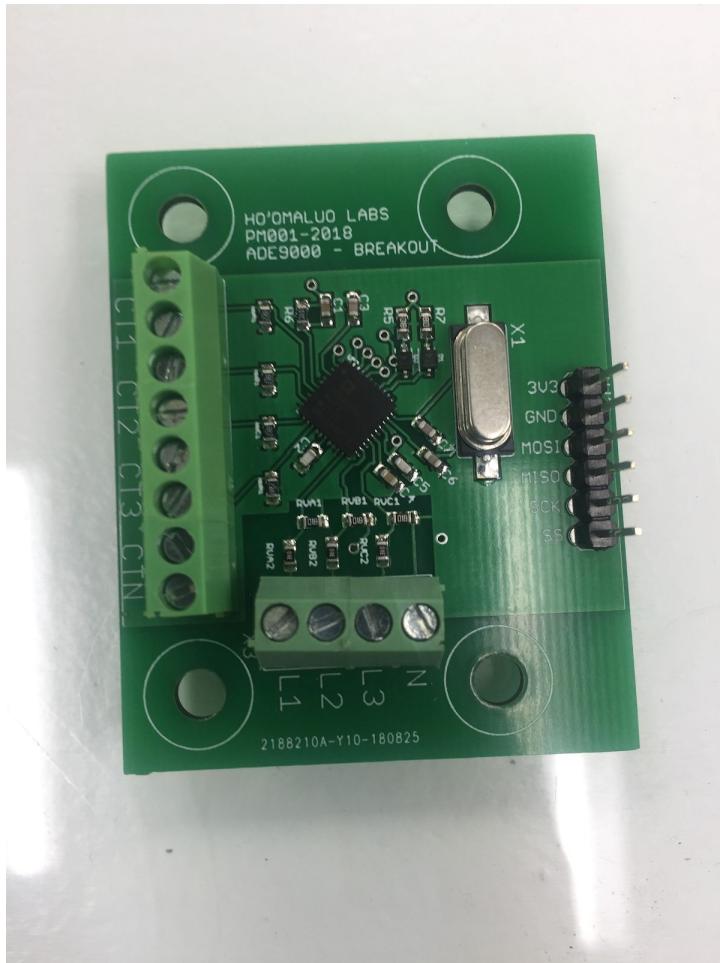


PM001-ADE9000 BREAKOUT ARDUINO (MAPLE MINI) TUTORIAL

SINGLE OR 3 PHASE POWER MONITORING IC BREAKOUT BOARD

By: Matsu



PRODUCT INFORMATION

This breakout board uses the ADE9000 power monitoring chip which uses SPI interface. This board is compatible with 32-bit microcontrollers such as the Maple Mini, ESP8266, and the

Arduino Zero or Due. This particular tutorial will use a Maple Mini as the primary microcontroller, but you should be able to adapt it to any of these other 32-bit microcontrollers using a similar approach.

DESCRIPTION OF PRODUCT FROM ANALOG DEVICES

The ADE9000 is a highly accurate, fully integrated, multiphase energy and power quality monitoring device. Superior analog performance and a digital signal processing (DSP) core enable accurate energy monitoring over a wide dynamic range. An integrated high end reference ensures low drift over temperature with a combined drift of less than $\pm 25 \text{ ppm}/^\circ\text{C}$ maximum for the entire channel including a programmable gain amplifier (PGA) and an analog-to-digital converter (ADC).

The ADE9000 offers complete power monitoring capability by providing total as well as fundamental measurements on rms, active, reactive, and apparent powers and energies. Advanced features such as dip and swell monitoring, frequency, phase angle, voltage total harmonic distortion (VTHD), current total harmonic distortion (ITHD), and power factor measurements enable implementation of power quality measurements. The $\frac{1}{2}$ cycle rms and 10 cycle rms/12 cycle rms, calculated according to IEC 61000-4-30 Class S, provide instantaneous rms measurements for real-time monitoring.

The ADE9000 offers an integrated flexible waveform buffer that stores samples at a fixed data rate of 32 kSPS or 8 kSPS, or a sampling rate that varies based on line frequency to ensure 128 points per line cycle. Resampling simplifies fast Fourier transform (FFT) calculation of at least 50 harmonics in an external processor.

