

PHASE 2

FLOOD MONITORING SYSTEM

INNOVATION:

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

The purpose of this project is to sense the water level in riverbeds 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 when the water level reaches beyond the limit.

HARDWARE COMPONENTS -

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

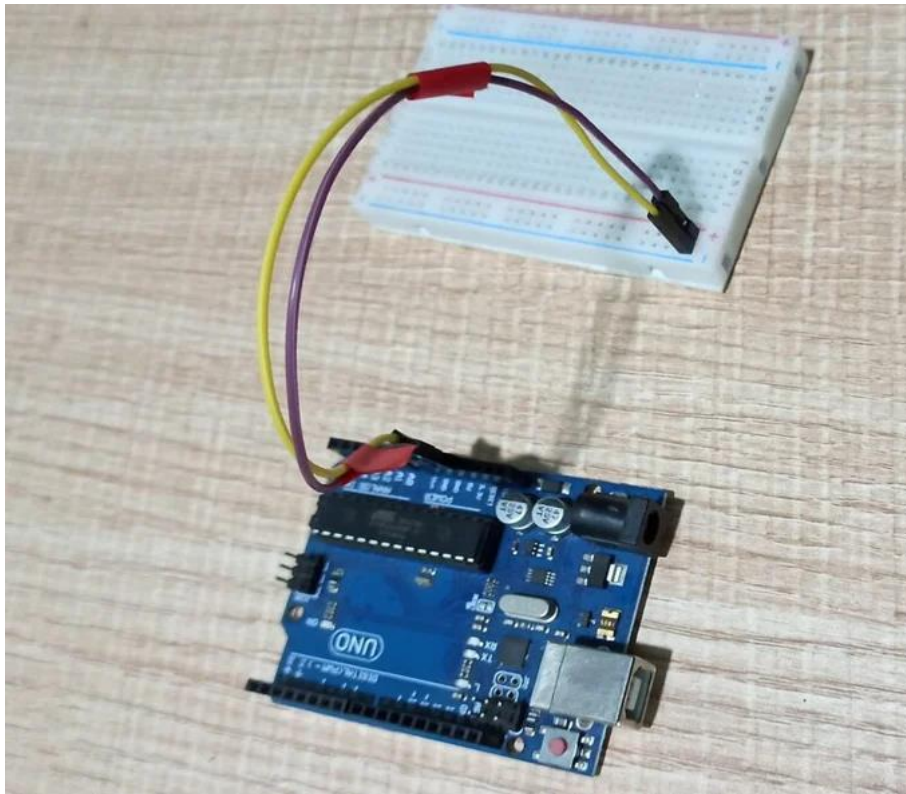
SOFTWARE COMPONENTS -

- Arduino IDE
- Python 3.7 IDLE
- Bolt IoT Cloud
- Bolt IoT Android App
- Twilio SMS Messaging API
- Mailgun EMAIL Messaging API Software components

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:

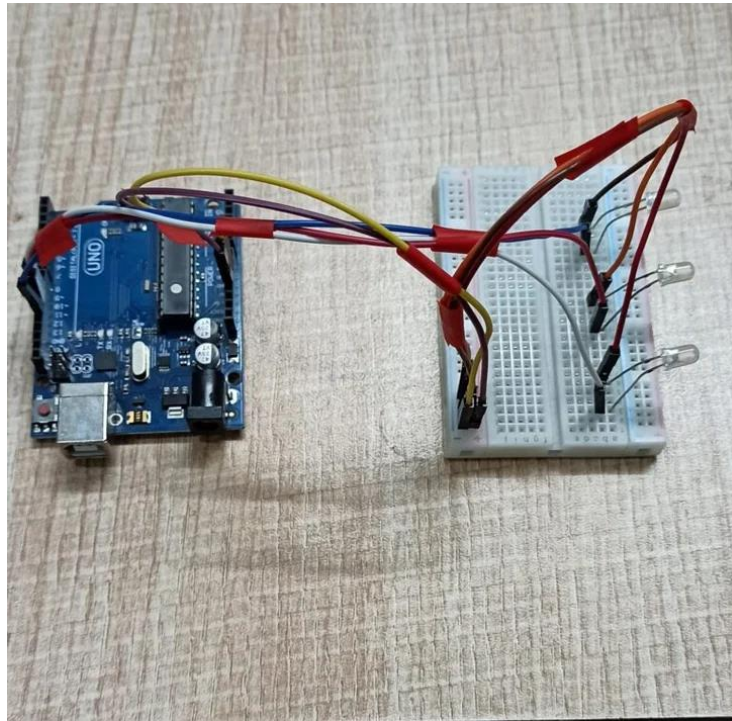
- VCC of Green Color LED to Digital Pin '10' of the Arduino.
- GND of Green Color LED to the GND of Arduino.

For Orange LED:

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

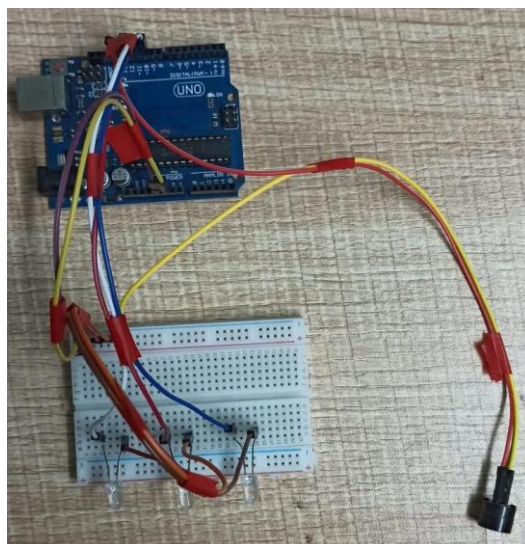
For Red LED:

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



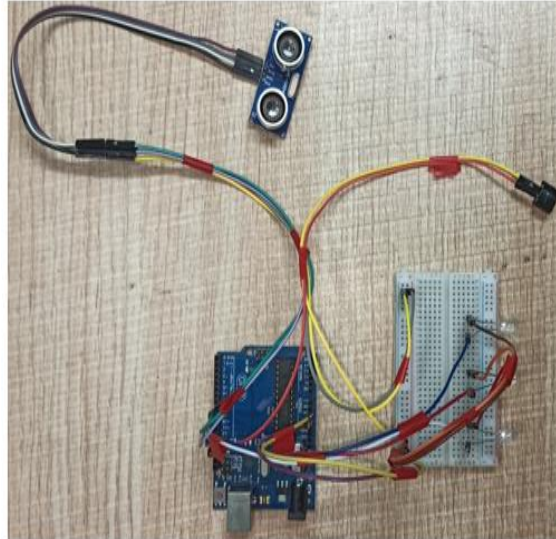
Step 3: Connecting Buzzer

- 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.

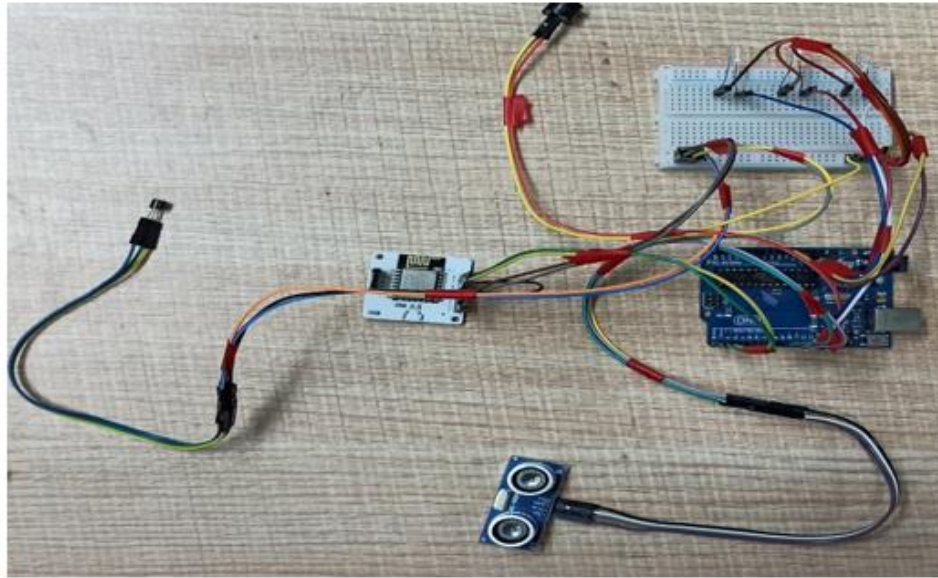


Step 5: Connecting Bolt Wi-Fi Module

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

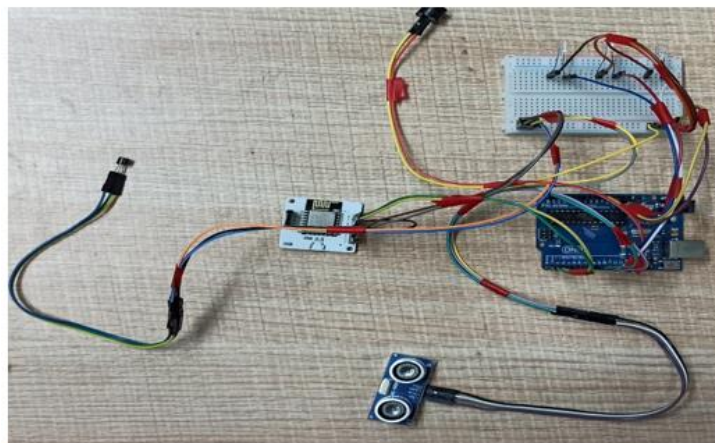
Step 6: Connecting LM35 Temperature Sensor

