THIRUMALAI ENGINEERING COLLEGE

The Right Place to Enrich Your Career...

Approved by AICTE, New Delhi & Affiliated to Anna University
An ISO 9001 : 2008 Certified Institution

FLOOD MONITORING AND EARLY WARNING

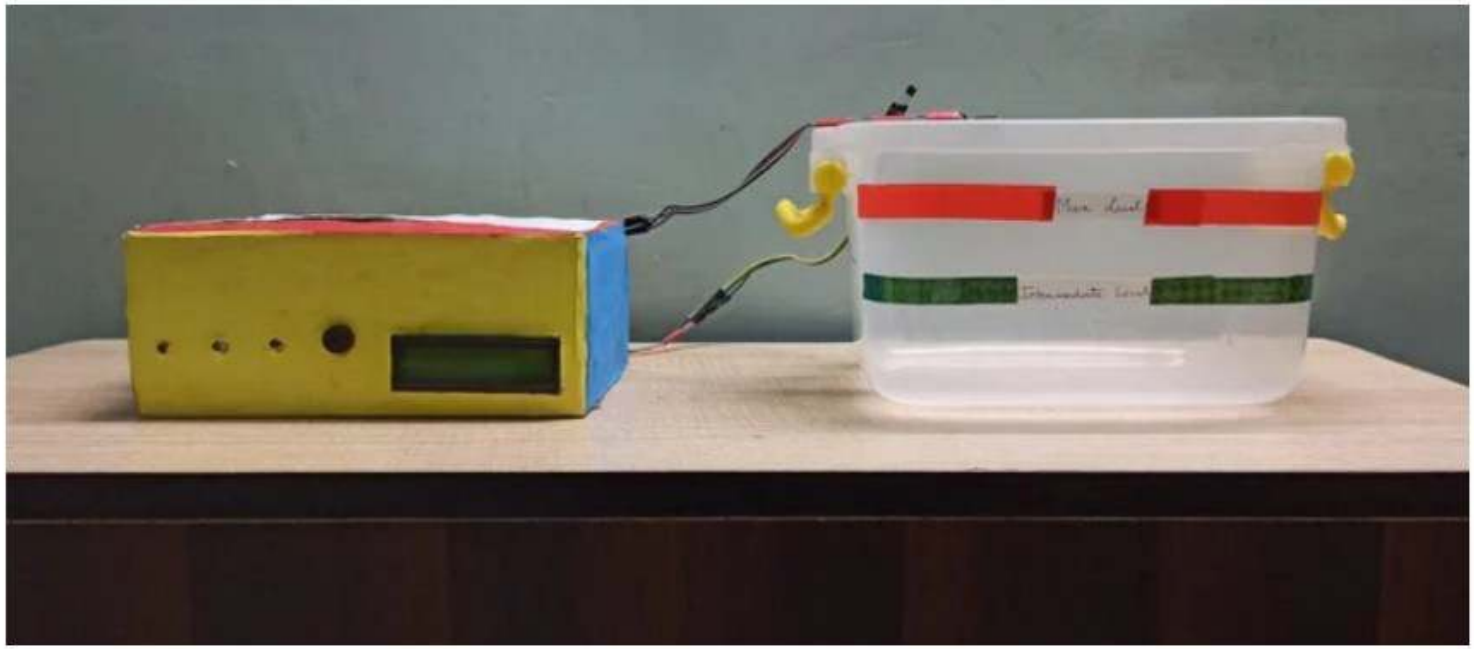
PHASE 3: DEVELOPMENT PART 1

A Project report submitted in partial fulfilment of the requirements for the degree of B.E in computer science and engineering

BY

M.BADRINATH (513221104005)

IoT Based Flood Monitoring And Alerting System



As we all know that Flood is one of the major well known Natural Disasters. When water level suddenly rises in dams, river beds etc. Alot of Destruction happens at surrounding places. It causes a huge amount of loss to our environment and living beings as well. So in these case, it is very important to get emergency alerts of the water level situation in different conditions in the river bed.

The purpose of this project is to sense the water level in river beds and check if they are in normal condition. If they reach beyond the limit, then it alerts

people through LED signals and buzzer sound. Also it alerts people through Sms and Emails alerts when the water level reaches beyond the limit.

Excited? Let's get started.

Things used in this project

Hardware components -

1. Bolt-IoT wifi module
2. Arduino uno
3. Breadboard- 400 tie points
4. 5mm LED:(Green, Red, Orange) and Buzzer
5. 16×2 LCD Display
6. LM35 Temperature Sensor
7. HC-SR04 Ultrasonic Sensor
8. Some Jumper Wires
    1. Male to Female Jumper Wires- 15 pcs
    2. Male to Male Jumper Wires- 10 pcs
    3. Female to Female Jumper Wires- 5 pcs
9. 9v Battery and Snap Connector
10. USB Cable Type B

