Energy harvesting shoe module with Arduino integration

Higher National Diploma in Software Engineering

Robotic project Final Report

HNDSE 2025.1F/CO

Student Index: COHDSE251F-041

Student Name: W.D.Y.Nathasha

Group project

School of Computing and Engineering

National Institute of Business Management

Colombo-7

2025

Abstract

This project focuses on making a smart shoe capable of producing electricity through walking , By using utilizing piezoelectric sensors. The mechanical energy generated with each step is converted into electrical power and that low capacity power increase 5v using boost converter. The electricity produced is stored in a rechargeable battery, which can later be used to charge small devices like mobile phones, smartwatches, or headphones.

In this project include main controller as Arduino Nano and LCD display that use to show Battery level ,voltage , step count and the reset (optional) and other one for ON/OFF. Also use voltage sensor to detect the voltage of the foot steps.

By integrating energy harvesting technology with smart electronics, this project not only tracks steps but also generates practical energy. It offers an innovative and convenient method to produce electricity while on the move.

Acknowledgement

- We would like to thank our Robotics Project lecturer, Mr. Supun Asanga, for his constant support, guidance, and valuable advice throughout this project. His insights and encouragement were crucial in helping us complete our work successfully.

- We also want to express our gratitude to the faculty and staff of NIBM for creating a supportive learning environment and providing academic resources that made this project possible.

- Special thanks to our team members for their hard work, commitment, and collaboration. Their dedication, creativity, and teamwork were essential in bringing this project to life.

Table of Contents

Table of Figures

Introduction

Today, people use many portable devices like phones, smartwatches, and wireless earphones. These devices need frequent charging, which can be difficult when you are outside or traveling. To address this problem, we developed a Smart Energy Harvesting Shoe Module. This module can be attached to your shoe and generates electricity as you walk.

How It Works

1. **Walking presses the piezoelectric discs:** Each time you take a step, the pressure applied on the discs generates a small amount of electricity.
2. **Diodes ensure the electricity flows in the correct direction:** The generated electricity needs to flow in one direction to charge the battery effectively, and diodes help achieve that.
3. **A boost converter raises the small voltage to charge the battery:** Since the voltage generated by the discs is small, a boost converter is used to increase it to a level suitable for charging the battery.
4. **A buck converter regulates the voltage from the battery to safely power the Arduino and LCD:** The Arduino and LCD require specific voltage levels to operate, so a buck converter is used to regulate the voltage coming from the battery.
5. **Finally, devices can be charged through a USB port using the stored energy:** The energy generated and stored in the battery can be used later to charge small devices such as phones or headphones through a USB port.

Features

The system also includes several features for monitoring and controlling its operation:

- **Step count display:** One of the push buttons allows you to view the total number of steps taken.
- **Battery level indication:** Another button shows you the current battery level, so you know when it's time to recharge.
- **Voltage monitoring:** By pressing a specific button, you can check the voltage output of the system.
- **Value reset option:** If needed, there's an option to reset certain values back to zero.
- **Power switch functionality:** One button is dedicated solely for turning the entire system on or off.

An LCD screen provides visual feedback by displaying information whenever any of these buttons are pressed. Additionally, a voltage sensor continuously monitors the battery capacity ensuring efficient usage of energy resources.

Methodology

I) Requirement Analysis

• First, we need to pinpoint the right power sources and sensors.

• Next, let's outline the essential features: durability, comfort, energy storage, and portability.

• Finally, we'll establish performance goals for voltage, current, and energy storage capacity.

II) Project Method

• We'll start by embedding piezo discs into the insoles.

• A bridge rectifier will be used to convert AC to DC.

• The TP4056 module will ensure safe charging for the Li-ion battery.

• We'll utilize an MT3608 boost converter to achieve a 5V output, along with a buck converter to manage the high power from the battery.

• An Arduino Nano paired with a 16x2 LCD or I2C OLED will handle data display.

• We'll monitor the battery level using an analog input.

• Lastly, we'll detect voltage with a voltage sensor.