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

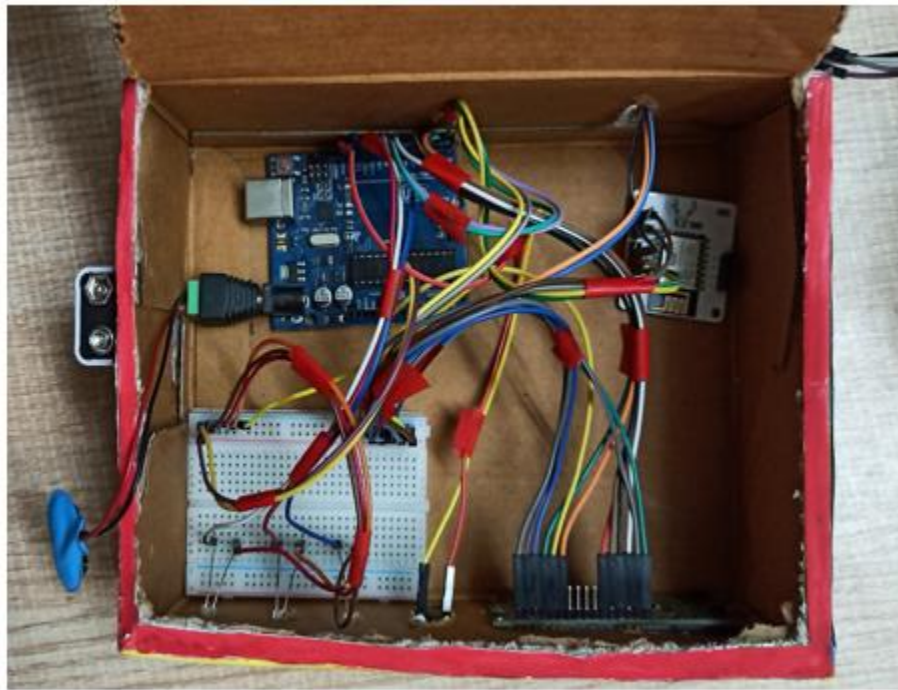


Step 7: Connecting 16×2 LCD Display

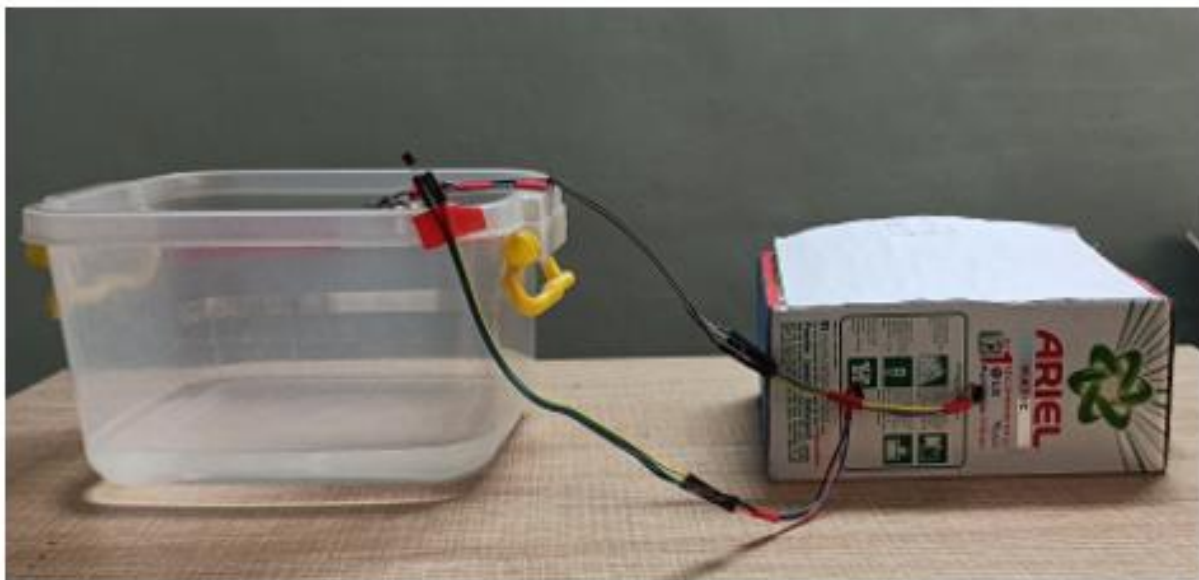
- Pin 1,3,5,16 of 16×2 LCD to GND of Arduino.
- Pin 2,15 of 16×2 LCD to 5v of Arduino.
- Pin 4 of 16×2 LCD to Digital Pin '2' of Arduino.
- Pin 6 of 16×2 LCD to Digital Pin '3' of Arduino.
- Pin 11 of 16×2 LCD to Digital Pin '4' of Arduino.
- Pin 12 of 16×2 LCD to Digital Pin '5' of Arduino.
- Pin 13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



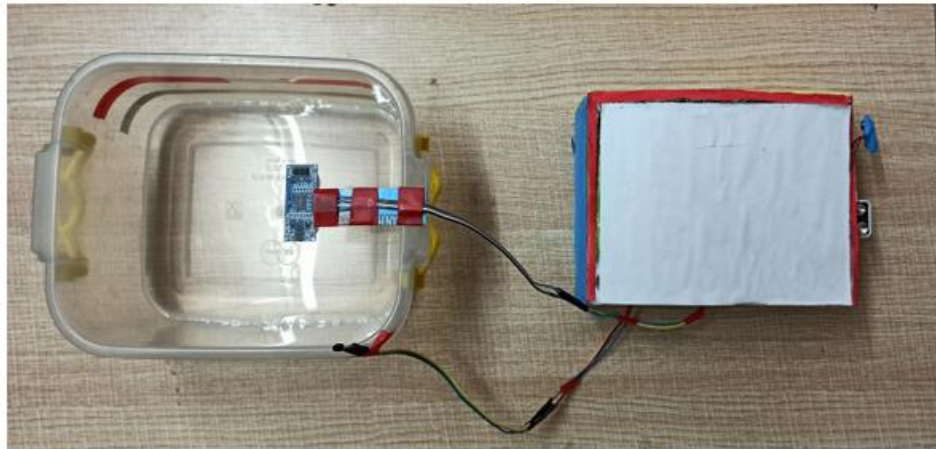
After making the hardware connection put all the hardware components in one box.



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



Also attach Ultrasonic sensor on the top of the container.

