DEVELOPMENT OF A PROTOTYPE SENSOR-INTEGRATED URINE BAG FOR REAL-TIME MEASURING

Atigorn Sanguansri

A thesis submitted in partial fulfilment of the
requirements of Bournemouth University
for the degree of Master by Research

May 2016

Bournemouth University

Copyright Statement

Abstract

A PROTOTYPE SENSOR-INTEGRATED URINE BAG FOR REAL-TIME MEASURING

The urine output is a rapid bedside test for kidney function, and reduced output is the common biomarker for an acute kidney injury (AKI). The consensus definition of the symptom is used urine output <0.5 ml/kg/hour for ≥6 hours to define AKI. If a patient is suspected to have this problem, the urine output monitoring needs to be done hourly, and this task consumes a lot of time, and easily affected by human errors. Moreover, available evidences in literatures indicate that more frequent patient monitoring could impact clinical decision making and patient's outcome. However, it is not possible for nurses to dedicate their precious time manually up to minute manually measurements.

To date, there is no reliable device has been used in the clinical routine. From the literatures, only a few automated devices were found with the ability to automatically monitor urine outputs, and could reduce nurse workload and at the same time enhance work performance, but these still have some limitations to measure human urine.

In this thesis presents the development and testing for such a device. The research was aimed at building a prototype that could be measured a small amount of urine output, and transit information via wireless to a Cloud database with inexpensive and less complex components. The concept is to provide a real-time measurement and generates data records in Cloud database without requiring any intervention by the nurse.

The initial experiment was done measure small amount of liquid using a drop-volume calculation technique. An optical sensor was placed in a medical dropper to record number of counted-drops, the Mean Absolute Percent Error from the test is reported ±3.96% for measuring 35 ml of liquid compared with the ISO standard. The second prototype was developed with multi-sensors, including photo interrupter sensor, infrared proximity sensor, and ultrasonic sensor, to detect the dripping and urine flow. However, the optical sensor still provided the most accuracy of all.

The final prototype is based on the combination of optical sensor for detecting drops to calculated urine flow rate and its volume, and weight scales to measurement the weight of collected urine in a commercial urine meter. The prototype also provides an alert in two scenarios; when the urine production is not met the goals, and when the urine container is almost full, the system will automatically generate alarms that warn the nurse. Series of experimentation tests have been conducted under consultant of medical professional to verify the proper operation and accuracy in the measurement. The results are improved from the previous prototype. The mean error found of this version is 1.975% or ≈ ±1.215 ml. when measure 35ml of urine under the average density value of urine (1.020). These tests confirm the potential application of the device by assisting nurse to monitor urine output with the accuracy in the measurement. The use of the Cloud based technology has not been previously reported in the literature as far as can be ascertained. These results illustrated the capability, suitability and limitation of the chosen technology.

Table of Contents

Contents

List of Figures

List of Tables

Acknowledgments

The study presented in this thesis was achieved with the help and kindness of many people for which I am very grateful. I would like to give special thanks to:

My special thanks go to my supervisor, Professor Hongnian Yu whose is more than a supervisor and supports me through this study with all its difficult moment from the very beginning. Professor Jim Roach, for his great suggestions, and Dr. Simon McLauhlin from the Royal Bournemouth Hospital, for his guidance and the most valuable discussion and consults through the journey.

My gratitude is also extended to all my study colleagues in faculty of Science and Technology, especially to my friends in the same office with whom I shared wonderful experiences.

This research was supported by the EU Erasmus Mundus Project-FUSION (Ref.545831-EM-1-2013-1-IT-ERAMUNDUSEMA21). I am grateful to this opportunity.

I would like to express my sincere of gratitude for my parents for their love, guidance and support throughout my entire study.

Abbreviations

Abbreviations used throughout the thesis are defined at first usage and listed below, where possible in their standard form.

AKI	Acute Kidney Injury
API	Application Programming Interface
cc.	Cubic Centimetres (cm^3)
CPS	Cyber-Physical System
CQC	Care Quality Commission
CRT	Capillary Refill Time
CSV	Comma Separated Values
ECG	Electrocardiography
EGG	Electrogastrogram
GND	Ground
Hz	Hertz
ICUs	Intensive Care Units
IDE	Integrated Development Environment
IoT	Internet of Things
IR	Infrared sensor
ITUs	Intensive Therapy Units
JSON	JavaScript Object Notation
kHz	Kilo-Hertz
MI	Microphone sensor
ml.	Millilitres (10^{-3}) litres
mm.	Millimetre (10^{-3}) meter
ms.	Mili-second (10^{-3}) second
NHS	National Health Service
NICE	National Institute for Health and Care Excellence
^{o}C	Temperature degree Celsius
pH	Potential Hydrogen
PI	Photo-interrupter sensor
PWM	Pulse Width Modulation
RFID	Radio-Frequency Identification
SD Card	Secure Digital memory card

US	Ultrasonic sensor
V.	Volts
VCC	Voltage at Common Collator
WEP	Wired Equivalent Privacy
WPA	Wireless Protected Access
XML	Extensible Markup Language

Chapter 1. Introduction

1.1 Background

Intensive care units (also known as ‘ICUs’) are a general term for specialist hospital wards that provide intensive care in the treatment and monitoring of patients with a critical illness or unstable condition, which require continuously close monitoring and support from the specialist equipment and medications in order to ensure normal bodily functions (Mosby's Medical Dictionary 2009). In some cases, ICUs are the place where life or death decisions are taken on professional judgement, relying on available evidence (Siegel 2009).

In general, ICUs treated a variety range of illness patients. As recommended in National Institute for Health and Care Excellence (NICE) clinical guidelines, monitoring adult patients vital signs in ICUs is often occurred least every 12 hours (National Institute for Health and Care Excellence 2007), but the diverse health status of each individuals in ICUs generates different requirements for health monitoring. Therefore, patients may require a different level of attention depends on their conditions. Some patients may require frequent care while the others of more stable conditions may require less. These are based upon the severity of patients’ conditions judged by nurses, which can be subjective.

At simplest, patient monitoring has been considered as a labour-intensive activity because it involves at least an observer (usually a nurse) to observe and determine changes in patients’ condition at least hourly. In the UK alone, it was found that an occupancy rate in intensive care was 80-88% out of around 4050 available beds (The Government Statistical Service 2015) compared to the number of nurses in average NHS hospitals, a survey indicated a ratio of nearly nine patients per registered nurse during daytime, and eleven at night (Ball and Pike 2009). Another survey from Care Quality Commission (2015) indicated that only 60% out of 57,996 patients felt there were ‘always or nearly always’ enough nurse on duty and, also found that 18% of 34,812 who pushed the call button for help request and, they claimed they had to wait for help ‘more than five minutes’ and 1% never got service. To reduce that number, and save more lives, would encourage hospitals to push up full intensive staffs in their ICUs. Unfortunately, the nursing staff shortage has still occurred. Thus, important factor that will reflect the patient monitoring and safe outcome is to reduce nurse since human resources are the most valuable input for this provision (Liu 2010).

Due to the fact of staff shortage, ICUs cannot only rely on a dedicated and hard working from nurses. A number of studies demonstrated that significant unintentional harm is caused to patients through nurses' failure to recognise the signs of clinical deterioration. The early UK study pointed out that 317 out of 477 hospital deaths occurred while requiring admission to the ICU during a 6-month period. 13 of these deaths were considered potentially avoidable: gradual deterioration was observed in physiological and/or biochemical variables, but the appropriate action was not taken (McGloin et al. 1999). The authors concluded that patients with obvious clinical indicators of acute deterioration are not infrequently overlooked or poorly managed on the ward. Another study from Rogers et al. (2008) reported that 365 errors discovered during the 28-day period from 12-hour worked shifts of nurses. Only 43 errors (11.7%) were detected before reached the patients, and the rest were overlooked when medications were ordered or dispensed. Those findings show the significant problems for nursing, since nurse staffs are in primary position to record changes in patient's clinical condition, but human errors still have been found in modern healthcare, such as diagnoses being written illegibly on paper, doctors not being able to easily access patient information, as well as limitations of time, space, and personnel for monitoring patients (Meingast et al. 2006).

Many researchers believed that the automated monitoring technology will help in replacing human work and at the same time enhance work performance with fewer mistakes (Kovner and Gergen 1998; Amaravadi et al. 2000; Pronovost et al. 2002). The automatic clinical tools were being used more frequently in clinical practice for identifying and monitoring, such as oxygen level indicator, heart rate, and electrocardiograph (ECG). These systems are variable in terms of physiological parameters and usages, but they share a common goal that is performing assessments of clinical changes in patients. However, urine output is the only parameter consistently used by clinical but not yet monitored electronically.

In light of the dramatic growth in the healthcare monitoring technology, only a few devices have been proposed for automated urine measurement in the last decade. The Urinfo 2000 is the first electronic and digit urine meter introduced to the market in 2009. It has been designed to especially estimate the urine output produced by the patients. Apart from a commercial device, a series of research were carried out by Otero et al. (2009; 2010a. 2010b; 2012; 2013). Those studies were based on sensor made up of different techniques to automated urine output measurement. However, most of the previous devices had drawback because some of them did not conform to regulations of indirect urine contact. Some have huge cost of development. And some required suppliers to redesign their mass production. Only a few devices are performed on animals in the University Hospital of Getafe (Otero et al. 2010a; Otero et al. 2010b; Otero et al. 2012; Otero et al. 2013).

The outcomes of the literature review showed that although the relevant information and background knowledge are plentiful but a small number of academic studies was recently address the importance of the urine output in the recognition of renal failure and kidney dysfunction and the studies in these domain are still in their infancy (Ricci et al. 2008; Hersch et al. 2009).

1.2 Problems Statement

The urine output is one of the major physiological criteria that used for indicating of kidney status. It is essential to calculate water balance and use for multi-protocols to observe the reaction of the patient toward the proper treatment. An adequate amount of urine produced means well perfused and oxygenated of the kidney (Legrand and Payen 2011). If a patient's urine output is too low, then he/she is suspected to have '*oliguria*' which is defined as patient produced urine less than 400 ml per day (an average value is around 15 ml per hour or 1 ml/Kilogram/hour) (Bellomo *et al.* 2004; McMahon *et al.* 2010; Cerda 2011). On the other hand, when a patient does not produce urine output at all, '*anuria*' will be suspected in most cases. Both oliguria and anuria are common problems in ICUs. These two conditions can cause variety of serious symptoms such as:

- *Prerenal azotemia* (Schrier et al. 2012) which leads to infection or heart failure,
- *Postrenal azotemia* which causes the blockage of the urine flow in an area below the kidneys,
- *Kidney damage* (McMahon *et al.* 2010), and increased risk of dearth (Uchino et al. 2010)

The average range for daily urine produced is generally around 800 to 2000 ml with a fluid intake around 2 litres. However, when the urine outcome is produced higher than 3000 ml per day, *diabetes* is most common reason (Tidy 2012).

Lewinton and Kanagasundaram (2011) reports that estimated up to 20% of critical ill patients in ICU suffered from deterioration of kidney function and they trends to promote the develop acute kidney injury (AKI) according to these criteria (Lewinton and Kanagasundaram 2011). An AKI, previously known as acute renal (or kidney failure) is a condition that relates to loss of kidney function over hours or days. In fact, not only kidney failure, AKI can also be caused by a problem with low blood flow which usually in a patient who is already unwell with another health condition (NHS 2013). If patients suddenly fall ill or have been diagnosis diagnosed with long-term disease, (e.g. chronic disease, a urinary system disease or signs of a disease affecting the kidneys and other organs). They are already suspected to be at risk of AKI (NHS Choices 2014). NHS Kidney Care reported that estimated cost to the NHS spent on treating AKI in the UK is as high as £620 million which is higher than combination cost of lung and skin cancer treatment (NHS 2013).

According to the press release of National Institute for Health and Care Excellence, early detection is a key priority and will prevent the patient's condition to become critical by following set out in guideline developed by the NICE. These include early identification of the condition and monitoring patient’s urine output (National Institute for Health and Care Excellence 2013). When the patient is suspected to have anuria or oliguria, the urinary output should be monitored hourly, and then ICU nursing staffs need to manually record the reading and empty a urine container for every patient under their care. This measurement is a repetitive 24 times a day protocol.

Based on our observation at the Royal Bournemouth Hospital, it was found that the current protocol for urine measurement is to read the scale from graduated container which connected to a Foley catheter with patient's bladder consuming approximately 2-3 minutes per measurement (see Figure 1.1). Dr Simon McLauhlin (MBBS., MD., MRCP.) from The Royal Bournemouth Hospital explains that since the measurement is taken hourly, nursing staffs will spend nearly half hour for every 15 patients under their care. He also says that such a repetitive task can increase stress and be considered as overloading work for each nursing staff in the ICU. This may also lead to human errors which can define as a cause affecting patient's treatments. Additionally, it has been found that the fluid input measurement is monitored with automatic pumps connected to the monitoring system which are installed in the modern ICUs. However, urine output is still monitored by manual registration as reference. This issue shows the significant role of fluid management in the ICUs that should not be overlooked.

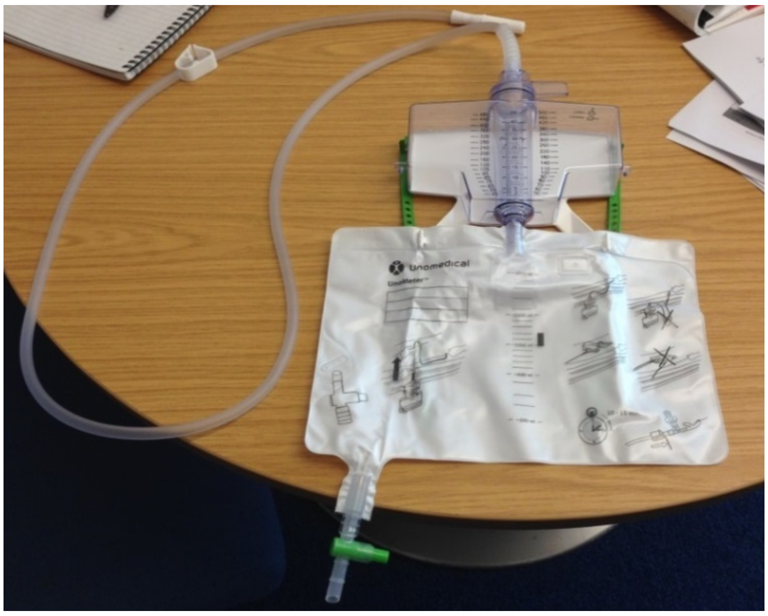


Figure 1.1 Urine meter currently used in the Royal Bournemouth Hospital

Implementing an automatic urine meter instead of a manual measurement seems to help reduce human errors and ease the nurse staff's workload. The automatic measurement trends to require less manpower to operate as well as the nurse staffs make less effort and time to register the patients' condition hourly. An automatic device could also provide clinicians for better quality of information to monitor and plan patient fluid therapy and even offer benefits to patients.

Nowadays, there are a few numbers of automatic urine measurement devices presented. Firstly, as reported in Otero *et al.* (2009), the Urinfo 2000 was released its first medical device product initially to markets in Israel. This device is a closed and standalone system with limitation to transmit data. Therefore, the nurse does not much advantage from having a digital recording of this parameter even though the study has proven that there was significantly more accurate of automatic urine data collection. Since this device does not transmit data to a central station for the clinician, it is still required a nurse to manually record the measurement from the device.

Apart from Urinfo2000, a series of research prototypes were built by Otero *et al.*, based on sensors made of different techniques (e.g. containers equipped with float sensor, high-precision industrial scale, capacitor sensors, and siphon container) offering high accuracy. However, most of the prototypes have drawbacks as only some of them were compliant with all laws applicable against indirect contact with urine, some designs need a very high development, some required suppliers to redesign their mass production, and they were also more complicated and difficult for nurses to use in practical situations (Otero *et al.* 2009; Otero *et al.* 2010a; Otero *et al.* 2010b; Otero *et al.* 2012; Otero *et al.* 2013). After all, previous studies, it should be highlighted that though some electronic devices have already been presented, a fully integrated system for general usage is yet to be developed and the details of each devices are presented in Chapter 2 Literature reviews.

Clearly, there is a need for carrying out for such studies. A suitable device to automate monitoring urine output is yet determined. The ICT (Information and Communication Technology) challenges raised by the integration of health and care service are large but have not been given as the priority they need. To address these key problems, we would like to investigate the possible solution of how to measure filled status of urine fluids, and transmit information with wireless to a secured database using less complex prototype that alleviates nurse workloads and avoid human errors. Nevertheless, small inexpensive integrated technologies (e.g., sensors, controller unit, and open source software) are now available in cheaper price increasing the possibility to achieve the solution for mentioned problems.

1.3 Aim and Objectives

This study aims to investigate the possible solutions of how to measure filled status of urine fluids, and transmit information via wireless to a Cloud database using inexpensive and less complex prototype that is alleviates nurse workload while reduce human errors in monitoring task. This aim will be achieved through the following objectives:

- To experiment through the feasibility studies of sensor selection and their attributes.
- To clearly study the capability those sensors to measure the urine output, epically patients who produce small amount of urine output.
- To identify requirements and expectation of the users.
- To develop a prototype system to collect real-time data with low-cost, off-the-shelf sensors.
- To implement an application to store transmitted data via a wireless connection.
- To test, evaluate and analyse the accuracy of the prototype device.

1.4 Research Methodology

This research is mainly conduct an intervention study and embeds qualitative data within the experiment procedures using prototyping technique, where the qualitative is used to find a foundation of knowledge in this research area, presented in Chapter 2.

To discover whether our prototype, as an automated urine output measurement device, could help nurses measuring the urine to compromise and avoid human errors. The following research questions were posed:

- What are the requirements from the users?
- Which technologies are currently being used in hospital?
- How prototype system should be designed and implemented according to the given the requirements?
- What is the level of user’s acceptance of the proposed prototype?

- How to verify and validate the output of the proposed prototype from user's acceptance?
- What will be the next generation of urine monitoring prototype?

The research questions aim to evaluate whether the concept of our prototype would reduce the nurse's workload meanwhile reduce human errors by doing monitoring task automatically using inexpensive and less complex prototype. The quantitative method will be used to evaluate the prototype in accuracy and level of user's acceptance. Therefore, the results of this research expected to be continued in clinical routine in the hospital. Thus, the experimentation had to conduct under suggestion of medical professionals and it has to be done similar to what it has been done in the hospital.

A comprehensive literature review was conducted to investigate the nature of physical properties and attributes of each selected sensor, comparatively study their advantages and disadvantages, carry feasibility study out of the current relevant technologies, and identify the requirement of the real-time integration system. Through the undertaking of the literature review, the foundation of knowledge in this research area was built, and the gap between existing studies and the aim of this research was identified.

In order to further identify the current process for measuring urine output in a hospital and medical industry, interviews and group discussion were conducted by medical professionals. Based on the results of a literature review and interviews, a deep understanding of the medical practice perspective and limitations of the current process was identified in the requirements to develop a prototype system that can achieve real-time data collection and defined integration of automatic sensor-based with wireless data transmitting in the application.

Considering the evaluation of drop counting as a volume measurement to be main basic ideas to solve this problem along with the characteristics of contactless sensor properties and their advantages in different aspects, a proposed prototype for the experiment was developed to test the integration of sensors and controller unit, real-time data integration with Cloud data storage and user interface. This method applied sensors as the data reading input connected with microcontroller board for automatically controlled all sensors, recorded the results across the sensor reading, displayed, and stored in an array for later processing. The wireless local area network was built to support the wireless data transmitting to Cloud storage.

An experimental study was adopted to investigate the technical feasibility of using off-the-shelf sensors for reading input data and transmitting them through a wireless network which controlled by a microcontroller. We were focusing on studying the sensor reading reliability and data transmission by using wireless network technology.

A series of tests were conducted to test the performance against the objectives and requirements. The disadvantages of the traditional method of urine measurement, benefits and limitations of the proposed prototype system were analysed from our evaluation tests. Medical professionals were consulted during the design of the prototype and experiments and discussed the results. They had given their concerns and suggestion for further improvement. Furthermore, it was an extension part of the post-development to draw research conclusion from the research knowledge and users' suggestions for future work.

1.5 Structure of the thesis

The thesis consists of five chapters, the content of which is briefly summarized as below:

Chapter Two: Literature review.

In this chapter, related research regarding the patient monitoring in ICUs is presented. It begins by reviewing relevant principle in fluid balance monitoring, techniques and technologies for monitoring clinical ill patient's parameter and provides an in-depth consideration of the urine output measurement. This chapter is followed by a review of the practical technologies on the necessity to develop a new model of integrating sensor-based prototype.

Chapter Three: Initial design of the proposed prototype system.

Based on the literature reviews, previous relevant works, and interviews with medical professionals, a proposed prototype system is introduced in this chapter. It begins with key requirements which indicated the features of the prototype regarding the current facts found in a hospital, a theory of operation and goes on the analyses the prototype design to make feasibility studies and showed technology selection, as well as systemic architecture design, which indicated benefits and limitation of selected technologies.

Chapter Four: Development and evaluation of an experimental prototyping approach.

In this chapter, the developments of two prototypes are presented and they functional operations will be discussed. Also, the relevant issues associated with the functionality and performances of the system are mentioned. The objective of the research is met through a serial of experiment trails. They were carefully conducted under consulting of medical professionals to provide evidence of the accuracy and to demonstrate the feasibility of the prototype which is the primary function of this study. Altogether, they provided the basis for the creation of a set of testing scenarios to evaluate overall satisfaction.

Chapter Five: Conclusions and suggestions for future work.

The final chapter presents the conclusion drawn from the research work. Discussion on the research limitations and recommendation for future work were made. The structure the research followed and the result chapters in this thesis are shown in Figure 1.2.

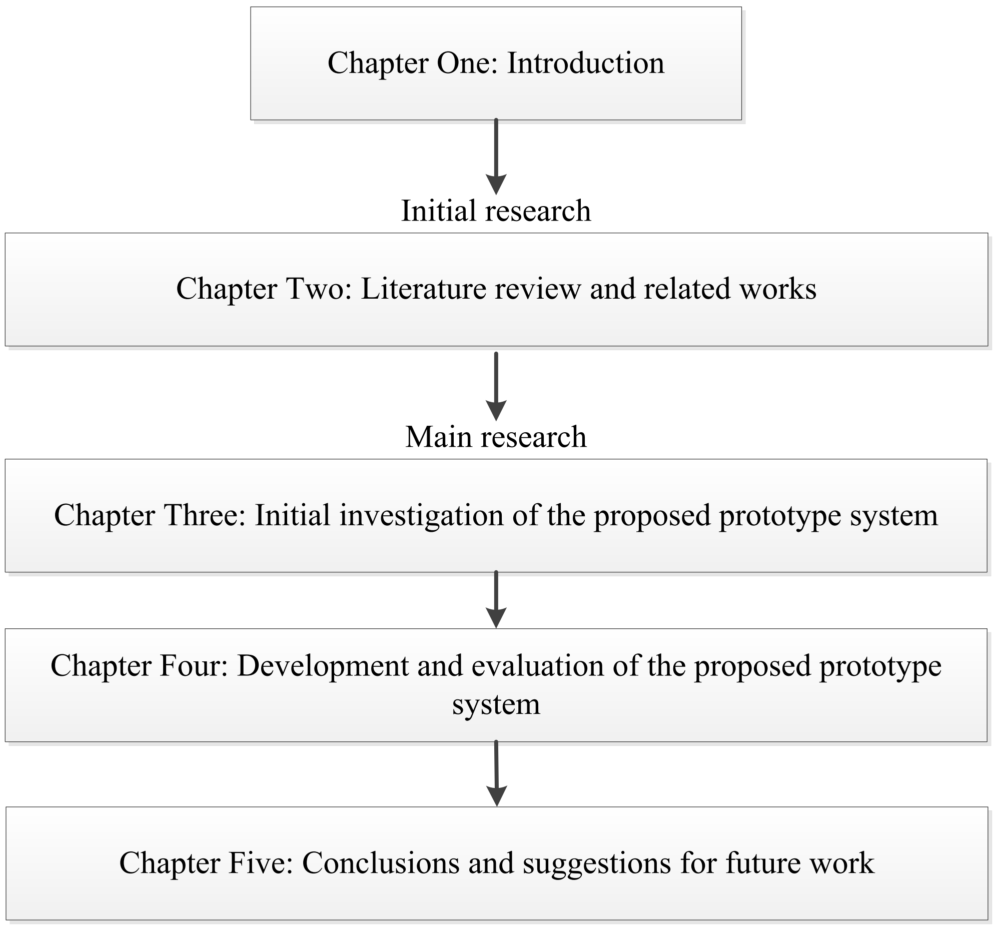

Chapter One: Introduction
Initial research
Chapter Two: Literature review and related works
Main research
Chapter Three: Initial investigation of the proposed prototype system
Chapter Four: Development and evaluation of the proposed prototype system
Chapter Five: Conclusions and suggestions for future work

Figure 1.2 An outline of the research structure and the chapters in the thesis

Chapter 2. Literature reviews

The aim of this chapter is to review the general background and relevant knowledge. In the first section, the fluid balance and urine monitoring in ICUs are described, with emphasis on the existing automated urine monitoring devices. The second section considers the technology literature relating to conducting research. The prototyping and the Internet of Things are stated.

2.1 Fluid balance and urine monitoring in ICUs

As described in the previous chapter, the aim of this study is to focus on urine output measurement. Despite from the urine measurement, we are briefly introduced the necessary background related to the main work presented in this thesis. In this section, the fundamental concept of water balance in human body and the basic methodology of physiological measurements are presented as well as apparent related studies and applications of the automated urine measurement device is discussed.

2.1.1 General Introduction

Intensive Care Society has given the definition of Intensive Care as followed:

"A service for patients who have potentially recoverable conditions, who can benefit from more detailed observation and invasive treatment than can be provided safely in an ordinary ward or high dependency area" (Intensive Care Society 1997).

The idea of an '*Intensive Care Units*', '*Intensive Treatment Units'*, and '*Critical Care Unit*' are generally considered to have the same meaning where they require constant medical attention and support to keep critical ill patient's body function in good condition. Patients may remain in such care depending on their specific conditions and recovery process. The Department of Health has defined four different levels of intensive care in hospital based on individual patients need (Department of Health 2000). The definitions of those levels of care can be summarized as listed:

- Level 0: Patients whose needs can be met through normal ward care in an acute hospital.
- Level 1: Patients at risk of their condition deteriorating, or those recently relocated from higher levels of care, whose needs can be met on an acute ward with additional advice and support from the critical care team.
- Level 2: Patients requiring more detailed observation or intervention, including support for a single failing organ system or postoperative care and those 'stepping down' from higher levels of care;
- Level 3: Patients requiring advanced respiratory care and support, or those who need monitoring and support for two or more organ systems, one of which may be basic or advanced respiratory support.

It is noted that Level 0 and 1 of care require a minimum of 4-hourly observations on the basis of clinical need while the patients in Level 2 and 3 need immediate care following major parameters; cardiovascular, renal and respiratory (Department of Health 2000).

According to study by Kipnis *et al.* (2012), monitoring of physical parameters has been mentioned as an essential in a critical ill patient. Physical parameters can collect an insight data from patient's body and can indicate sequential optimization of patient's health status, recognising early clinical deterioration and prevent them from critical harm or any organ failure (Kipnis et al. 2012). Elliott and Coventry (2012) agreed. They stated that a prompt detection and report of changes in those vital signs are absolutely essential as delays in the beginning of appropriate treatment can detrimentally affect the patient's outcome. The vital signs, as listed in

Table 2.1, are minimum physiological observations that should be accurately assessed before a serious change in a patient's physiology can be recognised.

Table 2.1 Basic vital signs monitoring in ICUs (Elliott and Coventry 2012; Kipnis et al. 2012)

Vital signs	Physiology	Influencing factors	Assessment issue
Pulse/Heart rate	Rhythmic expansion of an increased volume of blood pushed into the vessel.	Ages, oxygen demand, body temperature, and consumption	Regularity, strength and equality should also be assessed in 30 seconds counted.
Respiratory rate	Presents frequency of ventilation (number of breaths).	Varies with age, O2 level in blood, and Acidosis	Reference range for an adult is 16–20 breaths/minute. It is indicator of an oxygenation.
Oxygen saturation	Reflects the peripheral saturation of haemoglobin	Cardiac output, Haemoglobin, Fraction of O2	Does not reflect respiratory function overall
Blood pressure	Refers to the pressure exerted by blood against the arterial wall	Intravascular volume, Vascular tone, Contractility	For each heartbeat, blood pressure varies between systolic and diastolic pressure. A healthy adult ranges120/80 mm Hg.
Level of consciousness	Activating in the brain system.	Chemical substance, insufficient O_2 or blood flow, and the pressure in a skull.	Influenced by intra-cranial and extra-cranial factors. Commonly used Glasgow Coma Scale (GCS) for assessing brain injury patients.

Body temperature	Represents the balance between heat production and heat loss controlled by the hypothalamus	Age, infection, and medication	Core temperature of the body.
Pain	Detected by peripheral nerve	Patient’s perception	Pain is also a nurse-sensitive patient outcome and it should be treated promptly and effectively.
Urine output	Kidney’s drain functional	Renal perfusion, Cardiac Output	For an adult, the normal urine output is at least 0.5 ml/kg/hour

2.1.2 Fluid Balance Monitoring

Monitoring the patient’s fluid balance is usually done to prevent dehydration or overhydrating because vital signs such as pulse, blood pressure and respiratory rate, will change when the patient becomes dehydrated.

According to Welch (2010), the fluid balance is described as a balance between the input and output of fluids in the body which allows metabolic processes to function correctly. Around 52-60% of total body mass is fluid depending on age and gender (Shepherd 2011).

In the human body, the cell tissues contain water around two-third of the total water in body (Williams 1999). Nearly one-fourth of total body fluid is extracellular water (e.g. blood plasma, interstitial fluid, the space between cells, kidneys and skin). There is a small amount of water which is contained in joints, eyeballs and the cerebrospinal system. This fluid consists of water and molecules containing, sodium (mostly), chloride and potassium. It has been recommended to be one protocol of patient assessment in ICUs routine because water is the largest component of the human body (Katch et al. 1996).

For healthy adults, total fluid volume should balance between water intake and water loss, as well as fluctuates by less than 1%. Some evidences demonstrate that ICU patients who achieve a balance fluid balance status while admitted have had their desirable outcome and shorten their stay (Vincent et al. 2006; Boyd et al. 2011).

The minimum of fluid intake requirement must be equal to the amount of fluid losses to prevent insufficient water such as dehydration. However, determining actual consumption amount is difficult for varies reasons which mainly it affected by daily physical activities. The average minimum fluid consumption set by the World Health Organization (WHO) is volumes of 2,900 ml/day for men and 2,200 ml/day for women (The World Health Organization 2005). The European Food Safety Authority (EFSA) also suggested that 70-80% of daily intake should come from drink, and the remaining should come from food diet (European Food Safety Authority 2010).

In practical term, when the body has less water than it needs to function properly, '*dehydration*' is mostly suspected. In contrast, if patients obtain high sodium level in their body, it can cause '*fluid overload*' where retention of fluid occurred in extracellular. As a result, retention of fluid can lead to lung injury (Wiedemann et al. 2006) and brain damage (Fletcher et al. 2010).

Hydration status can be observed in several ways, starting from simple non-invasive measurements, for instance, body mass, intake and output measurements, and assessment from vital signs (e.g. temperature, heart rate and respiratory rate) in laboratory tests. There are several methods to evaluate fluid balance and hydration status as shown in Table 2.2.

Table 2.2 Assessment of fluid balance currently done in the hospitals

Method	**Protocol**
Observations	**Vital signs** will change if a patient becomes dehydration (Brooker et al. 2003; Scales and Pilsworth 2008).
	Skin elasticity will indicate a well-hydrated person when it immediately bounces back after pinching (Brooker et al. 2003; Scales and Pilsworth 2008).
	Mouth and **tongue** (Metheny 1987)

Thirst	Asked if a patient is **thirsty** (Epstein 2005).
Body mass	Increasing or decreasing of **weight** can be used to indicate fluid overloaded or dehydration through the decreasing (Scales and Pilsworth 2008; Shepherd 2011).
Capillary refill time (CRT)	Measured the fluid presented in the **intravascular fluid** volume (Scales and Pilsworth 2008). The measurement is assessed by holding the patient's hand and pressing on the pad of their middle finger. The pressure is released and the time measured in seconds until normal colour returns (Shepherd 2011).
Urine output	**Colour**: Urine should have a pale colour and it should be clear, with no debris or odour. In contrast, dehydrated patients, the kidneys attempt to conserve water producing urine with darker colour, concentrated and reduced in volume (Scales and Pilsworth 2008). **Volume**: Normal urine output is around 0.5 ml/kg of body weight per hour, in a range of 0.5-2ml/kg per hour (National PatientSafety Agency (NPSA) 2007; Scales and Pilsworth 2008; McMillen and Pitcher 2011).

The choice of the methods depends upon on many factors and situations such as a number of patients being assessed, the available resources and the testing environment. In most cases, when a patient losses a greater volume of water, the hypothalamus will evoke a sense of thirst and will cause a dry mouth (Guyton and Hall 2006). This assessment is only effective for patients who have ability to control their intake (Scales and Pilsworth 2008) and sensing of thirst can also be impaired in elderly patients (Cannella et al. 2009).

Assessing skin elasticity is quickly and simple test (Shepherd 2011) but it is unreliable indicator in elderly as skin elasticity reduces with age. It is highlighted that the limitation of the CRT method can sometimes be misleading, particularly when patients are suffered from sepsis (Scales and Pilsworth 2008).

It appears that urine parameter provides more accuracy and reliability (Kavouras 2002) because urine mostly contains water and other various other substances and the concentration of those substances increases with a reduction in urine volume. If a patient is dehydrated the kidney will attempt to conserve water and the urine output will be reduced in volume and become dark and concentrated (Scales and Pilsworth 2008). The body will have less water than it needs to function properly (Madden 2000). This increases risk factors for acute febrile illnesses, polypharmacy (diuretics, laxatives, drugs that decrease appetite or level of consciousness), and being bedridden (Rosner 2013) and often presents AKI with significant hypernatremia and, if untreated, the condition has a very high mortality rate (Weinberg et al. 1994).

Given the morbidity and mortality associated with AKI, preventing risk factors are clearly important. Generally, an early detection is a key priority and will prevent the patient's condition to become critical. The deterioration of AKI may be discovered by measure decrease in the urine output (National Institute for Health and Care Excellence 2013). The 24-hour urine monitoring is used as a simple non-invasive tool for assessing hydration status as compared with normal adults of similar body mass, and nursing staffs need to record the monitoring as part of the clinical assessment same as other vital signs (Health and Excellence 2007).

The NHS had announced the strategies to reduce the risk of AKI, this include identifying relevant risk factors, appropriate monitoring of urine and blood, rapid remedial action when AKI occurs, and appropriate referral of patients to specialist services (NHS 2013; NHS Choices 2014). The National Confidential Enquiry into Patient Outcome and Death (2009) suggested that the AKI should be easily recognised by the onset of oliguria, anuria and/or deteriorating biochemistry. However, missed opportunities to prevention was caused by poor assessment of risk factor. This also leads to delay in recognising AKI and will result in serious health condition and ultimately death. (National Confidential Enquiry into Patient Outcome and Death 2009). The consensus suggestion for preventing and managing the AKI in hospital is trough monitoring urine output and blood tests. The following section will be presented with background of urine output monitoring.

2.1.3 Conventional approach for urine output monitoring

In the human urinary system, it consists of two kidneys, ureters, urinary bladder, and urethra. The urine monitoring process is started with kidney filtration, where waste is extracted in the bloodstream, as well as excess water, sugar, and a variety of other compounds. The consequential urine contains high concentration of urea and other substances, including toxins. The urine then passes through the ureters to be stored in the bladder. During the urination, the urine is discharged from the body through the bladder and the urethra respectively.

As mentioned in previous chapter, disorders of water balance are exceedingly common in hospitalised patients, particularly those with critical illness who often get iatrogenic (Lewinton and Kanagasundaram 2011). A number of fluid balance assessments have been proposed (as listed in Table 2.2) but most of them are not established enough to replace urine monitoring as a marker of renal function.

Urine colour is the simplest method to determine the hydration status by using the amount of urochrome that is a breakdown product of haemoglobin (Ejlal 2012). Darker urine colour is accepted as an indicator of poor hydration status in individuals (Kavouras 2002). When a large volume of urine is produced, the solute is also excreted in large volume that is normally dilution with water which makes urine colour is very pale (Scales and Pilsworth 2008). A study by Armstrong et al. (1998) investigated the correlation between urine colour and its specific gravity using an eight-colour scale. A linear relationship between both the colour of urine and the amount of urine is found. Mentes et al. (2006) said urine colour testing is often found to be subjective (Mentes et al. 2006). Armstrong et al. (1998) agreed, they suggested that the colours can be used in the field setting to estimate hydration status where a high accuracy is not being needed (Armstrong et al. 1998).

The most accurate and precise biomarkers of hydration status that can show discrimination whether is a patient is ‘euhydrated’ (a normal state of body water content) or ‘dehydrated’ is using urine Specific Gravity (Chawla and Kellum 2012). The first of instrument used in measuring urine the Specific Gravity of urine was developed in the late 19th century (The National Meseum of American History n.d.). The principle of this device is often used to define the ratio of urine density to the density of water at specific temperature (Hall 2010). A scale hydrometer with graduated small stem is essentially utilised to pump in the sinking weighted float in test liquid which is proportion to Specific Gravity liquid as shown in Figure 2.1.

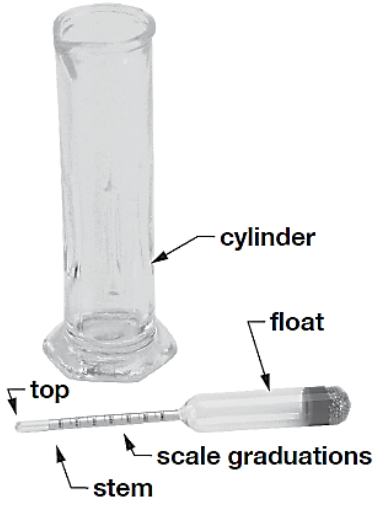


Figure 2.1 Urinometer (GF HealthProducts Inc. 2007).

In a Dictionary of Nursing (2008) defined the definition of '*Urinometer*' (A Dictionary of Nursing 2008) as:

"A hydrometer for measurement the specific gravity of urine. It is important in the determining the ability of the kidneys to concentrate or dilute the urine, which can be indicated renal structural damage, metabolic disorder, or endocrine disturbance".

The operation procedure is based on simple principle of liquid displacement, as followed instruction:

1. Fill the urinometer with urine sample approximately 20 ml.
2. Give the stem a slight spin for freely floating and do not allow it hung on the side of the container.
3. Elevate the reading at eye level, and make temperature correction as needed. The urinometer calibration temperature is 60°F (15.6°C). While increasing every 5.4°F (3°C), also add 0.001 to the reading. The density is expressed in grams per millilitre (g/ml).

In adult human, a normal Specific Gravity range is from 1.002 to 1.038 g/ml. A composition of human urine found in NASA report is described in detail concerning chemical analyses of urine that it consists approximately 95% of water, with the other 5% of organic solutes including urea, creatinine, uric acid, and trace amounts of enzymes, carbohydrates, hormones, and non-organic substance such as, Sodium

(Na^+), Potassium (K^+), Chloride (Cl^-), Magnesium (Mg^{2+}), Calcium (Ca^{2+}), Ammonium (NH_4^+), Sulfates (SO_4^{2-}), and Phosphates (e.g., PO_4^{3-}) as shown in

(Putnam 1971; Rose et al. 2015). Moreover, urine is usually acidic with different pH values ranging from 4.5 and 8 (Putnam 1971). As stated in Rose *et al.* (2015), the normal specific gravity of urine ranged from 1.002 to 1.037 was found in subjects aged 18–68. The pH value of fresh urine is largely neutral with a median of pH 6.2. However, an inverse relationship between Body Mass Index (BMI) and urine pH is found. Factors leading to low urinary pH include; high, weight, old age, and increased dietary (Rose et al. 2015).

Table 2.3 Main constituents of normal urine

Substance	**Origin**
Water	Diet, metabolism
Urea	Protein deamination
Creatinine	Metabolism of creatinine in muscle
Hippuric acid	Liver detoxification of benzoic acid
Uric acid	Catabolism of nucleic acid
Ketone bodies	Lipid metabolism
Sodium	Diet
Chloride	Diet
Potassium	Diet
Phosphates	Diet, metabolism of phosphate-containing compounds
Sulfates	Diet, metabolism of sulfate-containing compounds
Ammonia	Deamination of amino acids
Calcium	Diet and bone demineralisation
Magnesium	Diet

In most cases, urine SG increases linearly with increase urine osmolality. This relationship is altered when there are a significant amount of large molecular in the urine (Armstrong et al. 1998; Mentes et al. 2006; Ejlal 2012). The result from urinometer operation can be used to indicate a sign of diabetes or kidney problem (GF HealthProducts Inc. 2007).

A hydrometer scale is divided into four divisions. The highest division was marked with the letter "*W*". This is the level where the instrument rests in pure water. The next division was marked with the letter "*H*", which it rests when immersed in urine. The mark of letter "*S*" indicates good condition but not as healthy as "*H*". "*D*" for diabetes indicates when the instrument rests at the lowest division (The National Meseum of American History n.d.).

Explained by Dr. Simon McLaughlin (interviewed), this type of urinometer is found rarely to be used in clinical routine because it requires corrective calculation for temperature and increasing amounts of glucose and protein, and the urine volume of urine still requires for regular clinical assessment.

Generally, average urine produced in adult human is approximately 1.4 litres per day depending on state of hydration, activity level, environmental factors, weight, and the individual's health. If urine output is less than 30 ml per hour for extended periods of time in a patient eating an average diet, the patient is suspected as dehydrated and may risk having kidney problems (Cerda 2011; Legrand and Payen 2011). In other hand, when the patient is producing an excessive volume of urine, *Polyuria* is a fairly common suspected symptom. It is defined as an abnormally large production or passage of urine that is greater than 3 litres (over 24-hours in adults). Sometimes, the term "frequent urination" as well fits the definition (Tidy 2012).

In mid-1960, first currently used model of urine volume meter was introduced (see Figure 2.2). Basically, a 100-cc. graduated plastic cylinder was adapted to fit into a tube connecting to bladder catheter and pass urine to collecting jug. A simple pinchcock was applied at the lower end to let the urine flows down the tubing. Nurse was able to read and record the volume of urine in the cylinder on the hour, releases the pinchcock allows the urine to drain into collecting jug, as well as set the device ready for the next period. The graduated cylinder fully contained urine up to 5 cc and could permits accurate observation of volume within 2-3 cc. This concept provided accurate hourly measurement and, at the same time, enabled nurse to save all urine output for sample testing in later stage. The device was found useful, simple to clean and inexpensive (Haynes Jr. 1960).

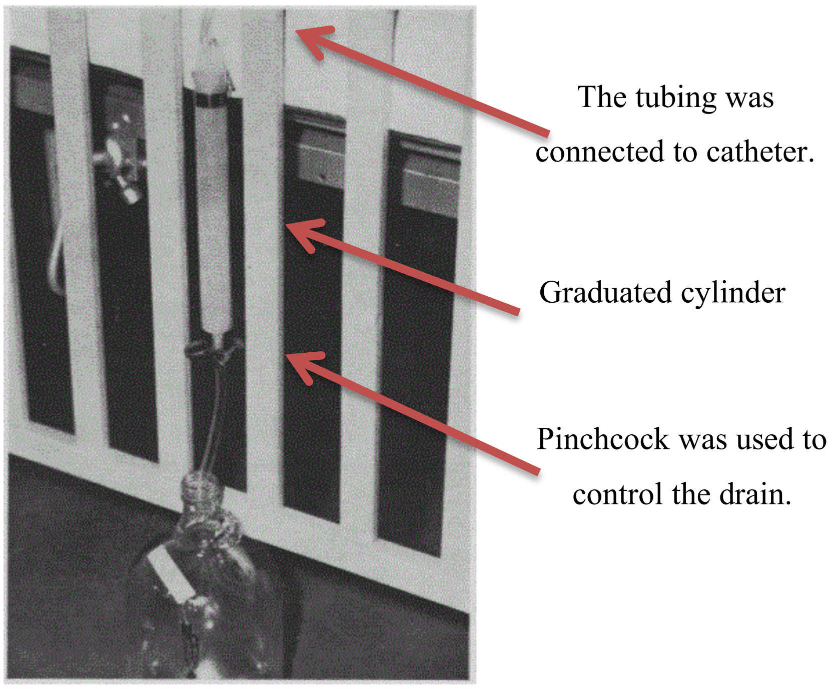

The tubing was connected to catheter.
Graduated cylinder
Pinchcock was used to control the drain.

Figure 2.2 First urine meter by Medical College of Virginia (Haynes Jr. 1960).

The urine meter and catheters nowadays come in a variety of designs, sizes and materials. The most common type of catheter in current used is the Foley catheter, developed by the American urologist, Frederick E.B. Foley (Lawrence and Turner 2005). It has been released since 1934 and remains relatively unchanged in design. This designed has 220 – 380 mm in length and consists a balloon that is inflated with sterile water to secure the catheter in the bladder at the proximal tip nears the drainage outlets at the end. The external diameter of a Catheter is measured in French gauge (F) or Charrier (Ch) unit, where 1 unit is equal 0.33 mm diameter (Jones 2005). Commonly, an adult catheter ranges between 12 to 30 units (or 4 to 10 mm) with the standard of 14 units (4.6 mm) and the 5 ml to 30 ml balloon size. A drainage bag is normally attached to the urine outlet managing a closed system. Some designed of urine meter are needed to be hung alongside the patient's bed as some are designed to be strapped to the patient's leg. Urine can either drain itself freely by the gravity into the drainage bag or released from the end of the Cather with a fitted on/off switch.

Additionally, it was reported up to 25% of patients who admitted to the hospital in the UK required a Urinary Cather at some point, and the catheterisation is not only found in ICUs but also amongst elderly residents in nursing homes (Warren 2001). Many of them need to be catheterised for months or years.

A main disadvantage of this urine meter model is the need to manually empty the measuring container hourly into the collecting bag to record urine volume and release the collecting bag when full (Hersch et al. 2009). Since, this protocol is taken hourly; nurses will spend nearly half hour for every fifteen patients under their care. Thus, they will perform the same task for 25 times in 24 hours, 365 days. Such a repetitive task may have considerably chances to human errors, and may affect the patient's treatments.

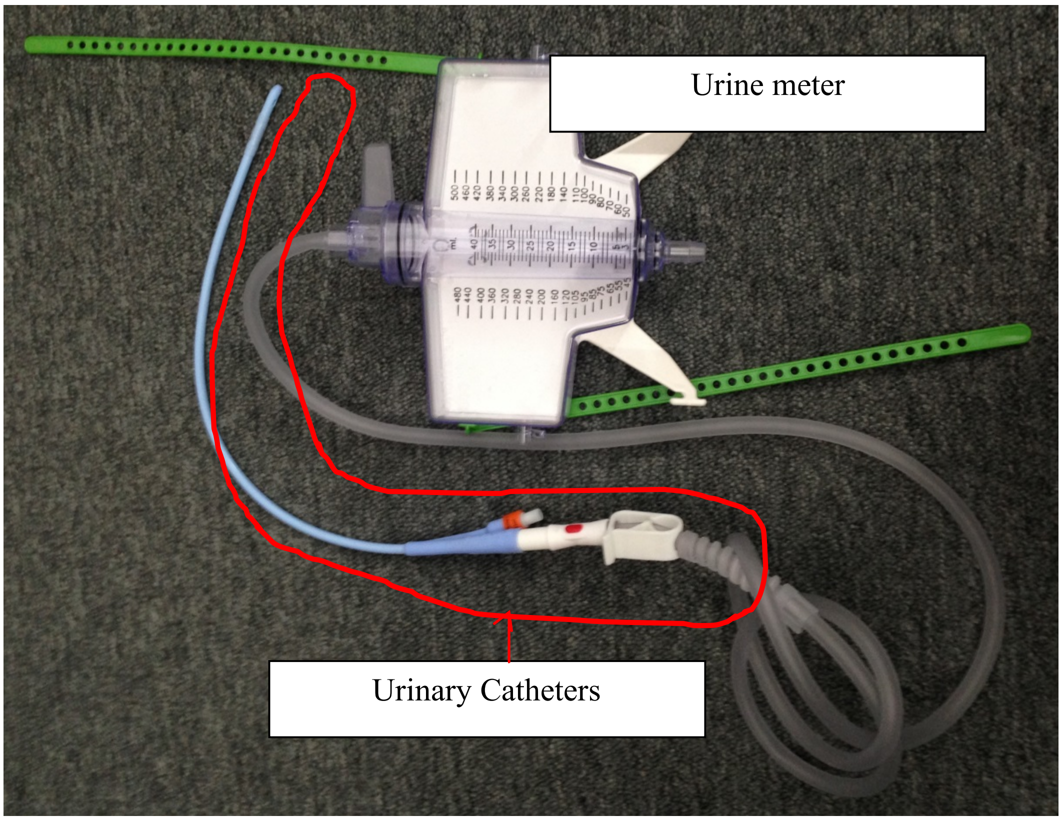


Figure 2.3 Urinary Catheters

2.1.4 Information flow in ICUs

When the measurement data is not directly go to clinicians, all assessments and measurement that done by nurses were recorded in accordance called '*ICU flow sheets*', manual records of the hour-by-hour (or minute-to-minute) attention given to the patient in a highly monitored setting. Nurses are responsible for monitoring and recording the findings, as their first main role is reporting the patients' condition trend and treatment (Andrews and Nolan 2006). This framework and necessary tools were provided to evaluate health monitoring and treatment through development of nursing process (Aust 2013).

The standard flow sheets have been designed as a single-page note for a 24-hour record in which all existing structures are recorded in real time. The data are recorded in either written form or as a diagram to be available for all treatment team members. The main goal of using a flow sheet in ICU is to save time. In Irajpour *et al.* (2014), they claimed that use of the monitoring flow sheets in ICU can help saving time for 2-3 minutes for each patient during the visit by a physician or a nurse. The time which is saved leads to an increased level of workload satisfaction and enhances their carefulness in recording. They had given the conclusion indicating that recording modification in ICU and the necessity of revision of recorded notes can lead to more effective nursing care (Irajpour et al. 2014). The use of precise monitoring and recoding leads to reduction of the overlooking from nurses and provides an early detection for physiological crisis through timely monitoring.

However, Gugerty et al. (2007) argued, they said reporting and recording patient's condition are parts of indirect care counted as 20% from all nursing work in each shift (Gugerty et al. 2007). Collins et al (2013) agreed. The study showed that sometime nurses recording document beyond what is required. In most cases they had concerns about the patient's worsening condition. One might expect this to lead to improved care giving between nurses and patients. However, the outcome of this action may reduce quality of cares. In fact, the documentation is often taking time away from providing hands-on care. Previous researches provide strong evidence that high nursing workload at the unit have negative impact on patient outcomes (Amaravadi et al. 2000; Lang et al. 2004). These studies suggested that to improving patient care are limited to increasing the number of nurses in a unit or decreasing the number of patients assigned to each nurse. However, it may not be possible to follow due to costs and the nursing shortage.

Moreover, Gugerty et al. (2007) reported that 25% from human errors was found in monitoring flow sheets, including calculation errors, deletion of data or formation of unreadable data. Shortage of standards in designing monitoring flow sheets as well as lack of specific information needed by physicians in ICU can also reduce the quality of care; consequently, other important information in the records is missing. The related problems were found when information is transferred from verbal instead of using flow sheets, around 3-4 additional minutes will be taken for each bedside visiting (Andrews and Nolan 2006). There is some evidence that conversation between nurse and doctor can be frustrated to a patient, particularly patient with serious or life threatening diseases (Kinnersley et al. 2008).

To improve quality of care in ICUs, many technologies were introduced in modern hospitals, such as Electronic Medical Records (EMRs), automatic vital signs monitoring devices, etc. These technologies will be discussed in the Section 2.2. Not surprising, the results from most of the previous research were found that nurses have positive attitude to a new automatic device to implement into their daily tasks. They suggested that automated devices can operate with reliability, sending information directly to responders and saving precious nurse's time.

2.1.5 Automatic urine output monitoring devices

As compared at the hospital setting, patient's fluid input is carefully recorded and mostly administered by the automatic devices. In the same setting, urine output is practically the only parameter consistently used by the clinic not yet monitored electronically. Currently, a few devices are presented to claim about the effective ability to measure urine output in assisting the hospital service.

There are few solutions described in literature that involve automate task in the urine monitoring process. Two ultrasonic sensor based solutions were developed in late 90's. The first study was implementing the Vitalmerics VM220 ultrasonic and temperature sensors interfaced with a Hewlett-Packard 1000 computer (Shabot et al. 1988). Another one was using 20 of Urotrack Plus-220 sensors connected with Hewlett-Packard 78709A (Shotts and Hauf 1986). Both had capabilities to transmit the urine volume and core temperature into a computerized patient data management system and allowed continuous observation, and displayed the data in the computer's screen. However, as explained in Otero et al. (2014), these two devices have low accuracy rate that make them unsuitable to measure minute-by-minute. Moreover, these products were discontinued and did not put in a market.

In 2009, another electronic-automatic urine meter was lunched by Medynamixt Company which was called '*Urinfo 2000*'. The device was announced to be first urine monitoring product and it was initially introduced to markets in Israel (Anwar 2010). It consists of three main components: (1) the electronic digital monitor, (2) the flow detector, and (3) the disposable measurement unit (collecting bag).

The concept of the measurement is based on an infrared sensor as the device turns the urine into a uniform flow passing through the disposable to the collecting bag same as the old-fashioned way. The digital display shows opportunely accumulated urine volume as the clinic requested. A patient's case history is kept for nine days.

Hersh et al. (200), had compared the accuracy of Urinfo2000 with the accuracy of standard nurse-handle DK-3460-Unometer. The average error of automatic device was found at 8% compared with up to 23% from the nurse records. Most of the nursing staffs we convinced that the use of the digital meter was every handy. This automatic device has been proven to be significantly more than manually recording.

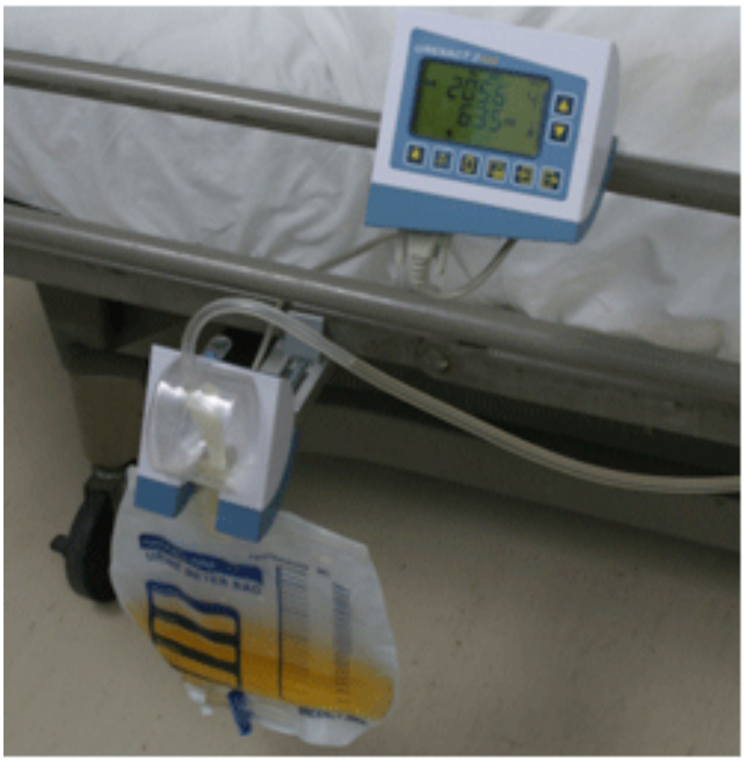


Figure 2.4 Urinfo 2000 device (Anwar 2010)

However, this current version is used as a standalone system where it lacks the ability to transmit the recorded data into centre computerises ICU monitoring system. Therefore, nurses did not much benefit from the advantage of having a digital recording parameter even though the device higher accurate. Since this device does not automatic transmit data into a work station for the clinician, nurse is still required manual recording the measurement from the device (Hersch *et al.* 2009; Otero *et al.* 2010a; Otero *et al.* 2010b; Bash 2012). As of today, Urinfo 2000 is no longer available in the market. The company had been bought by Flowsense, and the improvements had been discontinued.

