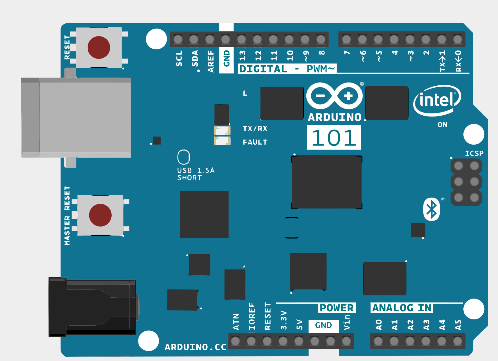
**Component List and Pin Connections for Footstep Power Generator Monitoring System**

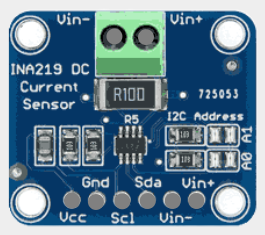
This document summarizes all essential components, their roles in the system, and how to connect them to the Arduino Uno. It ensures that the footstep power generator system operates effectively and safely while providing real-time monitoring.

**1. Arduino Uno**

****

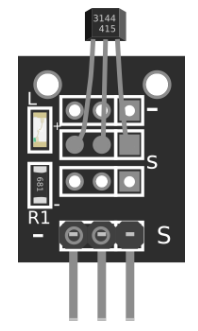
* **Purpose:** Acts as the central microcontroller to read data from sensors and control the OLED display.
* **Pins Used:** A0, A4, A5, D2, D3, D4, 5V, GND

**2. INA219 Voltage & Current Sensor**

****

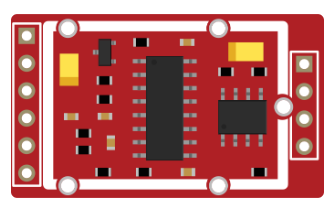
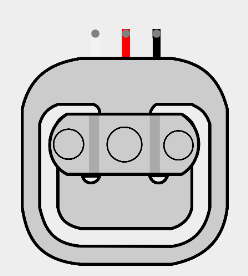
* **Purpose:** Measures DC voltage, current, and power generated by the footstep system safely up to 26V.
* **Why Used:** Provides safe and accurate measurement of both current and voltage without manual voltage dividers.
* **Connections:**
  + VCC → Arduino 5V
  + GND → Arduino GND
  + SDA → Arduino A4
  + SCL → Arduino A5
  + VIN+ → Generator output + (positive terminal)
  + VIN− → Load input + (positive terminal of the load)

**3. A3144 Hall Effect Sensor**

****

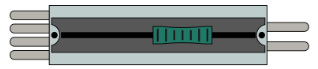
* **Purpose:** Measures the RPM (speed) of the rotating pinion gear.
* **Why Used:** Detects the presence of a magnetic field when a magnet on the rotating part passes by.
* **Connections:**
  + VCC → Arduino 5V
  + GND → Arduino GND
  + OUT → Arduino D2

**4. 4x 50kg Load Sensors with HX711 Amplifier**

****

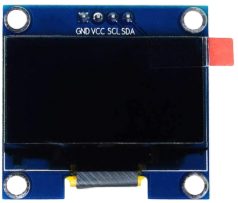
* **Purpose:** Measures the total weight applied to the footstep platform.
* **Why Used:** More durable and suitable for high weight loads compared to a single 5kg sensor. The four sensors are typically connected in a Wheatstone bridge configuration to the HX711.
* **Connections:**
  + HX711 DT → Arduino D3
  + HX711 SCK → Arduino D4
  + VCC → Arduino 5V
  + GND → Arduino GND

**5. B-103 Sliding Potentiometer (or Linear Displacement Sensor)**

****

* **Purpose:** Measures spring compression/deflection.
* **Why Used:** Converts spring movement into an analog voltage that correlates with deflection.
* **Connections:**
  + One outer pin → Arduino 5V
  + Other outer pin → Arduino GND
  + Middle pin → Arduino A0

