INVESTIGATION OF THE EFFECTIVENESS OF A DISASTER ALERT SYSTEM

CHUA LI YING

BACHELOR OF MECHATRONICS ENGINEERING

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

"I hereby declare that I have read through this report entitle "**Investigation of the Effectiveness of a Disaster Alert System**" and found that it has comply the partial fulfilment for awarding the degree of Bachelor of Mechatronics Engineering.

Signature	:	
Supervisor"s Name	:	Professor Madya Dr. Ahmad Zaki Bin Shukor
Date	:	6 June 2018

INVESTIGATION OF THE EFFECTIVENESS OF A DISASTER ALERT SYSTEM

CHUA LI YING

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Mechatronics Engineering

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2017/2018

I declare that this report entitle "*Investigation of the Effectiveness of a Disaster Alert System*" is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	
Name	:	CHUA LI YING
Date	:	6 June 2018

To my beloved mother and father

ACKNOWLEDGEMENT

First and foremost, I wish to express my greatest appreciation and deepest gratitude to my supervisor, Professor Madya Dr. Ahmad Zaki Bin Shukor for constantly guiding and encouraging me throughout the entirety of this final year project. Thanks a lot for giving me valuable advice and suggestion to bring this report to its final form. Without his support and interest, this report would not have been the same as presented here. I am very grateful to him for his patience and his constructive comments that enriched this project.

Next, deepest sense of gratitude is given to the Director of Melaka Meteorological Office, Mr. Nasrul Hakim Bin Hashim who willing to give explanation on the disaster alert system in Malaysia. In particular, my sincere thanks is also extends to all of my friends for their generous efforts and assistances provided in enlightening me. Their views and tips are useful indeed. I would like to acknowledge their comments and suggestions, which was crucial for the successful completion of this project. Also, I would take this opportunity to express my gratitude to my parent for their continuous shower of love, unceasing encouragement and support throughout all these years.

Last but not least, I place on record, my sense of gratitude to one and all who, directly or indirectly, have offered their helping hand during the entire period of final year project.

ABSTRACT

Earthquake is the most dreadful phenomenon among the 5 types of most common natural disaster. The existing earthquake alert system is costly and there may be faulty detections. Therefore, a low cost Internet of Thing (IoT) based real time earthquake early alert system is proposed where the system could estimate the earthquake epicenter and P-wave arrival time in an area around 25 kilometers apart from 3 sensor nodes. A disaster messaging system is develop which aims to send alert information and a site recorded video to Telegram channel for validation purpose when one of the accelerometers in 3 sensor node detects acceleration exceeds threshold value of 0.05 m/s^2. The time for delivering the alert message with photo and video attachment of the proposed system is calculated to check its efficiency. The map which illustrate the estimated epicenter coordinate and estimated P-wave arrival time using ThingSpeak MATLAB Visualization feature will then be sent to the Telegram channel for better understanding purpose. 3 shaking tables will be built to verify the proposed system through earthquake simulation at 6 different preset epicenter locations at each sensor node. Through the experiments, the final alert warning will be sent to Telegram channel within 68.1 seconds and earns an average short period of 35 seconds respond time for immediate evacuation or other purposes before P-wave hits the destination.

ABSTRAK

Gempa bumi adalah fenomena paling dahsyat di antara 5 jenis bencana alam yang paling biasa berlaku. Sistem amaran gempa bumi yang sedia ada adalah mahal dan berkemungkinan memberi data yang salah. Oleh itu, satu sistem amaran awal gempa bumi yang dibina dengan kos yang rendah dicadangkan di mana sistem tersebut boleh meramalkan lokasi gempa bumi dan menganggarkan masa impak P-gelombang di kawasan sekitar 25 kilometer jauh daripada 3 nod pengesan. Satu sistem pemesejan bencana dibangunkan untuk menghantar maklumat amaran dan tapak video yang dirakam ke Telegram untuk tujuan pengesahan apabila salah satu pecutan dalam 3 nod sensor mengesan pecutan melebihi 0.05 m/s^2. Masa untuk menyampaikan mesej amaran dengan lampiran foto dan video dikira untuk memeriksa kecekapan sistem yang dicadangkan. Peta yang menggambarkan lokasi gempa bumi dan masa ketibaan P-gelombang menggunakan ThingSpeak MATLAB Visualization akan dihantar ke Telegram untuk tujuan pemahaman yang lebih baik. 3 meja goncang akan dibina ke setiap nod pengesan untuk mengesahkan sistem yang dicadangkan melalui simulasi gempa bumi di 6 lokasi yang berbeza. Melalui eksperimen, amaran akhir akan dihantar ke Telegram dalam masa 68.1 saat dan memperoleh tempoh masa pendek purata 35 saat untuk bertindak atau tujuan lain sebelum gelombang P menemui destinasi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	-	Analog-to-Digital Converter
USB	-	Universal Serial Bus
CPU	-	Central Processing Unit
P-wave	-	Primary Wave
S-wave	-	Secondary Wave
GPIO	-	General Purpose Input/Output
SMS	-	Short Messaging Service
HDMI	-	High Definition Multimedia Interface
IoT	-	Internet of Things
IFRC	-	International Federation of Red Cross and Red Crescent Societies
UNISDR	-	United Nations Office for Disaster Risk Reduction
EM-DAT	-	International Disaster Database
CRED	-	Centre for Research on the Epidemiology of Disasters
WSAN	-	Wireless Sensor and Actor Network
WSN	-	Wireless Sensor Network
CSI	-	Camera Serial Interface
LAN	-	Local Area Network
GSM	-	Global System for Mobile Communications
AVI	-	Audio Video Interleave Video
MP4	-	MPEG-4 (Motion Picture Expert Group 4) Video
MKV	-	Matroska Video
GIF	-	Graphics Interchange Format

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

There are five subtopics to be presented in this chapter, which includes motivation, problem statement, objective and scope. Related statistical data will be shown to support the motivation of this project. Besides, limitation on current technology will also be discussed. Last but not least, the aim and research boundary of the project will also be discussed in this chapter.

1.2 Motivation

A natural disaster is known to hit any part of the world without any prior warning due to natural processes of the earth which includes cyclone, flood, earthquake and other geological processes [1]. In recent years, countless people died, injured or become homeless because of such disasters. Regardless of the cause of incident, disaster causes destruction in terms of economic and human lives in the history of mankind. Table 1.1 below shows the number of reported natural disaster and number of deaths, by type of phenomenon occurs worldwide between years 2011 until 2015.

Table 1.1: Total number of reported natural disaster and number of deaths, by type of phenomenon occur worldwide between years 2011 until 2015 [2].

Type / Criteria	Floods	Storms	Earthquakes	Droughts	Extreme Temperatures
Number of reported disaster	744	495	134	123	117
Number of deaths	26,529	17,495	33,076	10,035	13,048

Based on Table 1.1 shown above, there are 5 types of most common disaster, and earthquake results the highest total number of deaths although it was third highest reported.

Every year, at the global level, there are about 2 earthquakes above or equal to Richter scale of 7 and nearly 150 earthquakes of magnitude above Richter scale of 6 are reported occurred in inhabited regions, stated by the researchers at the Paris Institute of Earth Physics (IPGP) [3].

Recently, a major earthquake with Richter scale of 7.1 struck southern Mexico on 19 September 2017, 123km from Mexico City, in Puebla state which killed at least 230 people as shown in the Figure 1.1 below.

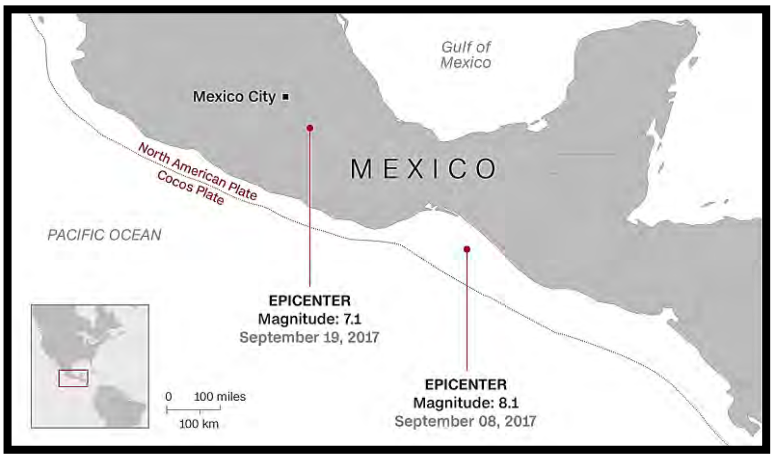


Figure 1.1: Earthquake disasters in Mexico [4].

The earthquake took place on the 32nd anniversary of a devastating earthquake that killed thousands in Mexico City in 1985, and came less than 2 weeks after another massive earthquake with a Richter scale of 8.1 killed at least 96 people and left 2.5 million in the need of aid in the state of Oaxaca. According to the government, more than 40 buildings in the country‟s capital, Mexico City near the quake‟s epicenter, have completely collapsed, with thousands more left damaged and unstable [4].

In year 2016, earthquake has again reminded the global population its existence by achieving top 2nd and 3rd rank on the 5 deadliest natural disasters happened in 2016 at Ecuador and Italy. On April, a powerful earthquake with a Richter scale of 7.8 hit the northwestern coast of Ecuador, South American nation, leaving nearly 300 dead and thousands injured. After a few months, another powerful earthquake disaster rattled central Italy on August. With a Richter scale of 6.2, more than 200 people are killed and more than 1000 people have been displaced by the quake [5].

In terms of seismic activity, Malaysia is classified as a country with low to medium seismic activity level. However, the seismic risk, in terms of damage potential should not be ignored since there have been numerous tremors on Malaysian soil due to the earthquakes disaster happened in nearby country over the past decade. Recently, an earthquake with a Richter scale of 6.4 hit southern Sumatra, Indonesia on 13 August 2017 and causes tremors felt in parts of Johor, Melaka and Singapore. Luckily, no injuries or deaths were reported [6]. In 5 June 2015, an earthquake with a Richter scale of 6.0 strikes Ranau, Sabah. It is the strongest earthquake ever recorded in Malaysia which kills 18 climbers due to the tremors felt during their climbing on Mount Kinabalu [7].

In conclusion, an efficient disaster management system is needed; especially on earthquake detection due to the harms it brings to the world. Occurrence of earthquake in Malaysia is less than other countries. However, Malaysia should have an up-to-date disaster management system to avoid earthquake such as the tragic incident 2015 at Sabah.

1.3 Problem Statement

With the aids of available technologies and preparation, alert messages are now able to be spread out when an earthquake is sensed nearby. The earthquake alert system implemented in Japan can even inform the citizen the area that will be hit by earthquake impact. However, due to its extremely high cost at around 1 billion dollars for implementation of a dense network with 1000 seismographs throughout the entire country to rapidly detect earthquakes, this system has not yet been popularized to other country, especially Malaysia. Besides, Taiwan has developed an earthquake early warning system which costs around 1.01 million dollars based on Japan‟s system. Over 700 strong-quake and 100 real-time monitoring stations have been set up nationwide. Thus, a low cost earthquake warning model with ability to estimate earthquake epicenter and arrival time should be designed to meet the needs of those countries which have low possibility of strong earthquake happening.

Next, validation is one of the main concerns of a disaster warning system. High false alarm rate of the system is triggered by faulty detection that may occur as a result of noise from accidents, lightning or device failure which in turns activates false alert. This will not only undermine public confidence towards the impact alerts, but also a waste of money due to unnecessary evacuation. One of the example is the false alert happened in 5th January 2018 in Japan which causes panic to millions of citizens and disrupted Tokyo‟s transport network. Therefore, an alert message should be sent to local authorities with an attachment of media file like video as a proof of an earthquake hit to avoid unnecessary wastes.

An effective disaster alert system depends on how quickly the situation is notified to the right people. Even a few additional seconds of warning can make a huge difference in saving hundreds or even thousands of human lives. The Japanese system gives citizens an average of 30 to 50 seconds warning while the Taiwan system allows at most a 10 seconds warning since the seismic activity take place closer to the island. On the other hand, Malaysia earthquake alert system takes 8 minutes to deliver the alert due to the latency of receiving alert information from neighboring countries. Thus, the efficiency of the system must be considered so that people could take appropriate action and respond to that situation effectively.

In conclusion, this research will focus on developing a real time earthquake monitoring and alert system by estimating the epicenter and time arrival of earthquake with the aid of low cost Internet of Things (IoT) sensing model.

1.4 Objective

The objectives of this research are stated as below:

1. To design a disaster alert system that estimates the epicenter and arrival time of earthquake by using accelerometer.
2. To develop a disaster messaging system for sending alert information and video recorded on earthquake scene by using Pi camera module through smartphone‟s application and web based network.
3. To calculate the efficiency for delivering the alert message with photo and video attachment.

1.5 Scope

1. The system is specialized for earthquake detection by taking into account only the P-waves.
2. The system sends warning message only when accelerometer reading exceeds threshold value of 0.05 m/s^2.

3. The project applicable for medium to high level of earthquake disaster at a range of Richter scale between 5.3 to 7.0 magnitudes.
4. The system consists of 3 sensor nodes with each built from an accelerometer as the sensor and a Raspberry Pi as the controller.
5. The distance between the sensor nodes is fixed to 5 kilometers apart each other.
6. The epicenter location is fixed to around 25 kilometers apart from 3 sensor nodes in 6 different directions.
7. The duration between each simulated earthquake scenario is equal to or greater than 15 seconds.
8. The designed shaking table is capable to simulate 3 different ranges of acceleration as shown in below. The ranges may change for each calibration due to hardware inconsistent with the absence of noise filtration.
   a) 0.800 m/s^2 to 1.231 m/s^2 (5.3-5.8 Magnitudes)
   b) 1.370 m/s^2 to 1.626 m/s^2 (5.9-6.4 Magnitudes)
   c) 1.973 m/s^2 to 2.448 m/s^2 (6.5-7.0 Magnitudes)
9. The distance between the earthquake epicenter and the destination is equal to or longer than 1700 km.

1.6 Summary

Overall, this chapter explains the importance of a disaster alert system worldwide to reduce earthquake hazards. The aim of this project is to develop a disaster alert system which could estimate the epicenter and arrival time of an earthquake by using accelerometer sensor. This project also aims to develop a disaster messaging system to send alert information with the attachment of a recorded video on earthquake scene and a photo displaying the map with earthquake information. Last but not least, to calculate the efficiency of this messaging system. The next chapter will discuss and summarizes the findings on recent journal related to this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter presents theoretical background which is related to this project, including theories on disaster and earthquake. Many journals and conference papers related to this research are studied and evaluated based on few specific criteria. Then, the most suitable setup is chosen to be used in this project.

2.2 Definition of Disaster

The „disaster" word comes from Old Italian word *disastro*, from the Latin prefix *dis*- and Latin *astro* which has the meaning of "star" during the late 16th century. Disaster is defined as a sudden accident or a natural tragedy that causes significant damage or loss of life in Oxford dictionary [8].

The International Federation of Red Cross and Red Crescent Societies (IFRC) and United Nations Office for Disaster Risk Reduction (UNISDR) define "disaster" as a sudden situation or event that overwhelms the functioning of a community or society and causes serious losses that exceed the community‟s or society‟s ability to get through using its own resources. According to the IFRC, disaster results from a combination of hazards, vulnerability and inability to reduce the potential of negative risk [9]. Thus, the relationship between hazard, vulnerability, and capacity is expressed in mathematical terms as shown in Equation (2.1) below.

$$\frac{\text{Hazard+Vulnerability}}{\text{Capacity}} = \text{Disaster} \tag{2.1}$$

Equation (2.1) above describes disaster as an inverse relationship between natural hazard plus social vulnerability and the responsive capacity of human organization. The hazard expresses by the risk of a sudden or calamitous event due to natural or environmental factor while vulnerability contributes from a social or human factor as shown in Figure 2.1 below [10].

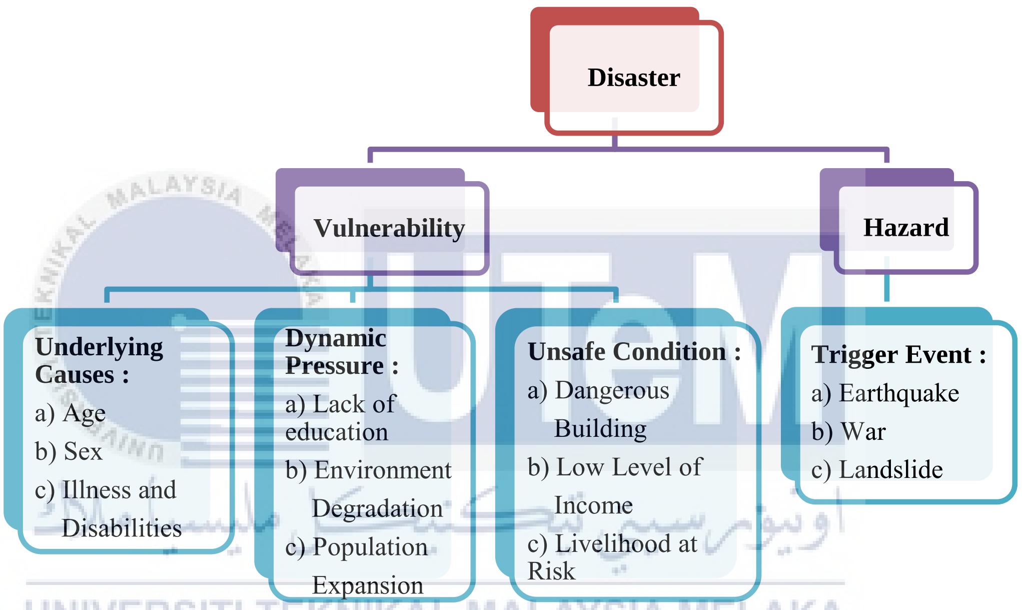


Figure 2.1: The relationship between disaster, vulnerability and hazard [10].

2.3 Classification of Disaster

The International Disaster Database (EM-DAT) of Centre for Research on the Epidemiology of Disasters (CRED) classifies disaster into two generic categories: natural disaster and technological disaster. The natural disaster is defined as naturally occurring physical phenomena caused by rapid or slow onset events which can be divided into 6 sub-groups: Biological, Geophysical, Meteorological, Hydrological, Climatological and Extra-Terrestrial. On the other hand, the technological disaster or man-made disaster is event that is caused by humans and occurs near to human settlements which can be divided into 3 sub-groups: Industrial accident, Transport accident and Miscellaneous accident which in turn cover 15 disaster types [11].

2.4 Natural Disaster

Natural disaster can be divided into 6 groups as mentioned before. In year 2016, Munich Re‟s NatCatSERVICE has recorded 750 events as natural catastrophes that happened globally. Among the 750 events, the hydrological events hit the peak with 50% number of occurrence throughout the year of 2016. 33% of all events recorded are meteorological events and 10% are climatological events. Finally, 7% of the events are geophysical [12]. The Table 2.2 below shows the definition of each type of event in natural disaster.

Table 2.1: Natural disaster sub-group definition [11].

Disaster Sub-group	Definition
Geophysical	An event originating from the ground of earth. This term can also be named as geological event.
Meteorological	An event caused by short-term and micro- to meso-scale extreme weather or atmospheric conditions.
Hydrological	An event caused by occurrence, movement and distribution of earth surface and subsurface of water.
Climatological	An event caused by long-term and meso- to macro-scale extreme atmospheric conditions.
Biological	An event caused by the exposure of living organisms to toxic substances or vector-borne diseases.
Extra-terrestrial	An event caused by strike of asteroids, meteoroids, and comets on the earth and also the changes in interplanetary conditions that effect the Earth‟s magnetosphere, ionosphere, and thermosphere.

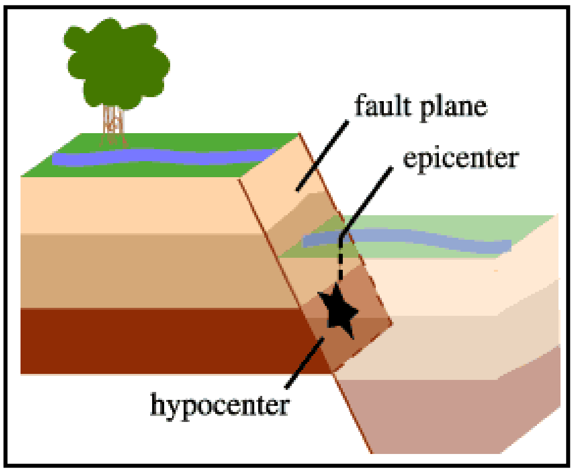
From Table 2.1 above, we can know that each sub-group of natural disaster covers different disaster types which has been tabulated and shown in Table 2.2 below.

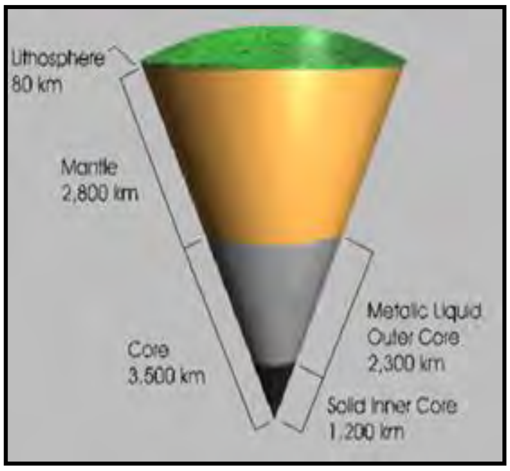
Table 2.2: General classification of natural disaster [11].

Disaster Category	Disaster Sub-group	Disaster Main Type
Natural	Geophysical	Earthquake
		Mass Movement (dry)
		Volcanic activity
	Meteorological	Extreme Temperature
		Fog
		Storm
	Hydrological	Flood
		Landslide
		Wave action
	Climatological	Drought
		Glacial Lake Outburst
		Wildfire
	Biological	Epidemic
		Insect infestation
		Animal Accident
	Extra-terrestrial	Impact
		Space weather

2.5 Earthquake

An earthquake is the result of sudden slippage between two blocks of the earth. The surface of slippage is called the fault or fault plane. The location below the earth"s surface where the earthquake starts is called the hypocenter, and the location directly above it on the surface of the earth is called the epicenter as shown in Figure 2.2 (a) below [13].





(a) (b)

Figure 2.2: a) Occurrence of earthquake [13] b) Layers of earth [13].

The earth has 4 major layers which are inner core, outer core, mantle and crust (lithosphere) as shown in Figure 2.2 (b) above. Lithosphere is the combination of the crust and the top of the mantle. It is made up from many pieces like a puzzle covering the surface of earth which are able to move around slowly and make collision with others. These puzzle pieces are called tectonic plates and the edges of plates are called plate boundaries which are made up of many faults. When the edges of the faults unstick due to the force of the moving blocks finally overcome the friction of the edges of the fault, the earthquake happens. All of the energy stored or a huge amount of energies propagate outward from the fault (epicenter) in all directions in the form of seismic waves. The seismic waves shake the earth as they move through it, and when the waves reach the earth‟s surface, they shake the ground and anything on it including houses and living organism on the land [13 - 17].

The seismic waves are categorized as body waves and surface waves. Body waves consist of two main types which are Primary Wave (P-wave) and Secondary Wave (S-wave) depending upon their physical properties. These two waves are mainly used for detection of the earthquake and its magnitude for earthquake disaster alert system. P waves are compressional or longitudinal wave which shakes the ground in the direction of their propagation using compression-rarefaction and their speed is the highest as compared to all other waves. On the other hand, S waves are shear waves which shakes ground in the direction perpendicular to the propagation direction of P waves with a speed approximately 1.7 times slower than P waves. However, the magnitude of S waves is stronger than P waves. Next, Surface waves are slower as compared to body waves but they bring the most intense shaking to the earth. They can also be categorized as Love waves (side-to-side) and Rayleigh waves (rolling) [14], [18 - 21]. The movements of body waves are shown in Figure 2.3 below.

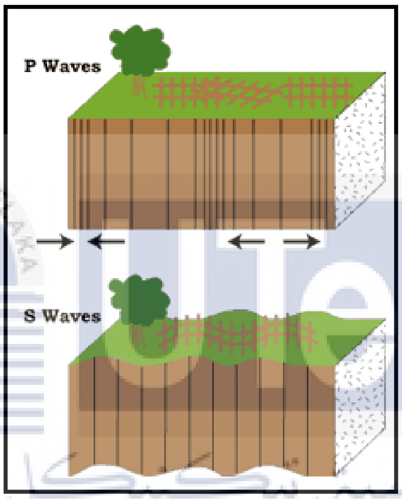


Figure 2.3: Propagation direction of body waves [13].

Earthquakes are recorded by an instrument called seismographs and the recorded graph is called seismogram. Scientists use the seismogram recordings made to determine how large the earthquake was. The length of the wiggle depends on the size of the fault, and the size of the wiggle depends on the amount of slip as shown in Figure 2.4 below [13].

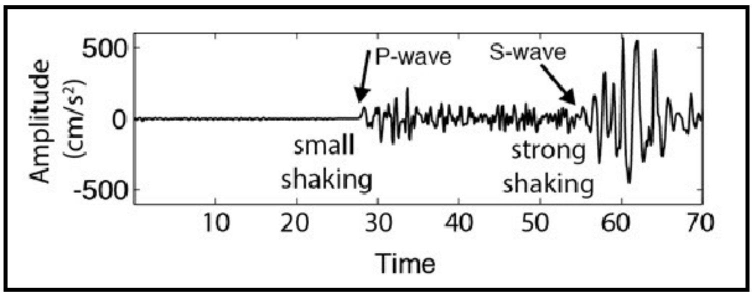


Figure 2.4: Seismogram recordings of body waves in an earthquake [19].

The most common measurement standard for an earthquake based on the amount of released energy at its source is the Richter magnitude scale, developed in 1935 by Charles F. Richter of the California Institute of Technology. The Richter magnitude of an earthquake is determined from the logarithm of the amplitude of waves recorded by seismographs [13]. The Richter scale with its corresponding acceleration is tabulated in Table 2.3 below.

Table 2.3: Richter scale with its corresponding acceleration [22].

Richter Scale	**Approximate Acceleration (cm/s^2)**
<3.5	1
3.5	2.5
4.2	
4.5	10
4.8	25
5.4	50
6.0	150
6.5	250
6.9	
7.3	500
8.1	780
>8.1	980

However, seismograph is able to determine how far the earthquake is, but not knowing the direction of the earthquake and its exact location. Thus, a method called triangulation is used to determine the exact point of the earthquake as shown in Figure 2.5 below [13].

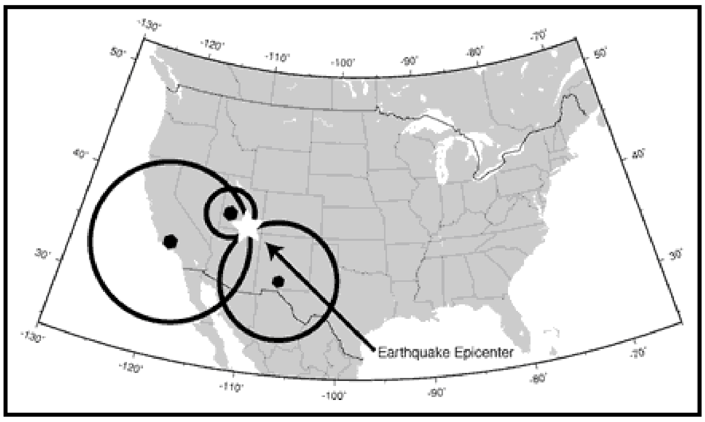
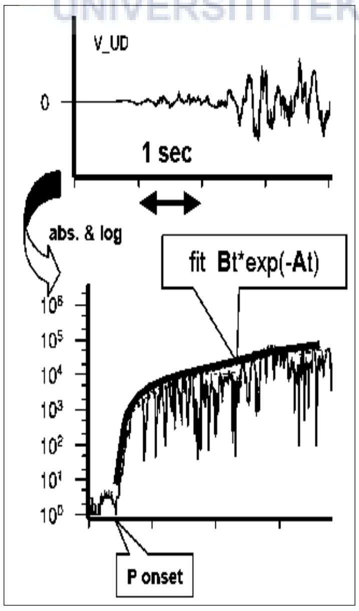
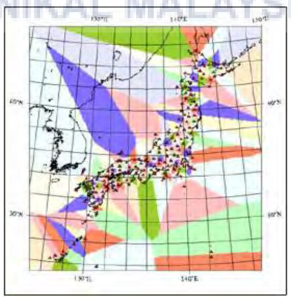


Figure 2.5: Triangulation method [13].

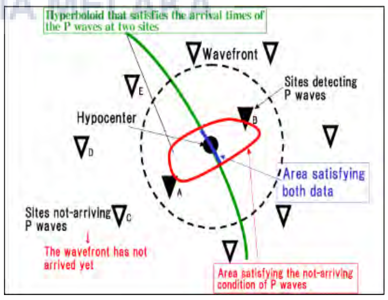
The name of triangulation is due to a triangle has three sides, and it takes three seismographs or 3 sensors as stated in [23] to locate an earthquake where the epicenter lies on the intersection between radius of each from that station to the earthquake. The radius can be known by referring the body waves travel time curve using the time difference between P-wave and S-wave [13], [18].

On the other hand, Japan�much earthquake early warning system can even estimate the hypocenter of an earthquake using both single station method and network method as shown in Figure 2.6 below.

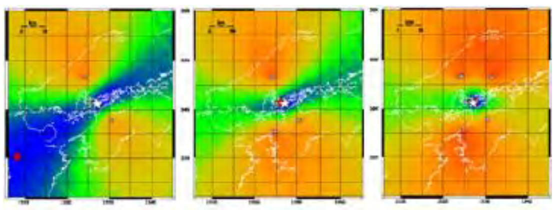




i) Territory method


Hyperboloid that satisfies the arrival times of the P waves at two sites
Wavefront
Sites detecting P waves
Hypocenter
Area satisfying both data
Sites not-arriving P waves
The wavefront has not arrived yet
Area satisfying the not-arriving condition of P waves

ii) Not-yet-arrived method



iii) Grid search method

(a) (b)

Figure 2.6: a) Single method [24] b) Network method [24].

The single station method which can also be known as B-delta method is used to know the estimate the epicenter location from 1 station by fitting Bt*exp(-At) function to the log-transformed acceleration waveform envelope of the first two seconds of P-wave. The network method can be divided into 3 methods which are territory method, grid search method and not-yet-arrived method. The territory method is used when there are 1 or 2 stations have detected arrival of P-wave. A polygonal region is defined as a "territory" to each station surrounded by perpendicular bisectors to the adjacent stations which aim to refine the epicenter estimation. When there is 2 or more stations detected arrival of P-wave, not-yet-arrived method is used by taking into account not only the P-wave arrival times of these stations but also the fact that P-wave have not yet arrived at surrounding stations in the network. This method is used to restrict the hypocenter location to a hyperbolic curve. When the number of stations reaches 3 to 5, grid search method is used to get the optimal grid which covers the P-wave arrival time differences in the least square manner as the hypocenter [24 - 26].

2.6 Disaster Management

The International Federation of Red Cross and Red Crescent Societies (IFRC) defines disaster management as the continuous and integrated process of organizing and managing the resources and responsibilities for dealing with all humanitarian aspects of emergencies, in particular preparedness, response and recovery in order to reduce the impact of disasters [27]. There are 3 stages of activity within disaster management which are known as Disaster Management Cycle. This cycle is then divided into 6 phases: Prevention, Mitigation, Preparedness, Response, Recovery and Reconstruction which is shown in Figure 2.6 below [28].

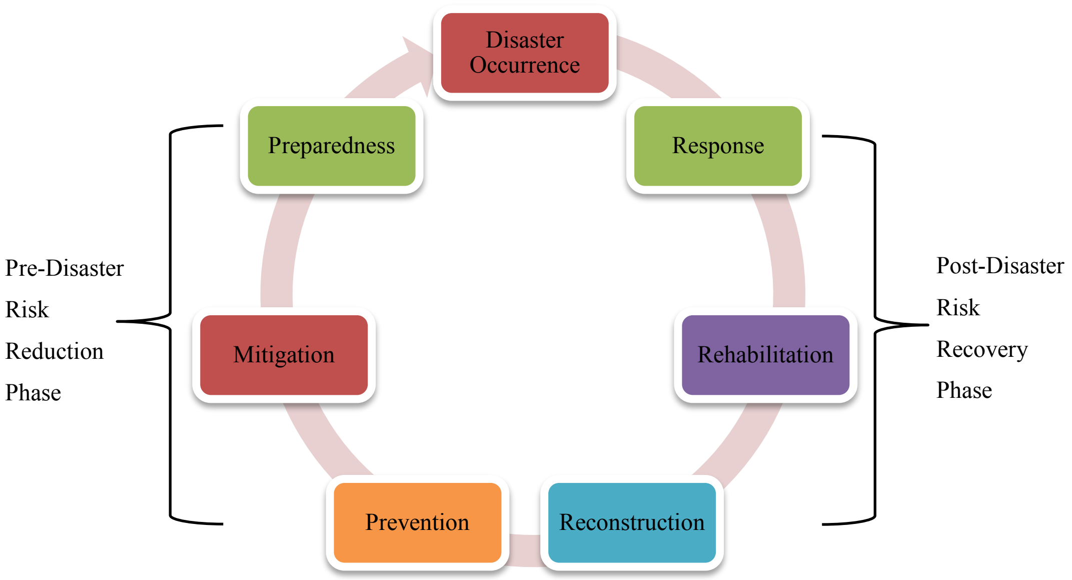


Figure 2.7: The Disaster Management Cycle [28].

The Figure 2.7 above has clearly shows that phases of Prevention, Mitigation and Preparedness are considered as pre-disaster activities which focus on reducing human death and property losses caused by a potential hazard. On the other hand, the remaining phases of Response, Rehabilitation and Reconstruction are considered as post-disaster activities with a purpose to achieve early recovery and rehabilitation of affected victims and communities after the impact of disasters.

Phases of prevention and mitigation refer to the activities which are undertaken to prevent or mitigate the effects of a disaster in short and long-term with the use of hazard, vulnerability and risk assessments or structural and non-structural measures. On the one hand, phase of preparedness can be divided into 2 parts which are contingency planning and also warning and evacuation which are essential to prevent or minimize the losses caused by a disaster. For example, the formulation of emergency plans in case of disaster, development of indicators and early warning systems, public awareness and education and the simulation exercise.

Besides, the phases of response refers to first stage response to any calamity such as putting the contingency plan in action, search and rescue and also fulfilling basic humanitarian needs of victims like shelter, food and clothes. The phase of rehabilitation involves the restoration of basic social function in duration between weeks to months to assist long-term recovery. Finally, reconstruction phase is a long-term recovery activity which takes from months to year which attempts to rebuild the buildings, infrastructure and lifeline facilities with higher capability to reduce or resist the impact of disaster [28].

2.7 Earthquake Early Alert System

An earthquake early alert system is considered as one of the pre-disaster management in preparedness section as mentioned before. Scientist believes that the resulting deaths and injuries of an earthquake are rarely brought by the shaking of ground but the event it triggers such as building collapse, fire and Tsunami [29]. Thus, the main purpose of this system is to provide sufficient time for citizens to evacuate to safe zones or to protect their properties before an earthquake strikes. Since this system detects P-waves of the earthquake to issue the alert, the difference in speed of P-wave and S-wave gives rise to a time delay about 10 to 15 seconds which in turns adding react time to the citizens [16]. In order to achieve this purpose, a real time monitoring and alerting system is needed since the alert message need to be distributed out within seconds to be of value. Thus, few research papers about the design of real time earthquake early alert system are studied and compared based on specific criteria as shown in Table 2.4 below.

Table 2.4: Comparison of early earthquake alert system from previous studied journals or paper.

JOURNAL/IEEE CONFERENCE / CRITERIA	**Towards Formalism of Earthquake Detection and Disaster Reduction using WSANs** [23]	**Disaster Alert System (DISAST)** [22]	**Earthquake Early Warning System by IOT using Wireless Sensor Networks** [20]	**Earthquake Monitoring and Warning System** [16]	**Seismic Early Warning Alert System** [21]
Hardware and/or Software System	Hardware and software using VDM-SL	Hardware and software using Python programming	Hardware and software using C programming and LABVIEW software	Hardware and software using C programming and LABVIEW software	Hardware and Software
Medium Used to Communicate	Internet	Internet	SMS and Internet	SMS and Internet	SMS, Phone Call and Internet
Software or Device Used for Alert	Social Networking Media and Radio Broadcast	Smartphone Application (Telegram)	Mobile Phone and Email	Mobile Phone	Mobile Phone and Email
Sensor Used	1) Animal Sensor (Attach to animal) 2) Water Pressure Sensor (Location with constant water flow) 3) Radon Sensor (Rocks, soil and underground water)	Accelerometer (Soil and sand)	Accelerometer (Surface of the soil)	1) Accelerometer (Surface of the soil) 2) Piezoelectric Sensor (Surface of the soil)	Magnetometer Sensor (Underground)
Controller Used	N/A	Raspberry Pi	Microcontroller	Arduino	N/A
Post or Pre Disaster System	Pre	Pre	Pre and Post	Pre and Post	Pre
Prediction Method	On-sight detection	On-sight detection	On-sight detection	On-sight detection	On-sight detection
Number of sensor node	24	1	1	2	1

Based on the papers shown in Table 2.4 above, there are 2 types of disaster systems which are pre and post-disaster system. Most of the systems are pre-disaster system which functions to alert people before an earthquake strikes. However, [16] and [20] systems shown in Table 2.4 consist of both pre and post-disaster system. Both systems are also considered as a post disaster system since the systems provide a database which stores the information regarding the disaster for future reference.

All of the papers shown in Table 2.4 involve hardware application which uses data from sensors as the source of disaster information in order to establish real time monitoring and alert system. [16], [20] and [22] use accelerometer to sense ground acceleration. [21] uses magnetometer sensors to detect earthquake with electromagnetic signal. Both accelerometer and magnetometer sensors used in [21] and [22] are placed underground while [16] and [20] place the accelerometers above the ground in order to detect the ground shaking caused by an earthquake. All of these sensors detect only the P-wave in different axes to issue warning. Besides, [16] uses piezoelectric sensor to trigger the activation of other components in the circuit only when it senses the vibration of ground for the purpose of long lasting usage of battery. [23] uses 3 different sensors which are animal sensor, water pressure sensor and radon sensor to detect earthquake. Animal sensor is attached to the animal body as mobile biological sensor to detect their body temperature with the aim of sensing abnormal behavior. Water pressure sensor is used to detect variation in water flow. Therefore, it is deployed on different locations where the ground water flow is constant. The radon sensor is used to detect the emission of radon gas which is produced due to the disturbance in water or soil or due to the breaking of rocks. It is placed on rocks, soil and underground water.

Based on Table 2.4 above, [16], [20] and [22] have controller in their system while [21] and [23] have not. The controllers used in the systems are Raspberry Pi, microcontroller and Arduino board respectively to handle sensor data. Most of the system uses different approach to program their system like LABVIEW and Python.

In terms of sensor nodes, [23] achieves the highest number of sensor nodes, which are 24. This system uses wireless sensor and actor network (WSANs) for early earthquake prediction. In this system, each subnet of the system is built up from many sensor nodes, few actor nodes and a gateway node. The predicted earthquake sensed from the sensor in sensor nodes will trigger the nearby actor node which in turns triggers the gateway node. The gateway node will send the earthquake information to the base station which has the function to send alert message. Besides, [16] has 2 sensor nodes with the use of wireless sensor network (WSNs). In each sensor node, there is a XBEE router, a piezoelectric sensor, an accelerometer and an Arduino Uno board. When the piezoelectric sensor senses earthquake, the XBEE router is triggered to send the sensor data to the XBEE coordinator which will push the data to laptop through serial communication for further analysis. The rest of the papers listed in Table 2.4 uses only 1 sensor node to establish simple earthquake early alert system.

Lastly, an earthquake early alert system has different ways to alert people when an earthquake strikes. Based on the research papers listed in Table 2.4 above, one of the methods is sending alert SMS to mobile phone through GSM modem. Another way to send alert message is through Internet in social networking media, smartphone application and email as a feature in Internet of Things (IoT) device. The radio broadcasting and recorded phone call are also one of the options among these systems. The alert information includes the details of the epicenter, magnitude and estimated S-wave arrival time.

2.8 Conclusion

Based on the comparison of early earthquake alert system from previous studied papers listed in Table 2.4, each system has its own pros and cons. In this section, the pros and cons of each system will be discussed in order to pick the desired setup for this project.

Nowadays, wireless sensor network (WSNs) is a common technique to be used to establish communication between sensor node and controller even the sensor node is placed far away from the controller. However, extra components which are router and coordinator are needed and the distance between these components is limited. Therefore, the cost of the system will be increased and the coverage area of the system is limited. The wireless sensor and actor network (WSANs) proposed in one of the previous studied paper has similar function like WSNs but the complexity of the system is much higher since it involves communication between multiple types of nodes in an area. Thus, the system will be more expensive although the coverage area is wider than WSNs. On the other hand, Internet of Things (IoT) has become tremendously popular in designing a system due to its unlimited coverage area and cost effective. Thus, this technique covers the cons of WSNs and WSANs as long as Internet connection is available.

There are few types of sensors are used in available early earthquake alert system based on the previous studied papers. Among these systems, accelerometer is the most popular sensor to detect the presence of earthquake due to the ease of detecting the ground acceleration in different axis which are x, y and z. The price of an accelerometer is cheaper and its size is smaller. Furthermore, an accelerometer can measure a large scale of magnitude. As for the magnetometer sensor, the sensor is extremely sensitive as it is also able to capture other natural or man-made noise. Thus, it is hard to find a suitable location to place the sensor and the system is less efficient to recognize the earthquake pulse since it involves complicated signal processing algorithm. Besides, the system with the combination of 3 sensors which are animal sensor, water pressure sensor and radon sensor to detect an earthquake is more costly and the detection method is more complicated as it involves participation of living things.

There are 2 most common ways to alert people based on the previous studied papers. One of the methods is sending SMS to mobile phone user through GSM modem. Thus, all mobile phone users including non-smartphone and smartphone users can receive the alert when an earthquake strikes. However, additional cost will be charged for each SMS. Another method is use of Internet service to send alert message through social networking media, smartphone application and email. Therefore, this system is limited to smartphone user only in order to receive the alert message at first hand without the aids of computer or laptop and the smartphone must be able to connect to Internet. Radio broadcasting and phone call are the least popular ways to alert people since both requires time to deliver the alert speech.

2.9 Summary

Overall, this chapter presents theoretical background related to this project and also the pros and cons of the studied research paper. Next chapter will cover the setup to be used in this project based on the studied papers and designed experiments to achieve the objectives of the project.

CHAPTER 3

METHODOLOGY

3.1 Overview

In this chapter, methods to achieve the objectives of the project as stated in Chapter 1 will be discussed and presented. Thus, this chapter can be divided into 3 main parts: hardware development, software development and experiment setup. The detailed procedure and list of material and apparatus used for each conducted experiment will be shown.

3.2 Gantt Chart and Milestone

The overall process of the entire project is listed in the Gantt chart as shown in Appendix A1 and A2 for proper schedule purpose. Then, the milestone of each section is listed down in Table 3.1 and Table 3.2 below.

Table 3.1: Milestone for Final Year Project 1.

No.	Activity	Date
1	Research Journal or Paper Review	3 November 2017
2	Selection of Hardware and Software	4 November 2017
3	Early Progress of Experiment 1	18 November 2017
4	Completion of Experiment 4	20 November 2017
5	Completion of Experiment 5	22 November 2017
6	FYP 1 Presentation	5 December 2017
7	Submission of FYP 1 Final Report	22 December 2017

Table 3.2: Milestone for Final Year Project 2.

No.	Activity	Date
1	Completion of Experiment 1	26 February 2018
2	Completion of Experiment 2	2 March 2018
3	Completion of Experiment 3	18 March 2018
4	Completion of Experiment 6	30 March 2018
5	Completion of Experiment 7	4 April 2018
6	Data Analysis and Discussion	12 April 2018
7	Report Writing	18 April 2018
8	Completion of Presentation Slide and Video	21 May 2018
9	FYP 2 Presentation	24 May 2018
10	Submission of FYP 2 Final Report	6 June 2018

3.3 Earthquake Early Alert System Design

The selection of hardware and software are done after comparing the pros and cons of each system discussed in Literature Review chapter. The system setup of the earthquake early alert system is shown in Figure 3.1 below.

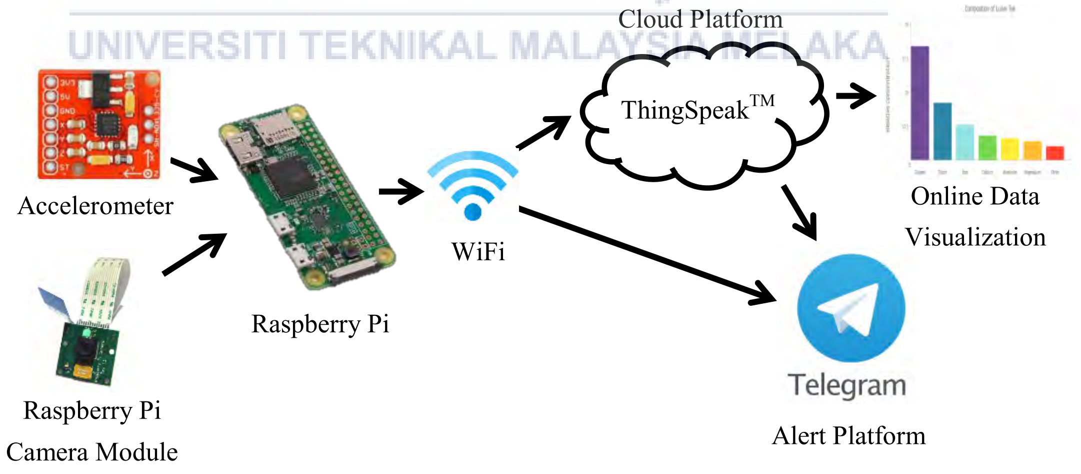


Figure 3.1: System Setup of Earthquake Early Alert System.

Based on Figure 3.1 above, accelerometer is selected as the sensor in this project to detect earthquake due to its low cost while being able to measure a large scale of ground acceleration in 3 different axes. Raspberry Pi Camera Module has the function of recording video for validation. Raspberry Pi is chosen as the controller of the system due to its ease of use to build an IoT model and cost effective. ThingSpeak^{TM} is chosen as the cloud platform for the existence of online MATLAB analytical tool. Next, smartphone application Telegram is used as the alert platform in this project since smartphone are widely used in the world nowadays and Telegram is faster than other smartphone messenger [22].

3.3.1 Hardware Development

Based on previous study, [22] has stated that Raspberry Pi has a higher processor capacity compared to Arduino Uno. Besides, Raspberry Pi is common in IOT model due to its ease of use to connect the Internet without the needs of purchasing extra module. Thus, Raspberry Pi is chosen as the controller of this project. Accelerometer is the most popular sensor among the previous discussed earthquake early alert system due to its low cost while able to measure a large scale of ground acceleration in 3 different axes. [22] also mentioned that accelerometer provides a faster and better responds compared to vibration sensor. Thus, accelerometer is chosen as the sensor to detect P-wave which is 1.7 times faster than S-wave in this project. Raspberry Pi Camera Module is chosen as a tool to record video when earthquake happens due to its very low impact on the CPU (Central Processing Unit) load of Raspberry Pi when recording video compare to USB camera which results to shorter footage processing time.

3.3.1.1 Raspberry Pi

A Raspberry Pi is a credit card-sized and high performance computer first developed by the Raspberry Pi Foundation in United Kingdom designed for programming skills and hardware understanding improvement in education field. It was then quickly adopted by makers and electronics hobbyists for projects that require more than a basic microcontroller, like Arduino board due to its accessible price and small size [30].

Recently, the Raspberry Pi Foundation has launched the Raspberry Pi Zero W, an improved version of the original Raspberry Pi Zero that adds 2 key elements of IoT, built-in Wi-Fi and Bluetooth provided by the same Cypress CYW43438 wireless chip with Pi 3 Model B. The 40- pin unpopulated GPIO header is same with previous model. The features of Raspberry Pi Zero W are shown in Figure 3.2 below [30].

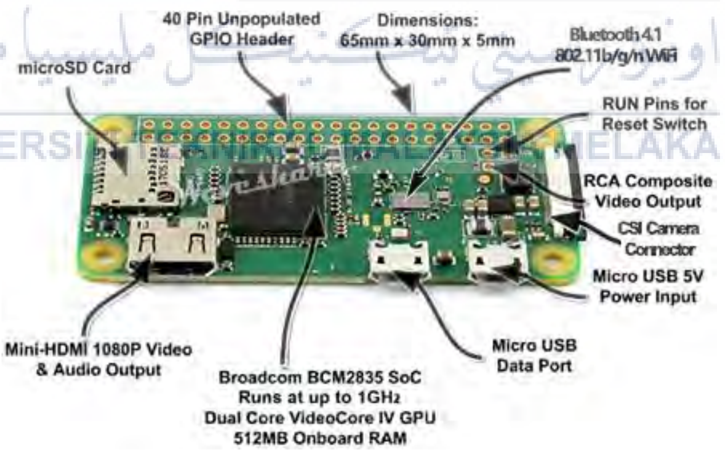

40 Pin Unpopulated GPIO Header
Dimensions: 65mm x 30mm x 5mm
Bluetooth 4.1 802.11b/g/n WiFi
microSD Card
RUN Pins for Reset Switch
RCA Composite Video Output
CSI Camera Connector
Micro USB 5V Power Input
Mini-HDMI 1080P Video & Audio Output
Micro USB Data Port
Broadcom BCM2835 SoC Runs at up to 1GHz Dual Core VideoCore IV GPU 512MB Onboard RAM

Figure 3.2: Raspberry Pi Zero W [30].

Comparison between Raspberry Pi Zero W and Raspberry Pi 2 Model B as used in [22] is shown in Table 3.3 below [31].

Table 3.3: Comparison between Raspberry Pi 2 Model B and Raspberry Pi Zero W.

Features	**Raspberry Pi 2 Model B**	**Raspberry Pi Zero W**
Central Processing Unit (CPU)	900MHz quad-core ARM Cortex-A7 CPU	1 GHz ARM11 CPU
Memory	1GB	512 MB
Storage	MicroSD card slot	MicroSD card slot
Connectivity	4 USB 2.0 ports, HDMI port, Ethernet port and 3.5mm audio jack	MicroUSB port, mini-HDMI port, 802.11n wireless LAN and Bluetooth 4.0
Operating System	Linux and Windows 10 IoT core	Linux
Connectors	Camera interface (CSI), 40- pin GPIO, SPI, I2C and JTAG	Camera interface (CSI), unpopulated 40-pin GPIO, SPI and I2C
Power Supply	5V	5V
Dimension	85 mm x 56 mm x 17 mm	65mm × 30mm × 5mm
Price	RM364.53	RM42.40

From Table 3.3, Raspberry Pi 2 Model B has better hardware specification than Raspberry Pi Zero W with higher processing capacity, larger memory and more operating system choice. However, Raspberry Pi Zero W has smaller size, ultra- low cost and has the most important feature, built in Wi-Fi and Bluetooth adapters. Thus, Raspberry Pi Zero W is chosen as the controller of this project since it is an excellent platform for IoT application.

3.3.1.2 Accelerometer

Accelerometer is a sensor used to measure the change in velocity, which can also be known as acceleration of an object. The unit of measurement is meters per second squared (m/s^2) or in G-forces (g) which is equivalent to 9.8 m/s^2 on Earth. It is useful in sensing the vibrations or orientation of the system in one or multiple axes depends on the sensor‟s structure [32].

Accelerometer ADXL335 from Analog Devices as shown in Figure 3.3 below is chosen as the sensor of the project to measure the earthquake magnitude based on the acceleration of the prototype model refers to [22].

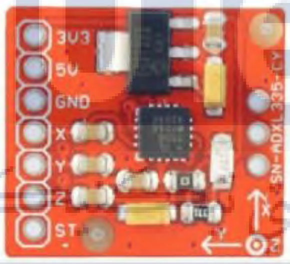


Figure 3.3: Accelerometer ADXL335 [22].

The Accelerometer ADXL355 is a small and thin analog sensor which is able to sense the vibration in 3 different axes, namely x, y and z. With its extremely low noise and power consumption around 300 uA, it is able to measure acceleration in full-scale range of ± 3 g [32]. The specification of the ADXL355 is listed in Table 3.4 below based on its datasheet.

Table 3.4: Specification of ADXL355 [32].

Parameter	Value
Operating voltage	2.5 V – 6 V
Typical current	300 uA
Range	± 3 g
Sensitivity	0.33 V/g
Voltage at 0g of X-axis and Y-axis	1.65 V
Voltage at 0g of Z-axis	1.80 V
Reference voltage, V_{ref}	3.3 V

To calculate the peak ground acceleration refer to the ADC value for different earthquake magnitude Richter scale as stated in Table 2.3 above in chapter 2, formula to be used are listed down below refer to [22]. Since P-wave propagates parallel to the ground, the calculation of this project involves x-axis only by assuming that the y-axis acceleration is same as x-axis.

The ground peak acceleration measured in m/s^2,

$$a = \left(\frac{adc * Vref}{r} - V_{0g}\right) \div \left(\frac{S}{g}\right) \quad (3.1)$$

where adc is the analog-to-digital converter (ADC) value

V_{ref} is the reference voltage (3.3 V)

r is the range of 10-bit resolution analog-to-digital converter (1023)

V_{0g} is the voltage at 0 g (1.65 V)

S is the sensitivity of ADXL355 (0.33 V/g)

g is the gravitational acceleration (9.81 m/s^2) [22]

3.3.1.3 Raspberry Pi Camera Module

In this project, Raspberry Pi Camera Module Version 1.3 as shown in Figure 3.4 below is used to validate the earthquake hit by recording video at the site. The camera attaches directly to the 15-pin MIPI Camera Serial Interface (CSI) of Raspberry Pi by a 15 Pin Ribbon Cable. This CSI bus is directly connected to the Raspberry Pi GPU (Graphics Processing Unit) which can process images without ARM intervention. It can record video at maximum 30 frames per seconds for 1920 x 1080 quality video, 60 frames per seconds for 1280 x 720 quality video and 60 or 90 frames per seconds for 640 x 480 quality video using a 5 Megapixels OmniVision OV5647 sensor [33].

Figure 3.4: Raspberry Pi Camera Module Version 1.3 [33].

3.3.1.4 Shaking Table

A unidirectional shaking table is designed to simulate the P-wave propagation of an earthquake consistently as shown in Figure 3.5 below.

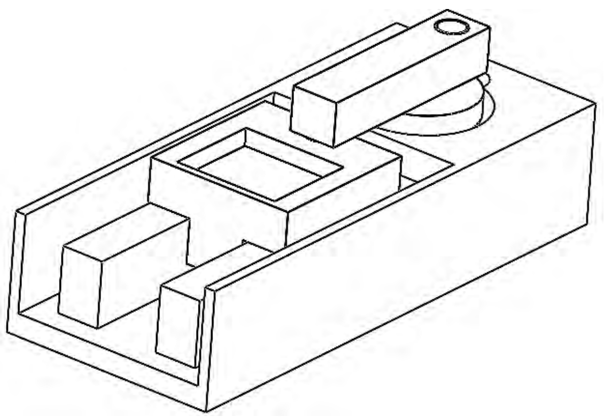


Figure 3.5: Shaking table design.

The design is inspired by [34] where the table is driven by a drill in a constant acceleration as shown in Figure 3.6 below.

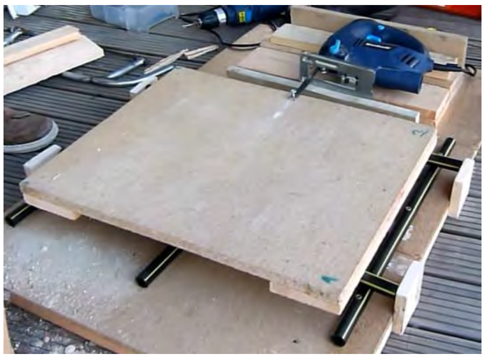
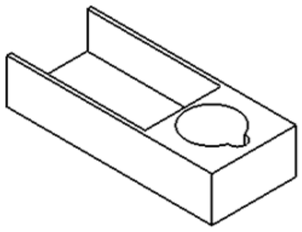
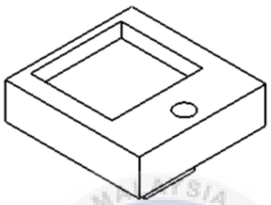
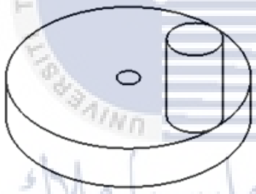
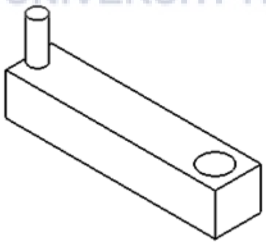
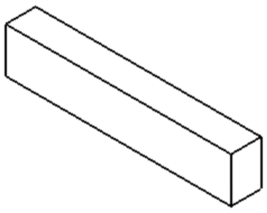


Figure 3.6: Shaking table using drill as actuator [34].

