NAME:HARISH M

REGISTER NUMBER:21BEC1142

COLLEGE NAME:VELLORE INSTITUTE OF TECHNOLOGY

BATCH NUMBER:06

MICROCONTROLLER BASED ENGINEERING APPLICATIONS-PROJECT STATEMENT

TITLE:ALERT SYSTEM FOR EARTHQUAKES

DESCRIPTION:DEVEOPING A PROTOTYPE DESIGN WHICH CAN DETECT AND ALERT THE EARTH QUAKE INCIDENCE

CONTENTS:

CODE FOR SOLUTION

SHORT OVERVIEW OF PROJECT:

Internet of Things Based Earthquake Early Warning SystemMany countries have used EEW (Early Earthquake Warning) systems to save people's lives. Istanbul Earthquake Warning and Rapid Response System is an example of an earthquake warning system. Since it is impossible to know when and where a seismic event will occur, it requires early detection, prompt design and good communication skills to reduce damage to buildings and people's lives. Earthquake warnings can be implemented via the Internet of Things in Wireless Sensor Networks (WSN). Since P waves travel faster than S waves and reach the sensor, they help provide early warning signals. Sensor values are connected to data on the NodeMcu and stored in the ThingSpeak account. ThingSpeak accounts allow data to be shared, collected and visualized in the cloud in real time. With the help of the acceleration sensor, the actual value of the earthquake is recorded and a warning signal is sent to the registered location with the help of the Wi-Fi module of the web server.

PROBLEM STATEMENT:

Design and implement an effective alert system for earthquakes that can provide timely and accurate notifications to individuals, communities, and relevant authorities, with the goal of minimizing the loss of life and property damage during seismic events.

Key Considerations and Objectives:

1. Timeliness: Develop a system that can detect seismic activity as quickly as possible and issue alerts in real-time or near-real-time.2. Accuracy: Ensure the system can distinguish between minor tremors and potentially devastating earthquakes, minimizing false alarms while maximizing true alerts.3. Accessibility: Create a system that can disseminate alerts through various communication channels, taking into account diverse demographics, languages, and abilities.4. Coverage: Ensure that the alert system covers a wide geographical

area, addressing the needs of urban and rural populations equally.5. Adaptability: Develop a system that can adapt to different seismic regions, considering variations in earthquake patterns and magnitudes.6. Redundancy: Implement multiple layers of redundancy to guarantee the system's reliability, even in the event of infrastructure damage during an earthquake.7. Public Awareness: Promote public awareness and education on how to respond to earthquake alerts and notifications effectively.8. Integration: Integrate with existing emergency response systems, such as first responder communication networks and government agencies responsible for disaster management.9. Cost-effectiveness: Design the system to be cost-effective, taking into account the budget limitations of the areas it serves.10. Legal and Ethical Considerations: Address privacy concerns and legal requirements for collecting and sharing data related to earthquake alerts.

The successful implementation of an alert system for earthquakes will significantly contribute to saving lives and minimizing damage during seismic events, making it a crucial project in the field of disaster preparedness and response.

SCOPE OF SOLUTION:

The scope of a solution for an alert system for earthquakes is broad and encompasses various components and considerations. To develop a comprehensive and effective earthquake alert system, the following aspects need to be considered within the scope of the solution:

1. Seismic Detection:

- Implementing a network of seismometers and sensors to monitor ground movements and detect seismic activity in real-time.

- Utilizing advanced algorithms and data analytics to assess the magnitude, location, and depth of earthquakes.

2. Alert Generation:

- Developing algorithms and protocols to evaluate the severity of an earthquake and generate alerts based on predefined thresholds.

- Incorporating early warning systems to provide alerts before the most damaging seismic waves arrive.

3. Alert Dissemination:

- Creating a multi-channel notification system to disseminate alerts to various stakeholders, including the general public, emergency services, and government agencies.

- Utilizing communication technologies such as SMS, mobile apps, sirens, social media, and broadcast systems to reach a wide range of audiences.

4. Geographic Coverage:

- Ensuring the system covers a wide geographical area, including urban and rural regions, as earthquakes can occur anywhere - Establishing a regional, national, or international network to coordinate alerts and responses for larger seismic events.

5. Data Integration:

- Integrating with existing seismic monitoring networks and government agencies responsible for disaster management to enhance data accuracy and response coordination.

- Collecting and analyzing data from various sources to improve alert accuracy and responsiveness.

6. User Education and Training:

- Developing educational materials and training programs to inform the public and relevant authorities about earthquake preparedness and how to respond to alerts effectively.

- Promoting earthquake drills and exercises to enhance community readiness.

7. Redundancy and Reliability:

- Implementing redundancy in the system to ensure reliability, including backup power sources, redundant communication channels, and failover mechanisms.

- Conducting regular maintenance and testing to guarantee the system's functionality.

8. Privacy and Legal Compliance:

- Addressing privacy concerns related to the collection and dissemination of personal data.

- Ensuring compliance with relevant legal and regulatory frameworks governing emergency alert systems.

9. Cost Management: - Developing a cost-effective solution that considers budget limitations while ensuring effectiveness.

- Evaluating the financial feasibility of the system and exploring potential funding sources, such as government grants or public-private partnerships.

10. System Scalability:

- Designing the system to accommodate future technological advancements and population growth.

- Ensuring that the alert system can be scaled up or down to meet changing demands and population dynamics.

11. Continuous Improvement:

- Establishing mechanisms for ongoing monitoring and evaluation to improve the system's accuracy and response times based on real-world data and feedback.

12. International Collaboration:

- Collaborating with neighboring countries in earthquake-prone regions to facilitate cross-border alerts and coordinated response

efforts.The scope of the solution for an earthquake alert system is comprehensive and requires a multidisciplinary approach, involving experts in seismology, data science, communication technology, emergency management, and public outreach. Implementing such a system is essential for saving lives and reducing the impact of earthquakes on communities and infrastructure.

REQUIRED COMPONENTS TO DEVELOP SOLUTIONS:(IDE NAME,SOFTWARE ,HARDWARE)

COMPONENTS REQUIRED:

Name	Quantity	Component
U1	1	Arduino Uno R3
R1	1	2 Ω Resistor
D1	1	Green LED
U5	1	LCD 16 x 2
R2	1	1 kΩ Resistor
R4	1	2 kΩ Resistor
PIEZO1	1	Piezo
TILT1 TILT2 TILT3	3	Tilt Sensor
R3 R5 R6	3	220 Ω Resistor

IDE NAMES:

ARDUINO DE,FRITZIG,THINGSPEAK

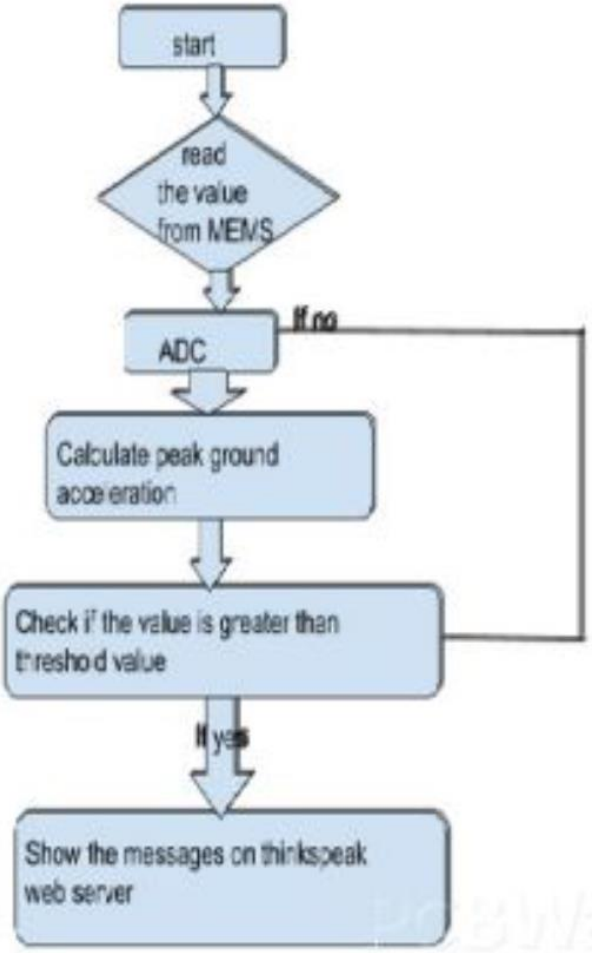
Hardware

- Arduino Uno
- Lcd Display 16x4
- MPU6050 Sensor
- ESP8266
- Buzzer
- Led

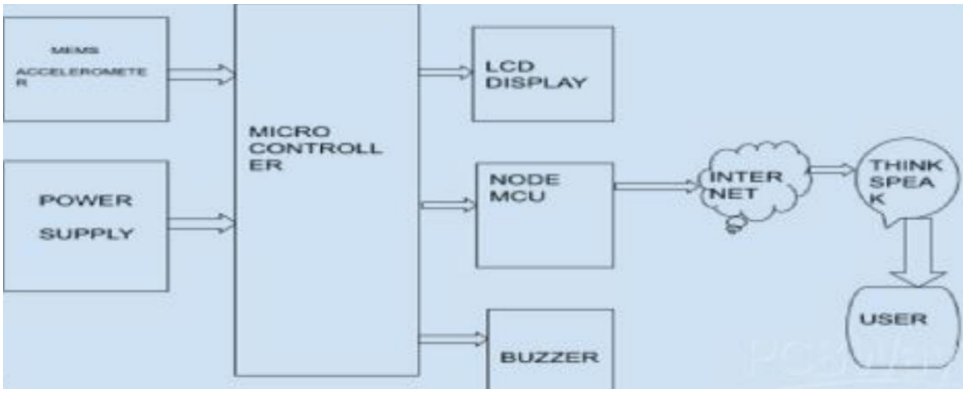
Software

- Arduino Ide
