

Implementation of Smart Marine Security System
Department of Electrical and Electronic Engineering
&
Department of Computer Science and Engineering

A Thesis

Submitted to the Department of Electrical and Electronics Engineering & Department of
Computer Science and Engineering

Of

BRAC University

By

Hamid Bin Zaman (12101020)

Moontasir Hassan (12121033)

Supervised by

Dr. Md. Khalilur Rhaman

Associate Professor

Department of Computer Science and Engineering

BRAC University, Dhaka



Declaration

We do here by declare that the thesis titled ‘Implementation of Smart Marine Security System’ is submitted to the Department of Electrical and Electronic Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronics Engineering. This is our original work and was not submitted elsewhere for the award of any other degree or any other publication.

Date: 20.08.2015

Authors

Dr. Md. Khalilur Rhaman

Thesis Supervisor

Associate Professor

Department of Computer Science and Engineering

BRAC University

Hamid Bin Zaman (12101020)

Moontasir Hassan (12121033)

Acknowledgement

We would like to take this opportunity to express our gratitude to Dr. Md. Khalilur Rhaman for his guidance, and instruction that he has given us from the time we have been doing the thesis project ‘Implementation of Smart Marine Security System’ under his supervision through the completion of our undergraduate program. The skills we have able to learn from him during our Thesis work have benefited us immensely and will continue to do so throughout our future endeavors. We also want to show our appreciation towards our mechanical helper Mr. Mohammad Munir for his huge exertion in building the vital structures that were utilized for drawing out this venture towards achievement.

Abstract

A standout amongst the most repeating and uprising issue in our nation is marine mishaps. Because of this disaster a great many individuals loses their lives sometimes. The real purposes for this issue are over-burden travelers boarding the boat and climate conditions along the stream course. One answer for this issue would be keep a check on the marine vessels and keep a track of the courses that it navigates. Therefore, we are interested in establishing the connections between a low cost marine black box and a web database and be able to store necessary data in it for monitoring the marine vessels. The database would be used to monitor marine vessels traversing the rivers carrying the passengers and as well as will act as a receiver of emergency signals that can provide the alert of taking necessary precautions and measures during accidents. To aid this operation we would be further integrating this system with android application software that would be able to fetch the necessary data from this web database and therefore would provide the end users with present conditions of the ship by constantly updating latest data from the web database. This can also provide warnings and alerts to facilitate preparation of necessary actions against marine accidents.

Table of Contents

TOPIC	PAGE NO
CHAPTER: 1 INTRODUCTION	6 - 15
1.1 Objective	6
1.2 Scenario in Bangladesh	7
1.3 Motivation	9
1.4 Literature Review	10
CHAPTER: 2 SYSTEM OVERVIEW	16 - 22
2.1 General overview of the System	17
2.2 Black Box	18
2.3 On Board Processor	19
2.3.1 Weather Station	19
2.3.2 Passenger Counter	20
2.4 Trigger Unit	21
2.5 Web Database and Android Application	22
CHAPTER: 3 SYSTEM IMPLEMENTATION	23 - 51
3.1 Black Box Design	23
3.1.1 Hardware	23
3.1.2 Control	24
3.1.3 Communication	25
3.1.4 Power	26
3.2 Onboard Processor	27
3.2.1 Weather Station	27
3.2.1.1 Control	28
3.2.1.2 Communication	29
3.2.1.3 Power	29
3.2.2 Passenger Counter	30
3.2.2.1 Control	31
3.2.2.2 Communication	32
3.2.2.3 Power	32
3.3 Trigger Unit	32
3.3.1 Water Level Sensor	32
3.3.2 Thrower Cannon/Trigger	33
3.3.3 Output	33
3.3.4 Control	34
3.3.5 Power	34

3.4 Software	35
3.4.1 Web Database	35
3.4.2 Android Application	41
CHAPTER: 4 DATA ANALYSIS	52 - 59
4.1 On Board Processor Analysis	52
4.1.1 Weather Data Analysis	52
4.1.2 Passenger Counter Analysis	54
4.2 Trigger Unit Efficiency Analysis	55
4.3 GPS Data Accuracy	56
4.4 Application Response Time vs. Internet Speed	58
4.5 Compatibility to various Android API	59
CHAPTER: 5 DISCUSSION	60 -64
CHAPTER: 6 REFERENCE	65 - 66
CHAPTER: 7 APPENDIX	67 - 81

Chapter - 1

Introduction

1.1 Objective

The Implementation of Smart Marine Security System's fundamental target is to find and research the essential reasons of the awful marine mischances. In the vast majority of the cases we see that the reason gets into covert as a result of the low quality hardware utilized as a part of the water vessel. In addition at last in the vast majority of the cases the casualty vehicle is not in the least found and discovered. Our goal is to make a framework by which we can assess and at most cases take a thought of the primary motivation to the particular mischance which can without a doubt be an essential approach to determine the issue and make the mishap rate low in a few ways. On the other hand again in the most cases the Black Box can find the influenced vehicle and measure of harm done to it because of the mishap.

Another target is to make an attention to the Water Vessel proprietors and drivers about the challenges, climate turbulences in some specific range and courses where they can keep away from the course in some basic climate figures. It can likewise be exceptionally useful to make the mishap issue reasonable to the significant piece of Water vessel driver and partners who are not all that actually stable. We discovered in our studies that amid turbulent climate conditions exceptional change happens to the Temperature, Pressure and Humidity in the particular territory or course. Utilizing the readings of past mischance situations the driver and partners can be prepared in a decent manner to give them a chance to take in the definite technic of disposing of those turbulent and potentially perilous conditions and circumstances.

In the large portion of the instances of mishaps now a days it is extremely hard to recognize the accurate number of travelers who are really influenced by the mischance. In addition in greatest cases the mishaps happen because of exorbitantly swarmed water vessels. For example, PINAK-6 had really more than 5 times a bigger number of travelers than the dispatch could hold. Yet, no confirmation arrived and the persons who were liable for this couldn't be rebuffed because of

absence of innovation there. Indeed, even numerous dead collections of the casualty couldn't be found because of intemperate flow through Padma River. By including a component called Passenger Counter our Black box can give the exact number of voyagers who were boarded and even under the minimum great conditions possible it can careful the explorers to not to board in a water vessel which is stuffed. It is also one of the basic focuses of our Low cost marine black box. Another target is to look for what and how adaption exercises should be considered in water transport structure organizing, endeavor progression, operations and upkeep and what are the variables and particular essentials to support possible and capable keeping up and directing marine transportation.

1.2 Scenario in Bangladesh:

Bangladesh is a riverine country and channels are key technique for correspondence in this locale. Since a long time, conduit framework has been seen as secured and monetarily astute course especially in the southern bit of Bangladesh. Reliably through this course (Bangladesh Inland Water Transport Authority) bears 87.80 million voyagers. This imperative technique for transport is ridden with stunning disasters reliably, bringing about a considerable toll of human lives. Around 3,869 people have gone on and 279 turned up lost in 458 dispatch disasters since 1976 (Department of Shipping). The inland courses of Barisal, Bhola, Chandpur and Patuakhali and their joined channels to Dhaka and Chittagong are seen to be clumsier. Boundless operation of unfit vessels, over-troubling of explorer, enlistment of unskilled gatherings, poor breaking point of critical government bodies and low standard backing of Inland Water Transport (IWT) channels are beginning these deadly accidents. There are a couple sorts of inland water transports working all through the country. For straightforwardness, these are considered under seven particular characterizations as: Payload Ships, Traveler Launches, and Traveler Trawlers, Traveler steamers, Payload Trawlers, Ships, Motor Boats, Nation Boats and Others

Payload vessels are very basic level greater vehicles which are made of steel body and every now and again arranged with sub-divisional bulkheads to give water coziness to the cargo holds. Similarly, payload water crafts contain cargo hatch openings on the upper deck through which the things are being stacked and discharged. Moreover some cargo pontoons contain free pumping workplaces to stack or vacant liquid cargoes on or off the heap opening

The voyager dispatches are generally made of steel structure with no allocated payload holds. Instead of evident burden compartments, voyager dispatches contain tinier private hotels to give some excess and security to the rich explorers. Regardless, in the lion's share of the explorer dispatches there stay broad open spaces on the decks where the economy class explorers live scattered in the midst of a voyage. It justifies saying that both payload pontoons and voyager dispatches are sketched out with mechanical or water driven controlling and fundamentally being used for medium to long partition going in Bangladesh.

Voyager trawlers and cargo trawlers are both relative kind of vehicles where the principle complexity lies on what they pass on in the midst of their voyages; i.e. in case they pass on explorers in the midst of a voyage, they are allocated as voyager trawlers and if they pass on burden, they are called as payload trawlers. In a far-reaching way these vehicles are sweeping wooden vessels with some steel plating took after at the outside skin besides have engines mounted at the posterior .Most of them contain neither payload holds nor explorer hotels beside a few encased spaces for the groups and along these lines, these vehicles are used for medium to short partition voyaging.

The engine watercrafts are likely the most standard technique for transportations for medium to short detachment voyaging. Such vessels are wooden made and instigated by agrarian multipurpose engines which are consistently known as shallow engines. These vehicles are respectably tinier than trawlers however noticeably greater than littler country watercrafts. No compartments or encased spaces are found in this sort of vessels and in a general sense these vessels have emerge deck to pass on explorers and their things and controls physically using by provincial models made rudders.

Country watercrafts in Bangladesh are various in numbers and changed in sorts with rich traditions that take after back quite a long while into the past. In any case, the consistent trademark that most of the country vessels have is that each one of them are non-mechanized and physically moved. The greater part of the country barges have the obtainment of being towed by the wind power using particularly ordinary looking sails, particularly in the inland waters of the country. Bangladesh has a broad mixed bag of marine vehicles numbers and in sorts

In Bangladesh, marine accidents have turned out to be most normal points in the features of the daily papers these days. There are different purposes for these mishaps like over-burden of travelers, extreme climate conditions and so forth. These mishaps happen on account of careless oversights however the impacts of the same are enduring and waiting. Numerous individuals passed on in our nation due to marine mishaps and we have seen that the boat couldn't be followed and additionally dead bodies couldn't be found. As of late, the impact between the oil tanker and the freight vessel happened at the Sela River in Sundarbans, Khulna division, Bangladesh. The boat sank and a few individuals passed on. Besides, an oil slick happened that lead numerous creatures and trees to die. This is by all account not the only illustration; there are different occurrences too of marine mishaps. So we choose to create a marine black box system that would be exceptionally proficient in identifying discriminating results of marine mishaps furthermore this will be practical.

1.3 Motivation

A powerful and proficient Smart marine system is a framework by which the entire thought of looking the unlucky water vessels which have been influenced by the maritime mishaps can be changed. In Fall 14 we turned out with the arrangement of building an ease gadget which can be recognized by the salvage groups of the deplorable casualty water vessels. Really before that the deplorable mishap of PINAK-6 happened in the course of MAWA – KAWRAKANDI course and at last because of the low administration framework and the low prepared frameworks the PINAK-6 dispatch was not identified and safeguarded from the waterway of PADMA. The real measure of travelers who died in the mischance was not likewise could be identified. We were truly enlivened to make a gadget that can at any rate recognize the entire reasons for mishap, all out travelers in the vessel, extreme reason of the mischance and so on. With our Marine black box and the elements we utilized it will be practically clear that why such mishap could be happened. Truly after that we went to a couple locales and read about the genuine recovery and examination technique for the BIWTA division of the organization who are fundamentally the power to research such sorts of setbacks. Regardless, consequent to encountering a couple locales and every one of those substance, we were genuinely confused watching at the old style of carrying data and with the usage of old rigging in a vast bit of the cases. The investigations were so back dated that by those things we could comprehend that why the Marine mishaps

are not researched appropriately. The main thing that took our consideration was the Sonar framework that has brought here from abroad. The intriguing and stunning part is that the upkeep of the station is not all that simple, that is the reason it obliges master professional's to control that. So from that exact instant we came to understand that the arrangement of BLACK BOX which are regularly utilized as a part of Air Crafts, if these are executed here in the water vessels requiring little to no effort utilizing nearby specialized help, it can be more less demanding to locate the unfortunate vehicle for the sonar framework or our salvage administrators or less master professionals to make utilization of this Low Cost Marine Black Box. So thinking for a Low cost form of Black box which will be financially savvy, versatile structure ,will have computerized information procurement framework, simple upkeep framework and reasonable in the connection of the monetary status of our nation.

Consistently we turn into the casualties of such disastrous Marine Accidents that is the reason we must consider this post mischance examination thing in a genuine note in the result of our Launch Drivers, Owners furthermore travelers can be more mindful and make the maritime course generally safe of influencing by Marine mishaps in Bangladesh. These Marine Accidents reason influence the life of hundreds and a large number of individuals of our nation consistently, which regularly prompts passing and immense loss of water vessel base. So to determine the smoldering issues a Low cost Marine Black Box has been produced that permits having the successful components to end up mindful and take pre-alerts to improve our life and in the meantime in an extremely reasonable expense for the water vessel proprietors in Bangladesh

1.4 Literature Review:

At the beginning of the examination for the undertaking we mulled over a proposal paper named ‘Riverine passenger vessel disaster in Bangladesh: Options for Mitigation and safety’. From that specific paper we came to think about the dangers and perils of the riverine courses of Bangladesh. We were likewise helped by the data of the explanations for the mishaps and the impact of the specific episodes on our economy and populace. We additionally comprehended the actuality of the measure of individuals and vital products for our economy are utilizing the marine course without huge measure of security measures. We figured out from the theory paper

huge number of 25% marine mishaps are a direct result of over-burdening furthermore critical number of 52.9% mishaps in Bangladesh happens because of climate issues (till 1974-2009). Indeed, even these high numbers couldn't make the power move to minimize the mishaps. Most shockingly we came to realize that by our studies that this area is in fact one of the slightest created segment in our nation. Another stunning part we came to know from daily papers and a few studies that high rate of 37.4% water vehicles influenced by mishaps couldn't be found through most recent 38 years. Also, unending terrible individuals were died and lost in those mishaps. These data really supported our goals to build up a framework where the mishaps ought not to be puzzling any longer and it gets to be simpler to low the measure of undetermined water vehicles influenced by mishaps. In the meantime make the particular division put under critical innovation.

In the initial, a theory paper named A Distance Machinery Controlling and Monitoring Guardian has been concentrated on. From this paper we came up to think about diverse purposes for marine mishaps. We have discovered wellbeing scope of machines which incorporates heat emanation, delivered sound and vibration. There were distinctive sorts of mishaps that happen in the boat motor room they are Crankcase Explosion of ship's motor, over speeding of generators, heater blast, Compressor Airline blast, High weight fuel line blasting, High weight steam spillages, Hydraulic high weight segments blasting, turbo charger blast, Electrical shock, Accidental CO₂ discharge.

For the Black Box part, we considered the unique metal that would resilient on the water as well as all that much water verification in within to make it an effective black box. We proceeded with a few papers about black box of aeroplanes yet no place had we discovered the resilient element. So at long last we utilized the Stainless Steel Ball which is 10 inch in breadth. Yet, we needed to consider the toughness of that thing we accompanied an answer about the strength by the theory paper named 'Minimal effort Car Black Box'. We utilized stainless Steel because of great sturdiness, stainless component and a definitive shock retaining force of the particular component. We could utilize more lavish tough molecule however we needed to make it extremely solid in the meantime it must be financially savvy. We again encountered a couple locales and papers we found that a complete Discovery definition would not sit on our endeavor. So we needed to make our way contrastingly where we can reload the arrangement of

examination in a specialized however in the meantime proficient to the less experienced field laborers.

Subsequent to being certain about the Black Box Design we needed to experience a few papers and connections which are identified with the information that we need to store or protect operating at a profit box. For that we have worked in parallel with another group named '**Low cost Marine Black Box**' to get the hardware and electrical part of the whole system. In the long run we figured out a few papers and daily paper articles where we discovered the center reasons of marine mishaps. We discovered more often than not stuffed water vessels experience the mishaps and measure of casualties the vast majority of the times couldn't be made sense of. So by using sonar sensors we made a equipment called Passenger Counter which is used to keep track of travellers who are boarding on the water vessel or dispatch. We can likewise make sense of the most extreme traveler that a water vessel or dispatch can take load. Also, surpassing that heap a notice ringer rings each time it surpasses the point of confinement.

Besides the climate inside of the course is a standout amongst the most imperative reasons of marine mishaps. We figured out by a few papers and sites as references that an intense change in air, basically in the temperature, mugginess and weight happens previously, then after the fact the tempest and tornedos occur. In our nation we all realize that storms and tornedos are a standout amongst the hugest reasons of the mishaps. So in the event that we continue following the climate conjecture each time by on load up climate station, it would assume an imperative part to track the reason of the mischances. So we chose and accompanied an on board weather station by utilizing particular sensors and to send the information to black box we additionally added to a Radio Frequency information transferable system by actualizing the RF sensors.

One of the elements of the black box is identifying the spot where the mishap has happened and the boat has sunk. In our nation the vast majority of the cases we see that the influenced vehicle is not really figured out. It is essential to discover the real vehicle to make the future events not to happen or make the rate of mishaps lower. With a specific end goal to do that, we have composed a Microcontroller based following and stockpiling framework where we pick Arduino Uno R3 microcontroller. We moreover used GSM and GPS to take after the spot. GPS can take after the spot and with the help of GSM a substance will go to the cell phones about the discovered spot. We have utilized GSM shield unit with module and GPS Shield with GPS

module. Likewise, the work of upgrading the territory of Google Maps is done. So at last we make it extremely adaptable with the view of some GPS and GSM related papers and distributions and related with our work. At any rate conceivable circumstance the black box can characterize the position of the mischance area and make it obvious on a Google guide or even tell the course of the influenced vehicle's position. We really balanced it inside the Black Box and sent the RF signal receivable from the on board climate station to focus the definite reason of the mishap happened to the Marine Vessel. Again we have additionally utilized RF sign to get the information from traveler counter to the genuine black box. Inside the black box alongside the GPS and GSM modules we set up an information store system where every one of those information of traveler counter and climate station are being stored with the help of SD card module. Each one of those information are likewise can be gotten to by a WEB database on a consecutive organization.

A standout amongst the most essential stage is to make the black box visible to rescuers. To do that part we have examined the fundamental material science standard of trebuchet which was really used to toss something to a specific separation. Really that trebuchet thing is indigent to the mass which it is appended to. In any case, we composed a thrower standard which is not really reliant on mass but rather subject to the responsive power connected by the spring. Really the versatility of the spring was utilized to do every one of the components. It tosses the entire black box by the power to a certain separation. An exceptionally solid and shock proof string is appended to both the water vehicle and Black box to really figure out the influenced vehicle. At first we utilized 120 feet of string. Until that profundity the string will turn out easily from the center we used to curl the string. Besides it is critical to make the trigger to make the thrower standard work. By a few investigations of actuator and spring system we figured out that we can utilize a water level sensor to make the thrower work like precisely a trigger unit. We built up three water levels on which each water level have its own detecting force of the water level that is perilous or not. At the risky point the water level the sensor triggers the actuator to open the lock from the mechanical gadget and triggers it on to the water level.

We finally built up an android application to make the entire framework work like a live monitoring system following framework where we can without much of a stretch see the ebb and flow condition of the water vessel and if the vehicle is influenced by a mischance or not. The

android application in like manner display the present range, atmosphere station updates, out and out voyagers thus on kind of data that are secured in the SD card and pushes on the web.

Before developing the Android application for our project, one of the many published papers we went through was "Implementation of Location based Services in Android using GPS and web services". From this particular article we studied about many benefits of the location based services for the end users in terms of retrieving the current location and process that data in order to acquire more useful information. Mobile devices using A-GPS and through web services using GPRS can also get handful of information such as routing information and nearby locations. We also got an idea about how to implement Location based services with the Google web services and built in Android APIs.

Another paper regarding such issues we have gone through was "Developing Android Mobile Map Application with Standard Navigation Tools for Pedestrians" where we learnt how it has been possible to develop applications using maps because of the mobility of the mobile devices. Though this paper provides suggestions regarding pedestrian navigation, we got a clear idea about the standard navigation tools that has already been developed. A handful number of limitations have been mentioned in this paper which helped us understand many important aspects regarding development of Android application using Google map services.

Since we were initially unsure about the format of interchanging data between web database and Android application, we read articles about interactions between them. A paper named "Accessing External Databases from Mobile Applications" demonstrated such an access is complicated since any queries cannot be run like often done in local databases since our raw data interface is remote. This paper solves the complication using Model-View-Controller (MVC) software design pattern and shows how communication can be established with remote databases irrespective of the platform used to develop the Mobile application which in our case is a native mobile application i.e. android.

Again to make things clearer we studied another paper named "Attendances System for College Students" which primarily discusses about the tools and APIs provided by Android. The paper also discuss how Android application interchange data i.e. send and retrieve from remote

database using JSON objects in detail. From this particular article we also came to know how Android connect with Web services.

Chapter - 2

System Overview

In fall'2014, we started planning to do our project in such a way that we could be successful. Our first step was working with microcontroller. We choose Arduino Uno R3 and started to learn the basic things about it. One of the features of our marine black box is to track the place where the accident has occurred. So to fulfill our purpose, we worked with GSM and GPS. After that, we worked with GPS Shield with GPS module and were successful to track a place. Lastly, we combined both GPS and GSM with Arduino in order to locate a place and send the information to mobile phones through message. Now we are working to update the location to Google Maps.

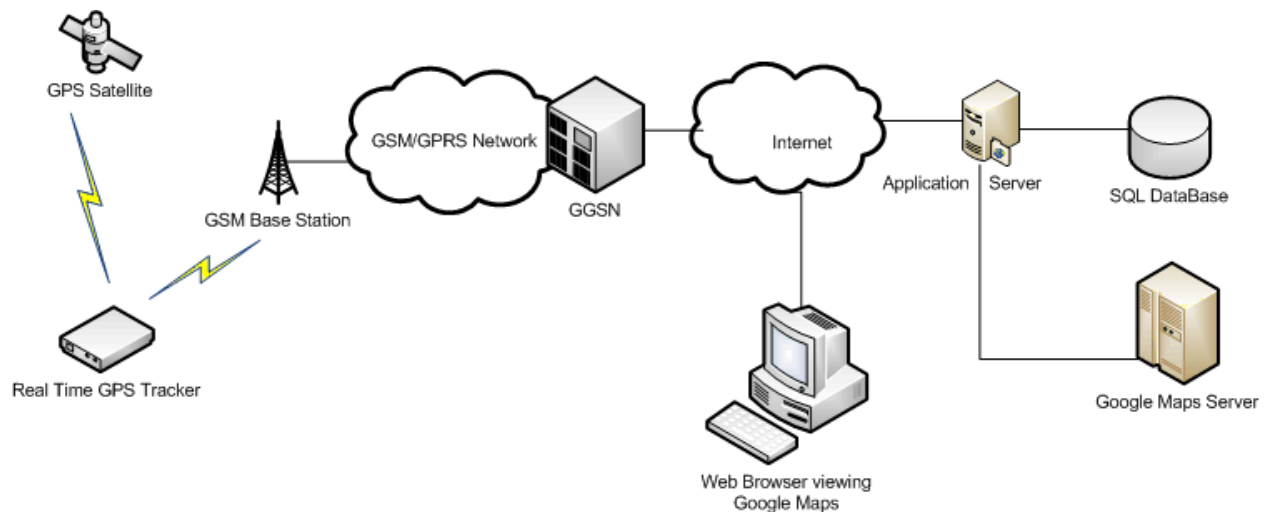


Figure 1.1: Overall Server system

So after every one of those experiments and research we made about the entire framework we really separated the entire framework into distinctive parts and began taking a shot at it. We are clarifying the general outline framework savvy in the accompanying piece of the section.

2.1 General Overview of the System:

The entire framework has Black box, weather station, Passenger counter and trigger unit. Firstly the weather station and passenger counter which are mutually approached board processor sends their particular information into the black box. Both the transmission is one way. The container gets the GPS area from the satellite. The weather station information, passenger counter information and GPS area is encouraged to the Web database by the container. This sending is a one way information transmission. The black box joined with a string is put on a round molded holder of the trigger unit.

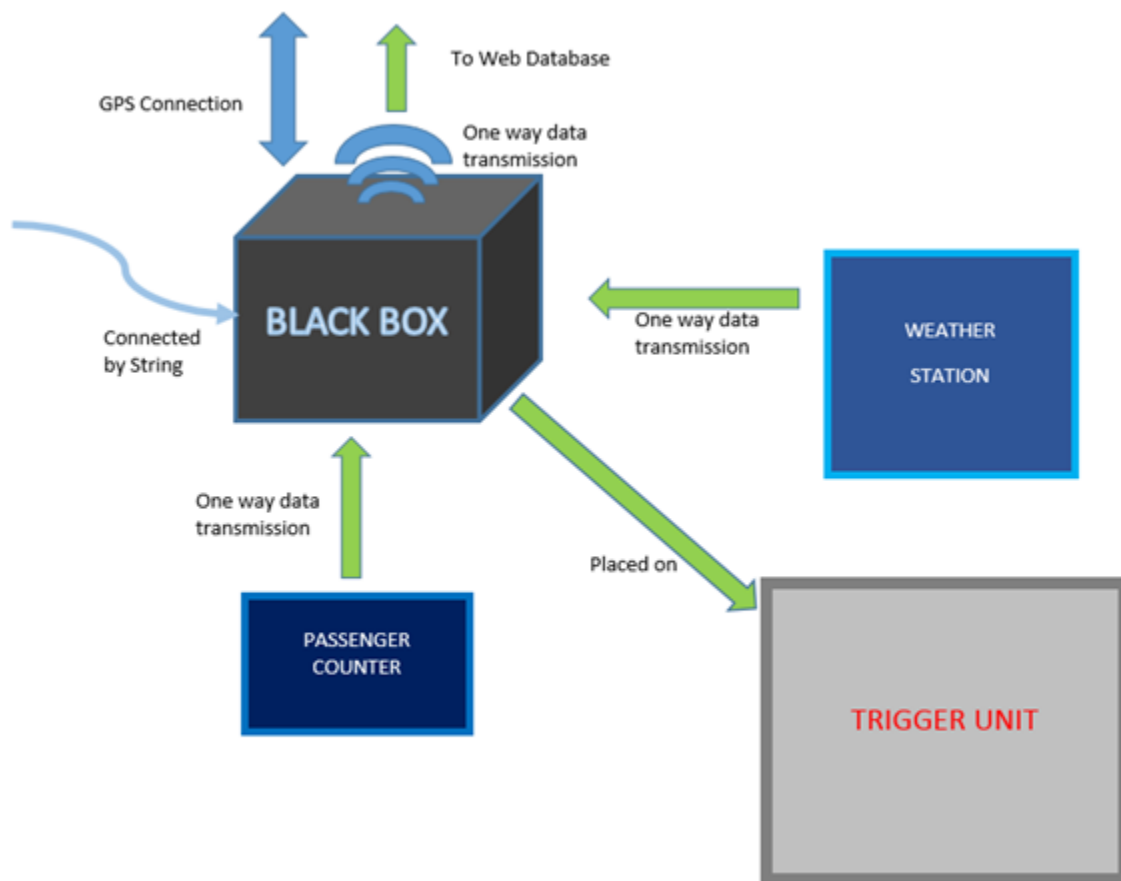


Figure 1.2: Overall system

2.2 Black Box:

Black box is a circular shaped sphere which consists of a GPS, GSM, SD card shield, and RF receiver. Arduino mini and UNO. The data of weather station sensors and passenger counter is received in the RF receiver with the help of Arduino Mini and it goes to the Arduino UNO. The GPS coordinates are collected via GPS and Arduino UNO. These coordinates and the sensor data of weather station are saved into a SD card through SD card shield for safe keeping. At the same time these data are sent into GSM. The function of GSM is to send all these data into the Web database for keeping a record. Then an Android device with an Android application is used to fetch these data and display GPS location in Google Maps and other values in the application.

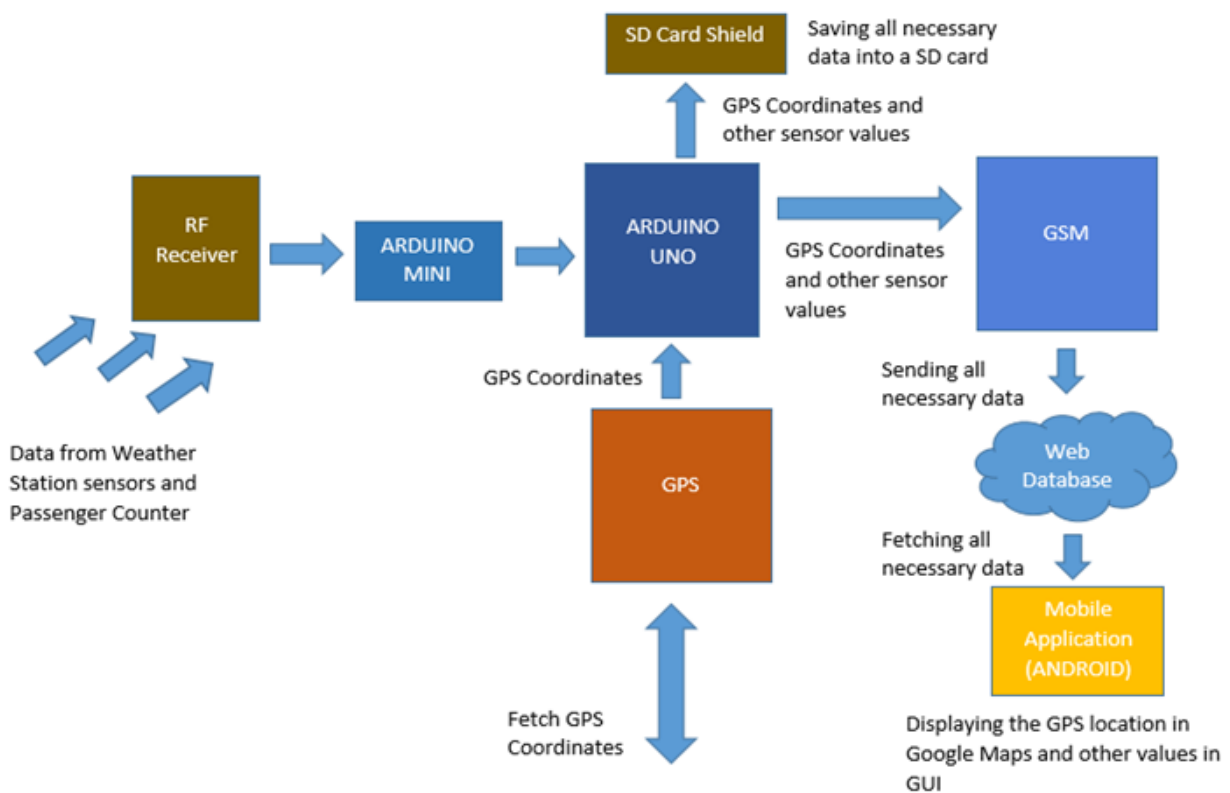


Figure1.3: Black box

2.3 On Board Processor:

The On board processor consist of weather station and passenger counter. On board processor components actually works independently, collects data and with the help RF transmitters the whole data is sent to the Black Box using the RF receivers.

2.3.1 Weather Station:

Temperature and Humidity sensor, air pressure sensor, RF transmitter, Arduino mini and UNO all together defines the weather station. The temperature, humidity and air pressure readings are taken with the help of an Arduino UNO. These readings are sent into the RF receiver of the black box unit via RF transmitter with the help of Arduino Mini, which in turn get processed, saved in the SD card and gets forwarded to the web database.

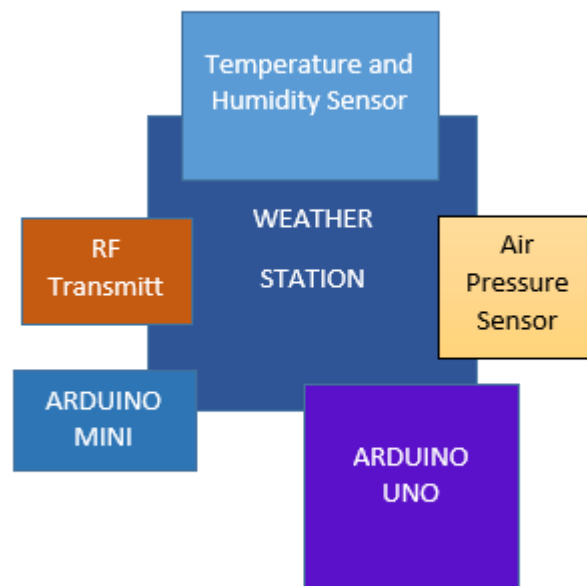


Figure 1.4: Weather station

2.3.2 Passenger Counter:

Passenger counter calculates and tracks the approximate total of the number of passengers present inside the ship at a given time. Two sonar sensors and Arduino UNO are used to add the passenger going inside the ship and also deduct the passenger coming out of the ship. So, at any given time the total number of passengers on board is being calculated. A RF transmitter is also placed along with the sonar and Arduino in order to send this data into the RF receiver of the Black box. This is again being processed in the black box, saved in the SD card and redirected to the web database.

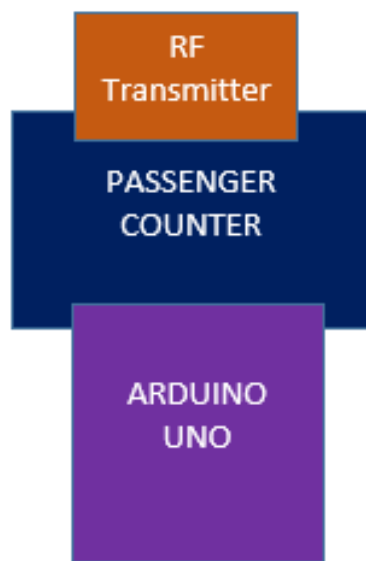


Figure1.5: Passenger counter

2.4 Trigger Unit:

The trigger unit is composed of a water level sensor, motor driver, actuator and a catapult. The water level sensor senses the presence of water and acts accordingly. With the help of Arduino Nano the action set to be triggered, after the presence of water is detected, is being controlled. A motor driver is connected with the Arduino and therefore is used to drive an actuator. The shaft of actuator is placed with a lock in such a way that the lock moves back and forth along with the direction of the shaft of the actuator. The black box placed in a circular shaped holder is thrown as soon as lock goes backward which is when the lock opens.

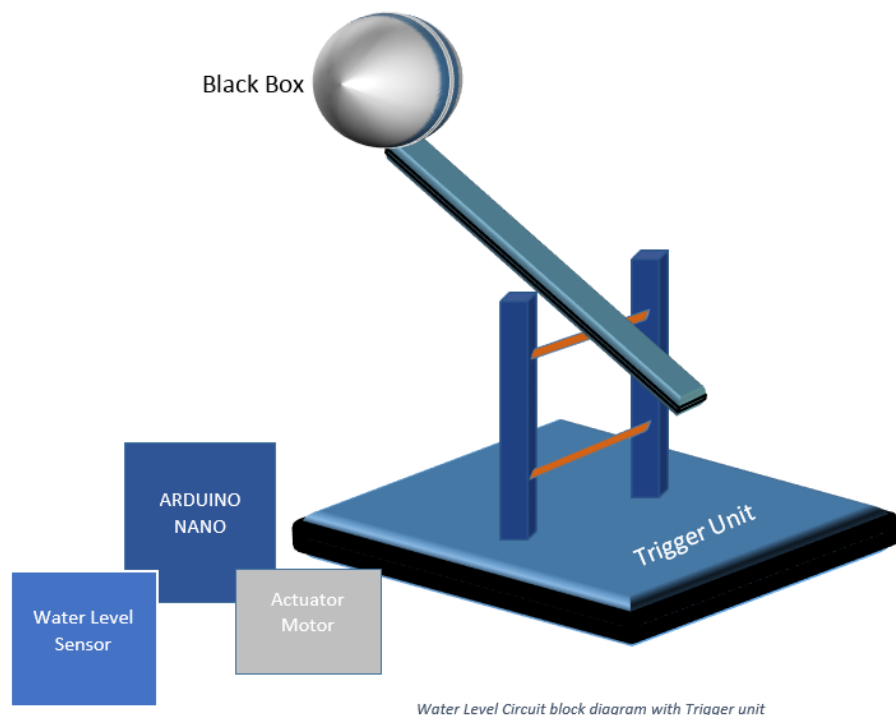


Figure1.6: Trigger Unit

2.5 Web database and Android Application

Sensor values acquired from the on board processor, number of passengers on board and location co-ordinates using GPS are pushed to our raw web database which we have set using the free accessible services provided by SparkFun technologies. We have developed an Android application to fetch and display the data so that it becomes more accessible and can also be used to notify the authorities concerned regarding any critical maritime situation. We have parsed the raw data using array of JSON objects. The application has been incorporated with Google map using their APIs for Android to show the exact location of the black box. For end users convenience, a route has also been shown starting from the current location to the marker placed on location co-ordinates of the black box. The Graphical User Interface (GUI) has been designed to make the application as user friendly as possible. Following figure illustrates a block diagram which explains the entire work flow.

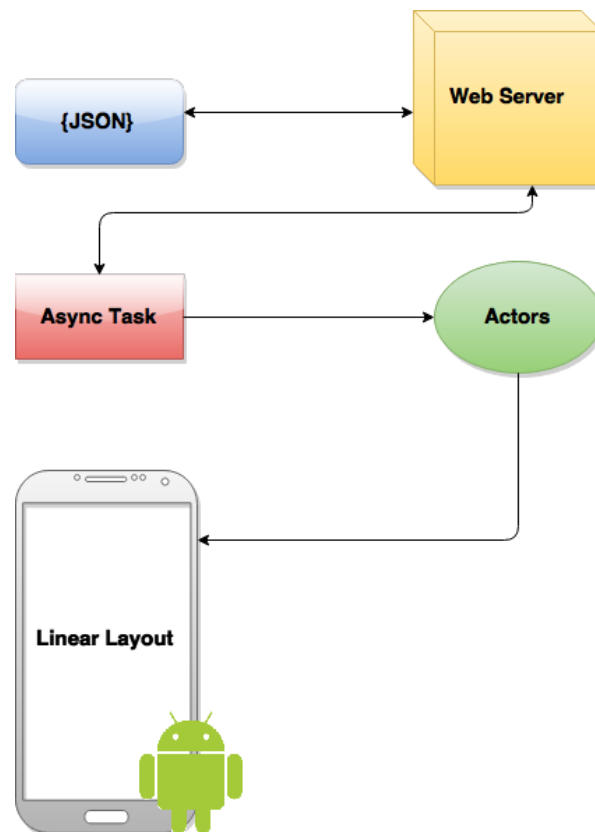


Figure1.7: Web Database and android application

CHAPTER - 3

System Implementation

3.1 Black Box

This is the most critical and the focal unit of our undertaking. With a specific end goal to battle the dynamic and resulting marine mishaps in Bangladesh, we have chosen to utilize a mechanical configuration for this unit that can be demonstrated truly worthwhile for gathering essential data from the marine vessel to screen and investigate certain parts of the mischances.

Remembering certain components alongside our destinations, similar to power utilization and responsiveness, we chose to load this unit with different gadgets that would viably perform. The gadgets utilized for this unit includes: Arduino UNO, GSM GPRS Shield, GPS Shield, Smaller scale SD card Shield, Arduino Mini and RF recipient. We have worked these gadgets in such an example and succession, to the point that it would work best for executing the procedure of getting and sparing fundamental information and all the more imperatively, sending it to web database, which is the key target of this examination. The accompanying areas would clearly portray how every one of the gadgets functions and how that speak with one another to transform the data and complete vital activities to push it web.

3.1.1 Hardware

To shield the delicate circuits from encompassing conditions and harm, we produced a suitable configuration which would be extreme and in the meantime oblige all the vital materials in it. There is additionally a component which we considered amid the arranging of the outline, which the lightness of this defensive intense covering. Since, we need this unit to remain skim on the surface of the water amid mishaps, we are utilizing an empty circle like packaging to encase the circuit, which is made of stainless steel. Moreover we have included battery holder underneath the compartment cover to encourage the arrangement of force supply and keep it out of span from water contact. In addition, there is cuboidal holder inside the circle to keep the discovery circuit set up solidly and give extra security to it. A slender layer of metal can without much of a stretch square GPS, GPRS and RF signs and in this way sticking them from further

correspondence. Since, our external holder is made of intense stainless steel, it hinders the sign from correspondence. To beat this issue, we have made gaps at the position of the circuit situation and brought out important receiving wire and gadgets needed for correspondence. To make these imparting parts water resistant, we have utilized defensive covering which permits them to stay waterproof in the meantime sufficiently conductive to perform vital communications.[15][14]

3.1.2 Control

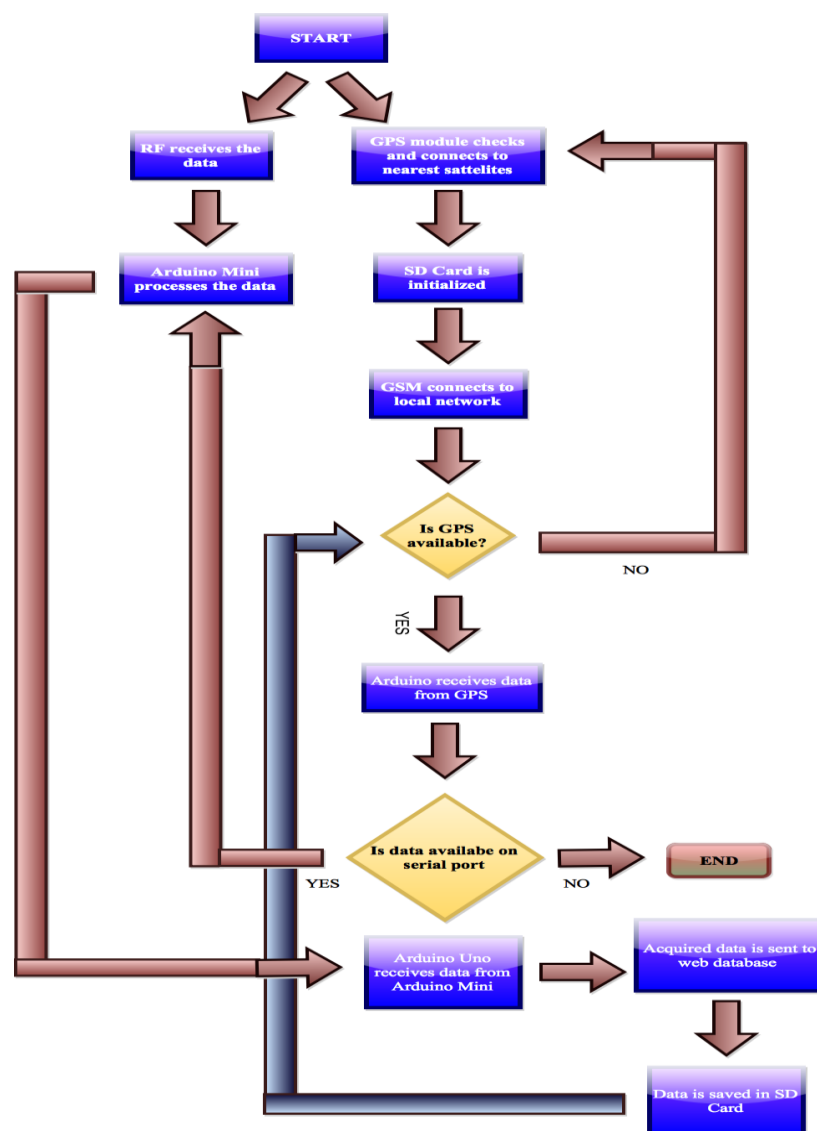


Figure1.8: Control of Black Box unit

3.1.3 Communication

There are mainly four types of communication involved for carrying out the process shown earlier.

- 1) RF to RF communication
- 2) GPS to satellite communication
- 3) Arduino to Arduino communication
- 4) GPRS to web communication

For RF to RF communication, we are using 3 units of RF module as per requirement of this project. Since we need a broad range wireless communication, we are using NRF24L01 as our RF module. NRF24L01 module is a transceiver, that is, it can act as a transmitter or receiver. As the black box unit only requires the operation of data reception, we used this module as the receiver for incoming data. The RF receiver responds to specific addresses, and in this case we have assigned specific address to the transmitter modules. From the transmitter end, the transmitter modules send the data as bytes and from the receiver end, if data is available, it receives these data as bytes and converts it to char array.[9]

For GPS to satellite communication, we are using EM406 GPS module in the GPS shield. The module communicates automatically on power and on connecting with the satellite, it acquires the coordinates of the current location as bytes then it is decoded to char array and finally stored in long type variables.

For Arduino to Arduino communication, we are using Arduino Pro Mini and Arduino Uno. Since, the RF module and SD card shield requires the same Serial Programming Interface (SPI) to be operational, we have connected the RF module to the Arduino Pro Mini and then we are communicating the Mini with the Uno through serial communications between the serial ports, RX and TX. Through this method, we are able to receive the data from the RF receiver without any interference. The data type received from Mini is byte type and is being casted to char type and moreover being appended to a String type variable. [10] [11]

For GPRS to web communication, we are using the SIM900 GSM/GPRS module. On power the module gets automatically activated and checks for the local network for the SIM card it holds. We are using specific AT commands, that it is sensitive to, that is required for serving the purpose of this project. In this case, the AT commands used are:

AT+CSQ: This checks the received signal quality in terms of signal strength

AT+CGATT : This allows to attach and detach from GPRS service.

AT+SAPBR=3,1,\"CONTYPE\", \"GPRS: This is to set bearer settings for applications based on IP

AT+SAPBR=3,1,\"APN\", \"gpineternet: This is used to activate applications like HTTP, FTP through Access Point Name (APN)

AT+HTTPINIT: This initializes the HTTP service that we need to send the data to web database.

AT+HTTPPARA: This sets the HTTP Parameters value, that is, the web address of the database along with the values we need to store in it.

AT+HTTPACTION=1: This is the HTTP method action which can perform action from 0 to 2 as predefined. Here, 1 corresponds to the action “POST”, that is, the given request will get submitted to the assigned web address. [12] [13]

3.1.4 Power

To control the entire unit, we are utilizing a 5,000 mAh 5V Lithium Polymer Force Bank, which are by and large utilized for charging cell telephones. Since, various gadgets are included alongside various correspondence forms, we accepted that it will oblige a considerable measure of energy to work and in this way we felt this is the ideal answer for driving this unit. This can control the entire unit persistently for over 5 hours with full charge when working disconnected from the net. By and large this force bank would be joined with the primary electrical cable of the boat, so it doesn't get depleted amid the voyage. It will just give the offline backup power supply during mishaps.

3.2 On-board Processor:

To facilitate and fulfill the operation of core unit Black box, we additionally required an on-board processor unit, which is further sub-divided into Weather Station unit and Passenger count unit. This On-board Processor unit, as a whole, would feed valuable data such as weather conditions and overall passenger boarding the ship to the Black Box unit, which in turn would process these data and carry out its operations.

3.2.1 Weather Station

The main purpose of this weather station is to acquire various weather data through various weather sensors. According to various research, there is a strong correlation between the change in air pressure and temperature and stormy weather. It has been seen that, for several hours before the storm, the air pressure tends to fall and just before the storm arises, the air pressure tends to rise and there is also a sudden rise of temperature that can be observed. There are also changes in humidity in the air just before the storm. [5][8]

We felt that the changes in weather condition are an important part of the data domain that is very essential to analyze for the marine ships. We have therefore incorporated the three basic weather sensors, temperature, humidity, and barometric pressure sensor in this unit for our whole system. The unit therefore consists of DHT11 temperature and humidity sensor and barometric pressure sensor for obtaining weather data, NRF24L01 RF module to transmit weather data to the main black box unit, Arduino mini to communicate with the Arduino UNO and receive data and transfer it to the RF transmitter module and Arduino UNO to control the whole unit. We are using these sensors because they are very sensitive to weather and therefore even a slight change in weather can be detected and thus taken into account.

3.2.1.1 Control

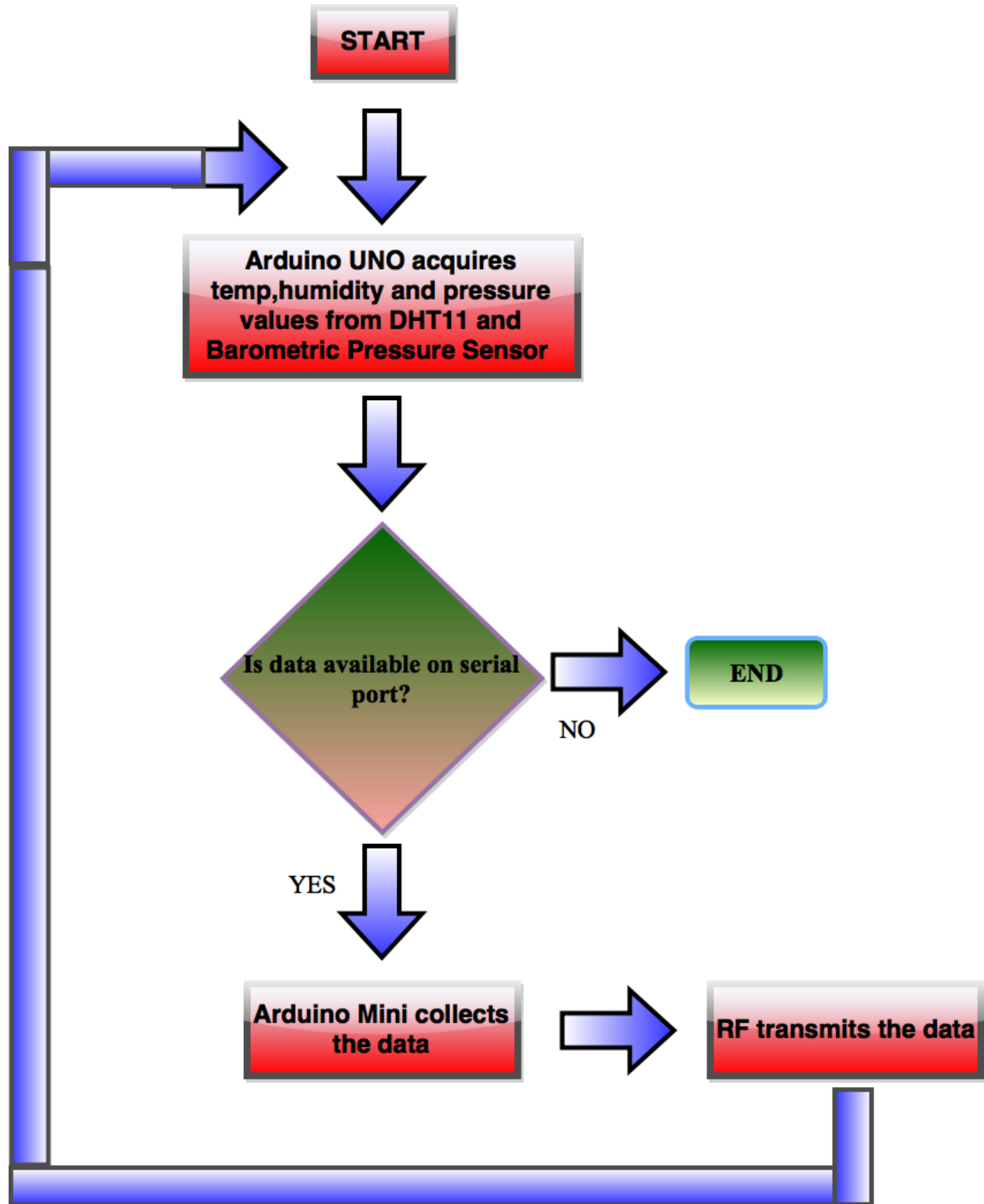


Figure1.11: Control of Weather Station unit

3.2.1.2 Communication

There are only two types of communication in this unit:

- 1) Arduino to Arduino Communication
- 2) RF to RF communication

For this unit, the Arduino to Arduino communication is also being carried out by Arduino Pro Mini and Arduino Uno. Since, the RF module and Barometric Pressure Sensor requires the same Serial Programming Interface (SPI) to be functional, we have connected the RF module to the Arduino Pro Mini and then we are communicating the Arduino Uno with the Arduino mini through serial communications between the serial ports, RX and TX. Through this method, we are able to send the data from the Barometric Pressure Sensor through Arduino Uno to the RF transmitter without any interference. The data type received from Uno is byte type and is being casted to char type and finally being appended to a String type variable.

RF to RF communication is achieved through the RF pairing of two NRF24L01 RF modules. Since in this unit, only wireless transmission of data is required so we are using the NRF24L01 transceiver module as a transmitter. For transmitting the data, first we are representing the RF as a modem to the Arduino mini, and saving its address in a global array. Then we are invoking a function that will receive the address of the receiver that the data will be sent to. By setting the address in this way, we are creating a recognition of the receiver module that the transmitter module will send to, establishing a pairing between both the modules.

3.2.1.3 Power

To power the whole unit, we are using a 5,000 mAh 5V LiPo Power Bank, which are generally used for charging mobile phones. This power bank will further be connected to the power supply of the marine ship, thus providing a seamless supply to the weather station and therefore the station can operate continuously as long as required.

3.2.2 Passenger Counter

Passenger counter is a device which is made to count the number of passengers. It will be placed at the entrance door of the ship. It will automatically count one as soon as the first passenger gets into the ship through the door. It will then keep adding with each passenger entering. Also if a passenger goes out from the door then the device will automatically minus a count. So we are having a perfect count of the number of people traveling in the ship. This device also has another feature. The user can set passenger limit in this device. As soon as the limit exceeds the buzzer turns on. It will only turn off when the person who has exceeded the limit of the number of passengers gets out. Now coming into the things that were used for making this device, two HC-SR04 sonar sensors, Arduino Uno, breadboard and a buzzer were used. The sensors and buzzer is connected with the Arduino. Sonar sensors work as input which gives signal to the Arduino and then the Arduino process it and count the number. The device will only count the passenger if he/she passes through the two sensors. If any one of the sensor is missed by the passenger then it will not count. If the passenger passes through the door crossing sonar sensor 1 and then sonar sensor 2, the passenger count is being incremented. At this point, it is considered that the person is entering into the ship. But if anyone passes from the opposite direction crossing sonar sensor 2 first and then sonar sensor 1, the passenger count gets decremented, indicating a person is getting out of the ship. This is how the whole system works.