Software components -

1. Arduino IDE
2. Python 3.7 IDLE
3. Bolt IoT Cloud

4. Bolt IoT Android App
5. Twillo SMS Messaging API
6. Mailgun EMAIL Messaging APISoftware components

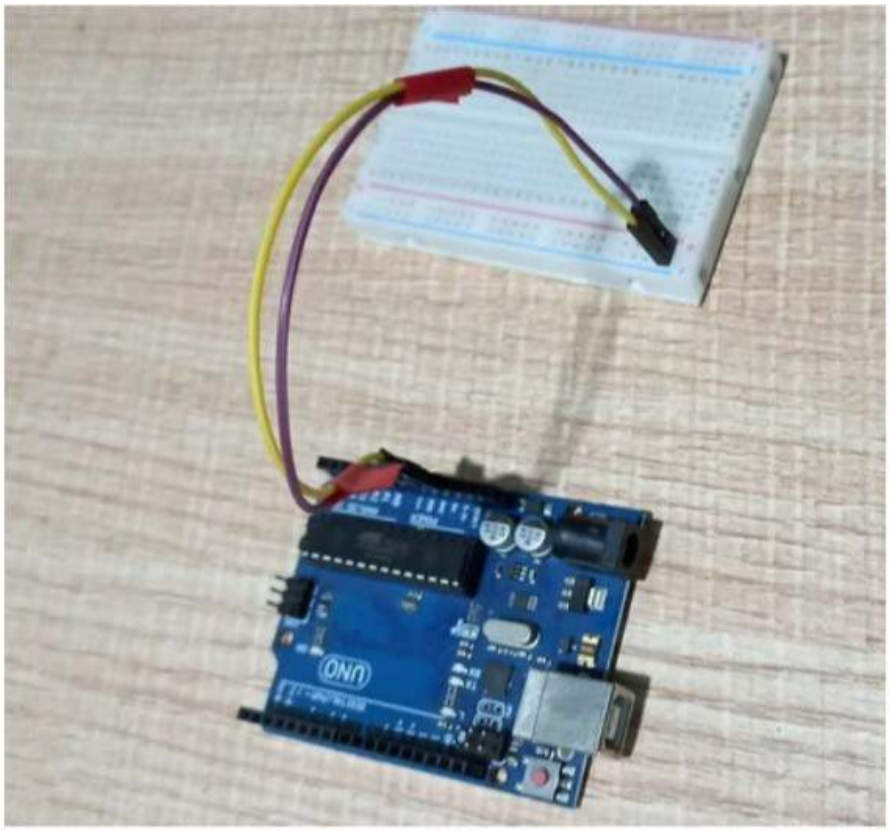
Hand tools and fabrication machines

1. Electrical Tape
2. Green Cello Tape

Hardware Setup

For Building this project we first configure the hardware connections. Then later on moving to the software part.

Step 1: Connecting 5v and GND of Arduino to the Breadboard for power connection to other components.



Step 2: Connecting LED's

For Green LED:

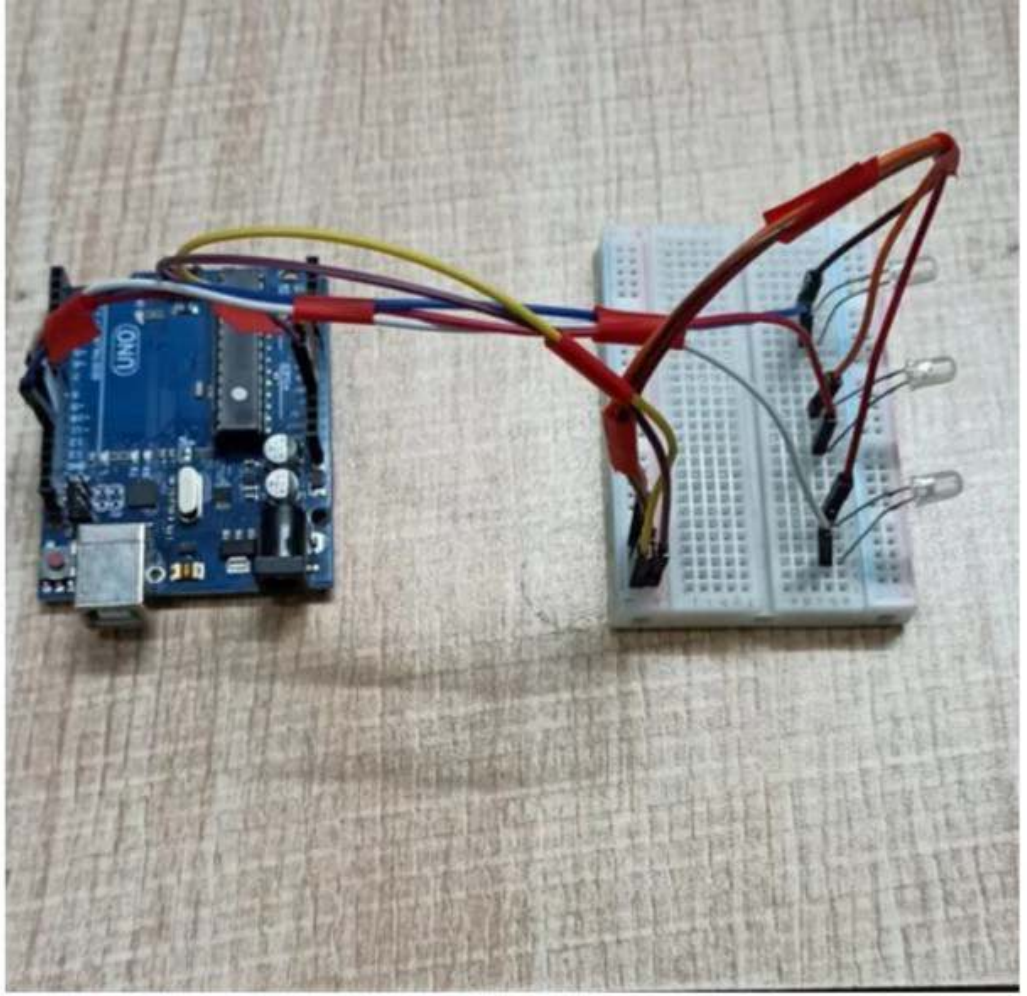
- VCC of Green Colour LED to Digital Pin '10' of the Arduino.
- GND of Green Colour LED to the GND of Arduino.

