Environmental Monitoring

Problem Statement:

Map plotted of Earth quake

Problem Definition:

The advancement of hardware and software technologies makes it possible to use smart phones or Internet of things for monitoring environments in real time. In recent years, much effort has been made to develop a smart phone based earthquake early warning system, where low-cost acceleration sensors inside a smart phones are used for capturing earthquake signals. However, because a smart phone comes with a powerful CPU, spacious memory, and several sensors, it is waste of such resources to use it only for detecting earthquakes. Furthermore, because a smart phone is mostly in use during the daytime, the acquired data cannot be used for detecting earthquakes due to human activities. Therefore, in this article, we introduce a stand-alone device equipped with a low-cost acceleration sensor and least computing resources to detect earthquakes. To that end, we first select an appropriate acceleration sensor by assessing the performance and accuracy of four different sensors. Then, we design and develop an earthquake alert device. To detect earthquakes, we employ a simple machine learning technique which trains an earthquake detection model with daily motions, noise data recorded in buildings, and earthquakes recorded in the past. Furthermore, we evaluate the four acceleration sensors by recording two realistic earthquakes on a shake-table. In the experiments, the results show that the developed earthquake alert device can successfully detect earthquakes and send a warning message to nearby devices, thereby enabling proactive responses to earthquakes.

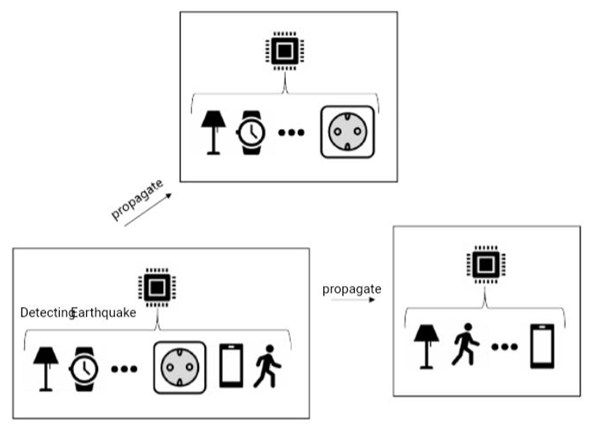
Design Thinking:

Related Works

In this section, we introduce the background technologies used in this work and the relevant projects that are somehow related to or have inﬂuenced our work.In the past few years, seismologists have adopted smart phones or low-cost acceleration sensors to detect earthquakes. In this regard, My shake is one of the most recent contributions which utilizes the smart phone as an earthquake detection sensors The sensor captures the data from the user phone using My Shake application and then process that data by using artiﬁcial neural network for earthquake detection. If the algorithm detects any earthquake like signals then it sends that timestamp data to the server. My Shake application runs on volunteers’ smart phones and they are asked to install the application. It is the ﬁrst world-wide earthquake alert system using smart phone sensors. However, because My Shake heavily relies on the volunteers smart phones, when the smart phones are in active mode, they cannot be used as a seismic station or earthquake detector. In addition, the earthquake detection model trained using earthquakes and human activities can be good enough to distinguish earthquakes from human activities but cannot detect earthquakes from various types of tremors generated from buildings.

Approach:

This section, we discuss our earthquake alert device. To develop the earthquake alert device, we ﬁrst compare several acceleration sensors and then provide benchmark results to select an appropriate sensor used for detecting earthquakes as a stand-alone sensor. Then, the implementation details of the earthquake alert device including hardware and software systems will be provided. The overview of our proposed earthquake detection and response system is shown in Figure 1.Because the earthquake alert device operates as a stand-alone device, it does collaborate with other earthquake alert devices or servers for further processing. However, we maintain a database server to preserve earthquake events.



Innovation:

Creating an environmental monitoring IoT project requires careful planning, integration, and ongoing maintenance to ensure that it effectively serves its intended purpose of monitoring and managing environmental conditions. Adapt these steps to the specific needs and goals of your project.

**1.Project Planning and Requirements Gathering:**

* Define the specific environmental parameters you want to monitor and the goals of your project, such as improving air quality, ensuring safety, or optimizing resource usage.

