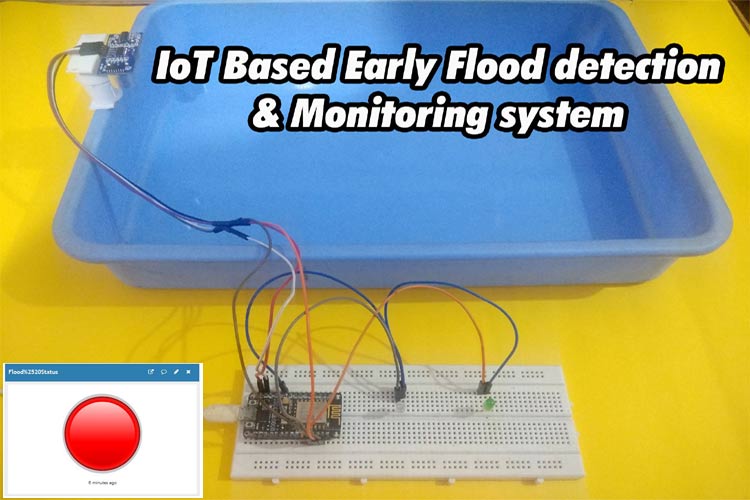
INTERNET OF THINGS

FLOOD MONITORING SYSTEM

INTRODUCTION:

In the face of escalating climate-related challenges, the development of a robust Flood Monitoring System emerges as a pivotal initiative. This system represents a sophisticated integration of cutting-edge technologies to address the increasing threats posed by flooding incidents. This introduction sets the stage for an exploration into the multifaceted components of the system, emphasizing its pivotal role in enhancing preparedness, minimizing damages, and ultimately fostering a more resilient response to the ever-growing risk of floods.

ABSTRACT:

The flood monitoring system is a crucial technological solution designed to enhance early detection and response to flooding events. Utilizing a network of sensors, data analytics, and real-time communication, this system aims to continuously monitor water levels, weather patterns, and potential flood risks. By employing advanced algorithms, it provides accurate predictions and timely alerts to authorities and affected community.

PROJE CT DEFITITION:

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. 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 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.

PROJECT OBJECTIVES:

Real-time flood monitoring objectives typically include:

1. Early Warning: Detecting flood events as they occur or even before they happen to provide early warnings to residents, emergency responders, and authorities.
2. Data Collection: Gathering real-time data on rainfall, river water levels, weather conditions, and other relevant parameters to assess flood risk.
3. Risk Assessment: Assessing the severity and potential impact of flooding in specific areas based on collected data.
4. Decision Support: Providing decision-makers with timely and accurate information to make informed choices about evacuation, resource allocation, and flood mitigation strategies.
5. Public Safety: Protecting lives and property by alerting residents to evacuate or take precautionary measures during a flood event.

These objectives collectively aim to reduce the impact of flooding on communities and enhance disaster preparedness and response efforts.

IOT SENSOR NETWORK DESIGN:

Deploying IoT sensors to monitor water levels involves several steps:

Sensor Selection : Appropriate water level sensors based on your specific requirements. Options include ultrasonic sensors, pressure sensors, or capacitive sensors, depending on the application.

Data Communication: Select a communication protocol (e.g., Wi-Fi, LoRa, cellular) and gateway devices to transmit data from the sensors to a central server or cloud platform.

Power Supply: Ensure a reliable power source for the sensors, which can be batteries, solar panels, or even energy harvesting methods, depending on the deployment location.

Sensor Placement: Strategically install sensors at key locations, such as rivers, reservoirs, or flood-prone areas. Consider factors like accessibility and environmental conditions.

Data Processing: Set up a central server or cloud platform to receive and process sensor data. Implement data analytics to detect trends, anomalies, or potential issues.

Alert System: Develop an alert system that can notify relevant stakeholders in real-time when water levels reach critical thresholds or unusual patterns are detected.

Data Visualization: Create user-friendly dashboards or mobile apps to visualize water level data for easy monitoring and decision-making.

Security: Implement robust security measures to protect the IoT network from cyber threats and unauthorized access.

Maintenance: Establish a maintenance schedule for sensor calibration, battery replacement, and system updates to ensure long-term reliability.

Data Storage: Store historical data for analysis and future reference. Consider data retention policies and compliance requirements.

Continuous monitoring of water levels using IoT sensors can help prevent disasters like floods, manage water resources efficiently, and protect the environment.

EARLY WARNING PLATFORM

AquaMonitor is a user-friendly web-based platform designed for real-time visualization and monitoring of water levels across various locations. This innovative system provides a comprehensive solution for users ranging from local authorities to concerned citizens, offering crucial insights into water level dynamics to facilitate informed decision-making and early response to potential flood events.

AquaMonitor represents a leap forward in flood monitoring technology, offering a dynamic and accessible platform for tracking water levels in real time. By combining advanced data visualization with user-friendly features, this web-based system empowers stakeholders at various levels to proactively manage flood risks and enhance community resilience in the face of changing environmental conditions.

INTEGRITY APPROACH:

IoT sensors can send data to an early warning system through a structured process that involves sensor deployment, data collection, and transmission. Here’s a general overview of how this connection is established:

1. Sensor Deployment:

- Deploy IoT sensors strategically in flood-prone areas, water bodies, or relevant locations. These sensors are equipped with various environmental monitoring capabilities, such as water level sensors, rain gauges, or weather sensors.

