**INTRODUCTION**

In this project, we will design our own IoT Based Electricity Energy Meter using ESP32 & monitor data on the Arduino Cloud. With the current technology, you need to go to the meter reading room and take down readings. Thus, monitoring and keeping track records of your electricity consumption is a tedious task. To automate this, we can use the Internet of Things (IOT). The IOT saves time and money by automating remote data collection. Smart Energy Meter has received quite a lot of acclaim across the globe in recent years.

**Components:**

**Current & Voltage Sensor:**

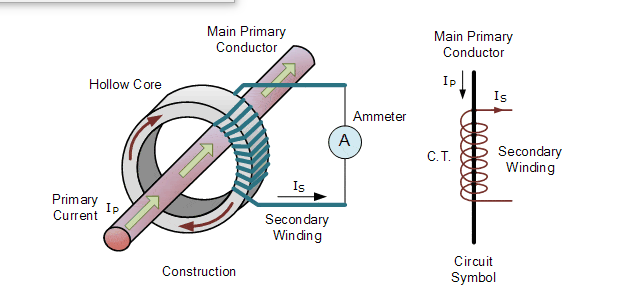
We need to select the current sensor as well as the voltage sensor so that the current & voltage can be measured and thus, we can know about the power consumption & total power consumed. The best current sensor available in the market is **SCT-013**. It is SCT-013 Non-Invasive AC Current Sensor Split Core Type Clamp Meter Sensor which can be used to measure AC current up to 30 amperes. Similarly, the best voltage sensor is the AC Voltage Sensor Module **ZMPT101B**. The ZMPT101B AC Voltage Sensor is the best where we need to measure the accurate AC voltage with a voltage transformer.

Using the SCT-013 Current Sensor & ZMPT101B Voltage Sensor, we can measure the all required parameters needed for Electricity Energy Meter. We will interface the SCT-013 Current Sensor & ZMPT101B Voltage Sensor with ESP32 WIFI Module & Send the data to Arduino cloud. The Arduino Cloud Dashboard will display the Voltage, Current, Power & total unit consumed in kWh.

**SCT 013 Current Sensor:**

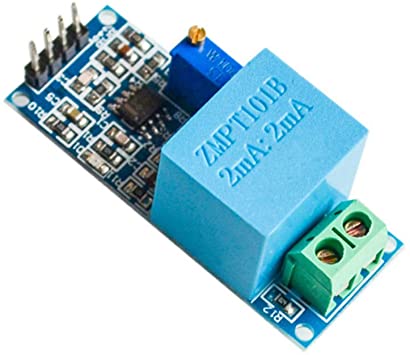
****

The SCT-013 is a Non-invasive AC Current Sensor Split Core Type Clamp Meter Sensor that can be used to measure AC current up to 30 amperes. Current transformers (CTs) are sensors are for measuring alternating current. They are particularly useful for measuring whole building electricity consumption. The SCT-013 current sensors can be clipped straight either to the live or neutral wire without having to do any high voltage electrical work.

****

Like any other transformer, a current transformer has a primary winding, a magnetic core, and a secondary winding. The secondary winding comprises many turns of fine wire housed within the casing of the transformer. It generates the DC signal between 0 and 1 V. For example, if 30A is flowing through the wire the coil will generate a signal of 1Vand if 15A is flowing through the wire then the voltage

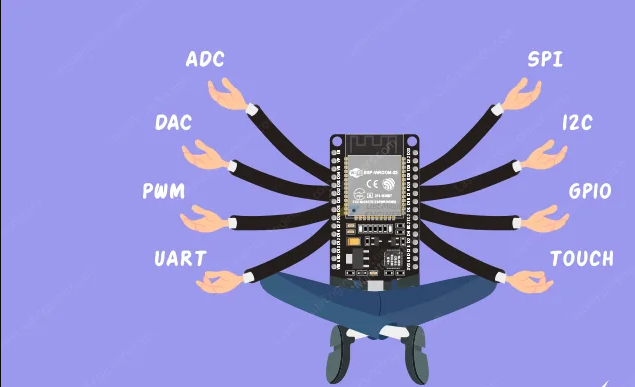
signal will be of 0.5V the through coding we can change this to the desire value of current.

