**FLOOD MONITORING AND EARLY WARNING**

By,

V.AJITH KUMAR

A. KAMALESH Mentor,

S. KISHORE Ms.R.Preethi,AP/ECE

E.R. HARRISH

M.SATHISH KUMAR

ABSTRACT

* Flood monitoring using the Internet of Things (IoT) is a rapidly developing field with the potential to revolutionize the way we detect, predict, and respond to flooding. IoT-based flood monitoring systems use a network of sensors and devices to collect data on water levels, rainfall, and other environmental conditions in real time. This data is then transmitted to a cloud-based platform for analysis and visualization

**INTRODUCTION**

Floods are among the most destructive and deadly natural disasters, causing extensive damage to property and posing a significant threat to human lives. Timely and accurate flood monitoring and early warning systems are critical for mitigating the impact of these catastrophic events. With the rapid advancement of technology, the integration of the Internet of Things (IoT) into flood monitoring and early warning systems has emerged as a promising solution to address these challenges.

The Internet of Things refers to the interconnected network of devices and sensors that can collect, transmit, and exchange data over the internet. In the context of flood monitoring, IoT offers a transformative approach by providing real-time data acquisition and analysis capabilities that enhance our understanding of flood events and facilitate faster response and mitigation efforts.

Traditionally, flood monitoring relied on limited data sources, often with delayed reporting, which hindered the ability to provide timely warnings and make informed decisions. IoT technology has revolutionized this landscape by enabling a vast array of sensors, including water level gauges, rainfall detectors, weather stations, and cameras, to be deployed in flood-prone areas. These sensors continuously collect data on crucial parameters, such as water levels, precipitation, and weather conditions, transmitting the information to a central platform in real-time. As a result, flood monitoring systems can now access a wealth of accurate, up-to-the-minute data, enabling more precise flood predictions and early warnings.

In this context, IoT-based flood monitoring and early warning systems offer several key advantages:

1. Real-time Data: IoT sensors provide continuous data streams, ensuring that flood events are detected and tracked as they occur, not after the fact.
2. Precision and Accuracy: The ability to gather data from various sources allows for more accurate flood predictions and assessments, leading to better-informed decision-making.
3. Early Warnings: With timely data and advanced analytics, authorities can issue early warnings to vulnerable communities, giving them more time to prepare and evacuate, ultimately saving lives.
4. Improved Disaster Response: Emergency services can use real-time information to allocate resources more efficiently and respond promptly to flood-related emergencies.
5. Sustainable Infrastructure Planning: IoT data can inform urban planners and policymakers to design flood-resilient infrastructure and improve land-use planning.

**PROPOSAL SYSTEM**

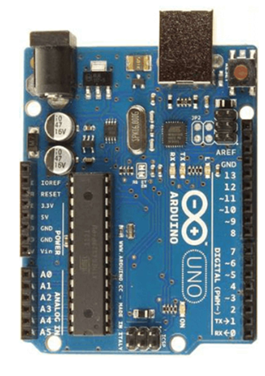
**COMPONENTS USED**

**Ultrasonic Distance Sensor - HC****- SR04 (5V)**



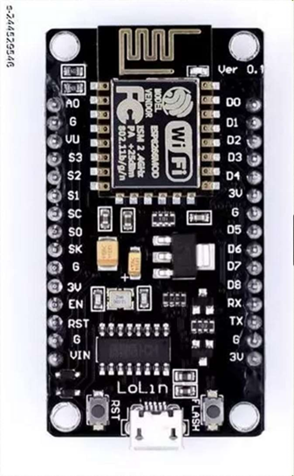
* This is the HC-SR04 ultrasonic distance sensor. This economical sensor provides 2cm to 400cm of non- contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.
* There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). You will find this sensor very easy to set up and use for your next range-finding project!
* This sensor has additional control circuitry that can prevent inconsistent "bouncy" data depending on the application.

ARDUNIO UNO



* Arduino Uno is a microcontroller board based on the ATmega328P microcontroller. It is a popular choice for building DIY electronics projects, including flood monitoring systems.
* Arduino Uno has several features that make it well-suited for flood monitoring systems:
* It is relatively inexpensive and easy to obtain.
* It is easy to program using the Arduino IDE software.
* It has a wide range of compatible sensors and other hardware modules.
* It is supported by a large and active community of developers and users.

**ESP8266**



* The ESP8266 can be programmed to generate alerts in a variety of ways, such as by sending an email, text message, or push notification.
* It can also be programmed to trigger an alarm or other device when the water level reaches a certain threshold.
* ESP8266 flood monitoring systems are a relatively inexpensive and easy-to-deploy way to monitor water levels and generate alerts when the water level reaches a certain threshold.
* This can be helpful for preventing flood damage to property and infrastructure.

**Alarm and buzzer**



* An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The
* Main function is to convert the signal from audio to sound.
* Based on the various designs, it can generate different sounds like alarm, music, bell & siren.
* The **pin configuration of the buzzer** is shown below. It includes two pins namely positive and negative.

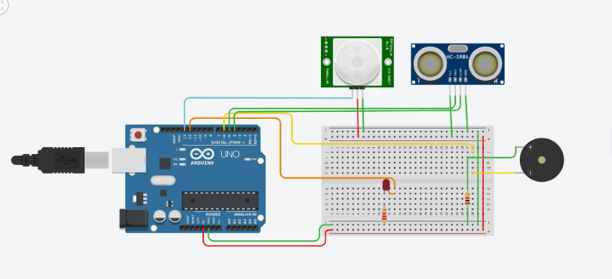
**Benefits**

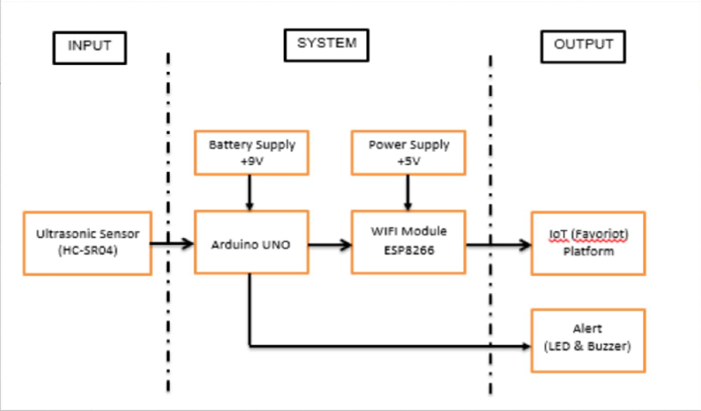
### Benefits of IoT-based Flood Monitoring and Early Warning Systems

* **Reduced flood losses:** IoT-based flood monitoring and early warning systems can help to reduce flood losses by providing early warning to residents and emergency responders, allowing them to take necessary precautions.
* **Improved public safety:** IoT-based flood monitoring and early warning systems can help to improve public safety by enabling early evacuation and other preparedness measures.
* **Enhanced decision-making:** IoT-based flood monitoring and early warning systems can provide valuable data to decision-makers, helping them to develop and implement more effective flood management strategies.
* **Reduced environmental impact:** IoT-based flood monitoring and early warning systems can help to reduce the environmental impact of flooding by enabling more efficient use of water resources and reducing the risk of damage to infrastructure and ecosystems.

# **PROPOSAL SYSTEM**

**Circuit diagram**





**BLOCK DIAGRAM**

The block diagram you sent shows a **flood monitoring and early warning system** using the **Internet of Things (IoT)**. This system uses a network of sensors to collect data on water levels, rainfall, and other environmental conditions. The data is then transmitted to a cloud-based platform for analysis and visualization. The system can then generate early warning alerts, operate flood control structures, or manage water resources to reduce the impact of flooding.

**Block diagram explanation:**

* **Sensors:** The sensors collect data on water levels, rainfall, and other environmental conditions. This data can be collected from a variety of sources, such as river gauges, rain gauges, and meteorological stations.
* **Gateway:** The gateway collects data from the sensors and transmits it to the cloud-based platform. The gateway may also be responsible for preprocessing the data before it is transmitted.
* **Cloud-based platform:** The cloud-based platform analyzes the data from the sensors and generates flood forecasts. The cloud-based platform may also be responsible for storing and managing the data, as well as providing access to the data to authorized users.
* **Actuators:** Actuators are devices that can be used to control flood control structures, such as dams and gates. The actuators can be activated by the cloud-based platform based on the flood forecasts.
* **Output:** The output of the system can be used to generate early warning alerts, operate flood control structures, or manage water resources.

**How the system works:**

The system works by continuously collecting data on water levels, rainfall, and other environmental conditions. This data is then transmitted to the cloud-based platform for analysis and visualization. The cloud-based platform uses the data to generate flood forecasts. Based on the flood forecasts, the cloud-based platform can then generate early warning alerts, operate flood control structures, or manage water resources.

