**Faculty of Electrical and Electronic Engineering Technology**

# DEVELOPMENT OF CRANE UTILIZATION SYSTEM BASED ON IOT

**NG WEN JUN**

**Bachelor of Computer Engineering Technology (Computer Systems) with Honours**

**2021**

# DEVELOPMENT OF CRANE UTILIZATION SYSTEM BASED ON IOT

**NG WEN JUN**

**A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Computer Engineering Technology (Computer Systems) with Honours**

**Faculty of Electrical and Electronic Engineering Technology**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2021**

**DECLARATION**

I declare that this project report entitled "DEVELOPMENT OF CRANE UTILIZATION SYSTEM BASED ON IOT" is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Student Name | : | NG WEN JUN |
| Date | : | 03/02/2022 |

# APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Engineering Technology (Computer Systems) with Honours.

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Supervisor Name | : | TS. GLORIA RAYMOND TANNY |
| Date | : | 03 FEBRUARI 2022 |
| Signature | : | [illegible] |
| Co-Supervisor Name (if any) | : | NORLEZAH BINTI HASHIM |
| Date | : | 03 FEBRUARI 2022 |

## DEDICATION

*To my beloved mother, Choong Chooi Peng, and father, Ng Chee Chong,*

*Thank you for supporting me when I decide to resign from the work in Singapore and return to Malaysia for completing a degree in UTeM.*

*To Yayasan Tenaga Nasional(TNB)*

*Thank you for giving me a chance to pursue my education toward the goal of achieving a better, brighter future.*

## ABSTRACT

Crane utilization rate and health is an important thing for a company to keep tracking. The condition of the crane cannot be decided by just looking at it, because it is always too late when the failure of the internal part of the crane was found. The abrupt accidents will interrupt or delay the work progress and also cause the cost of the work to increase. Due to that, a monitoring system will be needed. But most of the current monitoring systems are using wireless communication technology like BLUETOOTH and WIFI. This communication technique is good but still can be improved because the effective range for both of these communication technologies is limited. Due to that, LoRa is chosen in this project because of long-range of detections from LoRa is suitable for monitoring machines that require large space especially in the heavy engineering industry. For this crane monitoring system, some data will be transferred and the size for this data is not big. So that, the advantages for Lora to send and receive data through a long-range and the disadvantages of LoRa that only able to send data at a low rate is acceptable. Hence, the LoRa module is used in this project. This project will have a transmitter side that an Arduino UNO place at the crane part using some sensors to track the health of the crane by measuring temperature, power consumption, lifting activity, and working hours. Then the data will be sent to the receiver side which is a NodeMCU ESP8266. The data will be sent to the database if the NodeMCU is connected to the internet. In the end, the user will be able to do the monitoring for the crane by using the mobile application. This project aims to design and develop a crane monitoring system using an android application and it had been successfully done. The result shows that users will be able to monitor the crane utilization by using the Android application. In addition, the user will be able to view the real-time data or past data in graph view. For future works, a controlling system is possible to add and implement in this system so the approved user or admin can use the mobile application to stop the crane when they had found something wrong with the crane's data.

## *ABSTRAK*

Kadar penggunaan dan kesihatan kren adalah satu perkara penting bagi syarikat untuk memberi perhatian. Keadaan kren tidak dapat diputuskan dengan hanya melihat rupanya, ini kerana ketika kegagalan bahagian dalaman kren dijumpai maka sudah terlambat. Kemalangan yang berlaku secara tiba-tiba akan mengganggu atau melambatkan kerja dan juga menyebabkan kos meningkat. Oleh kerana itu, sistem pemantauan akan diperlukan. Tetapi kebanyakan sistem pemantauan semasa menggunakan teknologi komunikasi tanpa wayar seperti BLUETOOTH dan WIFI. Teknik komunikasi ini bagus tetapi masih boleh diperbaiki kerana jarak efektif untuk kedua-dua teknologi komunikasi ini adalah terhad. Disebabkan itu, LoRa telah dipilih dalam projek ini kerana kemampuan LoRa untuk dikesan dalam jarak yang jauh sesuai digunakan untuk pemantauan mesin yang memerlukan ruang yang besar terutamanya dalam industri kejuruteraan berat. Untuk sistem pemantauan kren ini, beberapa data akan dihantar dan terima tetapi saiz data ini tidak berapa besar. Oleh itu, kelebihan Lora untuk menghantar dan menerima data melalui jarak jauh dan kelemahan LoRa yang hanya dapat menghantar data dalam kadar rendah dapat diterima. Justeru itu, modul LoRa akan digunakan dalam projek ini. Projek ini akan mempunyai bahagian pemancar yang ditempatkan dengan Arduino UNO di bahagian kren menggunakan beberapa sensor untuk mengesan kesihatan kren dengan mengukur suhu, penggunaan kuasa, aktiviti mengangkat dan waktu kerja. Kemudian data akan dihantar ke sisi penerima yang merupakan NodeMCU ESP8266. Data akan dihantar ke pangkalan data jika NodeMCU disambungkan ke internet. Pada akhirnya, pengguna akan dapat melakukan pemantauan untuk kren dengan menggunakan aplikasi mudah alih. Projek ini bertujuan untuk merancang dan mengembangkan sistem pemantauan kren menggunakan aplikasi android dan` ia telah berjaya dicapai. Hasil menunjukkan bahawa pengguna boleh memantau data masa nyata atau data lepas dalam paparan graf. Bagi kerja masa depan, satu sistem kawalan boleh ditambah dan dilaksanakan dalam sistem ini supaya pengguna yang telah diluluskan atau admin boleh menggunakan aplikasi mudah alih untuk menghentikan kren apabila mereka telah mendapati sesuatu yang tidak kena dengan data kren.

# ACKNOWLEDGEMENTS

To complete this project, many people had helped and inspired me. I have received a lot of support from each of them.

First, I would like to thank my supervisor, TS. GLORIA RAYMOND TANNY, and co-supervisor, NORLEZAH BINTI HASHIM that help me to research this project. Without their support and guidance, I feel like I will not be able to do this project.

Then, I would also want to thank my classmates from BEEC for assisting me and giving me support when I need it. They are never stingy to share their knowledge, information, and experience with me.

Last but not least, I would like to thank my family that always encouraged me during the time I am researching to complete this project.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | | |
|---|---|---|
| LoRa | - | Long Range |
| RF | - | Radio Frequency |
| GPS | - | Global Positioning System |
| RTC | - | Real-Time Clock |
| IoT | - | Internet of Things |
| GSM | - | Global System for Mobile Communications |
| GPRS | - | General Packet Radio Service |
| LCD | - | Liquid Crystal Display |
| $V$ | - | Voltage |
| $A$ | - | Current |
| $W$ | - | Watt |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

In this chapter, the project's detail will be explained including the used method and way to implement the project. This is the first chapter for this thesis and will be started with the project background, problem statement, objective project scope, and finally the thesis outline.

## 1.2 Project Background

This project is created based on the idea to propose and build a crane monitoring system that can monitor and inspect the crane's health and utilization of the crane at any time by using the mobile application. When the utilization percentage of the crane and the health status of the crane is known, then future management, maintenance, and planning will be able to complete and executed to prevent an unexpected event like crane shutting down or accidents that will cause the delay to the working schedule.

When an unexpected problem occurs before we found out, many issues will appear. Therefore, a crane monitoring system should be proposed to resolve this problem. The monitoring system should be able to show the condition of the crane so that the user will be able to notice the condition of the crane when it is bad.

## 1.3 Project Statement

The condition of the crane will directly affect the chance for an accident to happen and also the cost to maintain the crane. This is because the cranes are usually exposed to harsh environments and always operate at high duty cycles. So that, the issues of the cranes should be identified before it occurs to prevent tragedy(Rahim Abdul Hamid *et al.*, 2019).

To solve this problem, a monitoring system will be needed. This monitoring system should be able to monitor four basic information which is the running hours, the lifting activity(tonnage), the power consumption, and the temperature of the motor or hydraulic or other similar things.

The running hours of a crane will tell the user how long the crane had been operated and should be stopped after a fixed operating time. This action can help to extend the lifespan of the crane and also make the maintenance schedule work easier.

Besides that, the lifting activity which shows the weight the crane is lifting allows the user to monitor and control the maximum capacity that a crane can lift. Whenever a crane is overloaded, it causes irreversible damage and may cause hazards. The most common case will be the drop of the load or crane is swinging and hard to be controlled.

Lastly, it is the crane’s temperature that should be observed for its motor part and hydraulic part if available. The motor of the crane is especially important as the temperature of the motor keep increasing it may cause the system of the crane to fail and stop from working normally.

## 1.4 Objective

i. To design and develop a crane monitoring system using an android application.

ii. To analyze the system design in terms of its functionality.

## 1.5 Scope of Project

To achieve the objective of the project, some of the scopes must be identified. The project should be able to develop a monitoring system for the crane. The project scope is listed below:

i. The monitoring system is based on the android application so that the result can be displayed on the mobile platform.

ii. The wireless module for this project is LoRa (Long Range) RF (RADIO FREQUENCY) wireless transceiver module. To ensure the function range is better than a normal wireless module like the ESP8266 module.

iii. The microcontroller for this project is Arduino Uno.

The crane's data will be sent through the Arduino Uno by LoRa wireless module to the user's mobile. Then the data will be displayed on the application, the user will be able to monitor the crane's data immediately. The crane data mentioned in this project are only simulation data, these data are not the real crane's data but the data type is the same as the real crane. For example, the temperature data in this project is refer to the temperature of the crane, the current and power data, the weight data refer to the load of the crane, and data from the ultrasonic sensor refer to the fuel level of the crane.

## 1.6 Thesis Outline

The first chapter in this thesis is chapter 1, the introduction for this project. The project background, problem statement, objectives, and project scope are all written in this chapter. This chapter helps the reader to understand the reason for the making of this project.

The next chapter will be chapter 2, literature review. In this chapter, the related research done by the other researcher will be studied and analyzed one by one. Comparison and review of the research will be done to understand each project's advantages and disadvantages.

After that, chapter 3 is talking about the methodology which is the method and technique applied in this project. Besides that, the design of the procedure used on this project that represents the process of the hardware and software will also be included.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

For this chapter, some research had been reviewed to apply the knowledge, technique, method from previous research into this project. By evaluating other's people project, problems or issues that occur on their projects can be seen. So that the same issues will not be repeated in own project. Besides that, reviewing others people's literature will be able to give the chance to compare, contrast, synthesize and analyze different people's works. In this chapter, 10 different pieces of literature will be reviewed. The collected literature was a search based on related IoT projects.

## 2.2 Smart Bus Tracking and Management System Using IoT

This project is done by K. Sridevi, A. Jeevitha, K.Kavitha, K.Sathya, and K. Narmadha in 2017. They implemented this project based on IoT and using android as the application. Aside from that, it has two applications which one for client and one for server. First, they put the GPS (Global Positioning System) devices on the bus, the bus will GPS devices will be able to track their positions. Then the positions of the buses will be updated periodically to the server, the client application will display a map that shows the current location of the bus. This project allows users to track a bus and the distance between each station along its route by using Android App on the smartphone. They use the Arduino UNO microcontroller as their central unit and added a GPS module, RTC (Real Time Clock) module, and Wi-Fi module for the hardware section. This project uses the DS3231 RTC module to keep track of the current time so that the GPS location of the bus and the time of the bus at that moment are accurate. To upload the GPS location, it needs the help of a Wi-Fi or internet connection. So, the problem occurs when the bus is on a location that a weak internet connection will affect the GPS upload time and accuracy. GPS can work without an internet connection but still better with a stable internet connection to function faster(K, 2017).

## 2.3 IoT Based Monitoring System in Smart Agriculture

This project was done by Prathibha S.R., Anupama Hongal, Jyothi M.P. in 2017. For this project, they aim to make use of IoT technology and smart agriculture using automation. The purpose of this project can monitor the temperature and humidity in the agricultural field with sensors using CC3200 and a camera. Then, the images captured by the camera will be sent to farmers by MMS (Maintenance Management System) using Wi-Fi. The authors choose to use CC3200 Launchpad because they said that CC3200 is a low-cost and faster development evaluation platform for a microcontroller and its focus on low power. Besides that, CC3200 has features such as programmable user buttons, RGB LED, and onboard emulation for debugging. For the disadvantages of this project, the method to send the picture with MMS is outdated and this project already installed with a camera so it can be evolved to a real-time monitoring system through camera to allow the user to monitor the farm on the go with the camera on their mobile application(Prathibha, Hongal and Jyothi, 2017).

## 2.4 Garbage Bin Monitoring for Smart Residence

As an introduction, this project aims to help garbage collecting companies to enhance their garbage collection more efficiently by using IoT technology. The author Ng Tian Xun design the system with the concept of receiving, process, and send which mean the processing station (Arduino Uno) receive the data from all the bins through 433MHz Radio Frequency protocol followed by process and data filtering, then send the data to Raspberry Pi via Ethernet Shield. The project is divided into 3 blocks, it has a remote site, a base station, and a server site. The remote site is built with ultrasonic sensors, tilt switches, power supplies, Arduino UNO, and a 433 MHz RF transmitter. Then, the base station block consists of a 433 MHz RF Receiver, Arduino UNO, Ethernet Shield, and a router. Finally, the server site is simply just a Raspberry Pi 3 Model B. In the end, this project can display the collected data on a PC or mobile platform after gathering the data from the remote microcontroller and send to the central server. Even though the project is good, but the fact is the RF modules in this project are only able to work below 10 meters before equipping antenna or 25 meters after equipping antenna. If the garbage bin is accidentally moved by other people, the monitoring system will be failed(NG TIAN XUN, 2018).

## 2.5 Weight Monitoring Waste Bin Based on Internet of Things (IoT)

The authors of this project Nor Akmal Ahmad and Mohd Hakimi Zohari were done this project in 2020. The authors said that the main objective of this project is to monitor the dustbin by using their developed weight monitoring waste bin system. The second objective can create a dustbin that has a function to open and close automatically with the help of an ultrasonic sensor. Then, the third one can tell the user or waste administrator about the condition of the dustbin when it reaches the max level of capacity. Finally, is the function to observe the temperature and humidity condition for the bin on the Blynk Application. To build this project, they choose to use Arduino Uno as their main controller to control this project, then follow by some sensors include weight sensor, temperature, humidity sensor, and ultrasonic sensor. To implement the function to open and close the lid automatically, they installed servo moto for the bins. Then, as an IoT project, the help of a Wi-Fi module is a must for providing the communication function between the bins and user through server and client. The client for this project is the Blynk Application, this is a platform the allows users to create amazing interfaces for projects with various widgets to control Arduino, Raspberry Pi, and other stuff. For the server, they use the ESP8266 to function as a weight monitoring system. Besides all of this, the disadvantage for this project is very obvious which is the Wi-Fi module ESP8266 range is only able to reach around 100 meters on space. This means that every 100 meters will need an extra ESP 8266 module, this will become a problem to the cost if the quantity of the bins needs to monitor is many for a place like a residential area will be hard to implement(Akmal, 2020).

## 2.6 Vehicle Tracking and Monitoring System to Enhance the Safety and Security Driving Using IoT

In 2017, the authors A. Anusha and Syed Musthak Ahmed designed this vehicle monitoring and tracking system to monitor the vehicles that move from place to place to secure the safety of the road. With the help of the GPS and GSM technology, the vehicle's present location will be provided by GPS and send the tracking data to the server by GPRS. At the same time, and message will be generated and sent to the vehicle owner as an alert message. The alert message has 2 main purposes which can alert the driver if the driver is driving in the wrong direction or when the driver feels drowsy or drunk, also the buzzer will make a warning sound if the alcohol sensor detects the alcohol level reach the allowed level. Furthermore, a temperature sensor is also installed and used to detect the status of the vehicle engine’s temperature for further safety enhancement. The microcontroller in this project is LPC2148 while the GPS and GSM module is used to send the location. Any status will be displayed on the LCD screen of the project hardware module. Now that the disadvantage of this project is the lack of the function for the user to monitor the status on computer or mobile. This project does not have a server to process and send the data to make the tracking system less effective, if a server is created for this project, then monitoring huge numbers or car and driver's data will be more obvious and easier to be managed monitoring and tracking. In addition, only using GPS without an internet connection is less accurate and slow than with the help of the internet(Anusha and Ahmed, 2017).

## 2.7 IoT Enabled Smart and Secure Power Monitor

This project proposed an idea to monitor the power for an individual appliance by the authors Akshay Ramesh Jadhav and P. Rajalakshmi in 2017. The authors aim to build a project with lower cost, smaller, and easier to install. To build this project, they use only 10 USD dollars to make. The material includes ESP8266, ADE7757 IC, power supply, current sensor, and 3D printed casing. The authors said that they choose WIFI instead of ZigBee and Bluetooth because of the widespread availability of WIFI and smartphones also laptops are more preferred. In addition, they say Wi-Fi provides a longer range than others in the cost of consuming more power. The concept of the system is divided into three subsystems which are power acquisition module, processing, and communication module, and remote data logging unit. First, the power acquisition module builds with the current sensor and energy metering module to measure and calculate the values of main voltage and current flow through the wire connected to the appliance. Then second, the processing and communication module is simple a WIFI module ESP8266 to send the data collected to the server or remote data logging unit. Then the third one, the remote data logging unit is a computer with a server application running and connected on the same network with the power monitor sensor. To provide a more user-friendly environment for monitoring, viewing, and improving user experience, the authors developed an Android application. On the application, the user can view the collected information in data-view or graph-view if the user is connected to the internet connection. However, the drawback of this project is the ESP8266 module's range is limited and needs to be very close to the router to have a stable connection. There is no microcontroller in this project to create a low-cost project, but no microcontroller means the function of this project is also limited where this project lacks the function example like to turn on or turn off for the appliance(Jadhav and Rajalakshmi, 2017).

## 2.8 A smart IoT-based system for monitoring and controlling the sub-station equipment.

In the year 2019, Md. Sanwar Hossaina, Mostafizur Rahmana, Md. Tuhin Sarker b, Md. Ershadul Haquec and Abu Jahida implement a system for remote monitoring and controlling the sub-station equipment through IoT. The system allows objects to be sensed or controlled remotely across the network infrastructure. The system is set up with a microcontroller Atmel Pico power 8bit microcontroller Atmega328-p then is communicate with the ESP8266 Wi-Fi module by serial communication technique. But before that, some sensors and actuators will collect information from the transformer and oil circuit breaker (OCB). The sensors include an infrared sensor, ultrasonic sensor, and also actuators. If the quality and present amount of oil are in alarming condition, an alarm message will be sent to the operator with what corrective action needed to be taken immediately. Besides that, the operator will also be able to do the circuit breaker operation and relay plug/transformer tap changing operation remotely through this system. To achieve these functions, the microcontroller must be connected to the web server by internet connections through a WIFI module. After the WIFI module uploads all the data to the webserver, the operator will be able to give commands and instructions. At the same time, the monitoring and control for the sub-station will be able to display at a webpage from the stored web server on any authorized internet-enabled device. However, the problems are the connectivity issue for ESP8266 and lacking interface on a mobile platform. ESP8266 can only establish a stable connection below 100 meters without an antenna, as the sub-station is usually a very large area hence it is very less efficient using this WIFI module. Then lacking a mobile interface is optional but adding a mobile interface makes the works smoother because the operator might not need to bring a laptop which makes them easier to work(Hossain *et al.*, 2019).

## 2.9 Internet of Things (IoT) using LoRa technology.

The authors Alireza Zourmand, Andrew Lai Kun Hing, Chan Wai Hung, and Mohammad AbdulRehman presents the performance and the ability to cover a big area of the LoRa network in both indoor and outdoor condition in 2019. The authors also said that the quality of the LoRa network depends on both the distance between the gateway and also the structural element. The connection of the LoRa network is divided into three sections, LoRa end-devices, Integrated with an internet connection, and Cloud server and web interface. Besides that, the authors state that adding a gateway will improve the indoor coverage performance, then the outdoor maximum coverage area is 330 meters. In this project, they use sensors, sx1278(433MHz) and Arduino UNO as the LoRa end-devices while they use sx1278(433MHz) and ESP8266 for the LoRa gateway. The result shows that 330 meters are the maximum communication coverage if the LoRa gateway setup is put indoor which does not reach the minimum distance of 1 km stated in the LoRa datasheet. The authors state the reason is because of the higher noise interference and path loss that occur inside and outside of the site. Then, the authors added the reason for this is because the nature of the antenna used in the testing is omnidirectional which send the signal in all direction through the air and most of the power is lost due to the receiver antenna does not receive it when the penetration of signal power go through obstacle reduce the power of the sending signal. To solve this problem, the authors suggest that placing the LoRa gateway near the window is better than putting it in a room. The disadvantage of this project will be the interface of the LoRa system that the authors were using. The interface can be designed to become more user-friendly and better if there is an option to display all the information on the mobile platform(Zourmand *et al.*, 2019).

## 2.10 Tower Crane remote wireless monitoring system based on Modbus/TCP protocol

This project is developed by Bo Li, Geng Chen, Le Wang, Zhe Hao in 2017 that a wireless system to monitor and manage the tower crane for the aims to control tower cranes more efficiently and safely. The data information of the crane will be detected by the sensors and process by the ARM/STM32 smart control unit which is also the core Central Processing Unit (CPU). Then GPRS module is connected to the internet by UART RS485 to allow the data send to the remote monitoring terminal. In the end, the user will be able to monitor and supervise tower cranes operation by Internet on the web platform. The drawbacks are only the web interface is available and at the same time, the web interface is not user-friendly. The web interface is very complex and not direct to the point to show the crane. Other than that, the GRPS module used in this project is a very old model which will cause the connectivity to remote control the crane not smooth when the network ping high(Li *et al.*, 2017).

## 2.11 Warehouse inventory management system using IoT and open-source framework.

The authors B. Sai Subrahmanya Tejesh and S. Neeraja developed this system in 2017. This system helps the user to maintain the product detail and information and can tell the user where the product is present. The authors choose RFID among all other wireless communication technologies because they think RFID is the most suitable to be used on their warehouse management system. To explain this reason, the RFID's ability to identify the products or goods with attached tags is very unique and preferred. To set up this project, the authors divide the system into blocks diagram of RFID tags, RFID reader, ESP8266, Raspberry Pi 3, and a web server. In the beginning, the RFID tags will be placed in each different warehouse room. Then, by using an RFID reader to scan the tags, the information will be transmitted as a short string of data by the ESP8266 and sent to the Raspberry Pi 3 which functions as a central server. If a user is connected to the internet, the user will be able to get all the information by searching a specific tag number on the web page. But before the user does that, the user is also required to get authorization for entering the web page. All information includes tag number, product description, location, stockroom number, and time detail will be displayed on the web page in a table format. The weak point for this project will be the WIFI module and the interface of the web page. The WIFI module used in this project is ESP8266 which the range only able to reach a 100meter range. If the warehouse is far larger than this 100meter long, then the system will not be able to function well. Besides that, this project is only having a web page interface is not convenient for people to look for many items. Let say when someone wants to find many products in a different warehouse. Those people will need to use the computer in the office first to search for the products. This can be improved by implementing the interface on the mobile platform so that any authorized user will be able to search the products anytime on a small and always carried mobile(Tejesh and Neeraja, 2018).

## 2.12 Journal Comparison for Relevant Previous Research

Table 2.1 shows the comparison table for 10 different research that related to this project so that we can improve or avoid the drawback for this project after analyzing it.

Between ten of this research, most of the projects use ESP8266 WIFI module because it is cheap and WIFI is a very common wireless method, but the drawback is that WIFI spread and detection range is still limited to only around 100 meters. Although compared to Bluetooth or infrared, WIFI is better but for this project, a better wireless module is needed. Then, the research of the Internet of Things (IoT) using LoRa technology shows that LoRa's communication can reach 1 km in the paper although LoRa performance is only able to reach 330 meters on their project. To improve the detectable range for this project, LoRa has the best range among all the other wireless communication methods in this 10 related research.

Then, another thing most of this research does not put into consideration is that their projects do not have a mobile platform/GUI to monitor the data. When take about monitoring, monitoring the data should be more direct, simple, convenient, and fast. Even though monitoring data on the computer is good but in this era that almost all people have a mobile phone, a mobile platform to allow the user to monitor their data in a mobile application will be better.

Table 2.1 Comparison Table For 10 Different Related Research

| No. | Author | Title | Technique/Component Used | Advantages | Disadvantages |
|---|---|---|---|---|---|
| 1 | K. Sridevi , A. Jeevitha , K. Kavitha , K. Sathya and K. Narmadha 2017 | Smart Bus Tracking and Management System Using IoT | RTC (real-time clock), GPS (Global Positioning System), Arduino Uno | Track buses through Android App which buses carry GPS devices to allow tracking. The server monitors the location and stores its data in the database. | If a bus is on a weak internet connection area, the GPS will be slower to track the bus because GPS without an internet connection is less accurate. |
| 2 | Prathibha S.R., Anupama Hongal, Jyothi M.P. 2017 | IoT Based Monitoring System In Smart Agriculture | CC3200, GPRS, DHT11, LDR, web server NRF24L01, camera | Monitor temperature and humidity through sensors, use a camera to capture images and send the pictures through MMS to farmer's mobile using Wi-Fi. | MMS is an outdated technique to send pictures. |
| 3 | Ng Tian Xun 2018 | Garbage Bin Monitoring for Smart Residence | Raspberry Pi 3 Model B, RF module, Arduino Uno, Arduino Micro/Nano, UDP (User Datagram Protocol), MQTT (Message Queuing Telemetry Transport), Tilt Sensor, Ultrasonic Sensor | Display data on PC/mobile after data gather from the remote microcontrollers and send to the central server. | RF modules without antenna are only able to work below 10meter while after equipping antenna also around 25 meters. |

| | | | | | |
|---|---|---|---|---|---|
| 4 | Nor Akmal Ahmad and Mohd Hakimi Zohari 2020 | Weight Monitoring Waste Bin Based on Internet of Things (IoT) | Arduino Uno, Weight Sensor, Temperature and Humidity Sensor, Ultrasonic Sensor, WIFI-module | Since temperature and humidity change of the waste, also the weight of the rubbish. Then, send and display on the application. | The dustbin needs to be near the WIFI signal as ESP8266 does not have a good WIFI detection range. |
| 5 | A. Anusha, Syed Musthak Ahmed 2017 | Vehicle Tracking and Monitoring System to Enhance the Safety And Security Driving Using IoT | GPS, GSM, SIM800 module, GPRS, Microcontroller LPC2148 | Monitoring moving vehicle parameters and display on display board then send message to the mobile. | Only able to monitor through the LCD screen. |
| 6 | Akshay Ramesh Jadhav, P. Rajalakshmi 2017 | IoT Enabled Smart and Secure Power Monitor | ESP8266, current sensor | Monitor the power consumption and display value on the Android application which the power data stored on the server. | ESP8266 detection range is weak so that the module needs to close to the WIFI signal. No microcontroller means limited function. |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 7 | Md. Sanwar Hossaina , Mostafizur Rahmana , Md. Tuhin Sarker b , Md. Ershadul Haquec , Abu Jahida 2019 | A smart IoT based system for monitoring and controlling the sub-station equipment | Ultrasonic sensor, IR sensor, actuator, ESP8266, MQTT | Remote monitoring and controlling the sub-station equipment through IoT. The system allows objects to be sensed or controlled remotely across the network infrastructure. | Only display on a computer, no mobile interface. |
| 8 | Alireza Zourmand, Andrew Lai Kun Hing, Chan Wai Hung, Mohammad AbdulRehman 2019 | Internet of Things (IoT) using LoRa technology. | LoRa, ESP8266, sensors, sx1278, Arduino Nano | Able to reach the detectable distance of the LoRa communication network for 330meters. | GUI interface to monitor the data is web interface only. |
| 9 | Bo Li, Geng Chen, Le Wang, Zhe Hao 2017 | Tower Crane remote wireless monitoring system based on Modbus/TCP protocol | Modbus/TCP protocol, ARM9 Cortex A8, GPRS | Monitor and manage system of tower crane based on Modbus/TCP protocol wirelessly through GPRS. | Only has a web interface to monitor the crane. The interface is not user-friendly. |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 10 | B. Sai Subrahmanya Tejesh, S. Neeraja 2017 | Warehouse inventory management system using IoT and open-source framework | RFID, Raspberry Pi, AIDC (Automatic identification and data capture), ESP 8266 | Use Raspberry Pi as a central server, monitoring all the information. Then, created a webpage to provide convenience and an interface to the user to track the inventory data. | ESP8266 short WIFI range and only has a web interface. |

## 2.13 Summary

At the end of this chapter, some advantages, and disadvantages of projects are analyzed. It is easier to implement the techniques that previous researchers had been used in this project. Also, the disadvantages can be avoided as much as possible. The result for reviewing previous researchers' research affects this project in a good way because the proper and suitable method will be applied while keeping the weak point aside. After going through these journals, I want to use LoRa as the wireless communication method to send and receive data. It is because the usual and common WIFI, BLUETOOTH technique is not able to provide long-range communication.

# CHAPTER 3

# METHODOLOGY

## 3.1 Introduction

After doing the review and comparison of 10 different literature reviews in the previous chapter. The idea to make this project is clarified. To implement this idea, software and hardware components are involved, so the detail of both the type of the component must be determined. After the hardware is decided, the block diagram to represent the relationship for each hardware component will be created. At the same time, the flowchart will also be created to show the workflow of this project. Besides that, an algorithm flowchart will also be created to show the workflow for the programming coding. To finish this project on time, a Gantt chart will be needed for planning and finishing each of the tasks without confusion.

The previous research in chapter 2 shows the data of the wireless connection method to transfer the sensor's value. So, in this chapter, I will show the advantage of LoRa for its ability to reach how long the range will be. Besides that, the result of the initial result for the sensors will also include in this chapter.

## 3.2 Software

### 3.2.1 Arduino IDE

Arduino IDE is a software that allows users to communicate with Arduino hardware by uploading the coding to the Arduino hardware board. The hardware is including the Arduino board itself, Arduino shields, modules, and sensors. For this chapter, the value for the sensors will be collected. To collect the data from the sensors, Arduino IDE is needed to upload the programming code to the Arduino board. Arduino IDE's environments use C and C++ languages to write and compile the code.