**ZMPT101B Voltage Sensor:**

The ZMPT101B AC Single Phase voltage sensor module is based on a high precision ZMPT101B voltage Transformer used to measure the accurate AC voltage with a voltage transformer. This is an ideal choice to measure the AC voltage using Arduino or ESP32. The Modules can measure voltage within 250V AC voltage & the corresponding analog output can be adjusted. The module is simple to use and comes with a multi-turn trim potentiometer for adjusting and calibrating the ADC output.

**Why we are using the ESP-32 microcontroller:**

When it comes to the ESP32 chip specifications, we find that:

* The ESP32 is dual core, this means it has 2 processors.
* It has Wi-Fi and Bluetooth built-in.
* It runs 32-bit programs.
* The clock frequency can go up to 240MHz and it has a 512 kB RAM.
* It comes with built-in hall effect sensor and built-in temperature sensor.

Having the ability of WI-FI the esp-32 can easily send data to Arduino cloud and input data form the voltage and current sensor.

**Some important pins that are used in connections:**

**Vin:** inputs/outputs 5V (used to power the current and voltage sensor)

**GND pin:** Ground pin

**GPIO34:** input pin (current sensor is attached to this pin)

**GPIO35:** input pin (Voltage sensor is attached to this pin)

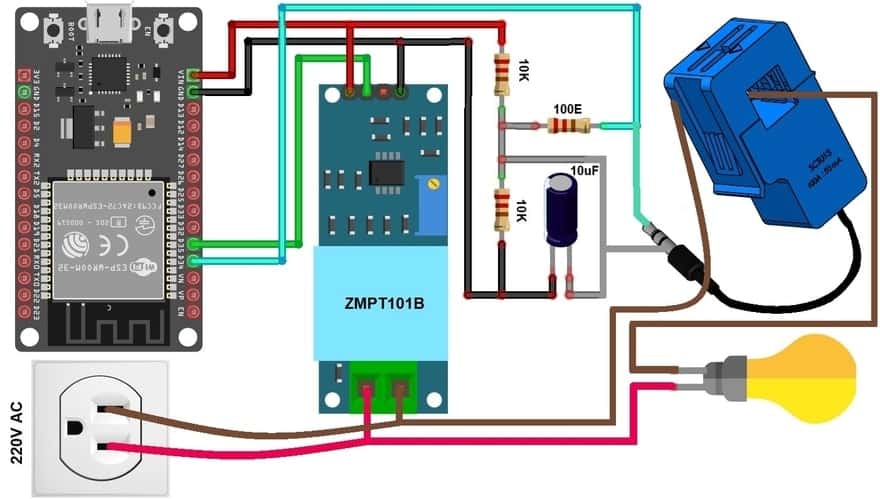


**Hardware Setup:**

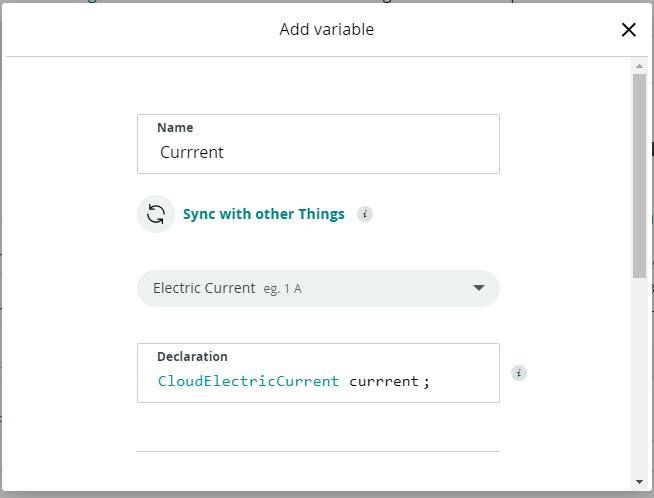
Connections in this circuit are very simple. SCT-013 Current Sensor & ZMPT101B Voltage Sensor VCC is connected to Vin of ESP32 which is a 5V Supply. The GND pin of both the modules is connected to the GND of ESP32. The output analog pin of the ZMPT101B Voltage Sensor is connected to GPIO35 of ESP32. Similarly, the output analog pin of SCT-013 Current Sensor is connected to GPIO34 of ESP32. You need a two resistor of 10K & a single resistor of 100 ohms connected along with a 10uF Capacitor.