A. Float sensors

In Otero *et al.* (2009), they presented the development of a device for automatic measuring and supervising the critical patient's urine output. According to their design, they wanted to achieve a robust and simple monitoring device that is capable of identifying the filling level of two different size containers as shown in

Figure 2.5. A smaller container has capacity of 15 ml, designed to be trigged in a half hour for 60 kg patient whereas the larger container has capacity of approximately 165 ml. Both were equipped with float sensor that moves vertically along a pole.

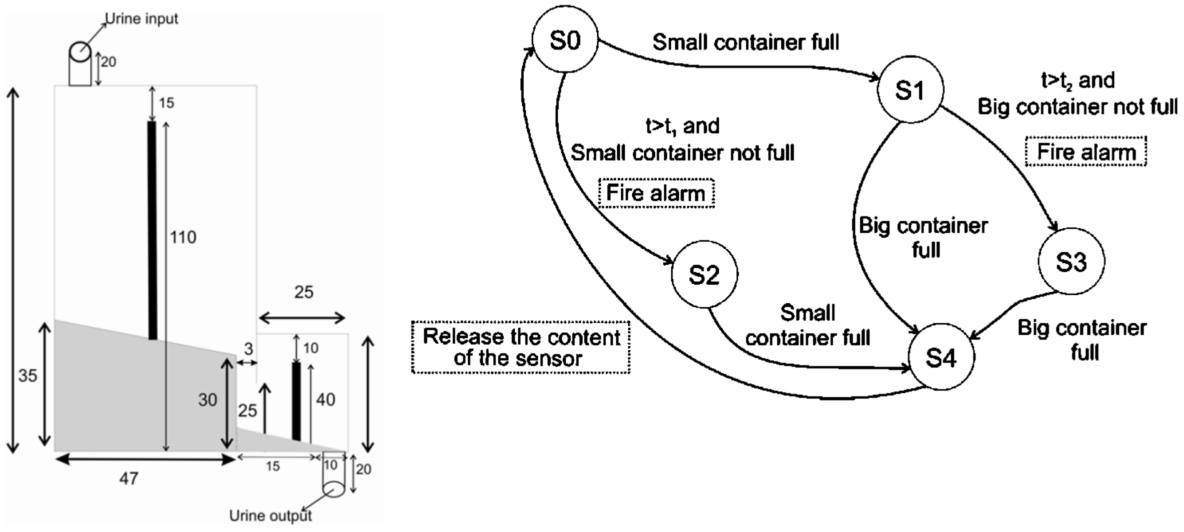


Figure 2.5 Design of the containers and component placement (Otero *et al.* 2009)

The floats allow each sensor to detect when a container is full. The smaller container is proposed to detect low urine output with a precise and continuous monitoring. The operation is based on the assumption that if patient produced at least 5 litres per day, the actuator will be triggered 25 times, but if does not fill in the expected time. Urine will be emptied after it got full and need to be measured again. For a patient who produces normal urine or suffers from polyuria, when the small container gets fill, the urine content will be not releases but it will start to fill the big container instead.

The use of smaller container that leads the release point is more frequently as well as reducing workloads with much work as possible. To achieve the duty without any supervision, the detectors must perform exactly at the moment that the container is completely full because once the container is filled, the urine produced will not be registered until it has been emptied. These release valves were adapted using linear solenoids controlled by microcontroller (Atmel AT89S52). This controller has maximum time to record around 327.5 hours (approximately 2 weeks). As long as patient produced at least 15 ml every 2 weeks, there will be enough time to read the filling status, release their content by activate an actuator when it is required, and trigger an alarm when goal is not being met.

The controlling software was developed using Java programing language that receives the reading from microcontroller through RS232 port. The therapeutic goals were established by physician based on weight of the patient. This program calculated the maximum time that the small container should require of filling, (t1) refers to state of the urine which is filled in the small container within the expected time, and (t2) is the maximum time required filling the larger container (as shown in Figure 2.5). When the microcontroller early receives data that indicates the full status before t1 period, the goals are met which is not necessary for precise monitoring. Therefore, the valve will not open and the larger container will be filled. Nonetheless, if the signal arrives after t1 period, the controller will turn on an LED light indicating that the goals are not met. In this case, precise monitoring is required. When the small container is full, the actuator will be active and the timer will be reset in order to measure the urine output again. At this point, if the small container is filled within the t1 again but the larger container is not filled before t2, the alarm will still turn on. In either case, when the larger one is full, the actuator will be active. Nonetheless, this prototype is beset with various mechanical problems on functioning the valve and actuator while in the operation period. Several legal problems were stated in this article, precluded the device from moving beyond the valid phase (Otero et al. 2009).

B. Reed switch and siphon principle

In Otero *et al.* (2010a), they proposed another device to measure the fluid flowing through the tube. Siphon principle was applied to empty the container and reed switches were used to measure the instant level of the containers. The concept of the device is placed one end in the bottom of the container, the liquid will go up inside that tube until it reaches the elbow part of the tube. Then, it will fall down the potation of the tube located on the exterior. Within a design structure, it limited only vertical movement of liquid sensor located inside the container. The float magnet is attached inside to interact with the reed switches which attached outside the wall (see Figure 2.6). The reed switch is operated by an applied magnetic field. It consists of a pair of magnetized metal reeds sealed in glass envelope. The contacts are normally opening, and closing when a magnetic field is present.

In general, this measurement is based on liquid that flows into the container from the time when the container begins to empty through the siphon mechanism to the time when it completely drained. The operation is described as; N refers to number of reed switch. At least one switch is located in an empty position (assumed N_0 as empty point). As the liquid begins to flow within the container, the magnet begins to rise. At first point, the reed switch N_1 will be detected, volume V_1 is recorded. At the point of additional volume, V_2 are being recorded after the reed switch N_2 is detected. The state of the reed switches continuously indicates the filling status at the same height as each sensor located until the magnet reaches the top.

An electrical controller is used to monitor the state of the reed switch sensors and send data through Bluetooth to a PC application based on Java language, similar to the previous prototype. From volume changes, the application will calculate the flowed urine and display a chart with information.

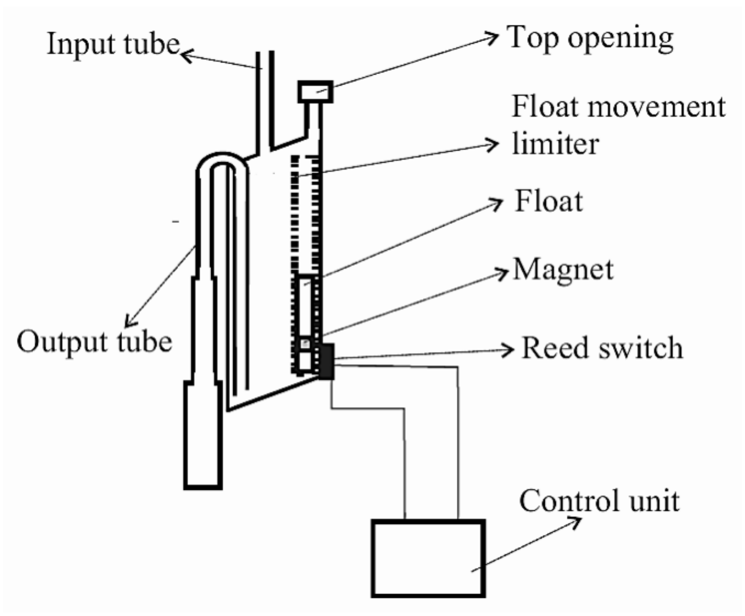


Figure 2.6 Drawing concept designed following siphon principle (Otero et al. 2010b)

Though this article, they reported the problems they had experienced by this design as follows:

1. *Complexity of the container design*: The container must be equipped with top opening for equalizing the internal and external pressure. It needs to be equipped with filter to avoid the risk of bacterial contamination. The distance between top opening and the top of output tube needs determination to keep liquid continously accumulated in the container and avoid immediat flowing when liquid reaches the top elbow.
2. *The bias toward the output tube's size*: The diameter of the output tube is needed to be balanced properly. If the output tube is too wide, the siphon principle will not work because air could rise against the liquid toward the tube. In contrast, to drain the container as fast as possible, the tube must be as wide as possible.
3. *The volumes will not being recorded during the empty through the siphon mechanism*: Two different sizes of the containers are being used to solve the problem, each of them working according the same principle. The output tube of the smaller container connected to the input tube of the big container. Thus, when the volume of the smaller container is not measured, it will be measured within the larger one instead.

4. *The connecting between two containers causes another problem*. When the smaller container releases its content and the larger one is nearly full. In such case, the content from the small container could not be measured at all.

This study here presented the device that was capable of providing feedback on patient's urine produced with at least the same manual supervision by nurse and in most case within 20-30 minutes. However, its device had several drawbacks from clinical studies. On the one hand, this device could only measures the instant time at which urine had been produced, but it did not provide rate of urine production. On the other hand, it is not easy to build a sterile device which is concerned on laws applicable to hygienic therapeutics (Otero et al. 2010a).

C. Scale based

That same year, another device was implemented using precision scale (PGW4502e from Adam Equipment Inc.) to measure the weight of a commercial urine meter, see Figure 2.7. This scale has advantage features to remove transient artefacts during the measurement and can be configured to send the reading when the detected weight is stable. A support framework made up of Bosch profiles focusing on the scale pan is to isolate the scale from transmission to patient's bed and guarantee for a continuous and smooth flow of the liquid through the urine meter's input tube. The urine meter consists of two containers, one is a 500 ml plastic container equipped with a top opening and filter. Another is a flexible polymer plastic bag with 2000 ml of capacity connected to a tube and valve.

A Java application receives the wireless reading through serial-port-to-Bluetooth adapter which is connected to RS232 port provided by the scale. When the urine is not obtained at the expected time, cubic interpolation is used to generate a measure at the desired instant time. The software part allows nurse to supervise the result by graph in millilitres per hour. The result of study showed the high accuracy rate, and has been used for highly accurate monitoring in renal function at the animal hospital in Spain (Otero *et al.* 2010b).

This device has shown high accuracy and acquisition rate compared with previously developed devices by the former authors. However, its size and operation procedure make it difficult to use in clinical routine, because it is not possible to be carried a balance scale with a hanger around the hospital while a patient is moving. The cost of development is also a weak point due to the price of the sensor materials. Furthermore, the new design would require new manufacturing processes and machinery.

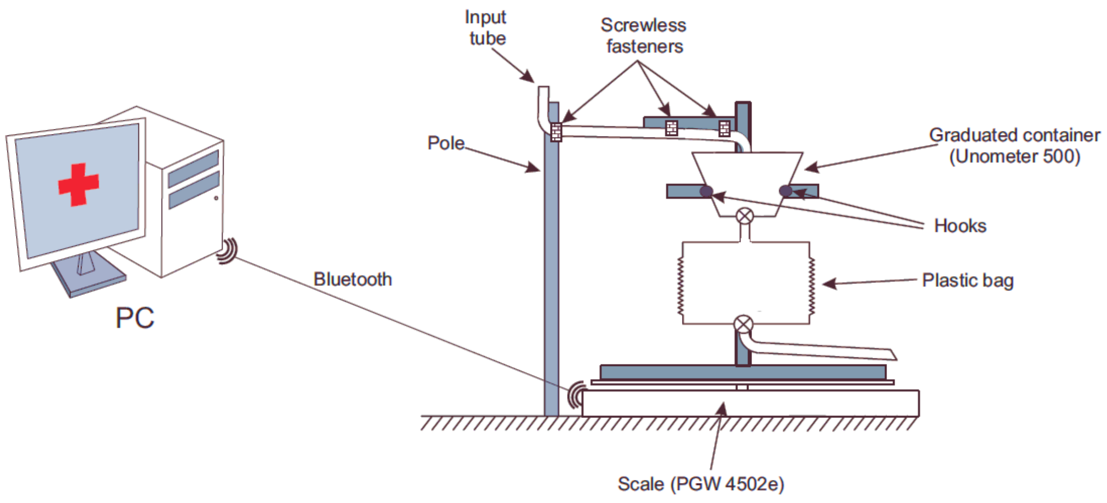


Figure 2.7 Device design based on scale (Otero et al. 2010a)

D. Capacitive containers

The ideal of new add-on part which can be manufactured separately and installed on the units is presented in (Otero et al. 2012). This article presented a device that applies with two capacitive sensors to measure the height of the column of liquid stored in container. The sensor is based on a coplanar-plate capacitor made up of two elongated conductive blades placed in parallel. The length of the blades is least equal to the maximum height of the column of liquid to be measured along with the container’s vertical dimension (see Figure 2.8).

According to this study, it should be able to provide accurate measurement of urine volume as small as 3–5 ml/hour. The error in measurement will increase proportionately with the area of the container when calculating the volume of liquid. Thus, to accurately measure small volume of liquid, the area of the container where the liquid accumulates must be small. Current clinical protocol specifies that urine output should be recorded hourly. They built a transparent plastic container with a dimension of 20 × 3 × 10 cm^3 and 3 mm of wall’s thickness, and divided into two chambers; the smaller one with 3 × 3 × 8 cm^3 is separated from the larger one with a dimension of 15 × 3 × 10 cm^3 and a wall of 8 cm in height. Each chamber is attached to a capacitive sensor, including circuit for providing an analog output ranging from 0.5V to 4.5V that is linearly proportional to the height of the liquid column. The

Analog to Digital module allows conversion of each analog output to signal from the capacitive sensors to a digital number for a full scale measurement between ranges of 0-5V.

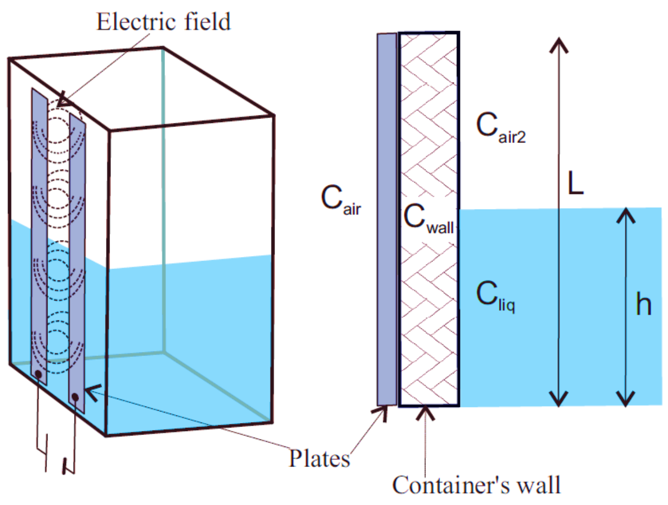


Figure 2.8 Design of capacitive sensor for measuring the height of the column (Otero et al. 2012).

The urine output readings are acquired every 5 seconds using a Java program runs on the PC. This program displays a graph showing the patient's urine output on a minute-by-minute basis. It also allows the healthcare staff to set the therapeutic goals for urine output by indicating the range of acceptable values.

In the conclusion, it has found that the automation of measuring and supervising urine output brings about a reduction in the healthcare staff workload. At the same time, the feedback on the patient's outcome provides more frequent than other currently available devices. This could result in patient outcome improvement. Moreover, the device presented in this article is considered as an added non-disposable part for further commercial urine meters.

2.1.6 Summary

The current literature has identified that urine output monitoring is an essential part of patient care in ICU. Since their alteration in vital signs may signify monitoring inadequacy that leads to harm in patient's being. They can be also used as multiple therapeutic protocols to assess reaction of treatment. Maintaining fluid balance is also important to avoid complications such as dehydration and overhydrating which can result in serious clinical consequences. One in five patient admission to ICUs trend to promote the development of AKI.

In order to determine whether patient is suspected to be at risk of AKI, the common indicator of early recognition is monitoring urine output. To measure the urine output, Foley's catheter is introduced into the patient's urethra. The other end of catheter is connected to a graduated container where the urine is collected and measured. Nevertheless, the nurse staff does not benefit from the advantages of having a digital display since it does not send information to patient's record. To promote effective care, the nurse needs to ensure that the fluid balance charts are recorded regularly.

Since urine monitoring protocol has to be processed manually, some studies found that repetitive and monotonous supervision of the patient's condition is prone to errors. Many of these errors can be avoided when they are done automatically. To the best of our knowledge, a series of prototypes have been proposed to discharge workload of nurse, reduce human errors, and provide a better care to the patients.

2.2 Technology literature relating to conducting research

Apart from the fluid balance monitoring, the basic concept of 'Prototyping' and the reasons for using a prototype as a demonstrator as a proof-of-concept model is discussed earlier of this Section, then, 'The Internet of Things' from the literature is introduced. It is apparent that conducting research into aspects of hardware and software development is fraught with difficulty so that the taxonomy of presented technologies in the healthcare industry will be utilized. Moreover, providing the reason why we adopted the Internet of Things to make digital senses the physical things is explained.

2.2.1 Prototyping development

The best answer of why we use prototyping in this study illustrated by Boar's justification (Boar 1984):

"Most currently recommended methods for defining business system requirements are designed to establish a final, complete, consistent, and correct set of requirements before the system is designed, constructed, seen or experienced by the user. Common and recurring industry experience indicates that despite the use of rigorous techniques, in many cases users still reject applications as neither correct nor complete upon completion.”

Because of the automation urine output measurement has only begun to be address in recently. Build a new device from scratch that provides similar functionality to previous products has considerable disadvantages and risks from the intellectual property rights that may be protected. The Royal Bournemouth Hospital has raised this concern and required a proven model that provides replicability, adaptability, and standardization to deliver operation efficiency while meeting the high standards and good value sought by patient. Thus, prototyping provides such a model.

The prototyping model was developed on the assumption that it is often difficult to know all of requirements at the beginning. Typically, users can identify many of the objectives that they wish to address with a system but they do not know all the details of the system features and capabilities (Houde and Hill 1997; Cho and Chung 2007). The Prototyping offers a development approach that yields results without first requiring all information up-front. Thus, a simplified version of the proposed system can be built and presented it to the users for consideration as part of the next development process.

2.2.2 Classes of prototype

As mentioned, prototyping is a technique that attempts to address the problems and provide possible feasibility. By using the prototype, the development can be performed much more quickly and cheaply, allowing an early version of the system to be put together and tried out on the user before the target system is built (Westen 2012). In this way, users can get an actual feeling of the product since the interactions with prototype can enable the user to better understand the requirements of their desired final product. Prototyping is an attractive idea for either complicated or large projects for which there is no manual process or existing system to help determining the requirements (Houde and Hill 1997). This means that the prototype is commonly not complete products and many of the details are not built in the prototype.

The term of ‘*Prototyping*’ is given to describe a wide range of methods where they used in multi disciplines. In the domain of product development, a prototype is referred as “a widely recognised core means of exploring and expressing designs” ,or "a test program, a block of code to try out an algorithm" is a definition for the field of Human Computer Interaction (Houde and Hill 1997).

Kelley and Littman (2001) said a prototyping is "a mechanism for problem solving". They expressed the key to successful use of prototyping is to ensure each prototype provides something that moves the project forwards, achieving parts of goal (Kelley and Littman 2001).

Another definition is given by Ulrich and Eppinger (2012) which defined prototypes as "being an approximation of the product along one or more dimensions of interest". They suggested that prototype can be usually be classified within two categories: (1) '*Physical or Analytic prototype*' which means a tangible artefact that built for testing and experimentation. (2) *'Comprehensive to focus*' is one that implements almost all aspects, closely resembling the final intended design in a working manner (Ulrich and Eppinger 2012).

Lidwell *et al.* (2010) agreed, but they introduced a different named for these two approaches of prototype: *Throwaway*, and *Evolutionary*, see in

Table 2.4.

Table 2.4 Summary of three stages of prototyping

Stages	**Characteristics**
Throwaway (Revolutionary)	• Useful for collecting information about certain aspects of a product • Understand and improve upon a design • Learn about a specific aspect
Evolutionary	• Useful when many requirements are unknown • Used for evaluation, but built upon in next iteration • A product of continuous refinement

Throw-away (or Revolutionary) presents an interactive simulation of the production which is usually iteratively refined until it represents a close likeness of the requirement. When the likeness is achieved, the revolutionary prototype is thrown-away and then the new product is constructed. In other hand, evolutionary prototypes are constructed for the final product with ever increasing functionality later on (Overmyer 1991).

2.2.3 The Benefits of Prototyping

Recognised benefits of prototyping are described in this section.

1. *Requirement capture and definition*: The main reason for prototyping is that it facilitates communication between developers and users (Acosta et al. 1994; Kinmond 1995). Wilson and Rosenberg said that this communication also provides a common reference for both developers and users (Wilson and Rosenberg 1988). The particular value of prototype is that it can be used to help understand the functional of the product (Damodaran 1991).
2. *Increased chance of product acceptance:* Described in users (Mason and Carey 1983), prototyping is widely recognized with increasing the chances that a final product will be accepted by users. Wilson and Rosenberg (1988) also said that a prototyping will increase the chances that the final product will work as users expected because they believed that prototyping will reduce the failure due to it generates understanding of requirements from brings developers and users closer together (Wilson and Rosenberg 1988).
3. *Flexible to implementation*: By using revolutionary prototype, it has capability to capture requirements and followed by evolutionary prototyping to develop the final product is considered as a very flexible way (Overmyer 1991) In this way, prototyping is able to handle with changing requirements. Although this method is not widely explored in the literature, this approach seems to represent the best practice of using both revolutionary and evolutionary approaches to prototypes.
4. *Reducing costs*: Some researchers regard that prototyping can reduce the cost of a development (Wilson and Rosenberg 1988). They found nearly 3:1 of budget is saved due to early changes and fixing problems at the prototype stage. Sibley et al. (1988) agreed, they claimed that nearly 25% is reduced compares with the cost when the produce released (Sibley et al. 1988). They also suggested that prototype should be done when the cost is less than a quarter of the total project costs, as design changes later would be likely to cost at least this much.

All of these positive attributes make prototyping an attractive approach for both physical prototype and software system development. However, not all these benefits are proven or are the inevitable result of prototyping. There are also disadvantages associated with prototyping. The main problems are seen as follows:

2.2.4 The Problems with using the Prototype

1. *Unrealistic expectation*: One of the main problems of prototyping in commercial product is that they trend to create unrealistic expectations (Wilson and Rosenberg 1988; Overmyer 1991; Kinmond 1995). For example, in short timescale, the prototype can lead a misunderstanding to users and even to developers to believe that the project is nearly completed. They may begin to think that a prototype that intended to be thrown away is actually a final system that merely needs to be polished. Sometime, users can also become attached to new features that were included in a prototype for consideration but then they were removed from the specification for a final product. If users are able to require all proposed features be included in the final system this can lead to conflict. Furthermore, adding new features may increase the complexity of the system as scope of the system may expand beyond original plans and create unrealistic expectations which can fail the project and dissatisfied users.
2. *Constrains and limitations* that apply to the real production are usually being ignored when constructing prototypes (Wilson and Rosenberg 1988). This overlooking can obviously lead to the unrealistic expectations and misunderstanding. Overmyer (1991) said the physical production can become worse than original plans because of constraints and limitations were found during the development.
3. *Management in a prototype process:* Wilson and Rosenberg (1988) highlighted that a prototyping process may be difficult to control and manage because too many changes can occur during the development. Overmyer (1991) agreed, the changes can occure due to the high visibility of prototyping. Developers can also become attached to prototypes because they have spent many efforts to develop them. It can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture. For this reason, throwaway prototyping is suggested to be used, rather than evolutionary prototyping (Overmyer 1991).

2.2.5 Discussions

As a solution to the problems we were facing (e.g. intellectual property right, lack of resources and time, unknown requirements) prototyping has been used and considered to act as a tool that provides a reflection between design and production which helps to improve the requirements and common understanding. Prototyping was used try out our ideas without the pressure of getting everything right straight away. In this way, we enable to involve users in design, implement, testing, and providing a better understanding of the requirements. This involvement is the most important part because it facilitates our design to users and they can have an actual feeling on the concepts. Moreover, it provides an iterative learning approach so the concepts can be developed and improved the functionality in the next stages.

In order give users to fully evaluate, it is necessary to create a physical prototype. It starts from the basic mock-ups written down in a paper to fully working prototypes. Physical prototype was considered to be a proof of the concept that our proposed prototype is met all given requirements. However, sometime physical prototype may require a high initial setup time followed by high running costs for both experiments and materials in each model. Thus, we need to find an alternative way to do a product prototyped fast and cost effectively for proof of our concept and get it into the hands of users.

As it is a part of identifying requirements, it is important to described the current technologies and compare them to evaluate state of the feasibility. Recently, several tools specifically designed to execute prototypes have emerged. These tools have the ability to quickly develop, test, and deploy ideas into prototype devices (Hodges et al. 2013) due to the impressive reduction of size, cost and energy consumption in electronic-hardware improvement (Payne and Macdonald 2004), and they are now available commercially, as open source, or as advanced research project (Kamogawa and Miranda 2013; Di Gennaro et al. 2014). One of these is the Arduino platform, a family of embedded processors that can be programmed with the C/C++ language via an accessible, minimalist integrated development environment (IDE). By using these tools (or gadgets), either stand-alone objects or online devices can be prototyped and they allows the prototype to integrate from a wide range of sensors, control a broad spectrum of output devices, and communicate with software running on a computer or even talk to each other over a network (Pearce 2012).

Stated by Banzi and Shiloh (2014), “With Aduino, a designer or artist can get to know the basics of electronics and sensors very quickly and can start building prototypes with very little investment” (Banzi and Shiloh 2014). We agreed on this statement, to get started with a prototype, the easiest way is implement it with development tool kits which their ability will accelerating the development process with fewer resources (including, man powers, budget and times). Additionally, the most important part of these toolkits is the integration of communication between machines to other machines, people or even objects. We think empower these gadgets to gather more information is a success of technology change, just as the Internet did. The reason is explained by Ashton (2009) "If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost" (Ashton 2009).

With so many possibilities options across the broad technology, the role of hardware and software platforms that expedite the reduction of ideas to working prototypes is an intriguing consideration (Hodges et al. 2013). The outline of the Internet of Things is further explained along with available services that facilitate the prototyping of networked embedded devices in next section.

2.3 The Internet of Things

2.3.1 Overview

As mentioned, the improvement of hardware is the great interest in nowadays technology. More devices have capability to collect data and transmit via the Internet to other devices. They are called ‘The Internet of Things’. It is closely related to machine-to-machine (M2M) technology (Yu et al. 2012). While the concept had been around for some time, the term "Internet of Things" was first used in 1999 (Ashton 2009). Since then, the idea has spread rapidly and widely. The original expression is composed by two words ‘*Internet*’ and ‘*Things*’, where ‘Internet’ is defined as “A world-wide network of interconnection computer network, based on standard communication protocol, the Internet suite (TCP/IP)” while ‘Things’ is referred to an object not precisely identifiable (Bassi and Horn 2008). Therefore, the meaning of IoT is:

“A world-wide network of interconnected objects uniquely addressable, based on standard communication protocols”

Another definition is given by the International Telecommunication Union (ITU) as:

“A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies” (ITU (n.d.)).

As point out from Fleisch (2010), when physical things connected to the Internet, they usually become smarter because they have certain computing resource and can do more smarter tasks than normal things (Fleisch 2010). Besides the Cyber-physical system (CPS) has very similar meaning to the IoT. The definition given to the CPS is ‘the combination of embedded software controlling the devices, networking capabilities and complicated physical dynamics exhibited by physical changes’ (Lee et al. 2012). Ahsanul Haque et al. (2014) given overview of CPS as connection between the virtual world and physical world with network integration (Ahsanul Haque et al. 2014).

2.3.2 The Internet of Things and the Internet

‘Is the IoT a separate part of the Internet?’ ‘Is the IoT similar to the Internet?’

These questions have been debated extensively. In one hand, the IoT had described as an extension of the Internet to reach out to the physical world of things and places that only can feature low-end computer (Gershenfeld et al. 2004). Fleisch (2010) argued, he said the IoT is not on the same level as the Internet, but it very much the same way as many existing Internet-enabled services (Fleisch 2010). We agreed on Fleisch’s statement and believed that the IoT is not the replacement of the Internet and they are not on the same level, but it could be considered either as parts or an application of the Internet.

The embracing the potential of the IoT will inevitably result in the generation of large amount of data, which need to be stored, processed and accessed. One paradigm that has long been recognized for such a big data storage and analytics is ‘Cloud computing’ which acts as a backbone to access Inter of Things(Ahsanul Haque et al. 2014) .

The National Institute of Standards (NIST) is given the definition of Cloud computing (Mell and Grance 2010) as described as;

“A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”

2.3.3 Cloud computing in the Internet of Things

We believed that large volumes of data will be generated as a result from the combination of many things to the Internet. This is based on the explanation of Canellos (2013), the combination of the Internet and the IoT will make sensing services and powerful processing of sensing data stream (Canellos 2013). Linthicum (2014) agreed, the advanced analytical services will let devices provide more critical information and it takes place based on remote analysis of this data in a real-time (Linthicum 2014). According to the Nation Institute of Standard and Technology

(NIST), they stated that the essential characteristics of Cloud for enabling a ubiquitous computing, as followed:

On-Demand Service means that the services from the Internet will be available when users need them (Gubbi et al. 2013). Cloud resources are normally in form of web-based services in three different levels: '*Application as a Services*', '*Platforms as a Services*', and '*Infrastructure as a Services*' and the need is to communicate with an Internet. These services have greatly helped to reduce cost and management tasks associated with virtualise resources, lowering the market entry threshold to businesses, and enabling provision of Sensing-as-a-Services for further services (Vermesan and Friess 2014).

Broad Network Access means the capabilities to access and connect through standard mechanisms (Mell and Grance 2010). With the advantages of the IoT, resources in Cloud can be access with multi-platforms and many devices at the same time (Gubbi et al. 2013).

Resource Pooling: The resources can be shared for those who authorised to access resources address. Resource pooling will make cooperation between people from anytime and anywhere as they want (Mell and Grance 2010) .

Rapid Elasticity: The Cloud computing offers an easy and quickly (in some cases can be automatically) provision to rapidly increase and decrease computing resources as needed. This gives the consumer impression that resources are infinite and that the application can always cope when in demand. When resources are no longer needed they are relinquished back into the resource pool (Sean and Kevin 2012).

Measured Service can be done as the usage on-demand service (e.g. storage, processing, bandwidth used). Any resources that were used are carefully monitored, controlled and recorded which allows the Cloud provider to be completely transparent with the consumer of the resources and facilities (Mell and Grance 2010). This means users only pay for the amount of resources they have used and always made aware of any discrepancies, spikes or abnormal behaviour regarding resources (Sean and Kevin 2012).

The recent advances in wireless sensor networks (WSNs), medical sensors, and Cloud computing are making both of them a candidate for healthcare applications including in-hospital and in-home patient care (Ahsanul Haque *et al.* 2014).

The key functionality of the both the IoT and CPS are to enable interaction between the ‘things’ and interface. This interface requires the ability for the digital to sense the physical world and make action on it. Unlikely traditional embedded system, these smart objects will become significantly more advanced than current embedded software-hardware system because, they are in particular based on cheap and small wireless devices with sensing, acting, communication, and advanced signal and information processing abilities. Secondly, wireless enabled technologies such as Wi-Fi make it possible to build low cost and reliable solutions and services (Dunkels et al., 2004) by connected various “things”.

Additionally, the IoT is typically designed as a network interacting elements with both physical input and output instead of as standalone devices. Conceptually, a network of medical devices and computer systems cooperating will interact in a given clinical scenario and provide a better care to patient.

2.3.4 Current practice of the IoT in hospitals

From a literature, there are several researches available to date. In this section, we are discussed the details and characteristics of each system to help visualize how the various systems proposed for healthcare applications were designed through the elements of: system architecture, sensor selection, data management, communication, and security. The combination between objects, things, or embedded hardware and Internet plays a significant role in a broad range of healthcare applications, from e-Medical records, monitoring system and other usage proposed in hospital. The sample practicable systems are introduced as following:

Simple vital signs monitoring based on Cloud (Rolim et al. 2010)
In this article, they have mentioned problems with the process manually done by nurses is undesirable and it is an error prone, as there is a possibility of incorrectly input. The simple architecture has been proposed: (1) sensors module collects the data from a serial port where it attached to the medical devices, then transforms and load information (2) the exchange server. (3) Content application displays the

information to nurse staff and using (4) unit computing to store, process, and delivery services. They claimed that the proposed service will help improving the quality of medical assistance delivery, especially in needy communities where it is difficult to gather expertise medical staff and the presented proposal will allow medical staff to remotely monitor patients located at another location. However, a detail of workable system did not present in this article.

CPeSC3 (Wang et al. 2011)

This architecture is integrated secured wireless sensor network, Cloud computing for U-life care application that responds intelligently to dynamic changes in real-world. A case scenario is presented based in two situations. In normal situation, the daily physical and medication is collected by sensors and stored in 3rd party Cloud via WSN-Gateway. Caregivers can regularly check for updated records and give some feedback (suggestions or prescriptions). In urgent situation, information will be sent to doctors and family members immediately so that fast respond can be taken in timely. However, authors did not highlight the details of sensors and specific data in this article.

iMedBox and iPackage (Pang et al. 2014)

The iMedBox architecture is consisted 4 elements: (1) Tracking and monitoring, (2) Remote service, (3) Information management, and (4) Cross-organization integration. A typical application scenario of the system has a powerful intelligent medicine box (iMedBox) that works not only as an in-home medicine container, but also as a “medication inspector”, and an “on-site examiner” in daily monitoring. The iMedBox is linked to public areas (e.g. hospital, medicine suppliers, and the emergency help centre) through wireless network. On the other side, iPackage controls a set of intelligent medicine packages and a wearable biomedical sensor tags (iTag) through RFID which links and wireless biomedical sensor network (WBSN) respectively. The software platform was running Android 3.1 (Honeycomb) operating system and capable to send all retrieved data into a web-based server as well as databases which implemented in Java. The system can connect to the backend system via either 3G or WiFi. A prototype system had been implemented to confirm the feasibility of the presented intelligent medicine packaging and container and it has been verifies by field trials.

BOX-Cloud-storage

The Oregon Health & Science University (2013) reported that more than three thousand patients records were stored on Internet-based storage service (The Oregon Health & Science University 2013) and for providing up-to-date information to other member in faculty about who was admitted to the hospital under the care of their division with an application named 'BOX', to store files, and share them with associates. It offers similar functionality with other Cloud providers but with added protections in place for confidential and restricted data. It was found that storage size is limited to 10GB with no requests for additional capacity accepted (Wilkes 2014).

Glucose Level Sensing

An m-IoT is found in (Istepanian et al. 2011). It based on IPV6 and 6LoWPAN protocol architecture where non-invasive diabetes sensors from the patients were linked low power microprocessor (MSP430) via USB connectivity and connected to the healthcare provider or the diabetes centre using wireless network based on IPv6 connectivity. An IPv6 access point used to send the relevant data to database. Data can be sent and collected both periodically or event-driven. Authors claimed to present a new concept that links the m-IoT connectivity for real-time non-invasive glucose level measurement. An experimental system had been implemented to verify the performance of the system.

Oxygen Saturation Monitoring

Ranken Jordan Pediatric Bridge Hospital in US recently announced to use a oximetry monitoring system into Masimo Radical-7, a monitoring system that uses sensors to collect patient data and wirelessly transmits into centre storage for ongoing display and notification. It has shown the more efficiency compared with standard screening. To date, the device is not available in the market yet (Zacks Equity Research 2015).

Huddle for South Tyneside NHS

Baldwin (2014) reported that South Tyneside NHS Trust in UK has begun to move the publishing of their board documents from paper to Huddle collaboration platform in iPads because users were already familiar with the iPad and they were already regarded as a secure platform (Baldwin 2014). According to report by Kelly (2014), security was important issue because they are dealing with health information, and iPad has good reputation for encryption and lower problems with malicious software, said by their Head of IT department (Kelly 2014). To protect their information, all the data are store inside the country and Mobile Device Management (MDM) software is used to secure the iPads and the Cloud backend. Moreover, the feedbacks were given much positive about the introduction of this iPad’s application.

Table 2.5 Mapping and comparison of the available IoT applications

Project	System architecture		Sensor		Data management & Communication			Security	
	Assisted	**Controlled**	**Type**	**Parameters**	**Computation**	**Storage**	**Protocol**	**Privacy**	**Encryption**
(Rolim et al. 2010)	Patients and clinicians	ICU	N/A	Multiple	Monitoring	Central	N/A	N/A	N/A
CPeSC3	Patients	In-home, hospital	Multi-sensors	Multiple	Daily living	Central	WSN	Data level	Network
iMedBox & iPackge	Elderly And patients	In-home, hospital, on-site	RFID (iMedbox) Temperature, EEG, and ECG (Bio-Patch)	Multiple	Monitoring & Reminding	Distributed	RFID-Gen2	Data level	N/A
BOX (The Oregon Health & Science University 2013)	Hospital staffs	Hospital	N/A	N/A	Storage	Central	N/A	Application	N/A
(Istepanian et al. 2011)	Patients	In-home Hospital	Non-invasive diabetes sensors	Glucose	N/A	Central	IPV6 & 6LoWPAN	N/A	N/A
Masimo Radical7 (Zacks Equity Research 2015)	Patients	Hospital	Multi-sensors	O2	Monitoring	Distributed	N/A	Data level	N/A
Huddle (Baldwin 2014)	Hospital staffs	Hospital	N/A	N/A	Storage	Central	N/A	Application	Physical

2.3.5 Discussions

The IoT is a concept to connect things, devices (e.g. smart phone or tablet to computers), and human of physical world to the digital world of information to create a ubiquitous computing world using enabling technologies; wireless communication and sensors. As explained in (Fleisch 2010), the IoT is not the replacement of the Internet but it is a future vision where everything will be connected to share a record of their interactions with people, context, and other objects. In order to do that, there is a need to extend Internet to things, such as; large database to store the information, network infrastructures, and software layer to handle the huge information flow. The database should be available in the Internet any anytime to be accessed by any device, person or thing. The Cloud computing has been recognized for such a service and it usually acts as a front-end to access Inter of Things. The combinations of these two technologies are already existed and can also be found in hospital settings. To build a ubiquitous computing, existing data, network infrastructures and identification systems, the services need to be integrated so that the contents can be exchanged and used seamlessly to realize the dream of IoT.

As mentioned, hospital staffs are squeezed by economic and regulatory factors; fewer healthcare professionals now manage sicker hospital patients. This has increased the demand for using remote-sensor to spot the early warning signs of patient deterioration. The existing monitoring installed in modern hospitals are commonly connected as a collaborative setting. The overall design of these systems is measure patient outcome and transmits information into administrative entities, so that the nursing staff in charge attending to the patients will be more efficient, and support needs to share a particular view of the electronic health records to participants group of clinicians and can also be useful to share links among different devices for individual clinician such as laptops, table computers, smart phone, or projector-connected computers.

There is no literature to directly mention the architecture of the IoT for hospital used, because the architecture of IoT-based healthcare hardware is more sophisticated than other usual devices and requires a real-time operating system with more stringent requirement, but there are mainly three elements that found in almost previous application; (1) Data element, (2) Connecting element, and (3) Processing element, and this can be said as a consensus of architecture design for healthcare and hospital.

So far, it is obvious that many efforts have been performed in this area but there is still lacking of secured and trustworthy prototype for testing, evaluation, and system developments. Proposed systems in healthcare to date are mostly still in design and simulation stage. For the very reason, there is inability to ensure the correctness in the uncertain environment scenario. The research challenge here is that the need of practical implementation which will be an excellent research direction and will open future research problems related to real-world implementation (Ahsanul Haque et al. 2014).

Chapter 3. Initial design of a prototype system

As stated in Section 2.1.2, the current protocol of urine measurement is considered to increase stress and workload for nurses, and it also leads human errors, which may affect with patient treatments. Many of these errors can be avoided by automating this task. An alternative solution is needed for non-invasive, less complex, cheap, portable, reliable and robust monitoring system. The above mentioned reasons have resulted in a surge of research in this area.

In this chapter, the observation learning from the Royal Bournemouth Hospital is presented; following by the given requirements by our consulters are identified in Section 3.2. In Section 3.3 and 3.4 are explained in details of prototype design in both hardware and software view.

3.1 The hospital observation

In March 2014, we had visited the Royal Bournemouth Hospital. The following lists are the finding in our visit:

1. Given the interview by Dr. Simon McLauhlin and his colleagues, they explained that the Royal Bournemouth Hospital is currently focusing on automatically equipment to measure patents vital signs (i.e. blood pressure, IV therapy, urine output etc.) to prevent human errors which may affect with patient treatments. Monitoring patient's condition is a repetitive task and work stress can be increased when nurses become overloaded. Furthermore, the demonstration of how to monitor patient's urine has been done, the procedure is described as follows:

   1.1. To insert a catherter into a patiant's body:

   I. Nurse will ask the patient to lie down in a supine position relexes the bladder and urethra, and the legs should also be spread.

   II. Ensure tha the catherter assembly came in sealed package and steriled.

   III. The nurse needs to sterilize and prepare the patient's genital area.

   IV. Then, apply lubricant to the tip of the catheter. If the patient is female, hold the labia open and insert the Catheter into the urethral meatus. If the patient is male, hold the penis and insert the catherter into the urethral opening. Continue pushing until the Catheter sits in the bladder.

V. Inflate the balloon with sterile water to ensure that the Catheter will be sit in place and connect the catherter to the drainage bag.

1.2. To empty a drainge bag:

I. Disconnect the valve at the end of the tube near the bag.

II. Hold the bag over the collecting container and open the spout at the bottom of the bag.

III. After the draginage bag is emptied, clean the spout with rubbing alchhol and then close it tighty.

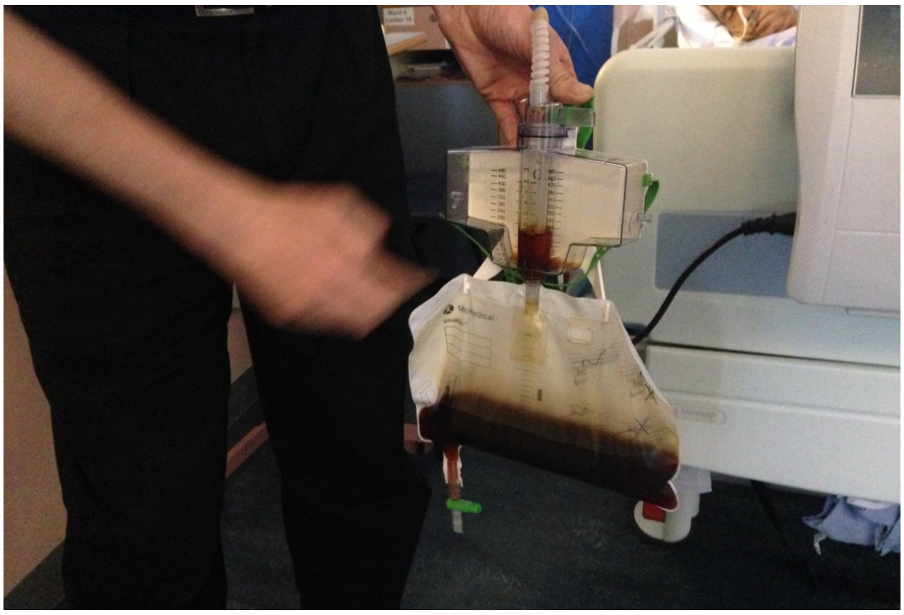


Figure 3.1 Demonstration of how to monitor patient's urine

2. Patient now have their condition monitored via a protable device system which automatically alerts staff if they start to deteriorate, using 'iPad' and 'iPad', and these devices were found extensively used for record and monitor a patient's conditions such as blood pressure, heart rate, and temperature.
   2.1. A system called VitalPAC Nurse is mainly used in tables, along with other in-home softwares in clinical computers.
   2.2. Nurses said that tablet based station is incredible protable and simple to use with a user-friendly interface and easily identifiable pictures.

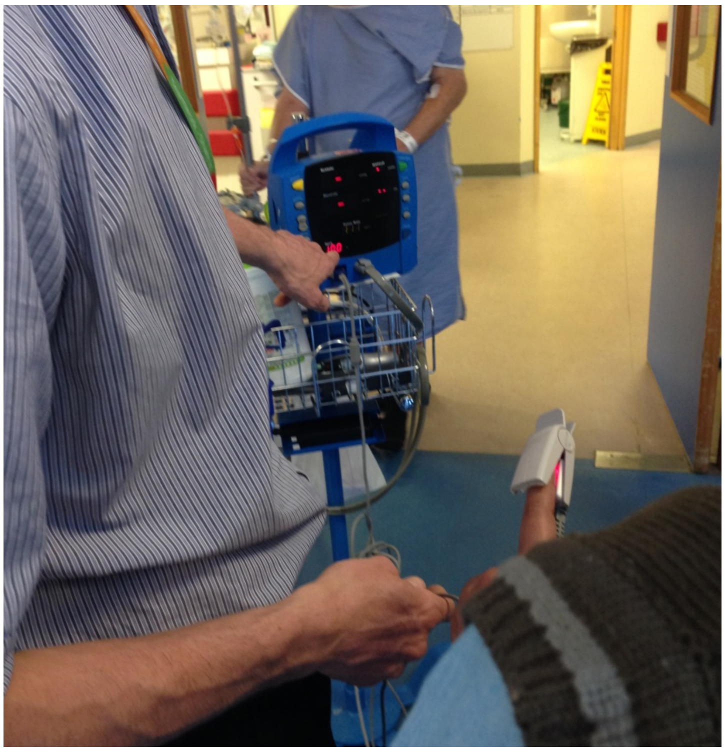


Figure 3.2 Blood-Oxygen level monitoring device

3. However, not all virtal signs were collected automatically. One of physiologicals that was not collected is fluid balance monitoring (exclude-IV theraphy where commercial machines are used for delivering and monitoring intake fluid).
4. One of the nurse had been interviewed and she told us that time taken to record all multitude of patient's condition is around five minutes per each patient, and it will be increased if paitient does not have stable condition.
   4.1. At the hospital, between 16 and 44 AKI alerts per day.
   4.2. Audit of 75 paitnets with AKI, more than half (≈35 patients) develop AKI after 48 hours of being admitted.
5. We found that incorrect records of fluid intake and output was occurred. The records were not matched the values displayed on the screen.
   5.1. Dr. Simon had raisen this issue and pointed out that under a heavy workload, nurses may not have sufficient time to perform tasks safely, apply safe practices, or overlooked some deti als.
   5.2. This issue had also found in CQC 2014/15 report, which stated that "The Trust should ensure that for patients who require their fluid intake and/or output to be monitored that this is accurately recorded" (Care Quality Commission 2014).

   5.3. There is an issue found 15 times where more than 4 hours nothing has been record records.

6. The UK government had set a new vision for the future of the NHS hospitals, which promise a ‘paperless’ health service by 2018 (Department of Health 2013).
   6.1. As a result, the Royal Bournemouth Hospital is now working on electronic patient’s records to replace some of the documents, but not all the paper based patient’s records will be gone.The hospital still needs to kept the records for at least five years.
   6.2. This changes will not direct affect the way they treat patients but instead medical staffs will be able to access records instantly from anywhere and anytime with any device and do not lose the time to wait for information to be sent from other wards or sites, reducing the time patient have to wait.
7. Nurses were having positive reposends when we were gathering requirements and introduce an automated acquisition of the data for urine measurement.
   7.1. They were felt that this concept seem to be useful to in clinical routine where the accuracy of data collection will be improved, and reduced some of their workload.

3.2 User requirements

After the observation, and interviews, pre-design for mock-up prototypes are presented and discussed in a group. A set of user requirements, and product matrices were identified. Here, the 1st research question has been answered. The represented requirements presented in this section are key factors to answer the 3rd question where successfully design and development process is translated into prototype functionalities from users themselves.

For this study, the specific requirements were mainly given based on guidelines and consultants of Dr Simon McLaughlin. He had given the following six major requirements that need be met:

1. All sensor components of the prototype must not contaminate, or enter into urine for patient’s hygiene reasons.
2. A minimum acceptable accuracy measurement provides by the prototype is 10%.

3. Rather than overall accuracy, a prototype must provide information for the patient with *oliguria* who produced urine as low as 10-15 millilitres, at least hourly same as the frequency with currently done by a nurse.
4. The prototype should be less complex and required less physical interaction with nurse, and provides mobility and comfort to patients.
5. Information must be stored and displayed in a Spreadsheets format. He also acknowledged that there are multi-level, experience and expertise of nurses in technology. Some of them may familiar with modern technologies, but some do not utilize these technologies and donot want incorporate something new.
6. The prototype should be adapted into currently used technologies in The Royal Bournemouth Hospital:
   6.1. Portable tablets are available for a nurse in working station.
   6.2. Instant access to records for multiple clinicians and authorised staff at the same time through the Internet connection within the hospital is required.

3.3 Design of the prototype system

Given the lack of sensors for measuring patients' urine output, there are no guidelines that specify the accuracy that they should provide, or any consensus on what is an acceptable error when measuring this parameter. Although many efforts have already been made in the design of automated urine measurement devices, as presented in Chapter 2, but there are still presented some limitations where we considered not unsuitable to work in actual clinical routine because:

1. Using a balance scale seems to not give enough portability to take the measurement while patients have movment.
2. The previous devices were designed to achive a high accuracy, so that their opreation were more complex and difficult to use in the actual situations.
3. Some of them were compliant with laws against indirect urine contaminated.
4. The cost of development was not suitable for generally used in the hospital because the urine container needs to be changed at least twice a week. If the equipment is attached to those prototypes, all components need to be thrown away.
5. Some of them were discontinued on improvement either operation features or marketing.

From the summarised of the previous study presented Table 3.1, it is obvious that the following issues are worth pursuing as challenges to this research area:

1. Currently, no single prototype was presented and claimed about the ability to measure human urine output in crinical routines and the hospital services. Hospital models can create barriers to progress.
2. Most of the previous prototype did not consider to minimized patient's risk of infection through invasive. Some of the designs were introduced some parts of metals directly contract to urine, which usually is acidic. Therefore, the metals can be corrosive and body fluid will be contaminated.
3. Lacking of economic impact and cost effective study. Commercial device may improve diagnostic accuracy but it is expensive and difficult to maintenance whereas a disosable device could be cheap but it usually need to be changed at least twice a week. Commercialising a new design may directly affect on preventing copyrights from competitors and also has disadvantages caused by other new product line that provides similar functions to the future marketed.
4. As serveral efforts have been done before, but they conceptualizes nursing workload at a macro level, ignoring the contextual and organizational characteristics of a particular health care setting (e.g., physical layout, information technology available) that may significantly affect further improvement in this topic research.
5. Likewise, the new prototype should be less complex and not require technical knowledge. It is expected to be easily installed and removed by nurse, and should not involve any calibration process as long as the sensor is in proper placement.

Since the purpose of this study has an aim to measure small amount of urine flow as low as 10-15 millilitres per hour. In the design phase, we strived to develop a prototype that would follow the requirements and suggestion as it should be less complex and suitable to apply for continuous usage.

One practicable solution is using an intravenous dripping system which delivers varying dosage of necessary substances administered to the patient in precise quantities (Norgia et al. 2014). The intravenous is based on principle of drip by gravity. It is very simple and low cost. The process of operation is stored substances in a container on a pole connected to intravenous lines and an infusion needle (Husch et al. 2005). This solution has been applied to measure urine output from patients by estimating volume using number of counted-drop technique. We believed that the principle of the intravenous seems to give more benefit for measuring such a small volume of liquid flow due to its simplicity, widely used, and proposed to measure small amount of liquid flow presented from the past until now. A fundamental of this solution is required the physics of drops and drop models to record the falling drops and estimate from its unknown volume.

Table 3.1 Summary of the previous automatic urine output measurement devices.

Device	Sensor Used	Communication	System Architecture	Advantages	Disadvantages
Specific containers with vertical float sensors. (Otero et al. 2009)	Float sensors	Serial-port to Bluetooth.	Closed system based Java Programming language. Central data storage.	• 2 weeks run-time. • Wireless communication. • Colour-code for warning indicators. • Alert systems; colour-code.	• Stress of release vale mechanical problems. • Components were contaminated with urine.
Siphon principle. (Otero et al. 2010a)	Reed switches	Serial-port to Bluetooth.	Closed system. Central data storage.	• Provide feedback more frequently. • Wireless communication.	• Stress of release vale mechanical problems. • Reed switches were directly contract with urine. • Complex design of the containers. • The bias of the tube's size.
Scale-based. (Otero et al. 2010b)	Scale	Serial-port to Bluetooth.	Closed system. Central data storage.	• High accuracy. • Wireless communication. • Able to contain large amount of urine.	• Size and operation procedural make it difficult for clinical routine. • Cost of development.
Capacitance containers. (Otero et al. 2012)	Magnetic Capacitance	Serial-port to Bluetooth.	Closed system. Central data storage.	• High accuracy for small amount of urine.	• Not viable to use in practical. • Components were contaminated with urine. • Cost of development.
Urinfo 2000	Infrared	N/A Displayed data from LED screen.	Stand-alone system.	• High accuracy with range from 1ml to litres.	• No integrated soft-data details available with centre database.

3.3.1 Drop-volume measurement

Typically, the use of a medical dropper is transporting the liquid into continually single drops falling under the gravity from the liquid storage on the top to the bottom. The outflow of the liquid product occurs through a dripping chamber and a terminal (Norgia et al. 2012).

For drop-volume measurement, it is known as a classical, maybe historical measurement technique that still being used in routine operation in the modern time. Therefore, the fundamental drop can be measured in volumetric where the estimation of the speed of the drop at a certain time period or the mass flow of each drop that falls off a dispense tip of known diameter. Given a certain speed of drop as *Q*, amount of liquid volume as *V*, a time period as *T*, the formula for working out flow rates is expressed as:

$$Q = \frac{V\ (ml)}{Time\ (minute)} \qquad 3.1$$

$$V = N_i * D$$

Where the total of counted-drop as N_i and a drop factor as *D* (gtts/ml). Here, the volumetric flow rate is a measure of volume of fluid passing a certain point per unit time. The result of the equation is usually expressed as millilitres per second or litre per hour, respectively.

Assuming that patient weighted 70 kg is needed to monitor urine output. The maximum target urine output from the guideline is 35 ml per hour. In this case, **how many drops will be delivered**? **how long does medical dropper take to deliver 35 ml of urine produced**? **What is the percentage error of this technique** compared with the standards?

To answer all of the above questions, the experiments of the drop-volume had been done to obtain characteristics of drop flow rates in different levels of liquid input and the results will be explained the insight details of each question.

.

3.3.2 Pre-design prototype

The simple circuit of optical sensor was built to detect the water drop between the upper and lower part of dripping chamber. The schematic is shown in Figure 3.3. The photo interrupter is used, it has 3 pins where POWER is connected pin-10, GND to pin-8, and OUT to pin-9. Additional LED was added to indicate whether the emitted beam is broken or not. One end of 1K ohm resistor is connected to pin-13. The anode leg of the LED is connected to the other end of the resistor and the cathode leg is connected to the GND-pin.

