AC Digital Multifunction Meter Watt Power Volt Amp TTL Current Test Module PZEM-004T With Coil 0-100A 80-260V AC For Arduino

Friday, 29 July 2022 12:27

AC Digital Multifunction Meter Watt Power Volt Amp TTL Current Test Module PZEM-004T With Coil 0-100A 80-260V AC For Arduino

- https://github.com/mandulaj/PZEM-004T-v30
- https://github.com/olehs/PZEM004T
- https://www.arduino.cc/reference/en/libraries/pzem004tv30/

- https://github.com/mandulaj/PZEM-004T-v30/blob/master/examples/PZEMMultiDevice/PZEMMultiDevice.ino

PZEM-004T v3.0
LINKS & REFs |

Arduino library for Peacefair PZEM-004T-10A and PZEM-004T-100A v3.0 Energy monitor using the ModBUS interface.

The Version 3.0 PZEM is an upgraded version of the older PZEM-004T for which you can find the library Here

Main features

- Measures Voltage, Current, Power, Energy, Power Factor and Frequency (New in Version 3.0)
- 247 unique programmable slave addresses
  - Enables multiple slaves to use the same Serial interface PZEM MulitDevice Demo
- Internal Energy counter up to 9999.99kWh

Other features

- Over power alarm
- Energy counter reset
- CRC16 checksum
- Better, but not perfect mains isolation

Common issue:

Make sure the device is connected to the AC power! The 5V only power the optocouplers, not the actual chip. Also please be safe, AC is dangerous! If you don't know what you are doing, you can die! You are responsible for your own stupidity. So don't be stupid.

The module

This module is an upgraded version of the PZEM-004T with frequency and power factor measurement features, available at the usual places. It communicates using a TTL interface over a Modbus-RTU like communication protocol but is incompatible with the older @olehs library found here: https://github.com/olehs/PZEM004T. I would like to thank @olehs for the great library which inspired me to write this one.

Manufacturer (optimistic) specifications

Function	Measuring range	Resolution	Accuracy	TODO: Realistic specifications
Voltage	80~260V	0.1V	0.5%	
Current	0~10A or 0~100A*	0.01A or 0.02A*	0.5%	
Active power	0~2.3kW or 0~23kW*	0.1W	0.5%	
Active energy	0~9999.99kWh	1Wh	0.5%	
Frequency	45~65Hz	0.1Hz	0.5%	
Power factor	0.00~1.00	0.01	1%	

* Using the external current transformer instead of the built in shunt

Compatibility

MCU	Hardware Serial	Software Serial	Not Tested	Examples	Notes
ATmega168			X	HardwareSerial SoftwareSerial	
ATmega328 (Arduino Uno)	(✓)	✓		HardwareSerial SoftwareSerial	HW Serial conflicts with Debug output. It can be used however without having any Serial Console output
ATmega2560 (Arduino Mega)	✓	✓		HardwareSerial SoftwareSerial	
ESP8266	(✓)	✓		SoftwareSerial	HW Serial conflicts with Debug output Serial
ESP32	✓	✗		HardwareSerial	SW Serial not really needed as ESP32 has 3 HW serials with configurable pins
STM32 BluePill			X		

Examples

Hardware Serial

This example uses Hardware Serial2 in order to interface with the PZEM module. Note that not all MCUs feature multiple Serial ports. It won't for example work on the Arduino Uno.

```
#include<PZEM004Tv30.h>/*Hardware Serial2 is only available on certain boards. * For example the
Arduino MEGA 2560*/#ifdefined(ESP32)
PZEM004Tv30 pzem(Serial2, 16, 17);
#elsePZEM004Tv30 pzem(Serial2);
#endifvoidsetup() {
   Serial.begin(115200);
//Uncomment in order to reset the internal energy counter//pzem.resetEnergy()}
voidloop() {

   Serial.print("Custom Address:");
   Serial.println(pzem.readAddress(), HEX);
//Read the data from the sensorfloatvoltage = pzem.voltage();
   floatcurrent = pzem.current();
   floatpower = pzem.power();
   floatenergy = pzem.energy();
   floatfrequency = pzem.frequency();
   floatpf = pzem.pf();
//Check if the data is validif(isnan(voltage)){
      Serial.println("Error reading voltage");
   } elseif(isnan(current)) {
      Serial.println("Error reading current");
   } elseif(isnan(power)) {
      Serial.println("Error reading power");
   } elseif(isnan(energy)) {
      Serial.println("Error reading energy");
   } elseif(isnan(frequency)) {
      Serial.println("Error reading frequency");
   } elseif(isnan(pf)) {
      Serial.println("Error reading power factor");
   } else{
//Print the values to the Serial consoleSerial.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,3);     Serial.println("kWh");
      Serial.print("Frequency: ");    Serial.print(frequency, 1); Serial.println("Hz");
      Serial.print("PF: ");           Serial.println(pf);
}
Serial.println();
   delay(2000);
}
```

