Environmental Monitoring

Wokwi Project:

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

Simulate this project on <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;

int statusCode;

#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>

#include <ThingSpeak.h>

#include "DHTesp.h"

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

WiFiClient client;

unsigned long myChannelNumber =  2326269;

const char \* myWriteAPIKey = "93UDBCCJF9F3AOEL";

DHTesp dhtSensor;

TempAndHumidity data;

const int DHT\_PIN = 15;

Adafruit\_MPU6050 mpu;

char auth[] = BLYNK\_AUTH\_TOKEN;

// Your WiFi credentials.

// Set password to "" for open networks.

BlynkTimer timer;

void myTimerEvent()

{

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

}

void setup()

{

**Serial**.begin(115200);

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client);

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

**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();

  connectToCloud();

computeData();

writeData();

delay(1000);

}

void connectToCloud(){

if(WiFi.status() != WL\_CONNECTED) {

**Serial**.print("Attempting to connect");

while(WiFi.status() != WL\_CONNECTED) {

WiFi.begin(ssid, pass);

for(int i=0;i<5;i++) {

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

delay(1000);

}

}

**Serial**.println("\nConnected.");

}

}

void computeData(){

data = dhtSensor.getTempAndHumidity();

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

**Serial**.println("Humi: " + String(data.humidity));

**Serial**.println("Temp: " + String(data.temperature));

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

}

void writeData(){

ThingSpeak.setField(1, data.humidity);

ThingSpeak.setField(2, data.temperature);

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(15000); // data to be uploaded every 15secs

}

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": "m.selvanayaki",

  "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" }

    },

    { "type": "wokwi-dht22", "id": "dht1", "top": -66.9, "left": 311.4, "attrs": {} }

  ],

  "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" ] ],

    [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],

    [ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],

    [ "dht1:SDA", "esp:D15", "green", [ "v0" ] ]

  ],

  "dependencies": {}

}

Libraries:

# Wokwi Library List

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

Blynk

Accelerometer ADXL335

Adafruit MPU6050

DHT sensor library for ESPx

WiFi

ThingSpeak

TimerOne

DHT sensor library for ESPx

Simulation:



