Flood Monitoring and Early Warning system using IoT

Team Members:

- Gowsalya N – 921021104012
- Epshiba R - 921021104010
- Deepika E – 921021104005
- Meenakshi R - 921021104027

Phase-3 Document Submission

Flood Monitoring and Early Warning:

- Floods are a major natural disaster that can cause widespread damage and loss of life. When water levels rise suddenly in dams, riverbeds, and other bodies of water, it can lead to flooding in surrounding areas. This can cause significant damage to property and infrastructure, as well as loss of life.
- It is important to have early warning systems in place to alert people of rising water levels so that they can take necessary precautions. One way to do this is to use water level sensors to monitor water levels in real time. If the water level exceeds a certain threshold, the sensor can generate an alert that can be sent to people via a variety of methods, such as LED signals, buzzers, SMS messages, or emails.
- This project aims to develop a water level monitoring system that can alert people of rising water levels in riverbeds. The system will use water level sensors to measure the water level in real time. If the water level exceeds a certain threshold, the system will generate an alert that will be sent to people via LED signals, a buzzer, SMS messages, and emails.
- This system can help to protect people and property from the devastating effects of floods by providing early warning of rising water levels.

Components Required:

These are the approximately estimated components that are needed for the Flood monitoring and early warning system:

Hardware components:

- ESP32 wifi module
- NodeMCU/Arduino
- Breadboard
- 5mm LED and Buzzer
- 16×2 LCD Display
- LM35 Temperature Sensor
- HC-SR04 Ultrasonic Sensor
- Some Jumper Wires
- Male to Female Jumper Wires
- Male to Male Jumper Wires
- Female to Female Jumper Wires
- 9v Battery and Snap Connector

Software components:

- Arduino IDE
- Python 3.7 IDLE
- IBM IoT Cloud
- Twillo SMS Messaging API
- Mailgun EMAIL Messaging API Software components

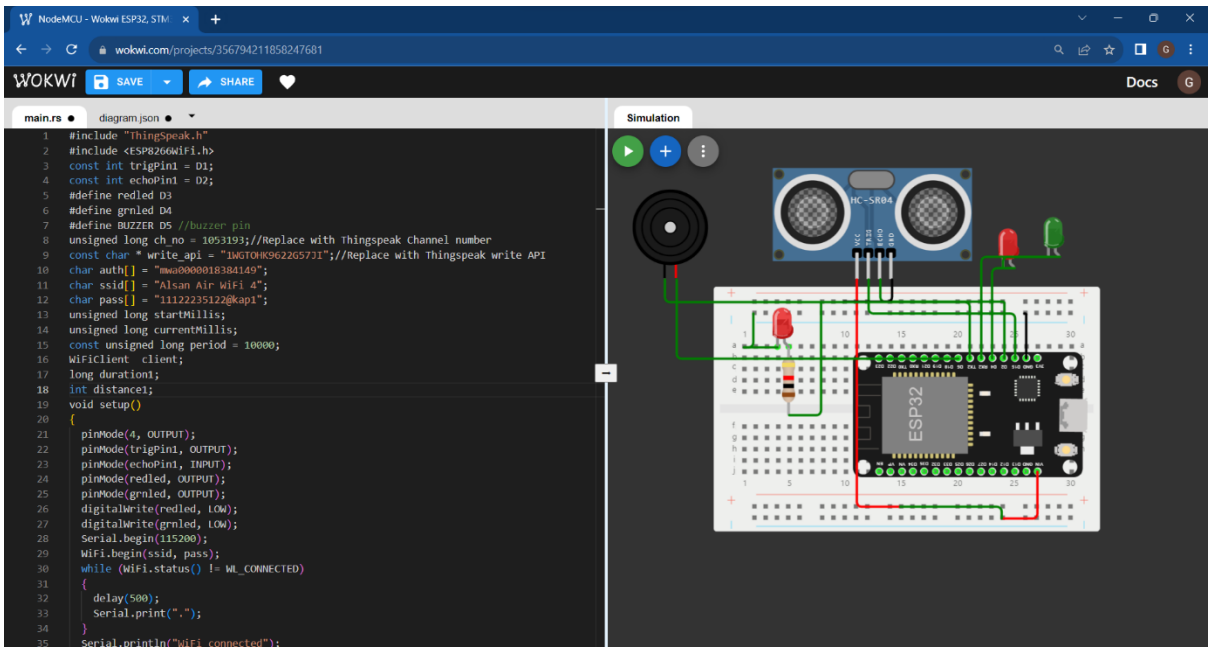
Working of the System:

- This IoT-based flood monitoring system uses an ESP8266 NodeMCU microcontroller to collect data from an ultrasonic sensor and send it to ThingSpeak for graphical monitoring and critical alerts. The NodeMCU also controls a red LED and buzzer, which are used to alert people in the event of a flood. A green LED is used to indicate normal conditions.
- In other words, the NodeMCU is the brain of the system. It reads the data from the ultrasonic sensor and determines if the water level has exceeded the threshold. If it has, the NodeMCU sends an alert to ThingSpeak, which will then send an alert to people via SMS, email, or another method. The NodeMCU also turns on the red LED and buzzer to alert people locally.
- The green LED is used to let people know that the water level is normal and there is no need for concern.

- This system is a valuable tool for flood monitoring and early warning. By providing early notice of rising water levels, it can help people to evacuate safely and protect their property.
- The system will generate an alert that will be sent to people via LED signals, a buzzer, SMS messages, and emails.

Simulating the system in Wokwi:

- Create a new Wokwi project and select the ESP8266 NodeMCU board.
- Add the above given components to our project.
- Connect the components to the NodeMCU board according to the block diagram.
- Write a program for the NodeMCU board. The program should read the data from the ultrasonic sensor and determine if the water level has exceeded the threshold. If it has, the program should generate the alerts and send the data to ThingSpeak.(Twillo and mailgun in case of sending SMS and mail)
- Upload the program to the NodeMCU board.
- Simulate the circuit in Wokwi.



Connecting the Twillo and Mailgun:

- To connect the above project to Twilio and Mailgun APIs for sending SMS and email messages, we will need to create accounts with Twilio and Mailgun.
- Once we have created accounts, we will need to obtain our Twilio account SID, Twilio auth token, Mailgun domain name, and Mailgun API key.
- Once we have obtained our Twilio and Mailgun credentials, we can update the NodeMCU program to send SMS and email messages.
- Write the following python script to connect the Twillo and Mailgun.

Python Script:

```
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)
```

Conclusion:

This project has demonstrated the design and implementation of an IoT-based flood monitoring system using an ESP8266 NodeMCU microcontroller and ThingSpeak. The system uses an ultrasonic sensor to detect the water level in a riverbed and sends the data to ThingSpeak for graphical monitoring and critical alerts. The system also uses a red LED and buzzer to alert people locally in the event of a flood.

The system can be easily extended to send SMS and email alerts using Twilio and Mailgun APIs. This would allow people to be alerted of rising water levels even if they are not near the riverbed.

The system can be used to help protect people and property from the devastating effects of floods by providing early warning of rising water levels.