Phase : 3 – Development part 1

Flood Monitoring and Early Warning System Project

Project Overview

This project aims to develop a Flood Monitoring and Early Warning System using IoT devices. The system will enable real-time monitoring of water levels in flood-prone areas and provide early warnings to mitigate potential disasters. This project involves deploying IoT devices in strategic locations and developing a Python script to collect and transmit data to an early warning platform.

Project Objectives

1. Deploy IoT devices with water level sensors in flood-prone areas.
2. Develop a Python script to collect and transmit water level data to an early warning platform.
3. Create a reliable system for monitoring and alerting of potential floods.
4. Ensure data accuracy and system resilience.

Project Requirements

Hardware:

- IoT devices with water level sensors.
- Power sources for the IoT devices.
- Network connectivity for data transmission.

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:

- Python programming environment.
- Early warning platform for data reception and analysis.

Software components -

1. Arduino IDE
2. Python 3.7 IDLE
3. Bolt IoT Cloud
4. Bolt IoT Android App
5. Twillo SMS Messaging API
6. Mailgun EMAIL Messaging APISoftware components

IoT Device Selection

The project will involve selecting suitable IoT devices with water level sensors. These devices must be capable of accurate data collection and reliable data transmission. The choice of devices should be based on their compatibility with the project's goals.

Sensor Deployment

The IoT devices will be deployed in flood-prone areas, chosen based on geographical and environmental factors. The deployment will be carried out ensuring that the devices are securely installed, well-calibrated, and protected from environmental conditions. Adequate power sources and network connectivity will be provided for the devices.

Python Script Development

A Python script will be developed to run on the IoT devices. The script will perform the following tasks:

1. Continuously collect data from the water level sensors.
2. Format and preprocess the collected data.
3. Transmit the data to the early warning platform at regular intervals.
4. Implement alerting mechanisms to trigger warnings when predefined thresholds are exceeded.

Writing the code in Python IDE

```
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)
```

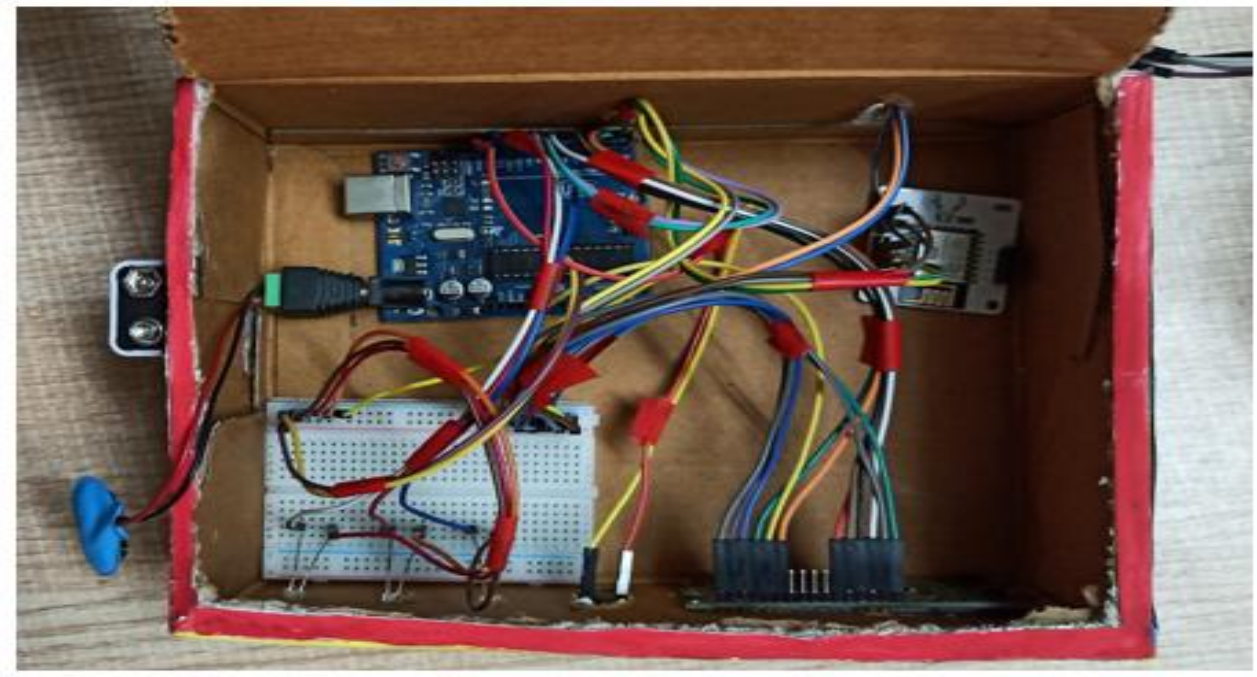
```
        if int(Water_level) >= max_value:
            message ="Red Alert!. Water level is increased by " + str(Water_level) + "% at your
place. Please Don't move out of the house. The Current Temperature is " + str(temp_value) +
"°C"
            head="Red Alert!"
            message_1="Water level 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)
```