### 3.2.2 Android Studio

Android Studio is used to develop an app to monitor the data easier. The health of the crane will be shown in this application by showing the value get from the sensors. To show the data in a graphical view, a library called MPAndroidChart had been used. The function of this library is mainly to display the graph in Android. Java is the official language for Android Studio development but there are much more support languages like Kotlin, C++ and many more.

## 3.3 Hardware

### 3.3.1 BOM (Bill of Material)

Table 3.1 BOM table

| No. | Components Name | Description | Quantity |
|---|---|---|---|
| 1 | Arduino UNO | Arduino UNO Board | 1 |
| 2 | LoRa Module | LoRa wireless module | 2 |
| 3 | NodeMcu ESP8266 | WIFI module | 1 |
| 4 | Load Cell(5Kg) | Weight sensor (with HX711) | 1 |
| 5 | DHT22 | Temperature sensor | 1 |
| 6 | ACS712 | Current sensor | 1 |

### 3.3.2 Arduino UNO

The reason for using Arduino UNO is due to its programming ease. Besides that, the low power consumption and low cost of Arduino UNO are also the reason for choosing it. In this project, Arduino UNO will be used to communicate with the sensors and then a LoRa module will be connected as a transmitter. So, the sensor data will be processed and sent to Arduino UNO, then the data will be sent using LoRa modules to the receiver side which in this project is the NodeMCU.

### 3.3.3 LoRa Module

LoRa module is a new technique that allows data transmission at low data rates to very long ranges. LoRa has a different type of operation frequency, it can operate in 433MHz, 868MHz, and 915 MHz based on the models. The advantage of LoRa is the long-range transmission that can at least reach more than 1km which is better than the common wireless communication method like infrared, Bluetooth, and WIFI. The disadvantage is also obvious, which is the low data rate transmission. Due to that, LoRa will only be able to send simple data, but it is still enough and suitable for this project because only the sensor's value needs to be sent.

### 3.3.4 NodeMcu ESP8266

ESP8266 modules allow the microcontroller to connect to WIFI. It is used in this project to make the Arduino able to send the collected sensor's value to the database through a WIFI connection. The sensor's value will be received through the connected LoRa module with this board, then it will be sent to the database.

### 3.3.5 Sensors

To monitor the health and utilization rate of the crane, some sensors will be needed. Some elements that are needed are the power consumption, working hours, temperature, fuel status and tonnage of the crane. But in this project, the prototype only will be created, so the related sensor will be used.

## 3.4 Block Diagram

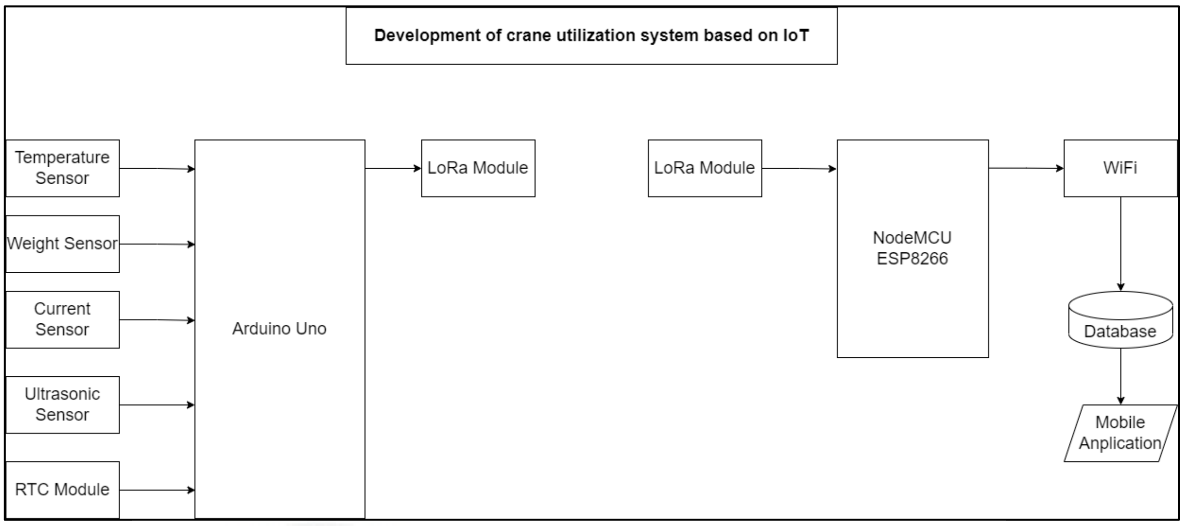


Figure 3.1 Block Diagram of Development of Crane Utilization System Based on IoT

Above Figure 3.5 is the block diagram of this development of crane utilization system based on IoT project. This project has two parts, the left side is the transmitter part, and the right side is the receiver part. This project start function from the transmitter part, the sensors will start working when the crane starts to work. Then, the value of the sensor will be sent to Arduino UNO to process. After that, the processed data will be sent to the receiver side through the LoRa module. Figure 3.6 shows the transmitter side block diagram of this project.

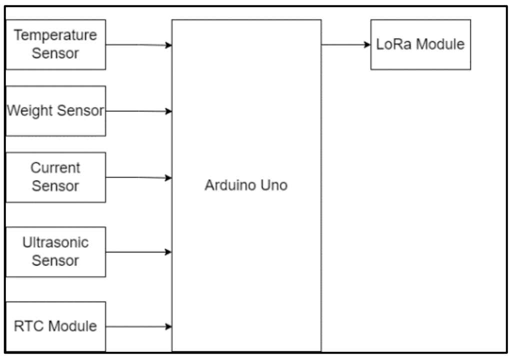


Figure 3.2 Block Diagram for Transmitter Side

Then, after the LoRa module on the receiver side receives the data, it will be sent to the database by using the WIFI connection from the NodeMCU ESP8266 internet module. Finally, the user will be able to start the crane monitoring by running the application on their mobile phone. Figure 3.7 shows the block diagram for the receiver side of this project.

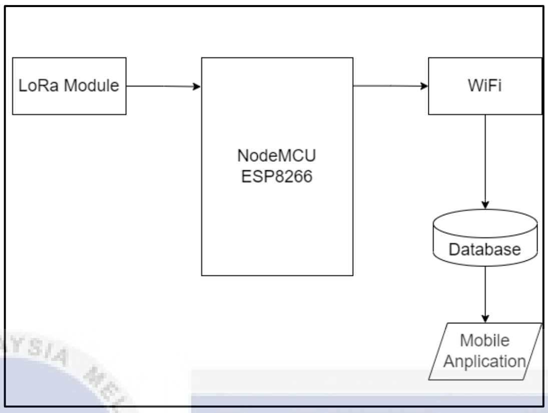


Figure 3.3  Block Diagram for Receiver Side

## 3.5 Project Overview

Table 3.2 shows a Gantt chart that represents the project activity execution schedule time every week until the PSM 1 presentation while table 3.3 shows  a Gantt chart  that represents the project activity for PSM2. In PSM 1, the Gantt chart is focusing on the research and methodology activity to determine the possible and suitable method could be use in this project. After PSM 1 completed, the method that obtain in PSM 1 will be execute so a project prototype can be create according to the plan from PSM 1. At the end, what PSM 1 do is planning while PSM 2 is to implement the plan.

### 3.5.1 Gantt Chart

Table 3.2 Gantt chart Bachelor Degree Project 1

| No. | Project Activity | Expected /Actual | Week | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | Registration for final year project | Expected | 🟥 | 🟥 | | | | | | | | | | | | | |
| | | Actual | 🟥 | 🟥 | | | | | | | | | | | | | |
| 2 | Final year project briefing | Expected | 🟥 | 🟥 | | | | | | | | | | | | | |
| | | Actual | 🟥 | 🟥 | | | | | | | | | | | | | |
| 3 | Title discussion and decision with supervisor | Expected | 🟥 | 🟥 | | | | | | | | | | | | | |
| | | Actual | 🟥 | 🟥 | | | | | | | | | | | | | |
| 4 | Study Related Research | Expected | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | |
| | | Actual | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | |
| 5 | Complete Chapter 2: Literature Review | Expected | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | |
| | | Actual | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | |
| 6 | Progress Update to supervisor | Expected | | | | | | 🟥 | | | | | | | | | |
| | | Actual | | | | | | 🟥 | | | | | | | | | |
| 7 | Complete Chapter 1:Introduction | Expected | | | | | | 🟥 | 🟥 | | | | | | | | |
| | | Actual | | | | | | 🟥 | 🟥 | 🟥 | | | | | | | |
| 8 | Complete Chapter 3: Methodology | Expected | | | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | | | |
| | | Actual | | | | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | |
| 9 | Submit report | Expected | | | | | | | | | | | | 🟥 | 🟥 | | |
| | | Actual | | | | | | | | | | | | 🟥 | 🟥 | | |
| 10 | Preparation for presentation | Expected | | | | | | | | | | | | | 🟥 | 🟥 | |
| | | Actual | | | | | | | | | | | | | 🟥 | 🟥 | |
| 11 | Submit presentation video | Expected | | | | | | | | | | | | | 🟥 | 🟥 | |
| | | Actual | | | | | | | | | | | | | 🟥 | 🟥 | |
| 12 | PSM 1 presentation | Expected | | | | | | | | | | | | | | | 🟥 |
| | | Actual | | | | | | | | | | | | | | | |

Table 3.3 Gantt chart Bachelor Degree Project 2

| No. | Project Activity | Expected /Actual | Week | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | Draft Material List | Expected | 🟥 | | | | | | | | | | | | | | |
| | | Actual | 🟥 | | | | | | | | | | | | | | |
| 2 | Test Hardware | Expected | | 🟥 | | | | | | | | | | | | | |
| | | Actual | | 🟥 | | | | | | | | | | | | | |
| 3 | Design application and hardware coding | Expected | | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | | | |
| | | Actual | | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | | | | |
| 4 | Set up Database | Expected | | | | 🟥 | 🟥 | | | | | | | | | | |
| | | Actual | | | | 🟥 | 🟥 | | | | | | | | | | |
| 5 | Analyse Result | Expected | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | |
| | | Actual | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | | | | |
| 6 | Complete Chapter 4:Result and Discussion | Expected | | | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | |
| | | Actual | | | | | | | | 🟥 | 🟥 | 🟥 | 🟥 | | | | |
| 7 | Complete Chapter 5:Conclusion | Expected | | | | | | | | | | | 🟥 | 🟥 | | | |
| | | Actual | | | | | | | | | | | 🟥 | 🟥 | | | |
| 8 | Submit draft report | Expected | | | | | | | | | | | | 🟥 | | | |
| | | Actual | | | | | | | | | | | | 🟥 | | | |
| 9 | Prepare Project Poster | Expected | | | | | | | | | | | | 🟥 | 🟥 | | |
| | | Actual | | | | | | | | | | | | 🟥 | 🟥 | | |
| 10 | Preparation for presentation | Expected | | | | | | | | | | | | | 🟥 | 🟥 | |
| | | Actual | | | | | | | | | | | | | 🟥 | 🟥 | |
| 11 | Presentation | Expected | | | | | | | | | | | | | | 🟥 | |
| | | Actual | | | | | | | | | | | | | | 🟥 | |
| 12 | Submit Final Report | Expected | | | | | | | | | | | | | | | 🟥 |
| | | Actual | | | | | | | | | | | | | | | 🟥 |

### 3.5.2 Project Flowchart

To start a project from scratch, a work flow is needed. Figure 3.8 show a project flow chart that had been used in this project. First, the title of the project need to be decided. Then, the project objective, problem statement and the project scope must be identified. By doing this, a possible and suitable method to build this project can be determine. After that, some research on previous research needs must be done to see how other researcher are working in their project that related to this project. Then, the methodology that suitable in this project will be define. Then, start the simulation and testing to find the problem, so troubleshoot can be done. At the end, if the project works without problems, an analysis of the result can be make.

An algorithm flow chart helps to build a new program because in a algorithm flow chart, it explain what is the process of a program step by step. Figure 3.9 shows the algorithm flow chart of this project. First, the Arduino Uno and the NodeMCU will connected with each other by LoRa transceiver module. Then, the sensors data of the transmitter side(Arduino UNO) will be collected and sent to the receiver side(NodeMCU). After that, the NodeMCU will upload the data to the database if it is connected to the internet. If no connect to internt, the NodeMCU will try to connect with internet again until it success. Next, after the data had been uploaded to the database, the user will able to view the data in the Android  mobile application.

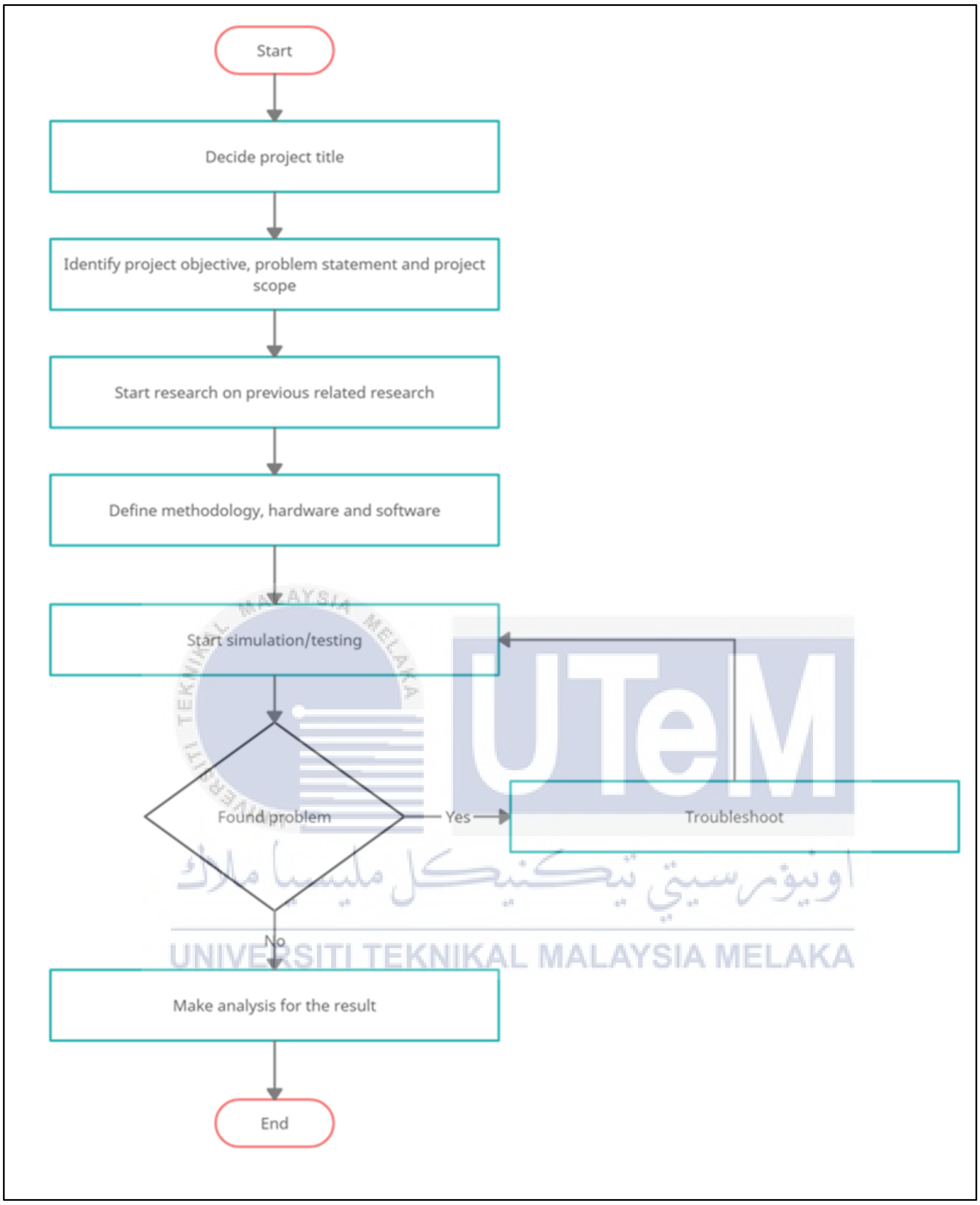

Start
Decide project title
Identify project objective, problem statement and project scope
Start research on previous related research
Define methodology, hardware and software
Start simulation/testing
Found problem
Yes
Troubleshoot
No
Make analysis for the result
End