Apart from the internal wiring, the AC wires where the current and voltage needs to measured are connected to the input AC Terminal of Voltage Sensor(Line with live and nuteral with nuteral terminal). Similarly, the current sensor clip doesn’t have any connection and a single live wire or neutral wire is inserted inside the clip part.

**Circuit Diagram:**

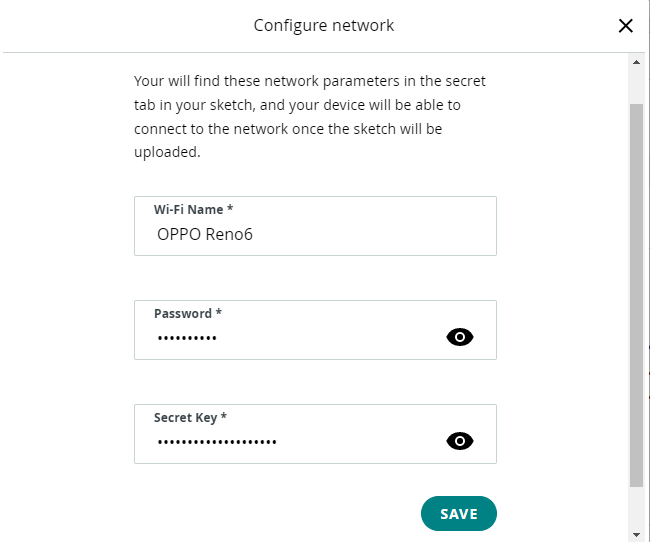


**Setting up Arduino Cloud**

First of all, we have to create variables in the Arduino cloud. Variables are the containers that will stores the values coming from current sensor and voltage sensor through the ESP 32.

First, we have to write the name and specify the type then a variable will be autogenerated in the Arduino sketch.

For this project we have to create four variables for different energy parameters like current, voltage, apparent power and energy in kWh.

After this we have to provide the WIFI cradientials so that our EPS 32 can connect to the internet and send data to the cloud. Secret Key is a code that is given to us by the cloud and it is required at te time of creating web dashboard.

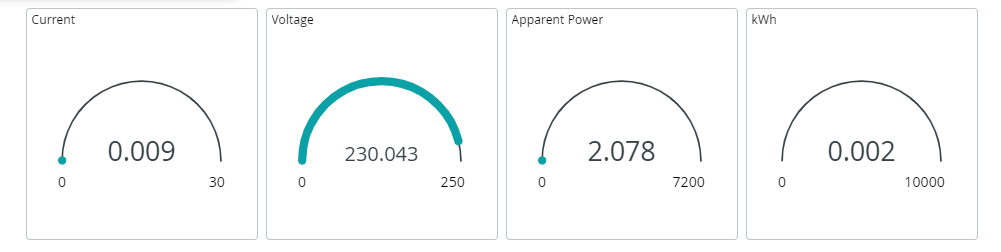
Now we have to create the web dashboard this IOT cloud has this ability that once our web dashboard is configured it automatically creates the mobile dashboard. The web dashboard of the Arduino cloud has many widgets like value, status, gauge, LED etc. But here for this project our concern is to display the values of different parameters of energy i.e., voltage, current, kWh, apparent power. That’s why we are using gauge. On the backend the gauge will require a variable whose value is to be displayed

