

PHASE 4

THE FLOOD MONITORING SYSTEM

PHASE 4: DEVELOPMENT PART 2

Performing different activities like feature engineering, model training, evaluation etc...

It sounds like you're interested in a flood monitoring system that utilizes IoT (Internet of Things) technology. These systems typically involve sensors placed in flood-prone areas to collect data and transmit it to a central software program for analysis and monitoring.

Certainly! A Flood Monitoring System using IoT typically involves the integration of sensors, communication devices, and a central software program to monitor and respond to flood-related data. Here's a simplified outline of how such a project might be structured:

1. Hardware Components:

- Sensors: These are placed in areas prone to flooding. They could be water level sensors, rain gauges, or even weather stations.
- Microcontrollers: These are used to interface with the sensors, process the data, and transmit it to a central server. Common platforms include Arduino, Raspberry Pi, or specialized IoT development boards.

2. Communication:

- IoT Protocols: MQTT, CoAP, HTTP, or other IoT protocols can be used to transmit data from the sensors to the central server.
- Connectivity: The system may use Wi-Fi, cellular networks, or LoRa (Long Range) communication for transmitting data, depending on the deployment location.

3. Central Server and Software:

- Database: Data from the sensors is stored in a database for analysis and historical reference.

- **Monitoring Software:** This software processes incoming data, performs analytics, and triggers alerts or notifications based on predefined thresholds.
- **User Interface:** A web-based or mobile application allows users to interact with and monitor the system. It might include features like real-time dashboards, historical data visualization, and alert management.

4. Alerts and Notifications:

- **Thresholds:** The system is configured to trigger alerts when specific conditions (e.g., water level surpasses a critical level) are met.
- **Notification Channels:** Alerts can be sent via email, SMS, or push notifications to relevant parties.

5. Data Analytics and Reporting:

- **Data Analysis:** The system may include tools for analyzing trends, patterns, and historical data to support decision-making.
- **Reporting:** Automatic generation of reports for stakeholders or authorities can be a crucial feature.

6. Integration with External Systems:

- **GIS (Geographic Information System):** Integration with mapping tools for visual representation of flood-prone areas.
- **Emergency Services Integration:** Integration with emergency response services for immediate action in case of a flood event.

7. Power Management:

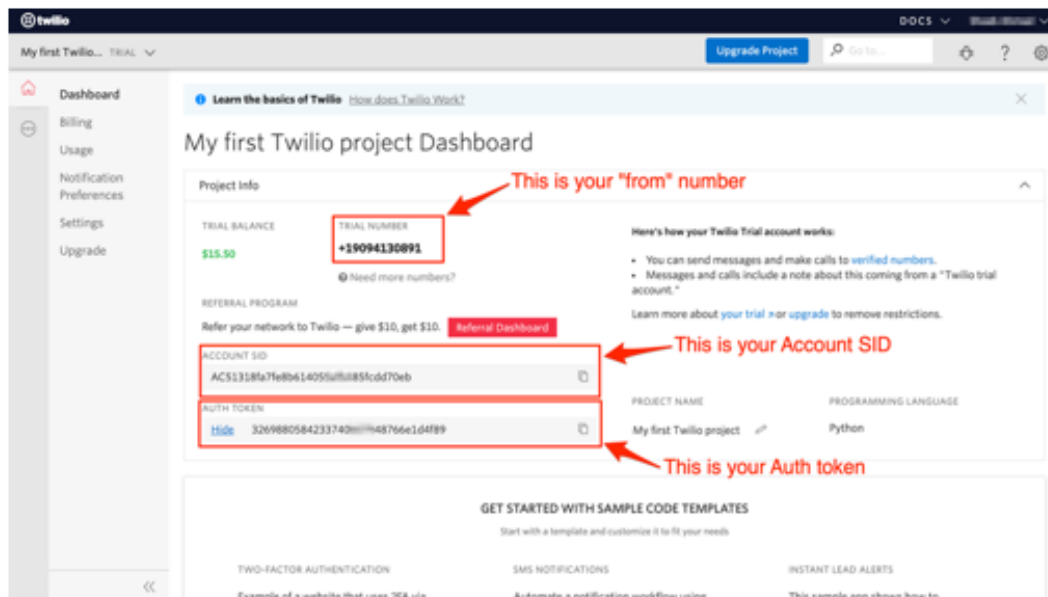
- Depending on the deployment location, power management solutions like solar panels or battery backups might be necessary to ensure uninterrupted operation.

Software Programming

After the successful completion of hardware setup. Now it's time to do software setup for the project. For that you must 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.

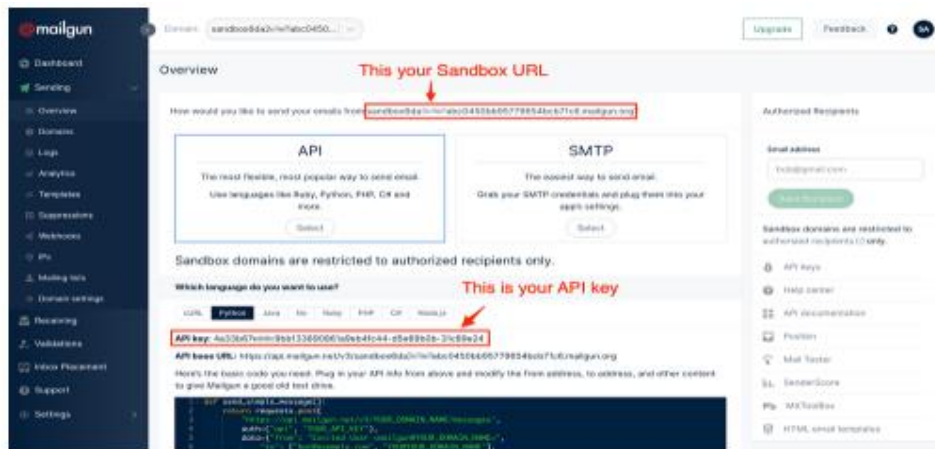
1. Visit <https://www.twilio.com/>.
2. Create an account by clicking sign up, fill required details.
3. Confirm your email.
4. You will need to authenticate your phone number on which the sms alerts will be notified.
5. Enter the code sent to your phone
6. When prompted "Do you write code?" Click yes
7. Select python as your programming language
8. When prompted "What is your target today?" "Choose" Twilio as a project.
9. When prompted "What do you want to do first?" "Choose" Send or receive a message.
10. My First Twilio Project Dashboard page will open. Now you can Edit your Project as "My Project".
11. Get a trial number and save it somewhere and then choose to use this number.
12. You will see the ACCOUNT SID and AUTH TOKEN.
13. 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.

1. Visit <https://www.mailgun.com/>.

2. Create an account by clicking on the start sending option and by filling in details.
3. Verifying your Account.
4. Once you have verified your Email, after that you have added your phone number.
5. After Entering your number. Click on send activation code. After some time, you will receive one OTP. Enter the OTP. Click on Enter.
6. After Creating account on Mailgun go to the overview option. Click on API and Click on Python.
7. After doing this you will receive 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.

1. Visit <https://cloud.boltiot.com>.
2. 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.)
3. After creating an account on cloud. Then Download Bolt Android App from playstore.
4. 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.
5. After successfully linking the device to the cloud then go to the cloud website. The Bolt device will show the device as online.
6. Go to API section make the API as enable. Copy the API and save it somewhere.
7. 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

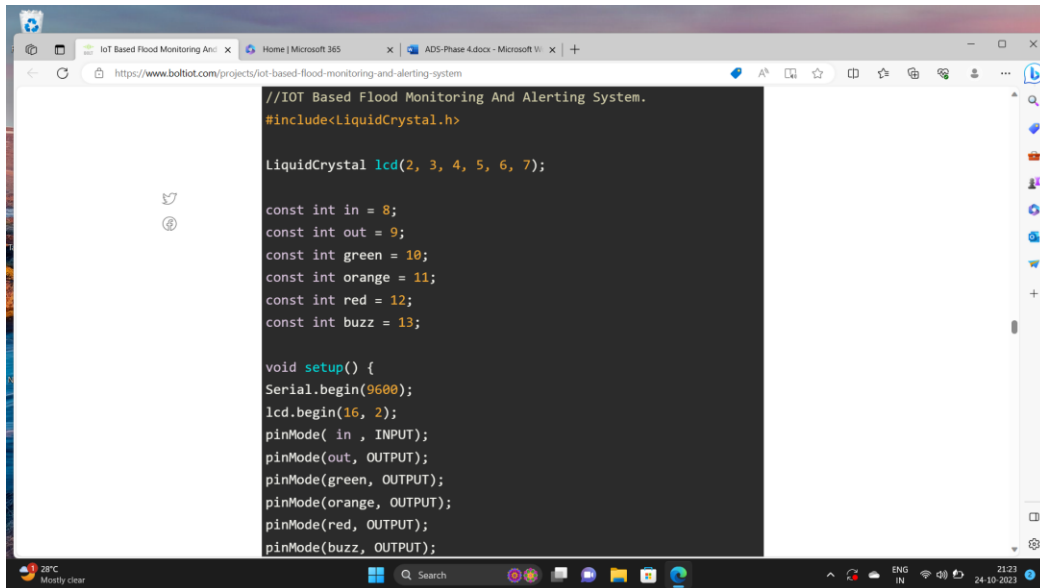
Step 5: Coding

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

1. Open the Arduino IDE (Downloaded from the above section).
2. 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.
3. 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.