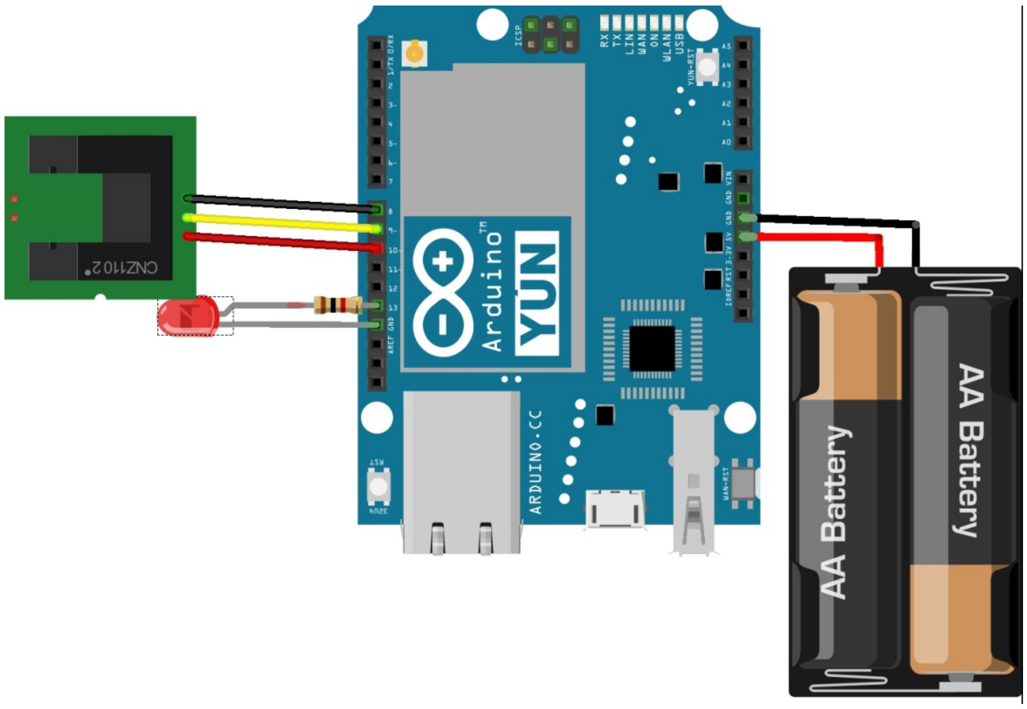


Figure 3.3 Breadboard view of a pre-design prototype

A graduated cup was used to fill with 5, 10, 15 and 20 ml of water and proceeded to make the drip from the point prepared, as visible in Figure 3.4. Fifteen-test set of each volume were performed. It is also necessary to highlight that in terms of a drop factor in these experiments does not take into account several factors such as; the morphological diversity of the drops, different temperature and specific weigh of the liquid, and body substance of the liquid.

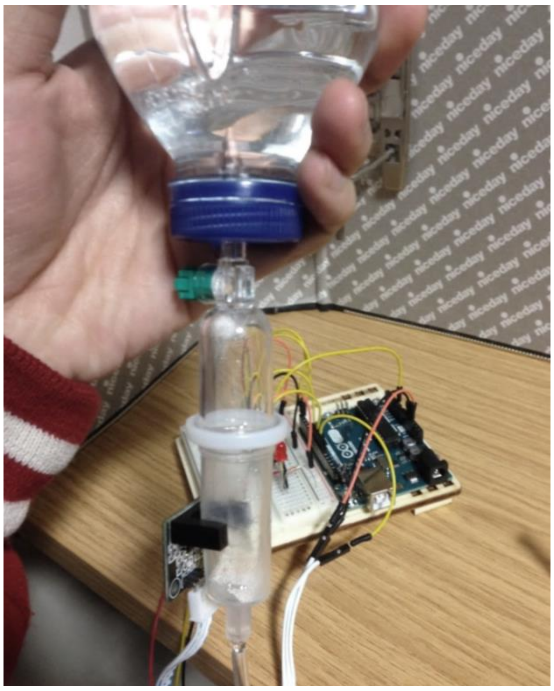


Figure 3.4 The preliminarily volumetric measurements

To define the speed of drops, each drop that passes through the sensor will be detected and kept time log where it is executed time range of millisecond, without using interrupts.

Figure 3.5 described the pseudo code of drop detect and time logger.

```
unsigned long time = millis();
unsigned long drop = 0;
LED = pin(13);
LED = LOW;

forever do:
   if (digitalRead(sensorPin)>0){
      drop = drop++;
      int sec = (time/1000)%60;
      int mins = ((time/(1000*60))%60);
      int hrs = ((time/(1000*60*60))%24);
      LED = HIGH;
   }
   else {
      LED = LOW;
   }
      print(hrs);
      print(“:”);
      print(min,2);
      print(“:”);
      println(sec,2);
   end
end
```

Figure 3.5 Pseudo code of drop detection and time logger

Nearly 5.5 hours of sensor data reading were collected from 60 times experiments. Each of the experiment was performed with no time limited to measure the time duration and the number of counted- drop, starts from the first drop passed the detected point until the input liquid was emptied. All the data were later transferred to Microsoft Excel for further transformation and analysis. Mean values and standard deviation of the performed measurements is illustrated in Table 3.2.

Table 3.2. The mean values and the standard deviations from the sensor reading.

Input (ml)	N	Avg. Time (minute)	STD	Flowrate (ml/min)	Avg. Drops-count
5	15	1.58	0.21	3.16	96
10		3.59	0.31	2.78	194
15		6.09	0.44	2.46	289
20		8.17	0.37	2.45	384

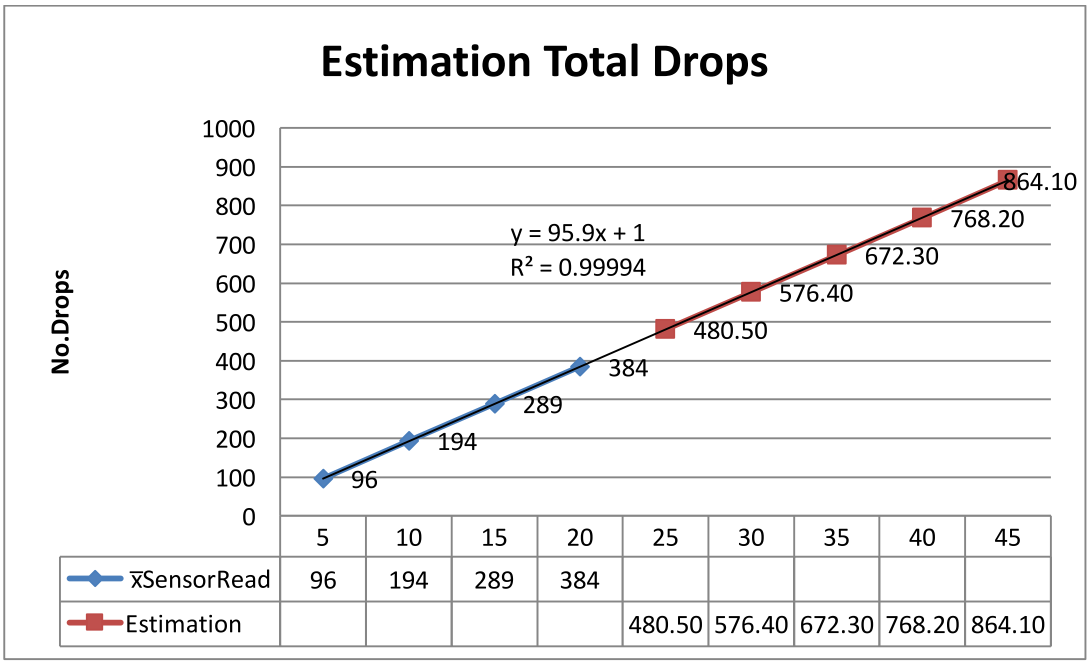
From the results, the average of flow rate without any tubing calibration is approximately 3.16, 2.78, 2.46 and 2.45 ml per minute for 5 to 20 ml of liquid, respectively. It was observed that increasing volume of liquid (in this case, the height of liquid column is increased), the flow rate is decreased. The reason is that fluid's viscosity is increased caused by increased pressure in the container.

It is a fact that the fluid pressure does not depend upon the total volume of the liquid, but it arises from the weight of the fluid and is given by the expression:

$$P_{statuc\ fluid} = H\rho G \qquad 3.2$$

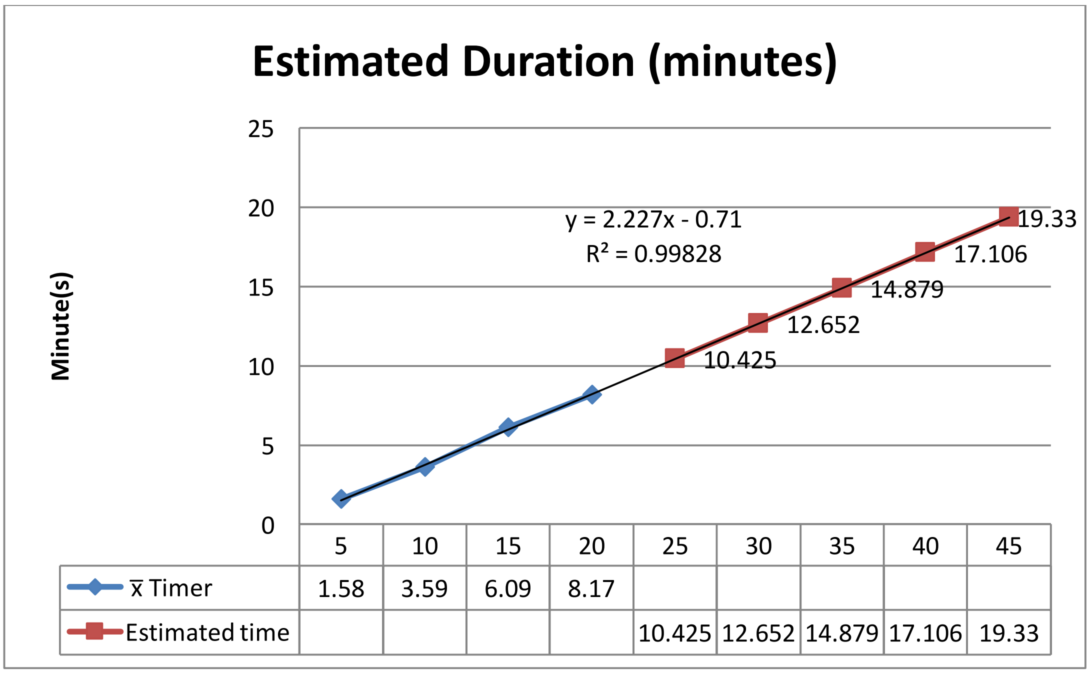
Where *P* is the fluid pressure, *H* is the height of fluid column (or depth in the fluid), ρ is density of liquid and *G* is the gravitational constant at 9.81 m/s^2. Regarding to this equation, the pressure exerted by the column of liquid is directly proportional of height of the column and the density.

From these statistics, the number of drops and expected duration emptied time can be estimated using the trend-line forecasting, as show in Figure 3.6 and Figure 3.7.



	5	10	15	20	25	30	35	40	45
x̄SensorRead	96	194	289	384					
Estimation					480.50	576.40	672.30	768.20	864.10

Figure 3.6 Estimation of counted-drop that will be delivered



	5	10	15	20	25	30	35	40	45
x̄ Timer	1.58	3.59	6.09	8.17					
Estimated time					10.425	12.652	14.879	17.106	19.33

Figure 3.7 Estimation time to empty the container using medical dropper

Table 3.3 Comparison between estimation using liner equation and with ISO 85364

Volume (ml)	**Est. Time (minutes)**	**Est. Drops**	**∑(Absolute Percentage of Error)**
5	1.58	96.90	0.830%
10	3.59	192.80	0.760%
15	6.09	288.70	0.543%
20	8.17	384.60	0.478%

To evaluate the error percentage, the MAPE (Mean Absolute Percent Error) is used to compare the size of the error in percentage terms where the average of the unsigned percentage error is presented, as shown in the equation below:

$$M = \frac{1}{n}\sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \qquad 3.3$$

Where A_t is the standard (actual) value and F_t is forecast value. The difference between A_t and F_t is divided by the actual again. The absolute value in this calculation is summed for every forecasted point in time and divided by the number of fitted points *n* multiplying by a hundred makes it a percentage error. The actual value is referenced by ISO 85364 :1987/ IS12655 where 20 drops will be delivered per ml (IEC International Electrotechnical Commission 1998). From these plots, when a patient produced 35 ml of urine, around 672 drops will be detected and used approximately 14.53 minutes (or 14.88 before converting a decimal to time format), and the percentage error of is ±3.96% or ±1.386 ml is different from ISO standard value.

After the first preliminary experiments, it can be said that the principle of drop-volume estimation is simple, proves to be of good precision for a small amount of liquid. The basic principle is simple and can be proceeds by deriving the urine flow from two basic parameters which can be easily accessible: (1) total of number drop-counting, (2) duration to empty the liquid input.

The error rate can be affected by increased uncertainty due to the level in infused container, the vertical alignment of the dropper is not precise, and properties of liquid. This uncertainty can lead a different size of the drop diameter and its weight (Lovick and Angeli 2004; Van Santvliet and Ludwig 2004). In Lovick and Angeli (2004), they claimed that the material from which is made will have an effect on droplet size in flows. Van Santvliet and Ludwig (2004) agreed, they stated that the size of drops delivered using drops count as volume measurement technique is influenced by three major factors included:

1. *The design and characteristics of the dropper tip*: the diameter of the dropper tip is directly variation to the size of the drop delivered. The surface tension of the liquid causes it to hang from the tip of the dropper and forming a pendant. When the drop exceeds a certain size it is no longer stable and detaches itself. The falling liquid is also a drop held together by surface tension (Cutnell and Johnson 2005).

A dropper tip was designed to delivery fluid flows into a drop followed Tate's law, common methods for measuring surface tension (see Equation 3.4), where *mg* is the weight when the drop fall, $2\pi r$ is the circumference, and σ, defined as the surface tension.

$$mg = 2\pi r\sigma \qquad 3.4$$

These drops grow as long as their weight is less than their holding force on the capillary. As soon as this weight has reached the same magnitude as the holding force, the drop falls and the volume of the falling drop is measured. In another word, the drop is falling when the weight is equal or greater than surface tension at the tip of dropper.

2. *The liquid-chemical properties*: The chemical substances in liquid may also influence surface tension measurement. In this case, urine substance is concerned that may reduce the accurate and reproducible of the measurement. However, Millis et al. (1988), they reported that a different only 2 mN/m was found between substance of water (73.0 mN/m) and urea (71.0 mN/m) at constant temperature. Only small changes will be found and may have small effect in the final estimation volume (Mills et al. 1988). This was confirmed by Jho and Carreras (1984), they concluded that an increasing of viscosity up to 15 mPa·s of liquid solution does not effect to drop size in different dynamic surface tension substances (Jho and Carreras 1984).

3. *The manner of handling the dropper:* It needs to define the shape form of drops as related to the drop's angles and the falling height. Vertical angles were assessed to indicate effective of the delivered volumes (Van Santvliet and Ludwig 2004). It found that 8.5% smaller drop were delivered due to the changing the dispensing angle to 45° angle compared to the vertical (90°), as well as different volume in a variety of drop shapes (see Figure 3.8 and Figure 3.9).

The theory of liquid shape is varying on the tilt manner is confirmed by Mazzola et al. (2012), they reported that two characteristic lengths of the drop shape that change with tilt grade. It is possible to see that increasing tilt grade, the drop become stretched and it shrinks orthogonally on the plate. Moreover, the shape of the drops is also varies due to the rate of a falling height (Hugli and Gonzalez 2000). They performed an experiment and obtained the drop shape changes in different falling heights, as shown in Figure 3.10. Clearly, the shape varies significantly with the manner of handling. In fact, when the drop leaves the tip, the sudden fall initiates an oscillation of the drop mass which goes on during the fall and is responsible for shape changing along the path.

Furthermore, the volume delivered is always smaller than the volume contained in the measuring instrument. This is caused by the fact, that a certain amount of liquid remains as a film on the inner surface of the instrument. The volume of this liquid film depends on the delivery time and should be taken into account when calibrating the measuring instrument.

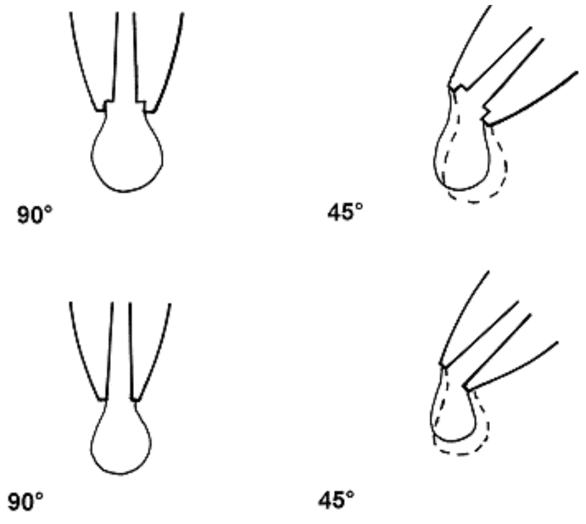


Figure 3.8 Drop delivering while changes the tip angle from 90°to 45° (Van Santvliet and Ludwig 2004)

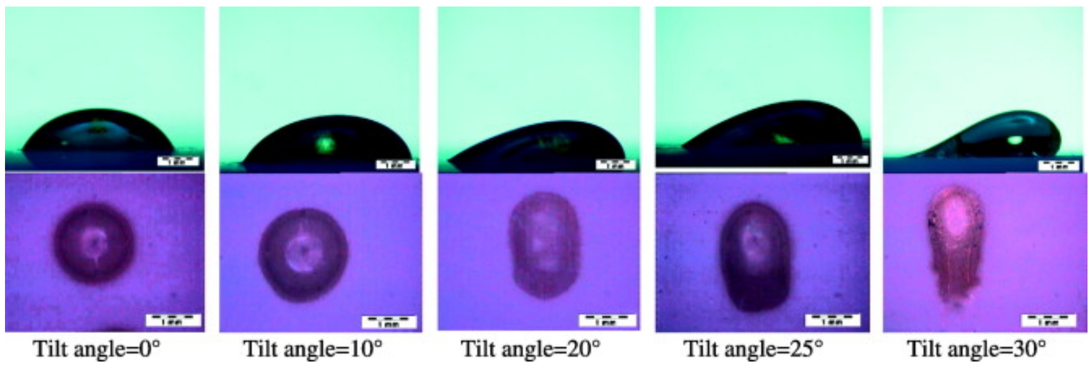


Figure 3.9 Different shapes of the liquid drops obtained varying the tilt angle (Mazzola et al. 2012)

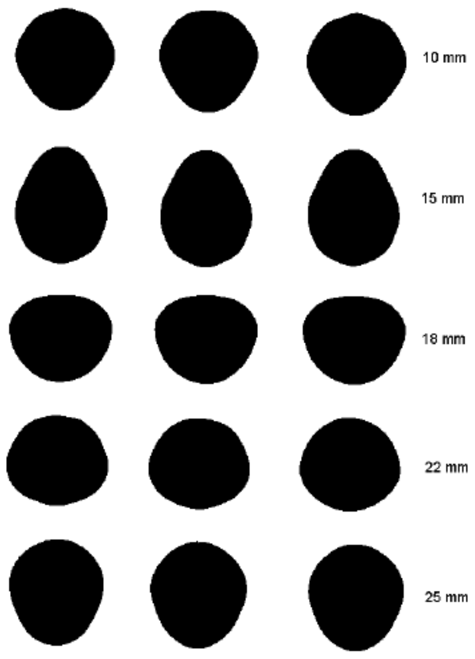


Figure 3.10 A variety of drop shapes from different falling height. (Hugli and Gonzalez 2000)

3.4 Design of system architecture

At the same time, the study is also focused on automation and real-time measurement techniques. A number of detailed reviews of flow measurement devices already exist and presented in many studies, but for this investigation only some non-intrusive sensors that can take a measurement in a small volume are considered. Some reasonable concepts were reviewed and provide a brief discussion of the theoretical and practical basis in Section 3.6.1.

The system architecture design was done by following the requirements and the general consensus of previous studies, the proposed system architecture is built under the integration of an embedded system connect and web application. It distinguished three elements of architectural given in Figure 3.11—(1) Data acquisition (or 'data element') which contains sensors connected to programmable microcontroller to record patient's urine output data to help in repetitive monitoring protocol. (2) Communication channels (or 'connecting element') refer either to a wired or a wireless, are in charge of providing sensor signal reading and redistributing the signal reading into a storage, and (3) The application (or 'processing element') which post those information to data storage and enables nurses to monitoring the output in real-time.

As mentioned, modern the ICU's monitoring applications are not only provide the current data to the clinical computer system but those data can also be shared with other clinicians who specialised in different areas. We consider our proposed system can be used to integrate with other bedside devices since they are based on microprocessor and having digital communication capabilities. However, these devices have difficult to interface together since each of them uses a different hardware interface and data representation protocol. Thus, custom software and hardware connections must be made to support this consideration. Technology of Cloud computing was applied to the architecture to reduce a cross-platform complexity between different of operation systems, and provide commonality and simplification in the software and hardware connections from embedded system to a web application.

We believed that this design will go beyond simple monitoring system because the ability that allows connect anything to the Internet lets the device operate with much more functional than usual machines. The automated-remotely data collection will be take place in the data acquisition (as a fundamental of the IoT concept) and data will spin out to a remote application where Cloud computing supports intelligently for smart monitoring and may add more sensor to act with the smart devices in the nearly future.

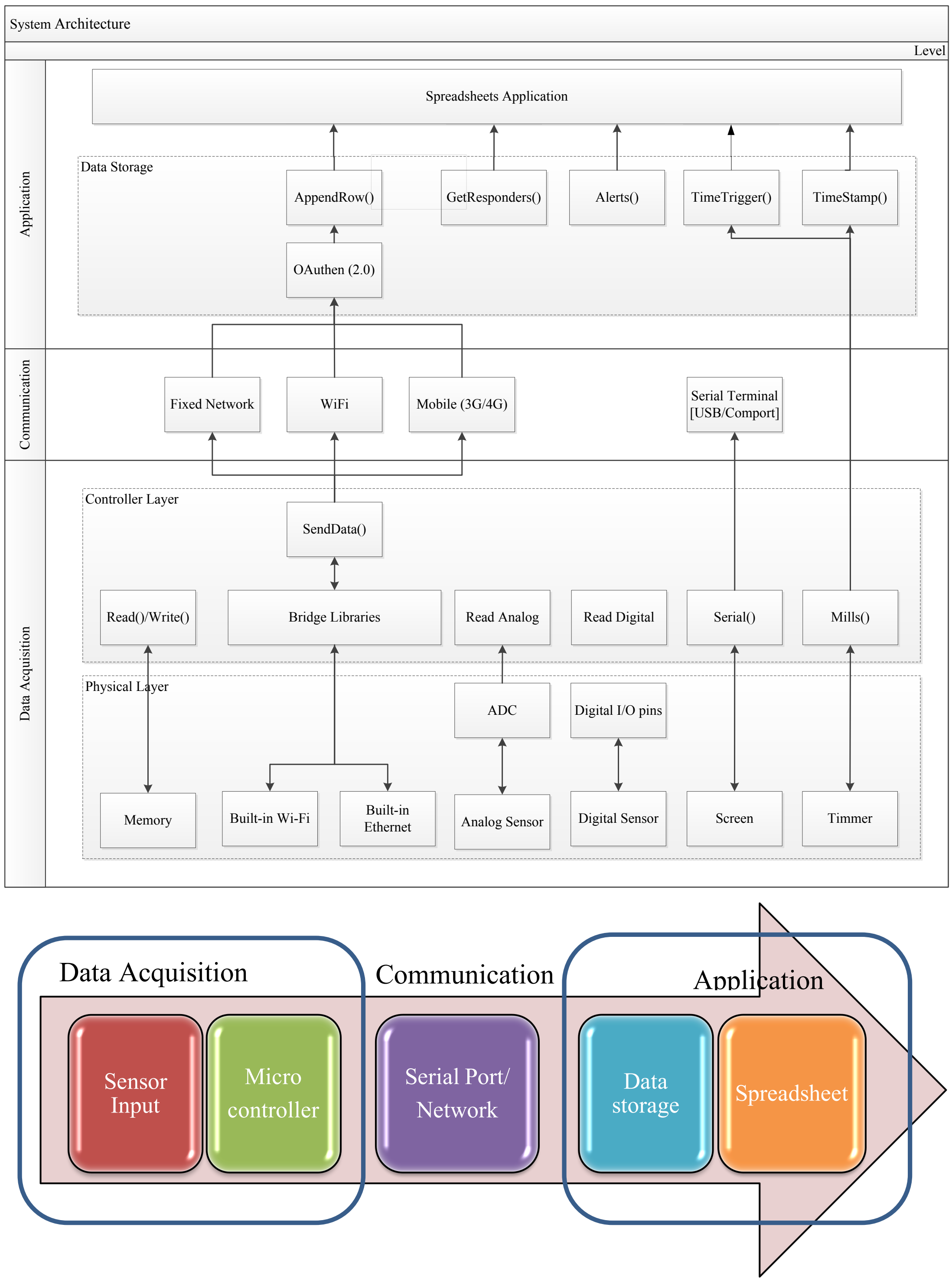


Figure 3.11 Design of the system architecture

3.5 Hardware for Data Acquisition

The hardware architecture of the prototype system can be divided into two modules: (1) *the sensor input (Physical layer)* and (2) the *microcontroller (Controller layer)*. A set of sensors is connected to a microcontroller using jumper wires and breadboard that helps to build and align electronic circuits. The USB cable can be connected between a microcontroller and a computer to provide power source and transmit data to a computer screen in an offline mode. A switch-off switch is included to the active microcontroller when it is desired to monitoring in order to reduce energy consumption. The device can be turn off when it is not necessary to measure.

Data transmission frequent can be configured as minimum as one time per hour. It obtains an increase operation time when microcontroller is connected with external power (e.g. power bank or batteries). Furthermore, the robustness and flexibility are key areas that are often insufficiently fulfilled in research prototype. Some embedded development toolkits were critically reviewed to find a suitable technology which will help reduce developing barriers.

3.5.1 Sensor input

The physical entity of the architecture is performed by sensing by mean of using sensor. The word '*sensor*' means an electronic device used to measure a physical quantity and convert it into electronic signal. It can be classified into two categories. *Analog sensors* most often produce a voltage proportional to measure quality and must be converted to digital with Analog-Digital Converter (ADC) before processing it. *Digital sensors* most often use serial communication to return numeric information (0 for LOW, 1 for HIGH) directly to the controller through a serial port (Dictionary.com n.d.).

Sensing is the key concern for the healthcare application as sensed values are used as input parameters to the system. Haque et al. (2014) stated that sometimes sensing may often be painful. For example, the blood glucose level (BGL) detection for diabetic patients requires pricking the finger and collecting sample. To remedy this, a non-invasive method of monitoring leveraging radio-based sensor is used (Istepanian et al. 2011) and many other non-invasive sensors had been used very widely.

From the literatures, there are different non-invasive sensors that have been used to detect drop or bubble in liquid flow, and these sensors can be divided into three main categories: (1) *Sound technique*, (2) *Optical technique*, and (3) *Image analysing*.

Therefore a feasibility study on the effectiveness in each sensor has been carried out from the pre-prototype, presented in Section 4.1, in order to detect number of counted-drop within the drip chamber.

A. Sound method

I. Ultrasonic sensor

Ultrasound sensors (US) had been proposed for using to measure of the speed of liquid, determine size of objects or particles, distance and range finding, or mapping the image of objects and it has been widely used in many healthcare applications because of its good beam directivity, penetration ability, harmless, high detection sensitivity and far transmission distance (Chen and Wu 2012), and there were two ultrasound urine meters presented in mid of 1980 (Shotts and Hauf 1986; Shabot et al. 1988).

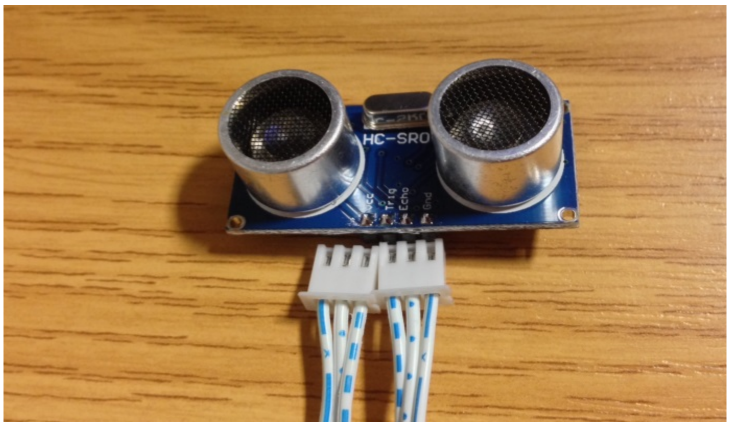


Figure 3.12 HC-SR04 ultrasonic components

HC-SR04 ultrasonic sensors (as shown in Figure 3.12) were used in the prototype. We believed that a small amount of sound energy should be reflected by a dripping in front of the sensors and returned to the detectors. The receiver amplifier will send this reflected signal to a microcontroller which times them to determine how long it takes to receive the reflected wave, by using the speed of sound in air.

From the datasheet, the sensor comes with 4 pins where two are dedicated to power supply and ground. It operates from a 5V DC supply and standby current is less than

2mA. Another two more pins are for active ultrasonic sensor generates high frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. Passive ultrasonic sensor is basically microphone that detects ultrasonic noise that is present under certain conditions. This sensor works by emitting a short burst of 40kHx ultrasound from an active transducer.

For the prototype, the ultrasonic was initially designed to emit 40 kHz of sound waves to detect the crossing drops between the sound fields. This detection method will be started when ultrasonic transmitter emitted an ultrasonic wave in a one direction, and the starts timing when it launched. If the emitted sound wave detects the obstacles on the way, it would return immediately and the different attitude of the tension will be occurred in the receiver, and timer will stop timing when the sound wave returns from the reflection. The constant density of air is known as 1.2 kg.m^{-3} and the speed of sound in the air is around 343 m.s^{-1}. The concept design is presented in Figure 3.13. The technique here is similarly to the bubbles detector device.

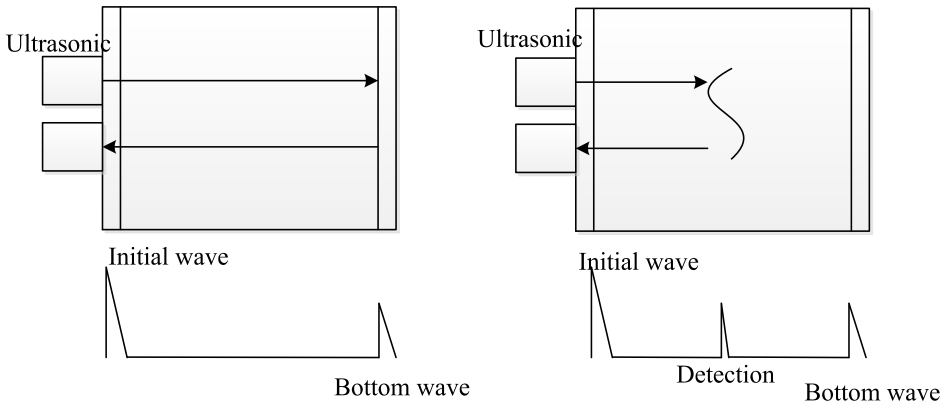


Figure 3.13 Detection concept diagram of ultrasonic pulse reflective technique

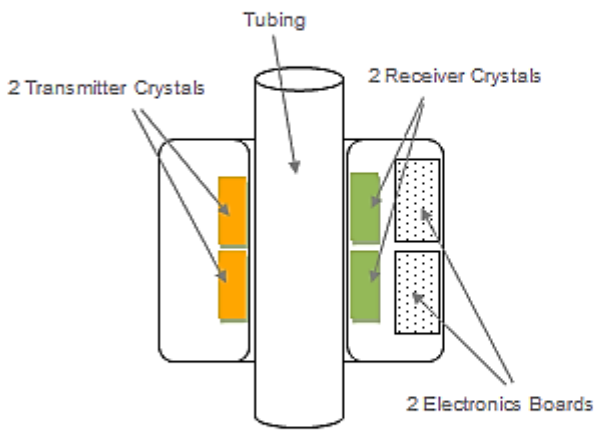


Figure 3.14 The principle of the air bubble sensor (SONOTEC 2015)

According to the timing diagram, presented in Figure 3.15, the sensor started when 10µS of HIGH pulse on the Trigger pin initiated an 8-cycle burst of 40 kHz ultrasonic pulses. The pulse return to the module and it emits on the Echo pin a high pulse between 150 µS and 25 ms. Approximately 58 µS is used to estimate the sound wave travel round-trip in 1 cm., this means sound travels forward and return in distance of 1 cm. The return value will illustrate where the sound waves were reached before it reflected and this value can calculated the reflected time using the equation:

$$L = C * (T/2) \qquad 3.5$$

Where *L* represents time travelled of the sound wave, *C* is the ultrasonic spread velocity 340 m/s, *T* is time where half of the time value was transmitted to receiver.

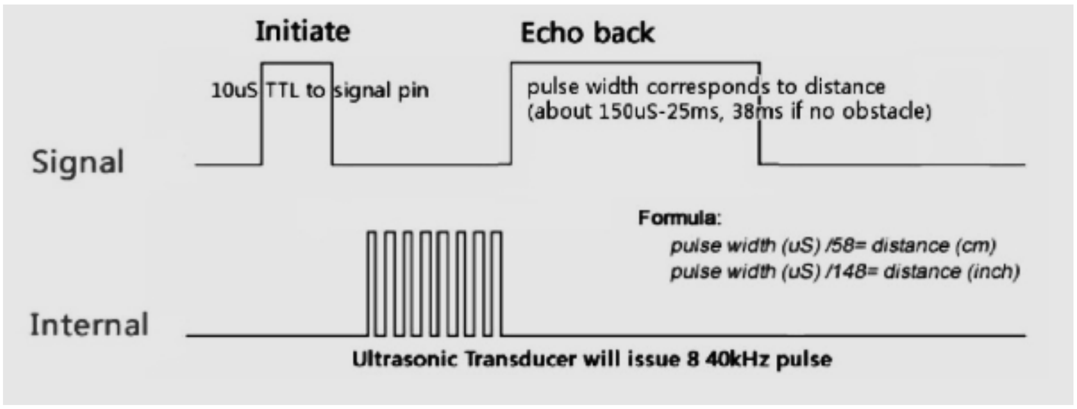

Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance (about 150uS-25ms, 38ms if no obstacle)
Signal
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Internal
Ultrasonic Transducer will issue 8 40kHz pulse

Figure 3.15 Ultrasonic sensor timing diagram

In Arduino community, the source code library is provided for the HC-SR04 module. It was written by Tim Eckel[1]. The code constructor will be provided in the Appendix section.

II. Microphone Sensor

Another approach for sensing an acoustic signal is the use of microphone sensor (MI) which we believed it is an effective approach for participatory sensing. It has been found in common health care applications such as, activity monitoring, noise detector, and hearing aids. Most microphone sensors today are based on electromagnetic or piezoelectricity to produce an electrical signal from air pressure variations. The operation design is to use microphone detecting noise when the drop fell of the dispenser tip and it hits the bottom part of the drip chamber. This should generate a noise where the sensor can detects and computes amplitude sound into digital signals to count numbers of time of the falling drop.

Audio signals in the prototype were recorded via 20 kHz electret microphone assembled with an amplifier, and it works from 2.5 up to 5V. In circuit, the amplifier on the board is wired to give a 2.5 volts signal when there is no sound. When sound has been detected, the return values from function *analogRead()*will show in voltage variations around 512 which is the half range of analog value from 0 to 1023.

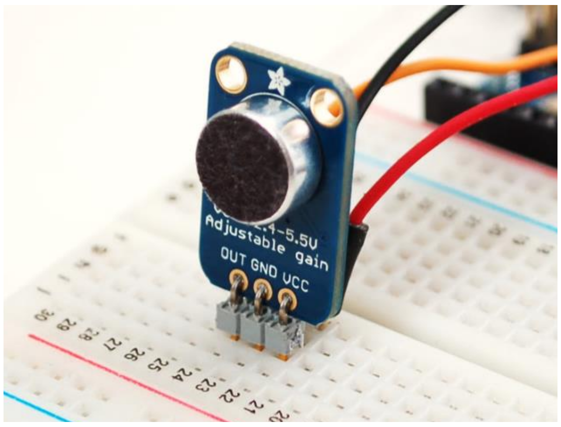


Figure 3.16 Microphone sensor

B. Optical method

An optical sensor is a device that converts light rays into electronic signals which similar to a photo resistor. A change in physical environment will modify such properties of the optical propagating light as its phase, polarization, amplitude or

[1] Appendix 5

spectre (Lecler and Meyrueis 2012). It measures the physical quantity of light and translates it into a form read by the instrument. As report in Ahuja and Parande (2010), one of the features of an optical sensor is its ability to measure the changes from one or more light beams. This change is most often based around alterations to the intensity of the light.

Other features of optical sensors include the distinction of whether it is placed internally or externally in a device (Ahuja and Parande 2012). There are many types of optical sensors found in numerous research and commercial application, many based on the use of lasers, imaging sensors, and fiber optics. If the optical sensor is only used to transmit the light, it is commonly known as an extrinsic optical sensor but if the modulation of the properties of light takes place in the optical sensor due to the physical phenomenon to measure, it is about an intrinsic optical fiber sensor (Lecler and Meyrueis 2012). The comparison of these two types is given in Table 3.4.

Table 3.4 Comparison of Extrinsic and Intrinsic optical sensors.

Extrinsic	**Intrinsic**
Applications measure temperature, pressure, liquid level and flow.	Applications measures rotation, acceleration strain, acoustic pressure and vibration.
Less sensitive	More sensitive
Easily multiplexed	Tougher to multiplex
Ingress/ egress connection problems	Reduces connection problems
Easier to use	More elaborate signal demodulation
Less expensive	More expensive

(Ahuja and Parande 2012)

I. Photo interrupter sensor

Photo interrupter (PI) is one of the extrinsic-through beam sensors. It made of an infrared LED for optical emitter on one side and a photo transition detector on the other to detect the change signal with amplify (Kim et al. 2013). The emitter uses a beam forming optics to project light onto the detector, and detector used to sense if the free path between the emitter and the detector called ‘Photogate’ is blocked (see Figure 3.17).

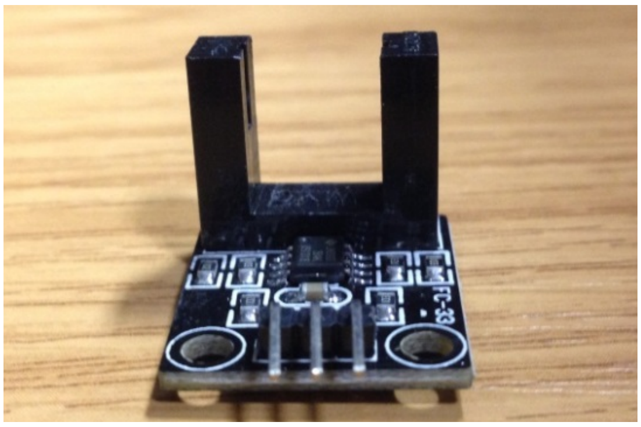


Figure 3.17 Proto interrupter components

According to the technical data sheet, it comes with 3 terminal pins which are: Power supply (VC), Output (OUT), and Ground (GND), respectively. The output voltage of the phototransistor depends on the position of the object between the photogate and it can be obtained from anolog-to-digital converter without an additional amplifier due to its Transistor-Transistor Logic (TTL) voltage level (0-5 v). Normally, the relationship between the output voltage and the position of the object is approximate linear close to the centre of the sensor (Gu et al. 2012), as shown in Figure 3.18. The reflective of output voltage from the phototransistor of the LED light from the drop changes according to the count.

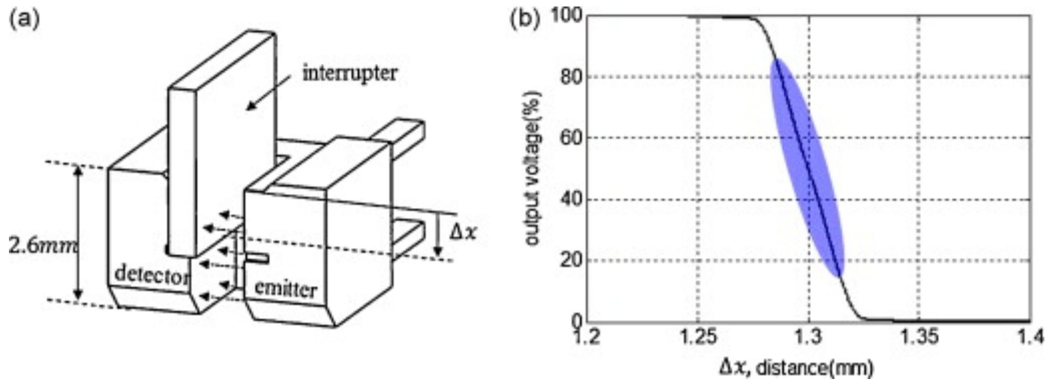


Figure 3.18 The output voltage and the position of the detected object (Gu et al. 2012)

II. Infrared proximate sensor

IR proximity sensors are widely used for sensing the presence of an object, its distance from a reference, or both. It can be classified into an intrinsic type where detector determined the wavelength of emitted infrared light. Common applications include speed detection, sensing of the hand in automatic faucets, automatic counting or detection of, object-edge detection, and many others.

To sense an object, the proximity sensor transmits IR (infrared) pulses toward the object, and then wait to detect any pulses reflected back (Figure 1). An LED transmits the signals, and any reflected signal is detected by a photo-detector. The strength of this reflected signal is inversely proportional to the distance of the object from the IR transceiver. Because the reflected IR signal is stronger when the object is close, this can be calibrated the output of the photodiode detector to determine the exact trigger distance of an object (the threshold distance for making a decision on whether the object is present or not).

There are several research report that IR sensor offers lower cost and faster response in detecting distance measurement of an object compare to the ultrasonic sensor where it is widely found as proximity detectors in robotic researchs[6].

Compare with the ultrasonic sensor, the infrared sensor has higher ability to measure shorter range compared to the sound[7]. The wavelength range of the infrared light is approximately 850 nm ± 70 nm and the transmitter signal from the sensor is depended on the distance of reflecting object, the signal returns at an angle where the distance can be determined[7].

In this study, the measure the small changes in light reflection that can be detected by the infrared sensors. IR sensors with part number GP2D120XJ00F (Sharp Corporation, Osaka, Japan) are specifically for analog output distance measuring sensor. A major advantage using this GP2D120 is the beam width where a signal that transmit from the IR sensor point directly to the require object[7], which is capable in detecting any changes in displacement within the range of 4 cm to 30 cm. Results showed that small changes in displacement can be detected by sensor, and a graph has been plotted for any changes in distance.

C. Image analysing

Another method for the determination of the drop-rvolume is the use of chromatographic or image analysing. In this way, the volume is calculated using geometric shape of drops and a specified reference volume to define number of drop which added to the system. Normally, a digital image acquisition and processing require high-end and complex technologies. For example, a high resolution camera was found in Hugli and Gonzales (2000). It has capability to capture an image at 1/16,000 second exposure time (Hugli and Gonzalez 2000). A CCD video camera was found in Song et al (2003). The video signal is first transformed into a digital signal and placed in the frame storage. It is converted by the vision logic and displayed on the terminal and sent to the computer via an interface for data processing (Song et al. 2003). Moreover, a specific software package and high end computer are needed to acquire and process 2D or 3D images (Del Río and Neumann 1997; Song et al. 2003; Song et al. 2005; Thurow et al. 2009).

To calculate the volume from the drop image, two coordinates pixel position (X, Y) of drop images must been known, then computerise with calculation model to convert those pixel points into volume units. Serval models were proposed (Hugli and Gonzalez 2000; Song et al. 2003; Song et al. 2005; Thurow et al. 2009) and the results of each model has presented with more accurate, efficiently and reliable of measurement drops and contents inside. With the advance technologies, a variety of physical and chemical parameters (e.g. surface tension, concentration, refractive index, turbidity, and chemical constitution) can also be potentially obtained directly or indirectly during a measuring cycle (Song et al. 2005).

Unfortunately, there are some circumstances in which we believe that this method may not yet suitable for real-time processing in the hospital. Firstly, it is required very complex measurement technology to support the measurement, and the expensive equipment are required to integrate hardware and software into a single system for processing online. Secondly, if 10 of non-kidney injured patients admitted in ICU and they have been monitoring urine output, more than forty thousand digital images (considered one image per drop) will be generated daily and they need large computer resources to preform real-time processing. Consequently, a greater bandwidth is needed to transmit all the information and this may reduce a computing time. Moreover, the methods presented were largely based on the development of specific algorithms with the specific hardware to extend its functional scope.

Since our goal is to find less complex solution for a real time measurement. The accuracy of the presented method here is higher than required, and more likely not necessary to monitor urine output as higher precision as it permits. Due to these circumstances, image analysis does not really represent simple method for volume estimation but it could be used as a reference procedure for the estimation and may future develop in next version of prototype. These findings also help define the balance between effort and benefit involved in integrating only some sensor for this automated task as simple and accurate as possible. This simple described the data give reason to predict future observations.

3.5.2 Microcontroller

A microcontroller is a small device on a single integrated circuit containing a processor, ROM memory, and RAM for programmable input and output memory (Howe n.d.). It is a core of the hardware system for controlling signal sending, detected changes from signal, and determined the count to an interruption produced by reflected signal. A microcontroller also controls the communication protocol by choose the correct configured channel by checking the data of the received signals and redistribute them to the either storage using or display them on the computer screen followed the designed communication protocol.

Over the last few years, an open source microcontroller board called 'Arduino' platform was introduced and it was quickly became popular and widely used in scientific community for controlling and monitoring experimental hardware (Faugel and Bobkov 2013). It was designed with the goal of inexpensive and cross-operation platform which runs on most common operation system; included Windows, Mac OS, and Linux, and it is easily to programme (Bell 2013).

Arduino board itself supports both, analog and digital pins that can use to connect to various sensor components. The basic layout of Arduino consists of a Universal Serial Bus connection (additionally Micro-USB for Arduino Yún) which also powers the a microcontroller and exchange information between board and the computer, a reset switch, LEDs for power and serial communication, and a standard spaced set of headers for attaching shields which allow the use of expansion additional capabilities such as Bluetooth, Ethernet, or Xbee (Arduino.cc n.d.). Arduino platform is also provided an intelligence development environment called 'Arduino Integrated Development Environment (Arduino IDE)' where all the source codes are write, comply and upload into hardware. Moreover, the Arduino IDE comes with a code editor and several features such as syntax highlighting, brace matching, automatic indentation to help introduce programming to newcomers who unfamiliar with software development.

A source code written for Arduino is called 'sketch' mainly based on C and C++ programming languages. The sketch only needs define two main function to make an executable cyclic loop program (Margolis 2012) as shown in Figure 3.19.

- setup () : a function runs once at the start to initialize all setting.
- loop () : a function called repeatedly cycle until the power is off

```
#define LED_PIN 13

void setup() {
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_PIN, HIGH);
    delay(1000);
    digitalWrite(LED_PIN, LOW);
    delay(1000);
}
```

Figure 3.19 Sample of Aruino sketch code to make LED blinks

The official Arduino boards were introduced and commercially produced in twenty versions. Most of them have same functional with an original board, but they have different in characteristics. To understand the compatibility of these technologies, six different versions of Arduino boards were selected and compared based on several criteria, such as: energy consumption, number of input/output pins, memory, and communication protocol (as shown in Table 3.5).

Table 3.5 Properties of Arduino boards

Board	**Processor**	**Operation Voltage**	**Digital/Analog I/O Pins**	**Memory [KB]**	**Serial Port/USB**	**Communication**	**Note**
Uno	ATmega328	5 V	14/6	32	1 port with Regular USB	USB-to-serial chip, Built-in Ethernet	Required additional shield for wireless connection.
Due	AT91SAM3X8E	3.3 V	54/12	512	4 ports with 2 Micro USB	2 USB-to-serial chip	Required both Wi-Fi and Ethernet Shield for Internet connection. Voltage and pin layout are different.
Mega	ATmega2560	5 V	54/16	256	4 ports with Regular USB	USB-to-serial chip, Built-in Ethernet	Required Wi-Fi shield for wireless connection. Supported Android Development kit for mobile app.
Mini	ATmega328	5 V	14/8	32	1 Port with no USB	Interfacing with other software	Required additional components to connect to Wi-Fi and Ethernet Shield
Nano	ATmega168 ATmega328	5 V	14/8	16	1 port with Mini-B USB	USB-to-serial chip, WiFi/Ethernet Shield	Pin layout is different and does not have a DC power jack
Yún	ATmega32u4	5 V	20/12	32	1 Port with Micro USB	USB-to-serial chip, Built-in WiFi/Ethernet	Contains an application processor running Linux, and modern flexible connectivity.

Arduino Yún is used because of its remarkable in flexible connectivity. According to a reviews described in an Arduino official blog (https://www.arduino.cc/), Arduino Yún is based on two processors, ATmega32U4 and Atheros AR9331. It also has built-in Ethernet and Wi-Fi support, a micro USB, micro-SD card slot, 20 digital input/output pins (of which 7 pins can be used as Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means output and 12 pins as analg inputs), a 16 MHz crystal oscillator, and a 3 reset buttons (as shown in Figure 3.20).

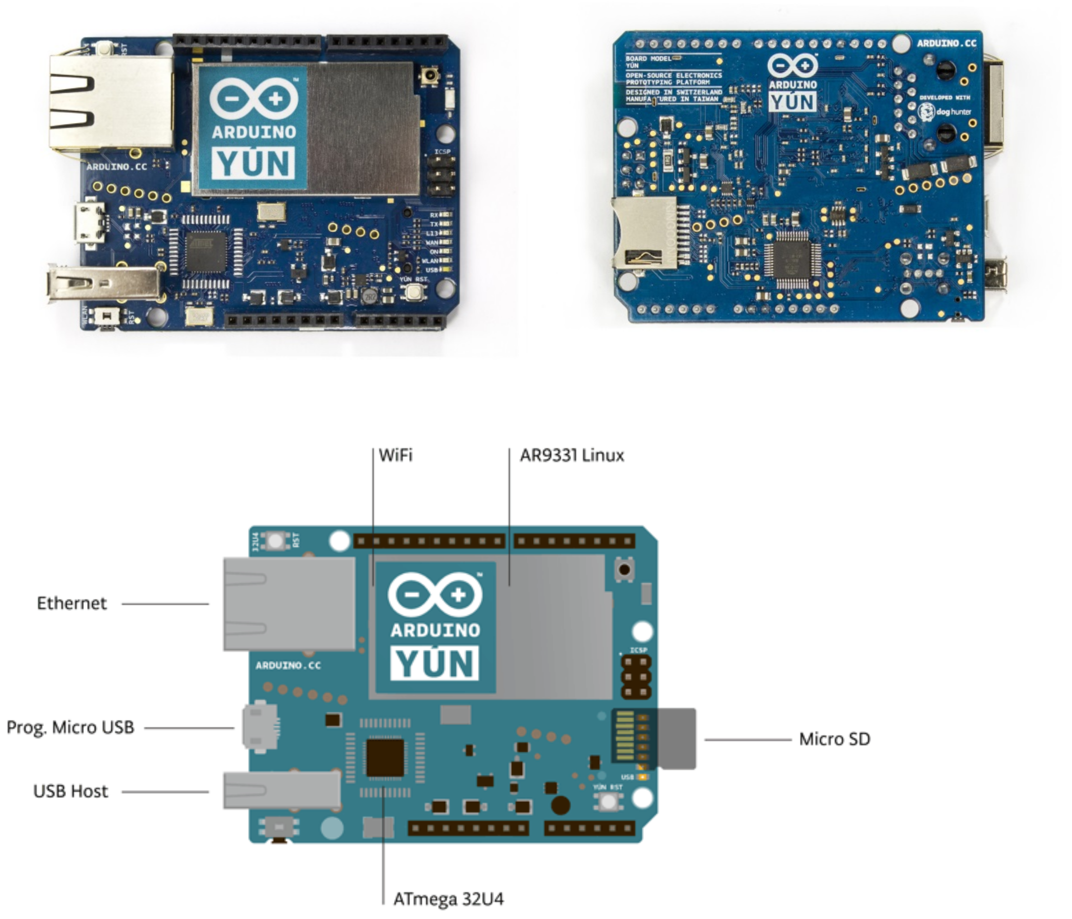


Figure 3.20 Arduino basic layer (Front and back side) (Arduino.cc n.d.)

Arduino Yún has the ability to act as an access point, but it can also connect to an existing network with WiFi 802.11b/g/n support as well as security algorithm networks that support Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), and Wi-Fi Protected Access II (WPA2) encryption. When the Yún is connected to a wired network with an Ethernet cable, it automatically connects with the DHCP. To give static IP address, or make it act as a DHCP server, can be done by configure in the computer's interfaces.

With all built-in facilities, Arduino Yún distinguishes itself from other boards in that it is able to communicate with Linux distribution on board. A wide range of user-space application, drivers and file management system are the main advantages of having Linux operation on Adriano board because it offers a compact solution for possibilities with developers respect, and it is a step forward in the Internet of Things. The prototyping time can be significantly reduced through the use of shell and python scripts that are typical of Linux systems (Balducci 2013).

A comparison the relative strength and weakness of other toolkits associate with other manufacturers had been made, as shown in Table 3.6. The relative strengths and weaknesses associated with each of the prototyping toolkits reviewed during the design phase. The strengths and weaknesses help to select a suitable tool for the usage during development process, based on its characteristics.

Table 3.6 Summary of strengths and weaknesses from the existing toolkits

Toolkits	Strengths	Weaknesses
d.tools	• Wide choice of I/O devices, including embedded screen. • Plug & Play components. This makes prototyping practicable. • IDE makes development easier than Linux.	• Large amount of supporting in hardware is required. This may increase complicated barriers towards rapid development. • Costumed interface platform is not widely used.
Beagleboard	• Better hardware specification and good set of input/output features. • Installed Linux operation system.	• Lack of wireless communication, makes it less mobility. • Does not have a supported user graphic interface. For beginner users, this makes Beagleboard harder to develop source code in command terminal.

Raspberry Pi	• Linux computer with video processing. • CPU is 40 times faster and has capability to run multitasked processes. • Have large and active communities to support to support.	• Lack of interface with external sensors.
Intel Edison	• Intel Atom system-on-a-chip. • Integrated Wi-Fi, Bluetooth, memory, and storage. • A web server on board. It supports mainly Internet of Thing platform.	• Costly • Required advance programming skills.

All models are all readily available, affordable, portable size (around 2x3 inches) and they can all be used for creating digital gadgets or modern IoTs. Each of them has the differences that make each of these gadgets shine in their own type of applications.

Firstly, the Arduino, Raspberry Pi, and BeagleBone are found inexpensive at under £30. Also worthy of note is that processor clock speed of the Arduino is about forty times slower than the other two. This makes the BeagleBone and the Raspberry Pi have a better performance. The reason for this is that they both run the Linux OS. This OS makes these platforms as tiny computers which are capable of doing multiple tasks at the same time. However, these two still require external flash memory (SD Card for Raspberry and Mico-SD Card for BeagleBone). In contrast, the Arduino is simple. It can run one program and it programmed in level C++.

