# Smart Power Tracking System.

---

## Abstract

The Smart Energy Monitoring System is a project that uses modern hardware and software technologies to measure and monitor electrical parameters in real time. To provide precise convenient energy monitoring, this project includes a NodeMCU ESP8266 microcontroller, a Peacefair PZEM-004T energy meter, and a 20x4 LCD with I2C connection. This article summarizes the project's design, delivery, and results focusing on its benefits, difficulties, and chances.

## Introduction

Due to rising energy costs, people are finding ways to monitor their energy consumption to make energy saving measures for their home. The goal for this project is to make a DIY power meter using the PZEM-004T v3 to monitor your power consumption, and since IoT is the new norm for remote monitoring, we can also interface the power meter with an IoT dashboard through Wi-Fi connection using the Wemos D1 mini microcontroller to visualize the meter readings online where you can access it using your computer or smartphone.

The Smart Energy Monitoring System is designed to measure and display various electrical parameters such as voltage, current, power, energy, frequency, and power factor in real-time. This project integrates the Peacefair PZEM-004T energy meter, a NodeMCU ESP8266 microcontroller, and a 20x4 LCD with I2C communication to provide a compact, efficient, and user-friendly solution for energy monitoring.

## Objective

The primary goal of this project is to develop a low-cost, real-time energy monitoring system that can:

- Accurately measure electrical parameters including voltage, current, power, energy, frequency, and power factor.
- Display the data in a clear and organized manner on an LCD.
- Provide scalability for integration with IoT platforms for remote monitoring.

## Scope

The scope of this project includes the following:

- Hardware assembly and connection of components.
- Software development for data acquisition, processing, and display.
- Testing and validation of the system in various scenarios.

## Components Used

1. **Peacefair PZEM-004T Energy Meter**: Measures electrical parameters such as voltage, current, power, energy, frequency, and power factor.
2. **NodeMCU ESP8266**: Acts as the microcontroller for processing data from the PZEM-004T and controlling the display.
3. **20x4 LCD with I2C Module**: Displays the measured parameters in an organized format.
4. **Miscellaneous Components**: Connecting wires, breadboard, and power supply.

## System Design

### Hardware Architecture

1. **Peacefair PZEM-004T Energy Meter**:
    - Functions: Measures voltage, current, power, energy, frequency, and power factor.
    - Communication: Serial communication with NodeMCU.
2. **NodeMCU ESP8266**:
    - Role: Processes data from the PZEM-004T and controls the LCD display.
    - Features: Built-in Wi-Fi for potential IoT applications.
3. **20x4 LCD with I2C Module**:
    - Purpose: Displays measured electrical parameters in a user-friendly format.
    - Communication: I2C protocol for efficient data transfer.
4. **Power Supply**:
    - Supplies 3.3V to NodeMCU and LCD, and 5V to the PZEM-004T.
5. **Miscellaneous Components**:
    - Includes connecting wires, resistors, and a breadboard for prototyping.

### Software Architecture

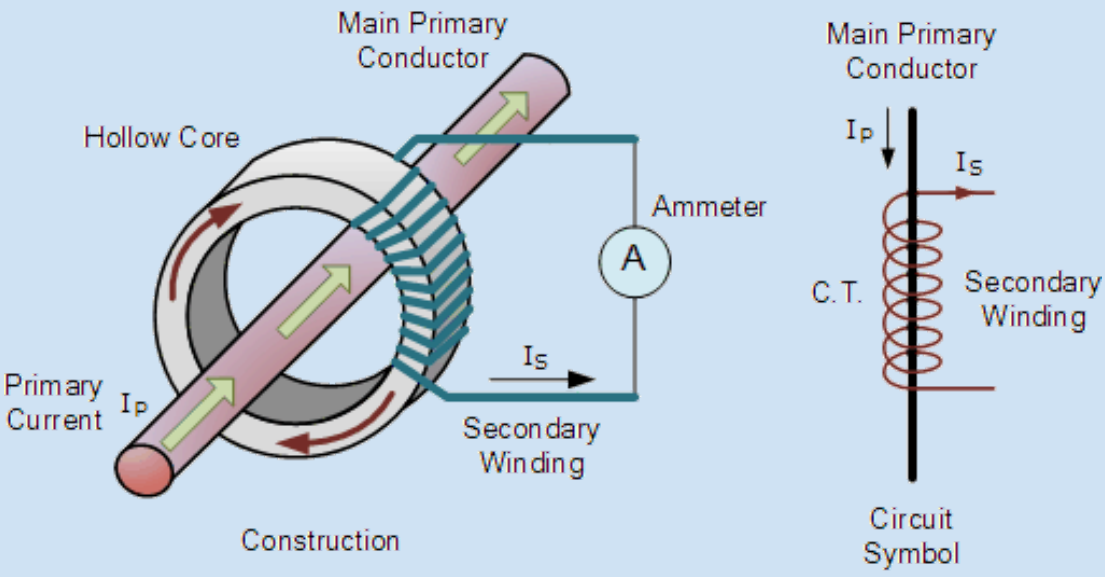
The software consists of several modules:

- **Initialization Module**: Sets up hardware and software components.
- **Data Acquisition Module**: Reads electrical parameters from the PZEM-004T.
- **Data Processing Module**: Formats the acquired data for display.
- **Display Module**: Outputs data to the LCD.

## Working Principle

1. The **PZEM-004T** energy meter continuously measures electrical parameters.
2. The data is transmitted to the **NodeMCU ESP8266** via serial communication.
3. The NodeMCU processes the received data and displays it on the **20x4 LCD** using I2C communication.
4. If any parameter is unavailable, an error message is displayed on the LCD.

**The meter uses a current transformer to measure current. It is designed to produce an alternating current in its secondary winding which is proportional to the current being measured in its primary. It reduces high voltage currents to a much lower value and provides a convenient way of safely monitoring the actual electrical current flowing in an AC transmission line.**




## Implementation

Hardware Setup

- Connect the PZEM-004T TX and RX pins to the RX and TX pins of the NodeMCU.
- Connect the I2C SDA and SCL pins of the LCD to the D2 and D1 pins of the NodeMCU.
- Provide appropriate power supply connections to all components.

Software Development

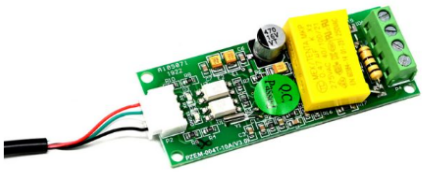
- Use the Arduino IDE for coding.
- Install required libraries:
  - PZEM00 4Tv30
  - LiquidCrystal_I2C

## Overview of the Peacefair PZEM-004T

The Peacefair PZEM-004T is an energy meter module used to measure:

- Voltage (V)
- Current (A)
- Power (W)
- Energy (kWh)

It uses a combination of sensors and circuits to perform these measurements, with the current transformer (CT) playing a key role in measuring current.



**Peacefair PZEM004Tv3**

---

## What is a Current Transformer (CT)?

The current coil in the PZEM-004T is a type of current transformer (CT). A CT is a device used to measure alternating current (AC) in a conductor without direct electrical contact. It works on the principle of electromagnetic induction.

---

## How the Current Coil (CT) Works

1. Electromagnetic Induction:
    - The current-carrying wire (e.g., a ground wire of an extension box) is passed through the CT.
    - The AC current flowing through the wire generates a magnetic field around it.
    - This varying magnetic field induces a proportional current in the coil of the CT (based on Faraday's Law of Electromagnetic Induction).
2. Step-Down Current Measurement:
    - The current transformer steps down the high current flowing through the conductor to a smaller, proportional current that can be safely measured by the PZEM-004T.
    - For example, if the main wire carries 10A, the CT might output 0.01A (depending on the turns ratio of the transformer).
3. Sensing and Conversion:
    - The induced current in the CT is fed into the internal circuitry of the PZEM-004T.
    - Inside the module, the current is converted to a voltage signal using a shunt resistor or other circuitry.
    - This voltage signal is then processed by the ADC (Analog-to-Digital Converter) of the microcontroller in the PZEM-004T.
4. Data Processing:
    - The microcontroller calculates the actual current value using the known turns ratio of the CT and the output voltage.
    - By combining this current measurement with voltage readings (measured separately), the module calculates power and energy.

## Key Components of the Current Measurement Process

1. Primary Conductor:
    - The wire carrying the AC current (your ground wire in the extension box) is the primary conductor.
2. CT Core:
    - The core of the CT is usually a toroidal (donut-shaped) magnetic core.
    - It enhances the magnetic flux and ensures accurate induction of current in the secondary coil.
3. Secondary Coil:
    - A coil of fine wire wound around the CT core. The number of turns in this coil determines the turns ratio.
4. Burden Resistor:
    - A resistor inside the PZEM-004T across which the secondary current flows.
    - The voltage across this resistor is proportional to the primary current and is used for measurement.

Let's dive into the details of how the **Peacefair PZEM-004T** calculates power, energy, and **power factor** and how these concepts are interrelated.

---

## Understanding Power Factor (PF)

The **power factor** (PF) is a measure of how effectively electrical power is being used by a device. It is defined as the ratio of **real power** (PPP) to **apparent power** (SSS).

Power Factor (PF) = Real Power (P)/Apparent Power (S)

- **Real Power (PPP)**: The actual power consumed by the device to perform useful work, measured in watts (W).
- **Apparent Power (SSS)**: The product of the RMS voltage and RMS current, measured in volt-amperes (VA).
- **Reactive Power (QQQ)**: Power that oscillates back and forth between the source and reactive components like inductors and capacitors, measured in VAR (volt-ampere reactive). This does not contribute to useful work.

In an ideal system with no reactive components, PF=1PF = 1PF=1. However, in real systems, PFPFPF is usually less than 1 due to inductive or capacitive loads.

---

## How PZEM-004T Measures Power Factor

The PZEM-004T measures the power factor by analyzing the phase relationship between the **voltage** and **current** waveforms.

**Step-by-Step Calculation:**

1. **Voltage and Current Measurement**:
    - The module samples the **voltage** and **current** waveforms in real-time using its internal ADCs.
    - The voltage and current signals are digitized for processing.
2. **Phase Difference Calculation**:
    - For AC signals, voltage and current may not be in phase due to reactive loads.
    - The PZEM-004T calculates the **phase angle (ϕ\phiϕ)** between the voltage and current waveforms.
3. **Real Power Calculation (PPP)**:
    - Real power is calculated as: P=Vrms⋅Irms⋅cos(ϕ)P = V_{rms} \cdot I_{rms} \cdot \cos(\phi)P=Vrms⋅Irms⋅cos(ϕ)
    - VrmsV_{rms}Vrms: Root-mean-square voltage
    - IrmsI_{rms}Irms: Root-mean-square current
    - cos(ϕ)\cos(\phi)cos(ϕ): Cosine of the phase angle, representing the power factor.
4. **Apparent Power Calculation (SSS)**:
    - Apparent power is calculated as: S=Vrms⋅IrmsS = V_{rms} \cdot I_{rms}S=Vrms⋅Irms
5. **Power Factor Calculation**:
    - Using the formula: PF=PSPF = \frac{P}{S}PF=SP
    - Since PPP and SSS are calculated internally, the module derives PFPFPF.

---

## Other Calculations by PZEM-004T

1. **Voltage Measurement**:
    - Measures voltage using a potential divider circuit and ADC.
2. **Current Measurement**:
    - Measures current via the CT (current transformer) and processes it through an ADC.
3. **Power Measurement**:
    - Power is calculated as: P=V⋅I⋅cos(ϕ)P = V \cdot I \cdot \cos(\phi)P=V⋅I⋅cos(ϕ)
4. **Energy Measurement**:
    - Energy is the integral of power over time: Energy (kWh)=∫P dt\text{Energy (kWh)} = \int P \, dtEnergy (kWh)=∫Pdt
    - The PZEM-004T sums the power readings over time to compute total energy consumed.

## Why is Power Factor Important?

1. **Efficiency**:
    - A low power factor indicates that a lot of power is wasted in the form of reactive power.
    - Power companies may charge penalties for industrial users with low power factors.
2. **System Design**:

- Power factor affects the sizing of wires, transformers, and generators. A low power factor leads to higher apparent power, requiring larger equipment.

---

### Practical Notes on PZEM-004T

- **Accuracy**:
  - The power factor calculation depends on the precision of the phase angle measurement. Noise or harmonics in the AC line can affect accuracy.
- **Real-Time Processing**:
  - The microcontroller in the PZEM-004T processes these calculations in real-time, providing continuous updates.

## Advantages of Using a CT in the PZEM-004T

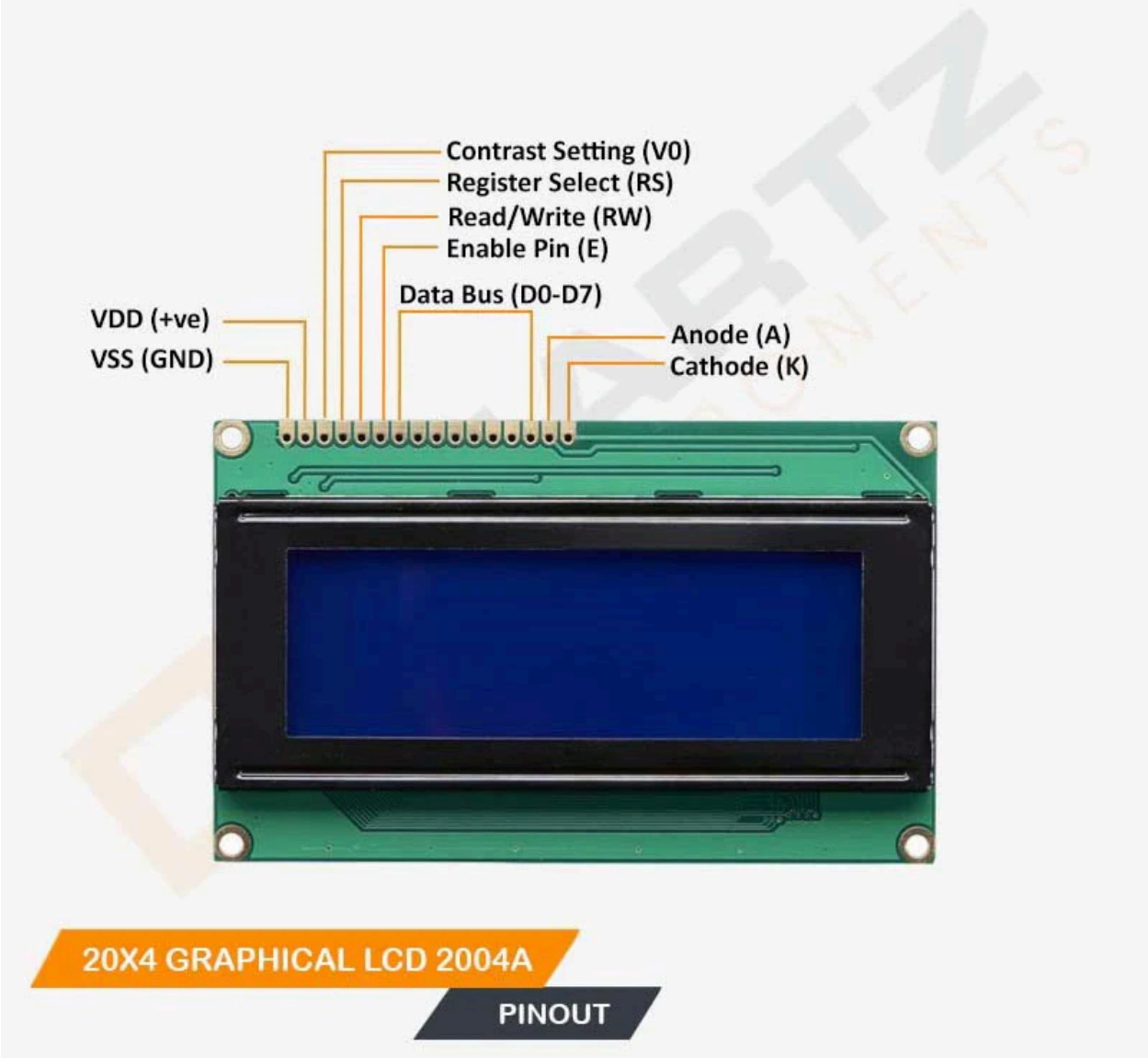
1. Non-Intrusive Measurement:
   - The CT does not need to make physical contact with the live wire. You only pass the wire through the CT core.
2. Electrical Isolation:
   - The CT provides complete electrical isolation between the primary circuit (high voltage) and the PZEM-004T module.
3. Wide Range:
   - The CT can measure a wide range of currents without the need for direct connections or bulky hardware.

## Practical Notes for Using the Current Coil

1. Direction Matters:
   - Ensure the wire is passed through the CT in the correct direction. Some CTs have a label indicating the correct orientation.
2. Single Wire Only:
   - Only one wire (live or neutral) should pass through the CT. Passing both wires will cancel the magnetic field, and the current measurement will be zero.
3. Accuracy:
   - The accuracy of the PZEM-004T depends on the quality of the CT and how well it is calibrated.

## 20X4 LCD interface using I2C :

Now, with only 3 pins from the microcontroller, you can display messages on this LCD. Compared to a parallel LCD which required at least 6 pins of I/O, this LCD offers a more cost effective solution. The LCD display is four lines by 20 characters and provides basic text wrapping so that your text looks right on the display.




The typical pinout for a 20x4 LCD is:

**VSS (Ground):** The pin connects to the ground (0V) of the supply.

**VDD (Power Supply):** Pin used to provide +5v power supply to the module.

**VO (Contrast Control):** Connect to a potentiometer to control the contrast of the display. Adjusting the voltage at this pin will change the contrast.

**RS (Register Select):** Used to select between data (RS=1) and command (RS=0) modes.

**RW (Read/Write):** Typically connected to ground (RW=0) to set the LCD in write mode.

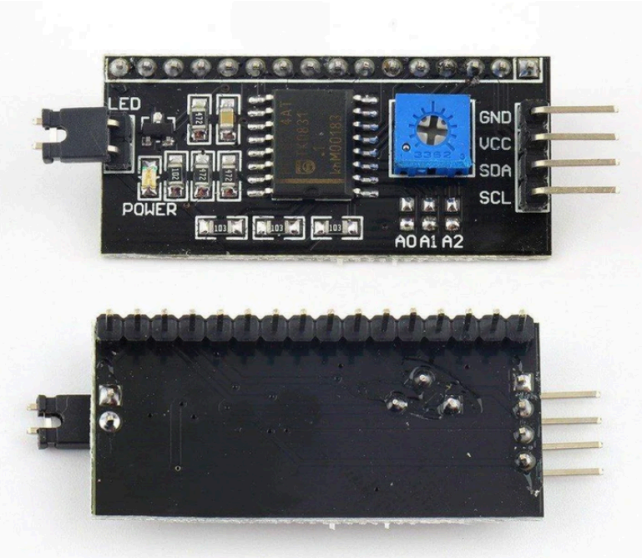
**E (Enable):** Enables the LCD to latch data present on the data bus when transitioning from high to low.

**D0-D7 (Data Bus):** These are the data lines. You can use either 4-bit mode (D4-D7) or 8-bit mode (D0-D7) depending on your setup. In 4-bit mode, you only need to connect D4-D7.

**LED+ (Backlight Anode):** Connect to the positive side of the backlight LED or resistor for current control.

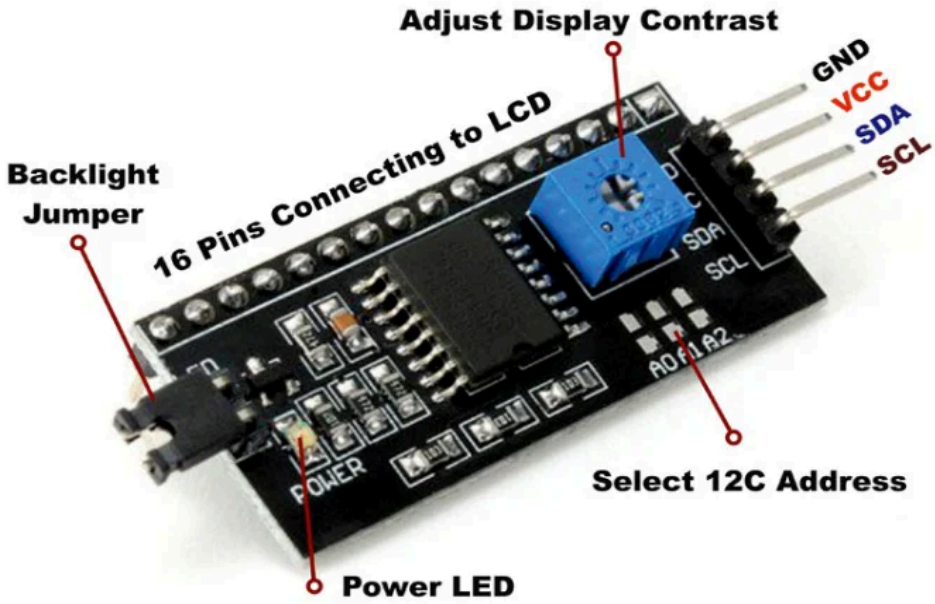
**LED- (Backlight Cathode):** Connect to the negative side of the backlight LED.

- 5V powered 4 x 20
- SPI communication
- Minimum 3 Pins interface to microcontroller
- Compatible with all types of microcontrollers
- Suitable for hobbyists and experts
- Back light and contrast control is available
- Size: 99mm x 60mm x 10mm




## Serial I2C control of LCD display using PCF8574.

**PINOUT :**




## Features of IIC/I2C Serial Interface Adapter Module:

- Serial I2C control of LCD display using PCF8574.
- Backlight can be enabled or disabled via a jumper on the board.
- Contrast control via a potentiometer
- RoHS-compliant I2C Serial LCD Daughter board.
- You can connect 16x4 and 20x4 LCD displays with I2C interface.
- Supports 4-bit mode, which Character Modules widely support.
- Utilizes a PCF8574 I2C chip to convert I2C serial data to parallel data for the LCD.
- The default I2C address is 0x3F, but it can be changed via 3 solder jumpers provided on the board.
- Allows control of up to 3 LCDs via a single I2C bus, each with its address.
- Enhances project efficiency and simplicity with its I2C communication protocol.

## Code Description

The code initializes the PZEM-004T and LCD modules, acquires data from the energy meter, processes it, and displays it on the LCD. Error handling mechanisms ensure that invalid data does not disrupt the display.

## Basic code without blynk :

```
#include <ESP8266WiFi.h>
#include <PZEM004Tv30.h>//https://github.com/mandulaj/PZEM-004T-v30
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
PZEM004Tv30 pzem(&Serial);

float voltage=0;
float current=0;
float power=0;
float energy=0;
float frequency=0;
float pf=0;
unsigned long lastMillis = 0;
LiquidCrystal_I2C lcd(0x27, 20, 4);

void setup()
{
  lcd.begin();   // initializing the LCD
  lcd.backlight(); // Enable or Turn On the backlight
  lcd.setCursor(0,0);  //Set cursor to first column of second row
  lcd.print("Smart Energy meter & ");
  delay(1000);
  lcd.setCursor(0,1);  //Set cursor to first column of second row
  lcd.print("Energy Monitor- IOT");
  delay(3000);
  lcd.clear();
  lcd.print("Device connecting...");
  delay(2000);
  lcd.setCursor(0,2);  //Set cursor to first column of second row
  lcd.print(".......please  wait");
  Serial.begin(9600);
  lcd.clear();
}
void loop()
{
    float voltage = pzem.voltage();
    if( !isnan(voltage) ){
        Serial.print("Voltage: ");
        Serial.print(voltage,1);
        Serial.println("V");
        lcd.setCursor(0,0);
        lcd.print("Voltage:      Volts");
        lcd.setCursor(8,0);
        lcd.print(voltage,1);
```

```
  } else {
      Serial.println("Error reading voltage");
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Voltage = 0.00Volts ");
      lcd.setCursor(0,1);
      lcd.print("Current = 0 Amps  ");
      lcd.setCursor(0,3);
      lcd.print("  No Power  Supply  ");
  }

  float current = pzem.current();
  if( !isnan(current) ){
      Serial.print("Current: ");
      Serial.print(current,3);
      Serial.println("A");
      lcd.setCursor(0,1);
      lcd.print("Current:      Amps");
      lcd.setCursor(8,1); lcd.print(current,3);
  } else {
      Serial.println("Error reading current");

  }

  float power = pzem.power();
  if( !isnan(power) ){
      Serial.print("Power: ");
      Serial.print(power,0);
      Serial.println("W");
      lcd.setCursor(0,2);
      lcd.print("     Watts");
      lcd.setCursor(0,2);
      lcd.print(power,0);
  } else {
      Serial.println("Error reading power");
          }

  float energy = pzem.energy();
  if( !isnan(energy) ){
      Serial.print("Energy: ");
      Serial.print(energy, 3);
      Serial.println("kWh");
      lcd.setCursor(12,2); lcd.print("     Kwh");
      lcd.setCursor(12,2); lcd.print(energy,3);
  } else {
      Serial.println("Error reading energy");
  }
```

```
    float frequency = pzem.frequency();
    if( !isnan(frequency) ){
        Serial.print("Frequency: ");
        Serial.print(frequency,1);
        Serial.println("Hz");
        lcd.setCursor(0,3);
        lcd.print("Freq:    Hz");
        lcd.setCursor(5,3);
        lcd.print(frequency,1);
    } else {
        Serial.println("Error reading frequency");
    }

    float pf = pzem.pf();
    if( !isnan(pf) ){
        Serial.print("PF: ");
        Serial.println(pf);
        lcd.setCursor(12,3);
        lcd.print("PF: ");
        lcd.setCursor(15,3);
        lcd.print(pf);
    } else {
        Serial.println("Error reading power factor");

    }

    Serial.println();
    delay(2000);
}
```

## Code Explanation :

This code is designed to monitor and display the electrical parameters (voltage, current, power, energy, frequency, and power factor) of a system using the **PZEM-004T energy meter** module and an **I2C-based LCD display** connected to an ESP8266 NodeMCU. Here's a detailed explanation of each line:

---

### Header Files and Libraries

`#include <ESP8266WiFi.h>`

- This library is specific to the **ESP8266 NodeMCU** microcontroller and is required for networking functions. Although not used in this program, it's included as a placeholder for potential IoT extensions.

```
#include <PZEM004Tv30.h>
```

- This library supports communication with the **PZEM-004T V3.0 energy meter**. It allows reading voltage, current, power, energy, frequency, and power factor via serial communication.

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

- Provides I2C communication functionality for interfacing with devices like the **I2C LCD**.

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

- Provides functions to control the I2C LCD display. This library simplifies displaying text on LCDs connected via I2C.

---

## Object Declarations

```
PZEM004Tv30 pzem(&Serial);
```

- Creates a `PZEM004Tv30` object to communicate with the energy meter. The `&Serial` argument specifies the hardware serial port for communication with the module.

```
float voltage=0, current=0, power=0, energy=0, frequency=0, pf=0;
```

- Declares floating-point variables to store the electrical parameters read from the PZEM-004T.

```
unsigned long lastMillis = 0;
```

- Variable to track time (not used in this program).

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

- Creates an LCD object.
  - `0x27`: I2C address of the LCD module.
  - `20, 4`: Specifies a 20x4 character display.

---

## `setup()` Function

```
lcd.begin();
```

- Initializes the LCD.

```
lcd.backlight();
```

- Turn on the LCD backlight.

```
lcd.setCursor(0,0);
lcd.print("    Energy meter    ");
```

- Sets the cursor to the first row and prints the title "Energy meter."

```
lcd.setCursor(0,1);
lcd.print("   Monitor by IOT   ");
```

- Displays a welcome message line by line with a delay of 1 second between each line.

```
lcd.clear();
lcd.print("Device connecting...");
delay(2000);
lcd.setCursor(0,2);
lcd.print(".......please  wait");
```

- Clears the LCD and displays a message indicating the device is connecting.

```
Serial.begin(9600);
```

- Initializes the hardware serial port for communication with the PZEM-004T module at a baud rate of 9600.

---

## `loop()` Function

**Reading Voltage**

```
float voltage = pzem.voltage();
if( !isnan(voltage) ){
```

1. Reads the voltage using the `voltage()` method.
2. `isnan()` checks if the reading is valid (not a "Not-a-Number" value).

```
lcd.setCursor(0,0); lcd.print("Voltage:      Volts");
lcd.setCursor(8,0); lcd.print(voltage,1);
```

3. Displays the voltage value on the LCD with 1 decimal place.

```
lcd.clear();
lcd.setCursor(0,0); lcd.print("Voltage = 0.00Volts ");
lcd.setCursor(0,1); lcd.print("Current = 0 Amps  ");
lcd.setCursor(0,3); lcd.print("  No Power  Supply  ");
```

4. If the reading is invalid, clear the LCD and display an error message.

---

**Reading Current**

```
float current = pzem.current();
if( !isnan(current) ){
```

1. Reads the current and checks for validity.

```
lcd.setCursor(0,1); lcd.print("Current:      Amps");
lcd.setCursor(8,1); lcd.print(current,3);
```

2. Displays the current value on the LCD with 3 decimal places.

---

**Reading Power**

```
float power = pzem.power();
if( !isnan(power) ){
```

1. Reads the power in watts.

```
lcd.setCursor(0,2); lcd.print("     Watts");
lcd.setCursor(0,2); lcd.print(power,0);
```

2. Displays the power value on the LCD with no decimal places.

---

**Reading Energy**

```
float energy = pzem.energy();
if( !isnan(energy) ){
```

1. Reads the energy in kilowatt-hours.

```
lcd.setCursor(12,2); lcd.print("     Kwh");
lcd.setCursor(12,2); lcd.print(energy,3);
```

2. Displays the energy value on the LCD with 3 decimal places.

---

**Reading Frequency**

```
float frequency = pzem.frequency();
if( !isnan(frequency) ){
```

1. Read the frequency in hertz.

```
lcd.setCursor(0,3); lcd.print("Freq:    Hz");
lcd.setCursor(5,3); lcd.print(frequency,1);
```

2. Displays the frequency on the LCD with 1 decimal place.

---

**Reading Power Factor**

```
float pf = pzem.pf();
if( !isnan(pf) ){
```

1. Read the power factor (unitless).

```
lcd.setCursor(12,3); lcd.print("PF: ");
lcd.setCursor(15,3); lcd.print(pf);
```

1. Displays the power factor on the LCD.

---

**Delay Between Readings**

```
delay(2000);
```

Waits for 2 seconds before taking the next set of readings.

## Step-by-Step Process

## Hardware Assembly

1. **Gather Components:**
    - Peacefair PZEM-004T energy meter.
    - NodeMCU ESP8266.
    - 20x4 LCD with I2C module.
    - Power supply.
    - Connecting wires and a breadboard.
2. **Connect the PZEM-004T:**
    - **TX (PZEM)** -> **RX (NodeMCU)**.
    - **RX (PZEM)** -> **TX (NodeMCU)**.
    - Power the PZEM-004T using a 5V power supply.
3. **Connect the LCD with I2C:**
    - **SDA (I2C LCD)** -> **D2 (NodeMCU)**.
    - **SCL (I2C LCD)** -> **D1 (NodeMCU)**.
    - **VCC** -> **VIN (NodeMCU)**.
    - **GND** -> **GND (NodeMCU)**.
4. **Power Connections:**
    - Provide 3.3V to the NodeMCU from an external power source or USB.
    - Ensure all grounds are connected together for proper operation.

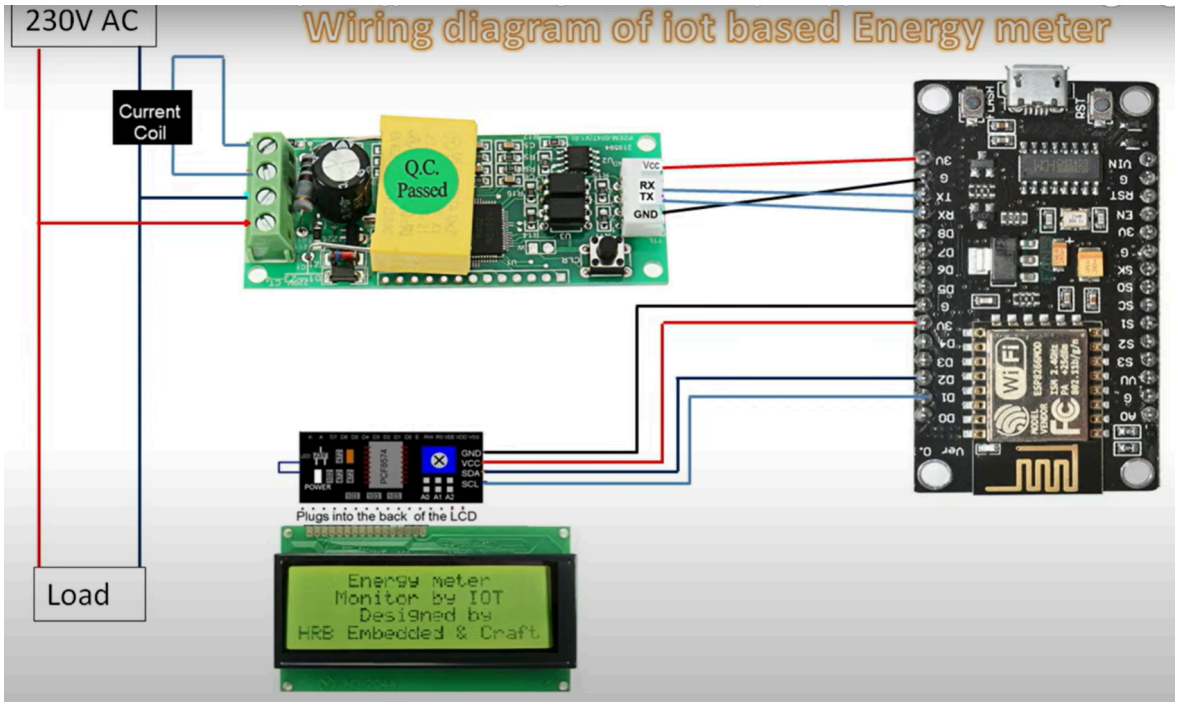
## Software Setup

1. Install the Arduino IDE on your computer.
2. Install the required libraries:
    - `PZEM 004Tv30` for interfacing with the energy meter.
    - `LiquidCrystal_I2C` for controlling the LCD.
3. Open the provided Arduino code.
4. Configure the correct COM port for your NodeMCU in the Arduino IDE.
5. Upload the code to the NodeMCU.

## Testing the System

1. Power on the setup.
2. Observe the initialization sequence on the LCD.
3. Connect a load to the PZEM-004T and check if the voltage, current, and other parameters are displayed correctly on the LCD.
4. Verify serial output using the Arduino IDE Serial Monitor.

---




## Features

- Real-time monitoring of voltage, current, power, energy, frequency, and power factor.
- Error handling for invalid or missing readings.
- User-friendly display interface.
- Potential for IoT integration via the Wi-Fi capabilities of the NodeMCU ESP8266.

### Pin Configuration

| Component | Pin on Component | Pin on NodeMCU |
|---|---|---|
| PZEM-004T | TX | RX |
| | RX | TX |
| | VCC | 5V |
| | GND | GND |
| I2C LCD | SDA | D2 |
| | SCL | D1 |
| | VCC | VIN |
| | GND | GND |

## Advantages

- Compact and portable design.
- Accurate and reliable measurements.
- Low power consumption.
- Scalable for IoT applications.

## Applications

- Residential energy monitoring.
- Industrial power management systems.
- Educational purposes for demonstrating energy measurement.

## Results

The system successfully measures and displays the following parameters:

1. **Voltage**: Accurate to 1 decimal place.
2. **Current**: Accurate to 3 decimal places.
3. **Power**: Displayed in watts (W).
4. **Energy**: Cumulative energy usage in kilowatt-hours (kWh).
5. **Frequency**: Displayed in hertz (Hz).
6. **Power Factor (PF)**: Unitless value representing system efficiency.

**Readings :**



Here is the schematic representation of the pin connections for your Arduino-based project with the **PZEM-004T** module and the **I2C LCD** display:

**Arduino code:**

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>
```

```
#include <PZEM004Tv30.h>

#include <SoftwareSerial.h>



// Define LCD and PZEM objects

LiquidCrystal_I2C lcd(0x27, 20, 4); // Adjust address if needed

SoftwareSerial pzemSerial(10, 11);  // RX = Pin 10, TX = Pin 11

PZEM004Tv30 pzem(pzemSerial);



void setup() {

  // Initialize LCD

  lcd.init();

  lcd.begin(20, 4);

  lcd.backlight();

  lcd.setCursor(0, 0);

  lcd.print("Smart Power Tracker ");

  lcd.setCursor(0, 1);

  lcd.print("Initializing...");

  delay(2000); // Show initializing message for 2 seconds

  lcd.clear();



  // Initialize Serial for debugging

  Serial.begin(9600);

  Serial.println("PZEM Data Logger");

}



void loop() {
```

```
// Read PZEM data
float voltage = pzem.voltage();
float current = pzem.current();
float power = pzem.power();
float energy = pzem.energy();

// Update LCD with PZEM data
lcd.setCursor(0, 0);
lcd.print("Voltage: ");
lcd.print(voltage, 1);
lcd.print("V  ");

lcd.setCursor(0, 1);
lcd.print("Current: ");
lcd.print(current, 2);
lcd.print("A  ");

lcd.setCursor(0, 2);
lcd.print("Power: ");
lcd.print(power, 0);
lcd.print("W  ");

lcd.setCursor(0, 3);
lcd.print("Energy: ");
lcd.print(energy, 3);
lcd.print("kWh");
```

```
  // Debug values to Serial Monitor
  Serial.print("Voltage: "); Serial.print(voltage); Serial.println(" V");
  Serial.print("Current: "); Serial.print(current); Serial.println(" A");
  Serial.print("Power: "); Serial.print(power); Serial.println(" W");
  Serial.print("Energy: "); Serial.print(energy); Serial.println(" kWh");
  Serial.println("---------------------");

  delay(2000); // Update interval
}
```

---

### Includes and Object Declarations

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <PZEM004Tv30.h>
#include <SoftwareSerial.h>
```

- **`#include <Wire.h>`**: This library is for I2C communication, required by the LCD module.
- **`#include <LiquidCrystal_I2C.h>`**: This library allows control of the LCD via I2C protocol.
- **`#include <PZEM004Tv30.h>`**: This library interfaces with the PZEM-004T module to read voltage, current, power, and energy.
- **`#include <SoftwareSerial.h>`**: This library enables software-based serial communication on specified pins.

---

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

- Creates an **LCD object** with the address `0x27` and specifies that the LCD has 20 columns and 4 rows.
- Adjust the address if your LCD has a different one (e.g., `0x3F`).

```
SoftwareSerial pzemSerial(10, 11);

PZEM004Tv30 pzem(pzemSerial);
```

- **`pzemSerial(10, 11)`**: Creates a software serial port on digital pins 10 (RX) and 11 (TX).
- **`PZEM004Tv30 pzem(pzemSerial)`**: Initializes the PZEM module using the software serial port.

---

### Setup Function

```
void setup() {
```

- The **`setup()`** function runs once when the Arduino is powered on or reset.

```
  lcd.init();

  lcd.begin(20, 4);

  lcd.backlight();
```

- **`lcd.init()`**: Initializes the LCD module.
- **`lcd.begin(20, 4)`**: Configures the LCD for 20 columns and 4 rows.
- **`lcd.backlight()`**: Turns on the LCD backlight for better visibility.

```
  lcd.setCursor(0, 0);

  lcd.print("Smart Power Tracker ");

  lcd.setCursor(0, 1);

  lcd.print("Initializing...");
```

- **`lcd.setCursor(0, 0)`**: Moves the cursor to the first row, first column (0-based index).
- **`lcd.print("Smart Power Tracker ")`**: Displays the message "Smart Power Tracker " on the first row.
- **`lcd.setCursor(0, 1)`**: Moves the cursor to the second row, first column.
- **`lcd.print("Initializing...")`**: Displays the message "Initializing..." on the second row.

```
  delay(2000);

  lcd.clear();
```

- **`delay(2000)`**: Pauses the program for 2 seconds to show the initializing message.
- **`lcd.clear()`**: Clears the LCD display.

---

```
  Serial.begin(9600);
```

```
  Serial.println("PZEM Data Logger");
```

- **`Serial.begin(9600)`**: Starts serial communication at a baud rate of 9600 for debugging.
- **`Serial.println("PZEM Data Logger")`**: Prints "PZEM Data Logger" to the Serial Monitor.

---

## Loop Function

```
void loop() {
```

- The **`loop()`** function runs repeatedly after `setup()`.

---

#### PZEM Data Reading

```
  float voltage = pzem.voltage();
```

```
  float current = pzem.current();
```

```
  float power = pzem.power();
```

```
  float energy = pzem.energy();
```

- **`pzem.voltage()`**: Reads the voltage in volts from the PZEM module.
- **`pzem.current()`**: Reads the current in amps from the PZEM module.
- **`pzem.power()`**: Reads the power in watts from the PZEM module.
- **`pzem.energy()`**: Reads the energy in kilowatt-hours (kWh) from the PZEM module.
- These readings are stored in `float` variables for display.

---

#### Updating the LCD

```
  lcd.setCursor(0, 0);
```

```
  lcd.print("Voltage: ");
```

```
  lcd.print(voltage, 1);
```

```
  lcd.print("V  ");
```

- Moves the cursor to the first row, first column.
- Displays "Voltage: " followed by the voltage reading with one decimal place (1) and appends "V".

```
  lcd.setCursor(0, 1);
```

```
  lcd.print("Current: ");
```

```
  lcd.print(current, 2);
```

```
  lcd.print("A  ");
```

- Moves the cursor to the second row.
- Displays "Current: " followed by the current reading with two decimal places and appends "A".

```
  lcd.setCursor(0, 2);
```

```
  lcd.print("Power: ");
```

```
  lcd.print(power, 0);
```

```
  lcd.print("W  ");
```

- Moves the cursor to the third row.
- Displays "Power: " followed by the power reading with no decimal places and appends "W".

```
  lcd.setCursor(0, 3);
```

```
  lcd.print("Energy: ");
```

```
  lcd.print(energy, 3);
```

```
  lcd.print("kWh");
```

- Moves the cursor to the fourth row.
- Displays "Energy: " followed by the energy reading with three decimal places and appends "kWh".

---

**Debug Output to Serial Monitor**

```
  Serial.print("Voltage: "); Serial.print(voltage); Serial.println(" V");
```

```
  Serial.print("Current: "); Serial.print(current); Serial.println(" A");
```

```
  Serial.print("Power: "); Serial.print(power); Serial.println(" W");
```

```
  Serial.print("Energy: "); Serial.print(energy); Serial.println(" kWh");
  Serial.println("--------------------");
```

- Prints the voltage, current, power, and energy values to the Serial Monitor for debugging and logs them.

---

**Delay Between Updates**

```
delay(2000);
```

- Pauses the program for 2 seconds before taking the next set of readings and updating the LCD.

## Pin Connections

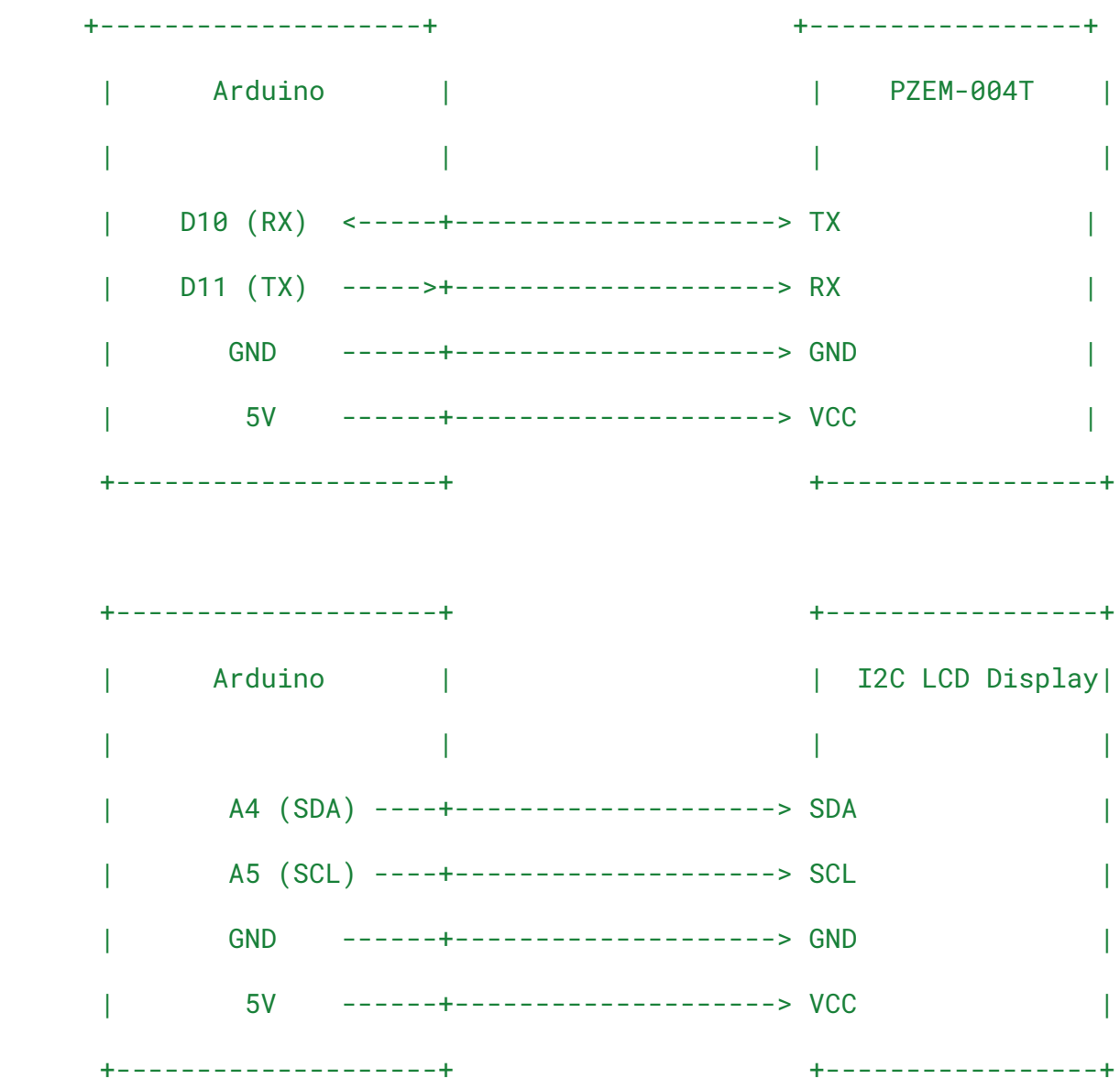
---

**Pin Connections**

| Component | PZEM Pin | Arduino Pin | Notes |
|---|---|---|---|
| PZEM-004T | TX | D10 (RX) | Use a voltage level shifter or resistor divider to step down Arduino's 5V to 3.3V for PZEM RX. |
| | RX | D11 (TX) | Direct connection; Arduino's TX sends 5V signals to PZEM RX. |
| | GND | GND | Common ground between Arduino and PZEM module. |
| | VCC | 5V | Power the PZEM module from Arduino's 5V pin. |
| I2C LCD Display | SDA | A4 | I2C data line. |
| | SCL | A5 | I2C clock line. |
| | GND | GND | Common ground between Arduino and LCD. |
| | VCC | 5V | Power the LCD from Arduino's 5V pin. |

## Schematic Diagram

**ASCII Representation:**

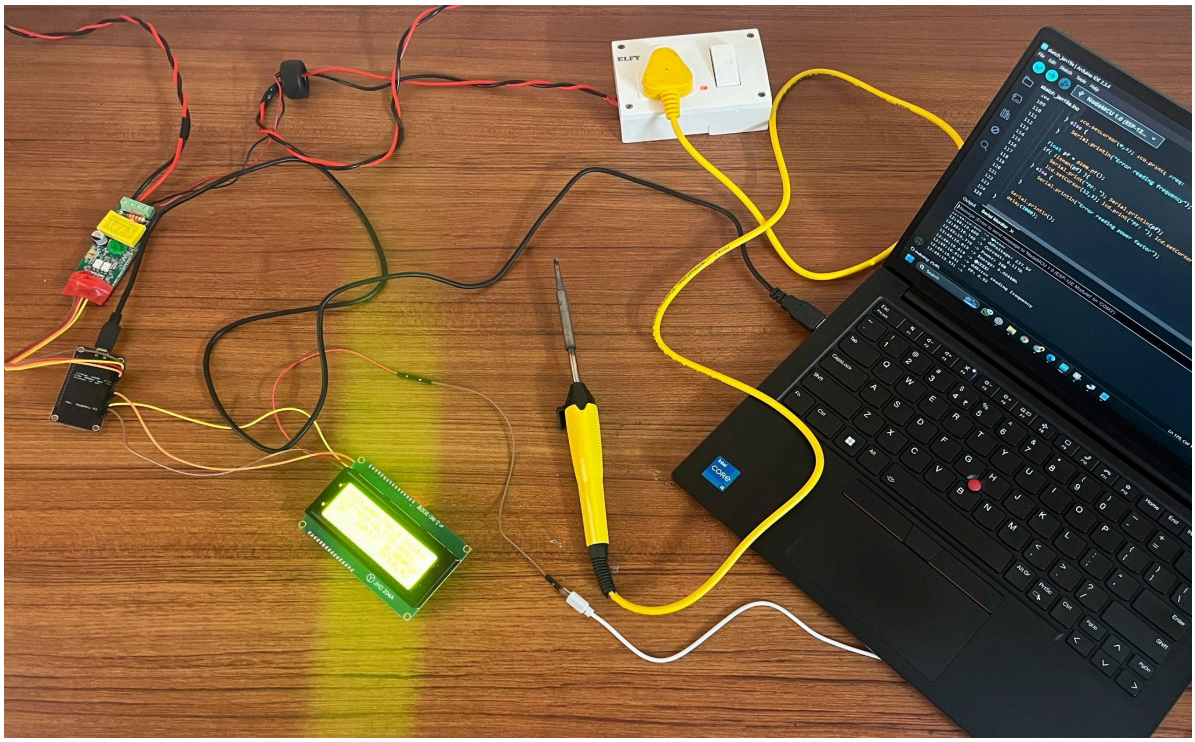



---

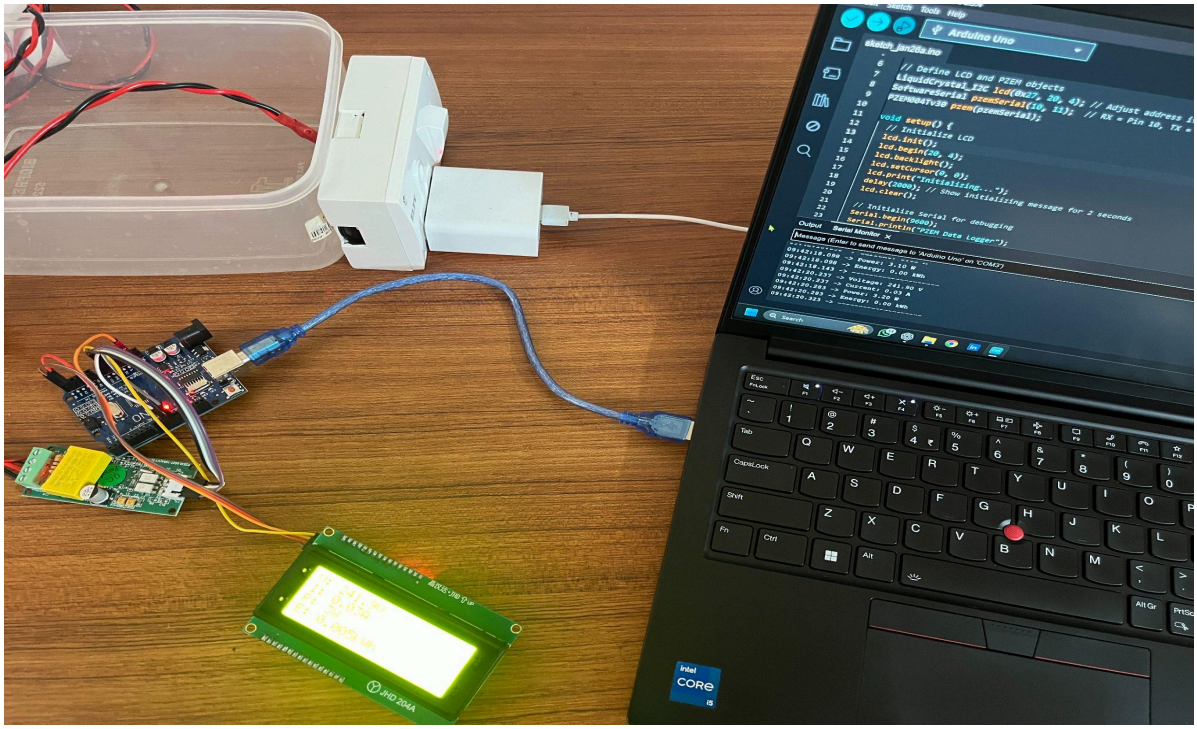
## Visual Schematic:

For a professional schematic, you can use tools like **Fritzing** or **Tinkercad** to visualize the pinouts. Let me know if you want assistance creating a detailed graphical schematic.

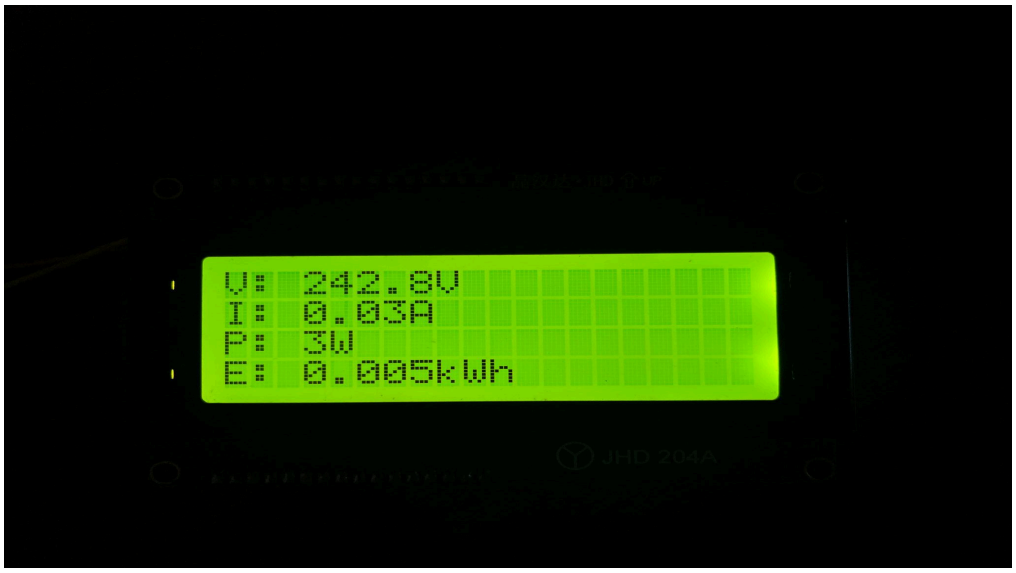
## Using ESP8266



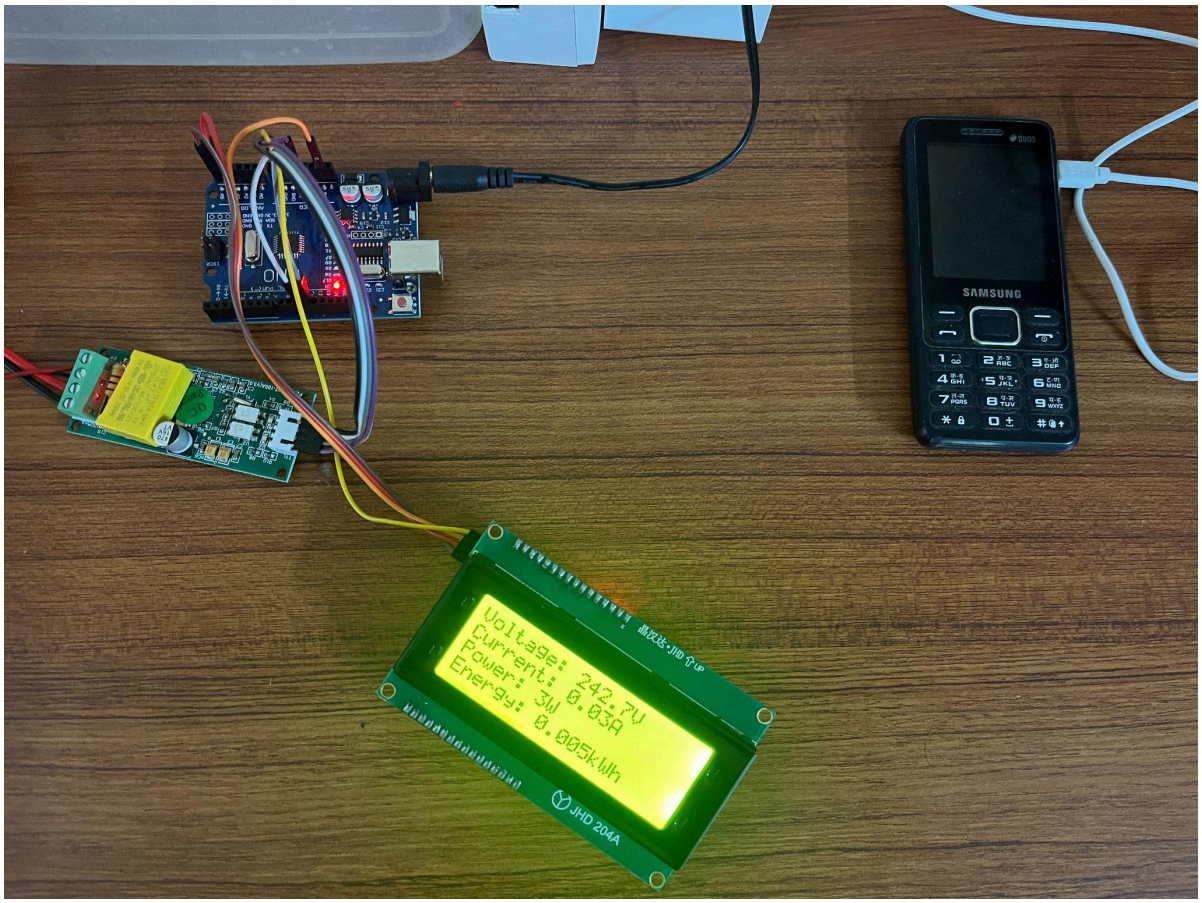
## Using Arduino uno R3



**Readings:**


V: 242.8V
I: 0.03A
P: 3W
E: 0.005kWh


Voltage: 242.3V
Current: 0.03A
Power: 3W
Energy: 0.005kWh




## Conclusion

The Smart Energy Monitoring System is a cost-effective solution for real-time energy parameter monitoring. It provides precise and reliable data, which can be further utilized for IoT-based applications. Future enhancements could include:

- Cloud integration for remote monitoring.
- Smartphone app interface for data visualization.
- Advanced analytics for energy usage optimization.

## References

1. Peacefair PZEM-004T Datasheet and User Manual.
2. LiquidCrystal_I2C Library Documentation.
3. Arduino IDE Documentation.
4. GitHub Repository for PZEM 004T v30 Library: https://github.com/mandulaj/PZEM-004T-v30

---