- tinkercad
- Fritzing
- ThinkSpeak

Block Diagram

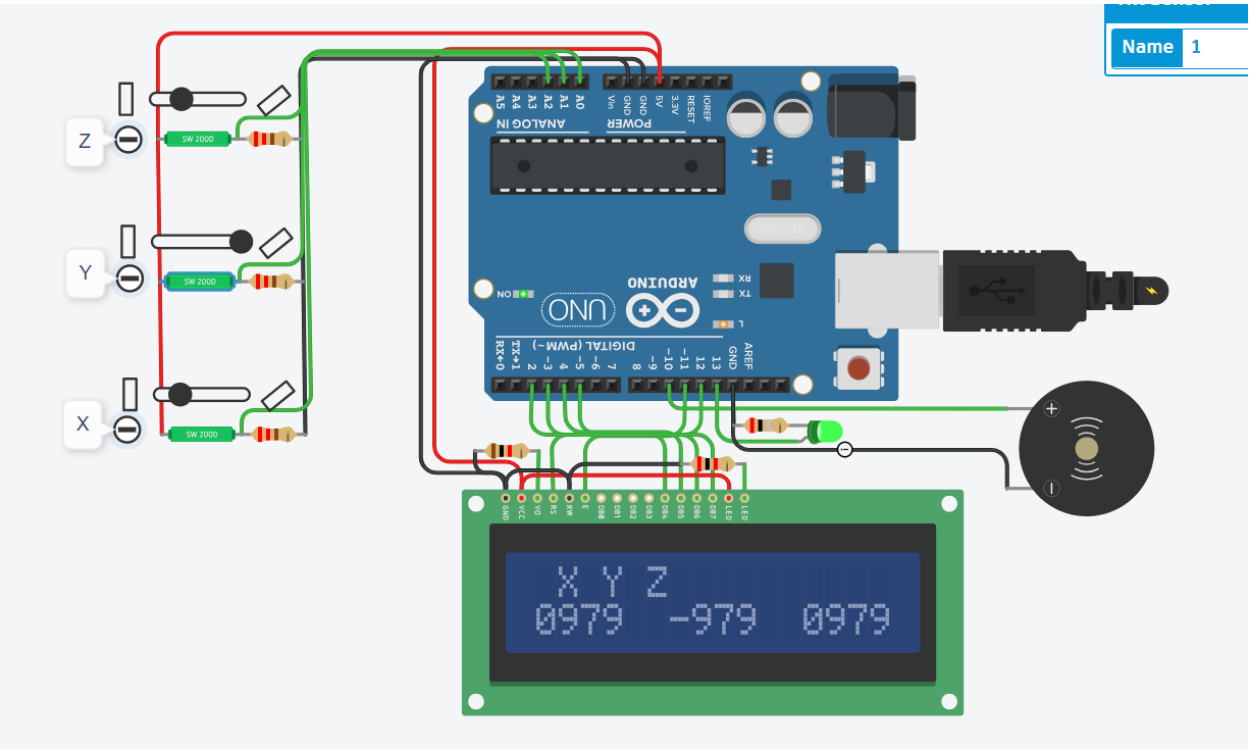


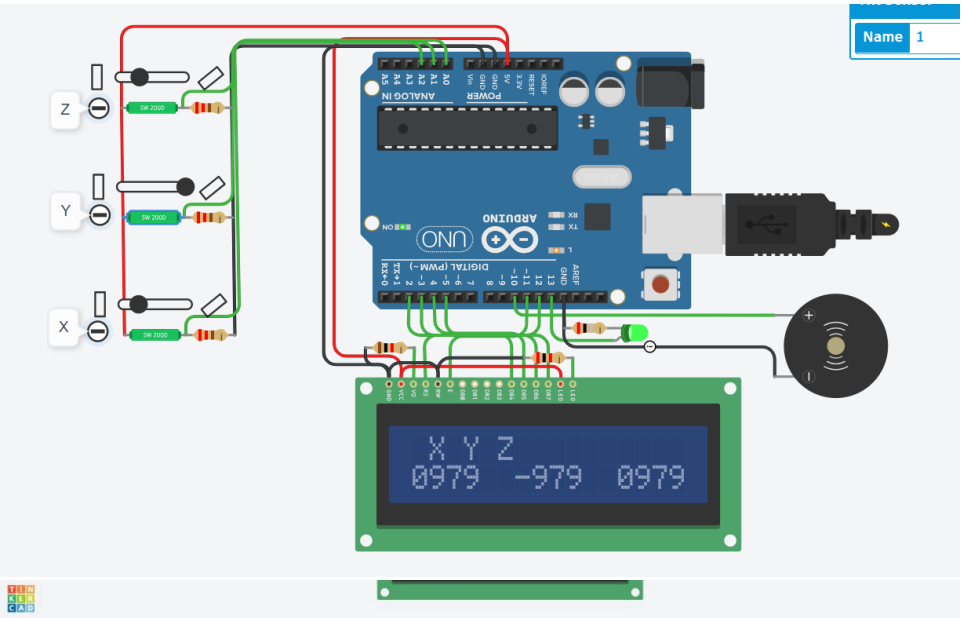
Flowchart



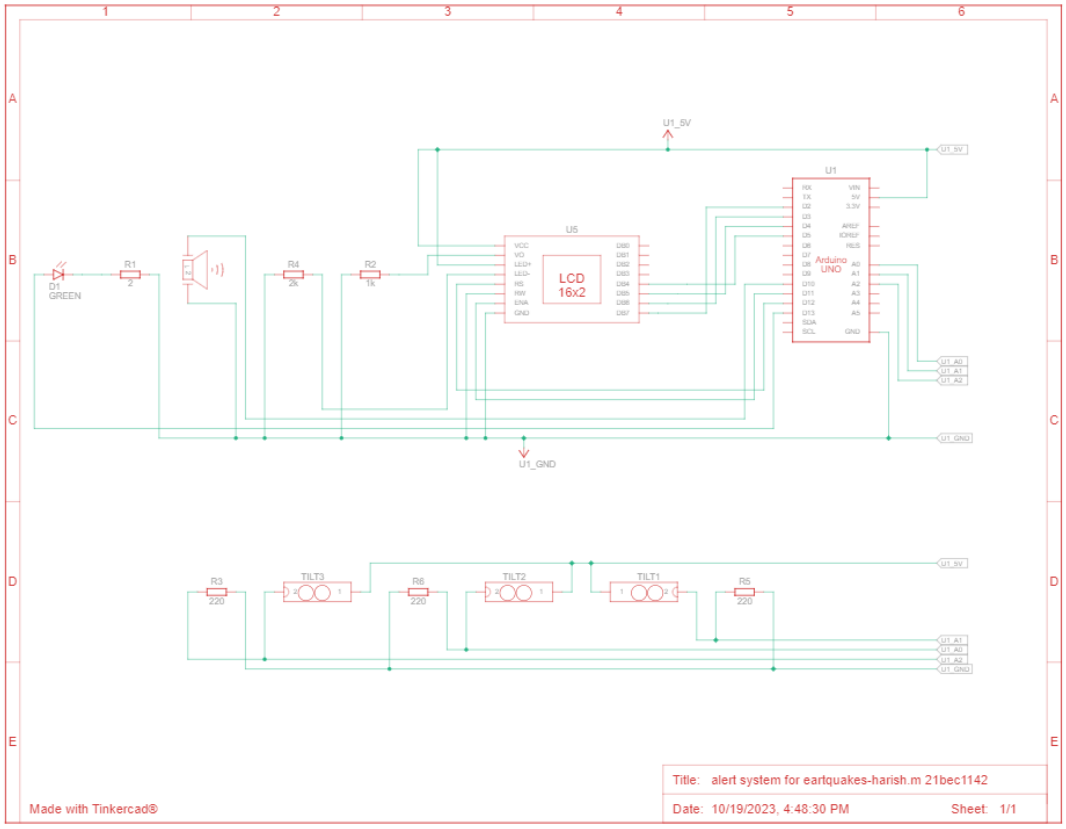
SIMULATED CIRCUIT:

Step 1 : Follow The Circuit And Construct





SCHEMATIC VIEW OF PROJECT



CODE SIMULATED IN TINKERCAD:

```
#include<LiquidCrystal.h> // lcd Header
LiquidCrystal lcd(12,11,5,4,3,2); // pins for LCD Connection

#define buzzer 10 // buzzer pin
#define led 13 //led pin

#define x A0 // x_out pin of Accelerometer
#define y A1 // y_out pin of Accelerometer
#define z A2 // z_out pin of Accelerometer

/*variables*/
int xsample=0;
int ysample=0;
int zsample=0;
long start;
int buz=0;

/*Macros*/
#define samples 50
#define maxVal 20 // max change limit
#define minVal -20 // min change limit
```

```
#define buzTime 5000 // buzzer on time


void setup()
{
lcd.begin(16,2); //initializing lcd
Serial.begin(9600); // initializing serial
delay(1000);
lcd.print("EarthQuake ");
lcd.setCursor(0,1);
lcd.print("Detector ");
delay(2000);
lcd.clear();
lcd.print("Calibrating.....");
lcd.setCursor(0,1);
lcd.print("Please wait...");
pinMode(buzzer, OUTPUT);
pinMode(led, OUTPUT);
buz=0;
digitalWrite(buzzer, buz);
digitalWrite(led, buz);
for(int i=0;i<samples;i++) // taking samples for calibration
{
xsample+=analogRead(x);
ysample+=analogRead(y);
zsample+=analogRead(z);
}

xsample/=samples; // taking avg for x
ysample/=samples; // taking avg for y
```

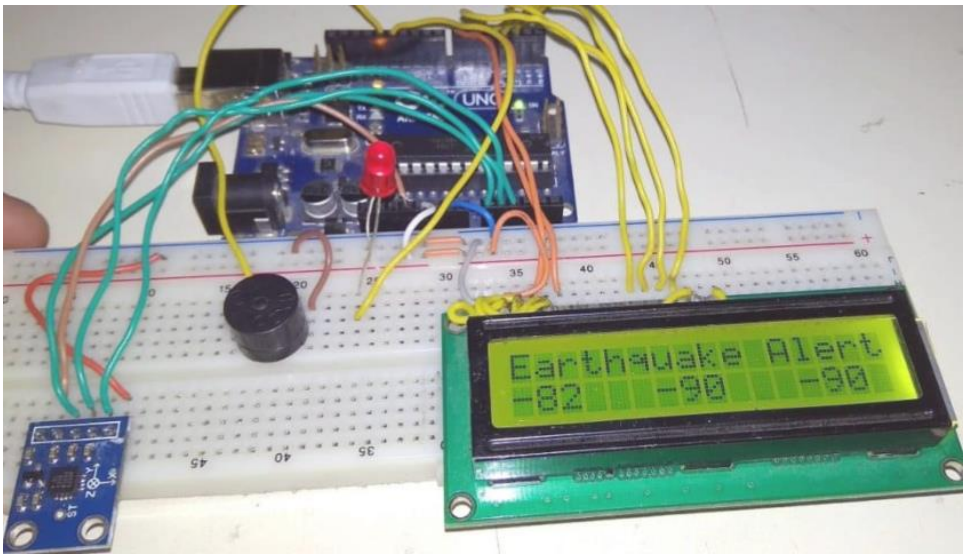
```
zsample/=samples; // taking avg for z

delay(3000);
lcd.clear();
lcd.print("Calibrated");
delay(1000);
lcd.clear();
lcd.print("Device Ready");
delay(1000);
lcd.clear();
lcd.print(" X Y Z ");
}

void loop()
{
int value1=analogRead(x); // reading x out
int value2=analogRead(y); //reading y out
int value3=analogRead(z); //reading z out

int xValue=xsample-value1; // finding change in x
int yValue=ysample-value2; // finding change in y
int zValue=zsample-value3; // finding change in z

/*displying change in x,y and z axis values over lcd*/
lcd.setCursor(0,1);
lcd.print(xValue);
lcd.setCursor(6,1);
lcd.print(yValue);
lcd.setCursor(12,1);
```

