LOWPAN AND AD-HOC NETWORKING

NATURAL DISASTER MONITORING AND EARLY WARNING SYSTEM

MODULE CODE	**CT120-3-2-LPAN**
INTAKE CODE	**APD2F2409IT(IOT)**
LECTURER NAME	**KAMALANATHAN A/L SHANMUGAM**
HAND IN DATE	**MAY 26TH 2025**
STUDENT ID	**TP072606**
STUDENT NAME	**ELEANOR PERMATA FRY**

TABLE OF CONTENTS

1. ABSTRACT

This project highlighted the details and the design behind a Natural Disaster Monitoring and Early Warning System made with Internet of Things (IoT) sensors and microcontroller that was able to provide real-time monitoring and early warning alerts for natural disasters, especially landslides and earthquakes. This system was specifically designed for disaster-prone areas to allow continuous monitoring of several environmental parameters.

The ESP32 microcontroller was used within this project to act as a central processing unit. The ESP32 was integrated with several sensors, including the DHT11 temperature and humidity sensor, soil moisture sensor, vibration sensor (SW-420), and MPU6050 gyroscope. These sensors would continuously monitor the environment's conditions to detect any potential natural disasters.

The system architecture consisted of two main components: the data collection and cloud communication. The ESP32 would collect data from the sensors and transmit it over Wi-Fi to an IoT platform which acts as a dashboard, here ThingSpeak will be used as the platform to monitor and visualize the data collected. To add, the system would provide immediate alerts through buzzer alarms, LEDs, and emergency push notifications through Telegram when critical conditions were detected.

The system's main objective is to assist communities and authorities in maintaining safety from any natural disasters. This system would be able to reduce causalities by providing real-time warnings about any potential environmental disasters, allowing efficient evacuation and emergency response.

Through this system, communities and authorities were able to receive reliable and real-time environmental monitoring solution through the help of IoT technology. The existence of this system would be able to enhance community safety and preparedness for these disasters while providing the authorities with important data for emergency response planning.

2. INTRODUCTION

2.1 BACKGROUND

On January 20th, 2025, heavy rainfall triggered floods and landslides in Petungkriyono Village, located in Pekalongan Regency, Central Java, Indonesia. The disaster claimed the lives of 25 residents and displaced 300; about a dozen were left missing under destroyed infrastructure (Agencies, 2025). The rural communities of Petungkriyono easily become vulnerable targets for these disasters, as most of the residents are aged and don't have access to mobile devices, leaving them without proper monitoring or warning systems. From the beginning of 2025 until today, about 1,150 natural disasters have happened, and about 3.4 million citizens were left displaced. Out of all the disasters, a third were caused by extreme weather, landslides, and earthquakes (Geoportal Data Bencana Indonesia, 2025).

With the increasing frequency of natural disasters due to climate change, a lot of communities worldwide faces the difficulties and effects from landslides, and earthquakes. These disasters can occur at any moment with little warning, resulting in great amount of lives losses and damage in properties (Arrel, et al., 2024). Due to limited coverage, traditional monitoring methods are often prescribed as unreliable, a lot of these methods have delay in their responses and lack in real-time data transmission capabilities (Gaitan & Hojbota, 2020).

Traditional or current disaster monitoring systems have limitations in their effectiveness as they are only able to focus on a single parameter and requires manual data collection. Modern IoT technology creates an opportunity in developing early warning systems that are able to monitor multiple environmental parameters at the same time (Esposito, Palma, Belli, Sabbatini, & Pierleoni, 2022). The integration of multiple sensors with cloud-based analytics and automated alerts can significantly improve disaster preparedness and emergency responses. And unlike traditional warning systems that may be expensive and limited in quantity, IoT-based systems are cost-effective and can provide continuous monitoring without the need of manual labor (Esposito, Palma, Belli, Sabbatini, & Pierleoni, 2022).

As communities becomes more vulnerable to these natural disasters, there is an urgent need for a reliable monitoring system that can also provide early warnings. IoT technology allows the existence of such systems that provides the benefits of reliability in data transmission, immediate alerts and be able to operate continuously.

2.2 OBJECTIVES

The main objective of this project to be able to provide to communities and authorities an environmental monitoring system using IoT technology and several sensors, that is able to capture data in real-time. The primary aim to improve disaster preparedness and reduce the risks of losses from natural disasters.

Other objectives include:

- Develop multi-sensor system for environmental monitoring
- Implement cloud-based monitoring and real-time data transfer
- Develop an automated early warning system with multiple alert features
- Provide historical data so that trends can be analyze and use for prediction
- Create several emergency alerts for immediate action and response

2.3 SYSTEM SCOPE

The current scope of this system includes comprehensive environmental monitoring for the detection of natural disasters, such as landslides and earthquakes. The system is integrated with several sensors, including the DHT11 for temperature and humidity monitoring, the soil moisture for groundwater monitoring, the vibration sensor for seismic activity detection, and the MPU6050 gyroscope for ground tilt measurement.

The monitoring system provides continuous monitoring for these environmental parameters and signals authorities when there’s a potential disasters or any indications of a developing disasters.

2.4 FUNCTION

The DHT11 sensor continuously monitors the humidity and temperatures, which can also sever to predict any weather conditions or events. The soil moisture sensors analyze the ground saturation levels, which relates to the possibility of landslides to happen. The vibration sensor detects ground movement or seismic activity that may indicate an earthquake or landslide.

Lastly, the MPU6050 gyroscope measures the ground tilt angles, if it passes a certain angle, it will be indicated as a landslide.