**2.Hardware Selection:**

* Choose the necessary IoT hardware components, including sensors, microcontrollers, communication modules, and power sources.
* Select sensors based on the parameters you plan to monitor (e.g., temperature sensors, humidity sensors, gas sensors, or particulate matter sensors).

**3.IoT Platform Selection:**

* Select an IoT platform or framework to manage data collection, storage, and device management. Common options include AWS IoT, Google Cloud IoT, Microsoft Azure IoT, or open-source platforms like MQTT.

**4.Sensor Deployment:**

* Install sensors in the locations you want to monitor, considering factors like accessibility and environmental conditions.
* Calibrate and configure sensors as needed to ensure accurate data collection.

**5.Data Collection and Connectivity:**

* Configure sensors to collect data at regular intervals or in response to specific events.
* Use wireless communication protocols like Wi-Fi, Bluetooth, Zigbee, LoRaWAN, or cellular connectivity to transmit data to the IoT platform.

**6.Data Processing and Analysis:**

* Use the IoT platform to process and analyze the data collected from the sensors.
* Implement algorithms to detect anomalies, trends, or events related to environmental parameters.

**7.User Interface:**

* Develop a user interface to visualize environmental data, such as a web-based dashboard or a mobile app.
* Provide real-time information on environmental conditions, historical data, and alerts.

**8.Alerting and Notifications:**

* Set up alerting mechanisms to notify relevant personnel or stakeholders when environmental parameters exceed predefined thresholds or when anomalies occur.

**9.Data Storage and Archiving:**

* Store historical data in a secure and scalable database to enable long-term analysis and reporting.
* Implement data retention policies based on your project's requirements

**10.Security and Privacy:**

* Ensure that the data collected from environmental sensors is secure and complies with privacy regulations.
* Implement encryption, authentication, and access control measures to protect sensitive data.

**11.Energy Efficiency:**

* Optimize power management to extend the lifespan of battery-powered devices.
* Implement sleep modes and low-power configurations for sensors and devices when possible.

**12.Testing and Quality Assurance:**

* Thoroughly test the entire system to ensure data accuracy, device reliability, and proper functioning of the IoT platform.
* Conduct field tests to validate the system's performance in real-world conditions.

**13.Deployment:**

* Deploy environmental monitoring devices in the target locations, ensuring they are properly installed, configured, and connected.

**14.Monitoring and Maintenance:**

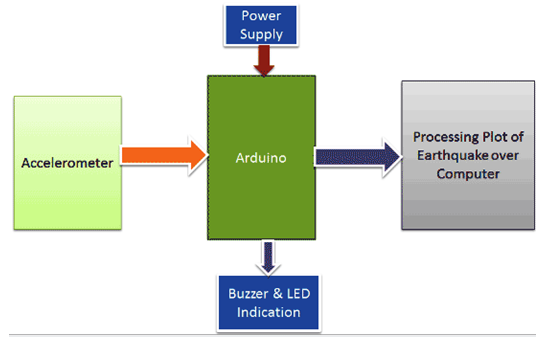
* Continuously monitor the system's performance and troubleshoot any issues that arise.
* Perform regular maintenance tasks, such as sensor calibration and firmware updates.

**15.Data Analytics and Reporting:**

* Analyze the collected data to gain insights into environmental trends and patterns.
* Generate reports and visualizations to inform decision-making and environmental management.

Earthquake detector:

An earthquake is an unpredictable natural disaster that causes damage to lives and property. It happens suddenly and we cannot stop it but we can be alerted from it. In today’s time, there are many technologies which can be used to detect the small shakes and knocks, so that we can take precautions prior to some major vibrations in earth. Here we are using [Accelerometer ADXL335](https://circuitdigest.com/tags/accelerometer) to detect the pre-earthquake vibrations. Accelerometer ADXL335 is highly sensitive to shakes and vibrations along with all the three axes. Here we are building an Ardunio based Earthquake detector using Accelerometer.

Block Diagram of Environmental Monitoring(Earthquake Detection) 

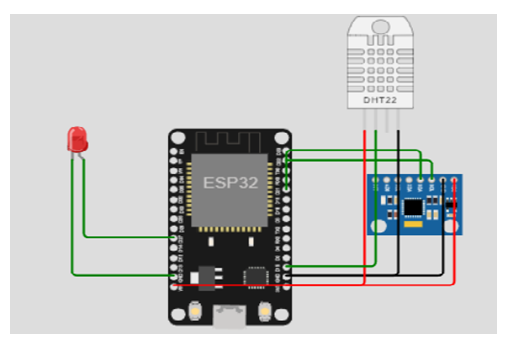
**Components Required**

* Adruino UNO
* Accelerometer ADXL335
* 16x2 LCD
* Buzzer
* BC547 transistor
* 1k Resistors
* 10K POT
* LED
* Power Supply 9v/12v
* Berg sticks male/female

**Tools Required**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Tools** | **Examples** |
| 1. | Arduino IDE | Arduino IDE |
| 2. | CAD Software (Optional) | Autodesk Eagle. |
| 3. | Version Control System | GitHub, GitHub Desktop. |
| 4. | Project Management Software | ThingSpeak |
| 5. | Simulation Software (Optional) | Wokwi |
| 6. | Data Analysis Tools | Excel. |
| 7. | Communication Tools | Microsoft Teams, Discord, Zoom |

**Circuit Diagram of Environmental Monitoring**

****

Working Principle:

Working of this **Arduino Earthquake Detector** is quite simple. As we mentioned earlier that we have used Accelerometer for detecting earthquake vibrations along any of the three axes so that whenever vibrations occur accelerometer senses that vibrations and convert them into equivalent ADC value. Then these ADC values are read by Arduino and shown over the 16x2 LCD. We have also shown these values on **Graph using Processing.** First we need to **calibrate the Accelerometer** by taking the samples of surrounding vibrations whenever Arduino Powers up. Then we need to subtract those sample values from the actual readings to get the real readings. This calibration is needed so that it will not show alerts with respect to its normal surrounding vibrations. After finding real readings, Arduino compares these values with predefined max and min values. If Arduino finds any changes values are more then or less then the predefined values of any axis in both direction (negative and positive) then Arduino trigger the buzzer and shows the status of alert over the 16x2 LCD and a LED also turned on as well. We can adjust the sensitivity of Earthquake detector by changing the Predefined values in Arduino code

Esp32 code:

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <Wire.h>

#include <DHTesp.h>

Adafruit\_MPU6050 m\_p\_u;

const int DHT\_PIN = 15;

DHTesp dhtSensor;

int a = 27;

void setup() {

  // Initialize Serial communication

**Serial**.begin(115200);

  while (!**Serial**); // Wait for the serial port to open

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

  pinMode(a, OUTPUT);

  // Initialize the MPU6050 sensor

  if (!m\_p\_u.begin()) {

**Serial**.println("MPU6050 not found. Please check wiring.");

    while (1) {

      delay(20);

    }

  }

}

void loop() {

  TempAndHumidity data = dhtSensor.getTempAndHumidity();

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

**Serial**.print(data.temperature, 2);

**Serial**.println(" °C");

  if (data.temperature <= 40) {

    digitalWrite(a, HIGH);

  } else {

    digitalWrite(a, LOW);

  }

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

**Serial**.print(data.humidity, 1);

**Serial**.println(" %");

**Serial**.println("---");

  sensors\_event\_t acc, gcc, temp;

  m\_p\_u.getEvent(&acc, &gcc, &temp);

**Serial**.print("Acceleration on X axis: ");

**Serial**.println(acc.acceleration.x);

  delay(1000);

**Serial**.print("Acceleration on Y axis: ");

**Serial**.println(acc.acceleration.y);

  delay(1000);

**Serial**.print("Acceleration on Z axis: ");

**Serial**.println(acc.acceleration.z);

  delay(1000);

**Serial**.print("Rotation on X axis: ");

**Serial**.println(gcc.gyro.x \* 180.0 / M\_PI);

  delay(1000);

}

**Wokwi -Project:**

**Downloaded from https://wokwi.com/projects/379651705792254977**

**Simulate this project on** [**https://wokwi.com**](https://wokwi.com)

**Sketch.INO:**

#define BLYNK\_TEMPLATE\_ID "TMPLRE0PlNsg"

#define BLYNK\_TEMPLATE\_NAME "Earthquake and Water Level Detector"

#define BLYNK\_AUTH\_TOKEN "uTZSm-N8bY5fn7\_I6ts0NvIVEN24mjgP"

#define BLYNK\_PRINT Serial

//set water level in cm

int emptyTankDistance = 70;

int fullTankDistance = 30;

const int trigPin = 18;

const int echoPin = 19;

//define sound speed in cm/uS

#define SOUND\_SPEED 0.034

#define CM\_TO\_INCH 0.393701

long duration;

float distanceCm;

float distanceInch;

#define pinBuzzer 2

#define pinLED1 5

#define pinLED2 4

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#include <Adafruit\_MPU6050.h>

#include <Adafruit\_Sensor.h>

#include <Wire.h>

Adafruit\_MPU6050 mpu;

char auth[] = BLYNK\_AUTH\_TOKEN;

// Your WiFi credentials.

// Set password to "" for open networks.

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

BlynkTimer timer;

void myTimerEvent()

{

Blynk.virtualWrite(V2, millis() / 1000);

}

void setup()

{

Serial.begin(115200);

Blynk.begin(auth, ssid, pass);

pinMode(pinBuzzer, OUTPUT);

pinMode(pinLED1, OUTPUT);

pinMode(pinLED2, OUTPUT);

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

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

timer.setInterval(1000L, myTimerEvent); //Staring a timer

// Try to initialize! MPU6050

if (!mpu.begin()) {

Serial.println("Failed to find MPU6050 chip");

while (1) {

delay(10);

}

}

Serial.println("MPU6050 Found!");

mpu.setAccelerometerRange(MPU6050\_RANGE\_16\_G);

mpu.setGyroRange(MPU6050\_RANGE\_250\_DEG);

mpu.setFilterBandwidth(MPU6050\_BAND\_21\_HZ);

Serial.println("");

delay(100);

}

void loop()

{

deteksigempa();

timer.run();

}

void deteksigempa(){

sensors\_event\_t a, g, temp;

mpu.getEvent(&a, &g, &temp);

int acX = a.acceleration.x;

int acY = a.acceleration.y;

int acZ = a.acceleration.z;

String v = ",";

String x = String(acX);

String y = String(acY);

String z = String(acZ);

// 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);

// Calculate the distance

distanceCm = duration \* SOUND\_SPEED/2;

// Convert to inches

distanceInch = distanceCm \* CM\_TO\_INCH;

delay(1000);

if(distanceInch > 70){

tone(pinBuzzer, 1000);

digitalWrite(pinLED2, HIGH);

Serial.println("SITUASI LEVEL AIR");

Serial.println("Level Air Tinggi !!!");

Serial.print("Level Air (inch): ");

Serial.println(distanceInch);

Serial.println("");

Blynk.virtualWrite(V6, distanceInch);

Blynk.virtualWrite(V7, LOW);

Blynk.virtualWrite(V8, HIGH);

}

else{

noTone(pinBuzzer);

digitalWrite(pinLED2, LOW);

Serial.println("SITUASI LEVEL AIR");

Serial.println("Level Air Aman");

Serial.print("Level Air (inch): ");

Serial.println(distanceInch);

Serial.println("");

Blynk.virtualWrite(V6, distanceInch);

Blynk.virtualWrite(V7, HIGH);

Blynk.virtualWrite(V8, LOW);

}

if(acX > 4 | acY > 4 | acZ > 13 ){

tone(pinBuzzer, 1000);

digitalWrite(pinLED1, HIGH);

Serial.println("SITUASI GEMPA");

Serial.println("AWAS GEMPA BUMI !!! "+ x + v + y + v + z);

Serial.println("");

Blynk.virtualWrite(V0, "AWAS !! GEMPA BUMI");

Blynk.virtualWrite(V1, x);

Blynk.virtualWrite(V4, y);

Blynk.virtualWrite(V5, z);

Blynk.virtualWrite(V2, LOW);

Blynk.virtualWrite(V3, HIGH);

}else{

noTone(pinBuzzer);

digitalWrite(pinLED1, LOW);

Serial.println("SITUASI GEMPA");

Serial.println("AMAN "+ x + v + y + v + z);

Serial.println("");

Blynk.virtualWrite(V0, "AMAN");

Blynk.virtualWrite(V1, x);

Blynk.virtualWrite(V4, y);

Blynk.virtualWrite(V5, z);

Blynk.virtualWrite(V2, HIGH);

Blynk.virtualWrite(V3, LOW);

}

delay(1500);

Blynk.run();

}

**Diagram.json:**

{

"version": 1,

"author": "Rajalakshmi.G",

"editor": "wokwi",

"parts": [

{

"type": "wokwi-esp32-devkit-v1",

"id": "esp",

"top": 31.4,

"left": 10.12,

"rotate": 90,

"attrs": {}

},

{

"type": "wokwi-mpu6050",

"id": "imu1",

"top": -60.61,

"left": 50.89,

"rotate": 270,

"attrs": {}

},

{

"type": "wokwi-buzzer",

"id": "bz1",

"top": 31.2,

"left": 203.4,

"attrs": { "volume": "0.1" }

},

{

"type": "wokwi-led",

"id": "led1",

"top": 107.42,

"left": 302.08,

"attrs": { "color": "red" }

},

{

"type": "wokwi-hc-sr04",

"id": "ultrasonic1",

"top": 333.57,

"left": 155.48,

"attrs": { "distance": "310" }

},

{

"type": "wokwi-led",

"id": "led2",

"top": 107.93,

"left": 364.52,

"attrs": { "color": "blue" }

}

],

"connections": [

[ "esp:TX0", "$serialMonitor:RX", "", [] ],

[ "esp:RX0", "$serialMonitor:TX", "", [] ],

[ "esp:GND.1", "led1:C", "black", [ "v91.44", "h284.44" ] ],

[ "bz1:1", "esp:GND.1", "black", [ "v99.44", "h-211.59" ] ],

[ "esp:D2", "bz1:2", "violet", [ "v43.83", "h148.14" ] ],

[ "esp:D5", "led1:A", "green", [ "v64.25", "h249.52" ] ],

[ "esp:VIN", "imu1:VCC", "red", [ "v0" ] ],

[ "esp:GND.2", "imu1:GND", "black", [ "v0" ] ],

[ "imu1:SCL", "esp:D22", "gold", [ "h-20.96", "v105.43", "h136.82", "v131.91", "h-44.17" ] ],

[ "esp:D21", "imu1:SDA", "blue", [ "v20.41", "h84.73", "v-152.77", "h-136.82", "v-85.28" ] ],

[ "ultrasonic1:VCC", "esp:VIN", "red", [ "h-281.13", "v-340.17" ] ],

[ "ultrasonic1:GND", "esp:GND.2", "black", [ "h-300.89", "v-340.71", "h63.73" ] ],

[ "ultrasonic1:TRIG", "esp:D18", "purple", [ "h-150", "v-243.87" ] ],

[ "ultrasonic1:ECHO", "esp:D19", "cyan", [ "h-150.45", "v-243.87" ] ],

[ "led2:C", "esp:GND.1", "black", [ "v133.13", "h-359.9" ] ],

[ "led2:A", "esp:D4", "green", [ "v112.63", "h-340.9" ] ]