Figure 3.4 Project flowchart

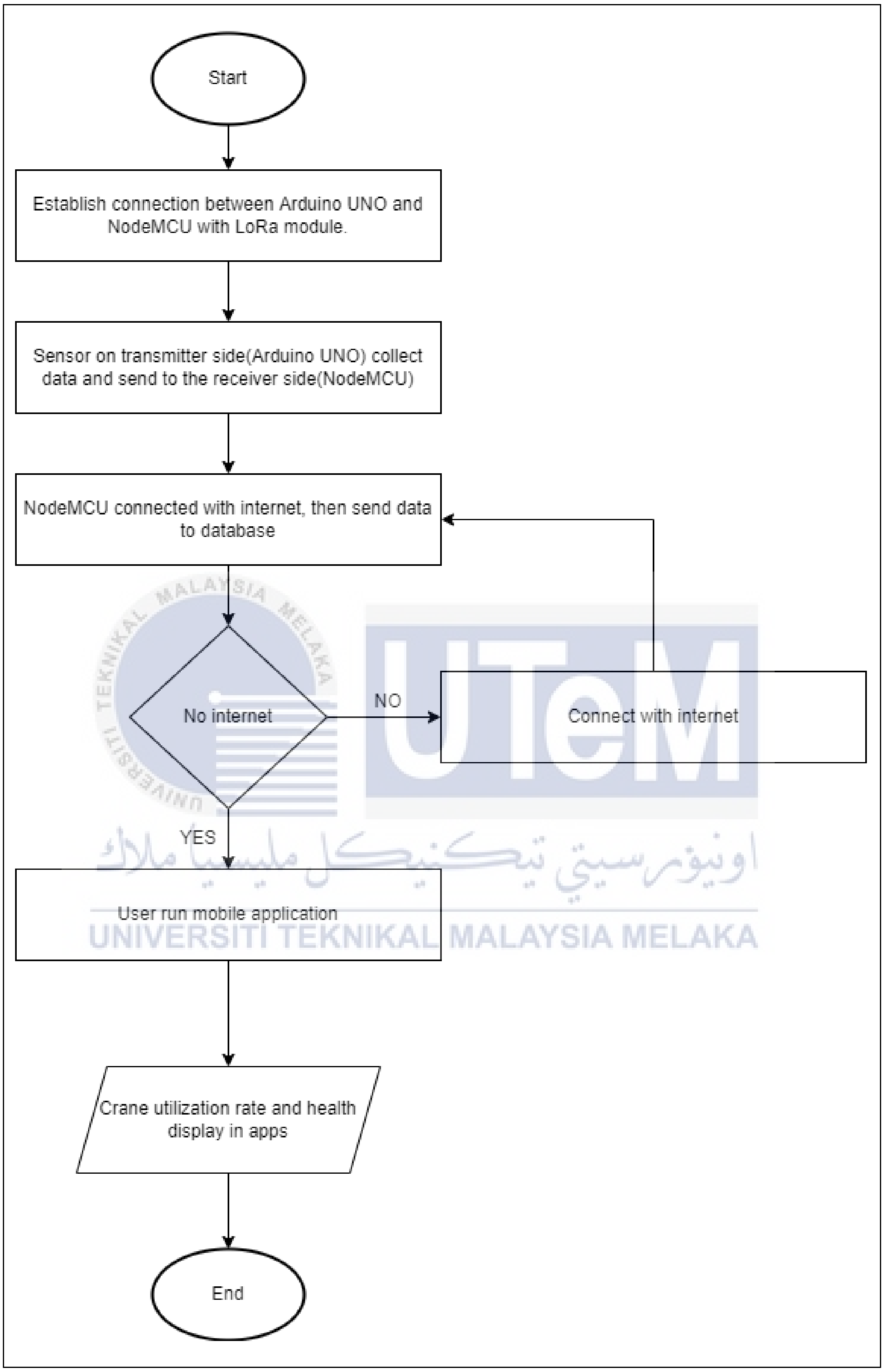

Start
Establish connection between Arduino UNO and NodeMCU with LoRa module.
Sensor on transmitter side(Arduino UNO) collect data and send to the receiver side(NodeMCU)
NodeMCU connected with internet, then send data to database
No internet
NO
Connect with internet
YES
User run mobile application
Crane utilization rate and health display in apps
End

Figure 3.5 Algorithm flowchart

## 3.6 Summary

In this chapter, the method that going to be used in this project and the development process had been explained. The reasons for the hardware and software used in this project are also explained one by one. This will help the reader to understand the development of this project more clearly.

To complete the project and achieve the project's objective, a plan will be needed. A plan will consist of planning the working flow and the project development schedule. This is to make sure each element in this project can be done on time and successfully. The plan can be carried out by creating a flow chart that shows all the processes and workflow for this project and makes a Gantt chart to schedule the needed time for completing each task every week. Last but not least, an algorithm flow chart will also be created.

# CHAPTER 4

# RESULT AND DISCUSSION

## 4.1 Introduction

When this project can work properly, the result and output will be created. In this chapter, those outcomes and results will be presented and discussed. In addition, the hardware implementation for this project Development of Crane Utilization System Based On IoT will be clarified. To get the data for crane utilization rate simulation, the hardware needs to be set up first. After the hardware had been tested and can run normally, coding that based on the programming flow chart needs to compile and install to the hardware. Then, the output from this project will be shown. At the same time, the obtained data can then be used for result analysis.

## 4.2 Hardware Setup

In this section, the implementation or design of the hardware will be discussed. For this project, two-part of hardware had been set up which are the transmitter side and receiver side. Figure 4.1 shows the transmitter side of the project while Figure 4.2 shows the receiver side of the project. At the transmitter side, the data will be collected using sensors then will be processed in the Arduino UNO and sent through the LoRa transceiver module. The sensors and modules are labelled in Figure 4.1. Then, the receiver side only has a NodeMCU ESP8266 connected with the LoRa module. The reason for this is because the receiver side is only used to process the collected data sent from the transmitter's LoRa transceiver, then upload to the database after NodeMCU is connected to the internet. The connection and hardware setup are labelled in Figure 4.2.

After both transmitter and receiver had been set up, some of the sensors need to add another object for measuring different data. Firstly, the ACS712 Current Sensor needs a load to allow the sensor for measuring the current data from the load. In this project, a TS90a Servo Motor has been used as the current sensor’s load. Then, an Arduino NANO is used to supply power and program the servo motor to rotate occasionally every 15 seconds. Second, the weight sensor also needs a load for measuring the weight data. So, an object which is a broken NodeMCU chip is added and placed on the 5KG load cell to become the weight sensor load. Figure 4.3 shows an ACS 712 Current Sensor that is connected to the load(TS90a Servo Motor). At the same time, Figure 4.4 shows a 5KG load cell that has a load(broken NodeMCU chip) placed on it.

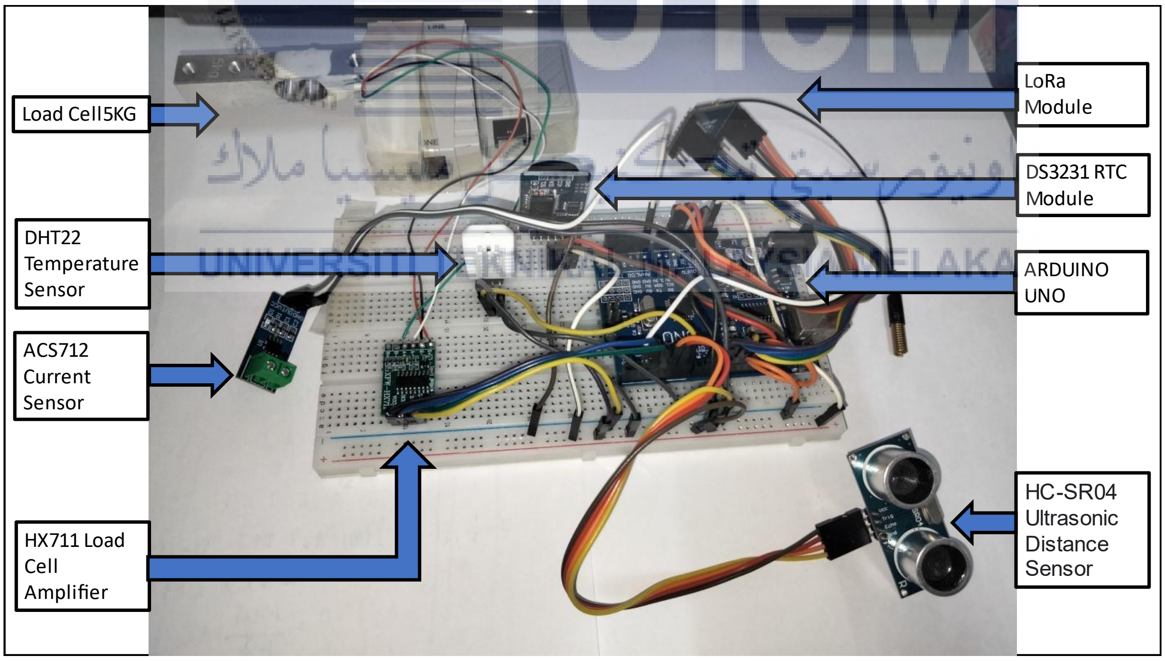


Figure 4.1 Transmitter Side Hardware Set-Up

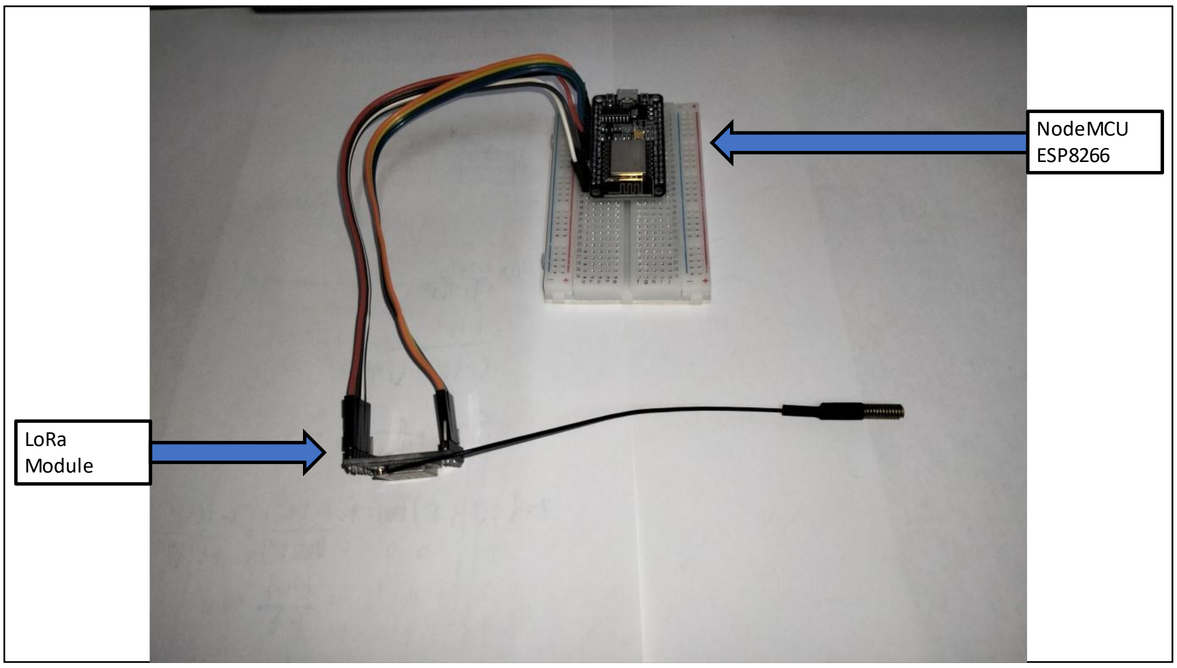


Figure 4.2 Receiver Side Hardware Set-Up

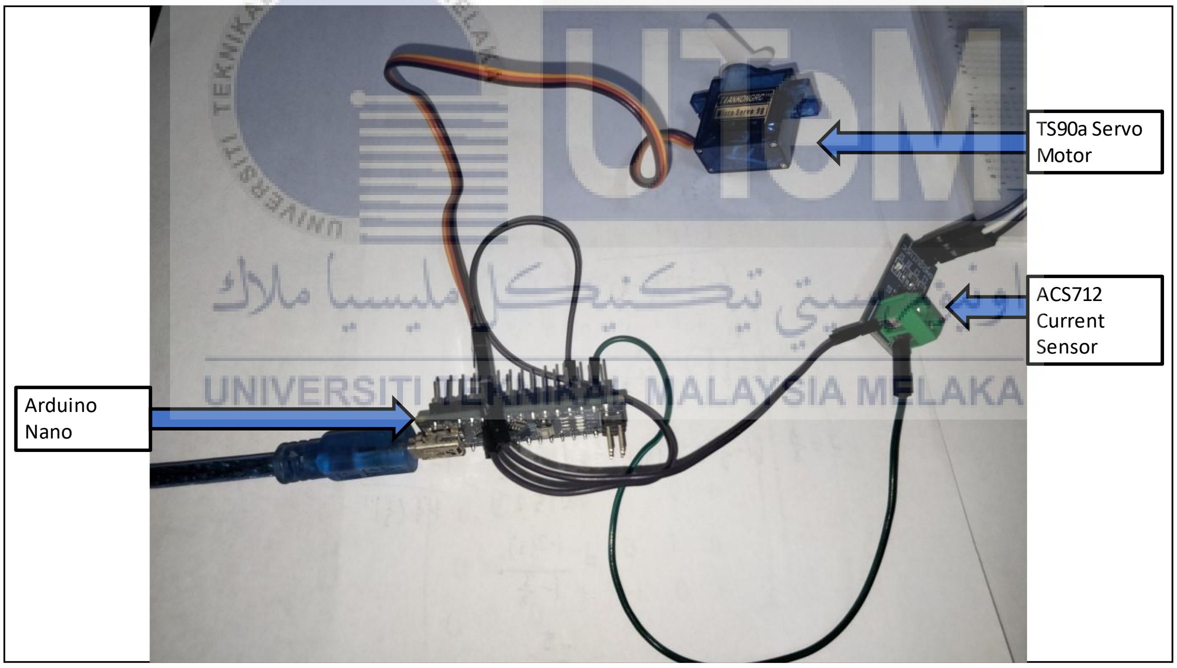


Figure 4.3 ACS712 Current Sensor Load(Servo motor)

