Department of Electrical and Computer Engineering
North South University

Senior Design Project

Aquatic Iguana: Water Waste Collector Drone with Water Quality Monitoring System

Abdullah Md. Humayun Kabir **ID # 1511828042**

Mirza Turesinin **ID # 1511819042**

Tanzina Mollah **ID # 1511777042**

Sadvan Sarwar **ID# 1511912042**

Faculty Advisor:

Dr. Shazzad Hosain
Associate Professor ECE Department

Spring, 2019

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' names & Signatures

1. Abdullah MD. Humayun Kabir

   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

2. Mirza Turesinin

   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

3. Tanzina Mollah

   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

4. Sadvan Sarwar

   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

APPROVAL

We, **Abdullah MD. Humayun Kabir (1511828042)**, **Mirza Turesinin (1511819042)** and **Tanzina Mollah (1511777042)**, **Sadvan Sarwar (1511912042)**, members of CSE: 499 (Senior Design) from the Electrical and Computer Engineering department of **North South University**; have worked on the project titled **"Aquatic Iguana: Water Waste Collector Drone with Water Quality Monitoring System"** under the supervision of Dr. Shazzad Hosain as a partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

Supervisor's Signature

.....

Dr. Shazzad Hosain
Associate Professor

Department of Electrical & Computer Engineering

North South University
Dhaka, Bangladesh.

Chairman's Signature

.....

Dr. K. M. A. Salam
Professor & Chairman

Department of Electrical & Computer Engineering

North South University
Dhaka, Bangladesh

ACKNOWLEDGEMENT

By mercy of the Almighty we have completed our senior design capstone project entitled "Aquatic Iguana: Water Waste Collector Drone with Water Quality Monitoring System".

Foremost, we would like to express our sincere gratitude to our advisor Dr. Shazzad Hosain for his continuous support in our capstone project progress throughout the whole 499A and 499B, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research, writing and completing of this project.

Our sincere thanks also goes to North South University, Dhaka, Bangladesh for providing an opportunity in our curriculum which enabled us to have an industrial level experience as part of our academics.

Last but not the least, we would like to thank our family as their inspiration and guidance kept us focused and motivated.

Abstract

"Aquatic Iguana", a water drone which moves around the surface of water and collects floating waste material such as plastic, packets, leaves etc. Aquatic Iguana also collects data from water such as temperature of water, pH level, measures turbidity of water and sends it to the ThingSpeak server. It is controlled by a Radio Controller. A camera provides the view of surroundings to make the control system easier for the operator. We have developed this product to ensure the cleaning of water resources and to create a strong dataset of water quality for future prediction.

List of Figures

Table of Contents

Table of Contents

Chapter 1
Overview

1.1 Introduction:

Water pollution is one of the major environment pollutions. All over the world water is being polluted due to the negligence of people. Industrial waste materials, medical or clinical waste, waste of households, waste of ships, boats all are dumped into water resources such as river, ponds, lakes etc. Among those wastes, some wastes such as solid wastes are not disposed with water and it creates a solid layer on the top of the water which is not good for the environment and especially for the water living animals. Collecting those floating solid waste material such as plastic, packets, leaves, cigarettes, plastic bags, food containers, bottle caps etc. can be a great initiative for cleaning waster waste and also preventing water pollution. So the motivation behind our undertaking is to clean the water sources and monitor the water quality so that it can reduce water pollution and also can distinguish the clean and contaminated water. In addition, our drone has the capacity to measure real time (using ThingSpeak server which is an open IOT platform) temperature level, pH level and turbidity level of the water. Besides this, there is live streaming option that can help the controller or user to have a better view to operate. As in today's technological field, artificial intelligence is a great sector to work; our drone gives an opportunity to build real-time dataset for future machine learning prediction. The point of building drone "Aquatic Iguana" in a country like Bangladesh is because we use human resource to clean water that is not efficient in respect to time and money. So our drone can be a good way to clean the water resource efficiently and also observe the water quality to use.

1.2 Project Definition:

Solid wastes such as plastic, polythene, packets, leaves etc. are unsafe for water. Because it takes years to get disposed. As we all know plastic is made to be strong and durable. According to Trash Travels estimates that plastic bags can take 20 years to decompose, plastic bottles up to 450 years, and fishing line, 600 years [1]. In Bangladesh the condition of water pollution is too alarming. The per capita waste collected in Dhaka per day is 0.2 kilograms (Islam & Rahman, 2002). The waste which is not managed by the DCC is dumped into the rivers by the people living near the river banks [2]. There is no proper management system to collect these solid waste materials continuously and keep the water resources clean.

The water quality is important for the use of water in different sectors such as household work, farming, fishing and drinking etc. Almost all the river of Bangladesh is highly polluted so it is must to know the quality of water before use. If the pH level is not in the optimum level, fish can't survive. Fish can become stressed in water with a pH ranging from 4.0 to 6.5 and 9.0 to 11.0 [3]. Another important element of water is its temperature. Water temperature is significant in lakes and streams since it can figure out where in the water certain plants and creatures can live. At last the turbidity level is considered as a decent proportion of the nature of water. Turbidity is a proportion of how much the water loses its transparency because of the polluted elements.

So the purpose of our project is to make such a drone that will collect floating waste from water and to detect the contaminated or polluted water by using some sensors which can be beneficial for human beings, animals and plants. Besides, our drone can be able to surveillance the surrounding environment for better collecting and controlling.

1.3 Purpose of the Project:

In Bangladesh, the water waste collecting procedure is human based. Sometimes it's not possible to use human resource to clean the rivers, if we use human to collect wastes, it takes more human energy, time and money, that's where our drone is needed. In Dhaka, 26 canals near the city still survive, but they are in their death throes because of continued grabbing, obstruction and dumping of industrial and household wastes [4]. Because of water pollution, fishes can't survive in the water. The water monitoring system can give better knowledge about the water so that it can be in light whether to use those water or not. As Bangladesh is a river-land country, and most of its people is dependent on farming. Temperature, pH and turbidity will give an idea to know whether the water can be used to farming such as grow plants. By this water monitoring system, the farmer will also be going to know whether he can cultivate fish in the water or not. All the result of the sensors will be found via ThingSpeak server which is an open IOT platform.

If we can build less costly drones which can collect wastes from water, monitor the temperature, pH and turbidity level of water, it can be of great help in reducing water pollution and using

water in different sectors. Using camera on the drone can give us the live video streaming through website or mobile app. The webpage or app can be accessed by connecting a personal router which will be able to create Wi-Fi and give us the access to monitor the drone in certain range. So, a person will be able to monitor the drone and its surrounding environment.

1.4 Project Goal:

In this period of current innovation and improvement strategies and highlights of mechanical technology and robot related works has developed to a great extent. There are some works done before and has some similarity with our work. Some of the drones used in different countries to clean their lakes and ponds. In addition, some are working on water quality monitoring separately. The difference we are making is our desired drone will go to collect floating water waste, monitor water quality and a live streaming system to surveillance in environment.

The goal of the project is to provide solution to water pollution problem. Bangladesh is a riverine country but the most regretful matter is that due to poor waste disposal system and people's negligence a huge amount of floating waste material such as plastics, bottles, packets, floating debris, alien vegetation and many other stuffs end up in rivers, canals lakes and ponds, which pollutes the water resources.

So to make sure that our water resources remain clean we have come up with the idea of "Aquatic Iguana". We not only intend to use this to clean the water resources but also monitor

the water quality and provide a good amount of data set for future purposes. Our goal will be achieved as long as we can use this technology for the betterment of our water sources.

1.5 Summary:

If we could make a drone in low cost that can collect floating water waste, it can be a great help for our rivers, pond, canals etc. The drones will go to minimize the human energy, time and cost. In addition, the water quality monitoring system using temperature, pH and turbidity sensors can be a great help for the farmers and for those people who wants clean water to use. The result of these sensors will be found in ThingSpeak.com and it will give real time result using graph and csv file. The results also can be found in our own mobile app. The live streaming in webpage or mobile app will help the operator to see the surrounding environment and also to collect waste more efficiently in distance place. The drones will be a great help for not only the people but also the government to clean the rivers, because it's a combination of three important factors of water management system.

Chapter 2

Existing System /Background study

2.1 Introduction:

In this present era, robot and drone related works has become influential to a greater extent because of the new technology, innovation and advancement strategies. There are a few works done before and has some similarity with our work. Those worked targets on single task such as either cleaning purpose or water quality monitoring system. In our project we tried to assemble those things with some new innovative feature. The principle reason for the task will be accomplished in the event that we get the most ideal results of these issues. The chapter emphasizes on presently existing systems and how they work, similarity with those existing systems and also new features of our project. It also demonstrates and explains the problems, alternative solutions and best possible solution to tackle the problems.

2.2 Existing Solution:

In this period of present day innovation and improvement techniques and highlights of applyng robot, drone and robot related works has developed generally. There are some works done before and has some similarity with our work. Most of the drones related to this field are made by developed country as their technology is much richer than the under-developing countries. Internet-Controlled Trash Robot Will Clean Up the Chicago River [5] is one of them.

Fig.2.1. Clean-Up robot

Another similar work is SEAVAX adaptation to river cleaning workboats [6]. Waste Shark, an aqua drone that cruises around urban waters such as harbours, marinas and canals, eating up marine litter like a Roomba of the sea [7].

Fig.2.2. Waste Shark drone

Amphibious clean - up robot use solar energy to rotate and collect waste from water [8]. Arduino-based Smart Garbage Monitoring System [9] is another similar work in this field. Besides these drones, there is some work related to observe water quality. Smart Water Quality Monitoring System [10] is one of them. It also uses sensors to monitor the water quality. Water Quality Monitoring System: Water quality, anywhere, anytime! [11].

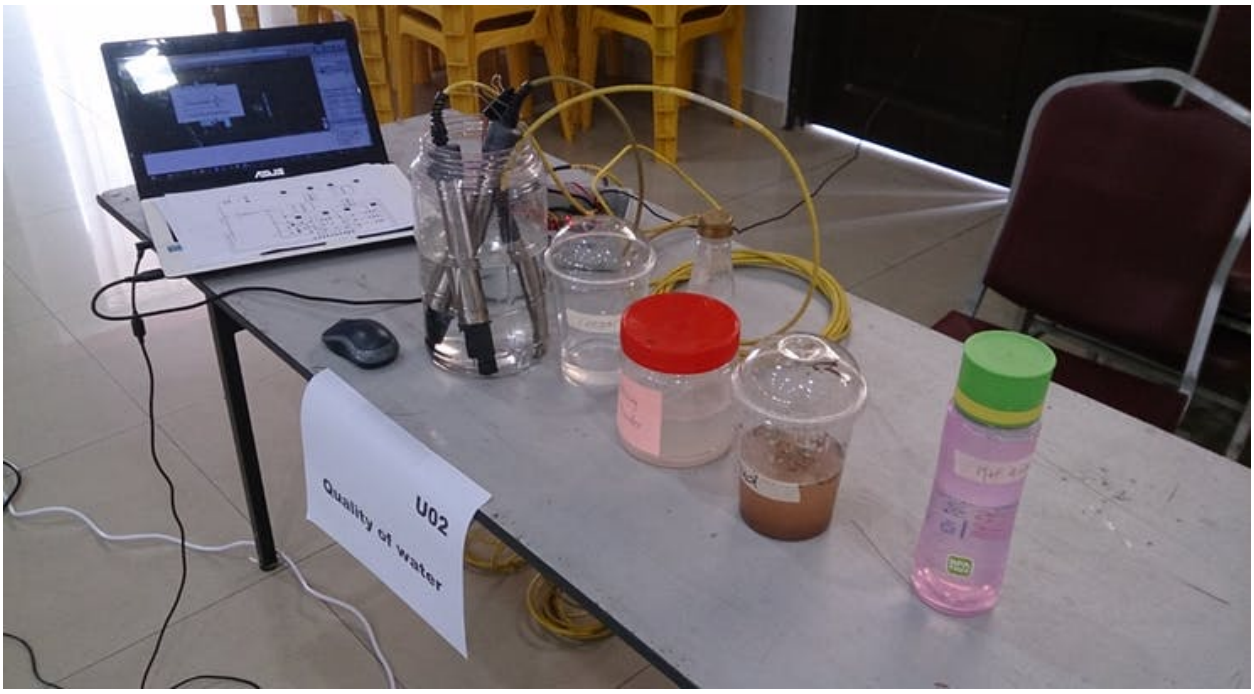

Fig.2.3. water quality monitoring system using sensors

Different drones have different features and specialized in one field. The difference is being made with the proposed drone as it will give a perfect combination of waste collection, water quality monitor and surveillance the surrounding environment. Thus it is expected to build a unique, improved and cost effective system.

2.3 Summary:

The objective of this chapter includes the utility of our system and background works proves that this system is needed in water cleaning and water quality monitoring purpose. There are a couple of works done before and has some similitude with our work. As we found that different work has been done in water cleaning and water quality monitoring separately. Our aim is to assemble this two system with a new feature known as live streaming to get a complete and efficient system which will reduce time, cost and using human resource in this field.

Chapter 3

Technical Description

3.1 Introduction

Here in this chapter the system block, characteristics, and the working principal of Aquatic iguana will described. The working principle of Arduino and Raspberry pi B+ will be discussed here as well. The brief description of the camera module used will also be discussed here. This chapter also includes the overall view of water quality monitoring system. And the reason of choosing individual component will also be given.

3.2 Overview of the system

As it is a water drone there are two wheels on both sides to make the drone move around the water surface. Both the wheels are made out of steel to maintain the balancing of the drone equally. And both of them are attached to two individual DC motors to move the drone in desire directions. All four directions are covered through this DC motors. The DC motors can be operated in 12V only. The shaft to collect the waste from the water is created from PVC fabric with steel blades in specific distance. PVC fabric is used because others materials such as rubbers can get ripped off when it is in contact of water and pulled continuously for a long time, but the PVC fabric always remains on the safe side and is waterproof. To pull the shaft upward and collect the waste another two DC motors are used. These 12V motors are capable of pulling approximately 30kgs at one time. So with the pressure of shaft pulling it towards the water and the weight of the waste, other low torque motors will not be capable of pulling it towards the collector. For water quality monitoring system three different sensors are used such as PH sensor, temperature sensor and turbidity sensor.

All of the sensors are connected to Node MCU 8086 (Wifi module) and another Arduino to collect the data from the water and put it in an online server named Thinkspeak.

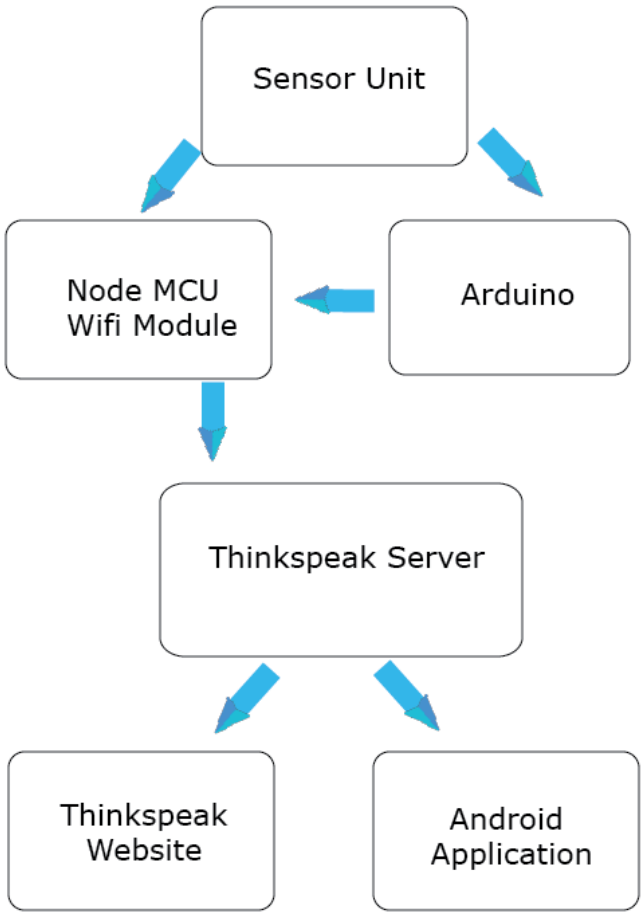

Fig 3.1: Workflow of water quality monitoring system

Later the data is collected on a custom made android application in JSON format, and decoded to text format and shown on the screen. The fetching procedure is done through API provided by thinkspeak.

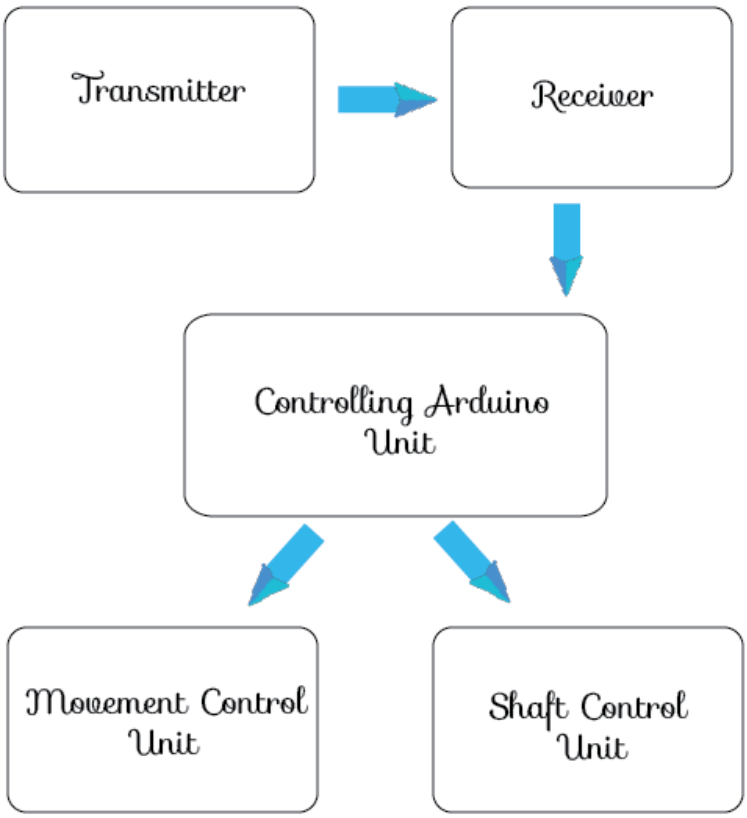

Fig 3.2: Block Diagram of controlling system

To control the drone wirelessly a communication link is established between the transmitter and a receiver. The receiver is bind with the wireless transmitter only, so it won't receive any other frequency coming towards it. For the live streaming Raspberry pi B+ and a Raspberry pi camera module is used. The Raspberry pi B+ gets the frames from the camera module and puts it to a specific hosting after processing the frames, later the frames are collected in the custom made android application and processed to the video streaming data. That's how the system works; detailed discussion will be done in next consecutive sections.

