

Flood Monitoring and Early Warning system with Wokwi

Name: Keerthana S

NM ID:B81DBD9FF719AC6A98D4D3D4BCD325F8

Email: 950321104025@gracecoe.org

1. Data Collection:

Gather historical data related to water levels, rainfall, and other relevant parameters in flood-prone areas. This dataset will be used for training and testing your machine learning model.

Date	Water_Level	Rainfall	Temperature	Humidity	Flood_Label	
-----	-----	-----	-----	-----	-----	
2023-10-01	2.5	0.2	20.0	60	0	
2023-10-02	2.7	0.1	21.5	58	0	
2023-10-03	3.2	0.5	19.8	65	0	
2023-10-04	4.0	1.2	18.6	70	0	
2023-10-05	5.2	2.0	17.3	75	1	
2023-10-06	6.1	0.8	18.9	72	1	
2023-10-07	7.0	0.3	20.2	68	1	
2023-10-08	6.5	0.2	22.0	64	1	
2023-10-09	5.8	0.1	21.3	70	0	
2023-10-10	5.3	0.3	20.5	75	0	

Date: The date of the data entry.

Water_Level: The water level measured at a specific point.

Rainfall: The amount of rainfall in millimeters.

Temperature: The temperature in degrees Celsius.

Humidity: The relative humidity in percentage.

Flood_Label: Binary label indicating if a flood occurred (1) or not (0).

With these additional features, you can now implement more sophisticated machine learning techniques like Random Forests, Support Vector Machines, or Neural Networks for flood prediction.

2. Feature Engineering:

Time-based Features: Extract features like day of the week, month, and hour to capture temporal patterns.

Lagged Variables: Include lagged values of water levels and rainfall to capture trends.

Rainfall Accumulation: Calculate accumulated rainfall over specific periods (e.g., daily, weekly).

Geospatial Information: If available, include geographical features like latitude, longitude, and elevation.

Statistical Measures: Mean, median, variance, etc., of water levels and rainfall can provide valuable information.

3. Model Training:

Select an appropriate machine learning algorithm for time series forecasting. Common choices include:

ARIMA (AutoRegressive Integrated Moving Average): Suitable for univariate time series forecasting.

LSTM (Long Short-Term Memory): A type of recurrent neural network (RNN) well-suited for sequence prediction tasks.

Prophet: Developed by Facebook, Prophet is designed for forecasting time series data that exhibit seasonal patterns.

4. Model Evaluation:

Split your dataset into training and testing sets. Train your model on the training set and evaluate its performance on the testing set using appropriate metrics:

Mean Absolute Error (MAE): Measures the average absolute errors between actual and predicted values.

Root Mean Squared Error (RMSE): Gives more weight to large errors, which can be useful if large errors are particularly undesirable.

R-squared (R²): Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

5. Implementation in Wokwi:

While Wokwi is primarily for hardware simulation, you can use it to simulate the behavior of the flood monitoring system's hardware components. Create a virtual environment with components like sensors (for water levels), a microcontroller (arduino), and actuators (for warnings).

Code:

```
#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);
    lcd.setCursor(0, 0);
    lcd.print("Flood Monitoring");
    lcd.setCursor(0, 1);
    lcd.print("Alert!!");
    delay(5000);
    lcd.clear();
}
void loop()
{
    long dur;
    long dist;
    long per;
```

```

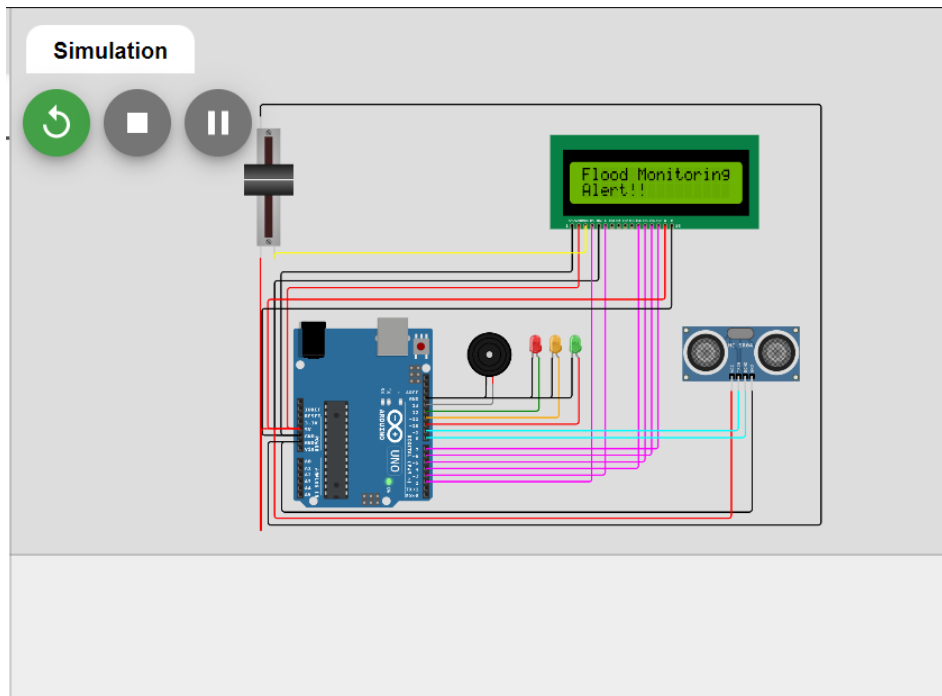
digitalWrite(out, LOW);
delayMicroseconds(2);
digitalWrite(out, HIGH);
delayMicroseconds(10);
digitalWrite(out, LOW);
dur = pulseIn(in, HIGH);
dist = (dur * 0.034) / 2;
per = map(dist, 10.5, 2, 0, 100);
if (per < 0)
{
    per = 0;
}
if (per > 100)
{
    per = 100;
}
Serial.print("Water Level:");
Serial.println(String(per));
lcd.setCursor(0, 0);
lcd.print("Water Level:");
lcd.print(String(per));
lcd.print("% ");
if (dist <= 3)
{
    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 (dist <= 10)
{
    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
{
    lcd.setCursor(0, 1);
    lcd.print("Green Alert! ");
    digitalWrite(green, HIGH);
    digitalWrite(orange, LOW);
    digitalWrite(red, LOW);
    digitalWrite(buzz, LOW);
}
}
}