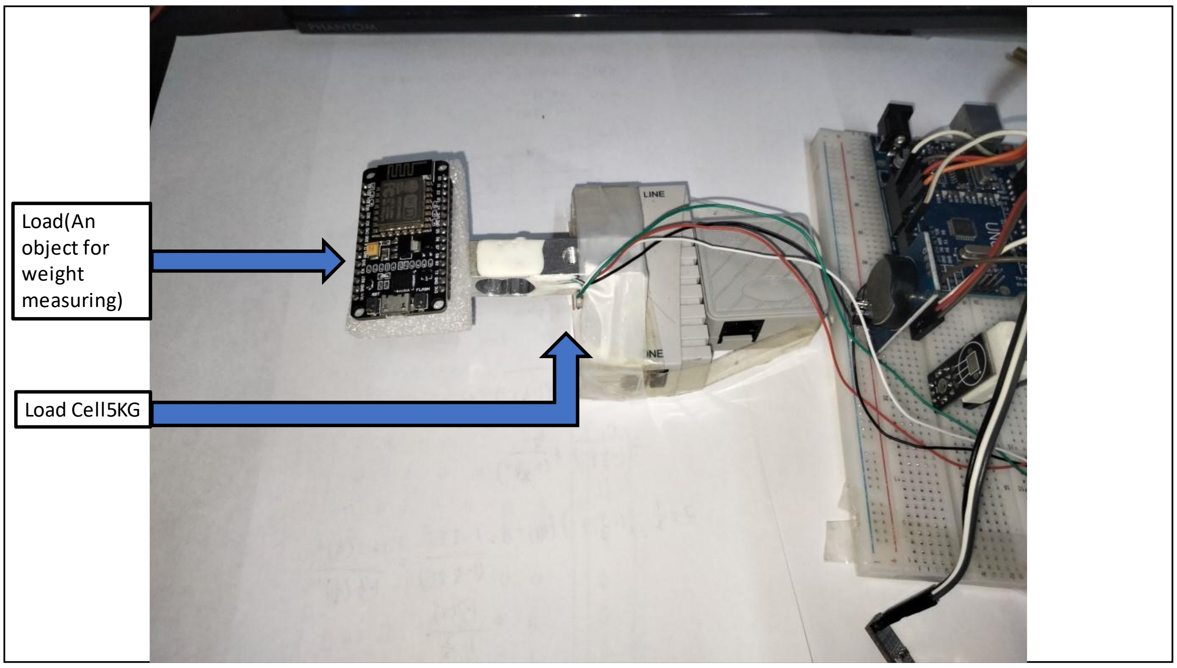
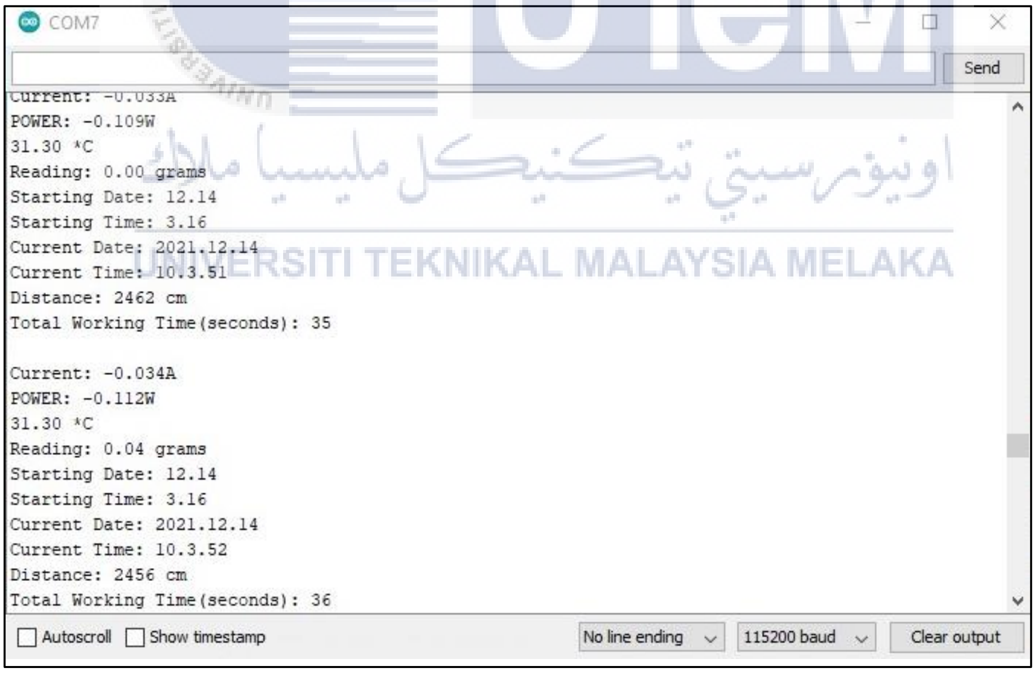


Figure 4.4  Load Cell 5KG and Load(Broken NodeMCU Chip)

## 4.3 Result Analysis

In this section, the output obtained from the project will be shown. This includes the data sent and received through the LoRa transceiver. First, the data will be collected by the sensor on the transmitter side. Then it will be transmitted and received by the LoRa module. Next, the data will be collected and processed on the receiver side. Figure 4.5 shows the serial monitor from the Arduino software when a transmitter sends data dan Figure 4.6 shows the serial monitor from the Arduino software when a receiver receives data. There are some symbols in Figure 4.6, the purpose for this is to capture and fetch data from the serial monitor which allows different data to store in a different variable. In addition, the SNR and RSSI values of the LoRa module are also displayed in the serial monitor of the receiver side.

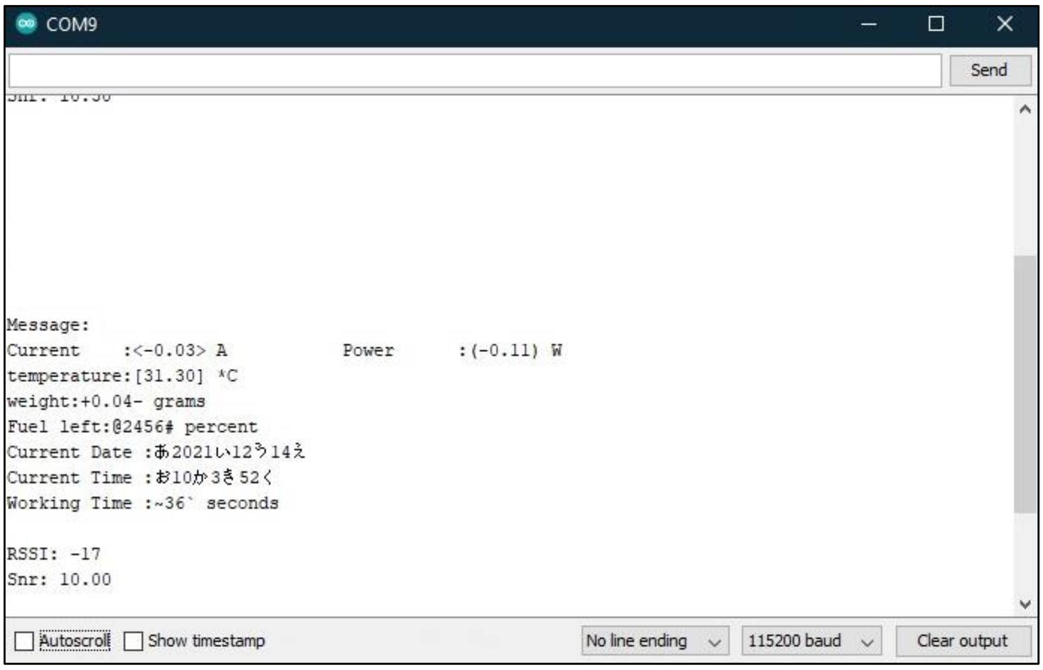
The database has two-part, one is real-time data and another one is recorded data. The reason to separate and store the data like this is that Google Firebase's real-time database is a NoSQL database. This means that the data uploaded to the database is unstructured data and hard to manage compared to SQL data. Due to that, this database has three parts. First part, the real-time data will be shown and displayed in real-time but never recorded, while the recorded part record and save the date and time as database key with the sensors data as database value. The reason for the real-time data not being recorded is because there is already a location to record the data for each second the prototype works. The second part is the database recording the data for each second the prototype works while the third part is the database recording the data for each hour the prototype works. By doing this, the data will be able to be analysed by applying different methods and depending on what the user wants to do. In this project, the real-time data will be shown directly with the value from the database. On the other hand, a graphical view will be used to display both recorded seconds and hours data getting from the database. Speaking of this, the graph view will be displayed in the android application.



```
Current: -0.033A
POWER: -0.109W
31.30 *C
Reading: 0.00 grams
Starting Date: 12.14
Starting Time: 3.16
Current Date: 2021.12.14
Current Time: 10.3.51
Distance: 2462 cm
Total Working Time(seconds): 35

Current: -0.034A
POWER: -0.112W
31.30 *C
Reading: 0.04 grams
Starting Date: 12.14
Starting Time: 3.16
Current Date: 2021.12.14
Current Time: 10.3.52
Distance: 2456 cm
Total Working Time(seconds): 36
```

Figure 4.5 Transmitter Side Serial Monitor



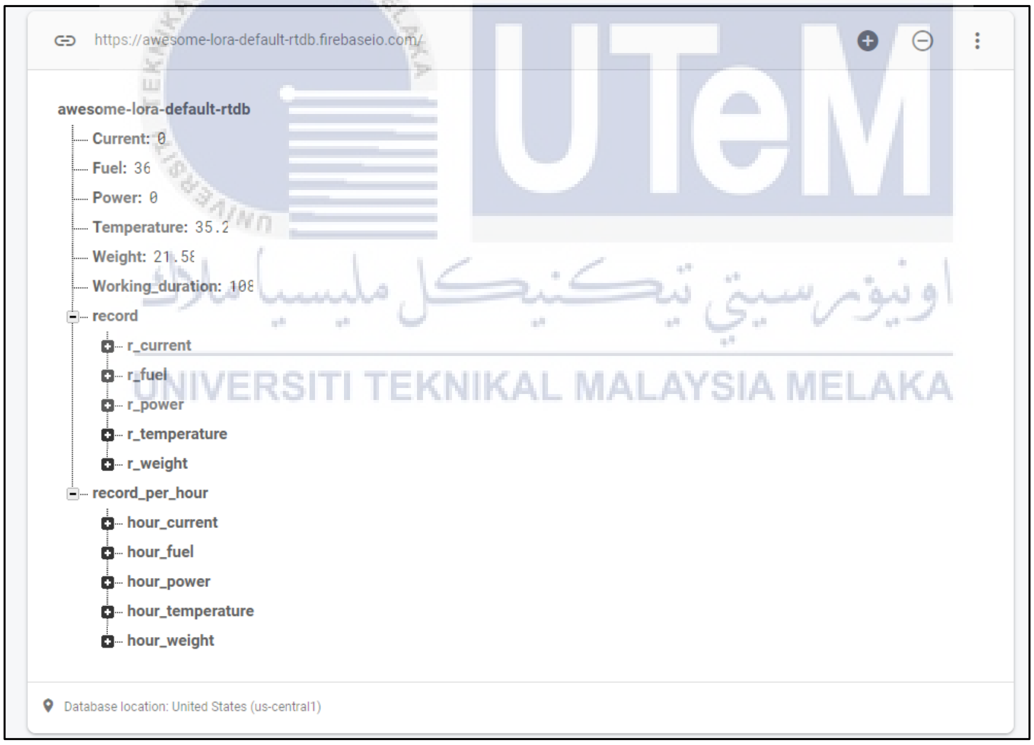
```
Snr: 10.00




Message:
Current    :<-0.03> A          Power      :(-0.11) W
temperature:[31.30] *C
weight:+0.04- grams
Fuel left:@2456# percent
Current Date :あ2021い12う14え
Current Time :お10か3き52く
Working Time :~36` seconds

RSSI: -17
Snr: 10.00
```

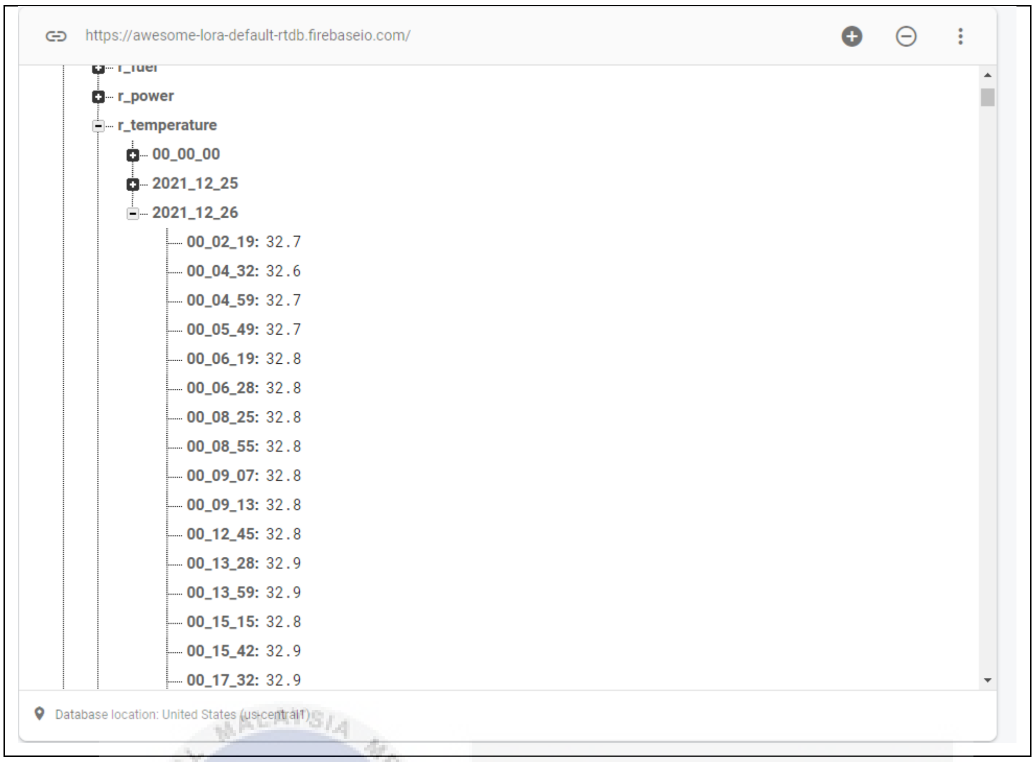
Figure 4.6 Receiver Side Serial Monitor


https://awesome-lora-default-rtdb.firebaseio.com/

```
awesome-lora-default-rtdb
  Current: 0
  Fuel: 36
  Power: 0
  Temperature: 35.2
  Weight: 21.58
  Working_duration: 108
  record
    r_current
    r_fuel
    r_power
    r_temperature
    r_weight
  record_per_hour
    hour_current
    hour_fuel
    hour_power
    hour_temperature
    hour_weight
```

