FREQUENCY OVERRIDING SYSTEM THROUGH SIGNAL AMPLIFICATION FOR FLOOD ALERT WITH SMS NOTIFICATION

A Design Project
Submitted to the Faculty
of
Computer Engineering Department
Adamson University

by

Rochelle G. Bastian
Elaine Grace S. Bayhon
Raymond Carl P. Chua
Emmanuel E. Jamoralin

In Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Science in Computer Engineering

Engr. Maria Concepcion A. Mirabueno
Adviser

September 2020

APPROVAL SHEET

The research project entitled **"Frequency Overriding System Through Signal Amplification for Flood Alert with SMS Notification**" prepared and submitted by Rochelle B. Bastian, Elaine Grace S. Bayhon, Raymond Carl P. Chua, Emmanuel E. Jamoralin in partial fulfilment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER ENGINEERING** is hereby approved and accepted.

Engr. Ma. Concepcion A. Mirabueno

Adviser

Approved by the committee on oral examination with the grade of_____.

Engr. Hubert Q. Temprosa

Panel Member

Engr. Yolanda D. Austria

Panel Member

Engr. Jonel R. Macalisang

Panel Member

Accepted and approved in partial fulfilment for the degree of **BACHELOR OF SCIENCE IN COMPUTER ENGINEERING**.

Engr. Yolanda D. Austria

Chairperson

Date: _____

Acknowledgments

To all the people who took part in our journey as we accomplish our design project *Frequency Overriding System Through Signal Amplification for Flood Alert with SMS Notification*, thank you.

We would like to thank God Almighty for giving us the strength to persevere to finish our design project, and keeping us safe amidst these trying times. Without His guidance, this achievement would not have been possible.

To our beloved parents, Thank You understanding and patience during the time doing this research especially with amidst trying times. We love you very much.

To our thesis advisor, Engr. Maria Concepcion A. Mirabueno, we are grateful for all the knowledge you have shared with us. Thank you for always taking time to provide us guidance. Thank you for your support and trust.

To our panelists, Engr. Hubert Q. Temprosa, Engr. Yolanda D. Austria, and Engr. Jonel R. Macalisang, thank you for giving us the trust that we can accomplish the research.

To Engr. James Santiago, a consultant at the Department of Information and Technology, for taking the time off of your busy schedule to meet us for consultation on such short notice.

To of Emmanuel Abalos of KBP – Kapisanan ng mga Brodkaster ng Pilipinas, for accommodating us for consultation, and for the further assistance by referring us to Engr. Santiago, thank you.

Lastly, we are grateful to the Bayhon family for the accommodation while we accomplish the project, and taking good care of us like a part of their family.

Table of Contents

List of Table

List of Figure

Abstract

Philippines' geographical location and physical environment—located near the equator where the ocean is warm, one of the major factors for typhoon formation—contributes to the country's high-susceptibility to tropical cyclones. And with global warming, warmer waters will only continue to intensify the storms to ever hit the country. As the storm intensifies, the flood occurrence is becoming more often than before. Flood monitoring systems' main role is to inform and also to possibly save people's lives.

Currently, there are various researches about the importance and implementation of flood monitoring. Most of the flood monitoring system being deployed nowadays use social media as its platform, as it is one of the current popular medium of communication. Sending an alert through social media does reach a larger audience for just a few clicks away. However, in this study, the researchers considered the downside of relying primarily on the convenience of the Internet and social media.

What the researchers have designed is a system that informs the people about the flood level in the area, with no internet connection needed. It is also designed to reach a specific audience by creating an SMS advisory system, which the subscribers can receive, and utilizing radio signals within a certain radius. In this study, the researchers will utilize FM radio frequency overriding and SMS advisory as the medium for flood level alerts to the people nearby and those who are subscribed.

The developed system can determine when to send SMS alerts and audio alerts show that the medium used to relay alerts are both SMS advisory and FM Radio. During

the simulation of floodwater, for all levels 1-5, the system was able to send alerts through SMS. For emergency levels 4 & 5, the system also sent alerts through FM radio overriding.

CHAPTER I.

THE PROBLEM AND ITS BACKGROUND

A. Introduction

Throughout the years, technology has changed the way we live life in general. Advancement in technology has changed the way we study, communicate, shop or purchase things, and many more. Since, the rise of the Internet and other technological advancement, the everyday lives of people were made easier and simpler--particularly in terms of communication. The continuous development of technology has paved numerous ways for hassle-free communication. Nowadays, information is readily available on different platforms of social media within seconds, provided one has access to the Internet. Social media has brought huge changes in the way people acquire news. Back then, people's primary source of information/news was not through the Internet--social media like Twitter, Facebook, YouTube, etc., but instead, people relied on TV, radio, or newspaper. As time goes by, a lot may have changed on the way people primarily obtain the news, but radio has proved its reliability--importantly, during times of emergencies or calamities e.g. typhoons.

Currently, there are multiple existing studies or researches about flood monitoring system. The research titled Early Warning System of Flood Disaster Based on Ultrasonic Sensors and Wireless Technology exists in Jakarta, Indonesia, utilizes ultrasonic sensor-based microcontrollers to track the flood water level, the social media platform, and SMS (short message service) for information broadcasting [1]. Another existing flood

monitoring system titled Flood Monitoring and Early Warning System Using Ultrasonic Sensor is from Isabela, Philippines. The design utilizes ultrasonic sensors as its component for flood water level detection and will be broadcasting information through their website or SMS [2]. Similar to the prior research is the paper titled Flood Detection Using Sensor Network and Notification via SMS and Public Network. The system consists of microcontrollers, sensors, modem, and computers. The same method of broadcasting was used in the system and it uses a GSM modem to send SMS alerts and social media platforms [3].

In this study, the researchers will utilize FM radio frequency overriding and SMS advisory as the medium for flood level alerts to the people nearby. Overriding radio frequency is often used by the military or the police to disrupt communications when an emergency is underway e.g. bomb threats, hostage situations, military action to confuse enemy radar, etc. Radio frequency is overridden by a device designed to prevent the reception of radio frequency to its receivers. The overriding device functions by emitting noise on a band, at a specific intensity that overrides or overwhelms the target receivers, makes the reception of signals impossible thus, this process enables the proposed system to insert legitimate information like news or status about the flood levels over the overridden radio frequency.

Another aspect of the alert transmission of the proposed system is Messaging Shortcode. A shortcode is ideally an easy to remember 5 or 6-digit number usually used for SMS marketing campaigns [4]. Commonly used by businesses to opt-in customers to their SMS promotions, the consumer interacts with an SMS short code by sending particular keywords [5].

B. Background of the Study

Over the past two decades, the Philippines endured a total of 274 natural calamities, making it the fourth most disaster-prone country in the world [6]. The Philippines is a country that has suffered from a large number of natural disasters—one of them is the deadly typhoons; others: earthquakes, landslides, volcanic eruptions, storm surges, etc. Located right amid the Pacific Typhoon Belt, the Philippines' area of responsibility is visited by approximately twenty tropical typhoons every year, five of which are destructive [7]. Its geographical location and physical environment—located near the equator where the ocean is warm, one of the major factors for typhoon formation—contributes to the country's high-susceptibility to tropical cyclones. And with global warming, warmer waters will only continue to intensify the storms to ever hit the country.

The Philippines is one of the countries most exposed to tropical storms in the world. In 2009, Typhoon Ondoy, internationally recognized as Typhoon Ketsana made landfall near the boundary of Aurora and Quezon provinces and flooded Metro Manila and Central Luzon for continuous 12 hours. A total of 239 barangays in Metro Manila were affected with a 341mm amount of rainfall, leaving a record of 464 fatalities. The typhoon battered Marikina City when it dumped the city with an equivalent of a month's rainfall, in such that the Marikina River rose to 23 meters above sea level, submerging 14 of the city's 16 barangays [8]. This is in a larger scale view of how these catastrophic disasters affect the country.

To view the effects of floods on a smaller scale, floods in Metro Manila these days are getting worse. Heavy downpour forces the Local Government Units to suspend classes both in private and public schools, as well as the Government workers because many parts of Metro Manila are getting submerged in floodwater. The Metropolitan Manila Development Authority (MMDA) told the Commission on Audit (COA) in a recent report that floods in the National Capital Region took longer to subside in 2018 than in 2017. In 2018 it took an average of 30 minutes for floods in Metro Manila to subside, compared to an average of 18.5 minutes in 2017 [9]—indicating that the Metro Manila floods are indeed getting worse. In the past, only the heaviest, consistent rain from strong typhoons triggered flooding. Unfortunately, nowadays, even regular monsoon rains and intermittent showers can cause flooding on the roads and streets, making them impassable. Following the heavy downpour is a high probability of heavy traffic due to floods.

In 2020, a recent report of flood following a heavy downpour flooded several streets of Quezon City that caused the streets to be impassable even to trucks [10]. While floods caused by sudden or consistent heavy rainfall are normal, a recent event in a low-lying riverside area in Santa Rosa, Laguna showed that floods are to happen even in the absence of rain in a particularly flood-prone area. The heavy rain occurred in the neighboring high-rise cities of Tagaytay and Cavite caused the flash flood [11].

Over the years, radio has proved its worth in times of emergency. Radio is easy and practical to use in times of calamities as it requires very low power, others are even battery-operated [12]. It can serve as a valuable source of information when the network is down, SMS traffic is heavy in networks, or when phone lines are cut.

No disasters indeed are the same, but the improvement in statistics could also be attributed to the better preparation and response of local governments and the locals. A well-established early warning system ensures timely evacuation. Flood warnings are a highly important adaptive measure to ensure public safety [13]. The system will serve its purpose of detecting flood water levels to avoid or lessen the chances of commuters and motorists getting stuck in the middle of a flood or heavy traffic. For life-threatening floods, the locals will receive preemptive alerts, therefore, they will be able to utilize the time to prepare, and to minimize the impact of the calamity. The system will be of help in regulating traffic flow and in mitigating massive loss of life and property by providing close monitoring of a particularly flood-prone area in a local community—factors like connectivity and power outage considered. It is known that inclement weather affects internet speed [14]; cases of large storms and high winds can be a cause for concern.

C. Objectives of the Study

The general objective of this study is to develop a Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification. In line with this, the project aims to achieve the following specific objectives:

In line with this, the project aims to achieve the following specific objectives:

- To develop a flood monitoring system that accurately detects the floodwater level in a certain flood prone area.
- To override FM radio frequency when flood reaches the last two emergency levels.

- To use short code in implementing SMS advisory for all 5 flood water level detections.

D. Significance of the Study

Below are list of significant contributions of the study.

- In a global context, the study will help government agencies such as the Strategic Emergency Management Council, Weather Bureau, and Local Government Unit in acquiring and disseminating flood water level information to its citizens.
- In an economic context, the study will help businesses that deal with transporting goods, manpower, etc. to be able to track down flood roads to avoid unnecessary delays.
- In an environmental context, the data gathered from this study will be able to help further projects and research concerning flood monitoring.
- In a societal context, the study will be able to help residents nearby to monitor the flood level easily, for them to be able to stay alert for possible evacuation if flood water continues to rise.

E. Scope and Delimitation

This study aims to monitor flooding in a particularly flood-prone area and send SMS notifications to the people around the area to prevent heavy traffic and casualties. The sensor beacon will be installed in a flood-prone area and it has several functionalities. The

flood level will be the trigger of the alert messages that the people will be receiving, either through SMS alone or through both SMS and Radio Frequency Overriding.

There will be five significant flood levels, the semi-critical, and the critical ones. The 3 semi-critical levels will only trigger the SMS functionality. The critical levels will make use of the two modes of relaying message alerts, both SMS notification and Radio Frequency Overriding. The depth measurement of the levels has 3 categories namely PATV (Passable to All Types of Vehicles), NPLV (Not Passable to Light Vehicles), and NPATV (Not Passable to All Types of Vehicles). Levels 1 and 2 will be under the PATV category, level 3 under NPLV, and levels 4 and 5 under the NPATV category respectively. The exact measurement for all of the levels that we have chosen will be referencing the MMDA's Flood Gauge, including their alias are Gutter Level = 9 inches, Half Tire Level = 13 inches, Knee Level = 19 inches, Waist Level = 37 inches, and Chest Level = 45 inches. The beacon will deliver the message that consists of the date and exact time the flood level occurred, level category, location, the exact measurement of the flood, and it's alias.

The SMS notification will be received by users who have subscribed to the shortcode. For the duration of the system sensing flood level data to sending the SMS notifications to receipt of the subscribed users, it takes a total of roughly 35 seconds. The ultrasonic sensor is programmed to scan for data for every 20 seconds interval. It scans for a total of 10 seconds, 10 readings with a 1-second delay for each reading. Upon computing for the average of the 10 readings, it takes roughly 5 seconds for this data to be sent to the Globe API. Once the data is sent, the Globe API will be the one who's responsible for processing the data and sending the alerts to the subscribed users. This applies to all of the 5 flood levels. The second medium that the researchers will be using is overriding the frequency

of FM radio stations. It is accessible even in times of blackout and absence of internet: radio receivers. Radio communication is extremely critical for public safety, national safety, and emergency communication systems. These radios are usually low cost, small size, portable, and require low power to operate [15].

All the FM frequencies within the radius covered will receive the broadcast once the flood reaches the critical level. For the duration of the system sensing flood level data to overriding an FM radio frequency, it takes a rough total of 65 seconds. The system senses data and gets an average reading for every 10 seconds. Acquiring data from web server takes 5 seconds, the process of converting these data from text to speech, converting mp4 file format into wave file format takes approximately 5 seconds. Transmitting to a particular frequency takes 45 seconds; depending on the length of the alert. For every successful audio transmission, the system has a cool down of 3 minutes and 30 seconds.

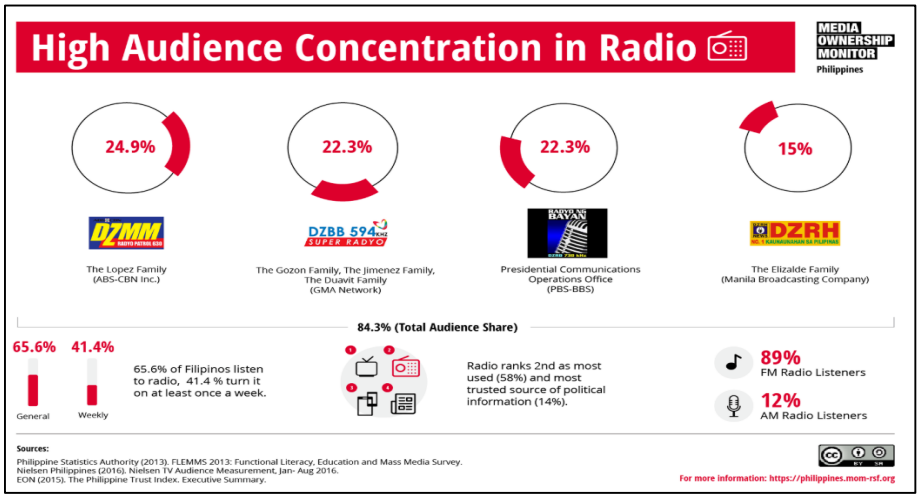


Figure 1. Radio Audience Concentration

Figure 1 shows the percentage of statistics that the majority of the radio audience commonly tune in to FM radio stations. Due to constraints, this study will not cover

overriding all local radio frequencies available. AM stations are excluded since vehicles commonly tune in to local FM stations. Overriding radio frequency can only be used once the urgency for evacuation is needed, either voluntary or forced.

Only those who subscribe to the shortcode will receive the SMS notification. Subscription to the system is not limited to Globe users; all networks are allowed to subscribe to the shortcode, free of charge. Once subscribed, the end-users also have the option to opt-out of the advisory services.

For a demonstration of the prototype, the frequency overriding using an SDR will only be limited to overriding FM stations within a certain area that's only the size of a room. Beyond that, there will be a violation of the law as this study is for research purposes only; for commercial use, a transmitting license is required. It is also not possible to override frequencies all at once, the system can only override stations one by one, consecutively. The messages will be relayed to the radio receivers with intervals, depending on the duration of the message. The frequency overriding may only be done strictly for emergency purposes.

For testing and demonstration, the researchers sent a letter seeking permission to the local government of Tanay, Rizal for the beacon to be situated in a certain flood-prone area.

F. Conceptual Framework

To successfully achieve the desired outcome of this study, certain procedures, requirements, and ideas were carefully discussed to conceptualize the project's design

and development. After long deliberation and brainstorming, one thought was agreed upon in which resulted in one concept.

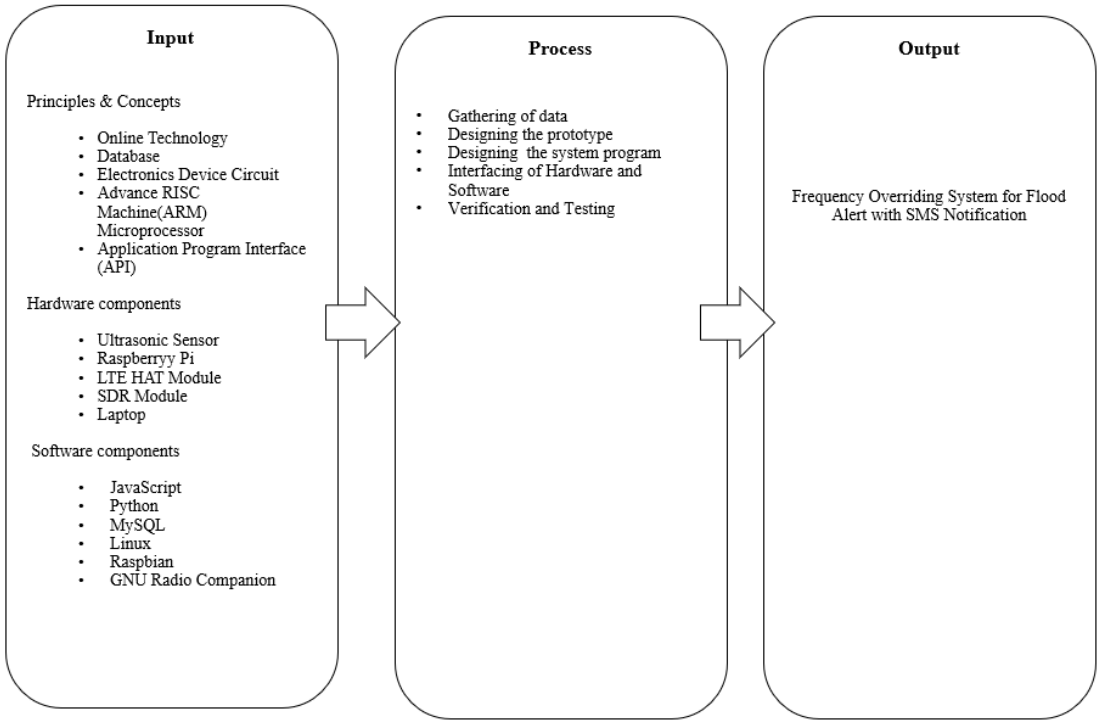


Figure 2. Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification Conceptual Framework

Figure 2 shows the conceptual framework of the research design. It covers the three major parts, namely: input, process, and output phases.

The input part of the conceptual framework covers the principles and concepts applied to the research. Background studies on online technology, database, electronic device circuit, advanced RISC machine (ARM) microprocessor, application program interface (API) are necessary for formulating the system. The study involves software and hardware components as well. For the hardware components to be used, ultrasonic sensor, Raspberry

Pi module, LTE module, SDR module, Laptop, and a power bank as a source. To integrate these components, the software to be used are JavaScript, Python, MySQL Database, Linux OS, Raspbian OS, and GNU Radio Companion. The process section of the conceptual framework shows the steps needed to be accomplished to finish the project.

The researchers identified the problem they aim to solve, then developed a system that addresses this problem. The researchers needed to develop a flood warning system that transmits a reliable and precise flood water level via SMS alerts and FM radio. The output generates a Frequency Overriding System for Flood Alert with SMS Notification.

For the setup, the Ultrasonic Sensor is controlled and configured by a Raspberry Pi with an LTE module. LTE is a cellular communication module used in this system for the Internet connection. Upon Internet connectivity, data transfer between the modules and the webserver is now established. The software used to program the Raspberry Pi is Python and MySQL for the database. These components needed to work together for the system to fully function successfully.

G. Operational Definition of Terms

- Database

  A database is a collection of organized information that is usually stored in a computer system. It is typically modeled in an efficient structure to easily access, manage, modify, update, and organize. It is usually stored in a computer system [17]. For this study the database to be used is MySQL. MySQL is an open-source Database Management System (RDBMS) that uses

Structured Query Language (SQL). SQL is a popular language for managing content in a database.

- Flood Prone Area

  An area wherein the flood occurs when rainfall intensity is more than 50mm per hour [18].

- Globe Labs API

  API stands for Application Programming Interface, a software that acts as an intermediary platform for two software applications to talk to each other. Globe Labs API is an API developed by Globe Telecom. It is a software tool that allows developers to program their solutions and services e.g. SMS-based or location-based solutions. Using Globe Labs' API, messaging services can be integrated into the application [19].

- GNU Radio

  GNU Radio is one of the free, open-source applications that is used to configure SDRs [20]. It provides signal processing blocks to configure software radios [21].

- LTE Base HAT

  A HAT is a rectangular board with similar measurement specifications with Raspberry Pi and is supposed to stack right on top of it, perfectly aligned with the four mounting holes on each corner. LTE Base HAT is a HAT that enables Raspberry Pi-based projects to access data networks all around the world. Making a remote controllable LTE Wi-Fi Hotspot, high-speed GPS tracking, more and more use case is possible with his add-on board [22].

- Overriding

  Interrupting an automatic process, e.g. radio frequency, typically to take manual control.

- Raspberry Pi

  Raspberry Pi is a credit-card sized computer with similar capabilities like that of a desktop computer [23]. It is a Single Board Chip Computer that will be used for driving the ultrasonic sensor and for interpreting the data that will be coming from the sensors. It is usually abbreviated as RPi.

- Raspberry Pi OS

  Previously called Raspbian, and Foundation's official supported operating system [24], will be the OS installed on the Raspberry Pi.

- SDR

  Acronym for Software Defined Radio. It is a radio communication tool wherein the typical hardware-implemented components of radio are instead implemented using software [25].

- SMS

  Stands for Short Message Service and is the most commonly used type of text messaging. Standard SMS are usually limited to 160 characters per message. If a message exceeds the 160-character limit, the messages are to be sent in segments [26].

- SMS Short Code

  Typically, an easy to remember 5-6-digit phone number used by business firms to initiate opt-in from consumers to their SMS-based marketing

platform. Short codes are capable of sending large quantities of SMS messages which makes it a potent marketing tool, especially for services that are time-sensitive [27].

- Ubuntu Linux

  Linux is an open-source operating system. Ubuntu is a distribution of Linux, and will be the OS of the SDR-controlling Laptop to be used in this research. Ubuntu is an example of a commercial project based on the Linux kernel [28]. The Ubuntu OS installed is on 18.04.4

- Ultrasonic Sensor

  One type of level sensing module. An instrument that measures distance using ultrasonic sound waves. Ultrasonic sensors measure the distance by measuring the time between ultrasonic wave emission and wave reception [29].

- VMWare Workstation

  Enables users to set up virtual machines that can execute their operating system. VMWare Workstation 15 Player will be installed in a Laptop running on x64-based Windows 10, and will be used to set up a Ubuntu Linux Virtual Machine 15.5.1 build [30].

- Web Server

  Web server is computer programs that provide the web pages when requested by the web client [31].

Chapter II.

Review of Related Literatures and Studies

Review of related literature and studies for flood monitoring shows diverse and varied means of implementation of data distribution. Back in 2018, a research design titled Road Flood Sensor with Web and Mobile Application Support was designed by Ado, et al [32]. It is a flood monitoring system that utilizes the technology of SMS and the Internet by deploying its services through text messages, web, and mobile applications. This research also includes a prototype design that senses water level on a flooded road through the use of an ultrasonic sensor attached to a gizDuino™.

Another existing paper that focuses on flood monitoring is by Indrasari—titled Early Warning System of Flood Disaster Based on Ultrasonic Sensors and Wireless Technology [1]. The proponents see the necessity of designing a system that sends alerts before the flood disaster. The research is not only based on developing a device that detects the water level by the use of the ultrasonic sensor, but the water flow velocity as well. For the medium of information, wireless technology and GSM were used.

One more study on flood monitoring implementation is Flood Monitoring and Early Warning System Using Ultrasonic Sensor by J G Natividad and JM Mendez [2]. This study is closely related to the prior mentioned research; its purpose is also to develop a flood monitoring and early warning system. The sensor used for sensing water level is also an ultrasonic sensor, the medium of information is also via SMS. The only difference between the two papers is that the prior has an additional feature to its flood monitoring system.

Natividad and Mendez's system design features gauging the flow rate of water through a water flow sensor.

All previously mentioned researches all have major similarities: the process of sensing water level is through the use of ultrasonic sensors. One advantage of ultrasonic sensing is ultrasound can propagate through solids, liquids, and gases. Ultrasonics are widely used as they can be used both indoors or outdoors. They can be reliably implemented for water level sensing [33]. Few to many of available research resources prove its reliability in gauging water level since it is the almost universally used component for flood monitoring.

The proponents created their design of the flood monitoring system. While the system utilizes the widely-used ultrasonic sensor used for measuring water level, a new medium of information is to be applied for emergency purposes. FM Radio frequency is to be overridden for the last two emergency flood levels, and SMS for all the 5 levels. Overriding of frequency is not relatively new in terms of implementation. A research patent under the title Emergency Signal Intercepting Unit by Boscacci [34] is a system that warns a vehicle that is proximal to an emergency vehicle (ambulance). A radio is associated with the vehicle, receiver, a speaker, and a decoder to send a warning signal to the speaker when the receiver receives a signal from the emitter. In this study, the emergency signal intercepting will be on radio signals—under strict regulations that comply with the urgency of declaring an emergency.

According to Engr. James Santiago, a former consultant at the Department of Information and Communications Technology, under the 1987 Philippine Constitution is a clause that discusses the legalities and validities of intercepting signals. Under Article XII Section 17, "In times of national emergency, when the public interest so requires, the State

may, during the emergency and under reasonable terms prescribed by it, temporarily take over or direct the operation of any privately-owned public utility or business affected with the public interest".

Referencing the book of KBP's Technical Standard and Operating Requirements for Broadcast Stations in the Philippines, considerations for allocation of frequencies for FM broadcast stations are stated. Purposes of this research fall under Class-D station, which is specified to have only authorized transmitter power not exceeding 10 watts. For the methodologies of this research, educational stations shall be allowed to operate with Class-D transmitter power [35].

Flood Monitoring

Timely and active detecting and monitoring of a flood event are critical for a quick response, effective decision-making, and disaster reduction [36]. Given that the Philippines is indeed prone to weather-related hazards such as floods, storms, and other natural calamities due to its geographical location, one of the precautionary measures deployed by the government is the Philippine Disaster Risk Reduction and Management Act. It aims to attain a "safer, adaptive, and disaster-resilient Filipino communities towards sustainable development" [37]. According to Wikipedia, 15-20 typhoons hit the country every year, making floods and disasters frequent specially to crowded cities like those in Metro Manila.

Floods in Metro Manila continue to grab headlines, not only because of the threat to life and property but also because of the nuisance it generates thru massive snarls in traffic. By monitoring floods in the streets of Metro Manila, worsening traffic may be avoided

through early identification of impassable roads and suggestion of alternative routes to motorists [38]

To lessen the considerable losses of these catastrophic events, this design project aims to develop a flood monitoring system by detecting an increase of flood water level at a certain flood-prone area and issue warnings direct to the covered public area through text message and radio.

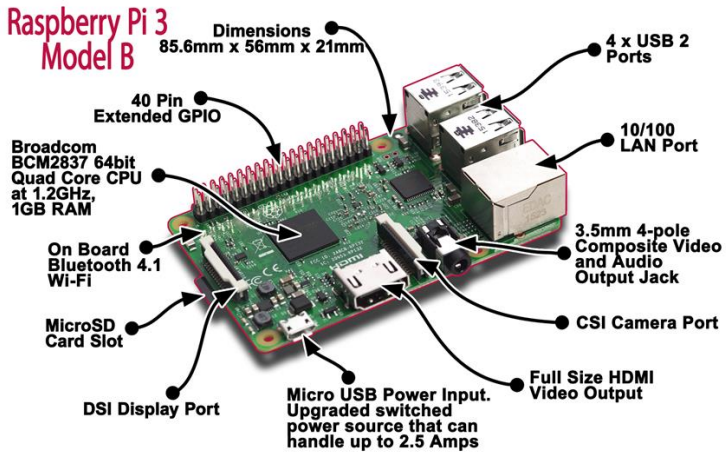


Figure 3. Raspberry Pi 3b+

Figure 3 shows a standard Raspberry Pi 3b+ model and its different available ports. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. Its capabilities are almost similar to those of desktop computers [39]. Raspberry Pi 3b+ is designed to let its users experiment with building and configuring hardware and software applications.

The Raspberry Pi does support similar functions to that of desktop computers—The Pi can perform most of the common tasks users would expect a decent desktop to do, just a bit slower. The difference between the Raspberry Pi and an average desktop PC is the

processor architecture, speed, and computing power. For the architecture, standard desktop manufacturers are most likely to run on a 64-bit CPU. On the other hand, the Raspberry Pi runs on a mobile processor—an ARM CPU that uses a different instruction set (RISC-Reduced Instruction Set Computer). CISC (Complex Instruction Set Computer) architecture based processor performs more operations in fewer clock cycles compared to ARM CPU. For Raspberry Pi 3b+'s CPU and RAM specifications, it runs on Broadcom BCM2837B0, Quad-core Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz CPU and 1GB LPDDR2 SDRAM [40].

Ultrasonic Sensor

Figure 4. Ultrasonic Sensor

Figure 4 shows an ultrasonic sensor module [29]. The component consists of 4 pins: VCC, GND, Trig, and Echo. The ultrasonic sensor works by using trigger and echo pins. In this study, the object of reference is flood water. The transceiver module triggers and sends the ultrasonic wave signal to the water, and the reflected signal is read by the echo

pin. With this data, the ultrasonic sensor calculates the distance of the object and returns the data level height of water [41]. Ultrasonic sensors measure the distance to the target by measuring the time between emission and reception of ultrasonic sound pulses [42].

When it comes to liquid level sensors, there are many different types to choose from. Just like any technology, they each have their advantages and disadvantages that vary depending on the implementation.

An ultrasonic sensor system does not require actual contact with the reference of measurement to gather data, therefore clogging on the prototype can be avoided. Since water is the reference of measurement in this study and is subjected to color and transparency, ultrasonic sensors reflect off the sound of objects, so this doesn't affect the sensor's reading. Prototype design shows that the sensor should be housed in a weatherproof enclosure, but unlike proximity sensors, dark environments do not affect its detection ability [43]. For this study, considering ultrasonic sensors are versatile and reliable, the fittest for the design, the researchers have chosen ultrasonic as a method of implementing the level measuring process.

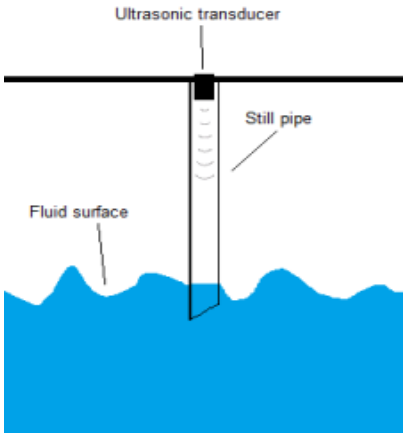


Figure 5. Ultrasonic Sensor Setup

Figure 5 shows a workaround for the possibility of result variation when dealing with agitated liquids i.e. flood as the point of measurement reference. In the setup, the ultrasonic sensor is mounted on top of a pipe that extends below the water level. Installing a pipe protects the liquid inside from turbulence and also improves the sensor's function by focusing the sound pulse on a more enclosed and concentrated area [44].

SDR

SDR is an acronym that stands for Software Defined Radio. A radio is any kind of device that wirelessly transmits or receives signals in the radio frequency part of the electromagnetic spectrum to facilitate the transfer of information. Simply put, SDR or Software Defined Radio is a radio in which some or all of the physical layer functions are defined in software [45]; some of the radio functions typically implemented in hardware are converted into software [46].

SDR implements a large part of radio functionality in software rather than hardware. It is a radio communication system wherein hardware functions like filters,

amplifiers, modulators, etc. are implemented in software [47]. The advancement in modern technology of computing and analog to digital converters allows most of these traditionally hardware-based components to be implemented in software [48]. It takes the analog signal processing to process the radio signal on a computer using software algorithms.

For this study, the SDR used is HackRF One. It has a tuning range of 1MHz – 6GHz; a half-duplex transceiver with a Hi-Speed USB 2.0 connection, and has a maximum quadrature sample rate of 20 million samples per second [49].

Frequency Overriding

In this design project, radio waves are to be used as a medium to send flood alerts to nearby residents. In times of emergency and disaster, radio is used as one of the primary means of emergency communication as it requires low power. Radio FM frequency waves are chosen as a medium to relay alerts as it highlights the importance and critical role of radios in times of calamities. When all other modern technologies fail, radios are much expected to retain.

Frequency overriding is to be implemented as a precautionary measure when severe levels of floodwater are increasingly detected. Radio receivers are to receive flood alerts through frequency overriding.

Referencing to the paper “Road Traffic Information Using a Dedicated Radio Beacon” [45], processing of frequency overriding is cited but with different applications. The prior system focuses on traffic monitoring; The proposed local beacon will override the network-wide radio frequency for a targeted road segment. The beacon is operated over

large audience frequencies. This beacon aims to deliver emergency information to drivers, using the existing unmodified car radios.

Another paper “Override Transmitter” utilizes frequency overriding [50], tackles the use of frequency overriding for emergency vehicles. The emergency transmitter apparatus will have the ability to transmit a siren or audio message through the radio, overriding the existing radio station that the motorist is tuned to, making the sound resistant qualities of the vehicle and the radio noise irrelevant.

For this study, the system of radio frequency overriding is designed for simultaneous overriding, wherein alert transmission is done one frequency at a time, consecutively. In this sense, the order of flood alert transmission is going to be first in the lower frequencies, followed by the higher radio frequencies. While the system for this study is not designed for simultaneous transmission—wherein the SDR is to transmit on multiple radio frequencies at the same time, it is definitely should be a reference for future studies. HackRF’s official forum on GitHub indicates that simultaneous FM transmission is possible but it depends on channel spacing, and as long as the signal to be transmitted is only less than 10MHz [51].

Another relevant related literature is a repository on GitHub for simultaneous transmission but only limited to specific signals i.e narrowband FM, wideband FM, PSK31 on specific frequencies. It is primarily built for BladeRF, but is adaptable to be implemented in HackRF [52]. A research documented on Google patents titled ‘Emergency Vehicle Radio Transmission System’ is a prior art closely related to this research, only it is meant for emergency vehicles like ambulances. The radio transmission apparatus is used

by emergency vehicles and is to be used for alerting vehicles nearby by broadcasting over the entire frequency spectrum regardless of what frequency is the receivers tuned into [53].

An article from informationweek titled 'Military Seeks Radio Override' seeks to provide a radio broadcast system that is capable of broadcasting messages to all received AM and FM radio station frequencies in a particular target area [54]. As of present, no simultaneous transmission on FM radio frequencies for flood monitoring exists, but these related literatures, though of different application, indicate that simultaneous transmission to multiple frequencies should be considered as an area of improvement for future studies.

