وزارة التعليم العالى

المعهـد العالــى

للهندسه والتكنولوجيا بالعبور

Posture Corrector using MPU6050

Prepared by

No.	Student name	Student code
1	Ramadan Mohamed Hassan Sokkar	221340
2	Abdelrahman Mohamed Hamad Mabrouk	221332
3	Abanoub Sabry Abdelsayed Elias	221352

Under supervision

Dr. Osama AbdelMoneim

Eng. Mina Osama

Medical Engineering

First semester (2025-2026)

1. Abstract:

This project presents the design and implementation of a wearable posture correction system using an MPU6050 accelerometer/gyroscope sensor and an Arduino Nano microcontroller. The system continuously monitors the user’s upper-body orientation and provides an alert via a buzzer whenever poor posture is detected. A 9V battery along with a 7805 voltage regulator ensures stable power delivery. The system is suitable for daily use and can assist in preventing back pain and spinal alignment issues

2. Introduction:

Poor posture is a common issue that leads to musculoskeletal problems, especially among students, office workers, and individuals who spend long hours sitting. Continuous monitoring of the spine angle can help reduce long-term damage. This project aims to create a simple, low-cost, and wearable device that alerts the user whenever the body tilts beyond a safe angle.

The device uses the MPU6050 sensor to detect posture orientation and the Arduino Nano as the main processing unit. A buzzer provides feedback through sound alerts, allowing the user to correct their posture in real time.

3. System Model:

3.1 Power Supply Unit

- **Battery (9V):** Provides portable power to the device.
- **Voltage Regulator (7805):** Regulates the battery voltage to a stable +5V.
- **Capacitors (C1 = 0.22µF, C2 = 0.1µF):** Remove noise and stabilize the voltage.

3.2 Control Unit

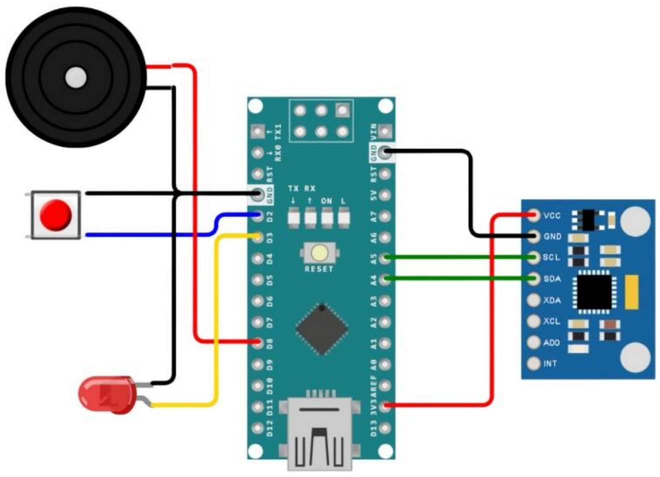
- **Arduino Nano:**
  - Reads sensor data from MPU6050.
  - Processes angle and compares it to the calibrated reference angle.
  - Controls the alert output.

3.3 Motion Sensing Unit

- **MPU6050 Accelerometer + Gyroscope Module:**
  - Measures pitch angle.
  - Communicates with Arduino via **I2C (SCL & SDA)**.

3.4 Alert/Feedback Unit

- **Buzzer (BUZ1):** Activates when poor posture is detected.
- **Transistor (2N2222 – Q1):** Drives the buzzer with sufficient current.
- **Resistors (R2 = 1 kΩ, R3 = 220 Ω):** Used for transistor base and buzzer protection.
- **Push Button:** Used for calibration of the neutral posture.