Output:

```
Custom Address:10
Voltage: 229.60V
Current: 0.10A
Power: 4.50W
Energy: 7.368kWh
```

```
Frequency: 50.0Hz
PF: 0.19
```

Software Serial

Using the <SoftwareSerial.h> library...

```
#include<PZEM004Tv30.h>#include<SoftwareSerial.h>/*Use software serial for the PZEM * Pin 11 Rx
(Connects to the Tx pin on the PZEM) * Pin 12 Tx (Connects to the Rx pin on the
PZEM)*/SoftwareSerial pzemSWSerial(11, 12);
PZEM004Tv30 pzem;
voidsetup() {
 Serial.begin(115200);

 pzem = PZEM004Tv30(pzemSWSerial);
}
voidloop() {

   Serial.print("Custom Address:");
   Serial.println(pzem.readAddress(), HEX);
//Read the data from the sensorfloatvoltage = pzem.voltage();
   floatcurrent = pzem.current();
   floatpower = pzem.power();
   floatenergy = pzem.energy();
   floatfrequency = pzem.frequency();
   floatpf = pzem.pf();
//Check if the data is validif(isnan(voltage)){
       Serial.println("Error reading voltage");
   } elseif(isnan(current)) {
       Serial.println("Error reading current");
   } elseif(isnan(power)) {
       Serial.println("Error reading power");
   } elseif(isnan(energy)) {
       Serial.println("Error reading energy");
   } elseif(isnan(frequency)) {
       Serial.println("Error reading frequency");
   } elseif(isnan(pf)) {
       Serial.println("Error reading power factor");
   } else{
//Print the values to the Serial consoleSerial.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,3);     Serial.println("kWh");
       Serial.print("Frequency: ");   Serial.print(frequency, 1); Serial.println("Hz");
       Serial.print("PF: ");          Serial.println(pf);
}
Serial.println();
   delay(2000);
}
```

Output:

```
Custom Address:11
Voltage: 229.60V
Current: 0.10A
Power: 4.50W
Energy: 7.368kWh
Frequency: 50.0Hz
PF: 0.19
```

Links

Check LINKS & REFs page for additional DOC's and related links

From <https://github.com/mandulaj/PZEM-004T-v30>

From <https://www.aliexpress.com/item/1005002875613737.html?algo_pvid=4d2ccc31-a121-49f1-96eb-8d55c104737c&algo_exp_id=4d2ccc31-a121-49f1-96eb-8d55c104737c-12&pdp_ext_f=%7B%22sku_id%22%3A%2212000029239124764%22%7D&pdp_npi=2%40dis%21EUR%21%214.11%21%21%212.3%21%21%21402103255b16590722993151214efb54%2112000029239124764%21sea>

PZEM-004T AC Digital Multifunction Meter Watt Power Volt Amp Current Test Module For Arduino TTL COM2\COM3\COM4 0-100A 80-260V

Model: PZEM-004T
Overview
This document describes the specification of the PZEM-004T AC communication module, the module is mainly used for measuring AC voltage, current, active power, frequency, power

factor and active energy, the module is without display function, the data is read through the TTL interface.
PZEM-004T-10A: Measuring Range 10A (Built-in Shunt)
PZEM-004T-100A: Measuring Range 100A (external transformer)