Secondly, Arduino is found to easily interface with a wide variety of sensors and effectors without external circuitry. It is recommended for applications that need to interface with external sensors. There are different version of Arduino board the operate at different voltages (3.3 – 5 V) to make it easier to connect to external devices, and they come with a small size, inexpensive, embedded system on a chip microprocessor from Atmel. BeagleBone only operates with 3.3V and will require a resistor or other external circuitry to interface to some devices. Both the Arduino and BeagleBone have analog to digital interfaces that easily connect components that output varying voltages.

Next, the Arduino is recommended as the least power used of the bunch, although, the BeagleBone is better in terms of computer power per watt. However, the Arduino has can work with a wide range of input voltages. This allows it to run from a variety of different types of batteries and keep working until the power is off.

Lastly, for applications that connect to Internet, the BeagleBone and Raspberry Pi and Intel Edison are considered better than most of the Arduino boards because they are based OS as tiny computers with Ethernet interfaces and USB, so they can connect to the network relatively painlessly. Edison is at the high end of the brunch where it comes with all built-in components that are needed to build an IoT projects and support advanced network capabilities. However, Intel Edison is cost twice but the processor speed is lower than others because it is meant to be deeply embedded IoT computing which reduce the power consumption for the portable application.

More recently, the Yún was introduced to compete with other toolkits, an application processor running Linux, and modern flexible connectivity (e.g. Ethernet, WiFi, and USB) with the same form factors as the typical Arduino. The Yún can be ordered in store for £38 which is a price that at a first glance competes with BeagleBone. But the applications for which these two products are targeted are very different. The Beagle boards are meant to run all process in full Linux environment, while the Yún is made to interface the sensors and actuators with the networked (either wire or wireless) or 3rd party web services.

3.6 Communication channels

In the hospital, two groups of patients can be found. First group are patients who cannot move and lay down on the bed. On other hand are patients who can move around the hospital's ward. In order to achieve the objective of real-time monitoring, both wired and wireless communications should be provided to use in both situations.

Wired connection is provided for a fixed location system for patients who cannot move and spend time in their bed. Most microcontrollers are supported through a serial communication over the USB and appeared as a virtual comport. Serial monitoring interface is included in Arduino IDE allows simple textual data to be sent from a microcontroller and displayed on the computer screen. Moreover, a microcontroller can be powered via USB without require an external power supply. Another wired connection within local area network (LAN) is provided using Ethernet protocol to the Internet. It is a trend to be more reliable, better performance in a local-based environment with a theoretical maximum speed of 100 Mbps, and likely more secured than a wireless connection. However, a wired network required physical cables that connect a machine to the router. Although this does not lend the design well to mobility, wired connections typically cost less and reduce the battery life of the microcontroller.

Wireless connection is provided to support mobility system that can be carried along within the hospital. An available add-on part such as WiFi Shield, it allows the connection to microcontroller board and the Internet wirelessly using 802.11 wireless specifications (Wi-Fi). For the most part, wireless connection is reliable. In this way, a proposed prototype system can be measured and collected information in real-time from the sensor. Thus, a proposed design is more likely to be achieved according to a major challenge for a continuous and seamless connectivity in this case.

3.6.1 Network setting

The structure of the IoT could have the similarity as the other Internet applications. In the low level, the device needs to be connected with Internet and other online applications, where the applications communicate using HTTP or HTTPS. The data is transmission similarly to the current done by either using DNS or TCP/IP. According to the article ‘Networking Best Practices for Large Deployments[2], it said that Google apps exist in a multi-tenant server environment. All the service shares the same IPv4 address space with a specific hostname, such as `mail.google.com` or `drive.google.com`. Hence, the network and specific protocol are presented in as Table 3.7.

The generally hospital’s network is constructed based on the enterprise network architecture where multiple devices (i.e. mobile phone, workstation, IP phone, surveillances and monitoring devices) are requiring IP network connectivity, large capacity network infrastructure is provided to support high bandwidth demands for

[2] http://goo.gl/L0ZzlG

telemedicine, and wireless facility is available for mobile carts and wireless medical devices and also covers to visitors. Both wired and wireless combination usually found in the hospital infrastructure where the original wired network architecture deploying the wireless network to realize the wireless coverage of all building and wards, similarly to Figure 3.9.

Table 3.7 Protocol stack of the designed prototype

Protocol stack		
Layers	**Web-application (Internet Protocol)**	**Prototype (Arduino)**
Application Layer	HTTP/HTTPS	HTTP
Transport Layer	TCP (APIs)	TCP
Network Layer	IPv4	IPv4/IPv6
		Ethernet: 802.3
		Wireless: 802.11
Data Layer	XML/JSON	XML/JSON

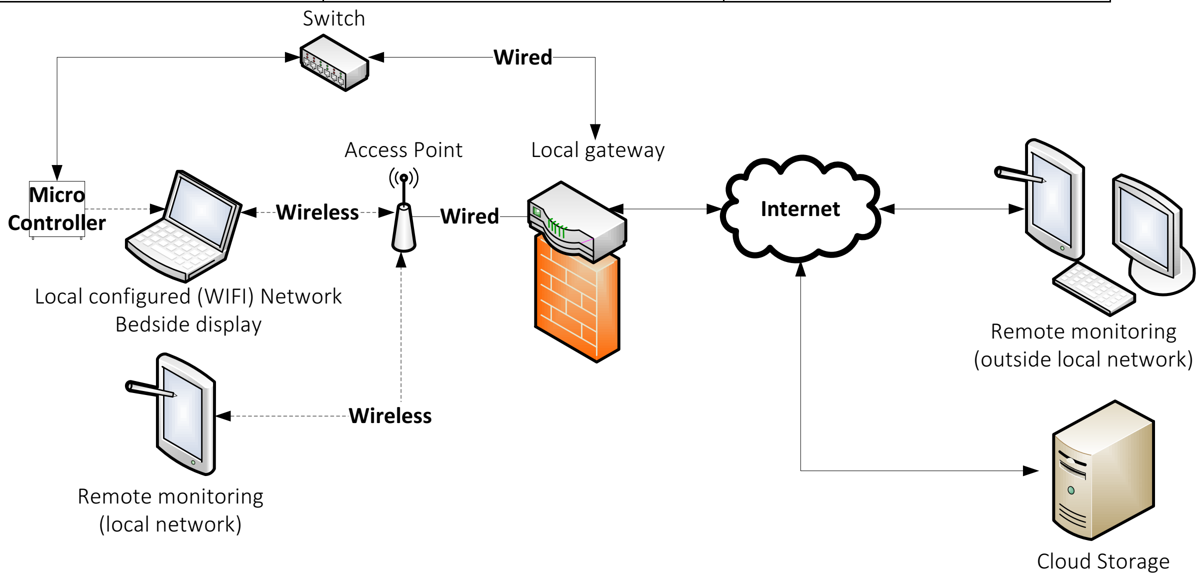


Figure 3.21 Simulated hospital network setting

A. Create a virtual Wi-Fi network

The Wi-Fi is based on IEEE 802.11 are the common standard of constructing hospital wireless network. The wireless access points are used to manage a centralized authentication and control access to a network. To simulate the similar environment, a local virtual Wi-Fi network was created. It had been configured to connect to another network and provides Internet connectivity as term of *'virtual router'*. At the end, the data from sensor reading will be uploaded on cloud storage. This Wi-Fi network was followed the latest IEEE 802.11n standard which is currently the fastest option for the wireless system, and it commonly supports over IP and MAC address. This Wi-Fi network uses SSID parameter for network name identification and KEY parameter for a WPA2 Personal Encryption with minimum of 8 characters password.

B. Configuring the on-board Wi-Fi

This virtual network also has capability to integrate with other features from the Wireless Hosted Network API's built-in that came with a modern System Operations. Two scenarios were tested:

- One computer laptop using its Ethernet connection for the Internet access and its Wi-Fi adapter to broadcast.
- A mobile modem avaliable with 3G or 4G connectivity.

Ethernet and Wi-Fi connection

Arduino Yún has two processors on board. One is an ATmega32U4 and another one is an Atheros 9331 with Linux OS and the OpenWRT wireless stack called, '*OpenWrt-Yun* that enables the board to connect to WiFi and Ethernet networks. OpenWrt-Yun is described as a Linux distribution for embedded devices which supports IEEE 802.11b/g/n for Wi-Fi connection and IEEE 802.3 10/100Mbit/s for wired port.

When interfacing with the OpenWrt-Yun system, there are two methods to configure the system: using command line or a web page that allows configuring. The web interface called 'LuCi' gives the access to the setting as needed for maintaining the Wi-Fi connection. We preferred to use web interface instead of command line because it reduces all the complexity and offered a fast and straightforward experience in setting up the Yún. Moreover, it provides an alternative way to use web APIs to communicate to Arduino sketch directly through browser.

When the Yún is powered, it will automatically act as an access point and it will create a Wi-Fi network named 'ArduinoYun-xxxxxxxxx', and then makes a connection to this network useing computer's network management. Once the Arduino board obtained an IP address, we navigate a web browser to 'http://arduino.local' or '192.168.240.1'.

Select the specified an ad-hoc network, security type, and enter the password. The Arduino Yún is able to connect to unencrypted networks, as well as networks that support WEP, WPA, and WPA2 encryption. After pressed the 'Configure & Restart icon', the board will be reset itself and join the specified network. The Arduino's network will shut down after a few moments.

Once the device have been identified and successfully become "Online". It can be operated and monitored by using the Bridge library. Bridge commands from the 32U4 are interpreted by Python on the AR9331. Its role is to execute programs on the Linux side when asked by Arduino, provide a shared space for sharing sensor reading data between the Arduino and the Internet, and receiving commands from the Internet and passing them directly to the Arduino.

3G and 4G mobile network

After the board is become 'Online', it is required to install the modem driver and related packages before configuration the specific modem and network. To do that, the following packages were installed to the board using the command line over Secure Shell (SSH). The command are available from the OpenWrt.org[3]:

```
oot@Arduino:~# opkg install kmod-usb-serial-option kmod-usb-serial-wwan
luci-proto-3g usb-modeswitch-data usb-modeswitch
```

[3] https://github.com/openwrt/packages

Table 3.8 Packes for mobile modem connection

Packages	Description
`kmod-usb-serial-option`	Modem drivers.
`kmod-usb-serial-wwan`	Modem drivers.
`luci-proto-3g`	Web-interface for configuration of 3G devices.
`usb-modeswitch-data`	Database for modeswitch
`usb-modeswitch`	Handles mode switching devices.

For 3G/UMTS and 4G/LTE networks the following settings are essential to get the modem working, see Table 3.9:

Table 3.9 Configuration detail to connect Arduino Yún to 3G modem

Required parameter	Description
APN	uk.lebara.mobi
SIM PIN	None (if any)
Username	wap
Password	wap
Modem interface	/dev/tty/USB0

3.7 Software design

From the interviews, the target users of this system are mostly nursing staffs. They gave the clearly requirements that the user interface should be simple and intuitive. As the system is not necessarily aimed at high computer skilled users, it should not require any special training or high level of computer knowledge to successfully run a prototype.

In general, a user interface should be clear, concise and easy to use. Three keywords sum up the main aims of good design.

- Simplicity
- Consistency
- Familiarity

Likewise, the complex operation functions should be minimised and the results should be presented in a clear and concise manner displayed similar to a familiar form, currently used in the hospital.

When creating the user interface for the system, two stage design process was done; *Design* and *Implementation*. The design stage is focused on verifying the best way to enter data and control the input of that information as well as identifying the optimum way to display the result. This way mostly conducted by interviewing and observing the processes done in the hospital. The second stage included coding, testing, and improving interface to the overall requirements.

3.7.1 Given requirements

- User interface design required a good understanding of the user's needs.
- Summary daily data in form of chart or graph with the trend line.
- Alerts to responsible nurses when trigger criteria is not fulfilled:
    - Catherised patient produced urine output less than expected hourly guideline (see Table 3.10).
    - Non-catherised patient produced urine output less than 8 hourly guideline.
    - Daily goal not being achieved.
- Hourly rows background is set to 'Yellow' as defaulted to reminder the nurse to calculate 4 hourly running total volumes.
- Display the information either on the computer or portable tablet.
- Use calculated weight based on height as shown in guide below.

Table 3.10 Urine measurement guideline form the Royal Bournemouth Hospital

Height (inches)	**Height (cm)**	**Calculated weight (kg)**	**Hourly**	**8Hrs**	**16Hrs**	**24Hrs**
54 - 61	138- 155	40 - 50	20	160	320	480
62 - 67	158 -169	51 – 60	25	200	400	600
68 – 72	170 – 183	61 – 70	30	240	480	720
73 – 77	184 – 195	71 – 80	35	280	560	840
≥ 78	≥ 196	≥ 81	40	320	640	960

The rules were developed by condition statement expressions (also known as 'IF-ELSE logical functions') which perform in different conditions depending on whether a criteria-specified Boolean evaluates to true or false. In this case patient's height and calculated weight are the important parameter to check which statement patient is suitable.

The following statements are defined according from the given guideline:

- If patient has height (inches) between 54 and 61 and/or calculated weight (kg) between 40 and 50. The minimum urine produced hourly should be 20 ml, 8hourly should be 160 ml; 16hourly should be 320 ml and daily should be 480 ml.
- But if patient has height (inches) between 62 and 67 and/or calculated weight (kg) between 51 and 60. The minimum urine produced hourly should be 25 ml, 8hourly should be 200 ml; 16hourly should be 400 ml and daily should be 600 ml.
- But if patient has height (inches) between 68 and 72 and/or calculated weight (kg) between 61 and 70. The minimum urine produced hourly should be 30 ml, 8hourly should be 240 ml; 16hourly should be 480 ml and daily should be 720 ml.
- But if patient has height (inches) between 73 and 77 and/or calculated weight (kg) between 61 and 70. The minimum urine produced hourly should be 35 ml, 8hourly should be 280 ml; 16hourly should be 560 ml and daily should be 840 ml.
- But if patient has height (inches) more than 78 and/or calculated weight (kg) more than 81. The minimum urine produced hourly should be 40 ml, 8hourly should be 320 ml; 16hourly should be 640 ml and daily should be 960 ml.
- Else, patient minimum urine produced should be 20.

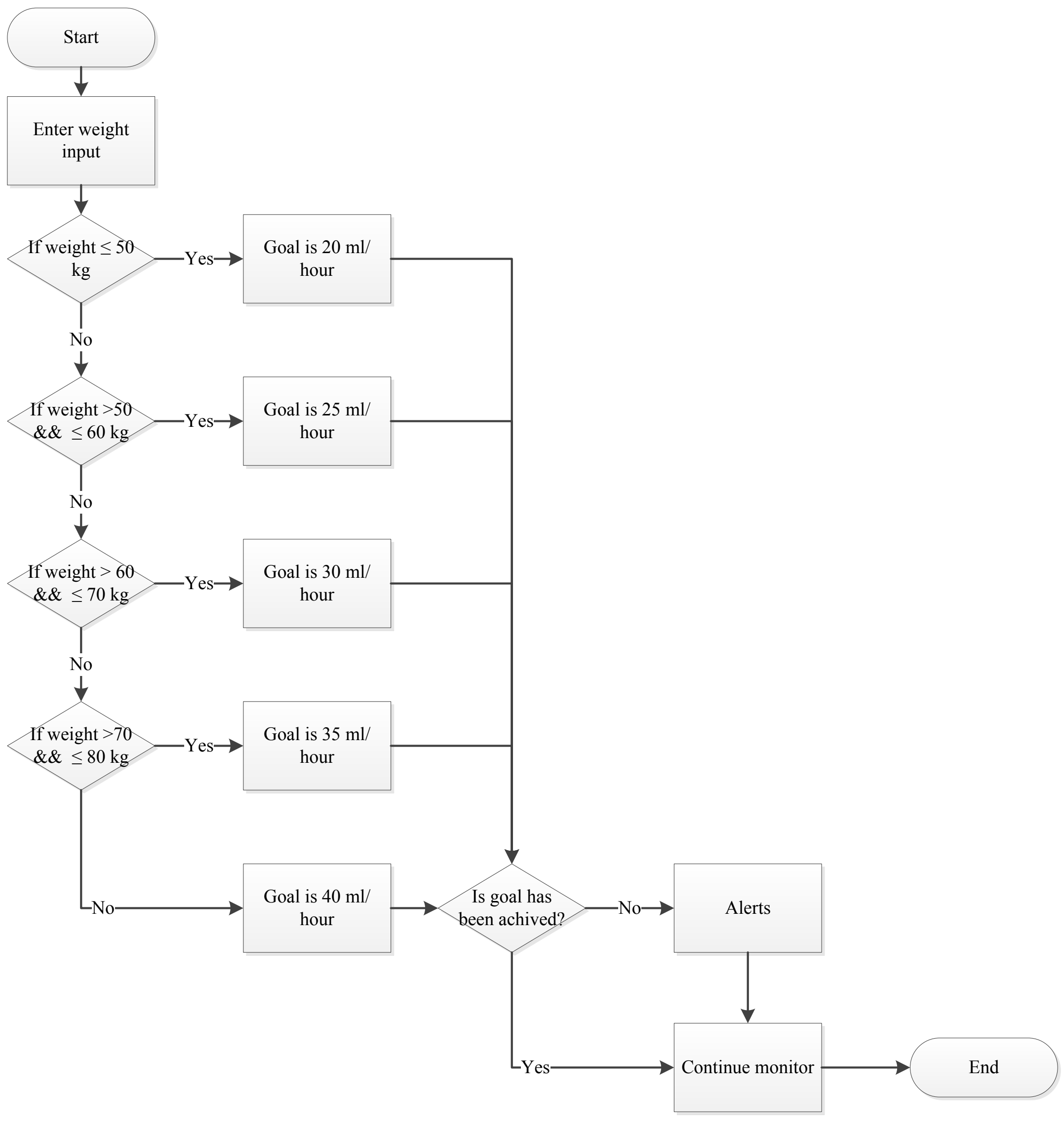
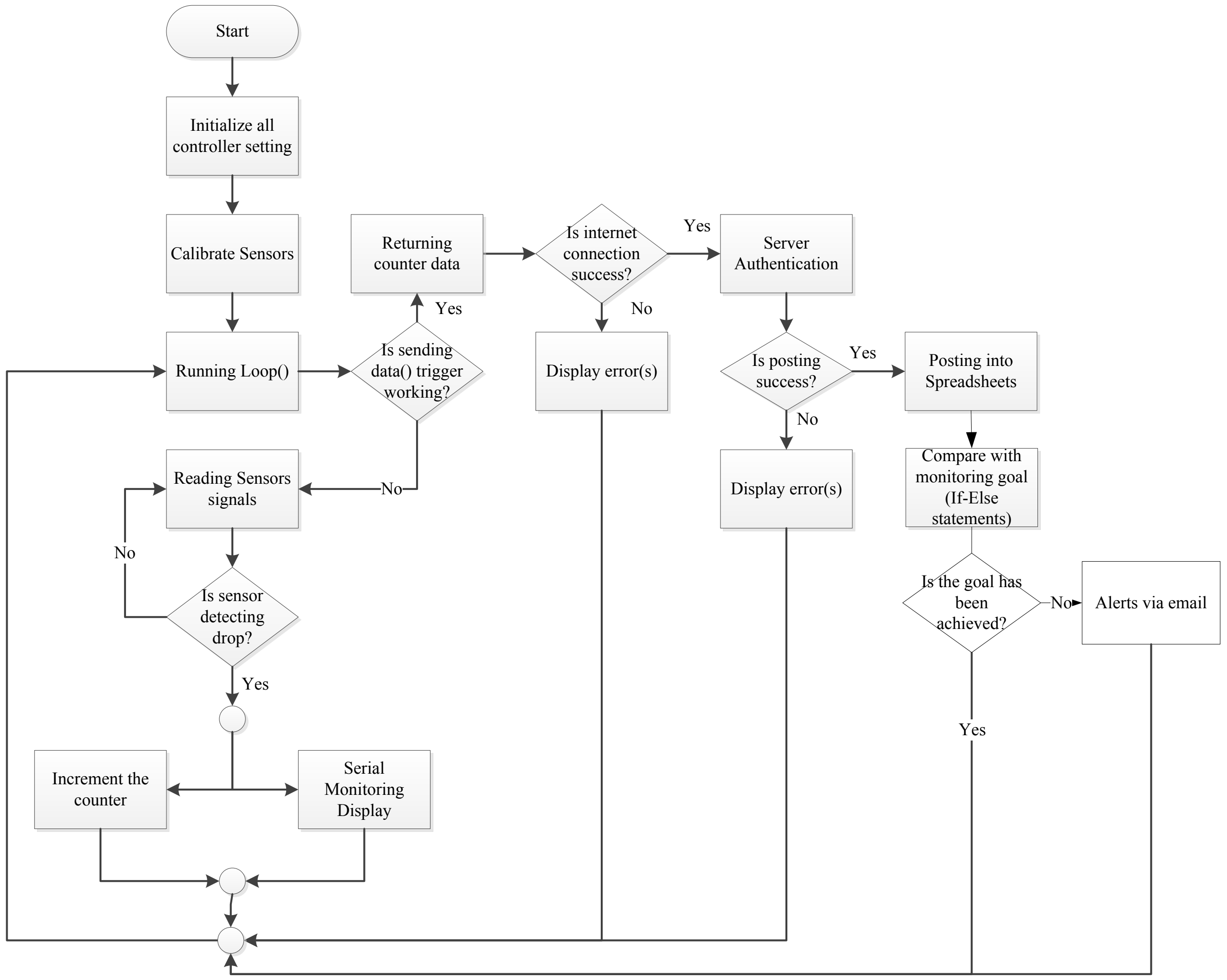


Figure 3.22 If-Elase logical functions for rine measurement guideline



Start
Initialize all controller setting
Calibrate Sensors
Running Loop()
Is sending data() trigger working?
Yes
Returning counter data
No
Reading Sensors signals
Is sensor detecting drop?
No
Yes
Increment the counter
Serial Monitoring Display
Is internet connection success?
Yes
No
Display error(s)
Server Authentication
Is posting success?
Yes
No
Display error(s)
Posting into Spreadsheets
Compare with monitoring goal (If-Else statements)
Is the goal has been achieved?
No
Alerts via email
Yes

Figure 3.23 System flow chart

The operation flow chart is designed and presented in Figure 3.23. It starts when the microcontroller is powered. The system will called the function *setup*() to initialize variables, pin modes and using included libraries. The setup function will only run once, after each power up or reset of the microcontroller board. Then, the Arduino takes sensor readings for five seconds during the start-up and tracks the highest and lowest values it gets from the analog signals. These sensor readings during the first five seconds of the sketch execution define the minimum and maximum of expected values for the readings taken during the loop.

After initialized and set the initial values, the *loop*() function will run loops consecutively according to programing. In the *loop*() part, the most important is detection function. If the drop is detected the number of drops will be added in to declared variables and display number of drops in the screen, while the timer is modified using *millis*() function to return the number of milliseconds since the Arduino board began running the loop cycle. When the timer is reached 10 minutes or 600,000 ms (as defaulted setting), the *dataSend*() will be called where it returns the value of total drops in .CVS object and set it the counted value back to 0, as consequently.

Next, the ATmega32U4 will invoke the communication channel to the Atheros 9331 processor using *Bridge.begin*() function. It is a blocking function. Once it has been called, the operation will be paused nearly 1 second until it has completed. If there is connectivity between microcontroller and the Internet, it will return code of zero indicates the success of connection. If not, it means connecting failure; the function returns and displays any error messages.

Once it had succeed, the operation will continue proceed to identify user performed to an authentication server. If the *dataSending()* is completed the .CSV object will be posted into Google Spreadsheets, but if not, will read and display any error messages. In the Google Spreadsheet, the built-in plotting functions were also used to plot received data as it comes in.

It is designed to establish communication between two different platforms because it acquires a signal reading from the sensors and microcontroller board, which is processed. This processing implies a control signal, at the same time, the system will send data to storage and display in computer device screen. For this reason, the serial communication must support both wired and wireless while the system is acquiring data, a control signal is received reading input from the sensors to generate a particular function to send data into the Cloud.

3.7.2 Data acquisition

Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer (National Instruments n.d.). As described in Guo et al (2014), defined the term of automated data acquisition is receiving, analysing and displaying important information without intervention from the operator (Guo et al. 2014). The examples are included the widespread use of level detectors, bubble detectors and pressure alerts and alarms to notify the abnormal clinical status (Raschke et al. 1998).

To facilitate running data acquisition module, we adopted a simple and extended framework from TEMBOO (https://temboo.com/). It was introduced in 2013 as a repository of programming processes that allows developers to connect to code utilities, databases and more than hundred APIs for online services, and recently partnered with Arduino (Pullen 2014). TEMBOO is one of the invaluable developer tools to application builders. It will save thousands of working hours and, their integrated service between hardware and software means for the future of the Internet of Things, also known as, hardware-based online processing device (Rodriguez 2013). Connecting to Internet services is the greatest advantage of Arduino Yún and TEMBOO service. The communication between hardware and software platform works by having a comma-seperted values file (CSV) formatted data record send TEMBOO. Table 3.11 is an example inputs needed to be specified to those processes.

Table 3.11 CVS file for information sending to Google Sheets

Data	**Describes (Notes)**
Time (Minutes)	Return value of timeStamp() function: (millis()/1000/60)
countPI	Return value of the sum Photo interrupter sensor increment.
countUS	Return value of the sum Ultrasonic sensor increment.
Example:	`String rowData(now);`

```
rowData += ",";
rowData += sensor1Values
rowData += ",";
rowData += sensor2Values


CVS formatted String = ‟12:00″,240,158
/*(Time, Value 1st Sensor, Value 2nd
Sensor)*/
```

A general standard for the CVS file format does not exist (Shafranovich 2005). However, the common format for the .txt/.cvs Internet media type registered with the Internet Assigned Numbers Authority (IANA) seems to be followed by most implementation rules:

1. The last record in the file may or may not have an ending line break, and the file does not required a specific character encoding, byte order, or line terminator format.
2. All records should have the same number of fields, in the same order.
3. Data within fields is interpreted as a sequence of characters, not as a sequence of bits or bytes.
4. Adjacent fields must be separated by a single comma. However, CSV formats vary greatly in this choice of separator character. In particular, in locales where the comma is used as a decimal separator, semicolon, TAB, or other characters are used instead.
5. If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be escaped by preceding it with another double quote.

Time	**Sensor**	**Model**	**Total No. of Drops**
19:27	IR	E350:“ Extended Edition”	
19:37	US	Venture (HC-SR04)	“OUT OF RANGE”
19:47	PI	OMRON EE-SX461-P11	1285

```
Year,Make,Model,Description,Price
“19:27”,”IR”,”E350:”Extended Edition””,””
“19:37”,”US”,”Venture(HC-SR04)”,””OUT OF RANGE””
“19:47”,”PI”,"OMRON EE-SX461-P11”,1285
```

Figure 3.24 Example of common CVS file format

Before sending the string into Spreadsheets, a cross application communication between Arduino board and Google Sheets is required to generate an authorisation URL that an application can use to complete the authentication process. As stated in Google Developers blog, any authentication requests to the Google Engine API for non-public user must be authorised by an authenticated user while keeping the username, passwords, and other information private (Google Developers 2015).

3.7.3 Spreadsheets and User interface module

The problem found from the process of passing information across the borders of paper works to electronic records is a labour-intensive task and resource consuming. In order to reduce nurse workloads, we aimed to collect real-time data by using online services from programming repository providers. Those services are leading our proposed system becomes more virtualized. The term of virtualization refers to the method of allowing the systems to run independently of one another, sharing the same underlying resources (Jansen and McGregor 2008), this means our proposed system will be run with a few lines of code with

A. Information Transmission Module

In traditional client-server authentication model the client requests an access-restricted resource (protected resource) on the server by authenticating with the server using the resource owner's credentials. In order to provide third-party applications access to restricted resources, the resource owner shares its credentials with the third party. This creates several problems and limitations, such as 3rd party applications required to store the resource for future use, typically a password in clear-text, or Servers are required to support password authentication, despite the security weaknesses.

OAuth foundation addresses these issues by introducing an authorization layer and separating the role of the client from that of the resource owner. The OAuth 2.0 framework is an open standard for authorisation provides client applications a secure delegated access to server resource on behalf of resource owner which was originally created in late 2006 (Hardt 2012; Jones and Hardt 2012). OpenID Connect performs many of the same tasks as OpenID 2.0. Whereas OpenID Connect required of using resource owner’s credentials to access protected

Recently, Google has announced the code engine migration from OpenID version 2.0 to OpenID Conncet which is an authentication layer on top of the OAuth 2.0 protocol for authentication and authorization by April 20, 2015. The Google OAuth 2.0 endpoint supports web server applications that written in most common programming languages and frameworks, such as PHP, Java, Python, Ruby, and ASP.NET.

The most widely used protocols and OpenID and OAuth. These protocols have slightly different functionality and use cases. OpenID provides a user authentication only, wherare OAuth provides an access to an API of a provider site.

OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol in Google Web Server Application. It enables clients to verify the identity of the end-user based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the end-user as illustrated in .

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Figure 3.25 OpenID Connect protocol flow (Hardt 2012)

Client is defined as an application making protected resource request on behalf of the owner and with its authorization. This does not imply any particular implementation characteristics.

Resources Owner is defined as an entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.

Authorization Server is defined the server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

Resource Server is defined the server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.

A. The client requests authorization from the resource owner. The authorization request can be made directly to the resource owner, or preferably indirectly via the authorization server as an intermediary.
B. The client receives an authorized grant, which is a credential representing the resource owner's authorization. The authorization grant type depends on the method used by the client's request and the types supported by the authorization server.
C. The client requests an access token by authenticating with the authorization server and presenting the authorization grant.
D. The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token.
E. The client requests the protected resource from the resource server and authenticates by presenting the access token.
F. The resource server validates the access token, and if valid, serves the request.

The combination of a data acquisition module and information transmission module are collecting data from sensors and sending them from the controller to the storage. Two type of signal were received from sensory input, and it is applied directly from both analog and digital pins. *Digital signals* were received from Photo interrupter and Ultrasonic. Generally, the digital output value is kept at 0 until interruption occurs between emitter and receiver; the output will pull up to 1. *Analog signals* were received from IR sensor in voltage corresponding to the detection distance. Data obtained from the sensor reading is saved in a text file (array) in CVS format for post processing. The serial monitoring function is included in Arduino IDE installed on a computer, receives the reading from sensors through a serial port and display on the screen at the moment when sensors were detected signal. Another, monitoring application is Google Sheets where data is stored on cloud engine and used for later stage analysed (more details in next section). It was developed by implementing Google Apps Script to customise automatic functions, such as; update time logging, sending an alert to nurses, calculate number of drops into millilitres, etc. The acquisition sampling rate can be configured from Arduino source code itself, ranging from one minute to an hour.

B. User Interface Module

The user interface has been confirmed by interviews, to use a Spreadsheets application to store and record urine output data for expert analysis at a later stage, it was decided to reduce the learning curve for using software. The Spreadsheet is the most widely used End User Computing (EUC) tool, and it is an integral part of the qualitative analysis and strategic decision-making processes (Madahar 2011). Indeed, it is multi-purpose and widely used software with simple needs as well as large companies as an integrated system for informing decision supports (Cunha et al. 2015).

The Oxford Dictionary (2010) is described a definition of the word 'Spreadsheets' as:

"An electronic document in which data is arranged in the rows and columns of a grid and can be manipulated and used in calculations"

As far as anyone knows, Spreadsheets contain cells in rows and columns, and some may have specific sets of function defined for the manipulation and presentation of those data, this issue makes a distributed data to cells can be quite difficult and complex. Although Spreadsheets are capable of manipulating vast amounts of data, there is no convenient way for placing data in the Spreadsheets. Typically, an easiest way to do the transferred a set of data from database into a Spreadsheet is looking up a file that is stored and copy each record into Spreadsheets. Another conventional method is retrieving necessary data from a database to input them into a table format. For this retrieval, it requires user to understand high level knowledge of the database, as well as the format of the file stored with data. Thus, it might be a very difficult method for selecting a specific set of data from a data source. Moreover, binding a data source to a practically cell in Spreadsheet, customised macros and coding need to be done. Writing macro code requires user who has knowledge of Visual Basic Programming, Excel objects methods and properties, and Structure Query Language (SQL).

From our point of views, data management needs to support collaboration among multiple users and at its very core. Second, data management systems need to appeal to a broader audience of users who are less technically skilled. Third, data management and the data source need to be integrated seamlessly. It means data collection; presentation and visualization should be immediately compatible with any devices.

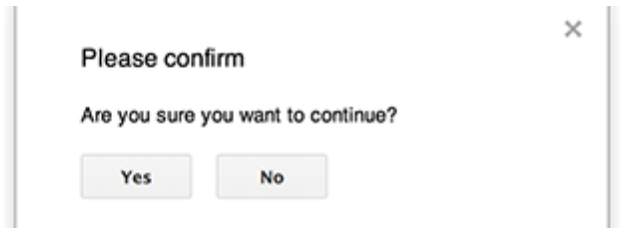
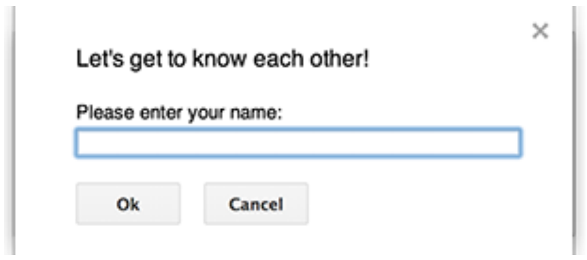
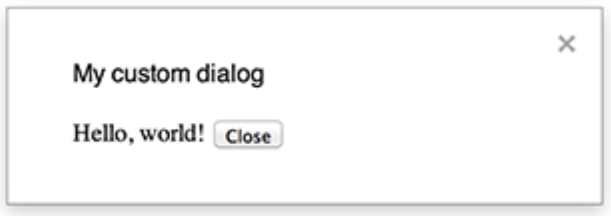
It has been decided to use Google Drive to store all the data using Googlesheets as front-end of user interface. Google Drive is one of the popular Cloud service providers, providing users a cost-effective, and in some cases free, ability to access, store, collaborate, and disseminate data (Quick and Choo 2014). By using Google Drive as a data storage has advantages including; a free and secured online Cloud based storage service with provided tools involved in data collection (Gan and Okonkwo 2014). It comes with version control, permissions-based sharing and instant messaging features that enhance the potential for collaboration, enabling multiple authors to work together in real-time (Rowe et al. 2013).

As a part of Google Drive features, Google launched '*Google Sheet*', web-based office suit that allows users to create and edit Spreadsheets online while collaborating with other users and platforms in real-time. Collaboration features are the main advantage of this application such task as simultaneous, real-time editing by a group of people, including chat and commenting with similar functions like Microsoft Excel. Information collected from any add-on data source from the Internet can be captured in a worksheet for analysing, tracking, and graphing with large amounts of data. Functions in Sheets can pull data and information from web-based sources, updating Spreadsheets, so they remain accurate automatically (Google 2015).

Google also provided APIs that allows users to create, retrieve, update, and delete, and collections data in real-time. These APIs scripting based on JavaScript, called 'Google Apps Script', and the basic features as listed in

Table 3.12.

Table 3.12 Features of Google Apps Script

Main Features	**Sub-feature**	**Description**
ADD	Custom menus	Extend Google products by adding user-interface elements in menu bar, with each menu item tied to a function in a script.
	Dialog & Side bars	Display several types of user-interface elements, such as, • Alert dialogs: a message and an "OK" button with a title and alternative buttons. Please confirm Are you sure you want to continue? • Prompt dialogs: a message, a text-input field, and an "OK" button; a title and alternative buttons are optional. Let's get to know each other! Please enter your name: • Custom dialogs and side bar: a custom message display an HTML or UI service. My custom dialog Hello, world!

WRITE	Custom functions	Google Sheets offers hundreds of built-in functions (AVERAGE, SUM, or VLOOKUP). However, Custom functions are created using standard JavaScript to support more complex system.
PUBLISH	Web apps	It contains a doGet(e) and doPost(e) function, and returns an HTML, UI, and content service objects.
INTERACT	Other Google Service	Google provides more than 30 built-in services for interacting with data, either internal or external systems including Google Analytics, Google Drive, Gmail, etc. Additional, Google Developer blog reported that this feature is based on JavaScript 1.6, and a few feature from JavaScript 1.7 and 1.8 which is commonly known objects, such as Array, Date, as well as, mathematical constants and functions, and Object global object.

With a number of advantages, Google Drive services can be fulfilled the requirement to store and display patient's information in secure online server with easy accessible to authorised staffs for real time monitoring, as well as predict future trends. Alternatively, the Google Apps Script can be utilized for additional automated functions. For example, time trigger for the data received from the monitoring devices, estimate the state of the patient's health, and automatically inform an alert to clinicians (in case of patients produced a tiny amount of urine or a container is full) by email or SMS text via other Google's services, such as, Gmail, or Google Text messages.

3.8 Designing tools

3.8.1 Computers

Computers are an additional element involved in this study. The main feature of the machine is used to develop source code, compiled and upload the source code from the development environment that is installed on the computers, also validate and verify the operation output. Two following machines were used in development process:

Table 3.13 Properties of computers

Specification/Computer	Laptop	Personal Computer
Operation System	OS X	Windows 7 Enterprise
Model	13-inchMacbook Air (2014)	HP EliteDesk 800 G1 SFF
Processor	Intel Core i5-1.6GHz	Intel Core i5-3.4GHz
Memory	4GB of 1600MHz	4GB
Storage	128GB	500GB

3.8.2 Fritzing (Software)

Knörig and Howell (2010) identified Fritzing as a free Electronic Design Automation tool (EDAs) for people who are not practicing electrical engineers. It is open-source and free to use. The advance of graphics formats allows users to replicate their physical Arduino breadboard circuit using on screen components from a drag and drop library. Additionally, Knörig *et al.* (2009) also reported that the aim of this software is to supports designers and artists of electronic artefacts through this prototyping barrier. Fritzing offers three alternative views on the circuit: *A breadboard liked view* is a classical schematic diagram view, and a *PCB design view* (see

Figure 3.26). Such a visualised software can only be described as a positive step in the drive forwards meeting designers needs for proving them with improved tools (Loudon *et al.* 2012).

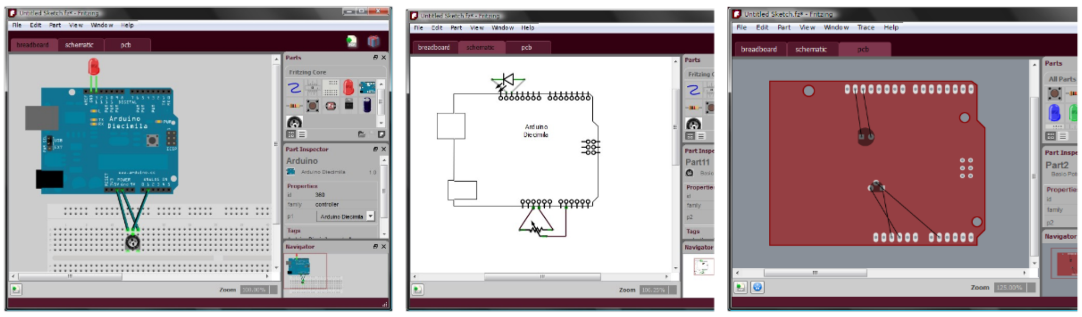


Figure 3.26 Fritzing Software User Interface (Knörig *et al.* 2009)

3.9 Summary

The proposed system architecture was discussed in the beginning of this Chapter. Our approach is not a replacement of the currently installed system in the Royal Bournemouth Hospital but it is a proof-of-concept of alternative solution to improve the performance for these systems in the nearly future.

The system utilizes off-the-shelf sensors as a data acquisition to detect a multitude of physical drop attributes. The sensors communicate directly with a microcontroller that provides ubiquitous communication, making it possible to access and sending information anywhere anytime. This system is presented under a concept of things that have capable to connect to Internet and present information from the sensor's data feed, as known as 'The Internet of Things'. This concept allows sensors to communicate information to people and system where in the most cases, this information is difficult to access or it was collected manually and frequently. Meanwhile, nurses are able to control and automated tasks remotely. Moreover, the IoT can help minimize the use of resources due to the volume and variety of data that will generated

A local area network provides the used for transmission the data from patients to a central databased in cloud computing platform. The Cloud computing is used to store, analyse and display a large amount of data that is generated as a result from connecting things. We believed that the proposed system has the capability of monitoring a large number of patients and provides further improvement to clinical practice and hospitals. Some relevant issues as well as an operational overview were also discussed. A development of prototype system will be discussed in the following chapter.

Chapter 4. Development of an Prototyping

The development of the physical prototype products is described in this study, communication and software prototype are also discusses. The approach adopted to realise a complete measurement, including communication and software system is detailed in sections 4.2 and 4.3, respectively.

4.1 Initial prototype system

The first version was focusing to evaluate the each custom sensor whether it is suitable (an overview of proposed techniques is given in Figure 4.1. and make a comparison of the exposed method for drop volumetric by detecting the same individual drop. The sensors were placed at sides of the closed environment, as shown in Figure 4.1. These sensors have two similar components; *Emitter* and *Detector* (or *Receiver*). An emitter is part where signals are sent to detect a drop, and a detector is sensed the change of the signal and amplifies it to microcontroller. When the drop falls down, it will change the path of the light for photo interrupter and infrared sensor, or sound wave for ultrasonic sensor. So the signal intensity received by detector is changed as well.

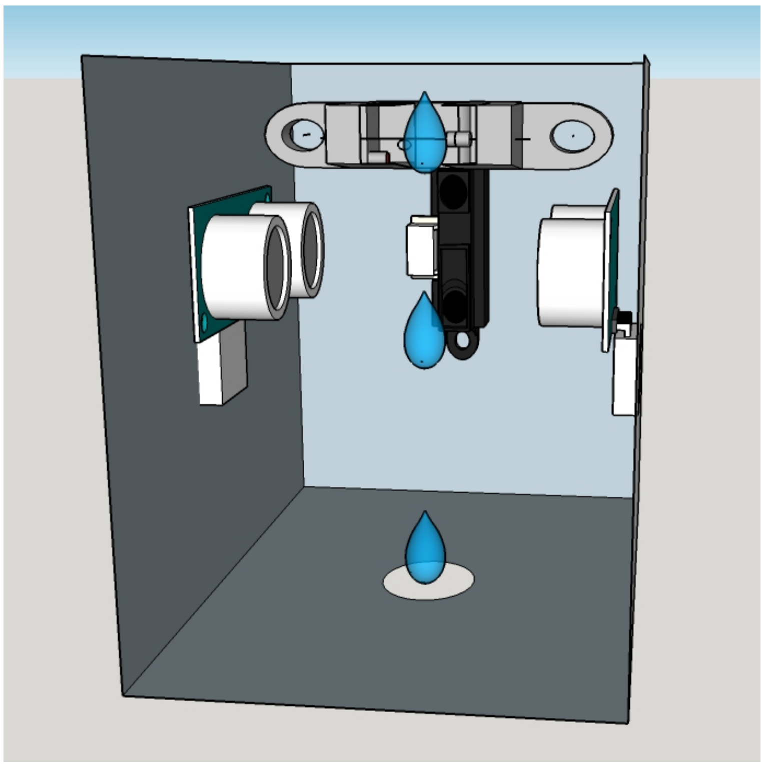


Figure 4.1 Model design for initial prototype

4.1.1 Initial hardware setup

In initial prototype, the fluid urine flows into the drip chamber through a 4-mm internal diameter and 30-cm length flexible plastic tube. One end of the output tube is connected to the closed container (11 inches length, 9 inches width, and 6 inches depth) where we placed a set of sensors inside the container, as present in Figure 4.3. The sealed container is added to facilitate the de-coupling; during the detection of the signals from the unwanted noises (e.g. sunlight or noises from other sources that may affect ultrasonic). Another end of the tube is connected to an intravenous bag on a pole to intravenous lines and an infusion needle. The purpose of the pole is to ensure that input tube always vertical and the tube does not hinder the flow of urine and it does not interface with the operation of the measurement. The height was set about 150 cm. All flow regulator devices, such as lure-connectors and flow control device were removed because we wanted to reduce the emptying duration and these devices may increase the resistance to flow in system, making operation less efficient.

The photo interrupter is used in the light intensity detection as a result of the passaged drop. It attached at the edge of a medical dropper to provide stable optical alignment. The medical dropper is adjusted vertical so that drops interrupt the beam as they form and then fall through a gap between emitter and phototransistor. The digital output of phototransistor used to calculate drop count. If the drop interrupted between emitter and receiver, the digital signal is pulled up to 1 until interruption is gone and it will be kept at.

```
void loop() {
val = ditialRead(12) // (defined signal pin as pin12)

  if (val = 1) {
         count = count++;
   }
```

The Sharp infrared sensor is placed at the far back inside the container. The theory of operation behind this arrangement is that the light will be emitted from the source and will reflect back into the detector after it detected the drop or other objects at the known reflection distance. The distance between light source and drop reflection is around 3.5 cm. In the case with no drop, the reflection will be occurred another end, around 6.3 cm from light source. Moreover, the known distance of the object refection can be determined by increase or decrease the value of the resistor. For the closer distance, it can be adjusted by adding additional small value of the resistor into a circuit.

To determine the optimal reflection distance, the 'DistanceGP2Y0A21YK.h' library has been modified and used for creating a data set of the two reflection distances. This library is an open-source, written by Jeroen Doggen[4]. The outcomes from this test are presented in Table 4.1 and Figure 4.2.

Table 4.1 A data set from IR sensor using analogRead()

Reflection point	**Maximum**	**Minimum**	**Average**	**Standard Deviation**
6.3cm (Container's wall)	6.4767	6.2874	6.4154	0.07
3.5cm (Dripping point)	3.6061	3.4416	3.6061	0.08

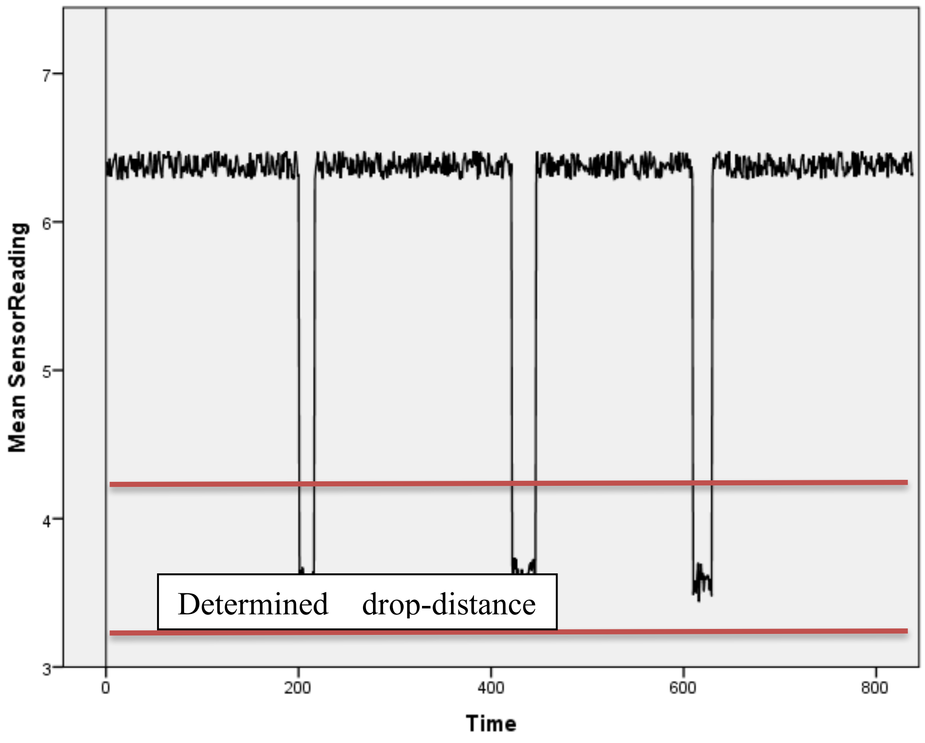


Figure 4.2 Sample of experiment results measure two different reflection distances.

[4] Appendix 7

Once the drop passed through the reflection beam, output from sensor reading is received range between 3.28 - 3.47 cm. In the case of no drops, the light will be reflected with the container wall where the output range is between 6.28 and 6.47 cm.

In order to count how many drops have been passed the light beam, the control sketch below were implemented to continually read analog signal and convert it into distance, then check whether the distance value is more than 3 cm and less than 4 cm.

```
#include <DistanceGP2Y0A21YK.h>
float distance;

void loop() {
  distance = Dist.getDistanceCentimeter();
  if ((distance >= 3) && (distance < 4)) {
        count = count++;
     if ((distance < 3) || (distance > 4)) {
         Serial.prinln “Out of Range”;
  }
}
```

Similar principle as the reflection IR-sensor, but ultrasonic sensor use sound instead of light for ranging and it is placed on the side inside the container. The ‘Ultrasonic.h’ is modified and used to reading digital signals from ultrasonic sensor. It is original written by Javier Rodrigo[5]. The distance between ultrasound emitter and the reflection position is around 3.5 cm, and 6.2 cm away from another side of the container. If the distance value is more than 3 and less than 4 cm, the total of drop-counting will be increased by 1.

```
#include <Ultrasonic.h>
Ultrasonic ultrasonic (5,6) // (define Trigger PIN as pin5, and Echo Pin as
pin 6)

void loop() {
  float distance = ultrasonic.Ranging(CM);
  if ((distance >= 3) && (distance < 4)) {
        count = count++;
     if ((distance < 3) || (distance > 4)) {
         Serial.prinln “Out of Range”;
  }
}
```

[5] Appendix 5

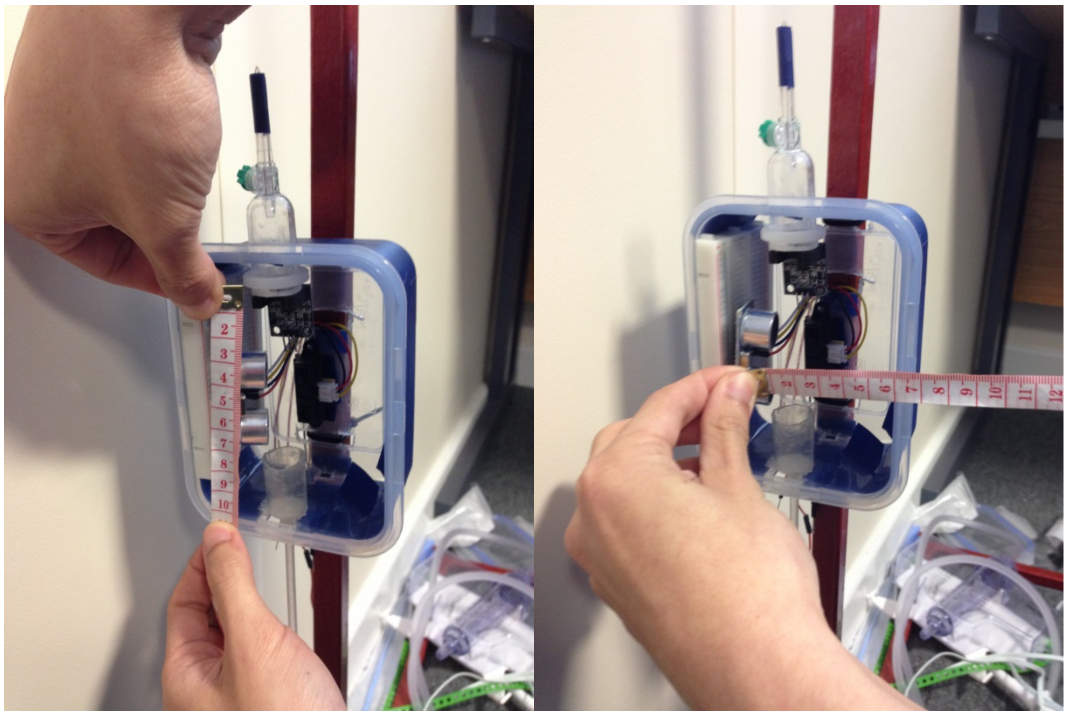


Figure 4.3 Initial Prototype setup

4.1.2 Evaluation of the initial prototype

The liquid amount from 5, 10, 15, 20, and 25 ml were being tested. In this scenario is 3 sensors detect the same drops that fall through the emitters and recovers. For a given time instant *t*, the collected data can be denoted as; A_t = ([$PI_{t,1}$, $IR_{t,1}$,$US_{t,1}$], [$PI_{t,2}$, $IR_{t,2}$,$US_{t,2}$],…, [$PI_{t,n}$, $IR_{t,n}$,$US_{t,n}$]), where *n* is the total number of drop collected by each sensors, and A is amount input liquid. For each of these amounts, fifteen experiments were carried out. The results from the measurements are presented in Figure 4.4.

The comparisons between average values from sensor reading, estimation using linear fitting, and the assumption that the medical dropper delivers 20 drops per ml (with acceptance error value should have per-drop volume of 0.05 ±0.005ml) as stated in the IEC 60601-2-24 (IEC International Electrotechnical Commission 1998) are illustrated in

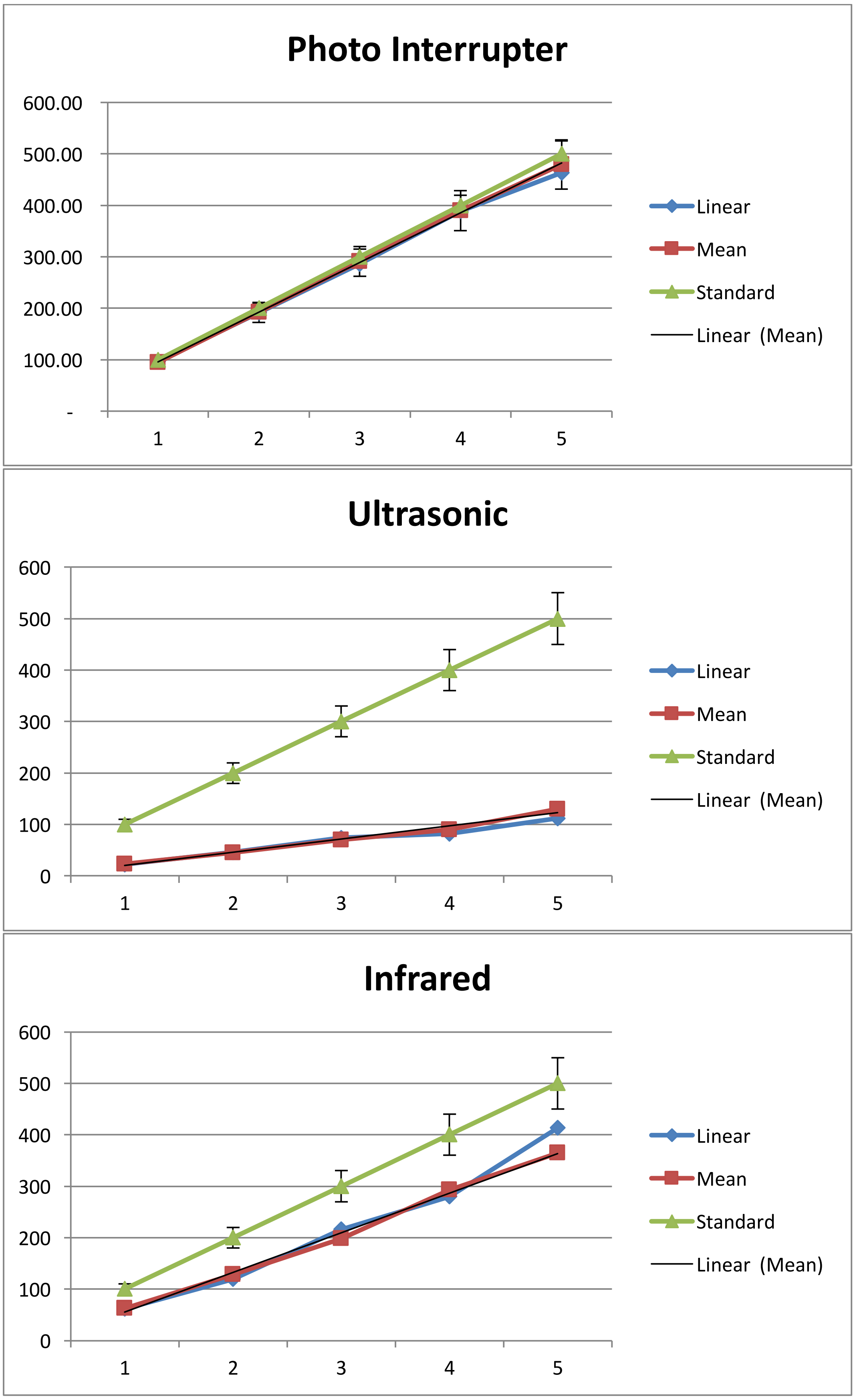
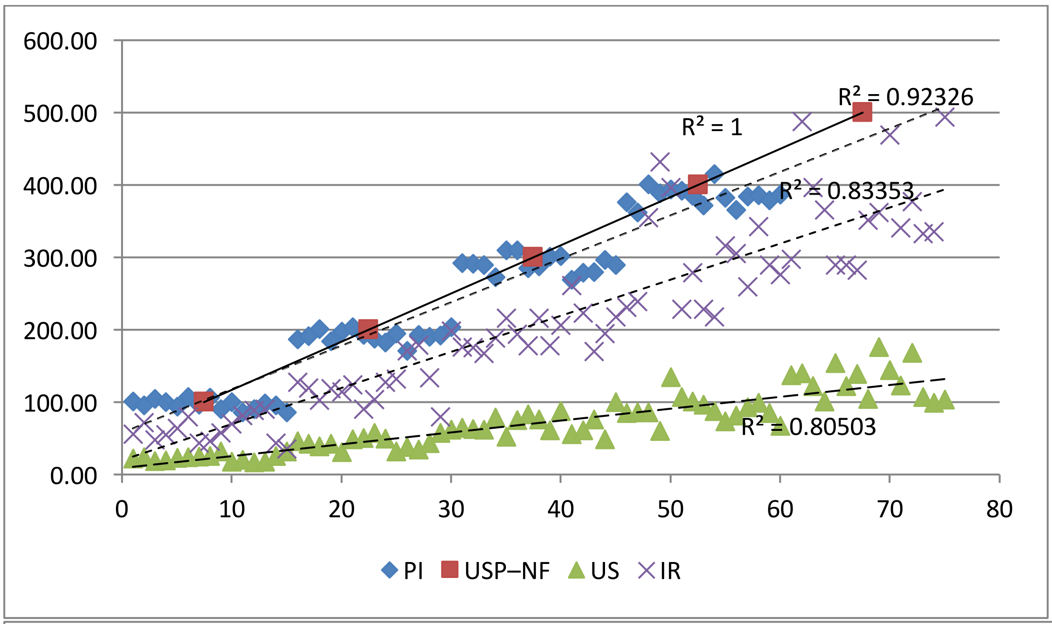
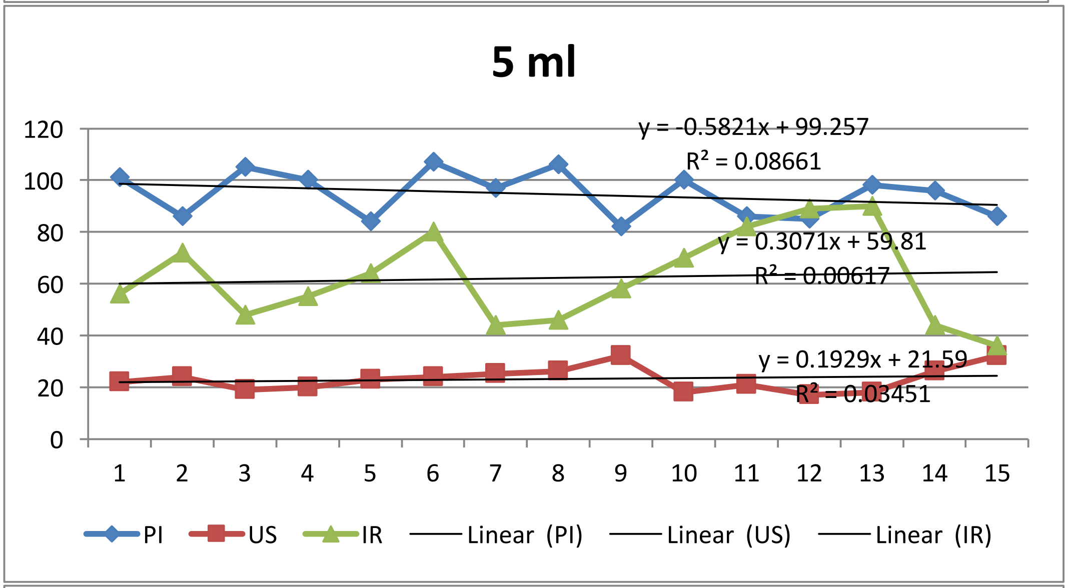


Photo Interrupter
600.00
500.00
400.00
300.00
200.00
100.00
-
1
2
3
4
5
Linear
Mean
Standard
Linear (Mean)
Ultrasonic
600
500
400
300
200
100
0
1
2
3
4
5
Linear
Mean
Standard
Linear (Mean)
Infrared
600
500
400
300
200
100
0
1
2
3
4
5
Linear
Mean
Standard
Linear (Mean)

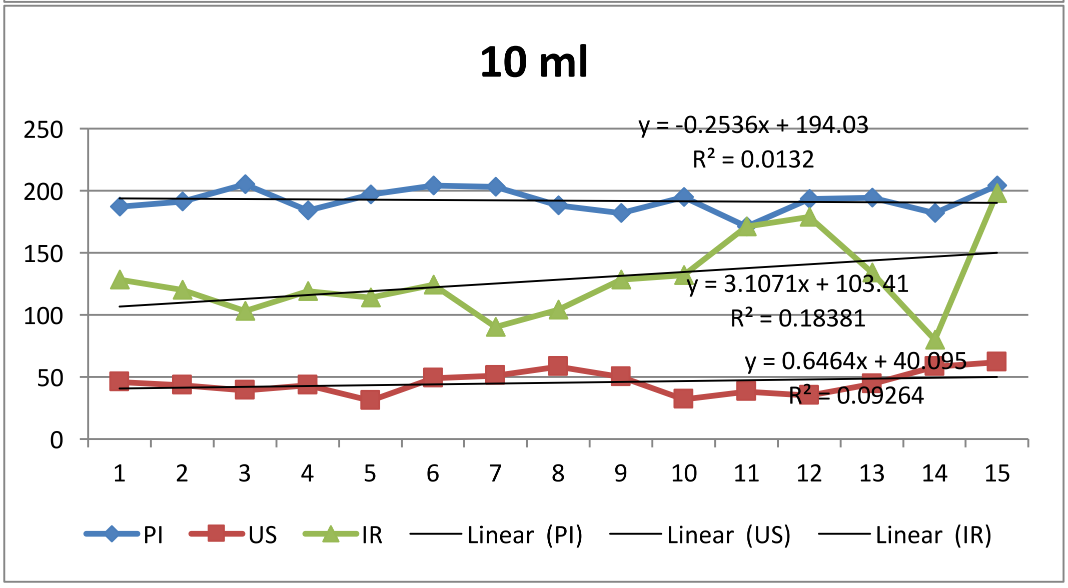
Figure 4.5.



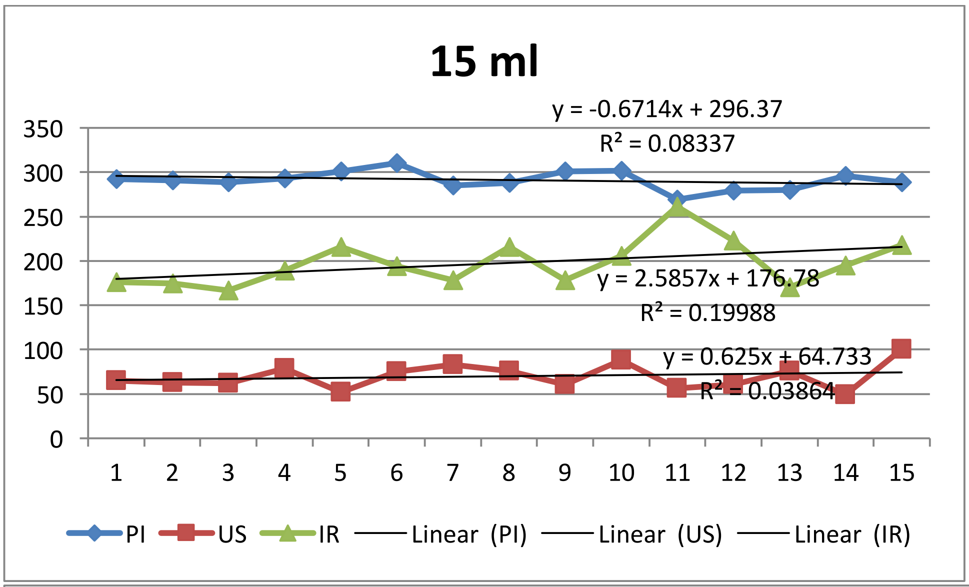
600.00
500.00
400.00
300.00
200.00
100.00
0.00
0
10
20
30
40
50
60
70
80
R² = 0.92326
R² = 1
R² = 0.83353
R² = 0.80503
PI
USP–NF
US
IR

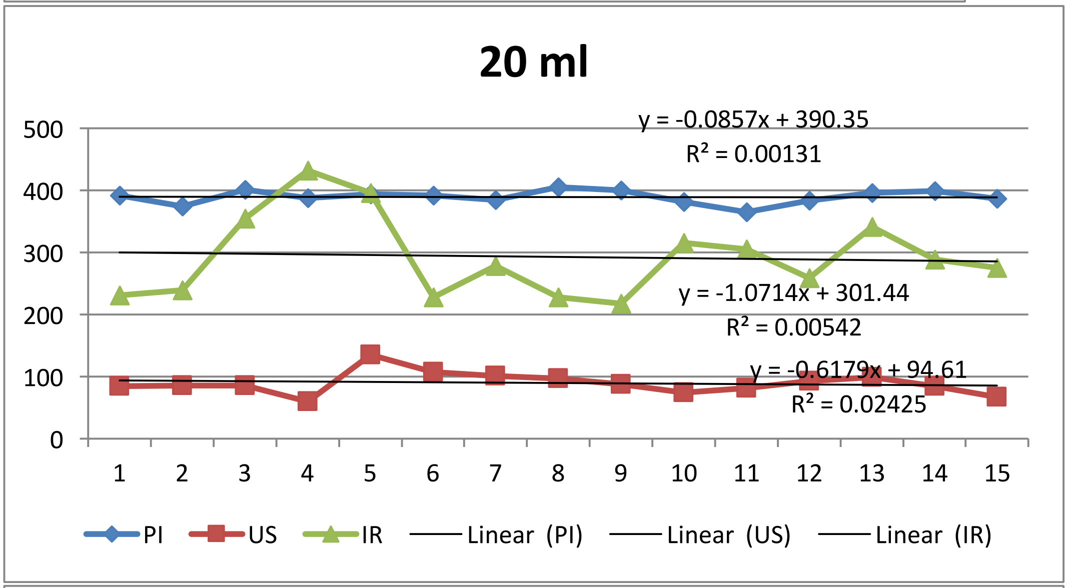


5 ml
120
100
80
60
40
20
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
y = -0.5821x + 99.257
R² = 0.08661
y = 0.3071x + 59.81
R² = 0.00617
y = 0.1929x + 21.59
R² = 0.03451
PI
US
IR
Linear (PI)
Linear (US)
Linear (IR)



10 ml
250
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
y = -0.2536x + 194.03
R² = 0.0132
y = 3.1071x + 103.41
R² = 0.18381
y = 0.6464x + 40.095
R² = 0.09264
PI
US
IR
Linear (PI)
Linear (US)
Linear (IR)





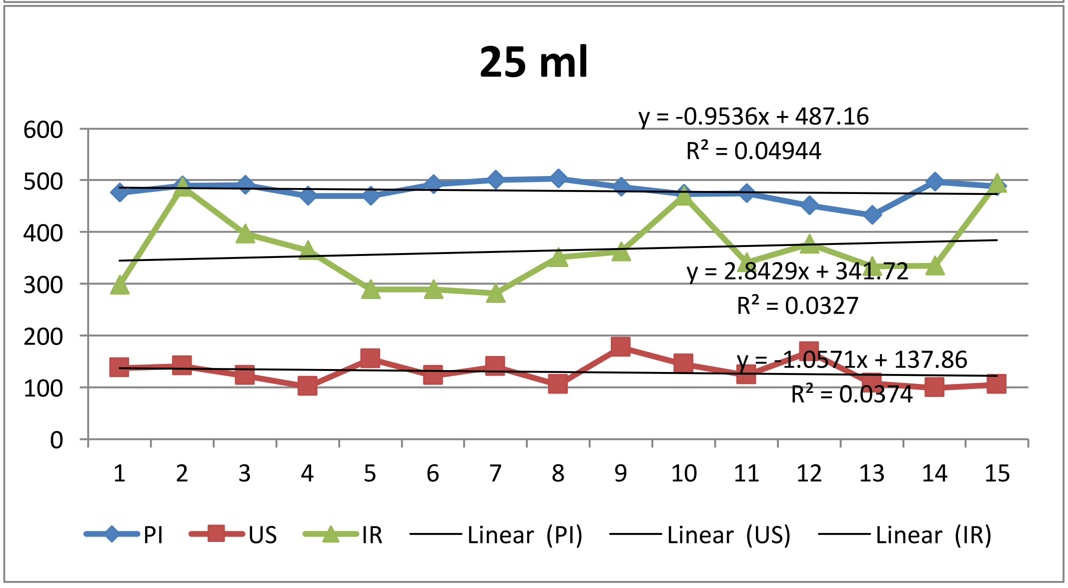


Figure 4.4 The values of 3 sensors reading from in different amount of input liquid

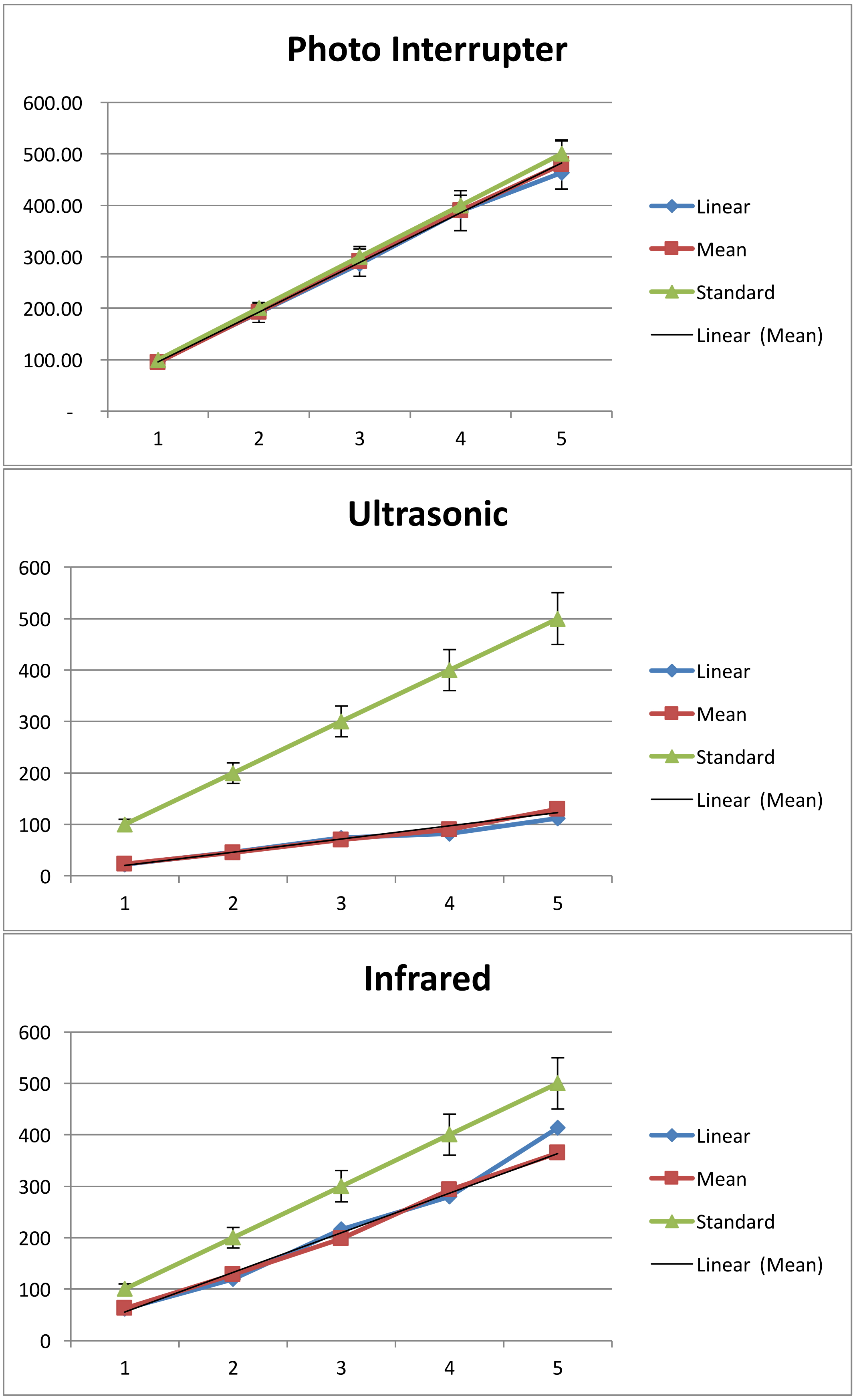


Figure 4.5 Total drop count of each sensor in different amount of liquid

Based on this information in Figure 4.5, clearly the results from photo interrupter are more stable, repeatable and controllable to detect drops at a constant rate. Both estimated value and mean values of drop detecting from this sensor are well within the required IEC-standard, as compared the reference point of 0.05 ±0.005ml per drop whereas ultrasonic sensor is the least effective choice. It should be pointed out that the experimental were taken place in a closed environment and would cause the sound wave bounce several time against the obstacles before reaching the receiver due to the multipath propagation effects of sound waves. These collisions may bounce off in a strange pattern causing lost echoes or shorten length of sound wave which directly affected the measurements.

It has been observed that the pulsed echo ultrasonic measurements suffer from some limitations, including that they need to be placed in the stable position for a correct angle, also the amplitude results in the receiver were depended on the obstacles that are between emitter and receiver. The use of the ultrasonic sensor may present future disadvantages in the clinical used, like the fact that the position of the sensor will be move when a patient moves, it will lead the ultrasonic detection may be in accurate, or even not work. Besides, having in mind that the material of the drip chamber is made of polyurethance, to detect the drop, this technique is required more advanced sensor to emit high frequency wave trough the thickness of drip chamber.

From these plots, the average weight of individual is presented in

Table 4.2. The results were come from the following equation:

$$Weight(drop) \ = \frac{Volume\ (ml)}{Total\ counted\ drop} \qquad 4.1$$

Table 4.2 The average drop volumes in 5 different liquid input from each sensor

Sensor/Amount	**PI**		**US**		**IR**	
Calculation	Linear	Mean	Linear	Mean	Linear	Mean
5 ml	0.0519	0.05285	0.22169	0.21614	0.08151	0.0803
Error (%)	3.82	5.7	343.38	332.28	63.02	60.6
10 ml	0.05222	0.05208	0.21478	0.22091	0.08407	0.07796
Error (%)	4.4	4.16	329.56	341.82	68.14	55.92
15 ml	0.05239	0.05155	0.20214	0.21511	0.06958	0.07596
Error (%)	4.78	3.1	304.28	330.22	39.16	51.92
20 ml	0.05146	0.05133	0.24316	0.22305	0.07143	0.06829
Error (%)	2.92	2.6	386.32	346.1	42.86	36.58
25 ml	0.05396	0.05213	0.22435	0.1932	0.06056	0.06859
Error (%)	7.92	4.26	348.7	286.4	21.12	37.18

4.1.3 Feedback

After demonstrated this version to our consulters, some concerns were raised about the operation procedure:

Firstly, adding more sensors means more power consumption. If the batteries need to be replaced often, it will not be much benefit, and might consider adding more tasks to nurse. By expanding functionalities, the number of wiring sensors will be increased but the mobility will be decreased.

Secondly, the flow control equipment had been removed. This makes the prototype lacks the ability to adjust the acquisition sample rate. When the volume of liquid is changed, it will be affected directly to the dripping rate and the drop size generated by the liquid level transfused. Given the assumption that patient produces normal amount of urine (or even worse, if they have diabetes), the error gap between the actual urine produced and the sensor reading may increase or may be exceed more than the requirements.

In additional, this prototype was relied on a careful positioning of the sensors in order to function properly. The problem with careful positioning is aggravated by the fact that after the nurse staff emptied the collecting bag, they would still have to connect the bag to the prototype, and possibly the position might be changed while doing so.

4.2 The improved version

A new improved version was designed to overcome the problems of the previous device. By modified some hardware supplies and added two load cells to determine volume using weight. However a fixed-scale balance that presented in Otero et al. (2010b) was considered not sufficiently to carry while patients are moving. To increase the portability of the prototype, we added two hanging weight scales instead. They were made from bar strain gauge based load cells connected with ‘S’ shaped hooks. These hooks designed to hang the prototype from either the bedside or a wheelchair, and to ensure that the input tube is always lie horizon for the proper flows. This issue has been mentioned in Van Santvliet and Ludwig (2004), they reported that 8.5% smaller drop were delivered due to the changing the dispensing angle to 45° angle compared to the vertical (90°), as well as different volume in variety of drop shapes. The theory of drop shape is varying on the tilt manner is confirmed by Mazzola et al. (2012), they found that two characteristic lengths of the drop shape that change with tilt grade.

4.2.1 Second Prototype setup

Only water was used to simulate the flow of urine. The input liquid container was placed around 35 cm above collecting bag. A commercial urine container from Unomedical is used to collect the fluid sample. It has capable to contain urine up to 2.5 litres. The urine meter is attached with a dropper (Intrafix® SafeSet), supplied by BRAUN and it equipped with a photo interrupter sensor. Another end of dropper is connected to another dropper where the dripping rate is controlled, the 3-D model was also built to generate for helping standardize on details and drafting, see Figure 4.6.

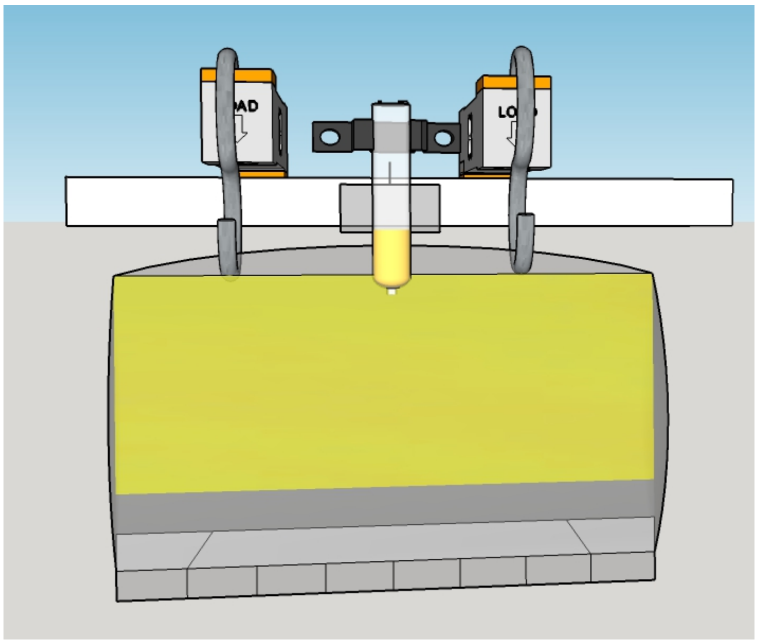


Figure 4.6 3-D model design for the second prototype

HX711 amplifiers are connected with each load cells to increase the output signal for the Arduino board. HX711 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

From the datasheet, the HX711 has two selectable input channels. Channel A can be programmed with a gain of 128 or 64, corresponding to a full-scale differential input voltage of ±20mV or ±40mV, respectively, when it powered with a 5V. Channel B has a fixed gain of 32. It comes with an On-chip power supply regulator eliminates the need for an external supply regulator to provide analog power for the ADC and the sensor. Clock input is flexible. It can be from an external clock source, a crystal, or the on-chip oscillator that does not require any external component.

This prototype was built to obtain multiple measures of the volume of liquid released by the container. The acquisition sampling rate can be configured manually which regulates the flow rate by nurse, then, the urine will flow though the 12.7 inches length of plastic tube to the second dripping chamber where it is detecting the drops before it go to the collected urine. The scales are configured to send measures every 10 minutes. The reason of this configure is to retrieved data more often but also reduced power consumption and the internet bandwidth when posting data to the storage. The density of urine has been estimated average value of approximately 1.02 g/ml from range between 1.005 and 1.035 g/ml. Thus by using this value, the additional error in the measurement will be added approximately 1.5%.

Despite from the physical prototype, a real-time processor was developed based upon a new design and evaluated. We can estimate the volume flow rate from average numbers of drop that passed through interrupter sensor multiplies by the weight of each drop in that period. Given is total of drop-count N_i taken at $Time_i$ where W_i is total weight obtained since prototype has started. Assumed average value of density of urine is 1.02. Therefore, the amount of urine produced during the interval at Time(i), measured in ml/g will be:

$$Volume = \frac{mass}{\text{density}} \tag{4.2}$$

$$\text{At Time}_i;\ Volume = \frac{\sum_{i=1}^{N_i}(W_i)}{1.02}$$

4.2.2 Evaluation of the new improved prototype

The equipment in the experiment was set as shown in Figure 4.7 and the volumes were tested with different flowrate as shown in Table 4.3. From the previous experiments, we knew that the most important concern is water level in the upper-container (h) that has to be constant throughout the experiments. The height was set around 15 cm for 500 ml liquid. Each flowrate had been tested for 15 times for 2 hours duration. The calibration was done for the IV Set corresponds to the standard. The weight sensors had been calibrated every time when the collecting bag is emptied. The weight data were collected at every 10 minutes and a photo interrupter sensor also equipped for solution drop counting. The drop volume of each setting was calculated as described in Equation 4.2 and will be compared with the reference value at 0.05±0.005.

Although the precise of these values depends on the patient's condition and weight, for usually 10 ml of urine per hour correspond to severe oliguria patient. 50 ml per hour corresponds to a normal—non kidney injured patient, and up to 250 ml/hour corresponds to polyuria patient.

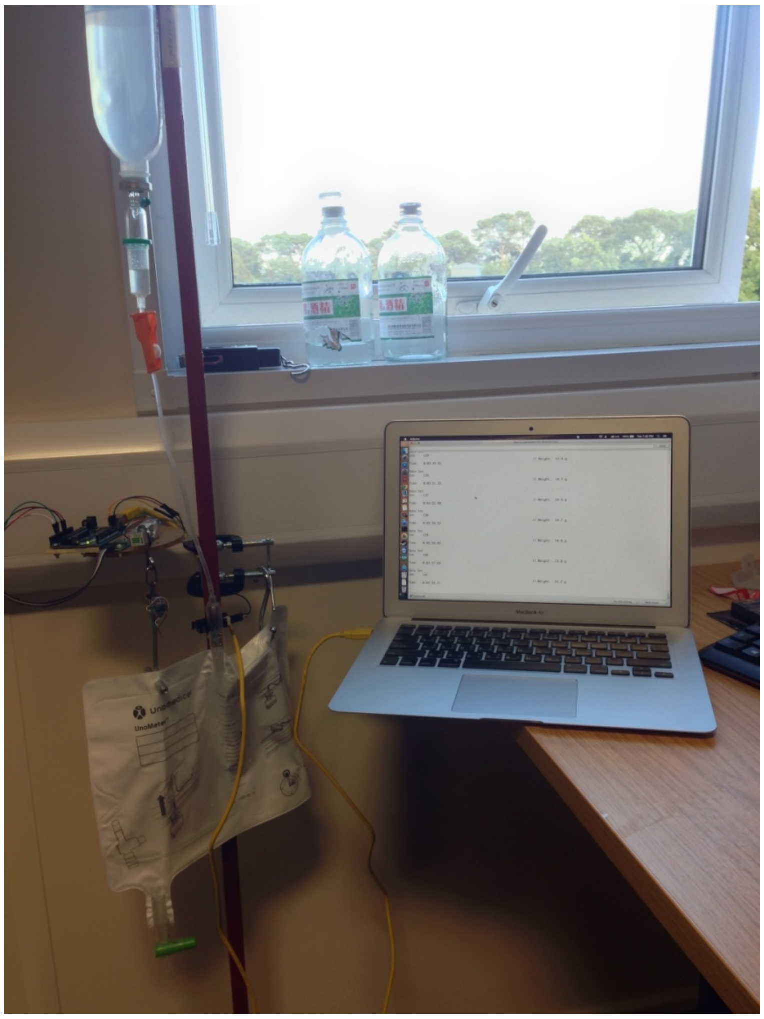


Figure 4.7 The experiment setting

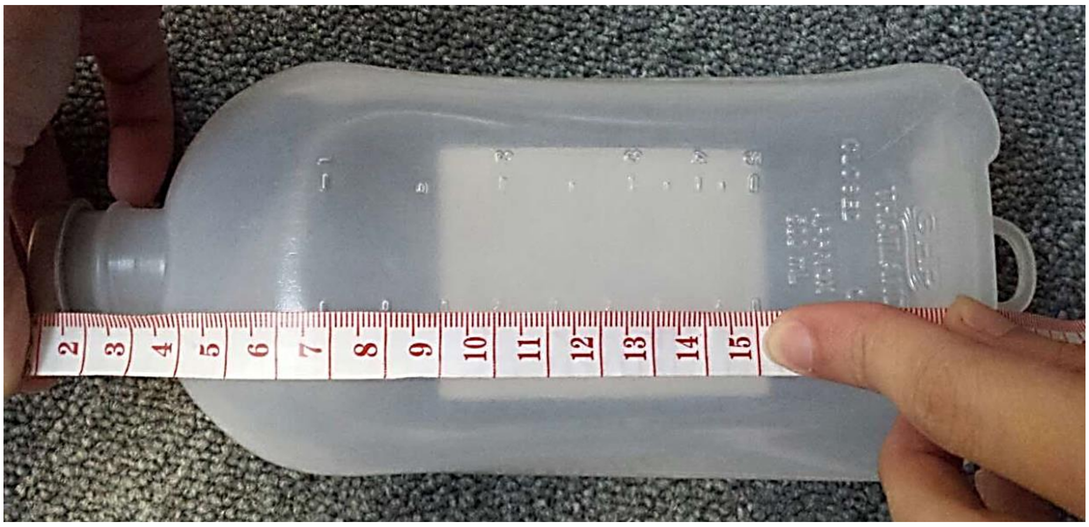


Figure 4.8 15 cm height of 500 ml in a container

Table 4.3 Comparison of drop volumes in different flowrate

Flowrate (ml/hour)	**Counted-drops (MEAN±STD)**	**Weight Sensors (grams) (MEAN±STD)**	**Avg. Drop-volume (ml)**
10	197.4±8.84	10.08	0.0510
50	974.6±9.80	51.08	0.0524
100	1942.2±10.39	102.29	0.0527
250	4874.5±14.60	258.43	0.0530

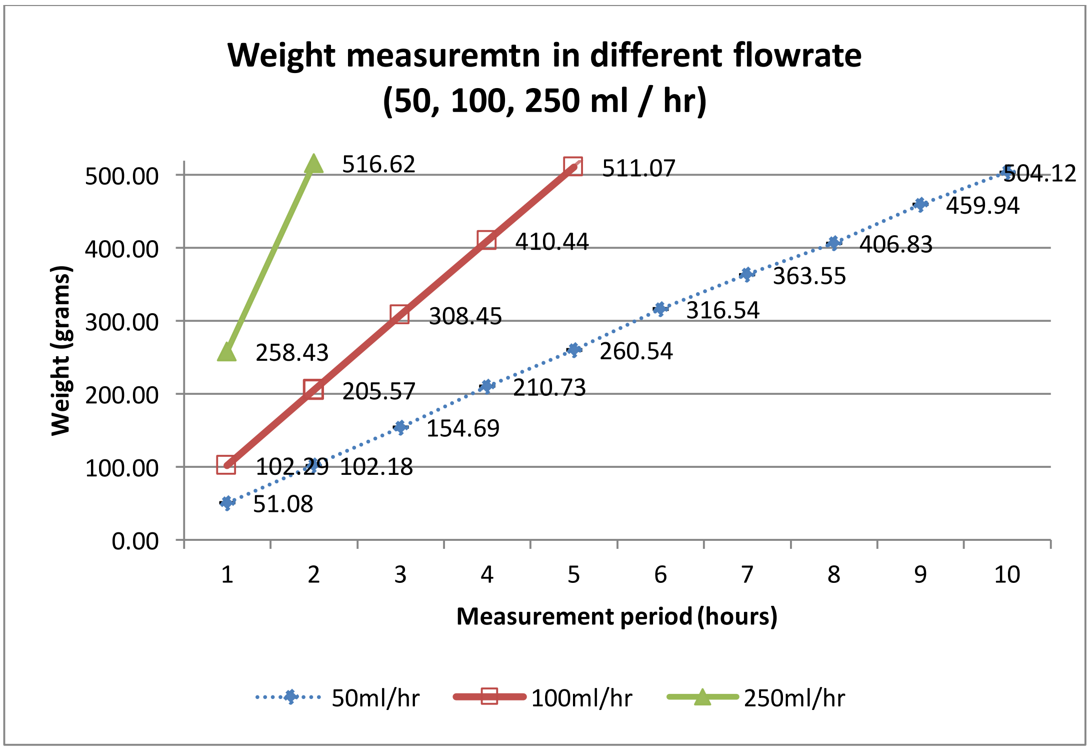


Figure 4.9 The drop volumes of 20drop/ml at flowrate 50, 100, 250 ml/h

The average drop volumes are calculated from four different flowrates of same amount of input liquid were acceptable within ±10% At the highest flowrate, the calculated volume per drop is around 0.053 ml whereas the slowest flow rate value is 0.051 ml per drop. As expected, when increasing the urine production rate, the weight of each drop is increased but the difference between these two values is considered very small and does not much affect the overall accuracy.

The average drop volumes are calculated from four different flowrates, equal 0.0522.Table 2 compares the error values between Equation 3.1, where number of drops multiplies by reference point (0.05 ml per drop) and Equation 4.2, where the weight sensors were added to measure the volume individual drop.

Table 4.4 The accuracy of the estimation methods between single sensor vs. combined sensors.

Flowrate	**Counted-drops**	**(Equation 3.1)**	**Error (%)**	**(Equation 4.2)**	**Error (%)**
10	197.4	9.87	1.3	10.30	3
50	974.6	48.73	2.54	50.87	1.74
100	1942.2	97.11	2.89	101.38	1.38
250	4874.5	243.725	2.51	254.45	1.78
MAPE			2.31	/////////////	1.975

4.2.3 Error sources

It should be highlighted that some errors were occurred during the experiments. Here, we characterised the source of errors found over that period. Firstly, approximately 1.5% will be added to the weight measurement due to the fact that using the average urine density of 1.020 g/ml.

The measurements were recorded over period of times and took place in public building where air conditioner is switched on and off at variety of times. However, in a hospital, room temperature is usually controlled by computer and remaining between 20-25 ^{o}C. These temperature variations can also affect the measurement and may produce an additional error.

A third potential source of error is caused by the experiment set up. A plastic input tube that connected between first dropper and the second one on top of the collecting bag, sometime it was not stable and it took several seconds to reach its position. The tube's movement can caused either increasing or decreasing of drop size and weight.

From this point, it can be said that the accuracy rate is improved by combined data from counted-drops and weight of collecting bag. This combination model can be used to monitoring either small or large amount of liquid input since their error values do not exceed more than 5%. Assume that a patient produces urine around 35 ml during the hour. In the worst case scenario, the maximum error of this prototype is ±(1.5%*35ml + 1.975%*35ml) ≈ ±1.215 ml.

4.2.4 Stress testing

The Arduino sketch was compiled in to the controller while the software interface is deployed on Cloud. As mentioned, the weight sensors were configured to send the measurement data every 10 minutes. When the urine measure is not obtained at the expected time, the application is needed to alert the nurse immediately. The test scenarios are listed in Table 4.5.

Table 4.5 List of test scenarios and test cases

Test scenario	Test case
When the daily goal is achieved	**[1]**Patient's weight ≤ 50kg, hourly goal must ≥ 20ml.
	[2]Weight ≤ 60kg && ≥ **[1]**, hourly goal must ≥ 25ml.
	[3]Weight ≤ 70kg && ≥ **[2]** , hourly goal must ≥ 30ml.
	[4]weight ≤ 80kg && ≥ **[3]**, hourly goal must ≥ 35ml.
	[5] Else urine goal ≥ 40ml. • True: filled cell with Green. • False: filled cell with Red, then send email/sms
	[6]Time trigger 4hourly && 4 records were post. • True: Sum(4 lastest rows record) • Flause: Send alert nurses to manual record, then check `AppendRowChoreo.available()`, o Return value = 0, then check Spreadsheets. o Else, inform network connection errors.
When the collecting container is nearly full	**[7]** If collecting container is weight > 1800 ml, • True: send email/sms. • False: continue monitoring.

The stress testing was performed continually 95 hours. After nearly 4 days, it was found that this new design prototype is able to work correctly, mainly in the software part. It still was able to perform drop counting and measure weight within the expected values.

All 100% of data records had been transmitted (557 times with 34.4 MB data size). These numbers were reported in activity logs from Cloud service. Details of information transmission and data storage will be presented in the following section.

4.3 Information Transmission features

The sequence of the authentication process before posting data into data storage is divided in to two stages: *InitializeOAuth* and *FinalizeOAuth* , as shown in Figure 4.10.

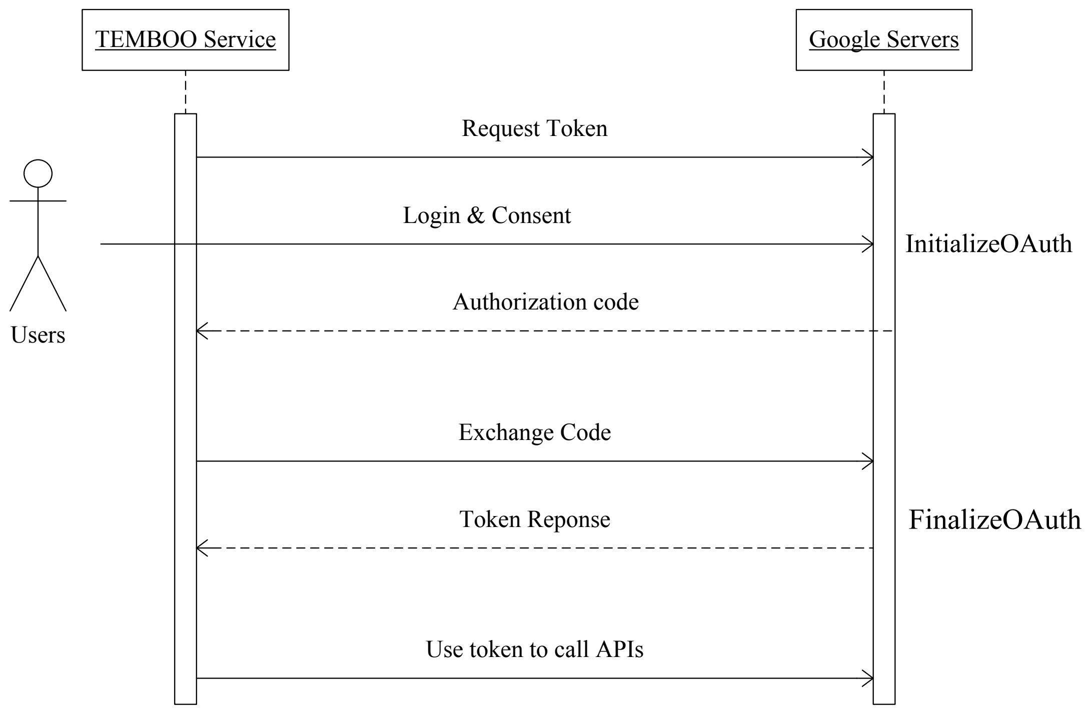
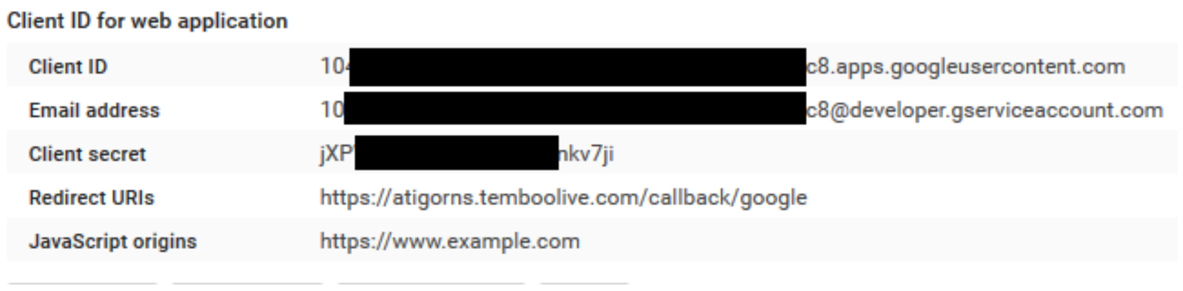


Figure 4.10 Using OAuth 2.0 in Google Web Server

4.3.1 Initialize OAuth

Initialize OAuth needs be done before the application can use Google's OAuth 2.0 authentication system. It requires end-user to login and register a project in the Google Developers Console to obtain OAuth 2.0 credentials. The Google Developers Console is a web terminal where user is capable to manage and viewing traffic data and authentication for the Google APIs in their projects used. The outcomes from registration show in figure Figure 4.11, these included ClientID, Google Email address, Client Secret, Redirect URLs, and JavaScript Origins where they were used as input parameters in Initialize OAuth



Client ID for web application	
Client ID	10[redacted]c8.apps.googleusercontent.com
Email address	10[redacted]c8@developer.gserviceaccount.com
Client secret	jXP[redacted]nkv7ji
Redirect URIs	https://atigorns.temboolive.com/callback/google
JavaScript origins	https://www.example.com

Figure 4.11 The output parameters from registered project in Developer Console

The initialize process can be done in TEMBOO web terminal where a Java repository is provided, see Figure 4.12. The functionality of this code is to generate an authorization URL that an application can use to complete the OAuth process. The required parameters are listed in Table 4.6., and the outcomes are presented in Table 4.7. These parameters will be temporarily stored and displayed in web terminal for easy retrieving and passing to the Finalize process.

Table 4.6 Input Parameters required for an authorisation token

Parameter	Description	Sample Values
ClientID	The Client ID provided by Google after registering the application.	…xxx.apps.googleusercontent.com
Scope	A space-delimited list of scopes to requests access. Additional, the scopes and tokens determine what user data the user gives your app permission to access**.**	https://docs.google.com/feeds https://docs.googleusercontent.com
CustomCallbackID (*Optional inputs)	A unique identifier that client can pass to eliminate delay to wait for Temboo to generate.	Call back identifiers is a random string that may only contain numbers, letters, periods, and hyphens.
ForwardingURL (*Optional inputs)	The URL that Temboo will redirect after they grant access to the application.	http:example.com/library/Google https:example.com/library/Google

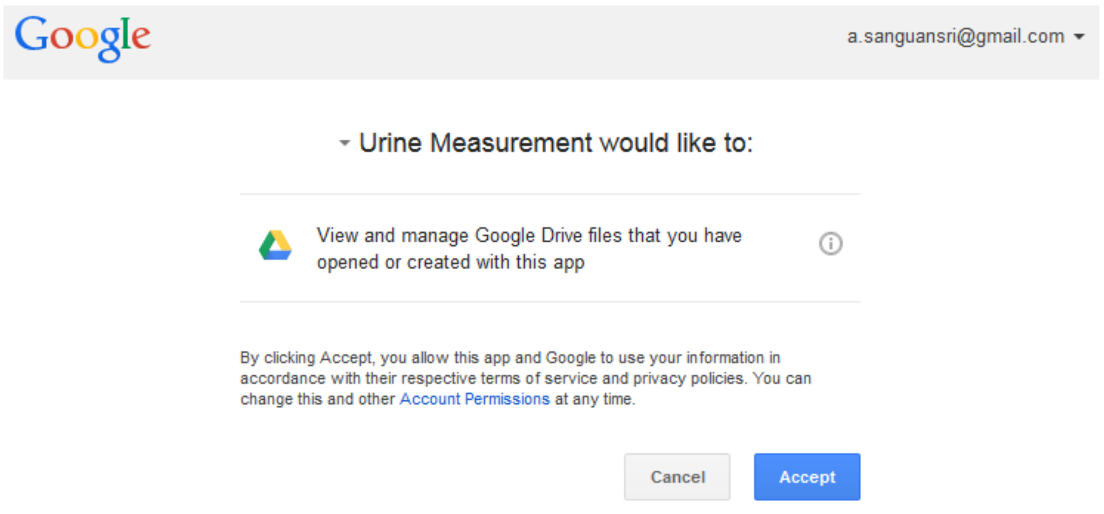
```
session = ("Account", "ApplicationName", "SessionID");

InitializeOAuth initializeOAuthChoreo = newInitializeOAuth(session);

InitializeOAuthInputSet initializeOAuthInputs = initializeOAuthChoreo.newInput
Set();

initializeOAuthInputs.setCredential("GoogleOAuthAccount");

initializeOAuthInputs.set_Scope("https://spreadsheets.google.com/feeds/");

InitializeOAuthResultSet initializeOAuthResults = initializeOAuthChoreo.execut
e(initializeOAuthInputs);
```

Figure 4.12 Java code for the Initiallize OAuth process

Table 4.7 Output parameters after received an authorization code

Parameter	**Description**	**Sample Values**
Authorization URL	The authorisation URL that the application's user needs to go to in order to grant access to your application.	https://accounts.google.com/o/oauth2/auth?client_id=clientID&responsetype=Type&redirect_uri=redirectURL&approval_prompt=force&access_type=offline&scope=https.xxx.doc.google.com
CallbackID	An ID used to retrieve the callback data that TEMBOO stores once your application's user authorises	Username/xxxxx-xxx-xxxxx

To grant access to Google account, application needs to be confirmed for private resources approval (see Figure 4.13) by navigating to an authorized URL. Google will handle the authentication session selection and user consent. This stage will be completed after the application is retrieving an access URL.


Google a.sanguansri@gmail.com

Urine Measurement would like to:

View and manage Google Drive files that you have opened or created with this app

By clicking Accept, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Figure 4.13 The consent window to view and mange Spreadsheets in Google Drive.

For additional security and reusable, two additional header files were created, *TembooAccount.h* and *GoogleAccount.h*. Both files are included into Arduino sketch for declaring authorized parameters in later stage. In Arduino, '#defined' constants does not take up any program memory space on the microcontroller but instead, the compiler will replace references to these constants with the defined value when it is compiled.

In *TembooAccount.h,* it contains TEMBOO account information included:

```
#define TEMBOO_ACCOUNT "ACCOUNT_NAME"
#define TEMBOO_APP_KEY_NAME "APP_NAME"
#define TEMBOO_APP_KEY "APP_KEY"
```

In *GoogleAccount.h,* it contains Google account information included:

```
#define GOOGLE_CLIENT_ID "Google-client-id"
#define GOOGLE_CLIENT_SECRET "Google-client-secret"
#define GOOGLE_REFRESH_TOKEN "Google-refresh-token"
```

4.3.2 Finalize OAuth

Finalize OAuth starts when an authorization code is received. Now, the application can exchange for an access token and a refresh token. The exchange code for access token is a one-time authorization between application and Google APIs. The exchange is made by sending an HTTPS POST request to the token endpoint. The input parameters are shown in Table 4.8.

Table 4.8 Input parameter required for exchange code for a token

Parameter	**Description**	**Sample Values**
ClientID	The Client ID provided by Google after registering the application.	.apps.googleusercontent.com
ClientSecret	The Client ID provided by Google after registering the application.	"Abcd1237_Xyz"
CallbackID	Callback token returned by the Initialize OAuth. Used to retrieve the authorization code after the authorize	Username/xxxxx-xxx-xxxxx
Timeout (*Additional input)	The amount of time (in seconds) to poll callback URL to see if app's user has allowed or denied the request for access. Defaults to 20 and max is 60.	60

Token endpoint is the endpoint of the authorization server where the client application exchanges the authorization code, client ID and client secret, for an access token. Figure 4.14 shows the sample of requesting code:

```
POST /oauth2/v3/token HTTP/1.1
Host: www.googleapis.com
Content-Type: application/x-www-form-urlencoded

code=4/{Speadsheets_ID} &
client_id={Clint_ID} &
client_secret={Client_Secret} &
redirect_uri={Redirect_URL} &
grant_type={Authorization_Code}
```

Figure 4.14 The POST method to send the token endpoint.

In

Table 4.9, the following parameters are received after the the access token exchange with Google APIs. Once the access token expired, the refresh token will be used as a new token.

Table 4.9 Sample of token returned from Google Server and its details.

Parameter	**Description**	**Sample Values**
AccessToken	The access token for the user that has granted access	…XXX…-AuthorizationURL
Expries	The remaining lifetime of the short-lived access token.	3600
RefreshToken	A token that maybe used to obtain a new acess when the short-lived access token is expried code after the authorize	1/xxxx/xxx-ClientSecret

An ID-Token is a JSON web Token that is a cryptographically signed Base64-encoded JSON objects (Google Developers 2015). It contains a set of name and value pairs (see Figure 4.15 and

Table 4.10 for details). Generally, it is critical to validate an ID token before use, but since we are communicating directly with Google over a HTTPS channel and using a client secret to authenticate ourselves to Google. We can be confident that the received token is came from Google and it is valid.

```
{"iss":"accounts.google.com",
 "at_hash":"HK6E_P6Dh8Y93mRNtsDB1Q",
 "email_verified":"true",
 "sub":"10769150350006150715113082367",
 "azp":"1234987819200.apps.googleusercontent.com",
 "email":"atigorn@example.com",
 "aud":"….xxx.apps.googleusercontent.com",
 "iat":1353601026,
 "exp":1353604926,
 "hd":"example.com" }
```

Figure 4.15 Sample of an ID token (formatted for readability).

Table 4.10 Google ID token descriptions (Google Developers 2015)

Parameter	**Description**
iss	Provided for the Issuer of the response. Always be: *accounts.google.com* or *https://accounts.google.com*.
at_hash	Access token hash. Provides validation that the access token is tied to the identity token.
email_verified	True if the user's e-mail address has been verified; otherwise false.
sub	An identifier for the user, unique among all Google accounts and never reused. A Google account can have multiple emails at different points in time, but the sub value is never changed.
azp	The client_id of the authorized presenter. This claim is only needed when the party requesting the ID token is not the same as the audience of the ID token.
Email	The email address.
aud	Identifies the audience that this ID token is intended for.
iat	The time the ID token was issued.
exp	The time the ID token expires.
hd	The hosted Google Apps domain of the user.

4.3.3 Operation Process

The operation process is controlled by Arduino sketch where has been installed into a microcontroller. After the microcontroller began to run the loop cycle, the *timer()* function is also started. This function is modified from the original 'SimpleTimer.h' library from Arduino Community written by Marcello Romani[6]. It runs based on millisecond, thus when the timer reaches 600,000 milliseconds (or 10 minutes), the *dataSend*() will be called while the value of current drop-count and time stamp will be converted into a string object, and set these two variables back to 0, as consequently. This function is used for posting the sting to Google Spreadsheets where these data will be stored and display in 24-hour records. The details of these functions are presented in Table 4.11

Table 4.11 Constructor summary of Arduino sketch

Functions

`timer.setInterval(600000, dataSend);`
`//Every 600000  ms, dataSend() will be called.`

`void timeStamp() {`
` time = millis()`
` sec = (time/1000)%60;`
` mins = ((time/(1000*60))%60);`
` hrs = ((time/(1000*60*60))%24);`
 	
` while (time>0) {`
` char currenttime[] = {hrs,“:”,mins,“:”,sec};`
` return currenttime[]`
`}`

`void dataSend() {`
 	
`     unsigned long sensorValue = getSensorValue();`
`     string now =getTimeStamp();`
 	
`     // invoke TEMBOO server with class.begin().`
`     TembooChoreo AppendRowChoreo;`
`     AppendRowChoreo.begin();`
 	
`     // set Temboo credentials`
`     AppendRowChoreo.setAccountName(TEMBOO_ACCOUNT);`
`     AppendRowChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);`
`     AppendRowChoreo.setAppKey(TEMBOO_APP_KEY);`

[6] Appendix 4

```
     AppendRowChoreo.setChoreo("/Library/Google/Spreadsheets/AppendR
ow");

     // set Google account credentials
     AppendRowChoreo.addInput("ClientID", GOOGLE_CLIENT_ID);
     AppendRowChoreo.addInput("ClientSecret", GOOGLE_CLIENT_SECRET);
     AppendRowChoreo.addInput("RefreshToken", GOOGLE_REFRESH_TOKEN);
     AppendRowChoreo.addInput("SpreadsheetTitle",
SPREADSHEET_TITLE);

     //add string : rowData[n]{"hh:mm:ss", unsigned_int}
     String rowData(now);
     rowData += ",";
     rowData += sensorValue;
     AppendRowChoreo.addInput("RowData", rowData);

     unsigned int returnCode = AppendRowChoreo.run();
     // return code of zero (0) means success
     if (returnCode == 0) {
        Serial.println("Success! Appended " + rowData);
        Serial.println("");
     } else {
        while (AppendRowChoreo.available()) {
          char c = AppendRowChoreo.read();
             Serial.print(c);
       }
     }

    AppendRowChoreo.close();
  }

  Serial.println("Waiting...");
  delay(5000); // wait 5 seconds between AppendRow calls
}

unsigned long getSensorValue() {
  return detectDrop();
}

Char getTimeStamp() {
 return timeStamp();
}
```

4.4 Data storage and User interface

4.4.1 Interface Design

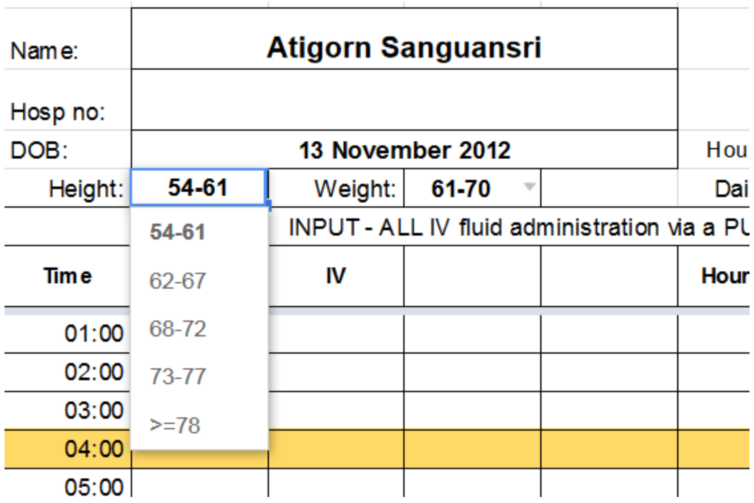
The interface was almost immediately modified to be simpler and similar to the record form currently used at the Royal Bournemouth Hospital. An interface was constructed to establish the best methods for defining the inputs to the system. These were initially sketched on paper. The design involved selecting the simplest way for the user to define the parameters required by the system. Some of these could be simple fields (e.g., patient's name or hospital number). Other parameters were better viewed, for example, drop-down lists which the user could be select one choice were applied to be used with height and weight (Figure 4.16), a calendar is tied to a standard input for patient's date of birth (Figure 4.17), and check box were applied to state the reason for using the record (Figure 4.18).