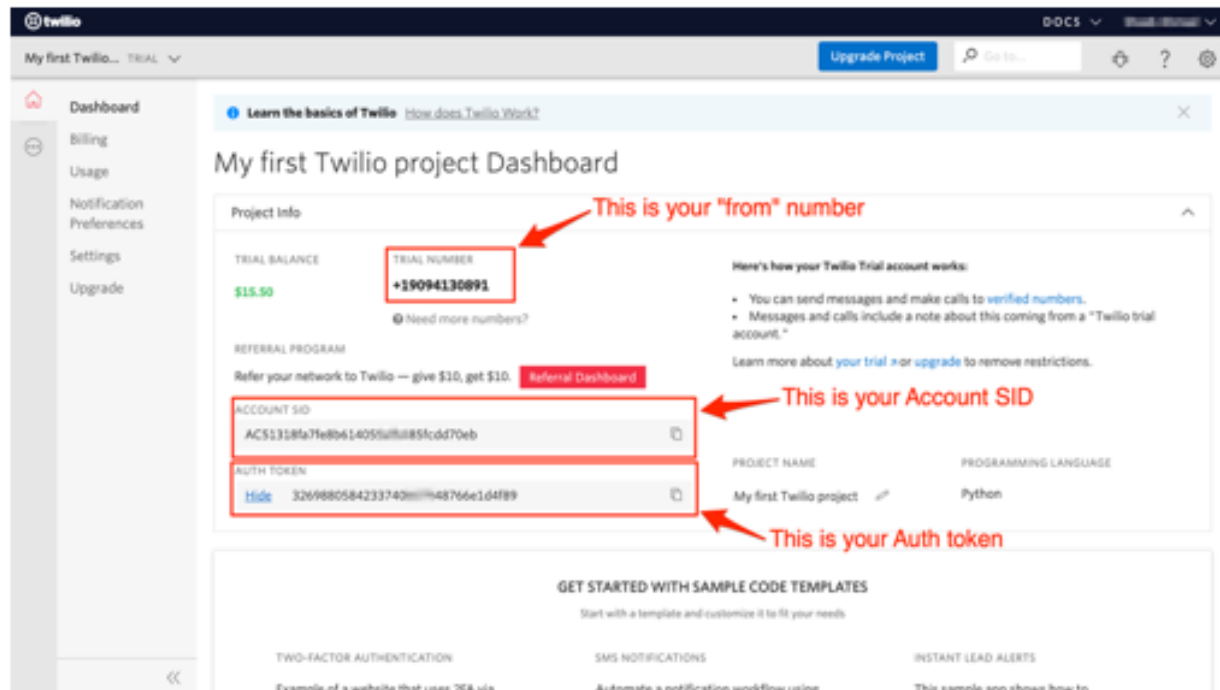


SOFTWARE PROGRAMMING:

After the successful completion of hardware setup. Now it's time to do software setup for the project. For that you have to first Download and Install Arduino IDE and Python IDE from the link given above in the software apps and online services section. Also Creating accounts on various online app services and noting down the important keys and ids. Below are all the steps given to create an account on online app services and note down the keys.

Step 1: Creating an account on Twilio and setting up Twilio for sending SMS alerts.

- Visit <https://www.twilio.com/>.
- Create an account by clicking sign up, fill required details.
- Confirm your email.
- You will need to authenticate your phone number on which the sms alerts will be notified.
- Enter the code sent to your phone
- When prompted "Do you write code?" Click yes
- Select python as your programming language
- When prompted "What is your target today?" "Choose" Twilio as a project.
- When prompted "What do you want to do first?" "Choose" Send or receive a message.
- My First Twilio Project Dashboard page will open. Now you can Edit your Project as "My Project".
- Get a trial number and save it somewhere and then choose to use this number.
- You will see the ACCOUNT SID and AUTH TOKEN.
- We will need Account Sid, Auth Token and Trial Number of these so save them somewhere.



Step 2: Creating an account on Mailgun and setting up Mailgun for sending Email alerts.

- Visit <https://www.mailgun.com/>.
- Create an account by clicking on the start sending option and by filling in details.
- Verifying your Account.
- Once you have verified your Email, after that you have added your phone number.
- After Entering your number. Click on send activation code. After some time, you will receive an OTP. Enter the OTP. Click on Enter.
- After Creating account on Mailgun go to the overview option. Click on API and Click on Python.
- After doing this you will receive an API Key and Sandbox URL. Save both these credentials somewhere you will be further using in this project.



Step 4: Creating an account on Bolt Cloud and Bolt Android App and Link the Bolt Module to Cloud.

- Visit <https://cloud.boltiot.com>.
- Create an account using Email-Id and password. (Use the same email which was used to order hardware kit also use same email for app for linking the hardware to cloud.)
- After creating an account on cloud. Then Download Bolt Android App from play store.
- Create an account on the Bolt app with the same email-Id then use the mobile hotspot for linking the Bolt Wi-Fi module to cloud.
- After successfully linking the device to the cloud then go to the cloud website. The Bolt device will show the device as online.
- Go to API section make the API as enable. Copy the API and save it somewhere.
- Also copy the Bolt Device Id which is present on Bolt IoT dashboard and save it somewhere.