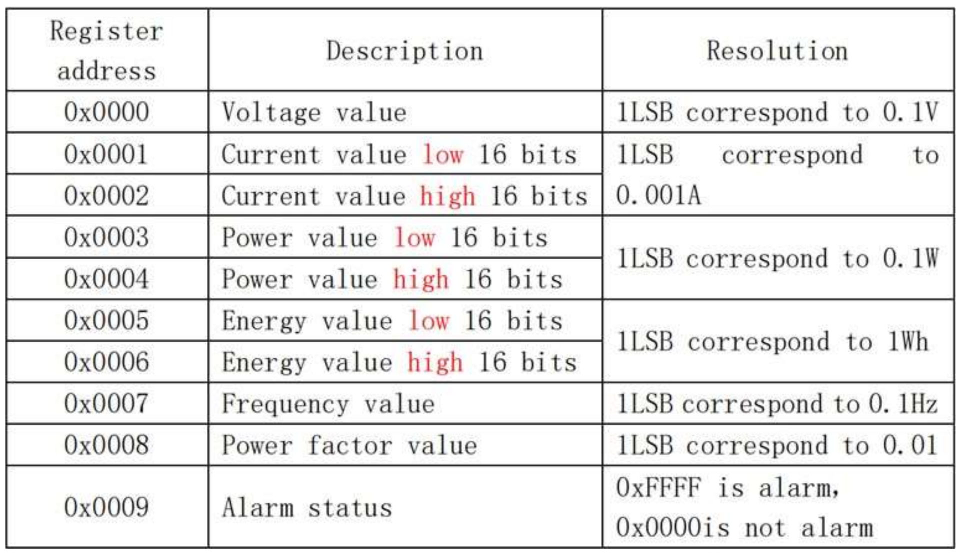
1.Function description
1.1 Voltage
1.1.1 Measuring range:80～260V
1.1.2 Resolution: 0.1V
1.1.3 Measurement accuracy: 0.5%
1.2 Current
1.2.1 Measuring range: 0～10A(PZEM-004T-10A); 0～100A(PZEM-004T-100A)
1.2.2 Starting measure current: 0.01A(PZEM-004T-10A); 0.02A(PZEM-004T-100A)
1.2.3 Resolution: 0.001A
1.2.4 Measurement accuracy: 0.5%
1.3 Active power
1.3.1 Measuring range: 0～2.3kW(PZEM-004T-10A); 0～23kW(PZEM-004T-100A)
1.3.2 Starting measure power: 0.4W
1.3.3 Resolution: 0.1W
1.3.4 Display format:
＜1000W, it display one decimal, such as: 999.9W
≥1000W, it display only integer, such as: 1000W
1.3.5 Measurement accuracy: 0.5%
1.4 Power factor
1.4.1 Measuring range: 0.00～1.00
1.4.2 Resolution: 0.01
1.4.3 Measurement accuracy: 1%
1.5 Frequency
1.5.1 Measuring range: 45Hz～65Hz
1.5.2 Resolution: 0.1Hz
1.5.3 Measurement accuracy: 0.5%
1.6 Active energy
1.6.1 Measuring range: 0～9999.99kWh
1.6.2 Resolution: 1Wh
1.6.3 Measurement accuracy: 0.5%
1.6.4 Display format:
＜10kWh, the display unit is Wh(1kWh=1000Wh), such as: 9999Wh
≥10kWh, the display unit is kWh, such as: 9999.99kWh
1.6.5 Reset energy: use software to reset.
1.7 Over power alarm
Active power threshold can be set, when the measured active power exceeds the threshold, it can alarm
1.8 Communication interface
RS485 interface。

2 Communication protocol
2.1 Physical layer protocol
Physical layer use UART to RS485 communication interface
Baud rate is 9600, 8 data bits, 1 stop bit, no parity

2.2 Application layer protocol
The application layer use the Modbus-RTU protocol to communicate. At present, it only supports function codes such as 0x03 (Read Holding Register), 0x04 (Read Input Register), 0x06 (Write Single Register), 0x41 (Calibration), 0x42 (Reset energy).etc.
0x41 function code is only for internal use (address can be only 0xF8), used for factory calibration and return to factory maintenance occasions, after the function code to increase 16-bit password, the default password is 0x3721
The address range of the slave is 0x01 ~ 0xF7. The address 0x00 is used as the broadcast address, the slave does not need to reply the master. The address 0xF8 is used as the general address, this address can be only used in single-slave environment and can be used for calibration etc.operation.

2.3 Read the measurement result
The command format of the master reads the measurement result is(total of 8 bytes):
Slave Address + 0x04 + Register Address High Byte + Register Address Low Byte + Number of Registers High Byte + Number of Registers Low Byte + CRC Check High Byte + CRC Check Low Byte.
The command format of the reply from the slave is divided into two kinds:
Correct Reply: Slave Address + 0x04 + Number of Bytes + Register 1 Data High Byte + Register 1 Data Low Byte + ... + CRC Check High Byte + CRC Check Low Byte
Error Reply: Slave address + 0x84 + Abnormal code + CRC check high byte + CRC check low byte
Abnormal code analyzed as following (the same below)
● 0x01,Illegal function
● 0x02,Illegal address
● 0x03,Illegal data
● 0x04,Slave error

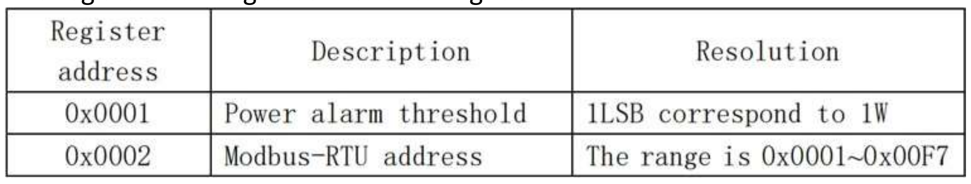
The register of the measurement results is arranged as the following table



