# AUTONOMOUS SMART ROBOT SURVEILLANCE SYSTEM (ASRSS)

GUAT SI JIE

Bachelor of Computer Science

UNIVERSITI MALAYSIA PAHANG

**UNIVERSITI MALAYSIA PAHANG**

**DECLARATION OF THESIS AND COPYRIGHT**

Author's Full Name : _____

Date of Birth : _____

Title : _____

_____

_____

Academic Session : _____

I declare that this thesis is classified as:

| | |
|---|---|
| ☐ CONFIDENTIAL | (Contains confidential information under the Official Secret Act 1997)* |
| ☐ RESTRICTED | (Contains restricted information as specified by the organization where research was done)* |
| ☑ OPEN ACCESS | I agree that my thesis to be published as online open access (Full Text) |

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

| | |
|---|---|
| _____ | _____ |
| (Student's Signature) | (Supervisor's Signature) |
| _____ | _____ |
| New IC/Passport Number | Name of Supervisor |
| Date: | Date: |

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

# THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
*Perpustakaan Universiti Malaysia Pahang*,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Guat Si Jie
Autonomous Smart Robot Surveillance System

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

**SUPERVISOR'S DECLARATION**

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Doctor of Philosophy/ Master of Engineering/ Master of Science in ……..

_____

(Supervisor's Signature)

Full Name :

Position :

Date :

_____

(Co-supervisor's Signature)

Full Name :

Position :

Date :

## STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : GUAT SI JIE

ID Number : 950423075557

Date : 12 December 2018

# AUTONOMOUS SMART ROBOT SURVEILLANCE SYSTEM (ASRSS)

GUAT SI JIE

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

December 2018

## ACKNOWLEDGEMENTS

I would first like to thank my supervisor Dr. Noraziah Ahmad of the FSKKP at University Malaysia Pahang. Dr. Noraziah Ahmad mentored me during the whole process of developing the project. She consistently guides me with care and wisdom and gave me advice to improve my project.

I also would like to appreciate the critics and suggestion by the lecturers who is involved in judging my project. Without their passionate participation and input, the presentation of project could not have been successfully conducted.

Finally, I must express my big thanks to my parents for providing me support and love throughout my years of studying in UMP and the period of writing this thesis. This achievement would not have been possible without them. Thank you.

# ABSTRAK

Tesis ini menerangkan sistem pengawasan robot pintar autonomi yang bertindak sebagai sistem pengawasan rumah. Robot pengawasan secara automatik boleh bergerak dari sukan tinjauan ke yang lain dan merakamkan serta menghantar pemberitahuan kepada pengguna jika ada objek yang bergerak yang direkam. Ia akan memuat naik video ke Google Drive. Ia boleh mengelakkan objek bergerak jika ia bergerak ke arah robot.

## ABSTRACT

This thesis describes an autonomous smart robot surveillance system which acts as a home surveillance system. The surveillance robot can automatically move from a surveilling sport to another and record plus send a notification to the user if there is a moving object recorded. It will upload the video to Google Drive. It can avoid moving object if it is moving in the direction of the robot.

# TABLE OF CONTENT

## LIST OF TABLES

## LIST OF FIGURES

```
21      def setup(self):
22          GPIO.setwarnings(False)
23          GPIO.setmode(GPIO.BCM)  # Numbers GPIOs by physical location
24          GPIO.setup(self.Gpin, GPIO.OUT)  # Set Green Led Pin mode to output
25          GPIO.setup(self.Rpin, GPIO.OUT)  # Set Red Led Pin mode to output
26          GPIO.setup(self.BTMSensorMid, GPIO.IN)  # Set BtnPin's mode is input, and pull up to high level(3.3V)
27          GPIO.setup(self.BTMSensorRight, GPIO.IN)
28          GPIO.setup(self.BTMSensorLeft, GPIO.IN)
29          GPIO.setup(self.BackSensorRight, GPIO.IN)
30          GPIO.setup(self.BackSensorLeft, GPIO.IN)
31
32      def LBTMInfra(self):
33          return GPIO.input(self.BTMSensorLeft)
34
35      def RBTMInfra(self):
36          return GPIO.input(self.BTMSensorRight)
37
38      def MBTMInfra(self):
39          return GPIO.input(self.BTMSensorMid)
40
41      def LBackInfra(self):
42          return GPIO.input(self.BackSensorLeft)
43
44      def RBackInfra(self):
45          return GPIO.input(self.BackSensorRight)
46
```

## LIST OF SYMBOLS

| | |
|---|---|
| SBPWM | Simple Boost Pulse Width Modulation |
| ZSI | Z source inverter |

## LIST OF ABBREVIATIONS

| | |
|---|---|
| SBPWM | Simple Boost Pulse Width Modulation |
| ZSI | Z source inverter |

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

As we are getting closer to Industrial Revolution 4.0, we are blessed with capable IoT technology that empowers our lives.(Moore, 2018) Soft computing and artificial intelligence are successfully applied to our everyday life, such as machinery control, robot manipulation and engineering application. With the integration of nonlinear system and communication technologies, creating a secure environment for our home is a non-issue now.(Brandon, 2014)

Home safety is one of the most critical issues as we tend to keep our valuable assets at home. The most convenient solution to secure our home is to use Closed Circuit TV (CCTV) technology that films a fixed area at fixed angle all the time. CCTV holds a record of film which is about what was happening in the area for the users. However, there is limitations for CCTV. Because that it only surveillance a fixed area, it is lacking flexibility to film a particular event ongoing. The filmed video can only cover a footage which is filmed at the fixed area.

A robot with sensors for detecting environment and motor for moving around surveillance area which is programmed to watch over an area is a promising solution as a conventional home surveillance system. To make surveillance job possible, ultrasonic sensors, a camera and a navigating algorithm is implemented. To enhance the ability of the robot to record important event, computer vision algorithm is implemented in the system to detect movements. If the system detects a moving person at home, it will send notification to the owner of the device.

## 1.2 PROBLEM STATEMENT

Thief crimes often happen at a regular basics. It often happens when the victim is away from the house. The thieves steal away assets of the victim and causes massive amount of lost to the victim.

Regular CCTV do have the problem of mobility since that it does not have the motor and the algorithm to adjust the angle. That means that there is always blind spot for crime to be happen. Some important incidents may not be recorded since it happened in the blind spot.

Installing surveillance camera to cover the whole house requires a high cost to implement it. The cost of the setup is mainly cause by the quantity of the components that is required to setup the system.

Often, thief crimes are not reported it is found out later. Normally it may take hours if not days for the victim to realise what is happening in his house. It will cause a long delay in reporting the incident.

## 1.3 GOAL AND OBJECTIVE

i. To build a smart surveillance system that detects nearby obstacles.

ii. To build a raspberry pi system that can move from a point to another.

iii. To apply the feature of sending notification and video to a phone.

## 1.4 SCOPE

i. User

Only one user can interact with the autonomous surveillance robot system at a time. The user can access the system using the ASRSS mobile app. User must be able to operate Linux system and command windows.

ii. Data

The data of video is stored in local sd card and cloud storage. The video recorded is only encoded in H.264 format. The database is only stored in the surveillance robot, Android smartphone and Azure SQL database. The internet connection must be available to the system and port 1433 must be unblock for the communication to the Azure SQL database.

iii. System

Autonomous surveillance robot system is built on Raspberry Pi system. It will have the ability to navigate from a point to another. It can detect moving object. If moving person is detected, it will send notification to the user. The system must be supplied with a 5 Volt 2 Ampere power supply.

iv. Environment

Autonomous surveillance robot system is designed to only work in indoor environment. The indoor environment must a guiding tile to navigate it from a point to another. The path of the guiding tile must be straight and without any obstacle.

## 1.5 SIGNIFICANCE

i. Reduce the cost of surveying a large surveillance area.

ii. Reduce the risk of thief.

iii. Reduce the time to make a police report.

## 1.6 THESIS ORGANIZATION

There are five chapters in this report which are introduction, literature review, Methodology, implementation and result discussion and finally conclusion. Each chapter will discuss its own aspects related to the project.

Chapter one, the introduction of the project is discussed. It includes introduction, problem statement, objectives that need to achieve, scope of the project and thesis organization.

Chapter two, is about literature review. It describes the existing problem or solution done by other parties. This chapters explains the detail techniques/method/hardware or technologies which are suitable to be adapted into project.

Chapter three, shall discuss method, technique or approach and methodology in details. This chapter include introduction, methodology, hardware and software requirement, Gantt chart and testing plan.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 INTRODUCTION

In this chapter, we will be covering a few subtopics. It includes concept, technologies, existing systems and methodology model.

## 2.2 CONCEPT

### 2.2.1 Autonomous Surveillance Concept

The core concept of the project is to create an autonomous surveillance device. To be able to survey an area autonomously, the system needs to have a navigation system. The navigation system supports the autonomous surveillance system by providing the mobility the robot needs. The device does the survey job by capturing video via a webcam. The device will send notification to the user if it senses a person's movements. The video captured can be streamed live to mobile devices.

### 2.2.2 Navigation System Concept

The device is equipped with a navigating algorithm to navigate with the navigation tiles in the indoor area. The system can move itself by using 4 wheels controlled by motors. In order to avoid obstacles, it is attached with ultrasonic sensor. Infrared sensors are attached to the device to enable it to avoid short distance obstacles.

## 2.3 TECHNOLOGIES

### 2.3.1 Single Board Computer

Single board computer is the most important component of the project. It serves as the brain of the robot. It executes the logics which are programmed to it to serve its role. In the market there is a few single board computers which may be suitable for the project.

#### 2.3.1.1 Raspberry Pi Model 3 B+

Raspberry Pi Model 3 B+ is a powerful single board computer.(Watson, 2017) It is powerd by Boradcom BCM2837B0 as its SOC. It has a 1.4GHz quad-core ARM Cortex-A53 as its CPU and Broadcom VideoCore IV (400MHz) as its GPU. It is powered by 1 GB of SDRAM. It supports MicroSDHC by having a MicroSDHC slot. It can boot from USB drive via USB Boot Mode. For networking, it supports up to 300Mbit/s Ethernet, 802.11ac dual band 2.4/5 GHz and wireless Bluetooth 4.2 LS BLE. Raspberry Pi 3 Model B+ has 4 USB 2.0 ports HDMI (v1.3) as video output and a 3.5mm headphone jack as audio output. It only supports digital inputs.

The operating system that Raspberry Pi Foundation recommends is the Raspbian OS. Raspberry Pi 3 Model B+ does support third party operating systems whether it is Linux-based and not Linux-based. Among the non-Linux-based operating system are RISC OS Pi, Free BSD, Net BSD and Windows 10 IoT Core. The popular Linux-based operating systems includes Android things, Arch Linux ARM, openSUSE and SUSE Linux Enterprise Server 12 SP2.(Foundation, 2018)

Raspberry Pi 3 Model B+ is supported by large variety of IDEs and programming language.(Nayyar, 2017) It is supported by BlueJ, Geany Ide, Adafruit WebIDE and AlgoIDE. It supports any languages as long as the language will be compiled for ARMv6. such as Python, C, C++, Java, Scratch, Ruby, HTML5, Javascript, JQuery, Perl and Erlang.(McManus, 2018)

Raspberry Pi is a strong compute device. It is capable enough to run face recognition algorithm and other complex algorithm thanks to its powerful CPU and large RAM. It has enough IO for both camera or a external hard drive and it support WiFi

connection natively. It has enough pins to accommodate the sensors, which is a total of 6.

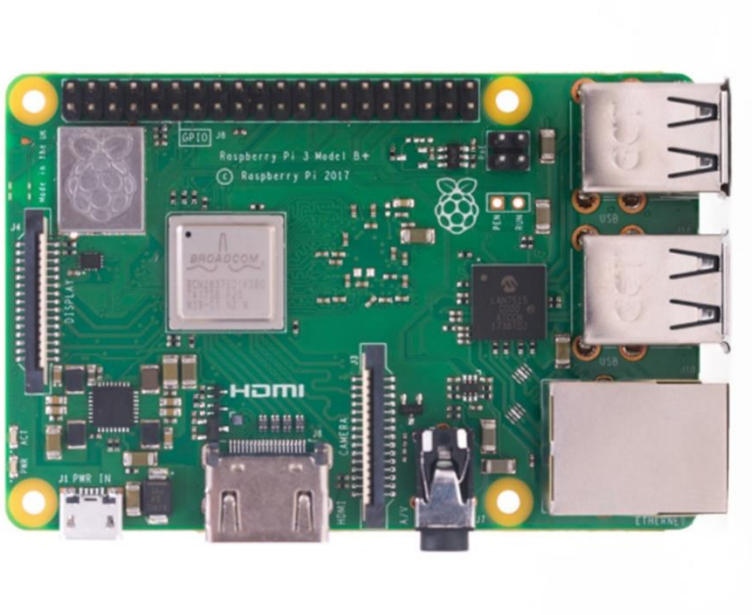



Figure 2.1 Raspberry Pi 3 Model B+

#### 2.3.1.2 Arduino Uno

Arduino Uno is a low powered system generally used for IoT devices. It is powered by 16 MHz ATmega328P processor.(Arduino, 2018a) It has 32 kB flash storage, 1 kB of EEPROM and 2 kB SRAM. It does support both digital input and analog input. It has a USB port and a dc power jack.

Arduino Uno can be programmed with any language as long as the compilers produce binary machine code for the target processor. Arduino IDE is the most popular IDE used by Arduinos. It supports languages C and C++.(Arduino, 2018b)

Arduino Uno has a very weak performance when it comes to computing. It has limitation of flash storage where you cannot programme it with a complex and long algorithm. It also lacks the USB port which can be connected to webcam or external hard drive.


Figure 2.2 Arduino Uno

#### 2.3.1.3 Beaglebone Black

Beaglebone Black is powered by AM3358/9 SoC. It has a 1 GHz Cortex-A8 and dual PRU (200MHz) as its CPU.(Beagleboard.org, 2018) The GPU that powers Beaglebone Black is PowerVR SGX530 running at 200 MHz. It has a 4GB 8-bit eMMC on-board flash storage and a 512 MB DDR3 SDRAM. For ports, the Beaglebone Black includes one standard A host port, one mini B device port and a Micro-HDMI port. It can connect to the internet via a 10/100 Ethernet port.

Beaglebone is can be coded in C, C++, Python, Perl, Ruby, Java, or shell script. It is commonly coded with Cloud9 IDE and Eclipse.(Richardson, 2012)

Beaglebone Black is a relative powerful device. It lacks enough ports for webcam and external hard drive. It has a large amount of GPIO that allows the user to connect to large quantity of sensors.

Figure 2.3 Beaglebone Black

#### 2.3.1.4 Comparison of Single Board Computer

Table 2.1 Comparison of Single Board Computer

| | Raspberry Pi 3 Model B+ | Arduino Uno | Beaglebone Black |
|---|---|---|---|
| CPU | 1.4 GHz quad core ARM Cortex-A53 | 16 MHz | 1 GHz Cortex-A8 + Dual PRU |
| GPU | Broadcom VideoCore IV 400 MHz | - | PowerVR SGX530 200 MHz |
| RAM | 1 GB | 2 KB | 512 MB DDR3 |
| Storage | MicroSDHC slot, USB Boot Mode | 32 KB | 4 GB |
| USB port | 4 | 2 | 1 |
| WiFi | 802.11ac dual band 2.4/5 GHz wireless | - | - |
| GPIO | Digital input | Analog and digital input | Analog and digital input |

### 2.3.2 Sensors

To interact with the environment, the robot needs to perceive the environment. The sensors act as the gate for the robot to learn about the situation of the world. There are a few sensors which are useful for this project.

#### 2.3.2.1 Webcam

It is used to record footage in real time. A webcam is typically made of a lens, an image sensor and supporting electronics. The lenses normally have a fixed focal length. The main function of the lenses is to allow the light to focus on the image sensor. The image sensor's job is to capture light. The sensors may be a CMOS or CDD. The supporting electronics reads the image from the sensor and transfer it as digital signal.(Layton, 2018)

The webcam works as follows:

1. Light reflected by the object enters the camera lens.
2. The image sensor of the camera splits the image of the object among the pixels of the sensor.
3. The sensor measures the colour and the brightness of each pixels.
4. The information of the colour and brightness is converted into binary numbers and transmitted via the USB cable.

This sensor is the most important sensor as it makes streaming video and other video analysing algorithm possible.


Figure 2.4 Webcam

#### 2.3.2.2 Ultrasonic Sensor

The main purpose of ultrasonic sensor is to detect the distance of an object by using ultrasonic waves. Way it measures the distance is by measuring the delay of ultrasonic waves bounce back from the object. It usually emits a 40kHz ultrasonic waves.(Academy, n.d.)

Firstly, the ultrasonic sensor sends out an ultrasonic wave. When the ultrasonic wave hits an object, it is reflected. Then the sensor measures the time lag in between the time it sends the wave and the time it receives the wave. The speed of the ultrasonic wave travels approximately 341m per second in air. With this information, the distance of the object from the sensor can be calculated by multiplying the lag time and speed of ultrasonic sound then divided by 2.

Ultrasonic sensor is extremely useful to detect obstacles. It can detect moving obstacles as well. It serves as the “eyes” of the robot and allows the robot to prevent collision with the obstacles. It also allows the robot to perceive the area of the indoor space.


Figure 2.5 Ultrasonic sensor

#### 2.3.2.3 Infrared Sensor

It is used to detect an object in a short distance or to detect the reflectiveness of a surface. The way it works is by emitting an infrared light to an object then detect the intensity of the light bounced back. The further the object, the lower the intensity of the infrared light reflected. The less reflective is the object, the lower the intensity of the infrared light reflected. Infrared sensor is extremely useful to detect short distance obstacles. It allows the robot from colliding with the wall.(Academy, 2018)

Figure 2.6 Infrared sensor

#### 2.3.2.4 Comparison of Single Board Computer

Table 2.2 Comparison of Sensors

| | Webcam | Ultrasonic Sensor | Infrared Sensor |
|---|---|---|---|
| Detect nearby object | No | Yes | Yes |
| Measure distance | No | Up to the limit of the sensor | No |
| Capture video | Yes | No | No |
| Detect reflectiveness of an object | Yes | No | Yes |

## 2.4 INVESTIGATION OF EXISTING SYSTEM

In the market, there are some existing system that serves the same purpose as surveillance robot. The existing systems are compared to learn more about surveillance systems in general.

### 2.4.1 Existing Surveillance System

#### 2.4.1.1 DAYTECH IP Camera CCTV

DAYTECH IP Camera CCTV is a bullet camera which is designed to be placed on ceiling or wall. Its position is fixed. It records the video in 960p and saves it to a micro sd card with H.264 format. It supports up to 128GB micro sd card. During night it can switch to night vision using infrared LED with IR-Cut filter. It can connect with its app to allow the user to live viewing and playback. The CCTV can send notification to the phone via email or SMS. It connects to internet with Wi-Fi/802.11/b/g technology.("DAYTECH. (2018), DAYTECH HD720P/HD960P CCTV IP Camera WiFi Surveillance Network Security Camera Waterproof Outdoor Two Way Audio Night Vision," 2018)

DAYTECH IP Camera CCTV lacks the flexibility to record video from different angle. The adjustment of the CCTV also takes a large amount of effort just to relocate it. It cannot detect faces of the subject that it films.

Figure 2.7 DAYTECH IP Camera CCTV

### 2.4.1.2 GOQ Q7 Robot Magnetic Surveillance CCTV

GOQ Q7 Robot Magnetic Surveillance CCTV is a system which is designed to be place at a flat surface. Although that it is fixed in a location, it has the flexibility of setting the angle of the camera. It records the video in 960p and saves it to a micro sd card with H.264 format. It supports up to 64GB micro sd card. During night it can switch to night vision using infrared LED with IR-Cut filter. It can connect with its app to allow the user to live viewing and playback. The CCTV can send notification to the app if it detects motion. It connects to internet with Wi-Fi 802.11 b/g/n technology.(Borong, 2018)

GOQ Q7 Robot Magnetic Surveillance CCTV does has a attractive design. However, it lacks the ability to recognise faces, which will result in too many less important notifications that is pushed to the user. It also lacks the ability to move in the indoor area.


Figure 2.8 GOQ Q7 Robot Magnetic Surveillance CCTV

### 2.4.1.3 ESCAM Robot QN02

ESCAM Robot QN02 is a surveillance robot that is design to survey indoor environment. It has 2 wheels and able to move around inside a indoor area. It can tilt its body to adjust camera angle. It can record up to 720p video and encode it with H.264

format. It can connect with its app to allow the user to live viewing and playback. The user can control the robot via an app control. It connects to internet with Wi-Fi 802.11 b/g/n technology.(ESCAM, 2018)

ESCAM Robot QN02 is great in terms of mobility. It has a limited autonomous navigating system. It can only guide itself to charging port automatically. It also has a great adjustable camera which allows the user to film from the desired angle.


Figure 2.9 ESCAM Robot QN02

#### 2.4.1.4 Comparison of Single Board Computer

Table 2.3 Comparison of existing systems

| | DAYTECH IP Camera CCTV | GOQ Q7 Robot Magnetic Surveillance CCTV | ESCAM Robot QN02 |
|---|---|---|---|
| Recording | 960p | 960p | 720p |
| Autonomous path searching | No | No | No |
| App | Yes | Yes | Yes |
| Network | Wi-Fi 802.11 b/g/n | Wi-Fi 802.11 b/g/n | Wi-Fi 802.11 b/g/n |
| Video live stream | Yes | Yes | Yes |
| Notification | Yes | Yes | Yes |
| Human detection | No | No | No |

| | | | |
|---|---|---|---|
| Price | RM 114<br>DAYTECH IP Camera CCTV | RM 129<br>GOQ Q7 Robot Magnetic Surveillance CCTV | RM 958<br>ESCAM Robot QN02 |

### 2.4.2 Existing Movement Tracking System

#### 2.4.2.1 SIP-enabled Surveillance Patrol Robot

The object tracking method used in the system is based on ultrasonic sensors. The surveillance patrol robot is equipped with two ultrasonic sensors. Then the sensors send ultrasonic wave to the target object. By calculating the time lag difference of the two ultrasonic sensors, the system can determine the location of the object.(Tseng, Lin, Shih, & Chen, 2013)

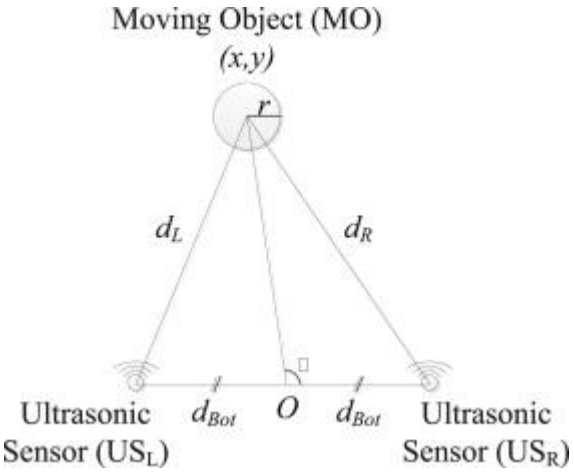



Figure 2.10 Position measurement using the triangulation method

The object in this case is assumed to be a circle. dL is the distance of the object from left ultrasonic sensor and dR is the distance of the object from right ultrasonic sensor. dBot is the distance of the centre from the sensor. r is the radius of the object.

$$x_{approx} = \frac{d_L^2 - d_R^2}{4d_{Bot}}.$$

$$y_{approx} = \sqrt{d_L^2 - (d_{Bot} + x_{approx})^2} = \sqrt{d_R^2 - (d_{Bot} - x_{approx})^2}.$$

Figure 2.11 Positioning formula

By applying the formula above, the approximate distance of the moving object can be calculated. The radius is not included in the calculation because that the radius of the moving object is often unknown in real world.