This interface had to be altered when the urine output data does not meet with criteria. The colour symbolism can be used for an indicator (Nemeth 2004). To achieve the mentioned requirement, we applied some customised conditional formatting into cells to indicate the current status of patient by using the colour code rules as followed:

- If the hourly urine produced data is less than minimum criteria, then the cell will filled background with 'Red'.
- If the hourly urine produced data is over the minimum criteria, then the cell background will turn into 'Green'.

The colour red and green were standardized as the colour for proceeding control. Red carried the strongest reaction of all the colours, and it is traditional colour of warning and danger. For this reason, we used the colour red to indicate the warning sign in our system. In contrast, the colour green widely used to indicate safety and permission to proceed which is opposite with red.

Figure 4.19 shown GUI displayed in Google's sheet format which allows to access on either on web browser in computer (A) or mobile application via mobile device by (B and C) the authored staff. In general the interface was improved with the adoption of clearer fronts and text size. The fields were also rearranged in to a neater format, and formatting condition are followed the rules given by the requirement. In one hand, most of the measurement inputs automatically inserted to a column in numerical vales from the prototype device. On other hand, related patient personal details are still required nurse to type down to the related cells. Thus version system has essentially the same format used in a hospital with a few alterations, relating mainly to the automated inputs.



Name: Atigorn Sanguansri

Hosp no:

DOB: 13 November 2012 | Hou

Height: 54-61 | Weight: 61-70 | Dai

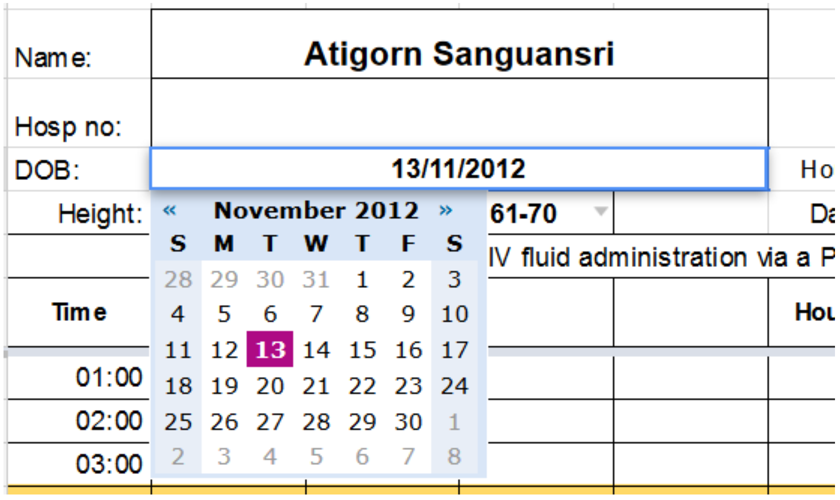
54-61
62-67
68-72
73-77
>=78

INPUT - ALL IV fluid administration via a PU

Time | IV | Hour

01:00
02:00
03:00
04:00
05:00

Figure 4.16 Drop-down list to choose range of height



Name: Atigorn Sanguansri

Hosp no:

DOB: 13/11/2012 | Ho

Height: | 61-70 | Da

IV fluid administration via a P

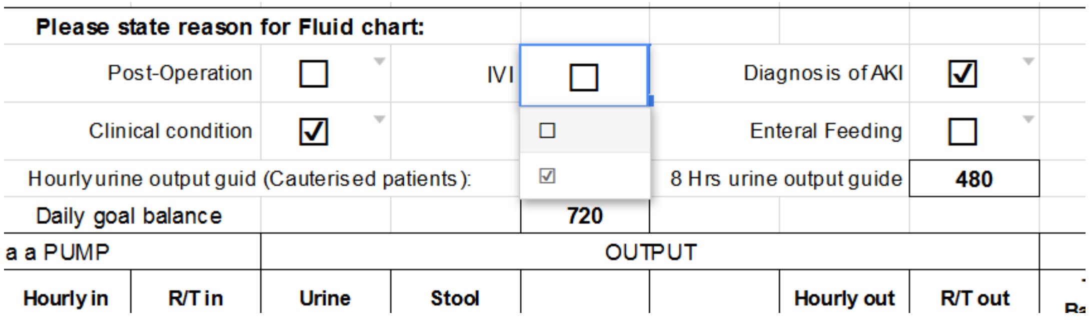
Time | Hou

01:00
02:00
03:00

« November 2012 »

S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Figure 4.17 Calendar applied for date of birth input



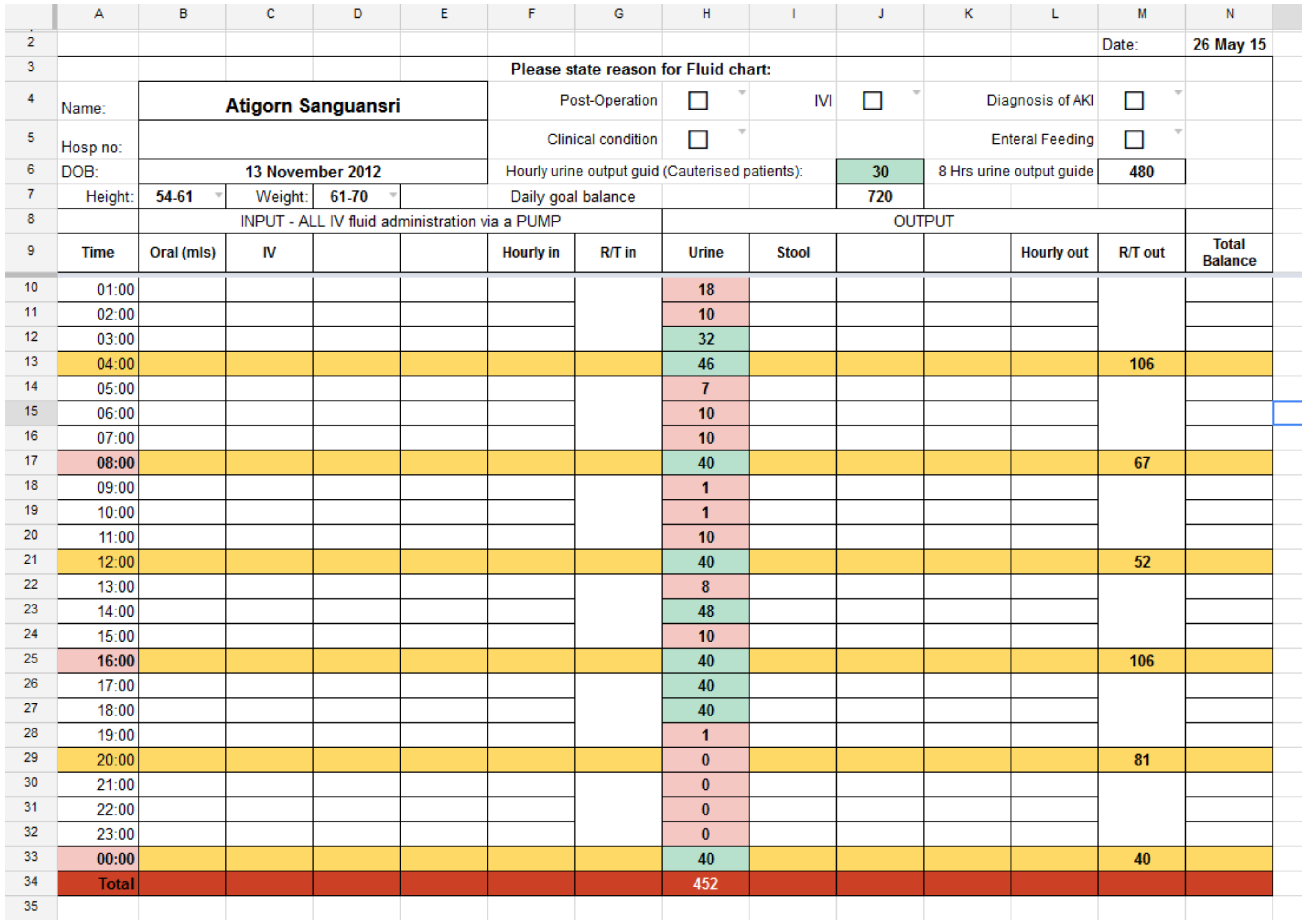
Please state reason for Fluid chart:

Post-Operation	☐	IVI	☐	Diagnosis of AKI	☑
Clinical condition	☑		☐	Enteral Feeding	☐
Hourly urine output guid (Cauterised patients):			☑	8 Hrs urine output guide	480
Daily goal balance			720		

a a PUMP | OUTPUT

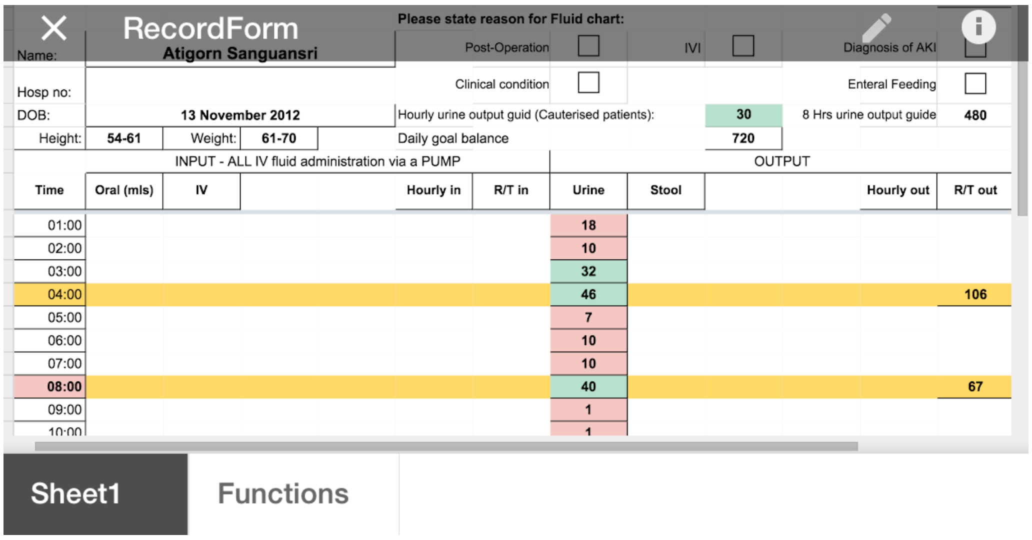
Hourly in | R/T in | Urine | Stool | | | Hourly out | R/T out

Figure 4.18 Check box selection to state the reason for fluid chart



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
2													Date:	26 May 15
3						Please state reason for Fluid chart:								
4	Name:	Atigorn Sanguansri				Post-Operation		☐	IVI	☐		Diagnosis of AKI	☐	
5	Hosp no:					Clinical condition		☐				Enteral Feeding	☐	
6	DOB:	13 November 2012				Hourly urine output guid (Cauterised patients):				30	8 Hrs urine output guide		480	
7	Height:	54-61	Weight:	61-70		Daily goal balance				720				
8		INPUT - ALL IV fluid administration via a PUMP						OUTPUT						
9	Time	Oral (mls)	IV			Hourly in	R/T in	Urine	Stool			Hourly out	R/T out	Total Balance
10	01:00							18						
11	02:00							10						
12	03:00							32						
13	04:00							46					106	
14	05:00							7						
15	06:00							10						
16	07:00							10						
17	08:00							40					67	
18	09:00							1						
19	10:00							1						
20	11:00							10						
21	12:00							40					52	
22	13:00							8						
23	14:00							48						
24	15:00							10						
25	16:00							40					106	
26	17:00							40						
27	18:00							40						
28	19:00							1						
29	20:00							0					81	
30	21:00							0						
31	22:00							0						
32	23:00							0						
33	00:00							40					40	
34	Total							452						
35														

Figure 4.19 Displayed GUI in Spreadsheet format



RecordForm

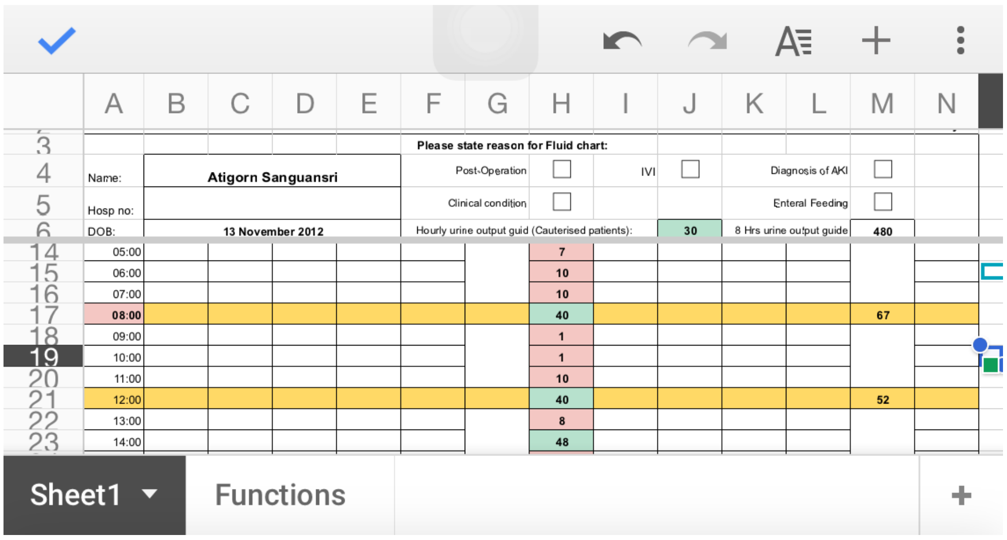
Please state reason for Fluid chart:

Name:	Atigorn Sanguansri	Post-Operation	☐	IVI	☐	Diagnosis of AKI ☐
Hosp no:		Clinical condition	☐			Enteral Feeding ☐
DOB:	13 November 2012	Hourly urine output guid (Cauterised patients):		30	8 Hrs urine output guide	480
Height: 54-61	Weight: 61-70	Daily goal balance		720		

	INPUT - ALL IV fluid administration via a PUMP					OUTPUT				
Time	Oral (mls)	IV		Hourly in	R/T in	Urine	Stool		Hourly out	R/T out
01:00						18				
02:00						10				
03:00						32				
04:00						46				106
05:00						7				
06:00						10				
07:00						10				
08:00						40				67
09:00						1				
10:00						1				

Sheet1 Functions

(B) From web browser on iOS



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
3						Please state reason for Fluid chart:								
4	Name:	Atigorn Sanguansri					Post-Operation	☐	IVI	☐		Diagnosis of AKI	☐	
5	Hosp no:						Clinical condition	☐				Enteral Feeding	☐	
6	DOB:	13 November 2012				Hourly urine output guid (Cauterised patients):				30	8 Hrs urine output guide		480	
14	05:00							7						
15	06:00							10						
16	07:00							10						
17	08:00							40					67	
18	09:00							1						
19	10:00							1						
20	11:00							10						
21	12:00							40					52	
22	13:00							8						
23	14:00							48						

Sheet1 Functions

(C) From mobile application in iOS

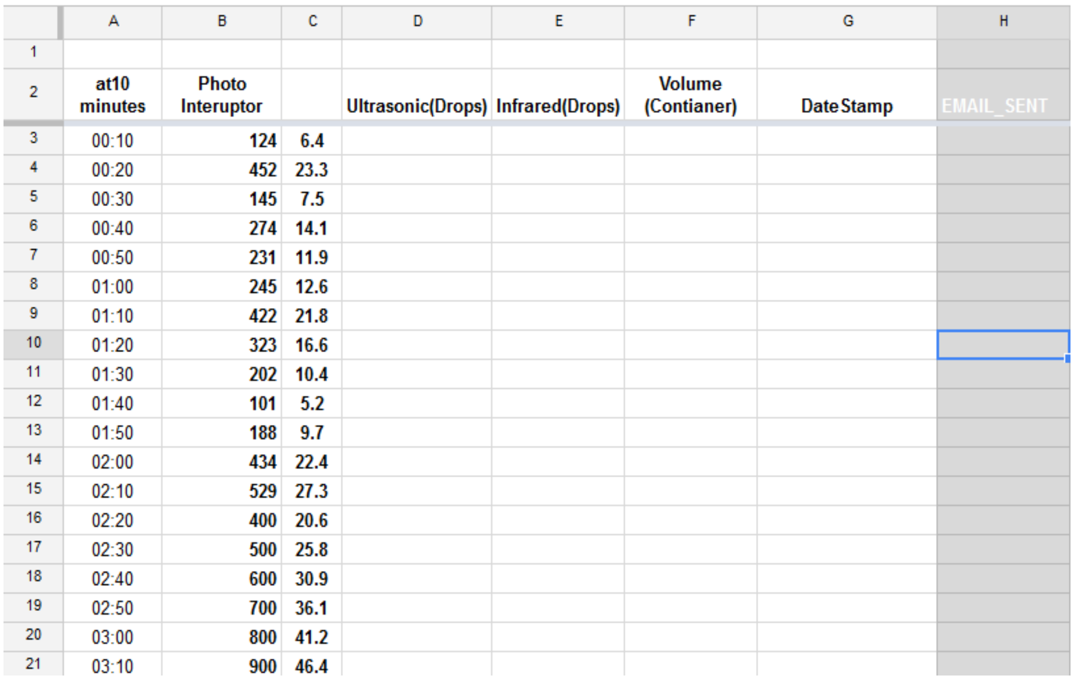
4.4.2 Data Storage

The data storage was also developed by using Google Drive to provide real-time collaboration as a service between authorised staffs to access files in Cloud. These collaboration APIs are a JavaScript library hosted by Google came with all of the functionality necessary for nurse to use a custom application that allows seamless and simultaneous editing.

Two simple Spreadsheets were created to store the necessary information (as shown in Figure 4.20 and Figure 4.21). In each sheet, additional functions were also implemented using Google Apps Scripts to programmatically read and edit the content inside. The scripts are modified using java.text package in Java platform Standard 7th Edition. It is a common platform for development and deployment of portable applications, desktop, or even web server environments. It is implemented by Oracle Corporation's Java Development, and can be applied into another language-independent application (Oracle Corporation (n.d.)-b, (n.d.)-a).

The java.text is a library that provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages. It is an idea of long term reuse of code. Once the study is going future, changing the programming language or switching the operating system should not force a source code to be completely rewritten; instead we can rely upon separated or dynamical localized resources, and make it more flexibility to add new localizations at any time.

A. Sheet!Datalog



	A	B	C	D	E	F	G	H
1								
2	at10 minutes	Photo Interuptor		Ultrasonic(Drops)	Infrared(Drops)	Volume (Contianer)	Date Stamp	EMAIL_SENT
3	00:10	124	6.4					
4	00:20	452	23.3					
5	00:30	145	7.5					
6	00:40	274	14.1					
7	00:50	231	11.9					
8	01:00	245	12.6					
9	01:10	422	21.8					
10	01:20	323	16.6					
11	01:30	202	10.4					
12	01:40	101	5.2					
13	01:50	188	9.7					
14	02:00	434	22.4					
15	02:10	529	27.3					
16	02:20	400	20.6					
17	02:30	500	25.8					
18	02:40	600	30.9					
19	02:50	700	36.1					
20	03:00	800	41.2					
21	03:10	900	46.4					

Figure 4.20 Sheet!Datalog

The first sheet is named ‘Sheet!Datalg’, contains eight columns (column A to H). Each column stores the different types and sources of data. The explanation is presented as follows:

Column A is representing a runtime of the hardware system which is sending data to Cloud storage every 10 minutes (by defaulted configuration) in the controller board. The data in this column came from the return value of function *timeStamp()* which has been set based on function *millis()*, a return value of milliseconds since it began running the current program, and this number will overflow (go back to zero).

Time variables were declared as unsigned long, making variables used to record in the elapsed millisecond for a fixed size. In C++ programming, unsigned long able to stores a 32-bit (4-byte value) range from 0 to 4,294,967,295 (2^32 - 1).

- defined ‘Seconds’ as unsigned long Sec = milli()/1000.
- defined ‘Minutes’ as unsigned long Mins = Sec/60.
- defined ‘Hours’ as unsigned long Hrs = Sec/3600.

Therefore it will overflow after 4294967296 milliseconds or 49.7 days and will produce strange results across this overflow. To prevent the overflow, we came up with the solution to reset time units by doing a subtraction of two unsigned numbers will produce the correct value even if one of them has over flowed.

- Sec = Sec - (Mins * 60).
- Mins = Mins - (Hrs * 60).

Here is an example; at 60 seconds, this function will automatically convert the mins (minutes) value equal to 1 and multiplies it by 60. When 60 subtract with 60, the final result will be equalled to zero. (Sec = 60 – (1*60)).

Column B to E is representing an increment number of drop captured by sensors; photo interrupter sensor, ultrasonic sensor, and infrared sensor, respectively.

Column F is representing the return values of the liquid weight measured by bar load cells. The liquid will be stored in a collecting bag after passed through the dropper. The maximum capacity volume of this container is 2000 ml. If the value is more than 1800 ml, the email will be sent to alert the nurse to empty the content of the container.

Column G is representing a date and time which an event is recorded for comparison of different records and tracking progress over time. Apps Script called '*autoDateTime()*', has been written to automatically generate data and time stamp when Column A has been inserted or edited. The details of the function are explained in the Table 4.12.

Table 4.12 Constructor summary of autoDateTime() function.

Class	**Parameters**	**Type**	**Description**
Range.getRowIndex()	N/A	Object	Returns the row position for defining the header row. It will only insert timestamp if header exists, but not in the header row itself.
Range.getValues()	N/A	Object	Returns the value of the top-left cell in the range.
Utilities.formatDate()	date	Date	a Date to format as a String- using: '*new Date()*' function to create date objects: ➔ Date (int year, int month, int date, int hrs, int min, int sec)
	timeZone	String	The output time zone set defaulted as GMT.
	format	String	a format per of date-time specification as "HH:mm:ss, [MM-dd-yyyy]"

Column H is representing a mark in each row every time an email is sent when a volume of produced urine does not meet the expected conditions. The script is called '*emailAlert()*', it will set the cells in column H to 'EMAIL_SENT' for each row after the function *emailAlert()* is called. The details of the function are explained in the

Table 4.13.

Table 4.13 Constructor summary of emailAlert() function.

Class	**Parameters**	**Type**	**Description**	**Parameter's Pattern**
Sheet.getRage()	row	Integer	The row of the cell to return	sheet.getRange(1, 1)
	column	Integer	The column of the cell to return	
Range.getValues()	N/A	Object	Returns the value of the top-left cell in the range.	
Mailapp.sendEmail()	recipient	String	The addresses of the recipients from return values in *getDate()* sub-function in 'Sheet!Schedule'.	"recipient1@example.com", "recipient2@example.com", "recipient3@example.com"
	subject	String	The subject line	"Sending emails from a Spreadsheets"
	body	String	The body of the email. Additional sub-function has been implemented to render HTML instead of the body traditional argument.	'<body>' + '<p> This is email sent from automated function for reminding you that the urine out does not meet the minimum requirement. </p>' '</body>' **or** '<body>'+ 'Now the urine collection is equal '+*volume*+' ml.</p>'+'</body>'

B. Sheet!Schedule

The details of nurse's work shift is located in Sheet!Schedule where administrator needs to insert a list of responders, email, telephone number, and assigned shift scheduling of seven working day. A scheduling table is divided into three shifts start from 08:01 to 16:00, 16:01 to 24:00, and 24:01 to 08:00, shown in

Figure 4.21.

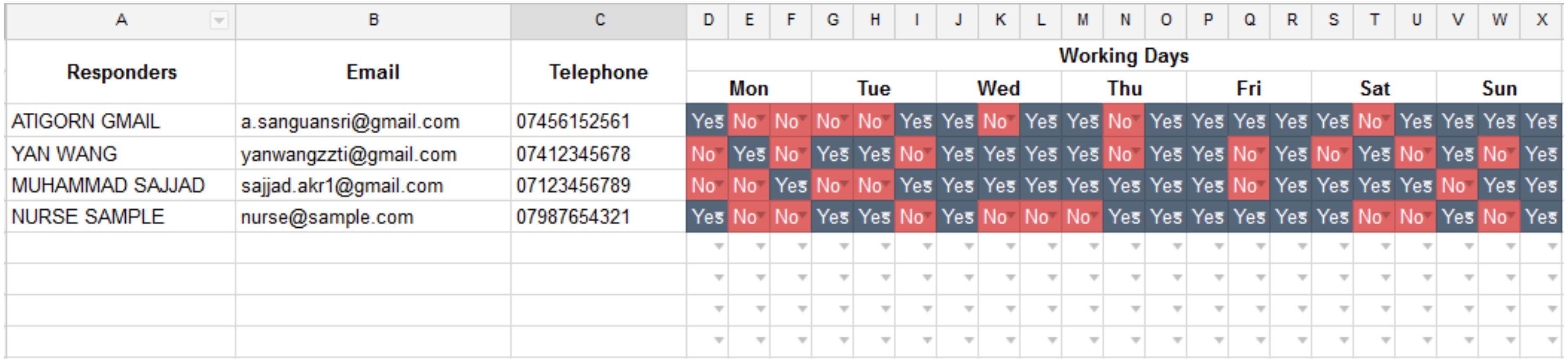
An time-trigger function called '*getDate()*' is implemented in this sheet to continually check whether each of responder is working in this shift or not by comparing with today's date and the current time. This function will continually update itself every 8 hours by time-driven triggered regarding to nurse working shifts. The function sequence is described as followed:

It defines the date and time in object, and splits it into an array format as: ("day"~time[hour]) using syntax '*getTodaysNumber().split('~')'*. The string output of day's parameter is range from 1 to 7 (as listed below), and time parameter is range from 1 to 24 (according to numbers of hour in a day).

1. is represented 'Monday'.
2. is represented 'Tuesday'.
3. is represented 'Wednesday'.
4. is represented 'Thursday'.
5. is represented 'Friday'.
6. is represented 'Saturday'.
7. is represented 'Sunday'.

The range value of Column A is defined as 0 by defaulted (e.g. Column B, C, and D are defined with number 1, 2, and 3, respectively). The working shifts were divided into three slots for each working day. In this case, to define a range in each *dayColumn* column, the calculation is done by multiplies by 3 to the results of the day's parameter, then, If-Elase statements have been applied to find a column value in a specified time slot in each working day:

- If current hour is >= 8 and <= 16, then, dayColum + 0.
- If current time is > 16 and <= 24:00, then, dayColum+1.
- Else, dayColum +2.

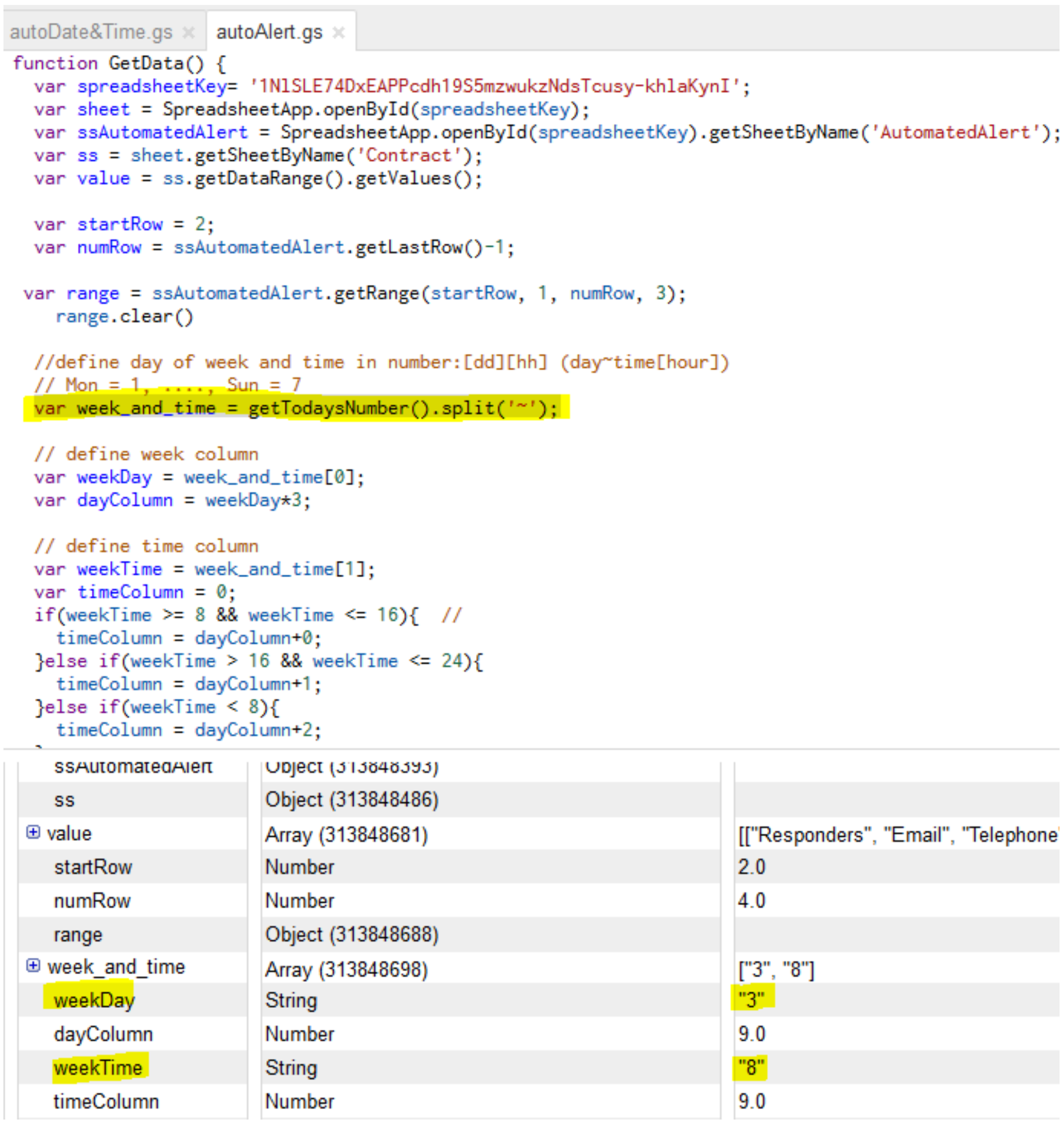


A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Responders	**Email**	**Telephone**	**Working Days**																				
			Mon			**Tue**			**Wed**			**Thu**			**Fri**			**Sat**			**Sun**		
ATIGORN GMAIL	a.sanguansri@gmail.com	07456152561	Yes	No	No	No	No	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
YAN WANG	yanwangzzti@gmail.com	07412345678	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	No	Yes	No	Yes
MUHAMMAD SAJJAD	sajjad.akr1@gmail.com	07123456789	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
NURSE SAMPLE	nurse@sample.com	07987654321	Yes	No	No	Yes	Yes	No	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes

Figure 4.21 Sheet!Schedule

The sub-function will store the values range from $(Row_{[2]}, Column_{[0]}: Row_{[last]}, Column_{[3]})$ where cells are contained 'Yes'. The stored values are in a string array formatted: ["Responders"], ["Email"], ["Telephone"]. For example, assume that today is November 04th, 2015 at 08:50.

Here, the result by running a debugging of *getTodateNumber()* function is [3][8], as highlighted in Figure 4.22. To find the working day, 3 is multiplied by 3, then add +0 for time condition, given *dayColum* = 9, which is indicated Column 9 or Column J (1st Shift of Wednesday).



autoDate&Time.gs × | autoAlert.gs ×

```
function GetData() {
  var spreadsheetKey= '1N1SLE74DxEAPPcdh19S5mzwukzNdsTcusy-khlaKynI';
  var sheet = SpreadsheetApp.openById(spreadsheetKey);
  var ssAutomatedAlert = SpreadsheetApp.openById(spreadsheetKey).getSheetByName('AutomatedAlert');
  var ss = sheet.getSheetByName('Contract');
  var value = ss.getDataRange().getValues();

  var startRow = 2;
  var numRow = ssAutomatedAlert.getLastRow()-1;

 var range = ssAutomatedAlert.getRange(startRow, 1, numRow, 3);
    range.clear()

  //define day of week and time in number:[dd][hh] (day~time[hour])
  // Mon = 1, ....., Sun = 7
  var week_and_time = getTodaysNumber().split('~');

  // define week column
  var weekDay = week_and_time[0];
  var dayColumn = weekDay*3;

  // define time column
  var weekTime = week_and_time[1];
  var timeColumn = 0;
  if(weekTime >= 8 && weekTime <= 16){  //
    timeColumn = dayColumn+0;
  }else if(weekTime > 16 && weekTime <= 24){
    timeColumn = dayColumn+1;
  }else if(weekTime < 8){
    timeColumn = dayColumn+2;
```

```
  ssAutomatedAlert      Object (313848393)
  ss                    Object (313848486)
⊞ value                 Array (313848681)      [["Responders", "Email", "Telephone"
  startRow              Number                 2.0
  numRow                Number                 4.0
  range                 Object (313848688)
⊞ week_and_time         Array (313848698)      ["3", "8"]
  weekDay               String                 "3"
  dayColumn             Number                 9.0
  weekTime              String                 "8"
  timeColumn            Number                 9.0
```

Figure 4.22 Debugging function to test the written scripts

The results of this function is shown four arrays have been found according to the given condition as shown Table 4.14, and they will be used as reference to send email(s) when *autoAlert()* in Sheet!Datalog is called. The details of the query in *getDate()* function are explained in the Table 4.15

Table 4.14 Queries in string arrays from getDate() function

getData()	rowcount	Number	4.0
	count[0]	Array	["ATIGORN GMAIL", "a.sanguansri@gmail.com", "07456152561"]
	0	String	"ATIGORN GMAIL"
	1	String	"a.sanguansri@gmail.com"
	2	String	"07456152561"
	count[1]	Array	["YAN WANG", " yanwangzzti@gmail.com ", "07412345678"]
	0	String	"YAN WANG"
	1	String	"yanwangzzti@gmail.com"
	2	String	"07412345678"
	count[2]	Array	["MUHAMMAD SAJJAD", "sajjad.akr1@gmail.com", "07123456789"]
	0	String	"YAN WANG"
	1	String	" yanwangzzti@gmail.com"
	2	String	" 7412345678"
	count[3]	Array	["NURSE SAMPLE", "nurse@sample.com", "07987654321"]
	0	String	" NURSE SAMPLE"
	1	String	"nurse@sample.com "
	2	String	"07987654321"

Table 4.15 Constructor summary of *getDate()* sub-function.

Class	**Parameters**	**Type**	**Description**
GetTodaysNumber()	N/A	Integer	Returns the string values of day in week and time (hours).
mailRange.getValues()	Responders	String	Returns the values from *getdate()* (see Table 4.14)

Chapter 5. Conclusions

Nearly, one-fifth of patients who admitted to ICUs suffered from the deterioration of kidney function, and nearly half need a body fluid balance measurement. An early detection is a key to prevent patients to become more critical. However, a fully functional of urine output is not yet introduced. To this end, the prototype has been designed, developed, and tested.

In this chapter, the objectives of the research are briefly reviewed and the research carried out to fulfil these goals is summarised as well as contribution to knowledge. Limitation of the research is a then discussed. Finally, areas of further work on the research are considered and the contribution of the research to the field of blood glucose sensors is discussed.

5.1 Review of the goals of the research

The objective of this research was to develop a prototype capable of continuously and reliably monitoring urine output. Its development could help reduce nurse's workload and prevent human errors in monitoring task.

Research was aimed to investigate existing urine monitoring devices, as discussed in Chapter 2 and producing a practical prototype that could be put in clinical routine as quickly as possible if the development proved successful. The concept of drop-volume measurement is proved to be used in small amount of urine, were it has been described in Chapter 2. To this end three versions of prototype have been presented. Details of the feasibility studies of sensor selection are their attributes were presented in Chapter 3 and Chapter 4.

In Chapter 4, the details of a proposed system implementation in both physical prototype as well as software for collecting real-time data are presented and its evaluation and analyse is discussed.

5.2 Summary of main finding

5.2.1 How prototype and system should be designed

For the physical prototype, a simple solution to building a law compliant urine meter capable of monitoring small amount of urine is using drop-volume estimation. This solution has been common practiced in hospital to delivery precise quantities of dosage to the patients. The evaluation of drop counting seems to be useful for this problem.

Furthermore, Though, there are no literatures directly mentioned the architecture of the IoT for hospital used, but mainly three elements found in almost publication are:

(1) Data acquisition (or 'data element') consists with sensors that connected to programmable microcontroller to record the sensor reading.

(2) Communication channels (or 'connecting element') which provides both wired and wireless connectivity.

(3) The application (or 'processing element') stores those information to data storage and enables nurses to monitoring the output in real-time.

The details of both physical prototype and software system design are discussed in Chapter 3.

5.2.2 The focus on sensor selection

There are different non-invasive sensors that have been used to detect drop obstructions and those sensors can be divided into three main categories:

(1) *Sound technique* detects of the tension variation produced by a drop that crosses a beam of ultrasounds. The sound wave response is independent upon the colour or light reflectivity of the object.

(2) *Optical technique* uses light beam to detect a deflection between the beam of light and the receiver. This method is lower in cost, has simple circuit, and low energy consumption.

(3) *Image analysing* acquires the images of the falling drops using high resolution of video camera. Various geometric parameters are used to estimate the total volume (e.g. drop shape and its diameter, elongation, etc.), makes this method to have very high accuracy.

A feasibility study had been carried out in initial prototype, presented in Section 4.1. Only sound (ultrasonic sensor) and optical sensors (photo interrupter and infrared sensor) were tested, for image analysis, it replied on a complex system and technology that is not fit in our aims.

The effectiveness of different sensors has been reported, the photo interrupter gave the most accuracy of all selected sensor, within standard as compared the reference point of (0.05 ±0.005ml per drop) whereas ultrasonic sensor is the least effective choice with average of 4.8 drops per ml due to the fact that ultrasonic sound bounces several time against the objects before reaching the receiver and these collisions created unwanted noises which directly affected the measurements.

A new designed prototype has presented in Section 4.2, where photo interrupter is equipped with medical dropper to count number of drop, weight sensors have been added. The main source of error in weight is produced by using an average urine density value of 1.020 g/mL to transform the scale measures (grams) to milliliters. Other sources of error are changes in room temperature, which affect the weight readings.

5.2.3 Comparison of a new improved prototype with previous research described in the published literature

In the published literature, the closest group of devices here are based on many types of sensor described in Chapter 2, namely (Otero et al. 2009; Otero et al. 2010a; Otero et al. 2010b; Otero et al. 2012; Otero et al. 2013). Float sensor equipped with mirrors have been proposed to measure the amount of liquid column height contained in a recipient, but the cost of the sensor makes it prohibitive for a device that must be disposable. Similarly in measurement functional, a device that has capability to empty itself autonomous was also presented. It equipped with reed switches to measure height of the urine in the container. However, it has a number of drawbacks that prevent it from being taken to the clinical routine and cannot provide an information minute-by-minute. A device based on capacitive sensors has been introduced to address the issue of providing information.

Compared with the device presented in this study, it is requiring only two types of sensors where they are available in the market and can be reduce the manufacturing cost. None of the above has been employed on Cloud computing where the monitoring data can be shared to another authorised clinical staffs in real-time. The software is programmed to transmit data every ten minutes which it provides similar information as done by previous system. Furthermore, the weakness of previous research is that it conceptualizes reducing nurse's workload at a macro level, ignoring the contextual and organizational characteristics of a particular health care setting (e.g., physical layout, information technology available) that may significantly affect workload.

However, the prototype presented here, does not have the functional to empty itself and still requires nurse to the bedside visit but, less often than they usually do. Another disadvantage is that when compared with the manual urine meter, it connected with electronic parts (e.g. sensors, hanging bar, microcontroller, etc.) where they can emit radio waves. If there are at the frequencies close to other monitoring devices, signals and reading cloud be corrupted. Moreover, this device needs to be disconnected from the patient's body before X-Ray or perform MRI.

5.3 Novel contribution to knowledge

The outcome of the research has made several contributions to both technological and medical practice perspectives as listed as follows:

1. The requirements in urine outcome measurement, espaically for the Royal Bournemouth Hosptical, have been deeply studied and analysed through literature review, interviews, and observation. The need for the sensor integrated for real-time data collection has been identified.
2. A wide range of related techniques were studied and analysed, included the device that is previously in the commercial market and those still under research. The limitations of those devices were identified and further demonstrated the need to improve a different type of integration system.
3. The proposed prototype and its system was developed to integrate off-the-shelf sensor products, open source software, and public Cloud data storage to monitoring the urine outcome. The prototype is included the creative hardware platform, controlled sensors and other components by microcontroller programmed with open source languages. Several experiments regarding this system were conducted and the feedback of medical professionals was analysed.

4. The results of the experiments showed that this prototype was compatible with the off-the-shelf components and available Cloud technologies. This combination is commonly known as an 'Internet-Of-Things' and it was helpful in shortening lab testing and improving user involvement.
5. This research has been succeeded in as it has identified the requirements for its proposed integrated sensor-based technology, analysed the weakness of the related studies, technologies, contribute alnernative approaches, and finally provided a new solution to address the problem.

5.4 Limitation of the research

1. This research has investigated the technical feasibility in developing an automatic system for monitoring urine outcome. However due to ethic concerns, the studies was restricted to a simulated environment only in a laboraoty. However, it is crucial to validate the designed prototype system on general hospital before it deploys.
2. Only off-the-shelf sensors have been used. The details on the electronic components and circuits are not carried out in the study. This leads to the operation of the sensor input becoming slightly 'black-box', means that the performance of the sensors could only be used in a very low tolerance. Although the proposed system is only the proof of concept. For futher development, it is required to improve the quality of the signals and reduce interfere with measurement environment.
3. Security is an important issue when dealing with patients health and privacy information. Although some seclected technology have been proposed for protecting the security of information transmission, there methods have not met fully tested in the designed prototype system.
4. As mentioned, this study was carried out only in laboratoty due to time and it must been approved by a ethics committee for the propose of conducting clinical routine studies. In this regard, the level of user appectance still reamin to be indetified. The impact of characteristic design (e.g. shapes, size, obstacles, materials, etc) and operation on this prototype shoulde be further studied.

5.5 Further work

Clearly, the new design prototype needs to be constructed and tested to overcome the current problems we are facing. With a great number of emerged technologies to date, this may help reducing the cost of equipment and provide more Cloud engine to process big data in real-time. The monitoring device must be continually developed and improved, possibly using an advance technology and different techniques.

To cover the full range of fluid monitoring, device should be integrated with other monitoring devices such as, IV infusion, liquid food, or smart cups, to cover the range of intake liquid. Combined with the prototype developed here, which cover the output liquid, this would enable fully monitoring of fluid balance in patient's body.

The power consumption when using external power source (e.g. power bank or battery) needs to be improved. Reducing power consumption means the device will be able to monitor longer period and may extend to be used as in-home device; this could possibly be achieved through hardware development, and also the emergence of IPv6 standard has resolved this issue in communication layer. However, this requires further investigation through practical testing.

The basic drop-volume calculation presented here is using drop counted by sensor under the assumption that drop volumes gained standard medical dropper are not different in sizes and shape. To provide reliable accuracy the calculation here should be further compared the other drop-volume methods such as, surface tension from Tate's Law, and Gravimetric Method. Moreover a new hysteresis in the sensors response should also be investigated in depth, both through mathematical models and experimental work to avoid this source of error in the measurement.

Finally, the security of healthcare services is very important for software platform. The quantitative measurement of security perspective in system architecture is needed to be guarantee because any type of system disaster can put live at danger in medical situations. For future work, we would like to investigate more in; (1) Data protection, (2) Resource-efficient security, (3) Physical security, (4) Data transparency, and (5) The security of handling big data from the IoT.

List of References

A Dictionary of Nursing, 2008. *urinometer* [online]. Encyclopedia.com: Available from: http://www.encyclopedia.com/doc/1O62-urinometer.html [Accessed 13 Febuary 2015].

Acosta, R. D., Burns, C. L., Rzepka, W. E. and Sidoran, J. L., 1994. Case study of applying rapid prototyping techniques in the requirements engineering environment (pp. 66-73): Publ by IEEE.

Ahsanul Haque, S., Mahfuzul Aziz, S. and Rahman, M., 2014. Review of Cyber-Physical System in Healthcare. *International Journal of Distributed Sensor Networks*, 1-20.

Ahuja, D. and Parande, D., 2012. Optical sensors and their applications. *Journal of Scientific Research and Reviews*, 1 (5), 060-068.

Amaravadi, R. K., Dimick, J. B., Pronovost, P. J. and Lipsett, P. A., 2000. ICU nurse-to-patient ratio is associated with complications and resource use after esophagectomy. *Intensive Care Medicine*, 26, 1857-1862.

Andrews, F. J. and Nolan, J. P., 2006. Critical care in the emergency department: monitoring the critically ill patient. *Emergency Medicine Journal*, 23 (7), 561-564.

Anwar, H., 2010. *Urinfo 2000 Precise Urine Meter from Flowsense* [online]. medgadget.com. Available from: http://www.medgadget.com/2010/10/flowsense_releasing_new_urinfo_2000.html [Accessed December 10th, 2014].

Arduino.cc, n.d. *Arduino Yun* [online]. Available from: http://arduino.cc/en/Main/ArduinoBoardYun [Accessed 8 November 2014].

Armstrong, L. E., Soto, J. A. H., Hacker Jr, F. T., Casa, D. J., Kavouras, S. A. and Maresh, C. M., 1998. Urinary Indices During Dehydration, Exercise, and Rehydration. *International Journal of Sport Nutrition*, 8 (4), 345.

Ashton, K., 2009. *That 'Internet of Things' Thing: In the real world, things matter more than ideas* [online]. Available from: http://www.rfidjournal.com/articles/view?4986 [Accessed

Aust, M. P., 2013. Introduction to Critical Care Nursing, 6th edition. *Critical Care Nurse*, 33 (1), 75-76.

Balducci, F., 2013. *Features of the new Arduino YÚN* [online]. Available from: https://balau82.wordpress.com/2013/09/02/features-of-the-new-arduino-yun/ [Accessed

Baldwin, C., 2014. *Case study: South Tyneside NHS Trust deploys Huddle on iPads for board meetings* [online]. Available from: http://www.computerweekly.com/news/2240214610/South-Tyneside-NHS-Trust-deploys-Huddle-on-iPads-for-board-meetings [Accessed September 30, 2015].

Ball, J. and Pike, G., 2009. Past imperfect, future tense : nurses' employment and morale in 2009: Royal College of Nursing (RCN), Royal College of Nursing (RCN).

Banzi, M. and Shiloh, M., 2014. *Make: Getting Started with Arduino: The Open Source Electronics Prototyping Platform*. Maker Media, Inc.

Bash, R., 2012. *FlowSense Medical Ltd.: Updated Company Presentation* [FlowSense Medical Ltd. Available from: http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCMQFjAA&url=http%3A%2F%2Ftrade.bankleumi.

co.il%2FTRADE%2Finfo%2Fshuk%2Fshukgetfile.asp%3Ffileid%3D107068%26&ei=AGeIVIGjA4SAU8CCgvgH&usg=AFQjCNFs4mi13HBpIzV8VtS8T7ay0zUIWA&bvm=bv.81456516,d.d24. [Accessed

Bassi, A. and Horn, G., 2008. Internet of Things in 2020: A Roadmap for the Future. *European Commission: Information Society and Media*.

Bell, C. A., 2013. *Beginning sensor networks with Arduino and Raspberry Pi* [online]. New York: Apress.

Bellomo, R., Ronco, C., Kellum, J. A., Mehta, R. L. and Palevsky, P., 2004. Acute renal failure - definition, outcome measures, animal models, fluid therapy and information technology needs: the Second International Consensus Conference of the Acute Dialysis Quality Initiative (ADQI) Group. *CRITICAL CARE*, 8 (4), R204-R212.

Boar, B. H., 1984. *Application prototyping : a requirements definition strategy for the 80s* [Non-fiction]. New York ; Chichester : Wiley, c1984.

Boyd, J. H., Forbes, J., Nakada, T. A., Walley, K. R. and Russell, J. A., 2011. Fluid resuscitation in septic shock: A positive fluid balance and elevated central venous pressure are associated with increased mortality. *Critical Care Medicine*, 39 (2), 259-265.

Brooker, C., Nicol, M. and Waugh, A., 2003. Problems associated with fluid, electrolyte and acid-base balance: Mosby, 2003.

Canellos, D., 2013. *How the "Internet of Things" Will Feed Cloud Computing's Next Evolution* [online]. Available from: https://blog.cloudsecurityalliance.org/2013/06/05/how-the-internet-of-things-will-feed-cloud-computings-next-evolution/ [Accessed March 24, 2015].

Cannella, C., Savina, C. and Donini, L. M., 2009. NUTRITION, LONGEVITY AND BEHAVIOR. *Archives of Gerontology and Geriatrics*, 49 (Supplement), 19-27.

Care Quality Commission, 2014. *Royal Bournemouth Hospital*

Quality Report 2014/15 [Care Quality Commission. Available from: http://www.cqc.org.uk/sites/default/files/new_reports/AAAA1845.pdf [Accessed

Cerda, J., 2011. Oliguria: an earlier and accurate biomarker of acute kidney injury? *Kidney International*, 80 (7), 699-701.

Chawla, L. S. and Kellum, J. A., 2012. Acute kidney injury in 2011: Biomarkers are transforming our understanding of AKI. *Nature Reviews. Nephrology*, 8 (2), 68-70.

Chen, X. and Wu, H. K., 2012. Ultrasonic Lamb Wave Measurement and Data Analysis System. *Advanced Materials Research*, 457 (1), 701.

Cho, I. S. and Chung, E. J., 2007. Assessment of a Prototype Diagnostic Nursing Decision Support System for Inpatients with Type II Diabetes Mellitus (pp. [2167]): IOS Press.

Cunha, J., Fernandes, J. P., Mendes, J. and Saraiva, J., 2015. Embedding, Evolution, and Validation of Model-Driven Spreadsheets. *IEEE Transactions on Software Engineering*, 41 (3), 241-263.

Cutnell, J. D. and Johnson, K. W., 2005. Essentials of physics. *Essentials of Physics, by John D. Cutnell, Kenneth W. Johnson, pp. 694. ISBN 0-471-71398-8. Wiley-VCH, March 2005.*, 1.

Damodaran, L., 1991. Towards a human factors strategy for information technology systems. *Human factors for informatics usability*, 291-324.

Del Río, O. I. and Neumann, A. W., 1997. Axisymmetric drop shape analysis: Computational methods for the measurement of interfacial properties from the shape and dimensions of pendant and sessile drops. *Journal of Colloid and Interface Science*, 196 (2), 136-147.

Department of Health, 2000. Comprehensive critical care: a review of adult critical care services. London: Department of Health.

Department of Health, 2013. *NHS challenged to go paperless by 2018* [online]. Available from: https://www.gov.uk/government/news/jeremy-hunt-challenges-nhs-to-go-paperless-by-2018--2 [Accessed April 16, 2015].

Di Gennaro, S. F., Matese, A., Mancin, M., Primicerio, J. and Palliotti, A., 2014. An Open-Source and Low-Cost Monitoring System for Precision Enology. *Sensors (14248220)*, 14 (12), 23388-23397.

Dictionary.com, n.d. *Sensor* [online]. The Free Online Dictionary of Computing. Available from: http://dictionary.reference.com/browse/sensor [Accessed March 02, 2015].

Ejlal, J., 2012. *Water consumption and factors influencing hydration status.* (Doctor of Philosophy). Loughborough University.

Elliott, M. and Coventry, A., 2012. Critical care: the eight vital signs of patient monitoring. *British journal of nursing (Mark Allen Publishing)*, 21 (10), 621-625.

Epstein, O., 2005. *Pocket guide to clinical examination* [Non-fiction]. Edinburgh) : Mosby, 2004, (2005 printing.

3rd ed.

European Food Safety Authority, 2010. Scientific opinion on Dietary Reference Values for water. *EFSA Journal*, 8 (3), Article 1459-Article 1459.

Faugel, H. and Bobkov, V., 2013. Open source hard- and software: Using Arduino boards to keep old hardware running. *Fusion Engineering and Design*, 88, 1276-1279.

Fleisch, E., 2010. WHAT IS THE INTERNET OF THINGS? AN ECONOMIC PERSPECTIVE. *Economics, Management & Financial Markets*, 5 (2), 125-157.

Fletcher, J. J., Bergman, K., Blostein, P. A. and Kramer, A. H., 2010. Fluid balance, complications, and brain tissue oxygen tension monitoring following severe traumatic brain injury. *Neurocritical Care*, 13 (1), 47-56.

Gan, A. and Okonkwo, O., 2014. The use of google drive in clinical audit. *British Journal of Hospital Medicine*, 75 (7), 397-401.

Gershenfeld, N., Krikorian, R. and Cohen, D., 2004. The Internet of things. *SCIENTIFIC AMERICAN*, 291 (4), 76-81.

GF HealthProducts Inc., 2007. *GF0700145RevA07-Urinometer: INSTRUCTIONS for use* [Available from: http://www.grahamfield.com/nosync/Documents/301_GF0700145RevA07_Urinometers.pdf [Accessed 13 February 2015].

Google, 2015. *Check out the new Google Sheets* [online]. Google Inc.,. Available from: https://support.google.com/docs/answer/3541068?hl=en&ref_topic=20322 [Accessed March 22, 2015].

Google Developers, 2015. *Using OAuth 2.0 to Access Google APIs* [online]. developers.google.com. Available from:

https://developers.google.com/identity/protocols/OAuth2 [Accessed May 15, 2015].

Gu, G. M., Shin, Y. K., Son, J. and Kim, J., 2012. Design and characterization of a photo-sensor based force measurement unit (FMU). *Sensors & Actuators: A. Physical*, 182, 49-56.

Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M., 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29 (7), 1645-1660.

Gugerty, B., Maranda, M. J., Beachley, M., Navarro, V. B., Newbold, S., Hawk, W., Karp, J., Koszalka, M., Morrison, S., Poe, S. S. and Wilhelm, D., 2007. The Maryland Nursing Workforce Commission Releases Documentation Work Group Report. *Maryland Nurse*, 8 (4), 14.

Guo, X., Zheng, L., Li, M. and Zhang, Y., 2014. Intelligent data acquisition and cloud services for apple orchard. *International Journal of Agricultural and Biological Engineering*, 7 (2), 146-153.

Guyton, A. C. and Hall, J. E., 2006. *Textbook of medical physiology* [Bibliographies

Non-fiction]. Philadelphia, Pa. : Elsevier Saunders, c2006.

11th ed.

Hall, J. E., 2010. *Guyton and Hall textbook of medical physiology*. Elsevier Health Sciences.

Hardt, D., 2012. The OAuth 2.0 authorization framework.

Haynes Jr., B. W., M.D., 1960. A New Device for Measuring Urinary Output Hourly (Vol. 174, pp. 890): J.A.M.A (The Journal of the American Medical Association).

Health, N. I. f. and Excellence, C., 2007. *Acutely ill patients in hospital- Recognition of and response to acute illness in adults in hospital* [Available from: http://www.nice.org.uk/guidance/cg50/evidence/cg50-acutely-ill-patients-in-hospital-full-guideline3 [Accessed

Hersch, M., Einav, S. and Izbicki, G., 2009. Accuracy and ease of use of a novel electronic urine output monitoring device compared with standard manual urinometer in the intensive care unit. *Journal of Critical Care*, 24, 629.e613-629.e617.