The ADE9000 simplifies the implementation of energy and power quality monitoring systems by providing tight integration of acquisition and calculation engines. The integrated ADCs and DSP engine calculate various parameters and provide data through user accessible registers or indicate events through interrupt pins. With seven dedicated ADC channels, the ADE9000 can be used on a 3-phase system or up to three single-phase systems. It supports current transformers (CTs) or Rogowski coils for current measurements. A digital integrator eliminates a discrete integrator required for Rogowski coils.

The ADE9000 absorbs most complexity in calculations for a power monitoring system. With a simple host microcontroller, the ADE9000 enables the design of standalone monitoring or protection systems, or low cost nodes uploading data into the cloud.

MAPLE MINI

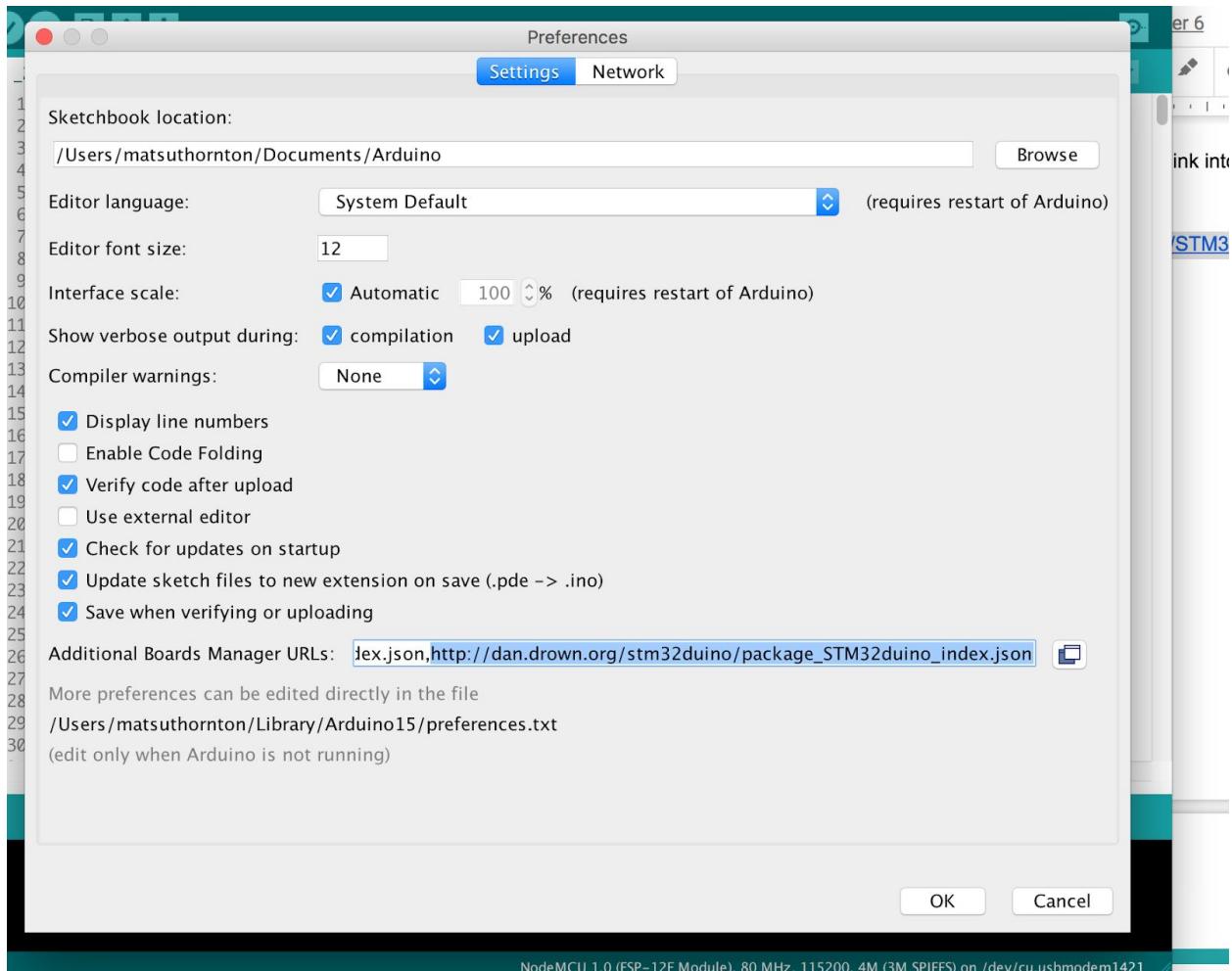
For this tutorial we will be using the Maple Mini microcontroller from Leaf Labs which is a 32-bit microcontroller. Please note that the Arduino UNO and many other boards are 8-bit microcontrollers and will not work using this tutorial, however, there are many 32-bit boards that will work with this board such as the Arduino Zero, Arduino Due, and ESP8266.



