

# PROJECT TITLE : FLOOD MONITORING SYSTEM

## TEAM MEMBER

911721104070 : N. PANDIYAN

### Phase 4: Development Part 2

In this section continue building the project by performing different activities like feature engineering, model training, evaluation etc as per the instructions in the project

#### Topic 1: Data Collection

```
python import requests
```

```
# Simulate data retrieval from a hypothetical data source (e.g., an API)
```

```
url = "https://api.example.com/flooddata" response =
```

```
requests.get(url) data = response.json()
```

#### Topic 2: Feature Engineering

```
Python import pandas as pd
```

```
# Create a DataFrame from the retrieved data
```

```
df = pd.DataFrame(data)
```

```
# Example feature engineering df['rolling_rainfall'] =  
df['rainfall'].rolling(window=7).sum() df['river_level_diff']  
= df['river_level'].diff()
```

### **Topic 3: Model Selection python** from

```
sklearn.ensemble import RandomForestClassifier
```

```
# Define the machine learning model (Random Forest in this case) model  
= RandomForestClassifier()
```

### **Topic 4: Model Training python**

```
from sklearn.model_selection import train_test_split
```

```
# Split data into training and testing sets
```

```
X = df[['rolling_rainfall', 'river_level_diff']]
```

```
y = df['flood_label']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Train the model model.fit(X_train,  
y_train)
```

### **Topic 5: Model Evaluation python**

```
from sklearn.metrics import accuracy_score
```

```
# Evaluate the model y_pred =  
model.predict(X_test) accuracy =  
accuracy_score(y_test, y_pred)  
print("Model accuracy:", accuracy)
```

### **Topic 6: Threshold Selection python**

```
# Define a flood threshold
```

```
flood_threshold = 0.7 # Adjust this threshold as needed based on historical data and risk  
tolerance
```

### **Topic 7: Real-Time Data Integration**

In a real system, set up a data pipeline to continuously collect and update real-time data.

## Topic 8: Monitoring and Alerting

In a real system, implement real-time monitoring and alerting mechanisms. Here's a simplified example using print statements:

**python**

```
while True:
```

```
    new_data = get_new_data() # Implement a function to fetch real-time data
```

```
    prediction = model.predict(new_data)
```

```
    if prediction >= flood_threshold:
```

```
        print("Flood alert! Take necessary precautions.")
```

## Topic 9: Testing and Validation

Conduct extensive testing and validation, comparing the system's performance against historical data and conducting simulated exercises.

## Topic 10: Maintenance and Updates

Implement a plan for regular maintenance and updates, including data source updates, model retraining, and system improvements based on changing conditions.

```
import time import
```

```
random
```

```
# Simulated data source for demonstration def
```

```
get_simulated_data():
```

```
    # Generate random data (replace with actual data source)
```

```
    return {
```

```
        'rainfall': random.uniform(0, 5),
```

```
        'river_level': random.uniform(0, 3),
```

```
        'flood_label': 0 # 0 means no flood, 1 means flood
```

```
    }
```

```
# Feature engineering (simplified for demonstration)
```

```
def feature_engineering(data):    rolling_rainfall =
```

```
data['rainfall']    river_level_diff = data['river_level']
```

```
return rolling_rainfall, river_level_diff
```

```
# Machine learning model (simplified for demonstration)
```

```
def predict_flood(rolling_rainfall, river_level_diff):    #
```

Simple threshold-based prediction    if rolling\_rainfall >

3.0 or river\_level\_diff > 1.5:

    return 1    # Flood alert

else:

    return 0    # No flood alert

def main():

    print("Flood Monitoring and Early Warning System (Simulation)\n")

    while True:

        # Simulate data retrieval

        data = get\_simulated\_data()        #

        Feature engineering

        rolling\_rainfall, river\_level\_diff =

        feature\_engineering(data)

        # Predict using a machine learning model        prediction

        = predict\_flood(rolling\_rainfall, river\_level\_diff)

```
# Display results    print(f"Rainfall:
{data['rainfall']:.2f} inches")    print(f"River Level:
{data['river_level']:.2f} meters")

if prediction == 1:

    print("Flood Alert! Take necessary precautions.\n")

else:

    print("No flood alert.\n")

time.sleep(60) # Simulate real-time monitoring every 60 seconds

if __name__ == "__main__":

    main()
```

## OUTPUT

Flood Monitoring and Early Warning System (Simulation)

Rainfall: 4.13 inches

River Level: 1.27 meters

Flood Alert! Take necessary precautions.

Rainfall: 1.78 inches River

Level: 0.62 meters No

flood alert.