By integrating these sensors, the disaster monitoring system provides communities and authorities about potential risks and alerts of any disasters. The system allows the authorities to take necessary preventive actions and emergency responses to ensure the safety of the communities.

3. DETAILS OF SOFTWARE AND HARDWARES

3.1 SOFTWARE

ARDUINO IDE

Arduino IDE servers as a platform that's able to program any microcontrollers, including the ESP32. It makes use of the C++ programming language for communication protocols between the IoT devices. The platform also provides extensive libraries for interfacing sensors, wireless communications, and data processing (Arduino, n.d.). Arduino's is an open-source software, allowing community-developed code examples and libraries, making it perfect for prototyping and deploying IoT systems. The platforms scalability and cost-effectiveness make it suitable for the development and deployment of this disaster monitoring system.

THINGSPEAK

ThingSpeak servers as the cloud based IoT platform that's able to collect, store, analyze and visualize data from IoT devices. It provides RESTful APIs for real-time data collection from the ESP32 to the cloud for continuous monitoring (MATLAB & Simulink, n.d.). The platform offers a variety of ways to visualize the data collected from graphs, charts, and real-time dashboards that supports monitoring of environmental trends. Advanced analytics for predictive modelling and analysis can also be done within ThingSpeak as its integrated with MATLAB. ThingSpeak can monitor multiple sensors to be visualized within one dashboard, making it ideal for community-wide disaster monitoring networks (ThingSpeak, n.d.).

TELEGRAM

Telegram serves as an alert communication system in this project. By integrating a Telegram bot with the ESP32, real-time notifications are sent directly to the user's mobile devices or

group chats when a landslide or earthquake risk is detected. Using HTTPS and secure APIs, the ESP32 posts a warning message *"⚠ EARLY WARNING: Landslide or Earthquake Detected*!" through the Telegram Bot API. This allows for instant alerts to residents, emergency responders, or monitoring stations. Telegram enhances responsiveness of the system by ensuring real-time, direct communication during environmental emergencies.

Meanwhile, the ESP32 acts as a central processing unit and collection server for data, gathering the information from all the connected sensors and transmitting in to ThingSpeak through Wi-Fi connectivity.

3.2 HARDWARE

Sensor Name	Qty	Description

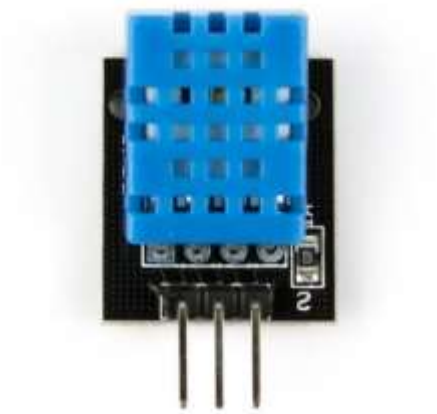
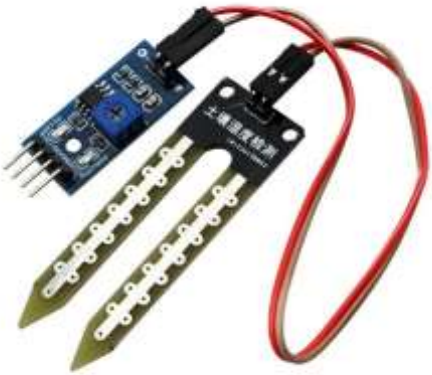
 *Figure 1. DHT11 Temperature and Humidity Sensor*	1	**Features:** Able to measure temperature from 0 – 50 degrees Celsius and humidity from 20% to 90% with an accuracy of +-1. Calibrated with digital signal output, fast in response, has an anti-interference capability and low in power consumption. **Function:** Monitors weather conditions that may contribute to the natural disasters and provide environmental data for future prediction.
 *Figure 2. Soil Moisture Sensor*	1	**Features:** Operates at voltage 2.5V – 7.0V with an analog output (HIGH/LOW). This sensor is double sided and corrosion resistant. **Function:** Monitors soil saturation levels to provide important data about ground stability and water inflation rates, as well as to predict an upcoming landslide.

 *Figure 3. Vibration Sensor Module SW-420*	1	**Features:** Supports 3.3V to 5V power and has a digital output. Compatible with many controllers including the ESP32. Integrated with adjustable sensitivity. **Function:** Detects seismic activity or ground movement that may indicate an earthquake or potential landslide.
 *Figure 4. GY-521 MPU6050 6DOF Accelerometer + Gyro*	1	**Features:** Combines a MEMS 3-axis gyroscope and a 3-axis accelerometer. Capable of processing 9-axis Motion Fusion algorithms. **Function:** Measures tilt angles of the ground for slope monitoring and landslide early warning.

Table 1. Hardware Components

Control Board	Qty	Description
 *Figure 5. NodeMCU ESP32*	1	**Features:** Dual core 240MHz processor. Built in Wi-Fi and Bluetooth with 3 GPIO pins and ADC/DAC capabilities. **Function:** Servers as the main control unit to collect data from the sensors, process and send alerts, as well as facilitate cloud communication through its built in features.

Table 2. Board Component

Output Components	Qty	Description

 *Figure 6. LED 3mm Red, Yellow, Green*	3	Provides visual alerts with different color-codes, green for normal conditions, yellow for warning, and red for danger.
 *Figure 7. Piezo Buzzer*	1	Generates audio alerts with different tone patterns for various alert levels, making sure alerts are noticed in a noisy environment.
 *Figure 8. Breadboard*	2	Facilitate the connections between components during the development and testing stage.