3.3 System Block diagrams

This section will show the whole system with the help of block diagrams in details.

i. Movement and shaft control:

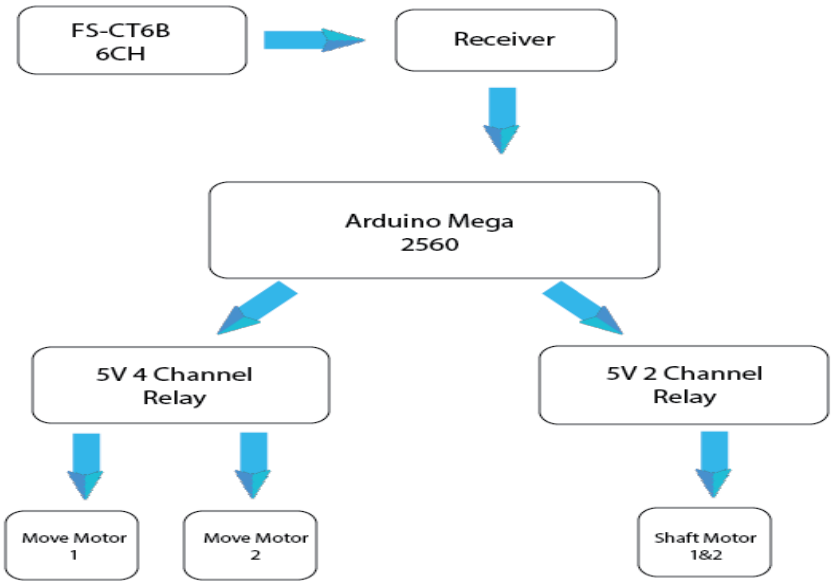

Fig 3.3: Movement control block diagram

ii. Live Streaming:

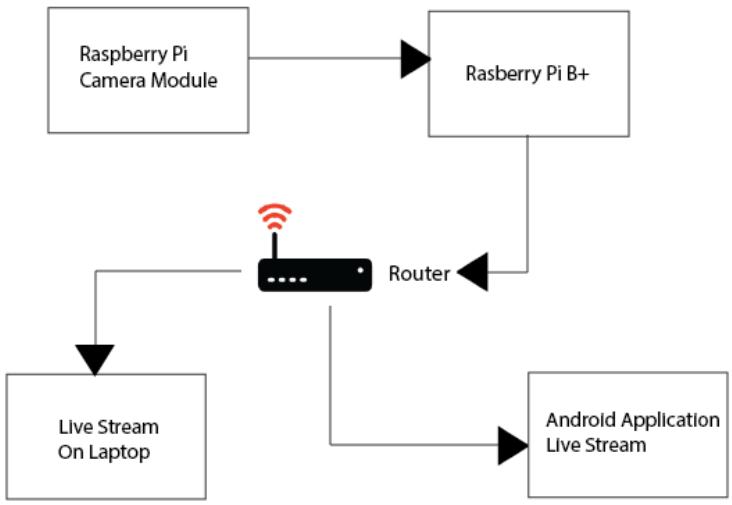

Fig 3.4: Live Streaming block diagram

iii. Sensor Communication:

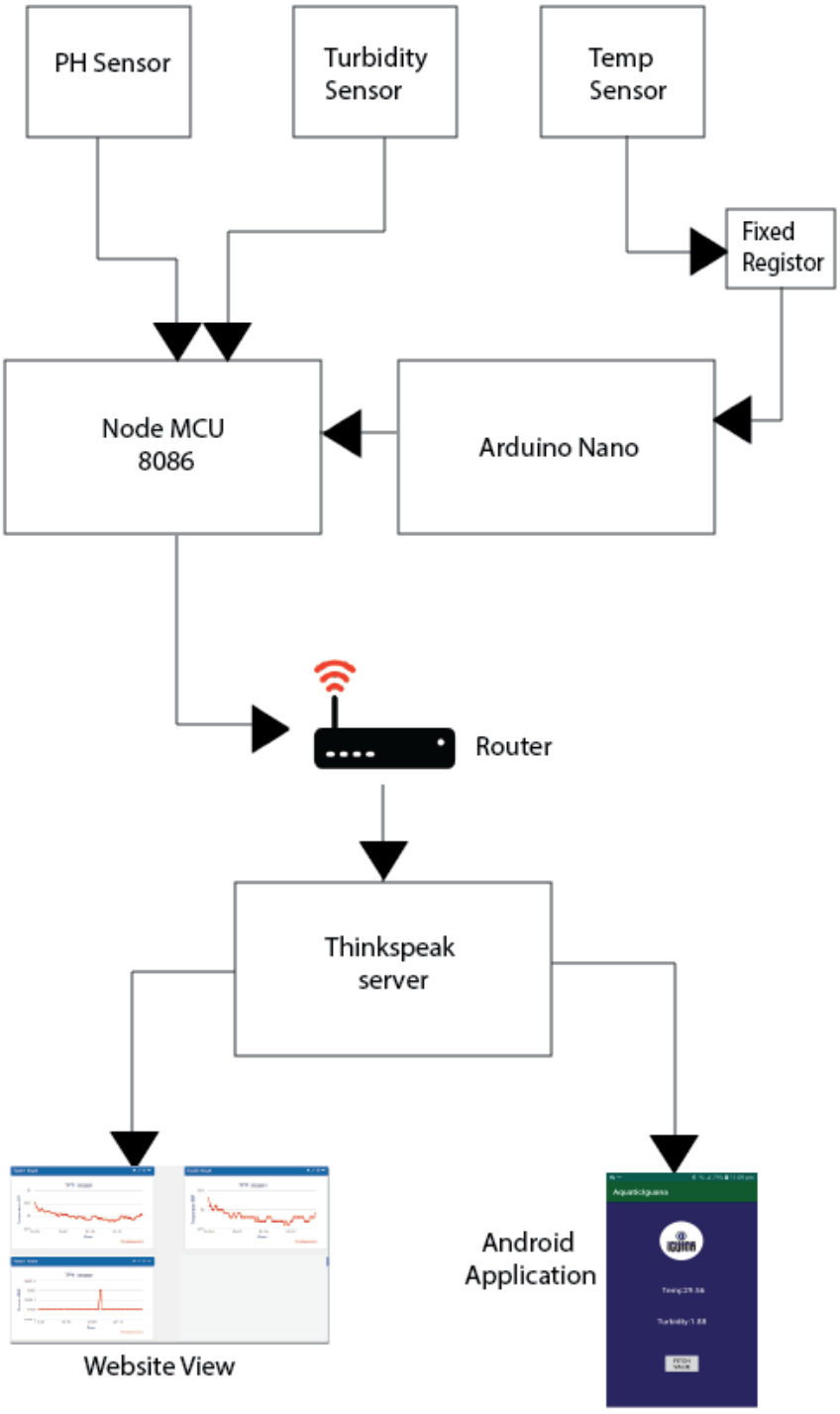

Fig 3.5: water quality monitoring system diagram

3.4 Description of subsystem:

As the project is a combination of three different sections, so this section is divided into three major sections, to briefly talk about the project.

- **Controlling**

  This section will discuss about the controlling mechanism, and will consist of a brief description of the components used including their mechanism and why we have used that.

- **Sensor**

  This sections will also have its component described and will also discuss some important issue about them.

- **Live streaming**

  This section will discuss about Raspberry pi and how we have used it for live streaming as well as it will discuss about the doors it had opened for us in further upgrade.

Now each section will be described.

3.4.1 Controlling

This section includes the controlling system of our project including the shaft control. To power up the controlling system we have used two 2200mah lipo batteries and eight [AAA] pencil sized batteries. The lipo batteries are used to power up the controlling unit and two 12V DC motors, and the AAA batteries are used to power up the transmitter. The lipo batteries are rechargeable.

3.4.1.1 Movement Controlling

Our designed system is a water drone and it will move along the rotation of the wheels. Two wheels are being used and those are driven by two DC motors. Two wheels as well as DC motors are fixed with the chassis of the drone. Two custom made steel wheels are attached to the motors to effect the rotation on the water surface. The wheels are also detachable to any kind of repair.

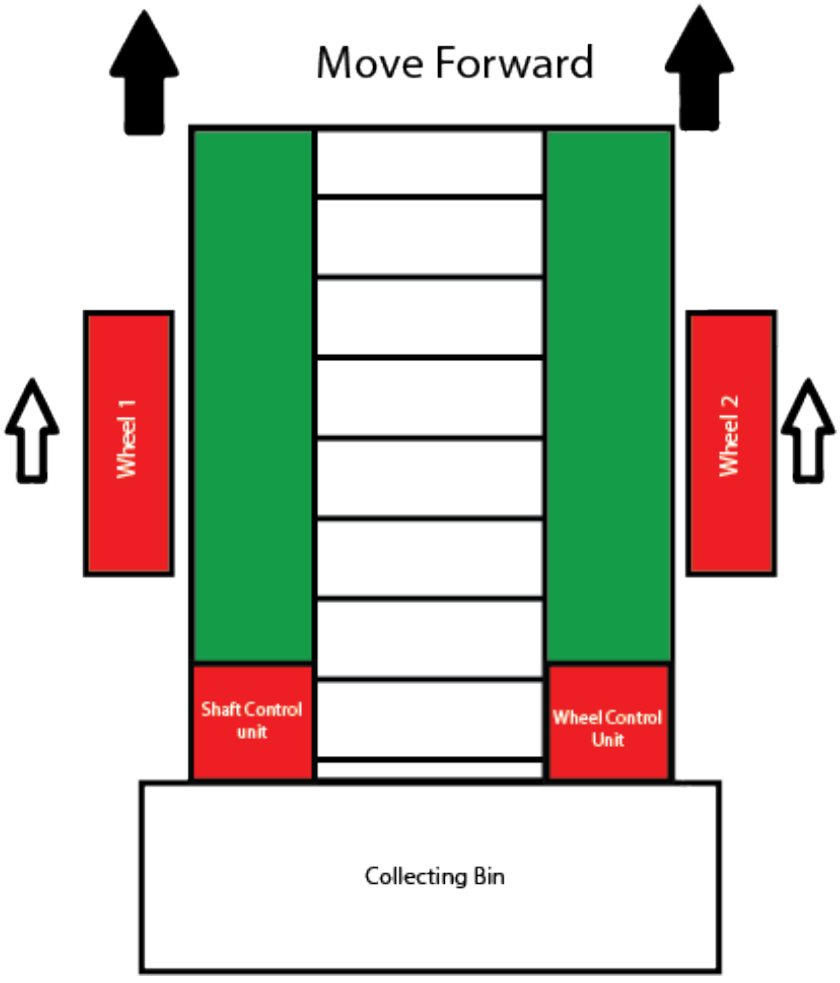

Fig 3.6: Strategy to move forward.

These DC motors can be rotated in both clockwise and anticlockwise direction. There is a motor on the right side of the drone and other one is on the left side. To move forward the motor on the left side are rotated in counter clockwise direction and the motor on the right side is rotated in

clockwise direction. As a result, all the motors will be rotated forward direction in perspective of the drone and the drone will move forward.

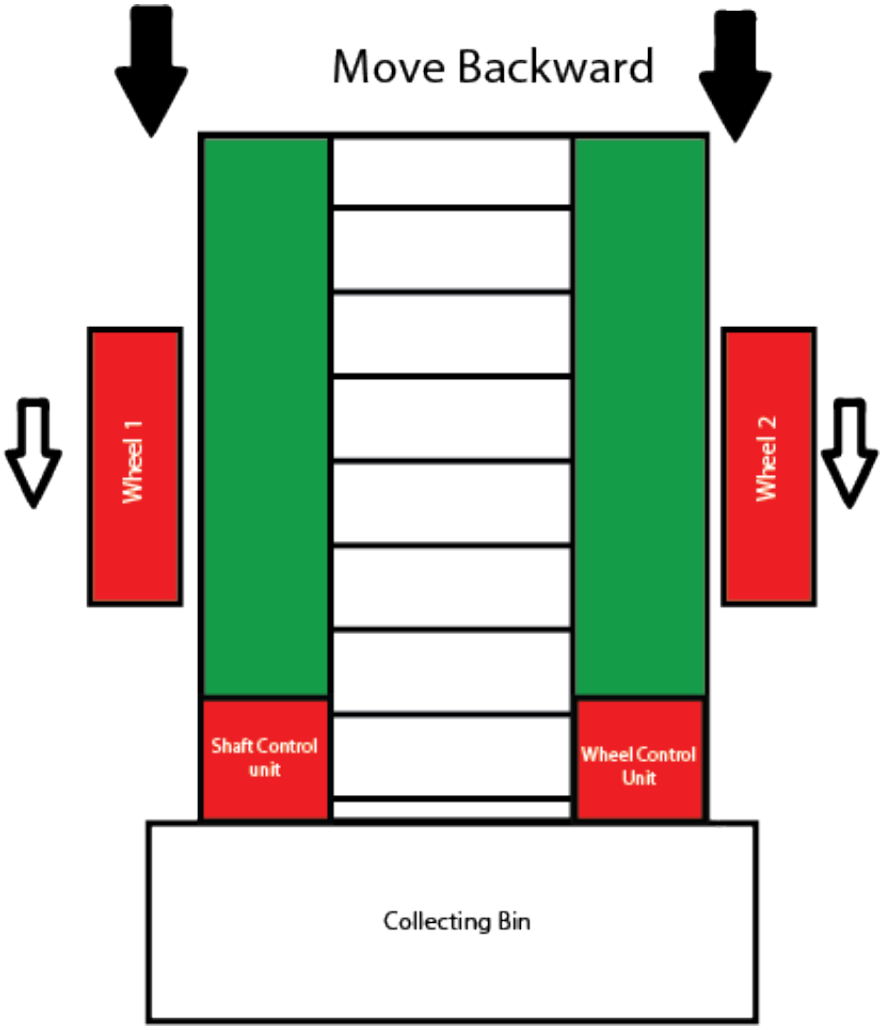

Fig 3.7:. Strategy to move backward.

To move backward the motor on the left side is rotated in clockwise direction and the motor on the right side is rotated in counter clockwise direction. As a result, all the motors will be rotated backward direction in perspective of the rover and the rover will move backward.

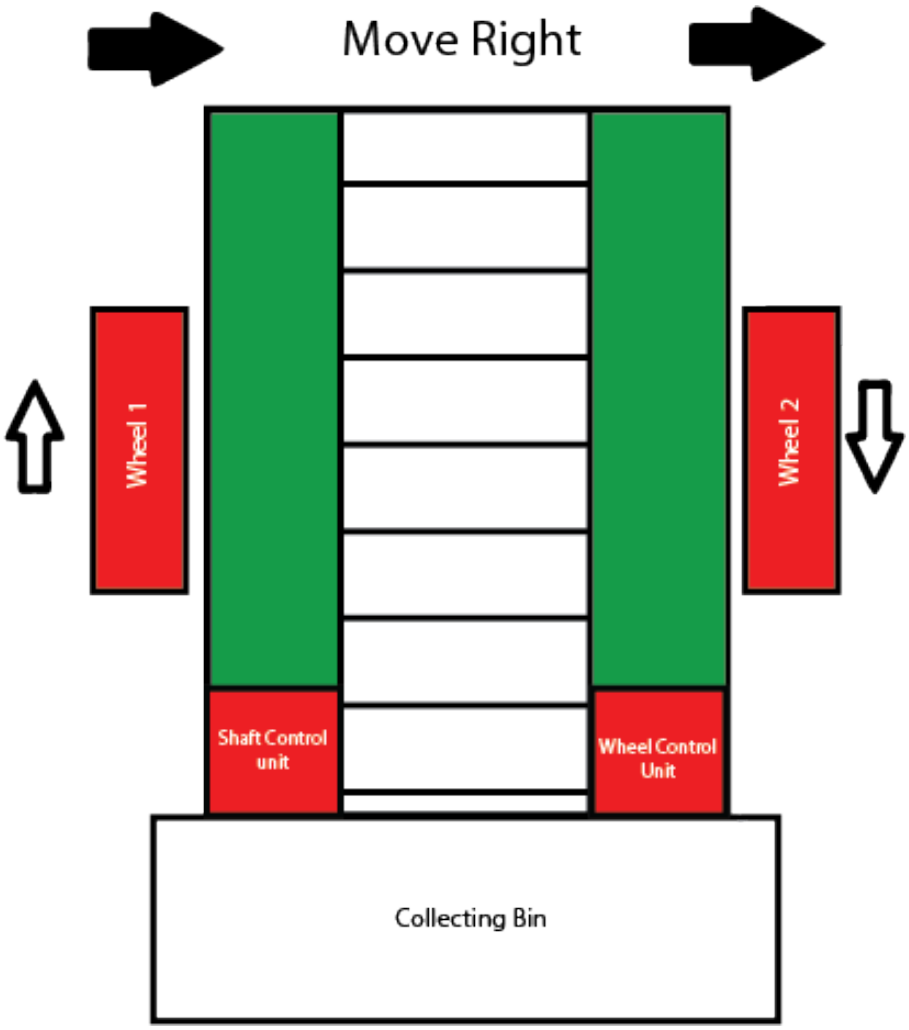

Fig 3.8: Strategy to turn right.

When all the motors run in counter clockwise direction the motor on the right side run in backward direction and other one on the left side run in forward direction. As a result, the drone tries to rotate in counter clockwise direction as we see this from the top view of the drone. This actually causes the drone to turn right.

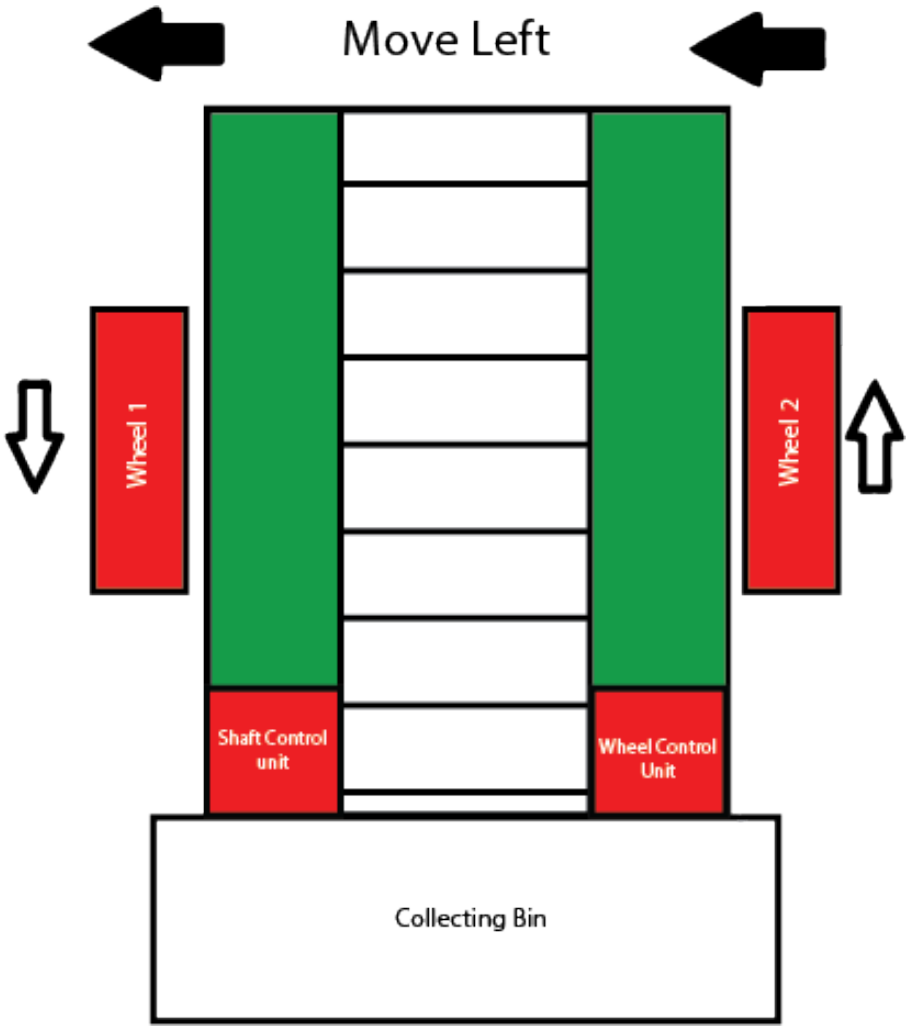

Fig 3.9: Strategy to turn left.

When all the motors run in clockwise direction the motor on the right side run in forward direction and other one on the left side run in backward direction. As a result, the drone tries to rotate in clockwise direction as we see this from the top view of the drone. This actually causes the drone to turn left.

Now let's get to know about the components used for this in brief and then we will discuss about their mechanism.