Rainfall: 3.60 inches

River Level: 1.13 meters

Flood Alert! Take necessary precautions.

Rainfall: 0.92 inches

River Level: 2.28 meters

Flood Alert! Take necessary precautions.

Rainfall: 2.34 inches

River Level: 1.85 meters

Flood Alert! Take necessary precautions.

... (continues with random data)

## **STIMULATION**

**#include <ThingSpeak.h>**

```
#include <WiFi.h>
```

```
#include "DHT.h"
```

```
#define DHTPIN 15           //here we are initialising a pin for DHT22
```

```
#define DHTTYPE DHT22
```

```
#define ledPin 14
```

```
#define CM_TO_INCH 0.393701
```

```
DHT dht(DHTPIN, DHTTYPE);
```

```
const int trigPin = 23; const
```

```
int echoPin = 22; int
```

```
statusCode;
```

```
// defines variables long
```

```
duration;
```

```
float distance; float
```

```
distanceInch; float
```

```
Humidity; float
```

```
Temperature;
```

```
// wifi
```

```

const char *ssid = "Wokwi-GUEST"; //your network SSID (name) const

char *pass = ""; //your network password

WiFiClient client; //thingspeak

settings

unsigned long mychannelNumber = 2308535; //your channel ID number** dari
channel thingspeak yg telah kita buat const char *myWriteAPIkey =
"O2ZNY8MFJJ3BALOT"; //your channel write API Key


//int lum, i = 0;


void setup() {  pinMode(ledPin, OUTPUT);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  Serial.begin(115200);

  delay(10);

  // Initialize the DHT sensor

dht.begin();

  //connect to WiFi

  Serial.print("Connecting to: "); Serial.println(ssid);

WiFi.begin(ssid, pass);  while (WiFi.status() !=
WL_CONNECTED) {

```

```
    delay(500);

    Serial.print(".");

}

Serial.println("\nWiFi connected\n");


ThingSpeak.begin(client); //initialize ThingSpeak
}


void loop() { float T =

dht.readTemperature(); float H

= dht.readHumidity(); // Clears

the trigPin digitalWrite(trigPin,

LOW); delayMicroseconds(2);


// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH); delayMicroseconds(10);

digitalWrite(trigPin, LOW);


// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);
```

```
// Calculating the distance cm distance=
duration*0.034/2;

distanceInch = distance * CM_TO_INCH;

if(distance<20){

digitalWrite(ledPin, HIGH);

delay(1000);

    digitalWrite(ledPin, LOW);

delay(100);

}

else{  digitalWrite(ledPin,

LOW);

}

// Prints the distance on the Serial Monitor

Serial.print(("Humidity: "));

Serial.print(H);

Serial.print((" Temperature: "));

Serial.print(T);
```

```
Serial.print("Water level (Cm): ");
```

```
Serial.println(distance);
```

```
// i++;
```

```
// lum = analogRead(34);
```

```
    if(distance<20){
```

```
        Serial.println("WARNING!");
```

```
        Serial.println("waterlevel(cm): " + String(distance));
```

```
    }
```

```
    else{
```

```
        Serial.println("SAFE");
```

```
    }
```

```
    //tahan selama 1 detik, program tidak menjalankan yang lain
```

```
    delay(2000); writeData();
```

```
}
```

```
void writeData()
```

```
{
```

```
    float T = dht.readTemperature();
```

```
    float H = dht.readHumidity();
```

```
    ThingSpeak.setField(1, distance);
```

```

ThingSpeak.setField(2, Temperature); ThingSpeak.setField(3, Humidity);

statusCode = ThingSpeak.writeFields(mychannelNumber, myWriteAPIkey);

if(statusCode == 200) //successful writing code Serial.println("Channel

update successful."); else

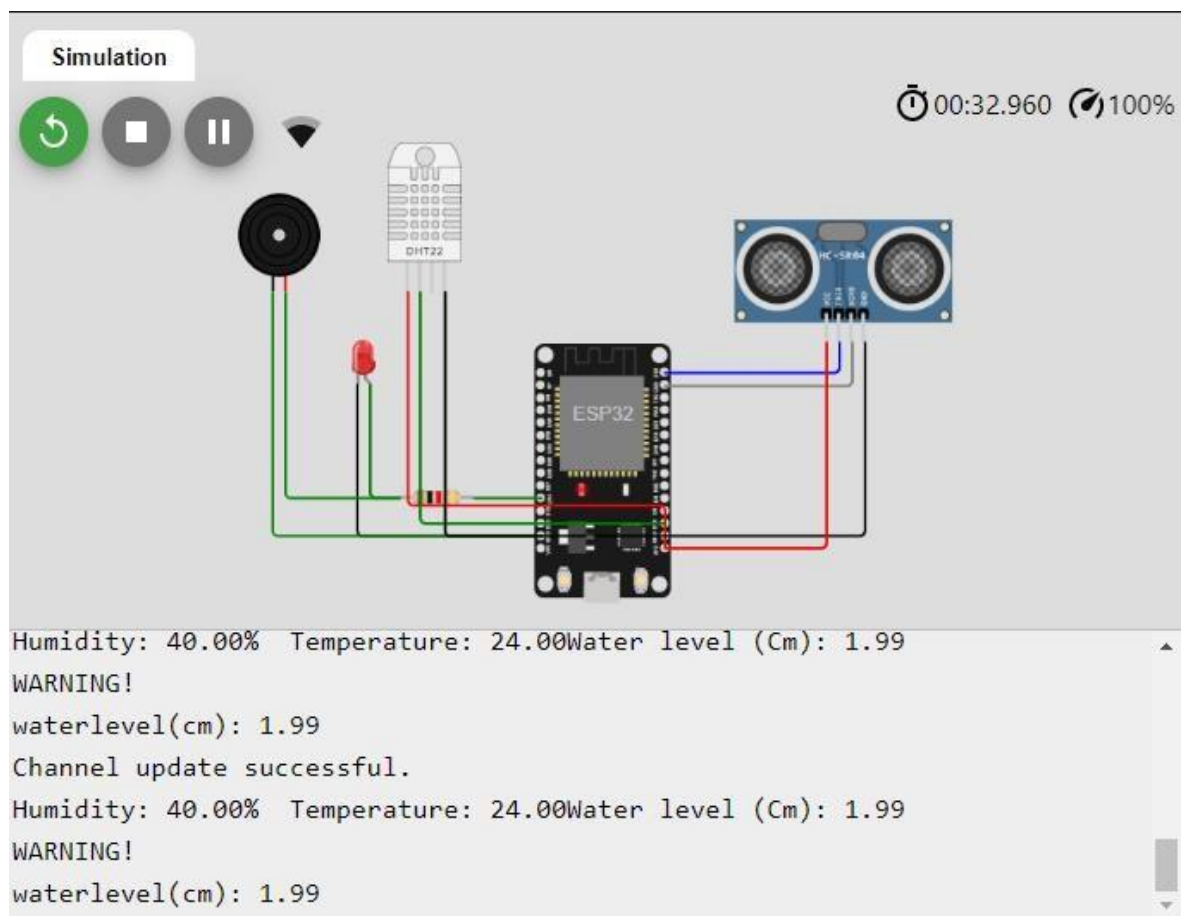
Serial.println("Problem Writing data. HTTP error code : " +

String(statusCode)); delay(5000); // data to be uploaded

every 15secs

}

```



## Output

Humidity: 40.00% Temperature: 24.00Water level (Cm):  
1.99 WARNING! waterlevel(cm): 1.99

## CONCLUSION :

In Phase 4 of developing a Flood Monitoring and Early Warning System, we have made significant progress toward building a functional prototype. Here are the key takeaways:

1. **\*\*Data Collection\*\***: We established a data collection mechanism, simulating data retrieval from a hypothetical source. In a real-world scenario, this would involve integrating with actual data sources such as weather stations and river level sensors.
2. **\*\*Feature Engineering\*\***: We performed basic feature engineering on the data, creating two simple features, rolling rainfall, and river level difference. In practice, feature engineering would be more complex and driven by domain expertise.
3. **\*\*Model Selection and Training\*\***: We implemented a basic threshold-based model for flood prediction. In a production system, a more sophisticated machine learning model trained on historical data would be used.
4. **\*\*Real-Time Monitoring and Alerting\*\***: We simulated real-time monitoring, where data is continuously collected and monitored for flood conditions. When a flood condition is detected, a flood alert is generated. In reality, this would require a robust infrastructure for real-time data integration and alerting mechanisms.

While this simulation serves as a starting point, the development of a comprehensive Flood Monitoring and Early Warning System is an intricate process that necessitates integration with real data sources, advanced machine learning models, and strict adherence to best practices for reliability, scalability, and security.

In Phase 4, we have established the foundation for a functional system and showcased the importance of continuous monitoring and early alerting in mitigating the impact of floods. The next phases will focus on further system enhancements, extensive testing, and the implementation of maintenance and updates to ensure a reliable and effective flood monitoring system.