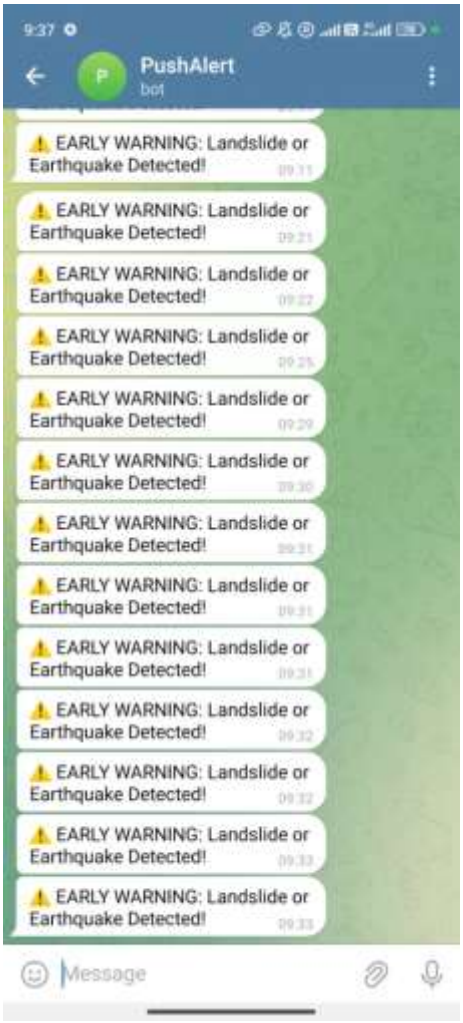
 *Figure 9. Jumper Wires (M to M, M to F)*	-	Provide connections between sensors and ESP32 control board.
 *Figure 10. Telegram Bot Channel*	-	Provide immediate push notifications for early warnings.

Table 3. Output Components

4. CODE

```
#include <Wire.h>
#include <MPU6050.h>
```

```
#include <DHT.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>

// ----- Pin Definitions -----
#define DHTPIN 15
#define DHTTYPE DHT11
#define SOIL_MOISTURE_PIN 34
#define VIBRATION_PIN 35
#define BUZZER_PIN 13
#define RED_LED 25
#define YELLOW_LED 26
#define GREEN_LED 27
#define BUTTON_PIN 12  // Buzzer silence button

// ----- WiFi & API Keys -----
const char* ssid = "<SSID_HERE>";
const char* password = "<PASSWORD_HERE>";
const String THINGSPEAK_API_KEY = "<THINGSPEAK_API_KEY_HERE>";

// Telegram Bot credentials
const char* TELEGRAM_BOT_TOKEN = "<ADD_TOKEN_HERE>";
const char* CHAT_ID = "<CHAT_ID_HERE>";

// ----- Sensor Objects -----
MPU6050 mpu;
DHT dht(DHTPIN, DHTTYPE);

// ----- Thresholds -----
const int MOISTURE_THRESHOLD = 2500;
const float TEMP_HIGH = 30.0;
const float HUMIDITY_HIGH = 70.0;
const float TILT_ANGLE_THRESHOLD = 70.0;

unsigned long lastUpload = 0;
const unsigned long uploadInterval = 20000;
unsigned long lastBuzzerTime = 0;
bool buzzerSilenced = false;

// ----- Setup -----
void setup() {
  Serial.begin(115200);
  Wire.begin(21, 22); // SDA, SCL
  mpu.initialize();
  dht.begin();

  pinMode(SOIL_MOISTURE_PIN, INPUT);
```

```
  pinMode(VIBRATION_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  connectWiFi();
  Serial.println("System initialized.");
}

// ----- Main Loop -----
void loop() {
  float temp = dht.readTemperature();
  float humidity = dht.readHumidity();
  int soilMoisture = analogRead(SOIL_MOISTURE_PIN);
  bool vibrationDetected = digitalRead(VIBRATION_PIN) == HIGH;

  // MPU6050 Acceleration
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);
  float ax_g = ax / 16384.0;
  float ay_g = ay / 16384.0;
  float az_g = az / 16384.0;
  float tiltX = atan2(ax_g, sqrt(ay_g * ay_g + az_g * az_g)) * 180 / PI; //
Convert to Precentage
  float tiltY = atan2(ay_g, sqrt(ax_g * ax_g + az_g * az_g)) * 180 / PI;

  // Conditions
  bool landslideRisk = abs(tiltX) < TILT_ANGLE_THRESHOLD;
  bool earthquakeRisk = vibrationDetected;
  bool environmentalRisk = (temp >= TEMP_HIGH || humidity >= HUMIDITY_HIGH ||
soilMoisture >= MOISTURE_THRESHOLD);

  // Buzzer silence button
  if (digitalRead(BUTTON_PIN) == LOW) {
    buzzerSilenced = true;
    Serial.println("🔕 Buzzer silenced manually.");
  }

  // LED & Buzzer logic
  if (landslideRisk || earthquakeRisk) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);
    if (!buzzerSilenced) buzzerPattern(1000, 500); // High Pitch 0.5s
    sendTelegramMessage("⚠️ EARLY WARNING: Landslide or Earthquake
Detected!");
```

```
  } else if (environmentalRisk) {
    digitalWrite(RED_LED, LOW);
    digitalWrite(YELLOW_LED, HIGH);
    digitalWrite(GREEN_LED, LOW);
    if (!buzzerSilenced) buzzerPattern(500, 1000); // Low Pitch 1s
  } else {
    digitalWrite(RED_LED, LOW);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, HIGH);