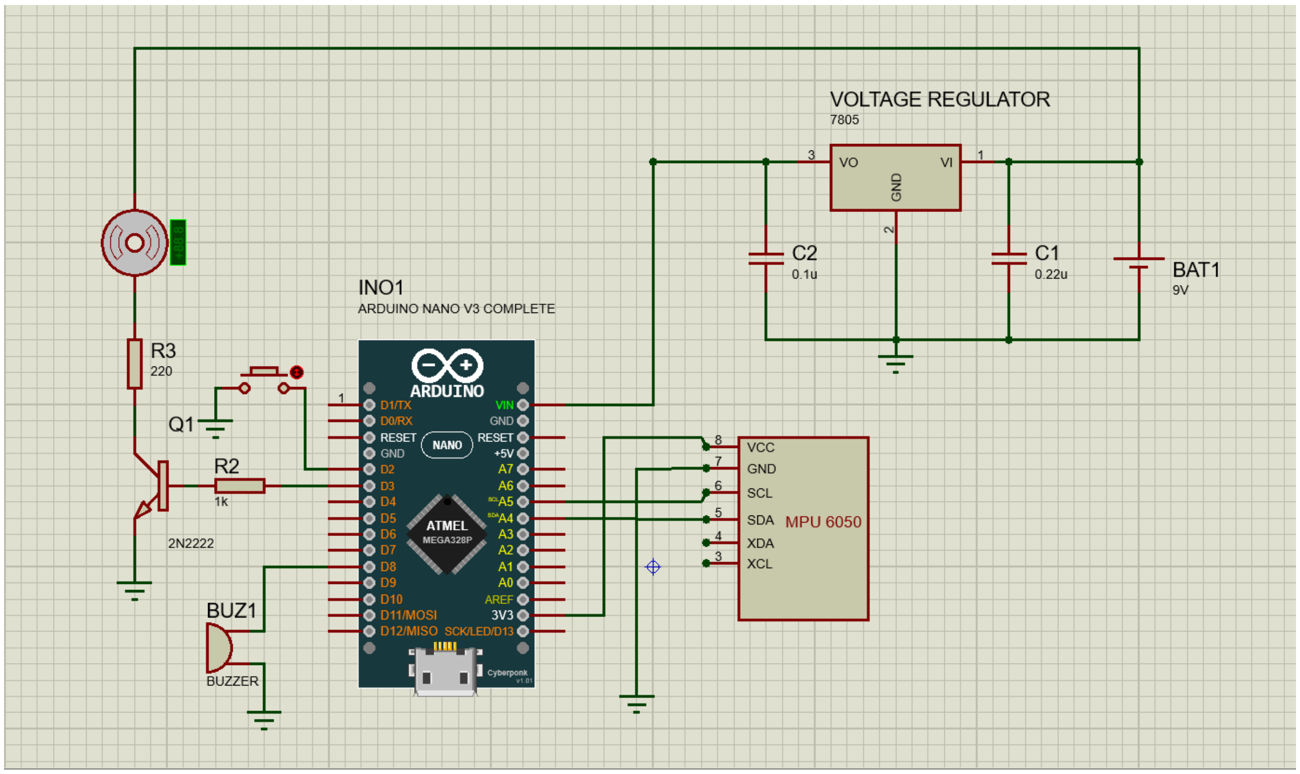


4. Simulation:

Simulation was conducted using **Proteus:**

- Verify correct wiring of the **MPU6050.**
- Test **buzzer activation** based on simulated **angle values.**
- Validate the 5V regulated power supply.
- Ensure the transistor switching operation is correct.

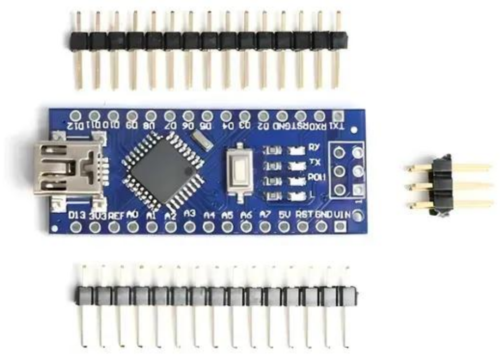
The simulation successfully demonstrated the expected behavior of the system.



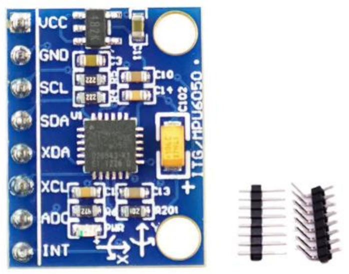
5. Hardware discerption:

No.	Component Name	Description
I.	Arduino Nano	Main microcontroller, executes program logic
II.	MPU6050	3axis accelerometer & 3axis gyroscope
III.	Regulator (7805)	Converts 9V to stable 5V
IV.	Push Button	Used to calibrate neutral posture
V.	Transistor (2N2222)	Acts as a driver for buzzer
VI.	Vibrating Disk Motor	Provides haptic vibration feedback for posture alerts
VII.	Buzzer	Generates audible alerts
VIII.	Resistors (1kΩ, 220Ω)	Base resistor & current limiting
IX.	Capacitors (0.22µF, 0.1µF)	Filtering capacitors
X.	Battery (9V)	Power source