#### 2.4.2.2 Implementation of Tracking of a Moving Object Based on Camshift Approach with a UAV

The object tracking system used by the system is implemented to a drone with a camera on it. The tracking system identify the tracked object then it uses Camshift algorithm to process the data. The principle of the Camshift algorithm is based on the Meanshift algorithm. Meanshift is used for static distribution meanwhile Camshift algorithm is used for dynamic distribution. It makes the Camshift algorithm to be more effective in tracking moving object. Camshift algorithm is based on colour characteristic. RGB colour space is more sensitive to changes in illumination brightness compared to HSV colour space. For this reason, the Camshift algorithm transforms the images from RGB color to HSV color space to improve the tracking performance.(Coşkun & Ünal, 2016)

The algorithm of Camshift works as follow:

1. Firstly, select an area in the meanshift search window to be tracked. It sets the hue component from HSV color space of the tracked object.

2. Calculate the probability distribution of the selected area centred at the meanshift window. It is represented as a histogram of colours that represents the object. The pixels that have hue component in the selected region can be recognised by means of the histogram.

3. Iterate the meanshift to find the centroid of the probability image. The point selected will be the new centre point of the search window and used as the zeroth moment.

4. During the next video frame, centre the search window at the new centroid and go to step two to repeat the process.

$$M_{00} = \sum_{x}\sum_{y} I(x, y)$$

Figure 2.12 Zeroth moment formula

I(x,y) is the intensity of the discrete probability image at the point (x,y) within search window and x and range over the serarch window. The zeroth moment is calculated in figure 2.12.

$$M_{10} = \sum_{x}\sum_{y} x \times I(x, y),\ M_{01} = \sum_{x}\sum_{y} y \times I(x, y)$$

$$x_c = \frac{M_{10}}{M_{00}},\ y_c = \frac{M_{01}}{M_{00}}$$

Figure 2.13 First order moment formula and coordination of xc and yc formula

$$M_{20} = \sum_{x}\sum_{y} x^2 \times I(x, y)$$

$$M_{02} = \sum_{x}\sum_{y} y^2 \times I(x, y)$$

$$M_{11} = \sum_{x}\sum_{y} x \times y \times I(x, y)$$

Figure 2.14 Second and first moment formula

$$\theta = \frac{\tan^{-1}\left(\frac{2 \times \left(\frac{M_{11}}{M_{00}} - x_c \times y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)}\right)}{2}$$

Figure 2.15 Object orientation formula

$$a = \frac{M_{20}}{M_{00}} - x_c^2$$

$$b = 2 \times (\frac{M_{11}}{M_{00}} - x_c \times y_c)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2$$

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}}$$

Figure 2.16 Length and width of probability distribution formula

#### 2.4.2.3 Comparison of Single Board Computer

Table 2.4 Comparison Existing Movement Tracking System

| | SIP-enabled Surveillance Patrol Robot | Implementation of Tracking of a Moving Object Based on Camshift Approach with a UAV |
|---|---|---|
| Sensor involved | Ultrasonic sensor | Camera |
| Compute power required | Low | High |
| Accuracy | High | Low |
| Direction of moving object | Detectable | Detectable |

## 2.5 METHODOLOGY MODELS

Software development life cycle is crucial element for a successful deployment of a software. A few SDLC models are studied to determine if the model is suitable for the project.

### 2.5.1 Waterfall Model

Waterfall model is a linear sequential flow. The progress only flow in one direction through he phases of software life cycle. One phase can only start if the previous phase is complete. The waterfall model disallows the process from flowing back for some changes.(Existek, 2017)

The advantages of the waterfall model are that it has a well defined stages and activities. Verification of each stages ensures that errors or misunderstandings are detected early on. The disadvantage of the waterfall model is that it lacks flexibility of adjusting scope. It makes the process is almost impossible to go back to any stage after it is completed. It requires more time and a very detailed plan to execute it.

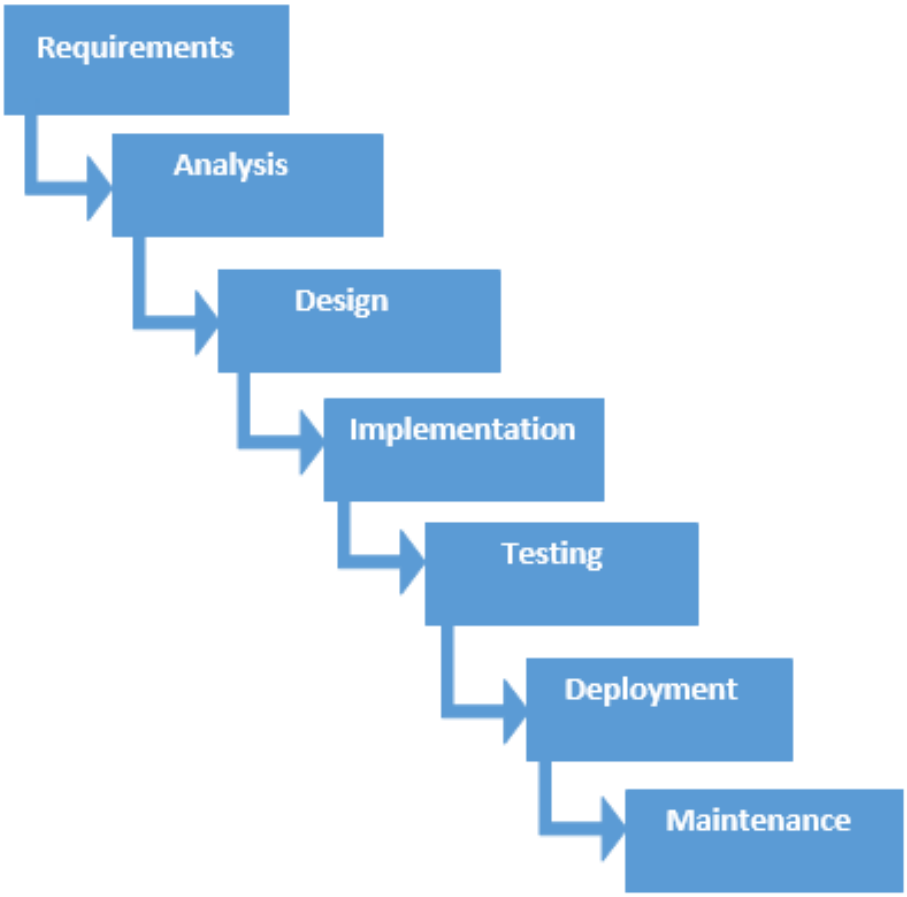



Figure 2.17 Waterfall Model

### 2.5.2 Iterative SDLC Model

The idea of iterative SDLC model is to develop a system through repeated cycles of smaller part of the system at a time. It is like repetition of mini-waterfall model. (Existek, 2017)

There are advantages of implementing this model. Iterative and incremental method produces a solid business value for the customer in a relative short time. It allows the developer team to use resources more effectively during the development process. Besides that, it can tolerate some change requests in between increments. During development, problems can be sniff out earlier. However, there are downsides of the model. It requires heavy documentation and customer engagement. During development, developers can only develop based on the dependencies on functions and features.

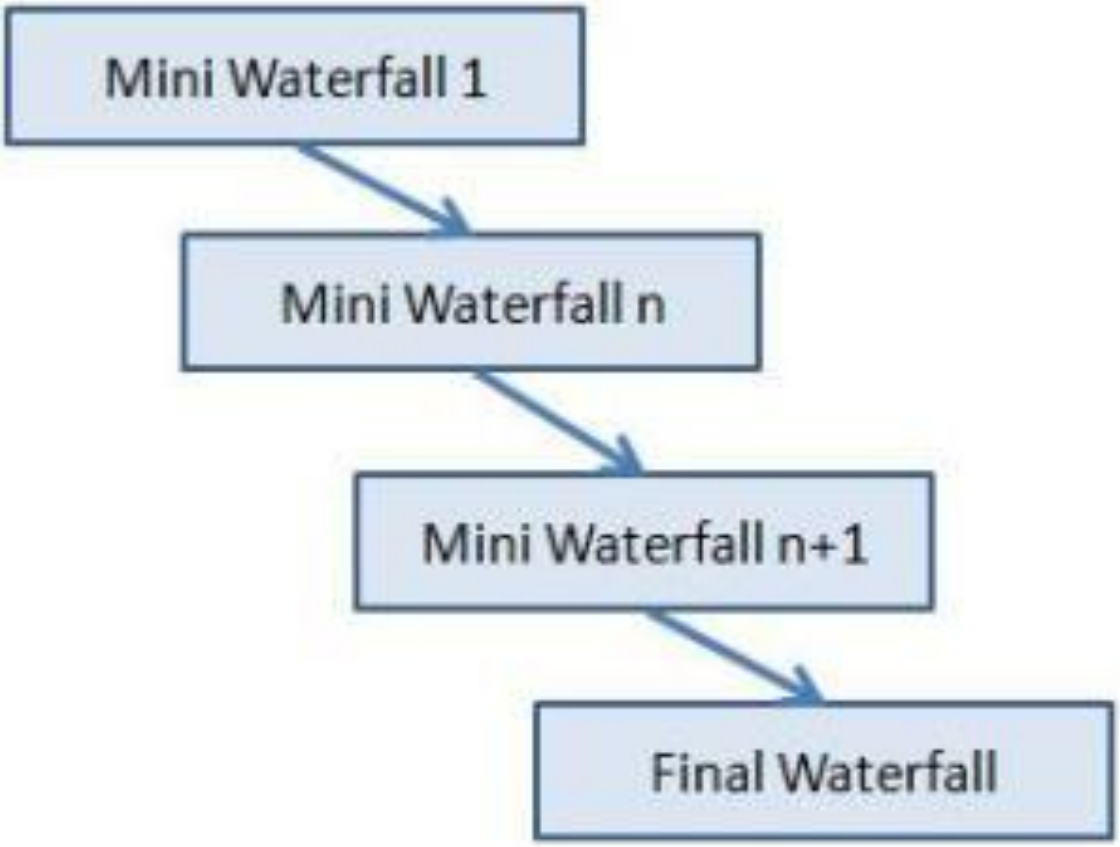



Figure 2.18 Iterative SDLC Model

### 2.5.3 Agile Model

Agile model is a combination of iterative and incremental process. Its focus is to deliver process adaptability and customer satisfactory to the customer. It is expected that the requirements and solutions evolved through collaboration between cross-function teams.(Existek, 2017)

The advantage of agile model is that it decreases the time required to deploy some system features. It also results in customer satisfactory as the development time is low. The disadvantages of the model is that is lacks scalability and the usability of the components is reduced from iteration to next iteration. It requires a strong communication with the customer. Documentation is down at later stages.

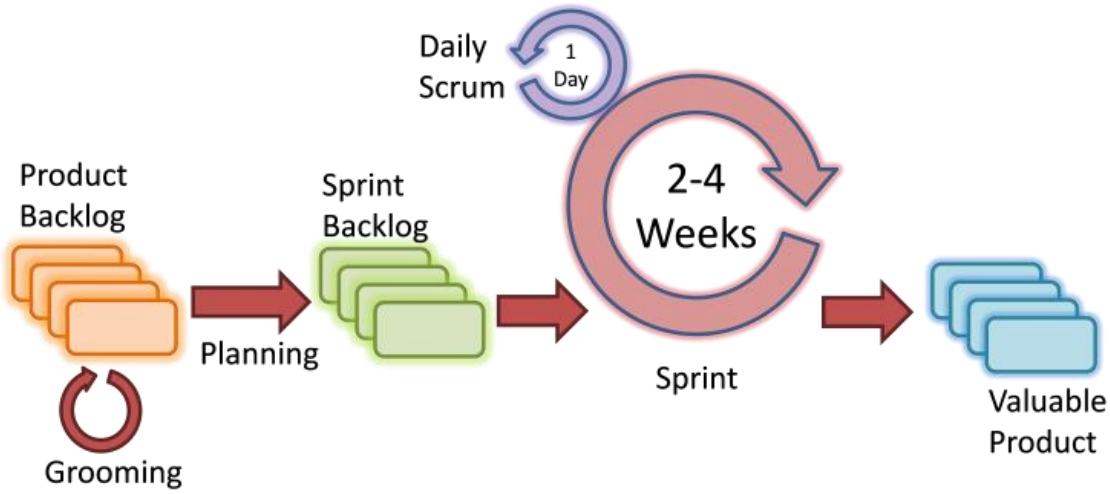



Figure 2.19 Agile Model

### 2.5.4 Comparison of Methodology Models

Table 2.5 Comparison of Methodology Models

| | Advantages | Disadvantage |
|---|---|---|
| Waterfall Model | - Enable the job of planning and scheduling the project becomes easy.<br>- Errors or misunderstanding can be detected when verification is done at each stage ensures early | - Near impossible to roll back to any stage after a stage is completed.<br>- Lack of flexibility and adjusting scope requires extensive resources.<br>- A detailed plan is needed, and it will cost a great fortune of money to develop it. |
| Iterative SDLC Model | - Delivers value to client early in the development lifecycle.<br>- Efficient way of using scarce resources by incrementally adding modules.<br>- Can tolerate change requests in between each increment.<br>- Customer value is prioritised.<br>- Obstacles and problems can be detected early in the development process. | - Requires high level of documentation.<br>- Increments are based on function and it is very feature dependencies.<br>- High level of customer involvement is needed if compared to the linear approaches.<br>- Integration may be a hassle if the developer does not consider the integration in the initial design |

| | | |
|---|---|---|
| Agile Model | - A small amount of time some system features.<br>- Close communication with customer representative makes the clarity of the project in line with customer expectation. | - Scalability is an issue if the application is scaling up.<br>- The requires high communication skills and great empathy to get the whole picture of the desired software.<br>- The components have a low potential for reusability. |

### 2.5.5 Conclusion of Methodology Models

As conclusion, iterative SDLC model is the most suitable model to be implemented in Autonomous Smart Robot Surveillance System. It allows for rapid development which gives room for change request as it will be issued in the early stage of development. It reduces the time wasted on misunderstanding the requirements. The development of the system does allow for flexibility in user requirements and handles the unexpected risk gracefully.

# CHAPTER 3

# METHODOLOGY

## 3.1 Overview

In order to develop an autonomous smart robot surveillance system, we need a SDLC model that allows the development to be executed in an order manner. SDLC models ensures that a system is developed in an ordered sequence and it saves resources and yet improve the reliability of the final product. SDLC model can be implemented by categorising the effort of developing the system into a general 5 categories. The categories may vary depends on the SDLC model chosen. It is usually consisting of "Planning", "Analysis", "Design", "Implementation" and "Maintenance". (Sami, 2012)

In planning stage, the main thing is to come up with the scope of the system and the feasibility of it. At the same time, the resources and the development schedule are also planned. In analysis stage, the main focus is to understand the key factor on how the current system used by the organisation is now satisfying their needs and how the system works. A few different approaches will be used to gather the information about it. In design phase, the main focus is to craft a solution based on the information that is collected in analysis phase. In this phase, SRS and SDD are written. In the implementation phase, the actual coding job starts. In this stage the programmers materialise the design which is stated in SDD into actual product. In the last phase of maintenance phase, the focus is on making sure that the system works. In this phase, often bug fixes and security patch is applied to the designed application.

In this project, the method of SDLC used is iterative model. Iterative model is a SDLC philosophy that generally designs the application a prat at a time and the functionality of the application is included in an iterative manner. This model is selected for this project because that it allows the project to improve from time to time and it can

get a reliable feedback form the client. It also offers a systematic orderly way of developing the system by splitting project into smaller parts and develop it in an order.

## 3.2 Methodology

### 3.2.1 Planning Phase

Planning phase is the phase when the task and estimated time is estimated. It also takes account of the resources used in the whole project. It is done on the initial of the SDLC and only once.

In this phase, the activities from PSM 1 to PSM 2 are planned with Microsoft Project. The project is decided to use iteration method. This is because the development which is separated into a few iterations to build is generally less risky compared to one shot method. The surveillance robot needs to improve its accuracy over time, so iteration is preferred over incremental method. The development will be done in 3 iterations. The budget for the project will be planned to have a maximum of RM 6 00. First Iteration is from 5/3/2018 to 18/9/18. Second iteration is from 19/9/18 to 19/11/18. The last iteration will be from 20/11/18 to 18/1/19.

### 3.2.2 Analysis Phase

Analysis phase is about understanding existing system and come up with requirements for the project. This phase is done by cycles. In each cycle, new requirements are added and the changes to the existing requirements may be included as well.

#### 3.2.2.1 Verification of Idea and Technologies

The concept and idea of the system is studied during the process of preparing chapter 2. Firstly, market research for similar products is search on Lazada and Lelong. The categories that are found are traditional CCTV and surveillance robot. It is found that the CCTV do lack of freedom of adjusting. The surveillance robot does lack of autonomous surveillance feature. The technologies that will be used by the system is studied by reading the IEEE thesis from sciencedirect.com. There are 2 ways of detecting environment that is usable in the project. The first way is to use the ultrasonic sensor to sense the motion of a moving object. It was studied at first because one of the potential

requirements may be tracking moving person. The second way of tracking person is via computer vision. The computer vision will be the main way of tracking people since that the surveillance robot only needs to adjust the angle of the camera according to the moving person. The library that is chosen for computer vision is OpenCV. It is chosen because there are tons of tutorials that lower the risk of developing the project. The potential list of service of the cloud storage are Google Drive, Dropbox and One Drive. All of it do support python and java. The capacity of the free storage is compared since that it is the bottleneck of the service. Google Drive is picked among these services since that it offers the most free space and it also integrates with Android application smoothly. Azure SQL database is chosen to be the online database since that it offers a large capacity of up to 250 GB and the service is free.

#### 3.2.2.2 Verification of Hardware Components

The on the hardware side. Firstly, the research is based on the single board computer. The choice is made by comparing the processing power. Processing power is a critical part of the system since that the application will use computer vision and neural network AI to detect people. Raspberry Pi Model 3 B+ is chosen over the older model Raspberry Pi Model 3 B because the improved power efficiency of processor and WiFi connectivity. Then the price of the Raspberry Pi Model 3 B+ is compared on internet. It is found that the retail store at Taobao is cheaper and available on Taobao which is around RM 50 cheaper than other websites while Lelong and Lazada do not sell any of those. In addition, the Taobao seller does sell the robot bundle which makes the purchasing procedure easier. The sensors which are studied were webcam, ultrasonic sensor and infrared sensor. All of those sensors are needed for the project. The webcam is used in detecting vision. The ultrasonic sensor is used for detecting distance object and the infrared sensor is used for detecting close up object.

#### 3.2.2.3 Getting Requirements from Client

In this phase, Mr. Law is appointed as the client. A casual interview was performed on 18th of April to understand the current system used by his home. The strengths and weakness are analysed to define some of the functional requirements. A list of questions is formed based on the first interview. On 27th of April, a questionnaire is designed for the client. It is used to interview Mr. Law to further understand the system.

Base on the requirements stated, a SRS document is created. The second iteration and third iteration of requirement elicitation will be performed after the first iteration ends.

The functional requirements are as follows:

1. The surveillance robot has the feature to move from one point to another using guiding tiles.

2. The surveillance robot can avoid object which is in front or at the back of the surveillance robot.

3. The surveillance robot is able to change the surveillance stance after 30 minutes.

4. The surveillance robot only records if there is moving object.

5. After recording, the surveillance robot will save in SD card and uploads to Google Drive.

6. The surveillance robot is able to send a notification to the user if there are hardware faults.

7. The user is able to view the saved video on Google Drive with the Android application.

8. The user is able to view the notification history in the Android application.

9. The user is able to delete the notification with the Android application.

#### 3.2.2.4 Use Case Definition

Based on the functional requirements, a total of 4 use cases are designed. Due to the nature of autonomous feature the ASRSS which does not require any actor to act, some of the functional requirements are not included in the use case.

Table 3.1 List of Use Case

| Use Case Name | Description |
|---|---|
| Record moving object | Record a moving object and uploads to Google Drive. |
| View saved video | View and play the saved videos from the Google Drive. |
| View notification | View notification list and able to delete the notification. |
| Report failure of hardware | Surveillance robot sends a notification if there is failure of hardware. |

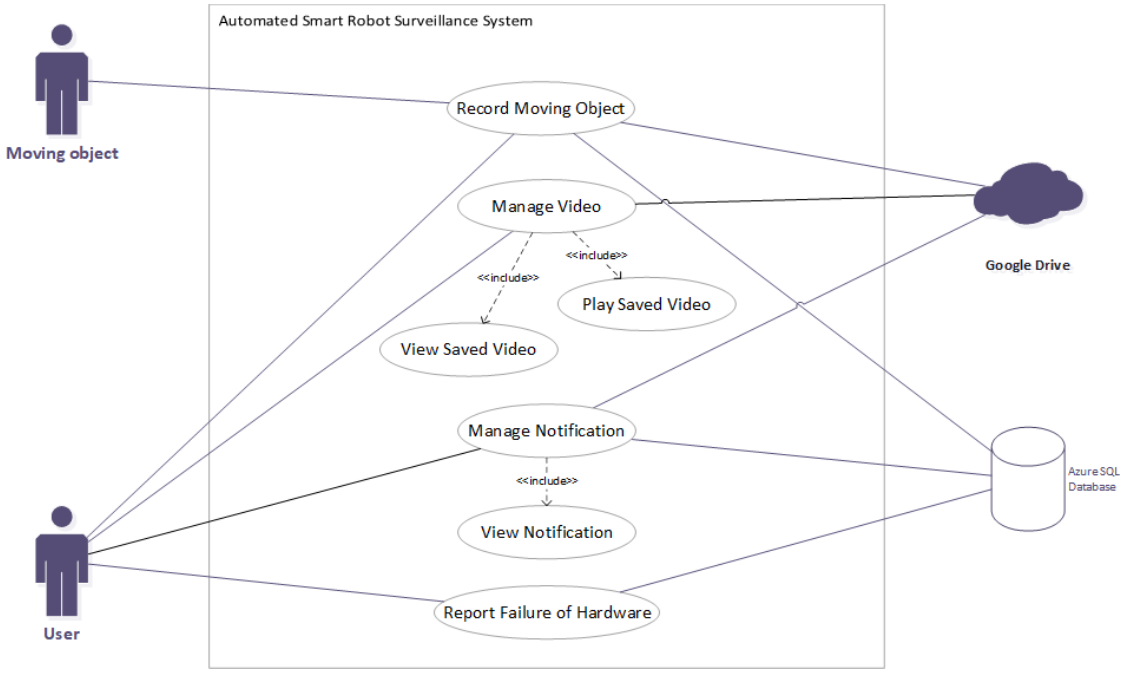



Figure 3.1 Use Case Diagram of ASRSS

### 3.2.2.5 Context Diagram Design

The system interacts with a total of 5 actors, including moving object, user, internet, Azure SQL Database and Google Drive.

