Monitoring Energy Consumption using Arduino Microcontroller

Kasandikromo Guillian, Karg Timothy, and Uraicia Sewpersad

Vanguard Community College - Department of Engineering

November 25, 2024

Abstract

This project explains how to set up an Arduino-based system to measure and monitor energy use, making the data accessible in real-time through cloud storage and a web app. As energy optimization becomes more important, this setup provides accurate and easily accessible energy data to help users make informed decisions.

The system uses an Arduino microcontroller with a YHDC Current Transformer SCT-013-000 sensor to measure current. The Arduino converts this data into energy consumption readings in kilowatt-hours (kWh) after first calculating power (watts) and then energy (watt-seconds). Every 10 seconds, this data is sent to a cloud-based PostgreSQL database via a Python script, which connects to the Arduino's serial output.

To make the data easy to view and analyze, a web app built with Flask and Dash lets users filter data by month or day to explore detailed energy usage patterns. The app also updates automatically, so the latest measurements are always available. With data stored in the cloud, users can monitor energy use from anywhere, helping them identify opportunities to save energy.

Overall, this system combines Arduino's compatibility with various sensors and the cloud's accessibility, creating a reliable and scalable solution for tracking energy usage in real time.

I. Introduction

This project aims to develop and deploy an Energy Monitor system, designed to provide real-time energy consumption data for a residential setting. The core component of the system is an Arduino-based microcontroller, responsible for collecting and processing energy consumption data from various sensors. This data is then transmitted to a web server, allowing for remote monitoring and analysis.

The web application, built using Python and Flask, provides a user-friendly interface to visualize energy consumption patterns, identify potential areas for energy savings, and generate insightful reports. This project leverages the power of open-source technologies to create a cost-effective and customizable solution for energy monitoring.

Key Features:

- Real-time monitoring: The system provides up-to-date energy consumption data.
- Remote access: Users can access and analyze data from anywhere with an internet connection.
- Data visualization: The web application offers various visualization tools for easy data interpretation.
- Energy efficiency insights: The system helps identify areas for energy savings and optimization.

This document will detail the system's architecture, hardware implementation, software development, and deployment process. It will also include a comprehensive evaluation of the system's performance and potential future enhancements.

II. Materials

This setup for measuring energy consumption includes an Arduino microcontroller and a Current Transformer (CT) sensor. The CT sensor detects the alternating current (AC) flowing through a conductor without direct electrical contact. It outputs a proportional current signal to the Arduino, which processes the data to calculate energy consumption. To improve accuracy, the system may include resistors, capacitors for filtering, and a voltage divider circuit for safe scaling, allowing the Arduino to measure and record energy data reliably for applications like household or small industrial monitoring.

i. Arduino Microcontroller UNO

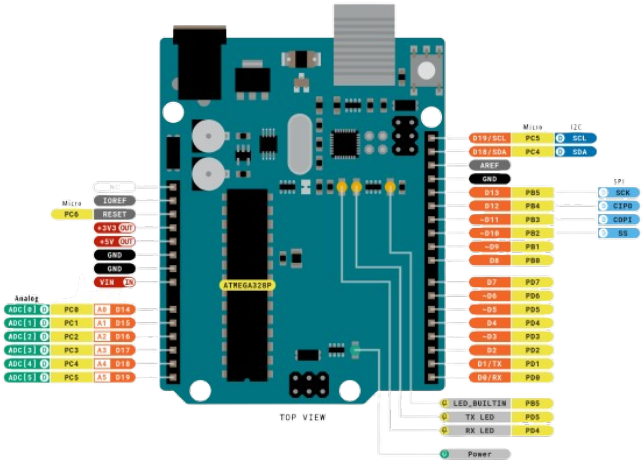


Figure 1: Arduino Uno Microcontroller

The Arduino UNO is the core microcontroller for this energy monitoring setup, handling data from various sensors and executing calculations for accurate energy measurements. It operates on a 5V logic level, suitable for reading data from components like the CT sensor (current transformer) and voltage divider circuits. Potential additional components include resistors to limit current, capacitors to filter noise, and an external analog-to-digital converter if higher accuracy is needed. [1]