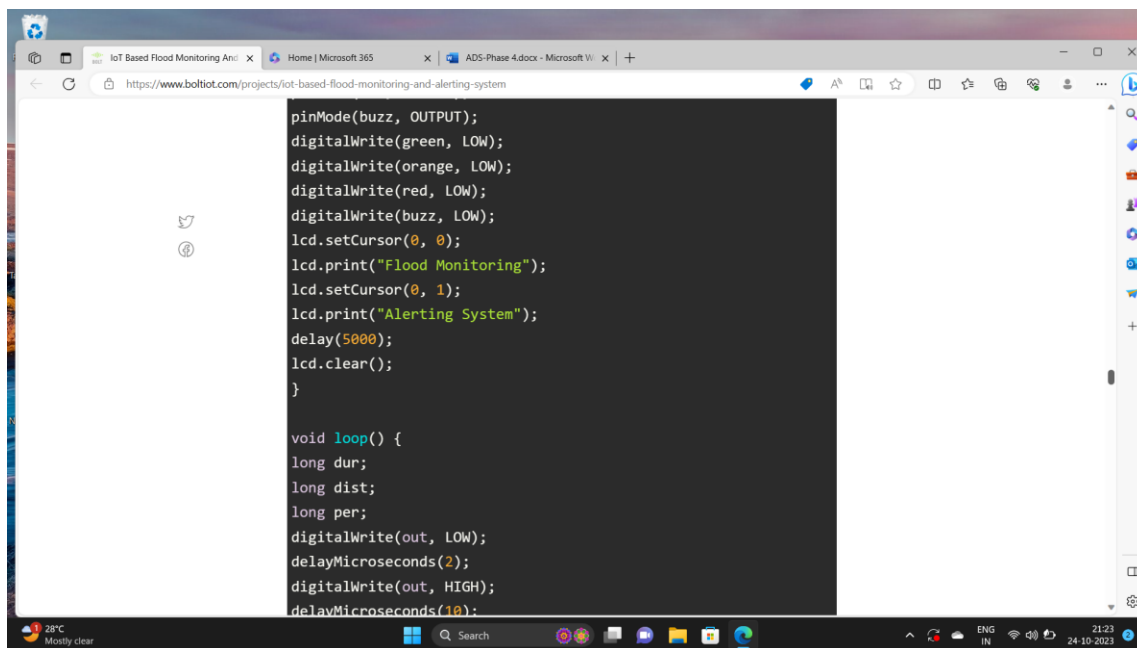


```
//IOT Based Flood Monitoring And Alerting System.
#include<LiquidCrystal.h>

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

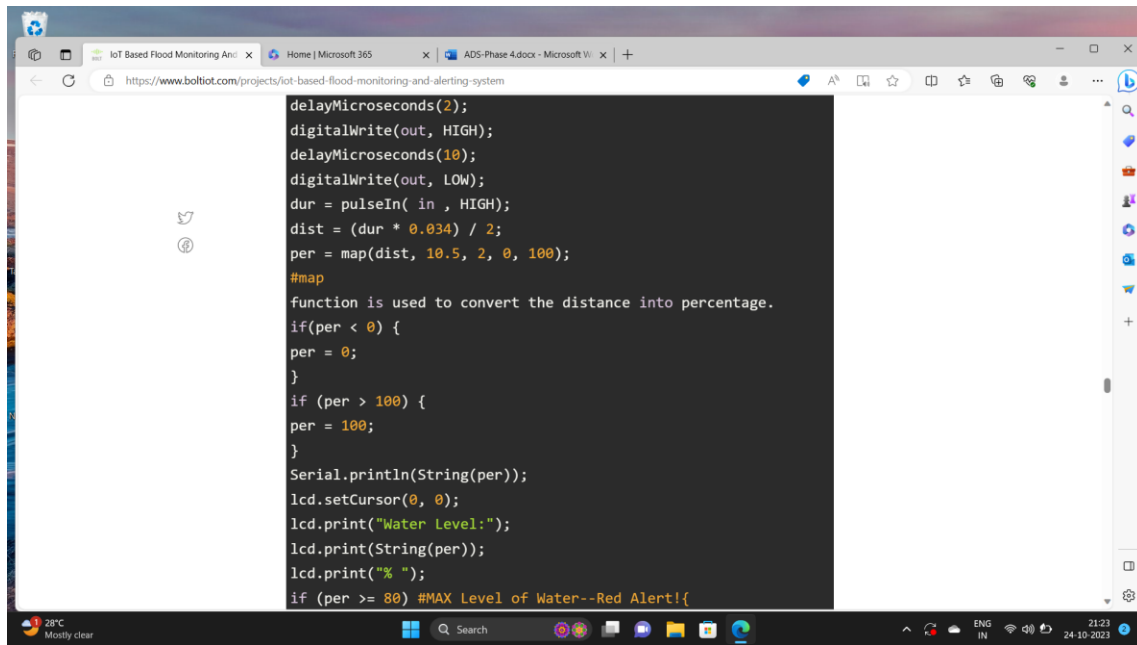
const int in = 8;
const int out = 9;
const int green = 10;
const int orange = 11;
const int red = 12;
const int buzz = 13;

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);
}
```



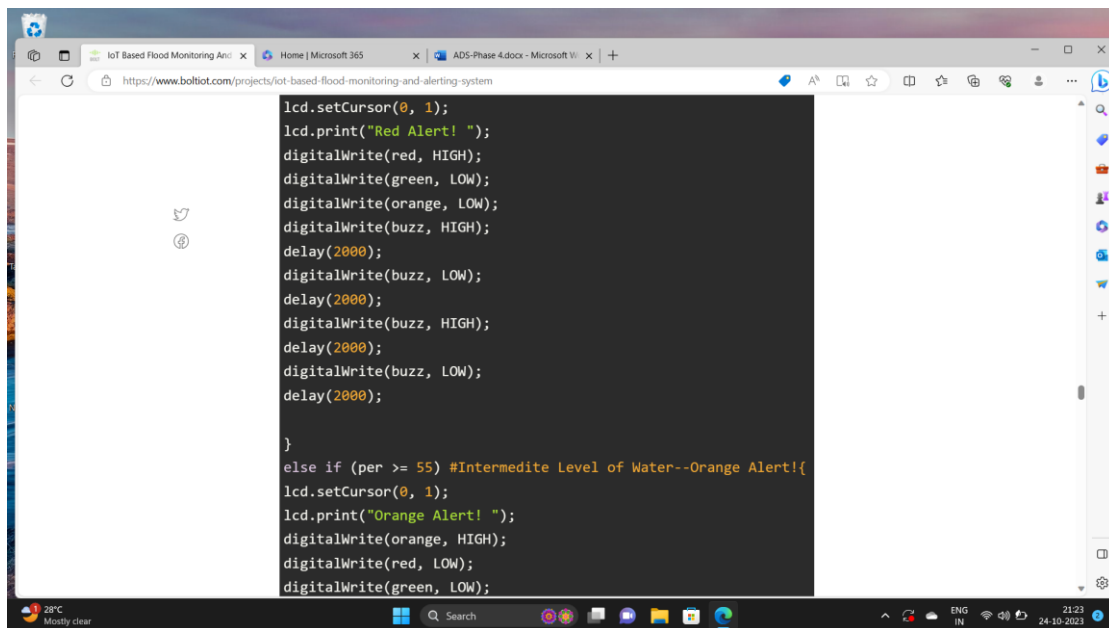
```
pinMode(buzz, OUTPUT);
digitalWrite(green, LOW);
digitalWrite(orange, LOW);
digitalWrite(red, LOW);
digitalWrite(buzz, LOW);
lcd.setCursor(0, 0);
lcd.print("Flood Monitoring");
lcd.setCursor(0, 1);
lcd.print("Alerting System");
delay(5000);
lcd.clear();
}

void loop() {
  long dur;
  long dist;
  long per;
  digitalWrite(out, LOW);
  delayMicroseconds(2);
  digitalWrite(out, HIGH);
  delayMicroseconds(10);
}
```



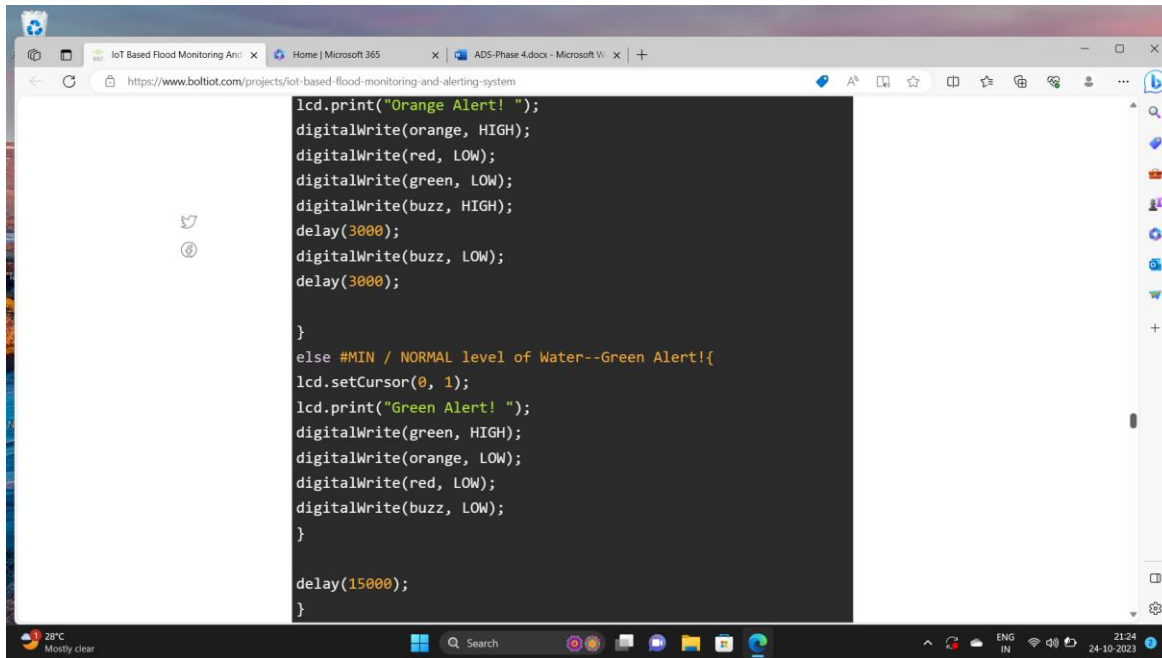
The screenshot shows a web browser window with the URL <https://www.bolttiot.com/projects/iot-based-flood-monitoring-and-alerting-system>. The page displays Arduino code for a flood monitoring system. The code includes a map function to convert distance into a percentage and a conditional statement to trigger a red alert when the percentage is greater than or equal to 80.

```
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!{
```



The screenshot shows the continuation of the Arduino code from the previous image. It includes a conditional statement to trigger an orange alert when the percentage is greater than or equal to 55. The code also includes a series of digitalWrite and delay statements to control the output pins and a buzzer.