```



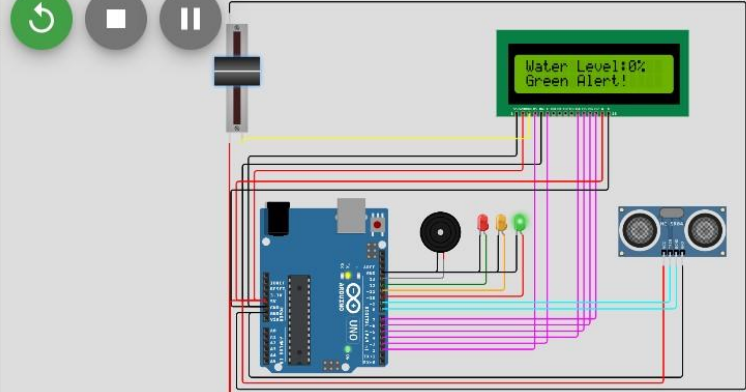
6. Integration:

Integrate the trained machine learning model into your system. The model will take inputs from sensors (water levels, rainfall) and provide predictions or alerts based on the trained patterns.

Docs

J

Simulation



03:10.709 78%

Water Level:0

Water Level:0

Water Level:0

Water Level:0

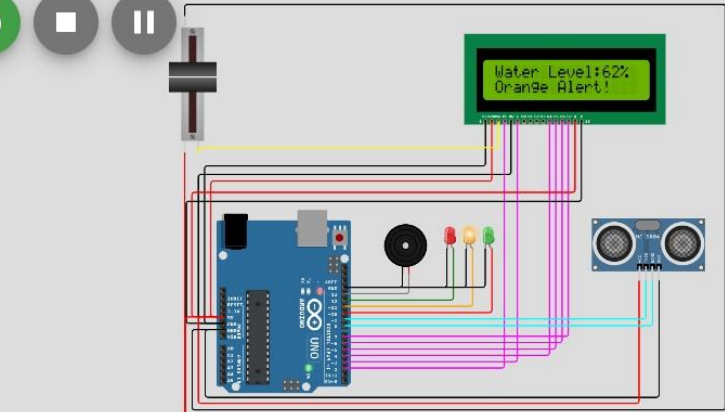
Water Level:0

Water Level:0

Docs

J

Simulation



00:24.412 100%

Water Level:0

Water Level:0

Water Level:0

Water Level:0

Water Level:50

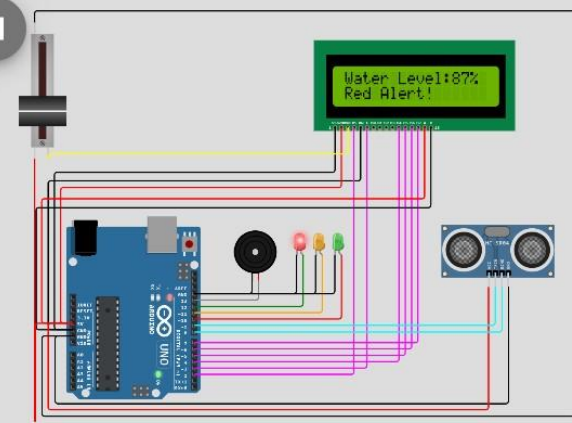
Water Level:62

Water Level:62

Docs
J

Simulation

⌚ 02:39.667
🔄 101%



```

Water Level:0
Water Level:0
Water Level:0
Water Level:0
Water Level:100
Water Level:87
Water Level:87

```

7. Testing and Fine-tuning:

Test the integrated system with simulated inputs in Wokwi. Fine-tune the model and system behavior based on the simulated results.

Docs
J

WOKWI
SAVE
SHARE
FMEW

sketch.ino
diagram.json
libraries.txt
Library Manager

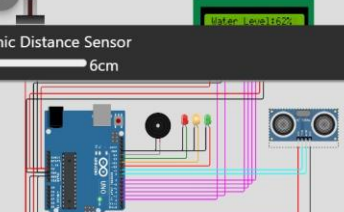
```

89   digitalWrite(buzz, LOW);
90   delay(3000);
91 }
92 else
93 {
94   lcd.setCursor(0, 1);
95   lcd.print("Green Alert! ");
96   digitalWrite(green, HIGH);
97   digitalWrite(orange, LOW);
98   digitalWrite(red, LOW);
99   digitalWrite(buzz, LOW);
100 }
101 }

```

Simulation

⌚ 00:45.560
🔄 100%



```

Water Level:0
Water Level:50
Water Level:62
Water Level:62
Water Level:62
Water Level:62
Water Level:62

```

8. Deployment:

Once satisfied with the simulation, the real-world deployment would involve interfacing with physical sensors and actuators. Make sure to follow best practices for hardware integration and deployment in flood-prone areas.

Conclusion:

This project aims to create an Internet of Things (IoT) flood monitoring and early warning system using a water level sensor, microcontroller, Wi-Fi module, and cloud platform. The system will continuously monitor the water level and send real-time data to a cloud platform. In case of a flood, the system will trigger notifications to alert users, enabling them to take necessary precautions.