ii. YHDC Current Transformer SCT-013-000

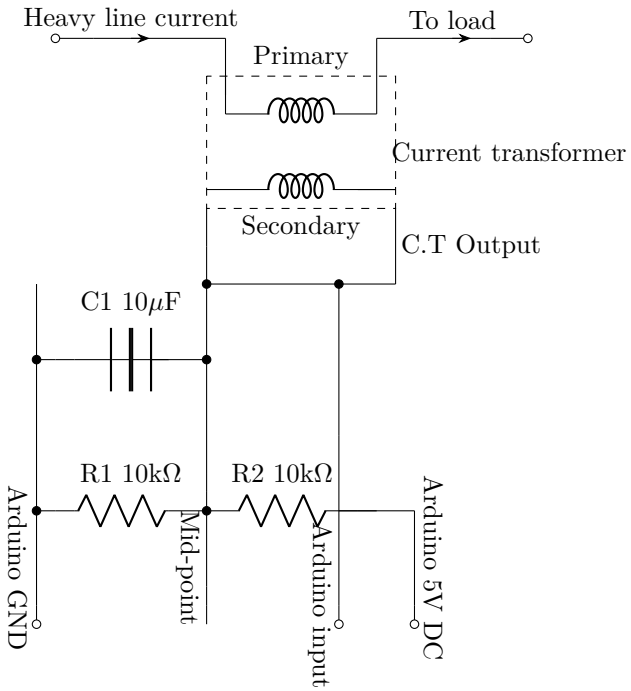


Figure 2: Current transformer circuit with Arduino

The YHDC SCT-013-000 is a non-invasive current transformer sensor used to measure AC current without direct contact with the high-voltage line. This CT sensor detects current based on electromagnetic induction and outputs a small, proportional AC signal. When connected to the Arduino, it allows accurate monitoring of the current flowing through the conductor. With the help of filtering capacitors and resistors for scaling, the output signal from the CT sensor is adapted to the Arduino's input range, enabling precise energy calculations. [1]

iii. Liquid Crystal Display

- Display: It displays information like voltage, current, power consumption, and other relevant data. [2]
- Backlight: It provides illumination for better visibility. [2]
- Interface: It connects to the Arduino board through digital pins to receive and display data.
- Types: There are various types of LCDs, including character LCDs and graphic LCDs.

iv. Potential Components:

- Voltage Sensor: To measure voltage, we initially tried the Open Energy Monitor (Emon-

Lib) approach, which utilizes an AC-AC adapter to capture the alternating voltage signal. However, due to issues with stability, we instead used a fixed AC voltage source of 127V for this setup. This consistent voltage ensures a reliable reference for calculating power and energy consumption.

- Temperature Sensor: Monitors the temperature of the system or environment.
- Power Supply: Provides the necessary power for the Arduino and other components.
- LCD (Liquid Crystal Display):

v. Voltage Divider

The voltage divider circuit is designed to scale down the high input voltage of 127V AC to a safe range of 0-5V DC for Arduino's analog input. It consists of two resistors (R_1 and R_2) that divide the voltage proportionally based on their resistance values. A capacitor (C_1) is added to filter noise, and a coupling capacitor (C_2) biases the signal within the 0-5V range. This enables the Arduino to read the AC signal safely without risking damage from high voltage. [3]

The output voltage, V_{out}, can be calculated as:

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

where V_{in} is the input AC voltage (127V). By selecting appropriate resistor values, V_{out} can be scaled to fall within the Arduino's input limits.

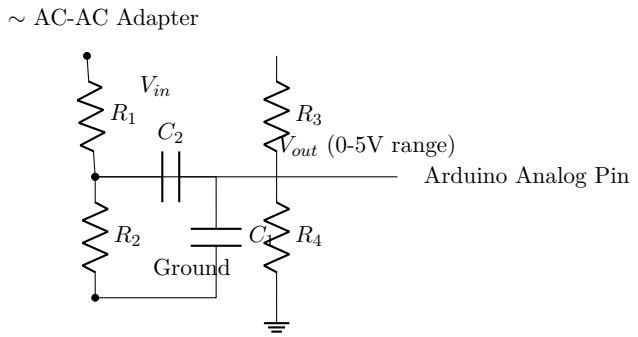
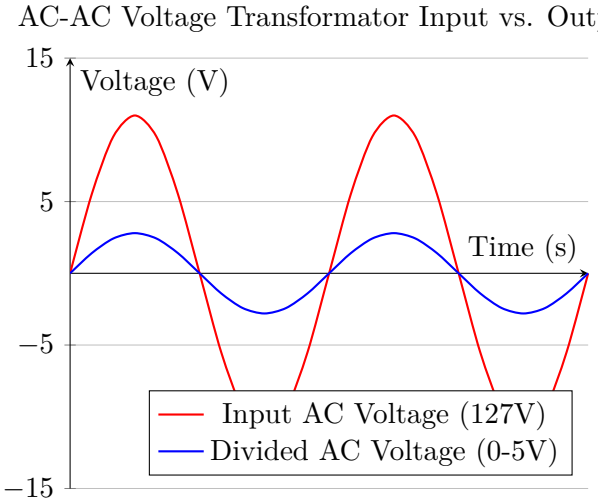