2. Data Collection:

- IoT sensors continuously collect data from the environment based on their specific monitoring functions. For flood monitoring, this could include data on water levels, precipitation, temperature, and other relevant parameters.

3. Data Processing and Aggregation:

- The collected data is processed within the IoT devices to ensure accuracy and relevance. Aggregated data may include real-time measurements, historical trends, and any anomalies detected by the sensors.

4. Communication Protocols:

- IoT sensors use communication protocols such as MQTT (Message Queuing Telemetry Transport), HTTP/HTTPS, or CoAP (Constrained Application Protocol) to transmit the collected data. The selection of the protocol depends on factors like energy efficiency, data volume, and network constraints.

5. Connectivity:

- IoT sensors typically use wireless communication technologies like Wi-Fi, cellular networks, LoRa (Long Range), NB-IoT (Narrowband IoT), or satellite communication to connect with the broader network.

6. Gateway Devices:

- In some cases, IoT sensors may communicate directly with the early warning system. However, in more extensive deployments, gateway devices might be employed to aggregate data from multiple sensors and transmit it collectively to the central system.

7. Cloud-Based Platforms:

- The transmitted data is often sent to cloud-based platforms or edge computing systems for further processing and storage. These platforms can handle large volumes of data, perform analytics, and ensure data integrity.

8. Integration with Early Warning System:

- The early warning system is designed to receive, process, and analyze incoming data. It may use APIs (Application Programming Interfaces) or specific protocols to integrate seamlessly with the IoT platform.

9. Alert Generation:

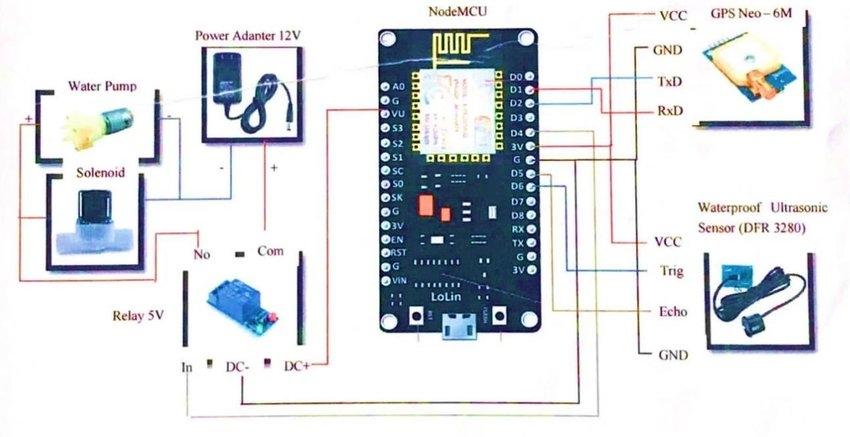
- Based on the analyzed data, the early warning system triggers alerts when predefined thresholds or abnormal patterns are detected. These alerts can be sent through various channels such as SMS, email, mobile apps, or sirens, depending on the urgency and severity of the situation.

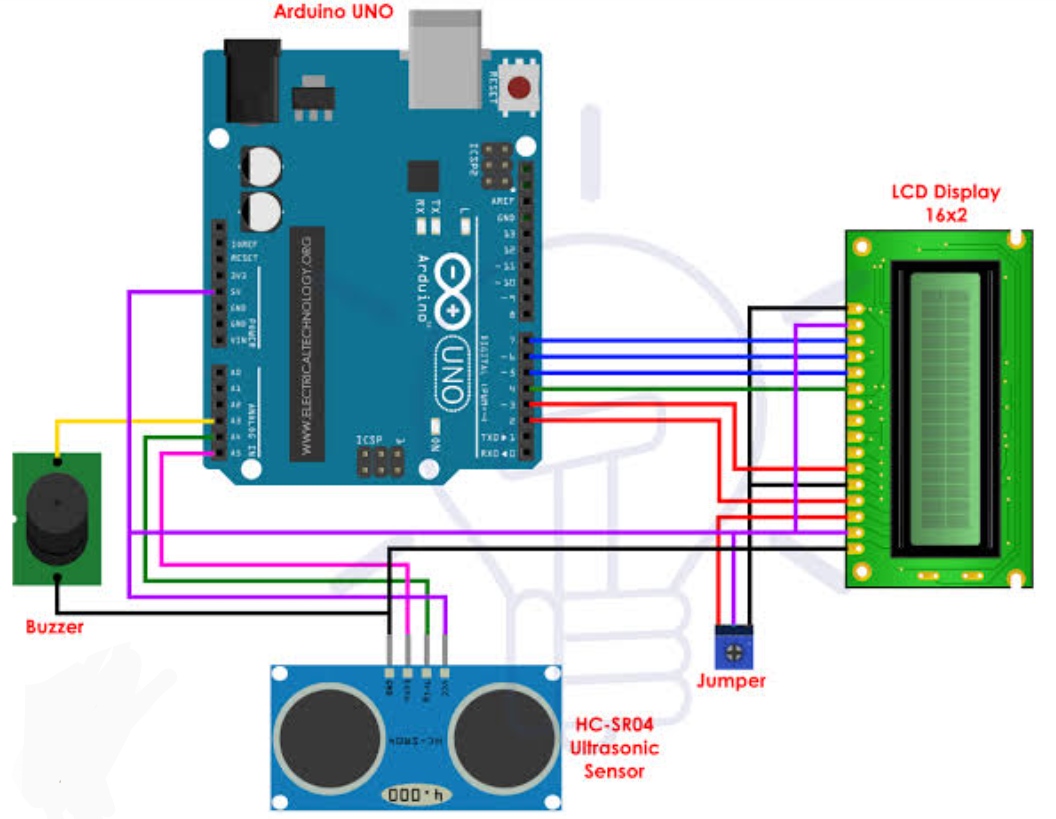
10. User Interface:

- The early warning system provides a user interface, such as a web-based dashboard or a mobile app, where users can visualize real-time data, historical trends, and receive alerts. This interface is critical for decision-makers, emergency responders, and the public.

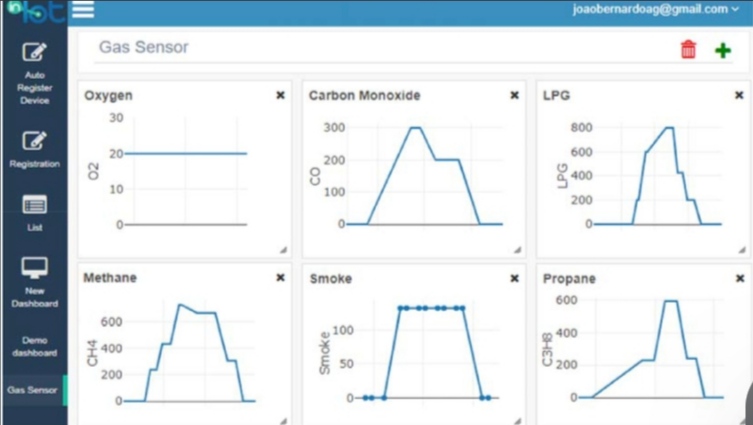
By following this process, IoT sensors seamlessly contribute to the early warning system, enhancing its capability to provide timely and accurate alerts for flood events.

SCHEMATIC DIAGRAM





SCREENSHOTS OF IOT SENSORS





WHY WE NEED IOT BASED FLOOD MONITORING SYSTEM?

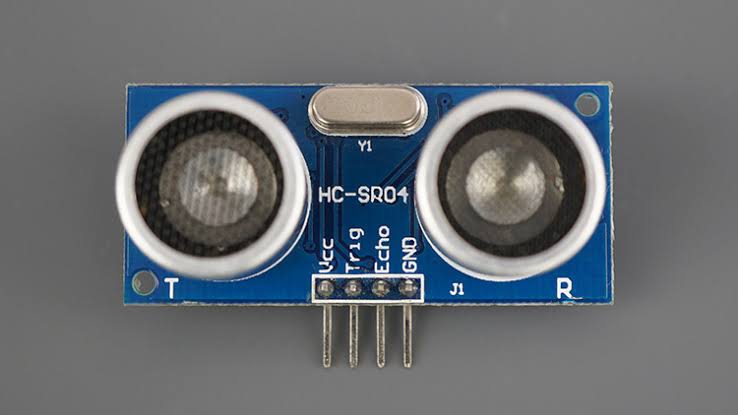
In the peninsular countries like Pakistan and India, with extreme weather and climatic conditions, the occurrence of heavy rainfall is normal. Multiple times, the arrival of very heavy rains results in the heavy discharge of water or because of the sudden melting of the glaciers due to global warming.

Especially, in the monsoon which normally begins in the mid of June and lasts till October, thousands of people lost their lives by drowning and their habitats were collapsed. The leftover were evacuated by the state and central disaster relief authorities. The severe water logging brought daily work to halt. In order to save the lives of the people, their habitat and the economy, the major step is to monitor the data on real time basis and if the situation is reaching a certain threshold, then alert is to be provided to the concerned person living in the area which is currently at risk. Even if it is difficult to abandon the natural calamity but the mandatory steps are to be taken by the government agencies to shift the population to a safe region and the losses will get reduced to less than 30%.

In this modern era, there are multiple systems working and are deployed at different locations but the alert notification is passed to government agencies and this ends up in slowing down the process. The reason behind this is that flood is very spontaneous disaster and government agencies have to follow multiple steps before reaching to a decision. In this case, awareness among the people is very necessary along with the government officials. So, that a combined and better result will be achieved.

In my proposed system the water level can be monitored in real-time using a cell phone, no matter you are in a meeting, outside your home, etc. You will be able to see everything in real-time. You can check the App at anytime and find out about the water level, if it’s increasing or decreasing. Different level can be defined. Alert messages are generated automatically when the desired conditions are met.

ULTRASONIC SENSOR HC-SR04:

This is the HC-SR04 Ultrasonic Sensor which will be used for monitoring the water level. This Module has a total of 4 male headers clearly labeled as VCC, Trig, Echo, and Gnd. The connection of the Ultrasonic Sensor with the Nodemcu ESP8266 is explained in the Circuit Diagram.