Bolt Device Id

ID: BOLT46	STATUS	PRODUCT	ACTIONS
BOLT46	OFFLINE	Not Linked	   

API Key

Generate Key

☒ Enable
☐ Disable

XXXXXXXXXXXXXXXXXXXX




GENERATE NEW API KEY

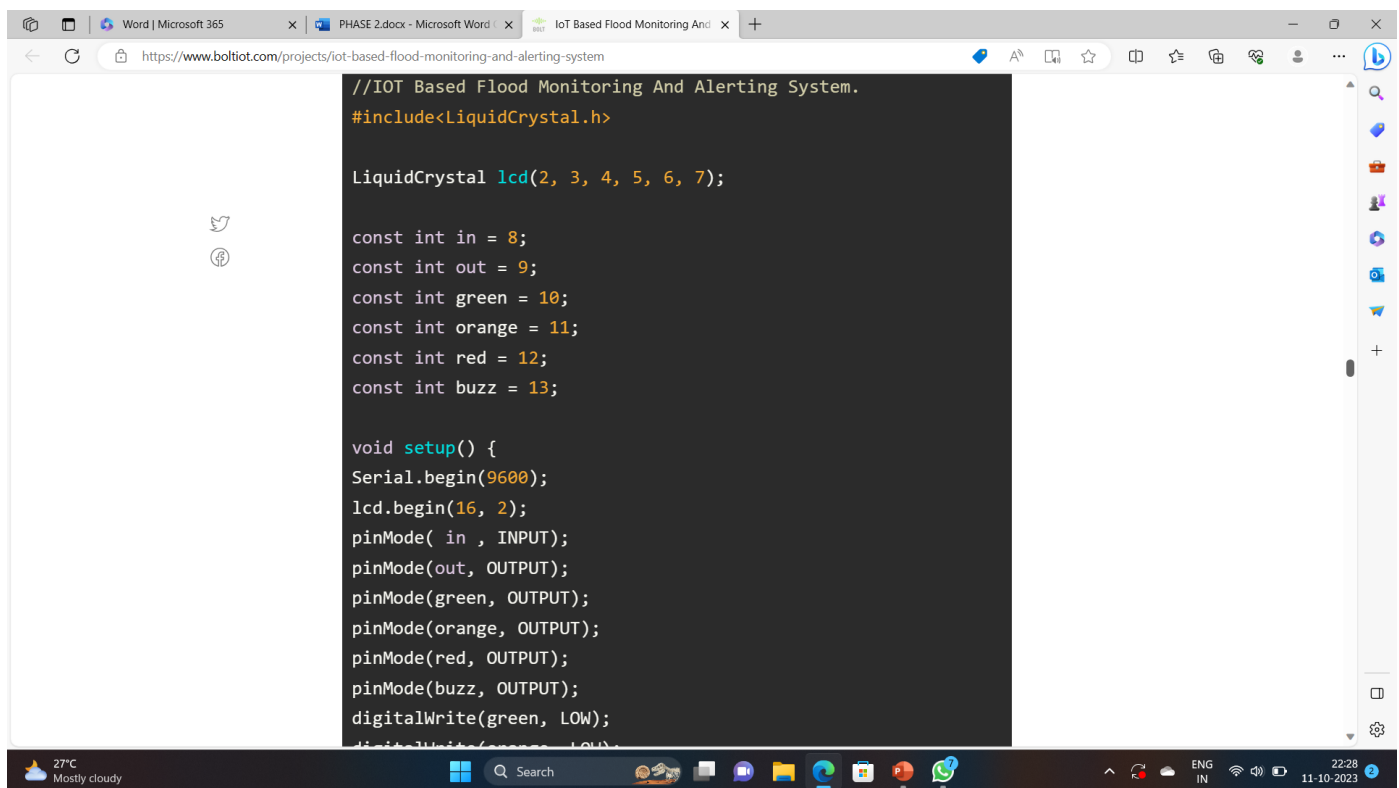
Step 5: Coding

After setting online app services and saving the keys somewhere. Now the most important thing 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 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 appropriate name to the file and add .ino extension 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.



```
//IOT Based Flood Monitoring And Alerting System.
#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);
}
```

```
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);
}
```

```
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, 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 Alert!{
  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);
}
```

```
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);

}

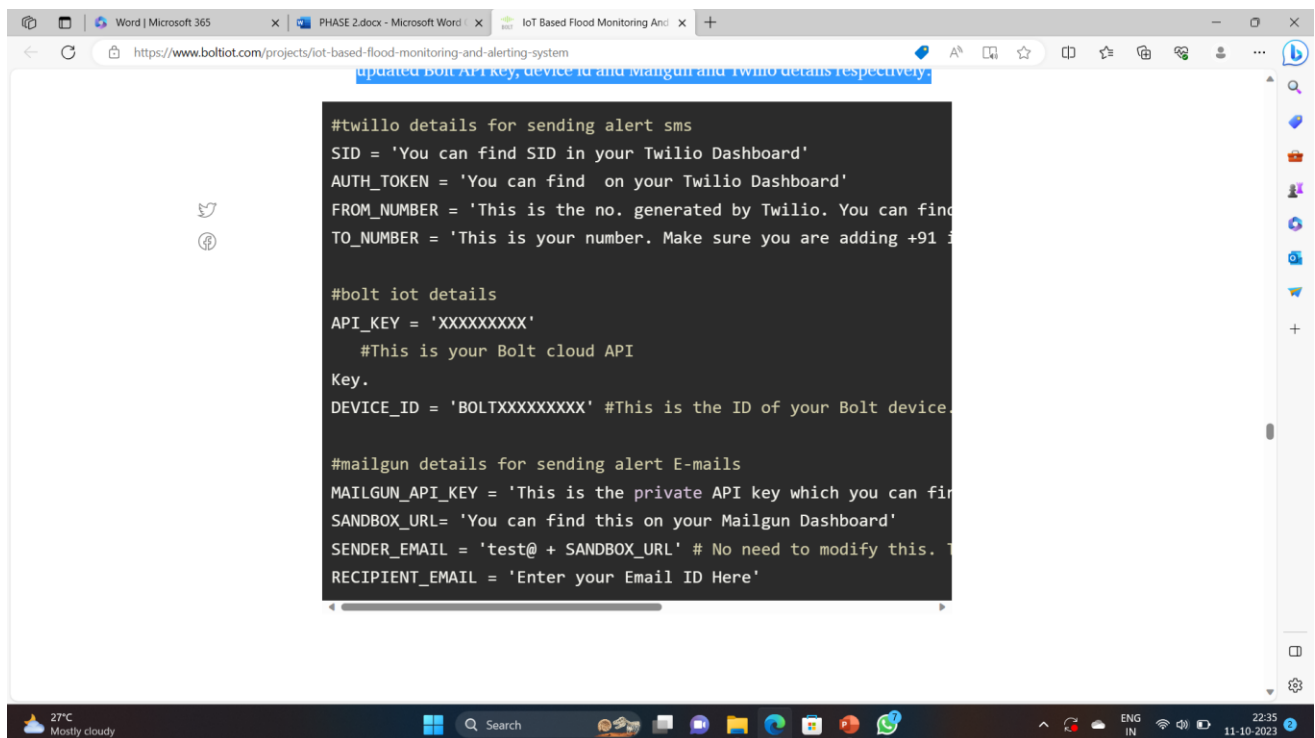
else #MIN / NORMAL level of Water--Green Alert!{
  lcd.setCursor(0, 1);
  lcd.print("Green Alert! ");
  digitalWrite(green, HIGH);
  digitalWrite(orange, LOW);
  digitalWrite(red, LOW);
  digitalWrite(buzz, LOW);
}

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 the other will be main.py.
- The purpose of making two files is to make the code understandable. Also, both these python files will be useful in sending SMS and emails alerts to users.
- Now the most important part is 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 Twilio and Mailgun respectively which will be further useful 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 Twilio details respectively:



The screenshot shows a web browser window with a code editor. The URL bar shows 'https://www.boltt.com/projects/iot-based-flood-monitoring-and-alerting-system'. The code editor contains the following Python code:

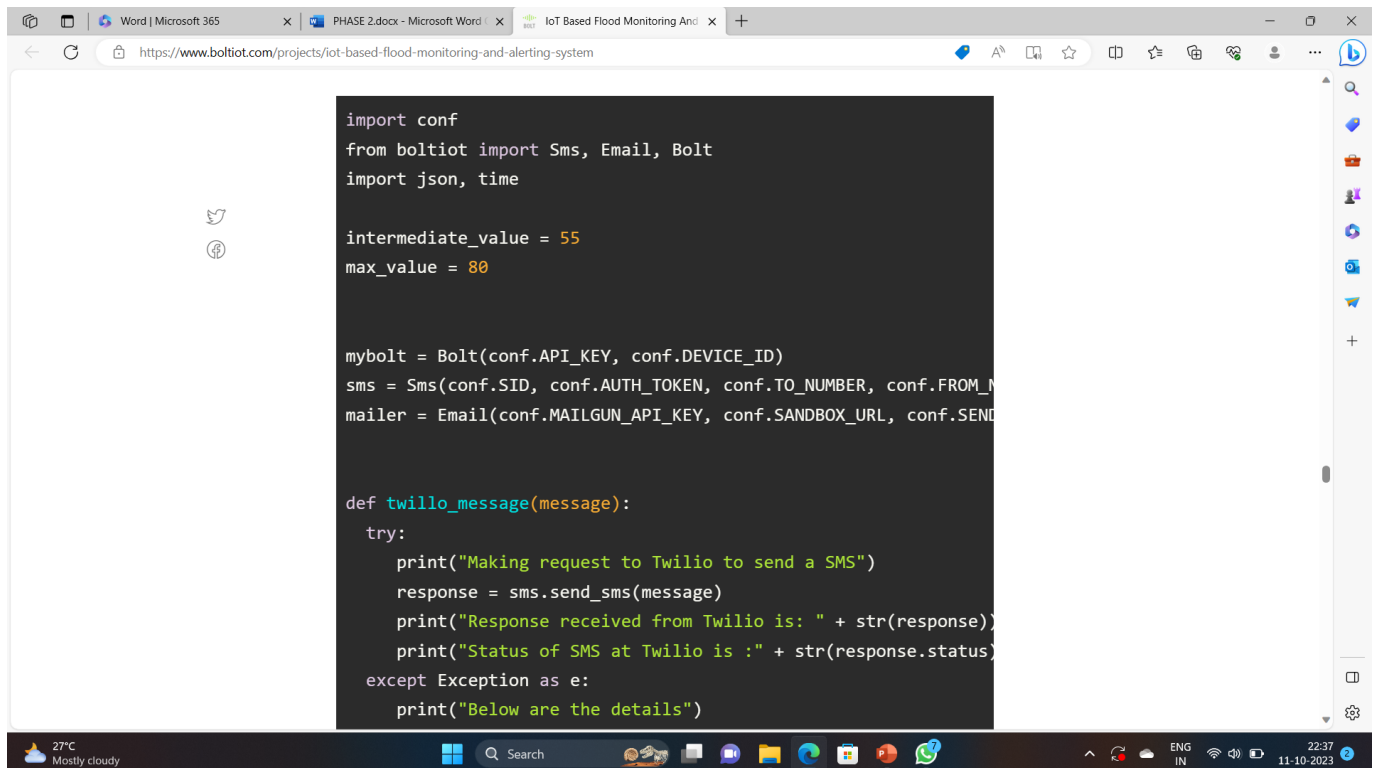
```
#twilio details for sending alert sms
SID = 'You can find SID in your Twilio Dashboard'
AUTH_TOKEN = 'You can find on your Twilio Dashboard'
FROM_NUMBER = 'This is the no. generated by Twilio. You can find'
TO_NUMBER = 'This is your number. Make sure you are adding +91 i'