Stretch Circuit Diagram

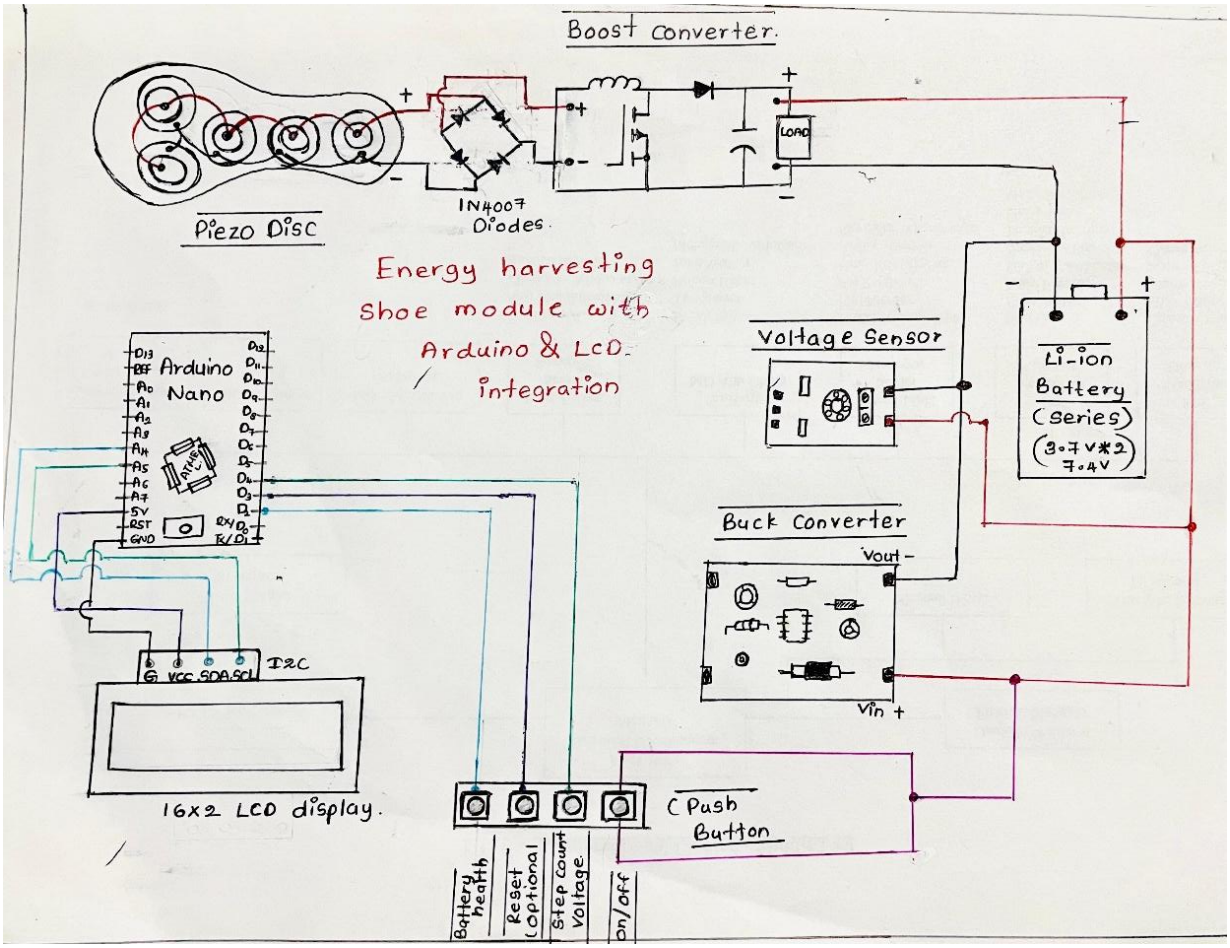


Figure 1 – stretch circuit diagram

3D Design and Completed Prototype

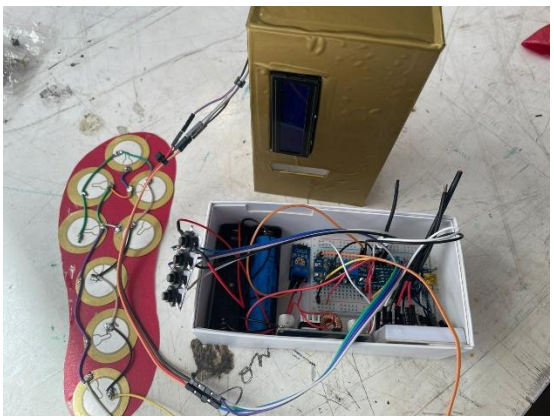


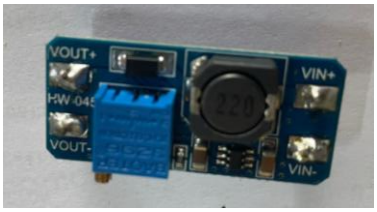
Figure 2- 3D Design of prototype

Components for Prototype

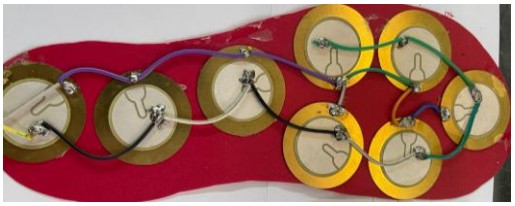
Arduino Nano - Main control unit of the prototype.

18650 Li-ion Battery – 2 batteries of 3.7v using series model and generate 7.4v output, store the electricity that generated by piezo disc.

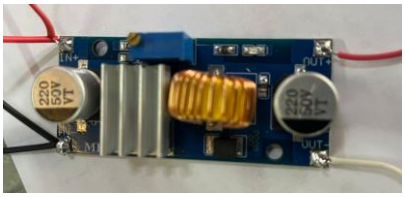
Voltage sensor – use for indicate the voltage generated by the foot steps.



Boost converter – This used to increase the generated piezo disc voltage for store the battery.

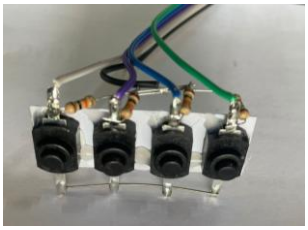


Piezoelectric Discs – Change the foot pressure in to the Electric power.

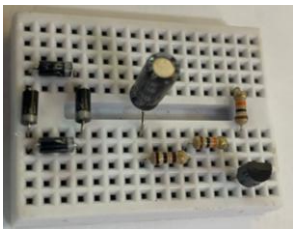


Buck converter - This converts the high voltage capacity into the low voltage.

I2C / LCD display – helps to display the outputs.



Push Buttons with 10k Om resistors



Circuit of the bread board – 1N4007 diodes / resisters / 10uf capacitors / Bc547 transistors.

Piezoelectric Energy Harvesting – Principle & Applications

- Piezoelectric sensors produce electricity when mechanical pressure or force is applied.
- Commonly used materials include PZT (Lead Zirconate Titanate), which has high efficiency in converting mechanical energy to electrical energy.
- These sensors are often used in shoe soles to generate energy from walking or running.

Integration with Arduino Nano and LCD Display

Using a microcontroller makes the system easier to use. In this project, the Arduino Nano is used to:

- Read battery voltage through analog input.
- Show real-time information like battery level and power output on an LCD.
- Control the boost converter and switching system.

Testing & Calibration

- Measure voltage and step counts at different walking speeds and conditions.
- Test different piezoelectric materials to see which works best.
- Adjust sensor placement on the shoe to produce maximum energy.

Energy harvesting shoe module video demonstration

Energy harvesting shoe module - Google Drive

Discussion

The Energy Harvesting Shoe Module has made a fascinating breakthrough by demonstrating that our everyday walking can actually be transformed into electrical energy. Early tests revealed that just by walking, we can generate enough electricity to charge small devices when it's stored in a rechargeable battery. The effectiveness of this system hinges on several factors, including how fast you walk, your weight, and the number of piezoelectric plates that are installed.

One of the hurdles faced was creating a circuit that could convert power efficiently without wasting energy, all while ensuring the shoe remains light and comfortable to wear. Looking ahead, there's plenty of room for enhancements, such as boosting energy output, increasing efficiency, and making the whole system more compact and user-friendly.