    noTone(BUZZER_PIN);
    buzzerSilenced = false;  // Reset for next time
  }

  // Serial Monitor Output
  Serial.println("===== SENSOR STATUS =====");
  Serial.printf("Temperature: %.1f °C\n", temp);
  Serial.printf("Humidity: %.1f %%\n", humidity);
  Serial.printf("Soil Moisture: %d\n", soilMoisture);
  Serial.printf("TiltX: %.2f ° | TiltY: %.2f °\n", tiltX, tiltY);
  Serial.printf("Vibration: %s\n", earthquakeRisk ? "Detected" : "None");

  if (landslideRisk) Serial.println("⚠ WARNING: Landslide Risk");
  if (earthquakeRisk) Serial.println("⚠ WARNING: Earthquake Vibration");
  if (environmentalRisk) Serial.println("⚠ Environmental Warning");
  if (!landslideRisk && !earthquakeRisk && !environmentalRisk)
    Serial.println("☑ All systems normal.");

  Serial.println("=========================\n");

  // Upload to ThingSpeak
  if (millis() - lastUpload > uploadInterval) {
    float tiltAverage = (abs(tiltX) + abs(tiltY)) / 2.0;
    sendToThingSpeak(temp, humidity, soilMoisture, tiltAverage,
earthquakeRisk);
    lastUpload = millis();
  }

  delay(200);
}

// ----- WiFi -----
void connectWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("\n☑ WiFi Connected");
}

// ----- ThingSpeak Upload -----
void sendToThingSpeak(float temp, float humidity, int moisture, float tilt,
bool earthquake) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "http://api.thingspeak.com/update?api_key=" +
THINGSPEAK_API_KEY +
                 "&field1=" + String(temp) +
                 "&field2=" + String(humidity) +
                 "&field3=" + String(moisture) +
                 "&field4=" + String(tilt) +
                 "&field5=" + String(earthquake ? 1 : 0);

    http.begin(url);
    int code = http.GET();
    if (code > 0) {
      String response = http.getString();
      Serial.print("📤 ThingSpeak: OK → Response: ");
      Serial.println(response);
    } else {
      Serial.print("❌ ThingSpeak Failed → Code: ");
      Serial.println(code);
    }
    http.end();
  }
}

// ----- Telegram Bot Alert -----
void sendTelegramMessage(String message) {
  static unsigned long lastSent = 0;
  const unsigned long cooldown = 10000; // Message Delay
  if (millis() - lastSent < cooldown) return;

  WiFiClientSecure client;
  HTTPClient http;
  client.setInsecure(); // Skip cert validation
  String url = "https://api.telegram.org/bot" + String(TELEGRAM_BOT_TOKEN) +
"/sendMessage";
  String payload = "{\"chat_id\":\"" + String(CHAT_ID) + "\",\"text\":\"" +
message + "\"}";

  http.begin(client, url);
  http.addHeader("Content-Type", "application/json");
  int code = http.POST(payload);
  if (code > 0) Serial.println("📨 Telegram Sent");
```

```
  else Serial.println("❌ Telegram Failed");
  http.end();
  lastSent = millis();
}

// ----- Buzzer Patterns -----
void buzzerPattern(int frequency, int interval) {
  static unsigned long lastTone = 0;
  static bool toneOn = false;

  if (millis() - lastTone >= interval) {
    lastTone = millis();
    toneOn = !toneOn;
    if (toneOn)
      tone(BUZZER_PIN, frequency);
    else
      noTone(BUZZER_PIN);
  }
}
```