Hodges, S., Taylor, S., Villar, N., Scott, J., Bial, D. and Fischer, P. T., 2013. Prototyping Connected Devices for the Internet of Things. *COMPUTER*, 46 (2), 26-34.

Houde, S. and Hill, C., 1997. Chapter 16: What do Prototypes Prototype? *Handbook of Human-Computer Interaction*, 367-381.

Howe, D., n.d. *Microcontroller* [online]. Dictionary.com,. Available from: http://dictionary.reference.com/browse/microcontroller [Accessed April 07, 2015].

Hugli, H. and Gonzalez, J. J., 2000. Drop volume measurements by vision. *Proceedings of SPIE*, (1), 60.

Husch, M., Sullivan, C., Rooney, D., Barnard, C., Fotis, M., Clarke, J. and Noskin, G., 2005. Insights from the sharp end of intravenous medication errors: implications for infusion pump technology. *Quality & Safety In Health Care*, 14 (2), 80-86.

IEC International Electrotechnical Commission, 1998. *IEC 60601-2-24 medical electrical equipment-part 2-24: Particular requirements for the safety of infusion pumps and controllers*. Technical report, IEC.

Intensive Care Society, 1997. Standards for intensive care units. London: Intensive Care Society.

Irajpour, A., Salimi, M., Mardanian, L. and Rahimi, M., 2014. Developing and validating a patient monitoring flow sheet in intensive care units. *Iranian Journal of Nursing & Midwifery Research*, 19 (4), 354-359.

Istepanian, R. S., Hu, S., Philip, N. Y. and Sungoor, A., 2011. The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2011, 5264-5266.

ITU, (n.d.). *Internet of Things Global Standards Initiative* [online]. Available from: http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx [Accessed

Jansen, S. and McGregor, A., 2008. Static virtualization of C source code. *SOFTWARE-PRACTICE & EXPERIENCE*, 38 (4), 397-416.

Jho, C. and Carreras, M., 1984. The effect of viscosity on the drop weight technique for the measurement of dynamic surface tension. *Journal of Colloid And Interface Science*, 99, 543-548.

Jones, G. L., 2005. *The development of a novel strategy for the control of encrustation and blockage of Foley catheters* [http://search.ebscohost.com/login.aspx?direct=true&db=edsble&AN=ethos.016615443&site=eds-live&scope=site]. Bibliographies

Theses

Non-fictionCardiff University.

Jones, M. and Hardt, D., 2012. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750, October.

Kamogawa, M. Y. and Miranda, J. C., 2013. USE OF "ARDUINO" OPEN SOURCE HARDWARE FOR SOLENOID DEVICE ACTUATION IN FLOW ANALYSIS SYSTEMS. *QUIMICA NOVA*, 36 (8), 1232-1235.

Katch, V. L., Katch, F. I. and McArdle, W. D., 1996. Exercise physiology: study guide, *Exercise physiology: study guide.* Baltimore; USA: Williams & Wilkins.

Kavouras, S. A., 2002. Assessing hydration status. *CURRENT OPINION IN CLINICAL NUTRITION AND METABOLIC CARE*, 5 (5), 519-524.

Kelley, T. and Littman, J., 2001. *The art of innovation : lessons in creativity from IDEO, America's leading design firm / Thomas Kelley ; with Jonathan Littman* [Non-fiction]. London : HarperCollinsBusiness, 2001.

Kelly, W., 2014. *How a UK hospital mobilized its board with Huddle for iPad* [online]. Techrepublic.com. Available from: http://www.techrepublic.com/article/how-a-uk-non-profit-mobilized-board-with-huddle-for-ipad/ [Accessed September 30, 2015].

Kim, J.-C., Kim, K.-S. and Kim, S., 2013. Note: A compact three-axis optical force/torque sensor using photo-interrupters. *Review of Scientific Instruments*, 84, 126109.

Kinmond, R. M., 1995. Survey into the acceptance of prototyping in software development (pp. 147-152): IEEE.

Kinnersley, P., Edwards, A., Hood, K., Ryan, R., Prout, H., Cadbury, N., MacBeth, F., Butow, P. and Butler, C., 2008. Interventions before consultations to help patients address their information needs by encouraging question asking: systematic review. *BRITISH MEDICAL JOURNAL*, 337 (7665).

Kipnis, E., Ramsingh, D., Bhargava, M., Dincer, E., Cannesson, M., Broccard, A., Vallet, B., Bendjelid, K. and Thibault, R., 2012. Monitoring in the Intensive Care (Vol. 2012, pp. 20): Critical Care Research and Practice.

Knörig, A., Wettach, R. and Cohen, J., 2009. Fritzing: a tool for advancing electronic prototyping for designers, *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (pp. 351-358): ACM.

Kovner, C. and Gergen, P. J., 1998. Nurse staffing levels and adverse events following surgery in U.S. hospitals. *Image: Journal of Nursing Scholarship*, 30 (4), 315-321.

Lang, T. A., Hodge, M., Olson, V., Romano, P. S. and Kravitz, R. L., 2004. Nurse-patient ratios: a systematic review on the effects of nurse staffing on patient, nurse employee, and hospital outcomes. *Journal of Nursing Administration*, 34 (7/8), 326-337 312p.

Lawrence, E. L. and Turner, I. G., 2005. Review: Materials for urinary catheters: a review of their history and development in the UK. *Medical Engineering and Physics*, 27, 443-453.

Lecler, S. and Meyrueis, P., 2012. *Intrinsic optical fiber sensor*. INTECH Open Access Publisher.

Lee, I., Sokolsky, O., Chen, S., Hatcliff, J., Jee, E., Roederer, E. and Venkatasubramania, K., 2012. Challenges and Research Directions in Medical Cyber-Physical Systems. Philadelphia, USA: Dept. of Computer & Information Sci., University of Pennsylvania.

Legrand, M. and Payen, D., 2011. Understanding urine output in critically ill patients. *ANNALS OF INTENSIVE CARE*, 1.

Lewinton, D. A. and Kanagasundaram, D. S., 2011. *Acute Kidney InjuryAcute Kidney InjuryAcute Kidney Injury* [online]. Available from: http://www.renal.org/guidelines/modules/acute-kidney-injury#sthash.BnpcDvuh.dpbs [Accessed 15 January 2015].

Lidwell, W., Holden, K. and Butler, J., 2010. *Universal principles of design : 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design.* [Non-fiction]. Beverly, Mass. : Rockport, 2010.

Linthicum, D., 2014. *The cloud is the secret weapon in the Internet of things* [online]. Available from: http://www.infoworld.com/article/2608029/cloud-computing/the-cloud-is-the-secret-weapon-in-the-internet-of-things.html [Accessed March 24, 2015].

Liu, Y., 2010. *Wireless remote patient monitoring on general hospital wards.* [http://search.ebscohost.com/login.aspx?direct=true&db=cat00012a&AN=bourne.599034&site=eds-live&scope=site]. Bournemouth University.

Loudon, G., Gill, S. and Culverhouse, I., 2012. Investigation into the insights generated through the application of interactive prototyping during the early stages of the design process: University of Wales, 2012.

Lovick, J. and Angeli, P., 2004. Droplet size and velocity profiles in liquid–liquid horizontal flows. *Chemical Engineering Science*, 59, 3105-3115.

Madahar, M., 2011. *Spreadsheet Use For Strategic Decision-Making An Analysis of Spreadsheet Use and Associated Risk* [http://search.ebscohost.com/login.aspx?direct=true&db=edsble&AN=ethos.016812519&site=eds-live&scope=site.]. Cardiff Metropolitan University.

Madden, V., 2000. Nutritional benefits of drinks. *Nursing standard (Royal College of Nursing (Great Britain) : 1987)*, 15 (13-15), 47-52; quiz 54-55.

Margolis, M., 2012. *Arduino cookbook / Michael Margolis* [Non-fiction]. 2nd edition. Sebastopol : O'Reilly,.

Mason, R. and Carey, T., 1983. Prototyping interactive information systems. *Communications of the ACM*, 26 (5), 347-354.

Mazzola, L., Bemporad, E. and Carassiti, F., 2012. An easy way to measure surface free energy by drop shape analysis. *Measurement*, 45 (3), 317-324.

McGloin, H., Adam, S. K. and Singer, M., 1999. Unexpected deaths and referrals to intensive care of patients on general wards. Are some cases potentially avoidable? *JOURNAL OF THE ROYAL COLLEGE OF PHYSICIANS OF LONDON*, 33 (3), 255-259.

McMahon, B. A., Phelan, D. and Murray, P. T., 2010. Oliguria and anuria (acute kidney injury part I) (Update 2010.): European Society of Intensive Care Medicine.

McMillen, R. and Pitcher, B., 2011. The balancing act: body fluids and protecting patient health. *British Journal of Healthcare Assistants*, 5 (3), 117-121.

Meingast, M., Roosta, T. and Sastry, S., 2006. Security and privacy issues with health care information technology. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 5453-5458.

Mell, P. and Grance, T., 2010. The NIST Definition of Cloud Computing. *Communications of the ACM*, 53 (6), 50-50.

Mentes, J. C., Wakefield, B. and Culp, K., 2006. Use of a urine color chart to monitor hydration status in nursing home residents. *BIOLOGICAL RESEARCH FOR NURSING*, 7 (3), 197-203.

Metheny, N. M., 1987. *Fluid and electrolyte balance : nursing considerations / Norma Milligan Metheny ; with 18 contributions* [Bibliographies

Non-fiction]. Philadelphia : Lippincott, c1987.

Mills, C. O., Elias, E., Martin, G. H., Woo, M. T. and Winder, A. F., 1988. Surface tension properties of human urine: relationship with bile salt concentration. *Journal Of Clinical Chemistry And Clinical Biochemistry. Zeitschrift Für Klinische Chemie Und Klinische Biochemie*, 26 (4), 187-194.

Mosby's Medical Dictionary, 2009. *Intensive Care Unit* [online]. Available from: http://medical-dictionary.thefreedictionary.com/intensive+care+unit [Accessed March 24 2015].

National Confidential Enquiry into Patient Outcome and Death, 2009. *Adding Insult to Injury: A Review of the Care of Patients who Died in Hospital with a Primary Diagnosis of Acute Kidney Injury (acute Renal Failure): a Report by the National Confidential Enquiry Into Patient Outcome and Death*. NCEPOD.

National Institute for Health and Care Excellence, 2007. *Monitoring patients in hospital and caring for them if their health becomes worse* [Available from: http://publications.nice.org.uk/ifp50 [Accessed 15 Jan 15].

National Institute for Health and Care Excellence, 2013. *New NICE kidney guideline to save thousands of lives* [online]. Available from: http://www.nice.org.uk/guidance/cg169/resources/new-nice-kidney-guideline-to-save-thousands-of-lives [Accessed 12 Feb 2015].

National Instruments, n.d. *What Is Data Acquisition?* [online]. Berkshire: National Instruments UK. Available from: http://www.ni.com/data-acquisition/what-is/ [Accessed April 09, 2015].

National PatientSafety Agency (NPSA), 2007. Safer care for the acutely ill patient : learning from serious incidents: National Patient Safety Agency (NPSA),.

Nemeth, C. P., 2004. *Human factors methods for design: Making systems human-centered*. CRC press.

NHS, 2013. *Better kidney care could save thousands of lives* [online]. Available from: http://www.nhs.uk/news/2013/08August/Pages/Better-kidney-care-could-save-thousands-of-lives.aspx [Accessed 10 February 2015].

NHS Choices, 2014. *Acute kidney injury* [online]. Available from: http://www.nhs.uk/conditions/acute-kidney-injury/Pages/Introduction.aspx#who [Accessed April 04,2015].

Norgia, M., Magnani, A., Melchionni, D. and Pesatori, A., 2014. Optical system for drop volume measurement. *2014 IEEE International Instrumentation & Measurement Technology Conference (I2MTC) Proceedings*, 1322.

Norgia, M., Pesatori, A. and Rovati, L., 2012. Self-Mixing Laser Doppler Spectra of Extracorporeal Blood Flow: A Theoretical and Experimental Study. *IEEE SENSORS JOURNAL*, 12 (3), 552-557.

Oracle Corporation, (n.d.)-a. *Java SE 7 Features and Enhancements* [online]. Oracle Corporation. Available from: http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html [Accessed May 1, 2015].

Oracle Corporation, (n.d.)-b. *Java SE at a Glance* [online]. Oracle Corporation. Available from: http://www.oracle.com/technetwork/java/javase/overview/index.html [Accessed May 1, 2015].

Otero, A., Fernandez, R., Apalkov, A. and Armada, M., 2012. An Automatic Critical Care Urine Meter. *Sensors*, 13109.

Otero, A., Palacios, F., Akinfiev, T. and Apalkov, A., 2010a. A Low Cost Device for Monitoring the Urine Output of Critical Care Patients. *Sensors (14248220)*, 10, 10714-10732.

Otero, A., Palacios, F., Akinfiev, T. and Fernández, R., 2010b. A Device for Automatically Measuring and Supervising the Critical Care Patient'S Urine Output. *Sensors (14248220)*, 10, 934-951.

Otero, A., Palacios, F., Apalkov, A. and Fernández, R., 2013. A Simple and Low Cost Device for Automatically Supervising Urine Output of Critical Patients. *Biomedical Engineering Systems & Technologies (9783642297519)*, 15.

Otero, A., Panigrahi, B., Palacios, F., Akinfiev, T. and Fernandez, R., 2009. A Prototype Device to Measure and Supervise Urine Output of Critical Patients: InTech, 2009-10-01.

Overmyer, S. P., 1991. *Revolutionary vs. Evolutionary Rapid Prototyping: Balancing*

Software Productivity and HCI Design Concerns. Amsterdam:: Elsevier Science Pulications.

Pang, Z. A., Tian, J. A., Chen, Q. A. and Kth, S. f. i.-o. k. E. O., 2014. Intelligent packaging and intelligent medicine box for medication management towards the Internet-of-Things. *International Conference on Advanced Communication Technology, ICACT*, 352.

Payne, R. and Macdonald, B., 2004. Ambient technology - now you see it, now you don't. *BT Technology Journal*, 22 (3), 119-129.

Pearce, J. M., 2012. Building Research Equipment with Free, Open-Source Hardware. *Science*, 337 (6100), 1303-1304.

Pronovost, P. J., Angus, D. C., Dorman, T., Robinson, K. A., Dremsizov, T. T. and Young, T. L., 2002. Physician staffing patterns and clinical outcomes in critically ill patients: a systematic review. *JAMA*, 288 (17), 2151-2162.

Pullen, J. P., 2014. THE DEVICE CODE. *Entrepreneur*, 42 (2), 44-44.

Putnam, D. F., 1971. Composition and Concentrative Properties of Human Urine: NASA Contractor Report.

Quick, D. and Choo, K.-K. R., 2014. Google Drive: Forensic analysis of data remnants. *Journal of Network and Computer Applications*, 40, 179-193.

Raschke, R. A., Gollihare, B., Wunderlich, T. A., Guidry, J. R., Leibowitz, A. I., Peirce, J. C., Lemelson, L., Heisler, M. A. and Susong, C., 1998. A computer alert system to prevent injury from adverse drug events: development and evaluation in a community teaching hospital. *JAMA*, 280 (15), 1317-1320.

Ricci, Z., Cruz, D. and Ronco, C., 2008. The RIFLE criteria and mortality in acute kidney injury: A systematic review. *Kidney International*, 73 (5), 538-546.

Rodriguez, N., 2013. *This Killer New Service Lets You Connect To Any Popular API—Without Wrappers* [online]. Fastcolabs.com. Available from: http://www.fastcolabs.com/3020690/this-killer-new-service-lets-you-connect-to-any-popular-api-without-wrappers [Accessed April 24, 2015].

Rolim, C. O., Koch, F. L., Westphall, C. B., Werner, J., Fracalossi, A. and Salvador, G. S., 2010. A cloud computing solution for patient's data collection in health care institutions (pp. 95-99).

Rose, C., Parker, A., Jefferson, B. and Cartmell, E., 2015. The characterisation of faeces and urine; a review of the literature to inform advanced treatment technology. *Critical Reviews in Environmental Science and Technology*, (just-accepted), 00-00.

Rosner, M. H., 2013. Acute Kidney Injury in the Elderly. *Clinics in Geriatric Medicine*, 29, 565-578.

Rowe, M., Bozalek, V. and Frantz, J., 2013. Using Google Drive to facilitate a blended approach to authentic learning. *British Journal of Educational Technology*, 44 (4), 594-606.

Scales, K. and Pilsworth, J., 2008. The importance of fluid balance in clinical practice. *Nursing Standard*, 22 (47), 50-58.

Schrier, R. W., Shchekochikhin, D. and Ginès, P., 2012. Renal failure in cirrhosis: prerenal azotemia, hepatorenal syndrome and acute tubular necrosis. *Nephrology, Dialysis, Transplantation: Official Publication Of The European Dialysis And Transplant Association - European Renal Association*, 27 (7), 2625-2628.

Sean, C. and Kevin, C., 2012. Cloud Computing Technologies. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, (2), 59.

Shabot, M. M., LoBue, M. and Leyerle, B. J., 1988. An automatic PDMS interface for the Urotrack Plus 220 urimeter. *International Journal of Clinical Monitoring and Computing*, 5 (2), 125-131.

Shafranovich, Y., 2005. Common format and MIME type for Comma-Separated Values (CSV) files.

Shepherd, A., 2011. Measuring and managing fluid balance... [corrected] [published erratum appears in NURS TIMES 2011 Aug 16-29; 107(32-33):9]. *Nursing Times*, 107 (28), 12-16.

Shotts, J. F. and Hauf, E., 1986. The Hewlett-Packard 1000 - Vitalmetrics VM220 Connection: A description of the automated ultrasonic urine output measurement system in the CICU of Genolier Clinic. *International Journal of Clinical Monitoring and Computing*, 3 (3), 175-182.

Sibley, E. H., Mantei, M. M. and Teorey, T. J., 1988. COST/BENEFIT ANALYSIS FOR INCORPORATING HUMAN FACTORS IN THE SOFTWARE LIFECYCLE. *Communications of the ACM*, 31 (4), 428-439.

Siegel, M. D., 2009. End-of-Life Decision Making in the ICU. *Clinics in Chest Medicine*, 30, 181-194.

Song, Q., Zhang, G. and Qiu, Z., 2003. Drop Growth Monitoring and Drop Volume Measurement Based on Image Drop Analysis with CCD. *Instrumentation Science & Technology*, 31 (1), 1.

Song, Q., Zhang, G. X. and Qiu, Z. R., 2005. Review of drop analysis technology for liquid property study. *OPTO-ELECTRONICS REVIEW*, 13 (1), 1-8.

SONOTEC, 2015. *Ready for 60601 3rd Edition with Redundant Air Bubble Sensor SONOCHECK ABD08* [SOTEC, Halle (Saale), Germany. Available from: http://www.sonotec.eu/fileadmin/media/DE/Produkte/Nicht_Invasive_Fluessigkeitsueberwachung/Applikationen/AN_Redundant_Bubble_Sensor_engl_Rev_2.1_2015-02-05.pdf [Accessed

Teddlie, C. and Tashakkori, A., 2009. *Foundations of mixed methods research : integrating quantitative and qualitative approaches in the social and behavioral sciences* [Bibliographies

Non-fiction]. Los Angeles, [Calif.] ; London : SAGE, c2009.

The Government Statistical Service, 2015. *STATISTICAL PRESS NOTICE: MONTHLY CRITICAL CARE BEDS AND CANCELLED URGENT OPERATIONS DATA, ENGLAND JANUARY 2015* [Available from: http://www.england.nhs.uk/statistics/wp-content/uploads/sites/2/2013/04/MSitRep-Pre-Release-List-September.pdf [Accessed 24 March 2015].

The National Meseum of American History, n.d. *Urinometer* [online]. Washington D.C.,: The Smithsonian Institution. Available from: http://americanhistory.si.edu/collections/search/object/nmah_735237 [Accessed 31 March 2015].

The Oregon Health & Science University, 2013. *OHSU notifies patients of 'cloud' health information storage* [online]. Portland: The Oregon Health & Science University,. Available from: http://www.ohsu.edu/xd/about/news_events/news/2013/07-28-ohsu-notifies-patients-o.cfm [Accessed April 13, 2015].

The World Health Organization, 2005. WHO Report: Nutrient Minerals in Drinking Water. *WATER CONDITIONING AND PURIFICATION INTERNATIONAL*, 62-65.

Thurow, K., Krüger, T. and Stoll, N., 2009. An optical approach for the determination of droplet volumes in nanodispensing. *Journal of Automated Methods and Management in Chemistry*, 2009.

Tidy, C., 2012. *Polyuria* [online]. Available from: http://www.patient.co.uk/doctor/polyuria [Accessed 17 November 2014].

Uchino, S., Bellomo, R., Bagshaw, S. M. and Goldsmith, D., 2010. Transient azotaemia is associated with a high risk of death in hospitalized patients. *Nephrology, Dialysis, Transplantation: Official Publication Of The European Dialysis And Transplant Association - European Renal Association*, 25 (6), 1833-1839.

Ulrich, K. and Eppinger, S. D., 2012. *Product design and development* [Non-fiction]. London: McGraw-Hill Education.

Van Santvliet, L. and Ludwig, A., 2004. Determinants of eye drop size. *Survey of Ophthalmology*, 49 (2), 197-213.

Vermesan, O. and Friess, P., 2014. Internet of Things-From Research and Innovation to Market Deployment.

Vincent, J. L., Sakr, Y., Sprung, C. L., Ranieri, V. M., Reinhart, K., Gerlach, H., Moreno, R., Carlet, J., Gall, J. R. l. and Payen, D., 2006. Sepsis in European intensive care units: results of the SOAP study. *Critical Care Medicine*, 34 (2), 344-353.

Wang, J., Abid, H., Lee, S., Shu, L. and Xia, F., 2011. A secured health care application architecture for cyber-physical systems. *Control Engineering and Applied Informatics*, 13 (3), 101-108.

Warren, J. W., 2001. Catheter-associated urinary tract infections. *International Journal of Antimicrobial Agents*, 17, 299-303.

Weinberg, A. D., Pals, J. K., Levesque, P. G., Beal, L. F., Cunningham, T. J. and Minaker, K. L., 1994. Dehydration and death during febrile episodes in the nursing home. *Journal of the American Geriatrics Society*, 968-971 964p.

Westen, D., 2012. FORUM: ADVANTAGES AND DISADVANTAGES OF A PROTOTYPE-MATCHING APPROACH TO PSYCHIATRIC DIAGNOSIS: Prototype diagnosis of psychiatric syndromes. *World Psychiatry*, 11, 16-21.

Wiedemann, H. P., Wheeler, A. P., Bernard, G. R., Thompson, B. T., Hayden, D., deBoisblanc, B., Connors, A. F., Hite, R. D. and Harabin, A. L., 2006. Comparison of two fluid-management strategies in acute lung injury. *NEW ENGLAND JOURNAL OF MEDICINE*, 354 (24), 2564-2575.

Wilkes, K., 2014. *Cloud storage now available for OHSU researchers* [online]. Portland: The Oregon Health & Science Universit,. Available from: http://www.ohsu.edu/blogs/researchnews/2014/08/05/cloud-storage-now-available-for-ohsu-researchers/ [Accessed April 13, 2015].

Williams, M. H., 1999. Nutrition for health, fitness and sport, *Nutrition for health, fitness and sport.* Boston; USA: WCB/McGraw-Hill.

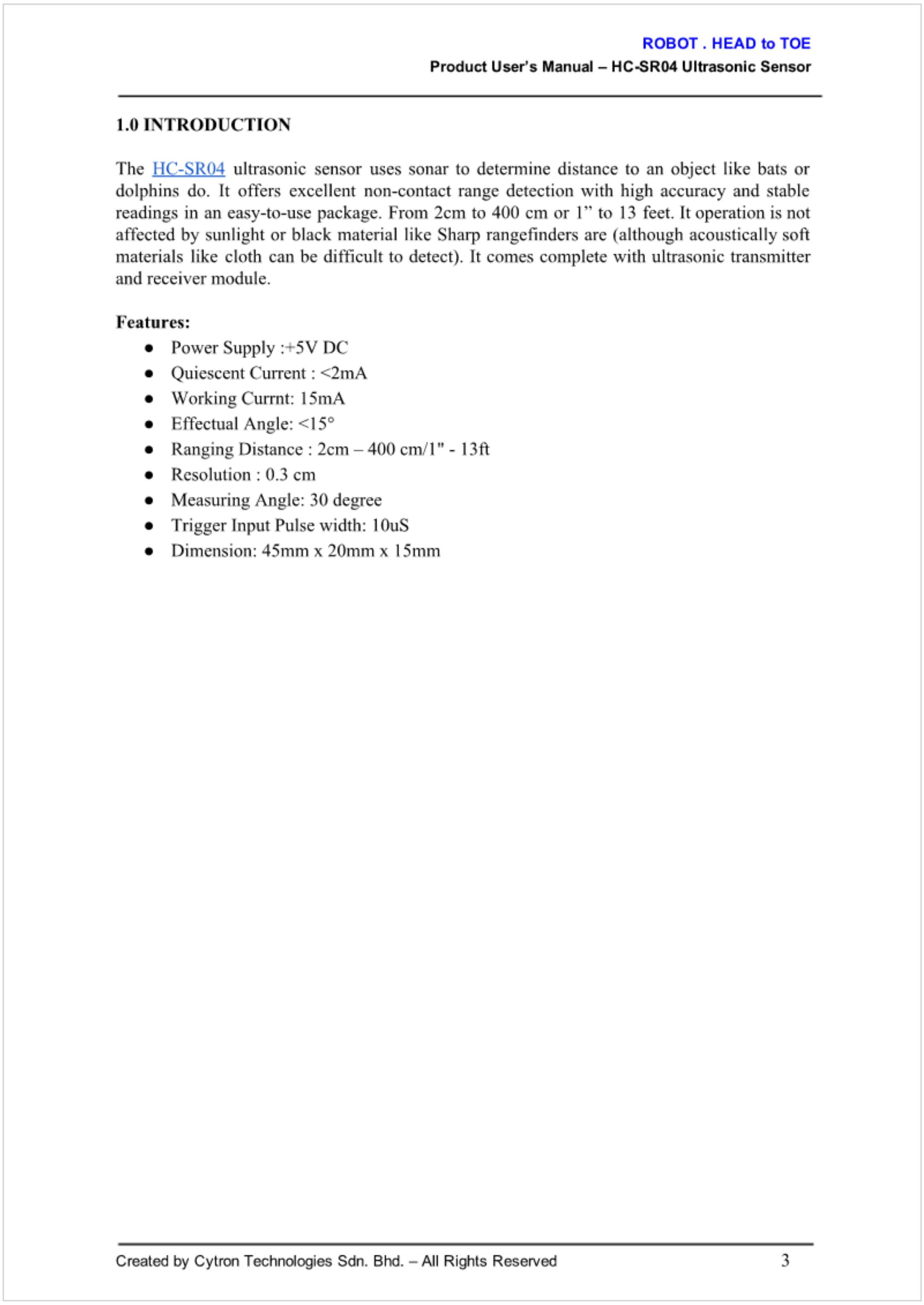
Wilson, J. and Rosenberg, D., 1988. Chapter 39: Rapid Prototyping for User Interface Design. *Handbook of Human-Computer Interaction*, 859-875.

Yu, L., Lu, Y. and Zhu, X., 2012. Smart hospital based on internet of things. *Journal of Networks*, 7 (10), 1654-1661.

Zacks Equity Research, 2015. *Masimo's Radical-7 Installed by Jordan Pediatric Hospital* [online]. Available from: http://www.zacks.com/stock/news/190126/masimos-radical7-installed-by-jordan-pediatric-hospital [Accessed September 30, 2015].

APPENDIX

Appendix 1. HC-SR04 Ultrasonic datasheet



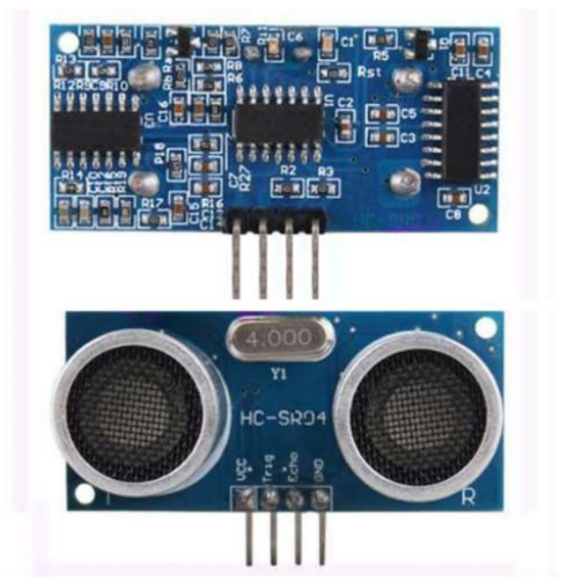
1.0 INTRODUCTION

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats or dolphins do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

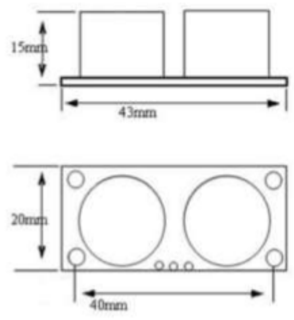
Features:

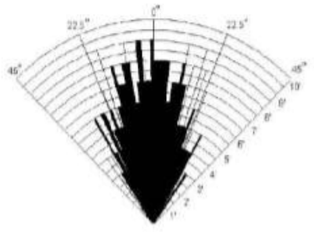
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Currnt: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" - 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

3.0 PRODUCT LAYOUT

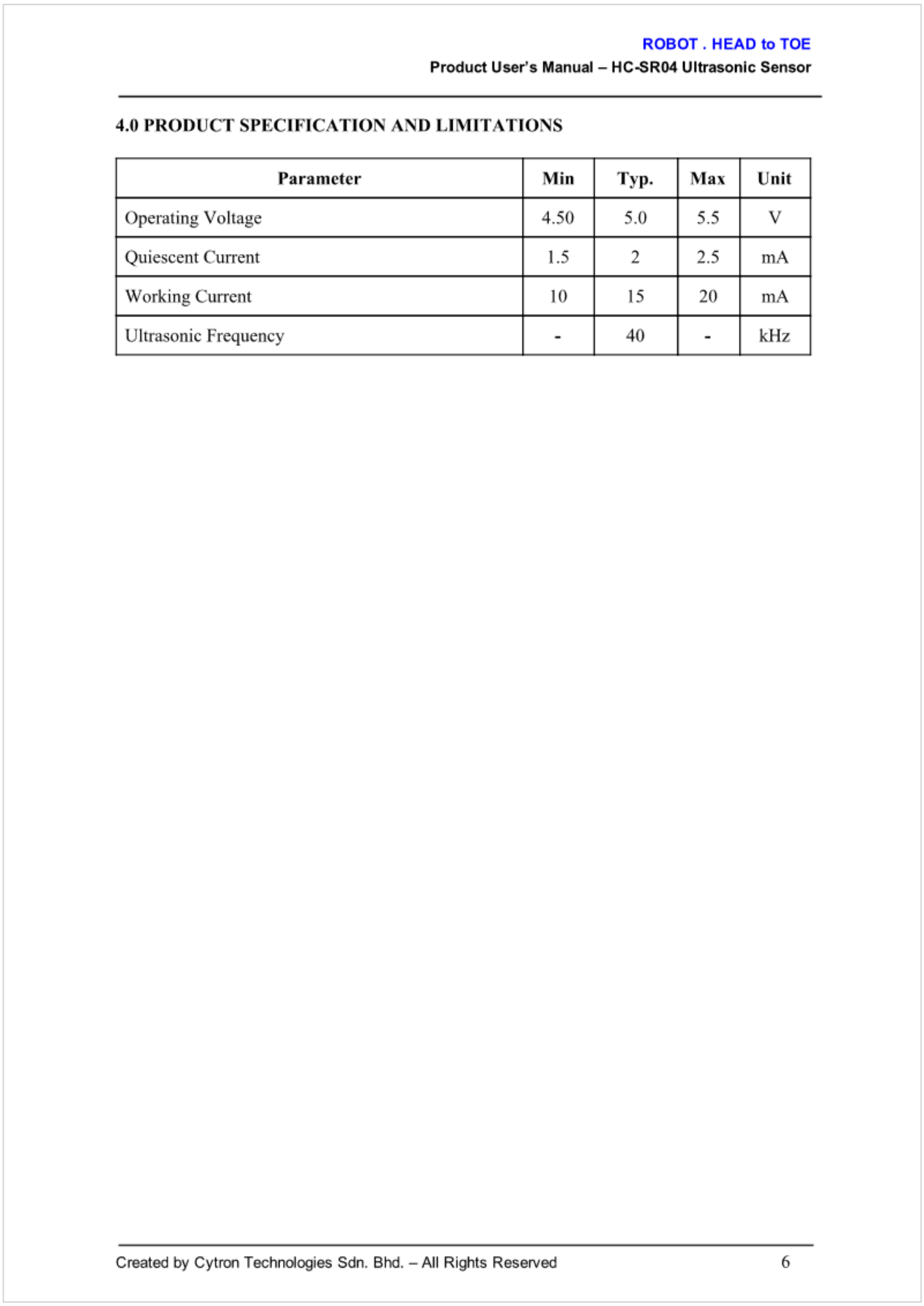


VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND





Practical test of performance,
Best in 30 degree angle



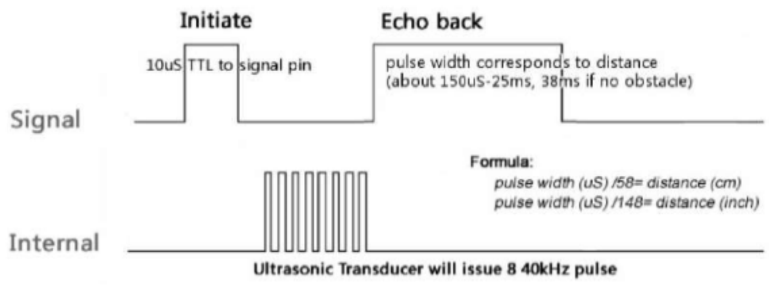
4.0 PRODUCT SPECIFICATION AND LIMITATIONS

Parameter	Min	Typ.	Max	Unit
Operating Voltage	4.50	5.0	5.5	V
Quiescent Current	1.5	2	2.5	mA
Working Current	10	15	20	mA
Ultrasonic Frequency	-	40	-	kHz

5.0 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)
- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s

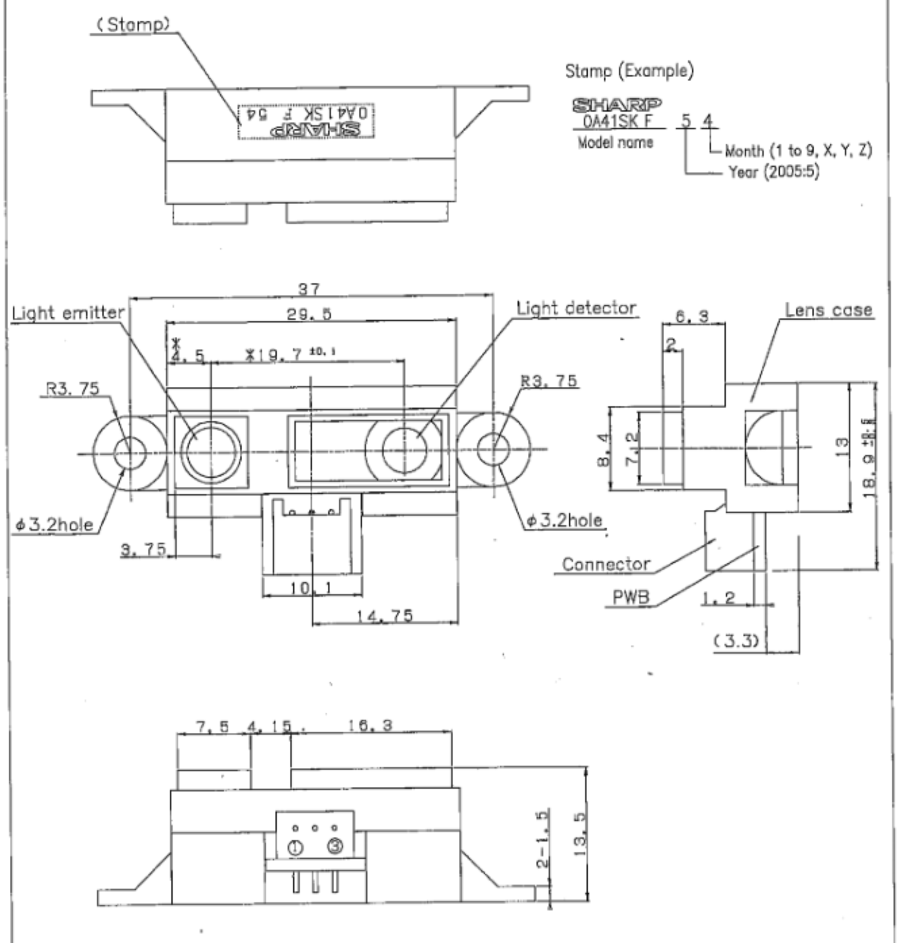

Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance
(about 150uS-25ms, 38ms if no obstacle)
Signal
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Internal
Ultrasonic Transducer will issue 8 40kHz pulse

Note:
- Please connect the GND pin first before supplying power to VCC.
- Please make sure the surface of object to be detect should have at least 0.5 meter^2 for better performance.

Appendix 2. Sharp GP2Y0A41SK0F: IR-Proximity

3/9
ED-05G101A GP2Y0A41SK0F
March 5, 2007

2. Outline



Connector signal

	Signal name
①	Vo
②	GND
③	Vcc

Connector :
J.S.T.TRADING COMPANY,LTD.
S3B−PH

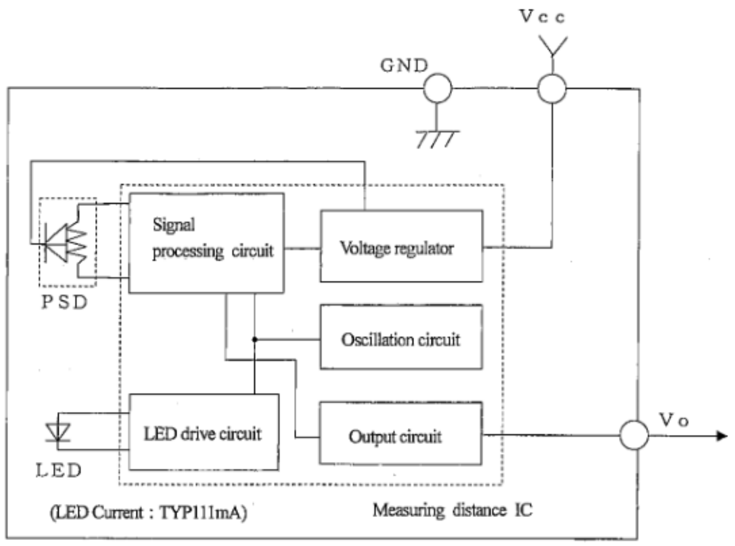
Materials
Lens: Acrylic acid resin
(Visible light cut−off resin)
Case: Carbonic ABS
(Conducttive resin)
PCB: Paper phenol

Unit: mm Scale: 2/1

Note 1: The dimensions marked ※ are described the dimensions of lens center position.
Note 2: Unspecified tolerance shall be ±0.3mm.
Note 3: The dimensions in parenthesis are shown for reference.

Name	GP2Y0A41SK0F Outline Dimensions
Drawing No.	CY13036j02

3-1 Schematic



3-2 Absolute maximum ratings (Ta=25℃, Vcc=5V)

Parameter	Symbol	Ratings	Unit	Remark
Supply voltage	Vcc	-0.3 to +7	V	-
Output terminal voltage	Vo	-0.3 to Vcc+0.3	V	-
Operating temperature	Topr	-10 to +60	℃	-
Storage temperature	Tstg	-40 to +70	℃	-

Operating supply voltage

Symbol	Rating	Unit	Remark
Vcc	4.5 to 5.5	V	-

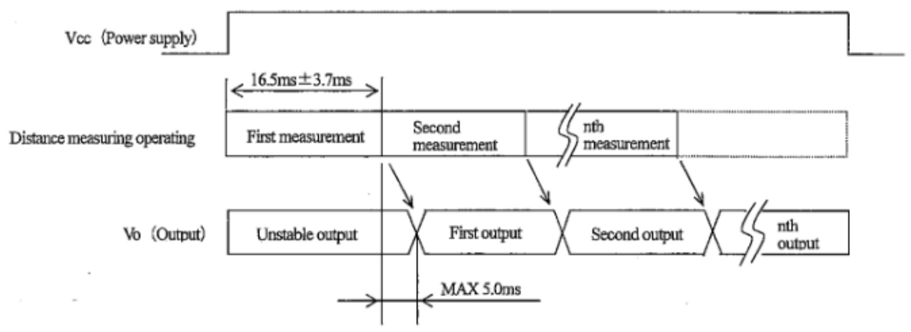
3-3 Electro-optical Characteristics

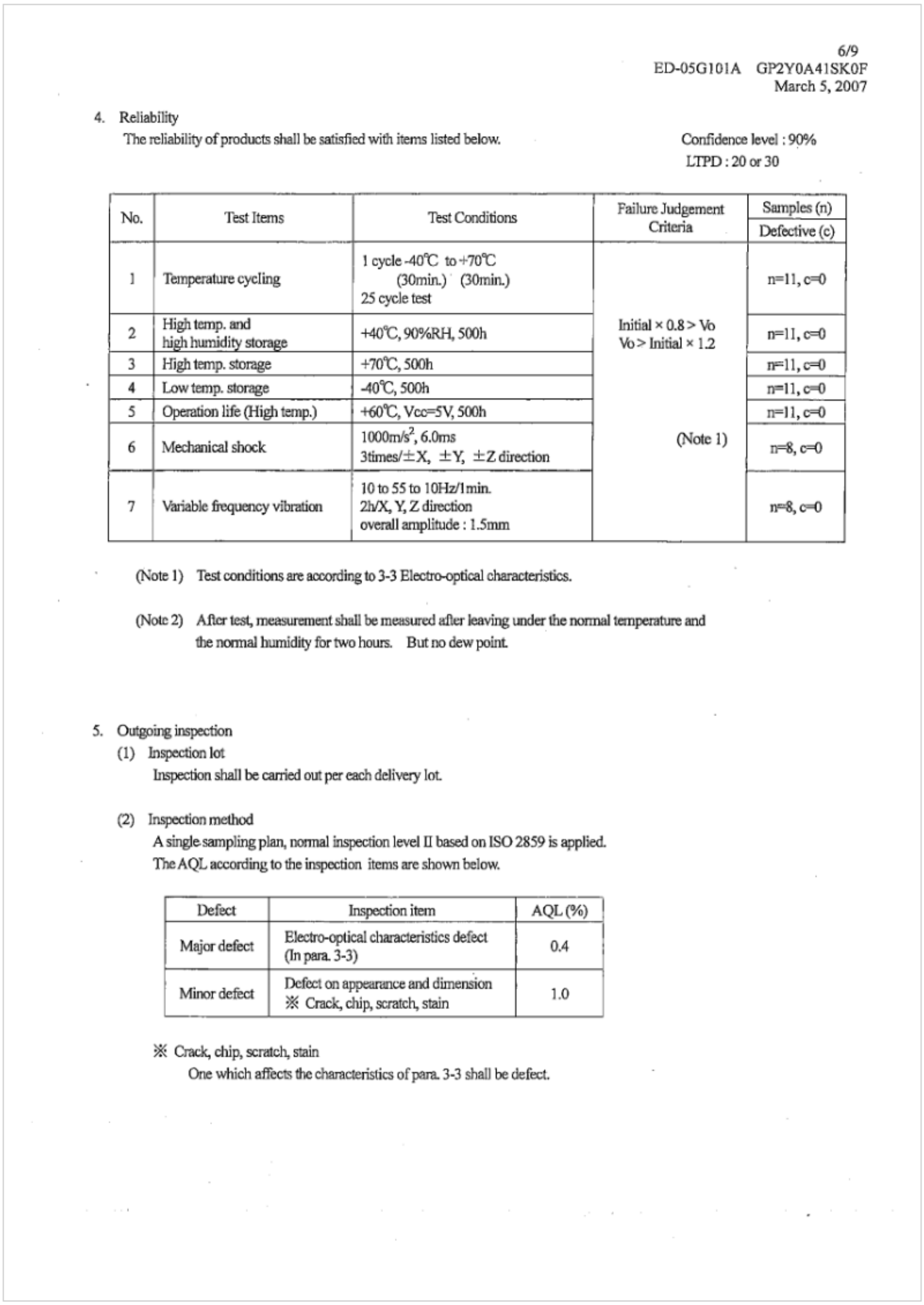
Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Measuring distance range	ΔL	(Note 1)	4	-	30	cm
Output terminal voltage	Vo	L=30cm (Note 1)	0.25	0.4	0.55	V
Output voltage difference	ΔVo	Output change at L change (30cm → 4cm) (Note 1)	1.95	2.25	2.55	V
Average supply current	Icc	L=30cm (Note 1)	-	12	22	mA

※ L : Distance to reflective object

(Note 1) Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27・white face, reflective ratio ; 90%)

3-4 Timing chart





4. Reliability

The reliability of products shall be satisfied with items listed below.

Confidence level : 90%
LTPD : 20 or 30

No.	Test Items	Test Conditions	Failure Judgement Criteria	Samples (n) Defective (c)
1	Temperature cycling	1 cycle -40℃ to +70℃ (30min.) (30min.) 25 cycle test	Initial × 0.8 > Vo Vo > Initial × 1.2 (Note 1)	n=11, c=0
2	High temp. and high humidity storage	+40℃, 90%RH, 500h		n=11, c=0
3	High temp. storage	+70℃, 500h		n=11, c=0
4	Low temp. storage	-40℃, 500h		n=11, c=0
5	Operation life (High temp.)	+60℃, Vcc=5V, 500h		n=11, c=0
6	Mechanical shock	1000m/s^2, 6.0ms 3times/±X, ±Y, ±Z direction		n=8, c=0
7	Variable frequency vibration	10 to 55 to 10Hz/1min. 2h/X, Y, Z direction overall amplitude : 1.5mm		n=8, c=0

(Note 1) Test conditions are according to 3-3 Electro-optical characteristics.

(Note 2) After test, measurement shall be measured after leaving under the normal temperature and the normal humidity for two hours. But no dew point.

5. Outgoing inspection

(1) Inspection lot

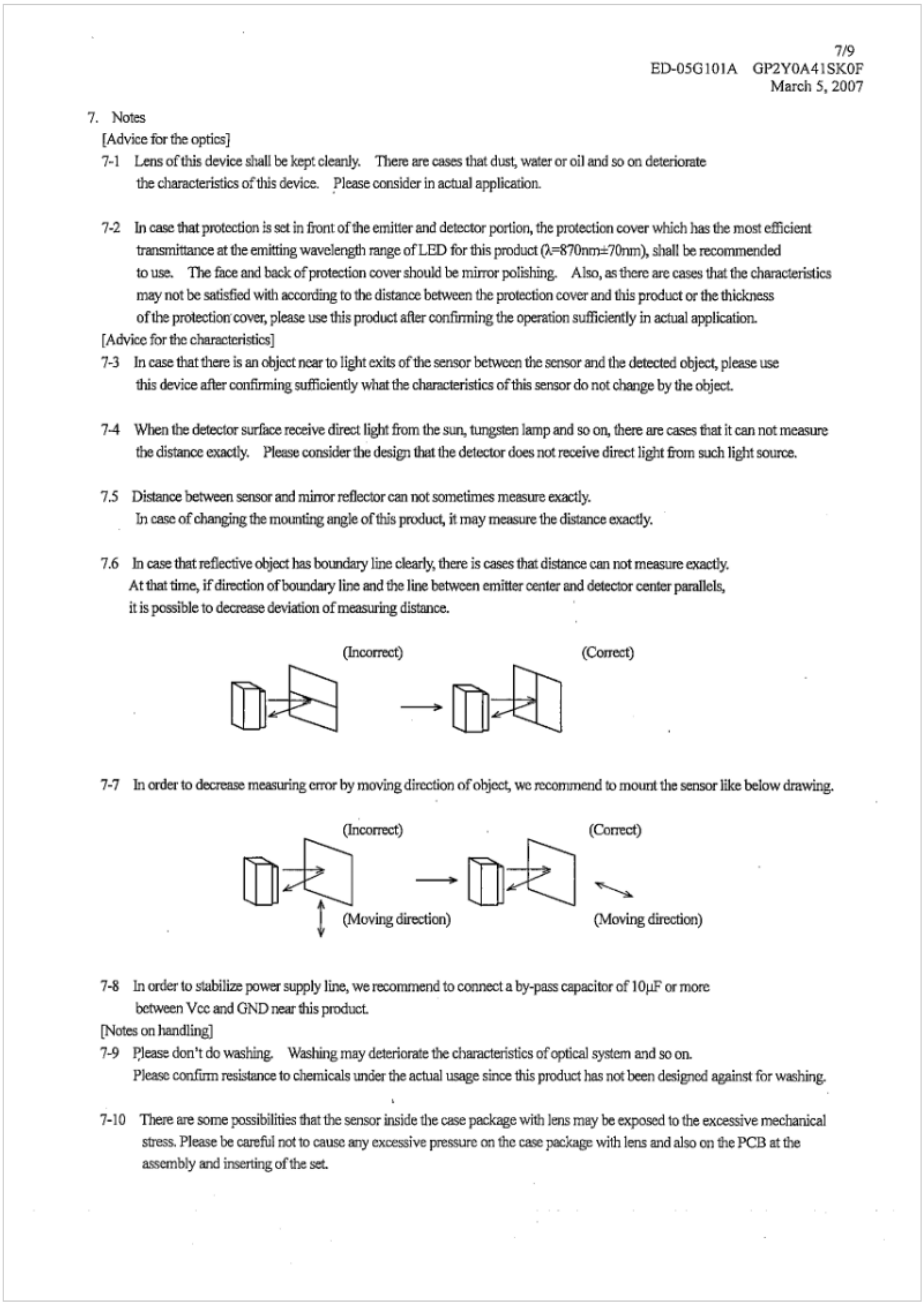
Inspection shall be carried out per each delivery lot.

(2) Inspection method

A single sampling plan, normal inspection level II based on ISO 2859 is applied.
The AQL according to the inspection items are shown below.

Defect	Inspection item	AQL (%)
Major defect	Electro-optical characteristics defect (In para. 3-3)	0.4
Minor defect	Defect on appearance and dimension ※ Crack, chip, scratch, stain	1.0

※ Crack, chip, scratch, stain
One which affects the characteristics of para. 3-3 shall be defect.



7. Notes

[Advice for the optics]

7-1 Lens of this device shall be kept cleanly. There are cases that dust, water or oil and so on deteriorate the characteristics of this device. Please consider in actual application.

7-2 In case that protection is set in front of the emitter and detector portion, the protection cover which has the most efficient transmittance at the emitting wavelength range of LED for this product (λ=870nm±70nm), shall be recommended to use. The face and back of protection cover should be mirror polishing. Also, as there are cases that the characteristics may not be satisfied with according to the distance between the protection cover and this product or the thickness of the protection cover, please use this product after confirming the operation sufficiently in actual application.

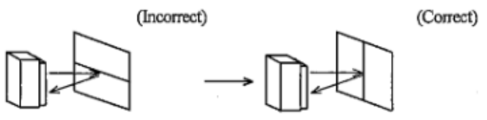
[Advice for the characteristics]

7-3 In case that there is an object near to light exits of the sensor between the sensor and the detected object, please use this device after confirming sufficiently what the characteristics of this sensor do not change by the object.

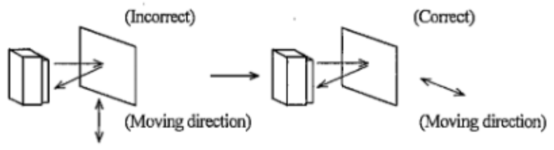
7-4 When the detector surface receive direct light from the sun, tungsten lamp and so on, there are cases that it can not measure the distance exactly. Please consider the design that the detector does not receive direct light from such light source.

7.5 Distance between sensor and mirror reflector can not sometimes measure exactly.
In case of changing the mounting angle of this product, it may measure the distance exactly.

7.6 In case that reflective object has boundary line clearly, there is cases that distance can not measure exactly.
At that time, if direction of boundary line and the line between emitter center and detector center parallels, it is possible to decrease deviation of measuring distance.



7-7 In order to decrease measuring error by moving direction of object, we recommend to mount the sensor like below drawing.



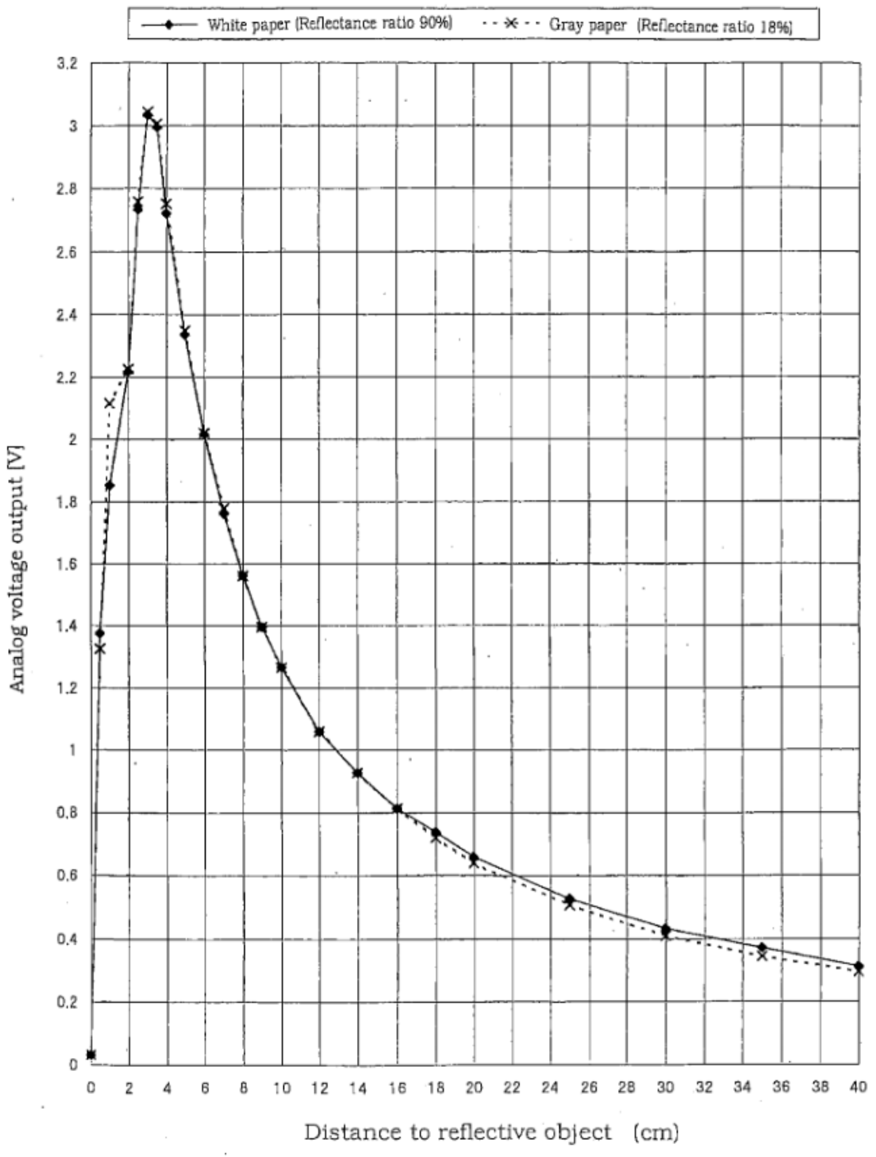
7-8 In order to stabilize power supply line, we recommend to connect a by-pass capacitor of 10μF or more between Vcc and GND near this product.