- **Arduino**

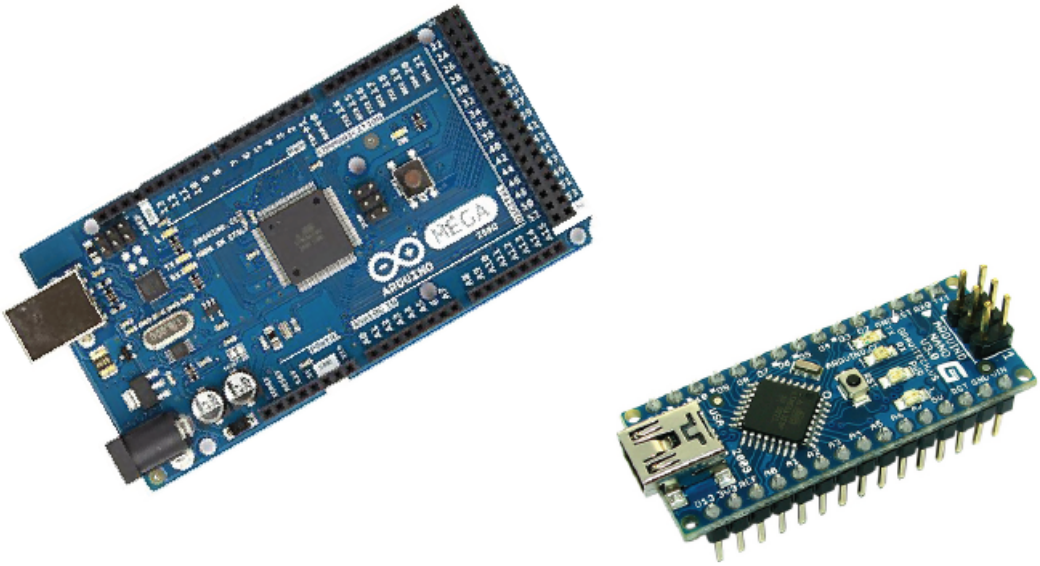

Fig 3.10: Arduino

Arduino is a microcontroller which can be used to control little component, in our case we used it for controlling relay module and sensor data calculation. A microcontroller is a basically a little PC (SoC) on a solitary coordinated circuit containing a processor center, memory, and programmable information/yield peripherals. Numerous installed frameworks need to peruse sensors that deliver simple signs. This is the reason for the simple to-computerized converter (ADC).Arduino is an open-source PC equipment and programming organization, task and client group that outlines and produces microcontroller-based units for building advanced gadgets and intuitive articles that can sense and control objects in the physical world. In our project we have used it to check the frequency the receiver is receiving so that the controlling can be done.

- **FS-CT6B 6CH Remote Control**

Fig 3.11: Radio Controller

FS-CT6B 6CH is a 6 channel wireless controller it includes a receiver in its package and is bind with it, so receiver will only communicate with one single transmitter. To power up the transmitter eight AAA sized batteries are needed. Using this radio controller the wireless connection can be done through 6 different channels and the channels can communicate all at once as well but it is recommended not to use the entire channels all at once as it builds a lot of pressure on the transmitter circuit board. The transmitter and the receiver have the range of 2.4GHz. To know about the channels at first, software needs to be installed which was proved in a CD inside the package. After installing the

software we connected the transmitter to the PC and found out that we need to use channel 2, 4 and 6 in our project. The receiver has 6 different ports for the transmitter to communicate wirelessly from which we took the 3 channels we needed. To communicate the receiver receives the frequency thrown from the transmitter through its channels, putting a condition over which we can do different things.

- **Relay**

As we have used motors which takes 12V to power up, we used relays for switching purpose. A relay is an electrically operated switch. Usually relays use an electromagnet to mechanically trigger a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuitry by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits as amplifiers. Relays were used extensively in telephone exchanges and early computers to perform logical operations. [12]

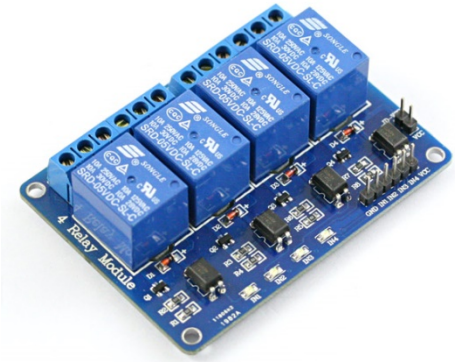

Fig 3.12: 4 channel relay module

Relays are switches that open and close circuits electromechanically or electronically. Relays control one electrical circuit by opening and closing contacts in another circuit. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized. When a relay contact is Normally Closed (NC), there is a closed contact when the relay is not energized. In either case, applying electrical current to the contacts will change their state. Relays are generally used to switch smaller currents in a control circuit and do not usually control power consuming devices except for small motors and Solenoids that draw low amps. Nonetheless, relays can "control" larger voltages and amperes by having an amplifying effect because a small voltage applied to a relays coil can result in a large voltage being switched by the contacts. Protective relays can prevent equipment damage by detecting electrical abnormalities, including overcurrent, undercurrent, overloads and reverse currents. In addition, relays are also widely used to switch starting coils, heating elements, pilot lights and audible alarms. [13]

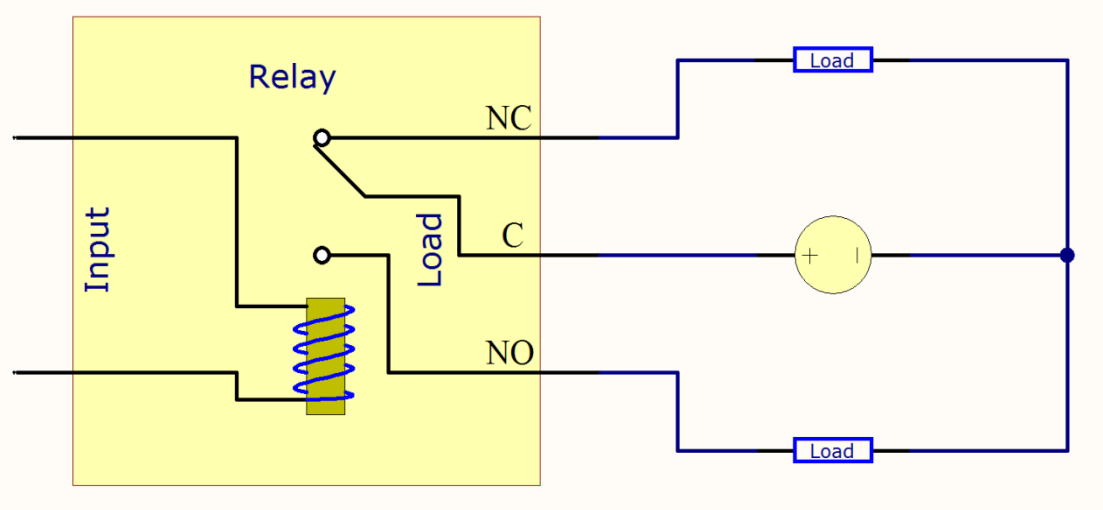

Fig. 3.13. Internal diagram of a relay.

- **DC Motor**

There are different types to DC motors some needs low voltage to power up where as some needs high voltage source to power up. Choosing motors for any project mainly depends on two factors; one is how much power is available to power up the motor and how much weight it needs to carry. In our project the motors had to take a huge amount of weight, to that's why we selected the motors accordingly. So we have used glass motors which can carry up to 20kgs each by calculating their torque.

Fig 3.14: motor

A DC motor consist basically of two parts, the stationary body of the motor called the "Stator" and the inner part which rotates producing the movement called the "Rotor". For D.C. machines the rotor is commonly termed the "Armature".

Generally in small light duty DC motors the stator consists of a pair of fixed permanent magnets producing a uniform and stationary magnetic flux inside the motor giving these types of motors their name of "permanent-magnet direct-current" (PMDC) motors.

The motors armature consists of individual electrical coils connected together in a circular configuration around its metallic body producing a North-Pole then a South-Pole then a North-Pole etc, type of field system configuration.

The current flowing within these rotor coils producing the necessary electromagnetic field. The circular magnetic field produced by the armatures windings produces both north and south poles around the armature which are repelled or attracted by the stator's permanent magnets producing a rotational movement around the motors central axis as shown.

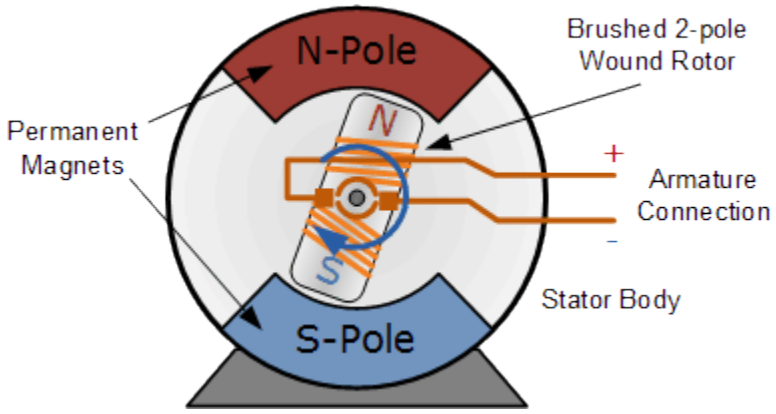

Fig. 3.15. Working principle of a 2 pole DC motor.

The direction of the force of a DC motor can be calculated from Fleming's left-hand rule and its magnitude is given by

$$F = BIL.$$

Here, **B = magnetic flux density,**
I = current and
L = length of the conductor within the magnetic field.

Direction of a DC motor rotation can be changed by changing the current flow direction.

3.4.1.2 Shaft Control

The shaft to collect the waste is controlled through another channel of the transmitter, so that it can be turned off when necessary. The condition was if the receiver receives frequency more the 1300 through the channel from the transmitter it will push a signal towards a 2 channel relay to turn on the shaft in one direction, when the frequency gets lower than 1100 it will push a signal to turn off.

3.4.2 Water Quality Monitoring System

Aquatic Iguana also consists of a water quality monitoring system. While collecting waste materials the drone is also able to collect data of different sensors and send it to the ThingSpeak server. The sensors and equipment which has been used are

- pH Sensor
- Turbidity Sensor
- Temperature Sensor
- NodeMCU, Arduino Nano
- ThingSpeak Server

3.4.2.1 pH Sensor

PH is a measure of acidity or alkalinity of a solution, the pH scale ranges from 0 to 14. The pH indicates the concentration of hydrogen [H] + ions present in certain solutions. It can accurately be quantified by a sensor that measures the potential difference between two electrodes: a

reference electrode (silver / silver chloride) and a glass electrode that is sensitive to hydrogen ion. This is what form the probe. We also have to use an electronic circuit to condition the signal appropriately and we can use this sensor with a micro-controller, such as Arduino.

Fig. 3.16. pH Sensor

As we can see that there are two potentiometers in the circuit. Which it is closer to the BNC connector of the probe is the offset regulation, the other is the pH limit.

Offset: The average range of the probe oscillates between negative and positive values. The 0 represents a pH of 7.0. In order to be able to use it with Arduino this circuit adds an offset value to the value measured by the probe, so the ADC will only have to take samples of positive voltage values. Therefore, we will force a pH of 7.0 by disconnecting the probe from the circuit and short-circuiting the inside of the BNC connector with the outside. With a multi-meter we measured the value of Po pin and adjust the potentiometer to be 2.5V.

pH limit: The potentiometer on the probe is there to set a limit value of the pH sensor circuit that causes the LED signal to light up and the D0 pin signal to turn ON.

In addition, we have to calculate the voltage conversion that will give us the pH sensor so we will need two pH reference value and measure the voltage returned by the sensor on the pin Po. The best thing to do is to use a calibration solution in powders, there are also in liquid but it is easier to preserve the powders. These solutions are sold in different values but the most common are pH 4.01, pH 6.86 and pH 9.18.

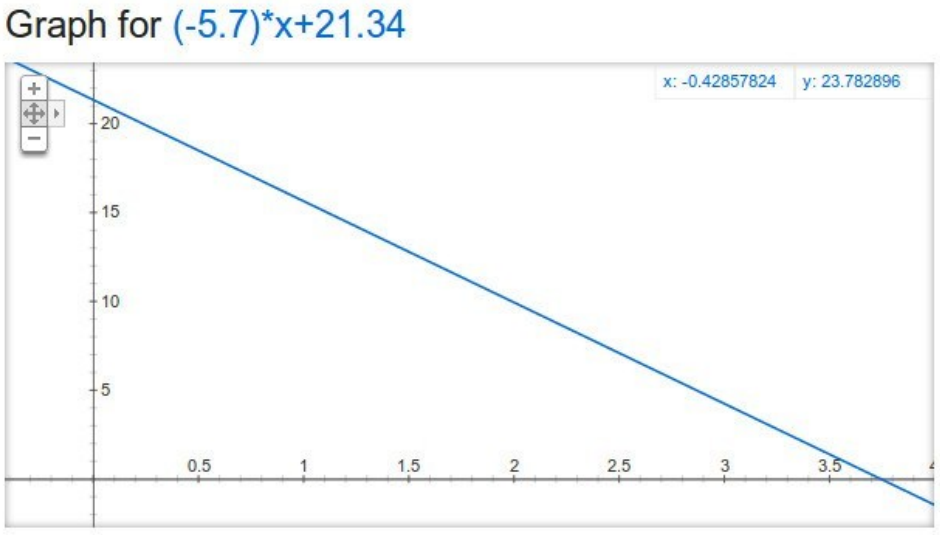

Fig. 3.17. pH Sensor calibration graph

Using the powders with pH 4.01 and pH 6.86 we obtain the voltages on the pin Po 3.04V and 2.54V respectively. The sensor is linear so by taking two points we can deduce the equation to convert the measured voltage to pH. The general formula would be y = mx + b, so we have to calculate m and b since x would be the voltage and y the pH. The result is y = -5.70x + 21.34 [14].

3.4.2.2 Turbidity Sensor:

The Arduino turbidity sensor detects water quality by measuring level of turbidity. It is able to detect suspended particles in water by measuring the light transmittance and scattering rate which changes with the amount of total suspended solids (TSS) in water. As the TTS increases, the liquid turbidity level increases.

This Arduino turbidity sensor have both analog and digital signal output modes. One can select the mode according to the MCU as threshold is adjustable in digital signal mode. Turbidity sensors can be used in measurement of water quality in rivers and streams, wastewater and effluent measurements, sediment transport research and laboratory measurements [15].

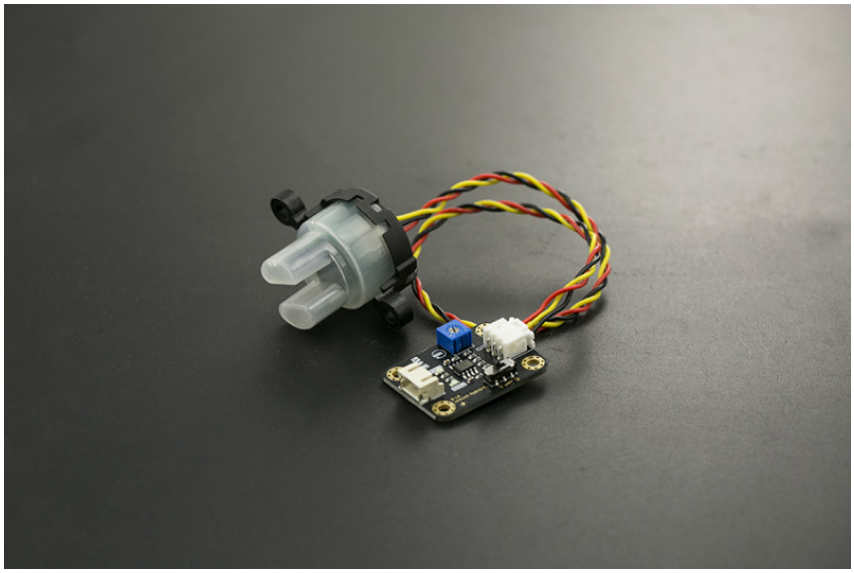

Fig. 3.18. Turbidity Sensor.

3.4.2.3 Temperature Sensor

The temperature of water will be sensed by temperature sensor. Its resistance is linearly dependent on the temperature. So increase in the water temperature changes its resistance and the microcontroller senses the changing in resistance. The temperature sensor sends data to the digital pin on arduino. For the purpose of water quality monitoring system we have used DS12B80 waterproof digital temperature sensor.

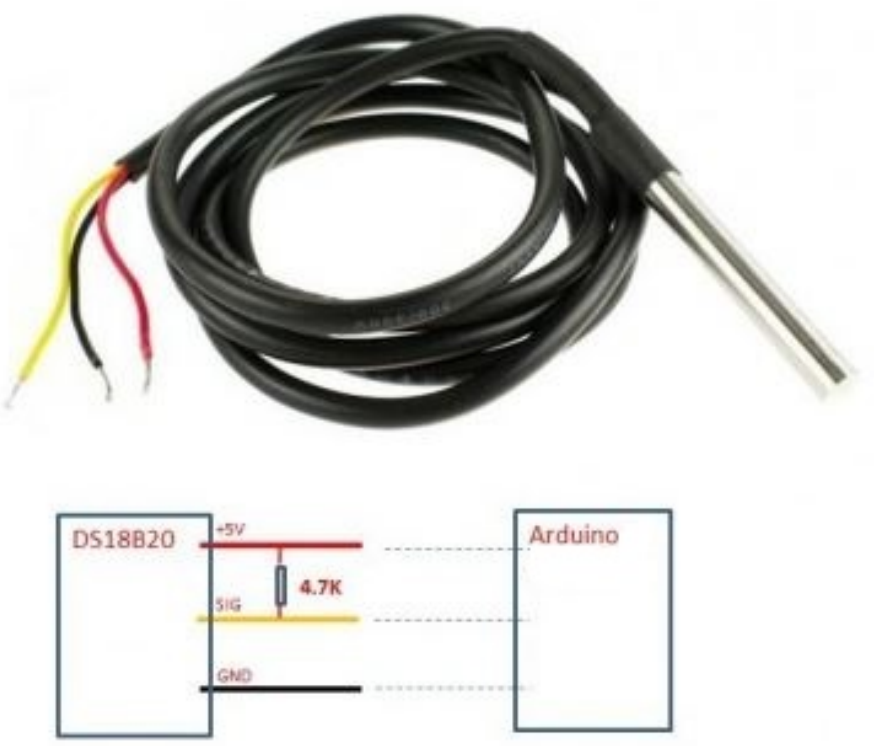

Fig. 3.19. DS12B80 waterproof temperature sensor [16].