For Orange LED:

- VCC of Orange Colour LED to Digital Pin '11' of the Arduino.
- GND of Orange Colour LED to the GND of Arduino.

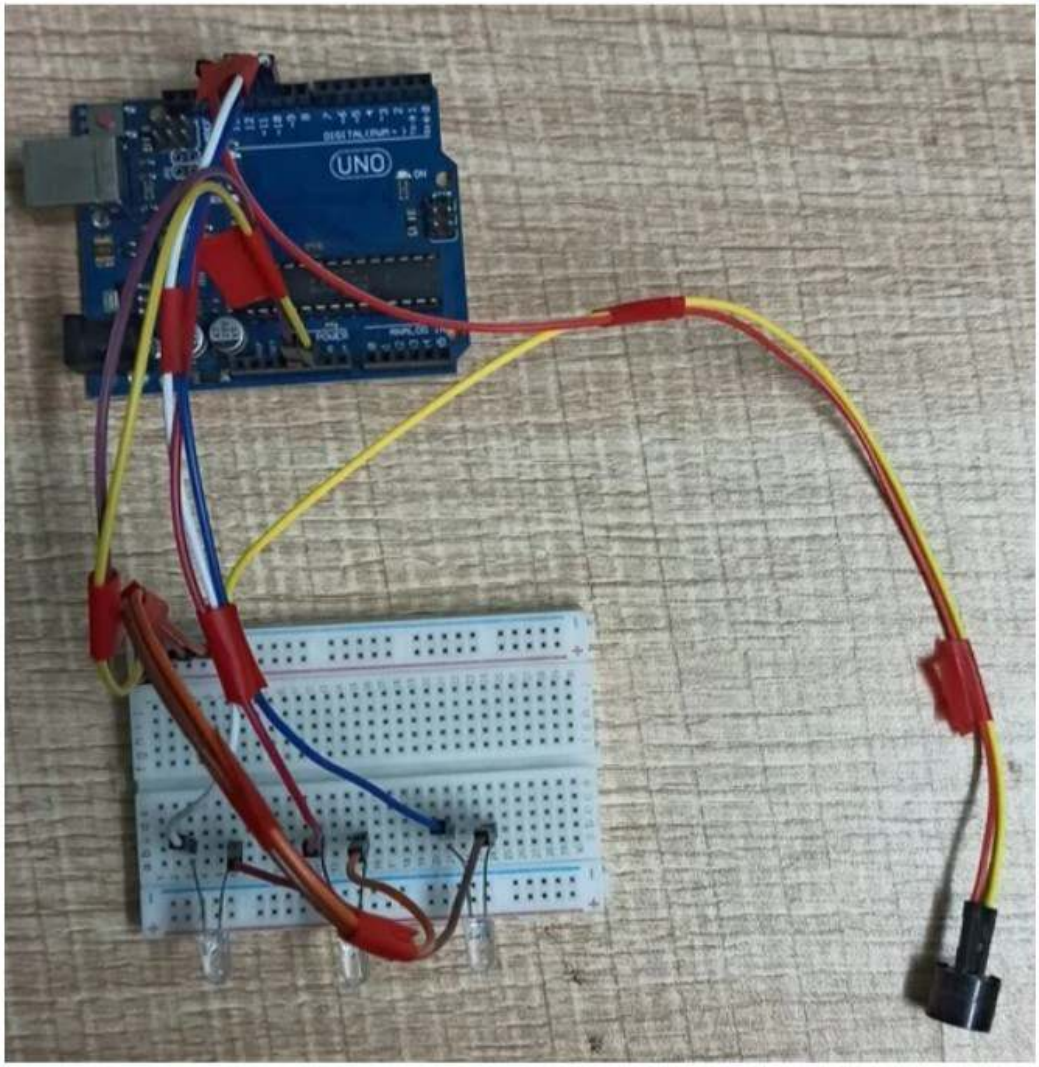
For Red LED:

- VCC of Red Colour LED to Digital Pin '12' of the Arduino.
- GND of Red Colour LED to the GND of Arduino.



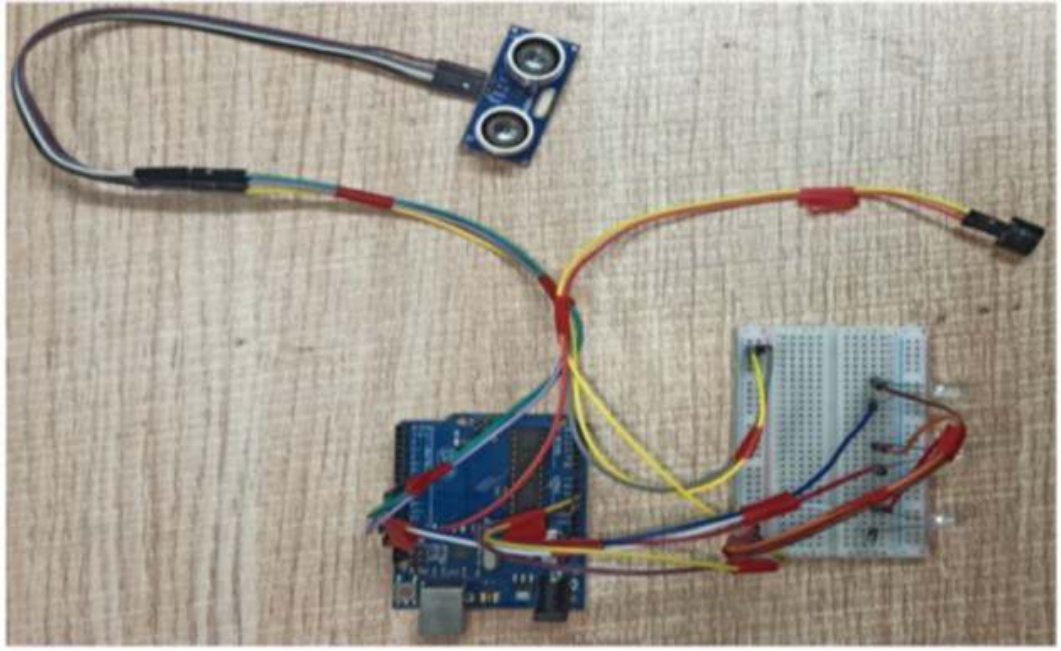
Step 3: Connecting Buzzer

- VCC of Buzzer to Digital Pin '13' of the Arduino.
- GND of Buzzer to the GND of Arduino.



Step 4: Connecting HC-SR04 Ultrasonic Sensor

- VCC of Ultrasonic Sensor to 5v of Arduino.
- GND of Ultrasonic Sensor to GND of Arduino.
- Echo of Ultrasonic Sensor to Digital Pin ‘8’ of Arduino.
- Trig of Ultrasonic Sensor to Digital Pin ‘9’ of Arduino.

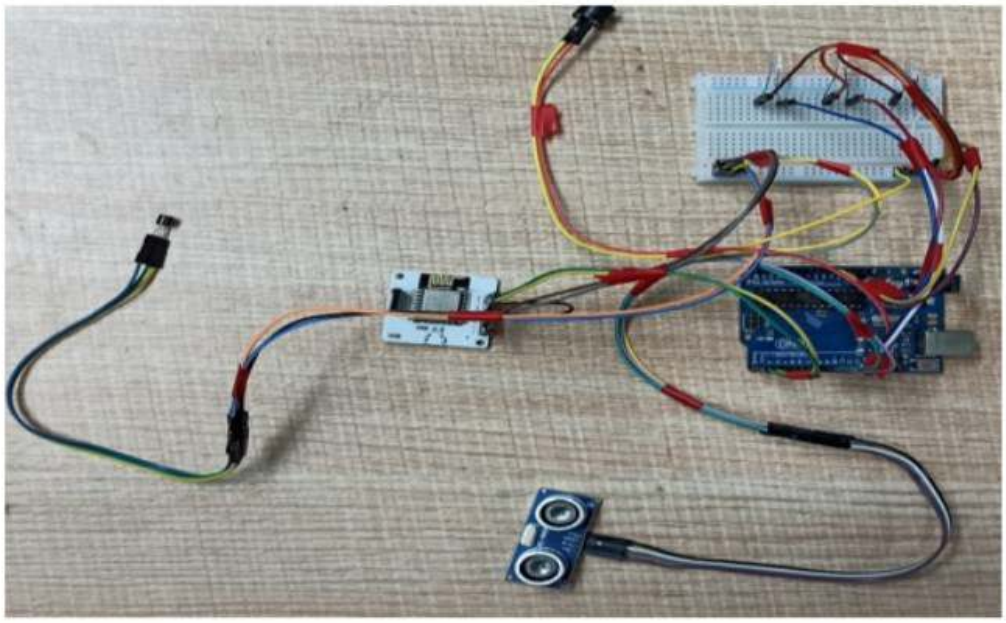


Step 5: Connecting Bolt WiFi Module

- 5v of Bolt WiFi Module to 5v of Arduino.
- GND of Bolt WiFi Module to GND of Arduino.
- TX of Bolt WiFi Module to RX of Arduino.
- RX of Bolt WiFi Module to TX of Arduino.