[Notes on handling]

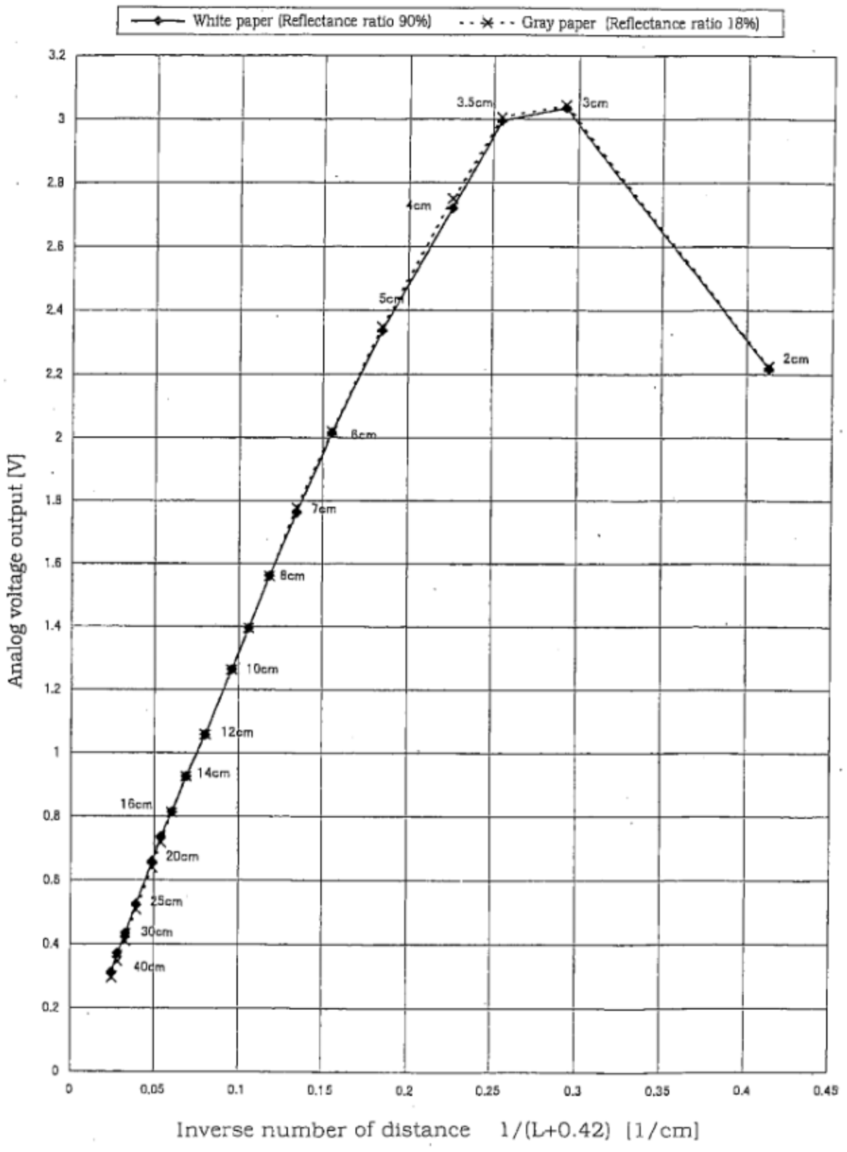
7-9 Please don't do washing. Washing may deteriorate the characteristics of optical system and so on.
Please confirm resistance to chemicals under the actual usage since this product has not been designed against for washing.

7-10 There are some possibilities that the sensor inside the case package with lens may be exposed to the excessive mechanical stress. Please be careful not to cause any excessive pressure on the case package with lens and also on the PCB at the assembly and inserting of the set.

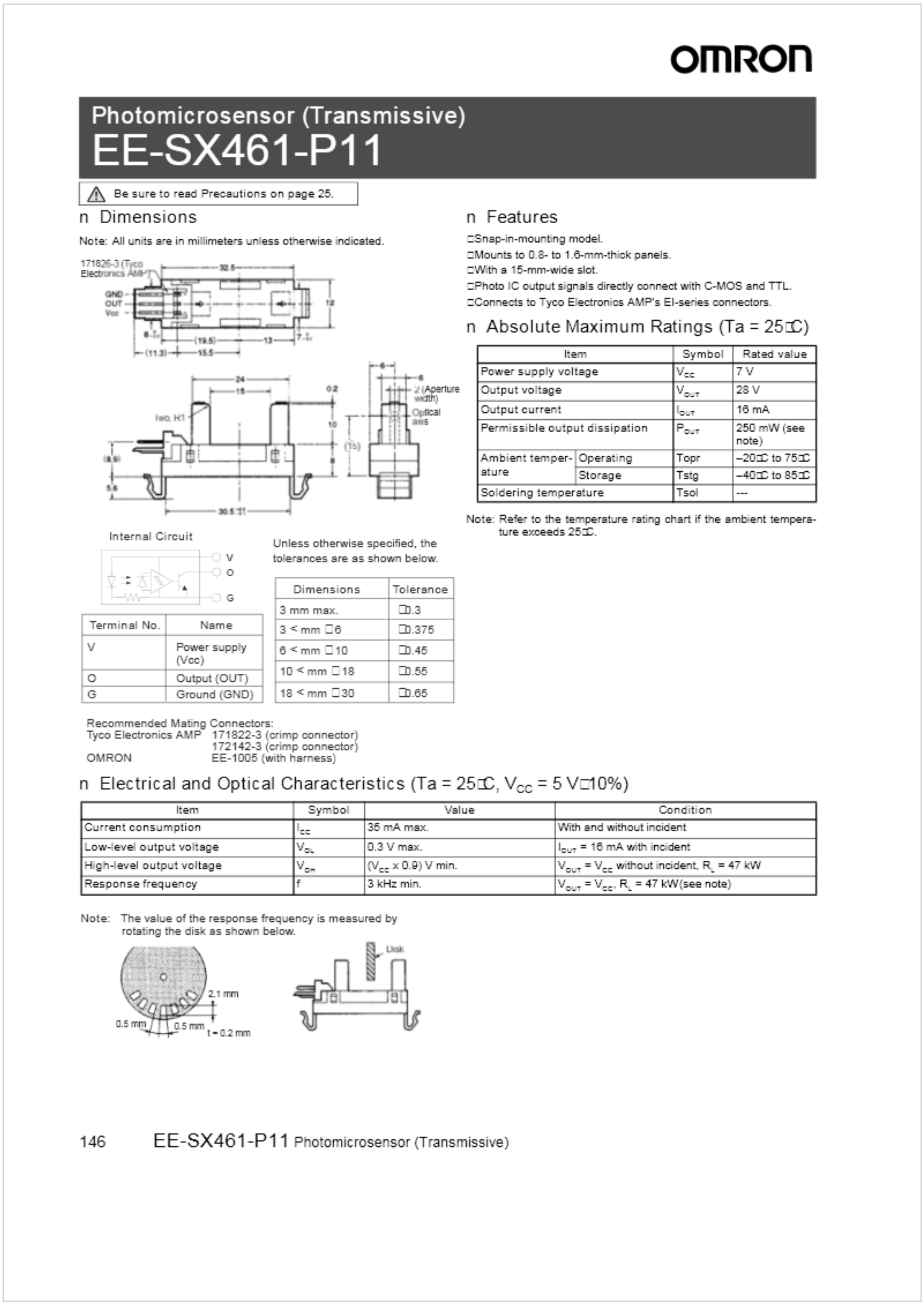
6-1 GP2Y0A41SK0F Example of output distance characteristics



6-2 GP2Y0A41SK0F Example of output distance characteristics with the inverse of distance



Appendix 3. OMRON EE-SX461-P11 Photo sensor



OMRON

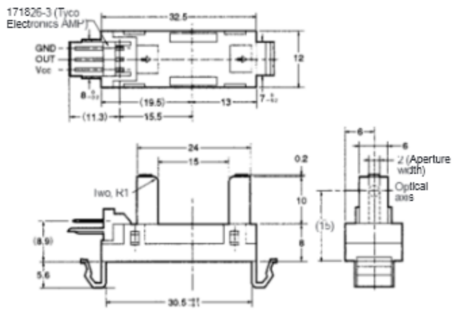
Photomicrosensor (Transmissive)

EE-SX461-P11

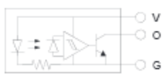
Be sure to read Precautions on page 25.

n Dimensions

Note: All units are in millimeters unless otherwise indicated.



Internal Circuit



Terminal No.	Name
V	Power supply (Vcc)
O	Output (OUT)
G	Ground (GND)

Unless otherwise specified, the tolerances are as shown below.

Dimensions	Tolerance
3 mm max.	□0.3
3 < mm □6	□0.375
6 < mm □10	□0.45
10 < mm □18	□0.55
18 < mm □30	□0.65

Recommended Mating Connectors:

Tyco Electronics AMP	171822-3 (crimp connector)
	172142-3 (crimp connector)
OMRON	EE-1005 (with harness)

n Features

□Snap-in-mounting model.
□Mounts to 0.8- to 1.6-mm-thick panels.
□With a 15-mm-wide slot.
□Photo IC output signals directly connect with C-MOS and TTL.
□Connects to Tyco Electronics AMP's EI-series connectors.

n Absolute Maximum Ratings (Ta = 25□C)

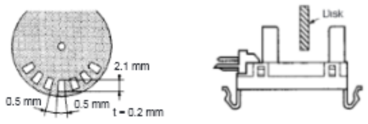
Item		Symbol	Rated value
Power supply voltage		V_{CC}	7 V
Output voltage		V_{OUT}	28 V
Output current		I_{OUT}	16 mA
Permissible output dissipation		P_{OUT}	250 mW (see note)
Ambient temperature	Operating	Topr	–20□C to 75□C
	Storage	Tstg	–40□C to 85□C
Soldering temperature		Tsol	---

Note: Refer to the temperature rating chart if the ambient temperature exceeds 25□C.

n Electrical and Optical Characteristics (Ta = 25□C, V_{CC} = 5 V□10%)

Item	Symbol	Value	Condition
Current consumption	I_{CC}	35 mA max.	With and without incident
Low-level output voltage	V_{OL}	0.3 V max.	I_{OUT} = 16 mA with incident
High-level output voltage	V_{OH}	(V_{CC} x 0.9) V min.	V_{OUT} = V_{CC} without incident, R_L = 47 kW
Response frequency	f	3 kHz min.	V_{OUT} = V_{CC}, R_L = 47 kW(see note)

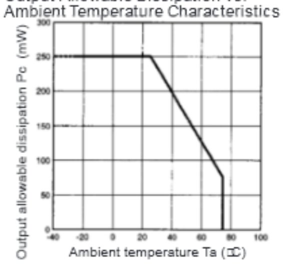
Note: The value of the response frequency is measured by rotating the disk as shown below.



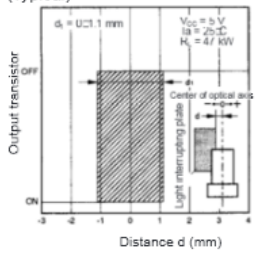
146 EE-SX461-P11 Photomicrosensor (Transmissive)

n Engineering Data

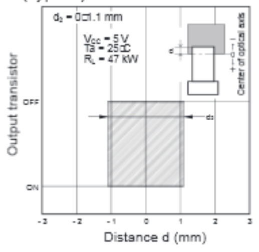
Output Allowable Dissipation vs. Ambient Temperature Characteristics



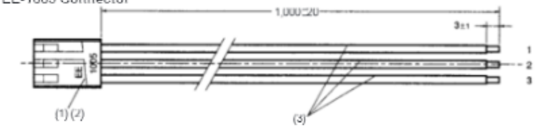
Sensing Position Characteristics (Typical)



Sensing Position Characteristics (Typical)



EE-1005 Connector

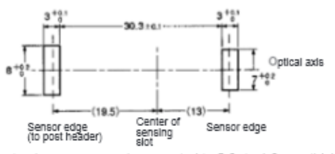


No.	Name	Model	Quantity	Maker
1	Receptacle housing	171822-3	1	Tyco Electronics AMP
2	Receptacle contact	170262-1	3	Tyco Electronics AMP
3	Lead wire	UL1007 AWG24	3	---

Wiring

Connector circuit no.	Lead wire color	Output when connected to EE-SX461-P11
1	Red	V_{CC}
2	Orange	OUT
3	Yellow	GND

n Recommended Mounting Hole Dimensions and Mounting and Dismounting Method



The Photomicrosensor can be mounted to 0.8- to 1.6-mm-thick panels.

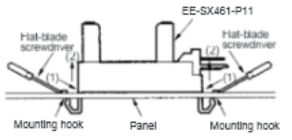
Refer to the above mounting hole dimensions and open the mounting holes in the panel to which the Photomicrosensor will be mounted.

Insert into the holes the Photomicrosensor's mounting portions with a force of three to five kilograms but do not press in the Photomicrosensor at one time. The Photomicrosensor can be easily mounted by inserting the mounting portions halfway and then slowly pressing the Photomicrosensor onto the panel.

There are two ways to dismount the Photomicrosensor. Refer to the following.

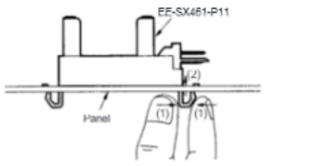
Dismounting with Screwdriver

Press the mounting hooks of the Photomicrosensor with a flat-blade screwdriver as shown in the following illustration and pull up the Photomicrosensor.



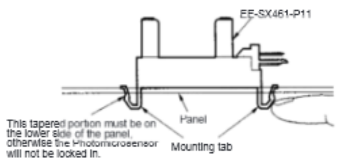
Dismounting by Hand

Squeeze the mounting tabs as shown in the following illustration and press the mounting tabs upwards.



Pressed mounting holes are ideal for mounting the Photomicrosensor. When mounting the Photomicrosensor to a panel that has pressed mounting holes for the Photomicrosensor, be sure to mount the Photomicrosensor on the pressing side of the panel, otherwise it may be difficult to mount the Photomicrosensor and an insertion force of five to six kilograms may be required.

When mounting the Photomicrosensor to a panel that has mounting holes opened by pressing, make sure that the mounting holes have no burrs, otherwise the lock mechanism of the Photomicrosensor will not work perfectly. After mounting the Photomicrosensor to a panel, be sure to check if the lock mechanism is working perfectly.


EE-SX461-P11
This tapered portion must be on the lower side of the panel, otherwise the Photomicrosensor will not be locked in.
Panel
Mounting tab

Appendix 4. Source code library for system timer (C++)

SimpleTimer.h

```
/*
 * SimpleTimer.h
 *
 * SimpleTimer - A timer library for Arduino.
 * Author: mromani@ottotecnica.com
 * Copyright (c) 2010 OTTOTECNICA Italy
 *
 */

#ifndef SIMPLETIMER_H
#define SIMPLETIMER_H

#if defined(ARDUINO) && ARDUINO >= 100
#include <Arduino.h>
#else
#include <WProgram.h>
#endif

typedef void (*timer_callback)(void);
class SimpleTimer {

public:
    // maximum number of timers
    const static int MAX_TIMERS = 10;

    // setTimer() constants
    const static int RUN_FOREVER = 0;

    const static int RUN_ONCE = 1;

    // constructor
    SimpleTimer();

    // this function must be called inside loop()
    void run();

    // call function f every d milliseconds
    int setInterval(long d, timer_callback f);

    // call function f once after d milliseconds
    int setTimeout(long d, timer_callback f);

    // call function f every d milliseconds for n times
    int setTimer(long d, timer_callback f, int n);

    // destroy the specified timer
    void deleteTimer(int numTimer);

    // restart the specified timer
    void restartTimer(int numTimer);

    // returns true if the specified timer is enabled
    boolean isEnabled(int numTimer);
    // enables the specified timer
    void enable(int numTimer);

    // disables the specified timer
    void disable(int numTimer);

    // enables the specified timer if it's currently disabled,
    // and vice-versa
    void toggle(int numTimer);

    // returns the number of used timers
    int getNumTimers();

    // returns the number of available timers
    int getNumAvailableTimers() { return MAX_TIMERS - numTimers; };

private:
    // deferred call constants
    const static int DEFCALL_DONTRUN = 0;       // don't call the
callback function
    const static int DEFCALL_RUNONLY = 1;       // call the callback
function but don't delete the timer
    const static int DEFCALL_RUNANDDEL = 2;      // call the
callback function and delete the timer

    // find the first available slot
    int findFirstFreeSlot();

    // value returned by the millis() function
    // in the previous run() call
    unsigned long prev_millis[MAX_TIMERS];

    // pointers to the callback functions
    timer_callback callbacks[MAX_TIMERS];

    // delay values
    long delays[MAX_TIMERS];

    // number of runs to be executed for each timer
    int maxNumRuns[MAX_TIMERS];

    // number of executed runs for each timer
    int numRuns[MAX_TIMERS];

    // which timers are enabled
    boolean enabled[MAX_TIMERS];

    // deferred function call (sort of) - N.B.: this array is only
used in run()
    int toBeCalled[MAX_TIMERS];

    // actual number of timers in use
    int numTimers;
};

#endif
```

SimpleTimer.cpp

```
    #include "SimpleTimer.h"

    // Select time function:
    //static inline unsigned long elapsed() { return micros(); }
    static inline unsigned long elapsed() { return millis(); }


    SimpleTimer::SimpleTimer() {
        unsigned long current_millis = elapsed();

        for (int i = 0; i < MAX_TIMERS; i++) {
            enabled[i] = false;
            callbacks[i] = 0;
// if the callback pointer is zero, the slot is free, i.e. doesn't
"contain" any timer
            prev_millis[i] = current_millis;
            numRuns[i] = 0;
        }

        numTimers = 0;
    }


    void SimpleTimer::run() {
        int i;
        unsigned long current_millis;

        // get current time
        current_millis = elapsed();

        for (i = 0; i < MAX_TIMERS; i++) {

            toBeCalled[i] = DEFCALL_DONTRUN;

            // no callback == no timer, i.e. jump over empty slots
            if (callbacks[i]) {

                // is it time to process this timer ?


                if (current_millis - prev_millis[i] >= delays[i]) {

                    // update time
                    //prev_millis[i] = current_millis;
                    prev_millis[i] += delays[i];



                  // check if the timer callback has to be executed
                    if (enabled[i]) {

                  // "run forever" timers must always be executed
                        if (maxNumRuns[i] == RUN_FOREVER) {
                            toBeCalled[i] = DEFCALL_RUNONLY;
                        }
                        else if (numRuns[i] < maxNumRuns[i]) {
                            toBeCalled[i] = DEFCALL_RUNONLY;
                            numRuns[i]++;
                            // after the last run, delete the timer
                            if (numRuns[i] >= maxNumRuns[i]) {
                                toBeCalled[i] = DEFCALL_RUNANDDEL;
                            }
                        }
                    }
                }
            }
        }

        for (i = 0; i < MAX_TIMERS; i++) {
            switch(toBeCalled[i]) {
                case DEFCALL_DONTRUN:
                    break;

                case DEFCALL_RUNONLY:
                    (*callbacks[i])();
                    break;

                case DEFCALL_RUNANDDEL:
                    (*callbacks[i])();
                    deleteTimer(i);
                    break;
            }
        }
    }

    // find the first available slot
    // return -1 if none found
    int SimpleTimer::findFirstFreeSlot() {
        int i;

        // all slots are used
        if (numTimers >= MAX_TIMERS) {
            return -1;
        }

        // return the first slot with no callback (i.e. free)
        for (i = 0; i < MAX_TIMERS; i++) {
            if (callbacks[i] == 0) {
                return i;
            }
        }

        // no free slots found
        return -1;
    }

    int SimpleTimer::setTimer(long d, timer_callback f, int n) {
        int freeTimer;

        freeTimer = findFirstFreeSlot();
        if (freeTimer < 0) {
            return -1;
        }

        if (f == NULL) {
            return -1;
        }
        delays[freeTimer] = d;
        callbacks[freeTimer] = f;
        maxNumRuns[freeTimer] = n;
        enabled[freeTimer] = true;
        prev_millis[freeTimer] = elapsed();

        numTimers++;

        return freeTimer;
    }


    int SimpleTimer::setInterval(long d, timer_callback f) {
        return setTimer(d, f, RUN_FOREVER);
    }


    int SimpleTimer::setTimeout(long d, timer_callback f) {
        return setTimer(d, f, RUN_ONCE);
    }


    void SimpleTimer::deleteTimer(int timerId) {
        if (timerId >= MAX_TIMERS) {
            return;
        }

        // nothing to delete if no timers are in use
        if (numTimers == 0) {
            return;
        }

        // don't decrease the number of timers if the
        // specified slot is already empty
        if (callbacks[timerId] != NULL) {
            callbacks[timerId] = 0;
            enabled[timerId] = false;
            toBeCalled[timerId] = DEFCALL_DONTRUN;
            delays[timerId] = 0;
            numRuns[timerId] = 0;

            // update number of timers
            numTimers--;
        }
    }


    // function contributed by code@rowansimms.com
    void SimpleTimer::restartTimer(int numTimer) {
        if (numTimer >= MAX_TIMERS) {
            return;
        }

        prev_millis[numTimer] = elapsed();
    }


    boolean SimpleTimer::isEnabled(int numTimer) {
        if (numTimer >= MAX_TIMERS) {
            return false;
        }
        return enabled[numTimer];
    }


    void SimpleTimer::enable(int numTimer) {
        if (numTimer >= MAX_TIMERS) {
            return;
        }

        enabled[numTimer] = true;
    }


    void SimpleTimer::disable(int numTimer) {
        if (numTimer >= MAX_TIMERS) {
            return;
        }

        enabled[numTimer] = false;
    }


    void SimpleTimer::toggle(int numTimer) {
        if (numTimer >= MAX_TIMERS) {
            return;
        }

        enabled[numTimer] = !enabled[numTimer];
    }


    int SimpleTimer::getNumTimers() {
        return numTimers;
    }
```

Appendix 5. Source code for ultrasonic sensor (C++)

NewPing.h

```
// Created by Tim Eckel - teckel@leethost.com
// Copyright 2015 License: GNU GPL v3
// http://www.gnu.org/licenses/gpl.html
// -----------------------------------------------------------------

#ifndef NewPing_h
#define NewPing_h

#if defined (ARDUINO) && ARDUINO >= 100
    #include <Arduino.h>
#else
    #include <WProgram.h>
    #include <pins_arduino.h>
#endif

#if defined (__AVR__)
    #include <avr/io.h>
    #include <avr/interrupt.h>
#endif

#define US_ROUNDTRIP_CM 57
// Microseconds (uS) it takes sound to travel round-trip 1cm
#define ONE_PIN_ENABLED true
#define ROUNDING_ENABLED false
// Set to "true" to enable distance rounding which also adds 64
bytes to binary size
#define URM37_ENABLED false
// Set to "true" to enable support for the URM37 sensor in PWM mode.
Default=false
#define TIMER_ENABLED true
// Set to "false" to disable the timer ISR (if getting "__vector_7"
compile errors set this to false). Default=true

#define NO_ECHO 0
// Value returned if there's no ping echo within the specified
MAX_SENSOR_DISTANCE or max_cm_distance. Default=0
#define MAX_SENSOR_DELAY 5800
// Maximum uS it takes for sensor to start the ping. Default=5800
#define ECHO_TIMER_FREQ 24
// Frequency to check for a ping echo (every 24uS is about 0.4cm
accuracy). Default=24
#define PING_MEDIAN_DELAY 29000
// Microsecond delay between pings in the ping_median method.
Default=29000
#define PING_OVERHEAD 5
// Ping overhead in microseconds (uS). Default=5
#define PING_TIMER_OVERHEAD 13
// Ping timer overhead in microseconds (uS). Default=13
#if URM37_ENABLED == true
    #undef  US_ROUNDTRIP_CM
    #undef  US_ROUNDTRIP_IN
    #define US_ROUNDTRIP_CM 50
// Every 50uS PWM signal is low indicates 1cm distance. Default=50
    #define US_ROUNDTRIP_IN 127
    // If 50uS is 1cm, 1 inch would be 127uS (50 x 2.54 = 127).
Default=127
```

```
#endif

// Detect non-AVR microcontrollers (Teensy 3.x, Arduino DUE, etc.)
and don't use port registers or timer interrupts as required.
#if (defined (__arm__) && defined (TEENSYDUINO))
    #undef  PING_OVERHEAD
    #define PING_OVERHEAD 1
    #undef  PING_TIMER_OVERHEAD
    #define PING_TIMER_OVERHEAD 1
#elif !defined (__AVR__)
    #undef  PING_OVERHEAD
    #define PING_OVERHEAD 1
    #undef  PING_TIMER_OVERHEAD
    #define PING_TIMER_OVERHEAD 1
    #undef  TIMER_ENABLED
    #define TIMER_ENABLED false
#endif

// Disable the timer interrupts when using ATmega128 and all ATtiny
microcontrollers.
#if defined (__AVR_ATmega128__) || defined(__AVR_ATtiny24__) ||
defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__) ||
defined(__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) ||
defined(__AVR_ATtiny85__) || defined(__AVR_ATtiny261__) ||
defined(__AVR_ATtiny461__) || defined(__AVR_ATtiny861__) ||
defined(__AVR_ATtiny43U__)
    #undef  TIMER_ENABLED
    #define TIMER_ENABLED false
#endif

// Define timers when using ATmega8 microcontrollers.
#if defined (__AVR_ATmega8__)
    #define OCR2A OCR2
    #define TIMSK2 TIMSK
    #define OCIE2A OCIE2
#endif

class NewPing {
    public:
        NewPing(uint8_t trigger_pin, uint8_t echo_pin, unsigned int
max_cm_distance = MAX_SENSOR_DISTANCE);
        unsigned int ping();
        unsigned long ping_cm();
        unsigned long ping_in();
        unsigned long ping_median(uint8_t it = 5);
        unsigned int convert_cm(unsigned int echoTime);
        unsigned int convert_in(unsigned int echoTime);
#if TIMER_ENABLED == true
        void ping_timer(void (*userFunc)(void));
        boolean check_timer();
        unsigned long ping_result;
        static void timer_us(unsigned int frequency, void
(*userFunc)(void));
        static void timer_ms(unsigned long frequency, void
(*userFunc)(void));
        static void timer_stop();
#endif
    private:
        boolean ping_trigger();
#if TIMER_ENABLED == true
        boolean ping_trigger_timer(unsigned int trigger_delay);
        boolean ping_wait_timer();
        static void timer_setup();
        static void timer_ms_cntdwn();
#endif
        uint8_t _triggerBit;
        uint8_t _echoBit;
        volatile uint8_t *_triggerOutput;
        volatile uint8_t *_echoInput;
        volatile uint8_t *_triggerMode;
        unsigned int _maxEchoTime;
        unsigned long _max_time;
};


#endif
```

NewPing.cpp

```
#include "NewPing.h"

NewPing::NewPing(uint8_t trigger_pin, uint8_t echo_pin, unsigned int
max_cm_distance) {
    _triggerBit = digitalPinToBitMask(trigger_pin);
    // Get the port register bitmask for the trigger pin.
    _echoBit = digitalPinToBitMask(echo_pin);
    // Get the port register bitmask for the echo pin.

    _triggerOutput =
portOutputRegister(digitalPinToPort(trigger_pin));
    // Get the output port register for the trigger pin.
    _echoInput = portInputRegister(digitalPinToPort(echo_pin));
    // Get the input port register for the echo pin.

    _triggerMode = (uint8_t *)
portModeRegister(digitalPinToPort(trigger_pin)); // Get the port
mode register for the trigger pin.

#if ROUNDING_ENABLED == false
    _maxEchoTime = min(max_cm_distance + 1, (unsigned int)
MAX_SENSOR_DISTANCE + 1) * US_ROUNDTRIP_CM; // Calculate the maximum
distance in uS (no rounding).
#else
    _maxEchoTime = min(max_cm_distance, (unsigned int)
MAX_SENSOR_DISTANCE) * US_ROUNDTRIP_CM + (US_ROUNDTRIP_CM / 2); //
Calculate the maximum distance in uS.
#endif

#if defined (__arm__) && defined (TEENSYDUINO)
    pinMode(echo_pin, INPUT);
    pinMode(trigger_pin, OUTPUT);
#endif

#if defined (ARDUINO_AVR_YUN)
    pinMode(echo_pin, INPUT);
#endif

#if ONE_PIN_ENABLED != true
    *_triggerMode |= _triggerBit;
#endif
}
```

```
unsigned int NewPing::ping() {
    if (!ping_trigger()) return NO_ECHO; // Trigger a ping, if it
returns false, return NO_ECHO to the calling function.

#if URM37_ENABLED == true
    while (!(*_echoInput & _echoBit))   // Wait for the ping echo.
        if (micros() > _max_time) return NO_ECHO;   // Stop the loop
and return NO_ECHO (false) if we're beyond the set maximum distance.
#else
    while (*_echoInput & _echoBit) // Wait for the ping echo.
        if (micros() > _max_time) return NO_ECHO;   // Stop the loop
and return NO_ECHO (false) if we're beyond the set maximum distance.
#endif

    return (micros() - (_max_time - _maxEchoTime) - PING_OVERHEAD);
// Calculate ping time, include overhead.
}

unsigned long NewPing::ping_cm() {
    unsigned long echoTime = NewPing::ping();          // Calls the
ping method and returns with the ping echo distance in uS.
#if ROUNDING_ENABLED == false
    return (echoTime / US_ROUNDTRIP_CM);               // Call the
ping method and returns the distance in centimeters (no rounding).
#else
    return NewPingConvert(echoTime, US_ROUNDTRIP_CM); // Convert uS
to centimeters.
#endif
}

unsigned long NewPing::ping_in() {
    unsigned long echoTime = NewPing::ping();          // Calls the
ping method and returns with the ping echo distance in uS.
#if ROUNDING_ENABLED == false
    return (echoTime / US_ROUNDTRIP_IN);               // Call the
ping method and returns the distance in inches (no rounding).
#else
    return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS
to inches.
#endif
}

unsigned long NewPing::ping_median(uint8_t it) {
    unsigned int uS[it], last;
    uint8_t j, i = 0;
    unsigned long t;
    uS[0] = NO_ECHO;

    while (i < it) {
        t = micros();  // Start ping timestamp.
        last = ping(); // Send ping.

        if (last != NO_ECHO) {
            if (i > 0)
                for (j = i; j > 0 && uS[j - 1] < last; j--)
                  // Insertion sort loop.
                    uS[j] = uS[j - 1
            } else j = 0;
            uS[j] = last;// Add last ping to array in position.
            i++;          // Move to next ping.
        } else it--;

        if (i < it && micros() - t < PING_MEDIAN_DELAY)
            delay((PING_MEDIAN_DELAY + t - micros()) / 1000); //
Millisecond delay between pings.

    }
    return (uS[it >> 1]); // Return the ping distance median.
}

boolean NewPing::ping_trigger() {
#if ONE_PIN_ENABLED == true
    *_triggerMode |= _triggerBit; // Set trigger pin to output.
#endif

    *_triggerOutput &= ~_triggerBit; // Set the trigger pin low,
should already be low, but this will make sure it is.
    delayMicroseconds(4);            // Wait for pin to go low.
    *_triggerOutput |= _triggerBit;  // Set trigger pin high, this
tells the sensor to send out a ping.
    delayMicroseconds(10);           // Wait long enough for the
sensor to realize the trigger pin is high. Sensor specs say to wait
10uS.
    *_triggerOutput &= ~_triggerBit; // Set trigger pin back to low.

#if ONE_PIN_ENABLED == true
    *_triggerMode &= ~_triggerBit; // Set trigger pin to input (when
using one Arduino pin this is technically setting the echo pin to
input as both are tied to the same Arduino pin).
#endif

#if URM37_ENABLED == true
    if (!(*_echoInput & _echoBit)) return false;            //
Previous ping hasn't finished, abort.
    _max_time = micros() + _maxEchoTime + MAX_SENSOR_DELAY; //
Maximum time we'll wait for ping to start (most sensors are <450uS,
the SRF06 can take up to 34,300uS!)
    while (*_echoInput & _echoBit)                          // Wait
for ping to start.
        if (micros() > _max_time) return false;              //
Took too long to start, abort.
#else
    if (*_echoInput & _echoBit) return false;               //
Previous ping hasn't finished, abort.
    _max_time = micros() + _maxEchoTime + MAX_SENSOR_DELAY; //
Maximum time we'll wait for ping to start (most sensors are <450uS,
the SRF06 can take up to 34,300uS!)
    while (!(*_echoInput & _echoBit))                       // Wait
for ping to start.
        if (micros() > _max_time) return false;              //
Took too long to start, abort.
#endif

    _max_time = micros() + _maxEchoTime; // Ping started, set the
time-out.
    return true;                         // Ping started
successfully.
}

#if TIMER_ENABLED == true
```

```
void NewPing::ping_timer(void (*userFunc)(void)) {
    if (!ping_trigger()) return;         // Trigger a ping, if it
returns false, return without starting the echo timer.
    timer_us(ECHO_TIMER_FREQ, userFunc); // Set ping echo timer
check every ECHO_TIMER_FREQ uS.
}


boolean NewPing::check_timer() {
    if (micros() > _max_time) { // Outside the time-out limit.
        timer_stop();             // Disable timer interrupt
        return false;             // Cancel ping timer.
    }

#if URM37_ENABLED == false
    if (!(*_echoInput & _echoBit)) { // Ping echo received.
#else
    if (*_echoInput & _echoBit) {    // Ping echo received.
#endif
        timer_stop();                  // Disable timer interrupt
        ping_result = (micros() - (_max_time - _maxEchoTime) -
PING_TIMER_OVERHEAD); // Calculate ping time including overhead.
        return true;                   // Return ping echo true.
    }

    return false; // Return false because there's no ping echo yet.
}


// Variables used for timer functions
void (*intFunc)();
void (*intFunc2)();
unsigned long _ms_cnt_reset;
volatile unsigned long _ms_cnt;
#if defined(__arm__) && defined(TEENSYDUINO)
    IntervalTimer itimer;
#endif


void NewPing::timer_us(unsigned int frequency, void
(*userFunc)(void)) {
    intFunc = userFunc; // User's function to call when there's a
timer event.
    timer_setup();      // Configure the timer interrupt.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4
(Teensy/Leonardo).
    OCR4C = min((frequency>>2) - 1, 255); // Every count is 4uS, so
divide by 4 (bitwise shift right 2) subtract one, then make sure we
don't go over 255 limit.
    TIMSK4 = (1<<TOIE4);                     // Enable Timer4 interrupt.
#elif defined (__arm__) && defined (TEENSYDUINO) // Timer for Teensy
3.x
    itimer.begin(userFunc, frequency);             // Really simple
on the Teensy 3.x, calls userFunc every 'frequency' uS.
#else
    OCR2A = min((frequency>>2) - 1, 255); // Every count is 4uS, so
divide by 4 (bitwise shift right 2) subtract one, then make sure we
don't go over 255 limit.
    TIMSK2 |= (1<<OCIE2A);                   // Enable Timer2 interrupt.
#endif
```

```
}


void NewPing::timer_ms(unsigned long frequency, void
(*userFunc)(void)) {
    intFunc = NewPing::timer_ms_cntdwn;  // Timer events are sent
here once every ms till user's frequency is reached.
    intFunc2 = userFunc;                 // User's function to call
when user's frequency is reached.
    _ms_cnt = _ms_cnt_reset = frequency; // Current ms counter and
reset value.
    timer_setup();                       // Configure the timer
interrupt.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4
(Teensy/Leonardo).
    OCR4C = 249;           // Every count is 4uS, so 1ms = 250
counts - 1.
    TIMSK4 = (1<<TOIE4);   // Enable Timer4 interrupt.
#elif defined (__arm__) && defined (TEENSYDUINO) // Timer for Teensy
3.x
    itimer.begin(NewPing::timer_ms_cntdwn, 1000);  // Set timer to
1ms (1000 uS).
#else
    OCR2A = 249;           // Every count is 4uS, so 1ms = 250
counts - 1.
    TIMSK2 |= (1<<OCIE2A); // Enable Timer2 interrupt.
#endif
}


void NewPing::timer_stop() { // Disable timer interrupt.
#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4
(Teensy/Leonardo).
    TIMSK4 = 0;
#elif defined (__arm__) && defined (TEENSYDUINO) // Timer for Teensy
3.x
    itimer.end();
#else
    TIMSK2 &= ~(1<<OCIE2A);
#endif
}


void NewPing::timer_setup() {
#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4
(Teensy/Leonardo).
    timer_stop(); // Disable Timer4 interrupt.
    TCCR4A = TCCR4C = TCCR4D = TCCR4E = 0;
    TCCR4B = (1<<CS42) | (1<<CS41) | (1<<CS40) | (1<<PSR4); // Set
Timer4 prescaler to 64 (4uS/count, 4uS-1020uS range).
    TIFR4 = (1<<TOV4);
    TCNT4 = 0;    // Reset Timer4 counter.
#elif defined (__AVR_ATmega8__)
    timer_stop();                     // Disable Timer2 interrupt.
    ASSR &= ~(1<<AS2);                // Set clock, not pin.
    TCCR2 = (1<<WGM21 | 1<<CS22); // Set Timer2 to CTC mode,
prescaler to 64 (4uS/count, 4uS-1020uS range).
    TCNT2 = 0;                        // Reset Timer2 counter.
#elif defined (__arm__) && defined (TEENSYDUINO)
    timer_stop(); // Stop the timer.
```

```
#else
    timer_stop();           // Disable Timer2 interrupt.
    ASSR &= ~(1<<AS2);      // Set clock, not pin.
    TCCR2A = (1<<WGM21);  // Set Timer2 to CTC mode.
    TCCR2B = (1<<CS22);   // Set Timer2 prescaler to 64 (4uS/count,
4uS-1020uS range).
    TCNT2 = 0;              // Reset Timer2 counter.
#endif
}

void NewPing::timer_ms_cntdwn() {
    if (!_ms_cnt--) {     // Count down till we reach zero.
        intFunc2();       // Scheduled time reached, run the main
timer event function.
        _ms_cnt = _ms_cnt_reset; // Reset the ms timer.
    }
}

#if defined (__AVR_ATmega32U4__)
ISR(TIMER4_OVF_vect) {
    intFunc();
}
#elif defined (__AVR_ATmega8__) // ATmega8 microcontrollers.
ISR(TIMER2_COMP_vect) {
    intFunc();
}
#elif defined (__arm__)
#else
ISR(TIMER2_COMPA_vect) {
    intFunc();
}
#endif


#endif

// ---------------------------------------------------------------------
// Conversion methods (rounds result to nearest cm or inch).
// ---------------------------------------------------------------------

unsigned int NewPing::convert_cm(unsigned int echoTime) {
#if ROUNDING_ENABLED == false
    return (echoTime / US_ROUNDTRIP_CM);              // Convert uS
to centimeters (no rounding).
#else
    return NewPingConvert(echoTime, US_ROUNDTRIP_CM); // Convert uS
to centimeters.
#endif
}
```

Appendix 6. Source code for weight sensor (C++)

HX711.h

```
#include <Arduino.h>
#include <HX711.h>

HX711::HX711(byte dout, byte pd_sck, byte gain) {
    PD_SCK  = pd_sck;
    DOUT    = dout;

    pinMode(PD_SCK, OUTPUT);
    pinMode(DOUT, INPUT);

    set_gain(gain);
}

HX711::~HX711() {

}

bool HX711::is_ready() {
    return digitalRead(DOUT) == LOW;
}

void HX711::set_gain(byte gain) {
    switch (gain) {
        case 128:       // channel A, gain factor 128
            GAIN = 1;
            break;
        case 64:        // channel A, gain factor 64
            GAIN = 3;
            break;
        case 32:        // channel B, gain factor 32
            GAIN = 2;
            break;
    }

    digitalWrite(PD_SCK, LOW);
    read();
}

long HX711::read() {
    // wait for the chip to become ready
    while (!is_ready());

    byte data[3];

    // pulse the clock pin 24 times to read the data
    for (byte j = 3; j--;) {
        for (char i = 8; i--;) {
            digitalWrite(PD_SCK, HIGH);
            bitWrite(data[j], i, digitalRead(DOUT));
            digitalWrite(PD_SCK, LOW);
        }
    }

    // set the channel and the gain factor for the next reading
using the clock pin
    for (int i = 0; i < GAIN; i++) {
        digitalWrite(PD_SCK, HIGH);
        digitalWrite(PD_SCK, LOW);
    }

    data[2] ^= 0x80;

    return ((uint32_t) data[2] << 16) | ((uint32_t) data[1] << 8) |
(uint32_t) data[0];
}

long HX711::read_average(byte times) {
    long sum = 0;
    for (byte i = 0; i < times; i++) {
        sum += read();
    }
    return sum / times;
}

double HX711::get_value(byte times) {
    return read_average(times) - OFFSET;
}

float HX711::get_units(byte times) {
    return get_value(times) / SCALE;
}

void HX711::tare(byte times) {
    double sum = read_average(times);
    set_offset(sum);
}

void HX711::set_scale(float scale) {
    SCALE = scale;
}

void HX711::set_offset(long offset) {
    OFFSET = offset;
}

void HX711::power_down() {
    digitalWrite(PD_SCK, LOW);
    digitalWrite(PD_SCK, HIGH);
}

void HX711::power_up() {
    digitalWrite(PD_SCK, LOW);
}
```

HX711.cpp

```
#include <Arduino.h>
#include <HX711.h>

HX711::HX711(byte dout, byte pd_sck, byte gain) {
    PD_SCK  = pd_sck;
    DOUT    = dout;

    pinMode(PD_SCK, OUTPUT);
    pinMode(DOUT, INPUT);

    set_gain(gain);
}

HX711::~HX711() {

}

bool HX711::is_ready() {
    return digitalRead(DOUT) == LOW;
}

void HX711::set_gain(byte gain) {
    switch (gain) {
        case 128:       // channel A, gain factor 128
            GAIN = 1;
            break;
        case 64:        // channel A, gain factor 64
            GAIN = 3;
            break;
        case 32:        // channel B, gain factor 32
            GAIN = 2;
            break;
    }

    digitalWrite(PD_SCK, LOW);
    read();
}

long HX711::read() {
    // wait for the chip to become ready
    while (!is_ready());

    byte data[3];

    // pulse the clock pin 24 times to read the data
    for (byte j = 3; j--;) {
        for (char i = 8; i--;) {
            digitalWrite(PD_SCK, HIGH);
            bitWrite(data[j], i, digitalRead(DOUT));
            digitalWrite(PD_SCK, LOW);
        }
    }

    // set the channel and the gain factor for the next reading
using the clock pin
    for (int i = 0; i < GAIN; i++) {
        digitalWrite(PD_SCK, HIGH);
        digitalWrite(PD_SCK, LOW);
    }

    data[2] ^= 0x80;

    return ((uint32_t) data[2] << 16) | ((uint32_t) data[1] << 8) |
(uint32_t) data[0];
}

long HX711::read_average(byte times) {
    long sum = 0;
    for (byte i = 0; i < times; i++) {
        sum += read();
    }
    return sum / times;
}

double HX711::get_value(byte times) {
    return read_average(times) - OFFSET;
}

float HX711::get_units(byte times) {
    return get_value(times) / SCALE;
}

void HX711::tare(byte times) {
    double sum = read_average(times);
    set_offset(sum);
}

void HX711::set_scale(float scale) {
    SCALE = scale;
}

void HX711::set_offset(long offset) {
    OFFSET = offset;
}

void HX711::power_down() {
    digitalWrite(PD_SCK, LOW);
    digitalWrite(PD_SCK, HIGH);
}

void HX711::power_up() {
    digitalWrite(PD_SCK, LOW);
}
```

Appendix 7. Source code for IR sensor (C++)

DistanceGP2Y0A21YK.h

```
/*****************************************************************
DistanceGP2Y0A21YK  -  Arduino  library  for  IR  Distance  sensor
Copyright 2011-2012 Jeroen Doggen (jeroendoggen@gmail.com)
*****************************************************************
#ifndef DistanceGP2Y0A21YK_h
#define DistanceGP2Y0A21YK_h
#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
  #include <pins_arduino.h>
#endif

class DistanceGP2Y0A21YK
{
    public:
        DistanceGP2Y0A21YK();
        void begin();
        void begin(int distancePin);
        void begin(int distancePin, int vccPin);
        int getDistanceRaw();
        int getDistanceVolt();
        int getDistanceCentimeter();

        boolean isCloser(int threshold);
        boolean isFarther(int threshold);
        void setAveraging(int avg);
        void setARefVoltage(int refV);
        void setEnabled(bool status);

    private:
        int _mapGP2Y0A21YK_V(int value);
        int _mapGP2Y0A21YK_CM(int value);
        int _distancePin;
        int _average;
        int _transferFunctionLUT3V[];
        int _transferFunctionLUT5V[];
        int _refVoltage;
        int _vccPin;
        bool _enabled;
};
#endif
```

DistanceGP2Y0A21YK.cpp

```
#include <DistanceGP2Y0A21YK.h>
#include <DistanceGP2Y0A21YK_LUTs.h>

DistanceGP2Y0A21YK::DistanceGP2Y0A21YK()
{
}

/// Begin function to set pins: distancePin = A0.
void DistanceGP2Y0A21YK::begin()
{
    begin (A0);
}

/// Begin variables
/// - int _distancePin: number indicating the distance to an object:
ANALOG IN
/// When you use begin() without variables standard values are
loaded: A0

void DistanceGP2Y0A21YK::begin(int distancePin)
{
  setEnabled(true);
  pinMode(distancePin, INPUT);
    _distancePin=distancePin;
    setAveraging(1);              //1: all samples passed to higher
level
    setARefVoltage(5);            // 5: default analog reference of
5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)
        //setARefVoltage(3);  // external analog reference: for 3.3V:
put a wire between the AREF pin and the 3.3V VCC pin.
        //This increases accuracy (and uses a different LUT)
}

/// Begin variables
/// - int _distancePin: number indicating the distance to an object:
ANALOG IN
/// - int vccPin: pin connected to the vcc pin of the sensor (switch
the sensor on/off)
/// When you use begin() without variables standard values are
loaded: A0
void DistanceGP2Y0A21YK::begin(int distancePin, int vccPin)
{
    _vccPin=vccPin;
    begin (distancePin);
    pinMode(_vccPin, OUTPUT);
    setEnabled(true);
    //digitalWrite(vccPin, HIGH);
}

/// setAveraging(int avg): Sets how many samples have to be averaged
in getDistanceCentimeter, default value is 1.
```

```
void DistanceGP2Y0A21YK::setAveraging(int avg)
{
    _average=avg;
}

/// getDistanceRaw(): Returns the distance as a raw value: ADC
output: 0 -> 1023
int DistanceGP2Y0A21YK::getDistanceRaw()
{
    if (_enabled == true)
    {
        return (analogRead(_distancePin));
    }
    else
    {
        return (1023);
    }

}

/// getDistanceVolt(): Returns the distance as a Voltage: ADC Input:
0V -> 5V (or 0V -> 3.3V)
int DistanceGP2Y0A21YK::getDistanceVolt()
{
    return _mapGP2Y0A21YK_V(getDistanceRaw());
}

/// getDistanceCentimeter(): Returns the distance in centimeters
int DistanceGP2Y0A21YK::getDistanceCentimeter()
{
    return _mapGP2Y0A21YK_CM(getDistanceRaw());
}

/// _mapGP2Y0A21YKV: calculates the input voltage of the ADC
int DistanceGP2Y0A21YK::_mapGP2Y0A21YK_V(int value)
{
    if (_refVoltage == 3)
    {
        return map(value,0,1023,0,3300);
    }
    if (_refVoltage == 5)
    {
        return map(value,0,1023,0,5000);
    }
}

/// _mapGP2Y0A21YK_CM: calculates the distance in centimeters using
a lookup table
///    -> Two different LUTs depending on ADC reference voltage
int DistanceGP2Y0A21YK::_mapGP2Y0A21YK_CM(int value)
{
    if (_refVoltage == 3)
    {
        int sum = 0;
        for (int i=0;i<_average;i++)
        {
            // this code is equivalent to:
sum=sum+transferFunctionLUT5V[(getDistanceRaw()/4)];
            // but this alternative syntax is needed to read the
program memory
```

```
            sum=sum + pgm_read_byte_near (transferFunctionLUT3V +
(getDistanceRaw()/4));
        }
        return(sum/_average);
    }
    if (_refVoltage == 5)
    {
        int sum = 0;
        for (int i=0;i<_average;i++)
        {
            sum=sum + pgm_read_byte_near (transferFunctionLUT5V +
(getDistanceRaw()/4));
        }
        return(sum/_average);
    }
}

/// setARefVoltage:set the ADC reference voltage: (default value: 5V,
set to 3 for external reference value, typically 3.3 on Arduino
boards)
void DistanceGP2Y0A21YK::setARefVoltage(int refV)
{
    if (_refVoltage == 5)
    {
        analogReference(DEFAULT);
    }
    if (_refVoltage == 3)
    {
        analogReference(EXTERNAL);
    }
    _refVoltage=refV;
}

/// isCloser: check whether the distance to the detected object is
smaller than a given threshold
boolean DistanceGP2Y0A21YK::isCloser(int threshold)
{
    if (threshold>getDistanceCentimeter())
    {
        return (true);
    }
    else
    {
        return (false);
    }
}

/// <summary>
/// isFarther: check whether the distance to the detected object is
smaller than a given threshold
/// </summary>
boolean DistanceGP2Y0A21YK::isFarther(int threshold)
{
    if (threshold<getDistanceCentimeter())
    {
        return true;
    }
    else
    {
        return false;
    }
```

```
}

/// setEnabled: enable or disable the sensor (only works when sensor
Vcc is connected to digital output pin)
void DistanceGP2Y0A21YK::setEnabled(bool status)
{
    _enabled = status;
    if (_enabled)
    {
        digitalWrite(_vccPin, HIGH);
    }
    else
    {
        digitalWrite(_vccPin, LOW);
    }
}
```

Appendix 8. Source code in Spreadsheets (Java Scripts)

```
function TimeStamp(event)
{
  var timezone = "GMT";
  var timestamp_format = "HH:mm:ss, [MM-dd-yyyy]"; // Format.
  var updateColName = "Photo Interuptor";
  var timeStampColName = "DateStamp";
  var sheet = event.source.getSheetByName('Datalog');


  var actRng = event.source.getActiveRange();
  var editColumn = actRng.getColumn();
  var index = actRng.getRowIndex();
  var headers = sheet.getRange(1, 1, 1,
sheet.getLastColumn()).getValues();
  var dateCol = headers[0].indexOf(timeStampColName);
  var updateCol = headers[0].indexOf(updateColName); updateCol =
updateCol+1;
  if (dateCol > -1 && index > 1 && editColumn == updateCol) {
// only timestamp if 'Last Updated' header exists, but not in the
header row itself!
    var cell = sheet.getRange(index, dateCol + 1);
    var date = Utilities.formatDate(new Date(), timezone,
timestamp_format);
    cell.setValue(date);
  }
}


function GetData() {
  var spreadsheetKey= '_SHEETKEY';
  var sheet = SpreadsheetApp.openById(spreadsheetKey);
  var ssAutomatedAlert =
SpreadsheetApp.openById(spreadsheetKey).getSheetByName('AutomatedAle
rt');
  var ss = sheet.getSheetByName('Contract');
  var value = ss.getDataRange().getValues();

  var startRow = 2;
  var numRow = ssAutomatedAlert.getLastRow()-1;

 var range = ssAutomatedAlert.getRange(startRow, 1, numRow, 3);
    range.clear()

  //define day of week and time in number:[dd][hh] (day~time[hour])
  // Mon = 1, ...., Sun = 7
  var week_and_time = getTodaysNumber().split('~');

  // define week column
  var weekDay = week_and_time[0];
  var dayColumn = weekDay*3;

  // define time column
  var weekTime = week_and_time[1];
  var timeColumn = 0;
  if(weekTime >= 8 && weekTime <= 16){  //
    timeColumn = dayColumn+0;
  }else if(weekTime > 16 && weekTime <= 24){
    timeColumn = dayColumn+1;
  }else if(weekTime < 8){
    timeColumn = dayColumn+2;
  }

  // Loop inside each Column to display only "Yes" column in each
day (Mon-Sun)
  for (var i=1; i < value.length; i++ ){
    var count = value[i];
    if(count[timeColumn] == "Yes"){
      var responder = count[0];
      var email = count[1];
      var telephone = count[2];
      ssAutomatedAlert.appendRow([responder,email,telephone]);
    }
  }
}

function lastNumber(){
  var spreadsheetKey= '_SHEETKEY';
  var sheet = SpreadsheetApp.openById(spreadsheetKey);
  var startRow = 3;
  var ssDatalog = sheet.getSheetByName('Datalog');
  var numRow = ssDatalog.getLastRow()-1;
  var position = 8;
  var sum = 0;
  //var range = ssDatalog.getRange(startRow, 1, numRow, 3);
  for(var n = startRow; n < numRow; n++){
    var range = ssDatalog.getRange(n, 2, 6);
    var values = range.getValues();

    for (var row in values) {
      for (var col in values[row]) {
        sum += values[row][col];
        //ssDatalog.getRange(row, 12).setValue(row + '-' + col);
      }
    }

    ssDatalog.getRange('L' + position).setValue(sum);
    sum = 0;
    position+=1;

  }

}

function checkContianer() {
  var spreadsheetKey= '1NlSLE74DxEAPPcdh19S5mzwukzNdsTcusy-khlaKynI';
  var sheet = SpreadsheetApp.openById(spreadsheetKey);
  var ssAutomatedAlert =
SpreadsheetApp.openById(spreadsheetKey).getSheetByName('AutomatedAle
rt');
  var ssDatalog = sheet.getSheetByName('Datalog');
  var startRow = 2;  // initial row of data to process
  var numRowCheck = ssDatalog.getLastRow()-1;   // Number of rows to
process not include header
  var numRowsMail = ssAutomatedAlert.getLastRow()-1;
  // Fetch the range of cells from validated data
  var dataRange = ssDatalog.getRange(startRow, 8, numRowCheck, 3);
  // Fetch the range of cells from list of responders
  var mailRange = ssAutomatedAlert.getRange(startRow, 1, numRowsMail,
2)
  var data = dataRange.getValues();

  var limit = 1800; //maximum of therapeutic goals

  for (var n = 0; n < data.length; ++n) {
    var row = data[n];
    var volume = row[0];
    var EMAIL_SENT_RANGE = ssDatalog.getRange(startRow, 10);
    var emailsent = row[2];
    var EMAIL_SENT = "EMAIL_SENT";

    if (volume >= limit) {
      if (emailsent !== EMAIL_SENT) {

        var mail = mailRange.getValues();
        for (i in mail) {
          var list = mail[i];
          var emailAddress = list[1];  // First column

          var html =
                '<body>' +
                  '<h2> To '+list[0]+',</h2><br />' +
                    '<p> <h3> This is email sent from automated
function for reminding you to empty the urine container  <br>'+
                      'Now the urine collection is equal (or more than)
'+limit+' cc.</p>'+
                        'Regards,</h5><br><br>' +
                          'Atigorn'+
                            '</body>'

          var message = html;
          var subject = "Sending emails from a Spreadsheet";
          MailApp.sendEmail(emailAddress, subject,
message ,{htmlBody: html});
          ssDatalog.getRange(startRow + n, 10).setValue(EMAIL_SENT);
        }
      }
    }
  }
}
```