**6. OLED Display (128x64, I2C)**

****

* **Purpose:** Displays all sensor readings including voltage, current, power, RPM, weight, and deflection.
* **Why Used:** Provides a compact and readable interface to view real-time data.
* **Connections:**
  + VCC → Arduino 5V
  + GND → Arduino GND
  + SDA → Arduino A4
  + SCL → Arduino A5

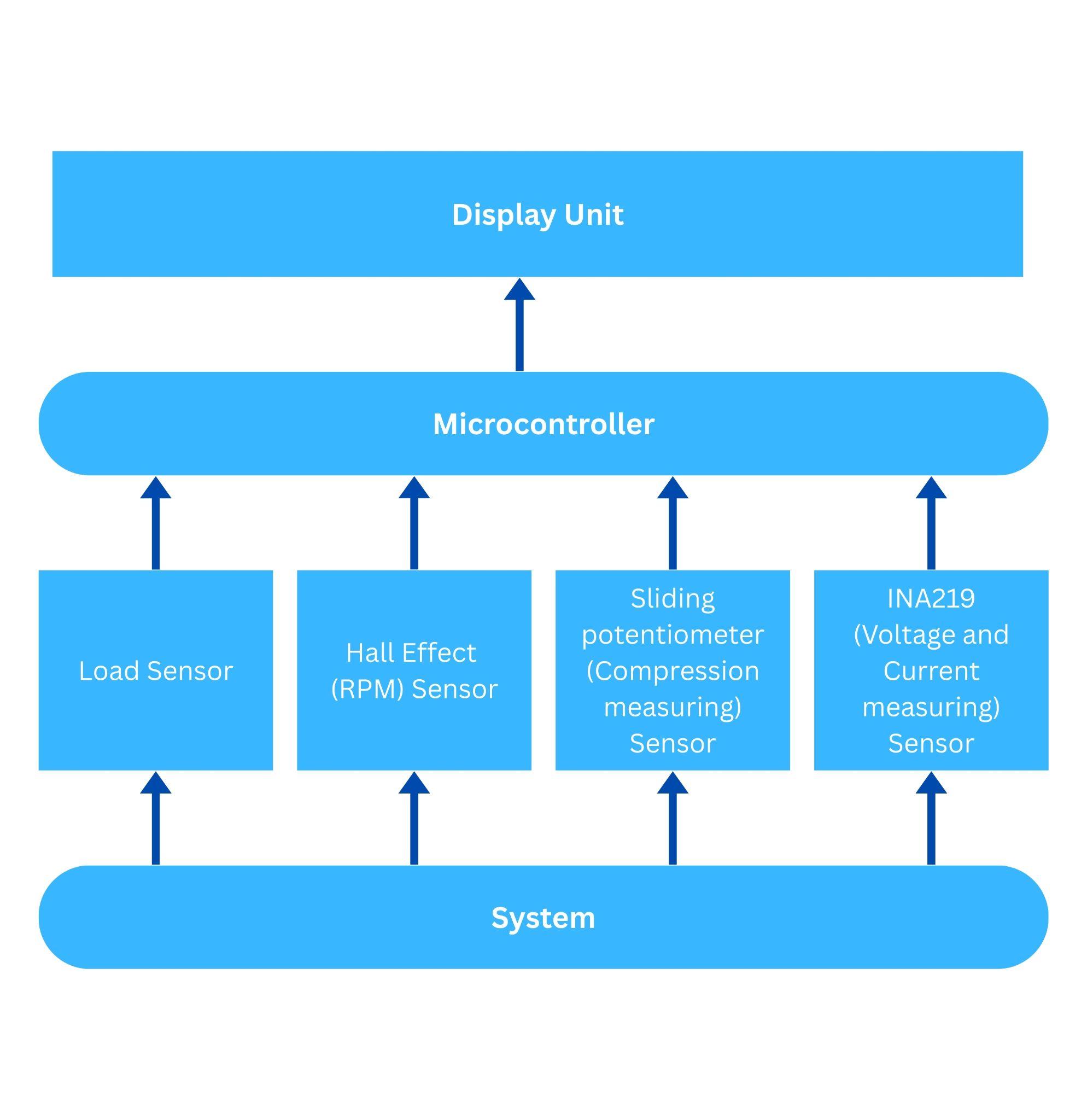
### **🔌 Summary Table: Arduino Pin Connections**

| **Arduino Pin** | **Connected Component** | **Description** |
| --- | --- | --- |
| A0 | Potentiometer (Spring sensor) | Measures deflection |
| A4 | INA219 / OLED (SDA) | I2C Data |
| A5 | INA219 / OLED (SCL) | I2C Clock |
| D2 | Hall Effect Sensor OUT | RPM detection input |
| D3 | HX711 DT | Load cell data |
| D4 | HX711 SCK | Load cell clock |
| 5V | All sensors/modules VCC | Power supply |
| GND | All sensors/modules GND | Common ground |

### **📦 Component Summary Table**

| **Component** | **Quantity** | **Purpose** |
| --- | --- | --- |
| Arduino Uno | 1 | Central controller |
| INA219 Sensor Module | 1 | Voltage, current, power sensing |
| A3144 Hall Sensor | 1 | RPM measurement |
| 50kg Load Sensors | 4 | High-capacity weight sensing |
| HX711 Amplifier | 1 | Signal amplification from load cells |
| Potentiometer | 1 | Spring deflection measurement |
| OLED Display (128x64 I2C) | 1 | Display sensor data |
| Connecting wires | - | Circuit wiring |
| Small Magnet | 1 | For Hall effect RPM detection |

**Block Diagram:**



**Arduino Code:**

// === Libraries ===

#include <Wire.h>

#include <Adafruit\_INA219.h>

#include <Adafruit\_SSD1306.h>

// === Objects ===

Adafruit\_INA219 ina219;

Adafruit\_SSD1306 display(128, 64, &Wire);

// === Pins ===

#define HALL\_PIN 2 // RPM Sensor

#define LOAD\_CELL\_DT 3 // HX711 DT

#define LOAD\_CELL\_SCK 4 // HX711 SCK

#define SPRING\_PIN A0 // Potentiometer for spring deflection

// === Libraries for other sensors ===

#include "HX711.h"

HX711 scale;

// === RPM Variables ===

volatile int rpmCount = 0;

unsigned long lastRPMTime = 0;

float rpm = 0;

void IRAM\_ATTR rpmISR() {

rpmCount++;

}

void setup() {

Serial.begin(115200);

Wire.begin();

// === INA219 Setup ===

ina219.begin();

// === OLED Setup ===

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);

display.clearDisplay();

display.setTextColor(SSD1306\_WHITE);

// === RPM Setup ===

pinMode(HALL\_PIN, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(HALL\_PIN), rpmISR, FALLING);

// === Load Cell Setup ===

scale.begin(LOAD\_CELL\_DT, LOAD\_CELL\_SCK);

scale.set\_scale(); // Adjust with calibration factor

scale.tare();

}

void loop() {

display.clearDisplay();

display.setCursor(0, 0);

// === Read INA219 ===

float voltage = ina219.getBusVoltage\_V();

float current = ina219.getCurrent\_mA() / 1000.0;

float power = ina219.getPower\_mW() / 1000.0;

display.print("V:"); display.print(voltage, 1); display.print("V ");

display.print("I:"); display.print(current, 2); display.println("A");

display.print("P:"); display.print(power, 1); display.println("W");

// === RPM Calculation ===

unsigned long currentTime = millis();

if (currentTime - lastRPMTime >= 1000) {

rpm = rpmCount \* 60.0; // assuming 1 pulse/rev

rpmCount = 0;

lastRPMTime = currentTime;

}

display.print("RPM:"); display.println(rpm);

// === Load Cell ===

float weight = scale.get\_units();

display.print("Wt:"); display.print(weight, 1); display.println("kg");

// === Spring Deflection ===

int rawDeflect = analogRead(SPRING\_PIN);

float deflection = map(rawDeflect, 0, 1023, 0, 50); // example range in mm

display.print("Defl:"); display.print(deflection); display.println("mm");

display.display();

delay(500);

}