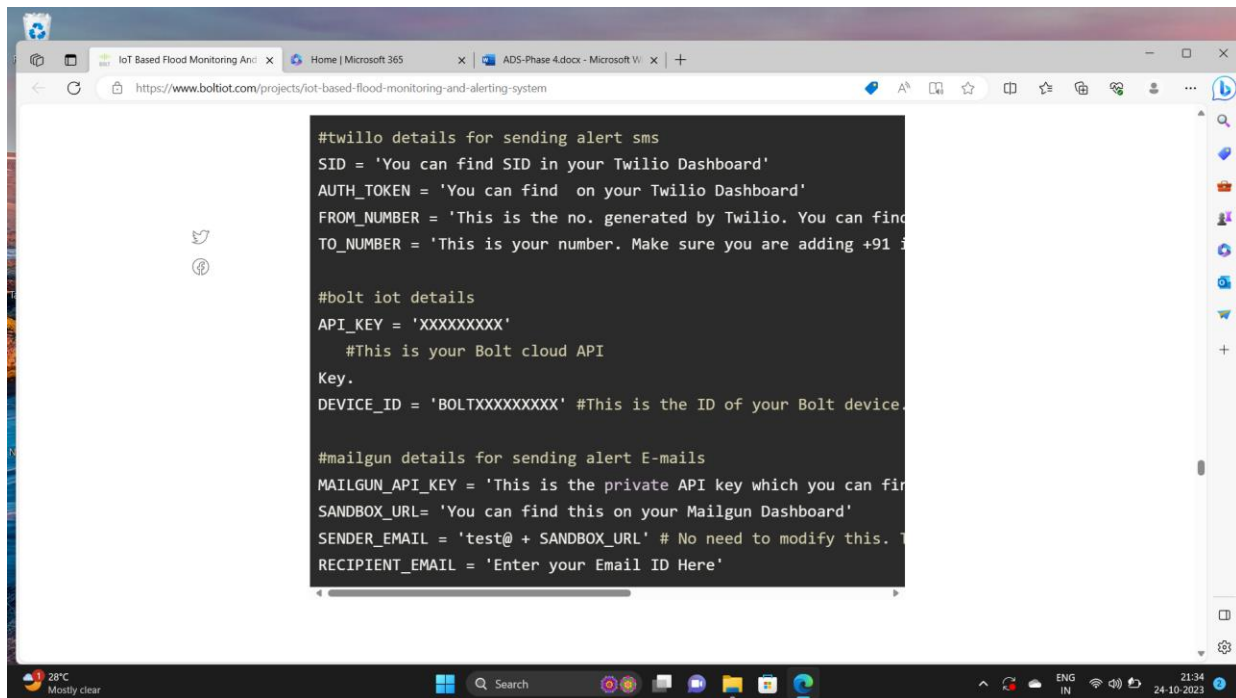
```
lcd.setCursor(0, 1);
lcd.print("Red Alert! ");
digitalWrite(red, HIGH);
digitalWrite(green, LOW);
digitalWrite(orange, LOW);
digitalWrite(buzz, HIGH);
delay(2000);
digitalWrite(buzz, LOW);
delay(2000);
digitalWrite(buzz, HIGH);
delay(2000);
digitalWrite(buzz, LOW);
delay(2000);
}
else if (per >= 55) #Intermedite Level of Water--Orange Alert!{
lcd.setCursor(0, 1);
lcd.print("Orange Alert! ");
digitalWrite(orange, HIGH);
digitalWrite(red, LOW);
digitalWrite(green, LOW);
```



- 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 Wi-Fi Module. Also, it consists of important keys of Twillo 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 Twillo details respectively:



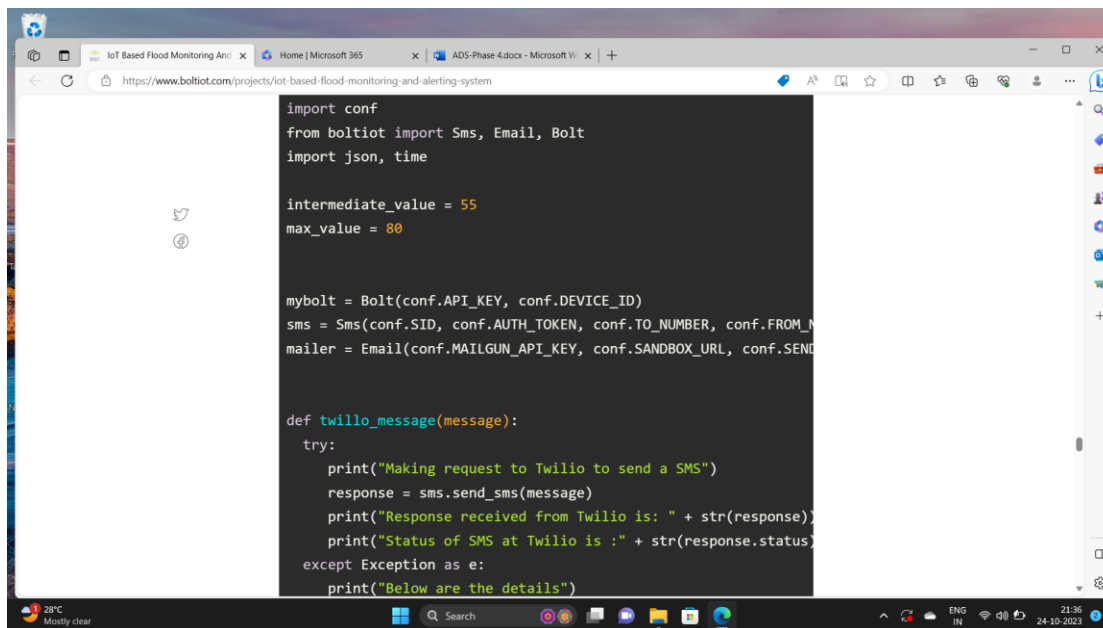
The screenshot shows a web browser window with the URL <https://www.bolttiot.com/projects/iot-based-flood-monitoring-and-alerting-system>. The browser has several tabs open, including 'IoT Based Flood Monitoring And...', 'Home | Microsoft 365', and 'ADS-Phase 4.docx - Microsoft W...'. The main content area displays a code editor with the following configuration details:

```
#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 3'

#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'
```

- 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 code editor displays the following Python code:

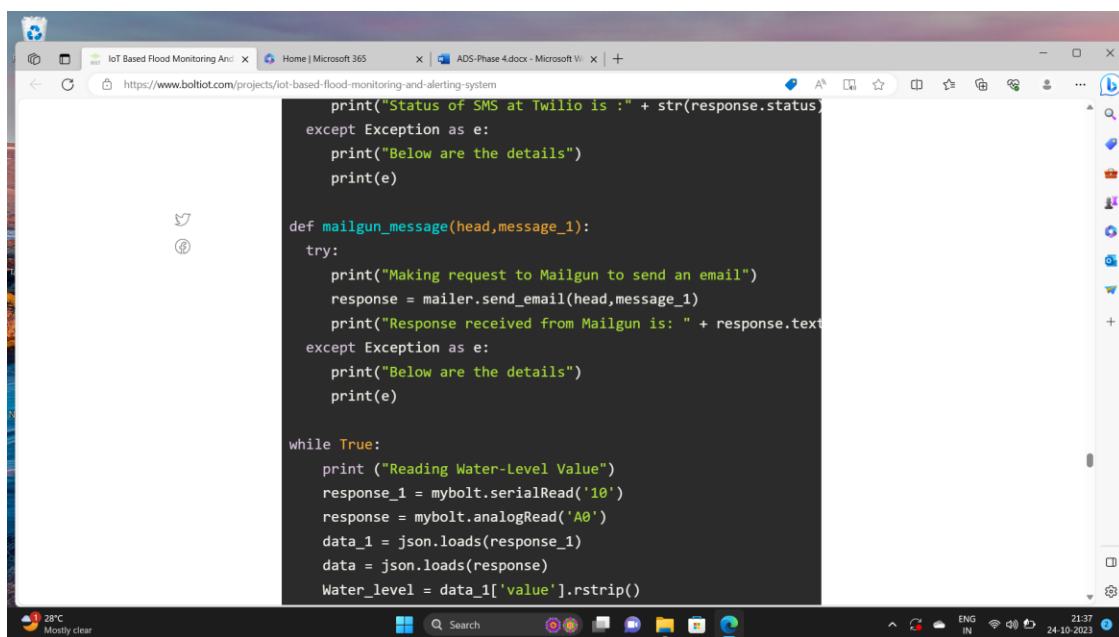
```
import conf
from bolttiot import Sms, Email, Bolt
import json, time

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.SEND_DOMAIN)

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(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 clock as 21:36 on 24-10-2023, with a weather widget indicating 28°C and 'Mostly clear'.



The screenshot shows the same web browser window with the URL <https://www.bolttiot.com/projects/iot-based-flood-monitoring-and-alerting-system>. The code editor displays the following Python code:

```
print("Status of SMS at Twilio is: " + str(response.status))
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.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()
```

The browser's taskbar at the bottom shows the system clock as 21:37 on 24-10-2023, with a weather widget indicating 28°C and 'Mostly clear'.

```
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:

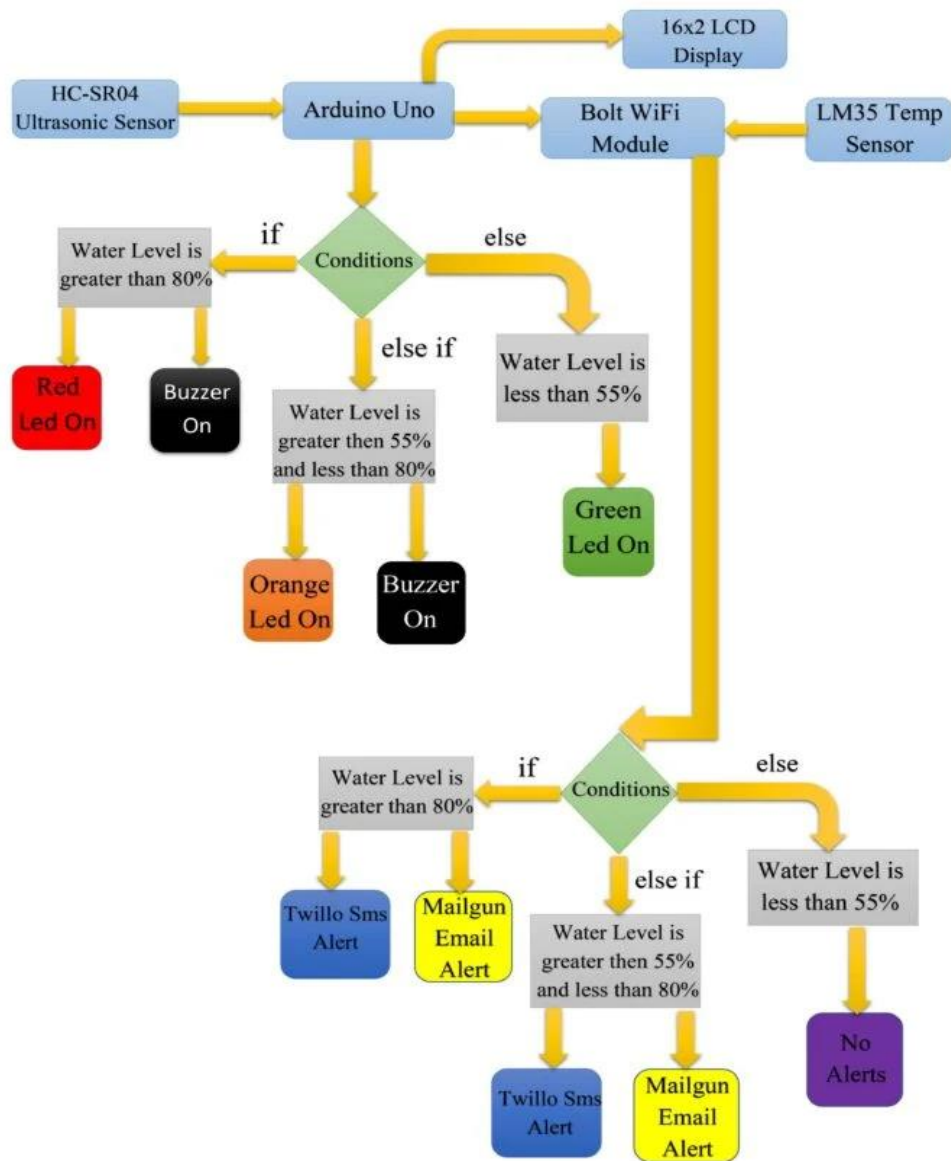
    if int(Water_level) >= intermediate_value:
        message = "Orange Alert!. Water level is increased by "
        head="Orange Alert"
        message_1="Water level is increased by " + str(Water_level)
        twilio_message(message)
        mailgun_message(head,message_1)

