# EE381 (EC) LAB PROJECT

## Pedometer and Movement Detection System using Arduino

**Name(s) - K.C Sree Samhitha (220517) & Gude Dayana (220414)**

**Table No/Day - 9 (Monday)**

**Section - A**

## 1. What problem are you trying to solve, and why is it important/interesting?

Our project tackles the need for a customizable, cost-effective, and open-source solution for step counting and movement detection — essential in fitness tracking, rehabilitation, and motion-aware applications. Most commercial pedometers are closed systems, offering limited control, expansion, and educational insight.

With our Arduino-based pedometer, we aim to:
- Provide real-time feedback on movement using raw accelerometer data.
- Enable transparent, modifiable systems for learning and experimentation.
- Deliver step count updates wirelessly via Bluetooth.
- Encourage physical activity through a buzzer-based feedback loop.

## 2. What are the existing solutions? Is your approach unique?

Existing Solutions:

- Commercial Pedometers: Compact and user-friendly but closed-source.
- Fitness Bands/Smartwatches: Expensive and proprietary.
- Smartphone Apps: Battery-draining and limited by phone placement.
- Buzzer: Can be used as a gamification element.

Shortcomings:

- Limited customizability and expandability.
- Costly for prototyping or learning environments.

Our Unique Approach:

- Bluetooth Integration: Real-time updates via HC-05 and we directly paired it to our mobile.
- Behavioral Feedback Loop: Buzzer for every 25 steps.
- Low-Cost Design: Uses only essential, affordable components.

## 3. Implementation

Working:

- MPU6050 measures acceleration and gyro data.
- Arduino detects step-like movement using logic from raw data.
- A buzzer activates every 25 steps.
- Step count is sent to a mobile device via Bluetooth.

Unique Feature: Fully modular and reusable — suitable for connecting it to a mobile.

## 4. Resources Required

Hardware:

- Arduino Nano
- MPU6050
- HC-05
- Buzzer
- Potentiometer
- Resistors
- Breadboard
- Jumper Wires
- USB Cable

Software:

- Arduino IDE

- Serial Monitor /Bluetooth Terminal App

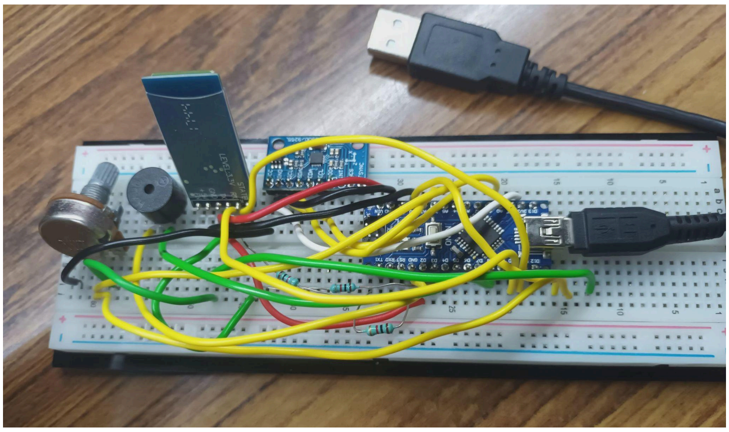
## 5. Weekly Work Breakdown

| Week | Tasks Completed |
|---|---|
| Week 1 | Component Research and Part Collection. Tested MPU6050 outputs. |
| Week 2 | Set up Arduino IDE and began code development for step detection logic. |
| Week 3 | Integrated Bluetooth communication, verified real-time data transmission. Finalized the hardware on the breadboard, added buzzer logic, and debugged the system. |

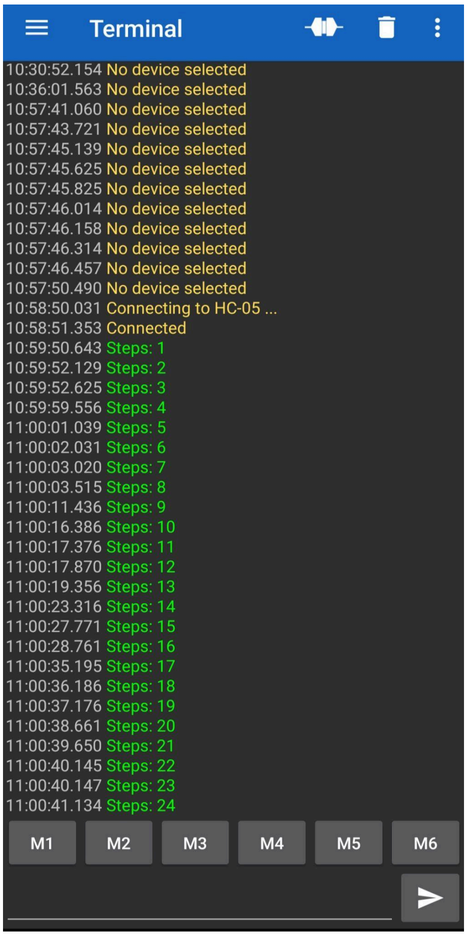
## 6. Code Used

```
#include <Wire.h>
#include <MPU6050_light.h>
#include <SoftwareSerial.h>

// Initialize Bluetooth Serial (D10 = RX, D11 = TX)
SoftwareSerial BT(10, 11); // HC-05: TX to D10, RX to D11 (via voltage divider)

// Initialize MPU6050 with I2C Wire
MPU6050 mpu(Wire);

// Step detection variables
int stepCount = 0;
bool stepDetected = false;
const float threshold = 1.2; // Acceleration threshold for step
unsigned long lastStepTime = 0;
const unsigned long debounceTime = 300; // Time gap between steps

// Buzzer
const int buzzerPin = 8;

void setup() {
  Serial.begin(9600);      // Debug via Serial Monitor
  BT.begin(9600);          // Bluetooth output
  Wire.begin();            // Start I2C

  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW);

  // Initialize MPU6050
  if (mpu.begin() != 0) {
    Serial.println("MPU FAIL!");
    BT.println("MPU FAIL!");
    while (1); // Stop here
  }

  // Calibrate MPU6050
  mpu.calcOffsets(true, true); // true = print offsets to Serial

  Serial.println("Pedometer Ready");
  BT.println("Pedometer Ready");
  delay(1500);
}

void loop() {
  mpu.update(); // Read accelerometer data

  float az = mpu.getAccZ(); // Get vertical acceleration
  unsigned long now = millis();

  // Step detection logic
  if (az > threshold && !stepDetected && (now - lastStepTime >
debounceTime)) {
    stepCount++;
    stepDetected = true;
    lastStepTime = now;

    // Optional buzzer every 25 steps
    if (stepCount % 25 == 0) {
      beep();
    }

    // Send data via Bluetooth and Serial
    BT.print("Steps: ");
    BT.println(stepCount);
    Serial.print("Steps: ");
    Serial.println(stepCount);
  }

  // Reset detection once below lower threshold
  if (az < 0.8) {
    stepDetected = false;
  }

  delay(100); // Polling delay
}

void beep() {
  digitalWrite(buzzerPin, HIGH);
  delay(200);
  digitalWrite(buzzerPin, LOW);
}
```

## 7. Project Results and Images



(a). Set up of the circuit



```
10:30:52.154 No device selected
10:36:01.563 No device selected
10:57:41.060 No device selected
10:57:43.721 No device selected
10:57:45.139 No device selected
10:57:45.625 No device selected
10:57:45.825 No device selected
10:57:46.014 No device selected
10:57:46.158 No device selected
10:57:46.314 No device selected
10:57:46.457 No device selected
10:57:50.490 No device selected
10:58:50.031 Connecting to HC-05 ...
10:58:51.353 Connected
10:59:50.643 Steps: 1
10:59:52.129 Steps: 2
10:59:52.625 Steps: 3
10:59:59.556 Steps: 4
11:00:01.039 Steps: 5
11:00:02.031 Steps: 6
11:00:03.020 Steps: 7
11:00:03.515 Steps: 8
11:00:11.436 Steps: 9
11:00:16.386 Steps: 10
11:00:17.376 Steps: 11
11:00:17.870 Steps: 12
11:00:19.356 Steps: 13
11:00:23.316 Steps: 14
11:00:27.771 Steps: 15
11:00:28.761 Steps: 16
11:00:35.195 Steps: 17
11:00:36.186 Steps: 18
11:00:37.176 Steps: 19
11:00:38.661 Steps: 20
11:00:39.650 Steps: 21
11:00:40.145 Steps: 22
11:00:40.147 Steps: 23
11:00:41.134 Steps: 24
```

(b). Output observed

## 8. Future Scope

- Enclosure design for a wearable application.
- Step detection optimization using machine learning.
- Integration with cloud-based health tracking platforms.
- Addition of OLED/LCD display for standalone use.