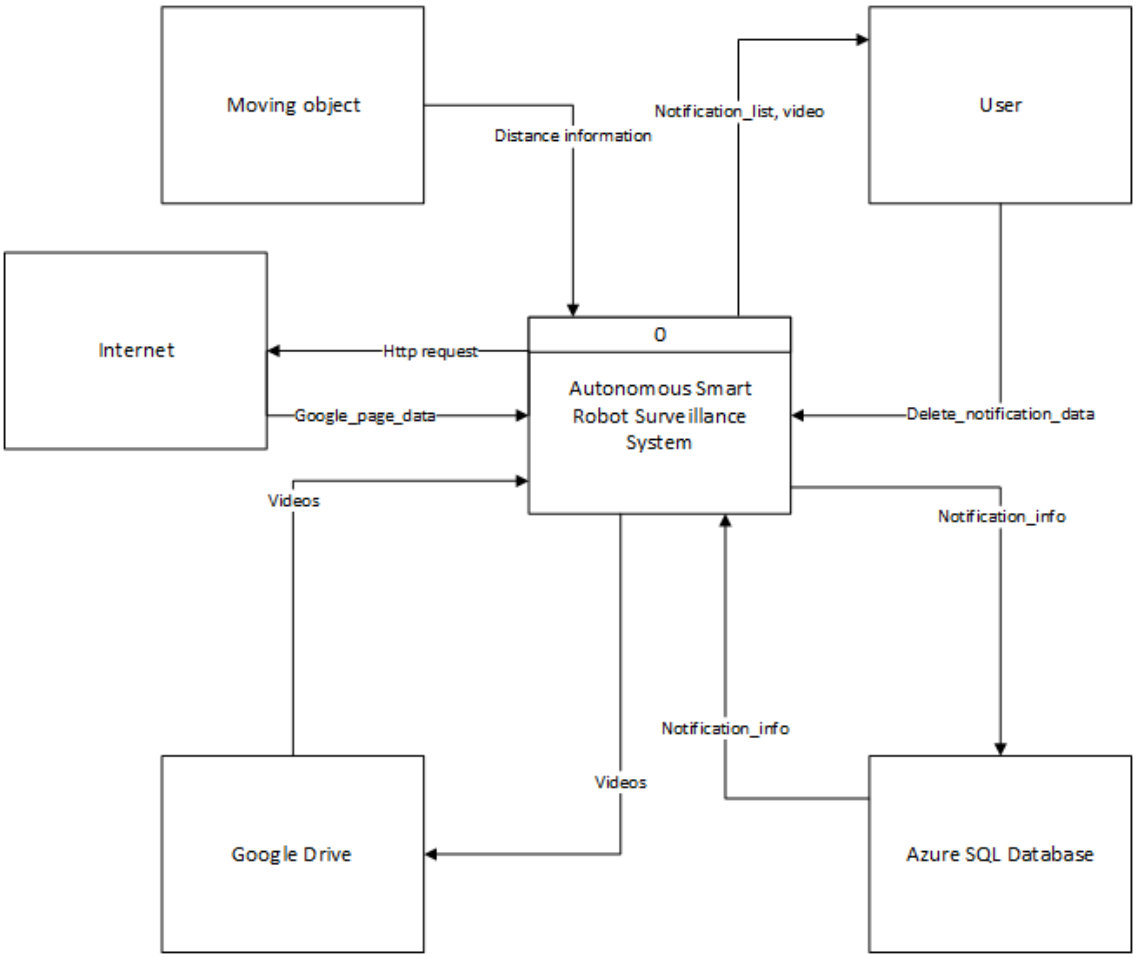



Figure 3.2 Context Diagram of ASRSS

### 3.2.3 Design Phase

Design phase is the phase where the software is designed virtually. The design scope is limited to the functionalities that are related to the added or modified requirements of the cycle. Based on the requirements gathered using the SRS, parts of the requirements are selected to be developed during the first iteration. Base on the chosen requirements, the design of algorithm is designed and documented into an SDD.

#### 3.2.3.1 Architecture Pattern

Initially the architecture pattern must be decided before any of other steps can be taken. After understanding layer architecture and MVC architecture. The model that will be used for the Android application is MVC. The main advantage of the architecture is that it makes the Android application more manageable as the functions of the

application can be grouped together and there is a defined separation of the group functions.

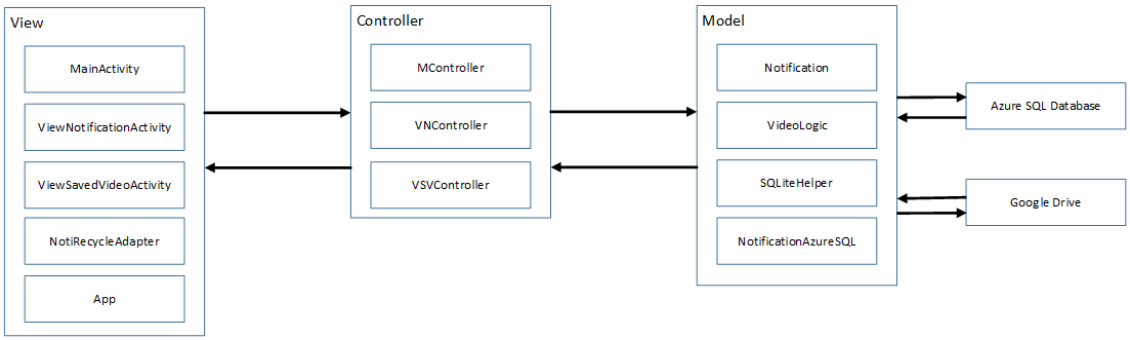



Figure 3.3 MVC architecture design of ASRSS_App

It mainly has three components including View, Controller and Model. Each component work together to get the application going. View is the component that handles the user interface. Controller is the component that handles the business logic. The model is the component that handles the calculations and the database. (Codecedemy, 2018)

The way of the components interacting with each other is as follows:

1. View calls the controller whenever the user interacts with the user interface.

2. The controller calls the application logic to perform the algorithm which is needed to complete the user's intention.

3. After succeeding in performing the task, the application logic component will return a value to controller to indicate the status of the task performed.

4. After performing the business logic, the controller will return a value back to the view to inform about the status of the application.

5. The view will update the interface according to the latest state of the application.

The model that will be used by the surveillance robot will be 3 tier layered architecture with all its layer being closed layers. It has a one directional relationship with the layers below it. It is consisting of level 1, level 2 and level 3. The layer 1 is mainly responsible for managing the whole system in a top-level view. Layer 2 is mainly in

charge of the business logic of the system. Layer 3 is responsible for IO communications and the movement of the system.

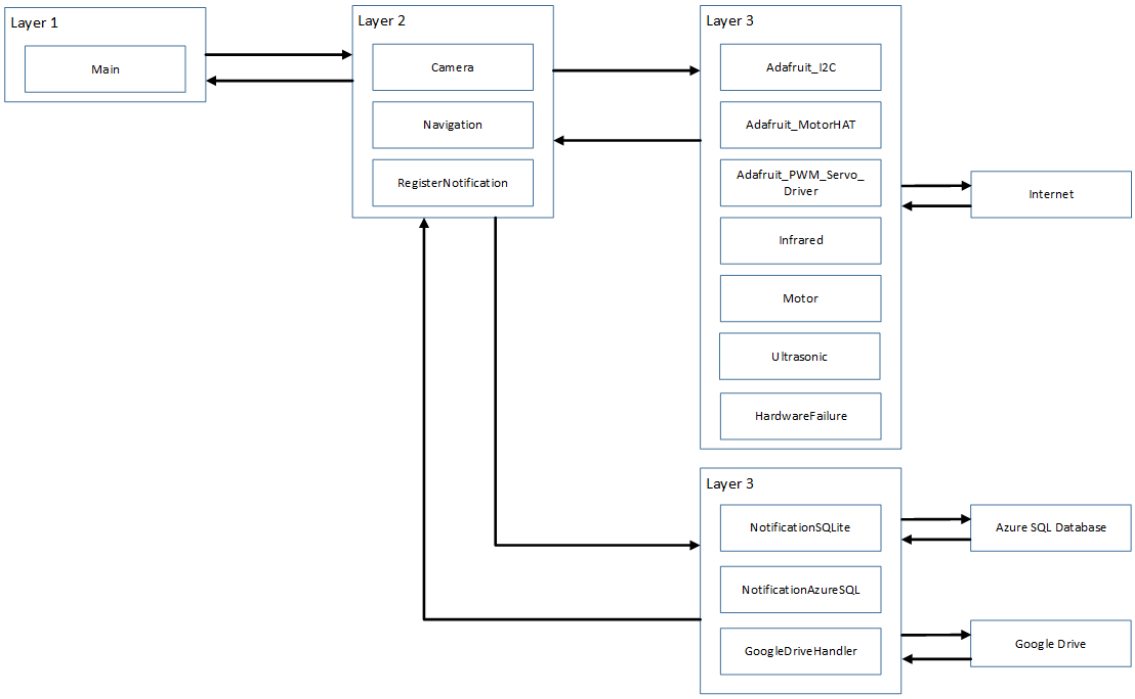



Figure 3.4 3 tier layered architecture design of ASRSS

### 3.2.3.2 Motion Detection Algorithm

The motion detection is mainly used to detecting an active person. The motion detection algorithm works by comparing the first frame and the current frame. The first frame is assumed to not to contain any moving person or object. If there is any difference, then the difference is highlighted as the moving person. The algorithm works as follows:

1. The first frame is captured and converted to greyscale.
2. Then it is stored into as an object.
3. The most current frame is captured and converted to greyscale.
4. The current frame is compared to the first frame and the difference is stored.
5. The stored difference frame is highlighted its difference by reducing values which are lower than 25 to 0.

6. The holes within the white spot is filled up
7. Contours are plotted based on the difference frame.
8. If the contour area is smaller than 500 it is ignored, else it is drawn with green boxes.

#### 3.2.3.3 Ultrasonic Distance Detection Algorithm

The ultrasonic distance detection is mainly used to calculate the distance of the sensor to an object. The method works by sending a ultrasonic wave and receiving a ultrasonic wave. The lag time is recorded and calculated to calculate the distance the measurement. The algorithm works as follows.

1. An ultrasonic wave is sent.
2. The time for sending is recorded.
3. Receive the reflection of the ultrasonic wave.
4. The receiving time is recorded.
5. The distance is calculated by lag time * speed of sound / 2 where the 2 indicates that it has travelled double the distance forward and backward.
6. The distance is multiplied by 100 to convert the result to cm unit.

#### 3.2.3.4 ASRSS Physical Design

In the ASRSS system consist of a total of 5 IR sensors, 1 webcam, 1 ultrasonic sensor, 4 motors, 1 battery holder, 1 Raspberry Pi Model 3B+, 1 extension board, 4 motors, the main frame and 4 wheels. The sketched design of the ASRSS is displayed in Figure 3.5 and Figure 3.6.

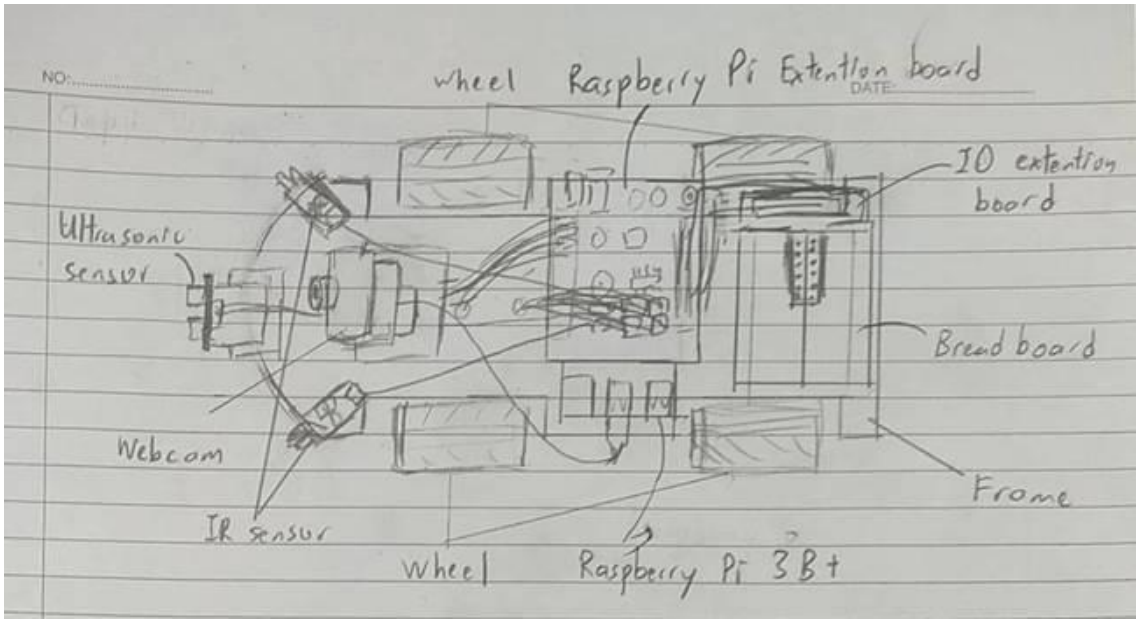



Figure 3.5 Top View of ASRSS Sketched Design

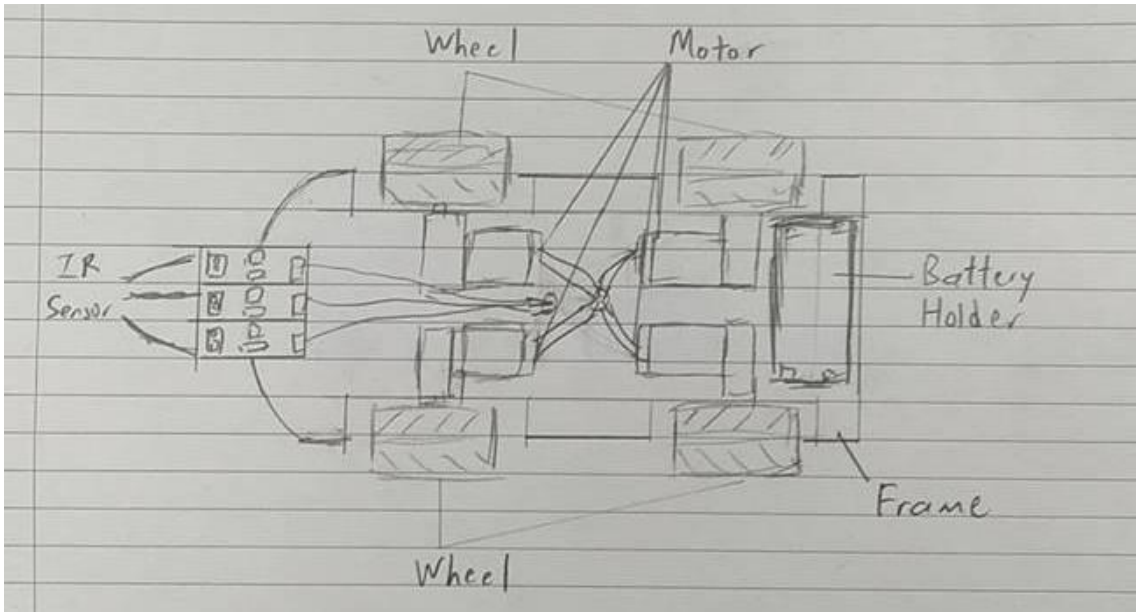



Figure 3.6 Bottom View of ASRSS Sketched Design

### 3.2.3.5 Static Detection

The motion detection is mainly used to detecting an active person. The motion detection algorithm works by comparing the first frame and the current frame. The first

frame is assumed to not to contain any moving person or object. If there is any difference, then the difference is highlighted as the moving person. The algorithm works as follows:

1. Every 5 second a frame is captured.

2. The frame is converted to grey scale and blurred to reduce the noise of the image.

3. The frame in inserted in a list as a queue.

4. If the queue is larger than 10, pop one frame from it.

5. If the queue size is 10, the last image in the queue is compared to other 9 images.

6. The stored difference frame is highlighted its difference by reducing values which are lower than 25 to 0.

7. The difference of the comparison is stored.

8. The difference is compared to the threshold value.

9. If any of the difference is larger than the threshold value, it means that the recording is not static.

#### 3.2.3.6 Module

The classes of the ASRSS system are classified as modules.

#### 3.2.3.2.1 ASRSS

##### 3.2.3.2.1.1 Main Module

Main module contains Main class. Its main functionality is to allow the surveillance robot to manage all the motion and capture activities.

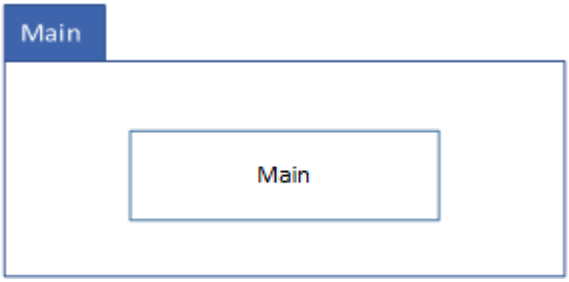



Figure 3.7 Main Module

##### 3.2.3.2.1.2 Sensor Module

Sensor module contains Adafruit_I2C class, Adafruit_MotorHAT class, Adafruit_PWM_Servo_Driver class, Infrared class, Motor class, Ultrasonic class, HardwareFailure class. Its main functionality is to allow the robot to detect the environment via the sensor, allow the movement of the motors and detecting the hardware failures of the sensors.

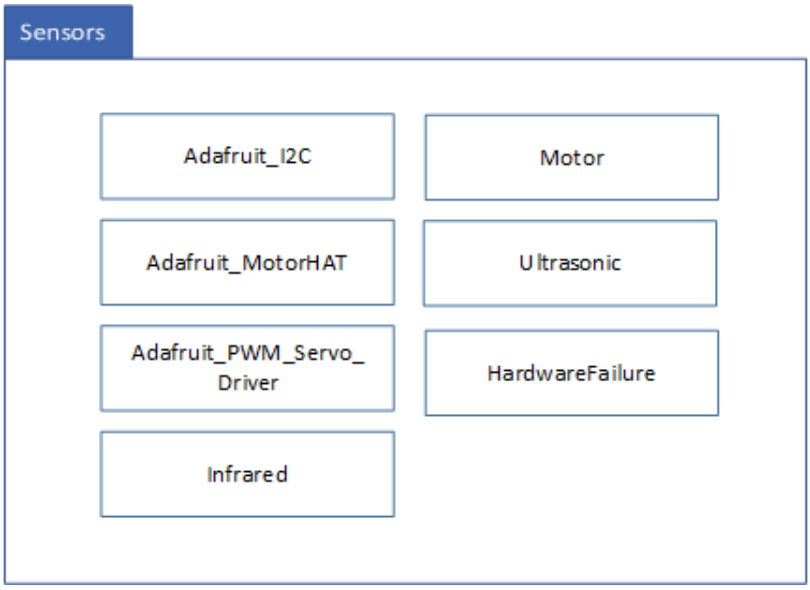



Figure 3.8 Sensor Module

#### 3.2.3.2.1.3 Navigation Module

Navigation module contains Navigation class. Its main functionality is to allow the robot to navigate in the surveillance environment and avoid objects.


Figure 3.9 Navigation Module

#### 3.2.3.2.1.4 Database Module

Database module contains RegisterNotification class, NotificationSQLite class, NotificationAzureSQL class, and GoogleDriveHandler class. Its main functionality is to allow the robot to interact with SQLite, Azure SQL database and Google Drive.

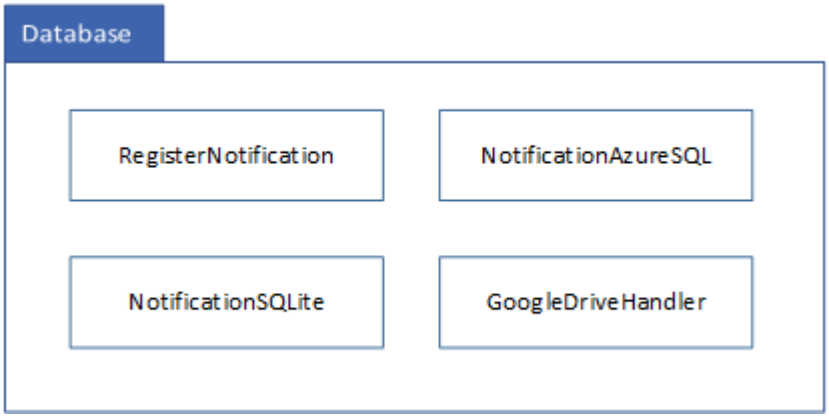



Figure 3.10 Database Module

#### 3.2.3.2.1.5 Camera Module

Camera module contains Camera class. Its main functionality is to record video and perform computer vision and the business logic behind recording videos.

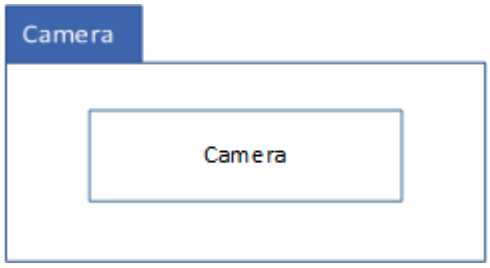



Figure 3.11 Recording Module

#### 3.2.3.2.2 ASRSS_App

##### 3.2.3.2.2.1 MainPage Module

MainPage module contains MainActivity class and MController class. Its main functionality is to redirect the user to other activities.

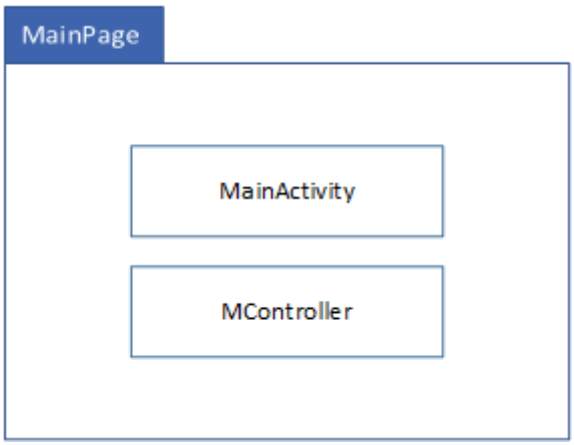



Figure 3.12 MainPage Module

##### 3.2.3.2.2.2 Database Module

Database module contains SQLiteHelper class and NotificationAzureSQL class. Its main functionality is to allow the robot to interact with SQLite, Azure SQL database.

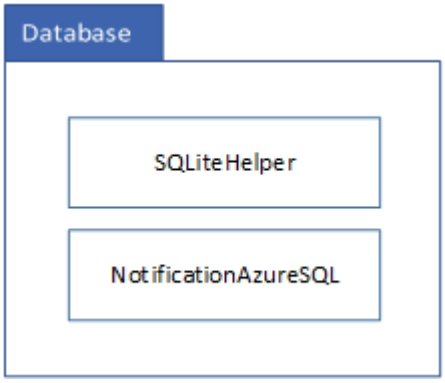



Figure 3.13 LiveStream Module

### 3.2.3.2.2.3 ViewVideo Module

ViewVideo module contains ViewSavedVideoActivity class, VSVController class, Video class, VideoLogic class and VideoPModel class. Its main functionality is allow the user to watch recorded video from Google Drive.

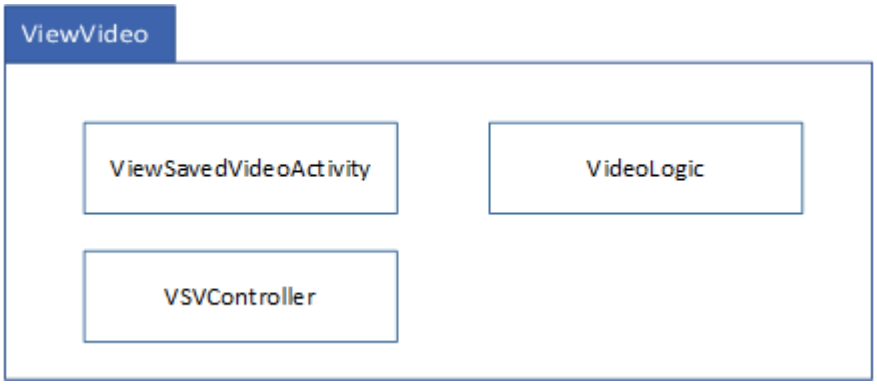



Figure 3.14 ViewVideo Module

### 3.2.3.2.2.4 Notification Module

Notification module contains ViewNotificationActivity class, NotiRecycleAdapter class, VNController class, Notification class and App class. Its main functionality is to send push notification to the user and allows the user to check notification history.