The movement of the table is fixed by wood blocks at 2 sides. Pipes are placed under the table to reduce friction when it is driven by the drill. However, the drill is expensive and this project requires total of 3 set of shaking tables for each station. Therefore, a 6V DC motor is used as the actuator and the size of the table is highly reduced to cut down the cost. All 3 tables are 3D printed using polylactic acid (PLA) plastic as its material. The size of the table is 5.1cm x 12.4cm x 7.577cm and it can be divided into 5 parts listed in Table 3.5 below. The detail measurement of each part of the designed shaking table is shown in Appendix B.

Table 3.5: List of parts of the designed shaking table.

Part	**Quantity**	**Function**
i) Base 	1	To provide space for the DC motor and unidirectional movement of the body.
ii) Body 	1	To hold the accelerometer sensor and drive by the DC motor.
iii) Roller 	1	To transform the rotational movement of the DC motor to linear movement.
iv) Link 	1	
v) Beam 	2	To limit the movement of the body and reduce the friction between base and body during shaking.

3.3.2 Software Development

The term “Internet of Things” (IoT) by Kevin Ashton in 1999 has been popularized by MIT in business field for market analysis at MIT“s Auto-ID center. IoT is used to achieve “Machine to Machine” (M2M) interaction which makes networked devices “smart” to exchange information and perform actions without assistance of humans. It is predicted that there will be more than 50 billion devices connected to IoT by 2020 [35].

3.3.2.1 ThingSpeak^{TM}

ThingSpeak™ is an open IoT analytics platform service that allows one to collect and store sensor data in the cloud to develop IoT application. With the ability to execute MATLAB, one of the application provided by ThingSpeak^{TM}, one can perform online analysis, visualization and react automatically upon live data streams to achieve system standalone mode without the necessity of a computer as shown in Figure 3.7 below [36], [37].

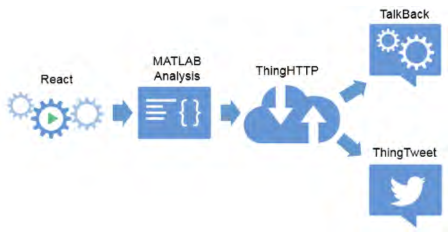


Figure 3.7: ThingSpeak applications [35].

ThingSpeak^{TM} is the only open data platform specifically designed for the IoT in the cloud. It has the possibility of creating public channels to present own IoT application without the needs of setting up servers or developing web software. Although it is simpler to be used as compared to other competitors like Carriots, SmartObject, Skynet and Sensorthings, there is a limit for one update per channel every fifteen seconds due to the excess bandwidth for a free account. However, the data upload rate can be reduced to 1 second if purchased [35], [38]. Based on the benefits mentioned above, ThingSpeak^{TM} is chosen as the cloud service to build a real time earthquake monitoring and alert system.

3.3.2.2 Telegram

Telegram is a cloud-based mobile and desktop messaging application which enables user to access their messages from multiple devices. It supports all operating system known today like Android, iOS, Windows, MacOS and even Linux. One of the most important features of Telegram is enabling third-party developers to create bots. Bot is a special account operated by software which does not require an additional phone number for set up. It is considered as text-based service built for specific purposes like remind user, integrate with other services or even pass instruction to IoT platform [39]. Therefore, Telegram is chosen as the alert platform to receive the alert information sent by the created bot from Raspberry Pi as done in [22].

3.3.3 System Architecture

The overall chart of earthquake early alert system is shown in Figure 3.8 below.

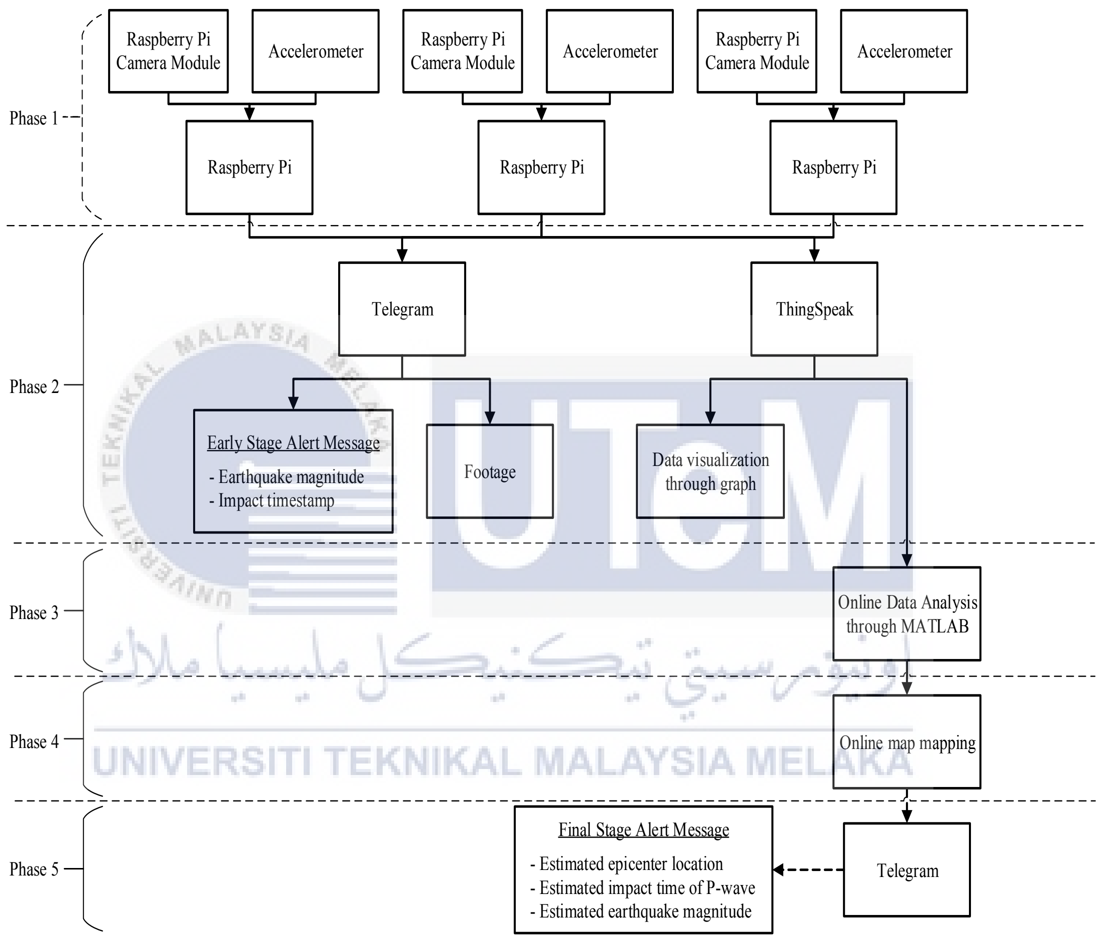

Phase 1
Phase 2
Phase 3
Phase 4
Phase 5
Raspberry Pi Camera Module
Accelerometer
Raspberry Pi Camera Module
Accelerometer
Raspberry Pi Camera Module
Accelerometer
Raspberry Pi
Raspberry Pi
Raspberry Pi
Telegram
ThingSpeak
Early Stage Alert Message
- Earthquake magnitude
- Impact timestamp
Footage
Data visualization through graph
Online Data Analysis through MATLAB
Online map mapping
Final Stage Alert Message
- Estimated epicenter location
- Estimated impact time of P-wave
- Estimated earthquake magnitude
Telegram

Figure 3.8: Overall chart of earthquake early alert system.

Based on Figure 3.8, earthquake early alert system can be divided into 5 phases. Each phase has its own functionality which is explained in Figure 3.9 below. The detailed process of the system is shown in a flow chart in Figure 3.10 below.

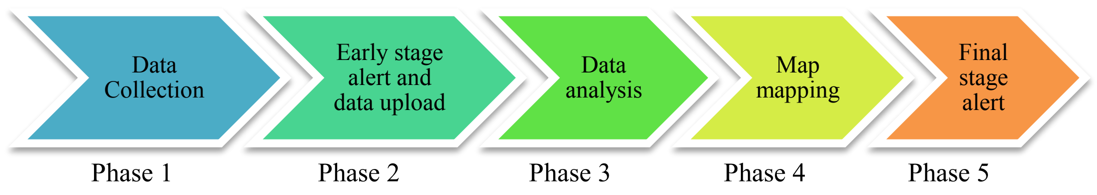
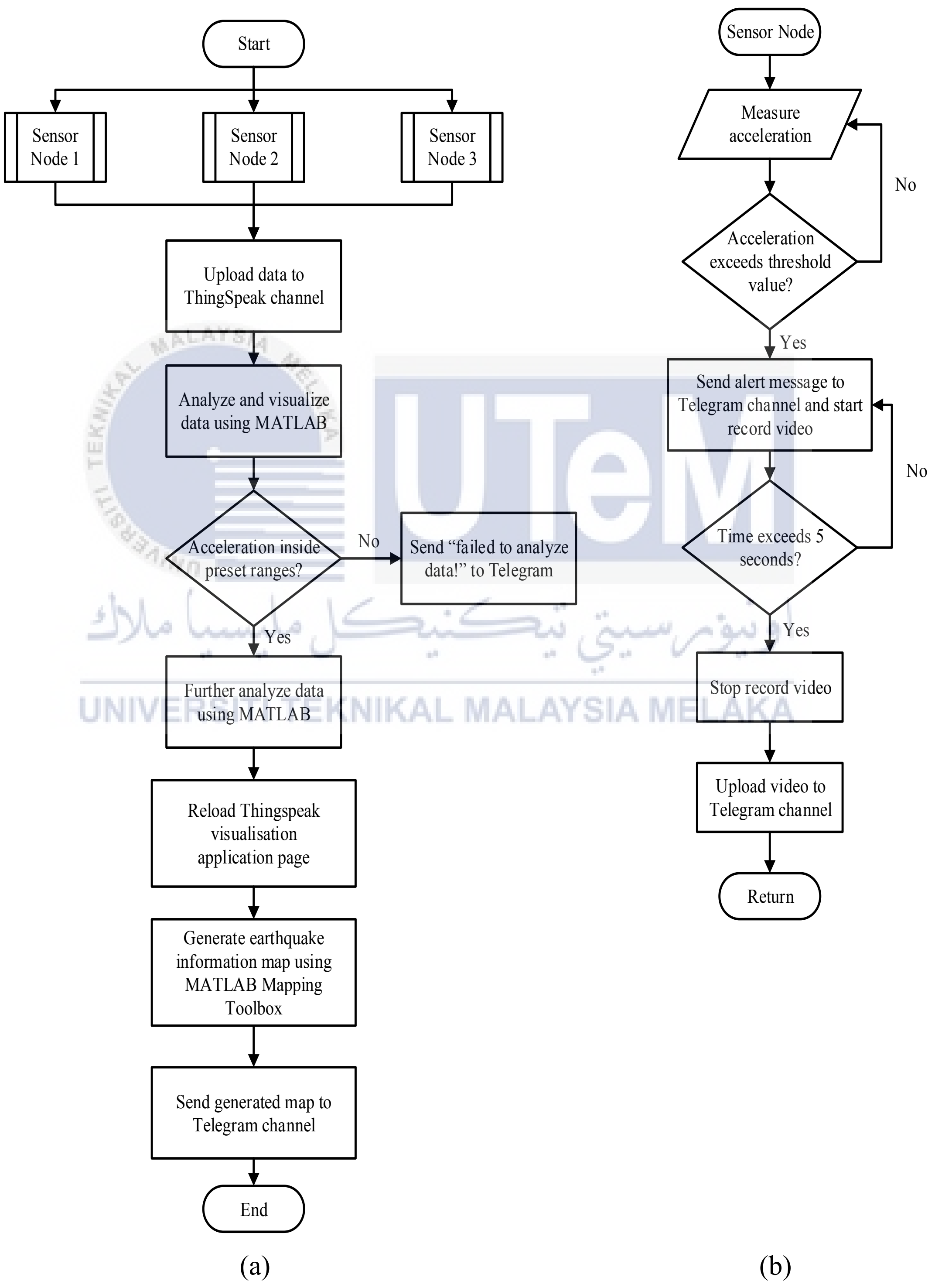


Figure 3.9: Functionality of each phases of earthquake early alert system.



Start
Sensor Node 1
Sensor Node 2
Sensor Node 3
Upload data to ThingSpeak channel
Analyze and visualize data using MATLAB
Acceleration inside preset ranges?
No
Send "failed to analyze data!" to Telegram
Yes
Further analyze data using MATLAB
Reload Thingspeak visualisation application page
Generate earthquake information map using MATLAB Mapping Toolbox
Send generated map to Telegram channel
End
(a)
Sensor Node
Measure acceleration
Acceleration exceeds threshold value?
No
Yes
Send alert message to Telegram channel and start record video
Time exceeds 5 seconds?
No
Yes
Stop record video
Upload video to Telegram channel
Return
(b)

Figure 3.10: a) Flow chart of alert system b) Flow chart of sensor node.

Figure 3.10 (a) shows the entire flow of the earthquake alert system while Figure 3.10 (b) shows the flow of each sensor node implemented in the system. All the processes will be explained by phases in next section.

3.3.3.1 Initial Setup

This section involves the installation of the circuit for each sensor node and its location before entering the phase explanation. The schematic diagram for each sensor node is shown in Figure 3.11 below. The pin number for Pi Zero refers to the Pi4J documentation.

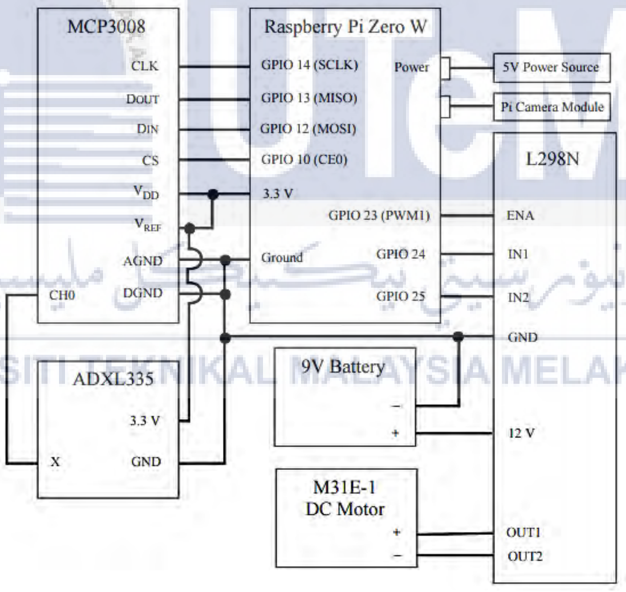


Figure 3.11: Schematic diagram of sensor node.

Each sensor node will have components:

a) Raspberry Pi Zero W (Controller)

b) MCP 3008 (8-channel 10-bit analog to digital converter)

c) L298N (Motor driver)

d) ADXL 335 Accelerometer (Sensor)

e) M31E-1 (6V DC motor)

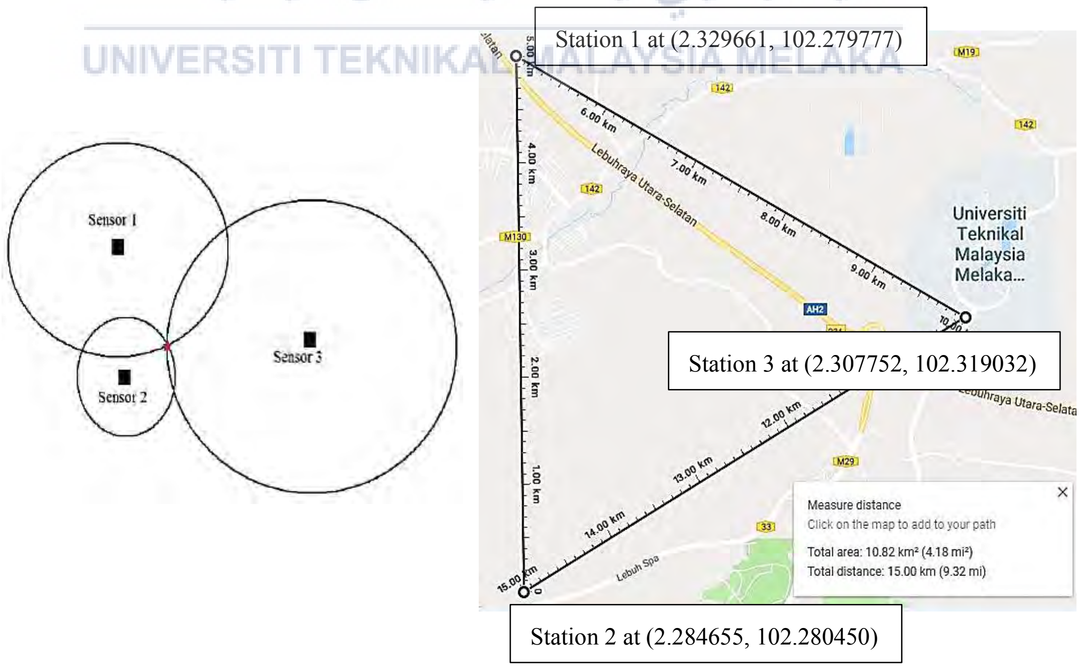
f) 9V Battery (Power supply for motor and motor driver)

g) 5V Power Supply (Power supply for Raspberry Pi Zero W)

h) Raspberry Pi Camera Module Version 1.3 (Video recorder)

For the 5V power supply as mentioned in (g), Raspberry Pi will be connected to a DC power source converted from an AC power plug which in turns provides electrical energy for the sensor and camera. Raspberry Pi Camera Module Version 1.3 will be installed only at the Sensor Node 2 due to the camera is used for experimental purpose on video recording to validate the earthquake hit and save project cost. Another 2 sensor nodes will only upload the saved video when the accelerometer data exceeds threshold value.

All 3 sensor nodes will be arranged at a location referring the triangulation method as mentioned in [18] to predict the epicenter of the earthquake as shown in Figure 3.12 (a) below. Each sensor node is 5km apart from each other as shown in Figure 3.12 (b) with the location coordinates in terms of latitude and longitude marked beside each sensor node.


Sensor 1
Sensor 2
Sensor 3
Station 1 at (2.329661, 102.279777)
Station 3 at (2.307752, 102.319032)
Station 2 at (2.284655, 102.280450)
Lebuhraya Utara-Selatan
Universiti Teknikal Malaysia Melaka...
Measure distance
Click on the map to add to your path
Total area: 10.82 km² (4.18 mi²)
Total distance: 15.00 km (9.32 mi)

(a) (b)

Figure 3.12: a) Triangulation method [18] b) Location of each sensor node.

A simple algorithm is used to calculate the theoretical acceleration using the distance between the epicenter and sensor node and the earthquake magnitude. The distance is the radius for each sensor node referring to the triangulation method as mentioned in Chapter 2.

Using the Donovan (1973) ground motion model,

$$a = 13 * (e^{0.67M}) * (D + 25)^{-1.6} \quad (3.2)$$

where a is the ground peak acceleration in m/s^2

M is the earthquake magnitude

D is the distance from the earthquake epicenter in km [40], [41]

By using Equation (3.2), it was found that the acceleration value at a distance of 25km away from the earthquake epicenter is the closest to the corresponding acceleration of Richter magnitude scale in Table 2.3 above which the magnitude ranges from 5.3 to 7.0 magnitudes. Using this distance, the earthquake epicenter is preset to 6 different locations marked from alphabet A to F surrounding the stations as shown in Figure 3.13 below which is plotted using MATLAB Mapping Toolbox. The red dots are the epicenters while the plus symbols are the stations. The location of the station is plotted using the coordinates in Figure 3.12 (b).

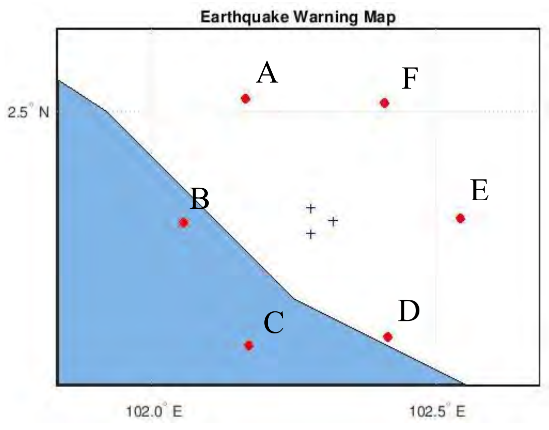


Figure 3.13: Map showing the location of epicenters and sensor nodes.

The distance between each epicenter point and the sensor nodes is found by illustrating the coordinates of all 3 stations and measure the distance of around 25 km from each station in AutoCAD software. The geometrical coordinate of each epicenter point is then found using MATLAB function *scxsc*. This function finds the intersection geometrical coordinate between 2 circles drawn from either 2 stations using the distance between epicenter point and the sensor nodes as shown in Table 3.6.

Table 3.6: Epicenter distance to each sensor node and its coordinate.

Epicenter	**Distance to station 1/km**	**Distance to station 2/km**	**Distance to station 3/km**	**Coordinate**
A	25.00	29.44	29.44	2.522500 , 102.164000
B	25.12	25.12	29.33	2.303800,102.055200
C	29.44	25.00	29.44	2.088406, 102.170699
D	29.33	25.12	25.12	2.103400,102.415300
E	29.44	29.44	25.00	2.311100,102.544100
F	25.12	29.33	25.12	2.515000,102.409100

To simulate the theoretical acceleration found using the Equation (3.2), the following formulas are used to transform the angular acceleration of DC motor to linear acceleration through designed shaking table.

The angular velocity of the DC motor,

$$\omega = A * D * \frac{2\pi}{60} \tag{3.3}$$

where w is the angular velocity corresponding to motor‟s duty cycle in rad/s

A is the no load speed of M31E-1 DC motor refer to its datasheet (3700 rpm)

D is the duty cycle of the motor in percentage

The angular acceleration of DC motor,

$$\alpha = \frac{\Delta\omega}{\Delta t} \tag{3.4}$$

where α is the angular acceleration of the motor in rad/s^2

$\Delta\omega$ is the change in angular velocity in rad/s

Δt is the change in time in s

Finally, the linear acceleration or tangential acceleration of the motor,

$$a_t = \alpha r \tag{3.5}$$

where a_t is the linear acceleration of the motor in m/s^2

r is the radius of the roller in Table 3.5 (0.01 m/s^2) [42]

By using the Equations (3.3) to (3.5), the duty cycle required for the motor to generate the theoretical acceleration is found. Since the motor is inconsistent due to the absence of close-loop control system, the Richter scale of 5.3 to 7.0 magnitudes are divided into 3 ranges, each range contains 6 steps of magnitude. The theoretical acceleration refers to around the middle of each range. Assuming the P-wave is in constant acceleration during its arrival to the sensor nodes, the speed of the shaking table is running following the pattern of a sine waveform for 4 seconds to ensure constant acceleration is produced for 3 seconds sensor data calculation as shown in Figure 3.14 below.

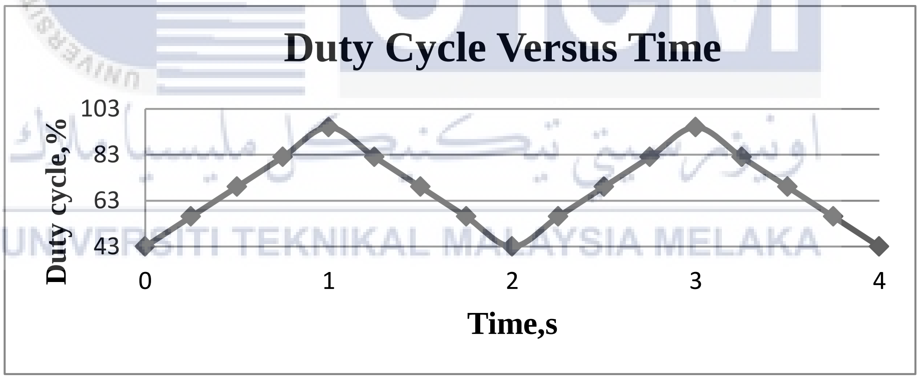


Figure 3.14: Changes of motor speed in time.

A gain is multiplied with the actual sensor acceleration data to reduce the percentage error of the data to within 10% as compared to the theoretical acceleration. This method can improve the accuracy of the sensing platform to be more approach to real situation.

Using the percent error formula,

$$e = \left|\frac{a_t - a_d}{a_t}\right| * 100\% \tag{3.6}$$

where e is the percentage error (10%)

a_t is the theoretical acceleration in m/s^2

a_d is the desired acceleration in m/s^2

Using Equation (3.6),

$$a_d = a_t \pm (\frac{10}{100} * a_t) \quad (3.7)$$

To find the gain,

$$x = \frac{a_d}{a_a} \quad (3.8)$$

where x is the gain

a_a is the actual acceleration from accelerometer in m/s^2

By using the Equations (3.6) to (3.8), the process of shaking table calibration is achieved through multiplication of gain with the sensor data. Table 3.7 below shows the range of Richter magnitude scale with its corresponding theoretical acceleration, desired acceleration and duty cycle of motor which calculated using the first second of shaking table movement. The time step and duty cycle step are the step size per increment of x and y-axis. Loop refers to the total loops required for the motor to produce the sine waveform pattern shown in Figure 3.14 for each second.

Table 3.7: Shaking table setup.

Richter magnitude scale	**Theoretical acceleration, m/s^2**	**Desired acceleration, m/s^2**	**Change in angular velocity, m/s**	**Change in time, t**	**Duty cycle range, %**	**Duty cycle step**	**Time step, s**	**Loop**
5.3-5.8	0.930	1.023(+) or 0.837(-)	93.0	1	35-59	3	0.125	8
5.9-6.4	1.356	1.492(+) or 1.220(-)	135.6	1	35-70	7	0.200	5
6.5-7.0	2.015	2.217(+) or 1.814(-)	201.5	1	43-95	13	0.25	4

All the data shown in Table 3.7 refers to the positive increment in Figure 3.14 and vice versa to the negative increment in duty cycle step. The negative sign given by the deceleration during negative increment is removed. Only the magnitude of the sensor data is considered.

To simulate the propagation of P-wave from an epicenter point to the sensor nodes using shaking tables, a time delay is essential since all 3 sensor nodes located at different coordinate during the earthquake simulation. Thus, P-wave travel time curve is used to know the theoretical arrival time of P-wave based on the distance of the sensor node from the epicenter as shown in Figure 3.15 below [43].

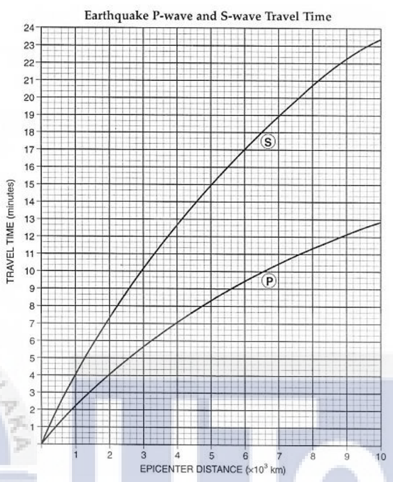


Figure 3.15: P-wave and S-wave travel time curve [43].

Using the MATLAB Curve Fitting Toolbox, the polynomial equation of P-wave travel time curve is obtained as shown below.

$$y = 0.0002x^5 - 0.0042x^4 + 0.0425x^3 - 0.2817x^2 + 2.4332x + 0.0174 \qquad (3.9)$$

where y is the travel time of P-wave in minutes

x is the distance from the epicenter in x10^3 km

Using the Equation (3.9), the time delay to simulate all the 6 epicenter location is calculated as shown in Table 3.8 below and is rounded off to 1 decimal place.

Table 3.8: Sensor nodes time delay for each epicenter point.

Epicenter	**Station 1**	**Station 2**	**Station 3**
A	0	0.6	0.6
B	0	0	0.6
C	0.6	0	0.6
D	0.6	0	0
E	0.6	0.6	0
F	0	0.6	0

3.3.3.2 Phase 1

The accelerometer acts as the sensor to measure ground peak acceleration every 0.1 seconds and send the measured data back to Raspberry Pi. For this project, a python script is running continuously in all the Raspberry Pi to receive command from a Telegram private channel, named info@pi using a Telegram bot. This channel is built to ease the communication between all the Raspberry Pi. Once the bot received the desired message, all 3 shaking tables will run which in turns trigger the accelerometer to measure the ground peak acceleration. Thus, a total of 4 telegram bots are needed for different Raspberry Pi where 3 bots is used to send message and video while the last one is to collect data sent by other bots since the bot cannot detect message send by itself. The last bot is used by Sensor Node 1 as it is responsible to upload the collected data to ThingSpeak channel in Phase 2. Thus, 2 bots are being used by Sensor Node 1 where 1 is used to detect message sent to the Telegram channel while another bot is used for sending message and video. The threshold value of acceleration to trigger the alert message is preset to a value larger than 1 cm/s^2 corresponding to earthquake shaking with Richter scale exceeding 3.0 magnitude which will be felt by many people. However, due to the absence of noise filter for the sensor, the threshold value is preset to 0.05 m/s^2. When the acceleration exceeds the preset threshold value, the system enters Phase 2.

3.3.3.3 Phase 2

First, 3 processes are undergone at the same time. Pi camera will be triggered to record a 5 seconds site video and an alert message containing the earthquake magnitude and earthquake impact timestamp will be sent to Telegram private channel named Test@pi. This channel is built to inform the authorities regarding the earthquake hit. The timestamp will also be sent to info@pi when the sensor data exceed the preset threshold value. At the same time, sensor will collect 100 times data within 3 seconds and the average value is multiplied with the preset gain before sending to info@pi. The 3 seconds duration is used to measure the P-wave acceleration as refers to [44] and [45] which stated that the initial several seconds of P-wave data can predict the earthquake size and magnitude. The recorded video is uploaded to Test@pi directly after 5 seconds recording time. When Sensor Node 1 finished collecting timestamps and average sensor data from all 3 Raspberry Pi, the information will be uploaded to ThingSpeak channel for data visualization and analysis. Since Node 1 timestamp delay due to the processing speed of its class 4 Micro SD card, the timestamp is minus by 1 second to increase the chance of successfully analyze the epicenter point. The average acceleration of each sensor node is directly plotted on the channel for visualization purpose which in turns enabling online monitoring feature. Sensor Node 1 will send “Warning cancel” as an assumption of faulty detection once one of the sensor nodes is not triggered and causes the system to reset back to Phase 1. When the data is successfully uploaded to ThingSpeak channel, the React app of ThingSpeak will call the MATLAB Analysis program saved in ThingSpeak and causing the system to enter Phase 3.

3.3.3.4 Phase 3

MATLAB analytical tool is used in this phase to compare the sensor data from each node with the preset acceleration ranges and the timestamps with the preset time refer to Table 3.8. If one of the data or timestamp falls outside the ranges, ThingSpeak will send “failed to analyze data!” to info@pi and causes the system to reset back to Phase 1. When all 3 sensor data fall within the same range, earthquake magnitude can be determined. Then, the timestamp regarding to the arrival time of P-wave to the sensor nodes are analyzed to estimate the earthquake epicenter location and the earliest P-wave arrival time refer to Table 3.8. Once the MATLAB Analysis program has finished analyzing the data, the estimated earthquake magnitude, estimated epicenter location and the earliest P-wave arrival time will be written to another ThingSpeak channel for map mapping purpose. Then, a “reload” message is sent to info@pi and the system enters Phase 4.

3.3.3.5 Phase 4

A MATLAB Visualization program is created to generate the earthquake information map showing the estimated earthquake location, P-wave arrival time to point X and the estimated earthquake magnitude. To do this, a MathWorks account with the license of using MATLAB Mapping Toolbox is required to log in to ThingSpeak. This toolbox is essential for processing geometrical coordinate data and generating world map. When the “reload” message is received by Telegram bot of Sensor Node 3, the MATLAB Visualization program page will be reloaded to run the program since ThingSpeak has not yet implement auto update on MATLAB Visualization application. The sensor data and the estimated magnitude is read from ThingSpeak channel to estimate the epicenter location refer to the triangulation method. Using the Equation (3.2), the circle radius in kilometer can be found. Due to the inconsistent problem of both the sensor and shaking table, the epicenter of the earthquake is expanded to cover the preset acceleration ranges for each set of magnitude ranges. The median of each acceleration range is the diameter of the epicenter circle with its coordinate as the center point. As for the P-wave arrival time to a destination point, point X with geographical coordinate (10.0156,115.8368) is used which is 1700 km far from epicenter point E. To estimate the shortest time for the arrival of P-wave to the point X, the following algorithm is used.

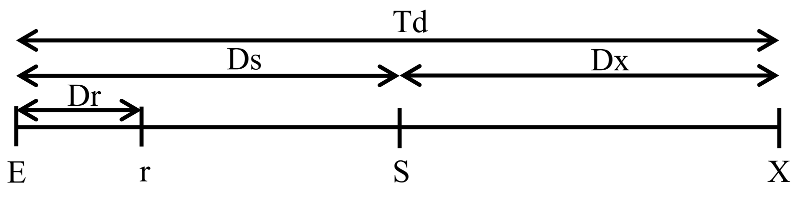


Figure 3.16: Details of the distance between point X and the epicenter.

where E is the epicenter coordinate

r is the epicenter radius coordinate

S is the sensor node coordinate

X is the point X coordinate

Dr is the radius of epicenter in km

Ds is the distance between epicenter and sensor node in km

Dx is the difference between Td and Ds in km

Td is the distance between point X and the epicenter in km

The estimated epicenter coordinate and the sensor node with the earliest timestamp of the arrival of P-wave are used. Using the Equation (3.9), the P-wave travel time in Dr and Dx distances are calculated.

$$t = t_S + t_X - t_r \tag{3.10}$$

where t is the estimated P-wave arrival time to point X in s

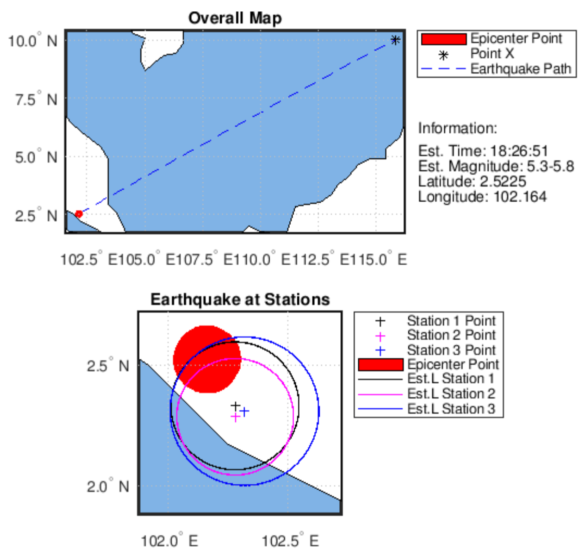
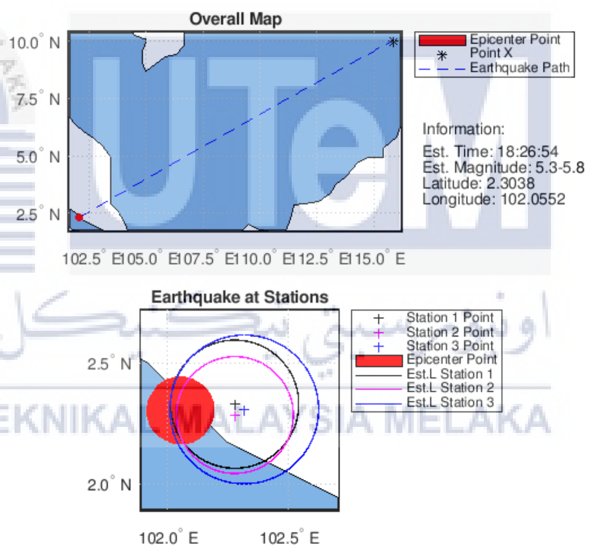
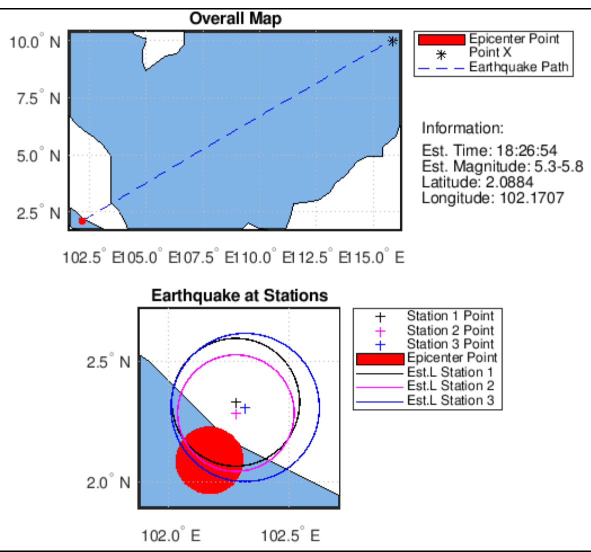
t_S is the earliest timestamp of the arrival of P-wave to the sensor node

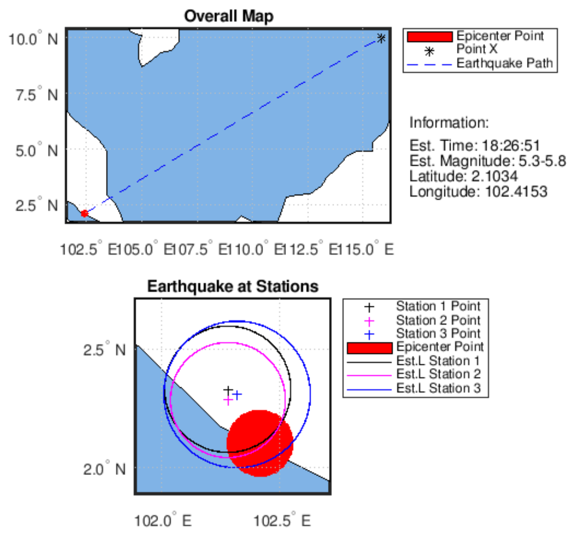
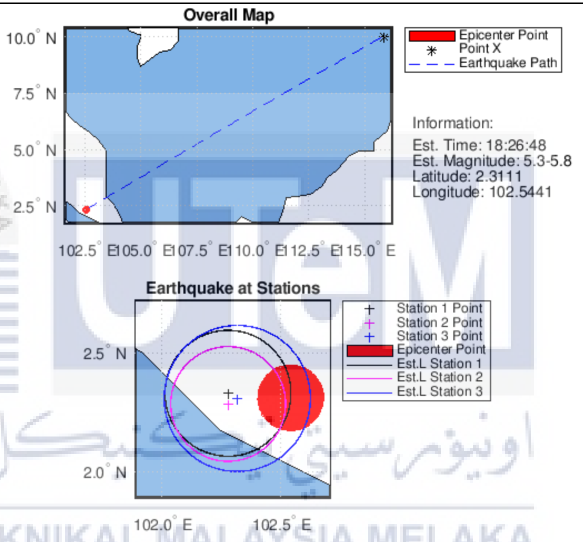
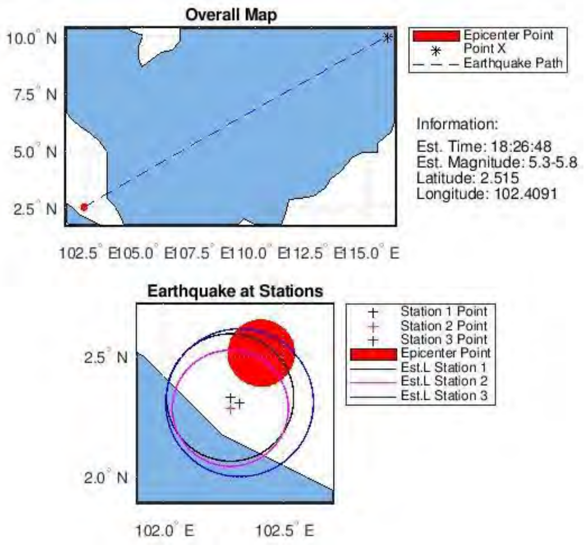
t_X is the P-wave travel time using Dx distance in s

t_r is the P-wave travel time using Dr distance in s

Table 3.9 below shows the generated map for each epicenter point. When the map is successfully plotted, a message “done” is sent to info@pi and triggers the system to move forward to last phase.

Table 3.9: Generated map for each epicenter point.

Epicenter	Earthquake Information Map
A	 Overall Map 10.0° N 7.5° N 5.0° N 2.5° N 102.5° E105.0° E107.5° E110.0° E112.5° E115.0° E Epicenter Point Point X Earthquake Path Information: Est. Time: 18:26:51 Est. Magnitude: 5.3-5.8 Latitude: 2.5225 Longitude: 102.164 Earthquake at Stations 2.5° N 2.0° N 102.0° E 102.5° E Station 1 Point Station 2 Point Station 3 Point Epicenter Point Est.L Station 1 Est.L Station 2 Est.L Station 3
B	 Overall Map 10.0° N 7.5° N 5.0° N 2.5° N 102.5° E105.0° E107.5° E110.0° E112.5° E115.0° E Epicenter Point Point X Earthquake Path Information: Est. Time: 18:26:54 Est. Magnitude: 5.3-5.8 Latitude: 2.3038 Longitude: 102.0552 Earthquake at Stations 2.5° N 2.0° N 102.0° E 102.5° E Station 1 Point Station 2 Point Station 3 Point Epicenter Point Est.L Station 1 Est.L Station 2 Est.L Station 3
C	 Overall Map 10.0° N 7.5° N 5.0° N 2.5° N 102.5° E105.0° E107.5° E110.0° E112.5° E115.0° E Epicenter Point Point X Earthquake Path Information: Est. Time: 18:26:54 Est. Magnitude: 5.3-5.8 Latitude: 2.0884 Longitude: 102.1707 Earthquake at Stations 2.5° N 2.0° N 102.0° E 102.5° E Station 1 Point Station 2 Point Station 3 Point Epicenter Point Est.L Station 1 Est.L Station 2 Est.L Station 3

Epicenter	Earthquake Information Map
D	 Information: Est. Time: 18:26:51 Est. Magnitude: 5.3-5.8 Latitude: 2.1034 Longitude: 102.4153
E	 Information: Est. Time: 18:26:48 Est. Magnitude: 5.3-5.8 Latitude: 2.3111 Longitude: 102.5441
F	 Information: Est. Time: 18:26:48 Est. Magnitude: 5.3-5.8 Latitude: 2.515 Longitude: 102.4091

3.3.3.6 Phase 5

The last stage of the system involves sending the generated map to Telegram channel Test@pi. When the Sensor Node 3 received message "done", it will download the generated map photo and upload it to Test@pi of telegram channel as final alert message. The alert information includes estimated earthquake epicenter, estimated impact time of P-wave to point X and the estimated earthquake magnitude.

3.4 Experiment Setup

A series of experiments to be conducted in this project are listed in this section according to the objectives to be achieved. A detailed explanation on each experiment is presented by listing out the materials and apparatus, hardware setup and the procedure which will be shown in Appendix D. Besides, short explanation about the experiment in terms of flow chart and expected outcome will also be discussed in this section.

There is rarely perfect flat surface on the ground layer of earth, thus the very small acceleration contributed by the slightly inclined accelerometer placed above the ground can be reduced by carrying out the calibration for every changed position. Simple calibration method called no-turn or single-point calibration will be used in the experiment to reduce the error of accelerometer using the Equation (3.11) shown below.

$$A_{OUT} = A_{OFF} + (\text{Gain} \times A_{ACT}) \tag{3.11}$$

where A_{OUT} is the output acceleration in g

A_{OFF} is the offset error in g

Gain is the gain of accelerometer

A_{ACT} is the real acceleration acting on the accelerometer in g

The output acceleration, A_{OUT} is the measured acceleration when the sensor is placed on a flat ground surface while the gain is set to an ideal value of 1 [46]. Since the acceleration is calculated by using the ADC value, thus the Equation (3.11) is changed from acceleration value to ADC value to simplify the calculation. The offset error, A_{OFF} is obtained by running a calibration test as shown in Appendix E. Since 0g is equal to 511.5 ADC value using the Equation (3.1), the A_{ACTUAL} is set to 511.5 to minus by the measured ADC value, A_{OUT} for 10 times repeat measurement. The offset errors of X-direction then can be obtained by averaging the difference. Then, the offset error could be inserted to the Equation (3.12) below to get the actual acceleration reading.

$$\mathrm{ADC}_{\mathrm{ACT}} = \mathrm{ADC}_{\mathrm{OUT}} - \mathrm{ADC}_{\mathrm{OFF}} \tag{3.12}$$

where ADC_{OUT} is the output ADC value

ADC_{OFF} is the offset error

ADC_{ACT} is the real ADC value sensed by the accelerometer

3.4.1 Objective 1

The first objective of the project is to design a disaster alert system that estimates the earthquake epicenter and arrival time of earthquake by using accelerometer.

3.4.1.1 Experiment 1: Experimenting Communication between Raspberry Pi

This experiment aims to check whether 2 Raspberry Pi can communicate to each other with a distance of exceeding 5km apart from each other. The communication is to be established with the success of receiving command from Telegram channel which run the shaking table and lastly uploads the sensor data to ThingSpeak channel. The result is tabulated to Table 3.10.

Table 3.10: Result of Experiment 1.

Test	Acceleration / ms^{-2}		Timestamp		Remark		
	Node 2	Node 3	Node 2	Node 3	Success	Fail	Reason

3.4.1.2 Experiment 2: Shaking Table Calibration

This experiment aims to compare the experimental acceleration measured from accelerometer with the theoretical actuator acceleration in order to find its corresponding gain using Equation (3.8). 3 different ranges of motor speed are to be found refer to Table 3.7 in order to simulate different earthquake scenarios in the 3 ranges of earthquake magnitudes where Range 1 is 5.3 to 5.8 magnitude, Range 2 is 5.9 to 6.4 magnitude and Range 3 is 6.5 to 7.0 magnitude . Then, the largest difference between the minimum and maximum of the acceleration values from each acceleration ranges of all 3 shaking tables will be taken to calculate the estimated epicenter radius. The sensor data will be tabulated in Table 3.11 below.

Table 3.11: Acceleration data of Experiment 2.

Range	1		2		3	
Duty Cycle (%)	35-59		35-70		43-95	
Test	a_1 / ms^{-2}	Gain	a_2 / ms^{-2}	Gain	a_3 / ms^{-2}	Gain

3.4.1.3 Experiment 3: Visualizing Predicted Earthquake Map in ThingSpeak

This experiment aims to check the reliability of designed earthquake early alert system with simulation of different earthquake scenarios using shaking table. Since there are 3 sensor nodes, thus total of 8 different earthquake scenarios can be simulated. However, the system is capable to sense earthquake hit only if the distance between the epicenter and the sensor nodes is around 25km. Thus, there are total of 6 different earthquake scenarios to be simulated. The generated earthquake map based on the analyzed sensor data from each sensor node will be evaluated and result will be tabulated in Table 3.12.

Table 3.12: Result of Experiment 3.

Test	**Acceleration / ms^{-2}**			**Timestamp**			**Evaluation**		
	Node 1	**Node 2**	**Node 3**	**Node 1**	**Node 2**	**Node 3**	**Success**	**Fail**	**Reason**

3.4.2 Objective 2

The next objective of the project is to develop a disaster messaging system for sending alert information and video recorded on earthquake scene by using Pi camera module through smartphone‟s application and web based network. There will be 3 experiments conducted to achieve this objective.

3.4.2.1 Experiment 4: Measuring Time Taken of Sending Message to Telegram

For this experiment, a Python script of sending earthquake information to Telegram using Telegram Bot is designed. Time required for the Telegram private channel to receive the message will be measured by comparing the timestamp of sending and receiving message recorded in Table 3.13.

Table 3.13: Comparison table for Experiment 4.

Test	**Timestamp before sending message**	**Timestamp after receiving message**	**Time difference / s**

The flow of this experiment is shown in Figure 3.17 below. The entire experiment is repeated for 10 times.

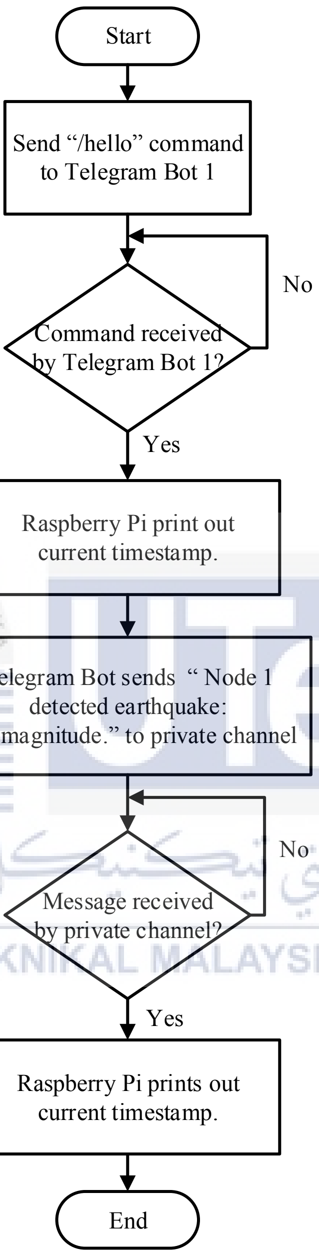

Start
Send "/hello" command to Telegram Bot 1
Command received by Telegram Bot 1?
No
Yes
Raspberry Pi print out current timestamp.
Telegram Bot sends " Node 1 detected earthquake: 3.7 magnitude." to private channel
Message received by private channel?
No
Yes
Raspberry Pi prints out current timestamp.
End

Figure 3.17: Flow chart of Experiment 4.

2 Telegram Bots will be created to send message and detect message sent since Telegram Bot is not able to read message sent itself. A Telegram private channel will be created and both Telegram Bots will be added as the administrators of the group. When the command "/hello" is received by Bot 1, the timestamp of sending message to the private channel by Telegram Bot 1 is printed out on a terminal just before

sending the alert message. When the message has been successfully delivered to the private channel, another Bot named Bot 2 is triggered to print out the delivered message and time the private channel received the message.

3.4.2.2 Experiment 5: Comparing Recorded Video Format

This experiment is conducted to find the video format that can be uploaded to Telegram channel in the shortest time. Thus, a few popular video formats are tested which includes Motion Picture Expert Group 4 (.mp4), Matroska (.mkv) and Audio Video Interleave (.avi). The flow of the experiment is similar to flow chart shown in Figure 3.17 above. The Telegram Bot 2 detects the video caption sent into the Telegram channel instead of detecting sent message like Experiment 3. All the videos will be set to the same duration of 5 seconds captured using the Pi camera module. All the video recorded with condition of 320x240 quality to reduce the size of the video for shorter upload time. Since different format of video may have different video processing time, thus the timestamp before capturing the video and the timestamp after the Telegram channel received the video will be compared to measure the time difference as recorded in Table 3.14 below. B is timestamp before record video, R is timestamp after receiving video in Telegram while D is the time difference between these 2 timestamp for each type of videos.

Table 3.14: Comparison table for Experiment 5.

Test	**MP4**			**MKV**			**AVI**		
	B	**R**	**D / s**	**B**	**R**	**D / s**	**B**	**R**	**D / s**

3.4.2.3 Experiment 6: Sending Earthquake Map to Telegram using ThingSpeak

To send the generated map to Telegram channel, the MATLAB Visualization program needs to be reloaded and the map is uploaded to Telegram using Telegram bot. Thus, this experiment aims to measure the time required for the Telegram private channel to receive the map after the reload command is received by Raspberry Pi. Using the acceleration data and timestamp from Experiment 3, the experiment is repeated for 10 times. The duration will be measured by comparing the timestamp of sending the reload command and receiving the map recorded in Table 3.15. S is timestamp after sending the reload command while R is timestamp receiving the map in Telegram channel. D is the time difference between S and R in seconds.

Table 3.15: Comparison table for Experiment 6.

Test	Generated Earthquake Information Map		
	S	R	D / s

3.4.3 Objective 3

The third objective of the project is to calculate the efficiency for delivering the alert message with video and photo attachment.

3.4.3.1 Experiment 7: Testing System Efficiency

The earthquake early alert system is tested to calculate its efficiency based on the time required of receiving final alert message in Telegram channel when a simulated earthquake scenario happens. 3 timestamp to be recorded in this experiment are: earliest time when a sensor data exceed the preset ADC threshold value (Timestamp 1), earliest time after Telegram channel received the early stage alert message (Timestamp 2), and also time for final alert message (Timestamp 3). Using the acceleration data and timestamp from Experiment 3, the experiment is repeated for 10 times and data will be recorded in Table 3.16 below. The time difference between 3 timestamps will be calculated where D1 is the time difference between timestamp 1 and 2, D2 is the time difference between timestamp 2 and 3 and finally D3 is the time difference between timestamp 1 and 3.

Table 3.16: Efficiency test.

Test	**Timestamp 1**	**Timestamp 2**	**Timestamp 3**	**D1 / s**	**D2 / s**	**D3 / s**

3.5 Summary

Overall, this chapter provides a detailed plan to ensure the entire project could be completed on time and the objectives of the project could be achieved. Most of the measurement involved in the experiments is repeated 10 times to verify the accuracy of the measurement since average of 10 sets of data has 80.5% more accuracy than a single trial [47]. A set of data is considered reliable if its deviation is small. The next chapter presents the results of the designed experiments and its corresponding data analysis.

CHAPTER 4

RESULT

4.1 Overview

All the results obtained from the conducted experiments listed in chapter 3 are recorded in either table or graph form for better understanding towards the parameters being measured from each experiment. All the timestamps collected through the experiments are to be obtained from programming codes or Telegram to avoid measurement errors.

4.2 Result of Experiment 1: Experimenting Communication between Raspberry Pi

The experiment setup of Experiment 1 is shown in Figure 4.1 below where 2 Raspberry Pi are placed in different locations.

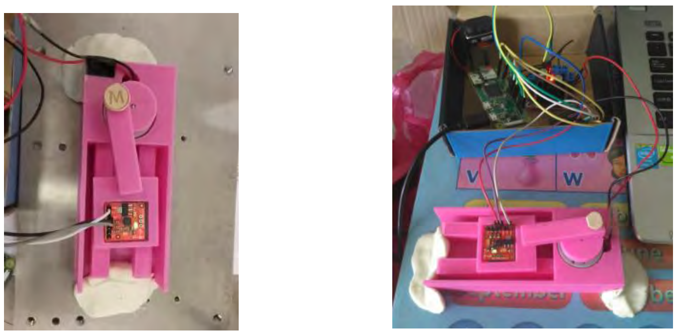


Figure 4.1: Experiment Setup of Experiment 1.

The locations of 2 Raspberry Pi is shown in Figure 4.2 below with coordinates of (2.245114, 102.279568) for Sensor Node 2 and (2.314750, 102.320278) for Sensor Node 3. 2 Raspberry Pi is 8.97 kilometers apart from each other. The Python script of this experiment is shown in Appendix F1.

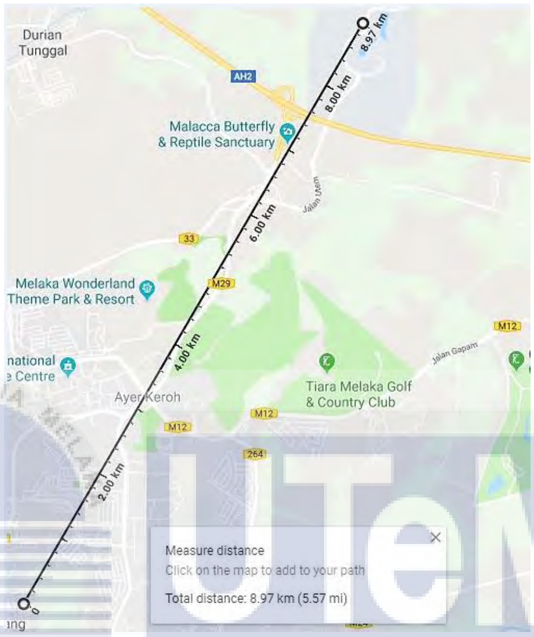

Durian Tunggal
AH2
Malacca Butterfly & Reptile Sanctuary
8.97 km
8.00 km
6.00 km
4.00 km
2.00 km
33
M29
M12
264
Melaka Wonderland Theme Park & Resort
national Centre
Ayer Keroh
Tiara Melaka Golf & Country Club
Jalan Gapam
Measure distance
Click on the map to add to your path
Total distance: 8.97 km (5.57 mi)

Figure 4.2: Distance between 2 Raspberry Pi.

The sensor data and the corresponding timestamp sent to Telegram from 2 Raspberry Pi are tabulated in Table 4.1 as refer to Table 3.10 in Chapter 3. The data is then compared with the plotted graph in ThingSpeak channel as shown in Figure 4.3 below. The first trial data mark has been left-shifted due to the incoming of new data.

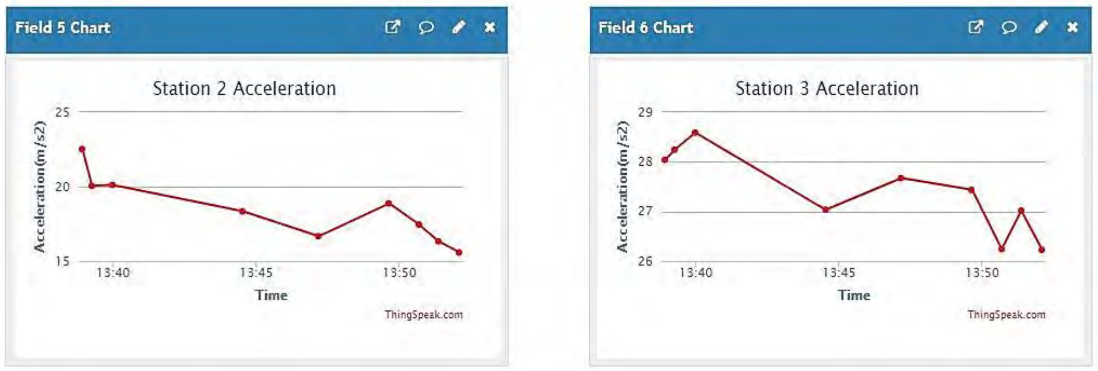


Figure 4.3: Plotted graph in ThingSpeak channel.

Table 4.1: Result of Experiment 1.

Test	Acceleration / ms^{-2}		Timestamp			Remark		
	Node 2	Node 3	Node 2	Node 3	Diff/s	Success	Fail	Reason
1	26.591	28.631	13:40:17	13:40:16	1	/		-
2	22.515	28.039	13:40:49	13:40:48	1	/		-
3	20.050	28.244	13:41:09	13:41:08	1	/		-
4	20.117	28.589	13:41:53	13:41:51	2	/		-
5	18.342	27.034	13:46:26	13:46:25	1	/		-
6	16.669	27.672	13:48:29	13:49:05	36	/		-
7	18.879	27.435	13:51:33	13:51:32	1	/		-
8	17.459	26.240	13:52:35	13:52:34	1	/		-
9	16.331	27.016	13:53:15	13:53:16	1	/		-
10	15.579	26.226	13:53:59	13:53:58	1	/		-

Refer to Table 4.1, 2 Raspberry Pi are able to run the shaking table based on the command sent to Telegram channel mostly within 1 seconds delay even though they are connected to different Wifi and located in different location. The data is able to be collected by Sensor Node 2 and uploaded to ThingSpeak channel in each trial. However, the time delays will increases as shown by the 6th trial when the Internet speed is slow. The acceleration data shown in Table 4.1 has not yet being multiplied with gain.

Besides, the timestamp tabulated in the table above refers to the timestamp on the Telegram Web when the data has been successfully sent to the Telegram channel. It delays about 2 minutes as compared to timestamp in ThingSpeak as shown in Figure 4.3 since Telegram follows the time of the laptop which open the Telegram Web while ThingSpeak follows global time. Since this experiment has proven that all 3 Raspberry Pi can communicate to each other with very small delay provided the Wifi speed is fast and consistent, hence the distance between all 3 Raspberry Pi can be simulated by manually inserting the coordinates of all 3 Raspberry Pi with distance of 5 kilometers apart into the coding. All the Raspberry Pi can be placed side by side and connect to different Wifi. This step is to ease the experiment setup and troubleshooting process.

4.3 Result of Experiment 2: Shaking Table Calibration

The experiment setup of Experiment 2 is shown in Figure 4.4 below where using all 3 Raspberry Pi and its shaking table. Clay is used on the front and end side of the shaking tables to fix their position during earthquake shaking simulation.

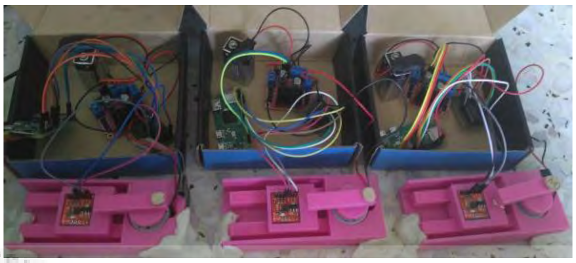


Figure 4.4: Experiment setup of Experiment 2.

All 3 Raspberry Pi will run at the same time when they are triggered by the command sent to Telegram channel. The acceleration data from each station measured by the accelerometer will be sent to Telegram channel for easier tabulation. Each Raspberry Pi has its own acceleration and its corresponding gain to be tabulated in table below which is built referring Table 3.11.

Table 4.2: Calibration data for Sensor Node 1.

Range	**1**		**2**		**3**	
Duty Cycle (%)	**35-59**		**35-70**		**43-95**	
Test	**a_1 / ms^{-2}**	**Gain**	**a_2 / ms^{-2}**	**Gain**	**a_3 / ms^{-2}**	**Gain**
1	15.641	0.065405	16.234	0.091906	15.921	0.139250
2	12.709	0.080494	17.817	0.083740	18.491	0.119896
3	14.644	0.069858	15.966	0.093449	15.031	0.147495
4	17.786	0.057517	17.859	0.083543	18.325	0.120982
5	12.113	0.084455	15.533	0.096054	16.877	0.131362
6	18.285	0.055947	15.258	0.097785	14.905	0.148742
7	16.852	0.060705	18.111	0.082381	17.924	0.123689
8	18.649	0.054855	15.963	0.093466	18.094	0.122527
9	15.159	0.067485	16.236	0.091895	15.755	0.140717
10	16.147	0.063355	17.829	0.083684	17.177	0.129068
	Average	0.066	**Average**	0.090	**Average**	0.132

Table 4.3: Calibration data for Sensor Node 2.

Range	**1**		**2**		**3**	
Duty Cycle (%)	**35-59**		**35-70**		**43-95**	
Test	**a_1 / ms^{-2}**	**Gain**	**a_2 / ms^{-2}**	**Gain**	**a_3 / ms^{-2}**	**Gain**
1	19.172	0.053359	20.303	0.073487	19.268	0.115061
2	21.220	0.048209	20.205	0.073843	19.584	0.113205
3	20.473	0.049968	20.626	0.072336	20.610	0.107569
4	20.136	0.050805	19.104	0.078099	21.204	0.104556
5	20.809	0.049161	20.069	0.074344	19.852	0.111676
6	19.894	0.051423	19.746	0.075560	19.124	0.115928
7	21.014	0.048682	19.71	0.075698	20.381	0.108778
8	20.217	0.050601	19.668	0.075859	19.530	0.113518
9	20.303	0.050387	19.639	0.075971	20.447	0.108427
10	19.780	0.051719	19.443	0.076737	18.988	0.116758
	Average	0.050	**Average**	0.075	**Average**	0.112

Table 4.4: Calibration data for Sensor Node 3.

Range	**1**		**2**		**3**	
Duty Cycle (%)	**35-59**		**35-70**		**43-95**	
Test	**a_1 / ms^{-2}**	**Gain**	**a_2 / ms^{-2}**	**Gain**	**a_3 / ms^{-2}**	**Gain**
1	26.904	0.047502	26.480	0.056344	26.023	0.085194
2	26.339	0.038840	26.162	0.057029	26.261	0.084422
3	27.163	0.037662	25.853	0.057711	25.412	0.087242
4	25.886	0.039519	26.142	0.057073	26.215	0.084570
5	26.401	0.038749	26.219	0.056905	26.465	0.083771
6	26.916	0.038007	26.044	0.057288	25.523	0.086863
7	26.534	0.038554	25.507	0.058494	25.877	0.085675
8	25.528	0.040074	26.241	0.056858	26.100	0.084943
9	26.202	0.039043	25.807	0.057814	26.306	0.084277
10	25.796	0.039657	26.374	0.056571	25.135	0.088204
	Average	0.040	**Average**	0.057	**Average**	0.086

Refer to above tables, the acceleration data are similar for each magnitude range although the speed of motor is different. This may due to the limitation where the sensor can only detect limited speed of shaking. Besides, since the motor rotates very fast for each magnitude range, the motor might achieve its own highest rotation speed for all 3 magnitude ranges. Thus, this causes motor to generate similar acceleration although the preset duty cycle is different for each magnitude range. The minimum and maximum ranges for Range 1 to Range 3 and its corresponding gains are found and tabulated to Table 4.5 below. The above gains are found using Equation (3.8) with the desired acceleration referring Table 3.7 for each magnitude range.

Table 4.5: Overall analysis on calibration data for each sensor node.

Range	**Gain**			**Final acceleration range (min, max), m/s^2**	**Median, m/s^2**	**Epicenter diameter/km**
	Sensor Node 1	**Sensor Node 2**	**Sensor Node 3**			
1	0.066	0.050	0.040	(0.800, 1.230)	1.015	30.83
2	0.090	0.075	0.057	(1.370, 1.626)	1.498	31.29
3	0.132	0.112	0.086	(1.973, 2.448)	2.211	31.75

From Table 4.5, the gain for each range of magnitude is found by averaging the gain of Table 4.2 to Table 4.4 for 10 trials. The gains will be used to calibrate the sensor data so that it lies within percentage error of 10% as refer to the theoretical acceleration. The minimum and maximum of the acceleration data from each range in Table 4.2 to Table 4.4 are multiplied with its corresponding average gain. Then, the final acceleration range is found by comparing all the minimum and maximum values from each sensor node. The median of the final acceleration range is found to calculate the epicenter diameter for each magnitude range by using Equation (3.2). However, due to the absence of filter for the sensor and closed-loop control system for the motor, the data shown in Table 4.5 will change for each calibration.

4.4 Result of Experiment 3: Visualizing Predicted Earthquake Map in ThingSpeak

The experiment setup of Experiment 3 is same as Experiment 2 as shown in Figure 4.4. Epicenter point A is chosen for the earthquake simulation throughout this entire experiment for all 3 magnitude ranges by assuming the result tabulated in Table 4.6 to Table 4.8 is same and applicable to other epicenter locations. The timestamp includes the milliseconds since the time delay for Station 2 and 3 is 0.6 seconds refer to the Table 3.8.

Table 4.6: Result for Range 2 with ranges from 1.370 to 1.626 m/s^2.

Test	Acceleration / ms^{-2}			Timestamp			Evaluation		
	Node 1	Node 2	Node 3	Node 1	Node 2	Node 3	Success	Fail	Reason
1	1.530	1.399	1.375	17:25:01.581	17:25:01.583	17:25:01.180		/	T
2	1.520	1.371	1.348	17:27:54.718	17:27:54.534	17:27:54.322		/	R,T
3	1.306	1.310	1.314	17:30:01.267	17:30:00.630	17:30:00.375		/	R,T
4	1.489	1.415	1.504	17:42:44.329	17:42:35.350	17:42:36.383		/	T
5	1.349	1.293	1.197	17:44:18.443	17:44:18.122	17:44:18.098		/	R,T
6	1.489	1.382	1.569	17:48:23.430	17:48:23.341	17:48:23.224		/	T
7	1.453	1.356	1.189	17:49:31.538	17:49:30.345	17:49:30.120		/	R,T
8	1.392	1.409	1.099	17:50:13.547	17:50:14.324	17:50:14.375		/	R
9	1.404	1.553	1.320	17:50:57.085	17:50:57.128	17:50:57.697		/	R,T
10	1.459	1.298	1.457	17:51:41.025	17:51:40.753	17:51:40.629		/	R,T

Table 4.7: Result for Range 3 with ranges from 1.973 to 2.448 m/s^2.

Test	Acceleration / ms^{-2}			Timestamp			Evaluation		
	Node 1	Node 2	Node 3	Node 1	Node 2	Node 3	Success	Fail	Reason
1	2.118	2.027	2.335	17:52:48.350	17:52:48.087	17:52:47.971		/	T
2	2.337	1.971	2.561	17:53:49.232	17:53:49.200	17:53:49.148		/	R,T
3	2.046	1.826	2.677	17:55:30.896	17:55:30.799	17:55:30.743		/	R,T
4	2.053	1.981	1.701	18:05:03.974	18:05:01.879	18:05:01.912		/	R,T
5	2.139	2.023	2.007	18:08:07.081	18:08:07.057	18:08:10.147		/	T
6	2.257	2.108	1.643	18:09:06.451	18:09:06.334	18:09:06.258		/	R,T
7	2.179	2.047	1.941	18:10:00.768	18:10:02.906	18:10:03.158		/	R
8	2.178	1.935	1.712	18:14:07.042	18:14:05.266	18:14:05.113		/	R,T
9	2.145	2.043	1.686	18:16:54.991	18:16:55.210	18:16:54.773		/	R,T
10	2.095	2.604	1.713	18:20:33.634	18:20:31.233	18:20:31.111		/	R,T

Table 4.8: Result for Range 1 with ranges from 0.800 to 1.230 m/s^2.

Test	Acceleration / ms^{-2}			Timestamp			Evaluation		
	Node 1	Node 2	Node 3	Node 1	Node 2	Node 3	Success	Fail	Reason
1	1.055	1.141	0.923	18:22:59.521	18:23:00.371	18:23:00.358	/		-
2	1.130	1.115	0.900	18:32:02.245	18:32:03.357	18:32:03.314	/		-
3	1.054	1.161	0.879	18:34:48.480	18:34:49.706	18:34:49.316	/		-
4	1.075	1.080	0.614	18:37:39.066	18:38:06.671	18:37:40.634		/	R,T
5	1.034	1.073	1.227	18:39:27.905	18:39:26.745	18:39:26.537		/	T
6	0.967	1.074	0.826	18:41:06.231	18:41:05.030	18:41:04.828		/	T
7	1.025	1.099	0.480	18:42:10.217	18:42:09.306	18:42:09.453		/	R,T
8	1.067	1.075	1.116	18:43:26.330	18:43:26.275	18:43:26.062		/	T
9	1.138	1.082	0.620	18:46:08.928	18:46:11.770	18:46:07.424		/	R,T
10	0.996	0.994	1.069	18:52:07.834	18:52:08.886	18:52:07.567		/	T

Using the success and failure rates from above tables, a bar chart is plotted to illustrate the data clearly.

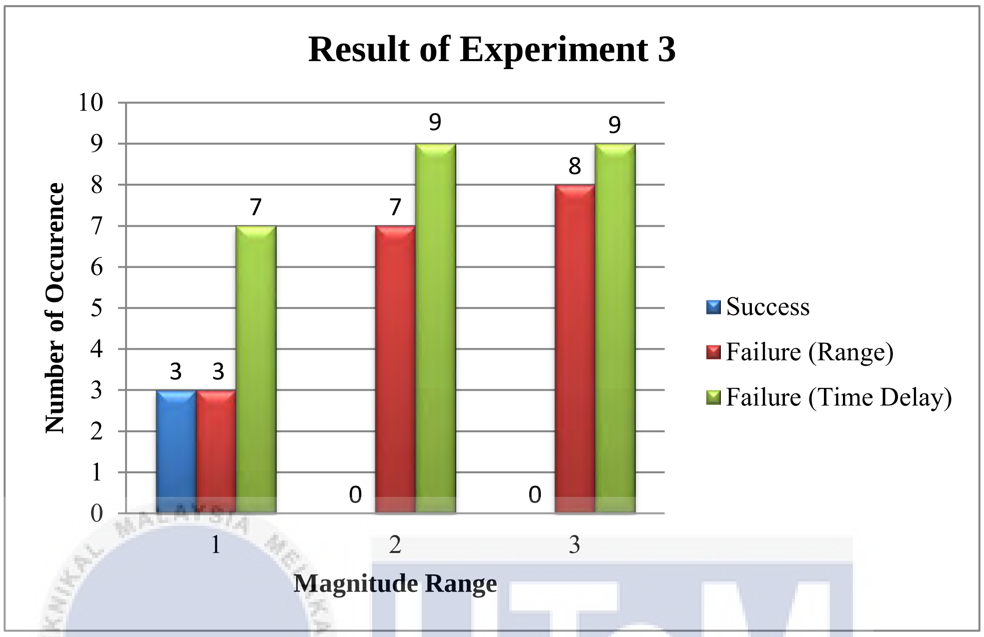
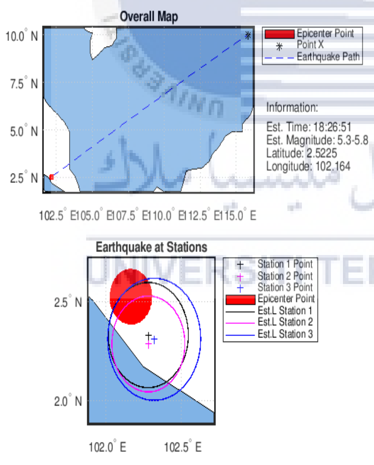


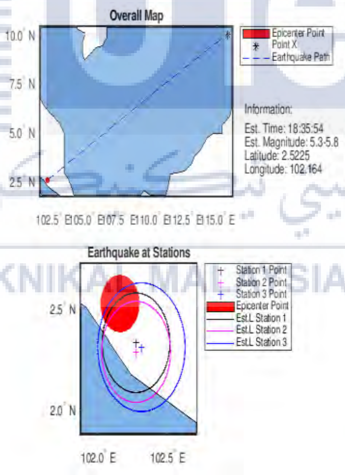
Figure 4.5: Bar chart showing the overall result of Experiment 3.

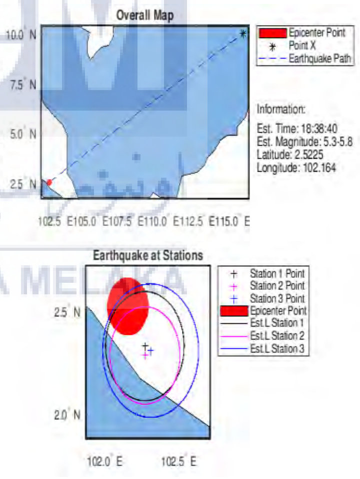
Refer to Figure 4.5, the success rate for the system to detect correctly the epicenter location A is very low around 10% for a total of 30 trials in 3 magnitude ranges. Only magnitude range 1 with acceleration ranges from 0.800 to 1.230 m/s^2 successfully trigger the system to send the final alert message.

The failure is divided into 2 categories where "R" relates to the failure of hardware. "T" relates to the inconsistent of Wifi connection speed and Raspberry Pi processing speed with the highest number of occurrence in all 3 magnitude ranges. Hardware failure happens when shaking table fails to simulate the expected acceleration or the noise existence in sensor data. For this earthquake scenario, Station 1 is expected to be the first station to receive the P-wave hit. However, due to the connection speeds of all 3 Raspberry Pi are different where Station 1 is connected to home Wifi with speed of 10 Megabit per seconds and the rest are connected to different smartphone hotspot with 3G Digi and 4G Digi Wifi speed respectively, the time of receiving the command to trigger the shaking table is delayed and thus giving the wrong timestamp. Besides, all 3 Raspberry Pi processing speed are different. Thus, time to run the scripts is delayed and hence giving the wrong timestamp especially in Sensor Node 1 since it uses class 4 Micro SD card with 8 GB storage which has slower processing speed and lower available storage compared to class 10 Micro SD card with 16 GB storage for another 2 sensor nodes. Mostly, the inconsistent Wifi speed causes the system to wrongly estimate the epicenter point. Only a few times that the system is unable to analyze the data as the duration of timestamps between the sensor nodes are out of range as refer to Table 3.8 caused by inconsistent Wifi speed.

The maps that are successfully sent into the Telegram channel are shown in Figure 4.6 below.







(a) (b) (c)

Figure 4.6: a) First trial b) Second trial c) Third trial.

The estimated P-wave arrival time shown in Figure 4.6 above is compared with the Telegram timestamp where the map is uploaded successfully to Telegram channel to know the duration for evacuation which the system is capable to do so. The data is tabulated into Table 4.9.

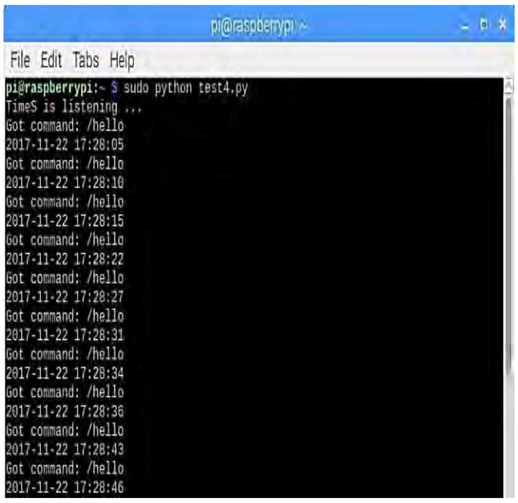
Table 4.9: Duration between the estimated P-wave arrival time and received time.

Trial of data set	P-wave arrival time	Telegram Timestamp	Duration/s
1	18:26:51	18:25:50	61
2	18:35:54	18:35:26	28
3	18:38:40	18:38:24	16
		Average	35

Based on Table 4.9, this system is capable to earn an average duration of 35 seconds for evacuation purpose before P-wave arrives Point X. However, this duration is obtained with the requirement of a minimum distance of 1700 km between destination Point X and epicenter point E. This may due to the processing time required for the system is too long since the process involves communication of data through Telegram and Chromium browser page reload in Sensor Node 3 for data update on ThingSpeak MATLAB Visualization program. Besides, estimation on the P-wave arrival time involves very simple calculation which might affect the accuracy of the calculated arrival time. Lastly, this issue may cause by the limitation on the hardware specification of Raspberry Pi Zero W which has slower performance speed and lower memory for running multiple scripts at the same time. The duration shown in Table 4.9 may be lengthened if the Internet connection is stronger and more stable.

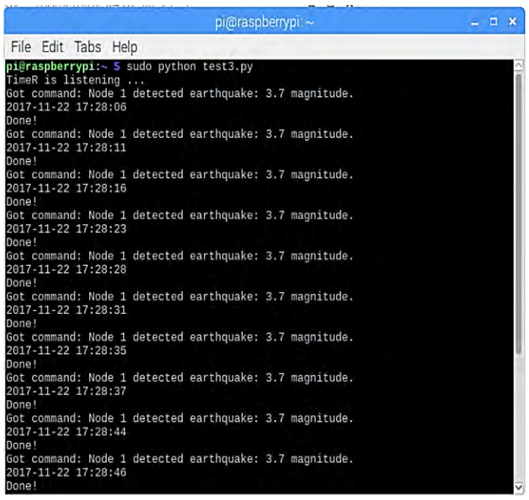
4.5 Result of Experiment 4: Measuring Time Taken of Sending Message to Telegram

The timestamp of sending message and receiving message are printed out in 2 different terminals executed in Raspberry Pi Zero W for 10 times experiments as shown in Figure 4.7 (a) and Figure 4.7 (b).



```
pi@raspberrypi:~ $ sudo python test4.py
TimeS is listening ...
Got command: /hello
2017-11-22 17:28:05
Got command: /hello
2017-11-22 17:28:10
Got command: /hello
2017-11-22 17:28:15
Got command: /hello
2017-11-22 17:28:22
Got command: /hello
2017-11-22 17:28:27
Got command: /hello
2017-11-22 17:28:31
Got command: /hello
2017-11-22 17:28:34
Got command: /hello
2017-11-22 17:28:36
Got command: /hello
2017-11-22 17:28:43
Got command: /hello
2017-11-22 17:28:46
```

(a)



```
pi@raspberrypi:~ $ sudo python test3.py
TimeR is listening ...
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:06
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:11
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:16
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:23
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:28
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:31
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:35
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:37
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:44
Done!
Got command: Node 1 detected earthquake: 3.7 magnitude.
2017-11-22 17:28:46
Done!
```

(b)

Figure 4.7: a) Timestamp sending message b) Timestamp receiving message.

Both timestamps shown in the terminals are then tabulated in Table 4.10 as refer to Table 3.13 in Chapter 3. The time difference between 2 timestamps is calculated and is recorded in the same table.

Table 4.10: Result of Experiment 4.

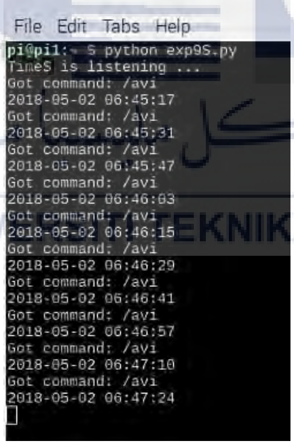
Test	**Timestamp before sending message**	**Timestamp after receiving message**	**Time difference / s**
1	17:28:05	17:28:06	1
2	17:28:10	17:28:11	1
3	17:28:15	17:28:16	1
4	17:28:22	17:28:23	1
5	17:28:27	17:28:28	1
6	17:28:31	17:28:31	0
7	17:28:34	17:28:35	1
8	17:28:36	17:28:37	1
9	17:28:43	17:28:44	1
10	17:28:46	17:28:46	0
		Average	0.8

The Python scripts for both sending and detecting message are shown in Appendix F4. The entire process of each trial will only be repeated when a message "Done!" is shown on the terminal to indicate both timestamp have been collected. This experiment is carried out by assuming the Wi-Fi speed connected by Raspberry

Pi Zero W is in good condition throughout every trial. From the calculated average value shown in Table 4.10, an alert message can be delivered to Telegram private channel within 0.8 seconds using Raspberry Pi connected to a 10 Megabit per seconds Wi-Fi. The time needed for sending message to Telegram may reduce from seconds to milliseconds if the Internet connection is strong. This is proven by the zero time difference obtained in test 6 and test 10 shown in Table 4.10.

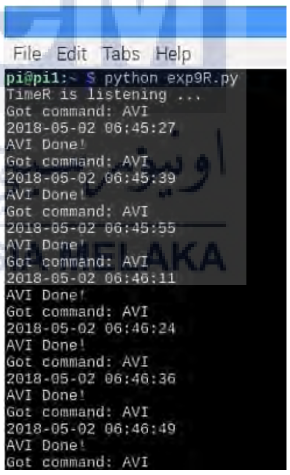
4.6 Result of Experiment 5: Comparing Recorded Video Format

The timestamp of sending video and receiving video are printed out in 2 different terminals executed in Raspberry Pi Zero W for 10 times experiments as shown in Figure 4.8 (a) and Figure 4.8 (b).



```
pi@pi1:~ $ python exp9S.py
TimeS is listening ...
Got command: /avi
2018-05-02 06:45:17
Got command: /avi
2018-05-02 06:45:31
Got command: /avi
2018-05-02 06:45:47
Got command: /avi
2018-05-02 06:46:03
Got command: /avi
2018-05-02 06:46:15
Got command: /avi
2018-05-02 06:46:29
Got command: /avi
2018-05-02 06:46:41
Got command: /avi
2018-05-02 06:46:57
Got command: /avi
2018-05-02 06:47:10
Got command: /avi
2018-05-02 06:47:24
```

(a)



```
pi@pi1:~ $ python exp9R.py
TimeR is listening ...
Got command: AVI
2018-05-02 06:45:27
AVI Done!
Got command: AVI
2018-05-02 06:45:39
AVI Done!
Got command: AVI
2018-05-02 06:45:55
AVI Done!
Got command: AVI
2018-05-02 06:46:11
AVI Done!
Got command: AVI
2018-05-02 06:46:24
AVI Done!
Got command: AVI
2018-05-02 06:46:36
AVI Done!
Got command: AVI
2018-05-02 06:46:49
AVI Done!
Got command: AVI
```

(b)

Figure 4.8: a) Timestamp sending video b) Timestamp receiving video.

Both timestamps shown in the terminals are then tabulated in Table 4.11 as refer to Table 3.14 in Chapter 3. The time difference between 2 timestamps is calculated and is recorded in the same table according to the tested video format. As mentioned before, B is timestamp before record video, R is timestamp after receiving video in Telegram while D is the time difference between these 2 timestamp for each type of videos.

Table 4.11: Comparison table for Experiment 5.

Test	MP4			MKV			AVI		
	B	**R**	**D / s**	**B**	**R**	**D / s**	**B**	**R**	**D / s**
1	07:06:12	07:06:24	12	06:57:17	06:57:29	12	06:45:17	06:45:27	10
2	07:06:27	07:06:39	12	06:57:36	06:57:48	12	06:45:31	06:45:39	8
3	07:06:42	07:06:56	14	06:57:51	06:58:04	13	06:45:47	06:45:55	8
4	07:07:01	07:07:13	12	06:58:07	06:58:19	12	06:46:03	06:46:11	8
5	07:07:18	07:07:30	12	06:58:20	06:58:32	12	06:46:15	06:46:24	9
6	07:09:03	07:09:16	13	07:00:42	07:00:54	12	06:46:29	06:46:36	7
7	07:09:18	07:09:30	12	07:00:57	07:01:09	12	06:46:41	06:46:49	8
8	07:09:34	07:09:46	12	07:01:15	07:01:28	13	06:46:57	06:47:06	9
9	07:10:05	07:10:17	12	07:01:31	07:01:43	12	06:47:10	06:47:19	9
10	07:10:21	07:10:33	12	07:01:47	07:01:58	11	06:47:24	06:47:33	9
		Average	12.3		**Average**	12.1		**Average**	8.5

A line graph illustrating the footage processing time, which covers the recording time until the receiving time by Telegram of all 3 video formats calculated in Table 4.11 is shown in Figure 4.9 below.

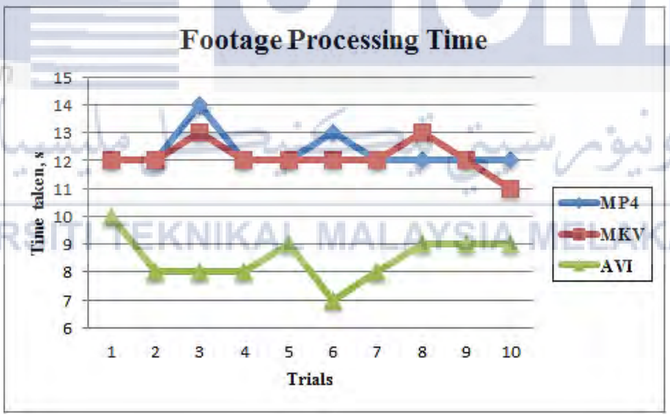


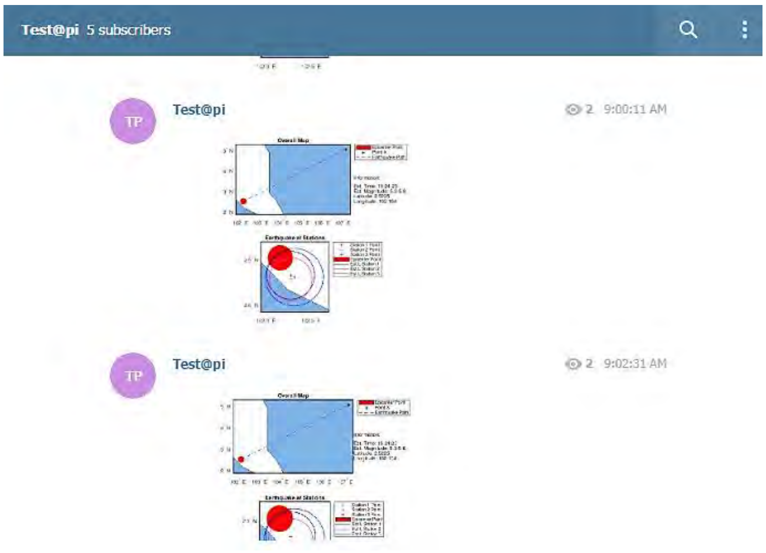
Figure 4.9: Footage processing time of 3 video formats.

Similar to Experiment 4, the Python scripts for both sending and detecting video are shown in Appendix F5. The entire process of each trial will only be repeated when a message "Done!" is shown on the terminal to indicate both timestamp have been collected. This experiment is carried out by assuming the Wi-Fi speed connected by Raspberry Pi Zero W is in good condition throughout every trial. In this experiment, the Raspberry Pi is connected to a 10 Megabit per seconds Wi-Fi.

The Figure 4.9 shows that Audio Video Interleave (.avi) takes less time for video processing in 10 time trials. Thus, the online source which mentioned Audio Video Interleave (.avi) has no encode and decode processes like Matroska (.mkv) and Motion Picture Expert Group 4 (.mp4) video files is verified through the data obtained in Table 4.11 [48]. Time required to record and deliver footage to Telegram private channel can be varies from an average of 8.5 seconds to 12.3 seconds with different video formats as shown in Table 4.11. Besides, only Matroska (.mkv) and Audio Video Interleave (.avi) remain as video files when they are sent to Telegram channel. The Motion Picture Expert Group 4 (.mp4) video is played in the form of Graphics Interchange Format (GIF). This has verified the online source that mentioned the conversion caused by size of Motion Picture Expert Group 4 (.mp4) video less than 10 Megabytes, without audio track and use of h.264 video codec [49]. Therefore, Audio Video Interleave (.avi) is suitable to be used in the project for validation purpose since it consumes the least time and achieves the smoothest video among 3 video formats.

4.7 Result of Experiment 6: Sending Earthquake Map to Telegram using ThingSpeak

Similar to Experiment 4 and 5, Raspberry Pi reloads the ThingSpeak Visualization program page opened using Chromium browser when it receives command of “reload” from Telegram channel. As mentioned before, S is timestamp after sending the reload command while R is timestamp receiving the map in Telegram channel. D is the time difference between S and R in seconds. S and R are obtained through Telegram Web since it can show the seconds unit of the timestamp as shown in Figure 4.10 below.


Test@pi 5 subscribers

Test@pi 2 9:00:11 AM

Test@pi 2 9:02:31 AM

Figure 4.10: Timestamp referring to Telegram Web.

Table 4.12: Comparison table for Experiment 6.

Test	Generated Earthquake Information Map		
	S	**R**	**D / s**
1	08:38:22	08:38:57	35
2	08:40:12	08:40:49	37
3	08:42:17	08:42:51	34
4	08:50:32	08:51:05	33
5	08:52:37	08:53:08	31
6	08:54:20	08:54:51	31
7	08:56:14	08:56:47	33
8	08:58:15	08:58:45	30
9	08:59:37	09:00:11	34
10	09:02:00	09:02:31	31
		Average	32.9

Due to the failure rate of the system is too high, the acceleration data and its corresponding timestamps for the P-wave arrival time of each sensor node are referred to the first trial success cases in Experiment 3 of Table 4.8. Based on the result shown in Table 4.12, the average of process time required for the Raspberry Pi to send the earthquake information map to Telegram channel as final alert message is 32.9 seconds with the assumption that the 10 Megabit per seconds Wi-Fi speed connected by Raspberry Pi Zero W is in good condition throughout every trial.

4.8 Result of Experiment 7: Testing System Efficiency

Similar to the previous experiment, the acceleration data and its corresponding timestamps for the P-wave arrival time of each sensor node are referred to the first trial success cases in Experiment 3 of Table 4.8. However, all the system phases mentioned in Chapter 3 will be run through to calculate system efficiency based on the time required of receiving final alert message in Telegram channel when a simulated earthquake scenario happens. The experiment setup is shown in Figure 4.11 below.

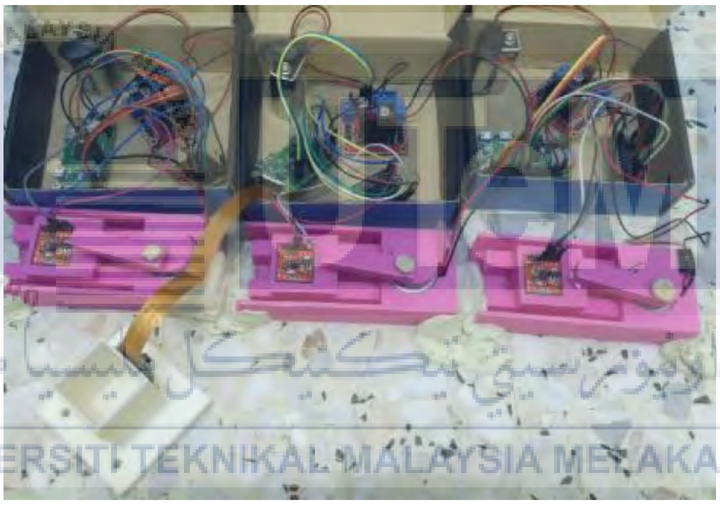
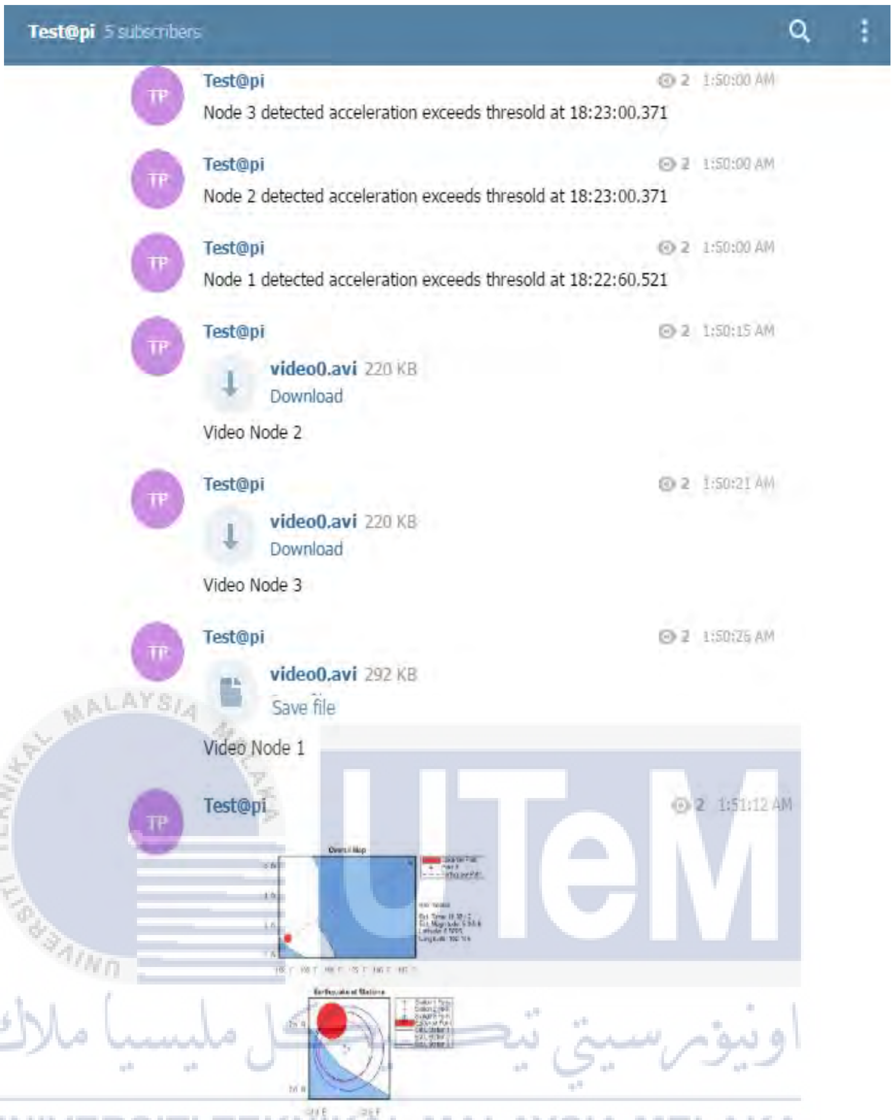


Figure 4.11: Experiment Setup of Experiment 7.

As mentioned before, 3 timestamp to be recorded in Table 4.13 are: earliest time when a sensor data exceed the preset ADC threshold value (Timestamp 1), earliest time after Telegram channel received the early stage alert message (Timestamp 2), and also time for final alert message (Timestamp 3). The time difference between 3 timestamps will be calculated where D1 is the time difference between timestamp 1 and 2, D2 is the time difference between timestamp 2 and 3 and finally D3 is the time difference between timestamp 1 and 3. The messages sent to Telegram channel for alert purpose are shown in Figure 4.12 below.


Test@pi 5 subscribers

Test@pi 2 1:50:00 AM
Node 3 detected acceleration exceeds thresold at 18:23:00.371

Test@pi 2 1:50:00 AM
Node 2 detected acceleration exceeds thresold at 18:23:00.371

Test@pi 2 1:50:00 AM
Node 1 detected acceleration exceeds thresold at 18:22:60.521

Test@pi 2 1:50:15 AM
video0.avi 220 KB
Download
Video Node 2

Test@pi 2 1:50:21 AM
video0.avi 220 KB
Download
Video Node 3

Test@pi 2 1:50:26 AM
video0.avi 292 KB
Save file
Video Node 1

Test@pi 2 1:51:12 AM

Figure 4.12: Alert messages in Telegram channel.

Table 4.13: Efficiency test.

Test	Timestamp 1	Timestamp 2	Timestamp 3	D1 / s	D2 / s	D3 / s
1	01:50:00	01:50:00	01:51:12	0	72	72
2	01:58:47	01:58:47	01:59:41	0	54	54
3	02:01:27	02:01:27	02:02:27	0	60	60
4	02:03:13	02:03:13	02:04:20	0	67	67
5	02:05:34	02:05:34	02:06:58	0	84	84
6	02:08:11	02:08:11	02:09:20	0	69	69
7	02:09:50	02:09:50	02:11:04	0	74	74
8	02:11:35	02:11:35	02:12:56	0	81	81
9	02:14:18	02:14:18	02:15:12	0	54	54
10	02:15:44	02:15:44	02:16:50	0	66	66
			Average	0	68.1	68.1

Based on Table 4.13 above, an average of 68.1 seconds processing time are required for the system to deliver the final alert message into the Telegram channel once one of the sensor node detected arrival of P-wave referring the value D3. This processing time can be reduced if the connected Wifi speed for each Raspberry Pi is very fast. Next, the average processing time is same as D2 average value due to the time for the early alert message to deliver into the Telegram channel after the sensor data exceeds preset threshold value is very short within one second as proven by Experiment 4.

4.9 Cost Evaluation

The components installed in each sensor node are listed in Table 4.14 below with its corresponding cost.

Table 4.14: Cost per sensor node.

No.	Component	Quantity	Cost / RM
1	Raspberry Pi Zero W	1	42.40
2	5V Micro USB Power Adapter	1	39.29
3	MCP 3008 ADC Converter	1	9.92
4	Raspberry Pi Camera Module Version 1.3	1	79.74
5	ADXL 335 Accelerometer	1	39.22
6	Micro SD card	1	33.03
		Total	243.60

Refer to Table 4.14, the cost for each sensor node is RM 243.60. Thus total of RM 730.8 are needed to build 3 sensor nodes for triangulation method.

4.10 Summary

Analysis and discussion on the outputs obtained from total of 6 conducted experiments are done with proper explanation on presented tables or figures.

Throughout the experiments, few issues which can affect the system performance and effectiveness are concluded with its corresponding solution. Firstly, the system is able to send the final alert message before the arrival of P-wave to the destination only when the distance between the earthquake epicenter coordinate and the destination equal or longer than 1700 km. This issue might be solved by further simplified the data analysis procedure and fasten the data delivery speed by using higher speed of Wifi. Next, the sensor data is highly inconsistent as proven in Experiment 3. This issue might solve by adding a filter circuit to the system and undergoing full calibration on the accelerometer. Lastly, the system is the very first prototype to establish a standalone IoT based earthquake early alert system without the aids of any personal computer. The algorithms used to predict the epicenter and P-wave travel time is not applicable to real situation due to the ignorance on the geographical characteristic of the earth. The next chapter concludes the project findings and further improvement suggestion.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

All the objectives are achieved where a low cost disaster alert system that estimates the earthquake epicenter and arrival time of P-wave of incoming strong earthquake towards the destination has been successfully developed. All 3 sensor nodes are able to communicate to each other through Telegram mostly within 1 seconds delay per message. Besides, each sensor node can be remotely accessed as long as Internet connection is available. Overall, the system has 10% success rate to estimate earthquake epicenter and its shortest arrival time to a destination correctly using simple algorithms provided that the epicenter falls within the preset 6 directions in a distance of around 25 kilometers away from the sensor nodes and Internet connection is always available for all sensor nodes. The cost per sensor nodes is RM 243.60. Next, this system is verified by using simulated earthquake scenario through 3D printed shaking table connected to each sensor node. The disaster messaging system is able to send alert information and video recorded on earthquake scene to the authorities through Telegram private channel when P-wave of the earthquake hits the sensor nodes. The sensor data can be monitored and stored online for future reference through ThingSpeak channel. Lastly, the proposed system is capable to deliver final alert message in the forms of map using online ThingSpeak MATLAB Visualization application for better understanding on the information within 68.1 seconds and earns an average period of 35 seconds respond time for immediate evacuation or other purposes before P-wave hits the destination.

5.2 Future Work

For further improvement, a closed-loop control system for the shaking table is recommended so that the generated acceleration can achieve the theoretical value without the needs of multiplying the sensor data with gain. Besides, a filter circuit should be added into the system to remove noise from the sensor data which in turns increase the data set accuracy. Next, all sensor nodes should use the same model of hardware including its Micro SD card to solve the difference in processing speed on the Python scripts. The way to handle the sensor data for alert purpose is recommended to be simplified to further enhance the processing time for final stage alert message delivery which in turns increase the respond time for the citizen to prepare for the earthquake impact. Last but not least, more sophisticated algorithm which involves the depth beneath the earth"s surface and the geographical characteristic for accurate estimation of earthquake hypocenter and P-wave arrival time for alert purpose.

REFERENCES

[1] S. A. Chowdhury et al., “Depiction of an Interactive Prevarication System during Exigency Situation,” in *2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, 2016, 978-1-5090-2906-8. Doi: 10.1109/CEEICT.2016.7873140

[2] “World Disasters Report 2016 | IFRC campaigns.” [Online]. Available: http://www.ifrc.org/Global/Documents/Secretariat/201610/WDR%202016-FINAL_web.pdf. [Accessed: 15-Sep-2017].

[3] “Natural disasters in the world between 2001-2016: breakdown of risks per region.” [Online]. Available: http://www.atlas-mag.net/en/article/natural-catastrophes-2001-2016-breakdown-of-risks-per-region. [Accessed: 19-Sep-2017].

[4] N. Chavez et al., “Central Mexico earthquake kills more than 200, topples buildings,” *CNN*, 20-Sep-2017. [Online]. Available: http://edition.cnn.com/2017/09/19/americas/mexico-earthquake/index.html. [Accessed: 22-Sep-2017].

[5] R. Sanchez et al., “At least 247 killed in earthquake in central Italy,” *CNN*, 25-Aug-2016. [Online]. Available: http://edition.cnn.com/2016/08/23/europe/italy-earthquake/index.html. [Accessed: 24-Sep-2017].

[6] K. A. Kili, “JB shaken by Indonesia earthquake,” *THE Star ONLINE,* 13-Aug-2017. [Online]. Available: https://www.thestar.com.my/news/nation/2017/08/13/jb-shaken-by-indonesia-earthquake/. [Accessed: 25-Sep-2017].

[7] “Sabah earthquake a 2015 shock for the nation.” [Online]. Available: http://www.themalaymailonline.com/malaysia/article/sabah-earthquake-a-2015-shock-for-the-nation#JUIyD4zUa5F7ytXH.97. [Accessed: 25-Sep-2017].

[8] “Disaster.” [Online]. Available: https://en.oxforddictionaries.com/definition/disaster. [Accessed: 26-Sep-2017].

[9] T. Songer, “Epidemiology of Disasters,” *University of Pittsburgh,* 23-April-1999. [Online]. Available: http://www.pitt.edu/~epi2170/lecture15/index.htm. [Accessed: 28-Sep-2017].

[10] J. Chen, “Modern Disaster Theory: Evaluating Disaster Law as a Portfolio of Legal Rules,” *Emory University School of Law*. [Online]. Available: http://law.emory.edu/eilr/content/volume-25/issue-3/symposium/modern-disaster-theory-evaluating-law-rules.html. [Accessed: 28-Sep-2017].

[11] “EM-DAT Disaster Classification.” [Online]. Available: http://www.emdat.be/classification. [Accessed: 28-Sep-2017].

[12] C. Bach et al., “Topics Geo,” Münchener Rückversicherungs-Gesellschaft Koniginstrasse 107, 80802 München, Germany. 302-09006, 2017

[13] L. Wald, “The Science of Earthquakes,” *USGS*. [Online]. Available: https://earthquake.usgs.gov/learn/kids/eqscience.php.[Accessed: 29-Sep-2017].

[14] M. S. Liew et al., “Ground Motion Prediction Equations for Malaysia due to Subduction Zone Earthquakes in Sumatran Region,” 2017. Doi: 10.1109/ACCESS.2017.2748360

[15] M. R. Chowdhury et al., “Response in the field intensity of Sferics at 40 kHz signal propagation caused by massive earthquake tremors in Kolkata,” *in 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, 2016, 978-1-5090-4432-0. Doi: 10.1109/ICCECE.2016.8009571

[16] R. Hoque et al., “Earthquake Monitoring and Warning System,” in *2015 3rd International Conference on Advances in Electrical Engineering*, 2015, pp. 109–112.

[17] A. N. Manafizad et al., “Estimation of Peak Ground Acceleration (PGA) for Peninsular Malaysia using geospatial approach,” in *8th IGRSM International Conference and Exhibition on Remote Sensing & GIS (IGRSM 2016)*, 2016. Doi: 10.1088/1755-1315/37/1/012069

[18] Y. Sherki et al., “Design of Real Time Sensor System for Detection and Processing of Seismic Waves for Earthquake Early Warning System,” in *2015 International Conference on Power and Advanced Control Engineering (ICPACE)*, 2015, pp. 285–289.

[19] M. Oguz et al., “Earthquake Preparedness Strategies for Telecom Backbone with Integration of Early Warning Systems and Optical WDM Networks,” in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016, pp. 181–188.

[20] Alphonsa A. and Ravi G., “Earthquake Early Warning System by IOT using Wireless Sensor Networks,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016, pp. 1201–1205.

[21] R. D. Singh et al., “Seismic Early Warning Alert System,” in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014, pp. 601–605.

[22] A. Z. Shukor et al., “Investigation on A New Earthquake and Flood Alert System,” in *International Symposium on Research in Innovation and Sustainability 2017 (ISoRIS ’17))*, July 2017.

[23] H. Afzaal. and N. A. Zafar, “Towards Formalism of Earthquake Detection and Disaster Reduction using WSANs,” in *2016 International Conference on Frontiers of Information Technology*, 2016, pp. 147–152.

[24] Osamu Kamigaichi, “JMA Earthquake Early Warning,” *Journal of Japan Association for Earthquake Engineering*, Vol.4, No.3 (Special Issue), pp. 134-137, 2004.

[25] Horiuchi et al., “An Automatic Processing System for Broadcasting Earthquake Alarms,” *Bulletin of the Seismological Society of America*, Vol. 95, No. 2, pp. 708–718, April 2005. Doi: 10.1785/0120030133

[26] J. Li et al., “Comparison of Two Earthquake Early Warning Location Methods,” *Earthquake Science*, Vol.26, Issue 1, pp. 15-22, February 2013. Doi: 10.1007/s11589-013-0002-7

[27] “About disaster management.” [Online]. Available: http://www.ifrc.org/en/what-we-do/disaster-management/about-disaster-management/. [Accessed: 10-Oct-2017].

[28] National Institute of Disaster Management, “Understanding Disaster,” *nidm* [Online]. Available: http://nidm.gov.in/PDF/Disaster_about.pdf [Accessed: 10-Oct-2017].

[29] M. R .M. Aldecimo and M. M. D. Leon, “Development of an OpenStreetMap Based Safe Zone Routing System for West Valley Fault Earthquake Disaster in the Makati Central Business District, Philippines,” in *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, 2016. Doi: 10.1109/ICT-DM.2016.7857219

[30] Matt, “Introducing Raspberry Pi Zero W,” *Raspberry Pi Spy* [Online]. Available: https://www.raspberrypi-spy.co.uk/2017/02/introducing-the-raspberry-pi-zero-w/. [Accessed: 14-Nov-2017].

[31] “Raspberry Pi Zero W Specification.” [Online]. Available: https://www.sparkfun.com/products/14277. [Accessed: 16-Nov-2017].

[32] “Accelerometer ADXL335 Datasheet.” [Online]. Available: https://drive.google.com/file/d/0BzFWfMiqqjyqVFRHTEJKQXotVXM/view. [Accessed: 18-Nov-2017].

[33] “Raspberry Pi Camera Board Version 1.3.” [Online]. Available: https://uk.pi-supply.com/products/raspberry-pi-camera-board-v1-3-5mp-1080p. [Accessed: 8-Apr -2018].

[34] F. A. Vieira, “VIII EBEC Porto - Team Design”, 22 October 2012. [Online]. Available: https://www.youtube.com/watch?v=wJJn_hwjhVM. [Accessed: 13-Mar-2018].

[35] M. A. G. Maureira et al., “ThingSpeak – an API and Web Service for the Internet of Things.”

[36] G. Jadhav et al., “Environment Monitoring System using Raspberry-Pi,” *International Research Journal of Engineering and Technology (IRJET),* vol. 3, issue. 4, pp. 1168–1172, Apr. 2016.

[37] R. Chandana et al., “Smart Surveillance System using Thing Speak and Raspberry Pi,” *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4, issue. 7, pp. 214–218, July. 2015.

[38] S. Gangopadhyay and M. K. Mondal, “A Wireless Framework for Environmental Monitoring and Instant Response Alert,” in *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*, 2016, 978-1-4673-6621-2. Doi: 10.1109/MicroCom.2016.752253516.8009571

[39] “Telegram.” [Online]. Available: https://telegram.org/. [Accessed: 18-Nov-2017].

[40] Nowroozi A.A and Ahmadi G., “Analysis of Earthquake Risk in Iran Based on Seismotectonic Provinces,” *Tectonophysics*, 122, pp. 89–114, 1986.

[41] E. M. D. Baer, “Shaking Ground – Linking Earthquake Magnitude and Intensity,” *National Science Foundation* [Online]. Available: https://serc.carleton.edu/introgeo/ssac/examples/eqshake.html. [Accessed: 21-Mar-2018].

[42] “Angular Acceleration.” [Online]. Available: https://courses.lumenlearning.com/physics/chapter/10-1-angular-acceleration/. [Accessed: 23-Mar-2018].

[43] “Earthquake and Seismic Education.” [Online]. Available: http://earthalabama.com/education.html. [Accessed: 23-Mar-2018].

[44] P. Rydelek and K. H. Kim, “A Study on Feasibility of Earthquake Early Warning in Korea: Determination of Locations and Magnitudes of Events,” *Geosciences Journal*, Vol. 14, No. 1, pp. 41-47, March 2010. Doi: 10.1007/s12303-010-0005-5

[45] Y. M. Wu and H. Kanamori, “Development of an Earthquake Early Warning System Using Real-Time Strong Motion Signals,” *Sensors (Basel)*, Vol. 8, No. 1, pp. 1-9, January 2008. Doi: 10.3390/s8010001

[46] C. J. Fisher, “Using An Accelerometer for Inclination Sensing,” *Digi-Key* [Online].Available: https://www.digikey.ca/en/articles/techzone/2011/may/using-an-accelerometer-for-inclination-sensing. [Accessed: 27-Nov-2017].

[47] S. Slutz and K. L. Hess, “Increasing the Ability of an Experiment to Measure an Effect,” *Science Buddies* [Online]. Available: https://www.sciencebuddies.org/science-fair-projects/competitions/experimental-design-increasing-signal-to-noise. [Accessed: 20-Nov-2017].

[48] “Why is converting WMV to MP4 so slow?.” [Online]. Available: https://superuser.com/questions/459896/why-is-converting-wmv-to-mp4-so-slow. [Accessed: 29-Nov-2017].

[49] “Sent MP4 files treated as GIF's.” [Online]. Available: https://github.com/telegramdesktop/tdesktop/issues/2053. [Accessed: 29-Nov-2017].

APPENDIX A1: Gantt Chart for Final Year Project 1

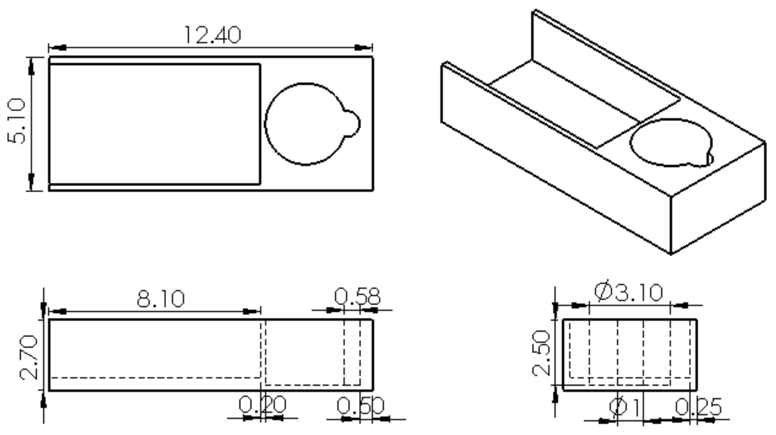
Project Activities of Final Year Project 1	**Week**														
	September			**October**				**November**				**December**			
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Research Journal or Paper Review	■	■	■	■	■	■	■								
Selection of Hardware and Software						■	■								
<u>Experiment Testing</u>								■	■	■					
1. Experiment 1								■	■						
2. Experiment 4									■	■					
3. Experiment 5										■	■				
FYP 1 Presentation												■			
Submission of FYP 1 Final Report													■	■	■

APPENDIX A2: Gantt Chart for Final Year Project 2

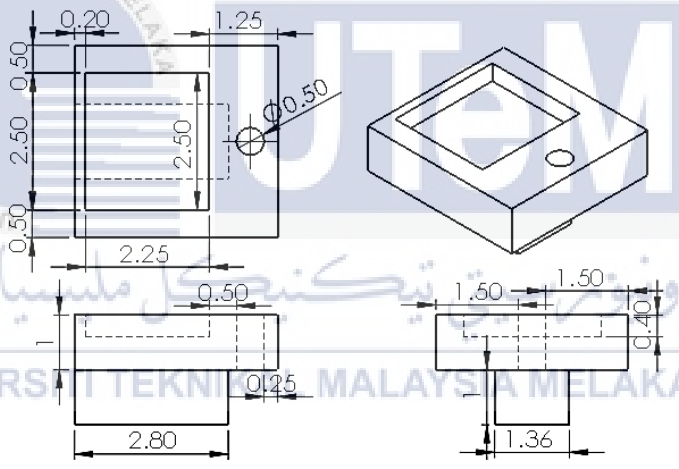
Project Activities of Final Year Project 2	**Week**																
	February				**March**				**April**				**May**				**June**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Experiment Testing	■	■	■	■	■	■	■	■	■								
1. Experiment 1	■	■	■	■													
2. Experiment 2			■	■													
3. Experiment 3				■	■	■											
4. Experiment 6						■	■	■									
5. Experiment 7								■	■								
Data Analysis and Discussion							■	■	■	■							
Report Writing										■	■	■	■	■			
Preparation Slide and Video															■		
FYP 2 Presentation															■		
Submission of FYP 2 Final Report															■	■	■

APPENDIX B: Shaking Table in Third Angle Orthographic Projection

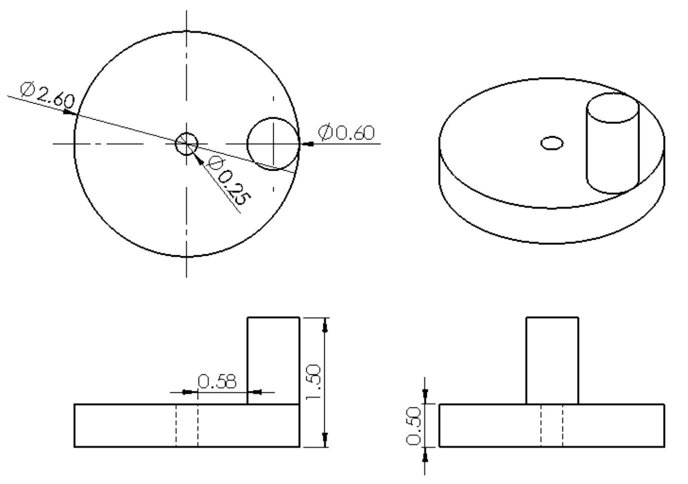
i) Base



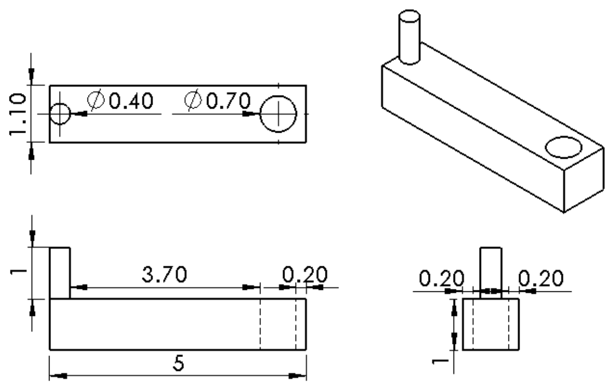
ii) Body



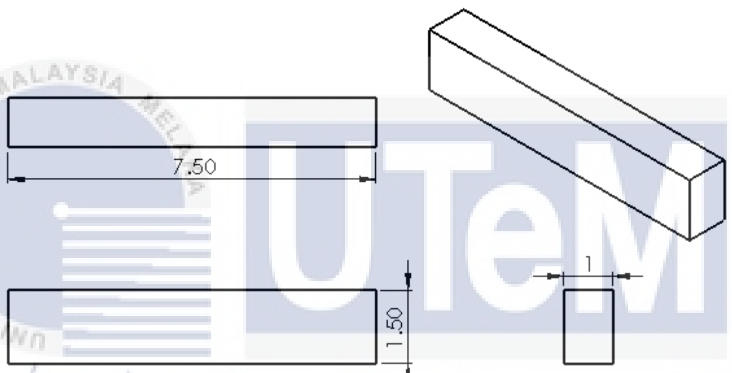
iii) Roller



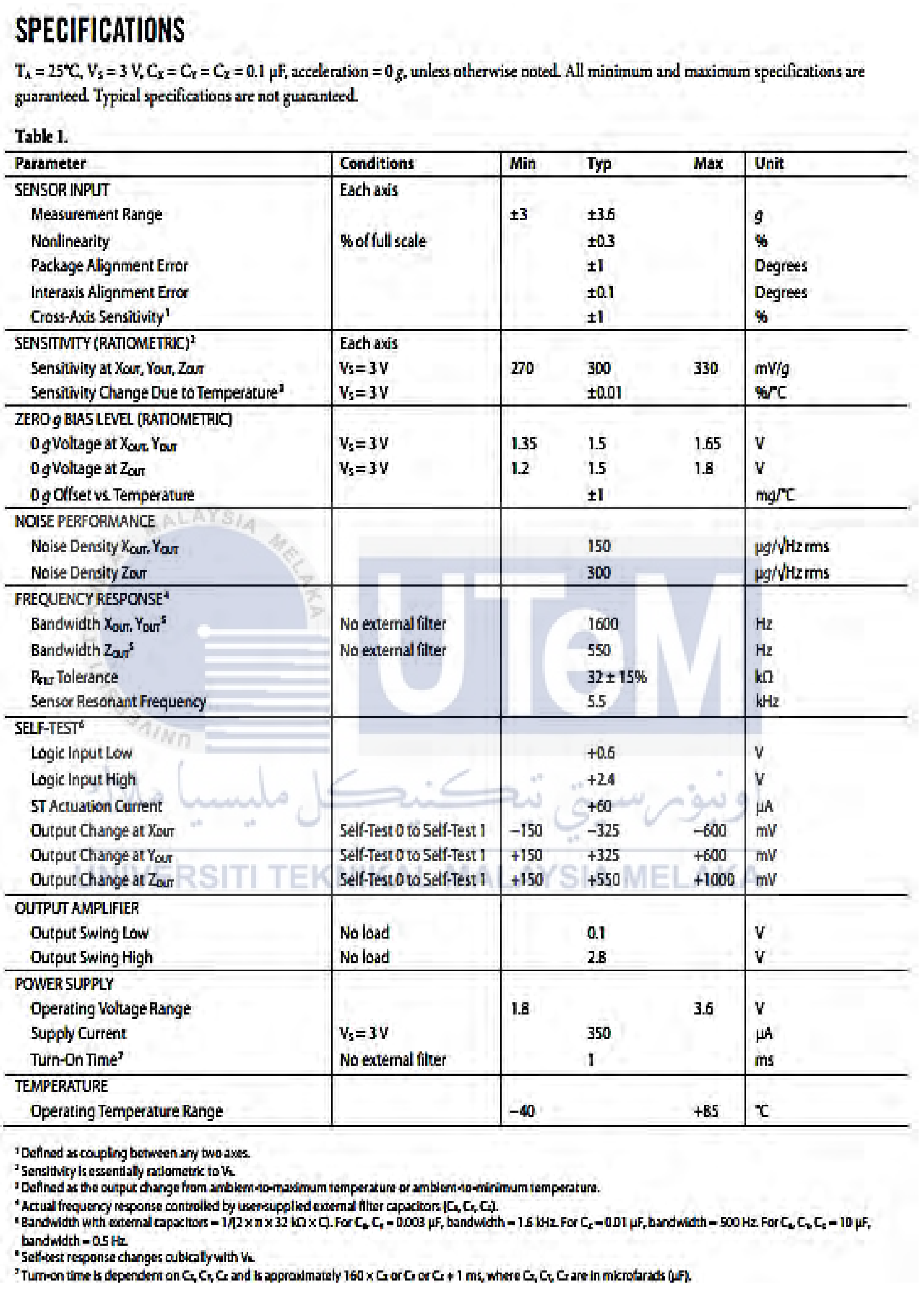
iv) Link



v) Beam



APPENDIX C1: Datasheet of Accelerometer ADXL 335



SPECIFICATIONS

$T_A = 25°C$, $V_S = 3$ V, $C_X = C_Y = C_Z = 0.1$ µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	$V_S = 3$ V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	$V_S = 3$ V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	$V_S = 3$ V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	$V_S = 3$ V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		m*g*/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µ*g*/√Hz rms
Noise Density Z_{OUT}			300		µ*g*/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	$V_S = 3$ V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = $1/(2 \times \pi \times 32\ k\Omega \times C)$. For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.
[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately $160 \times C_X$ or C_Y or $C_Z + 1$ ms, where C_X, C_Y, C_Z are in microfarads (µF).

APPENDIX C2: Datasheet of DC Motor M31E-1 Series

MITSUMI

DC Mini-Motors

M31E-1 Series

DC Mini-Motors

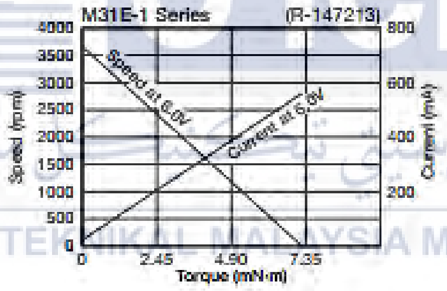
Applications

1. CD/MD Player
2. DVD Player
3. VCR Recorder
4. Power assistance

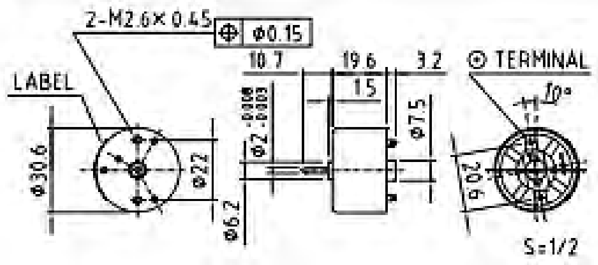
SPECIFICATIONS

Items \ SPEC-NO	R-147213
Rated Voltage (V)	6
Voltage Range (V)	1.5-9
Rated Load (mN·m)	1.37
No Load Speed (rpm)	3700
No Load Current (mA)	23
Starting Torque (mN·m)	7.25
Rotation	CW/CCW

CHARACTERISTICS



DIMENSIONS



Unit : mm

APPENDIX D1: Experiment 1 (Experimenting Communication between Raspberry Pi)

Material and Apparatus

1) 2 sensor nodes
2) Laptop
3) Clay

Procedure

1) 2 sensor nodes are placed on 2 different locations and the shaking tables are fixed using clay.
2) The power supplies on both sensor nodes are turned on.
3) The accelerometers of both sensor nodes are calibrated.
4) Python script of detecting Telegram message is executed in terminal of both sensor nodes.
5) Message “A1” is sent to Telegram channel.
6) The acceleration measured by both accelerometers and its corresponding timestamp are recorded in Table 3.10.
7) A remark on the result is done based on the plotted graph on ThingSpeak channel.
8) Steps 5 to 7 are repeated 9 times.
9) The recorded data in Table 3.10 is analyzed.

APPENDIX D2: Experiment 2 (Shaking Table Calibration)

<u>Material and Apparatus</u>

1) 3 sensor nodes
2) Laptop
3) Clay

<u>Procedure</u>

1) 3 sensor nodes are placed near to each other and the shaking tables are fixed using clay.
2) The power supplies on 3 sensor nodes are turned on.
3) The accelerometers of 3 sensor nodes are calibrated.
4) Python script of detecting Telegram message is executed in terminal of 3 sensor nodes.
5) Message "A1" which represent Range 1 is sent to Telegram channel.
6) The sensor data for each sensor node are recorded in Table 3.11.
7) The gain is calculated using Equation (3.8).
8) Steps 5 to 7 are repeated 9 times.
9) Steps 5 to 8 are repeated by changing the message to "A2" and "A3" which represent Range 2 and Range 3 respectively.
10) The recorded data in Table 3.11 is analyzed.

APPENDIX D3: Experiment 3 (Visualizing Predicted Earthquake Map in ThingSpeak)

Material and Apparatus

1) 3 sensor nodes
2) Laptop
3) Clay

Procedure

1) 3 sensor nodes are placed near to each other and the shaking tables are fixed using clay.
2) The power supplies on 3 sensor nodes are turned on.
3) The accelerometers of 3 sensor nodes are calibrated.
4) Python script of detecting Telegram message is executed in terminal of 3 sensor nodes.
5) Message “A2” which represent Range 2 is sent to Telegram channel.
6) The sensor data for each sensor node are recorded in Table 3.12.
7) The timestamp for the sensor data to exceed the preset threshold value for each sensor node are recorded in Table 3.12.
8) The result is evaluated based on the message or map completed by ThingSpeak to Telegram channel.
9) Steps 5 to 8 are repeated 9 times.
10) Steps 5 to 9 are repeated by changing the message to “A3” and “A1” which represent Range 3 and Range 1 respectively.
11) The recorded data in Table 3.12 is analyzed.

APPENDIX D4: Experiment 4 (Measuring Time Taken of Sending Message to Telegram)

Material and Apparatus

1) Raspberry Pi Zero W 1
2) Laptop

Procedure

1) Python scripts of sending and detecting message by 2 Telegram Bots are executed in 2 terminals of Raspberry Pi 1.
2) A message "/hello" is sent to Telegram Bot 1 through Telegram chat box.
3) The timestamps for both sending message and receiving message actions are recorded in Table 3.13.
4) Time difference between 2 timestamps are calculated and recorded in Table 3.13.
5) Steps 2 to 4 are repeated 9 times.
6) The recorded data in Table 3.13 is analyzed.

APPENDIX D5: Experiment 5 (Comparing Recorded Video Format)

<u>Material and Apparatus</u>

1) Raspberry Pi Zero W 1
2) Laptop
3) Raspberry Pi Camera Module

<u>Procedure</u>

1) Raspberry Pi Camera Module is connected to Raspberry Pi 1.
2) Python scripts of recording and uploading video and also detecting video by 2 Telegram Bots are executed in 2 terminals of Raspberry Pi 1.
3) A message "/avi" is sent to Telegram Bot 1 through Telegram chat box.
4) The timestamps for both recording video and receiving video by Telegram channel are recorded in Table 3.14.
5) Time difference between 2 timestamps are calculated and recorded in Table 3.14.
6) Steps 3 to 5 are repeated 9 times.
7) Steps 3 to 6 are repeated by using messages "/mkv" and "/mp4".
8) The recorded data in Table 3.14 is analyzed.

APPENDIX D6: Experiment 6 (Sending Earthquake Map to Telegram using ThingSpeak)

Material and Apparatus

1) Raspberry Pi Zero W 3
2) Laptop

Procedure

1) The preset sensor data and its corresponding timestamps for each sensor node refer to success case in Experiment 3 are uploaded to ThingSpeak channel.
2) ThingSpeak Visualization program page is opened using Chromium browser.
3) Python script of detecting message by Telegram Bot 3 is executed in terminal of Raspberry Pi 3.
4) A message "reload" is sent to Telegram Bot 3 through Telegram chat box.
5) The timestamps for both sending message and receiving map actions are recorded in Table 3.15.
6) Time difference between 2 timestamps are calculated and recorded in Table 3.15.
7) Steps 3 to 5 are repeated 9 times.
8) The recorded data in Table 3.15 is analyzed.

APPENDIX D7: Experiment 7 (Testing System Efficiency)

Material and Apparatus

1) 3 sensor nodes	3) Raspberry Pi Camera Module
2) Laptop	4) Clay

Procedure

1) 3 sensor nodes are placed near to each other and the shaking tables are fixed using clay.
2) The power supplies on 3 sensor nodes are turned on.
3) The accelerometers of 3 sensor nodes are calibrated.
4) Raspberry Pi Camera Module is connected to Sensor Node 2.
5) The sensor data for each sensor node is fixed to same value within Range 1 obtained through Experiment 2.
6) Python script of detecting Telegram message is executed in terminal of 3 sensor nodes.
7) A message of "A1" is sent to Telegram channel which represent earthquake epicenter point A.
8) The acceleration value of each sensor node is recorded in Table 3.16.
9) 3 timestamps are recorded in Table 3.16 including:
   a) Earliest timestamp when acceleration values of either one or more than one sensor nodes exceed the preset threshold value.
   b) Earliest timestamp when Telegram channel receive early stage alert message.
   c) Timestamp when Telegram channel finished receive final stage alert message.
10) Time difference between 3 timestamps are calculated and recorded in Table 3.16.
11) Steps 7 to 10 are repeated 9 times.
12) The recorded data in Table 3.16 are analyzed.

APPENDIX E: Python Script for Measuring Offset Error

```
import spidev
import time

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

offsetx = 0
averageX = 0

#Function to read SPI data from MCP3008 chip
def ReadChannelX(channel):
        adc = spi.xfer2([1,(8+channel)<<4,0])
        data = ((adc[1]&3) << 8) + adc[2] - offsetx
        return data

#Function to calculate acceleration
def ConvertAcc(data, places):
        Acc = (data*3.3/float(1023)-1.65)/0.33*9.81
        Acc = round(Acc,places)
        return Acc

#Define sensor channels
X_channel=0

while True:
        print "Start calibration!"
        time.sleep(1)
        for i in range(1,11):
                #Read the X sensor data
                X1_level = ReadChannelX(X_channel)
                offsetx = X1_level - 511.5
                averageX = offsetx + averageX
                offsetx=0
                time.sleep(0.5)
        offsetx = averageX/10
        X2_level = ReadChannelX(X_channel)
        Acc = ConvertAcc(X2_level,2)

#Print out results
        print("----------Calibration Result----------")
        print("X-direction:{}({}m/s2)").format(X2_level, Acc)
        print("Offset X:{}").format(offsetx)
        time.sleep(0.1)
        break
```

APPENDIX F1: Python Script for Experimenting Communication between Raspberry Pi

Sensor Node 2 and Sensor Node 3

a) Shaking table script

```
import RPi.GPIO as GPIO
from time import sleep
import telepot
import subprocess
import os
import sys

bot = telepot.Bot('BOT TOKEN')
GPIO.setmode(GPIO.BOARD)
Motor1A = 35
Motor1B = 37
Motor1E = 33
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)
pwm=GPIO.PWM(33,100)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    if command[0] in ('A','B','C','D','E','F'):
        subprocess.Popen(["python","exp1.py"])
    # Only for Sensor Node 2
    elif command[0] in ('1','2','3','X'):
        f=open('thingspeak.txt','a')
        f.write(command)
        f.write("\n")
        f.close()

    #Refer to Table 3.8
    if command[0] in ('A','B','C'):
        t1=0.6
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
    elif command[0] in ('D','E','F'):
        t1=0
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
```

b) Sensor data calculation script

```
import spidev
from time import sleep
import math
import sys
import telepot

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

#Refer to sensor calibration result
offsetx = -10
averageX = 0
bot = telepot.Bot('BOT TOKEN')

#Function to read SPI data from MCP3008 chip
#Channel must be an integer 0-7
def ReadChannelX(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2] - offsetx
    return data

#Function to calculate acceleration
def ConvertAcc(data, places):
    Acc = (data*3.3/float(1023)-1.65)/0.33*9.81
    Acc = round(Acc,places)
    return Acc

#Define sensor channels
X_channel=0

try:
    while True:
        for i in range(1,10):
            X_level = ReadChannelX(X_channel)
            Acc = ConvertAcc(X_level,2)
            Acc = abs(Acc)
            if (Acc >= 0.05):
```

```
def definespeed(command):
    if '1' in command:
        return (35,59,3,0.125)
    elif '2' in command:
        return (35,70,7,0.2)
    elif '3' in command:
        return (43,95,13,0.25)

def motor(s,e,r,t1,t2):
    sleep(t1)
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    for i in range(s,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(s+r,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    GPIO.output(Motor1E,GPIO.LOW)
    pwm.stop()

bot.message_loop(handle)

try:
    while True:
        sleep(0.1)

except KeyboardInterrupt:
    pwm.stop()
     GPIO.cleanup()
     sys.exit()
```

```
                for i in range(1,101):
                    X_level = ReadChannelX(X_channel)
                    Acc = ConvertAcc(X_level,2)
                    Acc = abs(Acc)
                    averageX = Acc + averageX
                    sleep(0.03)
                averageX = averageX/100
                #Refer to sensor node number
                bot.sendMessage(-1001190132929,
                "2A#{}".format(averageX))
                # Only for Sensor Node 2
                sleep(4)
                subprocess.Popen(["python","sendthingspeak.py"
                ])
                break
            sleep(0.01)
        if (Acc < 0.05):
            # Only for Sensor Node 2
            sleep(3)
            subprocess.Popen(["python","sendthingspeak.py"])
            #Refer to sensor node number
            bot.sendMessage(-1001190132929, "XXX2")
        break

except KeyboardInterrupt:
    sys.exit()
```

Sensor Node 2

a) Data upload to ThingSpeak script

```
import httplib, urllib
import sys
from time import sleep
import telepot
n=0
def upload(a2,a3,t):
    params = urllib.urlencode({'field5':a2,'field6':a3,'field7':t, 'key':'2OOORA6XXVHPV93R'})
    headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept": "text/plain"}
    conn = httplib.HTTPConnection("api.thingspeak.com:80")
    try:
        conn.request("POST", "/update", params, headers)
        response = conn.getresponse()
        print (response.status, response.reason)
        data = response.read()
        conn.close()
    except:
        print ("connection failed")
f = open('thingspeak.txt','r')
try:
    while True:
        command = f.readline()
        if command[0] in ('2','3'):
            if 'A' in command:
                if '2' in command[0]:
                    a2=command[3:]
                    n=n+1
                elif '3' in command[0]:
                    a3=command[3:]
                    n=n+1
        elif command[0] in 'X':
            sleep(15)
            bot = telepot.Bot('393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY')
            bot.sendMessage(-1001135228608, "Warning cancel.")
            f = open('thingspeak.txt','w')
            f.write(")
            f.close()
            break
        if (n == 2):
            t=1
            upload(a2,a3,t)
            f = open('thingspeak.txt','w')
            f.write(")
            f.close()
            break
except KeyboardInterrupt:
    sys.exit()
```

APPENDIX F2: Python Script for Shaking Table Calibration

All Sensor Nodes

a) Shaking table script

```
import RPi.GPIO as GPIO
from time import sleep
import telepot
import subprocess
import os
import sys

bot = telepot.Bot('BOT TOKEN')
GPIO.setmode(GPIO.BOARD)
Motor1A = 35
Motor1B = 37
Motor1E = 33
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)
pwm=GPIO.PWM(33,100)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    if command[0] in ('A','B','C','D','E','F'):
        subprocess.Popen(["python","exp1.py"])

    #Refer to Table 3.8
    if command[0] in ('A','B','C'):
        t1=0.6
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
    elif command[0] in ('D','E','F'):
        t1=0
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)

def definespeed(command):
    if '1' in command:
        return (35,59,3,0.125)
    elif '2' in command:
        return (35,70,7,0.2)
    elif '3' in command:
        return (43,95,13,0.25)
```

b) Sensor data calculation script

```
import spidev
from time import sleep
import math
import sys
import telepot

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

#Refer to sensor calibration result
offsetx = -10
averageX = 0
bot = telepot.Bot('BOT TOKEN')

#Function to read SPI data from MCP3008 chip
#Channel must be an integer 0-7
def ReadChannelX(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2] - offsetx
    return data

#Function to calculate acceleration
def ConvertAcc(data, places):
    Acc = (data*3.3/float(1023)-1.65)/0.33*9.81
    Acc = round(Acc,places)
    return Acc

#Define sensor channels
X_channel=0

try:
    while True:
        for i in range(1,10):
            X_level = ReadChannelX(X_channel)
            Acc = ConvertAcc(X_level,2)
            Acc = abs(Acc)
            if (Acc >= 0.05):
                for i in range(1,101):
                    X_level = ReadChannelX(X_channel)
```

```
def motor(s,e,r,t1,t2):
    sleep(t1)
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    for i in range(s,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(s+r,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    GPIO.output(Motor1E,GPIO.LOW)
    pwm.stop()

bot.message_loop(handle)

try:
    while True:
        sleep(0.1)

except KeyboardInterrupt:
    pwm.stop()
     GPIO.cleanup()
     sys.exit()
```

```
                    Acc = ConvertAcc(X_level,2)
                    Acc = abs(Acc)
                    averageX = Acc + averageX
                    sleep(0.03)
                averageX = averageX/100
                #Refer to sensor node number
                bot.sendMessage(-1001190132929,
                "1A#{}".format(averageX))
                break
            sleep(0.01)
        if (Acc < 0.05):
            #Refer to sensor node number
            bot.sendMessage(-1001190132929, "XXX1")
        break

except KeyboardInterrupt:
      sys.exit()
```

APPENDIX F3: Python Script for Visualizing Predicted Earthquake Map in ThingSpeak

All Sensor Nodes

a) Shaking table script

```
import RPi.GPIO as GPIO
from time import sleep
import telepot
import subprocess
import os
import sys
# Only for Sensor Node 3
import urllib

bot = telepot.Bot('BOT TOKEN')
GPIO.setmode(GPIO.BOARD)
Motor1A = 35
Motor1B = 37
Motor1E = 33
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)
pwm=GPIO.PWM(33,100)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    if command[0] in ('A','B','C','D','E','F'):
        f=open('command.txt','w')
        f.write(command)
        f.close()
        subprocess.Popen(["python","exp1.py"])
    # Only for Sensor Node 1
    elif command[0] in ('1','2','3','X'):
        f=open('thingspeak.txt','a')
        f.write(command)
        f.write("\n")
        f.close()
    # Only for Sensor Node 3
    elif command =='done':
        sleep(3)
```

b) Sensor data calculation script

```
import spidev
from time import sleep
import math
import sys
import telepot

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

#Refer to sensor calibration result
offsetx = -10
averageX = 0
bot = telepot.Bot('BOT TOKEN')

#Function to read SPI data from MCP3008 chip
#Channel must be an integer 0-7
def ReadChannelX(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2] - offsetx
    return data

#Function to calculate acceleration
def ConvertAcc(data, places):
    Acc = (data*3.3/float(1023)-1.65)/0.33*9.81
    Acc = round(Acc,places)
    return Acc

#Define sensor channels
X_channel=0

try:
    while True:
        for i in range(1,10):
            X_level = ReadChannelX(X_channel)
            Acc = ConvertAcc(X_level,2)
            Acc = abs(Acc)
            if (Acc >= 0.05):
                subprocess.Popen(["python","sendmsg.py"])
        urllib.urlretrieve("https://s3.amazonaws.com/images.thi
        ngspeak.com/plugins/224752/x577c3FtngVdv87dFkgJ
        hg.png", "1.png")
        bot.sendPhoto(chat_id=-1001135228608,photo=open('/
        home/pi/1.png'))
    elif command=='reload':
        os.system('/home/pi/refresh.sh')

    #Refer to Table 3.8
    if command[0] in ('A','B','C'):
        t1=0.6
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
    elif command[0] in ('D','E','F'):
        t1=0
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)

def definespeed(command):
    if '1' in command:
        return (35,59,3,0.125)
    elif '2' in command:
        return (35,70,7,0.2)
    elif '3' in command:
        return (43,95,13,0.25)

def motor(s,e,r,t1,t2):
    sleep(t1)
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    for i in range(s,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(s+r,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    GPIO.output(Motor1E,GPIO.LOW)
```

```
                for i in range(1,101):
                    Acc = ConvertAcc(X_level,2)
                    Acc = abs(Acc)
                    averageX = Acc + averageX
                    sleep(0.03)
                averageX = averageX/100
                subprocess.Popen(["python","sub.py",
                str(averageX)])
                # Only for Sensor Node 1
                sleep(4)
                subprocess.Popen(["python","sendthingspeak.py"])
                break
            sleep(0.01)
        if (Acc < 0.05):
            # Only for Sensor Node 1
            sleep(3)
            subprocess.Popen(["python","sendthingspeak.py"])
            #Refer to sensor node number
            bot.sendMessage(-1001190132929, "XXX1")
            break
except KeyboardInterrupt:
    sys.exit()
```

c) Gain multiplication on sensor data script

```
from time import sleep
import sys
import ast
import telepot
bot = telepot.Bot('BOT TOKEN')
while True:
    f = open('command.txt','r')
    command = f.read()
    f.close()
     sleep(0.1)
     #Refer to Table 4.5 for each sensor node
    if '1' in command:
        g=0.092
    elif '2' in command:
        g=0.121
    elif '3' in command:
        g=0.164
    Acc = sys.argv[1]
    Acc = ast.literal_eval(str(Acc))
    Acc = g * Acc
```

```
    Acc = round(Acc,3)
    #Refer to sensor node
    number
    bot.sendMessage(-10011
    90132929,
    "1A#{}".format(Acc))
    sleep(0.1)
    break
```

```
        GPIO.output(Motor1E,GPIO.LOW)
        pwm.stop()

bot.message_loop(handle)

try:
        while True:
                sleep(0.1)

except KeyboardInterrupt:
        pwm.stop()
          GPIO.cleanup()
          sys.exit()
```

d) Sensor exceeds preset threshold value timestamp script

```
from time import sleep
import datetime
import telepot

while True:
        bot = telepot.Bot('BOT TOKEN')
        t=datetime.datetime.now().strftime("%H:%M:%S.%f")[:-3]
        #Refer to sensor node number
        bot.sendMessage(-1001190132929, "1T#{}".format(t))
        sleep(0.1)
        break
```

Sensor Node 3

a) ThingSpeak Visualization program page refresh shell script

```
export DISPLAY=:0
XAUTHORITY=/home/pi/.Xauthority
xdotool search --onlyvisible --class chromium windowactivate
xdotool key "ctrl+R"
```

Sensor Node 1

a) Data upload to ThingSpeak script

```
import httplib, urllib
import sys
from time import sleep
import telepot

n=0

def upload(t1,t2,t3,a1,a2,a3,t):
        params = urllib.urlencode({'field1':t1,
        'field2':t2,'field3':t3,'field4':a1,'field5':a2,'field6':a3,
        'field7':t, 'key':'2OOORA6XXVHPV93R'})
        headers = {"Content-typZZe":
        "application/x-www-form-urlencoded","Accept": "text/plain"}
        conn = httplib.HTTPConnection("api.thingspeak.com:80")
        try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                print (response.status, response.reason)
                data = response.read()
                conn.close()
        except:
                print ("connection failed")

f = open('thingspeak.txt','r')

try:
        while True:
                command = f.readline()
                if command[0] in ('1','2','3'):
                        if 'T' in command:
                                if '1' in command[0]:
                                        t1=command[3:]
                                        n=n+1
                                elif '2' in command[0]:
                                        t2=command[3:]
                                        n=n+1
                                elif '3' in command[0]:
                                        t3=command[3:]
                                        n=n+1
                        if 'A' in command:
                                if '1' in command[0]:
                                        a1=command[3:]
```

```
                    n=n+1
                elif '2' in command[0]:
                    a2=command[3:]
                    n=n+1
                elif '3' in command[0]:
                    a3=command[3:]
                    n=n+1
        elif command[0] in 'X':
            sleep(15)
            bot =
            telepot.Bot('393328733:AAGbqKbVfYYcW4kOhRiUg
            FxsWKulH_EkEfY')
            bot.sendMessage(-1001135228608, "Warning cancel.")
            f = open('thingspeak.txt','w')
            f.write('')
            f.close()
            break
        if (n == 6):
            t=1
            upload(t1,t2,t3,a1,a2,a3,t)
            f = open('thingspeak.txt','w')
            f.write('')
            f.close()
            break
        sleep(0.1)

    except KeyboardInterrupt:
        sys.exit()
```

APPENDIX F4: Python Script for Measuring Time Taken of Sending Message to Telegram

a) Python script for Telegram Bot 1

```
import time
import datetime
import telepot

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print 'Got command: %s' % command
    if command == '/hello':
        # Print timestamp before sending alert message
        print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        bot.sendMessage(-1001135228608, "Node 1 detected earthquake: 3.7 magnitude.")
    else:
        bot.sendMessage(chat_id, "Invalid command.")
# Telegram Bot 1 token
bot = telepot.Bot('393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY')
bot.message_loop(handle)
bot.sendMessage(470295792, "Pls enter command!")
print 'TimeS is listening ...'
while True:
      time.sleep(0.1)
```

b) Python script for Telegram Bot 2

```
import time
import datetime
import telepot

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    print 'Got command: %s' % command
      if command == 'Node 1 detected earthquake: 3.7 magnitude.':
            # Print timestamp when alert message is detected by Telegram Bot 2
            print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
            print('Done!')
# Telegram Bot 2 token
bot = telepot.Bot('444050918:AAGEWUJxy_0deByevQ4t-46UxkFBJ-6f0Ys')
bot.message_loop(handle)
print 'TimeR is listening ...'
while True:
      time.sleep(0.1)
```

APPENDIX F5: Python Script for Comparing Recorded Video Format

a) Python script for Telegram Bot 1

```
import time
import datetime
import telepot
import os

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print 'Got command: %s' % command
      if command == '/avi':
        print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        os.system('raspivid -o video0.avi -t 5000 -w 320 -h 240 -vf')
        bot.sendDocument(-1001135228608, document =
        open('/home/pi/video0.avi'), caption = 'AVI')
    elif command == '/mkv':
        print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        os.system('raspivid -o video0.h264 -t 5000 -w 320 -h 240 -vf')
        os.system('ffmpeg -i video0.h264 -vcodec copy video0.mkv')
        bot.sendDocument(-1001135228608, document =
        open('/home/pi/video0.mkv'), caption = 'MKV')
    elif command == '/mp4':
        print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        os.system('raspivid -o video0.h264 -t 5000 -w 320 -h 240 -vf')
        os.system('ffmpeg -i video0.h264 -vcodec copy video0.mp4')
        bot.sendDocument(-1001135228608, document =
        open('/home/pi/video0.mp4'), caption = 'MP4')
    else:
        bot.sendMessage(chat_id, "Invalid command.")

# Telegram Bot 1 token
bot =
telepot.Bot('393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY')
bot.message_loop(handle)
bot.sendMessage(470295792, "Pls enter command!")
print 'TimeS is listening ...'

while True:
      time.sleep(0.1)
```

b) Python script for Telegram Bot 2

```
import time
import datetime
import telepot

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['caption']

    print 'Got command: %s' % command
    if command == 'AVI':
        print(datetime.datetime.now().strftime("
        %Y-%m-%d %H:%M:%S"))
        print('AVI Done!')
    elif command == 'MKV':
        print(datetime.datetime.now().strftime("
        %Y-%m-%d %H:%M:%S"))
        print('MKV Done!')
    elif command == 'MP4':
        print(datetime.datetime.now().strftime("
        %Y-%m-%d %H:%M:%S"))
        print('MP4 Done!')

# Telegram Bot 2 token
bot =
telepot.Bot('444050918:AAGEWUJxy_0deByevQ
4t-46UxkFBJ-6f0Ys')
bot.message_loop(handle)
print 'TimeR is listening ...'

while True:
    time.sleep(0.1)
```

APPENDIX F6: Python Script for Sending Earthquake Map to Telegram using ThingSpeak

Sensor Node 3

a) Shaking table script

```
import RPi.GPIO as GPIO
from time import sleep
import telepot
import subprocess
import os
import sys
import urllib

bot =
telepot.Bot('574331181:AAHcOPPzSKzo2L1mqciKt2Mi5_Kk
qDIW4Wg')
GPIO.setmode(GPIO.BOARD)
Motor1A = 35
Motor1B = 37
Motor1E = 33
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)
pwm=GPIO.PWM(33,100)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    if command[0] in ('A','B','C','D','E','F'):
        f=open('command.txt','w')
        f.write(command)
        f.close()
        subprocess.Popen(["python","exp1.py"])
    elif command =='done':
        sleep(3)
        urllib.urlretrieve("https://s3.amazonaws.com/imag
        es.thingspeak.com/plugins/224752/x577c3FtngVd
        v87dFkgJhg.png", "1.png")
        bot.sendPhoto(chat_id=-1001135228608,photo=op
        en('/home/pi/1.png'))
        os.remove("/home/pi/1.png")
    elif command=='reload':
        os.system('/home/pi/refresh.sh')
```

b) ThingSpeak Visualization program page refresh shell script

```
export DISPLAY=:0
XAUTHORITY=/home/pi/.Xauthority
xdotool search --onlyvisible --class chromium windowactivate
xdotool key "ctrl+R"
```

```
    if command[0] in ('A','B','C'):
        t1=0.6
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
    elif command[0] in ('D','E','F'):
        t1=0
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
def definespeed(command):
    if '1' in command:
        return (35,59,3,0.125)
    elif '2' in command:
        return (35,70,7,0.2)
    elif '3' in command:
        return (43,95,13,0.25)
def motor(s,e,r,t1,t2):
    sleep(t1)
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    for i in range(s,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(s+r,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    GPIO.output(Motor1E,GPIO.LOW)
    pwm.stop()

bot.message_loop(handle)

try:
    while True:
        sleep(0.1)
except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
    sys.exit()
```

APPENDIX F7: Python Script for Testing System Efficiency

All Sensor Nodes

a) Shaking table script

```
import RPi.GPIO as GPIO
from time import sleep
import telepot
import subprocess
import os
import sys
# Only for Sensor Node 3
import urllib

bot = telepot.Bot('BOT TOKEN')
GPIO.setmode(GPIO.BOARD)
Motor1A = 35
Motor1B = 37
Motor1E = 33
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1E,GPIO.OUT)
pwm=GPIO.PWM(33,100)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
    if command[0] in ('A','B','C','D','E','F'):
        f=open('command.txt','w')
        f.write(command)
        f.close()
        subprocess.Popen(["python","exp1.py"])
    # Only for Sensor Node 1
    elif command[0] in ('1','2','3','X'):
        f=open('thingspeak.txt','a')
        f.write(command)
        f.write("\n")
        f.close()
    # Only for Sensor Node 3
    elif command =='done':
        sleep(3)
```

b) Sensor data calculation script

```
import spidev
from time import sleep
import math
import sys
import telepot

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

#Refer to sensor calibration result
offsetx = -10
averageX = 0
bot = telepot.Bot('BOT TOKEN')

#Function to read SPI data from MCP3008 chip
#Channel must be an integer 0-7
def ReadChannelX(channel):
    adc = spi.xfer2([1,(8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2] - offsetx
    return data

#Function to calculate acceleration
def ConvertAcc(data, places):
    Acc = (data*3.3/float(1023)-1.65)/0.33*9.81
    Acc = round(Acc,places)
    return Acc

#Define sensor channels
X_channel=0

try:
    while True:
        for i in range(1,10):
            X_level = ReadChannelX(X_channel)
            Acc = ConvertAcc(X_level,2)
            Acc = abs(Acc)
            if (Acc >= 0.05):
                subprocess.Popen(["python","sendmsg.py"])
                subprocess.Popen(["python","sendvideo.py"])
        urllib.urlretrieve("https://s3.amazonaws.com/images.thingspeak.com/plugins/224752/x577c3FtngVdv87dFkgJhg.png", "1.png")
        bot.sendPhoto(chat_id=-1001135228608,photo=open('/home/pi/1.png'))
    elif command=='reload':
        os.system('/home/pi/refresh.sh')

    #Refer to Table 3.8
    if command[0] in ('A','B','C'):
        t1=0.6
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)
    elif command[0] in ('D','E','F'):
        t1=0
        s,e,r,t2 = definespeed(command)
        motor(s,e,r,t1,t2)

def definespeed(command):
    if '1' in command:
        return (35,59,3,0.125)
    elif '2' in command:
        return (35,70,7,0.2)
    elif '3' in command:
        return (43,95,13,0.25)

def motor(s,e,r,t1,t2):
    sleep(t1)
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1E,GPIO.HIGH)
    for i in range(s,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(s+r,e+1,r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    for i in range(e-r,s-1,-r):
        pwm.ChangeDutyCycle(i)
        sleep(t2)
    GPIO.output(Motor1E,GPIO.LOW)
```

```
                for i in range(1,101):
                    Acc = ConvertAcc(X_level,2)
                    Acc = abs(Acc)
                    averageX = Acc + averageX
                    sleep(0.03)
                averageX = averageX/100
                subprocess.Popen(["python","sub.py", str(averageX)])
                # Only for Sensor Node 1
                sleep(4)
                subprocess.Popen(["python","sendthingspeak.py"])
                break
            sleep(0.01)
        if (Acc < 0.05):
            # Only for Sensor Node 1
            sleep(3)
            subprocess.Popen(["python","sendthingspeak.py"])
            #Refer to sensor node number
            bot.sendMessage(-1001190132929, "XXX1")
        break
except KeyboardInterrupt:
    sys.exit()
```

c) **Gain multiplication on sensor data script**

```
from time import sleep
import sys
import ast
import telepot
bot = telepot.Bot('BOT TOKEN')
while True:
    f = open('command.txt','r')
    command = f.read()
    f.close()
    sleep(0.1)
    #Refer to Table 4.5 for each sensor node
    if '1' in command:
        g=0.092
    elif '2' in command:
        g=0.121
    elif '3' in command:
        g=0.164
    Acc = sys.argv[1]
    Acc = ast.literal_eval(str(Acc))
    Acc = g * Acc
    Acc = round(Acc,3)
    #Refer to first trial acceleration data for each sensor node on Table 4.8
    Acc = 1.055
    #Refer to sensor node number
    bot.sendMessage(-1001190132929, "1A#{}".format(Acc))
    sleep(0.1)
    break
    GPIO.output(Motor1E,GPIO.LOW)

    pwm.stop()

bot.message_loop(handle)

try:
    while True:
        sleep(0.1)

except KeyboardInterrupt:
    pwm.stop()
     GPIO.cleanup()
     sys.exit()
```

d) Sensor exceeds preset threshold value timestamp script

```
from time import sleep
import datetime
import telepot

while True:
    bot = telepot.Bot('BOT TOKEN')
    t=datetime.datetime.now().strftime("%H:%M:%S.%f")[:-3]
    #Refer to first trial timstamp for each sensor node on Table
    4.8
    t=" 18:22:59.521"
    #Refer to sensor node number
    bot.sendMessage(-1001190132929, "1T#{}".format(t))
    sleep(0.1)
    break
```

Sensor Node 1

a) Data upload to ThingSpeak script

```
import httplib, urllib
import sys
from time import sleep
import telepot

n=0

def upload(t1,t2,t3,a1,a2,a3,t):
    params = urllib.urlencode({'field1':t1,
    'field2':t2,'field3':t3,'field4':a1,'field5':a2,'field6':a3,
    'field7':t, 'key':'2OOORA6XXVHPV93R'})
    headers = {"Content-typZZe":
    "application/x-www-form-urlencoded","Accept": "text/plain"}
    conn = httplib.HTTPConnection("api.thingspeak.com:80")
    try:
        conn.request("POST", "/update", params, headers)
        response = conn.getresponse()
        print (response.status, response.reason)
        data = response.read()
        conn.close()
    except:
        print ("connection failed")

f = open('thingspeak.txt','r')

try:
    while True:
        command = f.readline()
        if command[0] in ('1','2','3'):
            if 'T' in command:
                if '1' in command[0]:
                    t1=command[3:]
                    n=n+1
                elif '2' in command[0]:
                    t2=command[3:]
                    n=n+1
                elif '3' in command[0]:
                    t3=command[3:]
                    n=n+1
            if 'A' in command:
                if '1' in command[0]:
                    a1=command[3:]
```

e) Footage record script

```
from time import sleep
import telepot
import os
import datetime

while True:
    bot = telepot.Bot('BOT TOKEN')
    # Only for Sensor Node 2
    os.system('raspivid -o video0.avi -t 5000 -w 320 -h 240 -vf')
    # Only for Sensor Node 1 and 3
    sleep(8.5)
    #Refer to sensor node number
    bot.sendDocument(-1001135228608, document =
    open('/home/pi/video0.avi'), caption = 'Video Node 1')
    # Only for Sensor Node 2
    os.remove("/home/pi/video0.avi")
    sleep(0.1)
    break
```

Sensor Node 3

a) ThingSpeak Visualization program page refresh shell script

```
export DISPLAY=:0
XAUTHORITY=/home/pi/.Xauthority
xdotool search --onlyvisible --class chromium windowactivate
xdotool key "ctrl+R"
```

```
                n=n+1
            elif '2' in command[0]:
                a2=command[3:]
                n=n+1
            elif '3' in command[0]:
                a3=command[3:]
                n=n+1
        elif command[0] in 'X':
            sleep(15)
            bot =
            telepot.Bot('393328733:AAGbqKbVfYYcW4kOhRiUg
            FxsWKulH_EkEfY')
            bot.sendMessage(-1001135228608, "Warning cancel.")
            f = open('thingspeak.txt','w')
            f.write('')
            f.close()
            break
        if (n == 6):
            t=1
            upload(t1,t2,t3,a1,a2,a3,t)
            f = open('thingspeak.txt','w')
            f.write('')
            f.close()
            break
        sleep(0.1)
    except KeyboardInterrupt:
        sys.exit()
```

APPENDIX G1: ThingSpeak MATLAB Analysis code

```
% Channel ID to read from
readChannelID = 480136;

% Channel Read API Key
readAPIKey = 'CMZPDLSUOP26GBQL';

% Channel ID to write
writeChannelID = 480139;

% Write API Key
writeAPIKey = 'EE0Z5DXZRXMU5OJ2';

a1 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',4)
a2 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',5)
a3 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',6)
q=0;
n=0;
m=0;
l=0;

if   (a1 >= 0.8 && a1 <= 1.231)
       n=1
elseif   (a1 >= 1.37    && a1 <= 1.626)
       n=3
elseif (a1 >= 1.973    && a1 <= 2.448)
       n=6
else
       n=20
end
if   (a2 >= 0.8 && a2 <= 1.231)
       m=1
elseif   (a2 >= 1.37    && a2 <= 1.626)
       m=3
elseif (a2 >= 1.973    && a2 <= 2.448)
       m=6
else
       m=20
end
if   (a3 >= 0.8 && a3 <= 1.231)
       l=1
elseif   (a3 >= 1.37    && a3 <= 1.626)
       l=3
elseif (a3 >= 1.973    && a3 <= 2.448)
```

```
    l=6
else
    l=20
end

t=n+m+l

if   (t==3)
    Magnitude=5.8
    q=1;
elseif   (t==9)
    Magnitude=6.4
    q=1;
elseif   (t==18)
    Magnitude=7
    q=1;
else
    q=0;
end

if (q==1)
    t1 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',1,'OutputFormat','timetable')
    t2 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',2,'OutputFormat','timetable')
    t3 = thingSpeakRead(readChannelID,'ReadKey',readAPIKey,'Fields',3,'OutputFormat','timetable')
    t1 = t1.TimestampStation1{1}
    t4=datestr(addtodate(datenum(t1,'HH:MM:SS.FFF'),-1,'second'),'HH:MM:SS.FFF')
    t5 = t2.TimestampStation2{1}
    t6 = t3.TimestampStation3{1}
    d1= etime(datevec(t4),datevec(t5))
    d2= etime(datevec(t5),datevec(t6))
    d3= etime(datevec(t4),datevec(t6))
    if (d1 < -0.6) && ((-0.6 < d2 && d2 <= 0 ) || (d2 >= 0 && d2 < 0.6))
        epicenter = 1
        T1=1
    elseif ((-0.6 < d1 && d1 <= 0 ) || (d1 >= 0 && d1 < 0.6)) && (d2 < -0.6)
        epicenter = 2
        if (d1 >= 0)
        T1=2
        elseif (d1 < 0)
        T1=1
        end
    elseif (d1 > 0.6) && (d2 < -0.6)
        epicenter = 3
        T1=2
    elseif (d1 > 0.6) && ((-0.6 < d2 && d2 <= 0 ) || (d2 >= 0 && d2 < 0.6))
```

```
            epicenter = 4
            if (d2 >= 0)
            T1=3
            elseif (d2 < 0)
            T1=2
            end
        elseif ((-0.6 < d1 && d1 <= 0 ) || (d1 >= 0 && d1 < 0.6)) && (d2 > 0.6)
            epicenter = 5
            T1=3
        elseif (d1 < -0.6) && (d2 > 0.6)
            epicenter = 6
            if (d3 >= 0)
            T1=3
            elseif (d3 < 0)
            T1=1
            end
        else
            epicenter=0;
        end
    else
        epicenter=0;
    end

    if (q~=0 && epicenter~=0)
        thingSpeakWrite(writeChannelID,[epicenter,T1,Magnitude],'WriteKey',writeAPIKey);
        telURL='https://api.telegram.org/bot393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY/sendmes
        sage?chat_id=-1001190132929&text=reload';
        webwrite(telURL,20)
    else
         telURL='https://api.telegram.org/bot393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY/sendme
         ssage?chat_id=-1001190132929&text=failed to analyze data!';
          webwrite(telURL,20)
    end
```

APPENDIX G2: ThingSpeak MATLAB Visualization code

```
a1 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',4)
a2 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',5)
a3 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',6)
epicenter = thingSpeakRead(480139,'ReadKey','PAMXKKQA6I5MRKAK','Fields',1)
T1 = thingSpeakRead(480139,'ReadKey','PAMXKKQA6I5MRKAK','Fields',2)
Magnitude = thingSpeakRead(480139,'ReadKey','PAMXKKQA6I5MRKAK','Fields',3)

d1= 10^(-0.625*log10(a1/(13*exp(0.67*Magnitude))))-25;
d1 = round(d1,2)
d2= 10^(-0.625*log10(a2/(13*exp(0.67*Magnitude))))-25;
d2 = round(d2,2);
d3= 10^(-0.625*log10(a3/(13*exp(0.67*Magnitude))))-25;
d3 = round(d3,2);
r1=km2deg(d1);
r2=km2deg(d2);
r3=km2deg(d3);
s1=[2.329661, 102.279777];
s2=[2.284655, 102.280450];
s3=[2.307752, 102.319032];
ptX=[10.0156,115.8368];

if (epicenter==1)
    epi=[2.5225,102.1640]
    latepi=[2.1, 2.52, 2.72, 1.88];
    lonepi=[102, 102.5,101.88,102.72];
elseif (epicenter==2)
    epi=[2.3038,102.0552]
    latepi=[2.1, 2.52, 2.72, 1.89];
    lonepi=[102, 102.5,101.89,102.71];
elseif (epicenter==3)
    epi=[2.088406, 102.170699]
    latepi=[2.1, 2.52, 2.72, 1.89];
    lonepi=[102, 102.5,101.88,102.71];
elseif (epicenter==4)
    epi=[2.1034,102.4153]
    latepi=[2.1, 2.52, 2.71, 1.89];
    lonepi=[102, 102.5,101.89,102.71];
elseif (epicenter==5)
    epi=[2.3111,102.5441]
    latepi=[2.1, 2.52, 2.72, 1.89];
    lonepi=[102, 102.5,101.89,102.71];
elseif (epicenter==6)
    epi=[2.5150,102.4091]
    latepi=[2.1, 2.52, 2.72, 1.89];
    lonepi=[102, 102.5,101.89,102.71];
end

if (Magnitude==5.8)
    epir=30.83/2
    epird=km2deg(epir);
    M='5.3-5.8'
elseif (Magnitude==6.4)
    epir=31.29/2
    epird=km2deg(epir);
    M='5.9-6.4'
elseif (Magnitude==7)
    epir=31.75/2
    epird=km2deg(epir);
    M='6.5-7.0'
end

if (T1==1)
    t1 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',1,'OutputFormat','timetable');
    t1 = t1.TimestampStation1{1}
    D = distance(epi,s1);
elseif (T1==2)
    t1 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',2,'OutputFormat','timetable');
    t1 = t1.TimestampStation2{1}
    D = distance(epi,s2);
elseif (T1==3)
    t1 = thingSpeakRead(480136,'ReadKey','CMZPDLSUOP26GBQL','Fields',3,'OutputFormat','timetable');
    t1 = t1.TimestampStation3{1}
    D = distance(epi,s3);
end

D = deg2km(D);
td = distance(epi,ptX);
td = deg2km(td);
d = (td-D)/1000;
epir = epir/1000;
t3 = ((0.0002*d)^5 - (0.0042*d)^4 + (0.0425*d)^3 - (0.2817*d)^2 + (2.4332*d) + 0.0174)*60;
t2 = ((0.0002*epir)^5 - (0.0042*epir)^4 + (0.0425*epir)^3 - (0.2817*epir)^2 + (2.4332*epir) + 0.0174)*60;
t3 = round(t3,0);
t2 = round(t2,0);
t=datestr(addtodate(datenum(t1,'HH:MM:SS'),t3,'second'),'HH:MM:SS')
t=datestr(addtodate(datenum(t,'HH:MM:SS'),-t2,'second'),'HH:MM:SS')

figure
```

```
subplot(2,1,1)
lat = [10.4, 4.589377, 1.7, 2.5];
lon = [116.2, 102, 101.6, 107.4];
[latlim,lonlim] = geoquadline(lat,lon);
buf = 0.01;
ax = worldmap(latlim,lonlim);
[latlim,lonlim] = bufgeoquad(latlim,lonlim,buf,buf);
oceanColor = [.5 .7 .9];
setm(ax, 'FFaceColor',oceanColor);
load coastlines;
hold on;
geoshow(coastlat,coastlon,'FaceColor','white','DisplayType','polygon');
[gclat,gclong] = track2('gc',epi(1),epi(2),ptX(1),ptX(2));
i3=plotm(gclat,gclong,'b--');
sca = scircle1(epi(1), epi(2), epird);
i1=patchm(sca(:,1), sca(:,2),'r');
plotm(sca(:,1), sca(:,2),'r');
i2=plotm(ptX ,'k*','MarkerFaceColor','k');
title('Overall Map');
legend([i1 i2 i3],{'Epicenter Point' 'Point X' 'Earthquake Path'},'Location','northeastoutside')
annotation('textbox', [0.62, 0.78, 0, 0], 'string', 'Information:');
annotation('textbox', [0.62, 0.67, 0.21, 0.07], 'edgecolor', 'w','string', ['Est. Time: ' t,'      Est. Magnitude: ' M,'
Latitude: ' num2str(epi(1)),'      Longitude: ' num2str(epi(2))]);

subplot(2,1,2)
[latlim,lonlim] = geoquadline(latepi,lonepi);
buf = 0.01;
ax = worldmap(latlim,lonlim);
[latlim,lonlim] = bufgeoquad(latlim,lonlim,buf,buf);
oceanColor = [.5 .7 .9];
setm(ax, 'FFaceColor',oceanColor);
load coastlines;
hold on;
geoshow(coastlat,coastlon,'FaceColor','white','DisplayType','polygon');
sce = scircle1(epi(1), epi(2), epird);
i7=patchm(sce(:,1), sce(:,2),'r');
plotm(sce(:,1), sce(:,2),'r');
sc1 = scircle1(s1(1), s1(2), r1);
i8=plotm(sc1(:,1), sc1(:,2),'k','LineWidth', 1);
i4=plotm(s1,'k+','MarkerFaceColor','k');
sc2 = scircle1(s2(1), s2(2), r2);
i9=plotm(sc2(:,1), sc2(:,2),'m','LineWidth', 1);
i5=plotm(s2,'m+','MarkerFaceColor','m');
sc3 = scircle1(s3(1), s3(2), r3);
i10=plotm(sc3(:,1), sc3(:,2),'b','LineWidth', 1);
```

```
i6=plotm(s3,'b+','MarkerFaceColor','b');
title('Earthquake at Stations');
legend([i4 i5 i6 i7 i8 i9 i10],{'Station 1 Point' 'Station 2 Point' 'Station 3 Point' 'Epicenter Point' 'Est.L Station 1'
'Est.L Station 2' 'Est.L Station 3'},'Location','northeastoutside')

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 0.5, 0.5]);

telURL='https://api.telegram.org/bot393328733:AAGbqKbVfYYcW4kOhRiUgFxsWKulH_EkEfY/sendmessage
?chat_id=-1001190132929&text=done';
webwrite(telURL,20)
```