3.2.2.1 Control

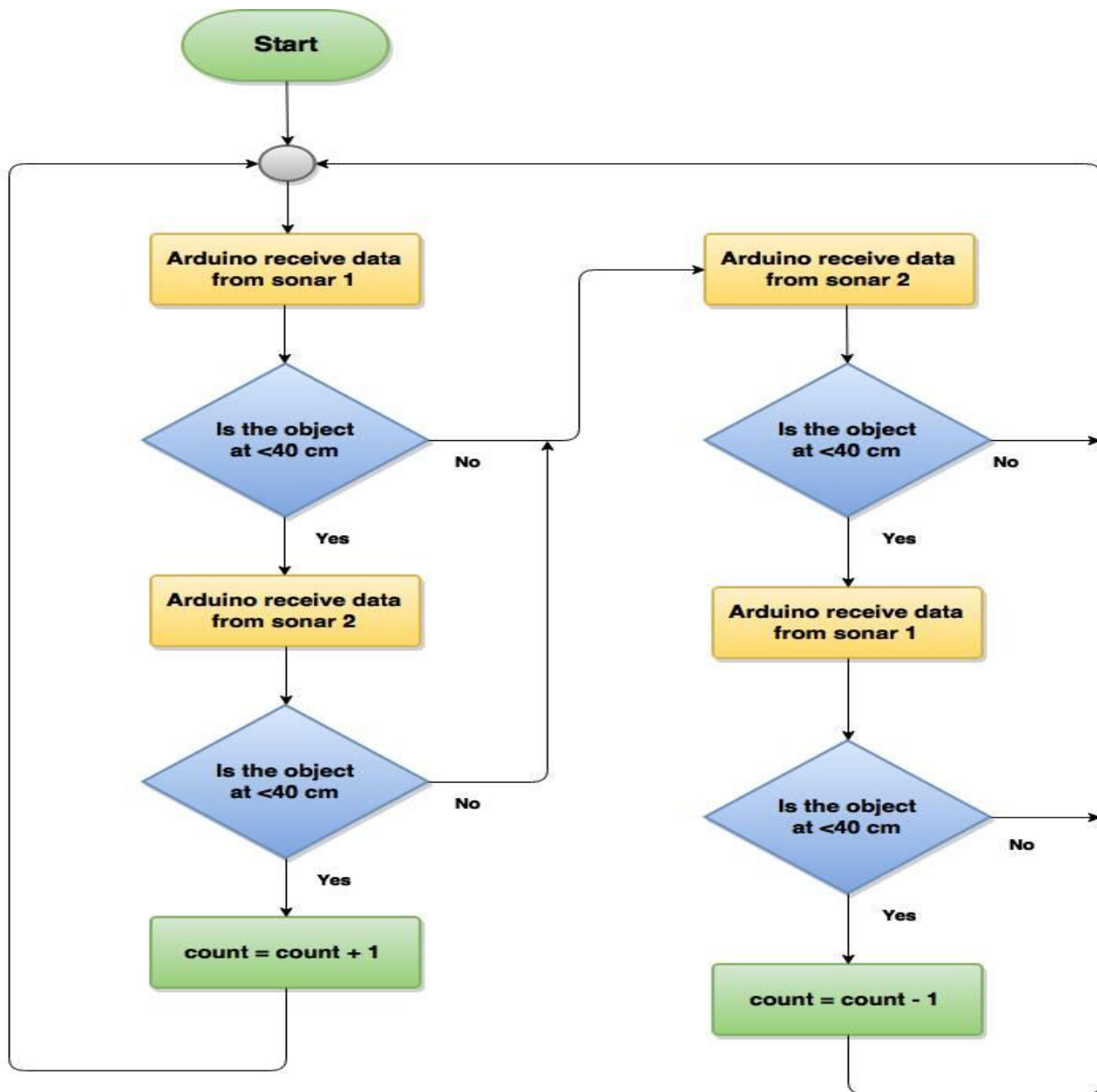


Figure1.13: Control of Passenger Counter unit

3.2.2.2 Communication

There is only one communication that occurs in this unit, which is the RF to RF communication. RF to RF communication is achieved through the RF pairing of two NRF24L01 RF modules. Since in this unit, only wireless transmission of data is required so we are using the NRF24L01 transceiver module as a transmitter. For transmitting the data, first we are representing the RF as a modem to the Arduino UNO, and saving its address in a global array. Then we are invoking a function that will receive the address of the receiver that the data will be sent to. By setting the address in this way, we are creating a recognition of the receiver module that the transmitter module will send to, establishing a pairing between both the modules.

3.2.2.3 Power

To power the whole unit, we are using a 5,000 mAh 5V LiPo Power Bank, which are generally used for charging mobile phones. This power bank will further be connected to the power supply of the marine ship, thus providing a seamless supply to the passenger counter and therefore the station can operate continuously as long as required.

3.3 Trigger unit:

The capacity of trigger unit is to trigger the black box amid the crisis time. This unit is made of a water level sensor and a thrower. Basically the trigger unit is the unit which meets desires simply single time when the water level is in highly danger level. Everything about the activating unit is portrayed in the accompanying part.

3.3.1 Water Level Sensor:

Water level sensor is one of the main parts of our thesis. We used this sensor to detect water levels while occurrence of marine accidents to activate the trigger. Here, we have worked out with three layers of water level (low, average and high) and the intensity of danger. To make water level sensor we have used BJT (Transistor), resistors and Arduino Nano. BC547 is a NPN

bipolar transistor that is used for amplification and switching purpose. It has three terminals collector, base and emitter. Resistors values used here are 9.84k, 10k, 0.43k, 0.48k, 1.3k.

3.3.2 Thrower Cannon:

The configuration of the thrower Cannon standard is similar to trebuchet to toss the black box amid any mishap.

A trebuchet is a barricade motor that was utilized as a part of the medieval times to annihilate workmanship dividers and to toss articles like unhealthy bodies into the château grounds to contaminate the tenants under attack. Because of innovative advances, trebuchets are no more utilized as a part of cutting edge fighting yet some still exist for tossing flame balls. (Lucas)

To make the thrower cannon, the outline of the trebuchet has been taken after. The thrower cannon is made of stainless steel so as to make it sufficiently solid to toss the black box and it is made stainless as it doesn't promptly consume, rust or stain with water which steel does.

The stand of the thrower cannon is 25 inch and this tallness is important as the thrower cannon will be set at the frame of the boat. The tossing arm of the group is made of spring. Four springs have been utilized and the explanation for utilizing these springs arrives is no stabilizer in this thrower cannon. The pressure of the springs are sufficient high and they are joined with a roundabout molded holder at the highest point of the stand. The Black Box is really balanced on the holder. So this spring made tossing arm will toss the Black Box in the water.

3.3.3 Output:

The entire framework is working in an extremely orderly manner. At the point when the lower level which is additionally viewed as the level-3 gets submerged then nothing will happen. At the point when center level that is level-2 get submerged, the ringer will turn on which is utilized to caution the individuals. In conclusion when level-1 goes submerged, Arduino Nano turns on the engine driver. At that point the engine driver which is associated with the actuator runs it. The actuator shaft opens the lock and after that the tossing arm tosses the crate. Again the trebuchet is definitely comprehended for the exactness anyway we obliged the power and force that is observed by the spring to hurl the black box a slight bit far from the Marine vehicle to make

effective usage of the string that is connected with the black box. Moreover just and just by the 120 feet string the affected vehicle can be taken after under the water.

3.3.4 Control:

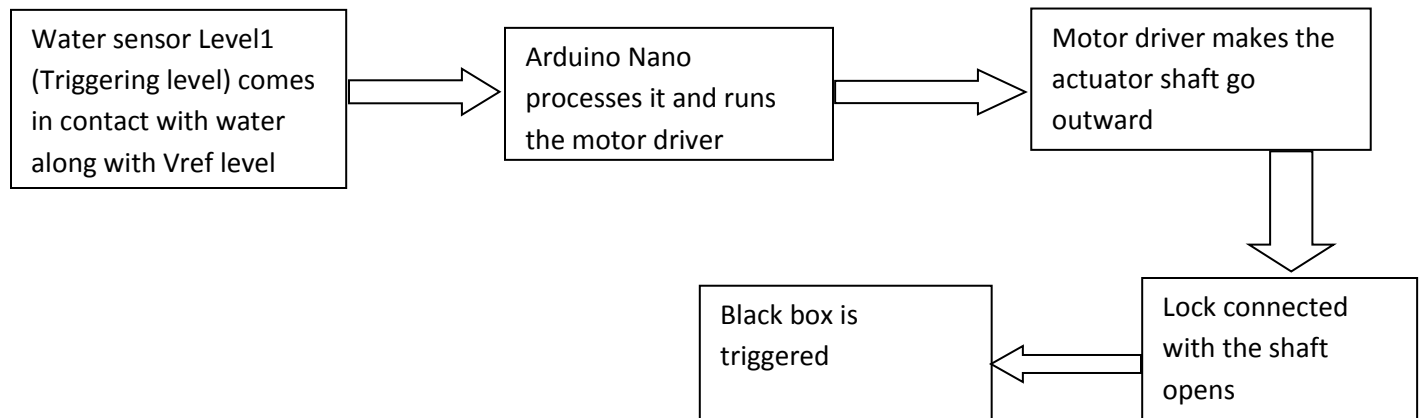


Figure1.15: Control of Trigger Unit

3.3.5 Power:

The motor driver will remain connected with the power supply of the ship. The actuator used here is of 12V. So 12V supply is needed to drive the motor driver.

3.4 Software:

As predicted by many, time has actually arrived when digital systems are not anymore meant by standalone hardware or software rather a blend of both better known as embedded systems. Hardware functions are being controlled by softwares or data received from sensors are being used to make decisions with the help of mobile applications. In our project, the black box using GPS technology gathers its current longitude and latitude and then pushes them along with other sensor values received from on board processor to destined web server. Basically the Graphical User Interface (GUI) part of our project has two parts:

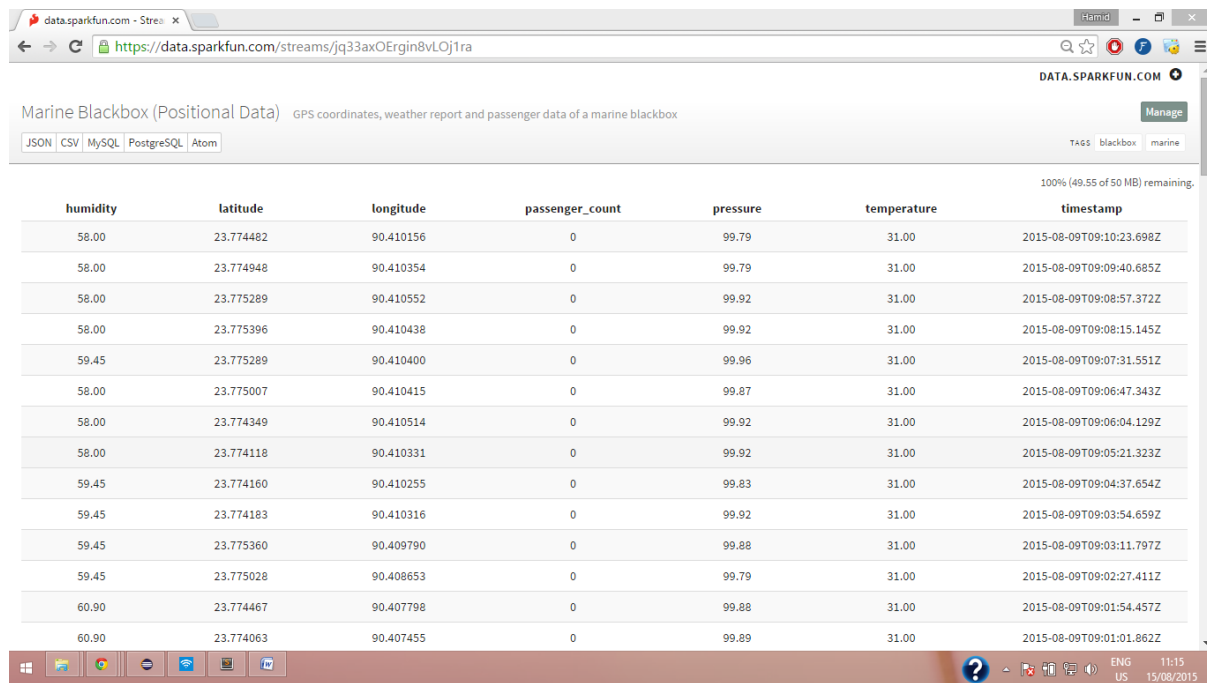
- a. Web Database
- b. Android Application

Since we intended to design a black box, as a typical feature of it, the exact location in terms of longitude and latitude of the water vehicle and data of various other sensors used, had to be stored in a reliable and accessible location. The reason behind we have chosen Android as the mobile platform is the popularity, this being the most used mobile operating system in the world.

3.4.1 Web Database

For our raw data interface we have gone for a readymade web data storage platform offered by SparkFun Electronics. They offer a free, robust service for the experimental projects with a storage limitation of maximum 50MB which we found sufficient for the time being. Data logging to their web interface also comes with a limitation of 100 pushes in every 15 minute window. Since we are log data sent from a water vehicle, we have maintained about 5 minutes delay between two consecutive pushes. While deciding on this time interval we have also been careful and considerate about the worst possible case of any water vehicle i.e. in case of any unwanted situations the black box will continuously logging data to our web data base so if the interval is more than the delay we have kept, we may find difficult to trace the exact location of the black box attached to the vehicle with a string since the force of the river stream will keep displacing it.

Another reason to choose the data storage service from SparkFun is that it comes up with different data formats namely JSON, CSV, MySQL, PostgreSQL and ATOM. The interface that SparkFun public data stream has, looks similar to a table we often see while working with MySQL database with columns containing variable names and rows having the corresponding values.

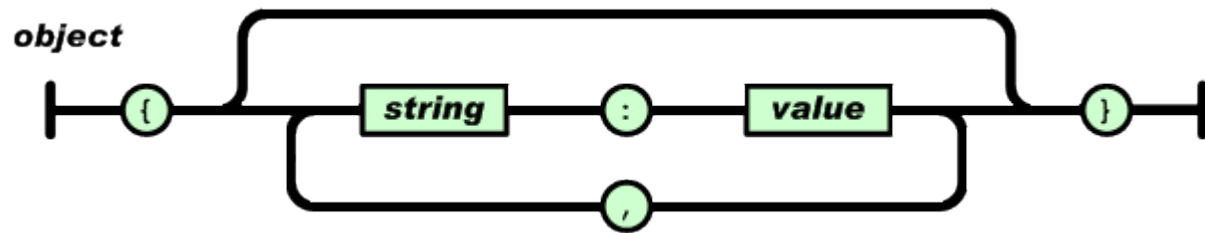


The screenshot shows a web browser window with the URL <https://data.sparkfun.com/streams/fjq33axOEgin8vLOj1ra>. The page title is "Marine Blackbox (Positional Data)" with a subtitle "GPS coordinates, weather report and passenger data of a marine blackbox". There are tabs for "JSON", "CSV", "MySQL", "PostgreSQL", and "Atom". A "Manage" button is visible. The table displays data with columns: humidity, latitude, longitude, passenger_count, pressure, temperature, and timestamp. The data is sorted by timestamp in descending order. A status bar at the bottom right indicates "100% (49.55 of 50 MB) remaining".

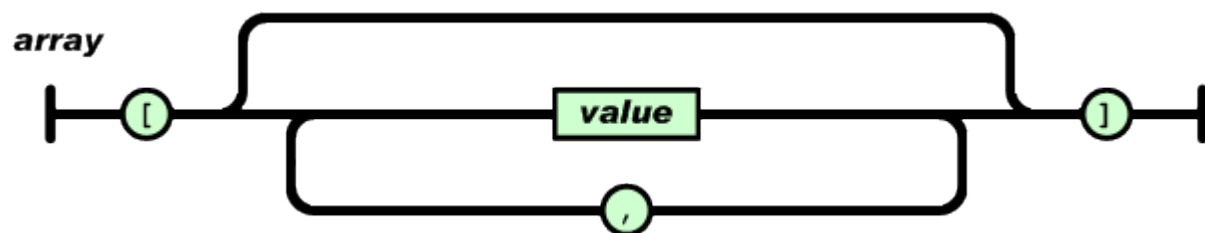
humidity	latitude	longitude	passenger_count	pressure	temperature	timestamp
58.00	23.774482	90.410156	0	99.79	31.00	2015-08-09T09:10:23.698Z
58.00	23.774948	90.410354	0	99.79	31.00	2015-08-09T09:09:40.685Z
58.00	23.775289	90.410552	0	99.92	31.00	2015-08-09T09:08:57.372Z
58.00	23.775396	90.410438	0	99.92	31.00	2015-08-09T09:08:15.145Z
59.45	23.775289	90.410400	0	99.96	31.00	2015-08-09T09:07:31.551Z
58.00	23.775007	90.410415	0	99.87	31.00	2015-08-09T09:06:47.343Z
58.00	23.774349	90.410514	0	99.92	31.00	2015-08-09T09:06:04.129Z
58.00	23.774118	90.410331	0	99.92	31.00	2015-08-09T09:05:21.323Z
59.45	23.774160	90.410255	0	99.83	31.00	2015-08-09T09:04:37.654Z
59.45	23.774183	90.410316	0	99.92	31.00	2015-08-09T09:03:54.659Z
59.45	23.775360	90.409790	0	99.88	31.00	2015-08-09T09:03:11.797Z
59.45	23.775028	90.408653	0	99.79	31.00	2015-08-09T09:02:27.411Z
60.90	23.774467	90.407798	0	99.88	31.00	2015-08-09T09:01:54.457Z
60.90	23.774063	90.407455	0	99.89	31.00	2015-08-09T09:01:01.862Z

JSON: JavaScript Object Notation better known shortly as JSON is commonly used lightweight data-interchange format. This format is easy for humans to understand and also for machines to parse and generate. Based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999, it is a text format that is completely language independent but uses conventions that are familiar to programmers of different programming languages which makes it ideal as a format to interchange data. In JSON most common forms are:

1. Object: Unordered set consisting name/value pairs beginning with a "{" and ending with "}". Each name is followed by ":" and the name/value pairs get separated by ",".



2. Array: Ordered collection of values unlike Object, beginning with "[" and ending with "]" and values are separated by ",".



Our raw data interface gives the following output if extracted as JSON format where each row of the table is turned into an object having multiple name/value pairs separated by "," and whole collection of objects are fit into an ordered array.



```
[{"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.79", "longitude": "90.410156", "latitude": "23.774482", "timestamp": "2015-08-09T09:10:23.698Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.79", "longitude": "90.410354", "latitude": "23.774948", "timestamp": "2015-08-09T09:09:40.685Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.92", "longitude": "90.410552", "latitude": "23.775289", "timestamp": "2015-08-09T09:08:57.372Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.92", "longitude": "90.410438", "latitude": "23.775396", "timestamp": "2015-08-09T09:08:15.145Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "59.45", "pressure": "99.96", "longitude": "90.410400", "latitude": "23.775289", "timestamp": "2015-08-09T09:07:31.551Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.87", "longitude": "90.410415", "latitude": "23.775007", "timestamp": "2015-08-09T09:06:47.343Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.92", "longitude": "90.410514", "latitude": "23.774349", "timestamp": "2015-08-09T09:06:04.129Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "58.00", "pressure": "99.92", "longitude": "90.410331", "latitude": "23.774118", "timestamp": "2015-08-09T09:05:21.323Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "59.45", "pressure": "99.83", "longitude": "90.410255", "latitude": "23.774160", "timestamp": "2015-08-09T09:04:37.654Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "59.45", "pressure": "99.92", "longitude": "90.410316", "latitude": "23.774183", "timestamp": "2015-08-09T09:03:54.659Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "59.45", "pressure": "99.88", "longitude": "90.409790", "latitude": "23.775360", "timestamp": "2015-08-09T09:03:11.797Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "59.45", "pressure": "99.79", "longitude": "90.408653", "latitude": "23.775028", "timestamp": "2015-08-09T09:02:27.411Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "60.90", "pressure": "99.88", "longitude": "90.407798", "latitude": "23.774467", "timestamp": "2015-08-09T09:01:54.457Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "60.90", "pressure": "99.89", "longitude": "90.407455", "latitude": "23.774063", "timestamp": "2015-08-09T09:01:01.862Z"}, {"passenger_count": "0", "temperature": "31.00", "humidity": "60.90", "pressure": "99.80", "longitude": "90.407585", "latitude": "23.774063", "timestamp": "2015-08-09T09:01:01.862Z"}]
```

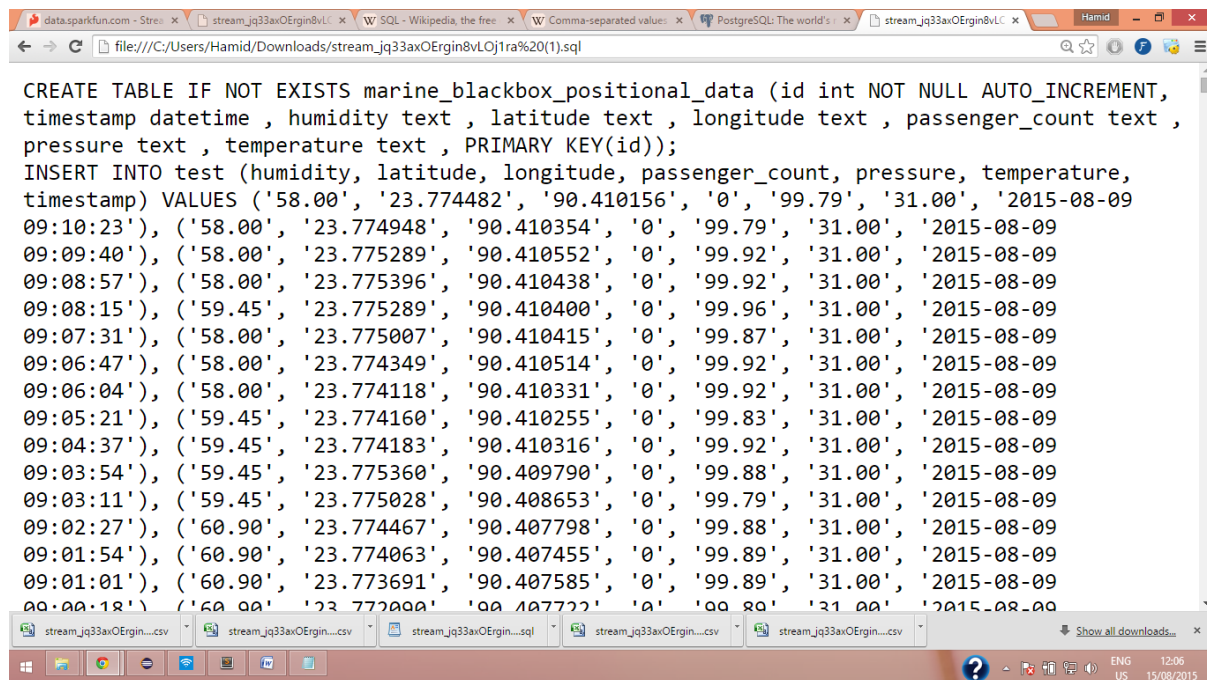
CSV: Comma-separated Values is also one of the widely used format for data exchange which stores tabular data in plain text. Each line of the format denotes a data record consisting of different fields and "," separates the fields. The format is human readable and often imported to spreadsheet programs when extracted.