Short Code

A short code is usually easy to remember a 5-6-digit contact number. It is usually set up to allow subscribers to receive SMS notifications. In this study, a shortcode is to be created using the Globe Labs' API. With the Globe Labs API, the project developers will be able to send targeted text messages to the target end-users [55]—thus, this platform is to be used for sending flood alerts to the subscribers. To receive such alerts, the end-users must be subscribed to the short code.

GNU Radio

GNU Radio is a free & open-source software development toolkit that provides its users access to signal processing blocks to implement software radio systems, e.g. radio receiver, radio transmitter, etc. It has block filters, demodulators, decoders, and many other elements

typically used in radio systems. GNU Radio has a method of managing data transfer by connecting these blocks [56]. In installing GNU Radio, Linux OS distributions are the optimal choice because they are kept well up to date. Therefore, in this study, GNU Radio is installed on VMWare running on Ubuntu Linux.

GNU Radio applications are primarily written in Python and C++. Since GNU Radio has a graphical user interface, systems can be built by drag-and-drop. Extending these applications i.e adding functionality through coding is also possible.

For its system requirements, a modern PC/laptop computer usually can perform most tasks such as receiving frequency signals, doing audio frequency processing, and many narrowband digital signals. GNU Radio supports both 32-bit and 64-bit operating systems. For processor support, Intel x86 architectures have the best support features and speed. On the other hand, support and capabilities are still currently in the works for ARM processors as they tend to be severely underpowered for math and signal processing. The average embedded ARM platform could not handle the usual sample rates of over 5 million samples per second [57].

For radio hardware, GNU Radio supports a wide range of commercially available SDRs. For this study, HackRF One is the hardware used.

Flood Prone

Below is a list of flood-prone areas in Metro Manila, as identified by the MMDA when rainfall intensity is more than 50mm per hour. [58].

NORTH MANILA FLOOD CONTROL OPERATION DISTRICT
1. Rizal Ave: Pampanga – Cavite St. (Southbound) 2. Rizal Ave Cor. R. Papa (EST. MAYPAJO) 3. Tayuman: Rizal Ave. Ipil St. 4. Tayuman: Almeda – A. Rivera St. 5. Abad Santos: Cor. Cavite St. (Northbound) 6. Moriones: Sevilla – Juan Luna St. 7. H. Lopez Blvd. Fronting Tondo Medical Hospital 8. R-10 Corner Jacinto St.

CENTRAL MANILA FLOOD CONTROL OPERATION DISTRICT
1. Blumentritt · Maria Clara St. to Calamba St.
2. Dimasalang Ave. · Makiling St. to Retiro St. / Maceda St.
3. Maceda St. · Piy Margal St. to Makiling St.
4. Piy Margal St. · Maceda St. to Blumentritt St.
5. Dapitan St. · Cristobal St. to Maceda St.
6. Laong Laan St. · Andalucia St. to A.H. Lacson St.
7. Andalucia St. · Dapitan St. to Maria Clara St. (North Bound) · Remigio St. to Malabon St. (South Bound)
8. Dapitan St. · A.H. Lacson St. to Andalucia St.
9. España Blvd. · Antipolo St. to Blumentritt
10. Rizal Ave. · C.M. Recto Ave to Doroteo Jose St.
11. C.M. Recto Ave.

· Rizal Ave. to Evangelista St.
12. Legarda St. · Corner Gastambide St.
13. Quezon Blvd. · foot of Quezon Bridge (North Bound)
14. Ramon Magsaysay Blvd. · D. Ampil St. to Ramon Magsaysay Bridge
15. V. Mapa St. · Guadal Canal St. to Old Sta. Mesa St.

SOUTH MANILA FLOOD CONTROL OPERATION DISTRICT
1. AROUND CITY HALL & VICINITY · Along P. Burgos, Victoria, Arroceros & Conception
2. TAFT AVENUE · Corner U.N Ave. (N.B.) · Corner Apacible St. (N.B.) · Corner Escoda St. (N.B.) · Corner P. Gil St. (N.B.) · Corner Malvar St. (N.B.) · Corner Pres. Quirino Ave. (Both Sides) · Corner Dagonoy St.
3. T.M KALAW ST. (Ma. Orosa – Taft Ave.)
4. P. GIL ST. (Railroad Track – Main St.)
5. Around Paco Park
6. PASIG LINE COR. A FRANCISCO ST.
7. SAN ANDRES (Singalong–Muños St.)
8. LEON GUINTO · San Andres – Galvan · Cor. Dagonoy St. · Escoda St. – Apacible St.
9. SYQUIA ST. (Cor. Cagayan St.)
10. LAGUSNILAD UNDERPASS

11. JONES BRIDGES UNDERPASS
12. PRES QUIRINO AVE. · P. Gil Northbound · A. Linao
13. AYALA BLVD. (ROMUALDEZ ST.)
14. U.N. AVE. COR. ROMUALDEZ ST.
15. MLLE MAGALLANES (Back of Post Office)
16. P. GIL (Taft Ave. – Pres Quirino)

Flood Gauge

Below is the flood level guide from the MMDA. This type of measurement is another way of estimating the flood level without using tools for measuring.

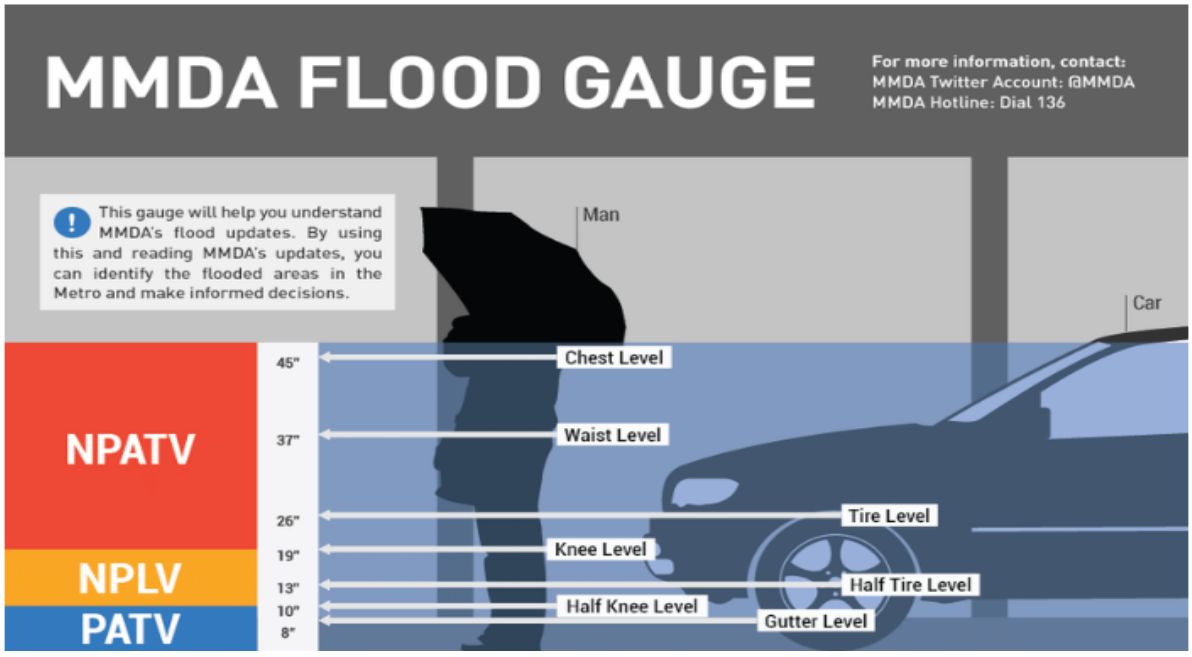


Figure 6. MMDA Flood Gauge

Chapter III.

Methodology

A. General Method Used

This study employed a descriptive method of research. In descriptive research, the population or phenomenon studied is being described. This type of methodology places emphasis on finding out "what is" rather than the "whys" of the research subject. Descriptive research is a method comprised of observations and descriptions of a particular subject--all without interfering or influencing its behavior in any way. Under types of descriptive research, the observational method is implemented. This method helps the researchers formulate a solution to the problem that this study aims to solve. Throughout the study, an observational method was used to set and design the functionalities of the flood monitoring system.

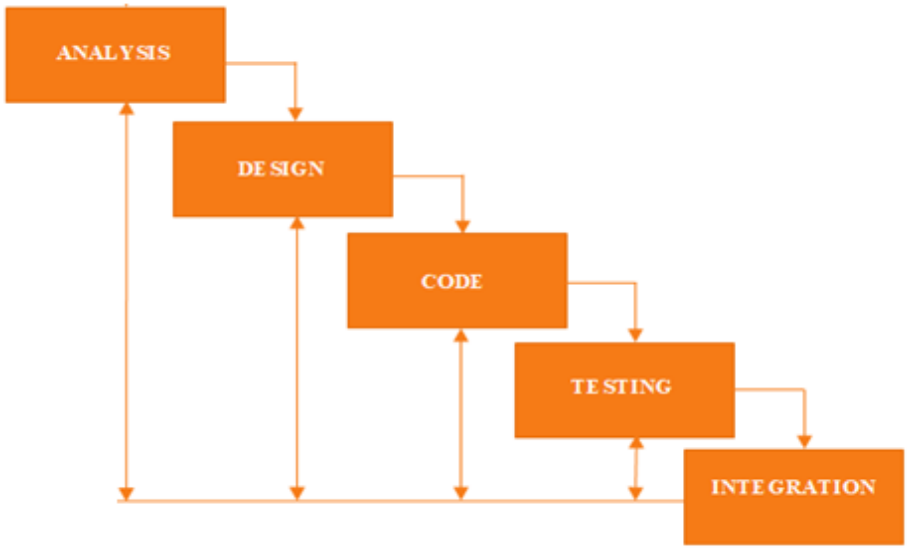


Figure 7. Iterative Waterfall diagram

Figure 7 shows the iterative waterfall diagram. The waterfall model emphasizes that a logical progression of steps be taken throughout the software development life cycle (SDLC), much like the cascading steps down an incremental waterfall. For the analysis stage, the probable system is analyzed to properly generate modules that will be used for the system. The design stage focuses on covering technical design requirements of the system, such as the programming languages to be used, data flow, services, etc. A design specification that outlines the processes covered in the analysis will technically be implemented. In the coding stage, the actual source code is written. Next is the testing stage, issues to be resolved are usually discovered. Commonly, this stage will undergo repetitions to debug the previous coding phase. The last stage is the integration wherein the data and the progress done from the previous stages are being combined until such time that the prototype is ready for deployment. The last stage is the operations wherein the system is ready for deployment [59].

B. Procedure

The study employed the common elements of the engineering design process described by Ford and Coulston [60].

1) Problem Statement

Heavy and continuous rains commonly cause traffic jams, commuters, and motorists stuck in traffic or flood. On the other hand, during severe cases of typhoons, power lines are vulnerable to damage due to strong winds. Another probable conflict that may arise is a poor Internet connection. Similar to cases of power outages, strong winds and heavy rains

pose a serious threat to physical cables and may lead to disruption of the signal transmitted by Internet providers. Two of the issues this research intend to address is the motorists stuck in flood and traffic, and the possibility of power outages and disruption of signal in times of severe typhoons. Several existing flood monitoring systems are implemented through the use of social media as its means of broadcasting, a platform that relies primarily on an Internet connection. Hence this project, the researchers aim to design a flood monitoring system that is comprised of two mediums of broadcasting—SMS alerts and another through FM radio frequency. With this, the researchers will design a flood monitoring system that will serve its function even in times of extreme calamities.

2) Requirements Specification

The requirement specification mentioned in this study provides a functionality outline of the system to be developed. It provides an overview of how the system is supposed to work, and its corresponding operational conditions to its target users. The requirement specification mentioned in this study allows people nearby the beacon device to receive flood alerts through SMS short messages and/or radio.

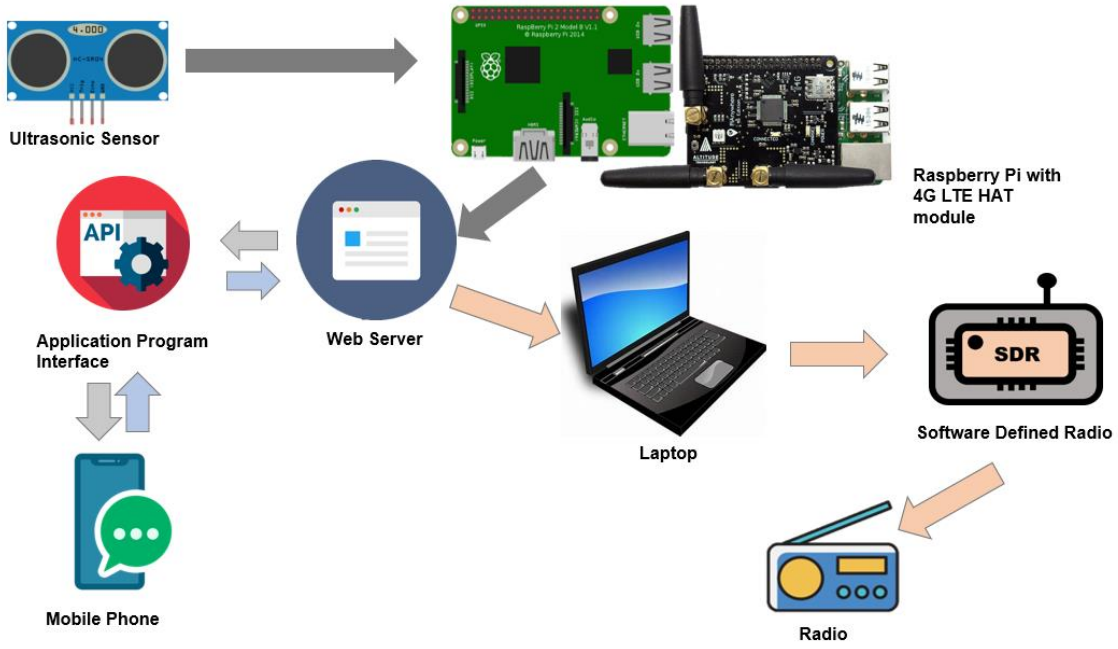


Figure 8. General System Architecture

Figure 8 shows the general system architecture of the project. In the prototype, attached to the Raspberry Pi are the ultrasonic sensor and LTE module. The LTE module is a communication module that supports LTE mobile Internet connectivity for the Raspberry Pi. The data gathered by the Ultrasonic Sensor are then sent to the server through the Raspberry Pi. One broadcasting medium of this study is through the SMS advisory service. For all the 5 levels, SMS advisory is going to be triggered. If the ultrasonic sensor detects the flood water reaching the last 4th and 5th levels, FM radio frequency overriding will also be implemented. For the process of FM radio frequency overriding, an x64-based laptop running on Ubuntu Linux will be used to run GNU Radio to configure the SDR (Software Defined Radio). The SDR will transmit the audio alerts to the FM radio units around the specified radius.

Figure 9. Ultrasonic Sensor HC – SR04

Figure 9 shows the sensor used in the prototype. Ultrasonic Sensor (HC-SR04) module consists of an ultrasonic transmitter, receiver, and a control circuit. It has four pins namely VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). Its operation voltage is 5 Volts DC, operating current of 15 mA, measure angle of 15°, and a ranging distance of 2cm to 4m [61]. It is a pioneering module that is responsible for detecting flood water levels in the project. The floodwater level is calculated as the difference between the measured distance and the distance from the sensor to the ground.

Figure 10. Raspberry Pi 3b+

Figure 10 shows the minicomputer used in the project--Raspberry Pi 3b+ model, the final revision in the Raspberry Pi 3 range. It runs on a 1.4GHz 64-bit quad-core processor and has dual-band wireless LAN, 4.2/BLE, faster Ethernet, and Power-over-Ethernet

support (with separate PoE HAT), contains a 1.2GHz ARM Cortex-A53 CPU ARM 64 architecture [62]. It is responsible for sending and receiving data from the sensor and the webserver.

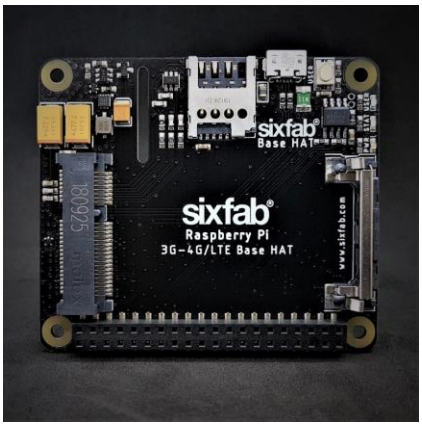


Figure 11. LTE Base HAT

Figure 11 shows the module used in establishing an internet connection for the Raspberry Pi in the project. This base HAT enables access to data networks on remote devices [63]. It is responsible for giving Raspberry Pi a high-speed internet connection that is used in sending and receiving data.

Figure 12. HackRF One

Figure 12 shows the module that is used in transmitting signals to radio frequencies within a specific radius in the project. HackRF One is a half-duplex transceiver, with an

operating frequency of 1MHz to 6GHz [64]. It is responsible for overriding radio signals and transmitting pre-recorded messages regarding flood water rising levels.

Figure 13. Python

Figure 13 shows the logo of Python, the programming language used in this project. Python is a high-level scripting language that is easy to read and simple to implement [65]. This is used to create programs that interact with the different modules of the system. Those programs are uploaded to a web server.

Figure 14. Python anywhere

Figure 14 shows the logo of Python anywhere. Python anywhere is a web hosting service wherein coding, running, and hosting python apps and scripts are made possible [66]. This is where all data from the application of the project are stored. Also, it serves as the database of the system.

Figure 15. Globe Labs

Figure 15 shows the logo of Globe Labs. Globe Labs allow developers to program their innovative applications, solutions, and services [67]. This is used as an API provider to send Flood alert SMS notification to all subscribers in the project.

Figure 16. GNU Radio Companion

Figure 16 shows the logo of the GNU Radio Companion. It is an open-source software development toolkit that enables developers to implement software radio operations and processes [56]. This is used to create a flowgraph that can enable to override FM radio signal and transmit prerecorded messages for high-level flood water level alert. It is also used to amplify radio signals for clear transmission.

Figure 17. Pycharm

Figure 17 shows the logo of PyCharm. PyCharm is an integrated development environment (IDE) specifically for Python [68]. This is the platform used to develop the application of the project.

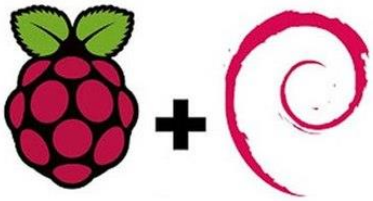


Figure 18. Raspbian

Figure 18 shows the logo of Raspbian. Raspbian is an operating system based on Debian and is optimized for Raspberry Pi's hardware [69]. This is used as the Operating System of the Raspberry Pi in the project.

Figure 19. jQuery

Figure 19 shows the logo of jQuery. It is a small Javascript library that can be used for making UI interactive [70]. This library is used for event handlings in the project.

Figure 20. D3.js

Figure 20 shows the logo of the D3. D3.js is also a JavaScript library that can create data visualization by combining HTML, CSS, and SVG [71]. This was used in plotting the line chart in the project.

TABLE I
WEIGHT MATRIX OF MARKETING REQUIREMENTS

	Reliability	Precision	Cost	Durability	Aesthetics	Weight	Total	Ranking
Reliability	x	0	1	1	1	0.75	0.30	2nd
Precision	1	x	1	1	1	1.00	0.40	1st
Cost	0	0	x	0	1	0.25	0.10	4th
Durability	0	0	1	x	1	0.50	0.20	3rd
Aesthetics	0	0	0	0	x	0.00	0.00	5th
						2.50	1.00	

Table I shows the project's weight matrix of marketing requirements, alongside with its total score and overall ranking. The primary requirements of the flood monitoring system are reliability, precision, cost, durability, aesthetics. Reliability indicates its integrity as a whole system, precision indicates the accuracy of the data gathered by the system, cost accounts for the total cost of raw materials to be used for the prototype, durability measures how these raw materials are functional to be used in the prototype, and

aesthetics for the overall prototype design.

After ranking these requirements, precision ranked out to be the most significant and the most important. It is of priority that the data gathered and sent by the system is precise, for the system to be considered reliable. Durability is prioritized compared to cost because the budget for materials is not to be neglected to sacrifice the quality and durability of the prototype.

TABLE II
REQUIREMENT SPECIFICATION OF FREQUENCY OVERRIDING AND SMS NOTIFICATION SYSTEM FOR FLOOD MONITORING SYSTEM

Marketing Requirement	Engineering Requirement	Justification
1	Utilize ultrasonic sensors that will accurately measure flood water levels.	The ultrasonic sensor will be able to detect the depth of floodwater
1, 2	The database should be developed using MySQL Database.	MySQL Database will store subscribers' mobile numbers as well as the flood level status payload.
2	Create an SMS advisory system using Globe Labs API	The system will use API as an intermediary software between the Web server and the users.
1, 2	Correctly interpret readings to determine which medium of broadcasting to utilize.	SMS advisory and radio frequency will be used as a broadcasting medium.
1, 2, 7	Push SMS alerts to the advisory subscribers for every water level reading.	SMS alerts will be sent for all the 5 levels
1, 2, 4, 5, 6	Override FM radio frequency using SDR as FM transmitter when the ultrasonic sensor detects flood water has reached the last 2 emergency levels.	Override radio frequency for the last 2 emergency levels.
1, 2, 4, 5, 6	SDR must be able to transmit and override radio frequency	SDR will act as an FM transmitter that sends audio cue alerts.

3	The beacon will utilize a power bank as a source of power.	The beacon should have a power source.

Marketing Requirements

1 The sensor should be programmed optimally to accurately measure the floodwater level.
2 The database should be able to store acquired data and be accessible.
3 The beacon device should have a source of power.
4 Transmitted FM radio alerts should be clear and understandable.
5 The system should transmit flood water level alerts to the target users within the specified range/coverage area of the transmitter.
6 The system should be able to override radio frequency for the last 2 emergency levels
7 The system should transmit flood water level alerts to the SMS advisory subscribers

Table II shows the marketing and engineering requirement specifications of the Frequency Overriding System Through Signal Amplification for Flood Alert with SMS Notification. Marketing requirements refer to the customer needs, how the system is supposed to work expressed layman's terms. While engineering requirements refer to the technical aspects of the system and system performance requirements.

The first engineering requirement describes the essential role of the ultrasonic sensor's precision in measuring flood water level to the reliability of the whole system. It addresses the marketing requirement of sensors' need to be programmed optimally to accurately measure the floodwater level. For the ultrasonic sensors to measure flood water levels accurately, data and results must be acquired and delivered with minimal delay. The ultrasonic sensors' output serves as input to the Raspberry Pi.

The second engineering requirement describes the function of a database in the system. In this design project, the MySQL database is to be used to store subscribers' mobile

numbers and the data gathered from the sensing component. This database should be accessed remotely.

The third engineering requirement highlights the use of API in the system. For this design project, Globe Labs API serves as the communication between the web server and the SMS subscribers. It fulfills the marketing requirement that the system must be able to store acquired data.

The fourth engineering requirement refers to the importance of reliable data gathered from the ultrasonic sensors and correctly interpreting these data to determine which of the two mediums of broadcasting to utilize. For this system, there are two mediums of broadcasting: SMS advisory subscription and radio frequency overriding. This addresses the need of integrity for the marketing requirements of the system. If programmed correctly, the system should be able to determine which broadcasting method to utilize.

The fifth engineering requirement specifies the application of SMS advisory in the system. For all the 5 water level measures, the system must be able to send SMS alerts to the subscribers. This addresses the marketing requirement of the sensor's accuracy, and data and results must be acquired and delivered with minimal delay.

The sixth engineering requirement describes the approach of overriding FM radio frequency when the ultrasonic sensor detects flood water level has reached the last 2 emergency levels. The process of overriding FM radio frequency is through the use of SDR as an FM transmitter. This addresses the marketing needs that the system must be able to define boundaries of range for the overriding of radio frequency, send alerts through the FM radio for the last 2 emergency levels, and the alerts must be clear and audible enough.

The seventh engineering requirement discusses the methodology dealing with the radio frequency overriding. The system will configure SDR on GNU Radio to transmit and override signals. The marketing requirement requires that the alerts should be clear and audible enough and that the transmitted water level alerts should reach the target users within the specific range.

The eighth engineering requirement describes where the beacon will acquire its source of power. The beacon will be having a power bank as its source of power.

3) Design

The design requirements depict an overview of the whole system in terms of system components, hardware implementations, diagram workflows, and overall measurement and design of the prototype. The aesthetics of the prototype is designed to house the necessary modules to be used in constructing the beacon. The physical structure and setup of the beacon are preferably designed to be elevated to meet the requirements of the project. For the 3D modeling, the application used is TinkerCAD

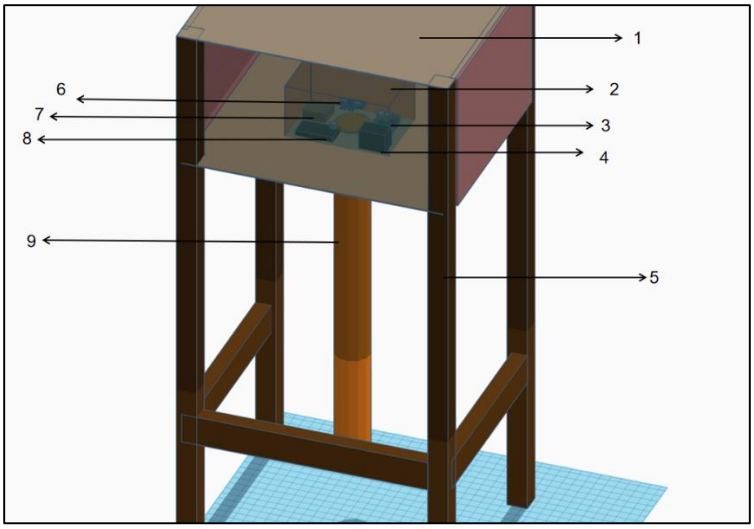


Figure 21. Prototype 3D Isometric View

Figure 21 shows the isometric view of the prototype. Labeled number 1 is the prototype box that houses the components and isolates the materials from extreme temperatures. 2 is the stand that elevates the components and modules above the ground. 3 is the IP-65 waterproof enclosure that houses the components and modules. Inside the enclosure, 4 is the Raspberry Pi; 5 is the power bank as the source; 6 is the ultrasonic sensor; LTE module; 8 is the circuit for the ultrasonic sensor, and 9 is a pipe for the ultrasonic slot.

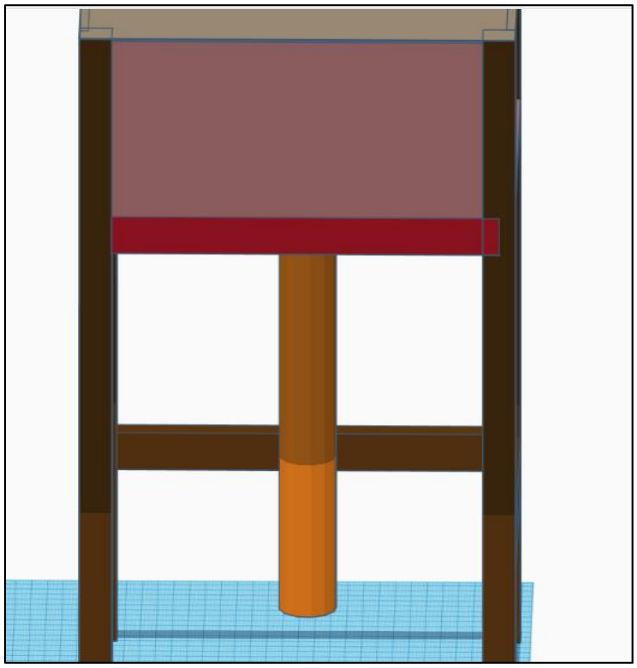


Figure 22. Prototype 2D Orthographic View: Front

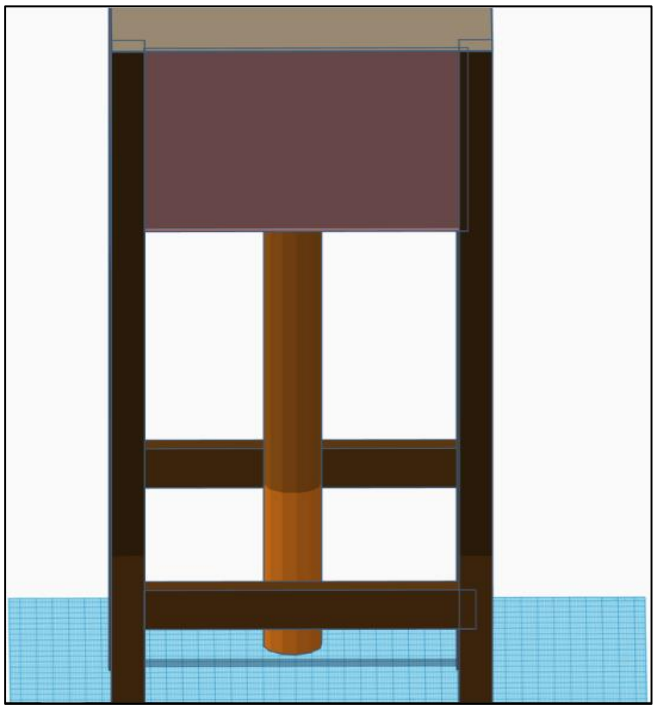


Figure 23. Prototype 2D Orthographic View: Side

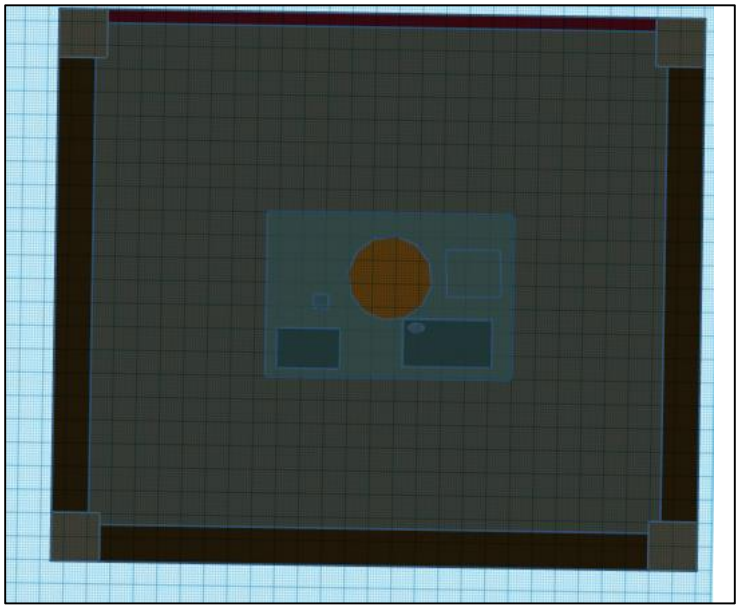


Figure 24. Prototype 2D Orthographic View: Bottom

Figure 25. Prototype 2D Orthographic View: Rear

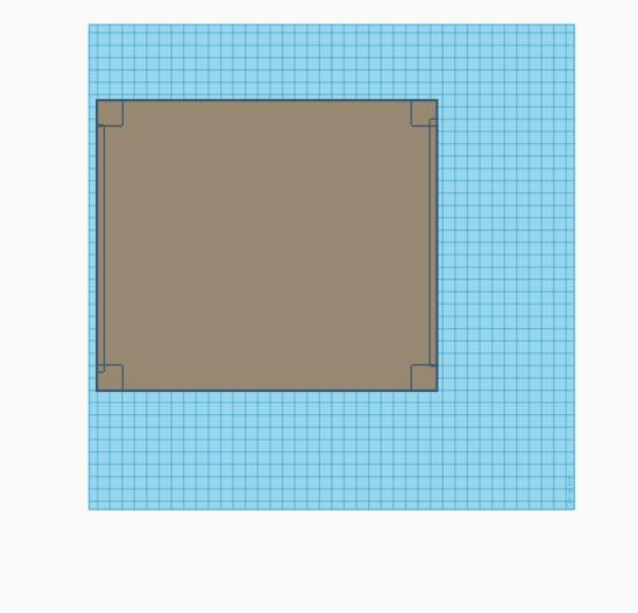


Figure 26. Prototype 2D Orthographic View: Top

TABLE III
TRADE-OFFS ANALYSIS BASED ON THE RELIABILITY OF THE DESIGN

Component	Description	Weight (a)	Cell Index Factor (b)	Score (c) a*b	Score Function (d) d=c*40%
DESIGN A					
Sensing Device	Ultrasonic Sensor	0.33	5	1.665	0.666
SDR	HackRF One	0.33	4	1.332	0.5328
Microcontroller	Raspberry Pi 3b+	0.33	4	1.332	0.5328
		0.99			**4.329**
DESIGN B					
Sensing Device	Float sensor	0.33	3	0.999	0.3996
SDR	RTL-SDR	0.33	3	0.999	0.3996
Microcontroller	Arduino Mega	0.33	4	0.5328	0.5328
		0.99			**3.33**
DESIGN C					
Sensing Device	Proximity Sensor	0.33	4	1.332	0.5328
SDR	Adalm Pluto	0.33	3	0.999	0.3996
Microcontroller	Arduino Uno	0.33	4	1.332	0.5328
		0.99			**3.663**

Table III shows the trade-offs analysis based on the reliability of the design. The comparison considers the 3 components namely: sensing device, SDR, and microcontroller used in the design. For the analysis of reliability, all three components are weighted similarly at 0.33 since these components function together and are dependent on each other in terms of their function as a whole in meeting the research objectives. The sensing device serves as the module that gathers data, SDR serves as the module that transmits into radio frequencies, and the microcontroller serves as the module that processes the data gathered by the sensing device and processes the different codes and scripts essential for the system to function. The cell index factor for each component is rated at 5 for 91-100, 4 for 81-90, 3 for 71-80, 2 for 51-70, and 1 for 0-50. The score column is computed by multiplying the weight to the cell index factor. Lastly, the score function column is obtained by multiplying the score with the weight of the trade-off analysis based on reliability which is 40%. The

score function per design is summed up to determine which is optimal out of the 3 designs, reliability wise. Among the three final designs to choose from, Design A garnered the highest score function out of the three with a score function of 4.329. It is then followed by Design C with a score function of 3.663, and Design B with a score function of 3.33.

TABLE IV
TRADE-OFFS ANALYSIS BASED ON THE FUNCTIONAL SUITABILITY OF THE DESIGN

Component	Description	Weight (a)	Cell Index Factor (b)	Score (c) a*b	Score Function (d) d=c*40%
DESIGN A					
Sensing Device	Ultrasonic Sensor	0.3	4	1.2	0.48
SDR	HackRF One	0.4	5	2	0.8
Microcontroller	Raspberry Pi 3b+	0.3	4	1.2	0.48
		1			**3.52**
DESIGN B					
Sensing Device	Float sensor	0.3	3	0.9	0.36
SDR	RTL-SDR	0.4	2	0.8	0.32
Microcontroller	Arduino Mega	0.3	2	0.6	0.24
		1			**1.84**
DESIGN C					
Sensing Device	Proximity Sensor	0.3	3	0.9	0.36
SDR	Adalm Pluto	0.4	3	1.05	0.48
Microcontroller	Arduino Uno	0.3	2	0.7	0.24
		1			**2.16**

Table IV shows the trade-offs analysis based on the functional suitability of the design. The comparison considers the 3 components namely: sensing device, SDR, and microcontroller used in the design. For the analysis of functional suitability, the SDR is selected as the top priority weighted at 0.40 as SDRs tend to be selective on the OS platform suitability. Followed by the sensing device and the microcontroller at similar weights of 0.30, since these two components are designed to work as a pair to function. The cell index factor for each component is rated at 5 for 91-100, 4 for 81-90, 3 for 71-80, 2 for 51-70,

and 1 for 0-50. The score column is computed by multiplying the weight to the cell index factor. Lastly, the score function column is obtained by multiplying the score with the weight of the trade-off analysis based on reliability functional suitability which is 40%. The score function per design is summed up to determine which is optimal out of the 3 designs, functional suitability wise. Among the three final designs to choose from, Design A garnered the highest score function out of the three with a score function of 3.52. It is then followed by Design C with a score function of 2.16, and Design B with a score function of 1.84.

TABLE V
TRADE-OFFS ANALYSIS BASED ON THE COST OF THE DESIGN

Component	Description	Weight (a)	Cell Index Factor (b)	Score (c) a*b	Score Function (d) d=c*10%
DESIGN A					
Sensing Device	Ultrasonic Sensor	0.3	5	1.5	0.3
SDR	HackRF One	0.35	2	0.7	0.14
Microcontroller	Raspberry Pi 3b+	0.35	4	1.4	0.28
		1			**2.88**
DESIGN B					
Sensing Device	Float sensor	0.3	4	1.2	0.24
SDR	RTL-SDR	0.35	4	1.4	0.28
Microcontroller	Arduino Mega	0.35	4	1.4	0.28
		1			**3.2**
DESIGN C					
Sensing Device	Proximity Sensor	0.3	5	1.5	0.3
SDR	Adalm Pluto	0.35	3	1.05	0.21
Microcontroller	Arduino Uno	0.35	4	1.4	0.28
		1			**3.16**

Table V shows the trade-offs analysis based on the cost of the design. The comparison considers the 3 components namely: sensing device, SDR, and microcontroller used in the design. For the analysis of cost, the SDR and microcontroller are selected as the top priority weighted at 0.35 respectively as these components tend to be expensive compared to the sensing device unit weighted as 0.30. The cell index factor for each component is rated at 5 for 91-100, 4 for 81-90, 3 for 71-80, 2 for 51-70, and 1 for 0-50. The score column is computed by multiplying the weight to the cell index factor. Lastly, the score function column is obtained by multiplying the score with the weight of the trade-off analysis based on the cost which is 20%. The score function per design is summed up to determine which is optimal out of the 3 designs, cost-effective wise. Among the three final designs to choose

from, Design B garnered the highest score function out of the three with a score function of 3.2. It is then followed by Design C with a score function of 3.16, and Design A with a score function of 2.88

Design Tradeoffs

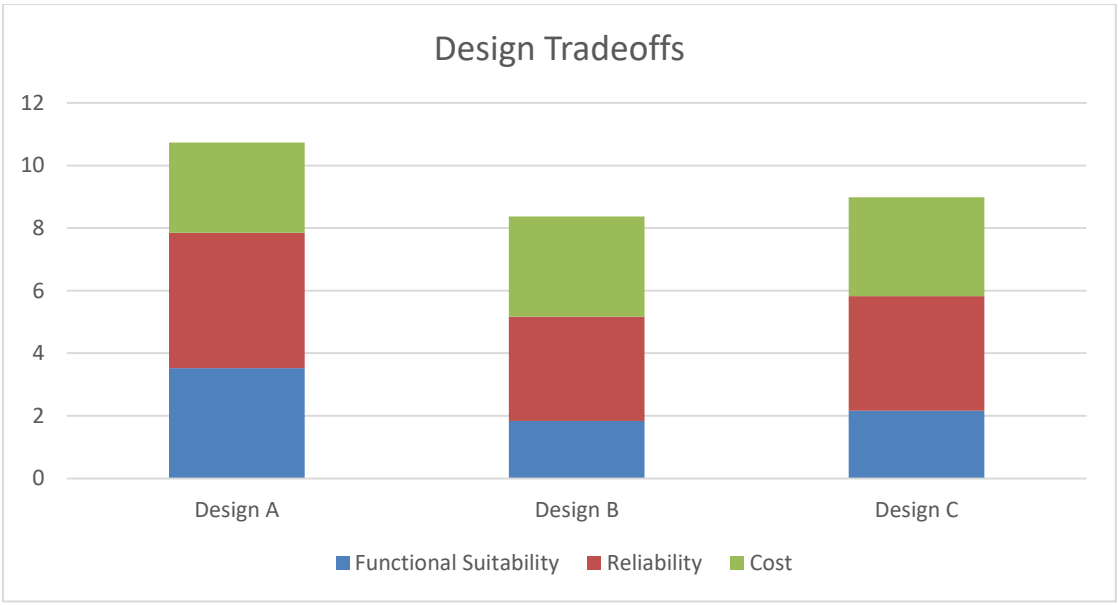


Figure 27. Design Tradeoffs

Figure 27 shows the total score obtained by the three designs in terms of their functional suitability, reliability, and cost. The scores were evaluated considering 40% for the functional suitability, 40% for the reliability, and 10% for the cost. Functional suitability and reliability are weighted at 40% and are the top priority in consideration for the design, with the cost only at 10%. In accordance with the weight matrix distribution of marketing requirements, it is important that the components used in this design are suitable and

functional with each other, for the system to be reliable. Functional suitability and reliability are prioritized compared to the cost.

After evaluation of each component, design A garnered the highest score among the three with a total score of 10.729. Design C came in second with a score of 8.983, and design B came in last with a score of 8.37.

Design A came in first as the most optimal design since it garnered the highest score function of 3.52 in its functional suitability evaluation and a score of 4.329 in its reliability evaluation—compared to design C with a total score of 8.983 which came in second with a score function of 2.16 in its functional suitability evaluation and a score function of 3.663 in its reliability evaluation. Design B came in third with a total score of only 8.37 and a score function of 1.84 in its functional suitability evaluation and a score of 3.33 in its suitability evaluation.

Design A came in short with the total score function of 2.88 for the cost evaluation compared to design B with a total score function of 3.2 and design C with a total score function of 3.16. But since the evaluation for the cost is the least prioritized among the three criteria for design tradeoffs, design A still came out the most optimal design for the flood monitoring designs that accurately detects flood water level.

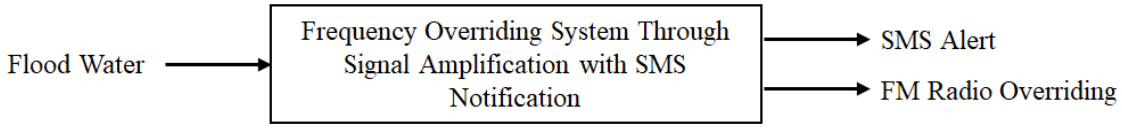


Figure 28. Level 0 Block Diagram of the Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification.

Figure 28 shows the level 0 block diagram of the Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification. It shows the level 0 input which is the flood water level data gathered by the ultrasonic sensor. The output is the two broadcasting mediums of the system: SMS alert and alerts received through radio.

TABLE VI
LEVEL 0 FUNCTIONAL REQUIREMENTS OF THE FREQUENCY OVERRIDING SYSTEM THROUGH SIGNAL AMPLIFICATION WITH SMS NOTIFICATION

Module	Frequency Overriding System Through Signal Amplification with SMS Notification
Inputs	- Data from flood water level
Outputs	- SMS Alert - Alert through FM radio
Functionality	The sensor accepts data from levels of floodwater. For all the 5 levels, the system will send SMS alerts to the SMS advisory subscribers. If the flood reaches the 4th and 5th level, then the alert message will be sent to radio listeners around the prototype.

Table VI shows the level 0 functional requirements of the Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification. The first row refers to the title of the system. The second row refers to the input of the system. The third refers to the output. The final row refers to the functionality of the system.

The input of the alert system is flood water data which acts as the fundamental input for the system. It shows that floodwater data is the fundamental input to the system. It serves as the input for triggering the flood alert system when the floodwater rises. The gathered data affects which would be the medium used for the output. The functionality is to precisely detect flood water and disseminate the detected flood water level in mobile phones and radio.

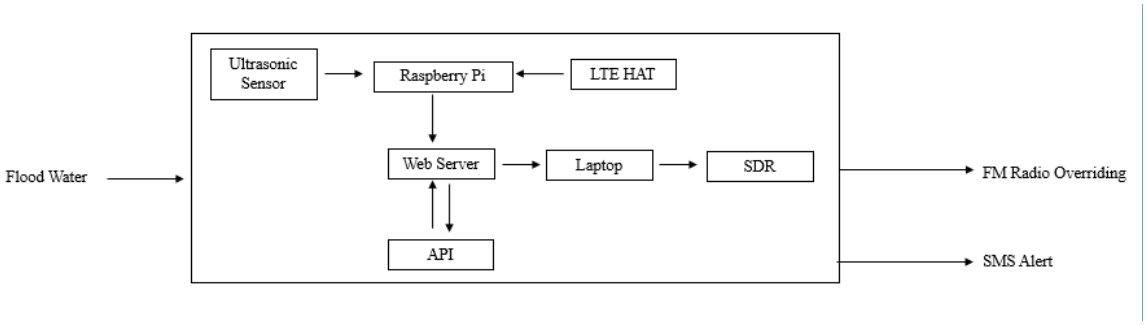


Figure 29. Level 1 Block Diagram of the Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification

Figure 29 shows the level 1 block diagram of the Frequency Overriding System Through Signal Amplification for Flood Alert and SMS Notification. The ultrasonic sensor will monitor the flood water level. For all the 5 levels, the Raspberry Pi will continuously send the data read from the ultrasonic sensor into the webserver via the Internet through the LTE module. The API will serve as the intermediary platform between the web server and the SMS advisory subscribers. If the floodwater reaches the last 2 levels, the data sent to the webserver will be processed through the SDR, the output will be audio advisory transmitted to FM radio frequency.

TABLE VII
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE ULTRASONIC SENSOR

Module	Ultrasonic Sensor
Inputs	- VCC, Ground, Trig, and Echo pinouts
Outputs	- Sends data to the Raspberry Pi; Data to be interpreted by the Raspberry Pi
Functionality	- Measures the flood water level.

Table VII refers to the functional requirements of the ultrasonic sensor. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The name of the module is the Ultrasonic sensor. Its input data will be coming from its pinouts. The output data is going to be interpreted by the Raspberry Pi. The functionality of this module is to measure the floodwater level.

TABLE VIII
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE LTE MODULE

Module	LTE Module
Inputs	- DC 5V (from Raspberry Pi USB Port) - Sim card - Rx, Tx (from Raspberry Pi GPIO pins)
Outputs	- LTE/4G Data
Functionality	- Enables the Raspberry Pi to be connected to the Internet network.

Table VIII refers to the functional requirements of the ultrasonic sensor. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The name of the module is the Ultrasonic sensor. Its input data will be coming from the flood water level it senses. The output data is going to be interpreted by the Raspberry Pi. The functionality of this module is to measure the floodwater level.

TABLE IX
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE SDR MODULE

Module	SDR Module
Inputs	- 5V (USB port)
Outputs	- Audio cue on radio frequency
Functionality	- Transmit programmable audio cue over FM radio frequency

Table IX shows the Level 1 functional requirements of the SDR Module. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The title of the module is the SDR Module. It has a DC input with 5V and data coming via a USB port. The output is to allow the device to send audio cue on FM radio frequency. The functionality of this module is to transmit audio cues over the FM radio frequency.

TABLE X
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE RASPBERRY PI 3B+

Module	Raspberry Pi 3b+
Inputs	- 5V (Micro USB Port)
Outputs	- Data interpreted from the ultrasonic sensor
Functionality	- Receives data from the ultrasonic sensor - Sends data to the webserver

Table X shows the Level 1 functional requirements of the Raspberry Pi 3b+. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The title of the module is Raspberry Pi 3b+. It has a DC input with a 5V source through the micro USB Port. The output is to interpret the data coming from the ultrasonic sensor and send data to the webserver. The functionality of this module is to receive data from the ultrasonic sensor, interpret data coming from the sensor, and send these data to the webserver.

TABLE XI
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE LAPTOP

Module	Laptop/PC
Inputs	- AC DC Adaptor 19.5V == 7.7A
Outputs	- 5V @ USB Port 3.0
Functionality	- Controls the SDR

Table XI shows the Level 1 functional requirements of the Laptop. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The title of the module is a laptop or PC. It has an AC input at 19.5V == 7.7A. The output is 5V at USB Port 3.0. The functionality of this is to control the SDR.

TABLE XII
LEVEL 1 FUNCTIONAL REQUIREMENTS OF THE POWERBANK

Module	Powerbank 10000mAh
Inputs	- 5V == 3A
Outputs	- 5V == 3A
Functionality	- Serves as the source for the Raspberry Pi.

Table XII shows the Level 1 functional requirements of the Powerbank. The first row refers to the title of the module. The second row refers to the input of the module. The third refers to the output. The final row refers to the functionality.

The title of the module is Powerbank. It has input and output of 5V 3A. The functionality of this module is to serve as the source for the Raspberry Pi.

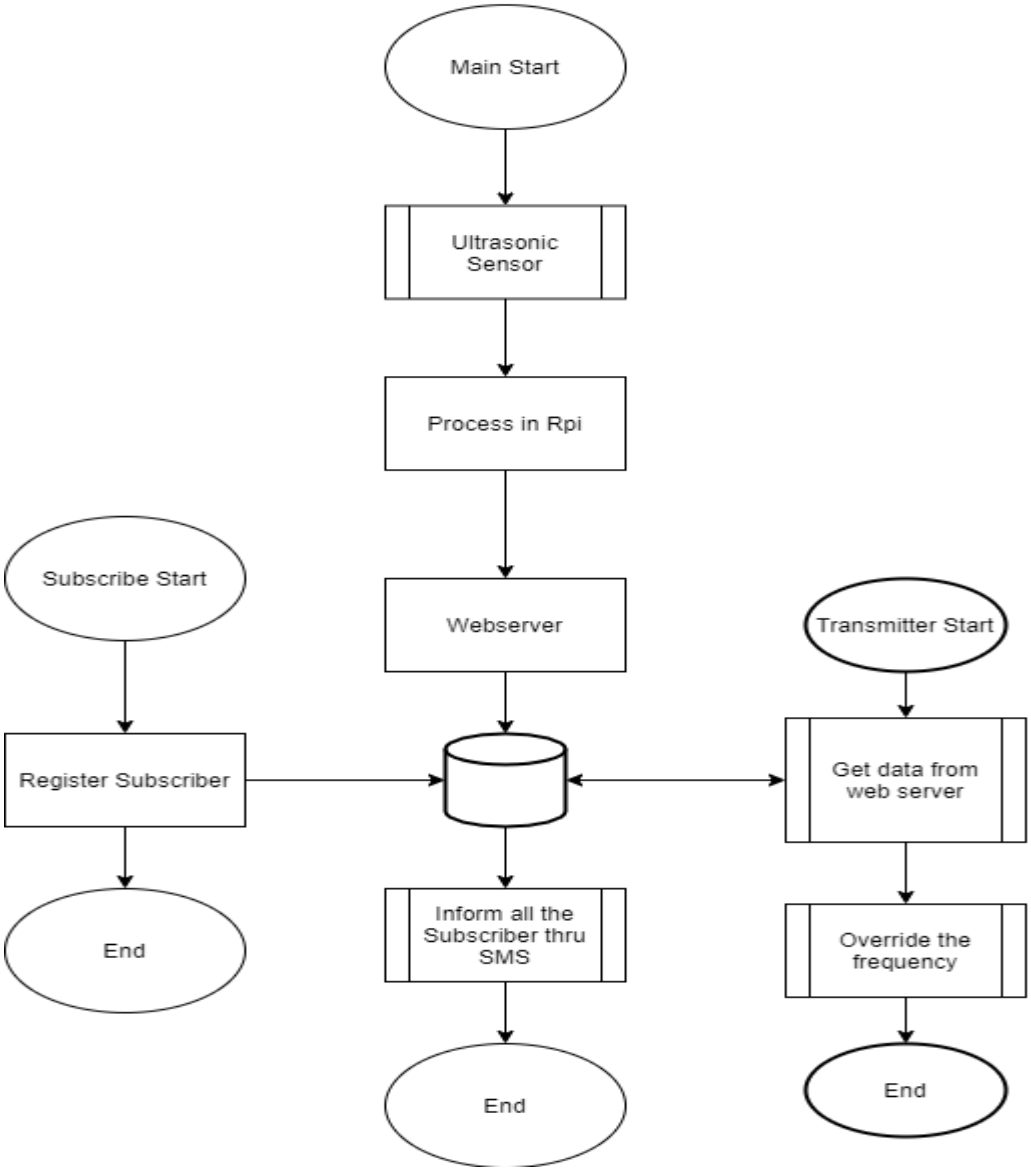


Figure 30. System Flowchart Diagram: Frequency Overriding System with SMS Notification

Figure 30 shows the general system flowchart diagram for this design project. Starting from the Ultrasonic module sensing data, Raspberry Pi processing and sending these data on to the web server, and on to the two branches of the mode of transmission: SMS and FM frequency transmission.

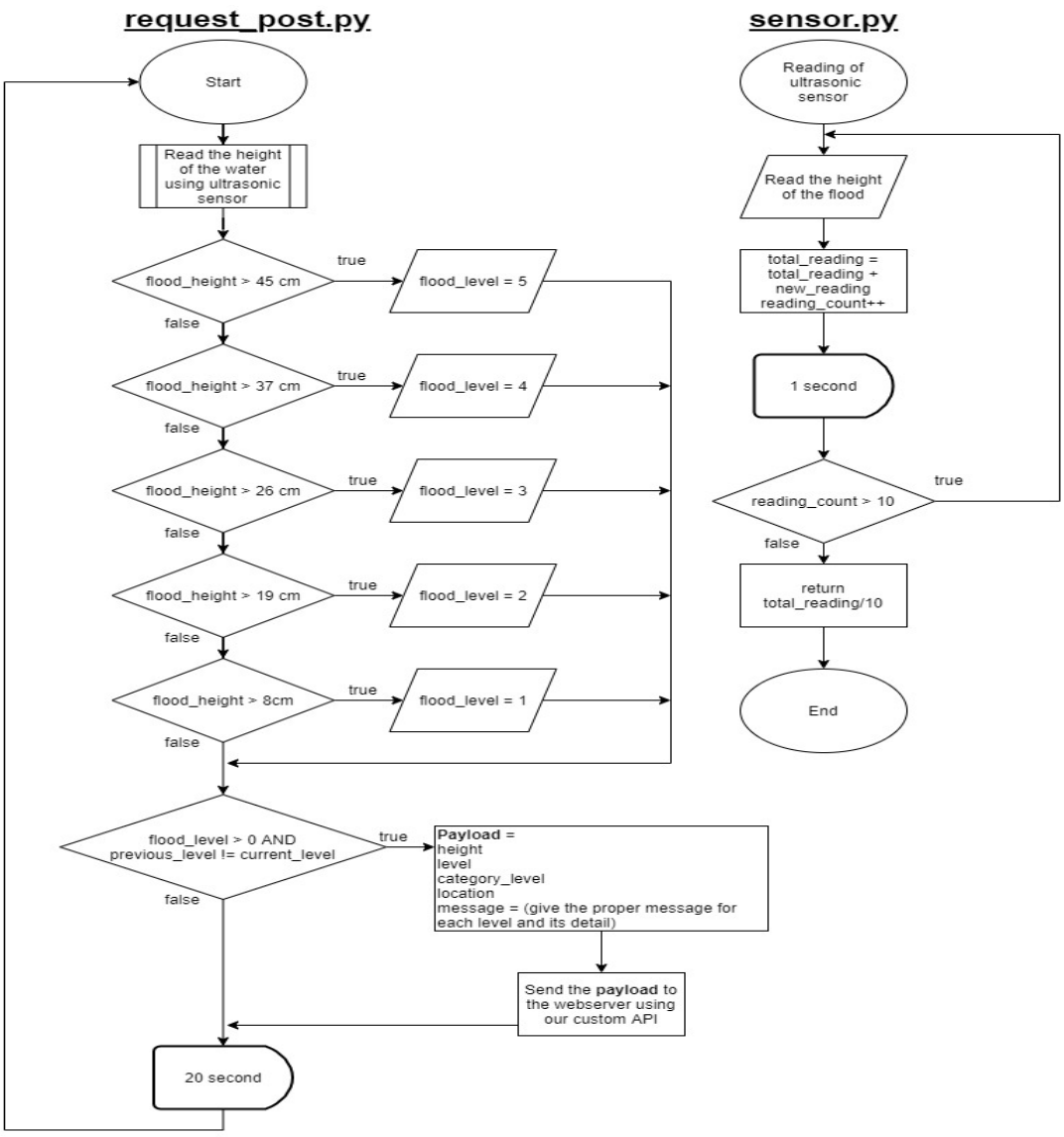


Figure 31. System Flowchart Diagram: Ultrasonic Sensor Data

Figure 31 shows the flowchart diagram of how the ultrasonic sensor module is programmed to acquire data and how this data is being passed on to the web server. This diagram shows the logic of the data flow.

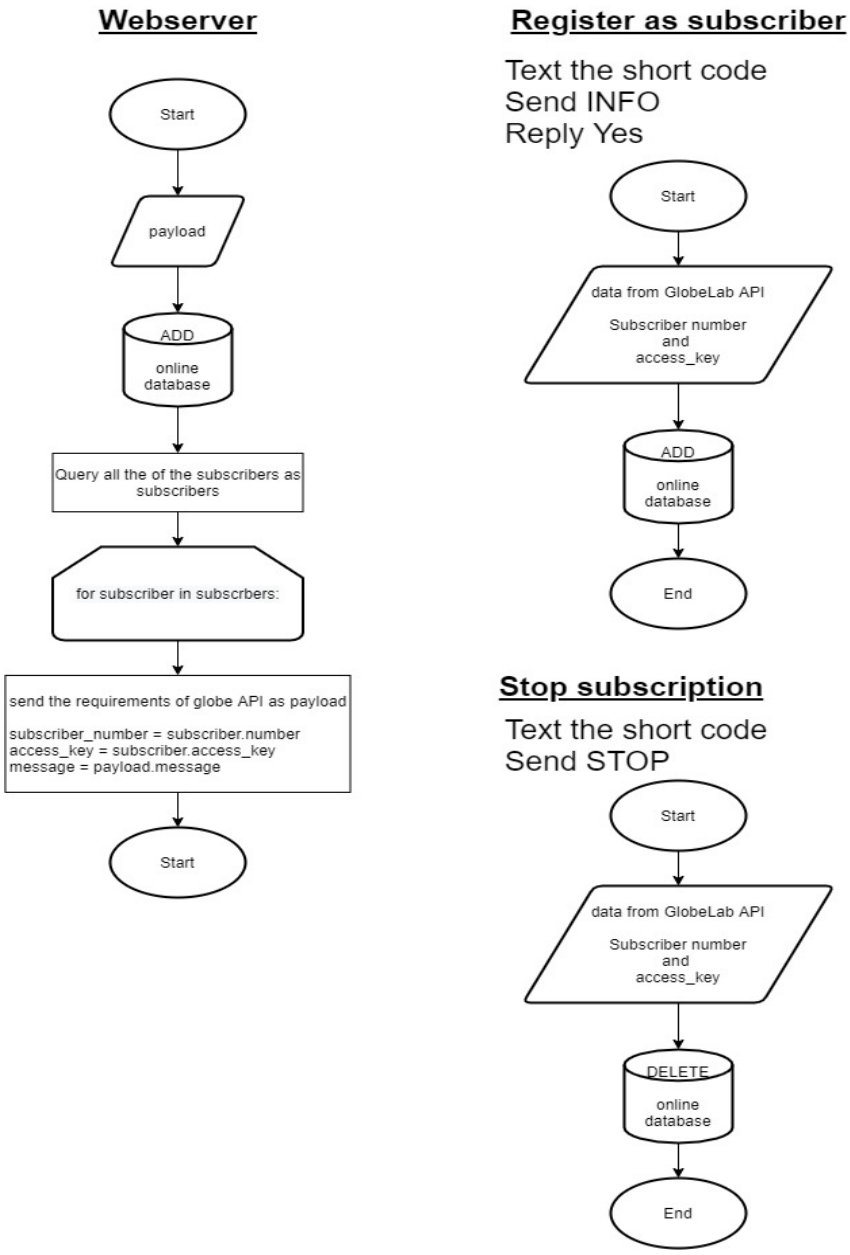


Figure 32. System Flowchart Diagram: SMS Advisory System

Figure 32 shows the system flowchart diagram for one of the medium of alert transmission used in this study: the SMS advisory system. This diagram shows how the data flows from the webserver and the passing of data between the system and the API.

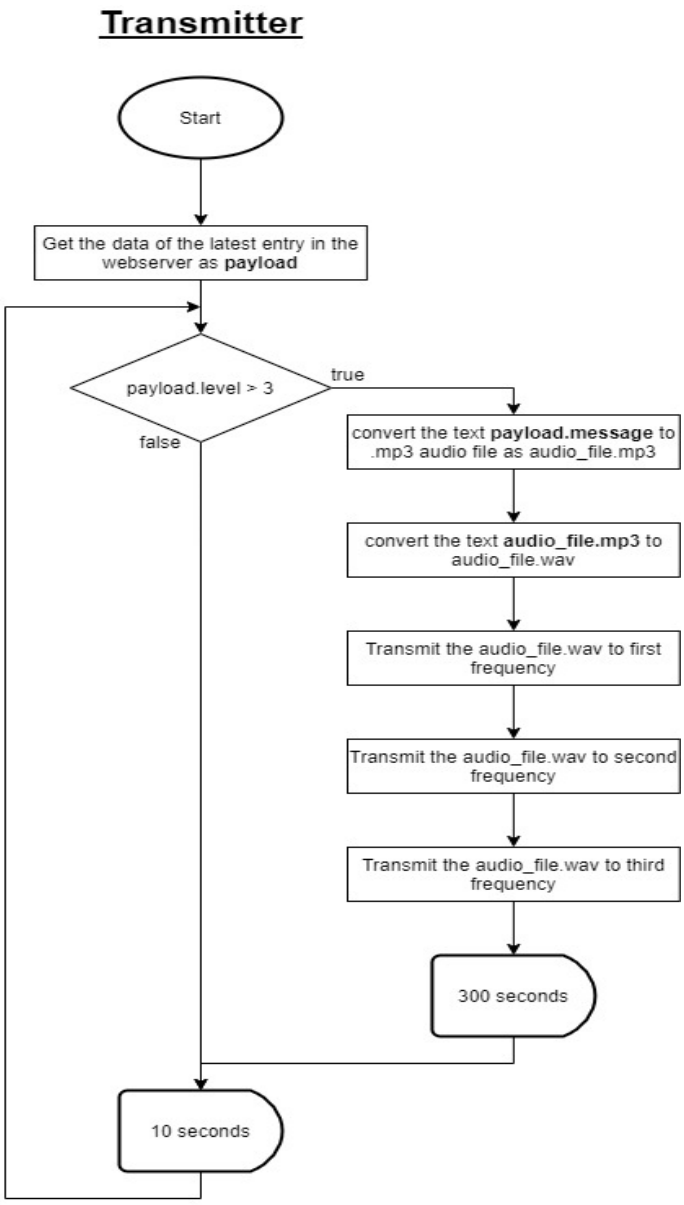


Figure 33. System Flowchart Diagram: FM Radio Overriding

Figure 33 shows the system flowchart diagram for another medium of alert transmission used in this study: the FM radio overriding. The flowchart shows the code logic of how the data is acquired from the web server and how the system interprets it. If

the data from the webserver indicates that the flood has reached level 4, it will trigger the FM radio overriding.

Figures 29-32 shows the main flow of the system. If the floodwater level reaches its trigger level, the data will be interpreted in the Raspberry Pi and sent to the web server, wherein it generates a text message of the flood water status and delivers the status to its subscribers. If the floodwater level reaches a critical level, the ultrasonic sensor will trigger the controller to transmit audio cues that contain data, time, level category, location, and exact measurement details extracted from the webserver. The acquired data will then be used to determine which medium of alert transmission will be used, either through SMS or FM radio overriding. For this study, the frequency of the first 3 FM radio stations will be overridden.

3) Verification and Testing

a. Unit Test

The unit test establishes that a system module performs a single unit of functionality to a prescribed specification. It tests the functionality of a system as a whole by inspecting its components.

Each of the modules was tested to see if there were defective components that needed to be replaced. The researchers individually checked every component used to create the hardware of the system.

TABLE XIII
TABLE FOR UNIT TESTING

Test ID	Description	Expected Result	Actual Result	Date of Testing
US-01	Checks the accuracy of the ultrasonic sensor for measuring distance	The overall test result should yield constant results upon repetitive tests to prove minimal to zero percent error regarding distance from the sensor to water measurement		
RP-01	Checks the ability of the Raspberry Pi to accept, interpret, and send out data	The overall test result is that the Raspberry Pi interprets the input water level and sends the data to the webserver depending on the height of the water.		
HAT-01	Checks the functionality of the HAT to access mobile data networks	The overall test should present blue LED blinking at specific intervals, indicating established Internet connection and data transmission/reception.		
SDR-01	Checks the ability of the SDR to override frequency channels and transmit signals	The overall test should present that the SDR can override and transmit radio signals and show a radio within radius receiving the sent signals from SDR.		

Table XIII shows the table that contains a list of Test IDs, descriptions, expected results, actual results, and dates of testing for a summary of the unit testing.

TABLE XIV

UNIT TEST OF THE ULTRASONIC SENSOR

Test Writer:		Bastian, Rochelle G.				
Test Case Name:		Ultrasonic Sensor Unit Test # 1			**Test ID#:**	US-01
Description:		Checks the accuracy of the ultrasonic sensor for measuring the distance to the water level.			**Type:**	☐ white box ☐ black box
Tester Information						
Name of Tester:		Jamoralin, Emmanuel E.			**Date:**	
Hardware Version:		Ultrasonic Sensor HC-SR04			**Time:**	
Setup:		The Vcc and ground wires of the Ultrasonic Sensor are connected to the 5V and GND pin of the Raspberry Pi. The Trig and Echo pins are wired to the Raspberry Pi's GPIO pins.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Set up the module.	The wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.				
2	Run the test program.	The sensor should return an accurate measurement of the water level.				
Overall test result:						

Table XIV shows the unit test details of the ultrasonic sensor. The component is wired to Raspberry Pi's Vcc, ground, and GPIO pins. For the test, the first step is setting the ultrasonic sensor module up. These wires should be correctly connected to a Vcc source, ground, and trig and echo GPIO pins. The next step would be running the test program. To determine if the module is wired and programmed properly, it should be able to return accurate measurements. The overall test result should yield constant results upon repetitive tests to prove minimal to zero percent error regarding distance from the sensor to water measurement.

TABLE XV

UNIT TEST OF THE RASPBERRY PI

Test Writer:		Bastian, Rochelle G.				
Test Case Name:		Raspberry Pi 3b+ Unit Test #2			**Test ID#:**	RP- 01
Description:		Checks the ability of the Raspberry Pi to accept, interpret, and send out data			**Type:**	☐ white box ☐ black box
Tester Information						
Name of Tester:		Jamoralin, Emmanuel E.			**Date:**	
Hardware Version:		Raspberry Pi 3b+			**Time:**	
Setup:		The Raspberry Pi is connected to a source.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Connect to the source.	The Raspberry Pi should boot.				
2	Run the test program.	Measurements should be read and interpreted into the system.				
Overall test result:						

Table XV shows the unit test details of the Raspberry Pi 3b+. For the test, the first step is setting the Raspberry Pi up by connecting it to a source. The microcomputer should boot up successfully without running into errors. The next step would be running the program. The measurements or water depth should be read and then be interpreted in the system. The overall test result is that the Raspberry Pi interprets the input water level and sends the data to the webserver depending on the height of the water. It specifies the frequency that the SDR will override.

TABLE XVI
UNIT TEST OF THE LTE HAT

Test Writer:		Chua, Raymond Carl P.				
Test Case Name:		LTE HAT Unit Test #3			**Test ID#:**	HAT-01
Description:		Checks the functionality of the HAT to access mobile data networks.			**Type:**	☐ white box ☐ black box
Tester Information						
Name of Tester:		Bayhon, Elaine Grace S.			**Date:**	
Hardware Version:		Raspberry Pi 3G/4G & LTE Base HAT			**Time:**	
Setup:		The LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through GPIO pins/Mini-USB.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Connect to the source.	When the module is powered up, the RED led should turn on.				
2	Configure the module.	The connection should be established and data should be transmitted/received. Blue led should blink at specific intervals.				
Overall test result:						

Table XVI shows the unit test details for the LTE Base HAT. The HAT is stacked on top of Raspberry Pi's GPIO pins and is connected to a source through GPIO pins/Mini-USB. For the test, the first step is connecting the module to the source. The red LED indicator should light up. The next step would be the module configuration. The connection should be established and data should be transmitted/received. As for indicators, blue LED should blink at specific intervals. The overall test should present blue LED blinking at specific intervals, indicating established Internet connection and data transmission/reception.

TABLE XVII

UNIT TEST OF THE SDR

Test Writer:		Chua, Raymond Carl P.				
Test Case Name:		SDR Unit Test #4			**Test ID#:**	SDR-01
Description:		Checks the ability of the SDR to override frequency channels and transmit signals.			**Type:**	☐ white box ☐ black box
Tester Information						
Name of Tester:		Bayhon, Elaine Grace S.			**Date:**	
Hardware Version:					**Time:**	
Setup:		The SDR is programmed using Linux OS.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N / A**	**Comments**
1	Configure the module.	The SDR should be recognized upon connecting through USB.				
2	Run the program.	The SDR should override frequency.				
3	Check signal transmission.	A radio should pick-up the sent audio signal				
Overall test result:						

Table XVII shows the unit test details of the SDR. The SDR is directly connected to the Raspberry Pi through a USB port. For testing, the first step is to configure the module. Upon successful configuration, the SDR should be recognized by the microcomputer. The next step would be running the test program. To test if the test program works, the SDR should be able to override frequency channels. The last step would be checking the signal transmission. A radio should be able to pick up the sent audio signal. The overall test should present that the SDR can override and transmit radio signals and show a radio within radius receiving the sent signals from SDR.

b. Integration Test

The integration test verifies the result of the integrated system. Integration tests are conducted after successful individual unit tests. Unit integration is of two or more components or modules that are supposed to function together for a specific task.

TABLE XVIII

TABLE FOR INTEGRATION TESTING

Test ID	Description	Expected Result	Actual Result	Date of Testing
Measure-IT-01	Checks if the ultrasonic sensor is wired to the Raspberry Pi correctly.	The overall test should yield accurate measurement results. If programmed correctly, the Raspberry pi should be able to interpret the measurement data acquired by the ultrasonic sensor.		
SMS-IT-02	Checks the capability of the LTE shield to establish mobile Internet connection for the Raspberry Pi	The overall test should show that through LTE HAT, the Raspberry Pi receives an Internet connection, and can send the data acquired by the ultrasonic sensor to the webserver.		
Audio-IT-03	Checks the ability of the SDR to send audio alerts when the measurement reaches alert levels 4 and 5.	The overall test should show that if the ultrasonic sensor acquires measurement data that reach alert levels 4 and 5, the SDR should override and transmit audio alerts to frequency channels.		

Table XVIII shows the table that contains a list of test IDs, descriptions, expected results, actual results, and dates of testing for a summary of the unit testing.

TABLE XXI

INTEGRATION TEST OF ULTRASONIC SENSOR AND RASPBERRY PI

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	Ultrasonic Sensor and Raspberry Pi Integration Test #1	**Test ID:**	Measure-IT-01
Description:	Checks if the ultrasonic sensor is wired to the Raspberry Pi correctly.	**Type:**	☐ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	The Vcc and ground pins of the ultrasonic sensor are wired to the 5V and GND pin of the Raspberry Pi. The trig and echo pins are wired to the Raspberry Pi's GPIO pins.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up modules.	The wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.			
2	Run the test program.	The sensor should return an accurate measurement of the water level.			
3	Loop the program.	The ultrasonic sensor should continuously acquire measurement level data for a set interval.			
Overall test result:					

Table XIX shows the integration test details of the ultrasonic sensor and Raspberry Pi. The component is wired to Raspberry Pi's Vcc, ground, and GPIO pins. For the test, the first step is setting the ultrasonic sensor module up. These wires should be correctly connected to a Vcc source, ground, trig, and echo GPIO pins. The next step would be running the test program. To determine if the module is wired properly and if the program codes run correctly, it should be able to return accurate measurements. The last step would be looping the program as long as the ultrasonic sensor senses flood, it should continuously

acquire data measurements. The overall test should yield accurate measurement results. If programmed correctly, the Raspberry Pi should be able to interpret the measurement data acquired by the ultrasonic sensor.

TABLE XX

INTEGRATION TEST OF ULTRASONIC SENSOR, RASPBERRY PI, AND LTE HAT

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Ultrasonic Sensor, Raspberry Pi, and LTE HAT Integration Test #2		**Test ID:**	SMS-IT-02	
Description:	Checks the capability of the LTE shield to establish mobile Internet connection for the Raspberry Pi		**Type:**	☐ white box ☐ black box	
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.		**Date:**		
Hardware Version:			**Time:**		
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.			
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.			
3	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.			
4	Run test program.	If Raspberry Pi is connected to the web server, it should be able to send alerts to the subscribers.			
4	Simulate water levels.	The sensor should return an accurate measurement of the water level.			
Overall test result:					

Table XX shows the integration test details of the ultrasonic sensor, Raspberry Pi, and LTE HAT. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. Right on top of the GPIO pins, the LTE HAT is stacked with the use of pin headers. For the test, the first step is setting the ultrasonic sensor module and Raspberry Pi up. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins. The next step would be setting up the LTE HAT. LTE HAT is stacked on the Raspberry Pi's GPIO pins and connected to the mini-USB port for the source. The next step would be booting the Raspberry Pi. Upon boot, it should automatically be connected to the Internet. Run the test program and then, the last step is simulating water levels. On this step, the Raspberry Pi already has access to the Internet and is, therefore, able to establish a connection to the webserver. The system should now be able to send SMS alerts to the subscribers every time the system reads data measurements for all the 5 levels. The overall test should show that through LTE HAT, the Raspberry Pi receives an Internet connection, and can send the data acquired by the ultrasonic sensor to the webserver.

TABLE XXI

INTEGRATION TEST OF ULTRASONIC SENSOR, RASPBERRY PI, LAPTOP, AND SDR

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	Ultrasonic Sensor, Raspberry Pi, and SDR Integration Test #1	**Test ID:**	Audio-IT-03
Description:	Checks the ability of the SDR to send audio alerts when the measurement reaches alert levels 4 and 5.	**Type:**	☐ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, SDR is connected to the Raspberry Pi through USB.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up the ultrasonic module and the Raspberry Pi. Boot Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins. Raspberry Pi should boot			
2	Set up Laptop	Boot Ubuntu in VMWare.			
3	Set up SDR.	Connect SDR to the laptop's USB port. The laptop should recognize SDR.			
4	Simulate water levels.	If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius.			
5	Check audio transmission.	Open GNURadio. A radio within the specified range of radius should pick up the audio sent by the SDR.			
Overall test result:					

Table XXI shows the integration test details of the ultrasonic sensor, Raspberry Pi, and SDR. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. the

test, the first step is setting up the ultrasonic sensor module and booting Raspberry Pi. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins. The second step would be setting up the laptop. Ubuntu should be booted using VMware. Next would be connecting the SDR to the laptop through the USB port. The next step would be simulating water levels. If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius. The last step is checking the audio transmission. A radio within the specified range of radius should pick up the audio sent by the SDR. The overall test should show that if the ultrasonic sensor acquires measurement data that reach alert levels 4 and 5, the SDR should override and transmit audio alerts to frequency channels.

c. Acceptance Test

The acceptance test verifies whether the system meets the requirement specification. It is conducted after the system has passed the integration test. The acceptance test is based on the engineering requirements stated in the previous chapter.

TABLE XXII

TABLE FOR INTEGRATION TESTING

Test ID	Description	Expected Result	Actual Result	Date of Testing
Measuring-AT-01	Utilize ultrasonic sensors that will accurately measure flood water levels.	The overall test result should show that the ultrasonic sensor is programmed correctly, and can acquire the depth of floodwater.		
Database-AT-02	Utilize the MySQL database.	The overall test result should show that the MySQL database is used to store subscribers' mobile numbers and flood level data measurements.		
Advisory-AT-03	Determine when to send SMS alerts and audio alerts.	The overall test result should show that the medium used to relay alerts are sending SMS and audio alerts.		
Audio-AT-04	Transmit audio alerts.	The overall test result should show that the system can transmit to radios within a specified radius if the ultrasonic sensor detects a flood that reaches the last 2 emergency levels.		
API-AT-05	Create SMS advisory system.	The overall test result should show that the Globe Labs API is used to develop the advisory system. The system will use API as an intermediary between the Web server and the users.		
SMS-AT-06	Send SMS alerts for all the water level readings.	The overall test result should show that for all the 5 level readings, the system should send SMS alerts to the advisory subscribers.		

Table XXII shows the table that contains a list of test IDs, descriptions, expected results, actual results, and dates of testing for a summary of the acceptance testing.

TABLE XXIII

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 1

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Measure Acceptance Test # 1		**Test ID:**	Measuring-AT-01	
Description:	Utilize ultrasonic sensors to allow the system to acquire flood level data by calculating the distance between floodwater and the sensor itself.		**Type:**	☐ white box ☐ black box	
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.		**Date:**		
Hardware Version:			**Time:**		
Setup:	The Vcc and ground pins of the ultrasonic sensor are wired to the 5V and GND pin of the Raspberry Pi. The Rx and Tx pins are wired to the Raspberry Pi's GPIO pins.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up an ultrasonic sensor and a Raspberry Pi.	Wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.			
2	Simulate water levels then run the test program.	The sensor should return an accurate measurement of the water level.			
3	Loop the program.	The ultrasonic sensor should continuously acquire measurement level data for a set interval.			
Overall test result:					

Table XXIII shows the acceptance test of the Measuring System. In step 1, the ultrasonic sensor must be set up with Raspberry Pi. Wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins. In step 2, run the test program together with a simulation for water levels. The sensor should return an accurate measurement of the water level. In step 3, the program must be looped for the system to continuously

receive measurement level data. The overall test result should show that the ultrasonic sensor is programmed correctly, and can acquire the depth of floodwater.

TABLE XXIV

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 2

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Database Acceptance Test # 2			**Test ID:**	Database-AT-02
Description:	Utilize the MySQL database.			**Type:**	☐ white box ☐ black box
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.			**Date:**	
Hardware Version:				**Time:**	
Setup:					
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Send INFO to the shortcode.	A message will be sent to the sender for confirmation			
2	After receiving a message, send YES to the shortcode.	The system should receive the confirmation message, granting the system the access token.			
3	The API will save the information of subscribers to the database.	The subscriber's details should be saved into the subscribers' table in the database			
4	Through post request, the API should be able to receive the data obtained by the sensor.	The data should be saved on the announcements table of the database. It should reflect on the web page			
Overall test result:					

Table XXIV shows the acceptance test of the Database System. In step 1, the end-users must text INFO to the shortcode. A message will be sent to the sender for information. In step 2, the end-user must send YES to the shortcode. Upon acceptance,

the system should receive the confirmation message, granting the system the access token. In step 3, the API will save the information of subscribers to the database. The subscriber's details should be saved into the subscribers' table in the database. In step 4, the API should be able to receive the data obtained by the sensor. The data should be saved on the announcements table of the database. It should reflect on the web page

The overall test result should show that the MySQL database is used to store subscribers' mobile numbers and flood level data measurements.

TABLE XXV

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 3

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	SMS and Audio Alerts Acceptance Test # 3	**Test ID:**	Advisory-AT-03
Description:	Determine when to send SMS alerts and audio alerts.	**Type:**	☐ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port. SDR is connected to the laptop through USB.		

Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.			
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.			
3	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.			
4	Set up SDR.	Connect SDR to the laptop's USB port. It should recognize the SDR.			
5	Simulate water levels.	The sensor should return an accurate measurement of the water level.			
6	Run test program.	If Raspberry Pi is connected to the Internet, it should be able to access the webserver for the subscribers' mobile numbers. The system should send SMS alerts.			
7	Check audio transmission.	If the simulated water level reaches levels 4 and 5, the system should transmit audio alerts as well, alongside with SMS alerts. A radio within the specified range of radius should pick up the audio sent by the SDR.			
Overall test result:					

Table XXV shows the acceptance test of the Advisory System. In step 1, the ultrasonic sensor and the Raspberry Pi must be set up. The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins. In step 2, set up the LTE HAT. Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink. In step 3, boot the Raspberry Pi. Upon boot, it should automatically have an Internet connection. In step 4, set up the SDR. Connect SDR to the laptop's USB port. The laptop should recognize SDR. In step 5, simulate water levels. The sensor should return an accurate measurement of the water level. In step 6, run the test program. If Raspberry Pi is connected to the Internet, it should be able to access the webserver for the subscribers' mobile numbers. The system should send SMS alerts. In the last step, check the audio transmission. If the simulated water level reaches levels 4 and 5, the system should transmit audio alerts as well, alongside with SMS alerts. A radio within the specified range of radius should pick up the audio sent by the SDR. The overall test result should show that the medium used to relay alerts are sending SMS and audio alerts.

TABLE XXVI

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 4

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	Audio Alerts Acceptance Test # 4	**Test ID:**	Audio-AT-04
Description:	Transmit audio alerts.	**Type:**	☐ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, SDR is connected to a laptop through USB.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up the ultrasonic module and the Raspberry Pi. Boot Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.			
2	Set up SDR.	Connect SDR to the laptop's USB port. The laptop should recognize SDR.			
3	Simulate water levels.	If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius.			
4	Check audio transmission.	A radio within the specified range of radius should pick up the audio sent by the SDR.			
Overall test result:					

Table XXVI shows the acceptance test details of audio transmission. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. the test, the first step is setting up the ultrasonic sensor module and booting Raspberry Pi. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins. The second step would be connecting the SDR to the laptop through the USB port. The third step would

be simulating water levels. If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius. The last step is checking the audio transmission. A radio within the specified range of radius should pick up the audio sent by the SDR. The overall test result should show that the system can transmit to radios within a specified radius if the ultrasonic sensor detects a flood that reaches the last 2 emergency levels.

TABLE XXVII

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 5

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Database Acceptance Test # 5		**Test ID:**	API-AT-05	
Description:	Create SMS advisory system.		**Type:**	☐ white box ☐ black box	
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.		**Date:**		
Hardware Version:			**Time:**		
Setup:					
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	The sensor detects flood level.	The flood level details will be sent to the web server through post request.			
2	The details will be sent to the API.	It should reflect on the home page of the web page.			
3	The API sends the message together with the subscriber's details to Globe Labs API.	The subscriber should receive the alert message.			
Overall test result:					

Table XXVII shows the acceptance test of the API System. In step 1, sensor detects flood level. The flood level details will be sent to the web server through post request. In step 2, The details will be sent to the API. It should reflect on the home page of the web page. In step 3, The API sends the message together with the subscriber's details to Globe Labs API. The subscriber should receive the alert message.

The overall test result should show that the Globe Labs API is used to develop the advisory system. The system will use API as an intermediary between the Web server and the users.

TABLE XXVIII

ACCEPTANCE TEST OF THE ENGINEERING REQUIREMENT 6

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	SMS Alerts Acceptance Test # 6			**Test ID:**	SMS-AT-06
Description:	Send SMS alerts for all the water level readings.			**Type:**	☐ white box ☐ black box
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.			**Date:**	
Hardware Version:				**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.			
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.			
3	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.			
4	Run test program.	The sensor should return an accurate measurement of the water level.			
4	Simulate water levels.	If Raspberry Pi is connected to the web server, it should be able to send alerts to the subscribers for all the 5 levels.			
Overall test result:					

Table XXVIII shows the acceptance test details of the SMS transmission. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. Right on top of the GPIO pins, the LTE HAT is stacked with the use of pin headers. For the test, step 1 is setting the ultrasonic sensor module and Raspberry Pi up. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins. Step 2 would be setting up the LTE HAT. LTE HAT is stacked on the Raspberry Pi's GPIO pins and connected to the mini-USB port for the source. Step 3 would be booting the Raspberry Pi. Upon boot, it should automatically be connected to the Internet. Run the test program and then, step 4 is simulating water levels. On this step, the Raspberry Pi already has access to the Internet and is, therefore, able to establish a connection to the webserver. The system should now be able to send SMS alerts to the subscribers every time the system reads data measurements for all the 5 levels. The overall test should show that through LTE HAT, the Raspberry Pi receives an Internet connection, and can send the data acquired by the ultrasonic sensor to the webserver.

Chapter IV.

RESULTS AND DISCUSSIONS

A. The Developed System

The system is developed through merging software applications and hardware components that each play specific functionalities. For the hardware part, different modules are assembled, all essential to function as a whole. Moreover, the comprising software applications bind the system as a whole by serving as interlink and providing support to the hardware components through programming.

The developed system is comprised of both hardware and software that provide certain functionality. The different modules and components are assembled to create the hardware of the system. Furthermore, the software was created using Python. The created program will be able to communicate with its beneficiaries utilizing mobile phones through SMS and radio. The program can run using a Raspberry Pi or any mini computers.

Prototype Model

Figure 34. Prototype Isometric View

Figure 35. Prototype Orthographic View: Front

Figure 36. Prototype Orthographic View: Back

Figure 37. Prototype Orthographic View: Bottom

Figure 38. Prototype Orthographic View: Top

Hardware Tools

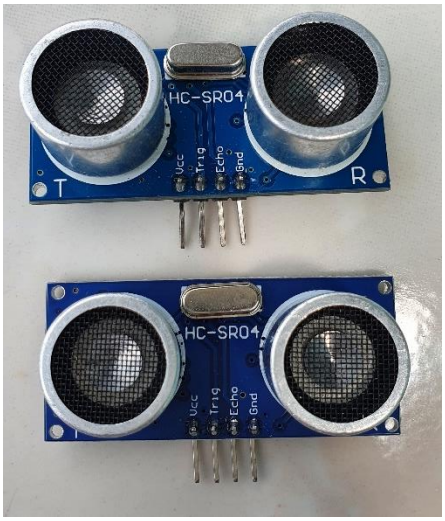


Figure 39. Ultrasonic Sensor HC – SR04

Figure 39 shows the sensor used in the prototype. Ultrasonic Sensor (HC-SR04) module is made up of an ultrasonic transmitter, receiver, and a control circuit. It has four

pins namely VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). Its operation voltage is 5 Volts DC, operating current of 15 mA, measure angle of 15°, and a ranging distance of 2cm to 4cm [29]. It is the module that is responsible for detecting flood water levels in the project. The ultrasonic sensor is programmed in a loop to get distance data from the sensor to the object continuously for 10 seconds before computing for its average.

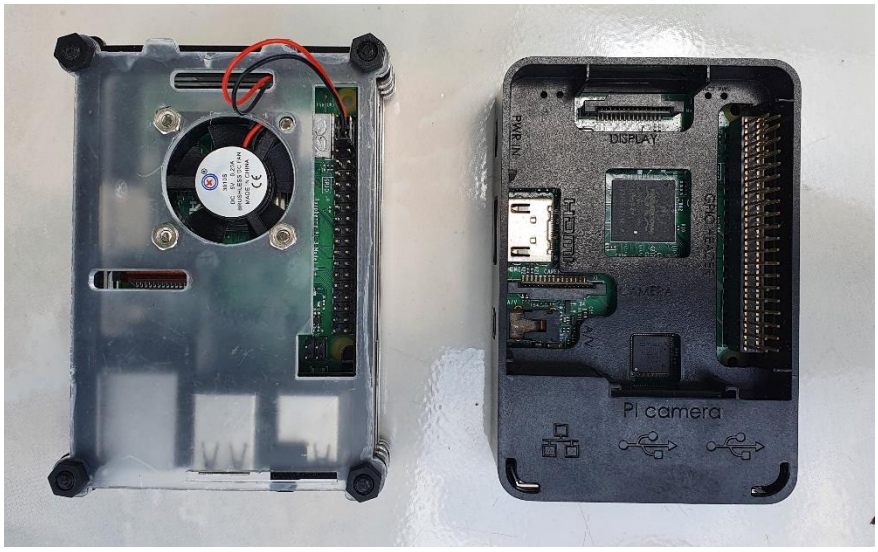


Figure 40. Raspberry Pi 3b+

Figure 40 shows the minicomputer used in the project--Raspberry Pi 3b+ model, the final revision in the Raspberry Pi 3 range. It runs on a 1.4GHz 64-bit quad-core processor and has dual-band wireless LAN, 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT) [23]. All the coded Python scripts that contain the logic and algorithm in between receiving and transmitting made in running the system are programmed using the Raspberry Pi.

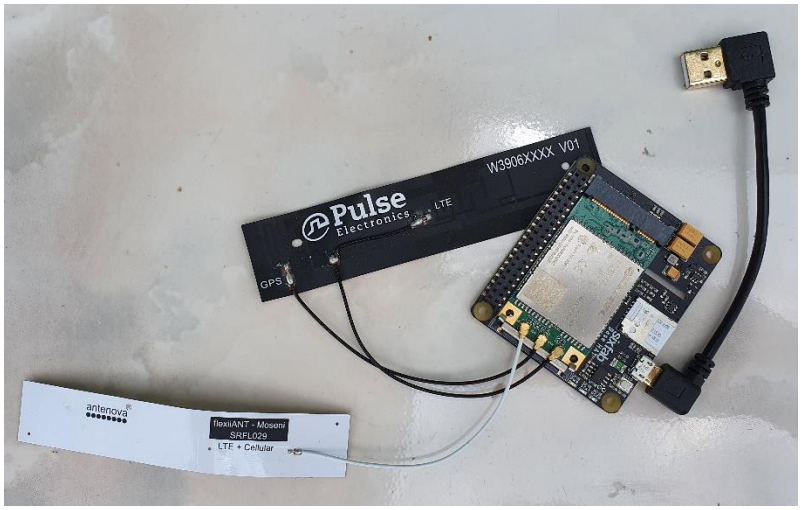


Figure 41. LTE Base HAT

Figure 41 shows the module used in establishing an internet connection for the Raspberry Pi in the project. This base HAT enables access to data networks on remote devices [22]. It is responsible for giving Raspberry Pi a high-speed internet connection that may use in sending and receiving data.

Figure 42. Powerbank

Figure 42 shows the power bank used in the prototype. The power bank serves as the source for the Raspberry Pi. It has an output of 5V and 3A.

Figure 43. HackRF One

Figure 43 shows the module that is used in transmitting signals to radio frequencies within a specific radius in the project. HackRF One is a half-duplex transceiver, with an operating frequency of 1MHz to 6GHz [60]. It is responsible for overriding radio signals and transmitting the audio alerts containing the date and time the flood level occurred, level category, location, and exact flood measurement data regarding flood water rising levels.

Software Tools

Python

The programming language used in this research. Python is a high-level scripting language that is easy to read and simple to implement [64]. This is used to create programs that interact with the different modules of the system. Those programs are uploaded to a web server

Python anywhere

Python anywhere is a web hosting service wherein coding, running, and hosting python apps and scripts are made possible [61]. This is where all data from the application of the project are stored. Also, it serves as the database of the system.

Globe Labs

Globe Labs allow developers to program their innovative applications, solutions, and services [64]. This is used as an API provider to send Flood alert SMS notification to all subscribers in the project.

GNU Radio

It is an open-source software development toolkit that enables developers to implement software radio operations and processes [21]. This is used to create a flowgraph that can enable to override FM radio signal and transmit audio alerts for high-level flood water level alert. The hardware to software implementation of a radio frequency transmission system is modified using GNU Radio.

Pycharm

PyCharm is an integrated development environment (IDE) specifically for Python [65]. This is the platform used to develop the application of the project.

Raspbian

Raspbian is an operating system based on Debian and is optimized for Raspberry Pi's hardware [68]. This is used as the Operating System of the Raspberry Pi in the project.

jQuery

jQuery is a small Javascript library that can be used for making UI more interactive [67]. This library is used for event handlings in the project.

D3.js

D3.js is also a JavaScript library that can create data visualization by combining HTML, CSS, and SVG [68]. This was used in plotting the line chart in the project.

Codes

```
@app.route('/globe/', methods=["GET"])
def opt_in():
    access_token = request.args.get("access_token")
    subscriber_number = request.args.get("subscriber_number")
    new_subscriber = Subscribers(access_token=access_token,
                                 subscriber_number=subscriber_number)

    if access_token is not None and subscriber_number is not None:
        Subscribers.save_subscriber(new_subscriber)

    subscribers = reversed(Subscribers.query.all())
    return render_template('subscribers.html', subscribers=subscribers, title='subscribers'), 200
```

Figure 44. API globe route with GET method

Figure 44 shows the codes for the API globe route with the GET method. This API route handles the incoming numbers and access tokens that the Globe API sends to the

developers. Its objective is to obtain numbers from the alert system subscribers then save it into the database. The mobile numbers of subscribers will be retrieved from the database once the system detects a flood level. The data that the Globe API sends in this route are query parameters.

```
@app.route('/globe/', methods=['POST'])
def stop_subscription():
    data = request.get_json()
    access_token = data['unsubscribed']['access_token']
    subscriber = Subscribers.query.filter_by(access_token=access_token).first()
    Subscribers.delete_subscriber(subscriber)
    subscribers = Subscribers.query.all()
    return render_template('subscribers.html', subscribers=subscribers, title='subscribers'), 200
```

Figure 45. API route globe with POST method

Figure 45 shows the codes for API route globe with POST method. Subscribing and unsubscribing have the same route but differ in the method. In the post method, Globe API will be sending us JSON data of the subscribers who do not want to receive alert messages from us. This route is responsible for deleting the subscriber's numbers from our database.

```
@app.route('/posts', methods=['POST'])
def posts():
    req_data = request.get_json()

    ActualHeight = format(req_data['ActualHeight'],  '.3f')
    Level = req_data['Level']
    CategoryLevel=req_data['CategoryLevel']
    FormatedDateTime = req_data['FormatedDateTime']
    Location = req_data['Location']
    MessageFromRpi = req_data['Message']

    data = Announcement(actualheight=ActualHeight,
                        level=Level,
                        message=MessageFromRpi,
                        categorylevel=CategoryLevel,
                        formateddatetime=FormatedDateTime,
                        location=Location)
    db.session.add(data)
    db.session.commit()

    if Level >= 0:
        subsribersID = Subscribers.query.all()
        for subscriber in subsribersID:
            outbound(Message = MessageFromRpi,
            access_token = subscriber.access_token,
            subscriber_number = subscriber.subscriber_number)


    return '''The Actual Height is {}.
    The Level is {}.
    The message: {}.
    Time Posted: {}.
    Category Level : {}.
    '''.format(ActualHeight, Level, MessageFromRpi, datetime.now, CategoryLevel)
```

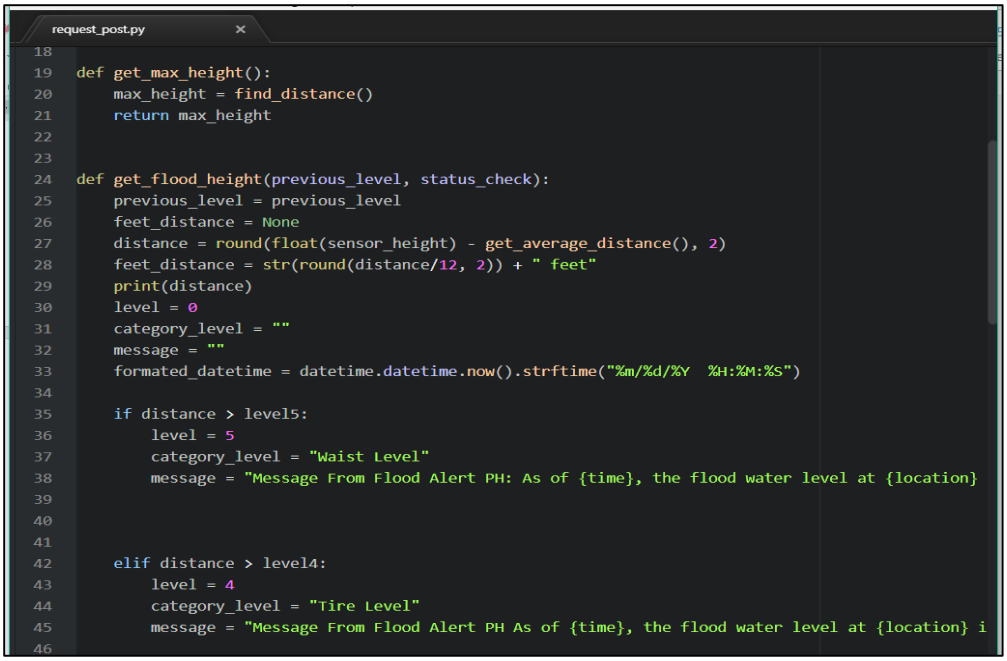
Figure 46. API route that accepts data from the sensor

Figure 46 shows the codes for the API route that accepts data from the sensor. This route displays the data that it receives from the sensor. It also calls the method for the system to send the alert message to the subscribers.

```
import requests

def outbound(Message, access_token, subscriber_number):


    shortcode = '21589064'
    access_token = access_token
    address = subscriber_number
    clientCorrelator = '21587128'
    message = Message

    url = "https://devapi.globelabs.com.ph/smsmessaging/v1/outbound/"+shortcode+"/requests"

    querystring = {"access_token":access_token}

    payload = "{\"outboundSMSMessageRequest\": { \"clientCorrelator\": \""+clientCorrelator+"\",
                \"senderAddress\": \""+shortcode+"\",
                \"outboundSMSTextMessage\": {\"message\": \""+message+"\"},
                \"address\": \""+address+"\" } }"

    headers = { 'Content-Type': "application/json", 'Host': "devapi.globelabs.com.ph" }

    response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

    # print(response.text)
    return response.text
```

Figure 47. Outbound method

Figure 47 shows the codes for the outbound method. This code includes the important details Globe API system needs to send the alert message to the subscribers. This method is being called on one of our API route.



```
request_post.py

def get_max_height():
    max_height = find_distance()
    return max_height


def get_flood_height(previous_level, status_check):
    previous_level = previous_level
    feet_distance = None
    distance = round(float(sensor_height) - get_average_distance(), 2)
    feet_distance = str(round(distance/12, 2)) + " feet"
    print(distance)
    level = 0
    category_level = ""
    message = ""
    formated_datetime = datetime.datetime.now().strftime("%m/%d/%Y  %H:%M:%S")

    if distance > level5:
        level = 5
        category_level = "Waist Level"
        message = "Message From Flood Alert PH: As of {time}, the flood water level at {location}



    elif distance > level4:
        level = 4
        category_level = "Tire Level"
        message = "Message From Flood Alert PH As of {time}, the flood water level at {location} i
```

Figure 48. Sending the flood level detected by the sensor

Figure 48 shows a snippet of the Python code used to acquire data from the ultrasonic sensor and the payload it sends to the API.

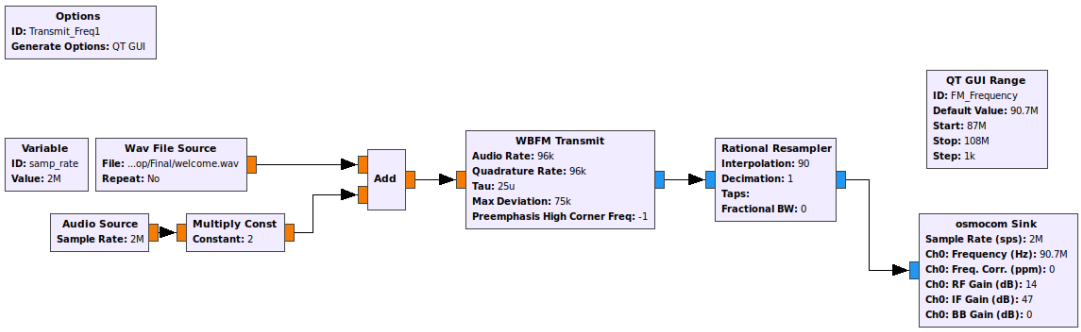


Figure 49. Flowgraph for Transmitting Audio Signals using GNU Radio

Figure 49 shows the hardware to software implementation flowchart for transmitting audio signals using the GNU Radio app. It overrides the radio frequency of nearby radio devices.

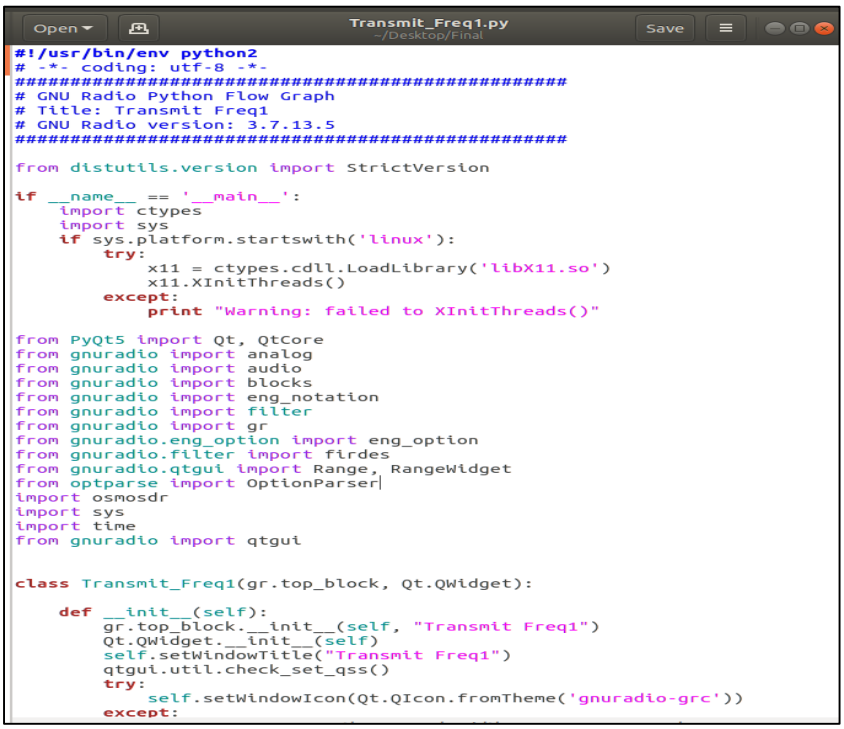


Figure 50. Python codes for Transmitting Audio Signals

Figure 50 shows the Python generated code for the flow graph shown in figure 49.

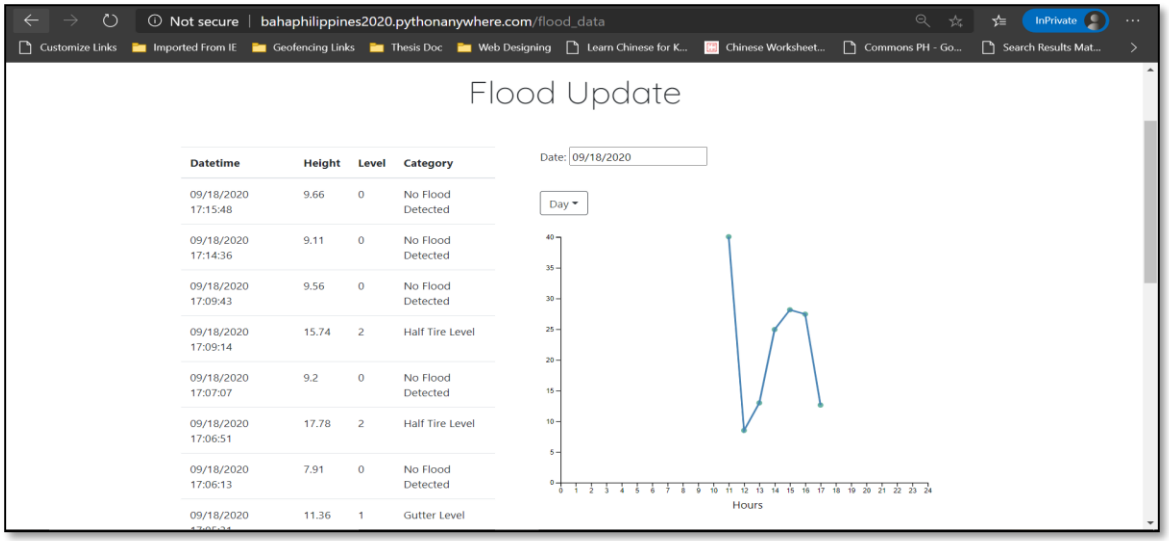


Figure 51. Home page

Figure 51 shows the data acquired from the beacon. The records of the flood detection and its line chart are included on this page.

B. Verification and Testing Result

a. Unit Testing

Each of the modules was tested to see if there were defective components that needed to be replaced. The researchers individually checked every component used to create the hardware of the system.

TABLE XXIX
UNIT TEST OF THE ULTRASONIC SENSOR

Test Writer:		Bastian, Rochelle G.				
Test Case Name:		Ultrasonic Sensor Unit Test # 1			**Test ID#:**	US-01
Description:		Checks the accuracy of the ultrasonic sensor for measuring the distance to the water level.			**Type:**	✓ white box ☐ black box
Tester Information						
Name of Tester:		Jamoralin, Emmanuel E.			**Date:**	January 18, 2020
Hardware Version:		Ultrasonic Sensor HC-SR04			**Time:**	
Setup:		The Vcc and ground wires of the Ultrasonic Sensor are connected to the 5V and GND pin of the Raspberry Pi. The trig and echo pins are wired to the Raspberry Pi's GPIO pins.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Set up the module.	The wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.	✓			The module was able to function well with the corresponding connection of Vcc, ground, trig, and echo GPIO pins.
2	Run the test program.	The sensor should return an accurate measurement of the water level.	✓			The module was able to measure based on the given algorithm of the equation.
Overall test result:			✓			The module was able to accurately and consistently measure the distance to the water level.

Table XXIX shows the unit test details of the ultrasonic sensor. The component is wired to Raspberry Pi's Vcc, ground, and GPIO pins.

For the test, step 1 is setting the ultrasonic sensor module up. These wires should be correctly connected to a Vcc source, ground, and trig and echo GPIO pins. Step 1: after 3 testings, in which 1 trial is failed and 2 trials are successful; the module was able to function well with the corresponding connection of Vcc, Tx, Rx, and GPIO pins.

Step 2 is running the test program. To determine if the module is wired and programmed properly, it should be able to return accurate measurements. Step 2: after 5 testings in which 2 trials are failed and 3 trials are successful; the module was able to measure based on the given algorithm of the equation.

The overall result showed the module was able to accurately and consistently measure the distance to the water level.

TABLE XXX
UNIT TEST OF THE RASPBERRY PI

Test Writer:		Bastian, Rochelle G.				
Test Case Name:		Raspberry Pi 3b+ Unit Test #2			**Test ID#:**	RP- 01
Description:		Checks the ability of the Raspberry Pi to accept, interpret, and send out data			**Type:**	✓ white box ☐ black box
Tester Information						
Name of Tester:		Jamoralin, Emmanuel E.			**Date:**	January 18, 2020
Hardware Version:		Raspberry Pi 3b+			**Time:**	
Setup:		The Raspberry Pi is connected to a power bank as a source.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Connect to the source.	The Raspberry Pi should boot.	✓			Raspberry Pi was able to boot properly and run the program successfully.
2	Run the test program.	Measurements should be read and interpreted in the system.	✓			Raspberry Pi was able to recognize the module and was able to interpret the detected measurement and send to the webserver
Overall test result:			✓			Raspberry Pi interprets the input water level and sends the data to the webserver depending on the height of the water

Table XXX shows the unit test details of the Raspberry Pi 3b+.