3.4.2.4 NodeMCU ESP8266

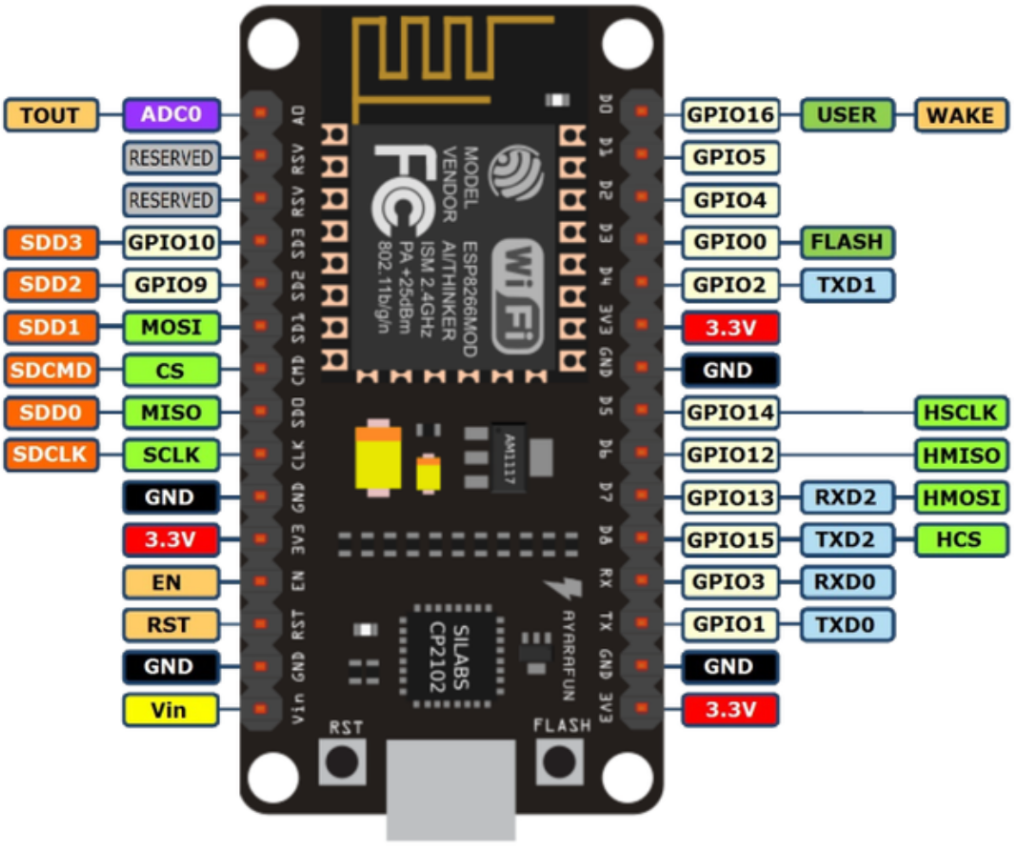

Fig. 3.20. NodeMCU ESP8266 wifi module

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS [17].

3.4.2.5 Arduino Nano

Arduino Nano is a small, compatible, flexible and breadboard friendly Microcontroller board, developed by Arduino.cc in Italy, based on ATmega328p (Arduino Nano V3.x) / Atmega168 (Arduino Nano V3.x).

It comes with exactly the same functionality as in Arduino UNO but quite in small size.

It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V.

Arduino Nano Pinout contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.

.

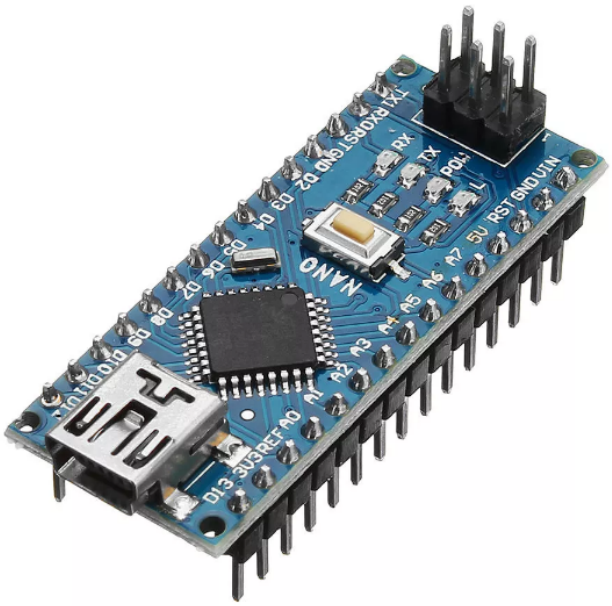

Fig. 3.21. Arduino Nano

Each of these Digital & Analog Pins are assigned with multiple functions but their main function is to be configured as input or output.

They are acted as input pins when they are interfaced with sensors, but if you are driving some load then use them as output. Functions like pinMode() and digitalWrite() are used to controlling the operations of digital pins while analogRead() is used to control analog pins.

The analog pins come with a total resolution of 10bits which measure the value from zero to 5V. Arduino Nano comes with a crystal oscillator of frequency 16 MHz. It is used to produce clock of precise frequency using constant voltage. This board doesn't use standard USB for connection with a computer, instead, it comes with Mini USB support. Tiny size and breadboard friendly nature make this device an ideal choice for most of the applications where a size of the electronic components are of great concern [18].

3.4.2.6 ThingSpeak

ThingSpeak is an IoT analytics platform service that allows one to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB code in ThingSpeak one can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

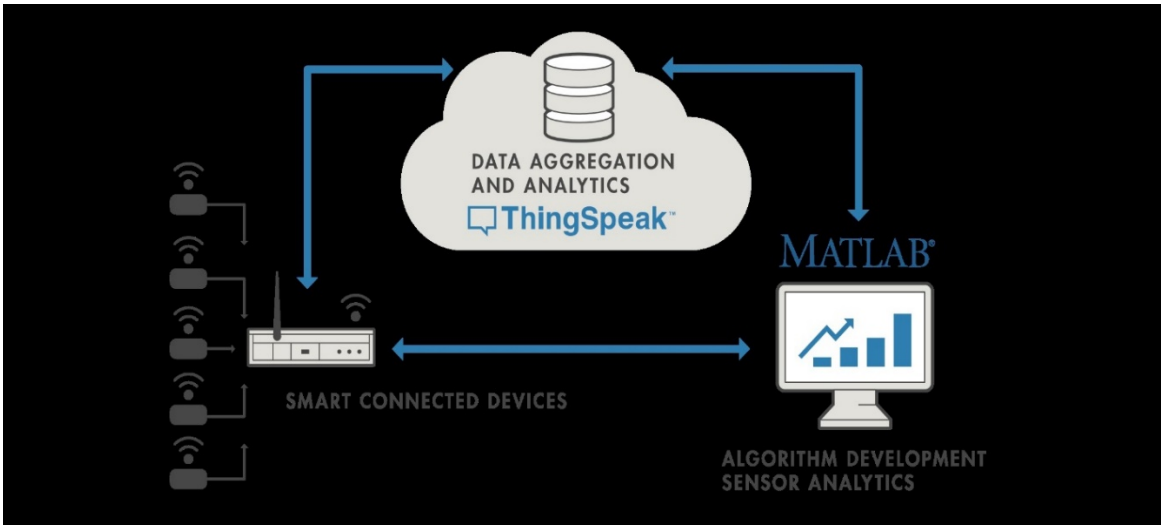

Fig. 3.22. ThingSpeak

ThingSpeak allows one to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of IoT data.
- Run IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software [19].

3.4.2.7 Water Quality Monitoring system working procedure

For the water quality monitoring system, we have used three sensors and they are pH sensor, temperature sensor and turbidity sensor. We have used NodeMCU and Arduino Nano to collect data from the sensors. NodeMCU sends those data to ThingSpeak server.

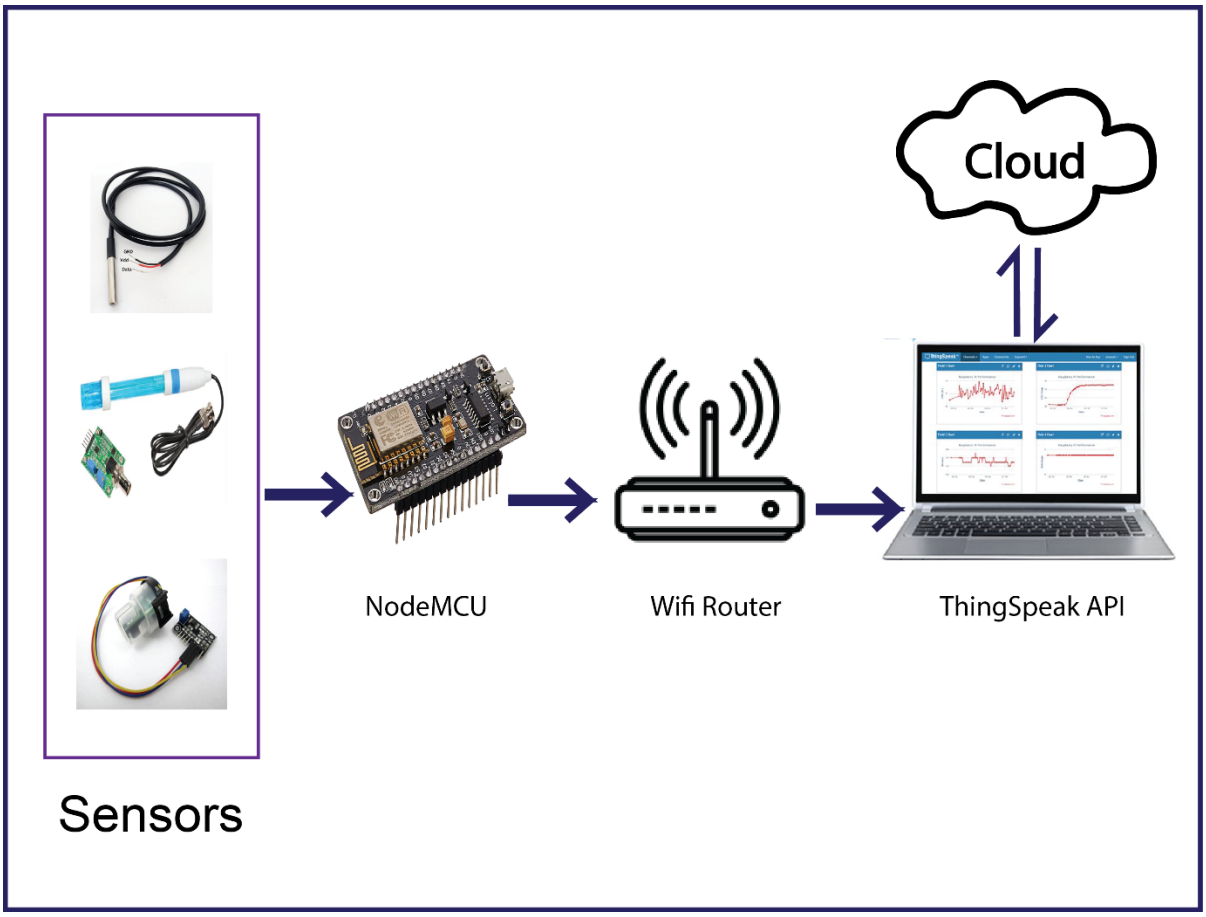

Fig. 3.23. Workflow of water quality monitoring system

Every 15 seconds data is sent to the ThigSpeak server and inserted into the database. One can see the visual representation of data in real-time. Inserted data can be accessed via csv file format which can be used later on for various purposes.

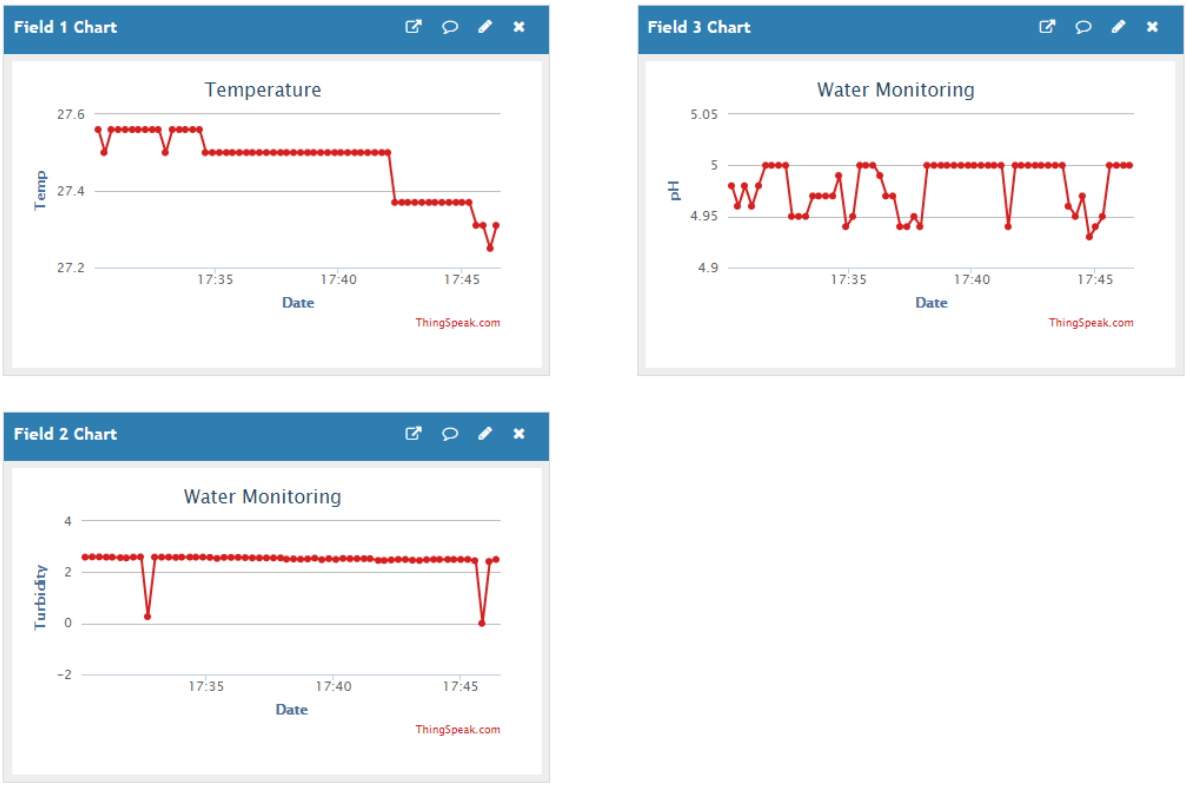

Fig. 3.24. Visual representation of stored data

On the above picture there are data of three sensors. This specific representation provides the data of HatirJheel.

3.4.3 Live Streaming

Live streaming is a continuous procedure of capturing frames and sending it to the host server. After sending it to the host server we had to fetch the frames and show it continuously to make it streaming. The smoothness of the live streaming depends on three factors, one is the FPS of the camera module, and second one is the sending net speed and lastly the internet speed of fetching. For our project we took the camera module of raspberry pi which had given us a fair amount of frames per second which was

approximately 3 FPS. And we had kept the server in our localhost to maintain uninterrupted and fast connection. Now let's get to know about the modules.

3.4.3.1 Raspberry Pi 3 B+

Raspberry Pi is a series of single board computer which is developed by Raspberry Pi Foundation; it promotes the teaching of basic computer science. It is really a handy equipment to have for robotics. It can help us to process and do many large implementations in smaller scale. Raspberry Pi 3 B+ is a very powerful tool which runs with 64-bit quad core processor running at 1.4 GHz.

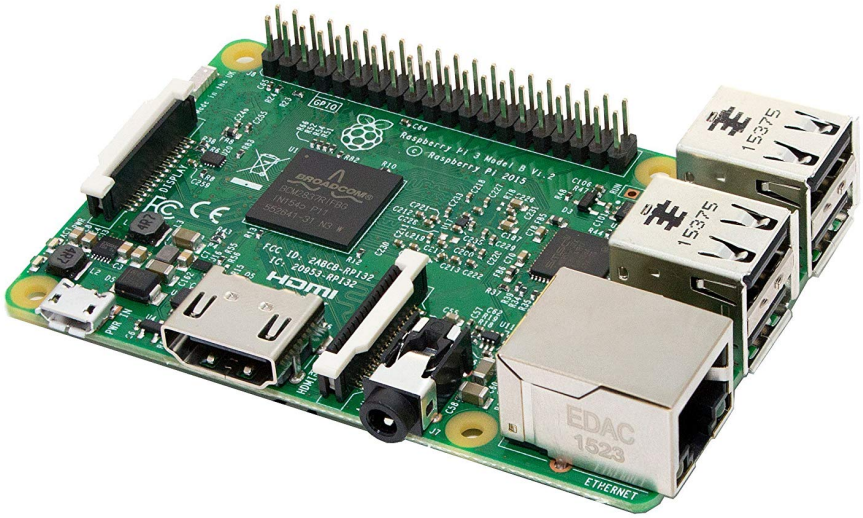

Fig 3.25: Raspberry Pi B+

In our project the high speed WLAN connection of the Raspberry Pi 3 B+ came in very handy which works in 5GHz speed.

3.4.3.2 Raspberry Pi Camera Module

Raspberry Pi camera module has various types depending on their mega pixels, in our project we had used the version of 5 mega pixel. The v2 Camera Module has a Sony IMX219 8-megapixel sensor. The Camera Module can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion, and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

Fig 3.26: raspberry pi camera module

We have used this camera module to continuously capture frame by frame.

3.4.4 Android Application

In our project we have also created an android application to make it easier for the user and have real time information. Our application starts with a splash screen which holds for 5 seconds and checks if the internet connection is available or not, if the internet connection is not available then the application will ask the user to turn on the internet.

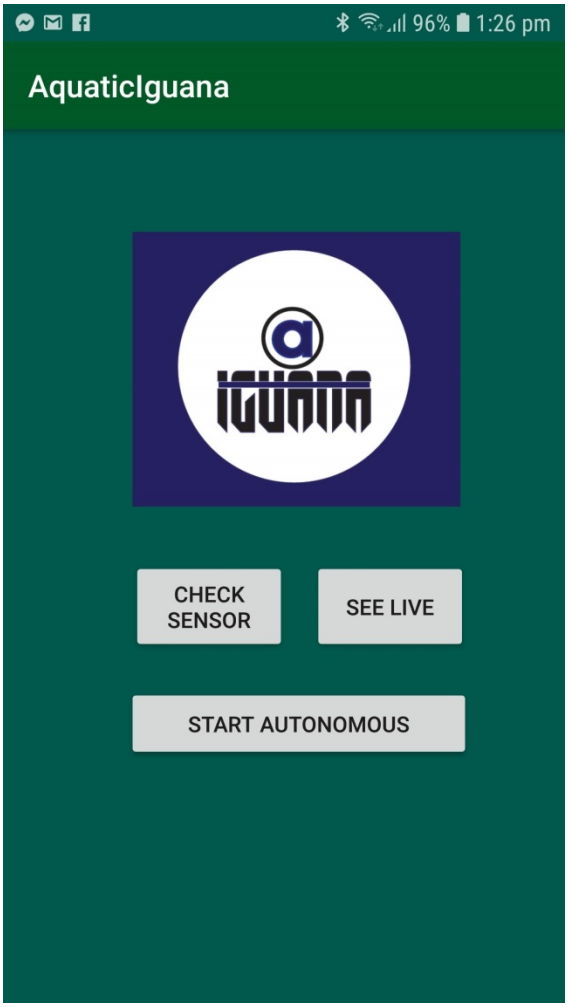

Fig 3.26: mobile application

After passing the splash screen the user will have two options to choose from one is to check the sensor value and the other one is to see the live streaming. When the user wants to check the sensor value he/she needs to just get inside the “Check Sensor” option and press a button to fetch the data. After pressing that the user will get the real time sensor value on their mobile phone, and also if the user wants to see the live streaming the user needs to select the “See Live” option and instantly the live streaming will be started on their mobile phone.

3.5 Summary

This chapter provides overall technical implementation of Aquatic Iguana. The control system is done using Radio Controller and Raspberry Pi based camera provides visual of the surroundings to the operator. This chapter also talks about the water quality monitoring system and its working procedure

Chapter 4

Design Implementation

4.1 Introduction

In this chapter the brief description of the whole design will be given. We will also be discussing about some important procedure we had to go through, also we will be talking about the software implementation and briefly discuss about the software. The internal circuitry will be explained in this chapter as well. We will be showing a work flow and talk about every little details about the implementation.

4.2 List of necessary hardware

At first we had to design the drone by 5mm PVC board and do some trial and error to come up with a finalized version of the actual chassis. As our drone was supposed to move around on a water surface we had to come up with a solution to make it float on the water, and also had to keep in mind to maintain the balancing after collecting the waste as well. Because more weigh on the back of the drone will make it tilt and drown. After the chassis was finalized and built then we went for the circuitry.

