ARDUINO NANO CODE OF ENERGY HARVESTING SHOE MODULE

```
#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);
  }
}
```