Figure 1 Web dashboard

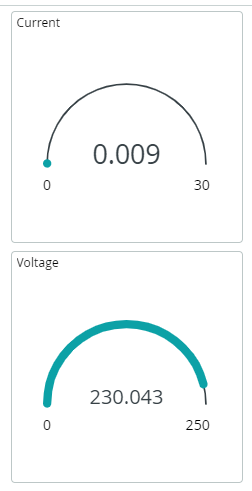
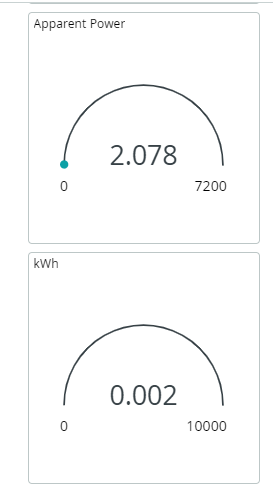


Figure 2 Mobile Dashboard

Figure 3 Mobile Dashboard

**Pre-Coding Essentials:**

For coding our esp-32 we are using the Arduino web editor which runs on the C++ language.

**Libraries:**

Some of the important libraries used in this coding are following.

**thingproperties.h:**

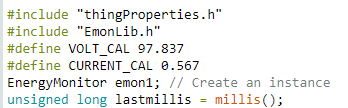
This is the library of the Arduino cloud which is automatically included when start coding in Arduino cloud IDE. All of the variable that we created in the Arduino Cloud are automatically defined in this library.

**emonlib.h:**

This is the energy monitor library that is used to convert data that is coming from the ESP 32 pin to numerical values using some mathematical calculations.

**Coding:**

**Including Libraries:**

We started coding by including mentioned above libraries and defining voltage and current calibration. Creating instance for the Energy monitor library some that we don’t have to write it over and over again. Here millis() is the time in milliseconds which starts when the code is uploaded in the

esp-32

**void setup() function:**

In the void setup function, we are initiating the communication with the Arduino cloud. We are also initiating the serial monitor so that we can display values on the serial monitor as well. After defining the voltage and current pin (34,35) we are done with the void setup function.

**void loop() function:**

following functions are done in the void loop function.

* Receiving data from the current and voltage sensor
* Output data on the serial monitor
* Output data on the Arduino IOT cloud
* Measurement of energy
* Conversion of energy in kWh
* Updating the Arduino cloud at every time the loop is executed

**COMPLETE SCRIPT**

#include "thingProperties.h"

#include "EmonLib.h"

#define VOLT\_CAL 97.837

#define CURRENT\_CAL 0.567

EnergyMonitor emon1; // Create an instance

unsigned long lastmillis = millis();

void setup() {

Serial.begin(9600); // Initialize serial and wait for port to open:

initProperties();

ArduinoCloud.begin(ArduinoIoTPreferredConnection);

setDebugMessageLevel(2);

ArduinoCloud.printDebugInfo();

emon1.current(34, CURRENT\_CAL); // Current: input pin, calibration.

emon1.voltage(35, VOLT\_CAL, 1.7);}

void loop() {

ArduinoCloud.update();

emon1.calcVI(20,2000); // Calculate all. No.of half wavelengths (crossings), time-out

float currentDraw = emon1.Irms;//extract Irms into Variable

float voltagedraw = emon1.Vrms;

float apparentPower = emon1.apparentPower;

kWh = kWh + apparentPower \* (millis() - lastmillis) / 3600000000.0;

Serial.print("Current:"); Serial.print(currentDraw);

Serial.print(" Voltage:"); Serial.print(voltagedraw);

Serial.print(" apparentPower:"); Serial.print(apparentpower);

Serial.print(" kWh:"); Serial.println(kWh);

lastmillis = millis();

current = currentDraw;

voltage = voltagedraw;

apparentpower = apparentPower; kWh = kWh;}