4.2.1 Required tools for design implementation

1. Soldering Iron

2. Wire cutter

3. Drill machine

4. Screw and bolts

5. Glue gun

6. Lathe

7. Digital Multi meter

8. 3mm wire

9. PVC fabric

10. 3 mm Steel sheets

11. PVC board

4.2.2 Required material and component for internal circuitry

1. Arduino mega 2560

2. Arduino nano

3. Node MCU 8086

4. Ph sensor

5. Turbidity sensor

6. Temperature sensor

7. 4.7k Ohm resistor

8. 40pin breadboard

9. Raspberry Pi 3 B+

10. Raspberry Pi camera module

11.1000 mAh powerbank

12. 5V 4 Channel relay

13. 5V 2 Channel relay

14. Transmitter and receiver

15. 12V 2200 mAh lipo battery

16. Jumper wire

17. 4 DC motors

18. Router

4.3 Description of the software

The drone is a microcontroller based project including some IOT based implementation as well. So some IDE(Integrated Development Environment) is needed to burn the program to the microcontroller as well as to the Node MCU, for both the purpose we have used the arduino IDE which is an open software provided by the arduino company itself. To create our android application we have used “Android studio” which is also free. And for the python script to run the live streaming we had used IDLE editor, which is given by the Raspbian operating system.

4.3.1 Programming Software

The electronic circuitry of the drone is based on microcontrollers. An AVR microcontroller from ATMEL has been used. The name of the microcontroller is ATMEGA 2560. Arduino microcontroller platform of ATMEGA has been used. In this case Arduino Mega 2560 board has

been used to implement the design. And for the water quality monitoring system Arduino nano was used which has a ATmega328 microcontroller built in to it.

Arduino programs may be written in any programming language with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in Java. It originated from the IDE for the Processing programming language project and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism for compiling and loading programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the C and C++ programming languages using special rules of code organization. The Arduino IDE supplies a software library called "Wiring" from the Wiring project, which provides many common input and output procedures. [26]

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. [27]

- The advantages of Arduino IDE have been discussed below.

- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money. [28]

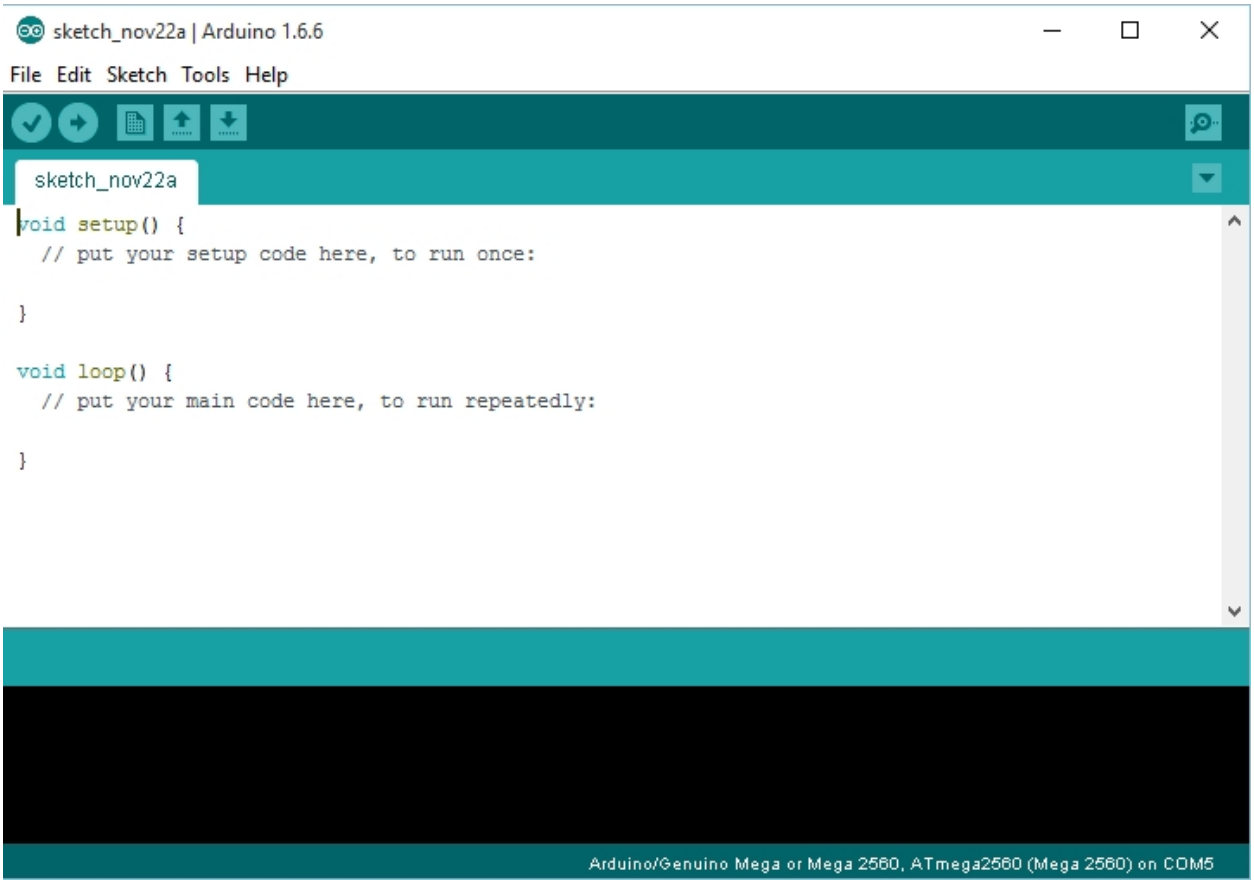

Fig. 4.1. Arduino IDE window.

Besides working with Arduino, a program was also written for Atmega32 microcontroller IC. But this was not done for hardware implementation purpose. This was done only for simulation purpose. The program was developed into ATMEL's own IDE called ATMEL STUDIO. Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® | SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel | SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits.[29]

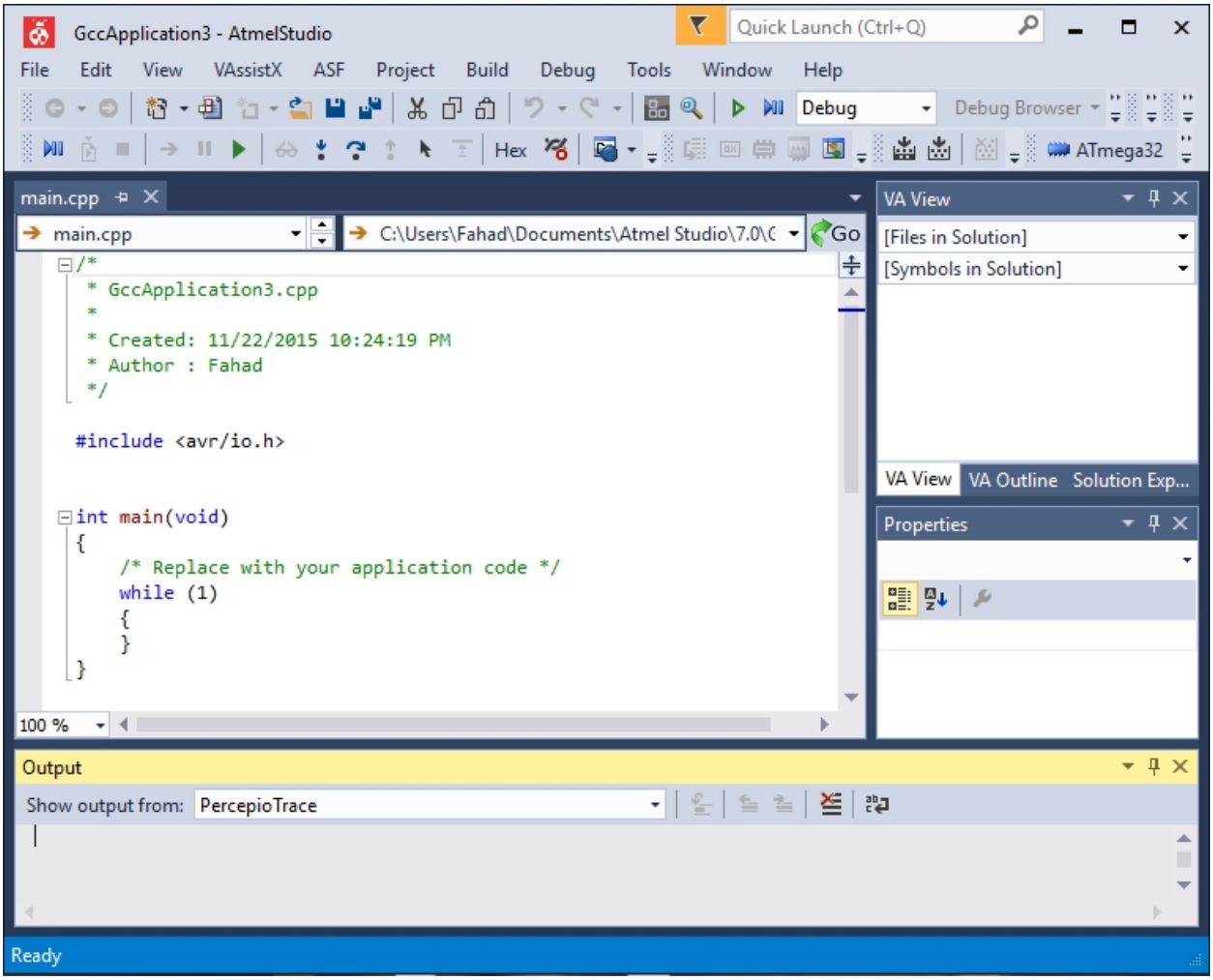

Fig. 4.2. Atmel Studio IDE

4.3.2 IDLE Editor

IDLE is Python's integrated development and learning environment. It was fully coded in python using the tkinter GUI toolkit. This editor works with most of the operating system, and comes built in with the raspbian operating system. IDLE has two main window types, the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously. On

Windows and Linux, each has its own top menu. Each menu documented below indicates which window type it is associated with.

Output windows, such as used for Edit => Find in Files, are a subtype of editor window. They currently have the same top menu but a different default title and context menu.

On macOS, there is one application menu. It dynamically changes according to the window currently selected. It has an IDLE menu, and some entries described below are moved around to conform to Apple guidlines.

4.3.3 Android Studio

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system
- Build variants and multiple apk file generation
- Code templates to help you build common app features
- Rich layout editor with support for drag and drop theme editing
- Lint tools to catch performance, usability, version compatibility, and other problems
- ProGuard and app-signing capabilities

- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- And much more.

4.3.4 Simulation Software

Computer simulation is a process of executing a real world model inside a computer program. It represents the function of the whole system or model. This gives us the capability of analyzing and understanding of how individual element interact and affect the simulated environment.

As simulation provides a way in which alternative designs, plans and policies can be evaluated without having to experiment on a real system, which may be prohibitively costly, time-consuming, or simply impractical to do it is a powerful and important tool for engineering.

In this section, we will discuss on which software and how we successfully simulated the several part of the robotic system.

We used “Proteus” for circuit simulation. Proteus is an application that allows the user to make schematic captures, simulate microprocessors and develop PCB (printed circuit board) designs. Most of the users download this software because its interface simplifies the different tasks. It comprises several modules that are combined to offer distinct services. This software combines ARES PCB layout and ISIS schematic capture to provide a powerful design program

ISIS (Intelligent Schematic Input System) module provides a wide number of features controlled by using a simple user interface. It allows you to design the plane of an electric circuit using many different components, such signal generators, simple resistors, power supplies, a different

microcontroller or microprocessor, and many others. Proteus it makes possible to define many of the aspects of the components appearance and it incorporates a powerful design environment.

The designs can be simulated in real-time through VSM (Virtual System Modeling). It is provided with a mixed-mode SPICE simulation, microprocessor models and animated components to facilitate the simulation to the user. VSM extension is integrated in the ISIS module and allows you to simulate the tasks that you want the program to carry out, being able to handle the microcontroller and its outputs. The simulation can be done with low level and high level code.

Another remarkable module that is integrated in Proteus is ARES (Advanced Routing and Editing Software) is a tool that allows you to route, edit and locate components used to for the production of printed circuit boards. The 3D viewer gives the customer the opportunity of having a view of the board as it would be in the real life. It generally gives you the option of editing the welding and surface layers.

Compared to other similar applications, Proteus provides the user with various tools that work in a simple and intuitively way. Maybe that's why this tool is frequently used in the education field. It is very friendly for novice users who are interested on obtaining high level simulation, schematics and board designs. You can download a free version to try it out.

Another positive feature of Proteus is that the developer provides free material to manage this software. In the official website there are several video demonstrations about Proteus VDM simulation and PCB design. There is also more support information to download and discussion forums that could help the user to utilize the application and share information.

Proteus 8 Features have been discussed below.

- Proteus incorporates a tool that automatically places a component in the netlist into the board.
- Make a basic simulation.
- Proteus has a Schematic capture module, PBC Layout module and a display technology that accelerates hardware.
- It is possible to execute the integrated based router loading custom scripts or using an interactive mode.
- Proteus provides a 3D visualization (VSM) of the board. It includes navigation and a 3D data footprints user application.
- Manage and configure dynamic teardrops.
- Export layouts using ODB++ CAD/CAM format.
- Proteus gives an automatic gate-swap optimization.
- There are unlimited shape-based power planes per each layer.
- Unlimited shape based power planes per layer.
- Unlimited number of pins in a netlist.

4.4 Hardware Implementation

In the hardware implementation we had to go through some trial and error fixing to come up with a finalized design to implement. We had to keep it in mind that the balancing issue may arrive when the waste will be collected. So at first we developed a prototype using PVC board and sticking it by glue and find the accurate shape.

4.4.1 Building Chassis

While building the chassis we had to think and the weight management on both sides. We had to keep in mind to make sure to make it floats upon water, that's where we came up with the idea to make the bottom part hollow so that it can get the push of the water upward the surface and float. We also kept the shaft on the as middle as possible as there will a pressure of the shaft pulling the upward, which will effect on the lower part of the drone. Then we measured the length of the drone and put the wheels exactly on the middle of the drone, because any change would have ruined the weight balancing and the rotation motion. We also tried to keep the motor as above as possible from the water, and wrap it around with cocksheet to water seal the area.

Fig 4.3: Body structure of Aquatic Iguana

4.4.2 Internal circuit for the drone

The whole process of the internal circuit of the drone will be discussed here in three different sectors.

4.4.2.1 Controlling and Shaft

At first 3mm electrical wire was soldered with the motor cutting them by measuring the length from the motor to the relay. Then the relays shorted and given power source according to the image below:

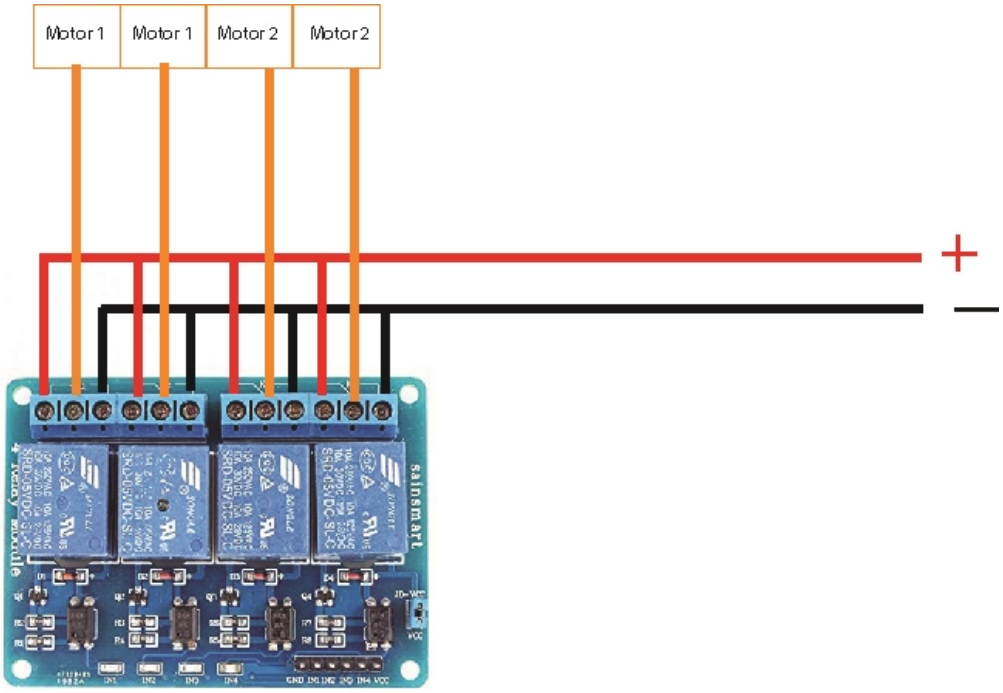

Fig 4.4: relay module connection

Then we started working on the connections from the receiver to the Arduino Mega, but at first we connected the transmitter to the PC and found out the channels we will be using. Then we connected those channels to the Arduino mega and burned a simple code to find the range of value which the receiver is receiving from the transmitter as it can differ from transmitter to transmitter. Then we burned the final code with the condition into the Arduino Mega and placed

it to its place and connected the input pins with the receiver. And got the output pins from the Arduino Mega and connect it accordingly to the code with the relay to push the output signal.

4.4.2.2 Live streaming

The implementation of the live streaming is very much straight forward, we had to first connect the camera module to the raspberry pi 3 B+ and turn on the raspberry pi using the power bank.

Fig 4.5: power bank

And find the IP address to build a SSH connection to a remote computer, after the IP address was found we the established a SSH connection with another computer using a software named 'Putty'. After tasting the connection we wrote the python script which will run the live steaming to the host. Then using the SSH connection we ran the python script and the live streaming had started.

4.5 Software implementation

In this section we mostly discuss about the codes and how do they work. We'll talk about Arduino and NodeMcu codes and also about the Android application.

4.5.1 Programming the Arduino

As a central microcontroller we are using ATMEGA 2560 which is based on an Arduino platform. Arduino uses a C language based programming language called "Wiring". All the programming codes have to be written on the Arduino IDE. There is a text editor in the Arduino IDE. This text editor is used to write the programming codes on it. There are two main parts of a typical Arduino program. One is "void setup()" and the other is "void loop()". "void setup()" is a function that runs once at the start of a program and that can initialize settings. "void loop()" is a function called repeatedly until the board powers off.

To declare a pin of the Arduino as a input "pinMode(pin number, INPUT)" is used. And to declare as a output "pinMode(pin number, OUTPUT)" is used. To get a digital data on an output pin "digitalWrite(pin number)" command is used. If we want to read a value from a sensor which is connected to an analog pin we use "analogRead(pin number)". If we have some conditional situation we can use "if" conditional statement. We have used all these things to write program for the Arduino.

There are many libraries for the Arduino. The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. We have used libraries for Motor shield, Ethernet shield, SPI communication etc.

After writing the whole code on the Arduino IDE editor, we compiled it through the compiler. To do this we select “Sketch” file menu on the Arduino window. Then we select “Verify/compile”. This will compile the code. If there is any error it will show that.