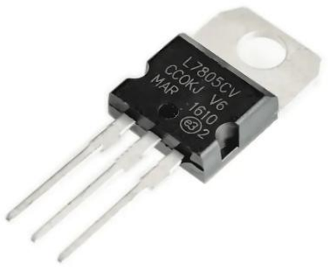
➢ **Arduino Nano:**



➢ **MPU6050:**



➢ **Regulator (7805):**



➢ **Push Button:**

- **Transistor (2N2222):**

- **Buzzer:**

- **Resistors (1kΩ, 220Ω):**

- **Capacitors (0.22µF, 0.1µF):**

- **Battery (9V):**

- **Vibrating Disk Motor:**

6. Code

```
#include <Wire.h>           // I2C library for communication
#include "MPU6050.h"        // MPU6050 sensor library

MPU6050 mpu;

const int calibButton = 2;    // Calibration button
const int ledPin = 8;         // LED turns on during calibration
const int vibPin = 3;         // Pin for vibration motor (PWM)

float neutralPitch = 0.0;     // Neutral sitting angle after calibration

const float POSTURE_THRESHOLD = 20.0; // Allowed difference before considering
posture incorrect (degrees)

unsigned long reminderInterval = 5UL * 60UL * 1000UL;
// Sitting time before sending gentle reminder (5 minutes for testing – should be 40
minutes in final project)

const int gentleDuty = 90;    // Soft vibration intensity (Reminder)
const int strongDuty = 255;   // Strong vibration intensity (Bad posture)
const unsigned long gentleDuration = 300;  // Soft vibration duration (ms)
const unsigned long strongDuration = 1000; // Strong vibration duration (ms)

unsigned long lastGoodStart = 0;    // Last time posture was good
bool inBad = false;                 // Was the user in bad posture recently?

// -------------------------------------
//              SETUP
// -------------------------------------
void setup(){
  Serial.begin(115200);  // Start serial monitor
  Wire.begin();          // Start I2C
  mpu.initialize();      // Initialize MPU6050

  pinMode(calibButton, INPUT_PULLUP); // Calibration button with pull-up
  pinMode(ledPin, OUTPUT);
  pinMode(vibPin, OUTPUT);

  analogWrite(vibPin, 0); // Ensure motor is off
  delay(200);
  // First calibration at startup
  calibrateNeutral();

  // Begin timer
  lastGoodStart = millis();
}

// -------------------------------------
//                LOOP
// -------------------------------------
void loop(){
  int16_t ax, ay, az, gx, gy, gz;

  // Read accelerometer and gyroscope
  mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);

  // Convert raw values to g
  float ax_f = ax/16384.0;
  float az_f = az/16384.0;

  // Calculate pitch angle
  float pitch = atan2(ax_f, az_f) * 57.295779513;

  // Difference between current and calibrated posture
  float rel = pitch - neutralPitch;

  unsigned long now = millis();

  // ----------- Calibration Button -----------
  if (digitalRead(calibButton) == LOW) {
    calibrateNeutral();   // Reset neutral posture
    lastGoodStart = now;  // Reset timer
    delay(300);
  }

  // ----------- Detect Bad Posture -----------
  if (abs(rel) > POSTURE_THRESHOLD) {

    // Strong vibration warning for incorrect posture
    analogWrite(vibPin, strongDuty);
    delay(strongDuration);
    analogWrite(vibPin, 0);

    // Reset timer after alert
    lastGoodStart = now;
    inBad = true;

    Serial.print("Bad posture! rel=");
    Serial.println(rel);

    delay(200);
    return;  // Exit loop and read again
  }

  // If posture was bad but now corrected
  else {
    if (inBad) {
      inBad = false;
      lastGoodStart = now;
      Serial.println("Recovered"); // Posture corrected
    }
  }

  // ----------- Gentle Reminder After Long Sitting -----------
  if (now - lastGoodStart >= reminderInterval) {

    // Soft vibration reminder
    analogWrite(vibPin, gentleDuty);
    delay(gentleDuration);
    analogWrite(vibPin, 0);

    Serial.println("Gentle reminder: time to move");

    // Reset timer after reminder
    lastGoodStart = now;
  }

  // Print data for debugging
  Serial.print("Pitch: ");
  Serial.print(pitch);
  Serial.print("  Rel: ");
  Serial.println(rel);

  delay(1000);
}

// ------------------------------------
//       FUNCTION: Posture Calibration
// ------------------------------------
void calibrateNeutral(){
  Serial.println("Calibrating... hold straight");

  long sum = 0;
  int samples = 30;

  // Take 30 readings and calculate average pitch
  for (int i=0;i<samples;i++){
    int16_t ax,ay,az,gx,gy,gz;
    mpu.getMotion6(&ax,&ay,&az,&gx,&gy,&gz);

    float ax_f = ax/16384.0;
    float az_f = az/16384.0;

    float p = atan2(ax_f, az_f) * 57.295779513;
    sum += p;

    delay(60);
  }

  // Set neutral posture angle
  neutralPitch = sum / (float)samples;

  Serial.print("Neutral set: ");
  Serial.println(neutralPitch);

  // Flash LED to indicate calibration success
  digitalWrite(ledPin, HIGH);
  delay(200);
  digitalWrite(ledPin, LOW);
}
```