Register address	Description	Resolution
0x0000	Voltage value	1LSB correspond to 0.1V
0x0001	Current value low 16 bits	1LSB correspond to 0.001A
0x0002	Current value high 16 bits	
0x0003	Power value low 16 bits	1LSB correspond to 0.1W
0x0004	Power value high 16 bits	
0x0005	Energy value low 16 bits	1LSB correspond to 1Wh
0x0006	Energy value high 16 bits	
0x0007	Frequency value	1LSB correspond to 0.1Hz
0x0008	Power factor value	1LSB correspond to 0.01
0x0009	Alarm status	0xFFFF is alarm, 0x0000is not alarm

For example, the master sends the following command (CRC check code is replaced by 0xHH and 0xLL, the same below)
0x01 + 0x04 + 0x00 + 0x00 + 0x00 + 0x0A + 0xHH + 0xLL
Indicates that the master needs to read 10 registers with slave address 0x01 and the start address of the register is 0x0000
The correct reply from the slave is as following:
0x01 + 0x04 + 0x14 + 0x08 + 0x98 + 0x03 + 0xE8+0x00 + 0x00 +0x08 + 0x98+ 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x01 + 0xF4 + 0x00 + 0x64 + 0x00 + 0x00 + 0xHH + 0xLL
The above data shows
- Voltage is 0x0898, converted to decimal is 2200, display 220.0V
- Current is 0x000003E8, converted to decimal is 1000, display 1.000A
- Power is 0x00000898, converted to decimal is 2200, display 220.0W
- Energy is 0x00000000, converted to decimal is 0, display 0Wh
- Frequency is 0x01F4, converted to decimal is 500, display 50.0Hz
- Power factor is 0x0064, converted to decimal is 100, display 1.00
- Alarm status is 0x0000, indicates that the current power is lower than the alarm power threshold

2.4 Read and modify the slave parameters
At present,it only supports reading and modifying slave address and power alarm threshold
The register is arranged as the following table



Register address	Description	Resolution
0x0001	Power alarm threshold	1LSB correspond to 1W
0x0002	Modbus-RTU address	The range is 0x0001~0x00F7

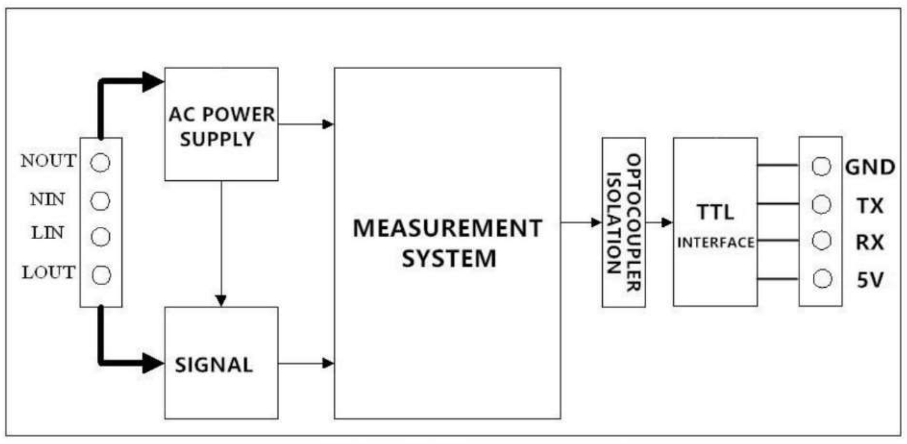
The command format of the master to read the slave parameters and read the measurement results are same(descrybed in details in Section 2.3), only need to change the function code from 0x04 to 0x03.
The command format of the master to modify the slave parameters is (total of 8 bytes):
Slave Address + 0x06 + Register Address High Byte + Register Address Low Byte + Register Value High Byte + Register Value Low Byte + CRC Check High Byte + CRC Check Low Byte.
The command format of the reply from the slave is divided into two kinds:
Correct Response: Slave Address + 0x06 + Number of Bytes + Register Address Low Byte + Register Value High Byte + Register Value Low Byte + CRC Check High Byte + CRC Check Low Byte.
Error Reply: Slave address + 0x86 + Abnormal code + CRC check high byte + CRC check low byte.
For example, the master sets the slave's power alarm threshold:
0x01 + 0x06 + 0x00 + 0x01 + 0x08 + 0xFC + 0xHH + 0xLL
Indicates that the master needs to set the 0x0001 register (power alarm threshold) to 0x08FC (2300W).
Set up correctly, the slave return to the data which is sent from the master.
For example, the master sets the address of the slave
0x01 + 0x06 + 0x00 + 0x02 + 0x00 + 0x05 + 0xHH + 0xLL
Indicates that the master needs to set the 0x0002 register (Modbus-RTU address) to 0x0005
Set up correctly, the slave return to the data which is sent from the master.

2.5 Reset energy
The command format of the master to reset the slave's energy is (total 4 bytes):
Slave address + 0x42 + CRC check high byte + CRC check low byte.
Correct reply: slave address + 0x42 + CRC check high byte + CRC check low byte.
Error Reply: Slave address + 0xC2 + Abnormal code + CRC check high byte + CRC check low byte

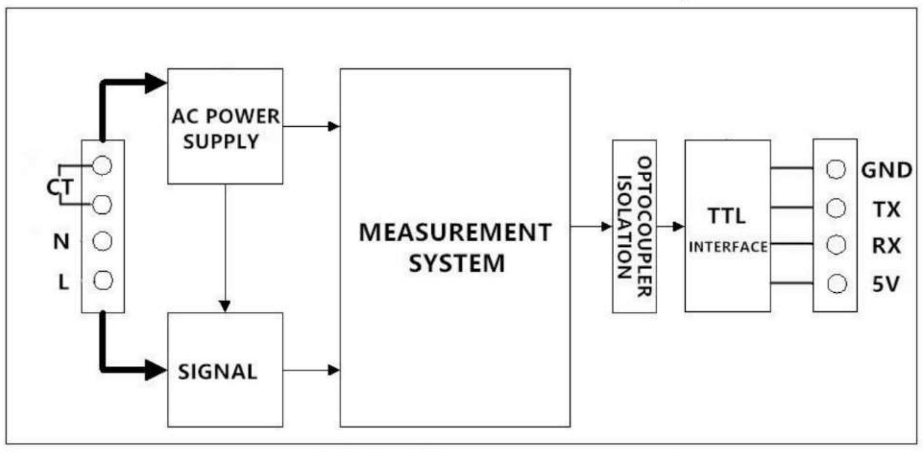
2.6 Calibration
The command format of the master to calibrate the slave is (total 6 bytes):
0xF8 + 0x41 + 0x37 + 0x21 + CRC check high byte + CRC check low byte.
Correct reply: 0xF8 + 0x41 + 0x37 + 0x21 + CRC check high byte + CRC check low byte.
Error Reply: 0xF8 + 0xC1 + Abnormal code + CRC check high byte + CRC check low byte.
It should be noted that the calibration takes 3 to 4 seconds, after the master sends the command, if the calibration is successful, it will take 3 ~ 4 seconds to receive the response from the slave.

2.7 CRC check
CRC check use 16bits format, occupy two bytes, the generator polynomial is X16 + X15 + X2 +1, the polynomial value used for calculation is 0xA001.
The value of the CRC check is a frame data divide all results of checking all the bytes except the CRC check value.

3 Functional block diagram

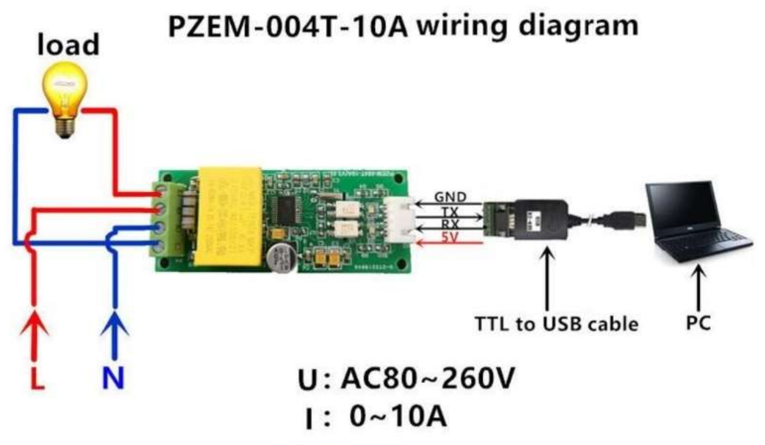


Picture 3.1 PZEM-004T-10A Functional block diagram

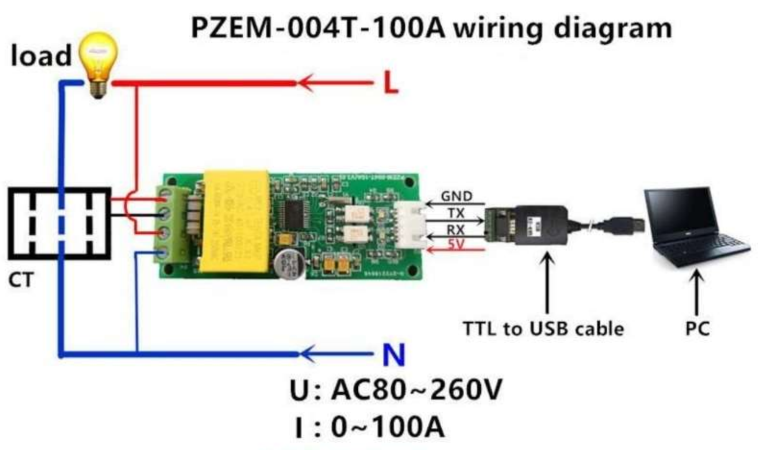


Picture 3.2 PZEM-004T-100A Functional block diagram

4 Wiring diagram



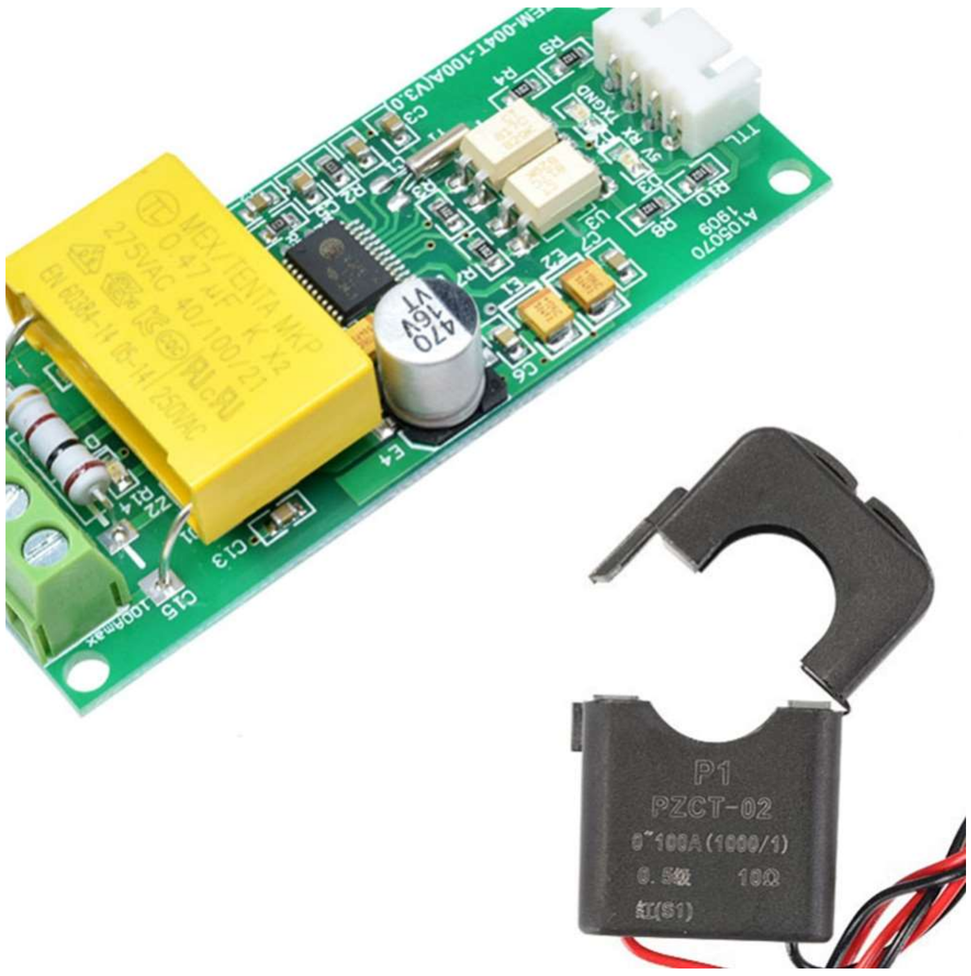
Picture 4.1 PZEM-004T-10A wiring diagram

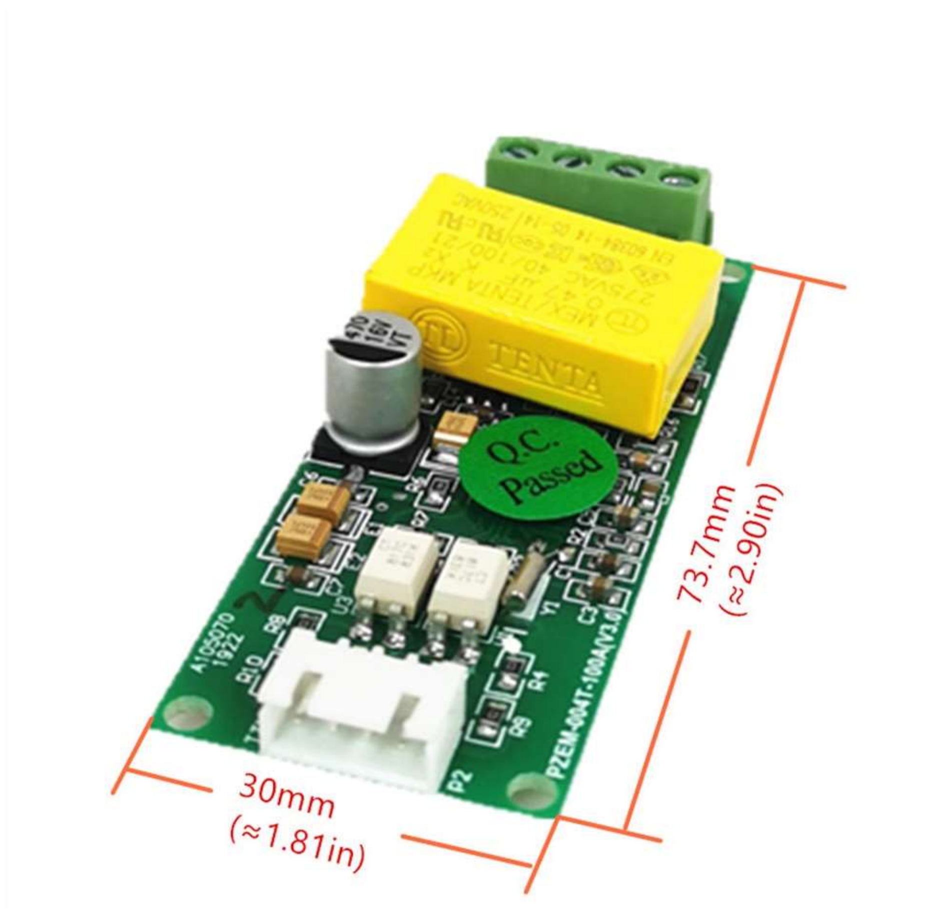


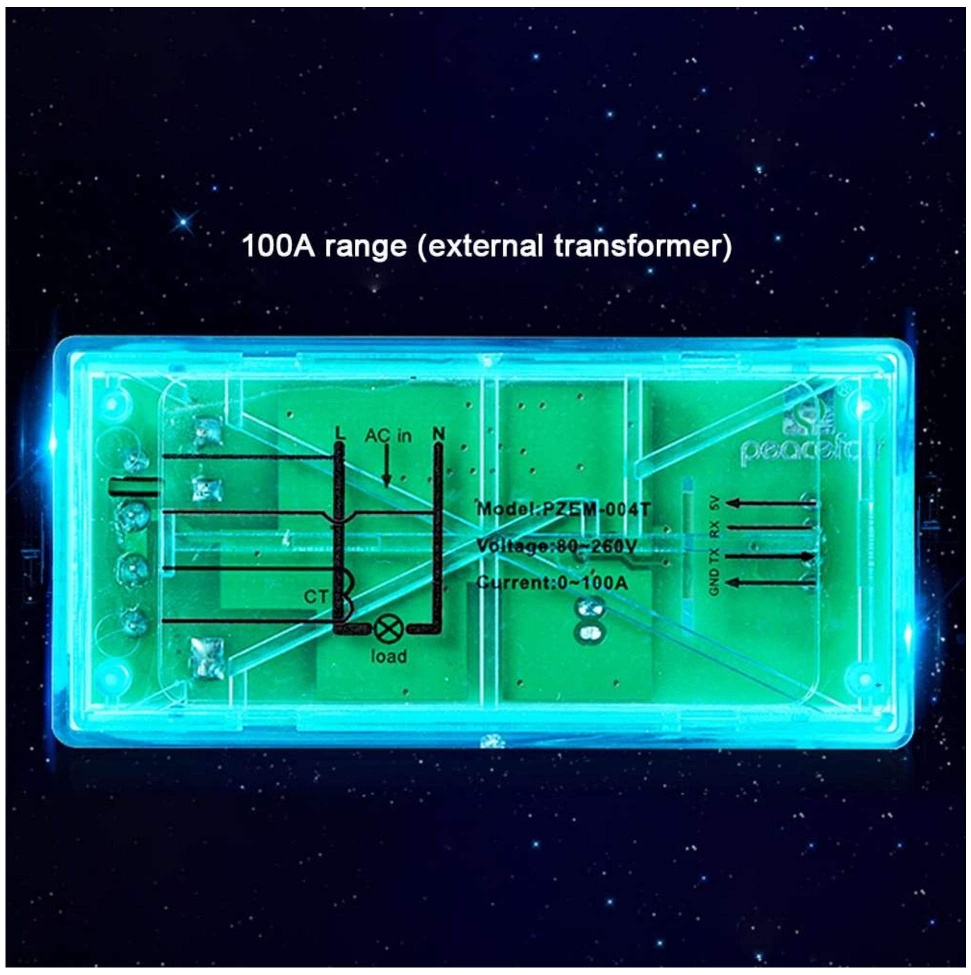
Picture 4.2 PZEM-004T-100A wiring diagram

5 Other instructi5.1The TTL interface ohich means, when come it cannot communica5.2 Working temper-20’C ~ +60’C。

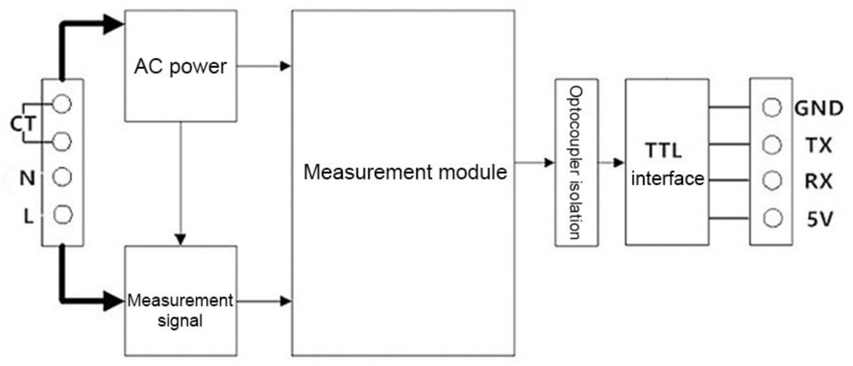
Package include:
1x Digital Multi-function Meter
1x Coil

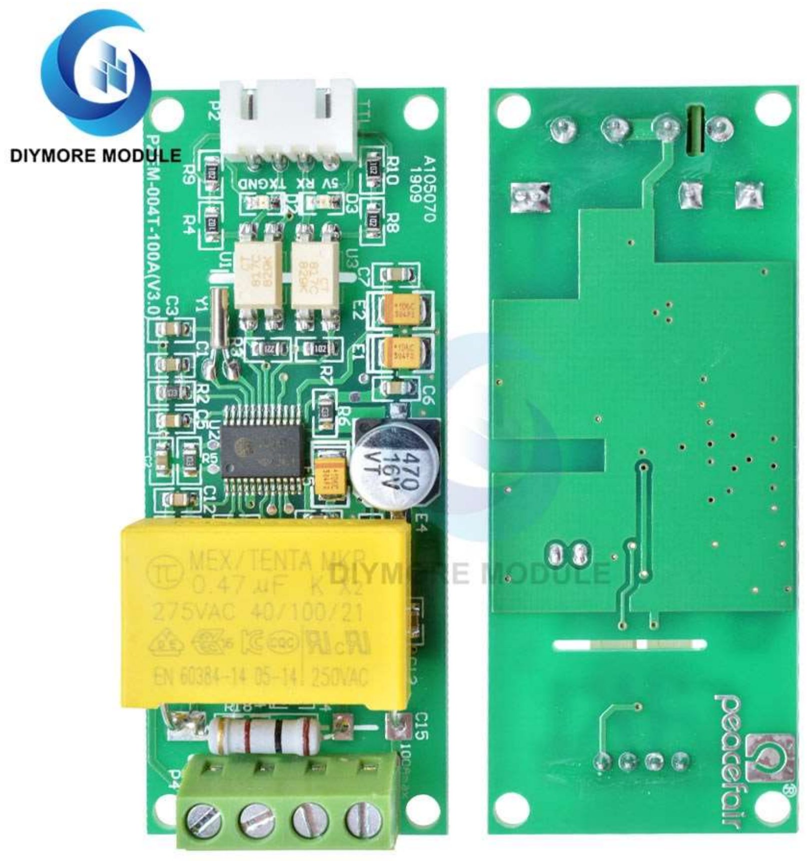

PZEM-004T-100A(V3.0)
P1
PZCT-02
0~100A(1000/1)
0.5级 10Ω
红(S1)


73.7mm
(≈2.90in)
30mm
(≈1.81in)
Q.C.
Passed
PZEM-004T-100A(V3.0)


100A range (external transformer)
L
AC in
N
CT
load
Model:PZEM-004T
Voltage:80~260V
Current:0~100A
GND TX RX 5V
peacefair

·Functional block diagram





From <https://www.aliexpress.com/item/1005002875613737.html?algo_pvid=4d2ccc31-

a121-49f1-96eb-8d55c104737c&algo_exp_id=4d2ccc31-a121-49f1-96eb-8d55c104737c-12&pdp_ext_f=%7B%22sku_id%22%3A%2212000029239124764%22%7D&pdp_npi=2%40dis%21EUR%214.11%21%21%212.3%21%21%21402103255b16590722993151214efb54%2112000029239124764%21sea>