#bolt iot details
API_KEY = 'XXXXXXXXXX'
#This is your Bolt cloud API
Key.
DEVICE_ID = 'BOLTXXXXXXXXXX' #This is the ID of your Bolt device.

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

The code is displayed in a dark-themed editor with syntax highlighting. The browser window also shows several open tabs and a taskbar at the bottom with various application icons and system status information.

- After writing the `conf.py` now the last part is to write the `main.py` code. This code will be helpful 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 helpful to keep a close monitor of water levels to send alerts whenever required.
- Below is the complete code of `main.py`.



The screenshot shows a web browser window with the URL `https://www.bolttiot.com/projects/iot-based-flood-monitoring-and-alerting-system`. The page displays a code editor with the following Python code for `main.py`:

```
import conf
from bolttiot 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.FROM_NUMBER)
mailer = Email(conf.MAILGUN_API_KEY, conf.SANDBOX_URL, conf.SENDER_DOMAIN)

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

The browser's taskbar at the bottom shows the system date and time as 11-10-2023, 22:37, and the language set to ENG IN.

```
try:
    print("Making request to Mailgun to send an email")
    response = mailer.send_email(head,message_1)
    print("Response received from Mailgun is: " + response.text)
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:
```

```
print("Temperature is: " + str(temp_value) + "°C")
try:
    if int(Water_level) >= intermediate_value:
        message = "Orange Alert!. Water level is increased by " + str(Water_level)
        head="Orange Alert"
        message_1="Water level is increased by " + str(Water_level)
        twillo_message(message)
        mailgun_message(head,message_1)

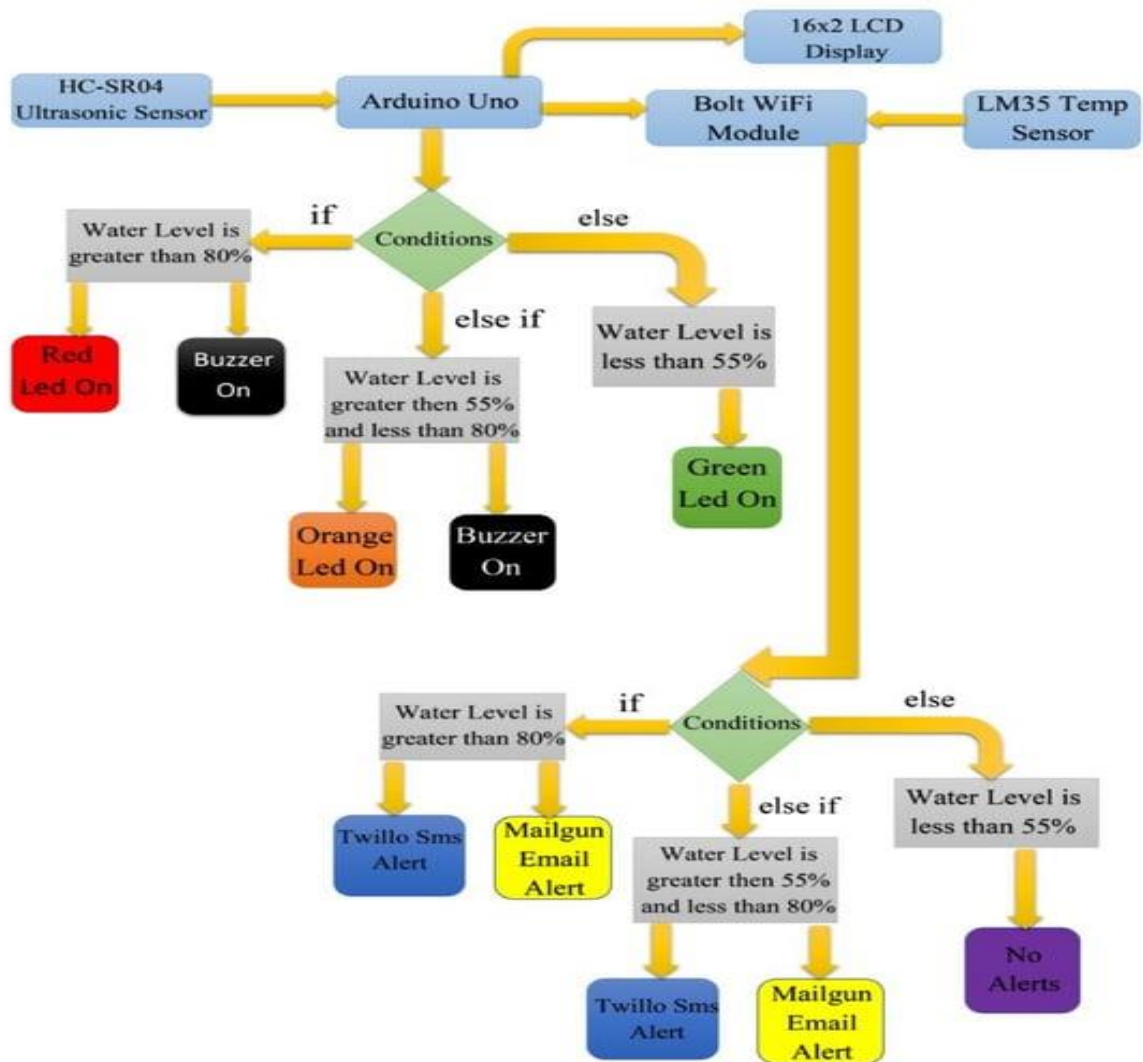
    if int(Water_level) >= max_value:
        message = "Red Alert!. Water level is increased by " + str(Water_level)
        head="Red Alert!"
        message_1="Water level is increased by " + str(Water_level)
        twillo_message(message)
        mailgun_message(head,message_1)

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 the next section for a demonstration of the project.

Demonstration

Let's First have a look at the workflow of this project.



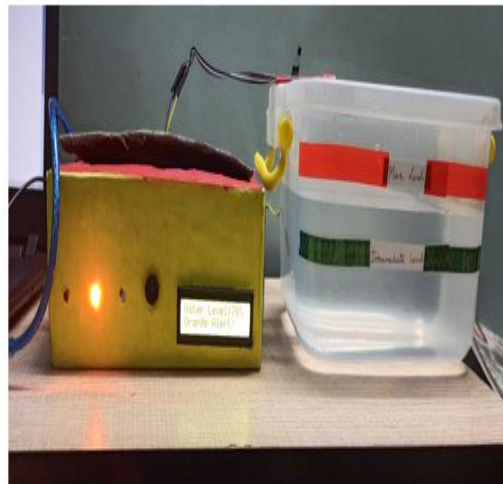
For doing the practical demonstration. First connect the USB cable type-B to the Laptop's USB slot for power supply. Also simultaneously run the python program (i.e Main.py). Firstly, the ultrasonic sensor will sense the water level in distance and then the Arduino program will help to convert it into percentage. Also, the sensed water level will be displayed on Lcd display (In

Percentage) along with zone/area the water level is present. The full water tank/container is divided into 3 zones i.e., Green, Orange and Red. Now let's look into each zone.

- When the water level is at Min/Normal level. That resembles 'Green Alert'. This means that the water is at a normal position and there is no sign of flood condition. Also, green lead will glow, and it will also show green alert in Lcd display with water level.

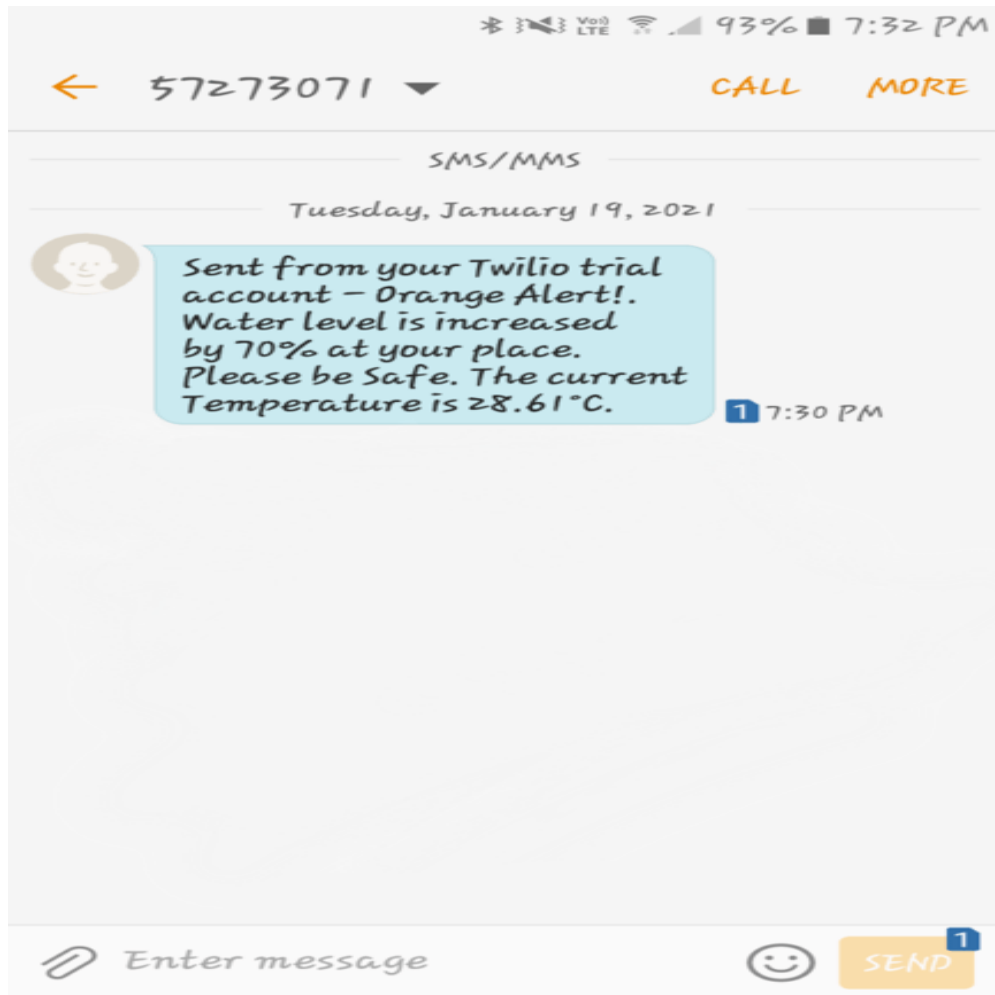


- When the water level crosses the Intermediate level. That resembles 'Orange Alert'. This means that water has crossed the 55% mark and there can be chances of flood condition at that place. With the increase in water level the system sends Sms and Email alerts to the authority or registered user from Twillo and Mailgun Services respectively. Also, orange lead will glow, and buzzer will buzz. It will also show orange alert in Lcd display.



Also, Sms and Email is sent to registered user with proper message and current temperature of that place.

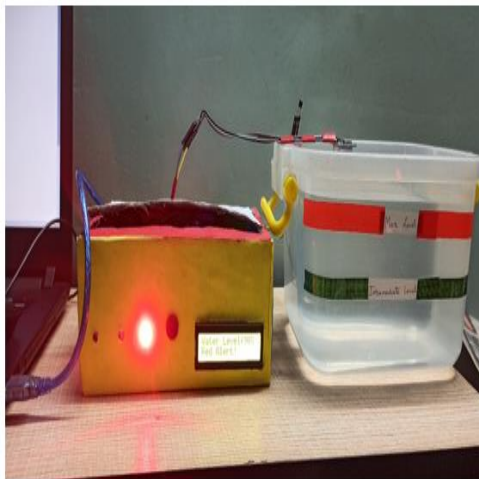
Twillo Sms Alert:



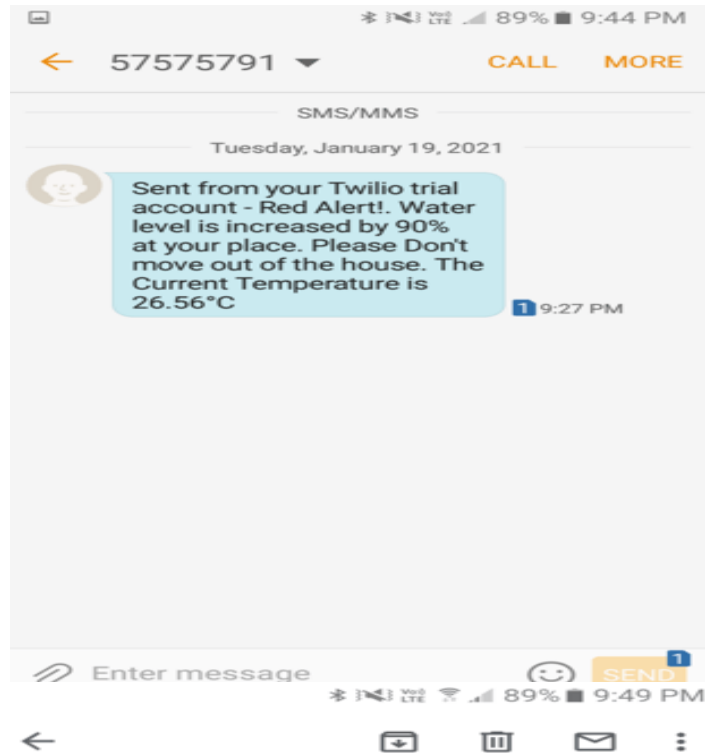
Mailgun Email Alert



- When the water level crosses the Max Level. That resembles 'Red Alert'. This means that the water level has exceeded 80% and a flood situation has occurred at that place. With the increase in water level the system sends Sms and Email alerts to the authority or registered user from Twillo and Mailgun Services respectively. Also red lead will glow, and buzzer will buzz two times. It will also show red alert in Lcd display.



Also Sms and Email is sent to registered user with proper message and current temperature of that place.



Red Alert!

Inbox



test@sandboxd116c6753d29... 7:28 PM

Water level is increased by 80% at your place. Please Don't move out of the house.



test@sandboxd116c6753d29... 7:31 PM

Water level is increased by 80% at your place. Please Don't move out of the house.



test@sandboxd116... 9:27 PM

to me ▾



Water level is increased by 90% at your place. Please Don't move out of the house. The Current Temperature is 26.56°C.

↩ Reply

↩↩ Reply all

➦ Forward