7. Working of the Circuit:

The circuit operation begins with the MPU6050 sensor continuously measuring the acceleration and angular motion of the user's upper body. These measurements are transmitted to the Arduino Nano through the I^2C communication interface (SDA and SCL pins).

The Arduino processes the sensor data and calculates the tilt angle relative to the calibrated neutral posture. When the measured posture angle exceeds the predefined threshold value ($25\ ^{O}$), the Arduino outputs a HIGH signal to the base of the 2N2222 transistor through a current-limiting resistor.

Once the transistor switches ON, it allows sufficient current to flow from the power supply through the vibrating disk motor. As current passes through the motor, its internal eccentric rotating mass spins, generating mechanical vibration. This vibration acts as a haptic alert, notifying the user to correct their posture.

In addition, a push button is used to recalibrate the neutral posture position. When pressed, the Arduino records the current sensor orientation as the new reference posture. Filtering capacitors connected to the voltage regulator ensure a stable 5V supply, improving system reliability and minimizing noise effects.

Operating steps:

i. The 9V battery feeds the 7805 regulator, producing a clean 5V output.

ii. The Arduino Nano is powered through its 5V input pin.

iii. The MPU6050 provides real-time pitch angle data to the Arduino.

iv. When the body tilts more than the threshold:

- Arduino sends a HIGH signal to the base of the 2N2222 transistor through the 1kΩ resistor.
- The transistor switches on and allows current to flow through the vibrating disk motor.
- The vibrating disk motor generating mechanical vibration to notify the user.
- The buzzer generates an alert sound to notify the user.

v. Pressing the calibration button resets the neutral posture angle.

8. **Result**:

The implemented system was tested under different posture conditions to evaluate its performance. During normal sitting posture, the calculated tilt angle remained within the allowed threshold, and the vibrating disk motor remained inactive.

When the user intentionally leaned forward or backward beyond the threshold angle, the system successfully detected the posture deviation.
The Arduino activated the transistor, allowing current to flow through the vibrating disk motor, which produced a clear vibration feedback.
This vibration effectively alerted the user to adjust their posture.

The system demonstrated fast response time, stable operation, and consistent vibration alerts. The results confirm that the designed hardware and control algorithm work together effectively to provide reliable posture monitoring and haptic feedback.

9. **Conclusion:**

A practical and low-cost posture correction device has been developed using Arduino and the MPU6050 sensor. The system is easy to wear, simple to use, and provides immediate audio feedback. The design is energy-efficient and suitable for portable operation using a 9V battery.

Future improvements may include:

- Adding vibration motor instead of buzzer
- Bluetooth connectivity for mobile app logging
- Using rechargeable battery module

10. References:

1. Real-Time Forward Head Posture Detection and Correction System Utilizing an Inertial Measurement Unit Sensor.
   https://www.mdpi.com/2076-3417/14/19/9075
2. The project mainly deals with a posture correcting hardware system using esp8266.
   https://www.hackster.io/the-circuit-breakers/posture-corrector-d729a3
3. Posture-Corrector-Robot.
   https://github.com/NorbertZare/Posture-Corrector-Robot-
4. Design and Implementation of a Novel System for Correcting Posture Through the Use of a Wearable Necklace Sensor.
   https://pmc.ncbi.nlm.nih.gov/articles/PMC6660123/#fn-group1
5. A DIY Wearable Posture Sensor.
   https://coretechrobotics.blogspot.com/2016/03/a-diy-wearable-posture-sensor.html