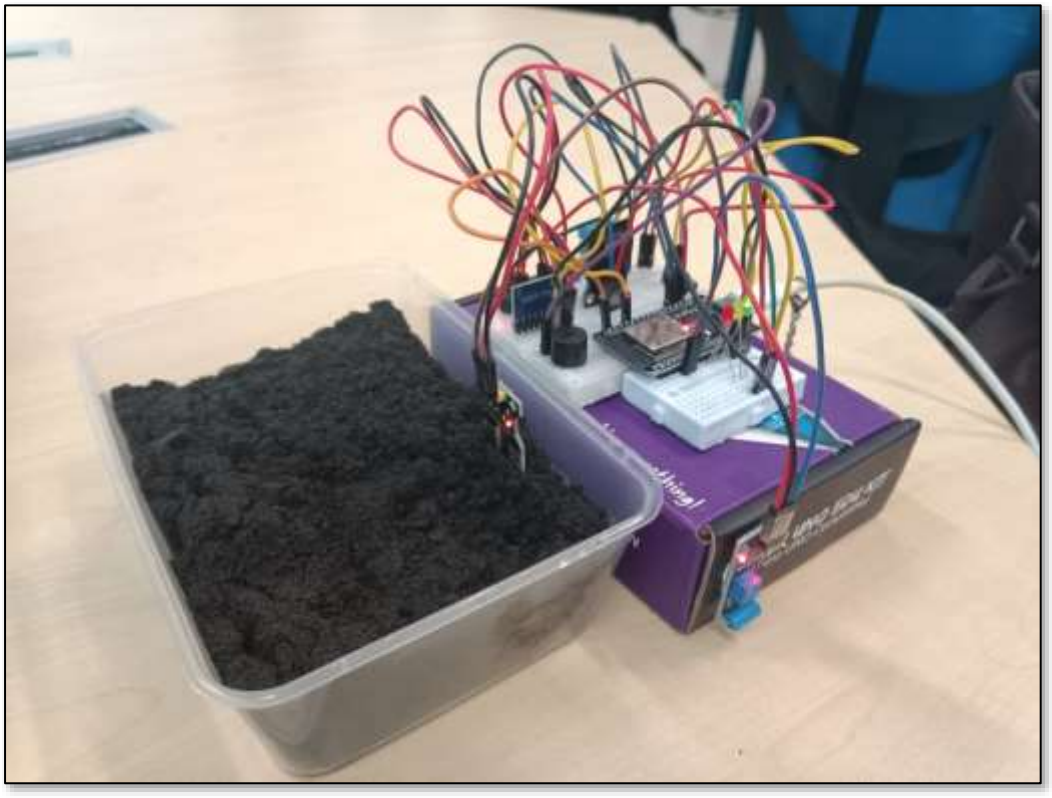


Figure 11. Natural Disaster Monitoring and Early Warning System Prototype

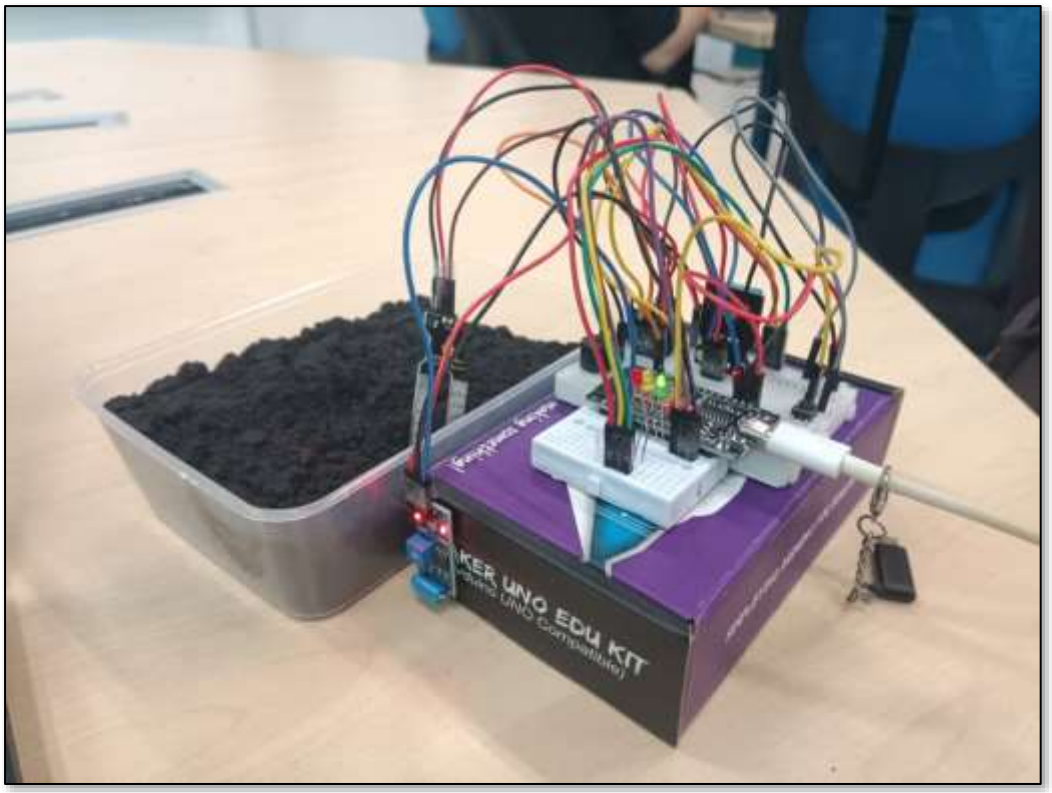


Figure 12. Prototype Photo 2

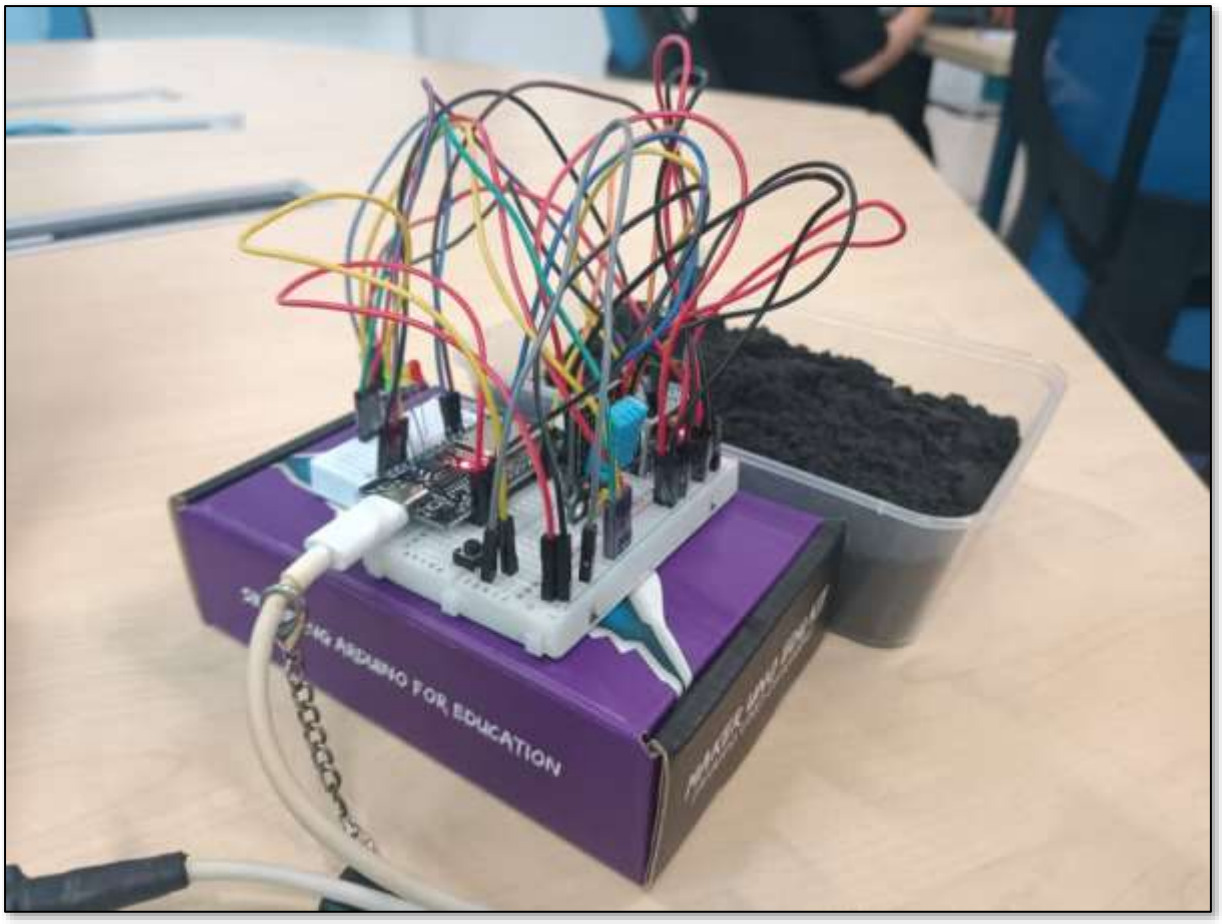


Figure 13. Prototype Photo 3

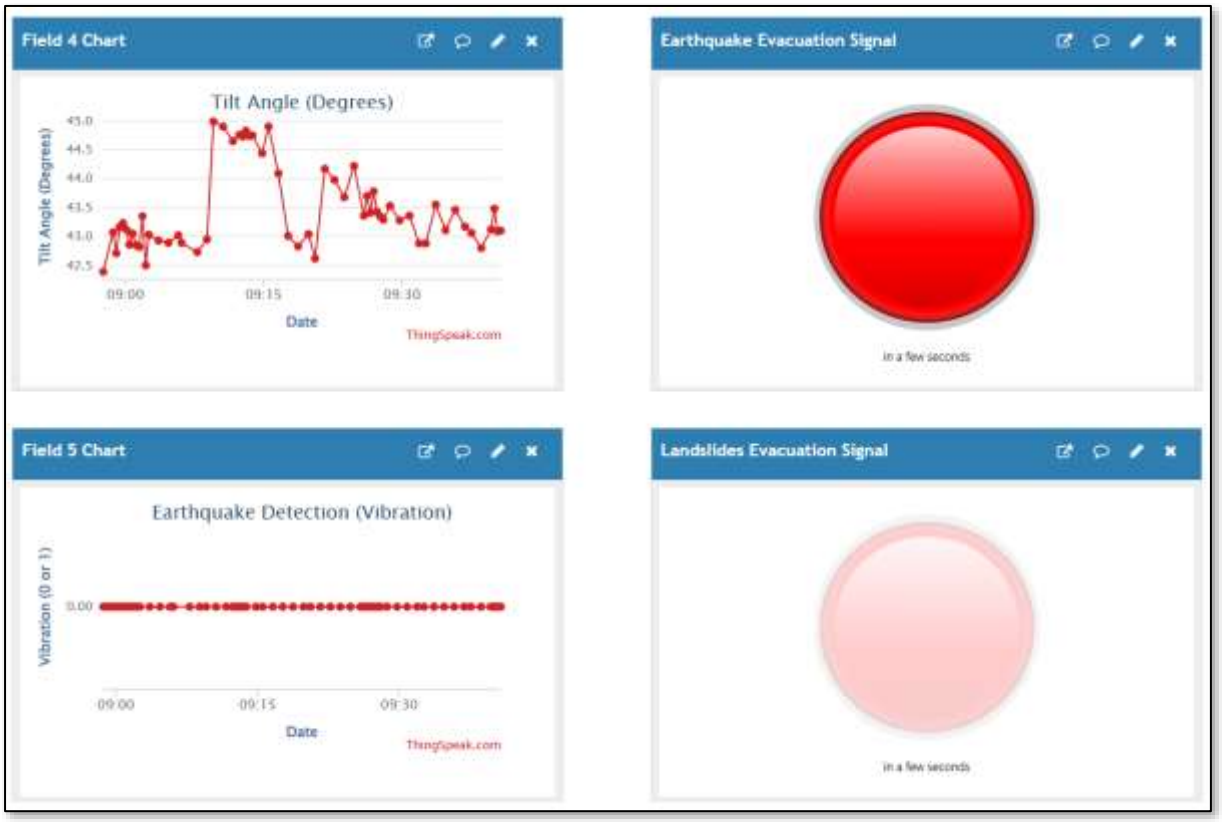


Figure 14. ThingSpeak Data for Tilt Angle and Vibration