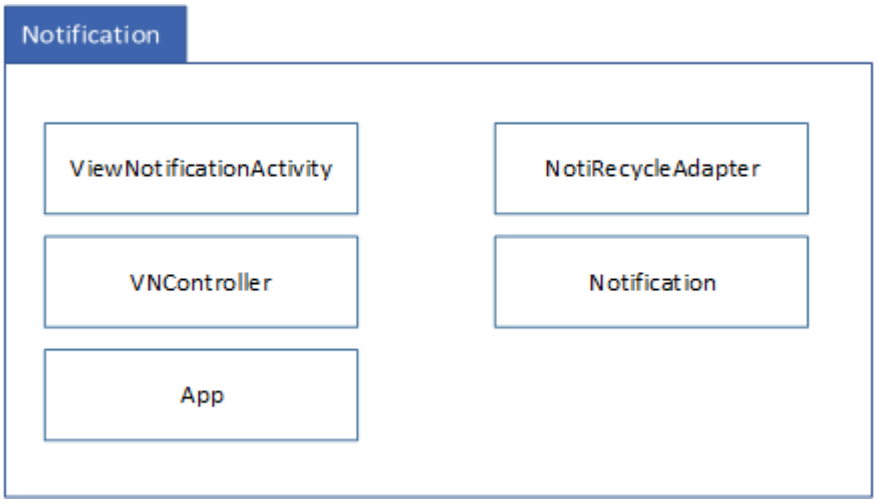



Figure 3.15 Notification Module

### 3.2.3.7 Database Design

The database used by the system will be divided into 3. The first database will be the local database used by the surveillance system. The second database is the Android application database. And The last database will be the Azure SQL database. The reason that the database does have a local database is to allow the application and the surveillance to work offline. All these 3 databases will work together and sync to perform the tasks together.

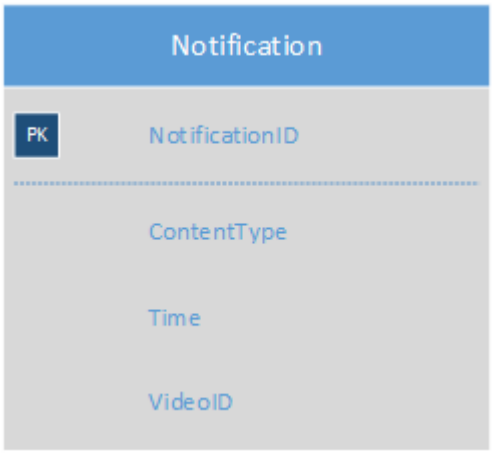



Figure 3.16 ERD design of ASRSS

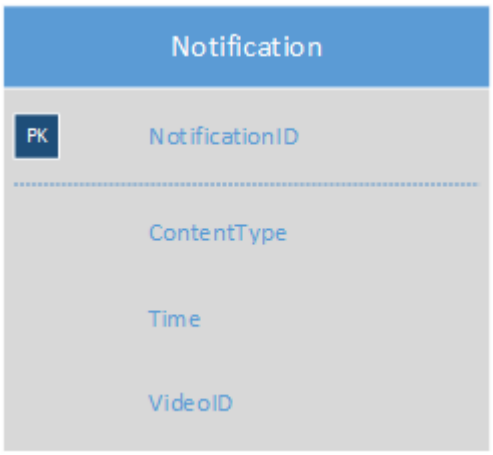



Figure 3.17 ERD design of ASRSS_App

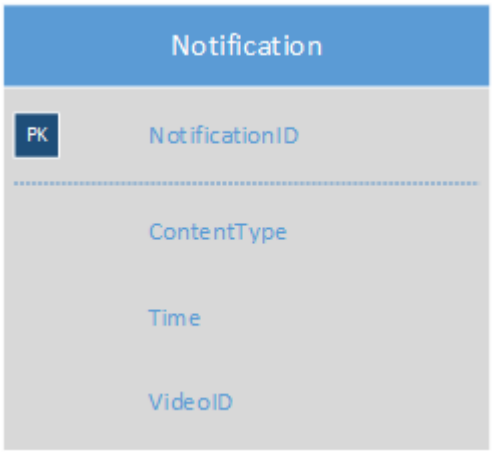



Figure 3.18 ERD design of Azure SQL database

#### 3.2.3.8 Interface Design

On the design, the requirements of the client are being prioritised in the design process. The client wants the user interface design which is simple and self-explanatory. There are 4 interfaces which include MainActivity, LiveStreamActivity, ViewSavedVideoActivity and ViewNotificationActivity.

#### 3.2.3.3.1 MainActivity

In this activity, it contains a total of 2 buttons including “View Saved Video” and “View Notification”. Each button will redirect the user to the respective interface.

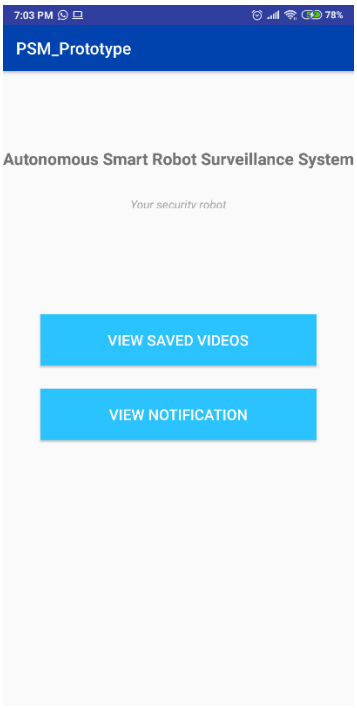



Figure 3.19 Interface design of HomeActivity

#### 3.2.3.3.2 ViewSavedVideoActivity

In this activity, it contains a video player and a “Find Video” button. In this activity, the user can click on “Find Video” button to search for videos on the Google Drive. Then after selecting the video from the gallery, the user will be able to play the video on this activity. The user is able to pause, forward and backward the video.

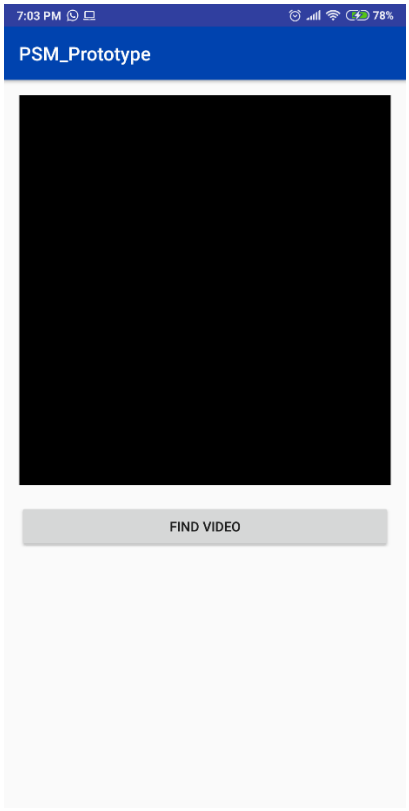



Figure 3.20 Interface design of ViewSavedVideoActivity

#### 3.2.3.3.3 ViewNotificationActivity

In this activity, it contains recycleview and “View Video” button. In this activity, the user can see the list of notification of the application. The user can delete the notification by a long press. By pressing the “View Video” button, the user is navigated to the ViewSavedVideoActivity.

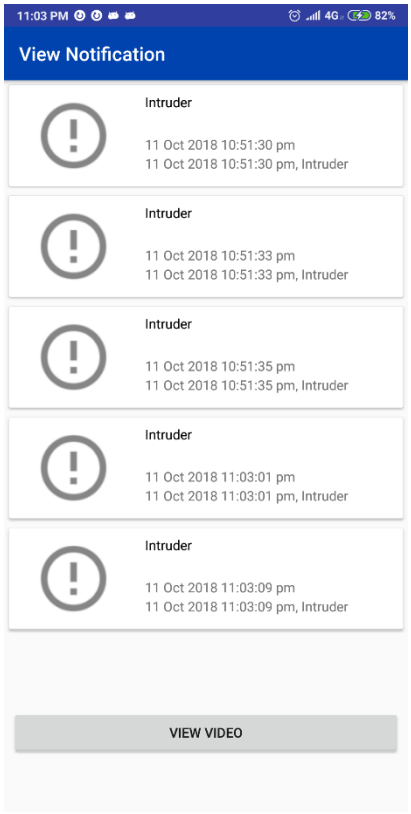



Figure 3.21 Interface design of ViewNotificationActivity

#### 3.2.3.3.4 Storyboard

In this activity, it contains a total of 2 buttons including “View Saved Video” and “View Notification”. Each button will redirect the user to the respective interface.

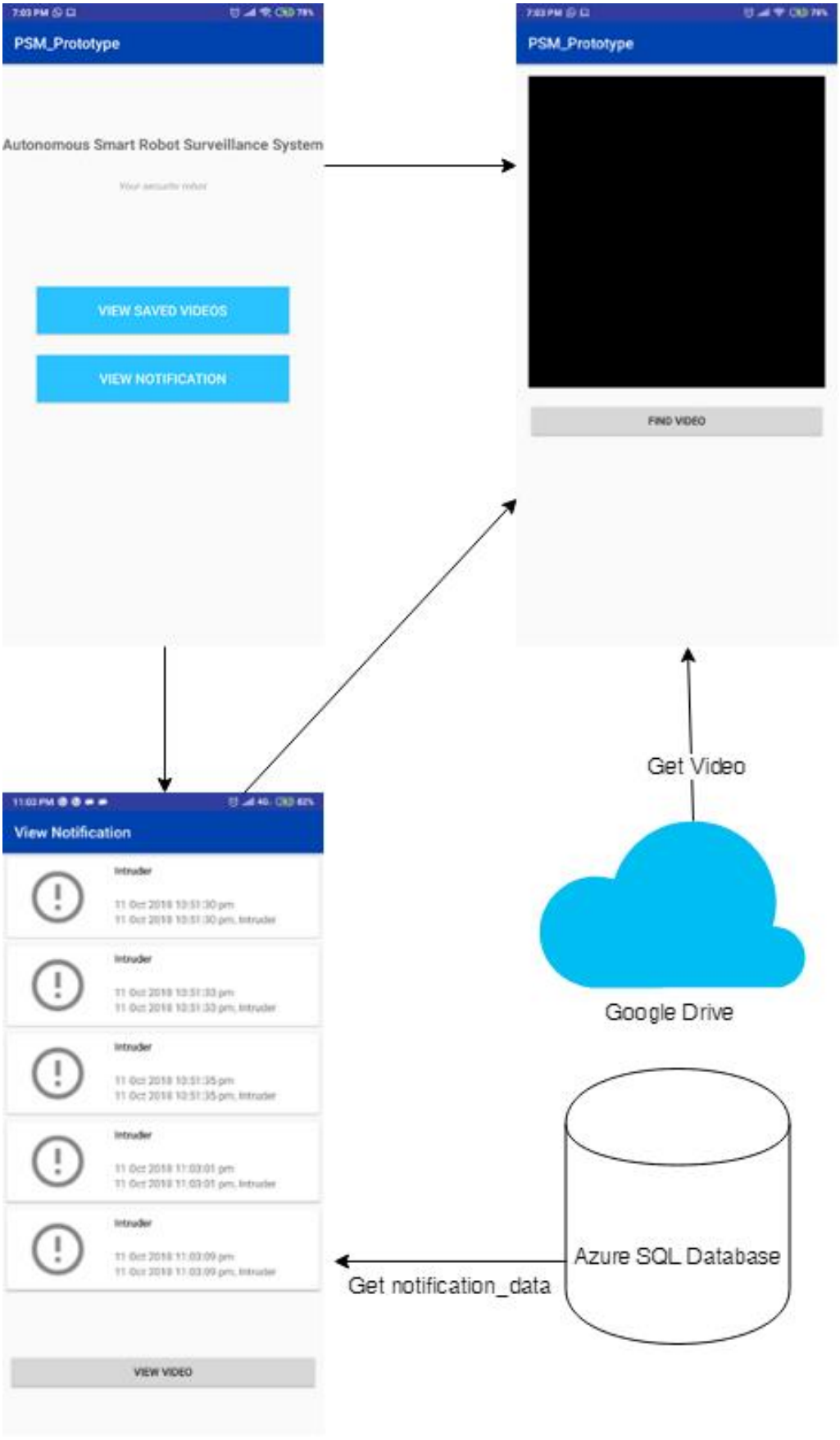



Figure 3.22 Storyboard of ASRSS_App

### 3.2.4 Development Phase

Development phase is where the designed algorithms are implemented in actual device. The development work only covers the designed algorithms that are mentioned in the previous phase of the cycle. At the end of this phase, we will expect a working prototype to be done. It cannot be determined as a product since that it is not the final product. The software is coding according to what is written in the SDD. After the code is done, the prototype is ready for testing purpose.

#### 3.2.4.1 First Iteration

On the first iteration, the development is started with the Android application. It is because that the hardware of the surveillance robot is expected to arrive on July. The application needs to fulfil a few functionalities. The Android application is able to connect to Google drive application and play video. The Android application is able to connect to Azure SQL database and retrieve data from the Azure SQL database. The application will be written using the Android Studio IDE and Java language as the primary coding language. The application will be written by the cycle of view to controller to application logic to model. The first functionality that will be written will be view notification. In this mini cycle, all the logic and design of the notification will be written. A local SQlite database and the Azure SQL database will be designed as well. Then on the second mini cycle, the connection to the Google Drive application and playing video will be written.

#### 3.2.4.2 Second Iteration

The second iteration which is planned will be mainly focus on developing the surveillance robot. It will cover a few functional requirements which includes the ability to navigate according to the guiding tiles, able to avoid nearby obstacles and the ability to detect the environment with sensors and be able to detect hardware failures. Before any of the software development starts, the surveillance robot will be build and the circuit will be configured. After that the sensors will be calibrated. The development will be on mini cycles which starts from the sensors layer to the business logic layer. The development of features will start from the ability to understand the environment with the

sensor, the ability to navigate according to the guiding tiles and the ability to avoid obstacles.

#### 3.2.4.3 Third Iteration

During the third or the final iteration, the remaining functional requirements will be developed. The requirements which are covered during this iteration includes recording a moving object, saves the recording into SD card and uploads it to Google Drive, the ability to change location once every 30 minutes of no recorded movement, the ability to send notification to the user and the ability to recalibrate still images. The development order will be same as always with mini cycles which starts from the camera module to the database module. The development of features will start from recording a moving object, saves the recording into SD card and uploads to Google Drive, detect still images, the ability to send notification to the user. The feature of detecting hardware faults and low battery are done at last since that it requires other features to be developed first.

### 3.2.5 Testing Phase

Testing phase is where the system is tested according to the new or modified requirements of the cycle. After this phase ends, the cycle will start again from analysis phase.

In this project will be tested using a few methods. At the lowest level of unit test, white box testing will be used to test the code. The code is tested with debugger tool provided by the Android Studio for the android application. For the python application of the Raspberry Pi system, unit testing module is used for the testing of the code. The value of the variables is checked upon each step to ensure the quality of the code. Positive testing strategy is used to ensure that the product do fulfil the functional requirements.

In integration testing, white box approach is carried out at the interface classes and black box approach is carried out for the overall system behaviour. The interfaces are tested with debugger tool provided by the Android Studio for the android application. For the python application of the Raspberry Pi system, unit testing module is used for the testing of the code. User acceptance testing will be carried out as well. The blackbox

testing is carried out by inputting dummy values and observe any system faults upon failures.

Use case testing method will be used. This method will be used since that the test method is similar to real world uses. The test case design will be based on the use cases and the most common practices that is expected to be performed by the user. The behaviour of the ASRSS system will be recorded during the testing. All the result of the testes will be documented in user acceptance test report.

## 3.3 Hardware and Software Requirements

To make the project possible a certain hardware and software tools must be utilized in the development process.

### 3.3.1 Hardware Requirements

There is a few hardware that is required by the project.

#### 3.3.1.1 Raspberry Pi Model 3 B+

The autonomous smart robot surveillance system needs a single board computer to control it. Raspberry Pi Model 3 B+ is the perfect candidate since that it has a relative powerful CPU for image recognition and enough GPIO and ports for sensors. It is used to control the infrared sensors, ultrasonic sensors, webcam and motors.

Raspberry Pi 3 Model B+ is a powerful single board computer. It is powerd by Boradcom BCM2837B0 as its SOC. It has a 1.4GHz quad-core ARM Cortex-A53 as its CPU and Broadcom VideoCore IV (400MHz) as its GPU. It is powered by 1 GB of SDRAM. It supports MicroSDHC by having a MicroSDHC slot. It can boot from USB drive via USB Boot Mode. For networking, it supports up to 300Mbit/s Ethernet, 802.11ac dual band 2.4/5 GHz and wireless Bluetooth 4.2 LS BLE. Raspberry Pi 3 Model B+ has 4 USB 2.0 ports HDMI (v1.3) as video output and a 3.5mm headphone jack as audio output. It only supports digital inputs.(Watson, 2017)

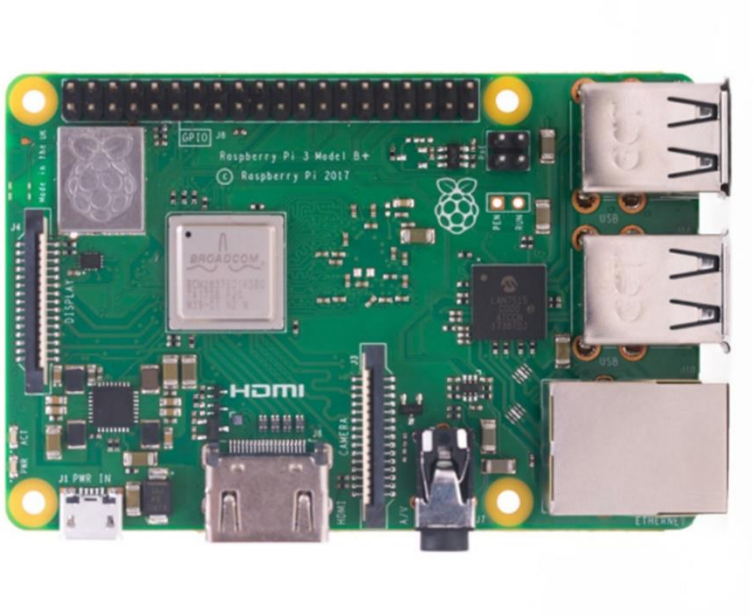



Figure 3.23 : Raspberry Pi Model 3 B+

### 3.3.1.2 Ultrasonic sensor

The systems need a sensor to detect distance object and moving object. Ultrasonic sensor enables the system to detect objects which is in front of the robot. It helps in avoiding the robot from obstacles and bumping into a wall.


Figure 3.24 Ultrasonic sensor

#### 3.3.1.3 Infrared sensor

The infrared sensor aids the robot from bumping into near distance obstacles and accidentally move into a void area. The infrared sensor does so by detecting close distance object and return a signal if something is detected.

Figure 3.25 Infrared sensor

#### 3.3.1.4 Webcam

The webcam allows the robot to see the surrounding. It allows the system to perform face recognition and object tracking. It also enables the system to record the footage. The web cam has the resolution of 720p HD at 30 fps recoding rate. It has a USB type b connector to connect with the Raspberry Pi Model 3 B+.


Figure 3.26 Webcam

#### 3.3.1.5 DC Motor

Motors are used to make the robot mobilise. A 4-motor system enables the robot to move around in a indoor area. Having the ability to mobilize increases the robot's surveillance coverage and allows the robot to have a flexibility in choosing the area for coverage.

Figure 3.27 DC Motor

#### 3.3.1.6 Car frame

The robot needs a frame to hold all the components and the wheels to enable actual movement. The frame has a holder for Raspberry Pi Model 3 B+ and 4 holders for DC motors.

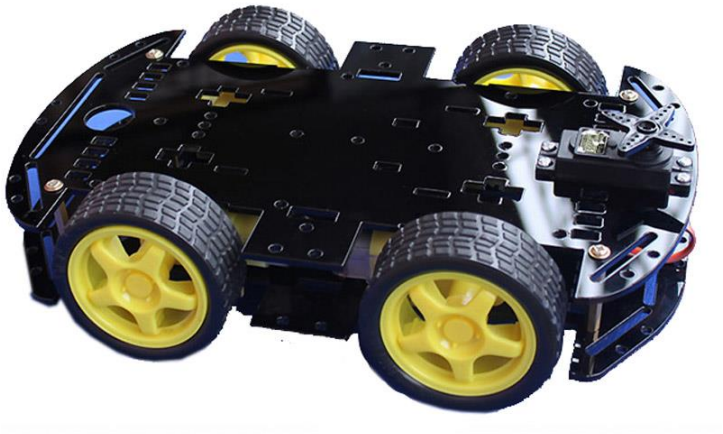


Figure 3.28 Car frame

#### 3.3.1.7 Android Phone

To allow the user to interact with the autonomous smart surveillance system, a smart phone is used for this project. Android operating system is preferred as majority of the market uses Android phone and its development is free. The chosen Android phone is Redmi 5 Plus.

Redmi 5 Plus has the Android 8.1 operation system. It is equipped with Snapdragon 625 SoC with 8 Cortex-A53 CPU clocked at 2.0GHz. It has 4GB ram and 64GB internal storage.(GSMArena, 2017)


Figure 3.29 Redmi 5 Plus

#### 3.3.1.8 Computer

The documentation of the thesis, SRS and SDD are written using an application. Using an application boast the speed of writing the documents compared to traditional hand-written way. To program the application and Raspberry Pi Model 3 B+, a computer is needed for the task. The chosen computer is HP Pavilion Notebook. The computer is mainly used for documentation and programming the robot and Android application.

HP Pavilion Notebook has the specs of 2.5GHz Intel i5 2500U processor, 8GB of LDDR3 ram, 512GB of SSD. It has the OS of Windows 10 Single Language Edition.


Figure 3.30 HP Pavilion Notebook

#### 3.3.1.9 Summary of Hardware Requirements

Table 3.2 Summary of Hardware Requirements

| Hardware requirements | Purpose |
|---|---|
| Raspberry Pi Model 3 B+ | To control the autonomous smart robot surveillance system |
| Ultrasonic sensor | To detect distant object to avoid obstacle |
| Infrared sensor | To detect close object to avoid obstacle |
| Webcam | To allow the autonomous smart robot surveillance system to record and perform face recognition |
| DC Motor | To move the autonomous smart robot surveillance system around |
| Car frame | To hold all the components and enable movement |
| Android Phone | To interact with the autonomous smart surveillance system |
| Computer | To make documents and program the autonomous smart robot surveillance system |

### 3.3.2 Software Requirements

In addition to hardware, the project cannot be completed without the support by the software.

#### 3.3.2.1 Microsoft Word 2016

The documents of this project need a proper way of documenting it. A great tool shortens the time to make the documents. Microsoft Word 2016 helps as a document editor.(Microsoft, 2018b) The latest version of Microsoft word is chosen for its ability to correct grammar mistakes. This allows the documents created using Microsoft Word 2016 to have a low mistakes rate. In conjunction to this, it also contains a wide variety of tools to accelerate the completion of the documents. The high familiarity to the software does contributes to the speed of completing the documents. It has a low cost as UMP provides a free version of it.

Figure 3.31 Icon for Microsoft Word 2016

#### 3.3.2.2 Ninja-IDE v2.3

The Raspberry Pi Model 3 B+ needs to be programmed. Ninja-IDE acts as the IDE for programming the Raspberry Pi Model 3 B+. The IDE is open source IDE specifically designed for python coding. It supports file handling, code search, go to any line, tabs, and auto-completion. It allows plugins to be installed to extend its function. These features allow the coding process to be smooth and it save time.(Ninja-IDE, 2018)

Figure 3.32 Ninja-IDE icon

#### 3.3.2.3 Microsoft Project 2016

To plan the development process properly, a tool is needed to do the planning. Microsoft Project is used to prepare the Gantt Chart. Microsoft Project is flexible and powerful. It is relatively easy to use for designing the Gantt Chart for a single project. It is free because UMP does provide a free soft copy via Microsoft Imagine.(Microsoft, 2018a)

Figure 3.33 Icon for Microsoft Project 2016

#### 3.3.2.4 Android Studio 3.1.4

In order to program the application into an Android device, we need an IDE to program it. Android Studio 3.1.4 is used to program the android application. Android Studio is the official IDE for programming Android devices by Google. Android Studio makes the programming job smooth with a few features. Instant Run features allows the edit, build and run cycles process faster. It also has Intelligent code editor. It helps the programmer to make less mistakes during the process. It also features an emulator. The emulator allows testing the fresh written code without a device. It also can monitor the performance by providing a build-in profiling tool to monitor the app's CPU, memory and network activity.(Developers, 2018)


Figure 3.34 Android Studio 3.1.4

#### 3.3.2.5 Summary of Software Requirements

Table 3.3 Summary of Software Requirements

| Software requirements | Purpose |
|---|---|
| Microsoft Word 2016 | To make documentation |
| Ninja-IDE v2.3 | To program the autonomous smart surveillance system |
| Microsoft Project 2016 | To plan the project and make the Gantt Chart |
| Android Studio 3.1.4 | To program the Android application |

## 3.4 Gantt chart

A Gantt Chart is designed for the development of the project. Refer to Appendix A.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Introduction

In this chapter, the focus will be on implementation and testing result obtained from this project. Implementations that are covered in this chapter includes hardware implementation and software implementation. The testing result of the project will be covering the user acceptance testing.