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.Python3.7 IDLE

3.Bolt IOT Cloud

4.Bolt IoT Android App

5.Twillo SMS Messaging API

6.Mailgun EMAIL Messaging APISoftware components.

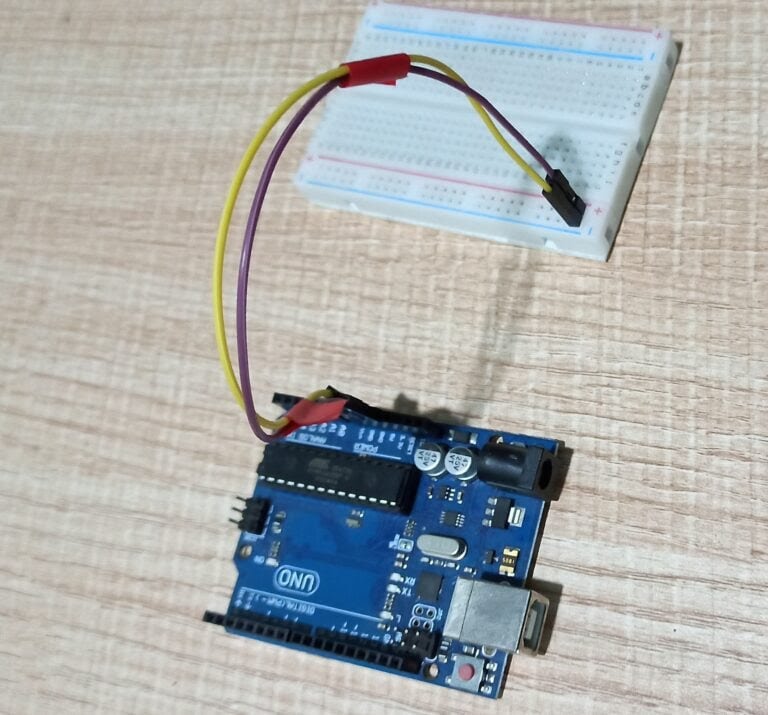
HAND TOOLS AND FABRICATION MACHINES.

1.Electrical Tape.

2.Green Cello Tape.

HARDWARE SETUP:

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.

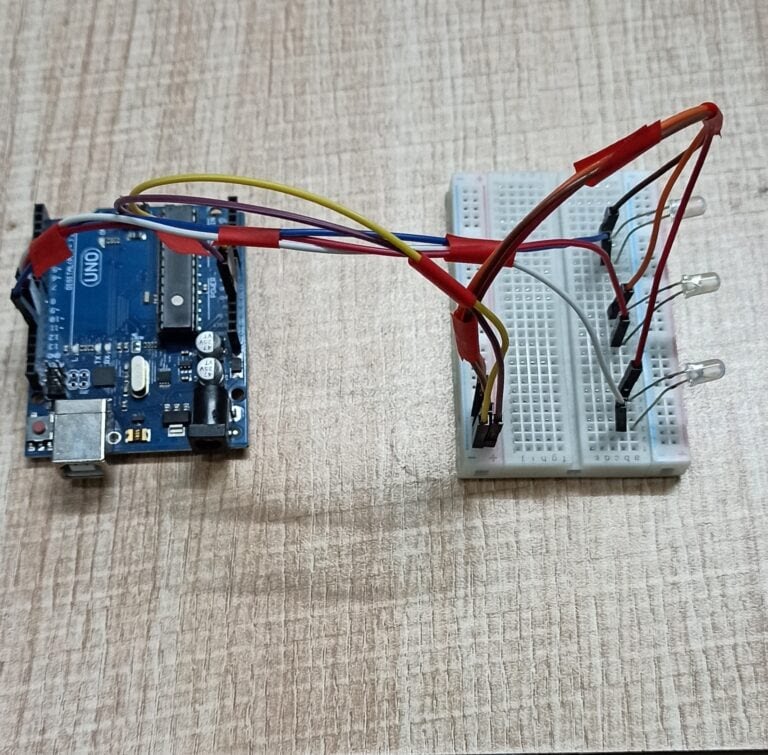
**For Orange LED:**

* 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.

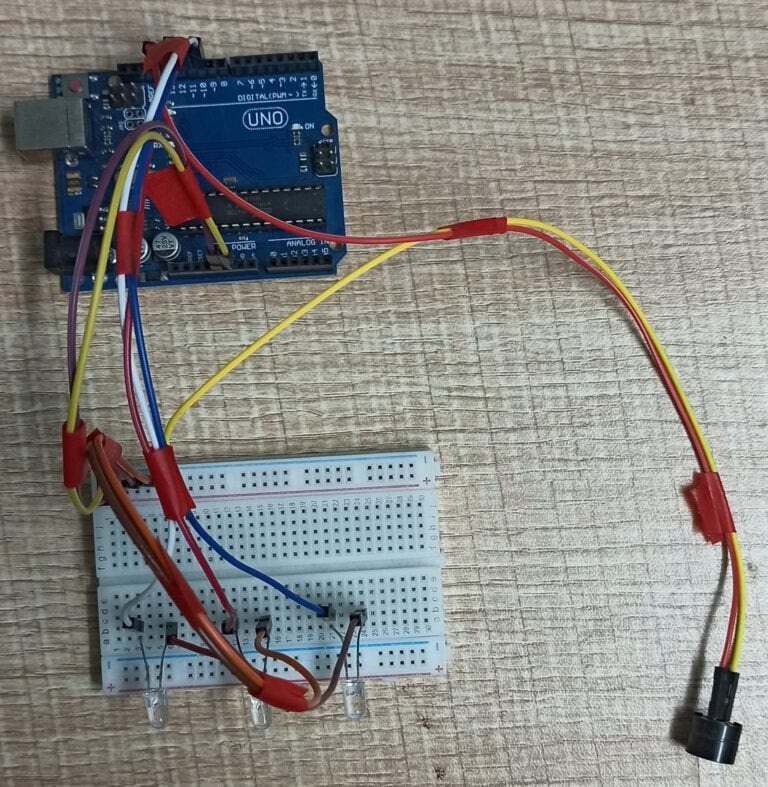
Step 3: Connecting Buzzer



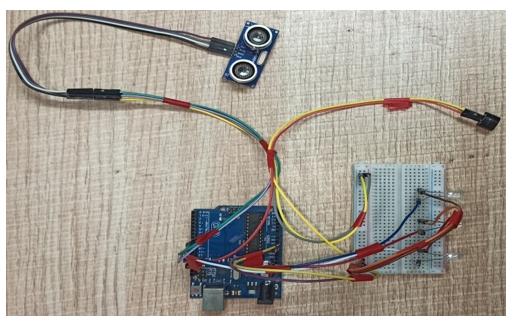
* VCC of  Buzzer to Digital Pin ‘13’ of the Arduino.
* GND of Buzzer to the GND of Arduino

**Step 4**: **Connecting HC-SR04 Ultrasonic Sensor**

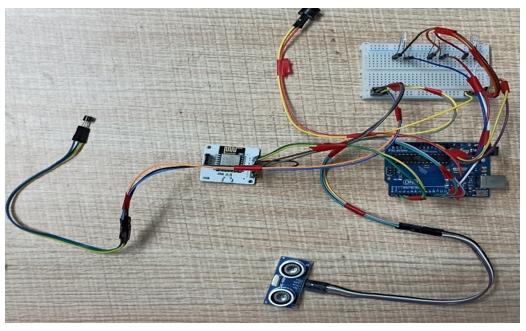
* VCC of Ultrasonic Sensor to 5v of Arduino.
* GND of Ultrasonic Sensor to GND of Arduino.
* Echo of Ultrasonic Sensor to Digital Pin ‘8’ of Arduino.
* Trig of Ultrasonic Sensor to Digital Pin ‘9’ of Arduino.

**Step 5: Connecting Bolt WiFi Module**

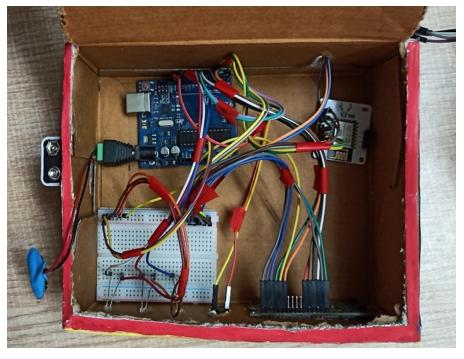
* 5v of Bolt WiFi Module to 5v of Arduino.
* GND of Bolt WiFi Module to GND of Arduino.
* TX of Bolt WiFi Module to RX of Arduino.
* RX of Bolt WiFi Module to TX of Arduino.

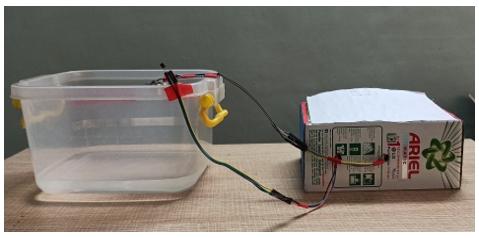
**Step 6: Connecting LM35 Temperature Sensor**

* 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.

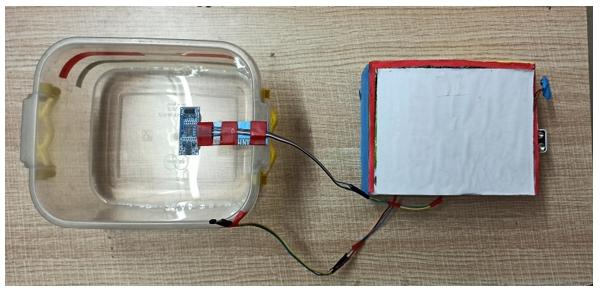
**Step 7:Connecting 16×2 LCD  Display**

* 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.

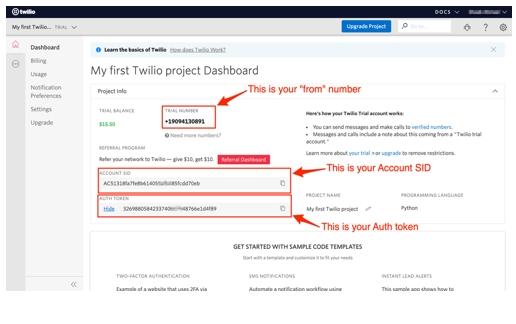


SOFTWARE PROGRAMMING:

After the successful completion of hardware setup. Now it’s the 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 account on various online app services and noting down the important keys and id’s. Below all the steps given to create account on online app services and noting down the keys.

