FLOOD MONITORING AND EARLY WARNING SYSTEM

IOT_PHASE 5

1.Introduction

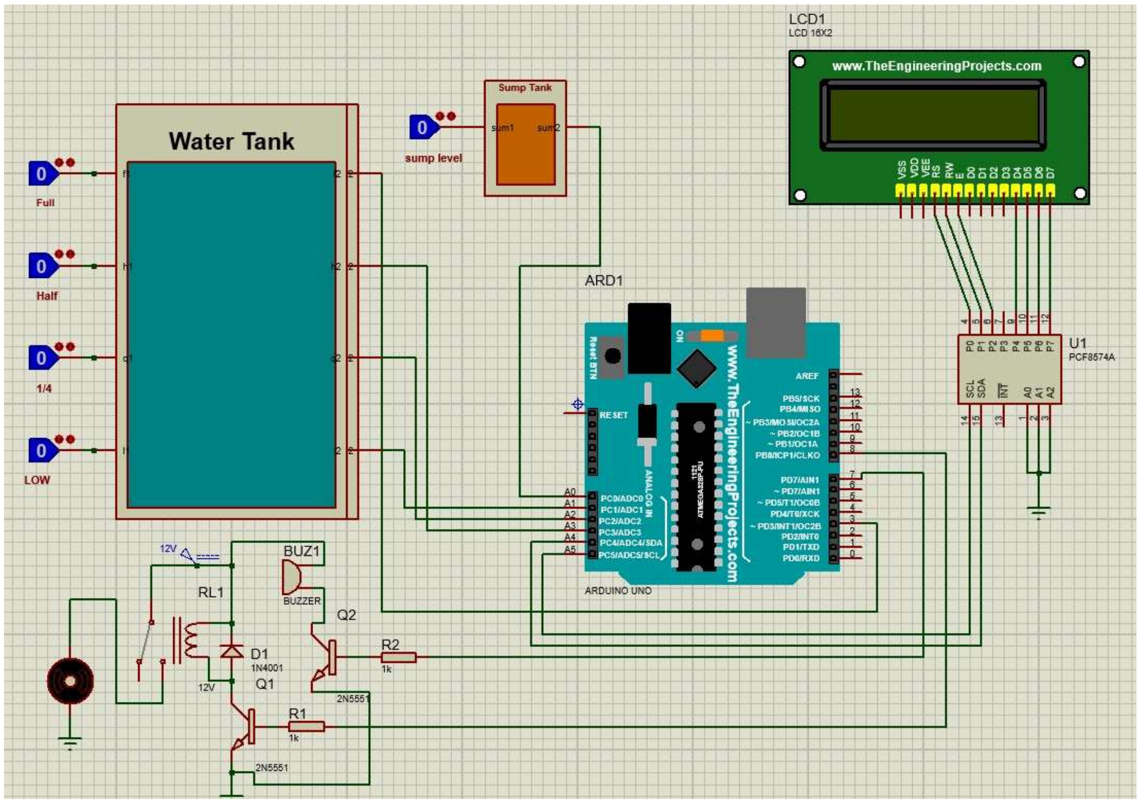
Floods are among the most destructive natural disasters, causing significant damage to infrastructure, property, and often resulting in loss of life. Timely and accurate flood monitoring and early warning systems are critical to mitigating the impact of floods and ensuring the safety of communities living in flood-prone areas. In this context, the integration of Internet of Things (IoT) technology offers a powerful solution to enhance flood monitoring and early warning systems.

The "Flood Monitoring and Early Warning System using IoT" project is designed to address this pressing issue by leveraging the capabilities of IoT technology. IoT refers to the interconnection of everyday physical objects to the internet, allowing them to collect and exchange data in real-time. In the case of flood monitoring, IoT devices, sensors, and data analytics play a pivotal role in continuously gathering relevant data, processing it, and providing actionable information for decision-makers and the public.

2.OBJECTIVES

The primary objective of the project is to bolster early warning capabilities by leveraging IoT technology. This involves the development of a robust system that can accurately detect and predict flood events in real-time. By doing so, the system aims to issue timely alerts to both local authorities and the public, enabling them to take proactive measures to mitigate the impact of floods.A crucial aspect of the project is the establishment of a comprehensive data collection and monitoring infrastructure. This includes the deployment of IoT sensors at strategic locations in flood-prone areas to continuously gather critical data, such as water levels, rainfall, and weather conditions. The goal is to ensure that the data collected is accurate, up-to-date, and readily accessible for analysis.Seamless integration with various communication channels is another key project objective. The system should be able to disseminate early flood warnings through multiple channels, including SMS, mobile apps, websites, and sirens. This ensures that the information reaches a wide audience and allows for rapid response.Collaboration with local authorities and emergency services is pivotal for an effective flood response. The project aims to establish strong partnerships with these organizations to ensure a coordinated approach to flood events. This includes the development of evacuation plans, emergency response protocols, and resource allocation.

3.CIRCUIT DIAGRAM



4.PROJECT CODE

```
#include 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);
}
  #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'
```

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)
```

5.PLATFORM DEVELOPMENT

HTML, CSS, and JavaScript Web Application:

Create a folder for your web application project and create the following files within it:

- index.html for the main HTML structure.
- Web.py for code integration.

Index.html

```
<!DOCTYPE html>

<html>

 <head>

   <script type="text/javascript">

     function load()

     {

     setTimeout("window.open('http://127.0.0.1:5000/', '_self');", 500);

     }

     </script>

    <style>

    .Line2 {

     border-left: 180px solid lightgray;

     height:{{ blue_line }}px;

    }

    .Line1 {
      border-left: 180px solid #28a1f7;
      height: {{ gray_line }}px;
    }
    .Div1 {
    width: 100%;
    height: 100%;
    margin: 1px 0;
    background-color: #ffffff;
    background-image: url({{url_for('static', filename='water_tank.jpeg')}});
    }
    </style>
</head>
<body onload="load()">
  <center>
    <br>
    <br>
  <table name="Table1" >
    <tr>
      <td class="Div1">
        <table align="bottom" name="Table2"  height=420px style="border-spacing:
       44px;">
          <tr>
            <td >
              <br>
```

```
 <br> <br>
                <br>
               <div class="Line2"></div>
               <div class="Line1"></div>
             </td>
           </tr>
         </table>
       </td>
     </tr>
   </table>
   </center>
  </body>
</html>
```

Web.py

You will need to install flask to receive data from the Arduino and process it.

pip install Flask

Code:

```
import time
from flask import *
import RPi.GPIO as GPIO
from datetime import datetime
```

```
#-----------------------------------------------------------

# Setup

app = Flask(__name__)

TRIG = 11

ECHO = 12


def setup():

    GPIO.setmode(GPIO.BOARD)

    GPIO.setup(TRIG, GPIO.OUT)

    GPIO.setup(ECHO, GPIO.IN)


def distance():

    GPIO.output(TRIG, 0)

    time.sleep(0.000002)

    GPIO.output(TRIG, 1)

    time.sleep(0.00001)

    GPIO.output(TRIG, 0)

    while GPIO.input(ECHO) == 0:

        a = 0

    time1 = time.time()

    while GPIO.input(ECHO) == 1:

        a = 1

    time2 = time.time()
```

```
        during = time2 - time1

        return during * 340 / 2 * 100

#-----------------------------------------------------------

# Flask

@app.route('/')

def index():

  setup()

  dis = distance()

  percent = (dis-3)/.24

  blue_line = round(percent*2.6)

  gray_line = 260-blue_line

  return render_template('index.html', blue_line=blue_line,
gray_line=gray_line)
```

To integrate above code into the early warning platform:

```
def index():

  return app.send_static_file('index.html')

@app.route('/get_data')

def get_data():

  water_level = 50  # Example value, replace with your actual data

  return jsonify({'water_level': water_level})

if __name__ == '__main__':

  app.run(host='0.0.0.0', port=5000)
```

5.FUTURE PROJECT REPLICATIONS AND IDEAS

- **IoT Sensors:** The project utilizes various sensors such as water level sensors, rainfall sensors, and weather sensors to collect data related to the water levels, precipitation, and weather conditions in flood-prone areas. These sensors are strategically placed at critical locations like riverbanks, reservoirs, and rain gauge stations.
- **Data Transmission:** The sensor data is transmitted in real-time to a central data hub or a cloud-based platform using wireless communication technologies such as Wi-Fi, LoRa, or cellular networks. This ensures that data can be accessed remotely and analyzed rapidly.
- **Data Analytics:** Once the data is collected, it is processed and analyzed to assess flood risk and predict potential flooding events. Data analytics algorithms are used to interpret the sensor data and generate actionable insights.
- **Early Warning System:** When the system detects elevated water levels, heavy rainfall, or other factors indicating a potential flood event, it triggers automated early warning alerts. These alerts can be sent to local authorities, emergency services, and the public through various communication channels like SMS, mobile apps, and sirens.
- **Public Engagement:** A vital aspect of the project is keeping the public informed about the flood situation. This can be achieved through mobile apps, websites, and public awareness campaigns. By providing clear and timely information, people can take necessary precautions.

6.Benefits

- **Data-Driven Decision-Making:** Decision-makers have access to real-time data and analytics, enabling them to make informed choices for disaster response and mitigation.

- **Cost Efficiency:** The use of IoT technology can lead to cost savings as it reduces the need for constant human monitoring and manual data collection.
- **Timely Response:** Rapid alerts enable emergency services to respond promptly, saving lives and minimizing damage.
- **Community Resilience:** By involving the public and increasing awareness, communities become more resilient to flood events and are better equipped to protect themselves.

7.conclusion

In conclusion, the "Flood Monitoring and Early Warning System using IoT" project represents a crucial and innovative approach to addressing the challenges posed by flooding in vulnerable areas. By leveraging the capabilities of Internet of Things (IoT) technology, the project aims to significantly enhance flood monitoring and early warning capabilities, ultimately contributing to the protection of lives and property.The project's objectives are rooted in the principles of accuracy, timeliness, and community engagement. Through the deployment of IoT sensors, data collection, and advanced data analytics, the system aims to provide real-time flood risk assessments and issue early warnings. This empowers local authorities, emergency services, and the public to take proactive measures, reducing the devastating impact of floods.Collaboration with local authorities and emergency services ensures a coordinated response to flood events, with well-defined protocols and community safety in mind. The project also recognizes the importance of public awareness and education, acknowledging that informed communities are better prepared and resilient.The system's adaptability, scalability, and cost-efficiency make it a sustainable solution that can be tailored to different geographical and climatic conditions. It also offers opportunities for continuous improvement through ongoing evaluation and feedback mechanisms. It stands as an essential endeavor in safeguarding our communities and fostering disaster resilience.