For instance, our web database show the following.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	passenger_count	temperature	humidity	pressure	longitude	latitude	timestamp								
2	0	31	58	99.79	90.4102	23.7745	2015-08-09T09:10:23.698Z								
3	0	31	58	99.79	90.4104	23.7749	2015-08-09T09:09:40.685Z								
4	0	31	58	99.92	90.4106	23.7753	2015-08-09T09:08:57.372Z								
5	0	31	58	99.92	90.4104	23.7754	2015-08-09T09:08:15.145Z								
6	0	31	59.45	99.96	90.4104	23.7753	2015-08-09T09:07:31.551Z								
7	0	31	58	99.87	90.4104	23.775	2015-08-09T09:06:47.343Z								
8	0	31	58	99.92	90.4105	23.7743	2015-08-09T09:06:04.129Z								
9	0	31	58	99.92	90.4103	23.7741	2015-08-09T09:05:21.323Z								
10	0	31	59.45	99.83	90.4103	23.7742	2015-08-09T09:04:37.654Z								
11	0	31	59.45	99.92	90.4103	23.7742	2015-08-09T09:03:54.659Z								
12	0	31	59.45	99.88	90.4098	23.7754	2015-08-09T09:03:11.797Z								
13	0	31	59.45	99.79	90.4087	23.775	2015-08-09T09:02:27.411Z								
14	0	31	60.9	99.88	90.4078	23.7745	2015-08-09T09:01:54.457Z								
15	0	31	60.9	99.89	90.4075	23.7741	2015-08-09T09:01:01.862Z								
16	0	31	60.9	99.89	90.4076	23.7737	2015-08-09T09:00:18.578Z								
17	0	31	60.9	99.89	90.4077	23.7721	2015-08-09T08:59:34.532Z								
18	0	31	62.35	99.85	90.4079	23.7707	2015-08-09T08:58:51.035Z								
19	0	31	62.35	99.85	90.4077	23.7698	2015-08-09T08:58:09.548Z								

MySQL : Another useful format in which our raw data can be extracted is the Structured Query Language(SQL) which is a special-purpose programming language designed to deal with data stored in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). After extracting queries can be run on them to manipulate the table data accordingly.

Again, our raw data takes shape in the following way when extracted as SQL format.

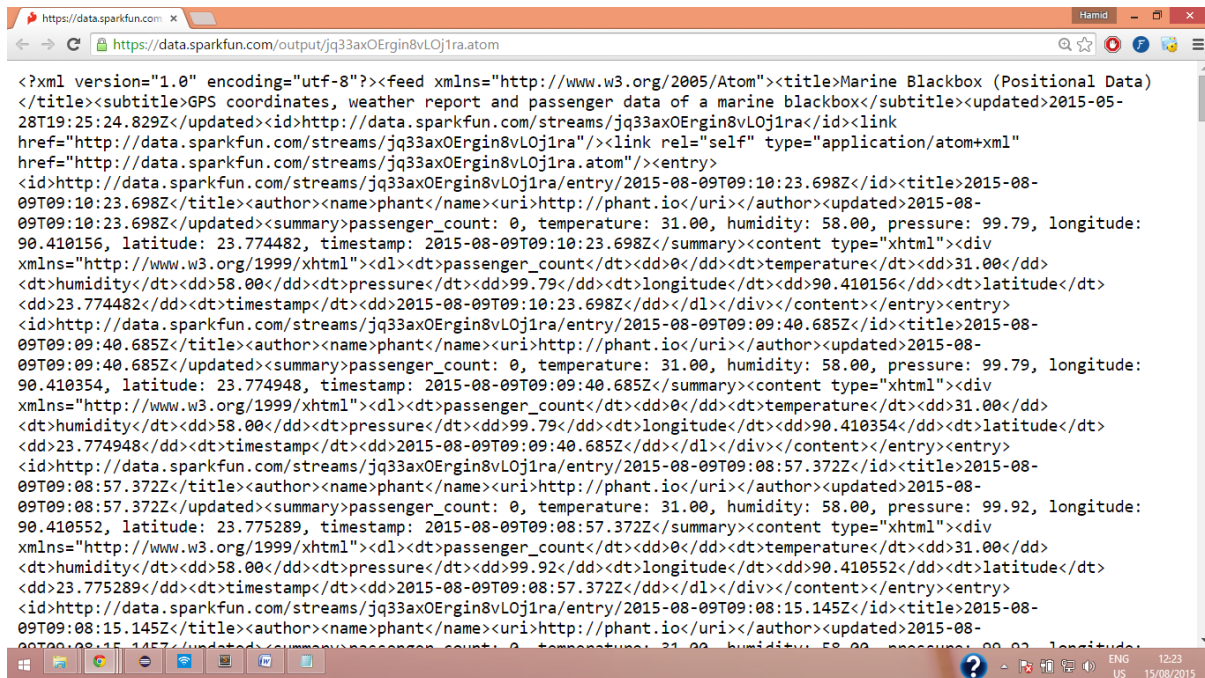
A screenshot of a Windows file explorer window. The address bar shows the file path: file:///C:/Users/Hamid/Downloads/stream_jq33axOErin8vLQj1tra%20(1).sql. The main pane displays the contents of the .sql file, which is a PostgreSQL script. The script starts with a CREATE TABLE statement for 'marine_blackbox_positional_data' with columns: id (int NOT NULL AUTO_INCREMENT), timestamp (datetime), humidity (text), latitude (text), longitude (text), passenger_count (text), pressure (text), and temperature (text). The PRIMARY KEY is set to (id). This is followed by an INSERT INTO statement for a table named 'test' with columns: humidity, latitude, longitude, passenger_count, pressure, temperature, and timestamp. The INSERT statement contains 20 rows of data, each representing a timestamp and corresponding sensor readings. The data is truncated at the bottom of the visible window. The taskbar at the bottom shows several open applications, including 'stream_jq33axOErin8vLQj1tra%20(1).sql', and the system clock shows 12:06 on 15/08/2015.

```
CREATE TABLE IF NOT EXISTS marine_blackbox_positional_data (id int NOT NULL AUTO_INCREMENT,
timestamp datetime , humidity text , latitude text , longitude text , passenger_count text ,
pressure text , temperature text , PRIMARY KEY(id));
INSERT INTO test (humidity, latitude, longitude, passenger_count, pressure, temperature,
timestamp) VALUES ('58.00', '23.774482', '90.410156', '0', '99.79', '31.00', '2015-08-09
09:10:23'), ('58.00', '23.774948', '90.410354', '0', '99.79', '31.00', '2015-08-09
09:09:40'), ('58.00', '23.775289', '90.410552', '0', '99.92', '31.00', '2015-08-09
09:08:57'), ('58.00', '23.775396', '90.410438', '0', '99.92', '31.00', '2015-08-09
09:08:15'), ('59.45', '23.775289', '90.410400', '0', '99.96', '31.00', '2015-08-09
09:07:31'), ('58.00', '23.775007', '90.410415', '0', '99.87', '31.00', '2015-08-09
09:06:47'), ('58.00', '23.774349', '90.410514', '0', '99.92', '31.00', '2015-08-09
09:06:04'), ('58.00', '23.774118', '90.410331', '0', '99.92', '31.00', '2015-08-09
09:05:21'), ('59.45', '23.774160', '90.410255', '0', '99.83', '31.00', '2015-08-09
09:04:37'), ('59.45', '23.774183', '90.410316', '0', '99.92', '31.00', '2015-08-09
09:03:54'), ('59.45', '23.775360', '90.409790', '0', '99.88', '31.00', '2015-08-09
09:03:11'), ('59.45', '23.775028', '90.408653', '0', '99.79', '31.00', '2015-08-09
09:02:27'), ('60.90', '23.774467', '90.407798', '0', '99.88', '31.00', '2015-08-09
09:01:54'), ('60.90', '23.774063', '90.407455', '0', '99.89', '31.00', '2015-08-09
09:01:01'), ('60.90', '23.773691', '90.407585', '0', '99.89', '31.00', '2015-08-09
09:00:18'), ('60.90', '23.772000', '90.407722', '0', '99.80', '31.00', '2015-08-09
```

PostgreSQL : An open source object-relational database system which runs stored procedures in more than a dozen programming languages, including Java, Perl, Python, Ruby, C/C++, and its own PL/pgSQL, which is similar to Oracle's PL/SQL. Hundreds of built-in functions that range from basic math and string operations to cryptography included with its standard function library makes it even more powerful as a format of data interchanging.

ATOM : Another format in which our raw data stored in web, can be extracted is ATOM which is an XML based format which defines XML elements and their meaning (feeds, entries, links), and OData protocol uses that along with its own XML elements to serialize the data.

To illustrate, our web database when extracted in ATOM format is shown below.



3.4.2 Android Application

Mainly implemented to make all the information, that are stored in our raw web database, more accessible and functional with some added features for the end users. Our primary target was to echo the entire system designed already for the web server and put together all the variable values so that they can be more usable for instance the number of people on board is displayed or the location co-ordinates fetched are being used to show route from our current location to the black box. The entire graphical user interface has been developed in such a way that it remains friendly to the users along with proper functionalities. We have used the followings to construct the application.

a. Eclipse IDE for Java Developers

Version: Luna Service Release 1a (4.4.1)

b. Android Software Development Kit (SDK)

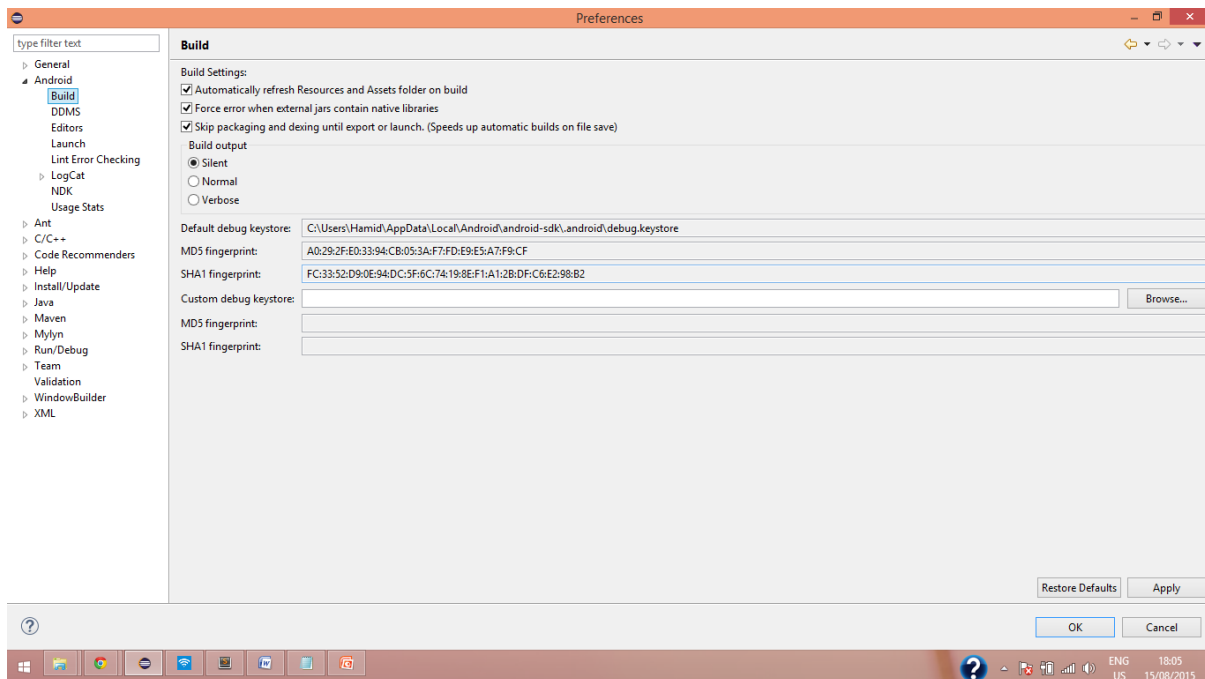
c. Android Debugging Tool (ADT)

Since one of the major part of our Android application is to show the route from our current location to the black box on Google map, we needed to incorporate things that are necessary to develop a Google map based Android application. Initially Google Maps Android API V1 was released which has now deprecated and now been replaced with API V2 with added features like search in plain English, Search by voice, Traffic view, Search along route, Satellite view and Street view.

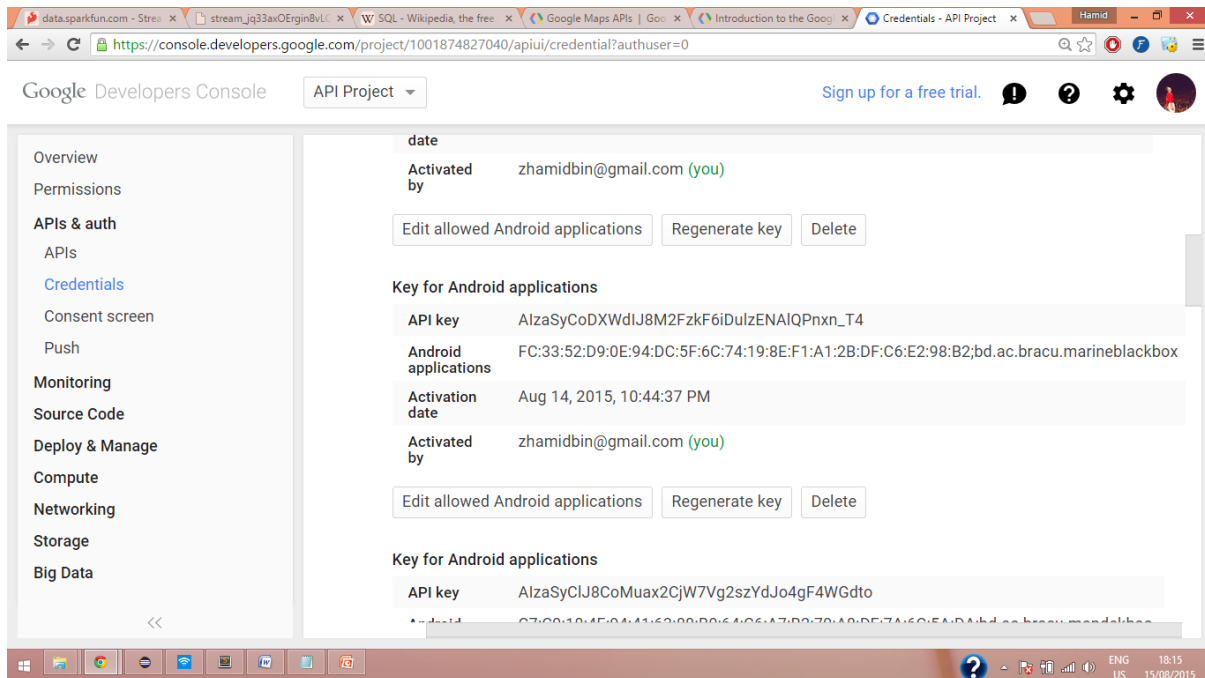
Google Maps Android API allows to add maps based on Google Maps data to the application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. API calls are used to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area. These items give extra data to map locations, and permit client connection with the map.

Following things are required to incorporate Google Maps Android API V2 are:

1. SHA1 Fingerprint: SHA-1 fingerprint is a unique text string generated by SHA-1 hash Algorithm and as it is unique, Google Maps uses it to identify the application. To obtain this fingerprint, we need to go through the following path i.e. in Eclipse IDE, Windows -> Preferences -> Android -> Build.

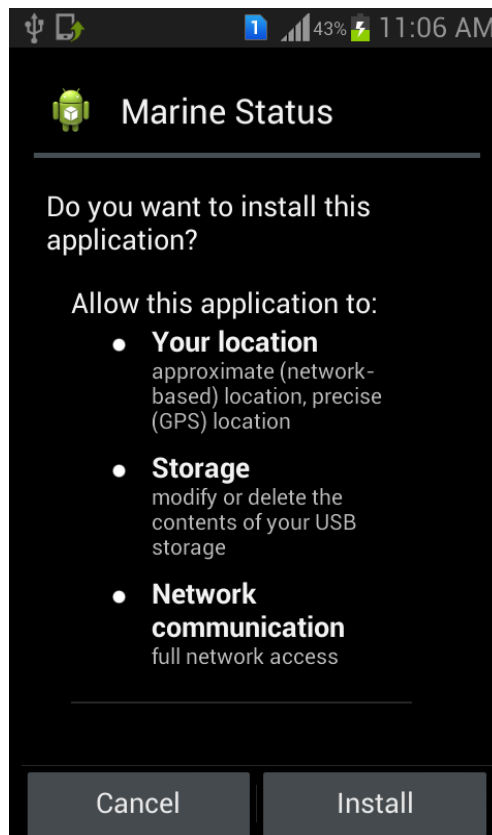


2. Map V2 API key : An application programming interface key (API key) is a code passed in by computer programs calling an API (application programming interface) to identify the calling program, its developer, or its user to the website. After enabling Android Map V2 service, API access was made to generate the API key for our application providing them with SHA1 fingerprint and the package name of the application. For instance, package name of our application is "bd.ac.bracu.marineblackbox;".