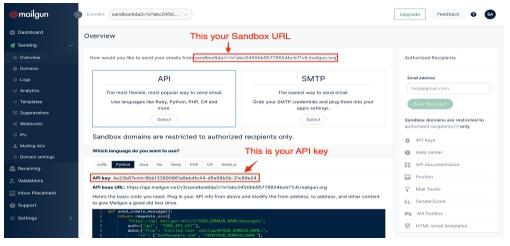
**Step 1**:**Creating an account on Twillo and setting up Twillo for sending Sms alerts.**

* Visit <https://www.twilio.com/>.
* Create 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 up details.
* Verifying your Account.
* Once you have verified your Email after that you have add your phone number.
* After Entering your number. Click on send activation code. After some time you will receive one 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 so you will receive API Key and Sandbox URL. Save this both credentials somewhere you will be further using in this project.



**Step 3:Creating an account on Bolt Cloud and Bolt Android App and Link the Bolt Module to Cloud.**

* Visit [https://cloud.boltiot.com](https://cloud.boltiot.com/).
* Create 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 account on cloud. Then Download Bolt Android App from playstore.
* Create a account on the Bolt app with the same email-Id then use the mobile hotspot for linking the Bolt WiFi module to cloud.
* After successful linking of 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 somewhere.
* Also copy the Bolt Device Id which is present on Bolt IoT dashboard and save it somewhere.

**Bolt Device Id**

**API KEY**

**Step 4: Coding**

**Step 4.1: 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.

/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);

(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.

L(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 (per >= 55)

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 4.2: 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 find this on your Twilio Dashboard'

TO\_NUMBER = 'This is your number. Make sure you are adding +91 in beginning'

#bolt iot details

API\_KEY = 'XXXXXXXXX'

#This is your Bolt cloud API

Key.

DEVICE\_ID = 'BOLTXXXXXXXXX' #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 on your Mailgun Dashboard'

SANDBOX\_URL= 'You can find this on your Mailgun Dashboard'

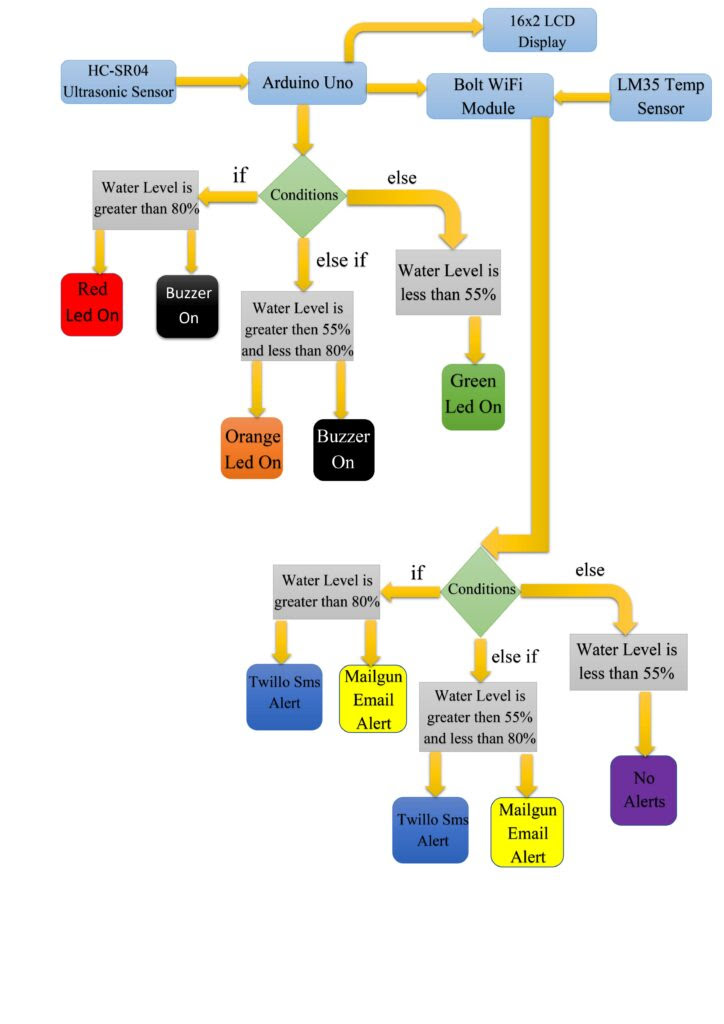
SENDER\_EMAIL = 'test@ + SANDBOX\_URL' # No need to modify this. The sandbox URL is of the format test@YOUR\_SANDBOX\_URL

RECIPIENT\_EMAIL = 'Enter your Email

* 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.
* boltiot import Sms, Email, Bolt  
  Below is the complete code of main.py
* import conf  
  from 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\_EMAIL, conf.RECIPIENT\_EMAIL)  
    
    
  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")  
   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()  
   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 " +str(Water\_level) + "% at your place. Please be Safe. The current Temperature is " + str(temp\_value) + "°C."  
   head="Orange Alert"  
   message\_1="Water level is increased by " + str(Water\_level) + "% at your place. Please be Safe. The current Temperature is " + str(temp\_value) + "°C."  
   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) + "% at your place. Please Don't move out of the house. The Current Temperature is " + str(temp\_value) + "°C"  
   head="Red Alert!"  
   message\_1="Water level is increased by " + str(Water\_level) + "% at your place. Please Don't move out of the house. The Current Temperature is " + str(temp\_value) + "°C."  
   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

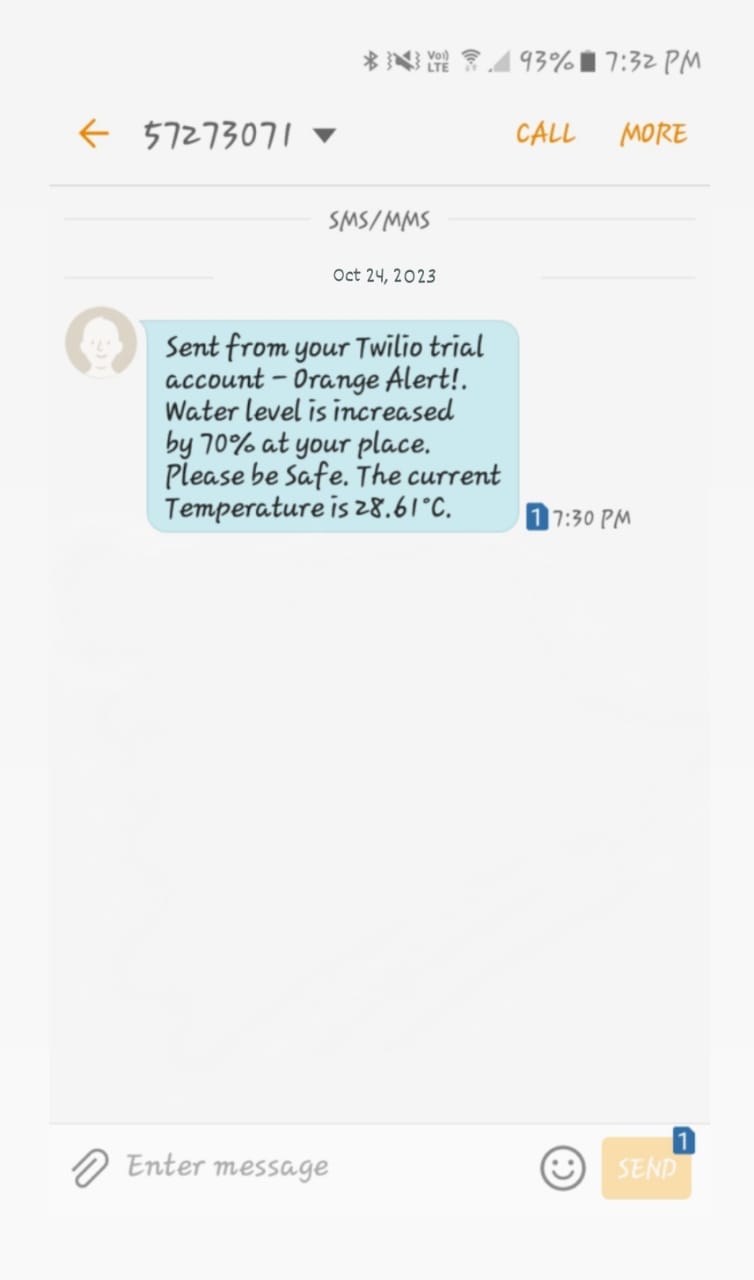
### Demonstration

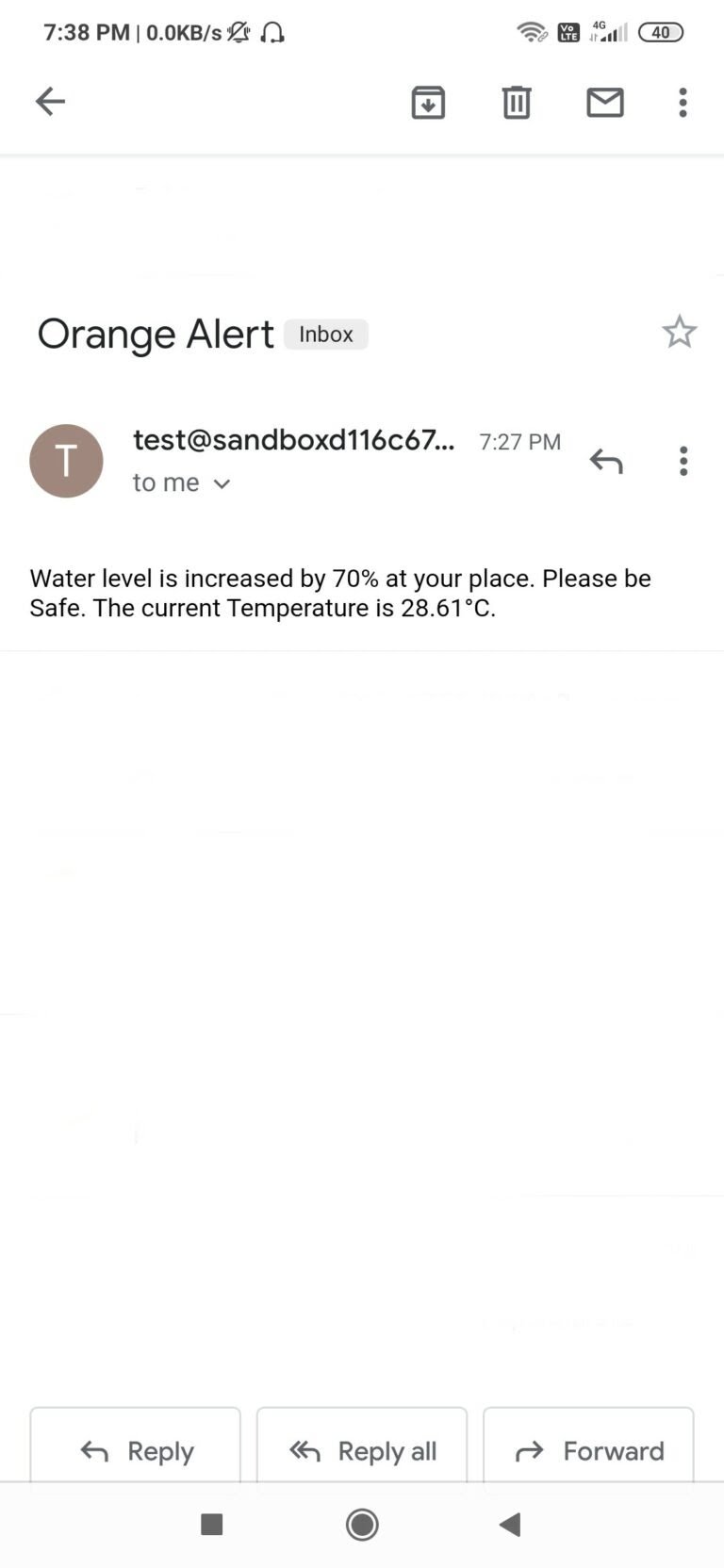
For doing the practical demonstration. First connect the USB cable type-B to the Laptop’s USB slot for power supply. Also simultaneously run the python program(i.e Main.py). Firstly the ultrasonic sensor will sense the water level in distance and then the arduino program will help to convert it into 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 lets look into each zone.