Figure 15. ThingSpeak Data for Soil Moisture

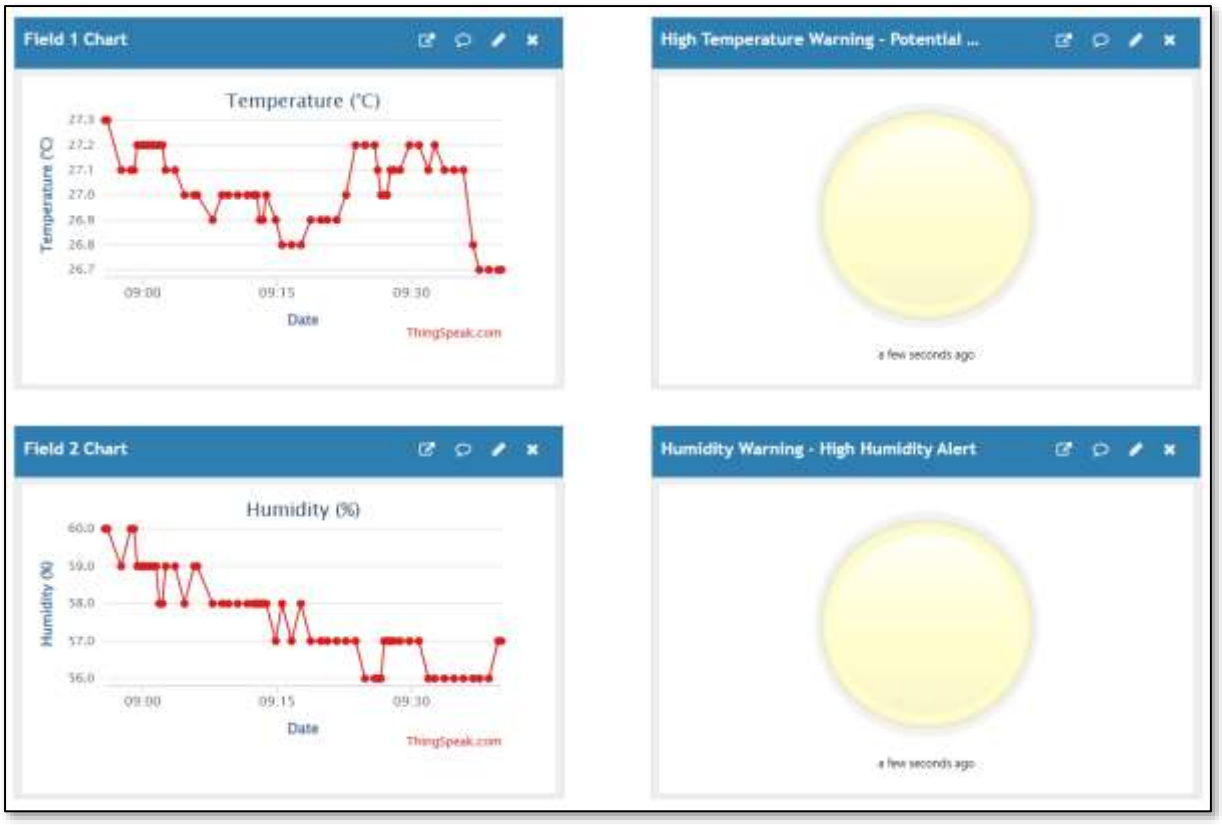


Figure 16. ThingSpeak Data for Temperature and Humidity

```
09:27:37.124 -> ===== SENSOR STATUS =====
09:27:37.124 -> Temperature: 27.1 °C
09:27:37.124 -> Humidity: 57.0 %
09:27:37.124 -> Soil Moisture: 2441
09:27:37.124 -> TiltX: 86.09 ° | TiltY: -0.46 °
09:27:37.124 -> Vibration: None