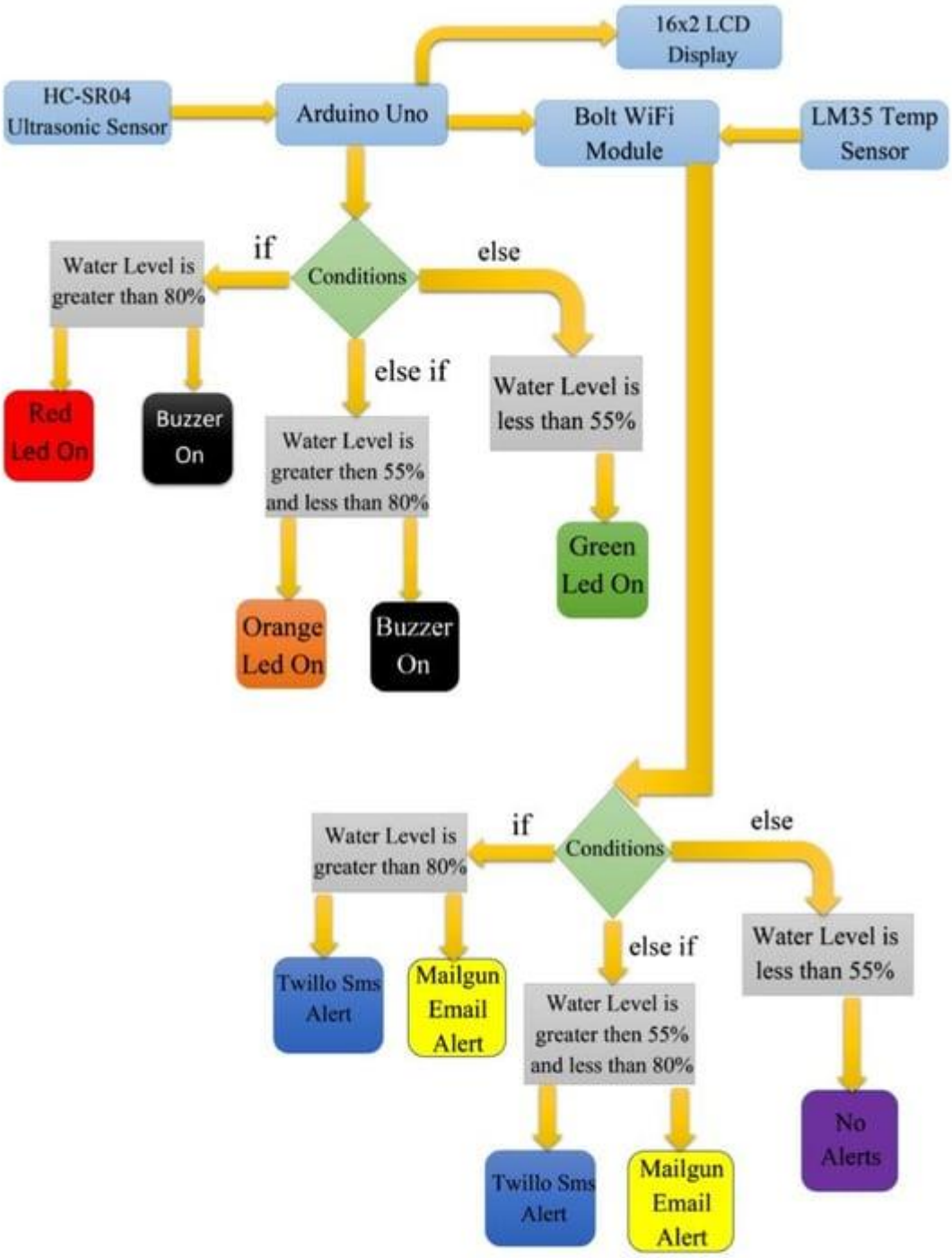
Header file :

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

Demonstration

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




16x2 LCD Display
HC-SR04 Ultrasonic Sensor
Arduino Uno
Bolt WiFi Module
LM35 Temp Sensor
Conditions
if
Water Level is greater than 80%
Red Led On
Buzzer On
else if
Water Level is greater then 55% and less than 80%
Orange Led On
Buzzer On
else
Water Level is less than 55%
Green Led On
Conditions
if
Water Level is greater than 80%
Twillo Sms Alert
Mailgun Email Alert
else if
Water Level is greater then 55% and less than 80%
Twillo Sms Alert
Mailgun Email Alert
else
Water Level is less than 55%
No Alerts

Early Warning Platform

An early warning platform will be set up to receive, process, and analyze the data transmitted by the IoT devices. This platform will have the capability to:

- Receive data via APIs or other communication protocols.
- Analyze incoming data for flood risk assessment.
- Trigger alerts and warnings based on real-time data analysis.

Testing and Validation

The system will undergo rigorous testing to ensure its accuracy and reliability. Testing will include:

- Verification of sensor accuracy and data transmission.
- Simulation of flooding events to validate the system's effectiveness.
- Error detection and troubleshooting.

Project Documentation

A comprehensive project document will be created for assessment, including:

- Project objectives and requirements.
- Details about the selected IoT devices and their deployment locations.
- The Python script with a detailed explanation of its functionality.
- Information about the early warning platform and data transmission protocols.
- Results of system testing and validation.
- Documentation of any challenges encountered and their solutions.
- Safety measures for deployment and maintenance.

Conclusion

This Flood Monitoring and Early Warning System project is essential for mitigating potential flood-related disasters. It incorporates IoT technology and data analysis to provide timely alerts to communities at risk. The project aims to deliver a reliable and accurate system for flood monitoring and early warning, contributing to the safety and well-being of communities in flood-prone areas.