Energy Consumption Monitoring System – Real-Time Electricity Usage Tracking

Group 04

Members:

- AGBOLOSOO, Dzidula Carl - 1842222
- ADJEI, Emmanuel Ako - 1841122
- ANSAH, Laud Appiah - 1844722
- AGYEI, Eugene - 1842422
- ANSU-AMPONSAH, Emmanuel Kwabena - 1844822
- ANYIDOHO, Daniel Mawuko - 1845222
- AGYENIM-BOATENG, Evans - 1842722
- ANTWI, Boasiako Jephthah - 1844922

Table of Contents

Problem Statement

Energy consumption monitoring is essential for understanding electrical usage, reducing waste, and managing electricity costs. This project aims to develop a real-time energy monitoring system using Arduino that can:

- Measure key electrical parameters (voltage, current, power).
- Display real-time consumption data.
- Track energy usage efficiently.
- Provide accurate power usage measurements.
- Offer insights into energy consumption patterns.

Methodology

Hardware and Simulation Tools Used:

- **Microcontroller:** Arduino Uno
- **Current Measurement:** Shunt resistor
- **Voltage Measurement:** Voltage Divider Circuit
- **Display:** 16x2 LCD (I2C) for real-time data
- **Load:** DC motor
- **Motor Control:** H-Bridge Motor Driver
- **Verification Tools:** Multimeter (for accuracy checks)
- **Simulation Platform:** Tinkercad

Measurement and Calculation Approach

The system calculates **energy consumption** using the following equations:

- **Energy (Ws) = Power × Time**
- **Power (Watts) = Voltage × Current**

Since Tinkercad lacks an ammeter that can directly measure current, an **analog pin** on the Arduino was used to **measure voltage drop across a shunt resistor** (10Ω). Using **Ohm's Law**, current is calculated as:

$$I = VR$$

$$I = \frac{\mathrm{V}}{\mathrm{R}}$$

The voltage across the motor is stepped down using a **voltage divider** to ensure the Arduino's **0V-5V measurement range** is not exceeded.

Time is tracked using Arduino's built-in millis() function.

Demonstration/Simulation

System Workflow

1. **Voltage Measurement**
    - Code:

    float voltage = analogRead(A1) * (30.0 / 1023.0);

    - A **voltage divider** scales the motor voltage to a safe range for the Arduino.
    - Formula: Vmotor=ADC reading×30.01023V_{motor} = \text{ADC reading} \times \frac{30.0}{1023}
2. **Current Measurement**
    - Code:

    float voltage1 = analogRead(A0) * (5.0 / 1023.0);

    float voltage2 = analogRead(A1) * (5.0 / 1023.0);

    float shuntVoltage = voltage2 - voltage1;

    float ledCurrent = shuntVoltage / 10.0;

    - The **shunt resistor creates a voltage drop** (V_shunt = V_A1 - V_A0).
    - Current is calculated using **Ohm's Law** (I = V_shunt / R_shunt).

3. **Power and Energy Calculation**
    - Code:
4. float ledPower = voltage * ledCurrent;
5. totalEnergy += ledPower * (elapsedTime / 1000.0);
    - Power = V×I
    - Energy is accumulated over time.

Output Workflow

- **Serial Monitor (Debugging Purpose)**
- Serial.print("Voltage: "); Serial.print(voltage, 2); Serial.println(" V");
- Serial.print("Current: "); Serial.print(ledCurrent * 1000, 2); Serial.println(" mA");
- Serial.print("Power: "); Serial.print(ledPower, 2); Serial.println(" W");
- Serial.print("Energy: "); Serial.print(totalEnergy, 2); Serial.println(" Ws");
- **LCD Display (User Interface)**
- lcd.clear();
- lcd.setCursor(0, 0);
- lcd.print("E=");
- lcd.print(totalEnergy, 2);
- lcd.print(" Ws");

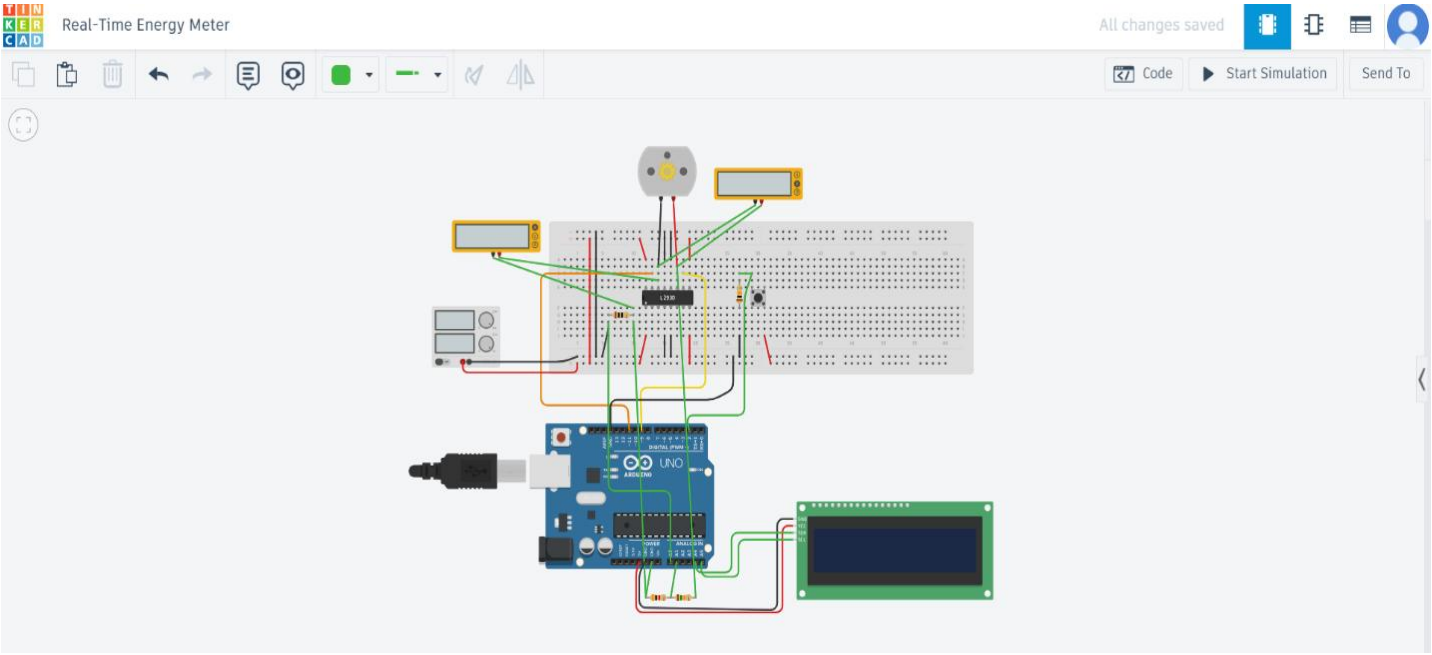
H-Bridge Motor Driver Role

- **Why an H-Bridge?**
    - Allows bidirectional motor control.
    - Provides electrical isolation, protecting the Arduino from back EMF.
    - Ensures energy calculations remain stable.
- **Motor ON/OFF Code:**
- digitalWrite(motor1, HIGH);
- digitalWrite(motor2, LOW);

Summary of System Operation

1. **Motor starts**, current flows through the **shunt resistor**.
2. **Voltage drop is measured** across shunt to **calculate current**.
3. **Motor voltage is measured** (scaled via divider on A1).
4. **Power is computed** (V × I), and energy accumulates over time.
5. **Energy is displayed** on the LCD, with details logged in the Serial Monitor.

Circuit Diagram



Conclusion

Findings

- The system successfully tracked **real-time energy consumption**.
- Power and energy calculations worked effectively within **Tinkercad's simulation limitations**.

Challenges Encountered

- **Tinkercad does not support direct current measurement**, requiring a workaround using a **shunt resistor**.
- **Measurement accuracy was limited** due to **low-resolution ADC (10-bit)**.
- **Tinkercad sometimes resets the circuit**, causing data loss.
- **Energy units were displayed in Watt-seconds (Ws)** instead of **kWh** due to small energy levels.

Suggested Improvements

- **Use a higher-precision sensor** (e.g., INA219) for **more accurate current measurements**.
- **Integrate Wi-Fi/Bluetooth** (e.g., ESP32) for **remote energy monitoring**.
- **Implement data logging** using an **SD card or cloud storage**