09:27:37.124 -> ✅ All systems normal.
09:27:37.124 -> ========================
09:27:37.124 ->
```

Figure 17. Sensors Status at Normal Status

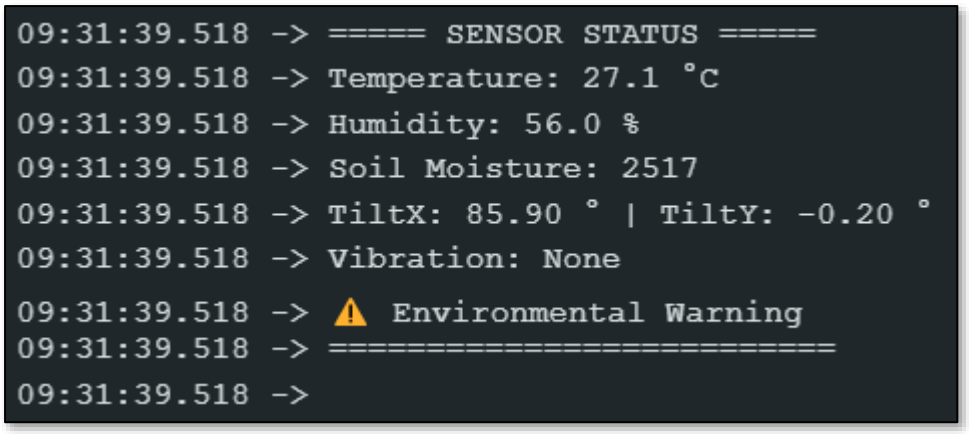


Figure 18. Sensors Status at Environmental Warning (Due to Soil Moisture Increase)

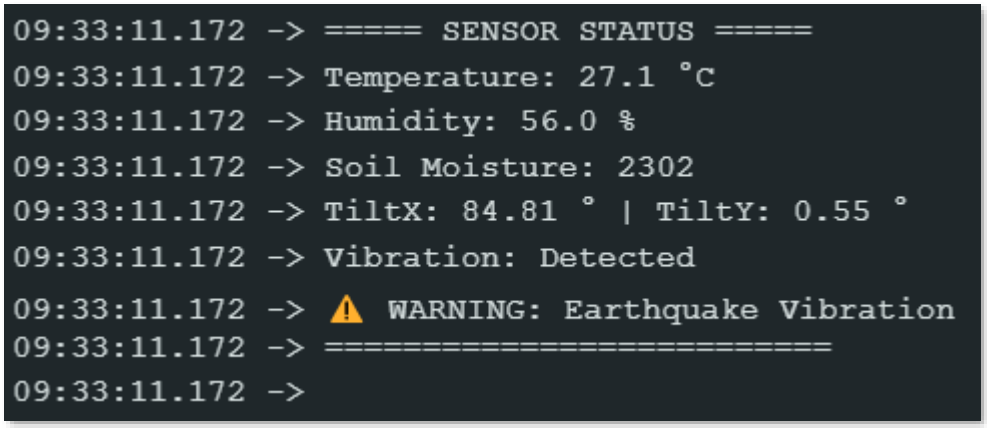


Figure 19. Sensors Status at Vibration Warning (Earthquake)