    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(Water_level)
```

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 investigate 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.

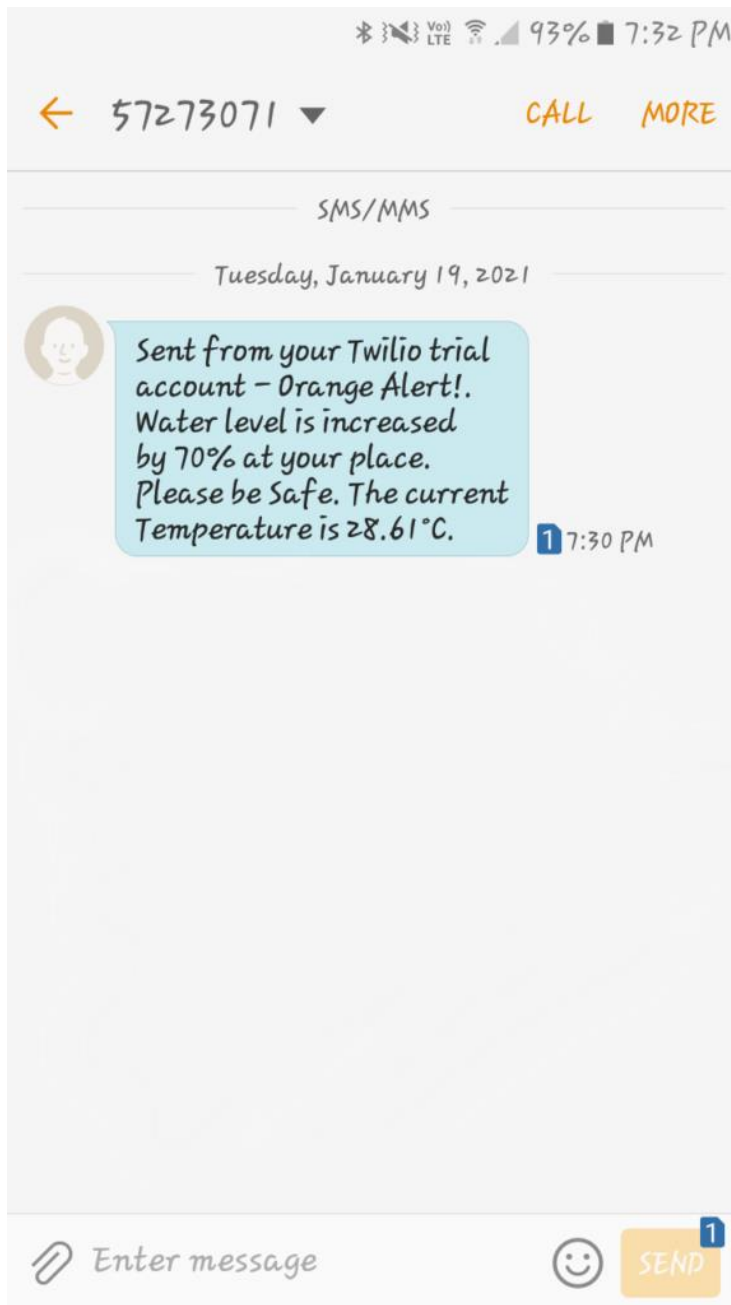


When the water level crosses the Intermediate level. That resembles 'Orange Alert'. This means that water has crossed the 55% mark and there can be chances of flood condition at that place. With the increase in water level the system sends Sms and Email alerts to the authority or registered user from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

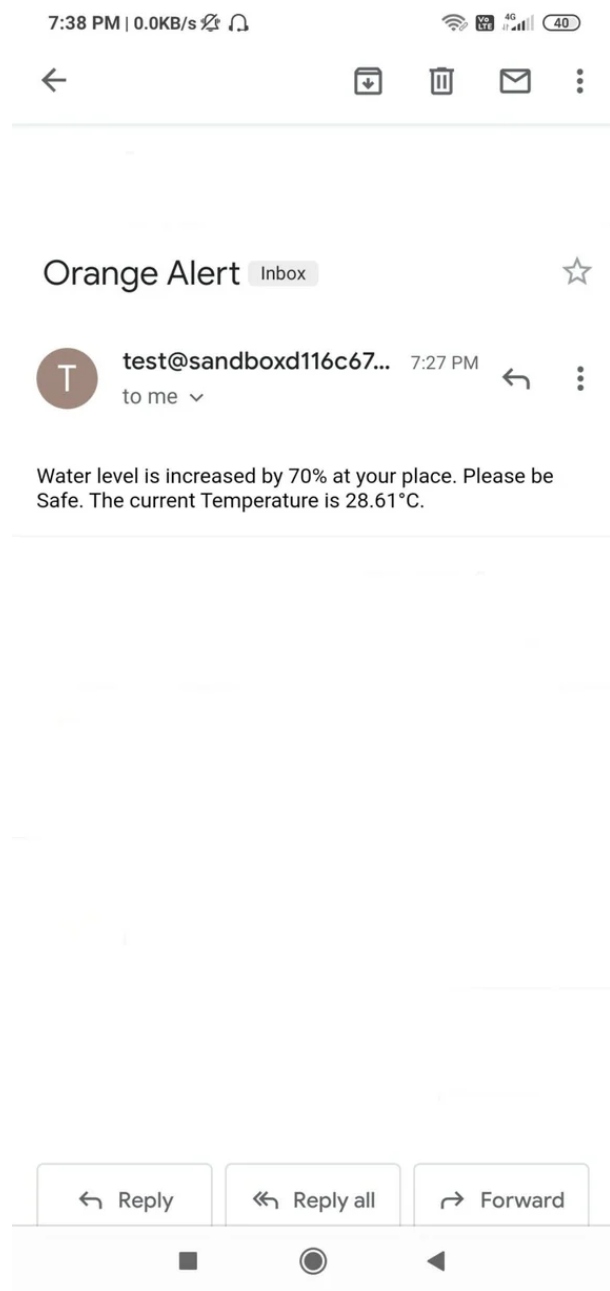


Also, Sms and Email is sent to registered user with proper message and current temperature of that place.

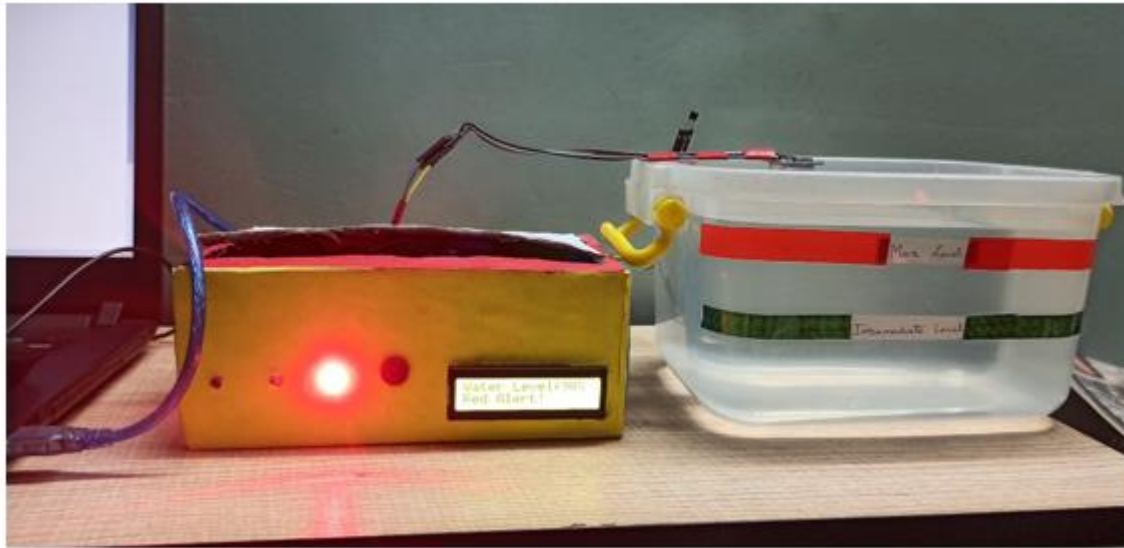
Twilio Sms Alert:



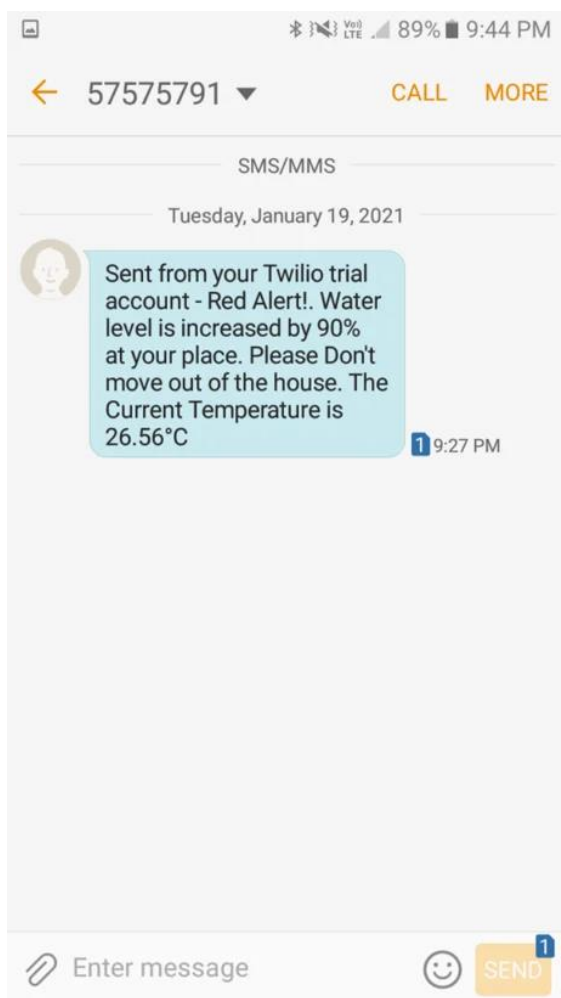
Mailgun Email Alert



When the water level crosses the Max Level. That resembles 'Red Alert'. This means that the water level has crossed 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