For the test, the step 1 is setting the Raspberry Pi up by connecting it to a source. The microcomputer should boot up successfully without running into errors. Step 1: after 1 testing, step 1 is successful; Raspberry Pi was able to boot properly and run the program successfully.

Step 2 is running the program. The measurements or water depth should be read and then be interpreted in the system. Step 2: after 5 testings in which 2 trials are failed

and 3 trials are successful; Raspberry Pi was able to recognize the module, interpret the detected measurement, and send it to the webserver.

The overall test result showed that Raspberry Pi interprets the input water level and sends the data to the webserver depending on the height of the water

TABLE XXXI
UNIT TEST OF THE LTE HAT

Test Writer:		Chua, Raymond Carl P.				
Test Case Name:		LTE HAT Unit Test #3			**Test ID#:**	HAT-01
Description:		Checks the functionality of the HAT to access mobile data networks.			**Type:**	✓ white box ☐ black box
Tester Information						
Name of Tester:		Bayhon, Elaine Grace S.			**Date:**	March 6, 2020
Hardware Version:		Raspberry Pi 3G/4G & LTE Base HAT			**Time:**	
Setup:		The LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is connected through Mini-USB				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Connect to the source.	When the module is powered up, the RED led should turn on.	✓			The LTE HAT module was able to boot up properly.
2	Configure the module.	The connection should be established and data should be transmitted/received. Blue led should blink at specific intervals.	✓			The LTE HAT module was able to establish an Internet connection.
Overall test result:			✓			The LTE HAT blinked specifying the establishment of an Internet connection.

Table XXXI shows the unit test details for the LTE Base HAT. The HAT is stacked on top of Raspberry Pi's GPIO pins and is connected to a source through GPIO pins/Mini-USB.

For the test, step 1 is connecting the module to the source. The red LED indicator should light up. Step 1: is successful after 1 testing; the LTE HAT module was able to boot up properly.

Step 2 is the module configuration. The connection should be established and data should be transmitted/received. As for indicators, blue LED should blink at specific intervals. Step 2: after 3 testings in which 2 trials are failed and 1 is successful; the LTE HAT module was able to establish an Internet connection.

The overall test result showed that the LTE HAT blinked specifying the establishment of an Internet connection.

TABLE XXXII
UNIT TEST OF THE SDR

Test Writer:		Chua, Raymond Carl P.				
Test Case Name:		SDR Unit Test #4			**Test ID#:**	SDR-01
Description:		Checks the ability of the SDR to override frequency channels and transmit signals.			**Type:**	✓ white box ☐ black box
Tester Information						
Name of Tester:		Bayhon, Elaine Grace S.			**Date:**	March 3, 2020
Hardware Version:					**Time:**	
Setup:		The SDR is programmed using Linux OS.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Configure the module.	The SDR should be recognized upon connecting through USB.	✓			The Raspberry Pi recognized the SDR.
2	Run the program.	The SDR should override frequency.	✓			The SDR was able to override the assigned frequency.
3	Check signal transmission.	A radio should pick-up the sent audio signal	✓			The radio received the audio sent by the SDR.
Overall test result:			✓			The SDR was able to override and transmit radio signals and show a radio within radius receiving the sent signals from SDR.

Table XXXII shows the unit test details of the SDR. The SDR is directly connected to the through a USB port.

For testing, step 1 is to configure the module. Step 1: after 2 testings in which 1 trial is failed and 1 trial is successful, the SDR is recognized by the laptop.

Step 2 is running the test program. Step 2: after 13 testings in which 6 trials are failed and 7 trials are successful, the SDR was able to override FM frequency channels.

Step 3 is checking the signal transmission. A radio should be able to pick up the sent audio signal. The overall test was SDR's able to override and transmit radio signals and show a radio within radius receiving the sent signals from SDR

b. Integration Test

Integration test verifies the result of the integrated system. Integration tests are conducted after successful individual unit tests. Unit integration is of two or more components or modules that are supposed to function together for a specific task.

TABLE XXXIII
INTEGRATION TEST OF ULTRASONIC SENSOR AND RASPBERRY PI

	Chua, Raymond Carl P.		
Test Case Name:	Ultrasonic Sensor and Raspberry Pi Integration Test #1	**Test ID:**	Measure-IT-01
Description:	Checks if the ultrasonic sensor is wired to the Raspberry Pi correctly.	**Type:**	✓ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	January 18, 2020
Hardware Version:		**Time:**	
Setup:	The Vcc and ground pins of the ultrasonic sensor are wired to the 5V and GND pin of the Raspberry Pi. The Rx and Tx pins are wired to the Raspberry Pi's GPIO pins.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up modules.	The wires should be connected to the correct Vcc, ground, trig, and echo GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, trig, and echo assigned GPIO pins
2	Run the test program.	The sensor should return an accurate measurement of the water level.	✓		The sensor module was able to measure the accurate water level and able to return the result instantly.
3	Loop the program.	The ultrasonic sensor should continuously acquire measurement level data for a set interval.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level
Overall test result:			✓		The sensor module was able to yield accurate measurement results. It was programmed correctly and the Raspberry pi was able to interpret the measurement data acquired by the ultrasonic sensor.

Table XXXIII shows the integration test details of the ultrasonic sensor and the Raspberry Pi. The component is wired to Raspberry Pi's Vcc, ground, and GPIO pins.

For the test, step 1 is setting the ultrasonic sensor module up. These wires should be correctly connected to a Vcc source, ground, trig, and echo GPIO pins. Step 1: after

1 successful testing; the module was able to function well with its current connection of VCC, Ground, trig, and echo assigned GPIO pins.

Step 2 running the test program. To determine if the module is wired properly and if the program codes run correctly, it should be able to return accurate measurements. Step 2: after 5 testings in which 2 trials are failed and 3 trials are successful; the sensor module was able to measure accurate water level and able to return the result instantly.

Step 3 is looping the program. As long as the ultrasonic sensor senses flood, it should continuously acquire data measurements. Step 3: after 11 trials in which 3 trials are failed and 8 are successful; the ultrasonic sensor should continuously acquire measurement level data for a set interval.

The overall test result showed the sensor module was able to yield accurate measurement results. It was programmed correctly and the Raspberry pi was able to interpret the measurement data acquired by the ultrasonic sensor.

TABLE XXXIV
INTEGRATION TEST OF ULTRASONIC SENSOR, RASPBERRY PI, AND LTE HAT

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	Ultrasonic Sensor, Raspberry Pi, and LTE HAT Integration Test #2	**Test ID:**	SMS-IT-02
Description:	Checks the capability of the LTE shield to establish mobile Internet connection for the Raspberry Pi	**Type:**	✓ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	March 6, 2020
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, trig and echo GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, Echo, and Trig assigned GPIO pins
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.	✓		The LTE HAT module was able to boot up properly. It was configured correctly to auto-connect upon startup.
3	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.	✓		Raspberry Pi was able to boot properly and was able to connect to the internet automatically.
4	Simulate water levels.	The sensor should return an accurate measurement of the water level.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level
5	Run test program.	If Raspberry Pi is connected to the web server, it should be able to send alerts to the subscribers.	✓		Raspberry Pi was able to recognize the module and was also able to interpret the detected measurement and send to the webserver
Overall test result:			✓		The result shows that through LTE HAT, the Raspberry Pi receives an Internet connection, and can send the data acquired by the ultrasonic sensor to the webserver.

Table XXXIV shows the integration test details of the ultrasonic sensor, Raspberry Pi, and LTE HAT. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. Right on top of the GPIO pins, the LTE HAT is stacked with the use of pin headers.

For the test, step 1 is setting the ultrasonic sensor module and Raspberry Pi up. The wires should be correctly connected to a Vcc source, ground, echo, and trig GPIO pins. Step 1: after 1 testing is successful; the module was able to function well with its current connection of VCC, Ground, Echo, and Trig assigned GPIO pins.

Step 2 is setting up the LTE HAT. LTE HAT is stacked on the Raspberry Pi's GPIO pins and connected to the mini-USB port for the source. The blue LED indicator should blink. Step 2: after 3 testings in which 2 trials are failed and 1 is successful; the LTE HAT module was able to boot up properly. It was configured correctly to auto-connect upon startup.

Step 3 is booting the Raspberry Pi. Upon boot, it should automatically be connected to the Internet. Step 3: after 2 testings in which 1 trial is failed and 1 is successful; Raspberry Pi was able to boot properly and was able to connect to the internet automatically.

Step 4 is to run the test program. Step 4: after 8 testings in which 3 trials are inaccurate and 5 trials are successful, the sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level.

Step 5 is simulating water levels. On this step, the Raspberry Pi already has access to the Internet and is, therefore, able to establish a connection to the webserver. Step 5: after 17 testings in which 5 trials are failed due to web server access, 12 trials are successful; Raspberry Pi was able to recognize the module and was also able to interpret the detected measurement and send it to the webserver.

The overall test result showed that through LTE HAT, the Raspberry Pi receives an Internet connection, and can send the data acquired by the ultrasonic sensor to the webserver.

TABLE XXXV
INTEGRATION TEST OF ULTRASONIC SENSOR, RASPBERRY PI, LAPTOP, AND SDR

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	Ultrasonic Sensor, Raspberry Pi, and SDR Integration Test #3	**Test ID:**	Audio-IT-03
Description:	Checks the ability of the SDR to send audio alerts when the measurement reaches alert levels 4 and 5.	**Type:**	✓ white box ☐ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	March 2020
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, SDR is connected to the Raspberry Pi through USB.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up the ultrasonic module and the Raspberry Pi. Boot Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins. Raspberry Pi should boot	✓		The module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins
2	Set up a Laptop.	Boot Ubuntu in VMWare.	✓		The Ubuntu Linux Virtual Machine should boot.
3	Set up SDR.	Connect SDR to the Raspberry Pi's USB port. The laptop should recognize SDR.	✓		The laptop was able to recognize the SDR.
4	Simulate water levels.	If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius.	✓		The FM radios inside the specified radius were overridden.
5	Check audio transmission.	Open GNURadio. A radio within the specified range of radius should pick up the audio sent by the SDR.	✓		The radio was able to pick up audio sent by the SDR.
Overall test result:					

Table XXXV shows the integration test details of the ultrasonic sensor, Raspberry Pi, and SDR. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins.

For the test, step 1 is setting up the ultrasonic sensor module and booting Raspberry Pi. After 1 successful testing, the wires are correctly connected to a Vcc source, ground, and Rx Tx GPIO pins.

Step 2 is setting up the laptop. After 7 testings in which 3 are failed due to OS issues and 4 trials are successful, the VMware Ubuntu Linux ran successfully on the laptop.

Step 3 is connecting the SDR to the laptop through the USB port. After 2 testings in which 1 is failed due to dependencies and 1 is successful, the laptop recognized the SDR.

Step 4 is simulating water levels. After 17 testings in which 4 trials are failed and 13 trials are successful, the SDR can override radio frequency around the radius when the water simulation reaches levels 4 and 5.

Step 5 is checking the audio transmission. After 8 testings in which 3 trials are failed and 5 trials are successful, radios within the specified range of radius were able to pick up the audio sent by the SDR.

The overall test showed that if the ultrasonic sensor acquires measurement data that reach alert levels 4 and 5, the SDR should override and transmit audio alerts to frequency channels.

c. Acceptance Test

The acceptance test verifies whether the system meets the requirement specification. It is conducted after the system has passed the integration test. The acceptance test is based on the engineering requirements stated in the previous chapter.

TABLE XXXVI
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 1

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Measure Acceptance Test #1		**Test ID:**	Measuring-AT-01	
Description:	Utilize ultrasonic sensors to allow the system to acquire flood level data by calculating the distance between flood water and the sensor itself.		**Type:**	✓ white box ☐ black box	
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.		**Date:**	January 18, 2020	
Hardware Version:			**Time:**		
Setup:	The Vcc and ground pins of the ultrasonic sensor are wired to the 5V and GND pin of the Raspberry Pi. The Rx and Tx pins are wired to the Raspberry Pi's GPIO pins.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up an ultrasonic sensor and a Raspberry Pi.	Wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins.
2	Simulate water levels then run the test program.	The sensor should return accurate measurements of the water level.	✓		The sensor module was able to measure the accurate water level and able to return the result instantly.
3	Loop the program.	The ultrasonic sensor should continuously acquire measurement level data for a set interval.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level.
Overall test result:			✓		The sensor module was able to yield accurate measurement results. It was programmed to calculate the distance between the floodwater and the sensor.

Table XXXVI shows the acceptance test of the Measuring System.

For step 1, the ultrasonic sensor must be set up with Raspberry Pi. Wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins. Step 1: after 1 successful testing, the module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins.

In step 2, run the test program together with a simulation for water levels. The sensor should return an accurate measurement of the water level. Step 2: after 17 testings in which 4 trials are failed and 13 trials are successful; the sensor module was able to measure the accurate water level and able to return the result instantly. Testing of the actual program for prototype demonstration underwent multiple runs and code modifications—optimization of codes for the ultrasonic sensor to measure the difference in theoretical and experimental measurements.

In step 3, the program must be looped for the system to continuously receive measurement level data. Step 3: after 1 successful trial as it only requires looping of codes, the sensor module was able to yield accurate measurement results. It was programmed to calculate the distance between the floodwater and the sensor.

The overall test result showed the sensor module was able to yield accurate measurement results. It was programmed to calculate the distance between the floodwater and the sensor. Successful overall testing of Measure Acceptance Test #1 meets the engineering requirement of utilizing ultrasonic sensors that will accurately measure flood water levels.

TABLE XXXVII
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 2

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Database Acceptance Test #2			**Test ID:**	Database-AT-02
Description:	Utilize the MySQL database.			**Type:**	✓ white box ☐ black box
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.			**Date:**	March 7, 2020
Hardware Version:				**Time:**	
Setup:	A mobile phone is needed in this testing. There should be a route in the API to receive the data from the sensor.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Send INFO to the shortcode	A message will be sent to the sender for confirmation	✓		A message was sent for confirmation of subscription.
2	After receiving a message, send YES to the shortcode	They should receive a message	✓		Replying to YES confirms the subscription.
3	The API will save the information of subscriber to the database	The subscriber's details should be saved into the subscribers' table in the database	✓		The subscriber's data were saved successfully in the database.
4	Through post request, the API should be able to receive the data the sensor was able to obtain	The data should be saved on the announcements table of the database. It should reflect on the web page	✓		The data sent from the sensor were saved successfully in the database. It also reflected on the web page.
Overall test result:			✓		The overall test result showed that the MySQL database is used to store subscribers' mobile numbers and flood level data measurements.

Table XXXVII shows the acceptance test of the Database System. In connection to the acceptance test #1, acceptance test #2 of database system is in correlation with the data gathered in the first acceptance test, whether the data acquired by the ultrasonic sensor can be sent and stored in a MySQL database.

In step 1, the end-users must text INFO to the shortcode. A message will be sent to the sender for information. Step 1: after 1 successful testing, a message was sent for confirmation of subscription.

In step 2, the end-user must send YES to the shortcode. Upon acceptance, the system should receive the confirmation message, granting the system the access token. Step 2: after 9 testings in which 4 trials are failed and 5 trials are successful, replying 'yes' confirms the subscription, unless Globe Labs API failed to send the mobile numbers to the API.

In step 3, the API will save the information of subscribers to the database. The subscriber's details should be saved into the subscribers' table in the database. Step 3: after 4 testings in which 2 trials are failed are successful, the subscriber's data were saved successfully in the database. The challenges in attaining the subscriber's information are giving Globe Labs API the wrong URI and failing to retrieve the data that was being sent to the API by Globe Labs API due to some errors in the codes.

In step 4, the API should be able to receive the data obtained by the sensor. The data should be saved on the announcements table of the database. It should reflect on the web page. Step 4: after 5 trials in which 2 trials are failed and 3 trials are successful, the data sent from the sensor were saved successfully in the database. It also reflected on the web page.

The overall test result showed that the MySQL database is used to store subscribers' mobile numbers and flood level data measurements. Successful overall testing of Database Acceptance Test #2 meets the engineering requirement that the database should be developed using MySQL database.

TABLE XXXVIII
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 3

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	SMS and Audio Alerts Acceptance Test #3	**Test ID:**	Advisory-AT-03
Description:	Determine when to send SMS alerts and audio alerts.	**Type:**	☐ white box ✓ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port. SDR is connected to the Raspberry Pi through USB.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.	✓		The LTE HAT module was able to boot up properly. It was configured correctly to auto-connect upon startup.
3	Set up SDR.	Connect SDR to the Raspberry Pi's USB port. The Raspberry Pi should recognize SDR.	✓		The Raspberry Pi was able to recognize the SDR.
4	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.	✓		Raspberry Pi was able to boot properly and was able to connect to the internet automatically.
5	Simulate water levels.	The sensor should return an accurate measurement of the water level.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a change of level category.
6	Run test program.	If Raspberry Pi is connected to the Internet, it should be able to access the webserver for the subscribers' mobile numbers. The system should send SMS alerts. If the water level reached level 4 and 5, the system should also trigger FM overriding as well.	✓		Raspberry Pi was able to recognize the module and was also able to interpret the detected measurement and send it to the webserver. The system sent SMS alerts for all the 5 level readings, and was able to override the FM frequency for levels 4 and 5.

7	Check audio transmission.	If the simulated water level reaches levels 4 and 5, the system should transmit audio alerts as well, alongside with SMS alerts. A radio within the specified range of radius should pick up the audio sent by the SDR.	✓		The radio was able to pick up audio sent by the SDR. Since the wave file being transmitted is connected to the web server, the system was able to acquire the most recent entry in the web server, and then convert it from text to speech.
Overall test result:					The overall test result showed that the medium used to relay alerts are sending SMS and audio alerts/

Table XXXVIII shows the acceptance test of the Advisory System. Acceptance test #3 is correlated to the first two acceptance tests. In correlation to acceptance test #2, data gathered in the webserver indicates what medium of alert transmission to utilize.

In step 1, the ultrasonic sensor and the Raspberry Pi must be set up. The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins. Step 1: after 1 successful testing, the module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins.

In step 2, set up the LTE HAT. Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink. Step 2: after 3 testings in which 2 trials are failed and 1 trial is successful, after installation of dependency, the LTE HAT module was able to boot up properly. It was configured correctly to auto-connect upon startup.

In step 3, set up the SDR. Connect SDR to the laptop's USB port. The laptop should recognize SDR. Step 3: after 7 testings in which 3 trials are failed and 4 trials are successful; after installation, the laptop can recognize the SDR.

In step 4, boot the Raspberry Pi. Upon boot, it should automatically have an Internet connection. Step 4: after 1 successful testing, the LTE HAT was already configured to auto-connect to the Internet.

In step 5, simulate water levels. The sensor should return an accurate measurement of the water level. Step 5: after 17 testings in which 4 trials are failed and 13 trials are successful, the sensor module was able to measure water level and return results. It was also able to measure when there is a change of level category.

In step 6, run the test program. If Raspberry Pi is connected to the Internet, it should be able to access the webserver for the subscribers' mobile numbers. The system should send SMS alerts. Step 6: after 7 testings in which 6 trials are failed and 1 trials is successful, the first 6 trials yielded error from web server connection. Once the code was modified, Raspberry Pi was able to recognize the module and was also able to interpret the detected measurement and send it to the webserver.

In step 6, run the test program. Step 6: after 15 testings in which 9 trials are failed and 6 trials are successful; multiple code modifications had to be done on getting the last entry on the webserver and then converting it from text to speech before the successful trial.

Step 7 asks for checking the audio transmission. If the simulated water level reaches levels 4 and 5, the system should transmit audio alerts as well, alongside with SMS alerts. A radio within the specified range of radius should pick up the audio sent by the SDR. The last step is successful after multiple trials as numerous prior trials showed jamming of radio reception after the process of overriding the frequency, code modification and trial and error on adjustment of parameters to decrease noise in the

transmitted audio, while increasing the volume and range. Step 7: after 8 testings in which 3 trials are failed and 5 trials are successful; after successful parameter modification, the radio was able to pick up the audible audio sent by the SDR.

The overall test result showed that the medium used to relay alerts are sending SMS and audio alerts. Successful overall testing of SMS and Audio Alerts Acceptance Test #3 meets the engineering requirement that the system must correctly interpret readings to determine which medium of broadcasting to utilize.

TABLE XXXIX
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 4

Test Writer:	Chua, Raymond Carl P.				
Test Case Name:	Audio Alerts Acceptance Test #4		**Test ID:**	Audio-AT-04	
Description:	Transmit audio alerts.		**Type:**	☐ white box ✓ black box	
Test Information					
Name of Tester:	Bayhon, Elaine Grace S.		**Date:**		
Hardware Version:			**Time:**		
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, SDR is connected to the Raspberry Pi through USB.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**Comments**
1	Set up the ultrasonic module and the Raspberry Pi. Boot Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins
2	Set up SDR.	Connect SDR to the laptop's USB port. The laptop should recognize SDR.	✓		The laptop was able to recognize the SDR.
3	Simulate water levels.	If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level
4	Check audio transmission.	A radio within the specified range of radius should pick up the audio sent by the SDR.	✓		The radio was able to pick up audio sent by the SDR. The audio indicates the latest data entry in the web server.
Overall test result:					The overall test result shows that the system can transmit to radios within a specified radius if the ultrasonic sensor detects a flood that reaches the last 2 emergency levels. The alerts transmitted are also the data acquired from the web server.

Table XXXIX shows the acceptance test details of audio transmission. Acceptance test #4 is correlated with the first three acceptance tests. In this acceptance test, the

audio alerts must show that the most recent data entry in the web server is gathered and is also converted from text to speech. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins.

For the test, the first step is setting up the ultrasonic sensor module and booting Raspberry Pi. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins. The 1st step is successful at the first trial, as it has already undergone and passed unit and integration tests for the Ultrasonic sensor module and Raspberry Pi.

The second step would be connecting the SDR to the Laptop through the USB port. The 2nd step is successful at 3rd trial, as VMware Workstation must be installed first as a virtual machine running on Linux. Once the virtual machine is set up, installation of dependencies of HackRF is required.

The third step would be simulating water levels. If the water level reaches level 4 and 5, the SDR should be able to override radio frequency channels around the radius. The last step, checking the audio transmission took 8 tests before the successful run. The overridden frequency is left jammed if the audio transmitted are too long compared to the set time in the codes. This means the transmit process is intercepted. Once the audio length is modified, this last step is successful. A radio within the specified range of radius should pick up the audio sent by the SDR.

The overall test result showed that the system can transmit to radios within a specified radius if the ultrasonic sensor detects a flood that reaches the last 2 emergency levels. Successful overall testing of Audio Alerts Acceptance Test #4 meets the engineering requirements that the system must override FM radio frequency using SDR as FM transmitter when the ultrasonic sensor detects flood water has reached the last 2 emergency levels, and that the SDR must be able to transmit and override radio frequency.

TABLE XXXX
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 5

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	API Acceptance Test #5	**Test ID:**	API-AT-05
Description:	Create SMS advisory system.	**Type:**	☐ white box ✓ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	March 7, 2020
Hardware Version:		**Time:**	
Setup:	A mobile number must be subscribed to the system already.		

Step	Action	Expected Result	Pass	Fail	Comments
1	The sensor detects flood level	The flood level details will be sent to the webserver through post request	✓		The post request sends the data to the API.
2	The details will be sent to the API	It should reflect on the home page of the web page	✓		An API route receives the data being sent from the sensor.
3	The API sends the message and the details of the subscribers to Globe Labs API	The subscriber should receive the alert message	✓		Receiving the alert message means that sending the data to the Globe Labs API
Overall test result:			✓		The overall test result should show that the Globe Labs API is used to develop the advisory system. The system will use API as an intermediary between the Web server and the users.

Table XXXX shows the acceptance test of the SMS Advisory System using API.

In step 1, the sensor detects the flood level. The flood level details will be sent to the webserver through post request. Step 1: after 5 testings in which 2 trials are failed and 3 trials are successful, the post request sent the data to the API.

In step 2, The details will be sent to the API. It should reflect on the home page of the web page. Step 2: after 6 testings in which 2 trials are failed and 4 trials are

successful; the API route receives the data being sent from the Raspberry Pi. The errors where some of the variable names from payload and the API are different are fixed after debugging.

In step 3, The API sends the message together with the subscriber's details to Globe Labs API. The subscriber should receive the alert message. Step 3: after 6 testings in which 2 trials are failed and 4 trials are successful, the alert system subscribers receiving the alert message indicates that the system met the Globe Labs API's requirements before sending the SMS.

The overall test result showed that the Globe Labs API is used to develop the advisory system. The system will use API as an intermediary between the Web server and the users. Successful overall testing of API Acceptance Test #5 meets the engineering requirements that the system must create an SMS advisory system using Globe Labs API.

TABLE XXXXI
ACCEPTANCE TEST OF ENGINEERING REQUIREMENT 6

Test Writer:	Chua, Raymond Carl P.		
Test Case Name:	SMS Alerts Acceptance Test #6	**Test ID:**	SMS-AT-06
Description:	Send SMS alerts for all the water level readings.	**Type:**	☐ white box ✓ black box
Test Information			
Name of Tester:	Bayhon, Elaine Grace S.	**Date:**	
Hardware Version:		**Time:**	
Setup:	While the ultrasonic sensor is connected to the Raspberry Pi's GPIO pins, the LTE HAT is stacked right on top of the Raspberry Pi's GPIO pins with the use of pin headers. The source is through the mini-USB port.		

Step	Action	Expected Result	Pass	Fail	Comments
1	Set up the ultrasonic module and the Raspberry Pi.	The wires should be connected to the correct Vcc, ground, and Tx, Rx GPIO pins.	✓		The module was able to function well with its current connection of VCC, Ground, Tx, Rx, and assigned GPIO pins
2	Set up LTE HAT.	Stack the LTE HAT right on top of Raspberry Pi's GPIO pins. Connect to the mini-USB port for the source. The blue LED indicator should blink.	✓		The LTE HAT module was able to boot up properly and was able to transmitted and received internet connection constantly
3	Boot Raspberry Pi.	Upon boot, the Raspberry Pi should automatically have an Internet connection.	✓		Raspberry Pi has able to boot properly and was able to connect to the internet automatically
4	Run test program.	The sensor should return an accurate measurement of the water level.	✓		The sensor module was able to measure the accurate water level and able to return the result instantly.
5	Simulate water levels.	If Raspberry Pi is connected to the web server, it should be able to send alerts to the subscribers for all the 5 levels.	✓		The sensor module was able to measure water level and return results. It was also able to measure when there is a gradual change in water level
Overall test result:			✓		It showed that Raspberry Pi receives an internet connection through LTE HAT and was able to send the data acquired by the ultrasonic sensor to the webserver.

Table XXXVIII shows the acceptance test details of the SMS transmission. The ultrasonic sensor is wired to Raspberry Pi's Vcc, ground, and GPIO pins. Right on top of the GPIO pins, the LTE HAT is stacked with the use of pin headers. For the test, step 1 is setting the ultrasonic sensor module and Raspberry Pi up. The wires should be correctly connected to a Vcc source, ground, and Rx Tx GPIO pins.

Step 1 is setting up the ultrasonic sensor module and booting Raspberry Pi. After 1 successful testing, the wires are correctly connected to a Vcc source, ground, and Rx Tx GPIO pins.

Step 2 is setting up the LTE HAT. LTE HAT is stacked on the Raspberry Pi's GPIO pins and connected to the mini-USB port for the source. The blue LED indicator should blink. Step 2: after 3 testings in which 2 trials are failed and 1 is successful; the LTE HAT module was able to boot up properly. It was configured correctly to auto-connect upon startup.

Step 3 is booting the Raspberry Pi. Upon boot, it should automatically be connected to the Internet. Step 3: after 2 testings in which 1 trial is failed and 1 is successful; Raspberry Pi was able to boot properly and was able to connect to the internet automatically.

Step 5 is simulating water levels. On this step, the Raspberry Pi already has access to the Internet and is, therefore, able to establish a connection to the webserver. Step 5: after 17 testings in which 5 trials are failed due to web server access, 12 trials are successful; Raspberry Pi was able to recognize the module and was also able to interpret the detected measurement and send it to the webserver.

The overall test showed that Raspberry Pi receives an internet connection through LTE HAT and was able to send the data acquired by the ultrasonic sensor to the webserver. Successful overall testing of SMS Alerts Acceptance Test #4 meets the engineering requirements that the system must push SMS alerts to the advisory subscribers for every water level reading.

Prototype Functionality Testing

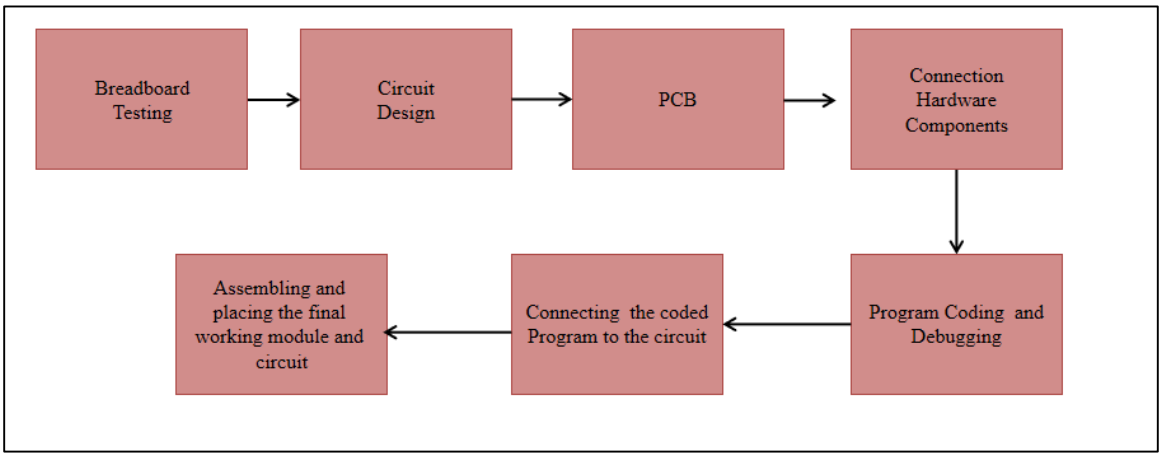


Figure 52. System Processes

Figure 52 shows the processes throughout the research. Started with breadboard testing and circuit design, etching into PCB, integration with other hardware components, program coding, system hardware, and software integration, and final prototype assembling.

TABLE XXXXII
MEASUREMENT COMPARISON BETWEEN EXPERIMENTAL AND THEORETICAL DATA VALUES

Sensor Detection (Experimental)	Actual Measurement (Theoretical)	Difference
8.71	8.5	0.21
14.37	14.6	0.23
16.16	16.4	0.32
17.28	17.6	0.32
23.47	23.8	0.33
25.22	25.8	0.58
26.6	27.5	0.9

Table XXXXII shows the data comparison between experimental and theoretical values gathered from a controlled flood simulation. The experimental values are the data acquired by the ultrasonic sensor while the theoretical values are measured using a measuring tape. Each experimental value entry is the average value of 10 ultrasonic sensor readings. The average percentage error of the flood level measuring system is 1.78092%.

$$Average\ Percentage\ Error = \frac{|Avg.Experimental\ measurement - Avg.Theoretical\ measurement|}{Avg.Theoretical\ measurement} \times 100$$

$$Average\ Percentage\ Error = \frac{|Avg.Experimental\ measurement - Avg.Theoretical\ measurement|}{Avg.Theoretical\ measurement}$$

$$Average\ Percentage\ Error = \frac{|(8.71 + 14.37 + 16.16 + 17.28 + 23.47 + 25.22 + 26.6) - (8.5 + 14.6 + 16.4 + 17.6 + 23.8 + 25.8 + 27.5)|}{(8.71 + 14.37 + 16.16 + 17.28 + 23.47 + 25.22 + 26.6)} \times 100$$

$$Average\ Percentage\ Error = \frac{|(131.81) - (134.2)|}{(131.81)} \times 100$$

$$Average\ Percentage\ Error = 1.78092\%$$

Chapter V.

SUMMARY OF FINDINGS, CONCLUSIONS AND RECOMMENDATIONS

A. Summary of Findings

This study summarizes the following findings:

- The acceptance test of utilizing ultrasonic sensors that will accurately measure flood levels shows that the sensor module was able to yield accurate measurement results. It was programmed to calculate the distance between floodwater and the sensor and its difference when subtracted to the distance between sensor and ground.
- The acceptance test of database development using MySQL database shows that MySQL database is used to store subscribers' mobile numbers and flood level data measurements. These acquired data are also reflected on the web page.
- The acceptance test of creating an SMS advisory system shows that Globe Labs API is used to develop the advisory system. The system used API as an intermediary between the Web server and the users.
- The acceptance test of the developed system can determine when to send SMS alerts and audio alerts show that the medium used to relay alerts are both SMS advisory and FM Radio. During the simulation of floodwater, for all levels 1-5, the system was able to send alerts through SMS. For emergency levels 4 & 5, the system also sent alerts through FM radio overriding.

- The acceptance test of sending SMS alerts for all the water level readings shows that Raspberry Pi receives an internet connection through LTE HAT and was able to send the data acquired by the ultrasonic sensor to the webserver.
- The acceptance test of transmitting audio alerts shows that the system can transmit to FM radios within a specified radius if the ultrasonic sensor detects flood that reaches the last 2 emergency levels.

B. Conclusions

This study concludes the following:

- Since ultrasonic sensors acquire data by measuring the difference in time between pulse transmission and echo detection, ultrasonic sensors are susceptible to many external variables as to when acquiring data. Depending on the application, these external variables can be handled as well.
- Since the ultrasonic sensor uses pulses of ultrasonic sound, mounting the sensor on top of a pipe improves the function of the sensor since the sound pulse is much more concentrated compared to when the echo pulses are in the open and are more susceptible to loss. With a pipe installed, the area of measurement is less exposed to turbulence.
- Since the pipe installed on the prototype is elevated 7 inches from the ground, water level measurements less than 7 inches are inaccurate. Pulse upon reaching the open area, some of the signals are reflected away, resulting in less echo detection; inaccurate results.

- Ultrasonic sensor distance measurement can also be affected by temperature. The speed of sound varies with temperature. Variation in temperature during testing results in variation and fluctuation in the results.
- SDR's transmission reach depends on the signal strength of the receiver. The clarity and strength of transmission vary from every FM radio station frequencies.
- To achieve a flood monitoring system that accurately detects floodwater in a certain flood-prone area, the system should utilize ultrasonic sensors. By calculating the distance between floodwater and the sensor, the sensor module can yield accurate measurement results.
- To override FM radio frequency, using SDR as an FM transmitter will implement radio hardware implementations into the software. Osmocom sink GNU radio blocks convert radio hardware into software API. Inputting audio samples in wave file source into the Wide Band FM transmitter produces Frequency Modulated output. The SDR can transmit and override radio frequency.
- To use shortcode in implementing SMS advisory, the database for subscribers' mobile numbers and flood level data measurements are to be stored in MySQL database. To implement the SMS advisory system, Globe Labs API was used.

C. Recommendations

This study recommends the following:

- To improve accuracy when measuring low levels of floodwater, the pipe installed should be leveled as close as possible to the ground. A good anti-clogging prototype design is advisable.