**Benefits of IoT-based flood monitoring and early warning systems:**

IoT-based flood monitoring and early warning systems offer several benefits over traditional flood monitoring systems, including

* **Real-time data collection:** IoT sensors can collect data on water levels and other environmental conditions in real time, providing early warning of potential flooding.
* **Spatial granularity:** IoT sensors can be deployed in remote and inaccessible areas, providing a more comprehensive picture of flood conditions.
* **Affordability:** IoT sensors are relatively inexpensive to deploy and maintain, making them a cost-effective solution for flood monitoring.

**Applications of IoT-based flood monitoring and early warning systems:**

IoT-based flood monitoring and early warning systems can be used in a variety of applications, including:

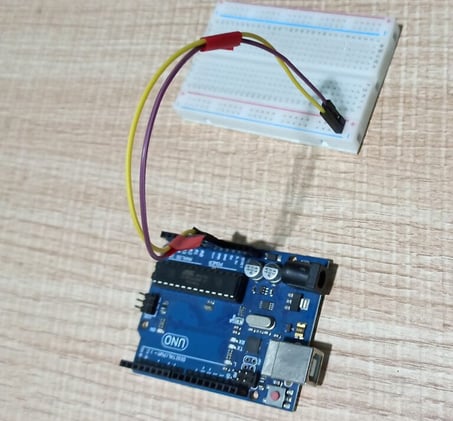
* **Early warning systems:** IoT sensors can be used to develop early warning systems that alert residents and emergency responders to potential flooding.
* **Flood forecasting:** IoT data can be used to develop more accurate and timely flood forecasts.
* **Flood control:** IoT data can be used to operate flood control structures more effectively.
* **Damage assessment:** IoT data can be used to assess damage after a flood and plan for recovery.

IoT-based flood monitoring and early warning systems are a promising new technology with the potential to significantly improve our ability to detect, predict, and respond to flooding. By providing real-time data collection, spatial granularity, and affordability, IoT-based flood monitoring systems can help us to reduce flood losses, improve public safety, enhance decision-making, and reduce the environmental impact of flooding.

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:**

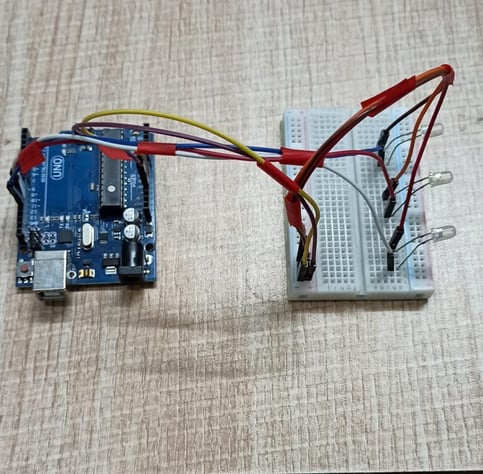
* VCC of Green Colour LED to Digital Pin ‘10’ of the Arduino.
* GND of Green Colour LED to the GND of Arduino.

**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.

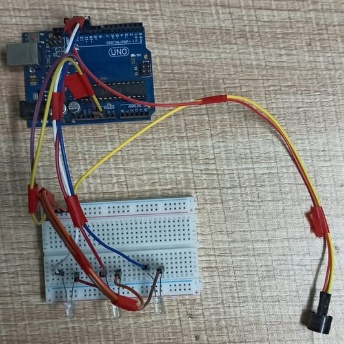
**For Red LED:**

* VCC of Red Colour LED to Digital Pin ‘12’ of the Arduino.
* GND of Red Colour LED to the GND of Arduino.



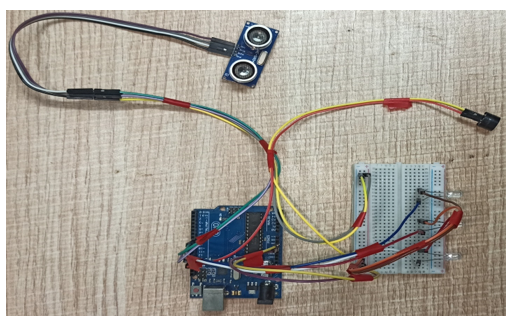
**Step 3**: **Connecting Buzzer**

* 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.

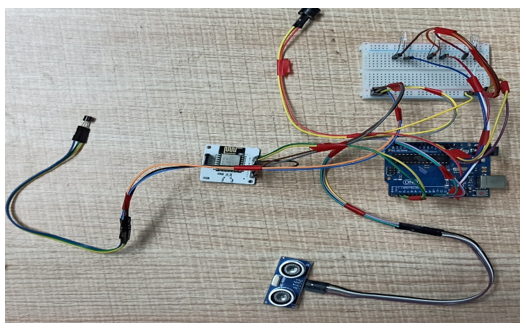


**Step 5: Connecting Bolt WiFi Module**

* 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.

