LEVELING SYSTEM CODE

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

MPU6050 mpu;
int16_t ax, ay, az;
int16_t gx, gy, gz;

struct MyData {
  byte X;
  byte Y;
  byte Z;
};

MyData data;

const int ledPin = 8;    // Pin connected to the LED
const int buzzer = 7;
const int sideLaser = 6;

int status = 0;
unsigned long previousAccelTime = 0;
const unsigned long accelInterval = 500;   // Interval for reading accelerometer data (in
milliseconds)

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();

  pinMode(buzzer, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(sideLaser, OUTPUT);
}

void loop()
{
  digitalWrite(sideLaser, HIGH);

  unsigned long currentMillis = millis();

  // Read accelerometer data at the specified interval
  if (currentMillis - previousAccelTime >= accelInterval)
  {
    previousAccelTime = currentMillis;

    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

```
    data.X = map(ax, -17000, 17000, 0, 255); // X axis data
    data.Y = map(ay, -17000, 17000, 0, 255);
    data.Z = map(az, -17000, 17000, 0, 255);  // Y axis data
  }

  if (data.Z >= 9 && data.Z < 13)
  {
    if (status == 0) {
      digitalWrite(ledPin, HIGH);
      status = 1;
      beep();
    }
  }
  else {
    // Blinking the LED
    digitalWrite(ledPin, LOW);
    delay(75);
    digitalWrite(ledPin, HIGH);
    delay(75);

    status = 0;
  }

  Serial.print("Axis X = ");
  Serial.print(data.X);
  Serial.print("  ");
  Serial.print("Axis Y = ");
  Serial.print(data.Y);
  Serial.print("  ");
  Serial.print("Axis Z  = ");
  Serial.println(data.Z);
}

// Function that makes the melody of the buzzer
void beep() {
  digitalWrite(buzzer, HIGH);
  delay(75);
  digitalWrite(buzzer, LOW);
  delay(75);
  digitalWrite(buzzer, HIGH);
  delay(75);
  digitalWrite(buzzer, LOW);
  delay(75);
  digitalWrite(buzzer, HIGH);
  delay(75);
  digitalWrite(buzzer, LOW);
  delay(75);
}
```

MAIN SYSTEM CODE

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

int potpin = 0;
int val;
Servo myservo;

const int trigPin = 9;
const int echoPin = 10;

//laser pins
const int servoLaser = 4;
const int constantLaser = 5;

const float rad = 0.01745329252;

unsigned long previousMillis = 0;
const unsigned long interval = 1000;

LiquidCrystal_I2C lcd1(0x26, 16, 2);
LiquidCrystal_I2C lcd2(0x27, 20, 4);

byte customChar[] = {
  B00000,
  B00000,
  B11111,
  B01010,
  B01010,
  B01010,
  B01010,
  B00000
};


void setup() {
  Serial.begin(9600);
  myservo.attach(3);  // Attaches the servo to pin 3
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(servoLaser, OUTPUT);
  pinMode(constantLaser, OUTPUT);

  lcd1.init();
  lcd2.init();

  lcd1.backlight();
```

```
  lcd2.backlight();
}

void loop() {
  unsigned long currentMillis = millis();

  //on the lasers
  digitalWrite(servoLaser, HIGH);
  digitalWrite(constantLaser, HIGH);


  // Read the potentiometer
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 184);

  // Measure distance with HC-SR04
  long duration;
  float distance;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  //Negating the Angle
  int ang = 0;
  if (val >= 90 && val < 180) {
    ang = val - 90;
  } else if (val >= 180) {
    ang = 90;
  } else {
    ang = -1 * (90 - val);
  }

  //Height and Hypotenuse Variables
  float height = (distance) * (tan(ang * rad));
  float hypotenuse = sqrt(pow(height, 2) + pow(distance, 2));

  //Negative Height
  if (ang < 0) {
    height = -1 * height;
  }


  // Print values every one second
  if (currentMillis - previousMillis >= interval) {
```

```
previousMillis = currentMillis;


//Trigonometric Ratios Formula
float sinx = sin(ang * rad);
float cosx = cos(ang * rad);
float tanx = tan(ang * rad);

float cscx = 1 / sinx;
float secx = 1 / cosx;
float cotx = 1 / tanx;


//Radians Calculation
int degrees = abs(ang);
float radians = degrees * 3.14159 / 180.0;