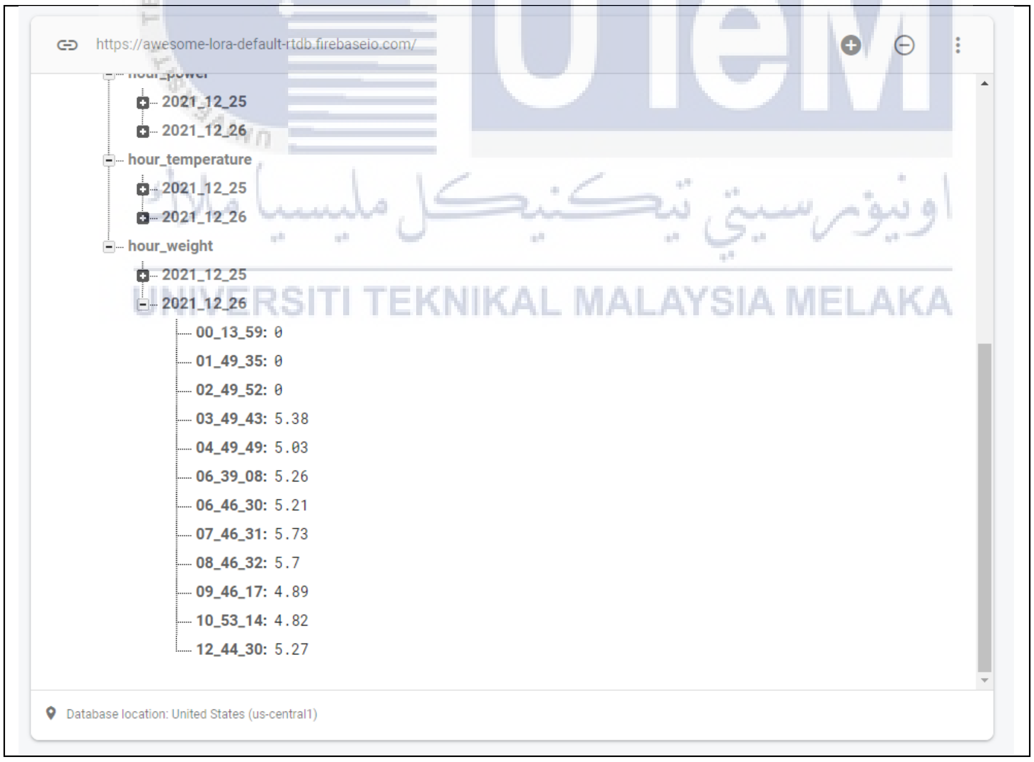
Database location: United States (us-central1)

Figure 4.7 Google Firebase Real-Time Database Root Directory



```
https://awesome-lora-default-rtdb.firebaseio.com/
  r_fuel
  r_power
  r_temperature
    00_00_00
    2021_12_25
    2021_12_26
      00_02_19: 32.7
      00_04_32: 32.6
      00_04_59: 32.7
      00_05_49: 32.7
      00_06_19: 32.8
      00_06_28: 32.8
      00_08_25: 32.8
      00_08_55: 32.8
      00_09_07: 32.8
      00_09_13: 32.8
      00_12_45: 32.8
      00_13_28: 32.9
      00_13_59: 32.9
      00_15_15: 32.8
      00_15_42: 32.9
      00_17_32: 32.9
Database location: United States (us-central1)
```

Figure 4.8 Google Firebase Real-Time Database Seconds View



```
https://awesome-lora-default-rtdb.firebaseio.com/
  hour_power
    2021_12_25
    2021_12_26
  hour_temperature
    2021_12_25
    2021_12_26
  hour_weight
    2021_12_25
    2021_12_26
      00_13_59: 0
      01_49_35: 0
      02_49_52: 0
      03_49_43: 5.38
      04_49_49: 5.03
      06_39_08: 5.26
      06_46_30: 5.21
      07_46_31: 5.73
      08_46_32: 5.7
      09_46_17: 4.89
      10_53_14: 4.82
      12_44_30: 5.27
Database location: United States (us-central1)
```

Figure 4.9 Google Firebase Real-Time Database Hours View

Figure 4.10 shows the initial page when a user launches the android application while Figure 4.11 shows the response when a user clicks on the “RealTime Monitoring button”.

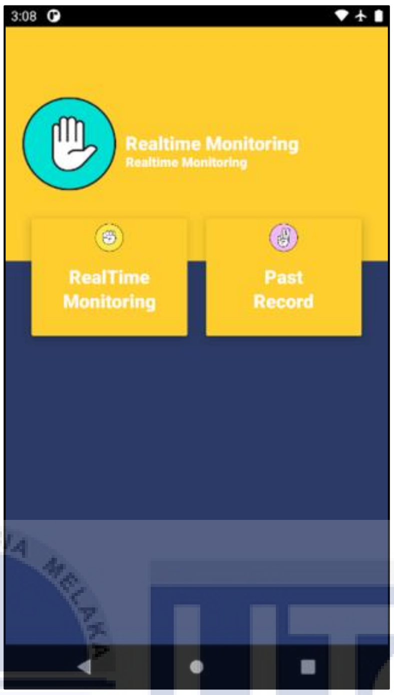
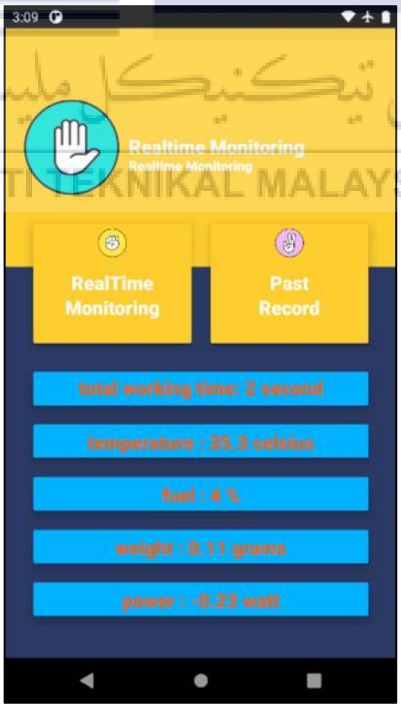


Figure 4.10 On Launch Interface



total working time: 2 second
temperature : 35.3 celsius
fuel : 4 %
weight : 0.11 grams
power : -0.23 watt

Figure 4.11 After Clicked “RealTime Monitoring” button

After the user clicks on that “RealTime Monitoring” button, the real-time data of the sensors will be shown. To view the graph, the user will need to click on the Past Record button. By default, there is a graph view without chart data. It is because the date of the graph needs to be input by the user first to show the data in graph view. On this page, the user will need to enter the date in a specific format and click submit date button. Besides that, there is a button labelled with the name “24H”, this button is used to change the data mode between seconds mode or hours mode. The default data display mode is seconds mode which shows all the recorded data in seconds view, while another option is hours mode that shows the data recorded in each hour the prototype works. Figure 4.12 shows the response when a user clicks on the Past Record button. Figure 4. shows the graph view after the date was entered by the user in a specific format.

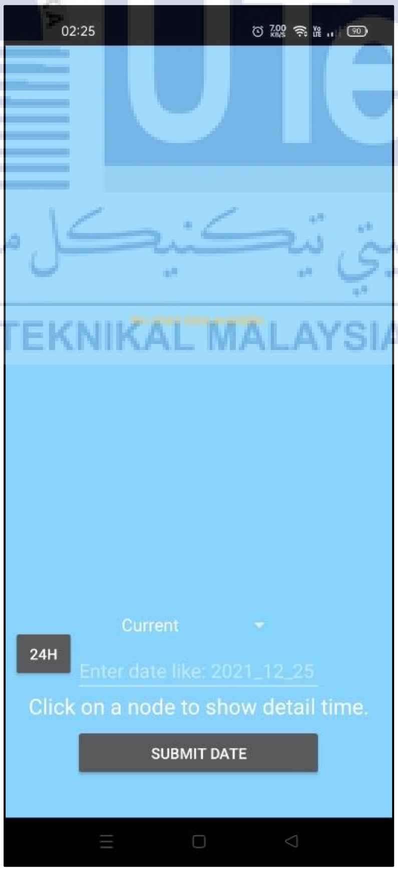

02:25
Current
24H
Enter date like: 2021_12_25
Click on a node to show detail time.
SUBMIT DATE

Figure 4.12 After Clicked Past Record button

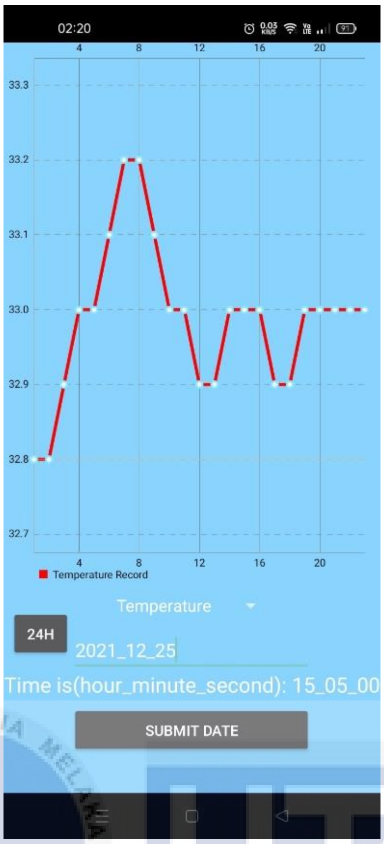


Figure 4.13 After Submitted Data in Specific Format

The user can change what sensor's data they want to monitor by clicking on the spinner. Figure 4.14 shows the sensors option in the spinner. The available option for the user to monitor is current data, fuel data, power data, temperature data, and weight data. The graph may have many data, but the user can click on a specific node to get the exact time for that data uploaded and recorded to the database. The user also can zoom in and zoom out the graph for a better view experience. The user can zoom in on the graph simply by double-tapping the graph or zooming in out by dragging and pinching the graph. Figure 4.15 shows the android application interface when the user zooms in on the graph view and selected a node. The graph in Figure 4.15 compared to the graph in Figure 4.13 is larger after zooming in, both graphs are displaying temperature data in the same date and time but just different in graph size because the latter one had been enlarged.

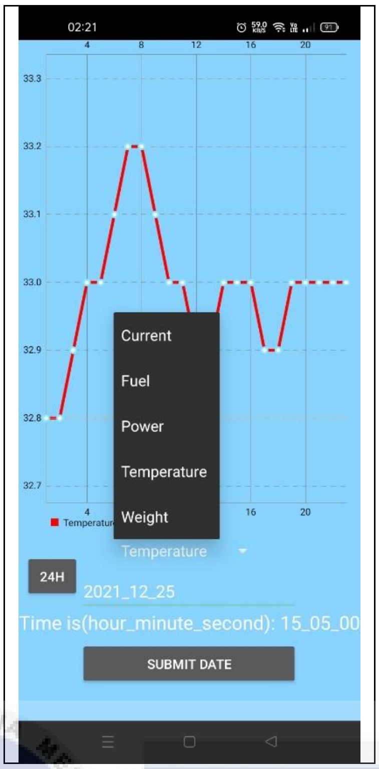


Figure 4.14 Spinner's Sensor Option

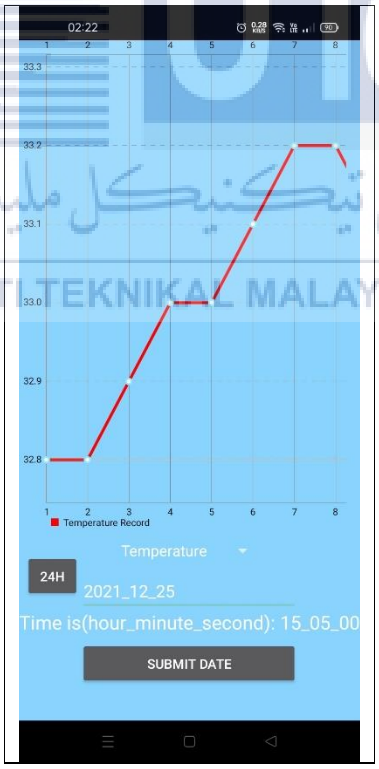


Figure 4.15 Zoom In and Selected A Node

The user can analyse the data by viewing the graph in seconds mode or hours mode. For better graph inspecting and analysing experience, the user can change the mode to show the recorded data in each hour by clicking on the "24H" button. After the user clicks on the button, the button label will show "SEC" instead of "24H", this means the data display mode had changed from seconds mode to hours mode. The following figures show the sensor's data recorded between 12.00 a.m. to 12.00 p.m. on 26 December 2021. Figure 4.16 shows the current data in per hour graph view. The current value is start with zero because at that time the load, servo motor havent connected to the current sensor. But after a while the servo motor had been connected to the current sensor and start to move periodically, so the current value increase from zero to 0.05 amphere. Figure 4.17 shows the fuel's data in the per hour graph view. The value for this fuel sensor is increasing and decreasing because the fuel data is obtain from an ultrasonic sensor. The ultrasonic sensor will detect the range by reading the reflection of the signal. The graph data shows unstable value because during the simulation, the ultrasonic sensor is place on a living room. When people passing, the value will be interrupted, this result also show that the ultrasonic sensor works without issues. If the ultrasonic sensor was place in a closed space, the value will be constant. Figure 4.18 shows the power data in the per hour graph view. For the power data, it is actually using the current sensor's data but after a multiplication with voltage value. So, the graph pattern is very similiar with the current graph. Figure 4.19 shows the temperature data in the per hour graph view. The temperature raise at 8 a.m. because at that time the weather is good and the sun is rising, then it causes the temperature to increase. But after that, weather become cloudy which cause the temperature decrease after 8 a.m.. Figure 4.20 shows the weight data in per hour graph view. In this project, 5KG load cell had been used but the load used in this project is a 5 grams load, this graph shows a good result that prove it is a 5 grams load.

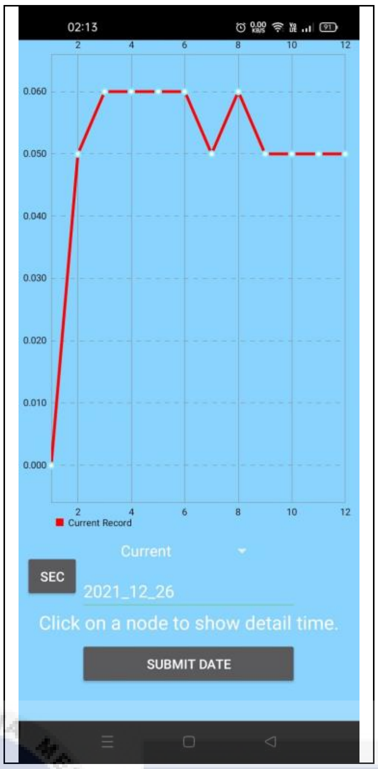


Figure 4.16   Graph Data Analysis-Current per hour

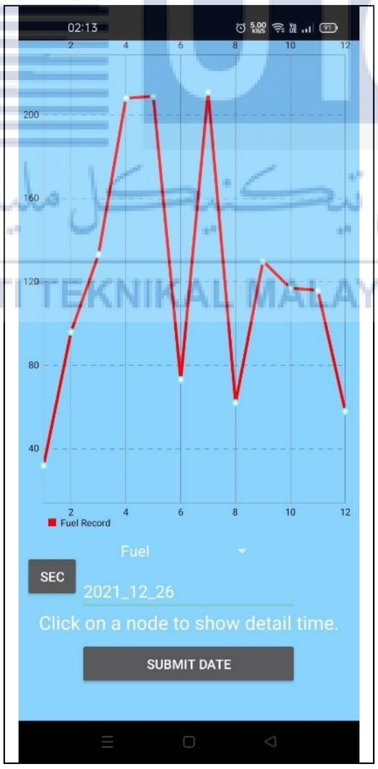
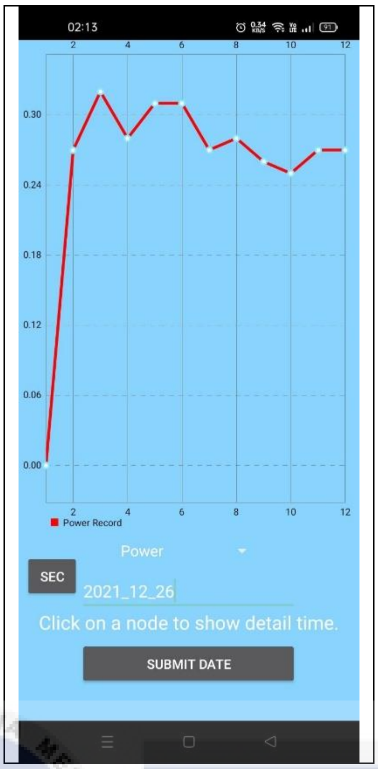
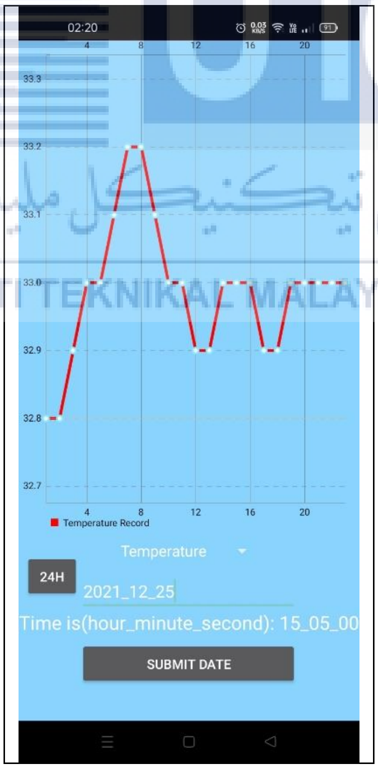


Figure 4.17   Graph Data Analysis-Fuel per hour



Click on a node to show detail time.

Figure 4.18 Graph Data Analysis-Power per hour



Time is(hour_minute_second): 15_05_00

Figure 4.19 Graph Data Analysis-Temperature per hour

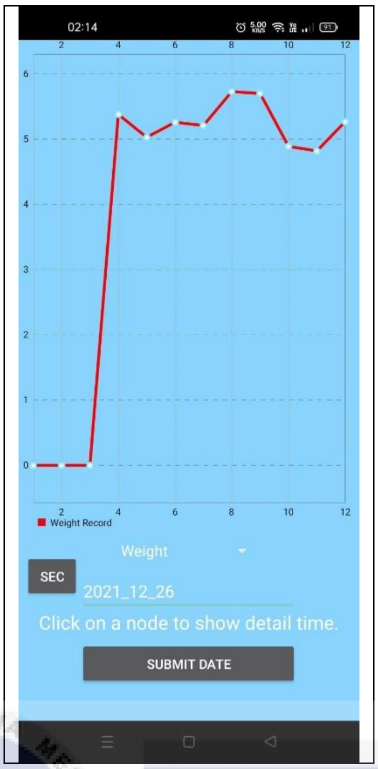


Figure 4.20 Graph Data Analysis-Weight per hour

## 4.4 Summary

In this chapter, several sensor data had been collected and shown for the result analysis. Some of the data getting from the sensors are unstable (i.e., current sensor's data, weight sensor's data, and ultrasonic sensor's data). The current sensor ACS 712 is getting data that is inaccurate because of the characteristic of itself that is very sensitive to the magnetic fields, the accuracy is of course less precise. Then, the ultrasonic sensor HC-SR04 is suitable for the fuel tank that shape is regular or symmetrical. To detect the fuel level for the fuel tank that other than this, the ultrasonic sensor HC-SR04 should be replaced with other sensors such as float level sensor. Besides that, the Google Firebase Realtime Database is not an SQL database based on the concept of documents. Hence, it is important to manage the database structure for avoiding the collected data messy. The flexibility of a no SQL database allows the developer to modify the database without providing the detailed database model before creation is one of its advantage of it.

# CHAPTER 5

# CONCLUSION

## 5.1 Conclusion

In this thesis, a method had been proposed for implementing the crane utilization monitoring system based on IoT using the LoRa transceiver module. The LoRa module that is able to receive and transfer data in a long-range that up to 10 kilometres are very suitable to implement in this system. Also, by implementing this project, the user will be able to monitor the crane utilization rate and health through the Android application. As mentioned before, this system is a simulation type prototype project, the sensors in this system should be replaced with proper sensors before being implemented in the crane. Then, when any unusual data is observed, the user can take action immediately. By doing this, the chance that an accident will happen due to the malfunction of the crane can be decreased to a minimum level. In addition, the crane owner or company can always monitor the crane and do maintenance or repairing jobs for the crane when one of the elements such as power consumption, fuel consumption or engine's temperature has an unusual value. In a conclusion, this project had successfully achieved both of the project's objectives. This project can perform well and has potential to be used in heavy engineering field that use crane in their works. By implementing this monitoring system, the crane data will not only able to be view by the crane operator but anyone that installed the mobile application in their smartphone. So, it is bmore convinient when people want to know about the crane performance anytime. Besides the monitoring function, many improvements can be done to this project to make it perform better and do other things. These future works will be discussed in the next part of this chapter.

## 5.2 Future Works

For this project, this crane utilization monitoring system is only used for monitoring the crane utilization rate and crane health. But, there will be great if the admin user can control the crane through the mobile application. For example, if the admin found that the crane is working in a bad or dangerous condition, the admin can stop the crane from working by using the mobile application. To do that, the application will need to implement a way to verify the user which is to request the user to log in to the application. By doing this, the user can be divided into many categories such as the normal user that only can monitor the data or admin that have the permission to interrupt the crane activity.

Then, some modifications can be done to make this project work on the crane for getting data from it. The sensors that had been used in this project are not the type of sensors that are suitable for measuring the real crane data, so they should be changed to the proper ones. Next, to improve the connectivity and coverage range of the LoRa module, it can be enhanced by using a better antenna.

# REFERENCES

Akmal, N. (2020) ‘Development of Software Weight Monitoring Waste Bin Based on Internet of Things ( IOT )’, 0(0), pp. 341–352.

Anusha, A. and Ahmed, S. M. (2017) ‘Vehicle Tracking and Monitoring System to Enhance the Safety and Security Driving Using IoT’, *Proceedings - 2017 International Conference on Recent Trends in Electrical, Electronics and Computing Technologies, ICRTEECT 2017*, 2017-Decem, pp. 49–53. doi: 10.1109/ICRTEECT.2017.35.

Hossain, M. S. *et al.* (2019) ‘A smart IoT based system for monitoring and controlling the sub-station equipment’, *Internet of Things*, 7, p. 100085. doi: 10.1016/j.iot.2019.100085.

Jadhav, A. R. and Rajalakshmi, P. (2017) ‘IoT enabled smart and secure power monitor’, *TENSYMP 2017 - IEEE International Symposium on Technologies for Smart Cities*. doi: 10.1109/TENCONSpring.2017.8070096.

K, S. (2017) ‘Smart Bus Tracking and Management System using Iot’, *International Journal for Research in Applied Science and Engineering Technology*, V(III), pp. 372–374. doi: 10.22214/ijraset.2017.3067.

Li, B. *et al.* (2017) ‘Tower Crane Remote Wireless Monitoring System Based on Modbus/TCP Protocol’, *Proceedings - 2017 IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017*, 2, pp. 187–190. doi: 10.1109/CSE-EUC.2017.217.

Prathibha, S. R., Hongal, A. and Jyothi, M. P. (2017) ‘IOT Based Monitoring System in Smart Agriculture’, *Proceedings - 2017 International Conference on Recent Advances in Electronics and Communication Technology, ICRAECT 2017*, pp. 81–84. doi:

10.1109/ICRAECT.2017.52.

Rahim Abdul Hamid, A. *et al.* (2019) ‘Causes of crane accidents at construction sites in Malaysia’, *IOP Conference Series: Earth and Environmental Science*, 220(1). doi: 10.1088/1755-1315/220/1/012028.

Tejesh, B. S. S. and Neeraja, S. (2018) ‘Warehouse inventory management system using IoT and open source framework’, *Alexandria Engineering Journal*, 57(4), pp. 3817–3823. doi: 10.1016/j.aej.2018.02.003.

Tian Xun Ng (2018) ‘A “missing” family of classical orthogonal polynomials’, *GARBAGE BIN MONITORING FOR SMART RESIDENCE*, 44(8), pp. 1689–1699. doi: 10.1088/1751-8113/44/8/085201.

Zourmand, A. *et al.* (2019) ‘Internet of Things (IoT) using LoRa technology’, *2019 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2019 - Proceedings*, (June), pp. 324–330. doi: 10.1109/I2CACIS.2019.8825008.

# APPENDICES

## Appendix A Transmitter Side Arduino IDE(Arduino UNO) coding

```
#include <SPI.h>
#include <LoRa.h>
#include <DHT.h>
#include <DHT_U.h>
#include "HX711.h"
#include "RTClib.h"

#define echoPin 5 // attach pin D5 Arduino to pin Echo of HC-SR04
#define trigPin 6 //attach pin D6 Arduino to pin Trig of HC-SR04
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement

#define DHTPIN 7 //dht22 pin
#define DHTTYPE DHT22//define sensor type
DHT dht(DHTPIN, DHTTYPE);//set pin and dht sensor model

#define VIN A0 //current sensor pin
const float VCC   = 5.0;// supply voltage is from 4.5 to 5.5V. Normally 5V.
const int model = 0;
const float QOV =   0.5 * VCC;// set quiescent Output voltage of 0.5V

HX711 scale(4, 3);//hx711 load cell amplifier pin

RTC_DS3231 rtc;//rtc module declare

String yearData,monthData,dayData,hourData,minuteData,
      secondData,dateData,timeData,starting_dateData,starting_timeData;
long startingSeconds,currentSeconds;

float calibration_factor = -711; // this calibration factor is adjusted according to my load
cell
float units,ounces;
float temp;
float cutOffLimit = 0;
float sensitivity[] ={
         0.185,// for ACS712ELCTR-05B-T
         0.100,// for ACS712ELCTR-20A-T
         0.066// for ACS712ELCTR-30A-T
        };
float voltage,current,power;// internal variable for voltage
```

```
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(115200);
  dht.begin();

  while (!Serial);
  Serial.println("LoRa Sender");
  if (!LoRa.begin(433E6)) { // or 915E6, the MHz speed of yout module
    Serial.println("Starting LoRa failed!");
    while (1);
  }

  scale.set_scale();
  scale.tare();  //Reset the scale to 0

   if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }
  if (rtc.lostPower()) {
    Serial.println("RTC lost power, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }
  trackTime();
  starting_dateData=dateData;
  starting_timeData=timeData;
  startingSeconds=currentSeconds;
}

void loop() {
  current_sensor();
  temperature_read();
  weighting();
  trackTime();
  detectFuel();
  Serial.print("Total Working Time(seconds): ");
  Serial.println(calculateTime(startingSeconds,currentSeconds));
  //delay(2000);
  Serial.println();

  LoRa.beginPacket();LoRa.println();
  LoRa.print("Current    :<");LoRa.print(current);LoRa.print("> A\t\t");
  LoRa.print("Power      :(");LoRa.print(power);LoRa.print(") W\t\n");
  LoRa.print("temperature:[");LoRa.print(temp);LoRa.println("] *C\t");
  LoRa.print("weight:+");LoRa.print(units);LoRa.print("- grams\n");
  LoRa.print("Fuel left:@");LoRa.print(distance);LoRa.print("# percent\n");//distance
need  adjust with range to get percent, now only distance
  LoRa.print("Current Date :あ");LoRa.print(yearData);LoRa.print("い");
  LoRa.print(monthData);LoRa.print("う");
  LoRa.print(dayData);LoRa.print("え \t\n");
  LoRa.print("Current Time :お");LoRa.print(hourData);LoRa.print("か");
  LoRa.print(minuteData);LoRa.print("き");
  LoRa.print(secondData);LoRa.print("く \t\n");
  LoRa.print("Working Time
:~");LoRa.print(calculateTime(startingSeconds,currentSeconds));LoRa.print("`
seconds\t\n");//new add
  LoRa.endPacket();
}

void temperature_read(){
  temp = dht.readTemperature();
  Serial.print(temp); Serial.println(" *C\t");
  }

void weighting(){

  scale.set_scale(calibration_factor); //Adjust to this calibration factor
  Serial.print("Reading: ");
  units = scale.get_units(), 10;
  if (units < 0)
  {
    units = 0.00;
  }
  //ounces = units * 0.035274;
  Serial.print(units);
  Serial.print(" grams");
  Serial.println();
}

void current_sensor(){

  for(int i=0;i<1000;i++){
  float voltage_raw =   (5.0 / 1023.0)* analogRead(VIN);// Read the voltage from sensor
  voltage = voltage+ voltage_raw ;// 0.006 is a value to make voltage zero when there is
no current
  delay(1);
  }
  voltage=voltage/1000-QOV-0.007;
  current = voltage / sensitivity[model];
  power = current*5;
  if(abs(current) > cutOffLimit ){

    Serial.print("Current: ");
    Serial.print(current,3); // print the current with 2 decimal places
    Serial.println("A");
    Serial.print("POWER: ");
    Serial.print(power,3);
    Serial.println("W");
  }else{
    Serial.println("No Current");
  }
  delay(1);
}

void trackTime(){
    DateTime now = rtc.now();

    yearData  =now.year()  ;  monthData =now.month() ;
    dayData   =now.day()   ;  hourData  =now.hour()  ;
    minuteData=now.minute();  secondData=now.second();
    dateData=monthData+'.'+dayData;
    timeData=minuteData+'.'+secondData;
    currentSeconds=now.unixtime();
    Serial.println("Starting Date: "+starting_dateData);
    Serial.println("Starting Time: "+starting_timeData);
    Serial.println("Current Date: "+yearData+'.'+dateData);
    Serial.println("Current Time: "+hourData+'.'+timeData);
    delay(1);
}

long calculateTime(long startAt, long endAt){
  long duration=endAt-startAt;
  return duration;
}

void detectFuel(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  //delay(1000);
}
```