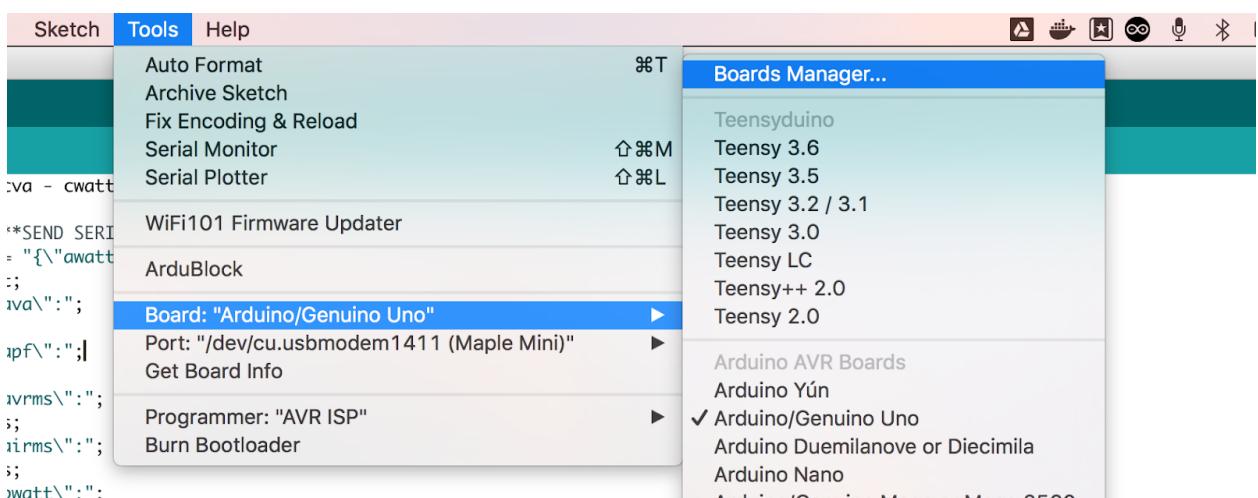
INSTALLING BOARDS

Open Arduino IDE and go to the preferences pane. Paste this link into the additional boards manager URLs:

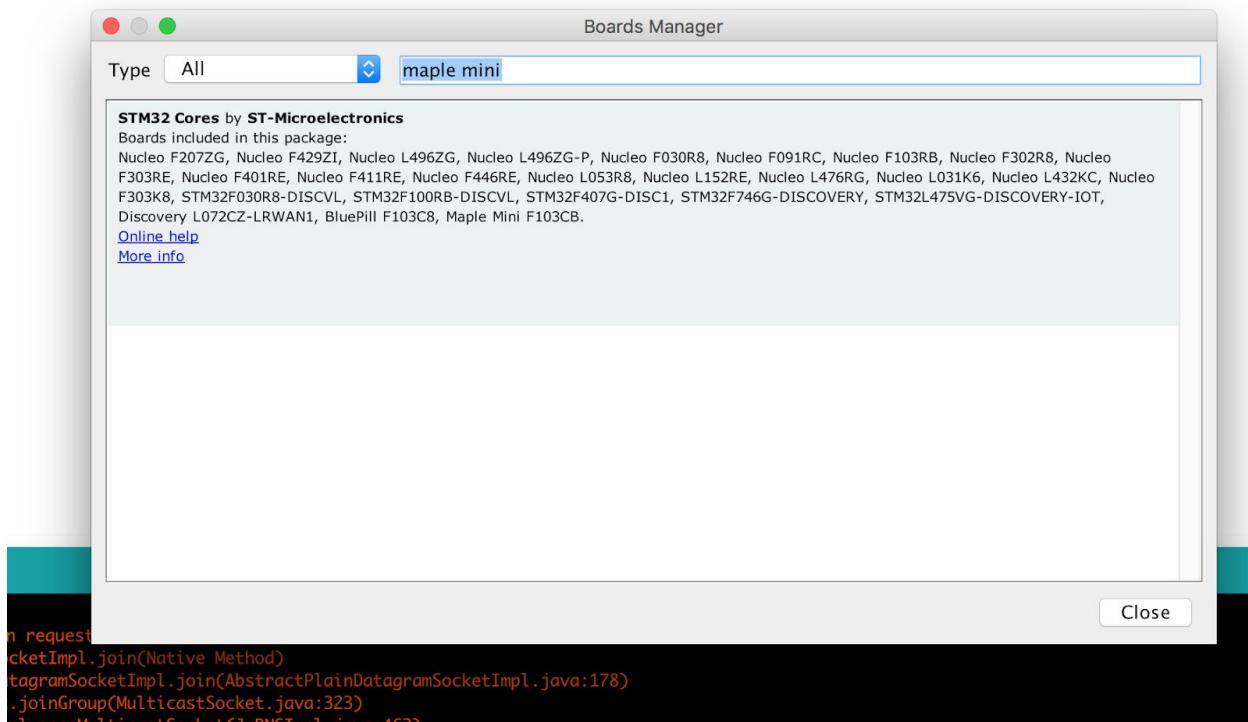
http://dan.drown.org/stm32duino/package_STM32duino_index.json



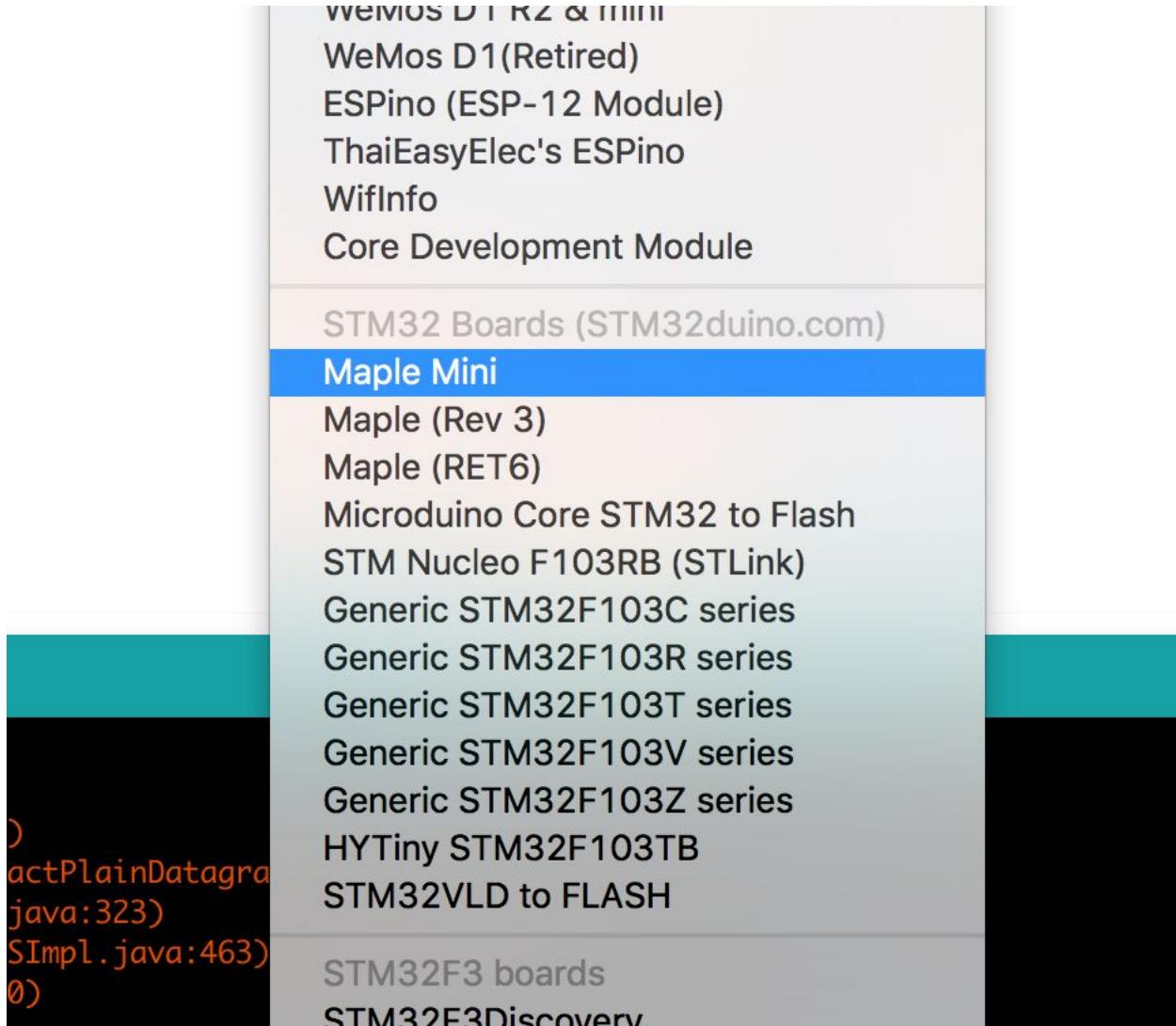
Click ok and go to the boards manager under the tools->Board menu:



Search for the maple mini board:



Install the STM32 cores and close the boards manager. You should now be able to see the maple mini in the available boards. Select the maple mini:



INSTALLING LIBRARIES

Navigate to:

https://github.com/tygermeow/hoomaluo_PM001-ADE9000

And download the zip file:

[tygermeow / hoomaluo_PM001-ADE9000](#)

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Arduino Libraries for Hoomaluo Labs PM001-ADE9000 Breakout Board

Add topics

9 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

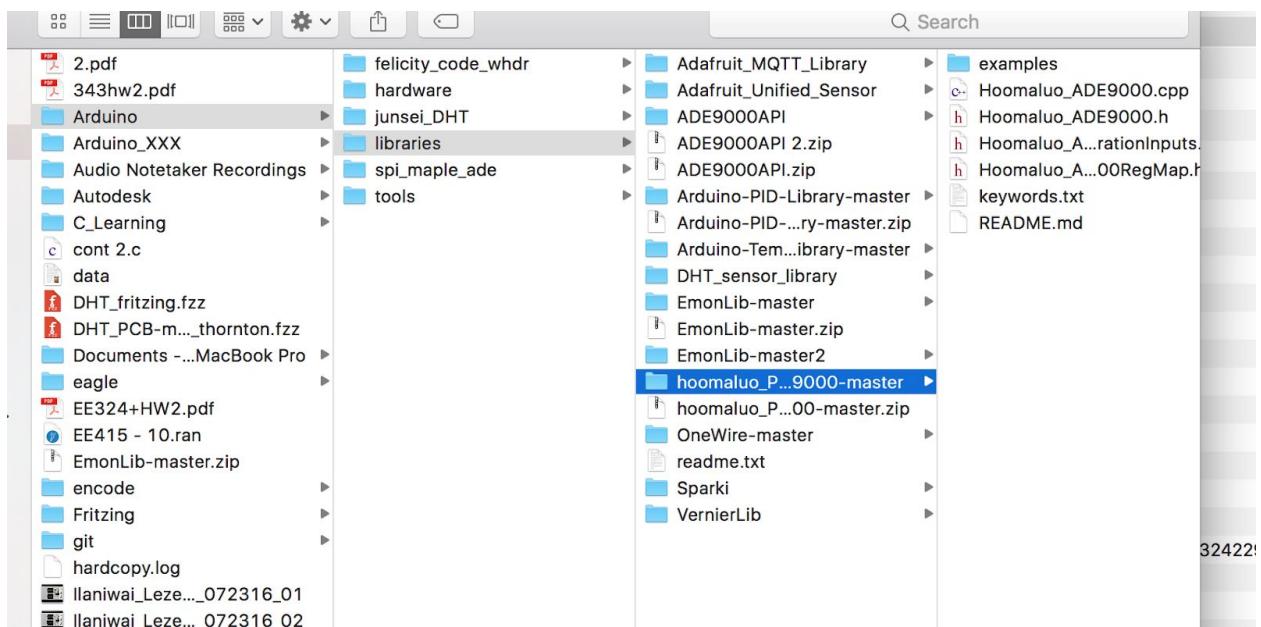
tygermeow Add files via upload
examples/Simple_Power_Acquisition Add files via upload
Hoomaluo_ADE9000.cpp Add files via upload
Hoomaluo_ADE9000.h Add files via upload
Hoomaluo_ADE9000CalibrationInputs.h Add files via upload
Hoomaluo_ADE9000RegMap.h Add files via upload
README.md first commit 14 minutes ago
keywords.txt Add files via upload 14 minutes ago