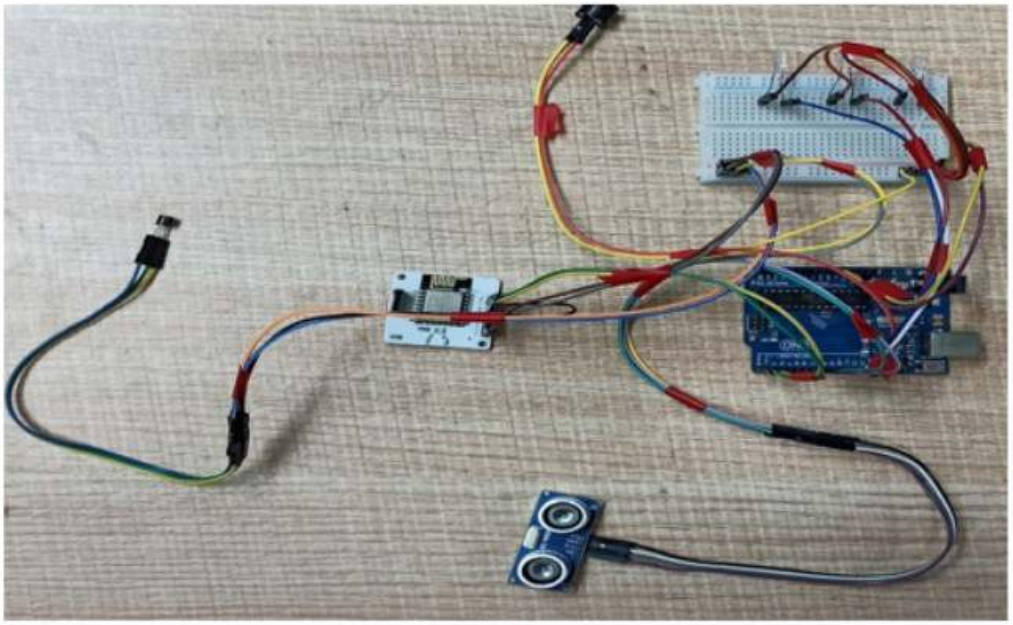
Step 6: Connecting LM35 Temperature Sensor

- VCC of LM35 to 5v of Bolt WiFi Module.
- Output Pin of LM35 to Pin 'A0' of Bolt WiFi Module.
- GND of LM35 to GND of Bolt WiFi Module.

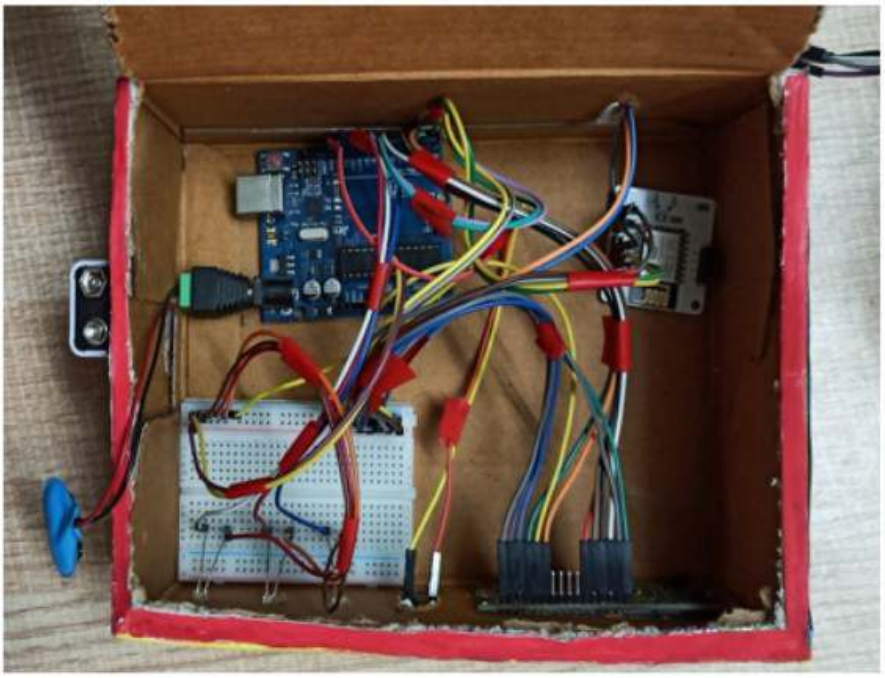


Step 7:Connecting 16×2 LCD Display

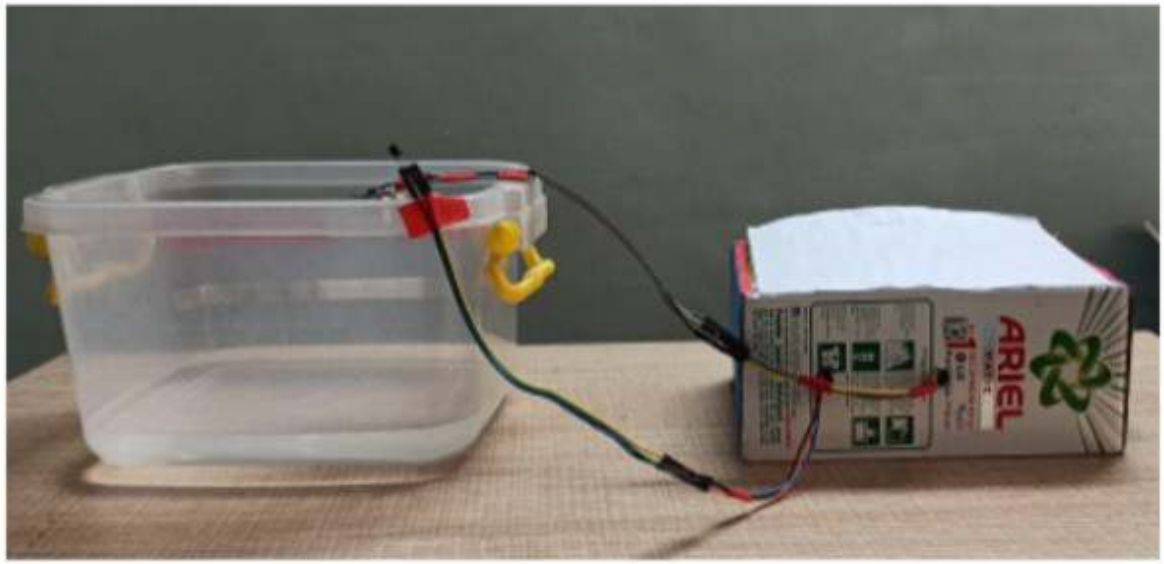
- Pin 1,3,5,16 of 16×2 LCD to GND of Arduino.
- Pin 2,15 of 16×2 LCD to 5v of Arduino.
- Pin 4 of 16×2 LCD to Digital Pin '2' of Arduino.
- Pin 6 of 16×2 LCD to Digital Pin '3' of Arduino.
- Pin 11 of 16×2 LCD to Digital Pin '4' of Arduino.
- Pin 12 of 16×2 LCD to Digital Pin '5' of Arduino.
- Pin 13 of 16×2 LCD to Digital Pin '6' of Arduino.
- Pin 14 of 16×2 LCD to Digital Pin '7' of Arduino.



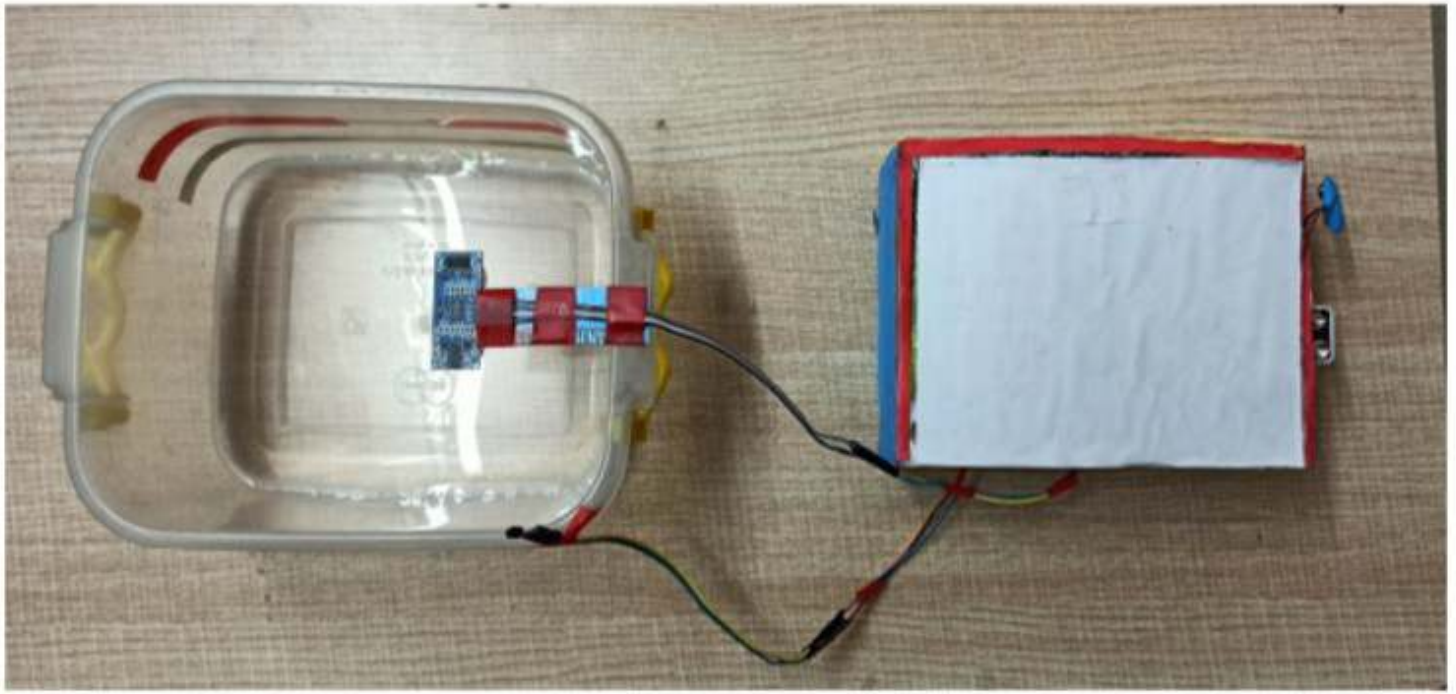
After doing the hardware connection put all the hardware components in one box.



Also attach LM35 Temperature Sensor on the side of the container.



Also attach Ultrasonic sensor on the top of the container.



Step 5: Coding

After setting online app services and saving the keys somewhere. Now most important is to write code and allow sensors attached to microcontroller to take specific decisions.

Basically this project contains two editors to write the code. First is Arduino IDE in that we will write the arduino code. Second the Python IDE in that we will write the configuration file and the main code. Also the download link of both the editor can find above in the online app services section.

Step 5.1: Writing the code in the Arduino IDE

- Open the Arduino IDE(Downloaded from the above section).
- Click on new file. Choose the correct file path to save the file. Give appropirate name to the file and add .ino extention to the file and save the file.
- Now the core part of the project is writing code for Arduino Uno. Below this line complete code is given. You can refer the below code.

```
#include<LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

const int in = 8;
const int out = 9;
const int green = 10;
const int orange = 11;
const int red = 12;
const int buzz = 13;

void setup() {
Serial.begin(9600);
lcd.begin(16, 2);
pinMode( in , INPUT);
pinMode(out, OUTPUT);
pinMode(green, OUTPUT);
pinMode(orange, OUTPUT);
pinMode(red, OUTPUT);
pinMode(buzz, OUTPUT);
digitalWrite(green, LOW);
digitalWrite(orange, LOW);
digitalWrite(red, LOW);
digitalWrite(buzz, LOW);
lcd.setCursor(0, 0);
lcd.print("Flood Monitoring");
lcd.setCursor(0, 1);
lcd.print("Alerting System");
delay(5000);
```

```
lcd.clear();
}

void loop() {
long dur;
long dist;
long per;
digitalWrite(out, LOW);
delayMicroseconds(2);
digitalWrite(out, HIGH);
delayMicroseconds(10);
digitalWrite(out, LOW);
dur = pulseIn( in , HIGH);
dist = (dur * 0.034) / 2;
per = map(dist, 10.5, 2, 0, 100);
#map
function is used to convert the distance into percentage.
if(per < 0) {
per = 0;
}
if (per > 100) {
per = 100;
}
Serial.println(String(per));
lcd.setCursor(0, 0);
lcd.print("Water Level:");
lcd.print(String(per));
lcd.print("% ");
if (per >= 80) #MAX Level of Water--Red Alert!{
```

```
lcd.setCursor(0, 1);
lcd.print("Red Alert! ");
digitalWrite(red, HIGH);
digitalWrite(green, LOW);
digitalWrite(orange, LOW);
digitalWrite(buzz, HIGH);
delay(2000);
digitalWrite(buzz, LOW);
delay(2000);
digitalWrite(buzz, HIGH);
delay(2000);
digitalWrite(buzz, LOW);
delay(2000);

}
else if (per >= 55) #Intermedite Level of Water--Orange Aler
lcd.setCursor(0, 1);
lcd.print("Orange Alert! ");
digitalWrite(orange, HIGH);
digitalWrite(red, LOW);
digitalWrite(green, LOW);
digitalWrite(buzz, HIGH);
delay(3000);
digitalWrite(buzz, LOW);
delay(3000);

}
else #MIN / NORMAL level of Water--Green Alert!{
lcd.setCursor(0, 1);
```

```
lcd.print("Green Alert! ");
digitalWrite(green, HIGH);
digitalWrite(orange, LOW);
digitalWrite(red, LOW);
digitalWrite(buzz, LOW);
}

delay(15000);
}
```

- After writing the code. Verify the code and then upload the code to the specific Arduino using USB Cable type A. Remember while uploading select specific board you want to upload.

Step 5.2: Writing the code in Python IDE.

- For writing python code we will be using python IDE.
- In this project we will be making two python files. One will be saved in the name of conf.py and other will be main.py.
- The purpose of making two files is to make the code understandable. Also this both python files will be usefull in sending sms and emails alerts to users.
- Now the most important part is arrived writing code in Python IDE. The full code is divided into two parts. The detailed code is given below.
- Open Python 3.7 IDE(Downloaded from the above section).
- Click on new file. Save the file in the name conf.py.
- **conf.py:** The file consists of important Api keys, Device id of Bolt IoT WiFi Module. Also it consists of important keys of Twillo and Mailgun respectively which will be further usefull in this project.

- Below is the complete structure of conf.py file. Make sure that you add the updated Bolt API key, device id and Mailgun and Twillo details respectively:

```
#twillo details for sending alert sms
SID = 'You can find SID in your Twilio Dashboard'
AUTH_TOKEN = 'You can find  on your Twilio Dashboard'
FROM_NUMBER = 'This is the no. generated by Twilio. You can
TO_NUMBER = 'This is your number. Make sure you are adding +

#bolt iot details
API_KEY = 'XXXXXXXXX'
   #This is your Bolt cloud API
Key.
DEVICE_ID = 'BOLTXXXXXXXXX' #This is the ID of your Bolt dev

#mailgun details for sending alert E-mails
MAILGUN_API_KEY = 'This is the private API key which you can
SANDBOX_URL= 'You can find this on your Mailgun Dashboard'
SENDER_EMAIL = 'test@ + SANDBOX_URL' # No need to modify thi
RECIPIENT_EMAIL = 'Enter your Email ID Here'
```

- After writing the conf.py now the last part is to write the main.py code. This code will be helpfull to send sms and email alerts when the water level crosses the threshold.
- Open the Python IDE.

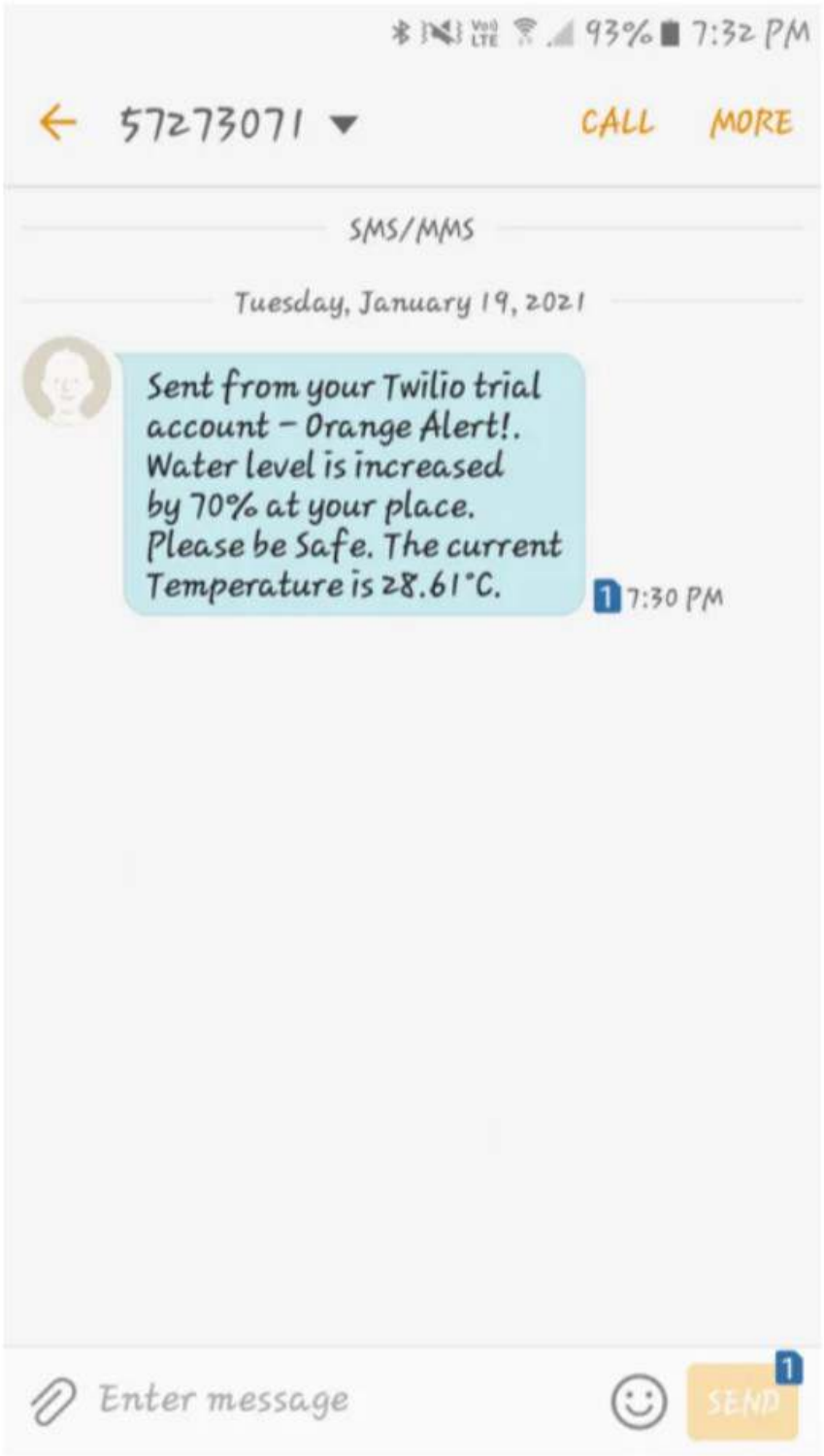
- Click on new file. Save the file in the name main.py. Save the file in the same path where conf.py is saved.
- **main.py:** This file consists of the main coding facility. Discussed earlier it will be used to send sms and emails alerts. It will be also helpfull to keep close monitor on water level to send alerts whenever required.
- Below is the complete code of main.py.

```
import conf
from boltiot import Sms, Email, Bolt
import json, time

intermediate_value = 55
max_value = 80


mybolt = Bolt(conf.API_KEY, conf.DEVICE_ID)
sms = Sms(conf.SID, conf.AUTH_TOKEN, conf.TO_NUMBER, conf.FR
mailer = Email(conf.MAILGUN_API_KEY, conf.SANDBOX_URL, conf.


def twillo_message(message):
  try:
     print("Making request to Twilio to send a SMS")
     response = sms.send_sms(message)
     print("Response received from Twilio is: " + str(respon
     print("Status of SMS at Twilio is :" + str(response.sta
  except Exception as e:
     print("Below are the details")
```

```
        print(e)

def mailgun_message(head,message_1):
  try:
      print("Making request to Mailgun to send an email")
      response = mailer.send_email(head,message_1)
      print("Response received from Mailgun is: " + response.
  except Exception as e:
      print("Below are the details")
      print(e)

while True:
    print ("Reading Water-Level Value")
    response_1 = mybolt.serialRead('10')
    response = mybolt.analogRead('A0')
    data_1 = json.loads(response_1)
    data = json.loads(response)
    Water_level = data_1['value'].rstrip()
    print("Water Level value is: " + str(Water_level) + "%")
    sensor_value = int(data['value'])
    temp = (100*sensor_value)/1024
    temp_value = round(temp,2)
    print("Temperature is: " + str(temp_value) + "°C")
    try:

        if int(Water_level) >= intermediate_value:
            message ="Orange Alert!. Water level is increase
            head="Orange Alert"
            message_1="Water level is increased by " + str(W
```

```
            twillo_message(message)
            mailgun_message(head,message_1)

        if int(Water_level) >= max_value:
            message ="Red Alert!. Water level is increased by
            head="Red Alert!"
            message_1="Water level is increased by " + str(Wa
            twillo_message(message)
            mailgun_message(head,message_1)

    except Exception as e:
        print ("Error occured: Below are the details")
        print (e)
    time.sleep(15)
```

After Successfully writing code for Arduino and Python. Now it is the time to test and demonstrate the project. Move to next section for demonstration of the project.



Mailgun Email Alert