Future Implementation

- Flexible Piezoelectric Materials - Upgrade comfort and mechanical power make.
- Dual Source Harvesting - Simple piezoelectric and thermoelectric modules working in tandem to capture and transform both motion and body heat into energy.
- Mobile App Monitoring - Monitor energy generation and usage statistics.
- Self-Powered Smart Footwear for IoT Connectivity" – which used an Arduino Nano with a piezo sensor and OLED display for live monitoring and data logging.
- Low-Cost Wearable Charger using Footsteps" (IJSER, 2023) – where the LCD displayed the number of steps, voltage, and charging status.
- Efficient Charging Circuits for Piezo-based Harvesters" (Springer, 2020) – discusses methods to minimize loss in rectification and boosting stages.

References

i. Design and Implementation of a Low-Cost Piezoelectric Footstep Energy Harvesting System

• Focus: Footstep energy harvesting through piezoelectric sensors, power conditioning, and LCD monitoring.

• Source: ResearchGate, 2023.

ii. StepNergy: Piezoelectric Energy Harvesting with Arduino Uno and LCD

• Focus: Footstep energy harvesting and Arduino-based monitoring with LCD display.

• Source: ResearchGate, 2023.

iii. Smart Shoe for Elderly Tracking with Energy Harvesting

• Focus: In rescue and tracking applications, this focus is on integrating energy harvesting in shoes so that it powers their sensing devices.

• Source: EJ-Engineering, 2023.

iv. Harvesting Energy with Integrated Piezoelectric Material into Shoe

• Focus: Shoes with integrated piezoelectric materials and force amplifiers for high energy generation.

• Source: ResearchGate, 2018.

v. Arduino-based Footstep Power Generation using Piezo

• Focus: Harvesting the energy generated from shoes based on Arduino Nano control, with reading on an LCD screen.

• Source: E3S Conferences, 2024.

Gann Chart

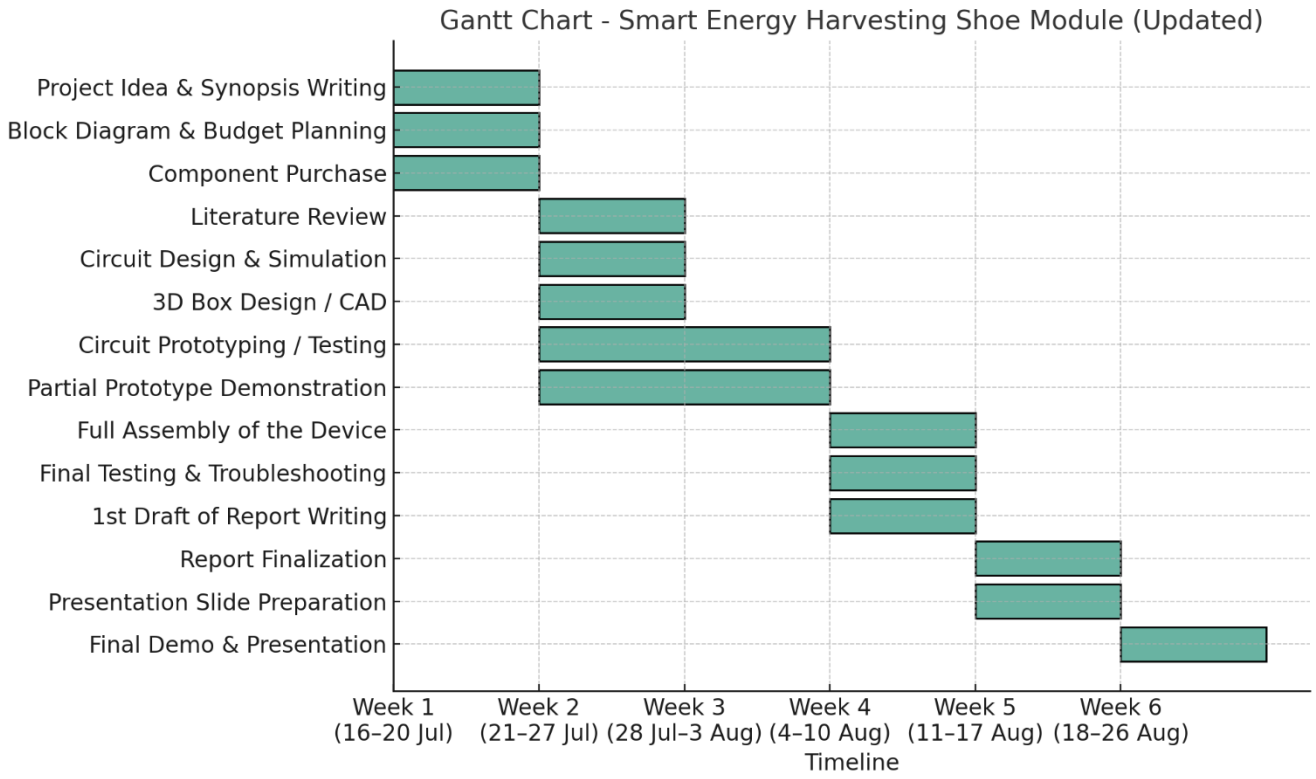


Figure 3 Gann chart

Arduino Nano CODE

(LCD display/ push buttons /voltage sensor)

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

// ---------------- Pins ----------------
const uint8_t LED_PIN        = 8;
const uint8_t BTN_HEALTH_PIN = 2;  // D2 -> Battery Health
const uint8_t BTN_MIXED_PIN  = 3;  // D3 -> Steps + V + %
const uint8_t BTN_STATS_PIN  = 4;  // D4 -> Steps + Voltage

// ---------------- LCD -----------------
LiquidCrystal_I2C lcd(0x27, 16, 2);

// ------------- ADC / Divider ----------
const float VREF      = 5.0f;       // ADC reference (UNO default)
const float DIV_RATIO = 0.040909f;  // Vout/Vin of your divider (adjust if needed)
const int   STEP_THRESHOLD = 5;     // Raw threshold to detect a step (avoid noise)

// ------------- Battery model ----------
// 12V Lead-Acid
const float BAT_FULL_V  = 12.7f;    // ~100%
const float BAT_EMPTY_V = 11.8f;    // ~0%

// ------------- App State --------------
long stepCount = 0;
```

```
bool prevStepActive = false;
unsigned long lastStepMs    = 0;    // LED timing
unsigned long lastRefreshMs = 0;    // display refresh

enum ScreenMode { SCREEN_STATS = 0, SCREEN_HEALTH = 1, SCREEN_MIXED = 2 };
ScreenMode screenMode = SCREEN_STATS;
ScreenMode lastScreen = (ScreenMode)255;

bool prevD2 = false, prevD3 = false, prevD4 = false;

// ------------ Helpers ----------------
float vinFromRawV(int raw) {
  float vout = (raw * VREF) / 1023.0f;   // ADC (0..1023)
  return vout / DIV_RATIO;               // real input voltage
}

int batteryPercent(float v) {
  float pct = (v - BAT_EMPTY_V) / (BAT_FULL_V - BAT_EMPTY_V) * 100.0f;
  if (pct < 0)   pct = 0;
  if (pct > 100) pct = 100;
  return (int)(pct + 0.5f);
}

const char* healthLabel(int pct) {
  if (pct >= 80) return "GOOD";
  if (pct >= 40) return "OK";
  return "LOW";
}
```

```
void drawStatsHeader() {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("STEP COUNT:");
  lcd.setCursor(0, 1); lcd.print("VOLTAGE:");
}

void drawHealthHeader() {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("HEALTH:");
  lcd.setCursor(0, 1); lcd.print("V:");
}

void drawMixedHeader() {
  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("STEPS & BATTERY");
}

void updateStatsValues(int raw) {
  float vinV = vinFromRawV(raw);
  lcd.setCursor(12, 0); lcd.print("    "); lcd.setCursor(12, 0); lcd.print(stepCount);
  lcd.setCursor(9,  1); lcd.print("       ");
  lcd.setCursor(9,  1); lcd.print(vinV, 2); lcd.print("V");
}

void updateHealthValues(int raw) {
  float vinV = vinFromRawV(raw);
  int pct = batteryPercent(vinV);
```

```
  lcd.setCursor(8, 0);  lcd.print("     ");
  lcd.setCursor(8, 0);  lcd.print(pct); lcd.print("% ");
  lcd.setCursor(2, 1);  lcd.print("             ");
  lcd.setCursor(2, 1);  lcd.print(vinV, 2); lcd.print("V ");
  lcd.print(healthLabel(pct));
}

void updateMixedValues(int raw) {
  float vinV = vinFromRawV(raw);
  int pct = batteryPercent(vinV);
  // Line 0: STEPS
  lcd.setCursor(0, 0); lcd.print("STEPS:        ");
  lcd.setCursor(7, 0); lcd.print("       ");
  lcd.setCursor(7, 0); lcd.print(stepCount);
  // Line 1: V and %
  lcd.setCursor(0, 1); lcd.print("V:");
  lcd.setCursor(2, 1); lcd.print("       ");
  lcd.setCursor(2, 1); lcd.print(vinV, 2); lcd.print("V ");
  lcd.setCursor(10, 1); lcd.print("B:");
  lcd.setCursor(12, 1); lcd.print("   ");
  lcd.setCursor(12, 1); lcd.print(pct); lcd.print("%");
}

// ------------- Setup ------------------
void setup() {
  Serial.begin(9600);

  pinMode(LED_PIN, OUTPUT);
```

```
  pinMode(BTN_HEALTH_PIN, INPUT_PULLUP);
  pinMode(BTN_MIXED_PIN,  INPUT_PULLUP);
  pinMode(BTN_STATS_PIN,  INPUT_PULLUP);

  lcd.begin();              // If this fails to compile, use lcd.init();
  lcd.backlight();

  // Splash
  lcd.print("FOOT STEP POWER");
  lcd.setCursor(0, 1); lcd.print("   GENERATOR");
  delay(2000);

  drawStatsHeader();
}

// ---------- Loop ----------
void loop() {
  unsigned long now = millis();
  int raw = analogRead(A0);
  bool stepActive = (raw > STEP_THRESHOLD);

  // Step detection with threshold + LED timing
  if (stepActive && !prevStepActive) {
    stepCount++;
    digitalWrite(LED_PIN, HIGH);
    lastStepMs = now;
    if (screenMode == SCREEN_STATS) { // quick refresh on stats screen
      lcd.setCursor(12, 0); lcd.print("    ");
```

```
    lcd.setCursor(12, 0); lcd.print(stepCount);
  }
}
if (now - lastStepMs >= 100) {
  digitalWrite(LED_PIN, LOW);
}
prevStepActive = stepActive;

// Read buttons (active LOW)
bool d2 = (digitalRead(BTN_HEALTH_PIN) == LOW);
bool d3 = (digitalRead(BTN_MIXED_PIN)  == LOW);
bool d4 = (digitalRead(BTN_STATS_PIN)  == LOW);

if (d2 && !prevD2) screenMode = SCREEN_HEALTH; // D2 -> Health
if (d3 && !prevD3) screenMode = SCREEN_MIXED;  // D3 -> Mixed
if (d4 && !prevD4) screenMode = SCREEN_STATS;  // D4 -> Stats

prevD2 = d2; prevD3 = d3; prevD4 = d4;

// Redraw header on screen change
if (screenMode != lastScreen) {
  if (screenMode == SCREEN_STATS)   drawStatsHeader();
  else if (screenMode == SCREEN_HEALTH) drawHealthHeader();
  else                              drawMixedHeader();
  lastScreen = screenMode;
  lastRefreshMs = 0; // force immediate update
}
```

```
  // Periodic refresh
  if (now - lastRefreshMs >= 200) {
    lastRefreshMs = now;
    if (screenMode == SCREEN_STATS)      updateStatsValues(raw);
    else if (screenMode == SCREEN_HEALTH) updateHealthValues(raw);
    else                                  updateMixedValues(raw);
  }
}
```