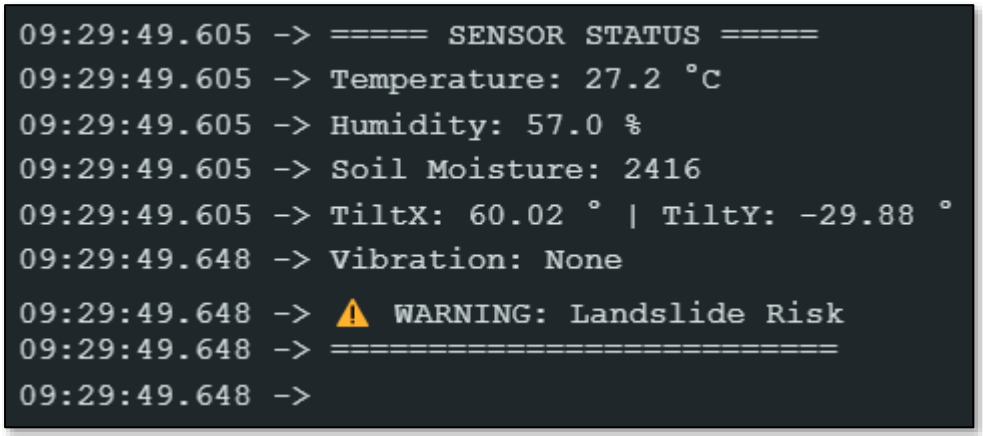


Figure 20. Sensors Status at Landslide Risk

From the output of the system, it creates real-time data from all sensors and warnings if the data collected by sensors passes the threshold assigned. The data will then be sent onto ThingSpeak at every 20 seconds and telegram every 10 seconds to allow data gathering and visualization through graphs and warning signals.

6. CONCLUSION

The Natural Disaster Monitoring and Early Warning System was able to achieve all its objectives successfully, by providing real-time monitoring and early warning systems for landslides and earthquakes through IoT technology. The integration of multiple sensors, like the DHT11, soil moisture, vibration, and gyroscope, creates a monitoring solution that can detect several natural disasters simultaneously.

The system was able to monitor environmental parameters including soil moisture, weather conditions, ground stability, and seismic activity. A cloud-based messaging alert is provided for connected users, as well as a multiple-alarm system that was able to provide both visual and audio alerts for disconnected users.

The limitation faced the making of this system would be the dependency on internet connectivity to transmit data from the sensors to ThingSpeak. As not all communities are able to provide reliable internet connectivity which can affect real-time response during emergencies.

The future enhancements of this project would be to include integration with government officials' response systems, implement machine learning to analyze predictions, add weather radar data integration and development of mobile application for community-alerts. In the real-world, the system could make user of solar panels for endless power and mesh networking for areas with limited internet connectivity.

This project was able to demonstrate the practical use of IoT technology using the ESP32 in disaster management and community safety. This system provides a scalable and cost-effective approach / solution for environmental monitoring and early warning systems.

REFERENCES

Agencies. (2025, January 25). *https://www.thejakartapost.com/indonesia/2025/01/25/rescuers-halt-evacuation-due-to-bad-weather-after-landslide-kills-25.html*. Diambil kembali dari The Jakarta Post: https://www.thejakartapost.com/indonesia/2025/01/25/rescuers-halt-evacuation-due-to-bad-weather-after-landslide-kills-25.html

Arduino. (t.thn.). *Software*. Diambil kembali dari Arduino: https://www.arduino.cc/en/software/

Arrel, K., Rosser, N. J., Kincey, M. E., Robinson, T. R., Horton, P., Densmore, A. L., . . . Pujara, D. S. (2024, August 21). The dynamic threat from landslides following large continental earthquakes. *National Library of Biotechnology Information, 19*(8). doi:10.1371/journal.pone.0308444

Cytron Technologies Malaysia - Simplifying Smart Technology. (t.thn.). Diambil kembali dari Cytron Technologies Malaysia: https://my.cytron.io/

Esposito, M., Palma, L., Belli, A., Sabbatini, L., & Pierleoni, P. (2022, March 9). Recent Advances in Internet of Things Solutions for Early Warning Systems: A Review. *Sensors, 22*(6), 2124. doi:10.3390/s22062124

Gaitan, N. C., & Hojbota, P. (2020). Forest Fire Detection System using LoRa Technology. *International Journal of Advanced Computer Science and Applications(IJACSA), 11*(5). doi:10.14569/IJACSA.2020.0110503

Geoportal Data Bencana Indonesia. (2025). Diambil kembali dari https://gis.bnpb.go.id/

MATLAB & Simulink. (t.thn.). *ThingSpeak*. Diambil kembali dari https://www.mathworks.com/products/thingspeak.html

ThingSpeak. (t.thn.). *IoT Analytics - ThingSpeak Internet of Things*. Diambil kembali dari ThingSpeak: https://thingspeak.mathworks.com/