## Appendix B Receiver Side Arduino IDE(NodeMCU) coding

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <LoRa.h>

// Set these to run example.
#define FIREBASE_HOST "awesome-lora-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "V2oU7wmf09bit8eCYm4hlIfYtfuuuL2MbgHzKBrR"
#define WIFI_SSID "no pay no gain"
#define WIFI_PASSWORD "huhu545225??"
#define ss 15//ss pin
#define rst 16
#define dio0 9

unsigned long lastMillis = 0;
unsigned long millisSend = 0;

uint32_t timer_1seconds = 0;
bool done;
char receivedChars[32];   // an array to store the received data
String msg,mergeTime;
float current,watt,temp,gram,fuel;
long duration,i=0;
int n = 0;
void setup() {
  Serial.begin(115200);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Serial.println("LoRa Receiver");
  LoRa.setPins(ss, rst, dio0);
  if (!LoRa.begin(433E6))
  {
    Serial.println("Starting LoRa failed!");
    delay(100);
    while (1);
  }
  Serial.println("LoRa Started");
}



void loop() {

  onReceive(LoRa.parsePacket());
  current=catchData('<','>');
  watt=catchData('(',')');
  temp=catchData('[',']');
  gram=catchData('+','-');
  duration=catchData('~','`');
  fuel=catchData('@','#');

  if(current<0)//if negative show 0
    current=0;
  if(watt<0)
    watt=0;
  if(gram<0)
    gram=0;

  Firebase.setFloat("Current", current);
  Firebase.setFloat("Power", watt);
  Firebase.setFloat("Temperature", temp);
  Firebase.setFloat("Weight", gram);
  Firebase.setFloat("Working_duration", duration);
  Firebase.setFloat("Fuel", fuel);
  updateDATA();
  yield();
  updateDATA_hour();

}

void onReceive(int packetSize) {
  if (packetSize == 0) return;// if there's no packet, return

  // read packet header bytes:
  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }
  msg=incoming;
  Serial.println("Message: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
}

float catchData(char opening, char closing){
  static byte ndx = 0;
  int starting,ending;
  String value;
  for(int i=0;i<msg.length();i++){
    if(msg.charAt(i)==opening){
      starting=i+1;
    }
    if(msg.charAt(i)==closing){
      ending=i-1;
    }
  }

  for(int i=0;i<=ending-starting;i++){
    receivedChars[i]=msg.charAt(starting+i);
  }
  value=String((char*)receivedChars);
  memset(receivedChars,0,sizeof(receivedChars));
  yield();
  return value.toFloat();

}

void readTime(char opening,char middle1,char middle2, char closing){//data process

  int a,b,c;
  String x,y,z;
  a=(int)catchData(opening,middle1);
  b=(int)catchData(middle1,middle2);
  c=(int)catchData(middle2,closing);
  if(a<10)//if data only 1 digit, make it 2digit,like 1 become 01
    x="0"+String(a);
  else
    x=String(a);

  if(b<10)
    y="0"+String(b);
  else
    y=String(b);

  if(c<10)
    z="0"+String(c);
  else
```

```
   z=String(c);

  mergeTime=x+"_"+y+"_"+z;

}



void updateDATA(){


      String a,b,c,d,e,f,g,h,i,j,temp1,temp2,temp3,temp4,temp5;

      readTime('あ','い','う','え');//catch data between these symbol
      String date=mergeTime;

      readTime('お','か','き','く');
      String timeNOW='/'+mergeTime;
      //seperate time here with _

      temp1=("record/r_current/");//database reference
      temp2=("record/r_power/");
      temp3=("record/r_temperature/");
      temp4=("record/r_weight/");
      temp5=("record/r_fuel/");

      a=temp1+date+timeNOW;
      b=temp2+date+timeNOW;
      c=temp3+date+timeNOW;
      d=temp4+date+timeNOW;
      e=temp5+date+timeNOW;

      Firebase.setFloat(a, current);
      Firebase.setFloat(b, watt);
      Firebase.setFloat(c, temp);
      Firebase.setFloat(d, gram);
      Firebase.setFloat(e, fuel);


}

void updateDATA_hour(){

      //read from database worktime =x
      //
      if(millis()-lastMillis>3600000){
      lastMillis=millis();
```

```
    String a,b,c,d,e,f,g,h,i,j,temp1,temp2,temp3,temp4,temp5;

    readTime('あ','い','う','え');
    String date=mergeTime;

    readTime('お','か','き','く');
    String timeNOW='/'+mergeTime;

    temp1=("record_per_hour/hour_current/");
    temp2=("record_per_hour/hour_power/");
    temp3=("record_per_hour/hour_temperature/");
    temp4=("record_per_hour/hour_weight/");
    temp5=("record_per_hour/hour_fuel/");

    a=temp1+date+timeNOW;
    b=temp2+date+timeNOW;
    c=temp3+date+timeNOW;
    d=temp4+date+timeNOW;
    e=temp5+date+timeNOW;

    Firebase.setFloat(a, current);
    Firebase.setFloat(b, watt);
    Firebase.setFloat(c, temp);
    Firebase.setFloat(d, gram);
    Firebase.setFloat(e, fuel);

    }

}
```

## Appendix C Android Application Code (MainActivity.java)

```
package com.example.fyplora;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
    private FirebaseDatabase database;
    private DatabaseReference fuelValue, powerValue, temperatureValue,
weightValue, workingTime;

    TextView fuel, power, temperature, weight, workTime;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fuel = findViewById(R.id.valueONE);
        power = findViewById(R.id.valueTWO);
        temperature = findViewById(R.id.valueThree);
        weight = findViewById(R.id.valueFour);
        workTime = findViewById(R.id.valueFive);

        database = FirebaseDatabase.getInstance();
        fuelValue = database.getReference("Fuel");//database reference
        powerValue = database.getReference("Power");
        temperatureValue = database.getReference("Temperature");
        weightValue = database.getReference("Weight");
        workingTime = database.getReference("Working_duration");


        fuelValue.addValueEventListener(new ValueEventListener() {//function
to get data from database
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String data = snapshot.getValue().toString();
                fuel.setText("fuel : " + data + " %");//set text to show
database data

            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "The reading action failed:
" + error.getCode(), Toast.LENGTH_LONG);
            }
        });

        powerValue.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String data = snapshot.getValue().toString();
                power.setText("power : " + data + " watt");
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "The reading action
failed: " + error.getCode(), Toast.LENGTH_LONG);
            }
        });

        temperatureValue.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String data = snapshot.getValue().toString();
                temperature.setText("temperature : " + data + " celsius");
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "The reading action
failed: " + error.getCode(), Toast.LENGTH_LONG);
            }
        });

        weightValue.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String data = snapshot.getValue().toString();
                weight.setText("weight : " + data + " grams");
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "The reading action
failed: " + error.getCode(), Toast.LENGTH_LONG);
            }
        });

        workingTime.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                String data = snapshot.getValue().toString();
                workTime.setText("total working time: " + data + " second");
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Toast.makeText(MainActivity.this, "The reading action
failed: " + error.getCode(), Toast.LENGTH_LONG);
            }
        });


    }

    public void goGraphMenu(View view) {
        Intent intent = new Intent(MainActivity.this, GraphMenu.class);//if
clicked button start this intent
        startActivity(intent);
    }


    public void showValue(View view) {
        if (findViewById(R.id.dataBOX).getVisibility() == View.VISIBLE)
{//show or hide real time data when clicked this button
            findViewById(R.id.dataBOX).setVisibility(View.INVISIBLE);
        } else if (findViewById(R.id.dataBOX).getVisibility() ==
View.INVISIBLE) {
            findViewById(R.id.dataBOX).setVisibility(View.VISIBLE);
        }
    }
}
```

## Appendix D Android Application Code (GraphMenu.java)

```
package com.example.fyplora;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.highlight.Highlight;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;

import com.github.mikephil.charting.listener.OnChartValueSelectedListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class GraphMenu extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

    private static final String TAG = "GraphMenu";

    String userInput = "000";
    FirebaseDatabase recordDatabase;
    DatabaseReference current, fuel, power, temperature, weight,
currentPerHour, fuelPerHour, powerPerHour, temperaturePerHour, weightPerHour;
    ArrayList<String> list = new ArrayList<>();
    ArrayList<String> keyList = new ArrayList<>();
    ArrayList<Entry> yValues = new ArrayList<>();
    private LineChart lineChart;
    private TextView time;
    private EditText textEdit;
    private Spinner spinner;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_graph_menu2);
        recordDatabase = FirebaseDatabase.getInstance();
        current = recordDatabase.getReference("record/r_current");
        fuel = recordDatabase.getReference("record/r_fuel");
        power = recordDatabase.getReference("record/r_power");
        temperature = recordDatabase.getReference("record/r_temperature");
        weight = recordDatabase.getReference("record/r_weight");

        time = findViewById(R.id.textViewshowtime);
        textEdit = findViewById(R.id.editTextInsertDate);
        lineChart = findViewById(R.id.linechartdemo);

        spinner = findViewById(R.id.spinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.spinner_options,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(this);

        lineChart.setDragEnabled(true);
        lineChart.setScaleEnabled(true);

        YAxis leftAxis = lineChart.getAxisLeft();
        leftAxis.removeAllLimitLines();

        leftAxis.enableGridDashedLine(10f, 10f, 0);
        leftAxis.setDrawLimitLinesBehindData(true);

        lineChart.getAxisRight().setEnabled(false);

    }

    public void searchFirebaseRecord(View view) {
        list.clear();
        keyList.clear();
        yValues.clear();
        String text = spinner.getSelectedItem().toString();
        userInput = textEdit.getText().toString();

        if (text.equalsIgnoreCase("Current")) {
            createGraph(current);
        } else if (text.equalsIgnoreCase("temperature")) {
            createGraph(temperature);
        } else if (text.equalsIgnoreCase("fuel")) {
            createGraph(fuel);
        } else if (text.equalsIgnoreCase("power")) {
            createGraph(power);
        } else if (text.equalsIgnoreCase("weight")) {
            createGraph(weight);
        }
    }

    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i,
long l) {//spinner
        String text = adapterView.getItemAtPosition(i).toString();
        if (text.equalsIgnoreCase("Current")) {
            createGraph(current);
        } else if (text.equalsIgnoreCase("temperature")) {
            createGraph(temperature);
        } else if (text.equalsIgnoreCase("fuel")) {
            createGraph(fuel);
        } else if (text.equalsIgnoreCase("power")) {
            createGraph(power);
        } else if (text.equalsIgnoreCase("weight")) {
            createGraph(weight);
        }
        Button swap24 = findViewById(R.id.button24);
        Button swapSeconds = findViewById(R.id.buttonSeconds);
        swap24.setEnabled(true);
        swap24.setVisibility(View.VISIBLE);
        swapSeconds.setEnabled(false);
        swapSeconds.setVisibility(View.INVISIBLE);
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
    }

    public void createGraph(DatabaseReference nodeName) {//function to create
graph
        list.clear();
        keyList.clear();
        yValues.clear();
        userInput = textEdit.getText().toString();//store user input
        try {
            nodeName.child(userInput).addListenerForSingleValueEvent(new
ValueEventListener() {//get data by searching user input in database
                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{//retrieve data from database
                        for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                            list.add(snapshot.getValue().toString());
                            keyList.add(snapshot.getKey());
                        }

                        try {
                            for (int i = 0; i < list.size(); i++) {//set node for
all data getting from database
                                float f = Float.parseFloat(list.get(i));//string
to float
                                yValues.add(new Entry(i + 1, f));
                                LineDataSet set1 = new LineDataSet(yValues,
spinner.getSelectedItem().toString() + " Record");//show variable label in
bottom left
```

```
                            XAxis xAxis = lineChart.getXAxis();
                            xAxis.setGranularity(1);

xAxis.setPosition(XAxis.XAxisPosition.BOTH_SIDED);

                            set1.setFillAlpha(110);
                            set1.setColor(Color.RED);
                            set1.setLineWidth(3f);
                            set1.setValueTextColor(Color.CYAN);
                            set1.setValueTextSize(10f);
                            set1.setDrawValues(false);

                            ArrayList<ILineDataSet> dataSets = new
ArrayList<>();
                            dataSets.add(set1);

                            LineData data = new LineData(dataSets);

                            lineChart.setDescription(null);
                            lineChart.setData(data);
                            lineChart.invalidate();//refresh the page for
chart
                            lineChart.setTouchEnabled(true);
                            lineChart.setOnChartValueSelectedListener(new
OnChartValueSelectedListener() {
                                @Override
                                public void onValueSelected(Entry e,
Highlight h) {

                                    try {
                                        Float split = e.getX();//get database
key
                                        int splitInt =
Math.round(split);//convert float to int
                                        time.setText("Time
is(hour_minute_second): " + keyList.get(splitInt));//show record time in the
apps
                                    } catch (Exception exception) {
                                    }
                                }

                                @Override
                                public void onNothingSelected() {
                                }
                            });
                        }
                    } catch (Exception e) {

                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {

                }
            });
```

```
        } catch (Exception e) {
        }
    }


    public void swap24hour(View view) {//button for switch mode to seconds
        Button swap24 = findViewById(R.id.button24);
        Button swapSeconds = findViewById(R.id.buttonSeconds);
        swap24.setEnabled(false);
        swap24.setVisibility(View.INVISIBLE);
        swapSeconds.setEnabled(true);
        swapSeconds.setVisibility(View.VISIBLE);
        Spinner mySpinner = findViewById(R.id.spinner);
        String text = mySpinner.getSelectedItem().toString();

        currentPerHour =
recordDatabase.getReference("record_per_hour/hour_current");//database
reference for hours data
        fuelPerHour =
recordDatabase.getReference("record_per_hour/hour_fuel");
        powerPerHour =
recordDatabase.getReference("record_per_hour/hour_power");
        temperaturePerHour =
recordDatabase.getReference("record_per_hour/hour_temperature");
        weightPerHour =
recordDatabase.getReference("record_per_hour/hour_weight");

        if (text.equalsIgnoreCase("current"))
            createGraph(currentPerHour);
        else if (text.equalsIgnoreCase("fuel"))
            createGraph(fuelPerHour);
        else if (text.equalsIgnoreCase("power"))
            createGraph(powerPerHour);
        else if (text.equalsIgnoreCase("temperature"))
            createGraph(temperaturePerHour);
        else if (text.equalsIgnoreCase("weight"))
            createGraph(weightPerHour);
    }


    public void swapSeconds(View view) {//button for switch mode to hours
        Button swap24 = findViewById(R.id.button24);
        Button swapSeconds = findViewById(R.id.buttonSeconds);
        swap24.setEnabled(true);
        swap24.setVisibility(View.VISIBLE);
        swapSeconds.setEnabled(false);
        swapSeconds.setVisibility(View.INVISIBLE);

        Spinner mySpinner = findViewById(R.id.spinner);
        String text = mySpinner.getSelectedItem().toString();
        if (text.equalsIgnoreCase("Current")) {
            createGraph(current);
        } else if (text.equalsIgnoreCase("temperature")) {
            createGraph(temperature);
        } else if (text.equalsIgnoreCase("fuel")) {
            createGraph(fuel);
        } else if (text.equalsIgnoreCase("power")) {
            createGraph(power);
        } else if (text.equalsIgnoreCase("weight")) {
            createGraph(weight);
        }
    }
}
```