],

"dependencies": {}

}

**Libraries:**

**# Wokwi Library List**

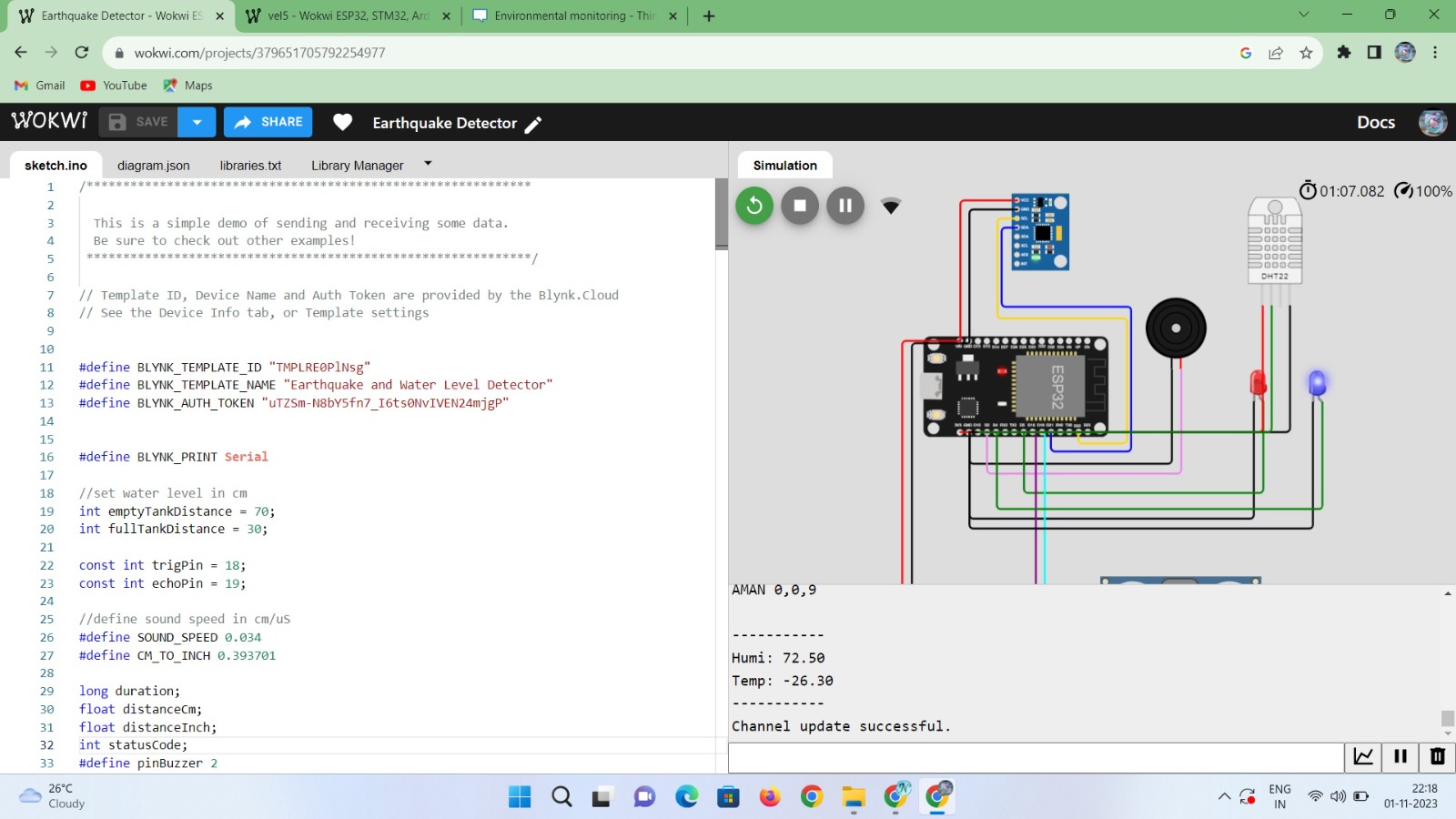
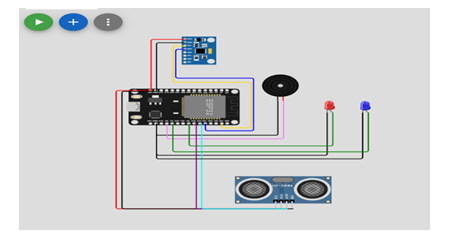
**# See https://docs.wokwi.com/guides/libraries**

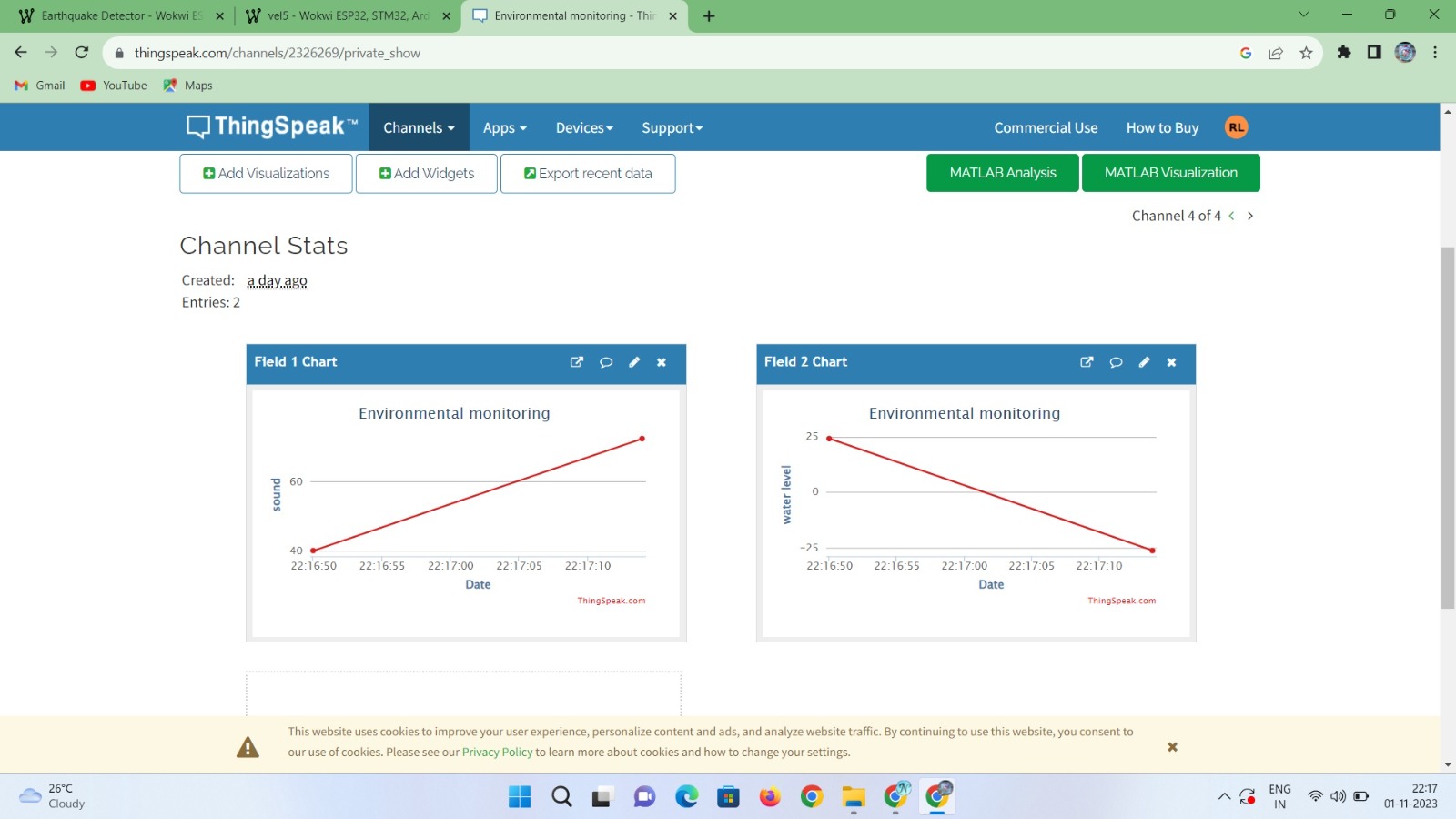
**Blynk**

**Accelerometer ADXL335**

**Adafruit MPU6050**

**Simulation:**

****

****