3. Google Play Services Library: Google Play Services make the access to Google Services easy and has strong integration with Android OS. Easy-to-use client libraries are provided for each service that let us implement the functionality we want easier and faster.

After incorporating things that are necessary to ensure that the Android application has access to Google Maps Services, we need to add necessary permissions to the Android Manifest file of the application.



```

1  <uses-permission android:name="com.androidbunch.drawroutev2.permission.MAPS_RECEIVE" />
2
3  <uses-permission android:name="android.permission.INTERNET" />
4  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
5  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
6  <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
7  <!--
8  The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
9  Google Maps Android API v2, but are recommended.
10 -->
11 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
12 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
13
14
15
16 <!-- Internet permission -->
17 <uses-permission android:name="android.permission.INTERNET" />

```

Following are the functions of some of these permissions being added to our application:

ACCESS_NETWORK_STATE: to check network state whether data can be downloaded or not

WRITE_EXTERNAL_STORAGE: to write to external storage as Google Maps store map data

ACCESS_COARSE_LOCATION: to determine user location using Wi-Fi and mobile cellular data

ACCESS_FINE_LOCATION: to determine user location using GPS

```
1 <uses-sdk
2   android:minSdkVersion="14"
3   android:targetSdkVersion="21" />
```

Minimum SDK version for our application has been set to Google API level 14 (platform version Android 4.0, 4.0.1, 4.0.2) which suggests this is the minimum API level required for the application to run. The Android system will not allow user to install this application if the system's API level is lower than the defined minimum SDK version.

Target SDK version for an application is meant by the SDK version it targets which suggests the system that the application have been tested against the target version and no compatibility issues should arise. The application will able to run on older versions down to the minimum SDK versions. In our case, we have set this to Google API level 21 (platform version Android 5.0)

We have compiled our application with Google API level 17 (platform version Android 4.2, 4.2.2).

In order to fetch data stored in raw web database to our Android application we have opted for JSON to interchange data across the platforms. In the following figures show how JSON array has been initialized and JSON Object has been created. For the application to response faster we have not iterated through the whole web database, rather we have parsed the latest logged data to know about the sensor values and last known location of the blackbox.

```

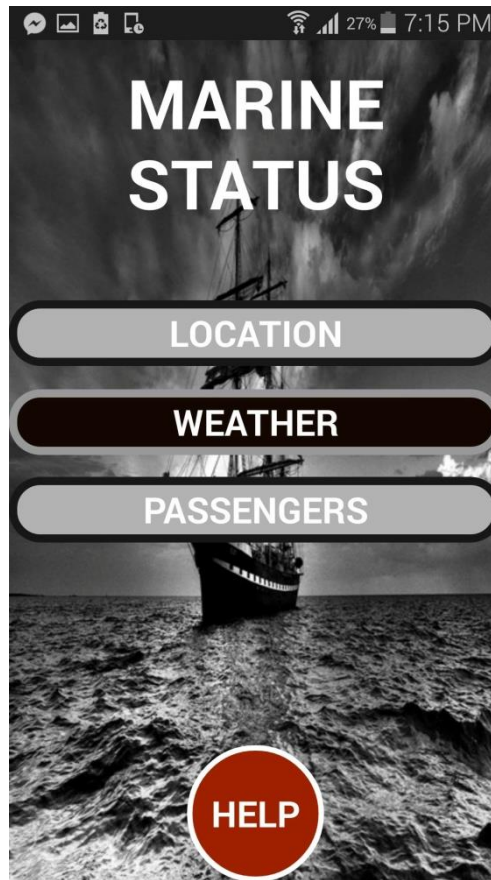
26         count = c.getString(TAG_COUNT);
27         temp = c.getString(TAG_TEMP);
28         hum = c.getString(TAG_HUM);
29         pre = c.getString(TAG_PRE);
30         longi = c.getString(TAG_LONG);
31         lat = c.getString(TAG_LAT);
32         time = c.getString(TAG_TIME);
33     } catch (JSONException e) {
34         e.printStackTrace();
35     }
36 } else {
37     Log.e("ServiceHandler", "Couldn't get any data from the url");
38 }
39
40 return null;
41
42 }
43
44 //initializing JSONArray
45 JSONArray points = null;
46 .
47 .
48 .
49 .
50 .
51 .
52 .
53 .
54
55 protected Void doInBackground(Void... arg0) {
56     // Creating service handler class instance
57     ServiceHandler sh = new ServiceHandler();
58     // Making a request to url and getting response
59     String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);
60     Log.d("Response: ", "> " + jsonStr);
61     if (jsonStr != null) {
62         try {
63             //JSONObject jsonObj = new JSONObject(jsonStr);
64             JSONArray json = new JSONArray(jsonStr);
65
66             // Getting JSON Array node
67             JSONObject c= json.getJSONObject(0);

```

Another class named ServiceHandler.java has been implemented to make the url and http requests. Inside this class, http client has also been initialized and steps taken according to the http request method types (GET and POST).

The main layout or the home of our application consists three main buttons namely "Location", "Weather" and "Passengers". To make the graphical user experience more friendly another button named "Help" has also been kept there to instruct the user about how the application works.

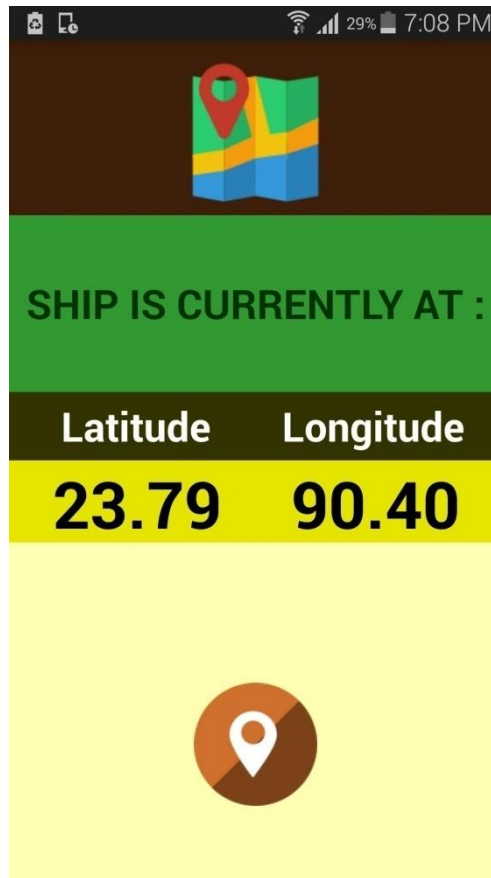
Following figure shows the main layout of the application.



When "Location" button is pressed a new activity named "LocationActivity" is called showing the last known latitude and longitude of the black box. To implement one of our main goal that is to show the exact location of the black box from our current location obtained by inbuilt feature of Google Map Services "Location".

```
1 Location location;  
2 .  
3 .  
4 .  
5 .  
6 .  
7 .  
8 .  
9 double currentLatitude = location.getLatitude();  
10 double currentLongitude = location.getLongitude();
```

Following figure shows the Location Activity layout.



From the MainActivity, values of latitude and longitude fields get sent through with the help of an Intent and those field values are shown in the designated fields of the LocationActivity.

```

1 loc = (Button) findViewById(R.id.button1);
2 .
3 .
4 .
5 loc.setOnClickListener(new OnClickListener(){
6
7     public void onClick(View v){
8         Intent intLoc = new Intent(MainActivity.this, LocationActivity.class);
9         intLoc.putExtra(TAG_LAT, lat);
10        intLoc.putExtra(TAG_LONG, longi);
11        startActivity(intLoc);
12    }
13 });

```

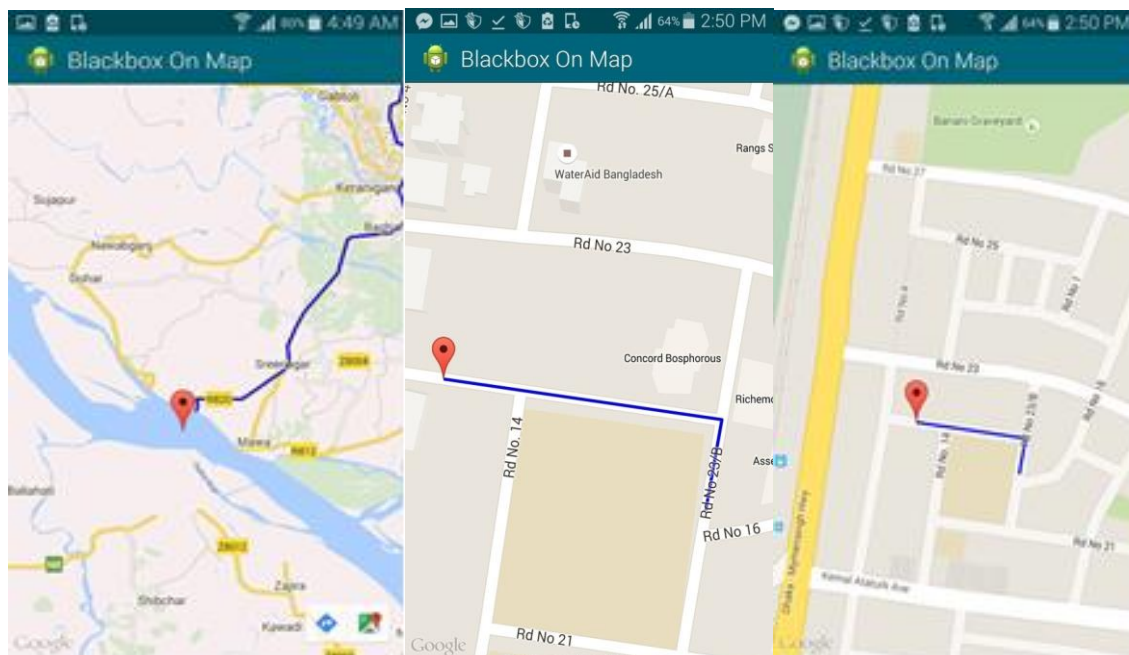
LocationActivity has a button which if pressed will load the Google map in our application and will keep our current location at the centre of the screen. A marker will be placed to the exact location of the black box and a route (in the Driving mode) will be shown from our current location. If our current location changes the application will reset the view again bringing the

new current location of the user to the centre of the screen, also updating the route to the black box every time a new location is handled.

```
1 protected String doInBackground(String... urls) {  
2     //Get All Route values  
3     document = v2GetRouteDirection.getDocument(fromPosition, toPosition,  
4                                             GMapV2GetRouteDirection.MODE_DRIVING);  
5     response = "Success";  
6     return response;  
7 }  
8 }
```

GMapV2GetRouteDirection.java class has been implemented to make the HttpClient, HttpContext, HttpPost, HttpResponse instantiation and add all geopoints to an arraylist so that it can be put together to be displayed as a complete route on the Google map.

MapActivity Class extends "FragmentActivity" and implements "LocationListener", "GoogleApiClient.ConnectionCallbacks", "GoogleApiClient.OnConnectionFailedListener".



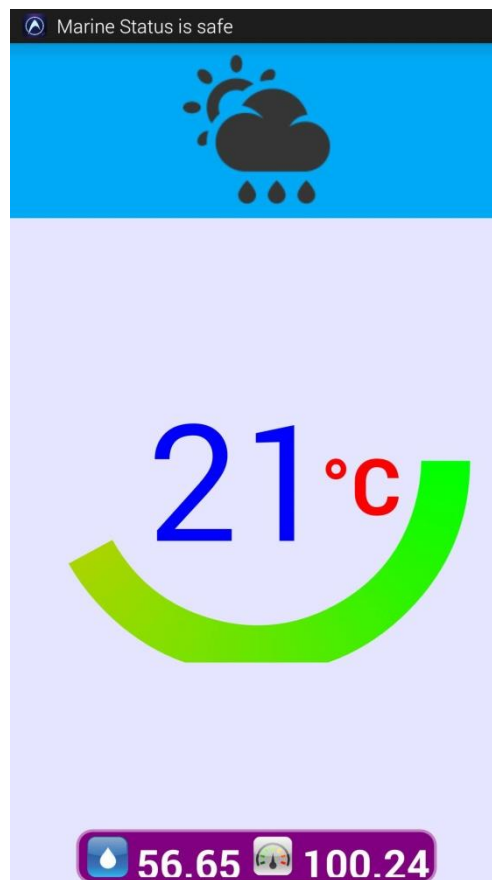
The figure above shows the Map Layout of our application showing a route to black box location where a marker has been placed from our current location.

The "Weather" button that has been placed in the main layout of our application opens a new activity named "WeatherActivity" where related sensor values obtained from the on boards

processor of our project such as temperature, humidity and pressure are passed through an intent so that they can be displayed. On a scale of 50 degree Celsius, there has also been implemented a progress bar to show how critical the temperature value is at that particular point of time.

```
1 wea = (Button) findViewById(R.id.button2);
2
3 wea.setOnClickListener(new OnClickListener(){
4
5     public void onClick(View v){
6         Intent intWeather = new Intent(MainActivity.this, WeatherActivity.class);
7         intWeather.putExtra(TAG_TEMP, temp);
8         intWeather.putExtra(TAG_HUM, hum);
9         intWeather.putExtra(TAG_PRE, pre);
10        startActivity(intWeather);
11    }
12 });
13
14
```

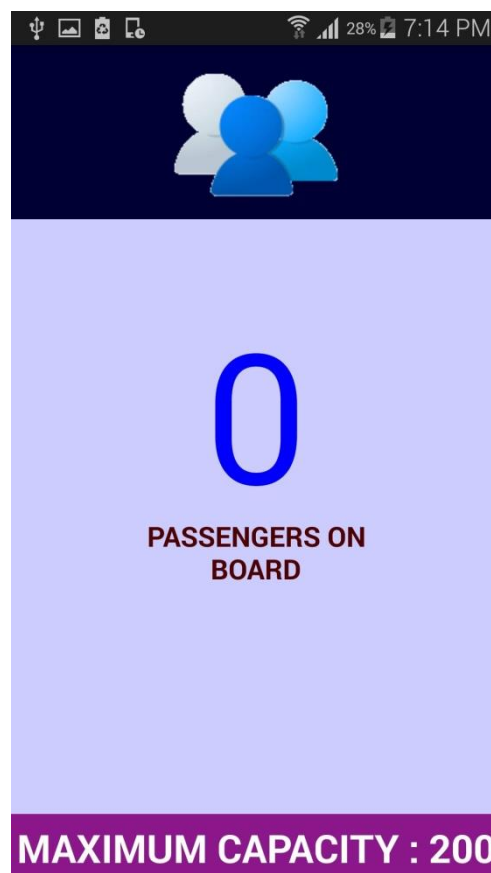
The figure above shows how an intent is carrying three sensor values to the WeatherActivity from MainActivity and the following figure shows the layout of the WeatherActivity.



The "Passengers" button that has been placed in the main layout of our application opens a new activity named "Passenger Activity" where number of people on board, computed by the passenger counter and then pushed to our raw data interface, is shown to regulate the excessive passenger carrying and to mitigate the risk of any maritime accidents. On a scale of 200 people (considered for our experimental project), there has been designed a progress bar to show the risk factor concerned depending on the ratio of people on board and the capacity of the water vehicle.

```
1 pas = (Button) findViewById(R.id.button3);
2
3
4 pas.setOnClickListener(new OnClickListener(){
5
6     public void onClick(View v){
7         Intent intPassenger = new Intent(MainActivity.this, PassengerActivity.class);
8         intPassenger.putExtra(TAG_COUNT, count);
9         startActivity(intPassenger);
10    }
11 });
12
```

The figure above shows how an intent is carrying three sensor values to the PassengerActivity from MainActivity and the following figure shows the layout of the PassengerActivity.



Chapter - 4

Data Analysis

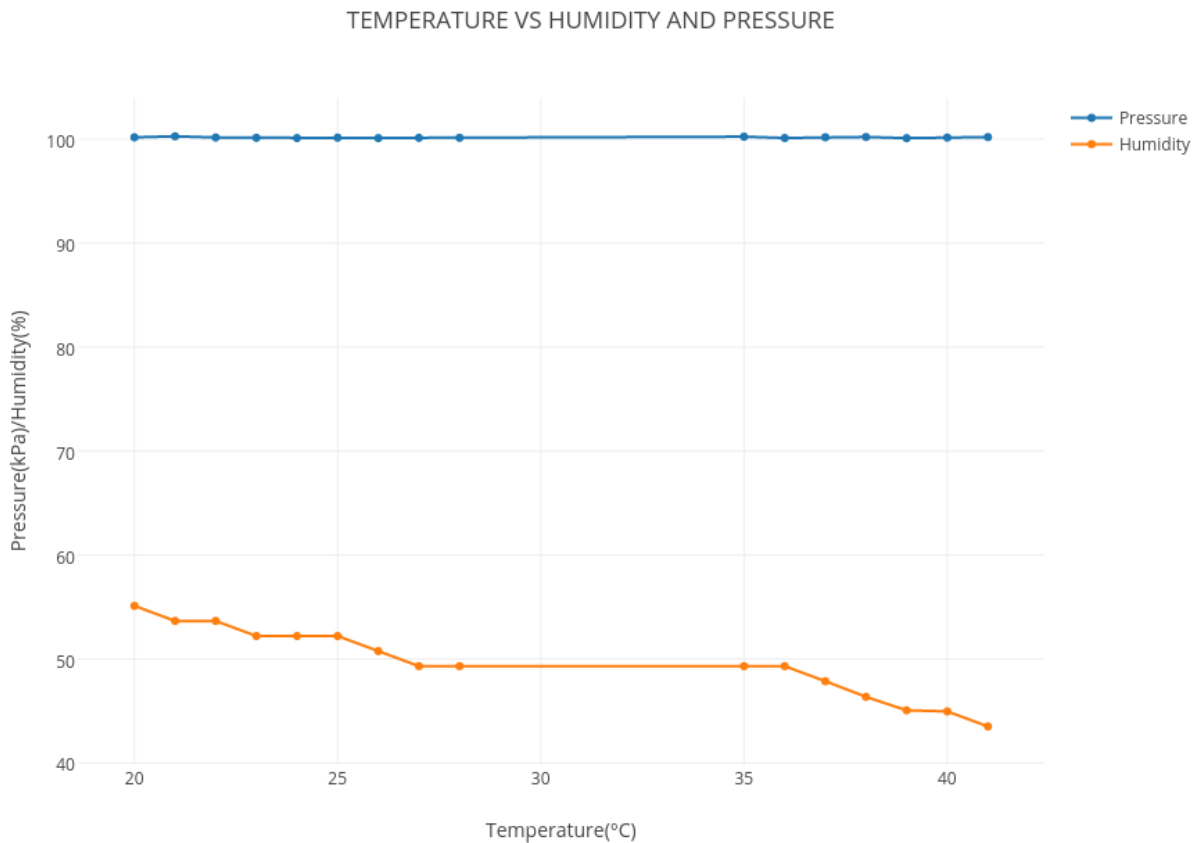
4.1 On-Board Processor Analysis

4.1.1 Weather Data Analysis

We have done several experiments with sensors in both constant temperature and also in various temperatures and took the corresponding readings of the sensor values. We tried to figure out a relationship between the temperature and pressure and also between temperature and humidity. For this experiment, we have taken sensor readings of 15 different temperatures and for each temperature we have taken 25 sets of data for humidity and pressure and then we took the average of the pressure and humidity for the corresponding temperature.