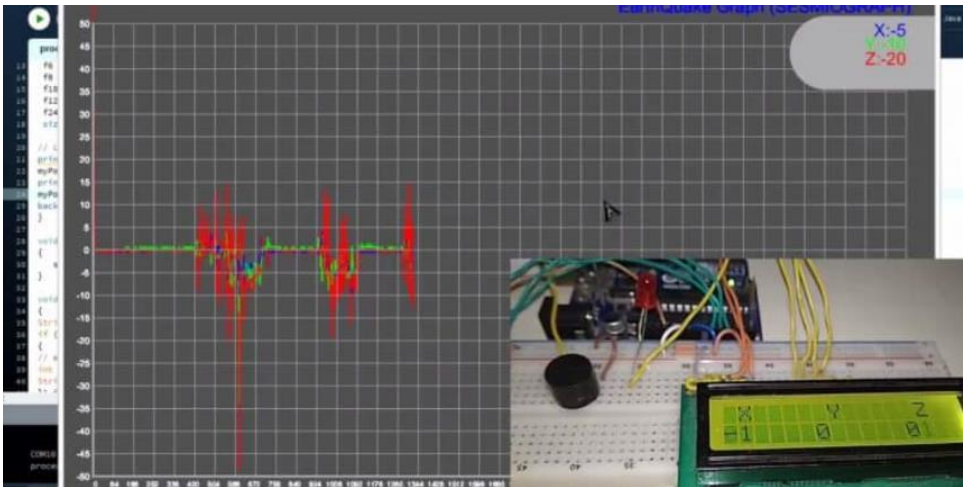
```
lcd.print(zValue);
delay(100);

/* comparing change with predefined limits*/
if(xValue < minVal || xValue > maxVal || yValue < minVal || yValue > maxVal || zValue < minVal ||
zValue > maxVal)
{
if(buz == 0)
start=millis(); // timer start
buz=1; // buzzer / led flag activated
}

else if(buz == 1) // buzzer flag activated then alerting earthquake
{
lcd.setCursor(0,0);
lcd.print("Earthquake Alert ");
if(millis()>= start+buzTime)
buz=0;
}

else
{
lcd.clear();
lcd.print(" X Y Z ");
}

digitalWrite(buzzer, buz); // buzzer on and off command
digitalWrite(led, buz); // led on and off command
```

```
/*sending values to processing for plot over the graph*/

Serial.print("x=");

Serial.println(xValue);

Serial.print("y=");

Serial.println(yValue);

Serial.print("z=");

Serial.println(zValue);

Serial.println(" $");

}
```

VIDEO OF THE PROJECT

https://www.dropbox.com/scl/fi/nhjyug9rtojah5la1hrr1/alert-system-for-earthquakes.mp4?rlkey=hfmhslt11jt14ixaxs3cvnd5i&dl=0





https://www.dropbox.com/scl/fi/nhjyug9rtojah5la1hrr1/alert-system-for-earthquakes.mp4?rlkey=hfmhslt11jt14ixaxs3cvnd5i&dl=0

https://www.dropbox.com/scl/fi/nhjyug9rtojah5la1hrr1/alert-system-for-earthquakes.mp4?rlkey=hfmhslt11jt14ixaxs3cvnd5i&dl=0

LINK FOR VIDEO KINDLY PLEASE SEE

CODE FOR SOLUTION

```
#include<LiquidCrystal.h> // lcd Header
LiquidCrystal lcd(7,6,5,4,3,2); // pins for LCD Connection

#define buzzer 12 // buzzer pin
#define led 13 //led pin

#define x A0 // x_out pin of Accelerometer
#define y A1 // y_out pin of Accelerometer
#define z A2 // z_out pin of Accelerometer

/*variables*/
int xsample=0;
int ysample=0;
int zsample=0;
```

```
long start;
int buz=0;

/*Macros*/
#define samples 50
#define maxVal 20 // max change limit
#define minVal -20 // min change limit
#define buzTime 5000 // buzzer on time

void setup()
{
lcd.begin(16,2); //initializing lcd
Serial.begin(9600); // initializing serial
delay(1000);
lcd.print("EarthQuake ");
lcd.setCursor(0,1);
lcd.print("Detector ");
delay(2000);
lcd.clear();
lcd.print("Calibrating.....");
lcd.setCursor(0,1);
lcd.print("Please wait...");
pinMode(buzzer, OUTPUT);
pinMode(led, OUTPUT);
buz=0;
digitalWrite(buzzer, buz);
digitalWrite(led, buz);
for(int i=0;i<samples;i++) // taking samples for calibration
{
xsample+=analogRead(x);
ysample+=analogRead(y);
zsample+=analogRead(z);
}

xsample/=samples; // taking avg for x
ysample/=samples; // taking avg for y
zsample/=samples; // taking avg for z

delay(3000);
```

```
lcd.clear();
lcd.print("Calibrated");
delay(1000);
lcd.clear();
lcd.print("Device Ready");
delay(1000);
lcd.clear();
lcd.print(" X Y Z ");
}

void loop()
{
int value1=analogRead(x); // reading x out
int value2=analogRead(y); //reading y out
int value3=analogRead(z); //reading z out

int xValue=xsample-value1; // finding change in x
int yValue=ysample-value2; // finding change in y
int zValue=zsample-value3; // finding change in z

/*displying change in x,y and z axis values over lcd*/
lcd.setCursor(0,1);
lcd.print(xValue);
lcd.setCursor(6,1);
lcd.print(yValue);
lcd.setCursor(12,1);
lcd.print(zValue);
delay(100);

/* comparing change with predefined limits*/
if(xValue < minVal || xValue > maxVal || yValue < minVal || yValue
> maxVal || zValue < minVal || zValue > maxVal)
{
if(buz == 0)
start=millis(); // timer start
buz=1; // buzzer / led flag activated
}