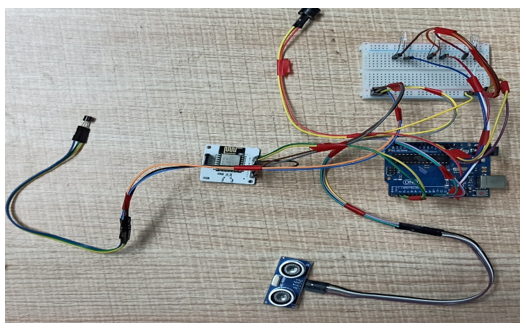
**Step 6: Connecting LM35 Temperature Sensor**

* 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.

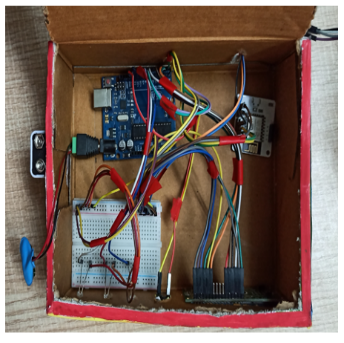


**Step** **7:Connecting 16×2** **LCD Display**

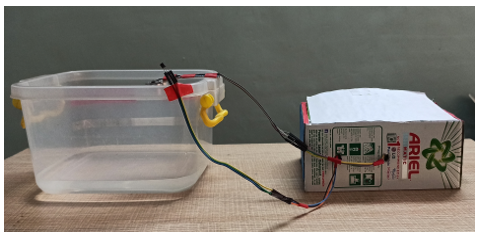
* Pin 1,3,5,16 of 16×2 LCD to GND of Arduino.
* Pin 2,15 of 16×2 LCD to 5v of Arduino.
* Pin 4 of 16×2 LCD to Digital Pin ‘2’ of Arduino.
* Pin 6 of 16×2 LCD to Digital Pin ‘3’ of Arduino.
* Pin 11 of 16×2 LCD to Digital Pin ‘4’ of Arduino.
* Pin 12 of 16×2 LCD to Digital Pin ‘5’ of Arduino.
* Pin 13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



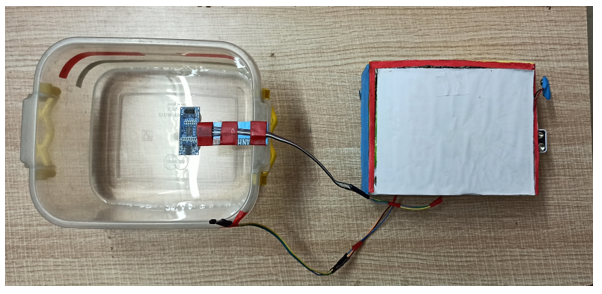
After doing the hardware connection put all the hardware components in one box.



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



Also attach Ultrasonic sensor on the top of the container.



## **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 Tillo and setting up Twillo for sending Sms alerts.**

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

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

**Step 5: Coding**

After setting online app services and saving the keys somewhere. Now most importantf 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 appropirate name to the file and add .into 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);

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 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);

}

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 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 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.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:

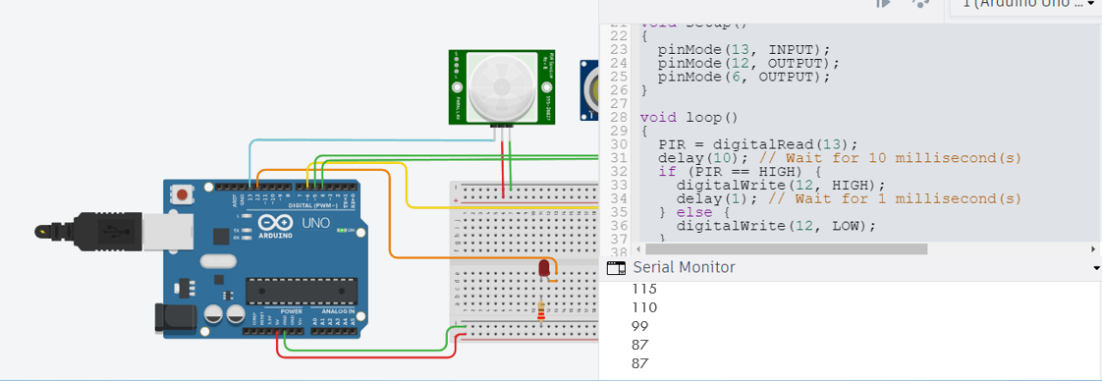
print ("Error occured: Below are the details")

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.

**Serial monitor output**



### 

### **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