Temperature	Average Pressure(kPa)	Average Humidity (%)
21	100.282	53.65
22	100.163	53.65
23	100.137	52.20
24	100.113	52.20
25	100.150	50.20
26	100.109	50.75
27	100.135	49.30
28	100.137	49.30
35	100.244	49.30
36	100.116	49.30
37	100.187	47.85

38	100.212	46.35
39	100.101	45.05
40	100.156	44.95
41	100.206	43.50



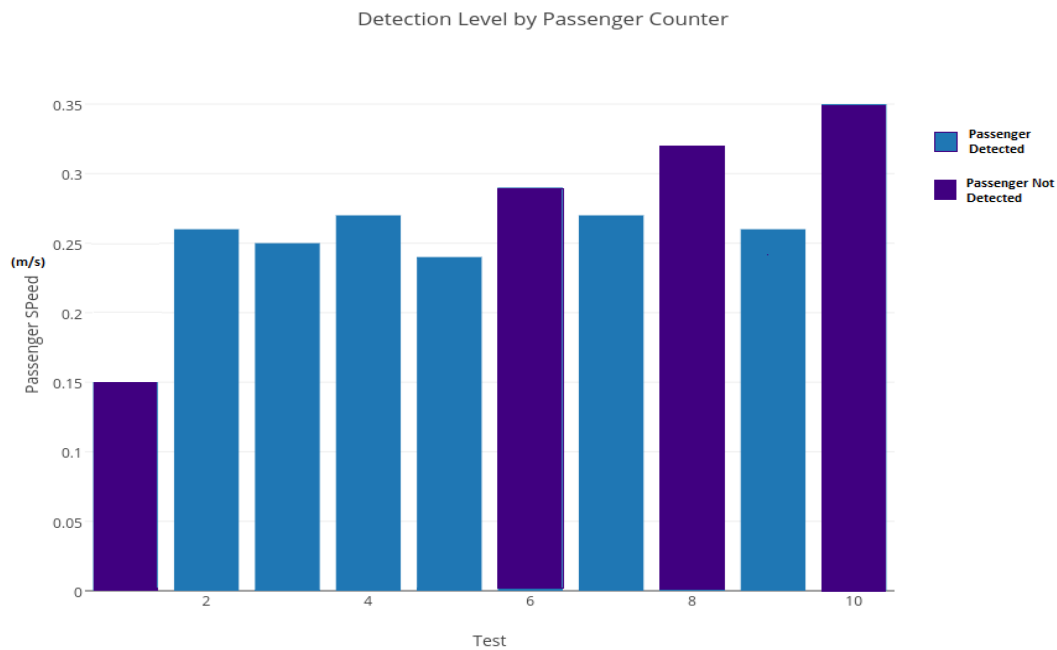
We plotted the Graph representing Temperature VS Pressure and Humidity. From Graph, it is understood that humidity changes with of temperature, that is, the percentage humidity decreases with the increase of temperature. But on the other hand, there was not any significant change in air pressure at normal conditions but just the rise in temperature. It can be deduced that, since there was no drastic change in air pressure the temperature conditions were absolutely normal and the change in humidity was quite clear as the temperature changed.

4.1.2 Passenger Counter Analysis

We have designed a passenger counter in our implemented part that can perform the operation of counting the number of passengers boarding the ship the increment or decrementing the counter when required. But it was necessary for us to test that, to how much extent this system can work. We ran an experiment, to see whether how many people can pass the counter and get detected. We acquired 10 sets of results for this experiment. Every time, we recorded the amount of time taken for the passenger to cross the sensors and from the distance between the sensors, we were able to calculate the speed of the passengers passing the passenger counter using formula $v = s/t$.

The table below shows the speeds and the detection by the passenger counter for each set of experiment.

Test	Passenger Speed (m/s)	Detection
1	0.15	Passenger Not Detected
2	0.26	Passenger Detected
3	0.25	Passenger Detected
4	0.27	Passenger Detected
5	0.24	Passenger Detected
6	0.29	Passenger Not Detected
7	0.27	Passenger Detected
8	0.32	Passenger Not Detected
9	0.26	Passenger Detected
10	0.35	Passenger Not Detected



From the above results, it is observed that the sensor for the passenger counter is for a specific range, which is about 0.20m/s to 0.28m/s of speed, beyond this range, the passenger getting into the ship cannot be recognized and thus give inaccurate information about the passengers on-board. Therefore, it would tend to work best when people get on board in queue and move inside the ship at a slow pace.

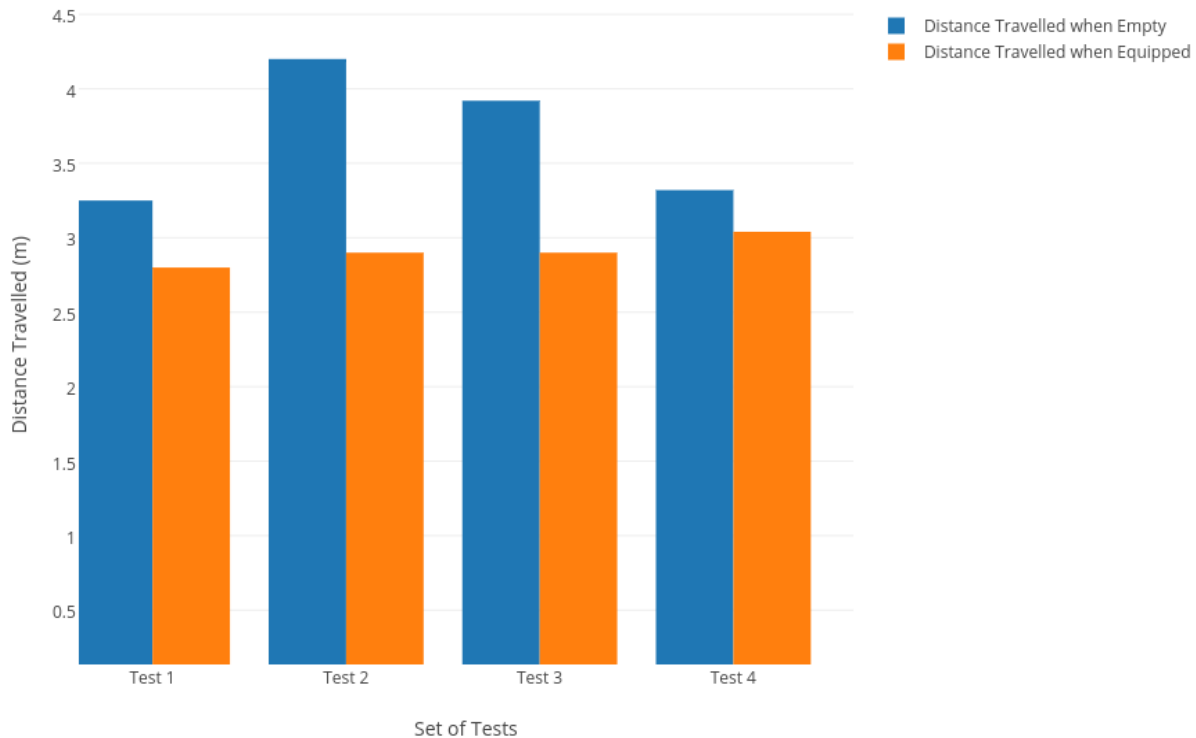
4.2 Trigger Unit Efficiency Analysis

Since the work of the trigger unit is to eject the Black Box unit with the thrower cannon, it is very important to determine how much efficient the thrower cannon can be to shoot the Black box unit during marine accidents. We have tested the thrower cannon with an empty Black Box container and noted how much distance it can cover when it is thrown from the thrower cannon. We repeated the same experiment but this time we equipped the Black box container with the Black Box unit and observed how much it differs from the previous result. We have taken four sets of data to analysis this, and these are given below:

Test	Distance traveled when empty	Distance traveled when equipped
1	3.25	2.8
2	4.2	2.9

3	3.92	2.9
4	3.32	3.04

Comparison of Distance Travelled by Black Box when Empty and when Equipped



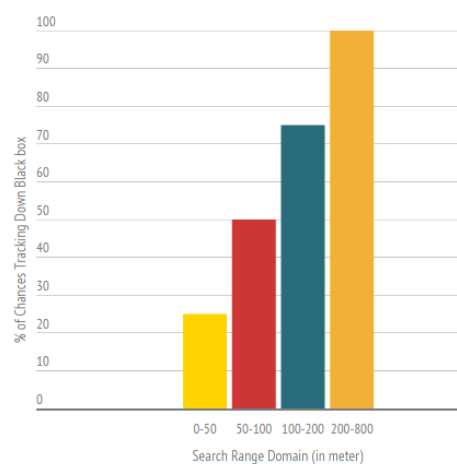
From the experiment, we understood that, there is not much variation in the distance covered as we can observe from the graph that the container covered $\frac{3}{4}$ th of the distance when empty, when it was equipped. So it can be easily deduced that, the distance that it cover quite a good amount of distance and good enough to get ejected from the ship while catastrophe.

4.3 GPS Data Accuracy

According to our project plan, on a critical maritime situation the black box will be triggered away from the marine vehicle whom it will remain attached until. In such cases if the black box can be tracked down the water vehicle which potentially faced an accident will be also tracked down since the black box will remain floating on the water and attached with a string to it. This makes the accuracy of the logged location co-ordinates from the black box to the raw data

interface extremely significant. The location co-ordinates the black box retrieves using its GPS module which gets pushed to the web server and the actual location co-ordinates may vary due to various circumstantial reasons. Therefore, we have taken a set of logged entries by our black box to the server and compared with the actual exact location of it to find the inconsistency of the GPS data and as the results suggest in the following graph, the location co-ordinates black box gets using GPS module and the actual co-ordinates are almost identical.

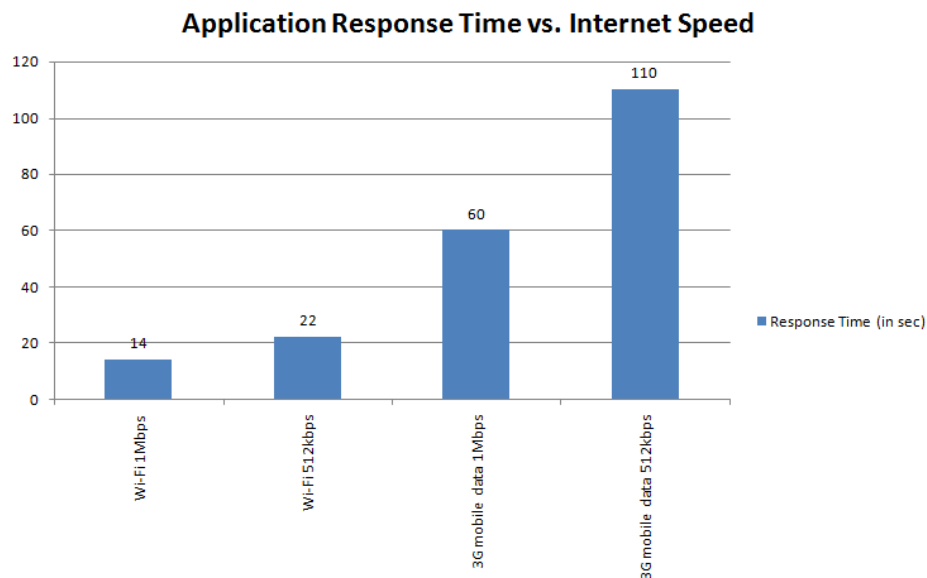
GPS Data Accuracy



On basis of numerous location co-ordinate data entry, we have found that if we search with 50 meters domain of the location co-ordinates logged to server from the black box, there is 25% chance to get the black box. Again if the search domain is increased to 100 meters the percentage of chance to track down the black box successfully, reaches to 50. When we tried to log data from our University Building no 5, the co-ordinates logged to the server was about 750 meters away from its exact location. We assume this happened due to presence of number of wireless communication setups installed in that building which hampered the performance of the black box in terms of lowering the accuracy of GPS data. That is why we have set the threshold to 800 meters considering the worst case conditions which suggests if black box is searched with a domain of the threshold, rescue team is bound to discover the black box meaning the percentage of chance going to 100.

4.4 Application Response Time vs. Internet Speed

Since our Android application retrieves data from remote database using web services, the amount of time it takes to parse data to Android device should be less than the interval (5minutes as we have set) in which the black box logs data i.e. location co-ordinates along with other sensor values got from the on board processor, to function the entire system as per plan. We understand the response time of our Android application has a relation with the speed of the internet device has been using and thus we have several times run our application to find average response times while using Wi-Fi and 3G mobile data.



When end users will be having 1Mbps Wi-Fi data enabled, the average time our Android application will take is 14 seconds while with 512kbps it would take 22 seconds on average for the application to respond. We also observed that the same variable has a larger value in region of a minute while using 3G mobile data of 1Mbps and around 2 minutes while using mobile data of 512kbps.

4.5 Compatibility across various Android API

The Android application we have developed has the minimum Android API level set to 14 i.e. Platform Version Android 4.0, 4.0.1, 4.0.2 and the target Android API level set to 21 i.e. Platform Version Android 5.0. According to the application manifest, any Android device with the system API level between 14 to 21 should not have any compatibility difficulties. In order to enquire, we installed our application in five Android devices running on five different system level API and found out that except for one to two minor user interface issues such as partial progress bar display, there were no significant difficulties faced by devices to function the application as it was intended to do.

Chapter - 5

Discussion

We need to specify that, we tried our task in a few circumstances. We found that, the outcomes are very acceptable. From the readings of RF sensors to pushing information on the WEB database, everything was near be flawless and very attractive. We composed the traveler counter and made the restricted stockpiling of the water vessel. In the wake of testing we went to the outcome that is working splendidly and it is really limiting the over-burdening of the Marine vehicles as it is one of the fundamental reason of mishaps. Also our configuration of black box is near impeccable as it is totally water resistant to spare the electrical gadgets. We additionally included two stages for security measures. The entire black box is made of stainless steel and all that much shock proof to make the framework more qualified to use in the marine vehicles as security instrument. We additionally inform to expand the number with respect to rafts and life coats to diminish the life misfortune in the disastrous marine mischances. We need to specify that our electrical circuit is all that much light and sturdy to stay quite a while on the water. Again we made it more water resistant by utilizing the silicon covering as it will be sturdy in the water.

The link connected to the black box is additionally much solid to take the power of the solid current that is regularly on the courses we said before. We utilized 120 feet link at first for trial purpose, later we will include more link after exploration of all the activity and profundity of the waterways.

In addition we had a short session where we scrutinized about the normal profundity of the waterways in our nation. We figured out that the normal profundity of the vast majority of the streams in our nation including Brahamaputra (124 feet), Buriganga (93 feet), Dhaleshwari (119 feet), Kushiya (33 feet), Naf (128 feet), Shitalakshya (33 feet), Titas (100 feet) and so on are between the scope of 33 feet to 128 feet. So on the majority of the circumstances, our system of tracking the affected vehicle must be pin point exact towards our arranging and execution.

Despite what might be expected our framework may confront critical difficulties for the huge streams where normal profundity is significantly more than our normal exploratory arrangements

of execution. Substantial waterways like Padma (968 feet), Meghna (1012 feet), Surma (282 feet) and so forth where our trial seeking configuration can be a major test to execute. However, we additionally need to guarantee that, augmenting the length of the string we utilized as a part of our analysis can totally pulverize the restrictions we clarified here for the vast and more normal profundity waterways.

As we said in the information examination area, our trigger unit reaction time is quick and precise and we really composed an exceptionally tough outline of thrower cannon which can't be influenced by the tempests and other common or human catastrophes. We utilized water level sensor which can recognize the water levels and make the trigger unit reaction when risk arrives. The exactness level of water level sensor is great and right around 100% time it worked appropriately on our various examinations.

We ought to likewise include that we have confronted some noteworthy difficulties to get our fancied framework working on practically exact parameters. Firstly we needed to confront the truth of picking an arrangement, an undertaking which is out of the container. We have seen secret elements of Planes, autos, and salvage frameworks. The majority of them were just to give an embodiment to fabricate an effective one. Be that as it may, in the marine side, we fundamentally had not very many papers to get some learning of the entire framework. We discovered a few frameworks and papers which are not identified with our framework. We really included the encounters and papers, included the electrical, communication learning to make it conceivable. Again another test was to locate a mechanical structure which is strong in the meantime water resistant. We looked into and discovered that the sphere like black box would be ideal for our wanted target.

Besides after these we needed to build up the correspondence segment from on board processor to the black box. That part was essentially the most difficult piece of our entire undertaking. We needed to utilize the RF transmitter and beneficiaries to make the on board processor speak with the Black Box. At first the procedure was not living up to expectations. Really the association between Arduino Uno and RF are direct connection. Since in both of the units we utilized various gadgets, for example, Barometric Weight Sensor and SD card Shield which required the Serial Fringe Interface alongside the RF module. We tackled this utilizing an additional Arduino

Pro Mini. We joined the RF specifically with the Arduino Pro Mini and a short time later made a serial correspondence between two arduino. This procedure obliged more mind boggling associations and materials which was extremely tedious to get the circuit working.

Another issue we confronted about the separation on the application we made to track the black box. At first the application separation was varying around 500m or 100m from the accurate position as we clarified in the information examination part. However, now at most extreme point it gives the accurate value or 50mmore or less on the Google map.

For the present the Android application we have composed has the base Programming interface level set to 14 i.e. stage adaptation Android 4.0,4.0.1,4.0.2, which implies Android gadgets with system lower than minimum API level would not be introduced. In future, we plan to reexamine the application with the goal that it can be introduced on gadgets running on all accessible Android Programming interface level systems.

Again we have some issue about the trigger unit particularly at the thrower part. We felt befuddled about how we can apply the power on the trigger unit. So eventually we got the opportunity to work with the springs to get the power connected to toss the ball. In any case, the issue still exists about the separation. It covers 10 or less in the wake of tossing. We recognized the issue. Presently we have utilized 2 springs to be correct on one side, 4 all in all unit. We recognized that on the off chance that we utilize more springs the flexibility would be better and the thrower will toss further from the influenced marine vehicle.

There are some tentative arrangements and viewpoints we have about our undertaking. We trust that the passenger counter we utilized that could be more overhauled and we could have utilized more defenses over the procedure. We have arrangements to make the passenger counter procedure to be involved with the particular ID of the passengers. Further we have arrangements to execute it with the biomedical example as fingerprints. Since at the greatest cases we see that the real individuals influenced by setbacks gets missing and even the dead bodies couldn't be found. So the genuine personality gets revealed. So we thought about these procedure.

In addition we have additionally arranged about the mechanical piece of the Black Box. Because of specialized challenges we needed to make the one side of the discovery level to make it glide.

We have arrangements to make the entire black box round and make the substance of the ball screw string to open it all the more productively.

Again on the Android application's climate sensor information part we have just shown the estimations of the sensors in the wake of parsing from the raw information interface. In future we have arrangements to change over this into a neighborhood climate station for the oceanic vehicles. We plan to contrast the sensor values and the authentic information as far as the sensor qualities recorded while water vehicles really confronted mischances or discriminating circumstances. Contingent upon the correlation results, we might want to give some kind of warning to the concerned power that sensor qualities being assembled are disturbing.

To get overhauled information in our Android application we now need to stop the application and after that begin from the earliest starting point. We plan to execute swipe revive to upgrade the most recent quality from the database without leaving the application.

Another major issue was how will be a route shown when the black box will logging data to the designated web server from the middle of the river. We have experimented by setting the black box mock location to be amidst the water and have seen that the route is shown to the nearest possible bank. We understand that showing the route up to the nearest possible bank of the river while the actual black box being within half or one kilometers of distance, is still considered to be success since the search domain is being reduced to one or two kilometers of radius.

Conclusion:

As a matter of first importance, we need to express appreciation towards Almighty Allah that we have completed our thesis successfully and in the meantime proficiently. From the soonest beginning stage to the fulfillment of the hypothesis there were different stories stacked with accomplishment and wretchedness. It is our pleasure that we picked such a theme where we could investigate ourselves, tossed ourselves at the last purpose of our test of adequacy and to the certain point that has its vast effect in the life of every one. When we encountered and understood the significance of our undertaking, taking a gander at the particular zones which are still such a great amount of absence of advancements in the time of digitization. So our venture can be the first stride of digitization in the field of Marine transportation arrangement of

Bangladesh. We fundamentally took a shot at the particular regions where we understood that there we can make such framework where heaps of lives can be spared as well as it can be exceptionally proficient and successful to the advancement of numerous zones in the late creating period of Bangladesh. We trust our framework is all that much simple to work and in the meantime it is all that much exact to make the rate of marine mishaps low by putting forth an in number expression to the consciousness of our Water Transport Power. On the off chance that our framework is upheld into the arrangement of BIWTA the marine mishaps can be lower and the courses will be extremely sheltered. We constructed the framework such proficiently that anybody can get to the fundamental information of the particular water vessel. The information are, for example, temperature, humidity, pressure and passengers on board. These data are all that much critical in the instances of mishaps and discovering the reason of mischances. We mulled over heaps of information about the late marine accidents. We discovered loads of circumstances furthermore clarified it prior about the rate of missing dispatches and water vessels. With our framework these missing disasters can be more confined to the way that we have composed the entire framework to limit the loss of missing accidents.

Chapter - 6

Reference

1. AZAD, ABUL KALAM. RIVERINE PASSENGER VESSEL DISASTER INBANGLADESH: OPTIONS FOR MITIGATION AND SAFETY. Thesis. BRAC UNIVERSITY, 2009. N.p.: n.p., n.d. RIVERINE PASSENGER VESSEL DISASTER IN BANGLADESH: OPTIONS FOR MITIGATION AND SAFETY. Postgraduate Programs in Disaster Management (PPDM). Web.
2. Mohosinul Karim. "Above 4,000 Deaths from Launch Accidents in 38 Years -." *Dhaka Tribune*. Dhaka Tribune, 05 May 2014. Web.
3. Musharraf, Yaseer, Maruf Hassan, and Shadman Sakib ‘ *A Distanced Machinery Controlling and Monitoring Guardian*.’ Thesis. BRAC UNIVERSITY, 2014. N.p.: Department Of Electrical and Electronic Engineering;, n.d. *A Distanced Machinery Controlling and Monitoring Guardian*. Web.
4. Paluska, Daniel, and Hugh Herr. "The Effect of Series Elasticity on Actuator Power and Work Output: Implications for Robotic and Prosthetic Joint Design." *Robotics and Autonomous Systems* 54.8 (2006): 667-73. Web.
5. "Storms Are Getting Stronger." [Http://earthobservatory.nasa.gov/](http://earthobservatory.nasa.gov/). NASA EARTH OBSERVATORY, n.d. Web.
6. "Future Voyage Data Recorder Based on Multi-sensors and Human Machine Interface for Marine Accident." *IEEE Xplore*. N.p., n.d. Web.
7. Kassem, Abdallah, Rabih Jabr, Ghady Salamouni, and Ziad Khairallah Maalouf. "Vehicle Black Box System." *Vehicle Black Box System* (2008): n. pag. [Http://ieeexplore.ieee.org/](http://ieeexplore.ieee.org/). IEEE, 10 Apr. 2008. Web. 02 Dec. 2014.
8. Grow, Erica. "Anatomy of a Storm: Pressure Rises and fall." [Http://www.wusa9.com/](http://www.wusa9.com/). [Http://www.wusa9.com/](http://www.wusa9.com/), 4 Sept. 2014. Web.

9. Nettigo. "*Connecting and Programming NRF24L01 with Arduino and Other Boards - Starter Kit.*" *Starter Kit RSS*. N.p., 04 Dec. 2014. Web.
10. Case, Jenn. "*Arduino to Arduino Serial Communication.*" *Arduino to Arduino Serial Communication*. Robotic Controls, 06 Feb. 2013. Web.
11. Matthew. "Projects from Tech." : *Arduino: Serial Communication Between Two Arduinos*. N.p., 11 May 2013. Web.
12. "Tutorial – Arduino and SIM900 GSM Modules." *Tronixstuff*. N.p., 08 Jan. 2014. Web.
13. "SIM900 AT Command Manual V1.03." (n.d.): n. pag. Web.
14. "Fundamentals:Radio Frequency Shielding." *Architecture Design Handbook: Fundamentals: Radio Frequency Shielding*. N.p., n.d. Web.
15. "4 Popular Methods of GPS Jammers." *GPS Systems*. N.p., 27 Apr. 2010. Web.
16. Patil, Chetan,Yashwant Marathe, Kiran Amoghimath, and Sumam David. "*Low Cost Black Box for Cars.*" *IEEE Xplore*. IEEE, 2013. Web. 12.
17. COHEN, ARIEL E., STEVEN M. CAVALLO, MICHAEL C. CONIGLIO, and HAROLD E. BROOKS. "*A Review of Planetary Boundary Layer Parameterization Schemes and Their Sensitivity in Simulating Southeastern U.S. Cold Season Severe Weather Environments.*" *American Meteorological Society*. N.p., n.d. Web. 2015.
18. Minkyun Noh, and Seung-Won Kim,. "*Flea-Inspired Catapult Mechanism for Miniature Jumping Robots.*" *IEEE Xplore*. IEEE, 5 Oct. 2012. Web.
19. "*Flea Inspired Catapult Mechanism with Active Energy Storage and Release for Small Scale Jumping Robot.*" *IEEE Xplore*. N.p., n.d. Web.
20. Sosnowski, Alex. "*Drastic Temperature Change For Midwest, East.*" www.accuweather.com, n.d. Web.
21. Toothman, Jessika. "*Is there really a calm before the storm*" 13 May 2008. HowStuffWorks.com. <<http://science.howstuffworks.com/nature/climate-weather/storms/calm-before-storm.html>
22. Nagesh, Anirudh, Keshav Khandelwal, and Carlos E. Caicedo. "Accessing External Databases from Mobile Applications, Technical Report, Center for Convergence and Emerging Network Technologies." Thesis. Syracuse University, 2014. *Accessing External Databases from Mobile Applications, Technical Report, Center for Convergence and Emerging Network Technologies*. Syracuse University. Web.

Chapter - 7

Appendix

Code Section:

Application source code:

MainActivity.java

```
package bd.ac.bracu.marineblackbox;
```

```
import java.util.ArrayList;
```

```
import java.util.HashMap;
```

```
import org.json.JSONArray;
```

```
import org.json.JSONException;
```

```
import org.json.JSONObject;
```

```
import android.app.Activity;
```

```
import android.app.ProgressDialog;
```

```
import android.content.Intent;
```

```
import android.os.AsyncTask;
```

```
import android.os.Bundle;
```

```
import android.util.Log;
```

```
import android.view.Menu;
```

```
import android.view.MenuItem;
```

```
import android.view.View;
```

```
import android.view.View.OnClickListener;
```

```
import android.widget.Button;
```

```
public class MainActivity extends Activity {

    private ProgressDialog pDialog;

    Button loc;

    Button wea;

    Button pas;

    String count = "";

    String temp = "";

    String hum = "";

    String pre = "";

    String longi = "";

    String lat = "";

    String time = "";

    // URL to get contacts JSON

    private static String url = "https://data.sparkfun.com/output/jq33axOErgin8vLOj1ra.json";

    // JSON Node names

    private static final String TAG_COUNT = "passenger_count";

    private static final String TAG_TEMP = "temperature";

    private static final String TAG_HUM = "humidity";

    private static final String TAG_PRE = "pressure";

    private static final String TAG_LONG = "longitude";
```

```

private static final String TAG_LAT = "latitude";

private static final String TAG_TIME = "timestamp";


// JSONArray

JSONArray points = null;

@Override

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    loc = (Button) findViewById(R.id.button1);

    wea = (Button) findViewById(R.id.button2);

    pas = (Button) findViewById(R.id.button3);


loc.setOnClickListener(new OnClickListener(){

    public void onClick(View v){

        Intent intLoc = new Intent(MainActivity.this, LocationActivity.class);

        intLoc.putExtra(TAG_LAT, lat);

        intLoc.putExtra(TAG_LONG, longi);

        startActivity(intLoc);

    }

    });

wea.setOnClickListener(new OnClickListener(){

    public void onClick(View v){

        Intent intWeather = new Intent(MainActivity.this,
WeatherActivity.class);

        intWeather.putExtra(TAG_TEMP, temp);

```

```

        intWeather.putExtra(TAG_HUM, hum);

        intWeather.putExtra(TAG_PRE, pre);

        startActivity(intWeather);

    }

});

pas.setOnClickListener(new OnClickListener(){

    public void onClick(View v){

        Intent intPassenger = new Intent(MainActivity.this,
PassengerActivity.class);

        intPassenger.putExtra(TAG_COUNT, count);

        startActivity(intPassenger);

    }

});

// Calling async task to get json

new GetPoints().execute();

}

/**
 * Async task class to get json by making HTTP call
 * */

private class GetPoints extends AsyncTask<Void, Void, Void> {

    @Override

    protected void onPreExecute() {

        super.onPreExecute();

        // Showing progress dialog

        pDialog = new ProgressDialog(MainActivity.this);

        pDialog.setMessage("Loading...");

```

```

        pDialog.setCancelable(false);

        pDialog.show();

    }

    @Override
    protected Void doInBackground(Void... arg0) {

        ServiceHandler sh = new ServiceHandler();

        String jsonStr = sh.makeServiceCall(url, ServiceHandler.GET);

        Log.d("Response: ", "> " + jsonStr);

        if (jsonStr != null) {
            try {

                JSONArray json = new JSONArray(jsonStr);

                JSONObject c= json.getJSONObject(i);

                count = c.getString(TAG_COUNT);

                temp = c.getString(TAG_TEMP);

                hum = c.getString(TAG_HUM);

                pre = c.getString(TAG_PRE);

                longi = c.getString(TAG_LONG);

                lat = c.getString(TAG_LAT);

                time = c.getString(TAG_TIME);

            } catch (JSONException e) {

                e.printStackTrace();

            }
        } else {

            Log.e("ServiceHandler", "Couldn't get any data from the url");
        }
    }
}

```



```

        }

        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);

        // Dismiss the progress dialog
        if (pDialog.isShowing())
            pDialog.dismiss();

        // Updating parsed JSON data into ListView
    }
}

```

LocationActivity.java

```
package bd.ac.bracu.marineblackbox;
```

```
import android.app.Activity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.view.View.OnClickListener;
```

```
import android.widget.Button;
```

```
import android.widget.TextView;
```

```
public class LocationActivity extends Activity{
```

```

// JSON node keys

    private static final String TAG_LONG = "longitude";

    private static final String TAG_LAT = "latitude";

    private static final String TAG_TIME = "timestamp";

    String longi = "";

    String lat = "";

    Button mapButton;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_location);

        // getting intent data

        Intent in = getIntent();

        // Get JSON values from previous intent

        longi = in.getStringExtra(TAG_LONG);

        lat = in.getStringExtra(TAG_LAT);

        TextView lblLongi = (TextView) findViewById(R.id.longi_label);

        TextView lblLat = (TextView) findViewById(R.id.lat_label);


        lblLongi.setText(longi.substring(0, 5));

        lblLat.setText(lat.substring(0, 5));


        mapButton = (Button) findViewById(R.id.btn_enable);

        mapButton.setOnClickListener(new OnClickListener() {

            public void onClick(View v){

```

```

        Intent intMap = new Intent(LocationActivity.this, MapActivity.class);

        intMap.putExtra(TAG_LAT, lat);

        intMap.putExtra(TAG_LONG, longi);

        startActivity(intMap);
    }

});

}

}

```

MapActivity.java

```

package bd.ac.bracu.marineblackbox;

import java.util.ArrayList;

import java.util.List;

import org.w3c.dom.Document;

import android.app.ProgressDialog;

import android.content.Intent;

import android.content.IntentSender;

import android.graphics.Color;

import android.graphics.drawable.Drawable;

import android.location.Location;

import android.location.LocationManager;

import android.os.AsyncTask;

import android.os.Bundle;

import android.support.v4.app.FragmentActivity;

import android.util.Log;

```

```

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.Overlay;

public class MapActivity extends FragmentActivity implements
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    public static final String TAG = LocationActivity.class.getSimpleName();

    private static final String TAG_LONG = "longitude";

    private static final String TAG_LAT = "latitude";

    private final static int CONNECTION_FAILURE_RESOLUTION_REQUEST = 9000;

    private GoogleMap myMap;

```

```

private GoogleApiClient myGoogleApiClient;

private LocationRequest myLocRequest;

List<Overlay> myMapOverlays;

GeoPoint p1, p2;

LocationManager myLocManager;

Drawable drawable;

Document document;

GMapV2GetRouteDirection v2GetRouteDirection;

LatLng currentLatLng;

LatLng fromPosition;

LatLng toPosition;

//GoogleMap mGoogleMap;

MarkerOptions myMarkerOptions, myMarker;

Location location;

String longi="";

String lat="";

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_map);

    Intent in = getIntent();

    // Get JSON values from previous intent

    lat = in.getStringExtra(TAG_LAT);

    longi = in.getStringExtra(TAG_LONG);

```

```

v2GetRouteDirection = new GMapV2GetRouteDirection();

setUpMapIfNeeded();

myGoogleApiClient = new GoogleApiClient.Builder(this)

    .addConnectionCallbacks(this)

    .addOnConnectionFailedListener(this)

    .addApi(LocationServices.API)

    .build();

myLocRequest = LocationRequest.create()

    .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)

    .setInterval(10 * 1000)    // 10 seconds, in milliseconds

    .setFastestInterval(1 * 1000); // 1 second, in milliseconds
}

@Override

protected void onResume() {

    super.onResume();

    setUpMapIfNeeded();

    myGoogleApiClient.connect();

}

@Override

protected void onPause() {

    super.onPause();

    if (myGoogleApiClient.isConnected()) {

        LocationServices.FusedLocationApi.removeLocationUpdates(myGoogleApiClient, this);

        myGoogleApiClient.disconnect();

```

```

    }
}

protected void onStop() {
    super.onStop();
    finish();
}

private void setUpMap() {
    myMap.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
}

private void handleNewLocation(Location location) {
    Log.d(TAG, location.toString());

    LatLng latLng2 = new LatLng(Double.parseDouble(lat), Double.parseDouble(longi));

    myMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng2, 14));

    double currentLatitude = location.getLatitude();
    double currentLongitude = location.getLongitude();

    LatLng currentLatLng = new LatLng(currentLatitude, currentLongitude);
    myMap.moveCamera(CameraUpdateFactory.newLatLng(currentLatLng));

    myMarkerOptions = new MarkerOptions();
    myMarker = new MarkerOptions();

    fromPosition = currentLatLng;
    toPosition = latLng2;

    GetRouteTask getRoute = new GetRouteTask();
    getRoute.execute();
}

private void setUpMapIfNeeded() {

```

```

if (myMap == null) {

    myMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map))
        .getMap();

    if (myMap != null) {
        setUpMap();
    }
}

@Override

public void onConnected(Bundle bundle) {

    Location location = LocationServices.FusedLocationApi.getLastLocation(myGoogleApiClient);

    if (location == null) {

        LocationServices.FusedLocationApi.requestLocationUpdates(myGoogleApiClient, myLocRequest,
this);

    }

    else {

        handleNewLocation(location);

    }
}

private class GetRouteTask extends AsyncTask<String, Void, String> {

    private ProgressDialog Dialog;

    String response = "";

    @Override

    protected void onPreExecute() {

        Dialog = new ProgressDialog(MapActivity.this);

```



```

        Dialog.setMessage("Showing Route...");

        Dialog.show();
    }

    @Override

    protected String doInBackground(String... urls) {

        document = v2GetRouteDirection.getDocument(fromPosition, toPosition,
GMapV2GetRouteDirection.MODE_DRIVING);

        response = "Success";

        return response;
    }

    protected void onPostExecute(String result) {

        myMap.clear();

        if(response.equalsIgnoreCase("Success")){

            ArrayList<LatLng> directionPoint = v2GetRouteDirection.getDirection(document);

            PolylineOptions rectLine = new PolylineOptions().width(8).color(

                Color.BLUE);

            for (int i = 0; i < directionPoint.size(); i++) {

                rectLine.add(directionPoint.get(i));
            }

            // Adding route on the map

            myMap.addPolyline(rectLine);

            myMarkerOptions.position(toPosition);

            myMarkerOptions.draggable(true);

            myMap.addMarker(myMarkerOptions);

```

```

    }

    Dialog.dismiss();
}

}

public void onConnectionFailed(ConnectionResult connectionResult) {

    if (connectionResult.hasResolution()) {

        try {

            connectionResult.startResolutionForResult(this,
CONNECTION_FAILURE_RESOLUTION_REQUEST);

        } catch (IntentSender.SendIntentException e) {}

    } else {

        Log.i(TAG, "Location services connection failed with code " + connectionResult.getErrorCode());

    }

}

@Override

public void onLocationChanged(Location location) {

    handleNewLocation(location);

}

}

```