Figure 3: Voltage Divider with Filtering and Biasing for Arduino Analog Input

The circuit is designed to measure AC voltage using an Arduino, an AC-AC power adapter, and a voltage divider. The AC-AC adapter supplies the input voltage (V_{in}) to the circuit. The voltage divider, made of resistors R_1 and R_2, steps down the AC voltage to a safe level for the Arduino's 0-5V input range. The resistors are selected to ensure the voltage stays within the Arduino's limits.

To allow the Arduino to read both positive and negative portions of the AC waveform, the signal is biased to a 2.5V reference using resistors R_3 and R_4. Capacitor C_1 filters high-frequency noise, ensuring clean signal measurement. A coupling capacitor (C_2) isolates the DC bias while passing the AC signal to the Arduino's analog input, where it is digitized for further analysis. This setup enables safe, accurate measurement of AC voltage for energy monitoring. [3]



vi. AC-AC Voltage Transformator

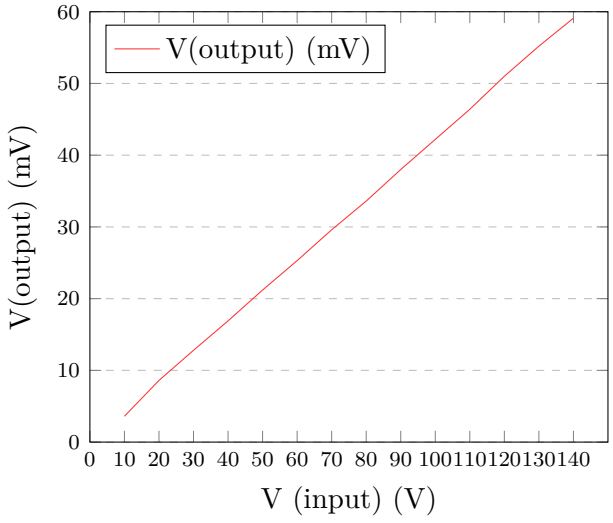
The table and graph display measurements from an AC-AC voltage transformer used to step down an input AC voltage to a lower output range. The input voltage values (in volts) increase incrementally from 10V to 140V, and the corresponding output voltage (in millivolts) is recorded in each case. The current drawn at each input voltage is also measured, showing a slight increase with higher input voltages.

Table 1: AC-AC Voltage Transformator Data

V (input) (V)	I (mA)	V(output) (mV)
10	0.59	3.6
20	1.15	8.6
30	1.7	12.8
40	2.25	16.9
50	2.8	21.2
60	3.4	25.3
70	3.96	29.6
80	4.52	33.6
90	5	38
100	5.67	42.2
110	6.21	46.4
120	6.77	51
130	7.38	55.2
140	7.9	59.1

In the plotted graph, there is a near-linear relationship between the input voltage and the output voltage, which aligns with the expected behavior of a linear voltage transformation. This data is useful for calibrating and verifying the transformer's performance, ensuring the output voltage remains within a safe and measurable range for connected circuits, such as an Arduino

AC-AC Voltage Transformator Input vs. Output



III. System Design

The energy monitoring system consists of three main components: the Arduino microcontroller, the YHDC Current Transformer (CT) SCT-013-000 sensor, and a cloud-based PostgreSQL database accessible through a web application. Each component plays a vital role in capturing, processing, storing, and displaying energy consumption data in real-time.Since we've covered the hardware components in Section II, here we'll go over the software components used.

i. Software Components

The software components of the system include:

- **Data Processing on Arduino:** The Arduino processes signals from the CT sensor and calculates power by combining current and voltage values. The calculated energy data is sent to a local SQLite database, which is later synced with a cloud database.

- **Python Data Transfer Script:** A Python script interfaces with the Arduino's serial monitor and transfers the processed data to a cloud-based PostgreSQL database at intervals of 10 seconds.

- **Flask Web Application:** A Flask-based application with Dash visualizes the data, allowing users to filter by month and day to observe consumption trends. This user interface provides an accessible and flexible way to monitor energy usage over time.

ii. Data Flow and Operation

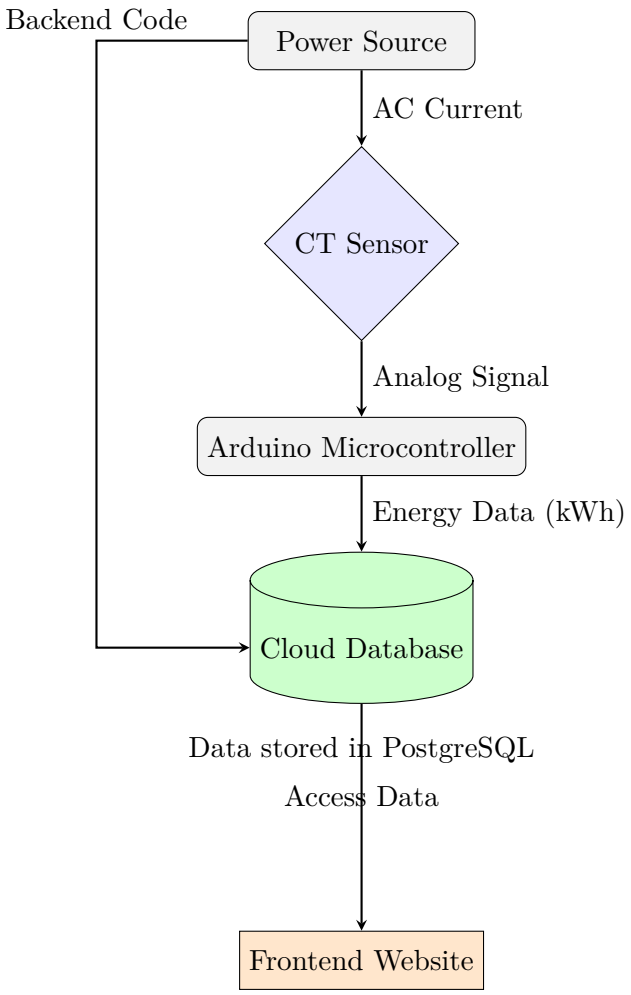


Figure 4: Process Flow Diagram of the Energy Monitoring System with Frontend Website for Data Viewing

The system operates as follows:

1. The Arduino reads analog signals from the CT sensor and the AC-to-AC adapter.

2. It processes these signals, calculates energy consumption in watt-seconds (later converted to kWh), and sends the data to the Python script.

3. The Python script logs the data in a local SQLite database, which then syncs to a PostgreSQL cloud database every 10 seconds.

4. The web app fetches the latest data from the cloud database, displaying it with interactive filters for user analysis.

This modular design, combining hardware and software elements, enables real-time, remote monitoring of energy usage.

IV. Energy Monitoring System Calculations

The energy monitoring system performs several key calculations to track and measure real-time energy consumption. Below is a detailed explanation of the important calculations implemented in the code for monitoring the energy usage.

i. Root Mean Square (RMS) Current Calculation

The energy monitoring system uses the Root Mean Square (RMS) current to measure the average current flowing through the load. The RMS current (I_{rms}) is calculated by the 'emon1.calcIrms()' function, which computes the RMS value of the current over a specified number of half-wavelengths (crossings) of the AC signal. This is essential for determining the power consumed by the load, as RMS values better represent the continuous power usage over time. The formula for RMS current is:

$$I_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} I_i^2}$$

where I_i represents the instantaneous current values over N half-wavelengths. The number of sample points $N = 1480$ is used in the code to calculate the RMS current, ensuring that the power consumption is accurately measured over the desired time period. [4]

ii. Power Calculation

Once the RMS current is obtained, the power consumed by the load is calculated using the following formula:

$$P = I_{rms} \times V_{rms}$$

where I_{rms} is the RMS current in amperes and V_{rms} is the RMS voltage, assumed to be $127\,V$ in the system. The power calculation is straightforward, as it directly multiplies the RMS current by the RMS voltage to obtain the instantaneous power in watts. The instantaneous power can be expressed as:

$$P = I_{rms} \times 127\,V$$

The computed power helps in determining the energy consumed over time by the load, which is the basis for further energy calculations.

iii. Energy Calculation

Energy consumption is calculated by multiplying the power by the time interval during which the measurement is taken. The energy in watt-seconds is computed as:

$$E = P \times \Delta t$$

where P is the instantaneous power (in watts) and $\Delta t = 1\,s$ is the time interval in seconds. The time interval is fixed at 1 second, corresponding to the frequency of measurements. This means that for every second, the energy consumed is accumulated to provide an accurate measure of energy usage in watt-seconds. The system performs this calculation for each measurement cycle, which is then added to the total energy consumed.

iv. Total Energy Accumulation

The total energy consumed over time is obtained by summing the energy measured during each time interval. The total energy is accumulated by adding the energy for each new measurement cycle:

$$E_{\text{total}} = E_1 + E_2 + \cdots + E_n$$

where E_n represents the energy calculated during the n-th interval. This value is continuously updated in the code and stored in the variable 'totalEnergy', which represents the total energy consumed since the system started operating.

v. Energy Conversion to kWh

Once the total energy is calculated in watt-seconds, it is converted into kilowatt-hours (kWh) for easier interpretation. Since 1 kilowatt-hour is equivalent to 3.6 million watt-seconds, the conversion is performed as follows:

$$1\,\text{kWh} = 3.6 \times 10^6\,\text{watt-seconds}$$

Thus, the total energy in kWh is given by:

$$kWh = \frac{E_{\text{total}}}{3.6 \times 10^6}$$

This conversion is critical for understanding energy usage on a larger scale, as kilowatt-hours are commonly used in electricity billing and consumption analysis.

vi. Time Simulation and Adjustment

The system simulates real-time data by incrementing the time in seconds by 10 for each measurement cycle. This time simulation ensures that the system continuously tracks energy usage in a time-ordered fashion. The time is adjusted for overflow, so that:

$$\text{seconds} \leftarrow \text{seconds} + 10$$

In the case that the seconds exceed 60, they are reset to 0, and the minutes are incremented. Similarly, if the minutes exceed 60, they are reset to 0, and the hours are incremented. If the hours exceed 24, the day counter is incremented, and the system adjusts for month and year changes. This simulation ensures that the system tracks time in a realistic manner, providing accurate data for each measurement.

vii. CSV Output Format

To make the data easily accessible for analysis, the system outputs the measured values in a CSV format. The format consists of the date, RMS current, energy usage in watt-seconds, and the converted energy in kilowatt-hours:

Output format: Date, Irms, Energy, kWh

This format is suitable for parsing by external software, such as a Python script, and allows for easy storage and further analysis of the energy consumption data. The output format makes it simple to visualize trends and identify areas where energy efficiency improvements can be made.

V. Energy Monitoring System Overview

This Arduino-based system uses an Arduino microcontroller and a YHDC Current Transformer SCT-013-000 sensor to monitor household energy consumption in real time. Power and energy calculations are performed on the Arduino, which converts data into kilowatt-hours (kWh) and sends it every 10 seconds to a cloud-based PostgreSQL database. A web app, built with Flask and Dash, enables users to view and analyze this data remotely.

Key features include:

- **Real-time monitoring:** Continuous data updates on energy use.
- **Remote access:** Data stored in the cloud allows for monitoring from any location.

- **Data visualization:** A user-friendly web interface displays insights and reports.
- **Energy efficiency insights:** Identifies potential areas for savings.

With a modular design that combines hardware and software, this system is scalable for future enhancements, providing a valuable tool for energy tracking and optimization.

For futher infromation and sourcew code of the project, click here for Github repository.

References

[1] OpenEnergyMonitor, "How to build an arduino energy monitor," accessed: 2024-11-15. [Online]. Available: https://docs.openenergymonitor.org/electricity-monitoring/ctac/how-to-build-an-arduino-energy-monitor.html

[2] TechTarget. (2024) Lcd (liquid crystal display). Accessed: 2024-11-15. [Online]. Available: https://www.techtarget.com/whatis/definition/LCD-liquid-crystal-display

[3] OpenEnergyMonitor, "Measuring voltage with an ac-ac power adapter," accessed: 2024-11-15. [Online]. Available: https://docs.openenergymonitor.org/electricity-monitoring/voltage-sensing/measuring-voltage-with-an-acac-power-adapter.html

[4] ——, "Emonlib calibration theory," accessed: 2024-11-15. [Online]. Available: https://docs.openenergymonitor.org/electricity-monitoring/ctac/emonlib-calibration-theory.html