When water level is at Min/Normal level. That resembles ‘Green Alert’. This means that water is at normal position and no sign about flood condition. Also green led will glow and it will also show green alert in Lcd display with water level.

When 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 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 led will glow and buzzer will buzz. It will also show orange alert in Lcd display.

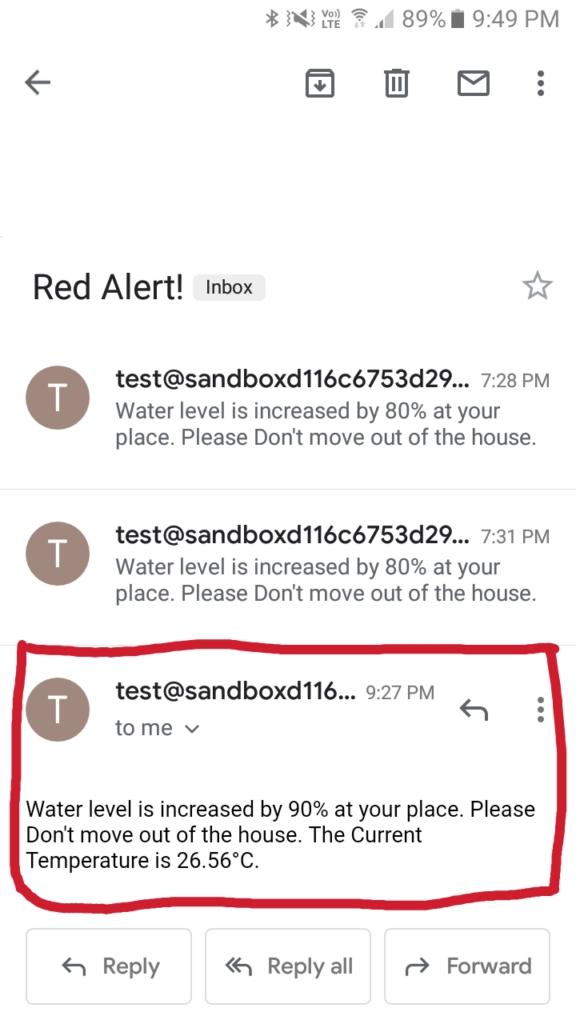
Also Sms and Email is send to registered user with proper message and current temperature of that place.





When water level crosses the Max Level. That resembles ‘Red Alert’. This means that water level has crossed the 80% and flood situation has occured at that place. With 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 led will glow and buzzer will buzz for two times. It will also show red alert in Lcd display

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



### Conclusion

Nowadays the Internet Of things (IoT) is broadly used in worldwide, this system will display the data of the water level measured on lcd display. This project can be very helpful to the Meteorological Department to continuously monitor the dams and river beds water level. With this project it can save many people lives by giving alerts when the water level crosses beyond the limit. This project is very cost-effective, flexible and productive in areas where flood conditions happens everytime.

Result:

