Institute of

Electrical Engineering Department
EE 20A

Project EMB
Measuring AC Power Using ESP32 and ZMPT101b, ZHT103 Modules

Submitted To:
Sir Asad ur Rehman
Submitted By:
Fahad Qalbi (210401060)

1. Introduction

The ESP32 Dev Kit v1 is a versatile, feature-packed development board produced by Espressif that is designed around their powerful ESP32 system on a chip (SoC). The ESP32 is a powerful dual-core microcontroller with integrated Wi-Fi and Bluetooth capabilities, making it suitable for a wide range of applications from simple prototyping to complex IoT deployments.

Key features of the ESP32 Dev Kit v1 include:

- Dual-core Xtensa® 32-bit LX6 microprocessor, operating at a frequency of up to 240 MHz.
- Integrated 520 KB SRAM.
- Built-in 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth (v4.2 BR/EDR and BLE) capabilities.
- 30 GPIO pins (input/output), which can support a variety of peripheral functions such as UART, SPI, I2C, etc.
- 2 analog to digital converters (ADCs) with 12-bit resolution.
- 2 digital to analog converters (DACs) with 8-bit resolution.
- 3 UART interfaces.
- Integrated LiPo Battery Charging Circuit.

ESP32 Dev Kit v1 Pinout:

The ESP32 Dev Kit v1 features a number of GPIO pins that support a variety of functions. Key pins include:

- -3V3 (3.3V): Power supply pin (3.3V). Provides the power supply to the chip and other components on the board.
- GND: Ground pin.
- EN: Reset/Enable pin. Pulling this pin LOW resets the chip.
- VP / VN: ADC1_CH7 (GPIO36) / ADC1_CH6 (GPIO39) respectively.
- RXD0 / TXD0: UART interface. RXD receives data, TXD transmits data.
- IO21 / IO22: Typically used for I2C communication (SDA / SCL respectively).
- IO19 / IO23: Typically used for SPI communication (MISO / MOSI respectively).
- IO18: Typically used for SPI communication (SCK).
- IO5: Typically used for SPI communication (CS).
- A0 – A3: Analog pins.
- IO25 / IO26: DAC1 / DAC2 respectively.

The specific functions of these pins can be customized through software configuration, giving you a lot of flexibility when designing your own projects.

The ZMPT101B module is a voltage transformer ideal for measuring AC voltage. It can transform the AC signal into a smaller amplitude signal. On the other hand, the ZHT103 is a current sensor module that can be used to measure AC current.

16x2 LCD display:

In this project, I also used a 16x2 LCD display, which is an alphanumeric display that can show up to 32 characters in two lines (16 characters each). It is connected to the ESP32 using an I2C module, a simple, reliable communication protocol, allowing the LCD to receive data from the ESP32 over just two wires. The I2C bus uses two lines: a serial data line (SDA) and a serial clock line (SCL). SCL stands for Serial Clock. It's one of the two signals used in the I2C communication protocol, the other being the Serial Data (SDA) line. In I2C communication, the SCL line is used to synchronize all data transfers over the I2C bus. It's controlled by the master device, which generates the clock signal.

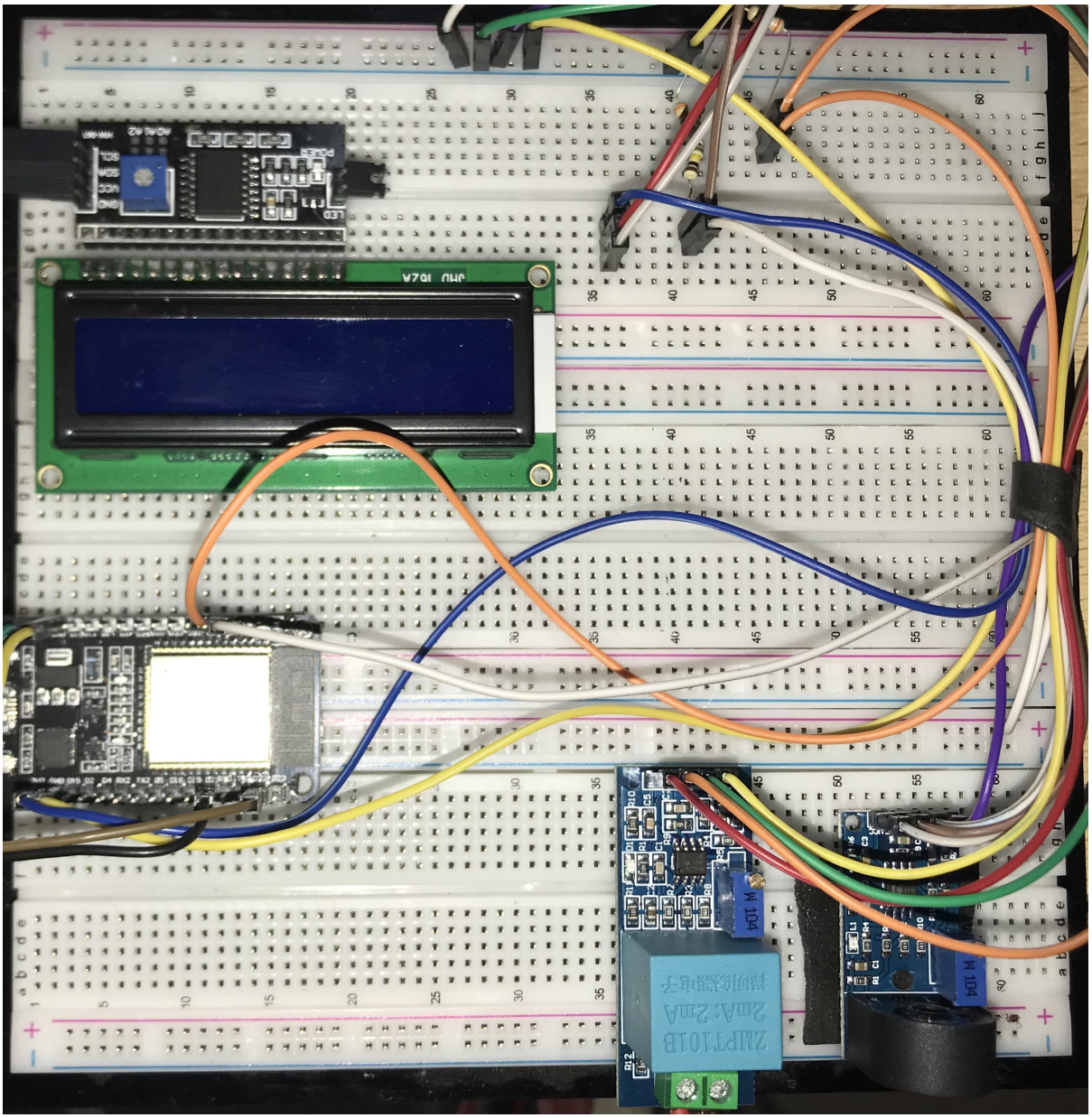
1.1 Emon Library

The Emon library is used. This library has been designed to make energy monitoring easier by providing calibration and phase correction for AC power measurements. The library calculates real power, apparent power, power factor, Vrms, and Irms.

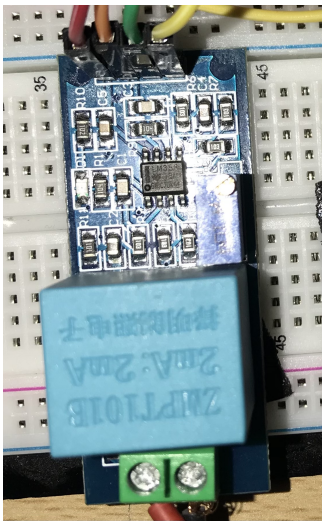
1.2 LiquidCrystal_I2C Library

The LiquidCrystal_I2C library is used to control the LCD display via I2C communication. This allows us to write to the display and control the backlight and cursor position, among other things.

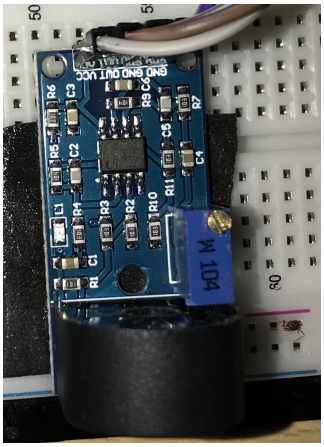
2. Component Interconnections:



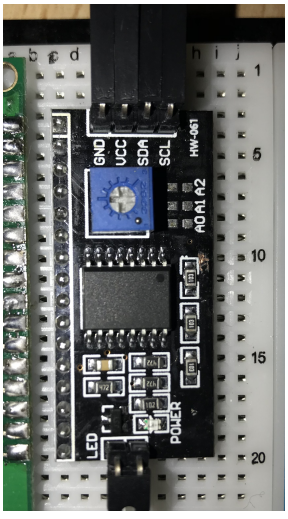
As shown in the wiring diagram above, the ZMPT101B module is connected to the ESP32 to measure the AC voltage.



The ZHT103 module is also connected to the ESP32 to measure AC current.



The LCD display module is connected to the ESP32 using an I2C module, allowing it to display voltage, current, and power readings.



The code handles the interaction between these components, collecting measurements from the voltage and current sensors and outputting the results to the LCD display.

3. Working:

1. Library and Objects Initialization:

At the top, the necessary libraries are included, and a LiquidCrystal_I2C object (lcd) and an EnergyMonitor object (emon1) are initialized. The LCD has 16 columns and 2 rows, and it uses the I2C address 0x27.

2. Setup:

In the setup() function, serial communication is initiated at a baud rate of 9600 for debugging purposes. The LCD is also initialized and a start-up message "AC Power Meter" is displayed. The `emon1.voltage()` and `emon1.current()` methods are used to set the calibration values for the voltage and current sensors.

3. Main Loop:

In the loop() function, the following sequence is repeated continuously:

- The string "Sequence: I+V,P" is printed on the LCD.
- RMS voltage and current are calculated using `emon1.calcVI()` for voltage and `emon1.calcIrms()` for current.
- The calculated voltage and current are multiplied to obtain the power in watts.
- The RMS voltage, current, and power are displayed on the LCD, one at a time, with a 1-second delay between each display.
- The RMS voltage and current values are also sent to the serial monitor.

The lcd.print() and lcd.setCursor() functions are used to control what is displayed on the LCD and where it is displayed. The Serial.print() and Serial.println() functions are used to send data to the serial monitor. The delay() function is used to create a pause between different readings, and lcd.clear() is used to clear the LCD before displaying new data.

3. Code :

```
// Include necessary libraries
#include <LiquidCrystal_I2C.h>
#include "EmonLib.h"

// Declare and initialize display characteristics
int lcdColumns = 16;
int lcdRows = 2;
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

// Create an instance of EnergyMonitor
EnergyMonitor emon1;

// The setup function runs once when you press reset or power the board
void setup() {
  // Open serial communications and set the baud rate to 9600 bps:
  Serial.begin(9600);

  // Initialize the LCD
  lcd.init();
  lcd.backlight();
  lcd.cursor_off();

  // Print a start-up message
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("AC Power Meter");

  // Wait for 1 second
  delay(1000);

  // Set calibration values for voltage and current sensors
  emon1.voltage(35,366, 0);
  emon1.current(34, 1.100999);           // Current: input pin, calibration.

  // Clear the LCD for next messages
  lcd.clear();
}

// The loop function runs over and over again forever
void loop() {
  // Print sequence
  lcd.print("Sequence: I+V,P");

  // Calculate Vrms and Irms
  emon1.calcVI(20, 2000);
```

```
double Vrms = emon1.Vrms;
double Irms = emon1.calcIrms(2000);  // Calculate Irms only

// Calculate Power
float watt=(Irms*Vrms);

// Clear the LCD for next messages
lcd.clear();

// Print Vrms on the LCD
lcd.setCursor(0,0);
lcd.print("Vrms:");
lcd.setCursor(6,0);
lcd.print(Vrms);
lcd.setCursor(14,0);
lcd.print("V");

// Print Vrms on the Serial Monitor
Serial.print(Vrms);
Serial.print(" V, ");

// Print Irms on the LCD
lcd.setCursor(0,1);
lcd.print("Irms:");
lcd.setCursor(6,1);
lcd.print(Irms);
lcd.setCursor(11,1);
lcd.print("A");

// Print Irms on the Serial Monitor
Serial.print(Irms);
Serial.println(" A, ");

// Wait for 1 second
delay(1000);

// Clear the LCD for next messages
lcd.clear();

// Print Power on the LCD
lcd.setCursor(0,0);
lcd.print("Power:");
lcd.setCursor(8,0);
lcd.print(watt);
lcd.setCursor(15, 0);
lcd.print("W");

// Wait for 1 second
```

```
  delay(1000);

  // Clear the LCD for next messages
  lcd.clear();
}
```

4. Conclusion

From this experiment, we have gained valuable experience in measuring AC voltage and current using the ESP32 along with the ZMPT101B and ZHT103 modules. We learned how to display these readings on an LCD display using the I2C communication protocol. We also became familiar with the EmonLib library, which simplifies the process of measuring power in AC circuits. This project can serve as a basis for many applications, including energy management systems, home automation, or condition monitoring in industrial settings. It also highlights the versatility and power of the ESP32 in interfacing with various hardware modules and sensors.