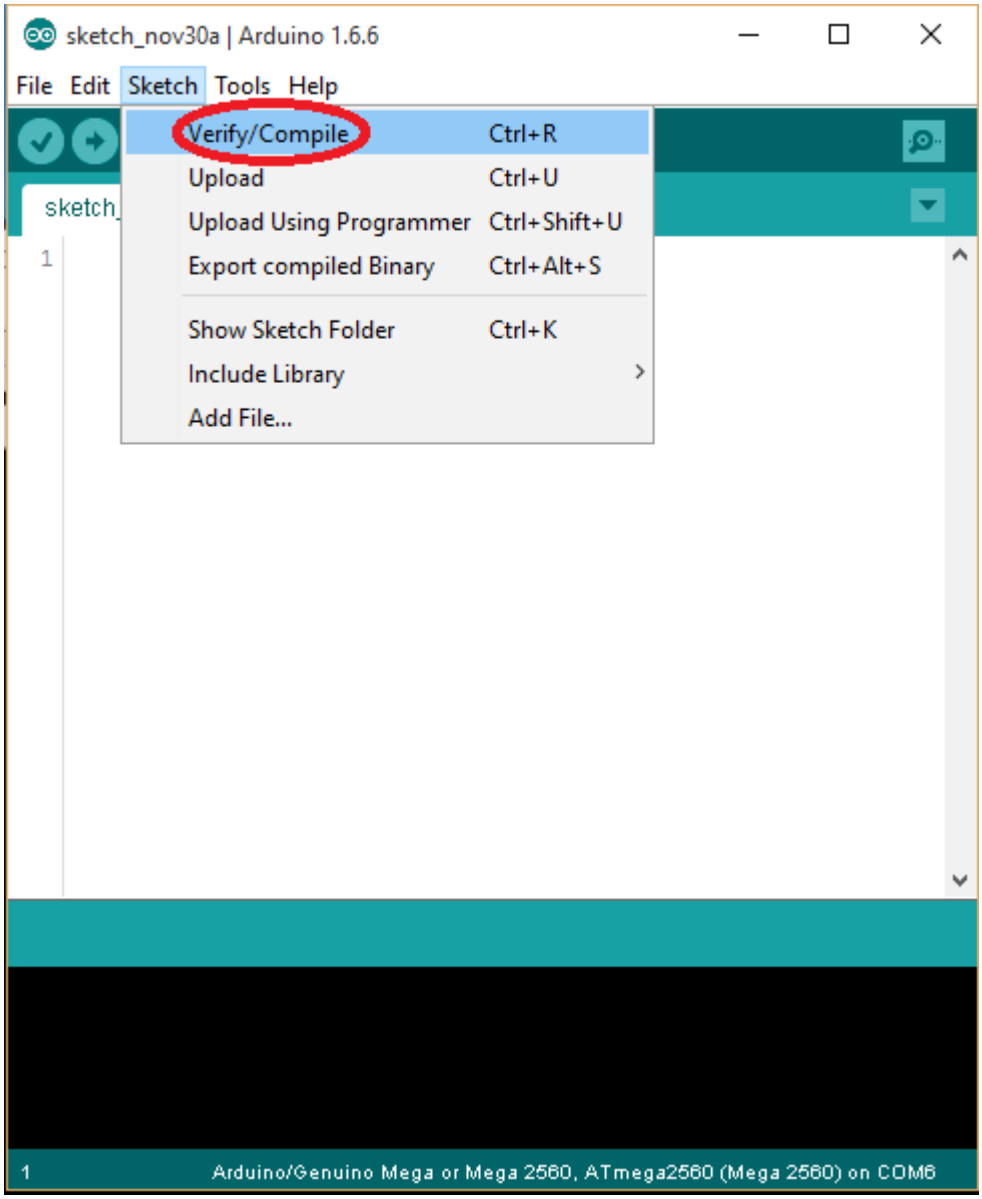

Fig. 4.6. Compiling Arduino code.

Another important thing should be remembered that for which board the codes will be compiled. For our case, it is Arduino Mega 2560. So, we need to select the appropriate board. To do this we go to Tools > Board > Arduino/Genuino Mega or Mega 2560.

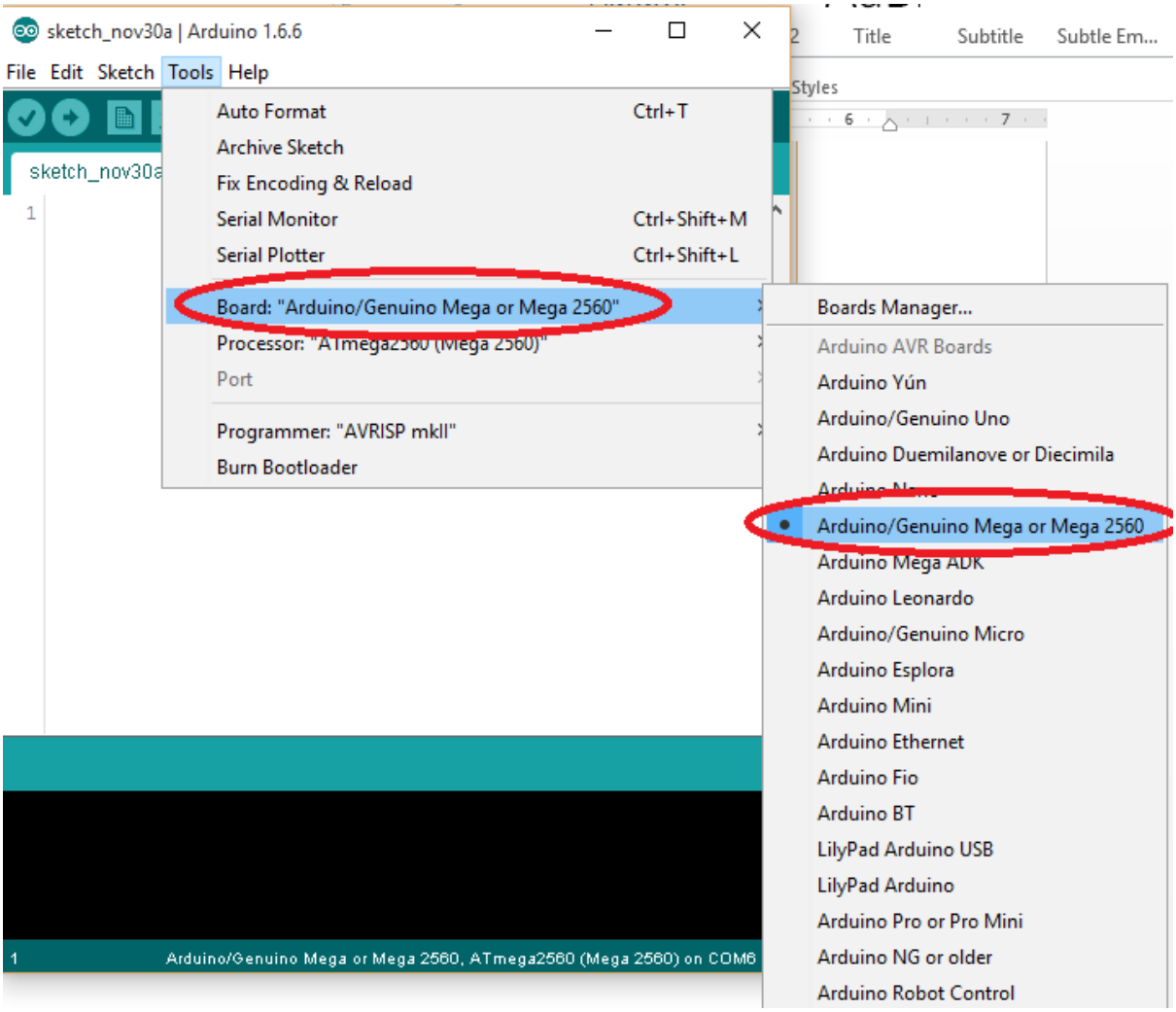

Fig. 4.7. Board selection.

After compiling it we have to upload the program to the Arduino. First we need to connect the Arduino board to the USB port through an USB cable provided with the Arduino. Then we need to select “Upload” at the top left corner on the Arduino IDE. If the program is uploaded

successfully it will show it through a message. Now the central microcontroller is programs. The whole programming code of the rover is in APPENDIX A.

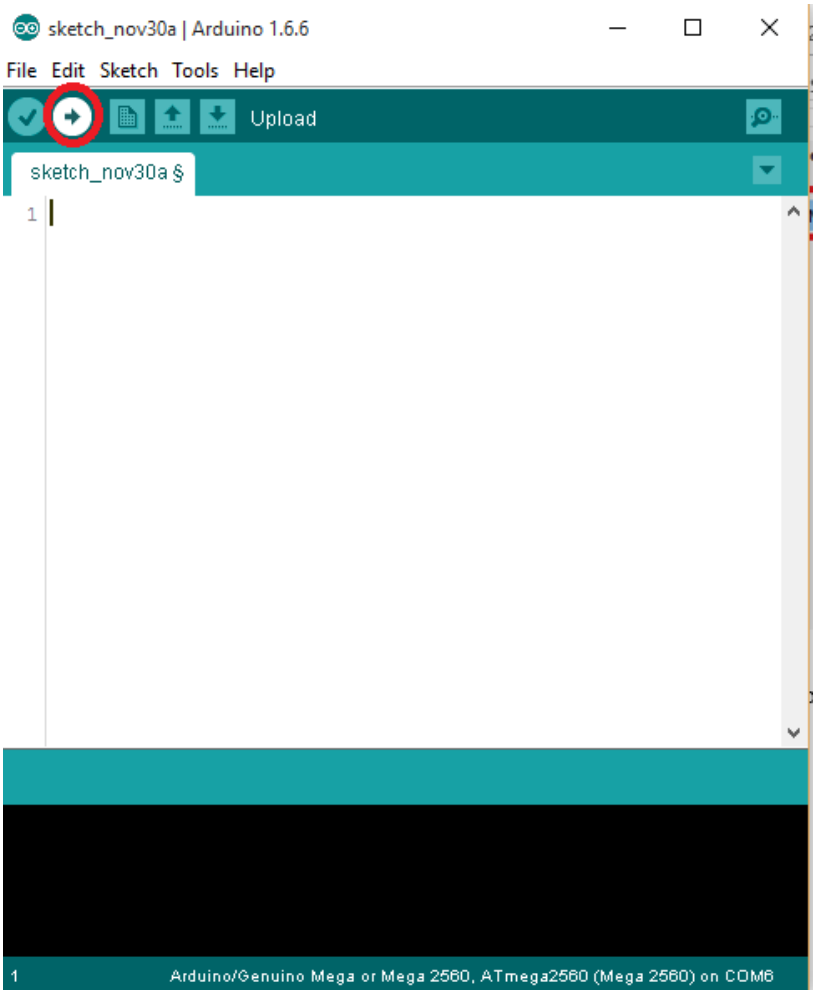

Fig. 4.8 Uploading program to an Arduino.

4.5.2 Programming for the NodeMCU and Arduino

The programming for controlling Aquatic Iguana is given below. For different frequency the drone moves to different direction. For channel frequency between 1300 to 1600 it remains stand still.

```
if ((Channel1 > 1300 && Channel1 < 1600 ) && (Channel2 > 1300 && Channel2 < 1600  ))
 {
          digitalWrite(leftMotF , HIGH);
          digitalWrite(leftMotB , HIGH);
          digitalWrite(rightMotF , HIGH);
          digitalWrite(rightMotB , HIGH);
  }


  if (Channel2 > 1700) // Checks if Channel1 is lesser than 1300
  {
      digitalWrite(leftMotF , LOW);
      digitalWrite(leftMotB , HIGH);
      digitalWrite(rightMotF , LOW);
      digitalWrite(rightMotB , HIGH);
  }
```

Fig. 4.9 Controlling code

For channel 1 value greater than 1700 it moves forward. For channel 1 value less than 1300 it moves backward.

```
if (Channel2 > 1700) // Checks if Channel1 is lesser than 1300
{
    digitalWrite(leftMotF , LOW);
    digitalWrite(leftMotB , HIGH);
    digitalWrite(rightMotF , LOW);
    digitalWrite(rightMotB , HIGH);
}
if (Channel2 < 1300) // Checks if Channel1 is greater than 1500
{
        digitalWrite(leftMotF , HIGH);
        digitalWrite(leftMotB , LOW);
        digitalWrite(rightMotF , HIGH);
        digitalWrite(rightMotB , LOW);
}
}
```

Fig. 4.10 Moving forward and backward

For channel 2 value greater than 1700 it moves left. For channel 2 value less than 1300 it moves right.

```
if (Channel1 < 1300) // Checks if Channel2 is lesser than 1300
{
        //delay(10);
        digitalWrite(leftMotF , HIGH);
        digitalWrite(leftMotB , LOW);
        //delay(100);
        digitalWrite(rightMotF , LOW);
        digitalWrite(rightMotB , HIGH);
        //delay(100);

if (Channel1 > 1700) // Checks if Channel2 is greater than 1500
{
      digitalWrite(leftMotF , LOW);
      digitalWrite(leftMotB , HIGH);
      digitalWrite(rightMotF , HIGH);
      digitalWrite(rightMotB , LOW);
}
}
```

Fig. 4.11 Moving left and right

If the channel 4 value is less than 1500 then the shaft remains off. If the channel 4 value is greater than 1500 then the shaft turns on.

```
if (ChannelWeapon < 1500) // Checks if Channel2 is lesser than 1300
{
        //delay(10);
        digitalWrite(weaponRun , HIGH);
}
if (ChannelWeapon > 1500) {
  digitalWrite(weaponRun , LOW);
}
```

Fig. 4.12 Shaft turn off and on

Arduino code is also used for water monitoring system as well.

```
StaticJsonBuffer<1000> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();

void loop() {

root["data"]= analogRead(A0);

  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);


if(Serial.available()>0)
{
root.printTo(Serial);
}
```

Fig. 4.13 Arduino code for data transfer

This code fraction is for data transfer from Arduino to nodemcu. Analog value of turbidity sensor is taken using Arduino Nano and other sensor values are taken using NodeMcu.

```
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
//float prevTemp = 0;
const char* server = "api.thingspeak.com"; //using the thingspeak server
String apiKey ="18QMHCWGIAS6PRUL";          //Insert the wrie key of thingspeak account
const char* MY_SSID = "Android";         //Give your own wifi name
const char* MY_PWD = "12345678";       //Your wifi password goes here
int sent = 0;

void setup() {

  Serial.begin(115200);
  connectWifi();

  while (!Serial) continue;

}
```

Fig. 4.14 NodeMcu code for wifi connection

The above code fraction sets up the connection of wifi and ThingSpeak server.

```
float temp;
float turb;
float pH;

int turbsensorValue = root["data"]; //thi
//Serial.write(turbsensorValue);
int PHsensorValue = analogRead(A0);   //r


//char buffer[10];
DS18B20.requestTemperatures();          /
temp = DS18B20.getTempCByIndex(0);

turb = turbsensorValue * (5.0 / 1024.0);

pH = PHsensorValue * (5.0/1024.0);
```

Fig. 4.15 Getting the sensor values

The above code collects data from sensor and converts value to 5V.

```
if (client.connect(server, 80)) { // use ip 184.106.153.149 or api.thingspeak.com
Serial.println("WiFi Client connected ");


String postStr = "api_key="+apiKey+"&field1="+String(temp)+"&field2="+String(turb)+ "&field3="+String(pH);
```

Fig. 4.16 Sending value to ThingSpeak server

The above code sends the three sensor values to ThingSpeak server every 15 seconds and it shows the visual representation in real time.

4.5.3 Simulation using PROTEUS

We simulated the L293D motor driver IC in PROTEUS. We used both arduino and Atmega32 IC for simulation purpose. Here steps of the simulation part will be discussed.

We used ISIS for draw the circuit diagram in PROTEUS. First in the proteus opening window we select ISIS below the menu bar at top left corner.

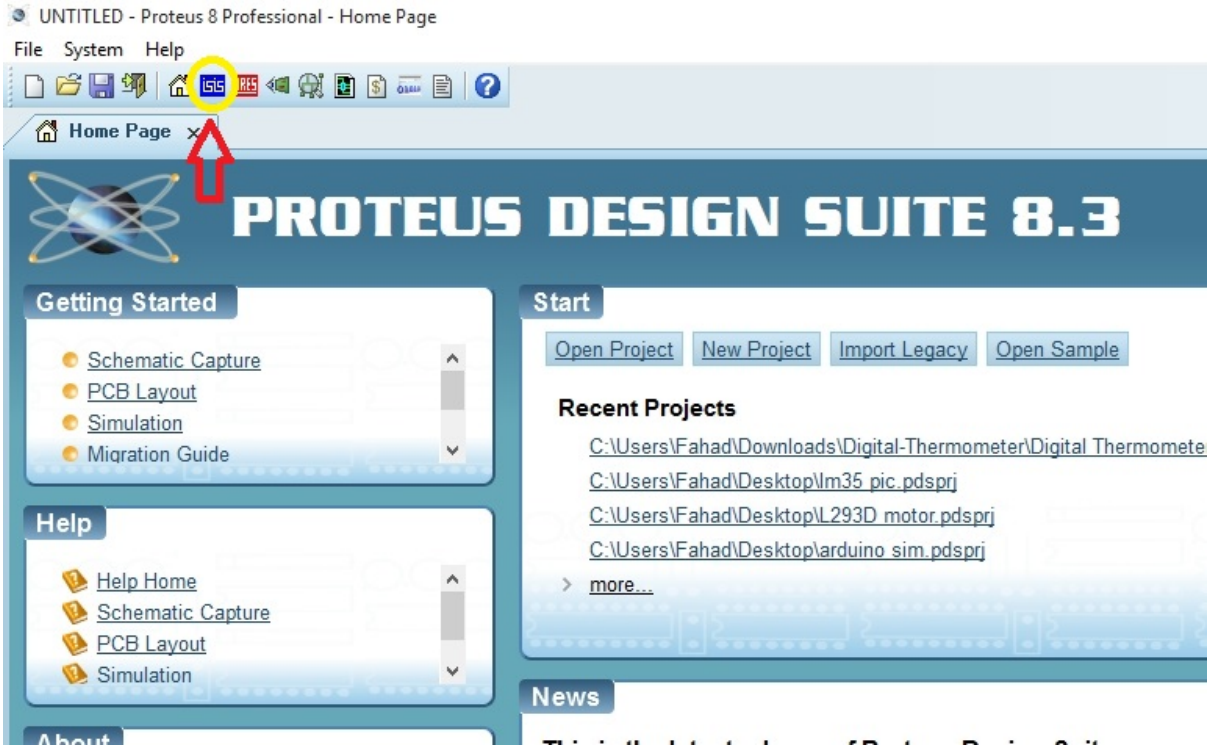

Fig. 4.17. ISIS selection.

Then we select component library. From the library we pick L293D, Atmega32, Arduino, capacitors, crystal oscillator, resistors, DC motor etc. Then using these components, we draw the circuit diagram. Then we uploaded the hex file generated by compiling the microcontroller codes on the microcontroller. To start the simulation, we select the “play” button on the bottom left.

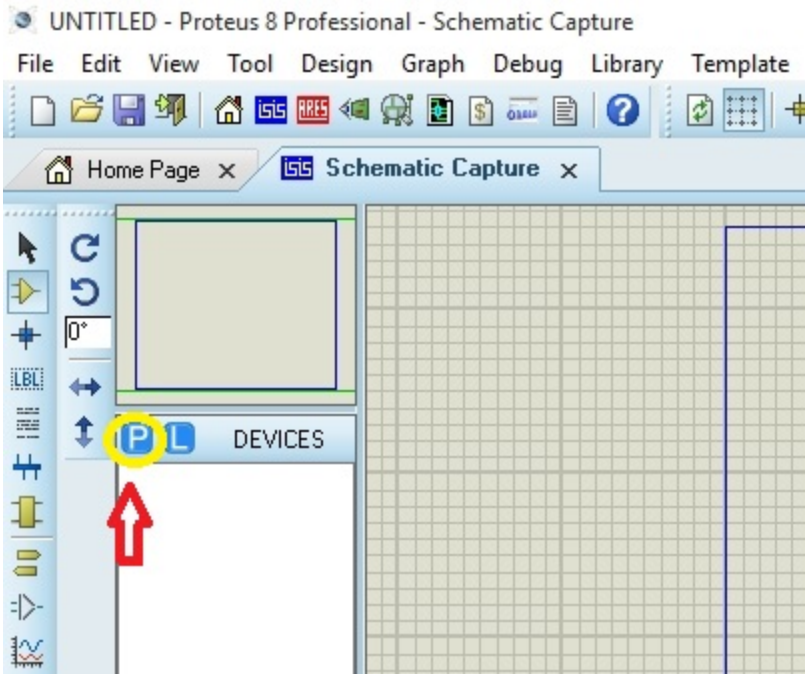

Fig. 4.18. Pick components from library.

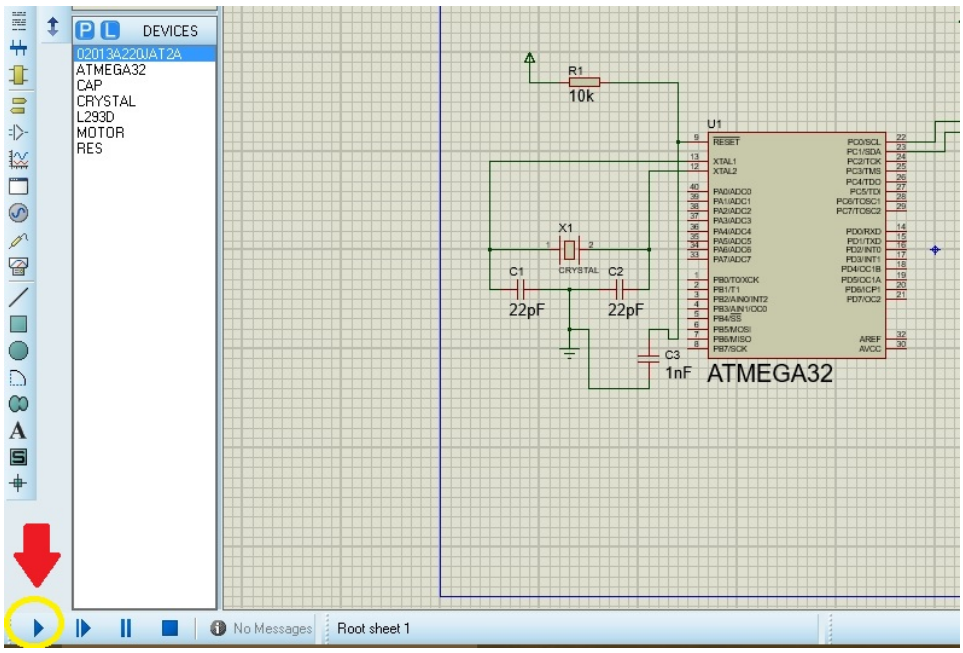

Fig. 4.19. Starting the simulation.

For the simulation part we wrote a program that will instruct the motor to run in clockwise direction for some time. Then it will be stopped for some time then it will be running in the opposite direction.

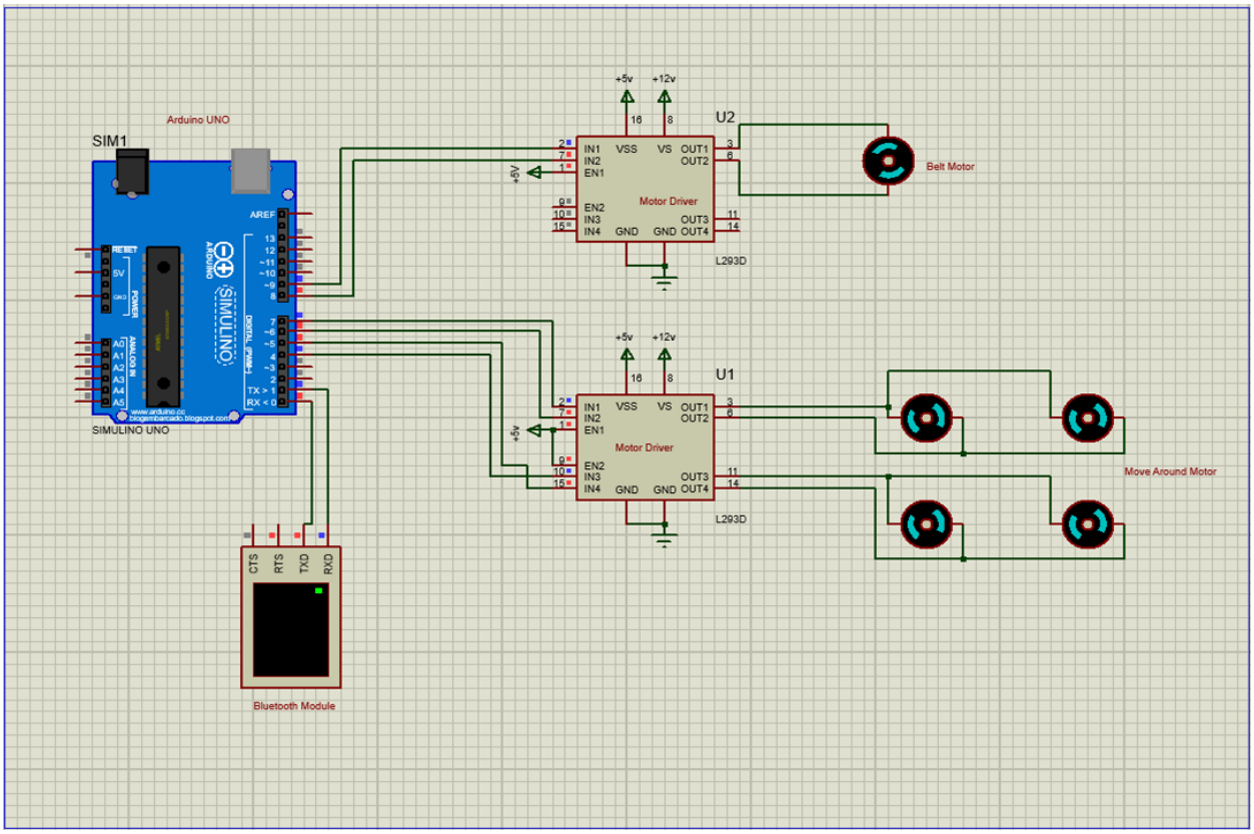

Fig 4.20: control system simulation

4.5.4 Programming in android studio

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. Running the android studio, we get a choice of creating a new project or run an existing project. After selecting to create a new project we get to select our working directory and give a name to our project then we get to choose the lowest API for the application we indicate what is the

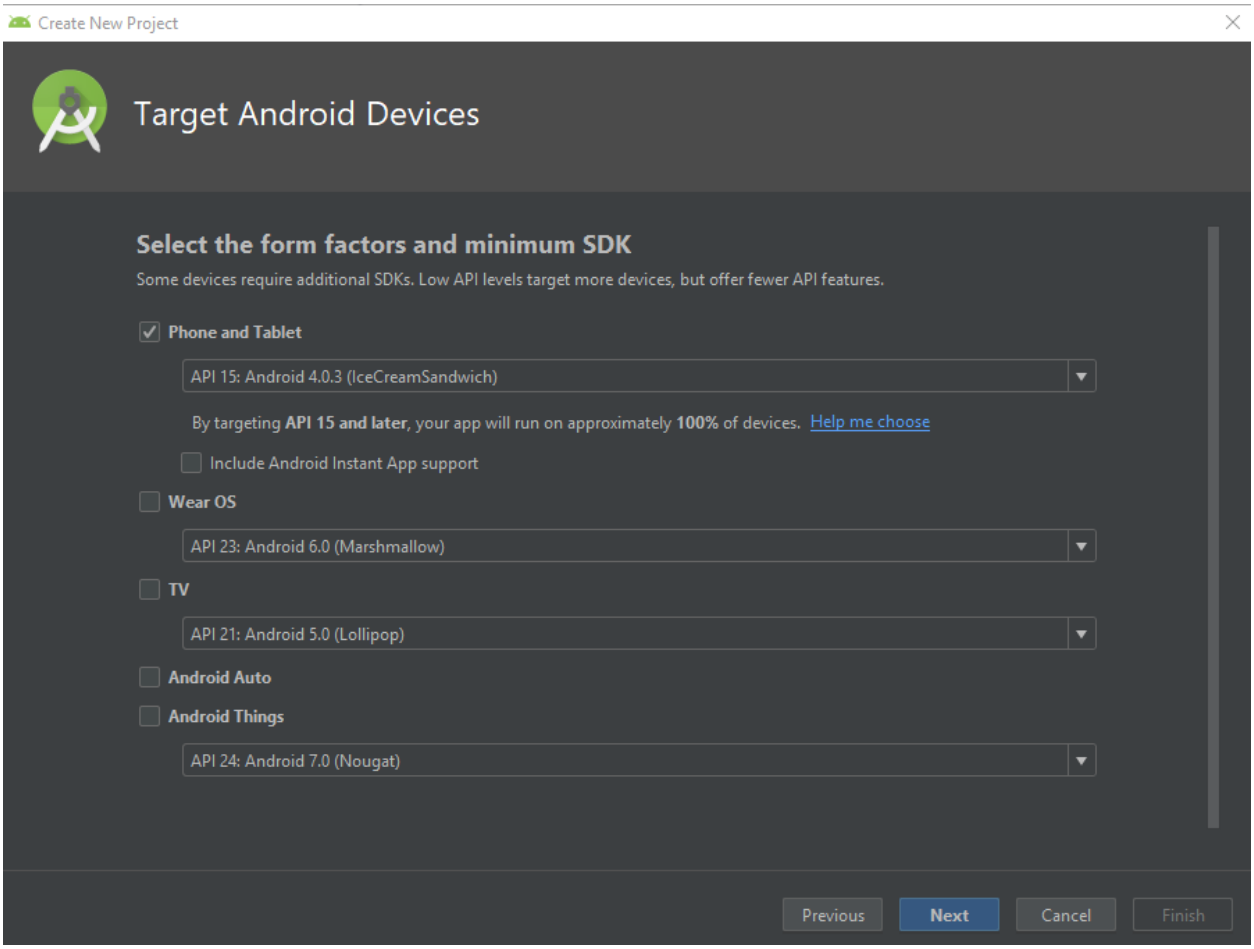

Fig 4.21: setting up Android Studio

lowest android version it will run upon. Then we get to select our first activity to be like the, then we have to make an xml file to decorate the application and then we start to write the java class to make the application work accordingly. After finishing the code we have two choices either we can run it on a physical device or test or we can create a virtual device to test the application. In our case we used our physical device to test it, to use the physical device one needs to start the developer mode on his/her device and turn on the debugging to make Android studio access its component. After finishing, there is a button (Shown in the image) on the top right corner to run the application.

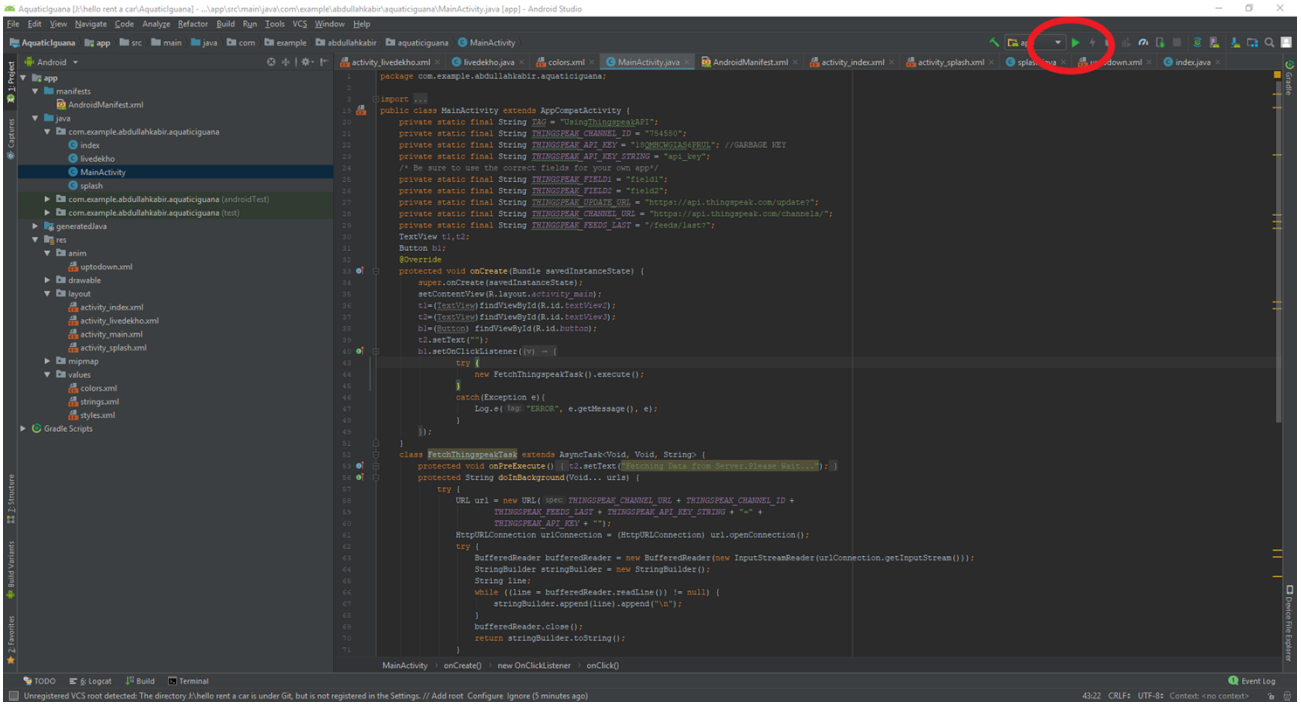

Fig 4.22 code run android studio

If after running the application any bugs are found, we can solve it but looking it on to the terminal.

4.6 Summary

Software plays a very important role in this project. The chapter mostly talks about different software used such as Android Studio, Arduino, Proteus ISIS. We have also discussed about the drone controlling codes and the water monitoring system codes as well.

Chapter 5

Design Impact

5.1 Economic Impact

Our radio controlled water waste collector is of optimal cost. The drone is built in such a way that it is cost effective and can serve its purpose smoothly. Within this cost it not only just collects waste materials but also monitors the water quality which can be used for future purposes. By making market price low and producing more products it can benefit and serve economical purposes as well. Through more research the cost of the product can be minimized even more which will be way more beneficial.

5.2 Environmental Impact

The main purpose of Aquatic Iguana is to collect floating water waste material and also monitor the water quality as well. Since most of the rivers, lakes of Bangladesh is not well maintained and water waste keeps polluting them, so our water drone will have a great environmental impact. Water quality monitoring system will help to predict the future condition of water and also provide wit sufficient data.

5.3 Manufacturability

Manufacturing of the drone is easy and less complex. The parts of the robots are easily available and it is easy to construct. When it will be manufactured commercially the cost will be reduced a lot. Furthermore, the body can be also made of plastic which will reduce the cost greatly.

5.4 Sustainability

Our robot has good speed and it is agile, it is easy to use and floats smoothly around water surface. The body structure is made of steel and as a result it is very strong. The shaft of the drone also rotates very smoothly which helps to collect ample amount of waste material.

Chapter 6

Total Cost for Implementation

Here is our total cost for implementation of the project. We tried our best to reduce the cost as much as possible. We think we are successful to implement the drone within a limited cost.

Components	Price(Taka)
Body	17000
Arduino Mega	800
Arduino Nano	319
4-Channel Relay(5V)	220
2-Channel Relay(5V)	130
Wiper Motors (4)	3600
Wild Scorpion 11.1v 2200mAh Li-Po battery (4)	7580
IMAX B6 Professional Balance charger with adapter	2169
Raspberry pi 3 model B+	3849
Raspberry pi Camera module	1290
Raspberry pi touch display(3.5'')	1949
Raspberry pi case with cooling fan	349
San Disk 16 GB micro SD card	749
pH sensor	4500
Temperature Sensors (2)	300
Turbidity Sensor	2500
Radio Controller	3000
NodeMCU ESP8266 Wi-Fi module	448
Jumper wire	300
Glue gun with sticks	400

Soldering Iron	450
Electrical + Duck + Sanitary Tape	350
PVC Pipe	500
8 inch self-locking Zip tie	150
T plug male and female connectors	105
PVC board	3000
Transportation cost	7500
Silicon Glue + Fevicol glue	500
Total Price	**64007**

Chapter 7

Result

7. Result

The main object of this drone was to collect floating waste material and collect data of water quality using different sensors. We've made the controlling system using radio controller. We've also set up Raspberry Pi based camera to make the operating system smoother. The control system works the way we desired. There is also a mobile application to see the live video streaming. The app also shows the value of the sensor data.

Fig 7.1: Aquatic Iguana collecting waste

The water quality monitoring system also works the way we wanted and sends sensor data to ThingSpeak server. ThingSpeak shows the visual representation of sensor data in real time. For

our experiment we have collected some data of Hatirjheel water and the sensors showed accurate values.

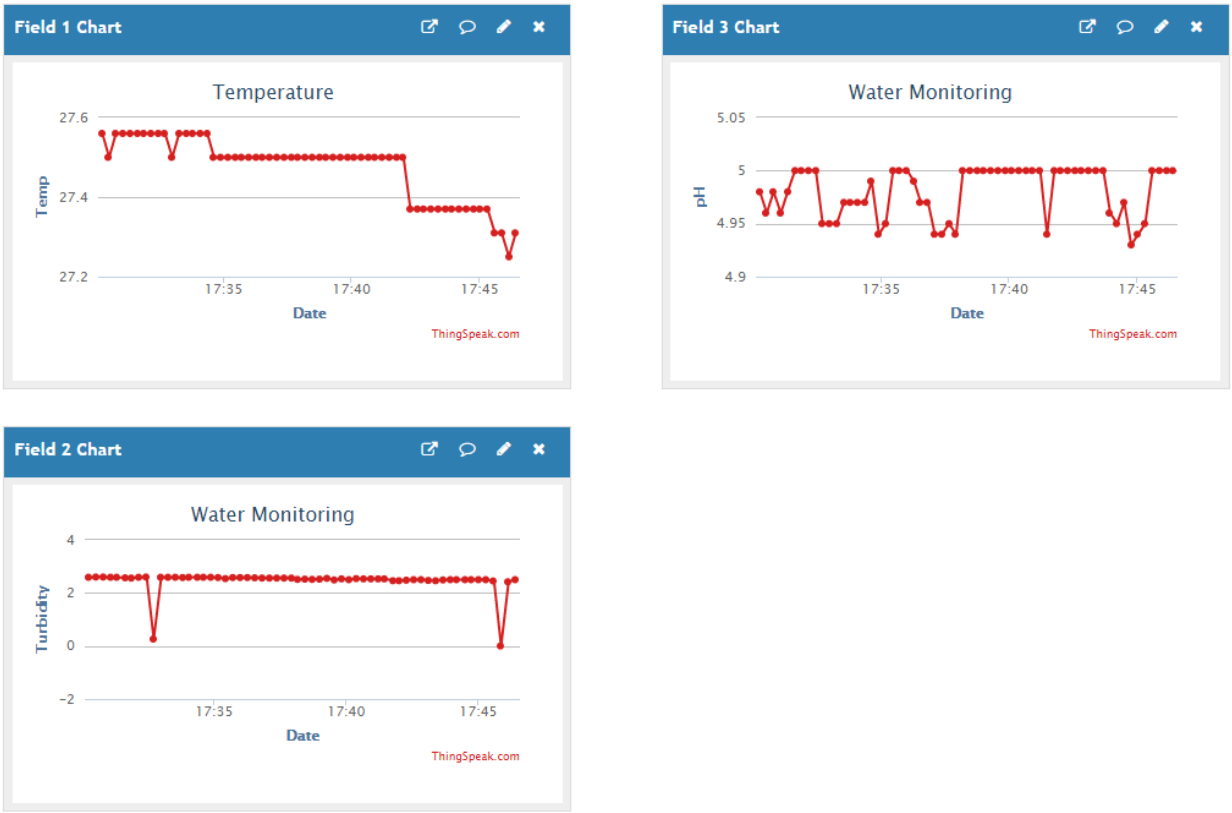

Fig 7.2: data collection result

This is the visual representation of sensor values.

