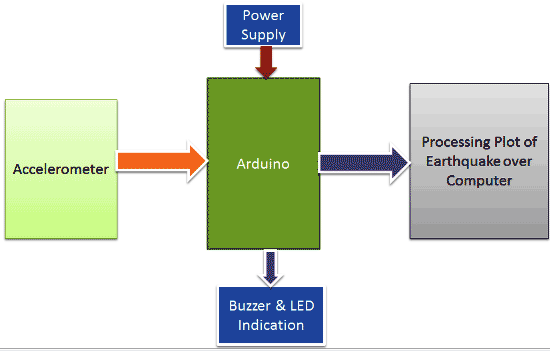
ENVIRONMENTAL MONITORING

(Earthquake Detector Alarm)

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**

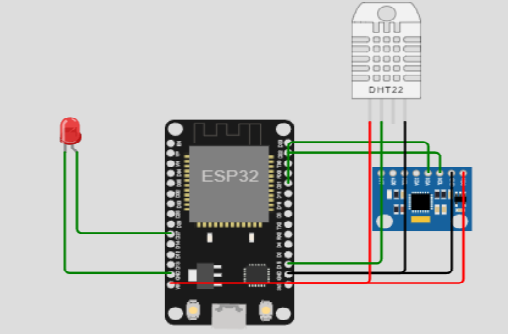
* Adruino UNO
* Accelerometer ADXL335
* 16x2 LCD
* Buzzer
* BC547 transistor
* 1k Resistors
* 10K POT
* LED
* Power Supply 9v/12v
* Berg sticks male/female

**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**

**(Earthquake Detection)**



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

Arduino 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);

}