// Calculate Fraction Form
int numerator = round(radians * 180.0 / 3.14159);
int denominator = 180;

// Find the greatest common divisor
int gcd = findGCD(numerator, denominator);

// Simplify the fraction
numerator /= gcd;
denominator /= gcd;



//Configuring the 2 LCD Displays
if (ang >= 90 || ang == -90) {
  //small lcd
  lcd1.clear();
  lcd1.setCursor(0, 0);
  lcd1.print(String(char(223)) + ": " + String(ang));
  lcd1.setCursor(0, 1);
  lcd1.print("A:INF");
  lcd1.setCursor(8, 0);
  lcd1.print("B:" + String(distance, 1));
  lcd1.setCursor(8, 1);
  lcd1.print("C:INF");

  //big lcd left
  lcd2.clear();

  lcd2.setCursor(0, 0);
  lcd2.print("DEG:" + String(ang));
  lcd2.setCursor(9, 0);
  lcd2.print("|R:");
```

```
if (ang < 0) {
  if (numerator == 1) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");
    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(13, 0);
    lcd2.write(0);

    lcd2.setCursor(14, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator < 10) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");

    lcd2.setCursor(13, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(14, 0);
    lcd2.write(0);

    lcd2.setCursor(15, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator >= 10) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");

    lcd2.setCursor(13, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(15, 0);
    lcd2.write(0);

    lcd2.setCursor(16, 0);
    lcd2.print("/" + String(denominator));
  }
} else {
  if (numerator == 1) {
    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(12, 0);
    lcd2.write(0);
```

```
      lcd2.setCursor(13, 0);
      lcd2.print("/" + String(denominator));
    }

    else if (numerator < 10) {
      lcd2.setCursor(12, 0);
      lcd2.print(numerator);

      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(13, 0);
      lcd2.write(0);

      lcd2.setCursor(14, 0);
      lcd2.print("/" + String(denominator));
    }

    else if (numerator >= 10) {
      lcd2.setCursor(12, 0);
      lcd2.print(numerator);

      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(14, 0);
      lcd2.write(0);

      lcd2.setCursor(15, 0);
      lcd2.print("/" + String(denominator));
    }
  }

  //left column
  lcd2.setCursor(0, 1);
  lcd2.print("sin:" + String(sinx, 2));
  lcd2.setCursor(0, 2);
  lcd2.print("cos:" + String(cosx, 2));
  lcd2.setCursor(0, 3);
  lcd2.print("tan:INF");

  //right column
  lcd2.setCursor(9, 1);
  lcd2.print("|csc:" + String(cscx, 2));
  lcd2.setCursor(9, 2);
  lcd2.print("|sec:INF");
  lcd2.setCursor(9, 3);
  lcd2.print("|cot:" + String(cotx, 2));
}

else if (ang >= -5 && ang <= -1) {
```

```
//small lcd
lcd1.clear();
lcd1.setCursor(0, 0);
lcd1.print(String(char(223)) + ": " + String(ang));
lcd1.setCursor(0, 1);
lcd1.print("A: " + String(height, 1));
lcd1.setCursor(8, 0);
lcd1.print("B: " + String(distance, 1));
lcd1.setCursor(8, 1);
lcd1.print("C: " + String(hypotenuse, 1));

lcd2.clear();

lcd2.setCursor(0, 0);
lcd2.print("DEG:" + String(ang));
lcd2.setCursor(9, 0);
lcd2.print("|R:");

if (ang < 0) {
  if (numerator == 1) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");
    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(13, 0);
    lcd2.write(0);

    lcd2.setCursor(14, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator < 10) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");

    lcd2.setCursor(13, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(14, 0);
    lcd2.write(0);

    lcd2.setCursor(15, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator >= 10) {
    lcd2.setCursor(12, 0);
    lcd2.print("-");
```

```
    lcd2.setCursor(13, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(15, 0);
    lcd2.write(0);

    lcd2.setCursor(16, 0);
    lcd2.print("/" + String(denominator));
  }
} else {
  if (numerator == 1) {
    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(12, 0);
    lcd2.write(0);

    lcd2.setCursor(13, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator < 10) {
    lcd2.setCursor(12, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(13, 0);
    lcd2.write(0);

    lcd2.setCursor(14, 0);
    lcd2.print("/" + String(denominator));
  }

  else if (numerator >= 10) {
    lcd2.setCursor(12, 0);
    lcd2.print(numerator);

    lcd2.createChar(0, customChar);
    lcd2.home();
    lcd2.setCursor(14, 0);
    lcd2.write(0);

    lcd2.setCursor(15, 0);
    lcd2.print("/" + String(denominator));
  }
}
```

```
  lcd2.setCursor(0, 1);
  lcd2.print("sin:" + String(sinx, 2));
  lcd2.setCursor(0, 2);
  lcd2.print("cos:" + String(cosx, 2));
  lcd2.setCursor(0, 3);
  lcd2.print("tan:" + String(tanx, 2));

  //big lcd right
  lcd2.setCursor(9, 1);
  lcd2.print("|csc:" + String(cscx, 2));
  lcd2.setCursor(9, 2);
  lcd2.print("|sec:" + String(secx, 2));
  lcd2.setCursor(9, 3);
  lcd2.print("|cot:" + String(cotx, 2));
}

else {
  //small lcd
  lcd1.clear();
  lcd1.setCursor(0, 0);
  lcd1.print(String(char(223)) + ":" + String(ang));
  lcd1.setCursor(0, 1);
  lcd1.print("A:" + String(height, 1));
  lcd1.setCursor(9, 0);
  lcd1.print("B:" + String(distance, 1));
  lcd1.setCursor(9, 1);
  lcd1.print("C:" + String(hypotenuse, 1));

  //big lcd left
  lcd2.clear();

  lcd2.setCursor(0, 0);
  lcd2.print("DEG:" + String(ang));
  lcd2.setCursor(9, 0);
  lcd2.print("|R:");

  if (ang < 0) {
    if (numerator == 1) {
      lcd2.setCursor(12, 0);
      lcd2.print("-");
      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(13, 0);
      lcd2.write(0);

      lcd2.setCursor(14, 0);
      lcd2.print("/" + String(denominator));
    }

    else if (numerator < 10) {
```

```
      lcd2.setCursor(12, 0);
      lcd2.print("-");

      lcd2.setCursor(13, 0);
      lcd2.print(numerator);

      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(14, 0);
      lcd2.write(0);

      lcd2.setCursor(15, 0);
      lcd2.print("/" + String(denominator));
    }

    else if (numerator >= 10) {
      lcd2.setCursor(12, 0);
      lcd2.print("-");

      lcd2.setCursor(13, 0);
      lcd2.print(numerator);

      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(15, 0);
      lcd2.write(0);

      lcd2.setCursor(16, 0);
      lcd2.print("/" + String(denominator));
    }
  } else {
    if (numerator == 1) {
      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(12, 0);
      lcd2.write(0);

      lcd2.setCursor(13, 0);
      lcd2.print("/" + String(denominator));
    }

    else if (numerator < 10) {
      lcd2.setCursor(12, 0);
      lcd2.print(numerator);

      lcd2.createChar(0, customChar);
      lcd2.home();
      lcd2.setCursor(13, 0);
      lcd2.write(0);
```

```
          lcd2.setCursor(14, 0);
          lcd2.print("/" + String(denominator));
        }

        else if (numerator >= 10) {
          lcd2.setCursor(12, 0);
          lcd2.print(numerator);

          lcd2.createChar(0, customChar);
          lcd2.home();
          lcd2.setCursor(14, 0);
          lcd2.write(0);

          lcd2.setCursor(15, 0);
          lcd2.print("/" + String(denominator));
        }
      }

      lcd2.setCursor(0, 1);
      lcd2.print("sin:" + String(sinx, 2));
      lcd2.setCursor(0, 2);
      lcd2.print("cos:" + String(cosx, 2));
      lcd2.setCursor(0, 3);
      lcd2.print("tan:" + String(tanx, 2));

      //big lcd right
      lcd2.setCursor(9, 1);
      lcd2.print("|csc:" + String(cscx, 2));
      lcd2.setCursor(9, 2);
      lcd2.print("|sec:" + String(secx, 2));
      lcd2.setCursor(9, 3);
      lcd2.print("|cot:" + String(cotx, 2));
    }
  }

  //writing the servo with no delay
  myservo.write(val);
  delay(15);
}


// Function to find the greatest common divisor (Euclidean algorithm)
int findGCD(int a, int b) {
  while (b != 0) {
    int temp = b;
    b = a % b;
    a = temp;
  }
  return a;
}
```