- To improve the reliability of the flood monitoring system, the prototype should be situated in a signal well-received area of a flood-prone area. Internet connectivity plays a huge role in sending data between the modules and the server, hence, the location of the prototype should be considered.
- A good solution for speed sound variation due to temperature is installing the ultrasonic sensor in an insulated housing. Using commercial-grade sensors that provide weather resistance is an option too.
- To lessen the delay in communication between the module and the server, a better Internet connectivity setup should be considered aside from the LTE module. i.e, long-range Wi-Fi connectivity.
- To improve the power lifespan of the prototype, consider including a power backup source—a solar system consisting of a solar panel and a reliable battery source. The solar panel should keep the sensor beacon running on sunny days.
- To cover more ground on flood monitoring, ideally, the transmitter should be situated in a center point of a community. Expansion of the system is possible by building a network of sensor beacons; install more sensor beacons situated in different streets.
- To improve the system of FM frequency overriding and to even lessen the total time in between alert transmissions, instead of consecutive overriding of radio frequencies, consider adding the feature of simultaneous FM frequency overriding for future studies.

References

[1] W. Indrasari, B. H. Iswanto, and M. Andayani, "Early Warning System of Flood Disaster Based on Ultrasonic Sensors and Wireless Technology", April 2018, [Online]. Available: https://www.researchgate.net/publication/324668205_Early_Warning_System_of_Flood_Disaster_Based_on_Ultrasonic_Sensors_and_Wireless_Technology [Accessed October 21, 2019].

[2] J G Natividad and J M Mendez, "Flood Monitoring and Early Warning System Using Ultrasonic Sensor", IOP Conf. Series: Materials Science and Engineering, 2017. doi: 10.1088/1757-899X/325/1/012020.

[3] Mohamed Khalaf, "Flood Detection using Sensor Network and Notification via SMS and Public Network", Student Conference On Research And Development, February 2011, [Online]. Available: https://www.researchgate.net/publication/263088726_Flood_Detection_using_Sensor_Network_and_Notification_via_SMS_and_Public_Network [Accessed October 20, 2019].

[4] Clickatel. “What is a short code” [Online]. Available: https://www.clickatell.com/products/sms/short-codes/ [Accessed October 21, 2019]

[5] Tatango. “SMS Short Codes What Every Business Needs To Know” [Online]. Available: https://www.tatango.com/blog/sms-short-codes-what-every-business-needs-to-know/ [Accessed October 21, 2019]

[6] CCN Philippines. "Countries Most Afflicted by Disasters". 2015, [Online].

Available: http://nine.cnnphilippines.com/news/2015/11/25/philippines-fourth-most-disaster-prone-country.html [Accessed October 21, 2019]

[7] Asian Disaster Reduction Center. "Information on Disaster Risk Reduction of the Member Countries" [Online]. Available: https://www.adrc.asia/nationinformation.php?NationCode=608&Lang=en&NationNum=14 [Accessed October 21, 2019]

[8] Inquirer. "Special Report on Storm 'Ondoy': Marikina remembers 'end of the world'" [Online]. Available: https://newsinfo.inquirer.net/818907/special-report-on-storm-ondoy-marikina-remembers-end-of-the-world [Accessed October 21, 2019]

[9] BusinessMirror Editorial. "Metro Manila floods are getting worse" [Online]. Available: https://businessmirror.com.ph/2019/08/06/metro-manila-floods-are-getting-worse/ [Accessed October 21, 2019]

[10] GMA News. "Heavy downpour causes flood, zero visibility in QC". 2020, [Online] Available: https://www.gmanetwork.com/news/news/metro/750273/heavy-downpour-causes-flood-zero-visibility-in-ncr/story/ [Accessed September 20, 2020]

[11] GMA News. "10-year-old dies after falling into Santa Rosa river amid flash flood". 2020, [Online] Available: https://www.gmanetwork.com/news/news/regions/752271/10-year-old-dies-after-falling-into-santa-rosa-river-amid-flash-flood/story/

[12] K. Rosen. "How to Pack an Emergency Kit for Any Disaster. [Online]. https://www.nytimes.com/2017/07/03/smarter-living/packing-emergency-kit-disaster.html

[13] “Flood Forecasting Systems” [Online]. https://www.ctc-n.org/technologies/flood-forecasting-systems#:~:text=Flood%20warnings%20are%20a%20highly,defence%20construction%20would%20be%20prohibitive. [Accessed September 20, 2020]

[14] D. Bonderud. “Does Weather Affect Internet Speed?” [Online]. https://www.bandwidthplace.com/does-weather-affect-internet-speed-article/ [Accessed October 20, 2019].

[15] L. Frenzel Jr. (2018). “Radio/Wireless” [Online]. https://www.sciencedirect.com/topics/engineering/antenna-length [Accessed October 20, 2019].

[16] Database Management. [Online]. Available: http://www.umsl.edu/~joshik/msis480/chapt06.htm [Accessed October 21, 2019]

[17] What is SQL and MySQL, (2020). [Online]. Available: https://en.wikipedia.org/wiki/MySQL [Accessed September 2, 2020]

[18] Definition of Flood Prone Area [Online]. Available: https://www.lawinsider.com/dictionary/flood-prone-area [Accessed October 21, 2019]

[19] Globe Labs API [Online]. Available: http://www.globelabs.com.ph/#!/developer/api/sms

[Accessed October 21, 2019]

[20] GNU Radio Guided Tutorial Introduction, (2020). [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction [Accessed September 20, 2020]

[21] Github GNU Radio [Online]. Available: https://github.com/gnuradio/gnuradio [Accessed July 24, 2020].

[22] “Raspberry Pi 3G/4G & LTE Base Hat” (2019). [Online]. https://sixfab.com/product/raspberry-pi-base-hat-3g-4g-lte-minipcie-cards/

[23] “What is Raspberry Pi” [Online]. https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Accessed October 20, 2019]

[24] “Raspberry Pi OS” [Online]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/ [Accessed September 20, 2020]

[25] SDR-Radio.com [Online]. Availabe: https://www.sdr-radio.com/ [Accessed September 20, 2020]

[26] “Difference Between SMS and MMS” [Online]. Available: https://twigby.zendesk.com/hc/en-us/articles/115010624828-What-s-the-difference-between-SMS-and-MMS- [Accessed October 20, 2019]

[27] “What is a Messaging Short Code?” [Online]. Available: https://support.twilio.com/hc/en-us/articles/223182068-What-is-a-Messaging-Short-Code- [Accessed October 20, 2019]

[28] Strickland. “What’s Ubuntu, and How Is It Different from Linux?”, (2020). [Online]. Available: https://computer.howstuffworks.com/ubuntu.htm [Accessed September 20, 2020]

[29] Burnett. “Understanding How Ultrasonic Sensors Work”, (2020). [Online]. Available: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm

[30] VMWare Workstation, (2020). [Online]. Available: https://en.wikipedia.org/wiki/VMware_Workstation [Accessed September 20, 2020]

[31] Web Server [Online]. Available: https://en.wikipedia.org/wiki/Web_server [Accessed October 20, 2019

[32] R. Ado, J. Bautista, K. Llleno, K. Malagday, M. Muya (2014). “Road Flood Sensor With Web and Mobile Application Support”. [Online]. https://www.slideshare.net/markanthonymuya/road-flood-sensor-with-web-and-mobile-application-support [Accessed October 21, 2019].

[33] M. Suleiman, G.I. Saidu, M. I. Ilyasu, O. Adeboye and M. Hamza , "Ultrasonic Fluid Level Measuring Device", International Journal of Recent Development in Engineering and Technology, Vol. 4, Special Issue 1, May 2015, [Online]. Available: http://www.ijrdet.com/files/ICMTSET2015/IJRDET_ICMTSET_01.pdf [Accessed October 21, 2019].

[34] Anthony Joseph Boscacci, "Emergency Signal Intercepting Unit", February 6, 2015, [Online]. Available: http://www.freepatentsonline.com/8928492.html [Accessed October 20, 2019].

[35] Technical Standards Committee of the Kapisanan Ng Mga Brodkaster sa Pilipinas, Technical Standards and Operating Requirements for FM Broacast Stations in the Philippines (1991, Revised)

[36] C. Zequiang, C. Nengcheng, D. Wenying, G. Jianya “An Active Monitoring Method For Flood Events” [Online]. https://ui.adsabs.harvard.edu/abs/2018CG....116...42C/abstract [Accessed January 20, 2019].

[37] “National Disaster Risk Reduction and Management Plan” [Online]. http://www.ndrrmc.gov.ph/attachments/article/41/NDRRM_Plan_2011-2028.pdf [Accessed January 20, 2019].

[38] F. C. C. Garcia, A. E. Retamar and J. C. Javier, "A real time urban flood monitoring system for metro Manila," (2015). [Online]. https://ieeexplore.ieee.org/document/7372990 [Accessed October 20, 2019].

[39] “How Raspberry Pi Is Different From A Desktop Computer” [Online]. https://raspberryinsider.com/how-raspberry-pi-is-different-from-a-desktop-computer/ [Accessed September 20, 2020]

[40] Hattersley. “Raspberry Pi 4 vs Raspberry Pi 3B+” (2020). [Online]. https://magpi.raspberrypi.org/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus [Accessed September 20, 2020]

[41] Hanan, A. Gunawan, M. Sumadiyasa “Water Level Detection System Based on Ultrasonic Sensors HC-SR04 and ESP8266-12 Modules with Telegram and Buzzer

Communication Media" March (2019). [Online]. Available: https://www.researchgate.net/publication/336440730_Water_Level_Detection_System_Based_on_Ultrasonic_Sensors_HC-SR04_and_ESP8266-12_Modules_with_Telegram_and_Buzzer_Communication_Media

[42] What is an Ultrasonic Sensor? (2020), [Online]. Available: https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/#:~:text=As%20the%20name%20indicates%2C%20ultrasonic,between%20the%20emission%20and%20reception. [Accessed September 20, 2020]

[43] "Ultrasonic Sensors: Advantages and Limitations" (2019), [Online]. Available: https://www.maxbotix.com/articles/advantages-limitations-ultrasonic-sensors.htm/ [Accessed September 20, 2020]

[44] "Float Switch and Level Sensing Solutions For Agitated Liquid and Turbulence" (2015), [Online]. Available: https://www.fluidswitch.com/2015/06/16/float-switch-and-level-sensing-solutions-for-agitated-liquids-and-turbulence/ [Accessed September 20, 2020]

[45] J. Mitola. "The software-defined radio architecture" IEEE Communication Magazine, 33 (5) (May 1995), pp. 26-37. [Online]. Available: https://www.cs.odu.edu/~cs752/papers/sdr-003.pdf

M.Heddebaut, JP. Ghys, M. Sanzc, and F. Elbahharab, "Road Traffic Information Using a Dedicated Radio Beacon", Transportation Research Part C: Emerging Technologies, Vol. 35, Pages 20-33, October 2013, [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0968090X13001253 [Accessed October 20, 2019].

[46] Salleh et al. (2013), "Design of Low Power Wideband Low Noise Amplifier for Software Defined Radio at 100 MHz to 1 GHz", Procedia Engineering, Vol. 53, Pages 368-375, 2013, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705813001677 [Accessed October 20, 2019].

FM Multiplex Broadcasting and Beacon [Online]. Available: https://www.vics.or.jp/en/know/structure/beacon.html [Accessed January 20, 2020]

[47] Walter H.W. Tuttlebee, "Software Defined Radio: Enabling Technologies", 2002, [Online]. Available: https://books.google.com.ph/books?hl=en&lr=&id=CyNCLZJCIs8C&oi=fnd&pg=PR5&ots=rEQ_ARObz9&sig=SG9cWM-fNue3QfiZpOXCutmPxT0&redir_esc=y#v=onepage&q&f=false [Accessed October 20, 2019].

[48] D. Curtis et al. (2019), "Software Defined Automotive Radar", Uhnder Inc, August 5, 2019, [Online]. Available: https://worldwide.espacenet.com/publicationDetails/biblio?II=1&ND=3&adjacent=true&locale=en_EP&FT=D&date=20190905&CC=US&NR=2019271776A1&KC=A1 [Accessed October 20, 2019].

[49] GNU Radio Hardware (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/Hardware [Accessed September 20, 2020]

[50] Nicholas Goldstein, "Override Transmitter-I.F.", October 8, 2016, [Online]. Available: https://patents.google.com/patent/US20060176172A1/en?q=car&q=radio+frequency&q=overriding&oq=car+radio+frequency+overriding [Accessed October 20, 2019].

[51] HackRF One Transmit Multiple Channel (2020), [Online]. Available: https://github.com/mossmann/hackrf/issues/486 [Accessed October 8, 2020]

[52] C. Smith. “SDR-Examples” (2020), [Online]. Available: https://github.com/argilo/sdr-examples [Accessed October 8, 2020]

[53] E. Argo, R. Baglino. “Emergency Vehicle Radio Transmission System”, [Online]. Available: https://patents.google.com/patent/US4764978A/en [Accessed October 8, 2020]

[54] P. Wait “Military Seeks Radio Override”, [Online]. Available: https://www.informationweek.com/software/information-management/military-seeks-radio-override/d/d-id/1111259 [Accessed October 8, 2020]

[55] “Globe Labs API” [Online]. Available: http://www.globelabs.com.ph/#!/developer/api/sms [Accessed October 20, 2019].

[56] “What is GNU Radio?” (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F [Accessed September 20, 2020]

[57] GNU Radio FAQ (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/FAQ [Accessed September 20, 2020]

[58] “Flood-Prone Areas in Metro Manila”, June 10, 2018, [Online]. Available: https://www.hoppler.com.ph/magazine/featured-articles/flood-prone-areas-in-metro-manila [Accessed October 20, 2019].

[59] “What is Waterfall Model?” [Online]. Available: https://www.toolsqa.com/software-testing/waterfall-model/ [Accessed October 20, 2019].

[60] “The Engineering Design Process” Available: https://www.slideshare.net/Fordlovers/the-engineering-design-process [Accessed October 20, 2019].

[61] Ultrasonic Distance Sensor [Online]. Available: https://www.sparkfun.com/products/15569 [Accessed July 24, 2020].

[62] Raspberry Pi 3 Model B+ [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/#:~:text=The%20Raspberry%20Pi%203%20Model,2.0%20(maximum%20throughput%20300%20Mbps) [Accessed July 24, 2020].

[63] Raspberry Pi 3G/4G & LTE Base HAT [Online]. Available: https://sixfab.com/product/raspberry-pi-base-hat-3g-4g-lte-minipcie-cards/ [Accessed July 24, 2020].

[64] HackRF One [Online]. Available: https://github.com/mossmann/hackrf/wiki/HackRF-One [Accessed July 24, 2020].

[65] TechTerms Python [Online]. Available: https://techterms.com/definition/python#:~:text=Python%20is%20a%20high%2Dlevel,use%2C%20even%20for%20commercial%20applications.&text=Python%20is%20considered%20a%20scripting,applications%20and%20dynamic%20Web%20content. [Accessed July 24, 2020].

[66] Ankur Tiwari, Should you host Python websites on PythonAnywhere?, January 21, 2020 [Online]. Available: https://medium.com/thoughtlytics/in-depth-review-should-you-host-python-website-on-pythonanywhere-594c31428336 [Accessed July 24, 2020].

[67] Globe Labs API [Online]. Available: https://www.programmableweb.com/api/globe-labs [Accessed July 24, 2020].

[68] PyCharm, June 21, 2020 [Online]. Available: https://en.wikipedia.org/wiki/PyCharm [Accessed July 24, 2020].

[69] Welcome to Raspbian [Online]. Available: https://www.raspbian.org/FrontPage [Accessed July 24, 2020].

[70] jQuery, 2020, [Online]. Available: https://jquery.com/ [Accessed July 24, 2020]

[71] M. Bostock. “d3.js”, Data-Driven Documents, 2019, [Online]. Available https://d3js.org [Accessed July 24, 2020]

Appendix A.

Executive Summary

(IEEE Format)

Frequency Overriding System Through Signal Amplification For Flood Alert with SMS Notification

Rochelle G. Bastian[1], Elaine Grace S. Bayhon[2], Raymond Carl P. Chua[3]
Maria Concepcion A. Mirabueno[4], Emmanuel E. Jamoralin[5]
Computer Engineering Department
Adamson University, Philippines
[1]budsbastian@gmail.com
[2]elainebayhon@gmail.com
[3]raymondcarlchua@gmail.com
[4]mariaconcepcion.amboy@adamson.edu.ph
[5]jamoralin.emmanuel@gmail.com

***Abstract*—** Philippines' geographical location and physical environment—located near the equator where the ocean is warm, one of the major factors for typhoon formation—contributes to the country's high-susceptibility to tropical cyclones. And with global warming, warmer waters will only continue to intensify the storms to ever hit the country. As the storm intensifies, the flood occurrence is becoming more often than before. Flood monitoring systems' main role is to inform and also to possibly save people's lives.

Currently, there are various researches about the importance and implementation of flood monitoring. Most of the flood monitoring system being deployed nowadays use social media as its platform, as it is one of the current popular medium of communication. Sending an alert through social media does reach a larger audience for just a few clicks away. However, in this study, the researchers considered the downside of relying primarily on the convenience of the Internet and social media.

What the researchers have designed is a system that informs the people about the flood level in the area, with no internet connection needed. It is also designed to reach a specific audience by creating an SMS advisory system, which the subscribers can receive, and utilizing radio signals within a certain radius. In this study, the researchers will utilize FM radio frequency overriding and SMS advisory as the medium for flood level alerts to the people nearby and those who are subscribed.

The developed system can determine when to send SMS alerts and audio alerts show that the medium used to relay alerts are both SMS advisory and FM Radio. During the simulation of floodwater, for all levels 1-5, the system was able to send alerts through SMS. For emergency levels 4 & 5, the system also sent alerts through FM radio overriding.

Keywords*—*Flood monitoring, FM radio overriding,

Introduction

Social media has brought huge changes in the way people acquire news. Back then, people's primary source of information/news was not through the Internet--social media like Twitter, Facebook, YouTube, etc., but instead, people relied on TV, radio, or newspaper. As time goes by, a lot may have changed on the way people primarily obtain the news, but radio has proved its reliability--importantly, during times of emergencies or calamities e.g. typhoons.

I. Objectives

- To develop a flood monitoring system that accurately detects the floodwater level in a certain flood-prone area.
- To override FM radio frequency when flood reaches the last two emergency levels
- To use shortcode in implementing SMS advisory for all 5 flood water level detections.

II. Methodology

The requirement specification mentioned in this study provides a functionality outline of the system to be developed. It provides an overview of how the system is supposed to work, and its corresponding operational conditions to its target users. The requirement specification mentioned in this study allows people nearby the beacon device to receive flood alerts through SMS short messages and/or radio.

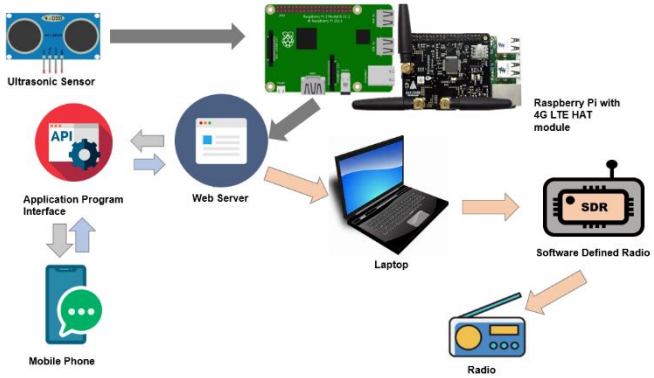


Figure 1. General System Architecture

In the prototype, attached to the Raspberry Pi are the ultrasonic sensor and LTE module. The LTE module is a communication module that supports LTE mobile Internet connectivity for the Raspberry Pi. The data gathered by the

Ultrasonic Sensor are then sent to the server through the Raspberry Pi. One broadcasting medium of this study is through the SMS advisory service. For all the 5 levels, SMS advisory is going to be triggered. If the ultrasonic sensor detects the flood water reaching the last 4th and 5th levels, FM radio frequency overriding will also be implemented. For the process of FM radio frequency overriding, an x64-based laptop running on Ubuntu Linux will be used to run GNU Radio to configure the SDR (Software Defined Radio). The SDR will transmit the audio alerts to the FM radio units around the specified radius.

III. Summary of Findings

- The acceptance test of utilizing ultrasonic sensors that will accurately measure flood levels shows that the sensor module was able to yield accurate measurement results. It was programmed to calculate the distance between floodwater and the sensor and its difference when subtracted to the distance between sensor and ground.
- The acceptance test of database development using MySQL database shows that MySQL database is used to store subscribers' mobile numbers and flood level data measurements. These acquired data are also reflected on the web page.
- The acceptance test of creating an SMS advisory system shows that Globe Labs API is used to develop the advisory system. The system used API as an intermediary between the Web server and the users.
- The acceptance test of the developed system can determine when to send SMS alerts and audio alerts show that the medium used to relay alerts are both SMS advisory and FM Radio. During the simulation of floodwater, for all levels 1-5, the system was able to send alerts through SMS. For emergency levels 4 & 5, the system also sent alerts through FM radio overriding.
- The acceptance test of sending SMS alerts for all the water level readings shows that Raspberry Pi receives an internet connection through LTE HAT and was able to send the data acquired by the ultrasonic sensor to the webserver.
- The acceptance test of transmitting audio alerts shows that the system can transmit to FM radios within a specified radius if the ultrasonic sensor detects flood that reaches the last 2 emergency levels.

IV. Conclusions

- Since ultrasonic sensors acquire data by measuring the difference in time between pulse transmission and echo detection, ultrasonic sensors are susceptible to many external variables as to when acquiring data. Depending on the application, these external variables can be handled as well.
- Since the ultrasonic sensor uses pulses of ultrasonic sound, mounting the sensor on top of a pipe improves the function of the sensor since the sound pulse is much more concentrated compared to when the echo pulses are in the open and are more susceptible to loss. With a pipe installed, the area of measurement is less exposed to turbulence.
- Since the pipe installed on the prototype is elevated 7 inches from the ground, water level measurements less than 7 inches are inaccurate. Pulse upon reaching the open area, some of the signals are reflected away, resulting in less echo detection; inaccurate results.
- Ultrasonic sensor distance measurement can also be affected by temperature. The speed of sound varies with temperature. Variation in temperature during testing results in variation and fluctuation in the results.
- SDR's transmission reach depends on the signal strength of the receiver. The clarity and strength of transmission vary from every FM radio station frequencies.
- To achieve a flood monitoring system that accurately detects floodwater in a certain flood-prone area, the system should utilize ultrasonic sensors. By calculating the distance between floodwater and the sensor, the sensor module can yield accurate measurement results.
- To override FM radio frequency, using SDR as an FM transmitter will implement radio hardware implementations into the software. Osmocom sink GNU radio blocks convert radio hardware into software API. Inputting audio samples in wave file source into the Wide Band FM transmitter produces Frequency Modulated output. The SDR can transmit and override radio frequency.
- To use shortcode in implementing SMS advisory, the database for subscribers' mobile numbers and flood level data measurements are to be stored in MySQL database. To implement the SMS advisory system, Globe Labs API was used.

V. Recommendations

- To improve accuracy when measuring low levels of floodwater, the pipe installed should be leveled as close as possible to the ground. A good anti-clogging prototype design is advisable.
- To improve the reliability of the flood monitoring system, the prototype should be situated in a signal well-received area of a flood-prone area. Internet connectivity plays a huge role in sending data between the modules and the server, hence, the location of the prototype should be considered.

- A good solution for speed sound variation due to temperature is installing the ultrasonic sensor in an insulated housing. Using commercial-grade sensors that provide weather resistance is an option too.
- To lessen the delay in communication between the module and the server, a better Internet connectivity setup should be considered aside from the LTE module. i.e, long-range Wi-Fi connectivity.
- To improve the power lifespan of the prototype, consider including a power backup source—a solar system consisting of a solar panel and a reliable battery source. The solar panel should keep the sensor beacon running on sunny days.
- To cover more ground on flood monitoring, ideally, the transmitter should be situated in a center point of a community. Expansion of the system is possible by building a network of sensor beacons; install more sensor beacons situated in different streets.

ACKNOWLEDGMENT

To all the people who took part in our journey as we accomplish our design project *Frequency Overriding System Through Signal Amplification for Flood Alert with SMS Notification*, thank you.

We would like to thank God Almighty for giving us the strength to persevere to finish our design project, and keeping us safe amidst these trying times. Without His guidance, this achievement would not have been possible.

To our thesis advisor, Engr. Maria Concepcion A. Mirabueno, we are grateful for all the knowledge you have shared with us. Thank you for always taking time to provide us guidance. Thank you for your support and trust.

To our panelists, Engr. Hubert Q. Temprosa, Engr. Yolanda D. Austria, and Engr. Jonel R. Macalisang, thank you for giving us the trust that we can accomplish the research.

To Engr. James Santiago, a consultant at the Department of Information and Technology, for taking the time off your busy schedule to meet us for consultation on such short notice.

To of KBP – Kapisanan ng mga Brodkaster ng Pilipinas, for accommodating us for consultation, and for the furth er assistance by referring us to Engr. Santiago, thank you.

REFERENCES

[1] W. Indrasari, B. H. Iswanto, and M. Andayani, "Early Warning System of Flood Disaster Based on Ultrasonic Sensors and Wireless Technology", April 2018, [Online]. Available: https://www.researchgate.net/publication/324668205_Early_Warning_System_of_Flood_Disaster_Based_on_Ultrasonic_Sensors_and_Wireless_Technology [Accessed October 21, 2019].

[2] J G Natividad and J M Mendez, "Flood Monitoring and Early Warning System Using Ultrasonic Sensor", IOP Conf. Series: Materials Science and Engineering, 2017. doi: 10.1088/1757-899X/325/1/012020.

[3] Mohamed Khalaf, "Flood Detection using Sensor Network and Notification via SMS and Public Network", Student Conference On Research And Development, February 2011, [Online]. Available: https://www.researchgate.net/publication/263088726_Flood_Detection_using_Sensor_Network_and_Notification_via_SMS_and_Public_Network [Accessed October 20, 2019].

[4] Clickatel. “What is a short code” [Online]. Available: https://www.clickatell.com/products/sms/short-codes/ [Accessed October 21, 2019]

[5] Tatango. “SMS Short Codes What Every Business Needs To Know” [Online]. Available: https://www.tatango.com/blog/sms-short-codes-what-every-business-needs-to-know/ [Accessed October 21, 2019]

[6] CCN Philippines. "Countries Most Afflicted by Disasters". 2015, [Online]. Available: http://nine.cnnphilippines.com/news/2015/11/25/philippines-fourth-most-disaster-prone-country.html [Accessed October 21, 2019]

[7] Asian Disaster Reduction Center. "Information on Disaster Risk Reduction of the Member Countries" [Online]. Available: https://www.adrc.asia/nationinformation.php?NationCode=608&Lang=en&NationNum=14 [Accessed October 21, 2019]

[8] Inquirer. "Special Report on Storm ‘Ondoy’: Marikina remembers ‘end of the world’" [Online]. Available: https://newsinfo.inquirer.net/818907/special-report-on-storm-ondoy-marikina-remembers-end-of-the-world [Accessed October 21, 2019]

[9] BusinessMirror Editorial. "Metro Manila floods are getting worse" [Online]. Available: https://businessmirror.com.ph/2019/08/06/metro-manila-floods-are-getting-worse/ [Accessed October 21, 2019]

[10] GMA News. “Heavy downpour causes flood, zero visibility in QC”. 2020, [Online] Available: https://www.gmanetwork.com/news/news/metro/750273/heavy-downpour-causes-flood-zero-visibility-in-ncr/story/ [Accessed September 20, 2020]

[11] GMA News. “10-year-old dies after falling into Santa Rosa river amid flash flood”. 2020, [Online] Available: https://www.gmanetwork.com/news/news/regions/75227

1/10-year-old-dies-after-falling-into-santa-rosa-river-amid-flash-flood/story/
[12] K. Rosen. “How to Pack an Emergency Kit for Any Disaster. [Online]. https://www.nytimes.com/2017/07/03/smarter-living/packing-emergency-kit-disaster.html
[13] “Flood Forecasting Systems” [Online]. https://www.ctc-n.org/technologies/flood-forecasting-systems#:~:text=Flood%20warnings%20are%20a%20highly,defence%20construction%20would%20be%20prohibitive. [Accessed September 20, 2020]
[14] D. Bonderud. “Does Weather Affect Internet Speed?” [Online]. https://www.bandwidthplace.com/does-weather-affect-internet-speed-article/ [Accessed October 20, 2019].
[15] L. Frenzel Jr. (2018). “Radio/Wireless” [Online]. https://www.sciencedirect.com/topics/engineering/antenna-length [Accessed October 20, 2019].
[16] Database Management. [Online]. Available: http://www.umsl.edu/~joshik/msis480/chapt06.htm [Accessed October 21, 2019]
[17] What is SQL and MySQL, (2020). [Online]. Available: https://en.wikipedia.org/wiki/MySQL [Accessed September 2, 2020]
[18] Definition of Flood Prone Area [Online]. Available: https://www.lawinsider.com/dictionary/flood-prone-area [Accessed October 21, 2019]
[19] Globe Labs API [Online]. Available: http://www.globelabs.com.ph/#!/developer/api/sms [Accessed October 21, 2019]
[20] GNU Radio Guided Tutorial Introduction, (2020). [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction [Accessed September 20, 2020]
[21] Github GNU Radio [Online]. Available: https://github.com/gnuradio/gnuradio [Accessed July 24, 2020].
[22] “Raspberry Pi 3G/4G & LTE Base Hat” (2019). [Online]. https://sixfab.com/product/raspberry-pi-base-hat-3g-4g-lte-minipcie-cards/
[23] “What is Raspberry Pi” [Online]. https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Accessed October 20, 2019]
[24] “Raspberry Pi OS” [Online]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/ [Accessed September 20, 2020]
[25] SDR-Radio.com [Online]. Availabe: https://www.sdr-radio.com/ [Accessed September 20, 2020]
[26] “Difference Between SMS and MMS” [Online]. Available: https://twigby.zendesk.com/hc/en-us/articles/115010624828-What-s-the-difference-between-SMS-and-MMS- [Accessed October 20, 2019]
[27] “What is a Messaging Short Code?” [Online]. Available: https://support.twilio.com/hc/en-us/articles/223182068-What-is-a-Messaging-Short-Code- [Accessed October 20, 2019]
[28] Strickland. “What’s Ubuntu, and How Is It Different from Linux?”, (2020). [Online]. Available: https://computer.howstuffworks.com/ubuntu.htm [Accessed September 20, 2020]
[29] Burnett. “Understanding How Ultrasonic Sensors Work”, (2020). [Online]. Available: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm
[30] VMWare Workstation, (2020). [Online]. Available: https://en.wikipedia.org/wiki/VMware_Workstation [Accessed September 20, 2020]
[31] Web Server [Online]. Available: https://en.wikipedia.org/wiki/Web_server [Accessed October 20, 2019
[32] R. Ado, J. Bautista, K. Llleno, K. Malagday, M. Muya (2014). “Road Flood Sensor With Web and Mobile Application Support”. [Online]. https://www.slideshare.net/markanthonymuya/road-flood-sensor-with-web-and-mobile-application-support [Accessed October 21, 2019].
[33] M. Suleiman, G.I. Saidu, M. I. Ilyasu, O. Adeboye and M. Hamza , "Ultrasonic Fluid Level Measuring Device", International Journal of Recent Development in Engineering and Technology, Vol. 4, Special Issue 1, May 2015, [Online]. Available: http://www.ijrdet.com/files/ICMTSET2015/IJRDET_ICMTSET_01.pdf [Accessed October 21, 2019].
[34] Anthony Joseph Boscacci, "Emergency Signal Intercepting Unit", February 6, 2015, [Online]. Available: http://www.freepatentsonline.com/8928492.html [Accessed October 20, 2019].
[35] Technical Standards Committee of the Kapisanan Ng Mga Brodkaster sa Pilipinas, Technical Standards and Operating Requirements for FM Broacast Stations in the Philippines (1991, Revised)
[36] C. Zequiang, C. Nengcheng, D. Wenying, G. Jianya “An Active Monitoring Method For Flood Events” [Online]. https://ui.adsabs.harvard.edu/abs/2018CG....116...42C/abstract [Accessed January 20, 2019].
[37] “National Disaster Risk Reduction and Management Plan” [Online]. http://www.ndrrmc.gov.ph/attachments/article/41/NDRRM_Plan_2011-2028.pdf [Accessed January 20, 2019].
[38] F. C. C. Garcia, A. E. Retamar and J. C. Javier, "A real time urban flood monitoring system for metro Manila," (2015). [Online]. https://ieeexplore.ieee.org/document/7372990 [Accessed October 20, 2019].
[39] “How Raspberry Pi Is Different From A Desktop Computer” [Online]. https://raspberryinsider.com/how-raspberry-pi-is-different-from-a-desktop-computer/ [Accessed September 20, 2020]
[40] Hattersley. “Raspberry Pi 4 vs Raspberry Pi 3B+” (2020). [Online]. https://magpi.raspberrypi.org/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus [Accessed September 20, 2020]

[41] Hanan, A. Gunawan, M. Sumadiyasa "Water Level Detection System Based on Ultrasonic Sensors HC-SR04 and ESP8266-12 Modules with Telegram and Buzzer Communication Media" March (2019). [Online]. Available: https://www.researchgate.net/publication/336440730_Water_Level_Detection_System_Based_on_Ultrasonic_Sensors_HC-SR04_and_ESP8266-12_Modules_with_Telegram_and_Buzzer_Communication_Media

[42] What is an Ultrasonic Sensor? (2020), [Online]. Available: https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/#:~:text=As%20the%20name%20indicates%2C%20ultrasonic,between%20the%20emission%20and%20reception. [Accessed September 20, 2020]

[43] "Ultrasonic Sensors: Advantages and Limitations" (2019), [Online]. Available: https://www.maxbotix.com/articles/advantages-limitations-ultrasonic-sensors.htm/ [Accessed September 20, 2020]

[44] "Float Switch and Level Sensing Solutions For Agitated Liquid and Turbulence" (2015), [Online]. Available: https://www.fluidswitch.com/2015/06/16/float-switch-and-level-sensing-solutions-for-agitated-liquids-and-turbulence/ [Accessed September 20, 2020]

[45] J. Mitola. "The software-defined radio architecture" IEEE Communication Magazine, 33 (5) (May 1995), pp. 26-37. [Online]. Available: https://www.cs.odu.edu/~cs752/papers/sdr-003.pdf

[46] M.Heddebaut, JP. Ghys, M. Sanzc, and F. Elbahharab, "Road Traffic Information Using a Dedicated Radio Beacon", Transportation Research Part C: Emerging Technologies, Vol. 35, Pages 20-33, October 2013, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X13001253 [Accessed October 20, 2019].

[47] Salleh et al. (2013), "Design of Low Power Wideband Low Noise Amplifier for Software Defined Radio at 100 MHz to 1 GHz", Procedia Engineering, Vol. 53, Pages 368-375, 2013, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705813001677 [Accessed October 20, 2019].

[48] Walter H.W. Tuttlebee, "Software Defined Radio: Enabling Technologies", 2002, [Online]. Available: https://books.google.com.ph/books?hl=en&lr=&id=CyNCLZJCIs8C&oi=fnd&pg=PR5&ots=rEQ_ARObz9&sig=SG9cWM-fNue3QfiZpOXCutmPxT0&redir_esc=y#v=onepage&q&f=false [Accessed October 20, 2019].

[49] D. Curtis et al. (2019), "Software Defined Automotive Radar", Uhnder Inc, August 5, 2019, [Online]. Available: https://worldwide.espacenet.com/publicationDetails/biblio?II=1&ND=3&adjacent=true&locale=en_EP&FT=D&date=20190905&CC=US&NR=2019271776A1&KC=A1 [Accessed October 20, 2019].

[50] GNU Radio Hardware (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/Hardware [Accessed September 20, 2020]

[51] Nicholas Goldstein, "Override Transmitter-I.F.", October 8, 2016, [Online]. Available: https://patents.google.com/patent/US20060176172A1/en?q=car&q=radio+frequency&q=overriding&oq=car+radio+frequency+overriding [Accessed October 20, 2019].

[52] "Globe Labs API" [Online]. Available: http://www.globelabs.com.ph/#!/developer/api/sms [Accessed October 20, 2019].

[53] "What is GNU Radio?" (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F [Accessed September 20, 2020]

[54] GNU Radio FAQ (2020), [Online]. Available: https://wiki.gnuradio.org/index.php/FAQ [Accessed September 20, 2020]

[55] "Flood-Prone Areas in Metro Manila", June 10, 2018, [Online]. Available: https://www.hoppler.com.ph/magazine/featured-articles/flood-prone-areas-in-metro-manila [Accessed October 20, 2019].

[56] "What is Waterfall Model?" [Online]. Available: https://www.toolsqa.com/software-testing/waterfall-model/ [Accessed October 20, 2019].

[57] The Engineering Design Process" Available: https://www.slideshare.net/Fordlovers/the-engineering-design-process [Accessed October 20, 2019].

[58] Ultrasonic Distance Sensor [Online]. Available: https://www.sparkfun.com/products/15569 [Accessed July 24, 2020].

[59] Raspberry Pi 3 Model B+ [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/#:~:text=The%20Raspberry%20Pi%203%20Model,2.0%20(maximum%20throughput%20300%20Mbps) [Accessed July 24, 2020].

[60] Raspberry Pi 3G/4G & LTE Base HAT [Online]. Available: https://sixfab.com/product/raspberry-pi-base-hat-3g-4g-lte-minipcie-cards/ [Accessed July 24, 2020].

[61] HackRF One [Online]. Available: https://github.com/mossmann/hackrf/wiki/HackRF-One [Accessed July 24, 2020].

[62] TechTerms Python [Online]. Available: https://techterms.com/definition/python#:~:text=Python%20is%20a%20high%2Dlevel,use%2C%20even%20for%20commercial%20applications.&text=Python%20is%20considered%20a%20scripting,applications%20and%20dynamic%20Web%20content. [Accessed July 24, 2020].

[63] Ankur Tiwari, Should you host Python websites on PythonAnywhere?, January 21, 2020 [Online]. Available: https://medium.com/thoughtlytics/in-depth-review-should-you-host-python-website-on-pythonanywhere-594c31428336 [Accessed July 24, 2020].

[64] Globe Labs API [Online]. Available: https://www.programmableweb.com/api/globe-labs [Accessed July 24, 2020].
[65] PyCharm, June 21, 2020 [Online]. Available: https://en.wikipedia.org/wiki/PyCharm [Accessed July 24, 2020].
[66] Welcome to Raspbian [Online]. Available: https://www.raspbian.org/FrontPage [Accessed July 24, 2020].
[67] jQuery, 2020, [Online]. Available: https://jquery.com/ [Accessed July 24, 2020]
[68] M. Bostock. “d3.js”, Data-Driven Documents, 2019, [Online]. Available https://d3js.org [Accessed July 24, 2020]

Appendix B.

Design Project Poster

Adamson University
College of Engineering
Computer Engineering Department

Frequency Overriding System Through Signal Amplification For Flood Alert with SMS Notification

Rochelle G. Bastian, Elaine Grace S. Bayhon, Raymond Carl P. Chua, Emmanuel E. Jamoralin, Maria Concepcion A. Mirabueno

Philippines' geographical location and physical environment—located near the equator where the ocean is warm, one of the major factors for typhoon formation—contributes to the country's high-susceptibility to tropical cyclones. And with global warming, warmer waters will only continue to intensify the storms to ever hit the country. As the storm intensifies, the flood occurrence is becoming more often than before. Flood monitoring systems' main role is to inform and also to possibly save people's lives.

Currently, there are various researches about the importance and implementation of flood monitoring. Most of the flood monitoring system being deployed nowadays use social media as its platform, as it is one of the current popular medium of communication. Sending an alert through social media does reach a larger audience for just a few clicks away. However, in this study, the researchers considered the downside of relying primarily on the convenience of the Internet and social media.

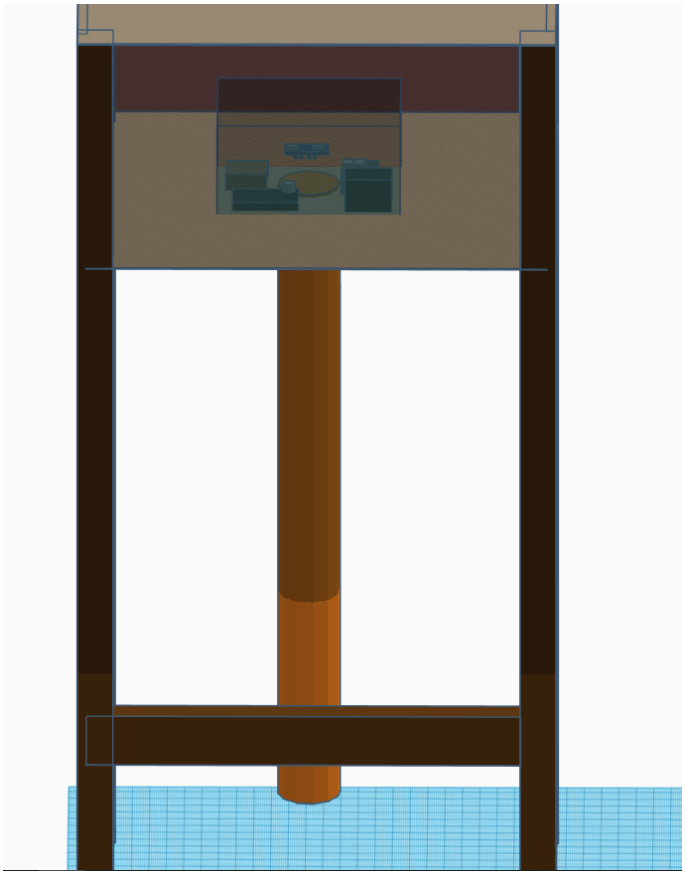
What the researchers have designed is a system that informs the people about the flood level in the area, with no internet connection needed. It is also designed to reach a specific audience by creating an SMS advisory system, which the subscribers can receive, and utilizing radio signals within a certain radius. In this study, the researchers will utilize FM radio frequency overriding and SMS advisory as the medium for flood level alerts to the people nearby and those who are subscribed.

The developed system can determine when to send SMS alerts and audio alerts show that the medium used to relay alerts are both SMS advisory and FM Radio. During the simulation of floodwater, for all levels 1-5, the system was able to send alerts through SMS. For emergency levels 4 & 5, the system also sent alerts through FM radio overriding.

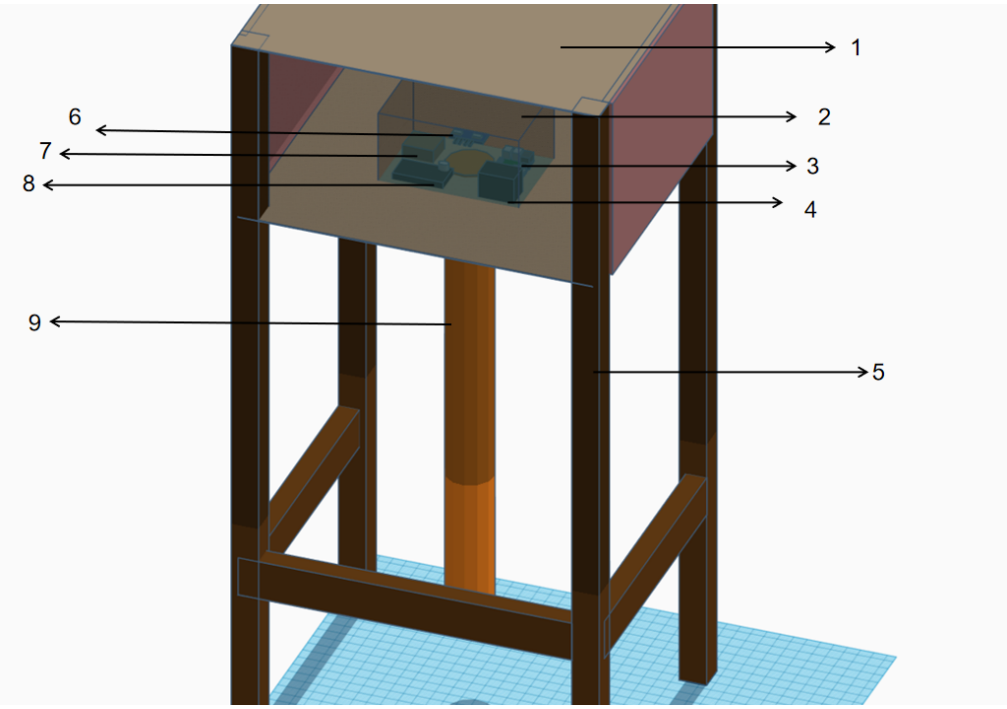
Appendix C.

Pictures of the Prototype

Actual Prototype

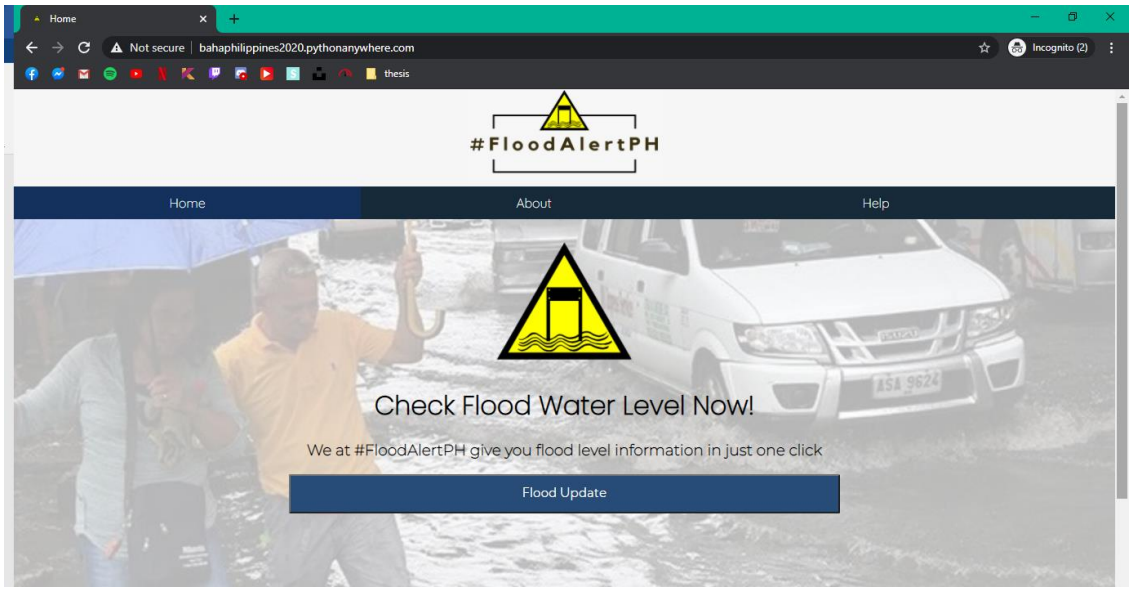


Prototype Design (Orthographic View)



Prototype Design (Isometric View)

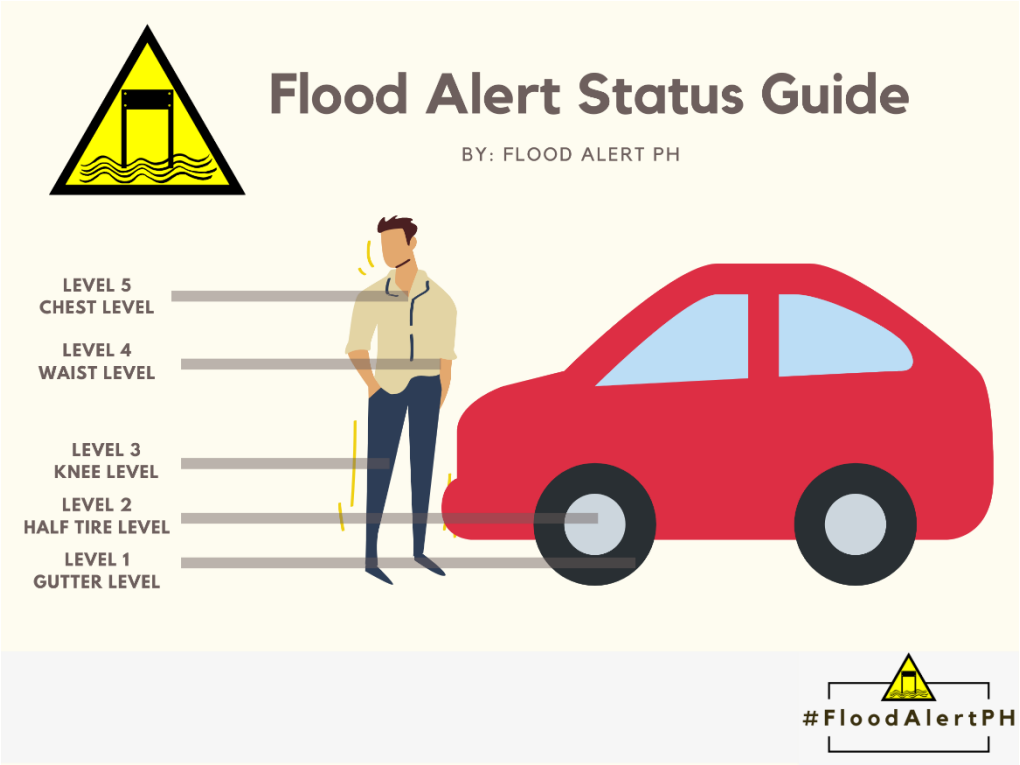
1 - Prototype box
2 - Prototype Stand
3 – IP 65 waterproof enclosure
4 – Raspberry Pi
5 – Power Bank
6 – Ultrasonic Sensor
7 – LTE Module
8 – Ultrasonic Sensor Circuit
9 – PVC Pipe

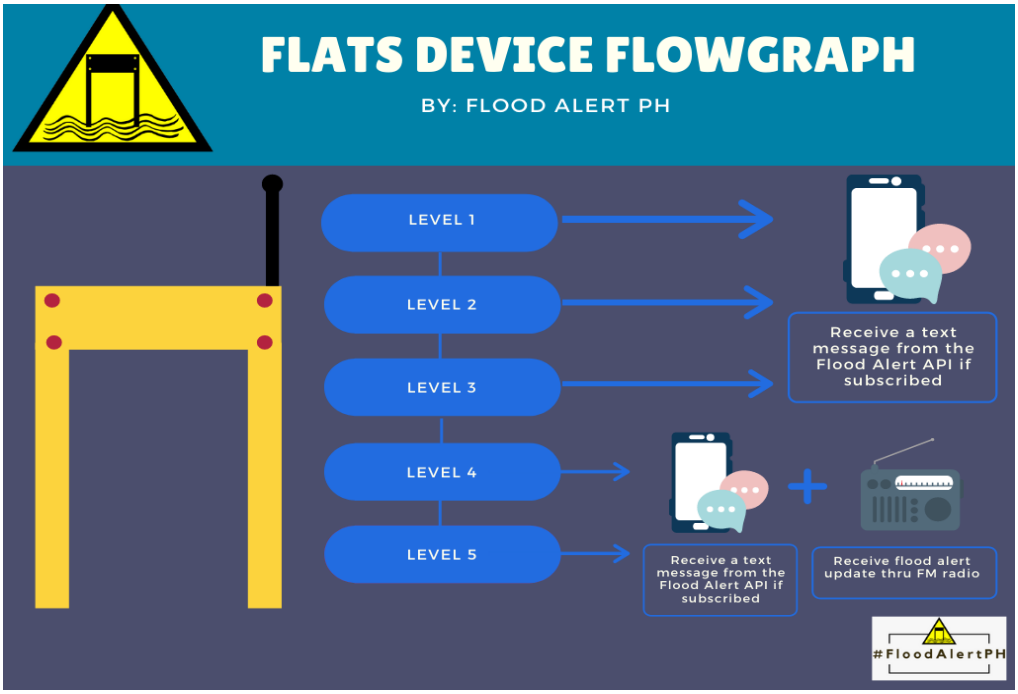


Flood Alert PH Website

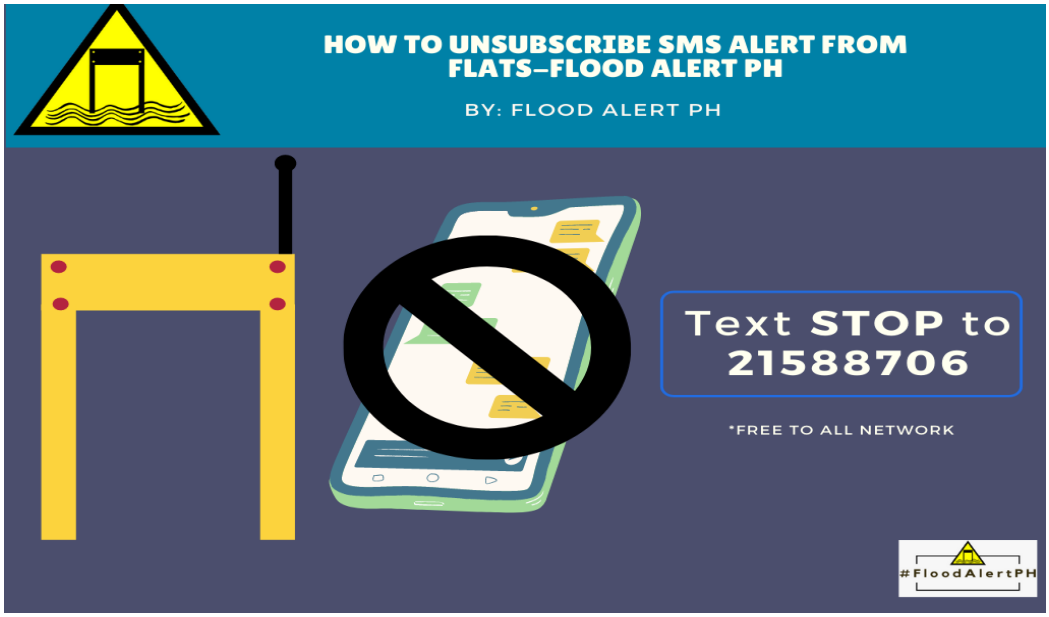
Appendix D.

User Manual


Flood Alert Status Guide
BY: FLOOD ALERT PH
LEVEL 5
CHEST LEVEL
LEVEL 4
WAIST LEVEL
LEVEL 3
KNEE LEVEL
LEVEL 2
HALF TIRE LEVEL
LEVEL 1
GUTTER LEVEL
#FloodAlertPH


FLATS DEVICE FLOWGRAPH
BY: FLOOD ALERT PH
LEVEL 1
LEVEL 2
LEVEL 3
LEVEL 4
LEVEL 5
Receive a text message from the Flood Alert API if subscribed
Receive a text message from the Flood Alert API if subscribed
Receive flood alert update thru FM radio
#FloodAlertPH


HOW TO SUBSCRIBE AND RECEIVE SMS ALERT FROM FLATS–FLOOD ALERT PH
BY: FLOOD ALERT PH
Text INFO to 21588706
To confirm, Text YES
*FREE TO ALL NETWORK
#FloodAlertPH


HOW TO UNSUBSCRIBE SMS ALERT FROM FLATS–FLOOD ALERT PH
BY: FLOOD ALERT PH
Text STOP to 21588706
*FREE TO ALL NETWORK
#FloodAlertPH

Appendix E.

Hardware Component Specification

COMPONENT	SPECIFICATION
 HC-SR04 Ultrasonic Sensor	• Operating voltage: +5V • Theoretical Measuring Distance: 2cm to 450cm • Practical Measuring Distance: 2cm to 80cm • Accuracy: 3mm • Measuring angle covered: <15° • Operating Current: <15mA • Operating Frequency: 40Hz
 Sixfab Base HAT	• Clip-in Mini PCIe socket for: • 4G/LTE Module (Quectel EC25) up to 150Mbps downlink and 50Mbps uplink data rates, GPS/GLONASS • 3G Module (Quectel UC20) up to 14.4Mbps downlink and 5.76Mbps uplink, GPS/GLONASS • Micro SIM card socket • USB – 1x micro USB port • Compatible with 40-pin Raspberry Pi header • Power Supply – 5V via micro USB port or external 5V source • Dimensions – 65 x 55 mm

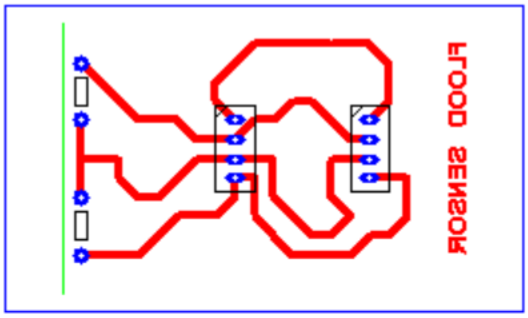
COMPONENT	SPECIFICATION
 ANT500 Telescopic Antenna	• Antenna Operation: 75MHz to 1GHz • Total length: 20cm to 88cm • 50 Ohm antenna designed to match HackRF One • Works for SDR reception, VHF, UHF, or ADSB
 Raspberry Pi 3b+	• CPU Type/Speed: ARM Cortex-A53 1.4GHz • RAM Size: 1GB SRAM • Integrated Wi-Fi: 2.4GHz and 5GHz • Ethernet Speed: 300Mbps • PoE: Yes • Bluetooth: 4.2

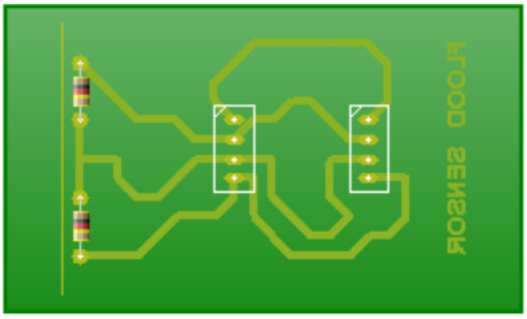
HackRF One SDR

- 1 MHz to 6 GHz operating frequency
- half-duplex transceiver
- up to 20 million samples per second
- 8-bit quadrature samples (8-bit I and 8-bit Q)
- compatible with GNU Radio, SDR#, and more
- software-configurable RX and TX gain and baseband filter
- software-controlled antenna port power (50 mA at 3.3 V)
- SMA female antenna connector
- SMA female clock input and output for synchronization
- convenient buttons for programming
- internal pin headers for expansion
- Hi-Speed USB 2.0
- USB-powered
- open source hardware

Appendix F.

Schematic Diagram and Layout





PCB Circuit Diagram for Ultrasonic Sensor

The ultrasonic sensor consist of 2 resistor (680 k and 1k) . The constant resistor in the circuit valued 1k resistance from the ground up to GPIO input. In order to protect the GPIO pins for possible damage from high input from the sensor(Echo Pin). It needed to have another resistor that the value can be determine by applying the voltage divider theorem.

Appendix G.

Data Sheet

HC-SR04 Datasheet

P. MARIAN

HC-SR04 is an ultrasonic ranging module that provides 2 cm to 400 cm non-contact measurement function. The ranging accuracy can reach to 3mm and effectual angle is < 15°. It can be powered from a 5V power supply.

HC-SR04 Specifications

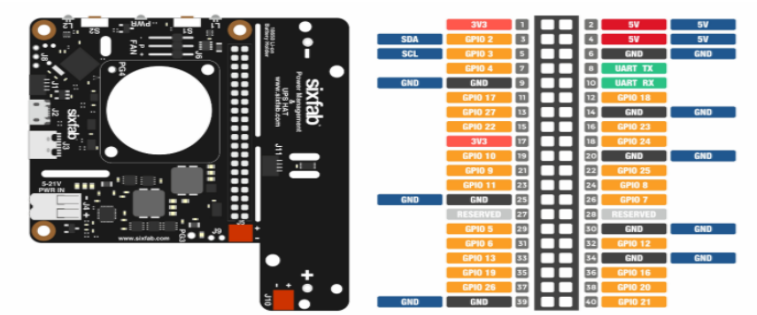
- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10µS TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Dimension 45 * 20 * 15mm

Technical Details

Hardware specifications

- Microchip ATSAMD21G18 ARM® Cortex®-M0+ Microcontroller
- Texas Instruments bq25703A Multi-Chemistry Battery Buck-Boost Charge Controller With System Power Monitor and Processor Hot Monitor
- Texas Instruments bq27441-G1 System-Side Impedance Track™ Fuel Gauge
- Texas Instruments INA219 Zerø-Drift, Bidirectional Current/Power Monitor

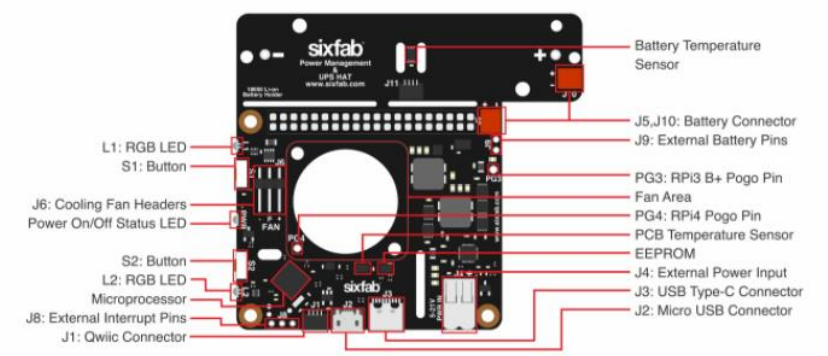
Pinout

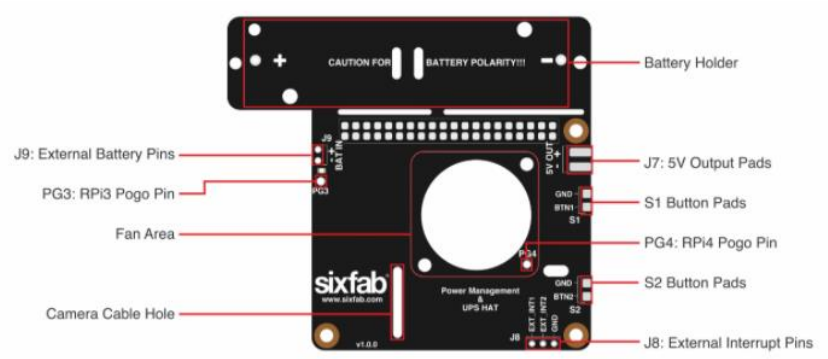


Pin Descriptions

Pin Number	BCM Pin	Pin Name	Description
8	UART_TX	PCI RX	This pin functions as the serial data input to the module for UART communication.
10	UART_RX	PCI TX	This pin functions as the serial data output from the module for UART communication
13	GPIO27	USER LED	Active HIGH, to switch on the USER LED, the pin's state should be HIGH.
15	GPIO22	USER BUTTON	This pin normally pulled-down to ground. When the button is pressed, pin switches to LOW.
31	GPIO6	RI	This pin is Ring indicator functions as the indication for receiving call or SMS, can be calibrated to HIGH or LOW using the AT commands.
33	GPIO13	DTR	When the module is in sleep mode, DTR pin allows to wake up the module up by pulling it to LOW.
35	GPIO19	W_DISABLE	This pin is used to turn Airplane Mode on the module, by pulling it HIGH.
37	GPIO26	HAT_PWR_OFF	Power regulator control. Normally pulled-down, when this pin drove to HIGH, Hat's power will cut off.

Layout





Power Inputs/Outputs

Inputs/Outputs	Description
J2	Micro USB 5V 2.5A max.
J3	USB Type-C 5V 3A max.
J4	Terminal Block 5V to 21V.
J7	5V rated 2.6A max output, solderable pads on the bottom side.

Power Supply/Adapter Selection

The recommended input voltage is 5V and the input current is 2A minimum. For the 12V input voltage, the minimum is 1.5A. We recommend the official Raspberry Pi power supply.

You can connect an external device to J7 for 5V output(underneath the HAT) and always check the polarity when you source via J7.

Use Single Input

Use only one of these three inputs(J2, J3, J4) at a time, i,e. Do not plug a USB(J2, J3) adapter while the power source is inserted through the terminal block(J4) or the other way.

Battery(Not Included)

The HAT comes with an integrated **18650 Flat Top Li-on battery** holder without a battery. It can be breakable if needed to use it in a different placement. Then you need to use the battery cable included in the package to connect the battery to the mainboard via **JST- SFH** connectors(J5 and J10)

You can plug an external **single-cell 3.7V Li-on or Li-po rechargeable batteries** when bigger capacities or special dimensions needed. Use **JST-SFH** connector(J5) with a suitable battery cable with it or you can solder cables to battery header(J9) by paying attention to the polarity.

> **Use One Battery**
>
> The UPS HAT has several Battery input options such as J5, J9.
> Do not use multiple Batteries at a time.

Buttons

There are two right-angle programmable push buttons(S1,S2) at the edge of the HAT.
The default behaviors of the buttons are as follows.

Behavior	Status
S1(Long Press)	Hard shutdown by cutting power of Raspberry Pi.
S1(Short Press)	Hard boot-up by powering Raspberry Pi.
S2(Long Press)	Soft shutdown via I2C then power off.
S2(Short Press)	Power on if power conditions are alright. (The pogo pin needs to be soldered)*

LEDs

There are three LEDs on the HAT as two of them are programmable RGB.

LED	Status
PWR	This red LED is ON when the Raspberry Pi is powered.
L1	Can be programmed as shows temperature level or heartbeat for the system.
L2	Used for battery charging level and status by default. Blinks if charging the battery, steady if the system powered by battery or fully charged

- Blinks if charging the battery, steady if the system powered by battery or fully charged.
- The color shows the percentage of the battery.
  - **RED** when the lower than 30%.
  - **YELLOW** between 30% and 60%.
  - **GREEN** when the battery is charged more than 60%.

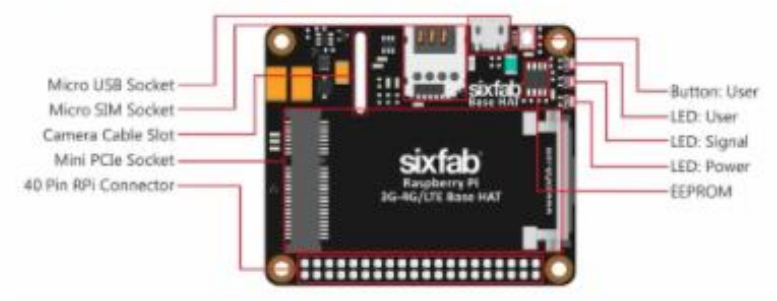
Qwiic Connector Note:
The Qwiic cable and the qwiic connectors(**J1-J11**) on board are used to read the temperature of the battery holder card when battery holder broke off from UPS HAT. This feature is under development.

The sensor on the battery uses **0x48** i2c address. Make sure it is not conflicting with another device using the same address.

Hardware Features

- True Uninterruptible Power Supply(UPS) battery rechargeable solution without Failing the Raspberry Pi.
- Compatible with Raspberry Pi A+, B+, 2B, 3B, 3B+, and 4 as well as Raspberry Pi Zero v1.3 and Raspberry Pi Zero Wireless.
- 3.9V to 21V input operating voltage range and open circuit voltage (Voc) up to 24V supports solar panels.
- 18650 Li-on Separable Battery Holder.
- 1-cell 3.7V External Battery Input.
- Compatible with Raspberry Pi Universal Power Supplies.
- Onboard Microcontroller manages scheduled power tasks, calendar events with Real-time Clock (RTC), gently shutdown or wake-up, sleep mode controls.
- Parallel Charging up to 3 Amps while powering the system.
- Watchdog Timer keeps the system alive when the Raspberry Pi freezes up.*
- Input, System, and Battery current monitor.
- Control over current limits of batteries and sources with 50mA resolution.
- Battery protection circuit enhanced with integrated temperature sensors.
- Thermal Shutdown and Input, System, Battery Overvoltage Protection.
- Dynamic power management boosts the system output from the battery if the needed power is more than the input supply provides.
- Onboard ARM® Cortex®-M0+ MCU manages scheduled power tasks with Real-time Clock (RTC), gently shutdown, or wake-up.
- Supports external GPIO interrupts to wake up or enter deep-sleep modes.
- Only I2C pins of Raspberry Pi used by HAT.
- Onboard STEMMA QT/Qwiic/JST SH connector.
- Programmable user buttons and RGB LEDs.
- Smart Cooling FAN with Pulse Signal Feedback.
- Battery level indicator LEDs with charging/discharging status.
- Slot for Raspberry Pi Camera Cable.
- Optional Pogo Spring Pins for hardware shutdown or wake up.
- Easy-to-use GUI to control features.*

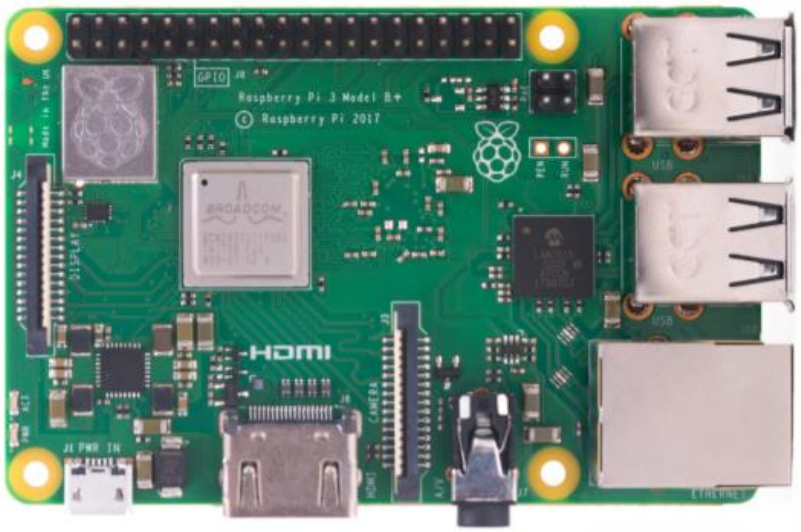
Layout



Compatible Mini PCIe Modules

- Quectel:
  - EC25 Mini PCIe 4G IoT Module
  - EC21 Mini PCIe 4G IoT Module
  - EC20 Mini PCIe 4G IoT Module
  - UC20 Mini PCIe 3G IoT Module
  - LTE-EP06
- Sierra
  - AirPrime MC Series
- Telit
  - LM960, LE910V2, HE910, LE910Cx, and more
- Huawei
  - ME909s-120, ME909s-821, and more
- Simcom
  - SIM7100, SIM7230, and more
- ZTE
  - ZM8620, and more
- U-Blox
  - MPCI-L2 Series
- Thales mini PCIe modules
  - PLS62W, mPLS8, mPLAS9

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5 V/2.5 A DC via micro USB connector ■ 5 V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50 °C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

Description

HackRF One, from Great Scott Gadgets, is a Software Defined Radio (SDR) peripheral capable of transmission or reception of radio signals from 1 MHz to 6 GHz. It covers many licensed and unlicensed ham radio bands. It is designed to enable test and development of modern and next generation radio technologies. HackRF One is an open source hardware platform that can be used as a USB peripheral or programmed for stand-alone operation.

HackRF One works like a sound card of computer. It processes Digital Signals to Radio waveforms allowing integration of large-scale communication networks. It is designed to test, develop, improvise and modify the contemporary Radio Frequency systems.

HackRF One has an injection molded plastic enclosure and ships with a micro USB cable. An antenna is not included. ANT500 is recommended as a starter antenna for HackRF One.

HackRF One is test equipment for RF systems. It has not been tested for compliance with regulations governing transmission of radio signals. You are responsible for using your HackRF One legally.

Features

•1 MHz to 6 GHz operating frequency
•half-duplex transceiver
•up to 20 million samples per second
•8-bit quadrature samples (8-bit I and 8-bit Q)
•compatible with GNU Radio, SDR#, and more
•software-configurable RX and TX gain and baseband filter
•software-controlled antenna port power (50 mA at 3.3 V)
•Open source hardware
•SMA female antenna connector
•SMA female clock input and output for synchronization
•convenient buttons for programming
•internal pin headers for expansion
•Hi-Speed USB 2.0
•USB-powered

Part List

1 x HackRF One
1 x Micro USB Cable

Documents

HackRF One Wiki
Source code and hardware design files
Github
FAQ
Learning SDR with HackRF

Appendix H.
Cost of Hardware Fabrication

Item	Quantity	Cost
Resistors	3	5.00
Ultrasonic sensor	1	100.00
Female pin header	1	20.00
Connecting wires (male)	10	50.00
Ckt board	1	110.00
Rapberry Pi 3b+	1	4,000.00
LTE HAT module	1	6,656.00
Hack RF One SDR Module	1	17,700.00
IP-65 Plastic Enclosure	1	850.00
Power bank	1	1500.00
Prototype	1	1694.00
	Total:	32,685.00

Appendix I.

Software Component Specification

Software	Specification
	• GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. • It can be used with readily-available low-cost external RF hardware to create software-defined radios
 Python	• Python is an open source high-level, general-purpose programming language. It allows users to distribute and contribute their own codes for public use. • Python provides a large standard library. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly. • The lightness of this interpreter made this a very good choice for prototyping small and big projects.

 Flask	• Flask is a micro web framework written in Python. It is capable of creating and running a simple web page. • Flask is more suited to smaller, less complicated applications. • Flask support SQLAlchemy it is a library that facilitates the communication between Python programs and databases. • Routing function of Flask makes it easier to navigate through the web pages.
 Bootstrap	• Front-end open source toolkit. • Offers extensive prebuilt components, and powerful JavaScript plugins. • It includes HTML and CSS based design templates for HTML components.
 D3	• JavaScript library for manipulating documents based on data. • Good library to use for data visualization in the front end. • Brings data to life using HTML, SVG, and CSS.

	• Small and fast JavaScript library • It can be used in HTML document manipulation, event handling, and animations to make it interactive. • Also offers prebuilt components like date picker.

Appendix J.

Source Code

1. Sensor Beacon Source Codes

get_sensor_height.py

- This python file gets the initial distance from sensor to the ground.

```
from sensor import get_distance

# distance = get_distance

def get_max_height():
    max_height = get_distance()
    return max_height

sensor_height = str(get_max_height())

f = open("sensor_height.txt", "w")
f.write(sensor_height)
print(sensor_height)
f.close()
```

request_post.py

- The codes inside request_post.py gets the actual level of the flood. The measurement of the levels and the location are being set in here. Once the floodwater reaches levels 1-5, it will send the data to the API.

```
#!/bin/bash
#This is working. Use this with the /posts page

import requests
from sensor import get_distance
import datetime
import time

# distance = find_distance()
location = "Boston Street"
scale = 1
level1 = 10 * scale #10
level2 = 15 * scale #19
level3 = 20 * scale #26
level4 = 25 * scale #37
level5 = 30 * scale #45

def get_max_height():
    max_height = find_distance()
    return max_height

def get_flood_height(previous_level):
    previous_level = previous_level
    feet_distance = None
```

```
    distance = round(float(sensor_height) - get_distance(), 2)
    feet_distance = str(round(distance/12, 2)) + " feet"
    print(distance)
    level = 0
    category_level = ""
    message = ""
    formated_datetime = datetime.datetime.now().strftime("%m/%d/%Y  %H:%M:%S")

    if distance > level5:
        level = 5
        category_level = "Waist Level"
        message = "Message From Flood Alert PH: As of {time}, the flood water level at {location} is at alert LEVEL 5: CHEST LEVEL {measurement}. Not passable to all vehicles. Please evacuate immediately.".format(time=formated_datetime, location=location, measurement=feet_distance)

    elif distance > level4:
        level = 4
        category_level = "Tire Level"
        message = "Message From Flood Alert PH As of {time}, the flood water level at {location} is at alert LEVEL 4: WAIST LEVEL {measurement}. Not passable to all vehicles. Please prepare for evacuation.".format(time=formated_datetime, location=location, measurement=feet_distance)

    elif distance > level3:
        level = 3
        category_level = "Knee Level"
        message = "Message From Flood Alert PH: As of {time}, the flood water level at {location} is at alert LEVEL 3: KNEE LEVEL {measurement}.  Not passable to light vehicles.".format(time=formated_datetime, location=location, measurement=feet_distance)

    elif distance > level2:
        level = 2
        category_level = "Half Tire Level"
        message = "Message From Flood Alert PH: As of {time}, the flood water level at {location} is at alert LEVEL 2: HALF TIRE LEVEL {measurement}. Passable to all types of vehicles.".format(time=formated_datetime, location=location, measurement=feet_distance)
    elif distance > level1:
        level = 1
        category_level = "Gutter Level"
        message = "Message From Flood Alert PH: As of {time}, the flood water level at {location} is at alert LEVEL 1: GUTTER LEVEL {measurement}. Passable to all types of vehicles.".format(time=formated_datetime, location=location, measurement=feet_distance)

    if level > 0 and level != previous_level:

        payload = {"Location": location,
                "ActualHeight": distance,
                "Level": level,
                "Message": message,
```

```
                "CategoryLevel": category_level,
                "FormatedDateTime": formated_datetime
                }

        r = requests.post('http://bahaphilippines2020.pythonanywhere.com/posts', json=payload)
        print(r.text)

    print("Level: {}".format(level))
    print("PLevel: {}".format(previous_level))
    previous_level = level
    print("AfterLevel: {}".format(previous_level))
    return level

f = open("sensor_height.txt", "r")
sensor_height = f.read()

#get_flood_height()
level = None
loading = ""
x = 1529
while True:
    while x < 100:

        if x % 2 == 0:
            loading = loading + "x"
        else:
            loading = loading + "+"
        x = x+2

        print("\r" + "Resting...  " + str(x) + "% Complete", end="")
        time.sleep(0.25)
    x = 0
    print('\nReading...')
    # Execute Command Here
    level = get_flood_height(level)
#    get_flood_height(level)
```

sensor_height.txt

44.0729434096

- This text file sets the total height of the sensor beacon; used to calculate for the total flood water height.

sensor.py

- This python script shows the connection of the ultrasonic sensor to the GPIO pins of the Raspberry Pi.

```
import RPi.GPIO as GPIO
```

```
import time

# 10 19 26 37 45
def get_distance():
    level = 0
    distance = 0
    GPIO.setmode(GPIO.BOARD)


    TRIG = 7
    ECHO = 12

    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.output(TRIG, 0)

    GPIO.setup(ECHO, GPIO.IN)

    time.sleep(0.1)

    print("Starting Measurement...")

    GPIO.output(TRIG,1)
    time.sleep(0.00001)
    GPIO.output(TRIG,0)

    while GPIO.input(ECHO) == 0:
        pass
    start = time.time()

    while GPIO.input(ECHO) == 1:
        pass
    stop = time.time()

    return((stop - start) * 17000 * 1/2.54) * 1.02

 #   def find_distance():
  #      return (stop - start) * 17000 * 1/2.54

    GPIO.cleanup()
```

2. Transmitter Station Source Codes

bash_test.sh

- This shell script runs multiple python scripts once called.

```
x=1
while [ $x -le 10 ]
do
```

```
  retn_value=$(python TextToSpeech.py)
  if [[ $retn_value -gt 3 ]]
  then
    echo "OVERRIDING!!"

    echo "Converting MP3 to wave file..."
    python convert.py

    echo "Transmitting audio to Radio..."
    echo "Transmitting to 90.7 FM"
    python Transmit_Freq1.py
    sleep 1
    echo "Transmitting to 91.5 FM"
    python Transmit_Freq2.py
    sleep 1
    echo "Transmitting to 97.1 FM"
    python Transmit_Freq3.py
    sleep 1

    echo "Overriding Complete"
    sleep 1
    echo "Cooling down..."
    sleep 180

  else
    echo "The level is equal or less than 3."

  echo "Resting..."
  sleep 10
done
```

convert.py

➢ These lines of codes convert the format of the audio that will be transmitted by the SDR, from mp3 to wav format.

```
from os import path
from pydub import AudioSegment

# files
src = "welcome.mp3"
dst = "welcome.wav"

# convert wav to mp3
sound = AudioSegment.from_mp3(src)
sound.export(dst, format="wav")
```

GetWebData.py

➢ The codes above use the python library Beautiful Soup to retrieve the html document of the webpage of FloodAlertPH in order to get the text message that will be converted from text to speech by the other python script.

```
import requests
from bs4 import BeautifulSoup

website = "http://bahaphilippines2020.pythonanywhere.com/flood_data"
class Website:
    def __init__(self, name):
        self.name = name

        webpage_response = requests.get(self.name)
        webpage = webpage_response.content
        soup = BeautifulSoup(webpage, "html.parser")

        contents = []
        turtle_links = soup.find_all("tr")
        i = 0
        for content in turtle_links[1]:
            contents.append(str(content).strip("<td>").strip(' class="col-md-10">').strip("</"))

        stripped_content = []
        #
        # print(turtle_links[1])
        i = 1
        while (i < 11):
            stripped_content.append(contents[i])
            i = i + 2

        string_content = ""

        for content in stripped_content:
            # print(content)
            string_content = string_content + str(content)

        self.time_date, self.height_inches, self.level, self.category_level, self.message = stripped_content

    def __repr__(self):
        return """Website: {website}
        Time: {time}
        Height: {height}
        Level: {level}
        Category Level: {category}
        Message: {message}""".format(website=self.name, time=self.time_date, height=self.height_inches,
level=self.level, category=self.category_level, message=self.message)
```

TextToSpeech.py

- The codes above convert the text message to speech, then saves it as mp3 file.

```
# Import the required module for text
# to speech conversion
# import subprocess
# from GetLatestPost
from gtts import gTTS

# This module is imported so that we can
# play the converted audio

# The text that you want to convert to audio
from GetWebData import Website

website = "http://bahaphilippines2020.pythonanywhere.com/flood_data"
web = Website(website)
print(web.level)
if web.level > 3:
    mytext = web.message
    mytext = mytext + " " + web.message

    # Language in which you want to convert
    language = 'en'

    # Passing the text and language to the engine,
    # here we have marked slow=False. Which tells
    # the module that the converted audio should
    # have a high speed
    myobj = gTTS(text=mytext, lang=language, slow=False)

    # Saving the converted audio in a mp3 file named
    # welcome
    myobj.save("welcome.mp3")

    # Insert trigger of SDR


# Playing the converted file
# os.system("vlc welcome.mp3")
# subprocess.call(['C:\Program Files\VideoLAN\VLC\vlc.exe', 'welcome.mp3'])
```

Transmit_Freq1.py

- Codes below shows the Python script equivalent of the flow graph generated from GNU Radio.

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
```

```
# Title: Transmit Freq1
# GNU Radio version: 3.7.13.5
##################################################

from distutils.version import StrictVersion

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from PyQt5 import Qt, QtCore
from gnuradio import analog
from gnuradio import audio
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.qtgui import Range, RangeWidget
from optparse import OptionParser
import osmosdr
import sys
import time
from gnuradio import qtgui

class Transmit_Freq1(gr.top_block, Qt.QWidget):

    def __init__(self):
        gr.top_block.__init__(self, "Transmit Freq1")
        Qt.QWidget.__init__(self)
        self.setWindowTitle("Transmit Freq1")
        qtgui.util.check_set_qss()
        try:
            self.setWindowIcon(Qt.QIcon.fromTheme('gnuradio-grc'))
        except:
            pass
        self.top_scroll_layout = Qt.QVBoxLayout()
     self.setLayout(self.top_scroll_layout)
        self.top_scroll = Qt.QScrollArea()
        self.top_scroll.setFrameStyle(Qt.QFrame.NoFrame)
        self.top_scroll_layout.addWidget(self.top_scroll)
```

```
        self.top_scroll.setWidgetResizable(True)
        self.top_widget = Qt.QWidget()
        self.top_scroll.setWidget(self.top_widget)
        self.top_layout = Qt.QVBoxLayout(self.top_widget)
        self.top_grid_layout = Qt.QGridLayout()
        self.top_layout.addLayout(self.top_grid_layout)

        self.settings = Qt.QSettings("GNU Radio", "Transmit_Freq1")
        self.restoreGeometry(self.settings.value("geometry", type=QtCore.QByteArray))

        ##################################################
        # Variables
        ##################################################
        self.samp_rate = samp_rate = int(2e6)
        self.FM_Frequency = FM_Frequency = 90.7e6

        ##################################################
        # Blocks
        ##################################################
        self._FM_Frequency_range = Range(87.0e6, 108.0e6, 1000, 87.5e6, 200)
        self._FM_Frequency_win = RangeWidget(self._FM_Frequency_range, self.set_FM_Frequency,
"FM_Frequency", "counter_slider", float)
        self.top_grid_layout.addWidget(self._FM_Frequency_win)
        self.rational_resampler_xxx_0 = filter.rational_resampler_ccc(
                interpolation=90,
                decimation=1,
                taps=None,
                fractional_bw=None,
        )
        self.osmosdr_sink_0 = osmosdr.sink( args="numchan=" + str(1) + " " + " " )
        self.osmosdr_sink_0.set_sample_rate(samp_rate)
self.osmosdr_sink_0.set_center_freq(FM_Frequency, 0)
        self.osmosdr_sink_0.set_freq_corr(0, 0)
        self.osmosdr_sink_0.set_gain(14, 0)
        self.osmosdr_sink_0.set_if_gain(47, 0)
        self.osmosdr_sink_0.set_bb_gain(0, 0)
        self.osmosdr_sink_0.set_antenna('', 0)
        self.osmosdr_sink_0.set_bandwidth(0, 0)

        self.blocks_wavfile_source_0 =
blocks.wavfile_source('/home/elainebayhon/Desktop/Final/welcome.wav', False)
        self.blocks_multiply_const_vxx_0 = blocks.multiply_const_vff((0, ))
        self.blocks_add_xx_0 = blocks.add_vff(1)
        self.audio_source_0 = audio.source(samp_rate, '', True)
        self.analog_wfm_tx_0 = analog.wfm_tx(
            audio_rate=96000,
            quad_rate=96000,
            tau=25e-6,
```

```
            max_dev=75e3,
            fh=-1,
        )

        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_wfm_tx_0, 0), (self.rational_resampler_xxx_0, 0))
        self.connect((self.audio_source_0, 0), (self.blocks_multiply_const_vxx_0, 0))
        self.connect((self.blocks_add_xx_0, 0), (self.analog_wfm_tx_0, 0))
        self.connect((self.blocks_multiply_const_vxx_0, 0), (self.blocks_add_xx_0, 1))
        self.connect((self.blocks_wavfile_source_0, 0), (self.blocks_add_xx_0, 0))
        self.connect((self.rational_resampler_xxx_0, 0), (self.osmosdr_sink_0, 0))

    def closeEvent(self, event):
        self.settings = Qt.QSettings("GNU Radio", "Transmit_Freq1")
        self.settings.setValue("geometry", self.saveGeometry())
        event.accept()

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.osmosdr_sink_0.set_sample_rate(self.samp_rate)

    def get_FM_Frequency(self):
        return self.FM_Frequency

    def set_FM_Frequency(self, FM_Frequency):
        self.FM_Frequency = FM_Frequency
        self.osmosdr_sink_0.set_center_freq(self.FM_Frequency, 0)

def main(top_block_cls=Transmit_Freq1, options=None):

    qapp = Qt.QApplication(sys.argv)

    tb = top_block_cls()
    tb.start()
    tb.show()
    time.sleep(45)
    quitting()

    def quitting():
        tb.stop()
        tb.wait()
    qapp.aboutToQuit.connect(quitting)
    qapp.exec_()
```

```
if __name__ == '__main__':
    main()
```

3. Web Server (pythonanywhere)

static: custom.css

- This CSS code sets the design of the flood data page of the website.

```
.row {
    margin-right: -15px;
    margin-left: -15px;
}

.container {
    padding-right: 15px;
    padding-left: 15px;
    margin-right: auto;
    margin-left: auto;
}

.header{
        background-color: #f6f6f6;
}

.nav{
    background-color: #172a3a;
    font-family: 'Quicksand', sans-serif;
}

.nav .nav-link:hover {
    background-color: #8da9c4;
    color: #ffffff;
}

.display-5{
    font-family: 'Quicksand', sans-serif;
    font-size: 48px;
}
```

static: style2.css

```
@charset "utf-8";
/* CSS Document */
body{
 margin: 0;
 border: 0;
}
```

```
.header{
	background-color: #f6f6f6;
}

.topnav ul{
  list-style-type: none;
  margin: 0;
  padding: 0px;
  overflow: hidden;
  background-color: #172a3a;
  font-family: 'Quicksand', sans-serif;
}

.topnav li{
  float: left;
}

.topnav li a{
	display: block;
  color: white;
  text-align: center;
  text-decoration: none;
	padding: 10px 190.3px;
	margin: 0;
}


/* Change the color of links on hover */
.topnav a:hover {
    background-color: #8da9c4;
    color: #ffffff;
}

/* Add a color to the active/current link */
.topnav a.active {
  background-color: #13315c;
  color: #ffffff;
}
```

static: style.css

- The CSS code below sets the design for the about and help page of the website.

```
body{
 margin: 0;
 border: 0;
}
```

```
.header{
	background-color: #f6f6f6;
}


.topnav ul{
  list-style-type: none;
  margin: 0;
  padding: 0px;
  overflow: hidden;
  background-color: #172a3a;
  font-family: 'Quicksand', sans-serif;
}

.topnav li{
  float: left;
}

.topnav li a{
	display: block;
  color: white;
  text-align: center;
  text-decoration: none;
	padding: 10px 190.3px;
	margin: 0;
}


/* Change the color of links on hover */
.topnav a:hover {
    background-color: #8da9c4;
    color: #ffffff;
}

/* Add a color to the active/current link */
.topnav a.active {
  background-color: #13315c;
  color: #ffffff;
}


.column {
  float: center;
  width: 50%;
}

 /*Clear floats after the columns */
```

```
.row:after {
  content: "";
  display: table;
  clear: both;
}


.content {
  background-image: url("/static/mnlflood1.png");
  height: 500px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  position: relative;
}

.info h1, h3, block{
  color: black;
}

.info h1{
        margin: auto;
        font-family: 'Poppins', sans-serif;
}

.info button{
        font-family: 'Kumbh Sans', sans-serif;
}
.info h3{
        font-family: 'Montserrat', sans-serif;
}

.block {
  display: block;
  width: 50%;
  border-radius: 2px;
  background-color: #274c77;
  padding: 14px 28px;
  font-size: 16px;
  cursor: pointer;
  text-align: center;
        color: white;
        margin: auto;
}
```

templates: about.html

- This HTML document is the structure of the about page where the description and the developers of FloodAlertPH can be found.

```
<!doctype html>
<html>
<head>
<link rel="stylesheet" href="/static/style2.css">
<link                    href="https://fonts.googleapis.com/css2?family=Roboto:wght@500&display=swap"
rel="stylesheet">
<link                  href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300&display=swap"
rel="stylesheet">
<link                     href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
<link                  href="https://fonts.googleapis.com/css2?family=Montserrat:wght@200&display=swap"
rel="stylesheet">
<link               href="https://fonts.googleapis.com/css2?family=Kumbh+Sans:wght@300&display=swap"
rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta charset="utf-8">
<title>About</title>
</head>

<body>

<div class="header">
                    <center><img src= "/static/A.png"  alt="" width="239" height="115"> </center>
          </div>


<div class="topnav">
                    <ul>
                              <li><a href={{url_for('home')}} >Home</a></li>
                      <li><a href={{url_for('about')}} class="active">About</a></li>
                      <li><a href={{url_for('guide')}}>Help</a></li>
                    </ul>
          </div>
<div class="content">
<center><img src= "/static/about1.png"  alt=""  width="100%" > </center>
<center><img src= "/static/about2.png"  alt=""  width="100%" > </center>
<center><img src= "/static/about3.png"  alt="" width="100%" > </center>

</div>

<div class="footer">
          <img src= "/static/footer.png"  alt="" width="100%" height="100" >
</div>

</body>
</html>
```

templates: data.html

➢ This HTML document includes the table of data that Raspberry Pi sends to the API. It mirrors the data stored in the database table, named Announcements.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
   <title>Flood Update</title>
   <link rel="stylesheet" href="/static/custom.css">
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
   <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
   <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300&display=swap">

</head>
<body>
   <div class="header">
            <center><img src= "/static/A.png"  alt="" width="260" height="130"> </center>
         </div>
   <div>
      <nav class="nav flex-column flex-sm-row nav-pills">
        <a class="flex-sm-fill text-sm-center nav-link text-light rounded-0"
href={{url_for('home')}}>Home</a>
        <a class="flex-sm-fill text-sm-center nav-link text-light rounded-0"
href={{url_for('about')}}>About</a>
        <a class="flex-sm-fill text-sm-center nav-link text-light rounded-0"
href={{url_for('guide')}}>Help</a>
      </nav>
   </div>
   <div>
      <h1 class="display-5 pl-3 pt-5 col text-center">Flood Update</h1>
   </div>
   <div class="wrapper container">
      <div class="row">
        <div class="col-md-5 pl-4 pt-5">
          <table class="table">
            <thead>
              <tr>
                <th>Datetime</th>
                <th>Height</th>
                <th>Level</th>
                <th>Category</th>
```

```
                    </tr>
                </thead>
                <tbody>
                    {% for announcement in announcements.items %}
                    <tr>
                        <td>{{announcement.formateddatetime}}</td>
                        <td>{{announcement.actualheight}}</td>
                        <td>{{announcement.level}}</td>
                        <td>{{announcement.categorylevel}}</td>
                        <!--<td>{{announcement.message}}</td>-->
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
        <div class="col-md-5 pt-5">
            <p class="pl-3 ml-3">Date: <input type="text" id="datepicker" name="datetime"></p>
            <br>
            <div class="dropdown pl-3 ml-3">
            <button class="btn btn-outline-dark dropdown-toggle" type="button" id="dropdownMenuButton"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Plot by</button>
                <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
                    <a class="dropdown-item" id="Day">Day</a>
                    <a class="dropdown-item" id="Month">Month</a>
                    <a class="dropdown-item" id="Year">Year</a>
                </div>
            </div><br>
            <div id="linechart"></div>
        </div>
    </div>
    <div class="col text-center">
        {% for page_num in announcements.iter_pages() %}
        {% if page_num %}
            <a class="btn btn-outline-dark mb-4 "href="{{url_for('flood_data', page=page_num)}}">{{
page_num }}</a>
        {% endif %}
        {% endfor %}
    </div>
    </div>
    <div class="footer">
        <center><img src= "/static/footer.png"  alt="" width="70%" height="90"> </center>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
```

```
   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
   <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
   <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
   <script src="https://d3js.org/d3.v4.js"></script>
   <!--<script src="{{ url_for('static', filename='linechart.js') }}"></script>-->
</body>
</html>
```

templates: help.html

- This HTML document consists of infographics that serves as a guide for the users.

```
<!doctype html>
<html>
<head>
<link rel="stylesheet" href="/static/style2.css">
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@500&display=swap"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300&display=swap"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@200&display=swap"
rel="stylesheet">
<link href="https://fonts.googleapis.com/css2?family=Kumbh+Sans:wght@300&display=swap"
rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta charset="utf-8">
<title>Help</title>
</head>

<body>

<div class="header">
        <center><img src= "/static/A.png" alt="" width="239" height="115"> </center>
</div>


<div class="topnav">
        <ul>
                <li><a href={{url_for('home')}} >Home</a></li>
                <li><a href={{url_for('about')}} >About</a></li>
```

```
                    <li><a href={{url_for('guide')}} class="active">Help</a></li>
            </ul>
</div>

<div class="content">
<center><img src= "/static/F1.png"  alt=""  width="100%" > </center>
<center><img src= "/static/help1.png"  alt=""  width="100%" > </center>
<center><img src= "/static/help2.png"  alt=""  width="100%" > </center>
<center><img src= "/static/help3.png"  alt="" width="100%" > </center>

</div>

<div class="footer">
            <img src= "/static/footer.png"  alt="" width="100%" height="100" >
</div>
</body>
</html>
```

templates: home.html

- This HTML document shows the structure of the home page of the website.

```
<!doctype html>
<html>
<head>
<link rel="stylesheet" href="/static/style.css">
<link                    href="https://fonts.googleapis.com/css2?family=Roboto:wght@500&display=swap"
rel="stylesheet">
<link                  href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300&display=swap"
rel="stylesheet">
<link                     href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
<link                  href="https://fonts.googleapis.com/css2?family=Montserrat:wght@200&display=swap"
rel="stylesheet">
<link               href="https://fonts.googleapis.com/css2?family=Kumbh+Sans:wght@300&display=swap"
rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta charset="utf-8">
<title>Home</title>
</head>

<body>

<div class="header">
                    <center><img src= "/static/A.png"  alt="" width="239" height="115"> </center>
            </div>
```

```
<div class="topnav">
                    <ul>
                                <li><a href={{url_for('home')}} class="active">Home</a></li>
                                <li><a href={{url_for('about')}}>About</a></li>
                                <li><a href={{url_for("guide")}}>Help</a></li>
                    </ul>
          </div>

<div class="content">
          <div class="info">
                    <center>
                                <img src="/static/LOG.png" width="300" height="200" alt="">
                                <form       action="http://bahaphilippines2020.pythonanywhere.com/flood_data"
method="get">
           <label><h1> Check Flood Water Level Now!</h1></label>
           <label><h3>We at #FloodAlertPH give you flood level information in just one click</h3></label>
           <button type="submit" value="submit" class="block">Flood Update</button>
                                </form>
                    </center>

          </div>
</div>

<div class="footer">
          <img src= "/static/footer.png"  alt="" width="100%" height="100" >
</div>

</body>
</html>
```

main.py

➢ The codes in main.py connects the front end and the back end. Here lie the routes of the web server, the connection from database, and imports the functions from other python scripts.

```
from flask import Flask, render_template, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from outbound import outbound
from plot import plot_daily, plot_monthly, plot_yearly
app = Flask(__name__)
app.config["DEBUG"] = True

SQLALCHEMY_DATABASE_URI                                                                    =
"mysql+mysqlconnector://{username}:{password}@{hostname}/{databasename}".format(
   username="BahaPhilippines2",
   password="thesis2020",
   hostname="BahaPhilippines2020.mysql.pythonanywhere-services.com",
```

```
    databasename="BahaPhilippines2$BahaDB",
)
app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False

db = SQLAlchemy(app)

class Announcement(db.Model):

    __tablename__ = "Announcements"

    id = db.Column(db.Integer, primary_key=True)
    actualheight = db.Column(db.Float())
    level = db.Column(db.Integer)
    message = db.Column(db.String(250))
    posted = db.Column(db.DateTime, default=datetime.now)
    categorylevel = db.Column(db.String(200))
    formateddatetime = db.Column(db.String(200))
    location = db.Column(db.String(200))


class Subscribers(db.Model):
    __tablename__ = 'Subscribers'

    id = db.Column(db.Integer, primary_key=True)
    access_token = db.Column(db.String(250))
    subscriber_number = db.Column(db.String(100))

    def save_subscriber(self):
        db.session.add(self)
        db.session.commit()

    def delete_subscriber(self):
        db.session.delete(self)
        db.session.commit()


@app.route("/homepage")
def index():
    return render_template("home_page.html")


@app.route('/flood_data', methods=['GET'])
def flood_data():
    page = request.args.get("page", 1, type=int)
    announcements = Announcement.query.order_by(Announcement.posted.desc()).paginate(page=page,
per_page=20)
```

```
    return render_template('data.html', announcements=announcements, title='subscribers'), 200

@app.route('/announcements', methods=["GET"])
def announcement():
    return                                                    render_template("announcement_page.html",
Announcements=reversed(Announcement.query.all()))


@app.route('/globe/', methods=["GET"])
def opt_in():
    access_token = request.args.get("access_token")
    subscriber_number = request.args.get("subscriber_number")
    new_subscriber = Subscribers(access_token=access_token,
                                 subscriber_number=subscriber_number)

    if access_token is not None and subscriber_number is not None:
        Subscribers.save_subscriber(new_subscriber)

    subscribers = reversed(Subscribers.query.all())
    return render_template('subscribers.html', subscribers=subscribers, title='subscribers'), 200


@app.route('/globe/', methods=['POST'])
def stop_subscription():
    data = request.get_json()
    access_token = data['unsubscribed']['access_token']
    subscriber = Subscribers.query.filter_by(access_token=access_token).first()
    Subscribers.delete_subscriber(subscriber)
    subscribers = Subscribers.query.all()
    return render_template('subscribers.html', subscribers=subscribers, title='subscribers'), 200


@app.route('/posts', methods=['POST'])
def posts():
    req_data = request.get_json()

    ActualHeight = format(req_data['ActualHeight'],  '.3f')
    Level = req_data['Level']
    CategoryLevel=req_data['CategoryLevel']
    FormatedDateTime = req_data['FormatedDateTime']
    Location = req_data['Location']
    MessageFromRpi = req_data['Message']


    dateTimeObj = datetime.now()
    timeStr = dateTimeObj.strftime("%m/%d/%Y  %H:%M:%S")
```

```
    # Message = "As of {}. Ang sukat ng tubig sa Consolacion St. ay {} inches at itinataas na ang warning level sa Level {} or {}".format(FormatedDateTime, ActualHeight, Level, CategoryLevel)
    # Message = MessageFromRpi

    data = Announcement(actualheight=ActualHeight, level=Level, message=MessageFromRpi, categorylevel=CategoryLevel, formateddatetime=FormatedDateTime, location=Location)
    db.session.add(data)
    db.session.commit()

    # Message = "Ang sukat ng tubig sa Consolacion St. ay {} inches at itinataas na ang warning level sa {} or {}   /n/n {}".format(ActualHeight, req_data['Message'], laymanlevel,  current_date.posted)

    if Level > 0:
        subsribersID = Subscribers.query.all()
        for subscriber in subsribersID:
            outbound(Message = MessageFromRpi, access_token = subscriber.access_token, subscriber_number = subscriber.subscriber_number)


    return '"The Actual Height is {}.
    The Level is {}.
    The message: {}.
    Time Posted: {}.
    Category Level : {}.
    '".format(ActualHeight, Level, MessageFromRpi, datetime.now, CategoryLevel)

@app.route('/home', methods=['GET'])
def home():
    return render_template('home.html')


@app.route('/about', methods=['GET'])
def about():
    return render_template('about.html')


@app.route('/help', methods=['GET'])
def guide():
    return render_template('help.html')


@app.route('/day', methods=['GET'])
def get_day_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    month = date[0]
    day = date[1]
```

```
    year = date[2]

    data = plot_daily(month=month, day=day, year=year)

    return jsonify(data)


@app.route('/month', methods=['GET'])
def get_month_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    month = date[0]
    year = date[2]

    data = plot_monthly(month=month, year=year)
    return jsonify(data)


@app.route('/year', methods=['GET'])
def get_year_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    year = date[2]

    data = plot_yearly(year=year)
    return jsonify(data)
```

outbound.py

➢ This set of code is the one that sends the payload to the Globe API for them to be able to send the subscribers the message.

```
import requests

def outbound(Message, access_token, subscriber_number):


    shortcode = '21589064'
    access_token = access_token
    address = subscriber_number
    clientCorrelator = '21587128'
    message = Message

    url = "https://devapi.globelabs.com.ph/smsmessaging/v1/outbound/"+shortcode+"/requests"

    querystring = {"access_token":access_token}
```

```
    payload = "{\"outboundSMSMessageRequest\": { \"clientCorrelator\": \""+clientCorrelator+"\", \"senderAddress\": \""+shortcode+"\", \"outboundSMSTextMessage\": {\"message\": \""+message+"\"}, \"address\": \""+address+"\" } }"
    headers = { 'Content-Type': "application/json", 'Host': "devapi.globelabs.com.ph" }

    response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

    # print(response.text)
    return response.text

def get_day_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    month = date[0]
    day = date[1]
    year = date[2]

    data = plot_daily(month=month, day=day, year=year)

    return jsonify(data)


@app.route('/month', methods=['GET'])
def get_month_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    month = date[0]
    year = date[2]

    data = plot_monthly(month=month, year=year)
    return jsonify(data)


@app.route('/year', methods=['GET'])
def get_year_plot():
    get_date = request.args.get("date")

    date = get_date.split('/')
    year = date[2]

    data = plot_yearly(year=year)
    return jsonify(data)
```

outbound.py

```
import requests
```

```
def outbound(Message, access_token, subscriber_number):

    shortcode = '21589064'
    access_token = access_token
    address = subscriber_number
    clientCorrelator = '21587128'
    message = Message

    url = "https://devapi.globelabs.com.ph/smsmessaging/v1/outbound/"+shortcode+"/requests"

    querystring = {"access_token":access_token}

    payload = "{\"outboundSMSMessageRequest\": { \"clientCorrelator\": \""+clientCorrelator+"\",
\"senderAddress\": \""+shortcode+"\", \"outboundSMSTextMessage\": {\"message\": \""+message+"\"},
\"address\": \""+address+"\" } }"
    headers = { 'Content-Type': "application/json", 'Host': "devapi.globelabs.com.ph" }

    response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

    # print(response.text)
    return response.text
```

plot.py

```
import requests

def outbound(Message, access_token, subscriber_number):

    shortcode = '21589064'
    access_token = access_token
    address = subscriber_number
    clientCorrelator = '21587128'
    message = Message

    url = "https://devapi.globelabs.com.ph/smsmessaging/v1/outbound/"+shortcode+"/requests"

    querystring = {"access_token":access_token}

    payload = "{\"outboundSMSMessageRequest\": { \"clientCorrelator\": \""+clientCorrelator+"\",
\"senderAddress\": \""+shortcode+"\", \"outboundSMSTextMessage\": {\"message\": \""+message+"\"},
\"address\": \""+address+"\" } }"
    headers = { 'Content-Type': "application/json", 'Host': "devapi.globelabs.com.ph" }

    response = requests.request("POST", url, data=payload, headers=headers, params=querystring)

    # print(response.text)
    return response.text
```

Appendix K.

Completed Testing Forms

*Testing tables found in Chapters 3 and 4.

Appendix L.

Evidence of Proofreading

ADAMSON UNIVERSITY

COLLEGE OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

PROOFREADING CERTIFICATION

This is to certify that this thesis manuscript titled:

Frequency Overriding System Through Signal Amplification for Flood Alert with SMS Notification

Prepared by:

Bastian, Rochelle G.

Bayhon, Elaine Grace S.

Chua, Raymond Carl P.

Jamoralin, Emmanuel E.

In partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER ENGINEERING**

has been proofread by the undersigned.

Jophiel Marie A. Onas	UST, BA Major in English Language Studies
Signature over Printed Name	Degree / Affiliation
jophielonas@gmail.com	September 28, 2020
Contact Number / Email	Date

Adamson University
Computer Engineeing Department
2F Ozanam Building, 900 San Marcelino St. Ermita 1000
Manila, Philippines (02)8524-20-11 loc 409