## 4.2 Implementation

Implementation describes the how is the project executed in both hardware and software aspect of the project. Both hardware and software aspects are required in this project as ASRSS cannot work with just either one. The hardware implementation will be focus on the configuration of the circuit board, sensor and the motor of ASRSS system. The software implementation will cover the interface and the coding of ASRSS application.

### 4.2.1 Hardware Implementation

As mentioned in chapter 3, 3.2.3.3 ASRSS Physical Design, the ASRSS is built according to the sketched design. First attach the 4 motors to the frame of the robot. Then, put on the wheel on the motor. After that attach Raspberry Pi Model 3B+ to the frame. Then attach the extension to the top of raspberry pi model 3B+. Attach 3 IR sensors to the bottom of the frame. After that, attach 2 IR sensors to each side of the frame. Then attach an ultrasonic sensor to the front of the frame. Then, attach the camera to the back of the ultrasonic sensor. Finally connect all the wires to the extension board in the

respective port. The system can be powered up by attaching the battery to the battery holder and switch on the switch on the extension board.

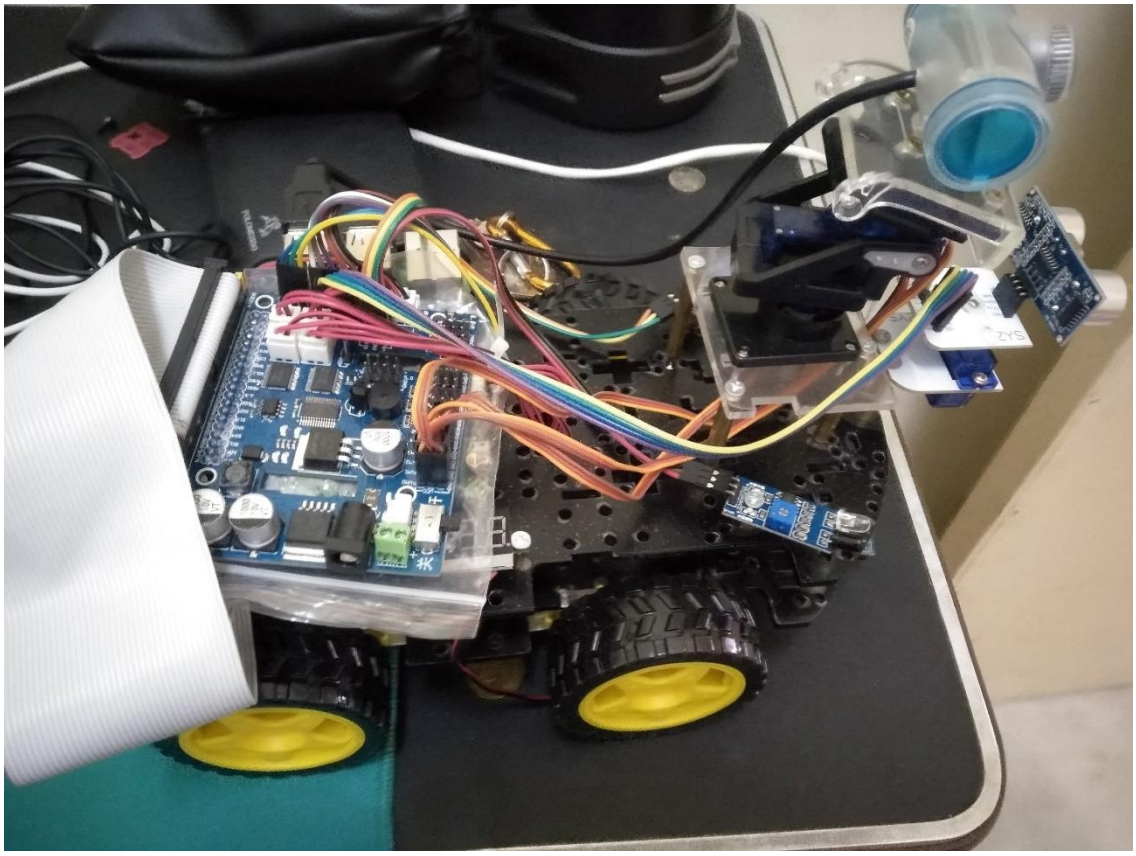


Figure 4.1 ASRSS system hardware configuration

### 4.2.2 Software Implementation

ASRSS system is incomplete without the software. The software implementation mainly consists of ASRSS system and ASRSS application. The application is responsible to display out the saved Google Drive video and receive the notification from the Azure SQL server. The ASRSS system is responsible to use machine vision to detect motions in the environment and record the movements.

#### 4.2.2.1 Home Activity

Home activity is responsible for navigating the user to use desired functionality of the application. In the interface, there is 2 buttons including “View Saved Video”

button and “View Notification” button. The user can navigate to the desired interface by clicking on the button.

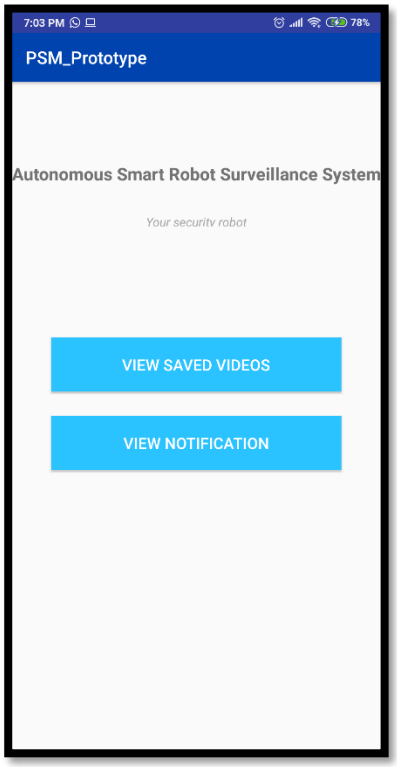



Figure 4.2 Home Activity

#### 4.2.2.2 View Saved Videos Activity

In View Saved Video Activity, the user can view the saved video on Google Drive. The user can click on “Find Video” button to open up gallery application in the user’s Android phone as shown in Figure 4.3. Then the user will be able to choose the targeted video to play as shown in Figure 4.4. After selecting the video, the user can play the video on the View Saved Videos Interface. The user can choose to pause, forward, backward the video as shown in Figure 4.5.

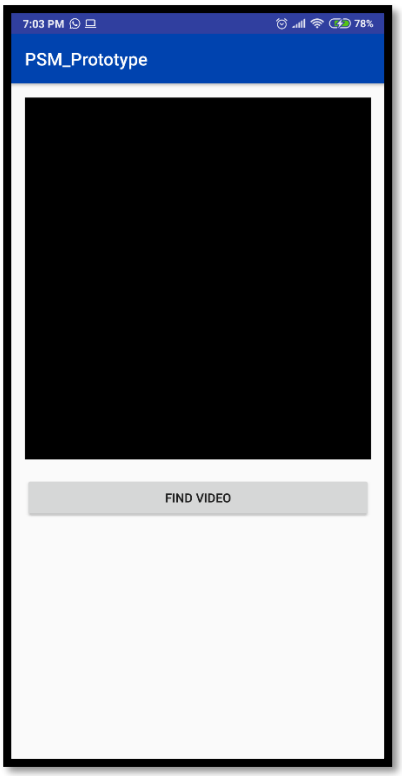



Figure 4.3 View Saved Videos Activity

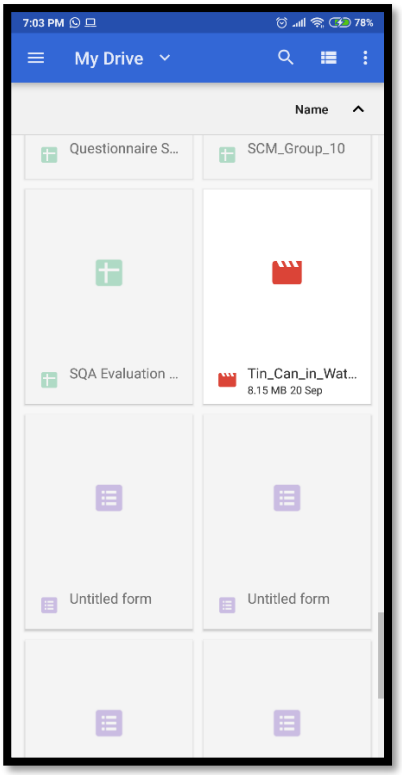



Figure 4.4 View Saved Videos Activity Opening Video

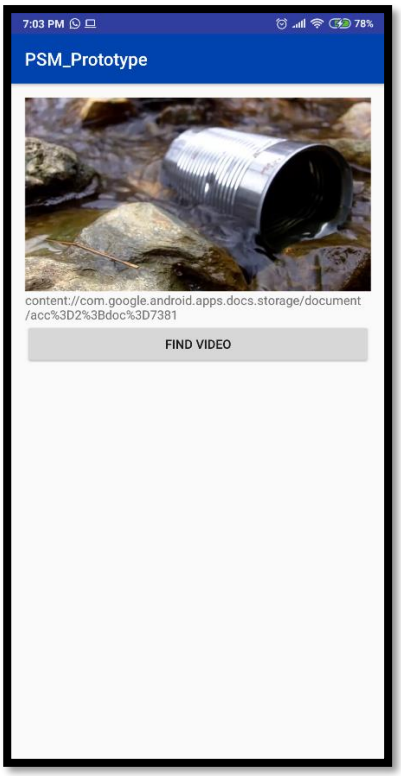



Figure 4.5 View Saved Videos Activity Playing Video

As shown in figure 4.6, there are two main functions are written. The buttonFindVideoOnClickListener function is responsible for opening the video from the gallery. Line 130 to line 132 define an intent to open a document. Line 133 specify that the intent can only open ".mp4" videos. Line 134 to line 137 starts the intent and proceed with opening the gallery app.

The second function which is onActivityResult is responsible to get the URL of the video and play the video. After selecting a video from the gallery application, this function will be called. On line 142 and line 143, there is an if condition to check if the video is successfully retrieved and the request is about opening a video. On line 145, the function gets the URI data from the selected video. On line 148, the video is played.

```
126     View.OnClickListener buttonFindVideoOnClickListener = new View.OnClickListener(){
127
128         @Override
129         public void onClick(View v) {
130             Intent intent = new Intent();
131             intent.setAction(Intent.ACTION_OPEN_DOCUMENT);
132             intent.addCategory(Intent.CATEGORY_OPENABLE);
133             intent.setType("video/mp4");
134             startActivityForResult(
135                     Intent.createChooser(intent,  title: "ACTION_OPEN_DOCUMENT"),
136                     RQS_OPEN_VIDEO);
137         }
138     };
139
140     @Override
141     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
142         if (resultCode == RESULT_OK) {
143             if(requestCode == RQS_OPEN_VIDEO){
144
145                 videoFileUri = data.getData();
146                 info.setText(videoFileUri.toString());
147
148                 prepareVideo();
149             }
150         }
151     }
```

Figure 4.6 Embedded Code for Searching Video and Getting Result from Search

#### 4.2.2.3 View Notification Activity

In View Notification Activity, a list of recycle view cards will be shown to user. As in Figure 4.7, the user can view the notification by reading the available information on the cards. As in Figure 4.8, the user can delete the notification by a long press on the cards. Then the deleted notification ID will be shown via Toast function. The “View Video” button will navigate the user to the “View Saved Video” activity.

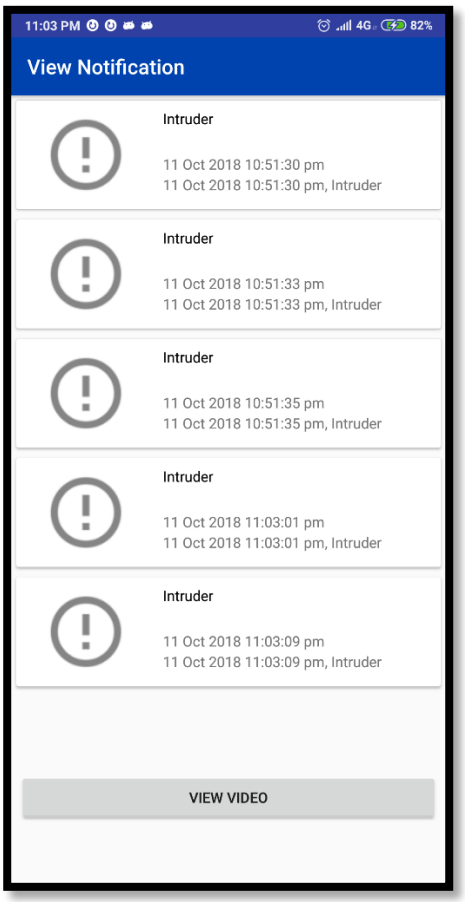



Figure 4.7 View Notification Activity

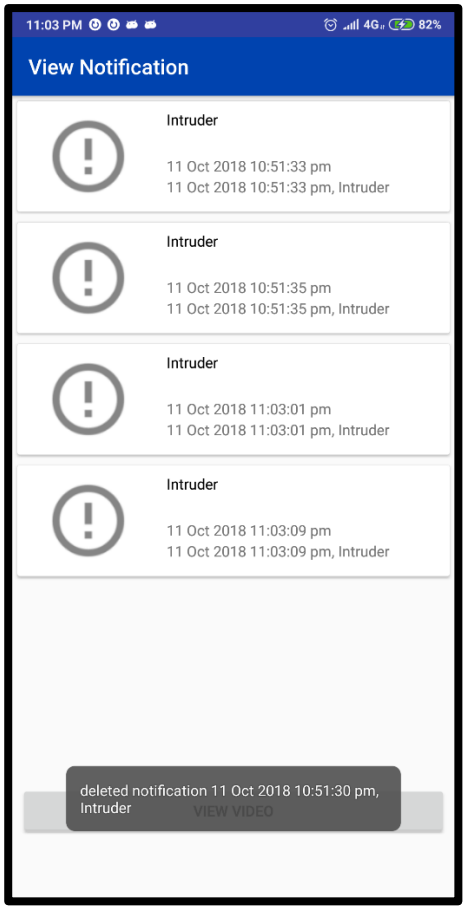



Figure 4.8 View Notification Activity

As shown in Figure 4.8, There are 2 function. The first function is responsible for deleting database. On line 50, the connection to SQLite is established. On line 52, a deletion SQL query is written. Then on line 55 the id value of the notification is inserted into the query. On line 57, the query is executed. On line 58, the connection to the database is closed.

In Figure 4.9, the second function is getData. Its role is to retrieve data from data base and return it as a cursor object. On line 62, the database is loaded from SQLite database. Then on line 63, it is returned as a cursor object with the query of selecting data from the database.

```
49      public  void deleteData(int id) {
50          SQLiteDatabase database = getWritableDatabase();
51
52          String sql = "DELETE FROM NOTIFICATION WHERE id = ?";
53          SQLiteStatement statement = database.compileStatement(sql);
54          statement.clearBindings();
55          statement.bindDouble( index: 1, (double)id);
56
57          statement.execute();
58          database.close();
59      }
60
61      public Cursor getData(String sql){
62          SQLiteDatabase database = getReadableDatabase();
63          return database.rawQuery(sql,  selectionArgs: null);
64      }
```

Figure 4.9 Algorithm for Retrieving and Deleting SQLite Database

In figure 4.10, the refresh function is responsible to retrieve the data from the database and display the newest notification. On line 56, the database data is retrieved and stored as cursor object. After that, on line 57 the recycle view cards are cleared. From line 58 to line 65, the function retrieves the data from the cursor object and load it into the recycle view cards one by one.

```
54      public void refresh()
55      {
56          Cursor cursor = this.sqLiteHelper.getData( sql: "SELECT * FROM NOTIFICATION");
57          notificationList.clear();
58          while (cursor.moveToNext()) {
59              int id = cursor.getInt( i: 0);
60              String content = cursor.getString( i: 1);
61              String time = cursor.getString( i: 2);
62              String videoID = cursor.getString( i: 3);
63
64              notificationList.add(new Notification(id, content, time, videoID));
65          }
66
67      }
```

Figure 4.10 Embedded Code for Refreshing Interface

In Figure 4.11, function is connectionClass. Its role is to connect to Azure SQL database. On line 101 and 102, it is to set the policy of the thread to monitor the processes of connecting to Azure SQL database. For line 108, it states the driver version for the connection. Online 109, the connection URL is stored in a string so that it can be further be used for connecting the server using the provided address, database name, user ID, user password. On line 110, the connection is launched using the connectionURL string.

```
99      public Connection connectionClass()
100     {
101         StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
102         StrictMode.setThreadPolicy(policy);
103         Connection connection = null;
104         String connectionURL = null;
105
106         try
107         {
108             Class.forName("net.sourceforge.jtds.jdbc.Driver");
109             connectionURL = "jdbc:jtds:sqlserver://asrssdb.database.windows.net:1433;DatabaseName=database;user=adminasrss@asrssdb;passw
110             connection = DriverManager.getConnection(connectionURL);
111         }
```

Figure 4.11 Algorithm for Connecting Azure SQL Database

In figure 4.12, the refresh function is responsible to retrieve the data from the Azure SQL database and display the newest notification. On line 65, the database data is connected. After that, on line 72 the query of selecting the data based on the ID is stored. From line 73 to line 74, the function retrieves the data by executing the statement query stored it as result set. From line 75 to line 85, the data is stored in a list of Notification object. The connection is closed on line 86.

```
59  public List<Notification> getAzureSQLNotificationNewData(int notificationID) {
60
61      List<Notification> listNotification = new ArrayList<>();
62
63      try
64      {
65          Connection con = connectionClass();
66          if(con == null)
67          {
68              Log.e( tag: "error", msg: "no internet connection");
69          }
70          else
71          {
72              String query = "SELECT * FROM NOTIFICATION WHERE ID > " + Integer.toString(notificationID);
73              Statement stmt = con.createStatement();
74              ResultSet rs = stmt.executeQuery(query);
75              while (rs.next())
76              {
77                  try
78                  {
79                      listNotification.add(new Notification(rs.getInt( i: 1),rs.getString( i: 2),rs.getString( i: 3),rs.getString( i: 4)));
80                  }
81                  catch(Exception e)
82                  {
83                      Log.e( tag: "error",e.getMessage());
84                  }
85              }
86              con.close();
87          }
88
89      }
```

Figure 4.12 Embedded Code for Retrieving Azure SQL Database

In figure 4.13, the DeleteAzureSQLNotificationData function is responsible to delete a query base on the notificationID. On line 103, the database data is connected. After that, on line 110 the query of deleting the data based on the ID is stored. From line 111 to line 112, the function deletes the data by executing the statement query.

```
98      public void DeleteAzureSQLNotificationData(int notificationID) {
99
100
101         try
102         {
103             Connection con = connectionClass();
104             if(con == null)
105             {
106                 Log.e( tag: "error", msg: "no internet connection");
107             }
108             else
109             {
110                 String query = "DELETE FROM NOTIFICATION WHERE ID = " + Integer.toString(notificationID);
111                 Statement stmt = con.createStatement();
112                 stmt.executeQuery(query);
113                 con.close();
114             }
115
116         }
117         catch (Exception e)
118         {
119             Log.e( tag: "error",e.getMessage());
120         }
```

Figure 4.13 Embedded Code for Deleting Azure SQL Database

#### 4.2.2.4 Notification

In figure 4.14, it shows the notification of the application. It shows the name of the application and the message of the application.

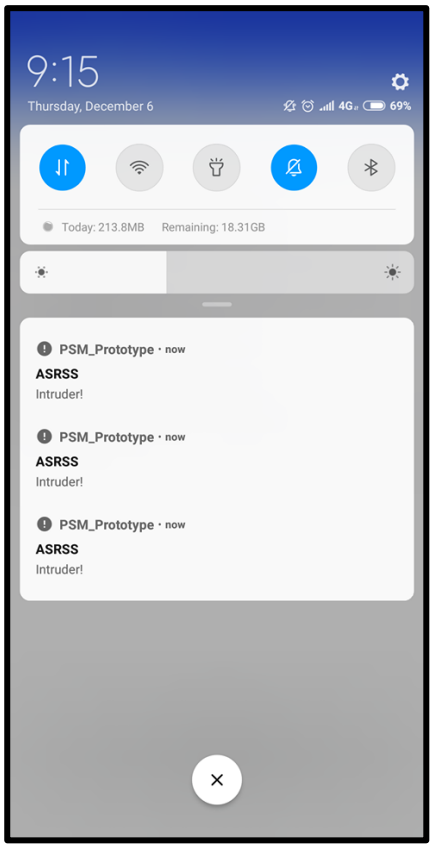
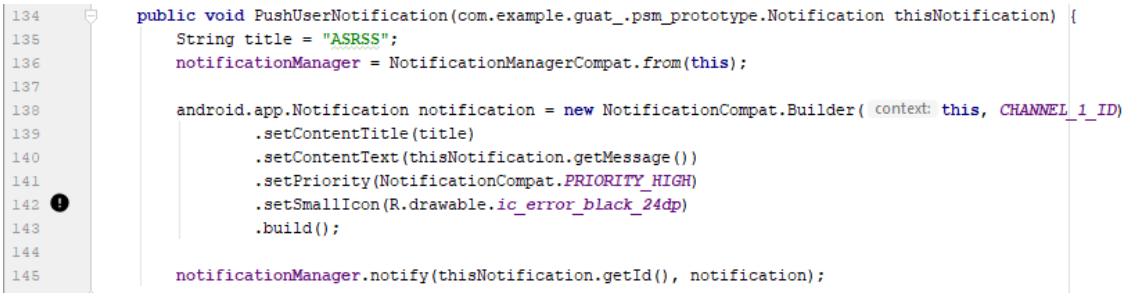



Figure 4.14 Notifications

In figure 4.15, the PushUserNotification function is responsible to display the newest notification. On line 136, the notification manager object is obtained. After that, on line 138 the builder of the notification compact is built with the notification channel of 1. From line 139 to line 142, the title, notification message, priority level, and the icon is set for the notification. The notification is built on line 143. Then on line 145, the notification is shown to the user.



```
134    public void PushUserNotification(com.example.guat_.psm_prototype.Notification thisNotification) {
135        String title = "ASRSS";
136        notificationManager = NotificationManagerCompat.from(this);
137
138        android.app.Notification notification = new NotificationCompat.Builder( context: this, CHANNEL_1_ID)
139                .setContentTitle(title)
140                .setContentText(thisNotification.getMessage())
141                .setPriority(NotificationCompat.PRIORITY_HIGH)
142                .setSmallIcon(R.drawable.ic_error_black_24dp)
143                .build();
144
145        notificationManager.notify(thisNotification.getId(), notification);
```

Figure 4.15 Embedded Code for PushUserNotification

In figure 4.16, the createNotificationChannels function is responsible to a notification channel. From line 18 to line 23, it checks the android version and if it is android O and above, it will assign channel name and the importance level to the channel. On line 24, the description of the channel is assigned to the channel object. On line 26, the manager object gets the service for notification manager. On line 27, the notification channel is created.

```
8   public class App extends Application {
9       public static final String CHANNEL_1_ID = "channel1";
10
11      @Override
12      public void onCreate() {
13          super.onCreate();
14          createNotificationChannels();
15      }
16
17      private void createNotificationChannels() {
18          if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
19              NotificationChannel channel1 = new NotificationChannel(
20                      CHANNEL_1_ID,
21                      name: "Channel 1",
22                      NotificationManager.IMPORTANCE_HIGH
23              );
24              channel1.setDescription("This is Channel 1");
25
26              NotificationManager manager = getSystemService(NotificationManager.class);
27              manager.createNotificationChannel(channel1);
28          }
29      }
30  }
```

Figure 4.16 Embedded Code for Creating Notification Channel

#### 4.2.2.5 Motion Detection

The motion detection works by using machine vision. The algorithm works by spotting the difference the first image with the current image to determine if there are movement present in the image. The first image taken is assumed to be static and there are no people present in the image. The embedded code in Figure 4.17 and Figure 4.18 shows all the implemented algorithms which is mentioned in chapter 3, 3.2.3.2 motion detection algorithm.