Chapter 8

Conclusion

8. Conclusion

In this project we have implemented water drone which moves around the water surface and collects waste materials. It also monitors the water quality. The controlling system is done using radio controller and a raspberry pi based camera is added to make the operating system easier to the controller. Three sensor data is sent every fifteen seconds to thingspeak server, which also provides a visual representation. The drone is able to collect ample amount of waste at a time. With one full charge it can move around and collect waste material for about thirty minutes. The body structure is also very strong.

Aquatic Iguana will bring an innovative solution to water pollution problem. With proper instruction we hope to develop it further in near future. We want to make two more versions of this water drone. One will be fully autonomous and another will be Wi-Fi based. These three models will bring revolutionary change to our technological solutions in terms of water pollution problems.

Chapter 9

Bibliography

9. Bibliography

[1] Cho, "Our Oceans: A Plastic Soup", *State of the Planet*. [Online]. Available: https://blogs.ei.columbia.edu/2011/01/26/our-oceans-a-plastic-soup/. [Accessed: 08- May- 2019].

[2] Hasan, M. (2011). *Pollution of rivers around Dhaka*. [online] The Daily Star. Available at: https://www.thedailystar.net/news-detail-201795 [Accessed 21 Oct. 2018].

[3] Stevens, "Fish Pond Water Quality: As Simple as Chemistry 101", *Noble Research Institute*, 2011. [Online]. Available: https://www.noble.org/news/publications/ag-news-and-views/2009/july/fish-pond-water-quality-as-simple-as-chemistry-101/. [Accessed: 08- May- 2019].

[4] Ali, T. (2017). *4 Dhaka Rivers: Little care to keep them alive*. [online] The Daily Star. Available at: https://www.thedailystar.net/frontpage/4-dhaka-rivers-little-care-keep-them-alive-1436353 [Accessed 22 Oct. 2018].

[5] H. Kuffel, "Internet-Controlled Trash Robot Will Clean Up the Chicago River", *Waste360*, 2019. [Online]. Available: https://www.waste360.com/fleets-technology/internet-controlled-trash-robot-will-clean-chicago-river. [Accessed: 08- May- 2019].

[6] "SEAVAX RIVER VAX RIVERVAX SEWAGE WASTE WATER POLLUTION CLEANING WORKBOATS", *Bluebird-electric.net*. [Online]. Available: http://www.bluebird-electric.net/oceanography/Ocean_Plastic_International_Rescue/RiverVax_River_Cleaning_Workboats.htm. [Accessed: 08- May- 2019].

[7] H. Magazine et al., "Garbage-collecting aqua drones and jellyfish filters for cleaner oceans | Robohub", *Robohub.org*, 2019. [Online]. Available: https://robohub.org/garbage-collecting-aqua-drones-and-jellyfish-filters-for-cleaner-oceans/. [Accessed: 08- May- 2019].

[8] Ieeexplore.ieee.org. (2017). *Amphibious clean — Up robot - IEEE Conference Publication*. [online] Available at: https://ieeexplore.ieee.org/document/8078972 [Accessed 22 Oct. 2018].

[9] Muyunda, N. (2017). *Arduino-based smart garbage monitoring system: Analysis requirement and implementation - IEEE Conference Publication*. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/8270394/ [Accessed 22 Oct. 2018].

[10] S. Geetha and S. Gouthami, "Internet of things enabled real time water quality monitoring system", *Smart Water*, vol. 2, no. 1, 2016. Available: 10.1186/s40713-017-0005-y.

[11] H. Chan and W. System, "Water Quality Monitoring System", *Hackster.io*, 2019. [Online]. Available: https://www.hackster.io/chanhj/water-quality-monitoring-system-ddcb43. [Accessed: 08- May- 2019].

[12] "Relay", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/Relay. [Accessed: 08- May- 2019].

[13] "How Relays Work | Relay diagrams, relay definitions and relay types", *Galco.com*. [Online]. Available: http://www.galco.com/comp/prod/relay.htm. [Accessed: 08- May- 2019].

[14] D. Caballero, "How to use a pH sensor with Arduino", *Scidle*. [Online]. Available: https://scidle.com/how-to-use-a-ph-sensor-with-arduino/. [Accessed: 08- May- 2019].

[15] "Gravity: Analog Turbidity Sensor For Arduino - DFRobot", *Dfrobot.com*. [Online]. Available: https://www.dfrobot.com/product-1394.html. [Accessed: 08- May- 2019].

[16] "Stainless Steel Waterproof Temperature Sensor | Sensor - Temp/Humid | Robot R Us", *Robot-r-us.com*. [Online]. Available: https://www.robot-r-us.com/vmchk/sensor-temp/humid/one-wire-waterproof-temperature-sensor.html. [Accessed: 08- May- 2019].

[17]"NodeMCU", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/NodeMCU#ESP8266_Arduino_Core. [Accessed: 08- May- 2019].

[18] A. Aqeel and G+, "Introduction to Arduino Nano - The Engineering Projects", *The Engineering Projects*. [Online]. Available: https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-nano.html. [Accessed: 08- May- 2019].

[19] "Learn More - ThingSpeak IoT", *Thingspeak.com*. [Online]. Available: https://thingspeak.com/pages/learn_more. [Accessed: 08- May- 2019].

Appendices

Appendix A

Drone control Code:

```
int leftMotF = 6;// defines pin 5 as connected to the motor
int leftMotB= 5;// defines pin 6 as connected to the motor
int rightMotF = 7;// defines pin 7 as connected to the motor
int rightMotB = 3;
int weaponRun=11;// defines pin 8 as connected to the motor

int channel1 = 13; // defines the channels that are connected
int channel2 = 10;
int channelWeapon= 12;
int reciverPower=32;
int reciverGnd=33;
int weaponGnd= 22;
// to pins 9 and 10 of arduino respectively

int Channel1 ; // Used later to
int Channel2 ;
int ChannelWeapon;
//int more = 4;// store values

void  setup ()
{
  pinMode(channelWeapon,INPUT);
   pinMode (leftMotF, OUTPUT);// initialises the motor pins
   pinMode (leftMotB, OUTPUT);
```

```
  pinMode (rightMotF, OUTPUT);
  pinMode (rightMotB, OUTPUT);// as outputs
  pinMode (channel1, INPUT);// initialises the channels
  pinMode (channel2, INPUT);// as inputs
  pinMode(weaponRun,OUTPUT);

  pinMode(reciverPower,OUTPUT);
  digitalWrite(reciverPower,HIGH);

  pinMode(reciverGnd,OUTPUT);
  digitalWrite(reciverGnd,LOW);

  pinMode(weaponGnd,OUTPUT);
  digitalWrite(weaponGnd,LOW);


  Serial.begin (9600); // Sets the baud rate to 9600 bps
 // pinMode (more ,OUTPUT);
   //digitalWrite(more , HIGH);
}

void  loop ()
{

 {
 Channel1 = (pulseIn (channel1, HIGH)); // Checks the value of channel1
 Serial.print("Channel 1: ");
```

```
 Serial.println (Channel1);
 //delay(2000);


 //Prints the channels value on the serial monitor


if ((Channel1 > 1300 && Channel1 < 1600 ) && (Channel2 > 1300 && Channel2 < 1600  ))
/*If these conditions are true, do the following. These are the values that I got from my
transmitter, which you may customize according to your transmitter values */
 {
        digitalWrite(leftMotF , HIGH);
        digitalWrite(leftMotB , HIGH);
        digitalWrite(rightMotF , HIGH);
        digitalWrite(rightMotB , HIGH);
 }



 if (Channel2 > 1700) // Checks if Channel1 is lesser than 1300
 {
    digitalWrite(leftMotF , LOW);
    digitalWrite(leftMotB , HIGH);
    digitalWrite(rightMotF , LOW);
    digitalWrite(rightMotB , HIGH);
 }
 if (Channel2 < 1300) // Checks if Channel1 is greater than 1500
 {
        digitalWrite(leftMotF , HIGH);
        digitalWrite(leftMotB , LOW);
        digitalWrite(rightMotF , HIGH);
```

```
        digitalWrite(rightMotB , LOW);
}
}
{
Channel2 = (pulseIn (channel2, HIGH)); // Checks the value of channel1
Serial.print("Channel 2: ");
Serial.println (Channel2);
// delay(2000);

//Prints the channels value value on the serial monitor

if (Channel1 < 1300) // Checks if Channel2 is lesser than 1300
{
        //delay(10);
        digitalWrite(leftMotF , HIGH);
        digitalWrite(leftMotB , LOW);
        //delay(100);
        digitalWrite(rightMotF , LOW);
        digitalWrite(rightMotB , HIGH);
        //delay(100);
}
if (Channel1 > 1700) // Checks if Channel2 is greater than 1500
{
     digitalWrite(leftMotF , LOW);
     digitalWrite(leftMotB , HIGH);
     digitalWrite(rightMotF , HIGH);
     digitalWrite(rightMotB , LOW);
```

```
  }
  }
  {
ChannelWeapon = (pulseIn (channelWeapon, HIGH)); // Checks the value of channel1
  Serial.print("Channel Wepon: ");
  Serial.println (ChannelWeapon);
//  delay(2000);
  if (ChannelWeapon < 1500) // Checks if Channel2 is lesser than 1300
  {
        //delay(10);
        digitalWrite(weaponRun , HIGH);
  }
  if (ChannelWeapon > 1500) {
    digitalWrite(weaponRun , LOW);
  }

  Serial.println (weaponRun);
  }
}
```

Appendix B

Water Quality Monitoring Code:

Arduino Code:

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
//SoftwareSerial s(5,6);
void setup() {


  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);

Serial.begin(115200);



}

StaticJsonBuffer<1000> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();

void loop() {

root["data"]= analogRead(A0);
```

```
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);

if(Serial.available()>0)
{
root.printTo(Serial);
}
}
```

NodeMcu Code:

```
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ArduinoJson.h>

#define myPeriodic 15 //in sec | Thingspeak pub is 15sec

#define ONE_WIRE_BUS 2  // DS18B20 on arduino pin2 corresponds to D4 on physical board

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
//float prevTemp = 0;
const char* server = "api.thingspeak.com"; //using the thingspeak server
String apiKey ="ET5Y5KOFCR3ACEAX";          //Insert the wrie key of thingspeak account
const char* MY_SSID = "Ayyan";         //Give your own wifi name
```

```
const char* MY_PWD = "mollah5292";       //Your wifi password goes here
int sent = 0;

void setup() {

  Serial.begin(115200);
  connectWifi();

  while (!Serial) continue;

}

void loop() {

  StaticJsonBuffer<1000> jsonBuffer;                    //Declaring JsonBuffer
  JsonObject& root = jsonBuffer.parseObject(Serial);            // creating JsonObject...Data through serial communication will be parsed through here
  if (root == JsonObject::invalid())
  {
     return;
  }
  /*
   Declaring the float temperature, turbidity, and ph
  */
  float temp;
  float turb;
  float pH;
```

```
  int turbsensorValue = root["data"]; //this vale is coming from arduino through serial port
  //Serial.write(turbsensorValue);
  int PHsensorValue = analogRead(A0);   //readinh the ph values through analog port

  //char buffer[10];
  DS18B20.requestTemperatures();          //Reading the temperature sensor values
  temp = DS18B20.getTempCByIndex(0);

  turb = turbsensorValue * (5.0 / 1024.0); // measuring the turbidity value in voltage.It can also
be converted into NTU.

  pH = PHsensorValue * (5.0/1024.0);        // Measuring the ph value in voltage

  //String tempC = dtostrf(temp, 4, 1, buffer);//handled in sendTemp()
  Serial.print(String(sent)+" Temperature: ");
  Serial.println(temp);

  Serial.println (String(sent)+" Turbidity_Voltage: ");
  Serial.println (turb);

  Serial.println (String(sent)+" PHsensor_Voltage: ");
  Serial.println (pH);
```

```
  sendTemperatureTurbidityTS(temp,turb,pH);   // Calling the function which sends three variable as float.


  int count = myPeriodic;       //Giving 15 seconds as there is 15 second interval to send data to thingspeak
  while(count--)
  delay(1000);


}


void connectWifi()
{
  Serial.print("Connecting to "+*MY_SSID);
  WiFi.begin(MY_SSID, MY_PWD);
  while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }


  Serial.println("");
  Serial.println("Connected");
  Serial.println("");
}//end connect



void sendTemperatureTurbidityTS(float temp, float turb, float pH)
{
   WiFiClient client;
```

```
  if (client.connect(server, 80)) { // use ip 184.106.153.149 or api.thingspeak.com
  Serial.println("WiFi Client connected ");


  String postStr = "api_key="+apiKey+"&field1="+String(temp)+"&field2="+String(turb)+
"&field3="+String(pH);  //Sending the data as string to thingspeak


  /*Client side status*/
  client.print("POST /update HTTP/1.1\n");
  client.print("Host: api.thingspeak.com\n");
  client.print("Connection: close\n");
  client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
  client.print("Content-Type: application/x-www-form-urlencoded\n");
  client.print("Content-Length: ");
  client.print(postStr.length());
  client.print("\n\n");
  client.print(postStr);
  delay(1000);


  }//end if
  sent++;
 client.stop();
}//end send
```

Appendix C

Android Application code

Index Activity XML:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".index"
    android:background="@color/colorPrimaryDark">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="210dp"
        android:layout_height="180dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginTop="64dp"
        android:layout_marginEnd="67dp"
        android:layout_marginRight="67dp"
        app:srcCompat="@drawable/logo" />

    <Button
        android:id="@+id/sensor"
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="82dp"
        android:layout_marginLeft="82dp"
        android:layout_marginBottom="224dp"
        android:gravity="center"
        android:text="@string/check_sensor" />

    <Button
        android:id="@+id/live"
```

```
        android:layout_width="100dp"
        android:layout_height="60dp"
        android:layout_alignTop="@+id/sensor"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginTop="0dp"
        android:layout_marginEnd="62dp"
        android:layout_marginRight="62dp"
        android:gravity="center"
        android:text="@string/see_live" />

    <Button
        android:id="@+id/button4"
        android:layout_width="221dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="79dp"
        android:layout_marginLeft="79dp"
        android:layout_marginBottom="155dp"
        android:text="Start Autonomous" />

</RelativeLayout>
```

Index Activity Java file:

```
package com.example.abdullahkabir.aquaticiguana;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class index extends AppCompatActivity {
    private Button Sensors;
    private Button LiveStream;
    private Button Autonomous;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_index);
        intcomponent();
    }
    private void intcomponent(){
        Sensors = (Button) findViewById(R.id.sensor);
        LiveStream = findViewById(R.id.live);
        Autonomous = findViewById(R.id.button4);
        intAction();
    }
    private void intAction() {
        Sensors.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(index.this, MainActivity.class);
                startActivity(intent);
            }
        });
        LiveStream.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(index.this, livedekho.class);
                startActivity(intent);
            }
        });
        Autonomous.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(index.this, "Coming Soon...", Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

Sensor data show XML:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@color/logo">


    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginStart="84dp"
        android:layout_marginLeft="84dp"
        android:layout_marginTop="226dp"
        android:layout_marginEnd="77dp"
        android:layout_marginRight="77dp"
        android:textSize="20sp"
        android:gravity="center"
        android:text="New Text"
        android:textColor="@color/white" />
    <TextView
        android:layout_width="wrap_content"
        android:textColor="@color/white"
        android:layout_height="wrap_content"

        android:textSize="20sp"
        android:text="New Text"
        android:id="@+id/textView3"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="58dp" />
```

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginStart="135dp"
        android:layout_marginLeft="135dp"
        android:layout_marginTop="412dp"
        android:layout_marginEnd="135dp"
        android:layout_marginRight="135dp"
        android:text="Fetch Value" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        app:srcCompat="@drawable/logo" />

</RelativeLayout>
```

Sensor data java file:

```
package com.example.abdullahkabir.aquaticiguana;

import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "UsingThingspeakAPI";
    private static final String THINGSPEAK_CHANNEL_ID = "754580";
    private static final String THINGSPEAK_API_KEY = "18QMHCWGIAS6PRUL";
//GARBAGE KEY
    private static final String THINGSPEAK_API_KEY_STRING = "api_key";
    /* Be sure to use the correct fields for your own app*/
    private static final String THINGSPEAK_FIELD1 = "field1";
    private static final String THINGSPEAK_FIELD2 = "field2";
    private static final String THINGSPEAK_UPDATE_URL =
"https://api.thingspeak.com/update?";
    private static final String THINGSPEAK_CHANNEL_URL =
"https://api.thingspeak.com/channels/";
    private static final String THINGSPEAK_FEEDS_LAST = "/feeds/last?";
    TextView t1,t2;
    Button b1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        t1=(TextView)findViewById(R.id.textView2);
```

```
        t2=(TextView)findViewById(R.id.textView3);
        b1=(Button) findViewById(R.id.button);
        t2.setText("");
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    new FetchThingspeakTask().execute();
                }
                catch(Exception e){
                    Log.e("ERROR", e.getMessage(), e);
                }
            }
        });
    }
    class FetchThingspeakTask extends AsyncTask<Void, Void, String> {
        protected void onPreExecute() {
            t2.setText("Fetching Data from Server.Please Wait...");
        }
        protected String doInBackground(Void... urls) {
            try {
                URL url = new URL(THINGSPEAK_CHANNEL_URL +
THINGSPEAK_CHANNEL_ID +
                        THINGSPEAK_FEEDS_LAST + THINGSPEAK_API_KEY_STRING + "=" +
                        THINGSPEAK_API_KEY + "");
                HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
                try {
                    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                    StringBuilder stringBuilder = new StringBuilder();
                    String line;
                    while ((line = bufferedReader.readLine()) != null) {
                        stringBuilder.append(line).append("\n");
                    }
                    bufferedReader.close();
                    return stringBuilder.toString();
                }
                finally{
                    urlConnection.disconnect();
                }
```

```
            }
            catch(Exception e) {
                Log.e("ERROR", e.getMessage(), e);
                return null;
            }
        }
        protected void onPostExecute(String response) {
            if(response == null) {
                Toast.makeText(MainActivity.this, "There was an error",
Toast.LENGTH_SHORT).show();
                return;
            }
            try {
                JSONObject channel = (JSONObject) new JSONTokener(response).nextValue();
                double v1 = channel.getDouble(THINGSPEAK_FIELD1);
                double v2 = channel.getDouble(THINGSPEAK_FIELD2);
                if(v1>=10) {
                    t1.setText("Temp:" + v1);
                    t2.setText("Turbidity:" +v2);
                }
                else
                    t1.setText("NO VALUES");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Live Streaming XML:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".livedekho">
    <WebView
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        tools:layout_editor_absoluteX="8dp"
        tools:layout_editor_absoluteY="8dp"
        android:id="@+id/web"
        tools:ignore="MissingConstraints" />

</android.support.constraint.ConstraintLayout>
```

Live Streaming Java file:

```
package com.example.abdullahkabir.aquaticiguana;

import android.app.Activity;
import android.app.ProgressDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class livedekho extends AppCompatActivity {

    private WebView webView;
    Activity activity;
    private ProgressDialog progDailog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_livedekho);

        activity = this;

        progDailog = ProgressDialog.show(activity, "Loading","Please wait...", true);
        progDailog.setCancelable(false);



        webView = (WebView) findViewById(R.id.web);



        webView.getSettings().setJavaScriptEnabled(true);
```

```
        webView.getSettings().setLoadWithOverviewMode(true);
        webView.getSettings().setUseWideViewPort(true);
        webView.setWebViewClient(new WebViewClient(){

            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                progDailog.show();
                view.loadUrl(url);

                return true;
            }
            @Override
            public void onPageFinished(WebView view, final String url) {
                progDailog.dismiss();
            }
        });

        webView.loadUrl("192.68.124:8000");
    }
}
```