```

```
93  else if(buz == 1) // buzzer flag activated then alerting
94  earthquake
95  {
96  lcd.setCursor(0,0);
97  lcd.print("Earthquake Alert ");
98  if(millis()>= start+buzTime)
99  buz=0;
100 }
101
102 else
103 {
104 lcd.clear();
105 lcd.print(" X Y Z ");
106 }
107
108 digitalWrite(buzzer, buz); // buzzer on and off command
109 digitalWrite(led, buz); // led on and off command
110
111 /*sending values to processing for plot over the graph*/
112 Serial.print("x=");
113 Serial.println(xValue);
114 Serial.print("y=");
115 Serial.println(yValue);
116 Serial.print("z=");
    Serial.println(zValue);
    Serial.println(" $");
    }
```

```
import processing.serial.*;
PFont f6,f8,f12,f10;
PFont f24;
Serial myPort; // The serial port
int xPos = 0; // horizontal position of the graph
float y1=0;
float y2=0;
float y3=0;

void setup ()
{
// set the window size: and Font size
f6 = createFont("Arial",6,true);
f8 = createFont("Arial",8,true);
f10 = createFont("Arial",10,true);
```

```
f12 = createFont("Arial",12,true);
f24 = createFont("Arial",24,true);
size(1200, 700);

// List all the available serial ports
println(Serial.list());
myPort = new Serial(this, "COM10", 9600);
println(myPort);
myPort.bufferUntil('\n');
background(80);
}

void draw ()
{
serial ();
}

void serial()
{
String inString = myPort.readStringUntil('$'); // reading incomming date
from serial
if (inString != null)
{
// extracting all required values of all three axis:
int l1=inString.indexOf("x=")+2;
String temp1=inString.substring(l1,l1+3);
l1=inString.indexOf("y=")+2;
String temp2=inString.substring(l1,l1+3);
l1=inString.indexOf("z=")+2;
String temp3=inString.substring(l1,l1+3);

//mapping x, y and z value with graph dimensions
float inByte1 = float(temp1+(char)9);
inByte1 = map(inByte1, -80,80, 0, height-80);
float inByte2 = float(temp2+(char)9);
inByte2 = map(inByte2,-80,80, 0, height-80);
float inByte3 = float(temp3+(char)9);
inByte3 = map(inByte3,-80,80, 0, height-80);
float x=map(xPos,0,1120,40,width-40);

//ploting graph window, unit
strokeWeight(2);
stroke(175);
Line(0,0,0,100);
textFont(f24);
fill(0,00,255);
textAlign(RIGHT);
xmargin("EarthQuake Graph (SESMIOGRAPH)",200,100);
```

```
fill(100);
strokeWeight(100);
line(1050,80,1200,80);

strokeWeight(1);
textAlign(RIGHT);
fill(0,0,255);
String temp="X:"+temp1;
Text(temp,100,95);

fill(0,255,0);
temp="Y:"+temp2;
Text(temp,100,92);

fill(255,0,0);;
temp="Z:"+temp3;
Text(temp,100,89);



//ploting x y and z values over graph
strokeWeight(2);
int shift=40;

stroke(0,0,255);
if(y1 == 0)
y1=height-inByte1-shift;
line(x, y1, x+2, height-inByte1-shift) ;
y1=height-inByte1-shift;

stroke(0,255,0);
if(y2 == 0)
y2=height-inByte2-shift;
line(x, y2, x+2, height-inByte2-shift) ;
y2=height-inByte2-shift;

stroke(255,0,0);
if(y2 == 0)
y3=height-inByte3-shift;
line(x, y3, x+2, height-inByte3-shift) ;
y3=height-inByte3-shift;

xPos+=1;

if (x >= width-30) // go back to begining
{
xPos = 0;
background(80);
}
}
```

```
}

void Line(int x1, int y1, int x2, int y2)
{
float xx1=map(x1,0,100,40,width-40);
float xx2=map(x2,0,100,40,width-40);
float yy1=map(y1,0,100,height-40,40);
float yy2=map(y2,0,100,height-40,40);

line(xx1,yy1,xx2,yy2);
xx2=map(100,0,100,40,width-40);
yy2=map(0,0,100,height-40,40);
line(xx1,yy1,xx2,yy2);

strokeWeight(1);
for(int i=1;i<21;i++)
{
yy2=map(i*10,0,200,height-40,40);
yy1=yy2;
line(xx1,yy1,xx2,yy2);
}
yy2=map(100,0,100,height-40,40);
yy1=map(0,0,100,height-40,40);
for(int i=1;i<41;i++)
{
xx1=map(i*5,0,200,40,width-40);
xx2=map(i*5,0,200,40,width-40);
line(xx1,yy1,xx2,yy2);
}

textAlign(RIGHT); // 100 degree
// result+=yy1;
fill(255);
strokeWeight(1);
textFont(f12);
for(int i=-10;i<11;i++)
{
String result="";
result+=5*i;
ymargin(result, x1,y1);
y1+=5;
}

x1=0;
y1=0;
strokeWeight(1);
textFont(f10);
for(int i=0;i<41;i++)
{
```

```
String result="";
result+=28*3*i;
xmargin(result, x1,y1);
x1+=5;
}

textAlign(RIGHT);

textAlign(RIGHT);
}

void ymargin(String value, int x1, int y1)
{

float xx1=map(x1,0,100,40,width-40);
float yy1=map(y1,0,100,height-40,40);
text(value,xx1-5,yy1+5);
}

void xmargin(String value, int x1, int y1)
{

float xx1=map(x1,0,200,40,width-40);
float yy1=map(y1,0,100,height-25,25);
text(value,xx1+7,yy1);
}

void Text(String value, int x1, int y1)
{

float xx1=map(x1,0,100,40,width-40);
float yy1=map(y1,0,100,height-25,25);
text(value,xx1,yy1);
}
```