As on line 42, first the first image is stored and converted into greyscale. Using GaussianBlur on line 43, we are able to smoothen out the image to remove the noise of the image as the noise of the image can trigger the motion detection. The first image is stored as firstFrame at line 46 to line 48. At line 52, the second image is compared with the first image and the difference is stored. Then the changes are highlighted by discarding values which is lower than 25 on line 53. Then, the image is dilated to fill in the hollow part on line 57, then the contours points are found on line 58 and 59. After that, all points of the contours are looped over to compute the contour box on line 63. On line 65 and 66, if the contour area is smaller than 500 then it is automatically dropped. Lastly the contour box is drawn on the colour image on line 71 and 72.

```
40      # resize the frame, convert it to grayscale, and blur it
41      frame = imutils.resize(frame, width=500)
42      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
43      gray = cv2.GaussianBlur(gray, (21, 21), 0)
44
45      # if the first frame is None, initialize it
46      if firstFrame is None:
47          firstFrame = gray
48          continue
49
50      # compute the absolute difference between the current frame and
51      # first frame
52      frameDelta = cv2.absdiff(firstFrame, gray)
53      thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
54
55      # dilate the thresholded image to fill in holes, then find contours
56      # on thresholded image
57      thresh = cv2.dilate(thresh, None, iterations=2)
58      cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
59          cv2.CHAIN_APPROX_SIMPLE)
60      cnts = cnts[0] if imutils.is_cv2() else cnts[1]
```

Figure 4.17 Embedded Code for Motion Detection

```
    # loop over the contours
    for c in cnts:
        # if the contour is too small, ignore it
        if cv2.contourArea(c) < args["min_area"]:
            continue

        # compute the bounding box for the contour, draw it on the frame,
        # and update the text
        (x, y, w, h) = cv2.boundingRect(c)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        text = "Occupied"

    # draw the text and timestamp on the frame
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

    # show the frame and record if the user presses a key
    cv2.imshow("Security Feed", frame)
    cv2.imshow("Thresh", thresh)
    cv2.imshow("Frame Delta", frameDelta)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key is pressed, break from the lop
    if key == ord("q"):
        break
```

Figure 4.18 Embedded Code for Motion Detection

The Figure 4.19 and Figure 4.20 show how the motion detection works. The left window shows the coloured frame. The middle window shows the highlighted difference of first frame with the current frame. The right window shows the difference of first frame with the current frame in greyscale. In Figure 4.20, the coloured frame shows a green contour box that highlight the difference of the current frame with the first static image. It is able to pin point out where exactly is the moving person.

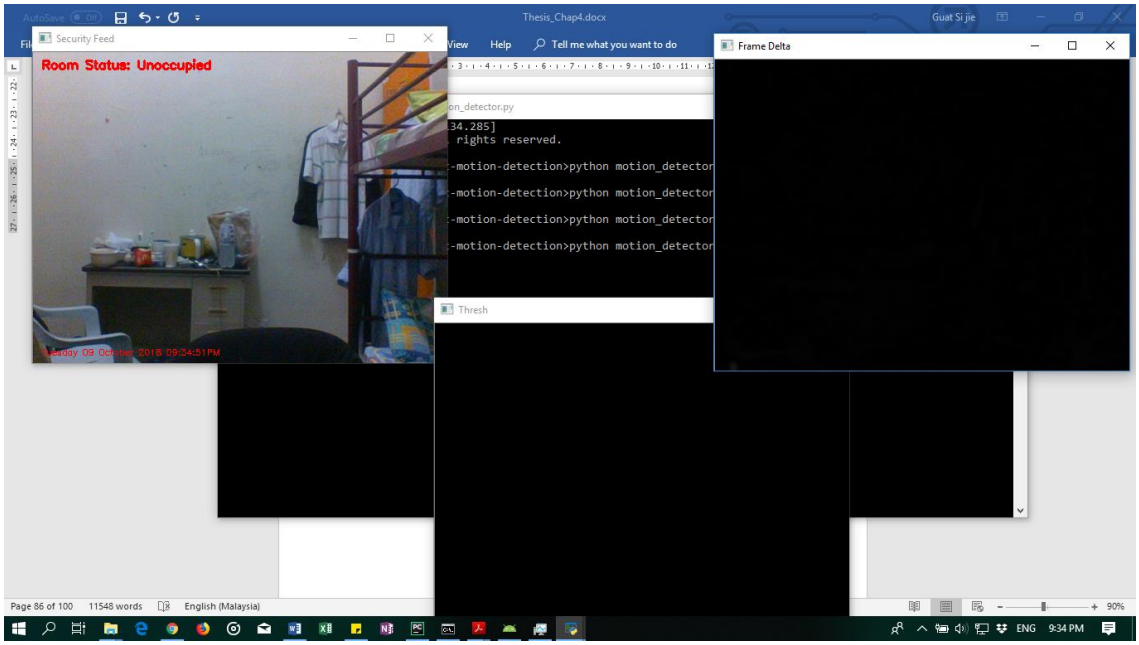



Figure 4.19 Demo for Static Video

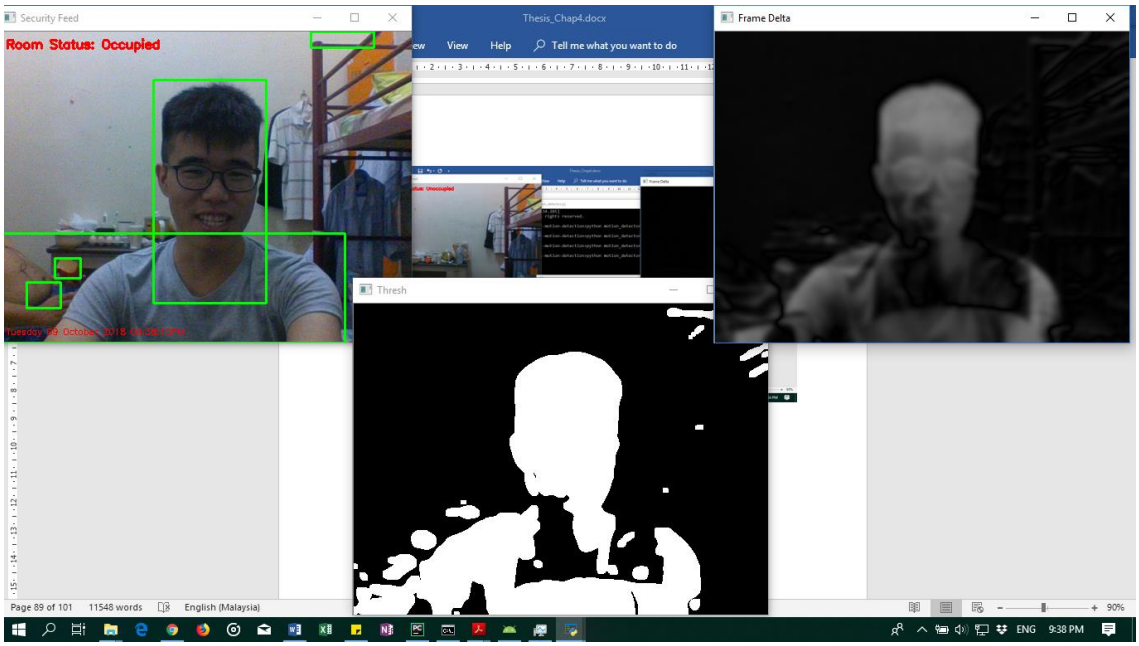



Figure 4.20 Demo for Detecting Moving Objects

#### 4.2.2.6 Static Detection

The static detection works by storing a frame, then compare with the subsequence 9 frames after it. If it is found that the frames are similar to the first frame, then it will determine that it is a static recording and replace the reference frame with the first stored static frame. The embedded code in Figure 4.21 is an implementation of 3.2.3.4 Static Detection algorithm.

```
75  def StaticDetection(self,curFrame):
76
77      if (self.staticDetectionTimer-datetime.now()).total_seconds()<0:
78
79          self.staticDetectionTimer = datetime.now() + timedelta(seconds=5)
80
81          grayFrame = cv2.cvtColor(curFrame, cv2.COLOR_RGB2GRAY)
82          grayFrame = cv2.GaussianBlur(grayFrame, (21, 21), 0)
83
84          self.frameQueue.insert(0,grayFrame)
85
86          if (len(self.frameQueue)>10):
87              self.frameQueue.pop()
88
89          if (len(self.frameQueue) is 10):
90              for i in range(9):
91                  frameDelta = cv2.absdiff(self.frameQueue[i], self.frameQueue[-1])
92                  frameDelta = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
93                  differenceFrame = cv2.countNonZero(frameDelta)
94
95                  if differenceFrame > self.threshold:
96                      return False
97
98              self.referenceFrame = self.frameQueue[-1]
99              return True
100
```

Figure 4.21 Embedded Code for Static Detection

The Figure 4.21 shows how the static detection works. On line 77 to line 79, a 5 sec gap is checked so that the frames will not be too close together in terms of time. On line 81 and line 82, the frame is converted to grey scale and blurred so that the noise of the frame can be reduced. On line 84, the grey scale image is stored to a queue of 10. On line 86 and 87, if the queue is larger than 10, it will deque the oldest frame. From line 90 to 99, all the frames are compared to the first frame. If the difference exceeds the threshold value, it will return false and it is considered that the recording is not static. Else, the reference frame is replaced by the oldest frame from the queue and return a true.

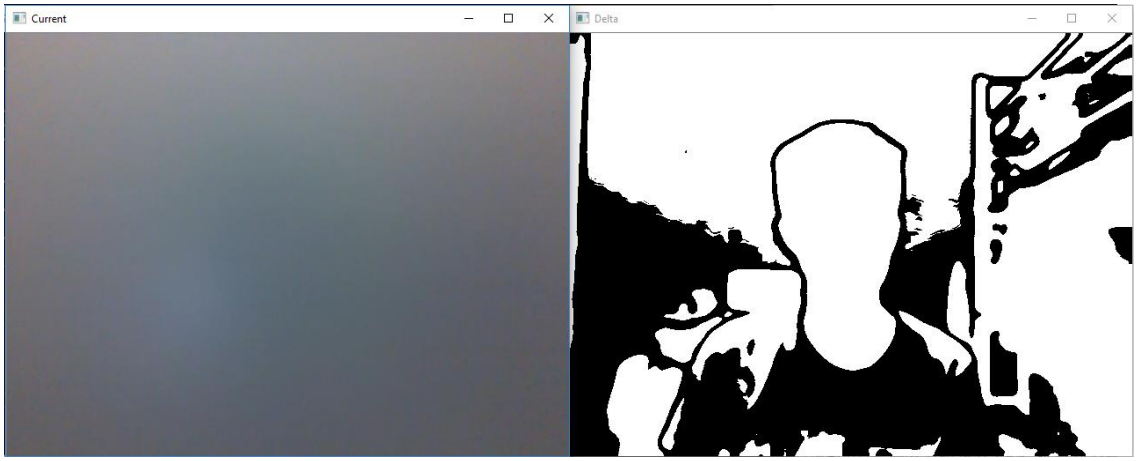



Figure 4.22 Demo for detecting static image

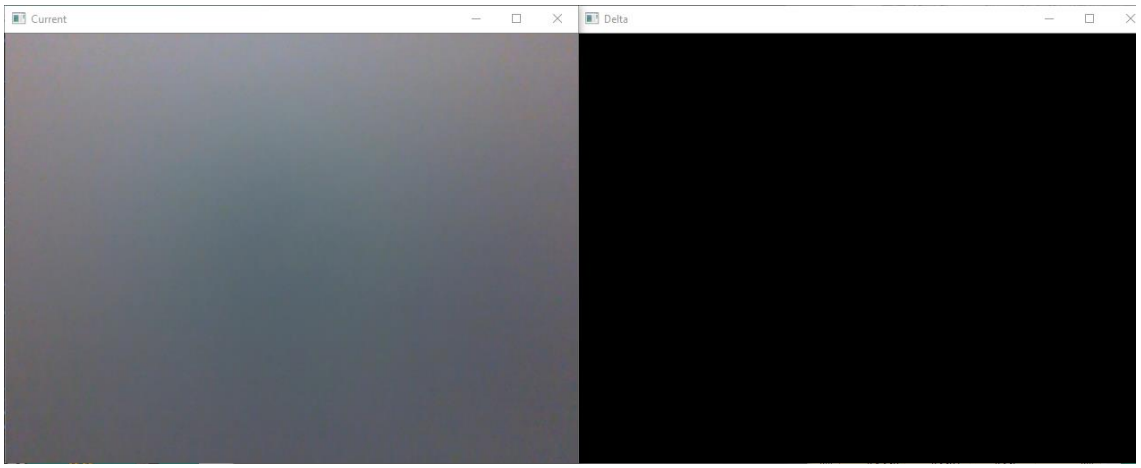



Figure 4.23 Demo for detecting static image after a minute

Figure 4.22 and Figure 4.23 shows how the static detection works. The current window shows normal colour frame and the delta window shows the difference of the reference frame with the current frame in grey scale. Firstly, the camera records a regular footage, then a sticker is applied to the camera to simulate a static recording. After one-minute passes, the delta window shows nothing at all which means that the recording is detected as a static recording. The reference frame is changed to the static frame.

#### 4.2.2.7 Record Video

```
102 def RecordVideo(self,curFrame):
103
104     if(self.recordingStatus is False):
105         self.videoName = datetime.now().strftime('%Y%m%d_%Hh%Mm%Ss%f') + '.mp4'
106         self.out = cv2.VideoWriter('./Video/'+self.videoName, self.fourcc, 16.0, (640, 480))
107         self.recordingStatus = True
108         self.registerNotification.RegisterNewNotification(content='Intruder!',time=datetime.now().strftime('%Y%m%d_%Hh%Mm%Ss%f'),videoID=self.videoName)
109
110     self.out.write(curFrame)
```

Figure 4.24 Embedded Code for Record Video

The Figure 4.24 show how the video is recorded. On line 104, the recording status is checked so that it can identify a new recording. Once the recording is confirmed to be a new recording, the name of the video is named after the time taken and the format of the video is set to MP4 on line 105 and 106. The recording status is set to true on line 107. Then, a notification is registered on line 108. On line 110, the video is written to the sd card.

#### 4.2.2.8 Register Notification

The register notification function registers the notification information into the SQLite database first. Then get the id for the notification from the SQLite database and insert the notification to the Azure SQL database.

```
 8     def RegisterNewNotification(self, content='',time='',videoID=''):
 9         notification = Notification(content=content, time=time, videoID=videoID)
10
11         self.notificationSQLite.InsertNotification(notification)
12         result = self.notificationSQLite.GetNotificationID(time)
13         notification.id = result[0][0]
14
15         self.notificationAzureSQL.InsertAzureSQLNotification(notification)
```

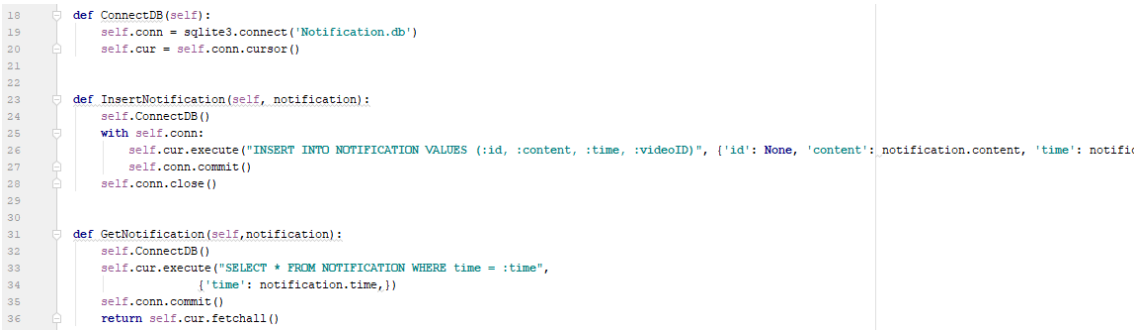
Figure 4.25 Embedded Code for Register Notification

The Figure 4.25 show how the register notification works. Firstly, it takes all the information passed to store it in a Notification object. From line 11 to line 13, the notification is stored in SQLite and retrieved the assigned notification id from SQLite. On line 15, it is stored in Azure SQL database.

```
15  def ConnectAzureSQLDB(self):
16      self.cnxn = pyodbc.connect(
17          'DRIVER=' + self.driver + ';SERVER=' + self.server + ';PORT=1433;DATABASE=' + self.database + ';UID=' + self.username + ';PWD=' + self.password)
18      self.cur = self.cnxn.cursor()
19
20
21  def InsertAzureSQLNotification(self, notification):
22      self.ConnectAzureSQLDB()
23      with self.cnxn:
24          self.cur.execute("SET IDENTITY_INSERT NOTIFICATION ON INSERT INTO NOTIFICATION(ID, Message, Time, VideoID) VALUES (?, ?, ?, ?)",
25                           notification.id, notification.content, notification.time, notification.videoID)
26          self.cnxn.commit()
27      self.cnxn.close()
```

Figure 4.26 Embedded Code for Connecting and Inserting Data to Azure SQL Database

The Figure 4.26 shows how to connect and insert data to Azure SQL Database. On line 16 to line 18, the Azure SQL Database is connected, and the cursor is stored. On line 22, the Azure SQL Database is stored. From 23 to line 26. The query is executed with the values stored in Notification object. Then on line 27, the connection is closed.



```
18  def ConnectDB(self):
19      self.conn = sqlite3.connect('Notification.db')
20      self.cur = self.conn.cursor()
21
22
23  def InsertNotification(self, notification):
24      self.ConnectDB()
25      with self.conn:
26          self.cur.execute("INSERT INTO NOTIFICATION VALUES (:id, :content, :time, :videoID)", {'id': None, 'content': notification.content, 'time': notifi
27          self.conn.commit()
28      self.conn.close()
29
30
31  def GetNotification(self,notification):
32      self.ConnectDB()
33      self.cur.execute("SELECT * FROM NOTIFICATION WHERE time = :time",
34                       {'time': notification.time,})
35      self.conn.commit()
36      return self.cur.fetchall()
```

Figure 4.27 Embedded Code for Connecting, Inserting Data and Getting Data from SQLite

The Figure 4.27 shows how to connect, insert data and getting data from SQLite. On line 19 and 20, the SQLite database is connected, and the cursor object is stored. On line 24, the database is connected. Frome line 25 to line 27, the notification data is inserted to the SQLite database. The connection is closed on line 28. On line 32, the database is connected. Frome line 33 to line 35, the notification data is fetched from the SQLite database. The fetched data is returned in line 36.

#### 4.2.2.9 Hardware Failure

The hardware failure covers the checking of internet connection and camera readiness. By using the OpenCV library we are able to check the status of the camera via isOpened() function. The internet connection is checked by accessing Google.

```
10  def CheckCam(self):
11      cap = cv2.VideoCapture(0)
12
13      if cap.isOpened():
14          cap.release()
15          cv2.destroyAllWindows()
16          return True
17      else:
18          cap.release()
19          cv2.destroyAllWindows()
20          self.registerNotification.RegisterNewNotification(content='Camera Fault',time=datetime.now().strftime('%Y%m%d_%Hh%Mm%Ss%f'))
21          return False
22
23
24  def CheckInternet(self):
25      try:
26          urllib.urlopen('http://216.58.192.142', timeout=1) #connect google
27          return True
28      except urllib.URLError as err:
29          return False
```

Figure 4.28 Embedded Code for Detect Camera Failure and Internet Connection

The Figure 4.28 show how the detection of camera failure and internet connection work. On line 11, the camera is targeted for capture of image. On line 13, the camera is check if it works. On line 14 to line 16, the cap object is release and the windows are all destroyed. Then return true as the camera is working. On line 18 to line 19, the cap object is release and the windows are all destroyed. On line 20, a notification is registered. On line 21, return false as the camera is not working. On line 28, the google is connected for testing purpose. On line 27 if it is connected, it will return true. Else, on line 29 it will return false.

#### 4.2.2.10 Swap Spot

The swap spot function works by reading the guiding tiles on the floor. It will start moving forward as long as it sense that the tile below is black in colour as in Figure 4.29 until all the three infrared sensors sense a white surface underneath the robot as in Figure 4.30. It means that it is the end of the path. After that, if the function is called again, it will move backward until the end of the path as shown in Figure 4.31.

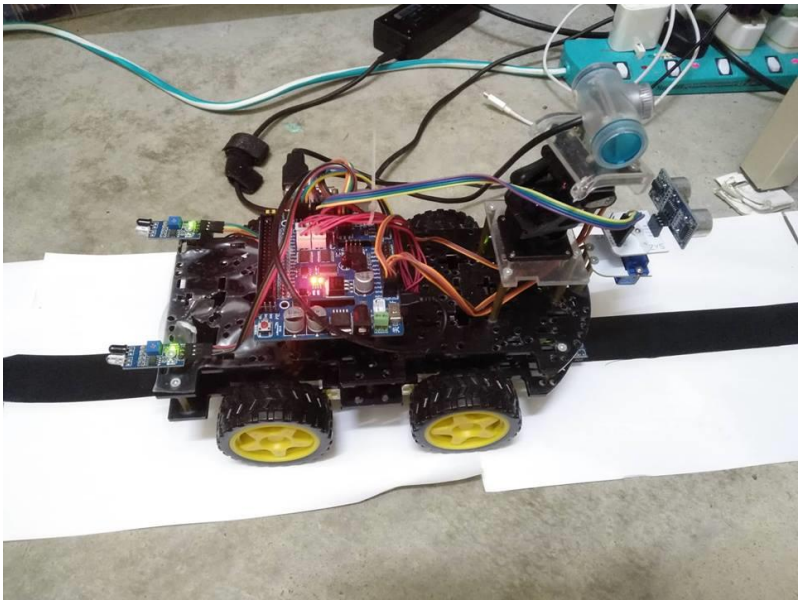


Figure 4.29 Demo for Swapping Spot

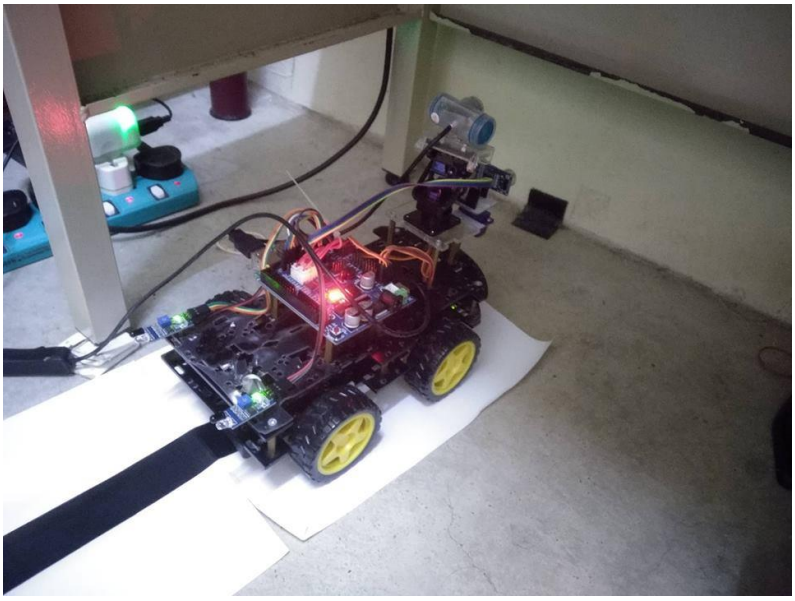


Figure 4.30 Demo for Swapping Spot

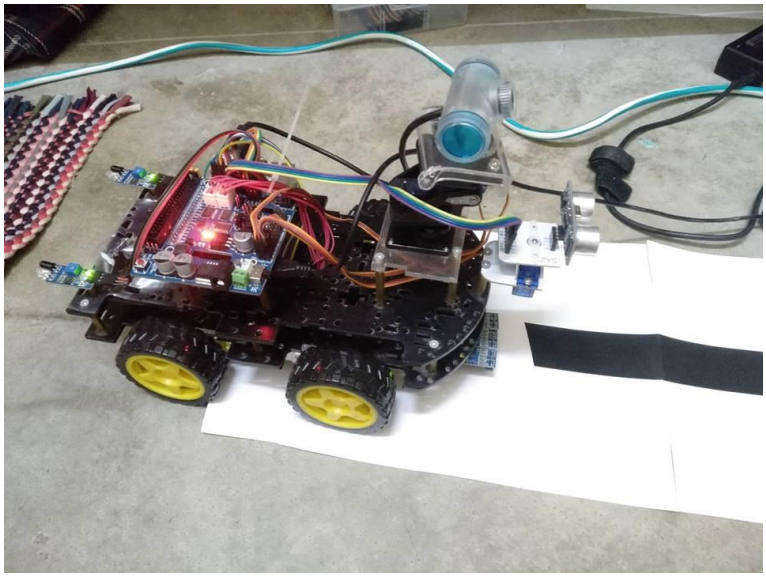


Figure 4.31 Demo for Swapping Spot

The hardware failure covers the checking of internet connection and camera readiness. By using the OpenCV library we are able to check the status of the camera via isOpened() function. The internet connection is checked by accessing Google.

```
17      def SwapSpot(self):
18          if self.location is 'front':
19              self.MoveBackwards()
20              self.location = 'back'
21          else:
22              self.MoveForward()
23              self.location = 'front'
24
25
26      def MoveForward(self):
27          self.motor.MoveForward(100, t_time=0.4)
28
29          while (self.infrared.LBTMInfra() is 1 or self.infrared.RBTMInfra() is 1 or self.infrared.MBTMInfra() is 1):
30              self.motor.MoveForward(100)
31              time.sleep(0.1)
32
33          self.motor.Stop()
34
35      def MoveBackwards(self):
36          self.motor.MoveBackward(100, t_time=0.4)
37
38          while (self.infrared.LBTMInfra() is 1 or self.infrared.RBTMInfra() is 1 or self.infrared.MBTMInfra() is 1):
39              self.motor.MoveBackward(100)
40              time.sleep(0.1)
41
42          self.motor.Stop()
```

Figure 4.32 Embedded Code for Swap Spot

The Figure 4.32 show the logic for swapping the surveillance spot. On line 18, it checks if the location is at front or back. On line 19 to line 20, it moves backwards if

the location is at front and then registered itself at the back. On line 22 to line 23, it moves forward if the location is at back and then registered itself at the front. On line 27, it moves forward for 0.4 second then it starts to detect the guiding tiles on line 29. On line 30 and 31, it moves forward for 0.1 second at a time. Finally, it stops on line 33. On line 36, it moves backward for 0.4 second then it starts to detect the guiding tiles on line 38. On line 39 and 40, it moves backward for 0.1 second at a time. Finally, it stops on line 42.

#### 4.2.2.11 Avoiding Object

The avoiding object function works by detecting the front with ultrasonic sensor and the back via infrared sensor. Then an object is detected by the ultrasonic at the front as in Figure 4.33, it will move backward for a second as in Figure 4.34. It is set to detect a distance of less of 6 cm so that the object is not too close to the robot. Then it will move back to the original spot as shown in Figure 4.35.

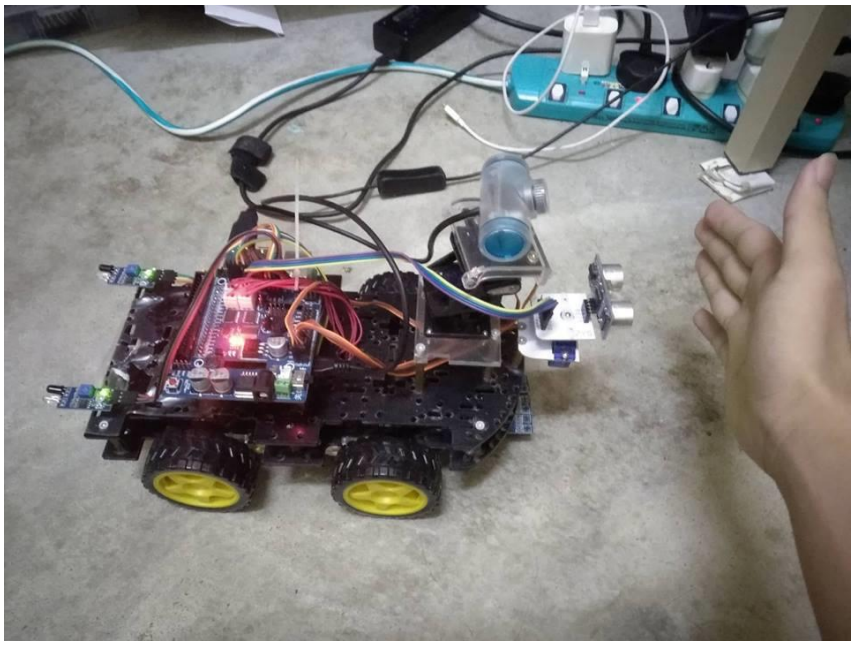


Figure 4.33 Demo for Avoiding Object and Detecting Object in Front

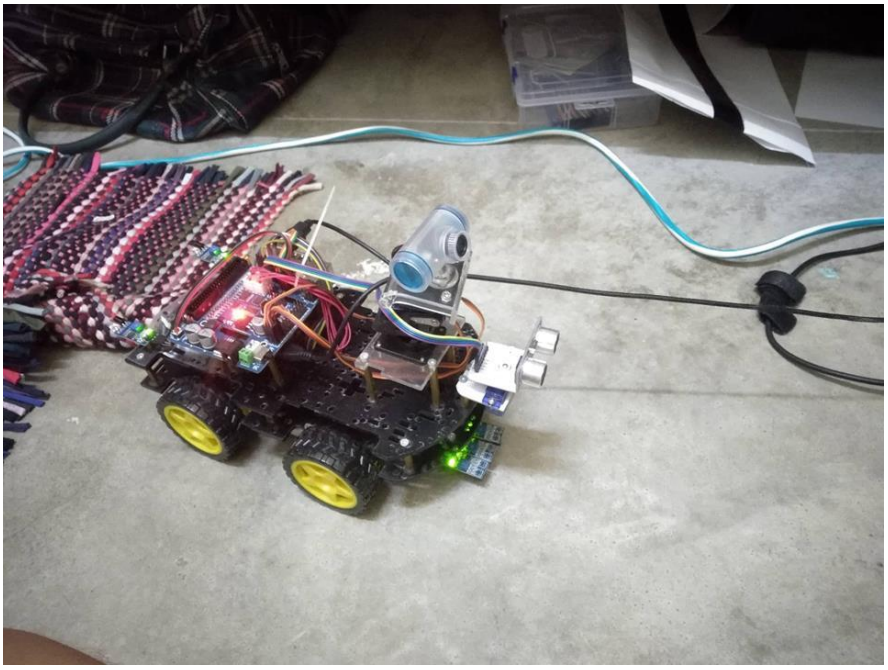


Figure 4.34 Demo for Avoiding Object and Detecting Object in Front

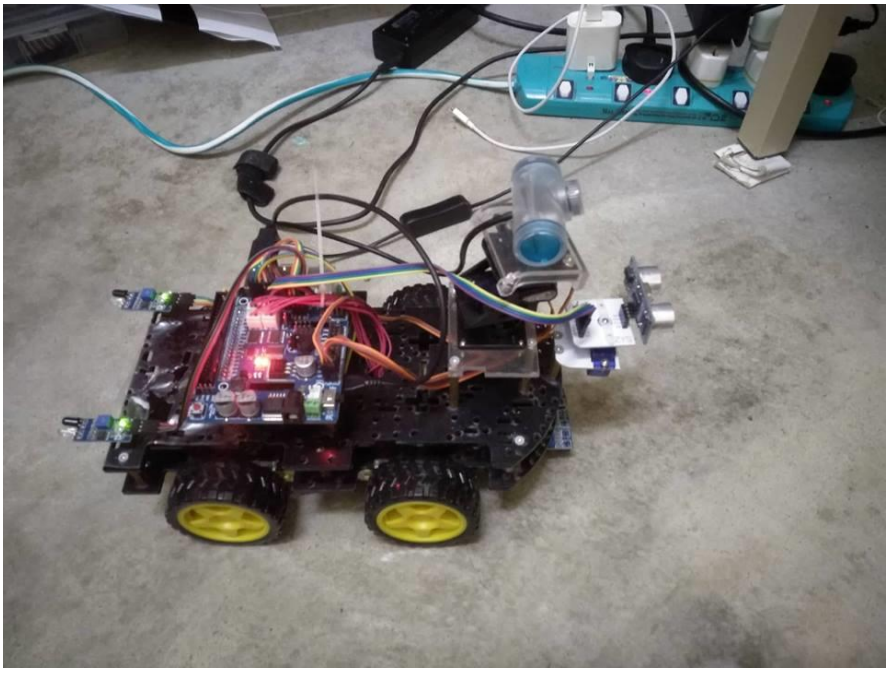


Figure 4.35 Demo for Avoiding Object and Detecting Object in Front

If an object is detected at the back with the infrared sensor as shown in Figure 4.36, it will move forward for a second as shown in Figure 4.37. The infrared sensor is configured to detect anything near 6 cm distance from the sensor so that the object will not be too close to the robot. Then, it will move back to the original spot as shown in Figure 4.38.

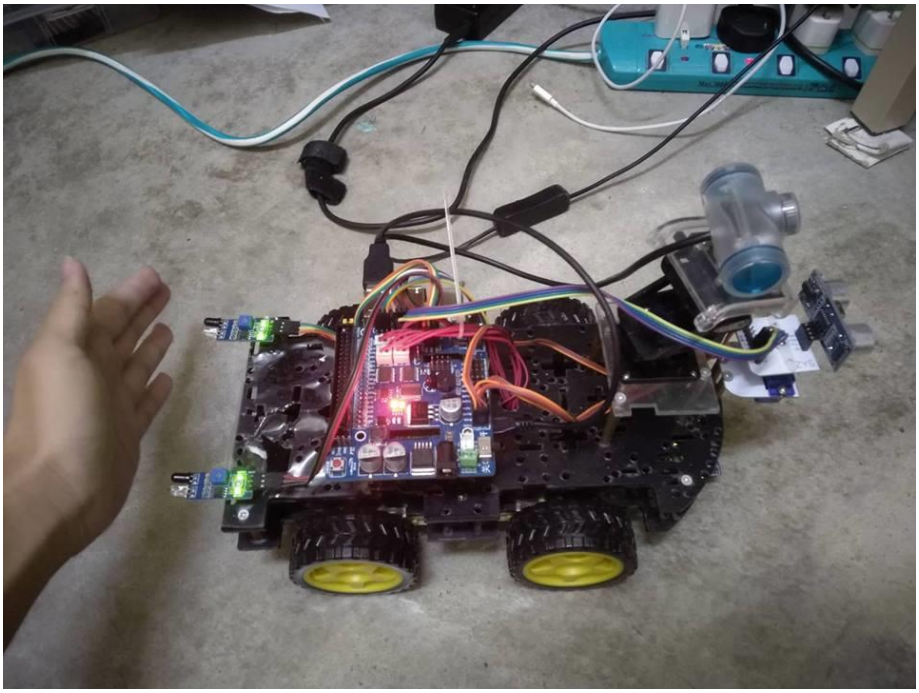


Figure 4.36 Demo for Avoiding Object and Detecting Object Behind

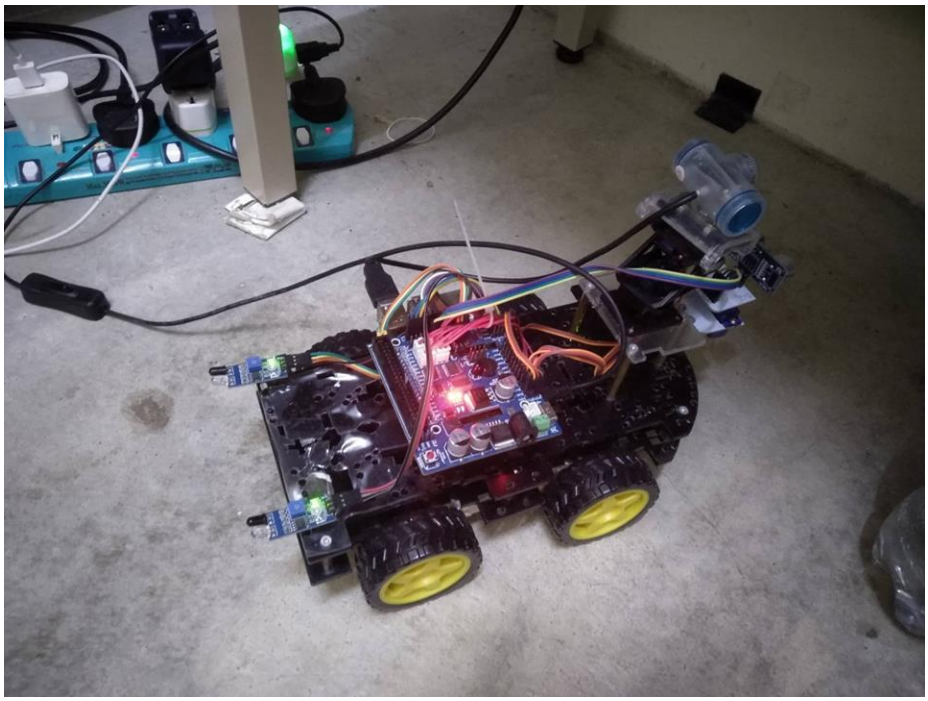


Figure 4.37 Demo for Avoiding Object and Detecting Object Behind

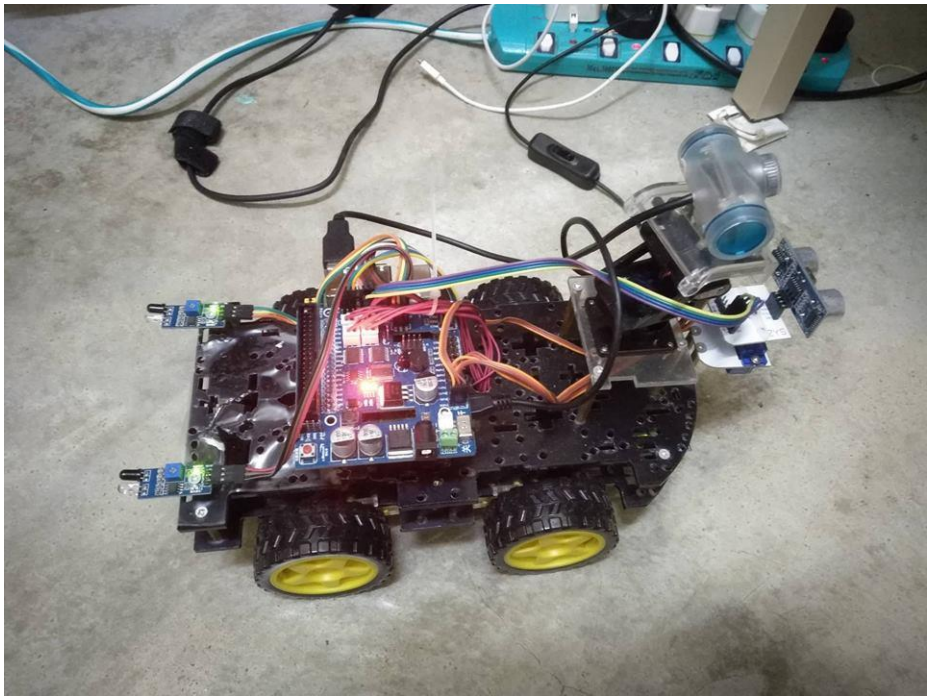


Figure 4.38 Demo for Avoiding Object and Detecting Object Behind

```
45      def AvoidObject(self):
46
47          self.DetectObject()
48          while -0.1<self.movementCorrection>0.1:
49              self.DetectObject()
50              if self.movementCorrection>0:
51                  self.movementCorrection -= 0.1
52                  self.motor.MoveBackward(100, t_time=0.1)
53              else:
54                  self.movementCorrection += 0.1
55                  self.motor.MoveForward(100, t_time=0.1)
56          self.motor.Stop()
57
58
59      def DetectObject(self):
60
61          self.motor.write(0,90)
62          if self.ultrasonic.USDistance() < 6:
63              self.motor.MoveBackward(100,t_time=1)
64              self.movementCorrection += 1
65
66          elif self.infrared.LBackInfra() is 0 or self.infrared.RBackInfra() is 0:
67              self.motor.MoveForward(100, t_time=1)
68              self.movementCorrection -= 1
```

Figure 4.39 Embedded Code for Avoiding Object and Detecting Object

The Figure 4.39 shows how the avoiding object and detecting object work. First, it detects the object on line 47. In DetectObject function, on line 61, it turns the camera to the middle at 90 degrees. On line 62, it detects the distance of the ultrasonic sensor's distance. If it is lesser than 6 cm, it will move backward on line 63 and the correction distance is recorded on line 64. On line 66, it detects the infrared sensor's distance. If it is lesser than 6 cm, it will move forward on line 67 and the correction distance is recorded on line 68. Then from line 48 to line 55, the corrective distance will be carried out. On line 49 if object is detected during the corrective movement, it will avoid the object. Else, it will smoothly carry out the corrective movement with an interval of 0.1 second.

#### 4.2.2.12 Infrared Sensor

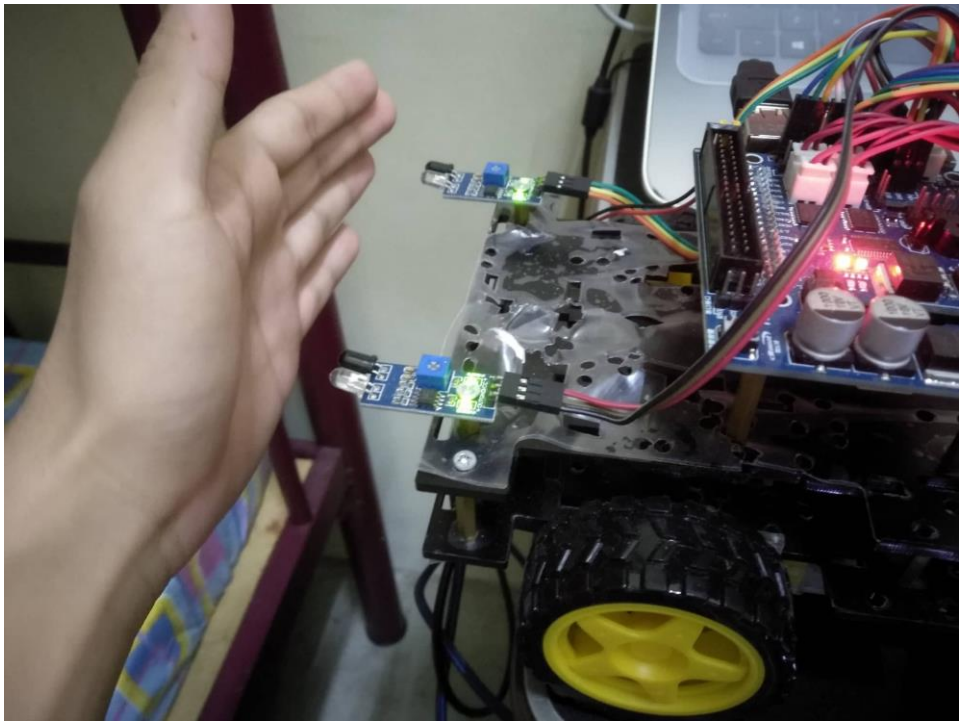


Figure 4.40 Testing the Infrared Sensor with an Object

```
21      def setup(self):
22          GPIO.setwarnings(False)
23          GPIO.setmode(GPIO.BCM)  # Numbers GPIOs by physical location
24          GPIO.setup(self.Gpin, GPIO.OUT)  # Set Green Led Pin mode to output
25          GPIO.setup(self.Rpin, GPIO.OUT)  # Set Red Led Pin mode to output
26          GPIO.setup(self.BTMSensorMid, GPIO.IN)  # Set BtnPin's mode is input, and pull up to high level(3.3V)
27          GPIO.setup(self.BTMSensorRight, GPIO.IN)
28          GPIO.setup(self.BTMSensorLeft, GPIO.IN)
29          GPIO.setup(self.BackSensorRight, GPIO.IN)
30          GPIO.setup(self.BackSensorLeft, GPIO.IN)
31
32      def LBTMInfra(self):
33          return GPIO.input(self.BTMSensorLeft)
34
35      def RBTMInfra(self):
36          return GPIO.input(self.BTMSensorRight)
37
38      def MBTMInfra(self):
39          return GPIO.input(self.BTMSensorMid)
40
41      def LBackInfra(self):
42          return GPIO.input(self.BackSensorLeft)
43
44      def RBackInfra(self):
45          return GPIO.input(self.BackSensorRight)
46
```

Figure 4.41 Embedded Code for Infrared Sensor

Figure 4.40 shows how the infrared sensor works. When an object is placed within 6 cm from the sensor, the second LED will light up indicating that the infrared sensor has sensed an obstacle. Figure 4.41 shows the embedded code for infrared sensor. the setup function mainly is responsible for setting up the GPIOs. On line 23, the mode of GPIO is set to BCM for the numbering of the GPIO. From line 26 to line 30, the GPIO of each infrared sensor are set as GPIO input. From line 32 to line 45 states all the infrared sensors. It returns 0 as no detection and 1 as detected something.

#### 4.2.2.13 Ultrasonic Sensor

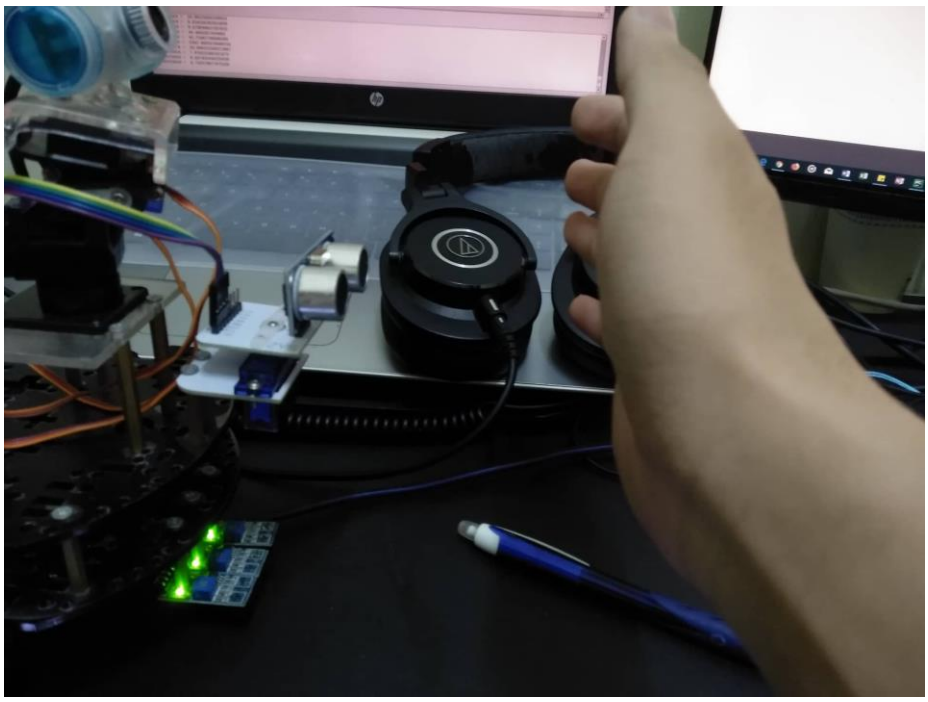


Figure 4.42 Testing for Ultrasonic Sensor

```
Shell
  distance =  8.085966110229492
  distance =  8.138656616210938
  distance =  2343.191385269165
  distance =  44.83151435852051
  distance =  42.71578788757324
  distance =  43.9600944519043
  distance =  42.241573333740234
  distance =  43.51019859313965
  distance =  42.594194412231445

>>>
```

Figure 4.43 Distance Result of the Ultrasonic Sensor

```
20  def setup(self):
21      GPIO.setwarnings(False)
22      GPIO.setmode(GPIO.BCM)
23      GPIO.setup(self.TRIG, GPIO.OUT)
24      GPIO.setup(self.ECHO, GPIO.IN)
25
26      GPIO.setup(self.Gpin, GPIO.OUT)  # Set Green Led Pin mode to output
27      GPIO.setup(self.Rpin, GPIO.OUT)  # Set Red Led Pin mode to output
28      GPIO.setup(self.BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Set BtnPin's mode is input, and pull up to high level(3.3V)
29
30
31  def USDistance(self):
32      GPIO.output(self.TRIG, 0)
33      time.sleep(0.000002)
34
35      GPIO.output(self.TRIG, 1)
36      time.sleep(0.00001)
37      GPIO.output(self.TRIG, 0)
38
39      while GPIO.input(self.ECHO) == 0:
40          a = 0
41      beforeTime = time.time()
42      while GPIO.input(self.ECHO) == 1:
43          a = 1
44      afterTime = time.time()
45
46      during = afterTime - beforeTime
47      cmDistance = during * 340 / 2 * 100
48
49      print('distance = ',cmDistance)
50      return cmDistance
51
```

Figure 4.44 Embedded Code for Ultrasonic Sensor

Figure 4.42 shows how the ultrasonic sensor works and the result of detected distance in Figure 4.43. The ultrasonic sensor works by sending an ultrasonic wave, then wait for the reflecting echo ultrasonic wave. The lag time is calculated by time difference * 340/2*100 where 340 is the speed of sound, 100 is the conversion to cm and 2 is for the distance of traveling back and forth. The Figure 4.43 shows the result of looping the ultrasonic sensor forever. The distance in this figure shows the distance in cm unit. The Figure 4.44 shows the embedded code for ultrasonic sensor which is an implementation of 3.2.3.3 Ultrasonic Distance Detection Algorithm.

The setup function maps the GPIO. Line 22 set the GPIO to BCM. Then, the GPIO of trigger and echo is set on line 23 and line 24. The USDistance function is responsible for detecting the distance. On line 32, the trigger is set to 0 to ensure that the ultrasonic wave is not sensed. On line 35, an ultrasonic wave is sent. On line 37, the ultrasonic wave sending is stopped. On line 35 waits the wave is sent and on line 41, the before time is recorded. On line 42, the wave is waited until received then on line 44, the after time is recorded. On line 46, the time lag is calculated by subtracting before time and after time. Then it is used to multiply 340 divided by 2 and again multiplied by 100. Then the distance is calculated and return.

## 4.3 Testing Result

Testing is carried out with user acceptance testing. Please refer to appendix f for the user acceptance testing report.

## 4.4 User Manual

Please refer to appendix g for the user manual.

# CHAPTER 5

# CONCLUSION

## 5.1 Introduction

In this chapter, the conclusion of this project will be concluded. The ASRSS system is able to detect nearby obstacles and avoid it. It also has the ability to move from a point to another. The ASRSS system is able to send notification via Azure SQL database with the help of Android ASRSS App. The advantages of this project are that the robot can work fully autonomously, record only moving objects and the video can be accessed from anywhere from Google Drive.

The main objective that are achieved are as follows:

i. To build a smart surveillance system that detects nearby obstacles.

ii. To build a raspberry pi system that can move from a point to another.

iii. To apply the feature of sending notification and video to a phone.

## 5.2 Constraint

During the development of ASRSS system, there are a few limitations that are found.

### 5.2.1 Power Limitation

The two 2280 3.7V batteries do not provide enough power for remote. The power provided is enough for non-motion related function. Once movement is attempted, the whole system got unstable and reboot. Hence, a wired power supply is needed for the operation of the ASRSS.

### 5.2.2 Motor Power Limitation

The four motors are not strong enough to allow the ASRSS to turn left or turn right. It is due to the resistance built up by turning two motors opposite to the other two motors. Hence, ASRSS can only move forward and backward.

### 5.2.3 Port 1433 Blocked

Port 1433 is blocked on UMP network. The port is used for communicating with Azure SQL database. As the result personal mobile data must be used to access the database.

### 5.2.4 Android Drive API Methods Depreciated

Some of the critical methods of the Drive API are depreciated. That includes accessing the files directly from Google Drive which is not available on the newest Drive API. As the result, Drive API is replaced with using an intent to Google Drive to access the video file.

### 5.2.5 Virtual Environment in Raspberry Pi Does Not Support I2C

I2C helps the raspberry pi to connect to the Adafruit motor driver and control the motors. The virtual environment does not support the I2C and all the dependencies and libraries must be installed in the raspberry pi environment instead. As the result, the time is wasted to reinstall all the libraries and dependencies on the Raspberry Pi system.

## 5.3 Future Improvements

The system can be improved by improving the battery power supply which will allows it to work wirelessly. The more powerful battery will allow the system to turn left and turn right making it more flexible in terms of movement. A GPS module and xxx can be used to ensure that the accuracy of the placement of the ASRSS which will allow the system to navigate better in the given indoor environment.

## 5.4 Conclusion

Autonomous Smart Robot Surveillance System was designed for indoor usage of surveying an area. The project was developed for Mr. Law which focus on monitoring the home using the ASRSS and the ASRSS application. Although there is some limitation to the system, all three objectives that were stated in Chapter 1 have been achieved. Future work also has been discussed to improve the system.

## REFERENCES


Academy, C. M. R. (n.d.). What Is an Ultrasonic Sensor? Retrieved from http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html

Academy, C. M. R. (2018). What is an IR Sensor? Retrieved from What is an IR Sensor?

Arduino. (2018a). Arduino Uno Rev3. Retrieved from https://store.arduino.cc/usa/arduino-uno-rev3

Arduino. (2018b). Download the Arduino IDE. Retrieved from https://www.arduino.cc/en/Main/Software

Beagleboard.org. (2018). BeagleBone Black. Retrieved from https://beagleboard.org/black

Borong, H. (2018). GOQ Q7 Robot Magnetic Wifi Surveillance IP Security Camera CCTV Cam 960P HD. Retrieved from https://www.lazada.com.my/products/goq-q7-robot-magnetic-wifi-surveillance-ip-security-camera-cctv-cam-960p-hd-i125945584-s138649433.html?spm=a2o4k.searchlist.list.3.38a1d7abfME3b2&search=1

Brandon, J. (2014). 5 Uses for the Surveillance Robot of Tomorrow. Retrieved from https://www.cio.com/article/2463903/robotics/5-uses-for-the-surveillance-robot-of-tomorrow.html

Codecedemy. (2018). MVC: Model, View, Controller. Retrieved from https://www.codecademy.com/articles/mvc

Coşkun, M., & Ünal, S. (2016). Implementation of Tracking of a Moving Object Based on Camshift Approach with a UAV. *Procedia Technology*, *22*(October 2015), 556–561. https://doi.org/10.1016/j.protcy.2016.01.116

DAYTECH. (2018), DAYTECH HD720P/HD960P CCTV IP Camera WiFi Surveillance Network Security Camera Waterproof Outdoor Two Way Audio Night Vision. (2018). Retrieved from https://www.lazada.com.my/products/daytech-hd720phd960p-cctv-ip-camera-wifi-surveillance-network-security-camera-waterproof-outdoor-two-way-audio-night-vision-i231719854-s307284688.html?spm=a2o4k.searchlist.list.7.dd7d17f2U9djQw&search=1

Developers. (2018). androidstudio. Retrieved from https://developer.android.com/studio/

ESCAM. (2018). ESCAM QN02. Retrieved from http://www.escam.cn/product/showproduct.php?lang=en&id=99

Existek. (2017). Sdlc Models Explained: Agile, Waterfall, V-shaped, Iterative, Spiral | Existek Blog. Retrieved from https://existek.com/blog/sdlc-models/

Foundation, R. P. (2018). Raspberry Pi Downloads - Software For the Raspberry Pi. Retrieved from https://www.raspberrypi.org/downloads/

GSMArena. (2017). Xiaomi Mi A1. Retrieved from https://www.gsmarena.com/xiaomi_mi_a1-8776.php

Layton, J. (2018). How Webcams Work. Retrieved from https://computer.howstuffworks.com/webcam1.htm

McManus, S. (2018). Top 10 Programming Languages Ported To the Raspberry Pi. Retrieved from http://www.dummies.com/computers/raspberry-pi/top-10-programming-languages-ported-to-the-raspberry-pi/

Microsoft. (2018a). Project. Retrieved from https://products.office.com/en-my/project/project-and-portfolio-management-software?tab=tabs-1

Microsoft. (2018b). Word 2016. Retrieved from https://products.office.com/en-my/word

Moore, M. (2018). What is Industry 4.0? Everything you need to know. Retrieved from https://www.techradar.com/news/what-is-industry-40-everything-you-need-to-know

Nayyar, A. (2017). Top 8 Ides For Raspberry Pi. Retrieved from http://opensourceforu.com/2017/06/top-ides-raspberry-pi/

Ninja-IDE. (2018). Ninja-IDE. Retrieved from http://ninja-ide.org/

Richardson, M. (2012). Get Started with BeagleBone. Retrieved from https://makezine.com/projects/make-32/get-started-with-beaglebone/

Sami, M. (2012). Software Development Life Cycle Models and Methodologies. Retrieved from https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

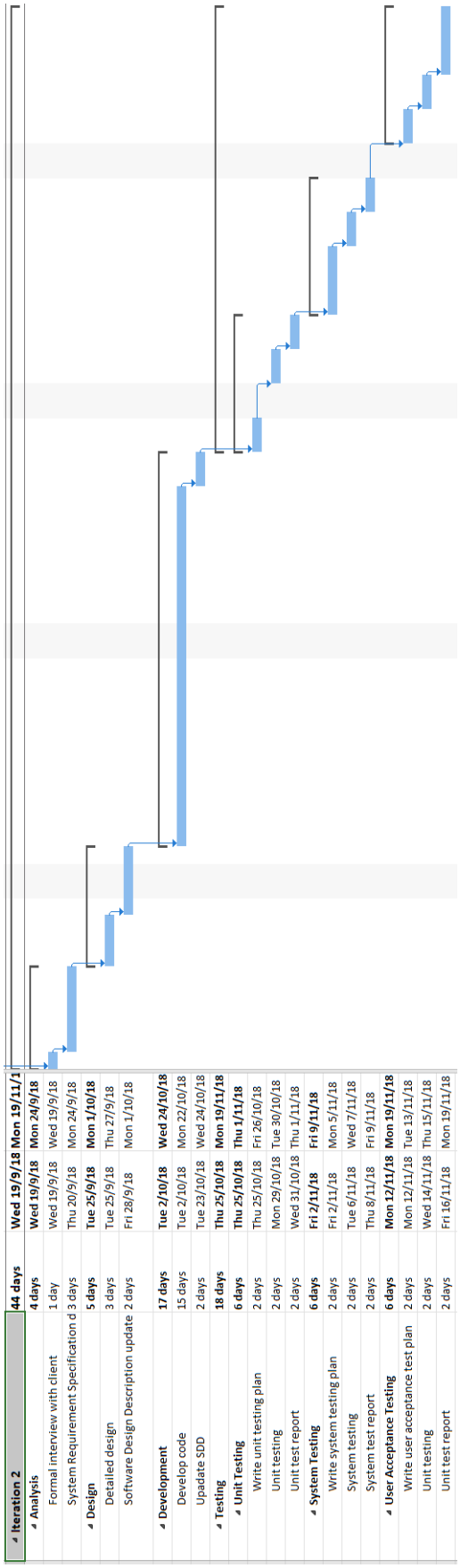
Tseng, C. C., Lin, C. L., Shih, B. Y., & Chen, C. Y. (2013). SIP-enabled Surveillance Patrol Robot. *Robotics and Computer-Integrated Manufacturing*, *29*(2), 394–399. https://doi.org/10.1016/j.rcim.2012.09.009

Watson, J. A. (2017). Hands-on with the Raspberry Pi 3 Model B+. Retrieved from http://www.zdnet.com/article/hands-on-raspberry-pi-3-model-b/

**APPENDIX A**
**GANTT CHART**

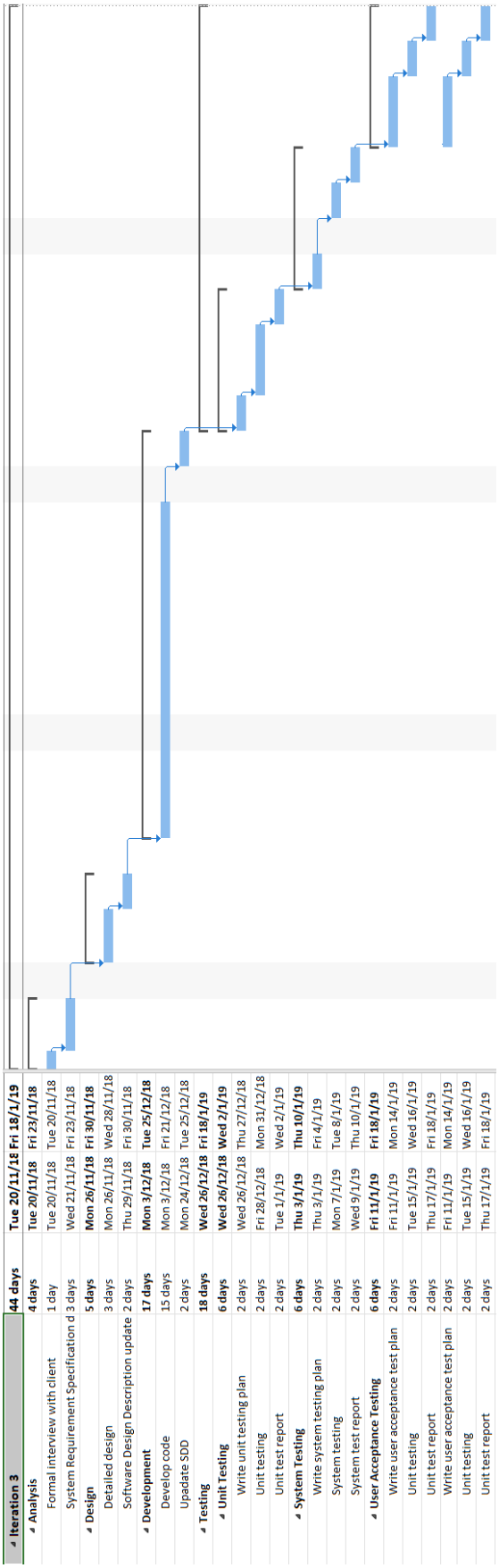
| | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 0 | **PSM project** | **230 days** | **Mon 5/3/18** | **Fri 18/1/19** |
| 1 | **Iteration 1** | **142 days** | **Mon 5/3/18** | **Tue 18/9/18** |
| 2 | **Planning** | **2 days** | **Mon 5/3/18** | **Tue 6/3/18** |
| 3 | Determine project scope | 1 day | Mon 5/3/18 | Mon 5/3/18 |
| 4 | Plan Gantt Chart | 1 day | Tue 6/3/18 | Tue 6/3/18 |
| 5 | **Analysis** | **39 days** | **Wed 7/3/18** | **Mon 30/4/18** |
| 6 | Hardware literature review | 2 days | Wed 7/3/18 | Thu 8/3/18 |
| 7 | Technology literature review | 2 days | Fri 9/3/18 | Mon 12/3/18 |
| 8 | Existing system review | 2 days | Tue 13/3/18 | Wed 14/3/18 |
| 9 | Methodology model analysis | 2 days | Thu 15/3/18 | Fri 16/3/18 |
| 10 | Update Gantt Chart | 1 day | Mon 19/3/18 | Mon 19/3/18 |
| 11 | Mid Term Break | 9 days | Tue 20/3/18 | Fri 30/3/18 |
| 12 | Analyse hardware requriements | 3 days | Mon 2/4/18 | Wed 4/4/18 |
| 13 | Analyse software requirements | 4 days | Thu 5/4/18 | Tue 10/4/18 |
| 14 | Analyse architecture pattern | 3 days | Wed 11/4/18 | Fri 13/4/18 |
| 15 | Casual interview with client | 1 day | Wed 18/4/18 | Wed 18/4/18 |
| 16 | System Requirement Specification documentation | 5 days | Thu 19/4/18 | Wed 25/4/18 |
| 17 | Design questionnaire | 1 day | Thu 26/4/18 | Thu 26/4/18 |
| 18 | Formal interview with client | 1 day | Fri 27/4/18 | Fri 27/4/18 |
| 19 | System Requirement Specification documentation update | 1 day | Mon 30/4/18 | Mon 30/4/18 |
| 20 | **Design** | **8 days** | **Tue 1/5/18** | **Thu 10/5/18** |
| 21 | Finalise software design of ASRSS application | 1 day | Tue 1/5/18 | Tue 1/5/18 |
| 22 | Software Design Description | 1 day | Wed 2/5/18 | Wed 2/5/18 |
| 23 | Detailed design | 6 days | Thu 3/5/18 | Thu 10/5/18 |
| 24 | **Development** | **20 days** | **Tue 15/5/18** | **Mon 11/6/18** |
| 25 | Develop code | 15 days | Tue 15/5/18 | Mon 4/6/18 |
| 26 | Upadate SDD | 5 days | Tue 5/6/18 | Mon 11/6/18 |
| 27 | **Testing** | **71 days** | **Tue 12/6/18** | **Tue 18/9/18** |
| 28 | **Unit Testing** | **10 days** | **Tue 12/6/18** | **Mon 25/6/18** |
| 32 | **System Testing** | **10 days** | **Tue 26/6/18** | **Mon 9/7/18** |
| 33 | Write system testing plan | 4 days | Tue 26/6/18 | Fri 29/6/18 |
| 34 | System testing | 3 days | Mon 2/7/18 | Wed 4/7/18 |
| 35 | System test report | 3 days | Thu 5/7/18 | Mon 9/7/18 |
| 36 | **User Acceptance Testing** | **51 days** | **Tue 10/7/18** | **Tue 18/9/18** |
| 37 | Write user acceptance test plan | 4 days | Tue 10/7/18 | Fri 13/7/18 |
| 38 | Sem break | 39 days | Wed 18/7/18 | Mon 10/9/18 |
| 39 | Unit testing | 3 days | Tue 11/9/18 | Thu 13/9/18 |
| 40 | Unit test report | 3 days | Fri 14/9/18 | Tue 18/9/18 |

Appendix A-1 Gantt Chart of Iteration 1



| Iteration 2 | 44 days | Wed 19/9/18 | Mon 19/11/1 |
|---|---|---|---|
| Analysis | 4 days | Wed 19/9/18 | Mon 24/9/18 |
| Formal interview with client | 1 day | Wed 19/9/18 | Wed 19/9/18 |
| System Requirement Specification d | 3 days | Thu 20/9/18 | Mon 24/9/18 |
| Design | 5 days | Tue 25/9/18 | Mon 1/10/18 |
| Detailed design | 3 days | Tue 25/9/18 | Thu 27/9/18 |
| Software Design Description update | 2 days | Fri 28/9/18 | Mon 1/10/18 |
| Development | 17 days | Tue 2/10/18 | Wed 24/10/18 |
| Develop code | 15 days | Tue 2/10/18 | Mon 22/10/18 |
| Upadate SDD | 2 days | Tue 23/10/18 | Wed 24/10/18 |
| Testing | 18 days | Thu 25/10/18 | Mon 19/11/18 |
| Unit Testing | 6 days | Thu 25/10/18 | Thu 1/11/18 |
| Write unit testing plan | 2 days | Thu 25/10/18 | Fri 26/10/18 |
| Unit testing | 2 days | Mon 29/10/18 | Tue 30/10/18 |
| Unit test report | 2 days | Wed 31/10/18 | Thu 1/11/18 |
| System Testing | 6 days | Fri 2/11/18 | Fri 9/11/18 |
| Write system testing plan | 2 days | Fri 2/11/18 | Mon 5/11/18 |
| System testing | 2 days | Tue 6/11/18 | Wed 7/11/18 |
| System test report | 2 days | Thu 8/11/18 | Fri 9/11/18 |
| User Acceptance Testing | 6 days | Mon 12/11/18 | Mon 19/11/18 |
| Write user acceptance test plan | 2 days | Mon 12/11/18 | Tue 13/11/18 |
| Unit testing | 2 days | Wed 14/11/18 | Thu 15/11/18 |
| Unit test report | 2 days | Fri 16/11/18 | Mon 19/11/18 |

Appendix A-2 Gantt Chart of Iteration 2



| Iteration 3 | 44 days | Tue 20/11/18 | Fri 18/1/19 |
|---|---|---|---|
| Analysis | 4 days | Tue 20/11/18 | Fri 23/11/18 |
| Formal interview with client | 1 day | Tue 20/11/18 | Tue 20/11/18 |
| System Requirement Specification d | 3 days | Wed 21/11/18 | Fri 23/11/18 |
| Design | 5 days | Mon 26/11/18 | Fri 30/11/18 |
| Detailed design | 3 days | Mon 26/11/18 | Wed 28/11/18 |
| Software Design Description update | 2 days | Thu 29/11/18 | Fri 30/11/18 |
| Development | 17 days | Mon 3/12/18 | Tue 25/12/18 |
| Develop code | 15 days | Mon 3/12/18 | Fri 21/12/18 |
| Upadate SDD | 2 days | Mon 24/12/18 | Tue 25/12/18 |
| Testing | 18 days | Wed 26/12/18 | Fri 18/1/19 |
| Unit Testing | 6 days | Wed 26/12/18 | Wed 2/1/19 |
| Write unit testing plan | 2 days | Wed 26/12/18 | Thu 27/12/18 |
| Unit testing | 2 days | Fri 28/12/18 | Mon 31/12/18 |
| Unit test report | 2 days | Tue 1/1/19 | Wed 2/1/19 |
| System Testing | 6 days | Thu 3/1/19 | Thu 10/1/19 |
| Write system testing plan | 2 days | Thu 3/1/19 | Fri 4/1/19 |
| System testing | 2 days | Mon 7/1/19 | Tue 8/1/19 |
| System test report | 2 days | Wed 9/1/19 | Thu 10/1/19 |
| User Acceptance Testing | 6 days | Fri 11/1/19 | Fri 18/1/19 |
| Write user acceptance test plan | 2 days | Fri 11/1/19 | Mon 14/1/19 |
| Unit testing | 2 days | Tue 15/1/19 | Wed 16/1/19 |
| Unit test report | 2 days | Thu 17/1/19 | Fri 18/1/19 |
| Write user acceptance test plan | 2 days | Fri 11/1/19 | Mon 14/1/19 |
| Unit testing | 2 days | Tue 15/1/19 | Wed 16/1/19 |
| Unit test report | 2 days | Thu 17/1/19 | Fri 18/1/19 |

Appendix A-3 Gantt Chart of Iteration 3

# APPENDIX B
# SOFTWARE REQUIREMENT SPECIFICATION (SRS)

# APPENDIX C
# SOFTWARE DESIGN DESCRIPTION (SDD)

# APPENDIX D
# FUNCTIONAL REQUIREMENTS GATHERING

## APPENDIX E
## FSKKP INFORMATION GATHERING APPROVAL LETTER

## APPENDIX F
## USER ACCEPTANCE TEST REPORT

**APPENDIX G**
**USER MANUAL**

**APPENDIX H**
**TURNITIN REPORT**