Clone with HTTPS Use SSH
 Use Git or checkout with SVN using the web URL.
<https://github.com/tygermeow/hoomaluc>

Open in Desktop Download ZIP 14 minutes ago

README.md

Unzip the folder into your Documents->Arduino->Libraries folder:



Restart Arduino IDE and open the simple power monitor example from the examples folder:

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The Examples menu is currently selected, showing a list of built-in examples categorized by board and library. The 'Simple_Power_Acquisition' example is highlighted in blue at the bottom of the list. The code editor on the left contains a sketch for the Hoomaluo library, which includes includes for SPI, OneWire, LiquidCrystal, stdint.h, math.h, pins.h, DallasTemp, and IRcarrier, along with variables for sendIR, sigTir, current_time, last_time, acquire_time, and update_timelcd. The Arduino Version is listed as unknown. The status bar at the bottom shows the file name PM001_A.ino and the version Arduino 1.8.6 Windows.

```
5 */
6
7 //*****INCLUDES*****
8 #include <SPI.h>
9 #include <Hoomaluo.h>
10 #include <OneWire.h>
11 #include <LiquidCrystal.h>
12 #include <stdint.h>
13 #include <math.h>
14 #include "pins.h"
15 #include <DallasTemp.h>
16 #include <DallasTemperature.h>
17 #include <SoftwareSerial.h>
18 #include "IRcarrier.h"
19
20
21 //*****GLOBAL VARS*****
22 uint8_t sendIR[199];
23 unsigned long sigTir;
24 float current_time;
25 float last_time = 0;
26 float acquire_time = 0;
27 float current_timelcd = 0;
28 float last_timelcd = 0;
29 float update_timelcd = 0;
30 double dcv_accum = 0;
31 double dci_accum = 0;
```

Arduino Version: unknown

File Edit Sketch Tools Help

Built-in Examples

- 01.Basics
- 02.Digital
- 03.Analog
- 04.Communication
- 05.Control
- 06.Sensors
- 07.Display
- 08.Strings
- 09.USB
- 10.StarterKit_BasicKit
- 11.ArduinoISP
- Teensy

Examples for any board

- Bridge
- Esploра
- Ethernet
- Firmata
- GSM
- LiquidCrystal
- Robot Control
- Robot Motor
- SD
- Servo
- SpacebrewYun
- Stepper
- Temboo
- RETIRED

Examples for Arduino/Genuino Uno

- EEPROM
- SoftwareSerial
- SPI
- Wire

Examples from Custom Libraries

- Adafruit MQTT Library
- ADE9000API
- DallasTemperature
- DHT sensor library
- EmonLib
- hoomaluo_PM001-ADE9000-master
- OneWire
- PID

CONNECTING SPI INTERFACE

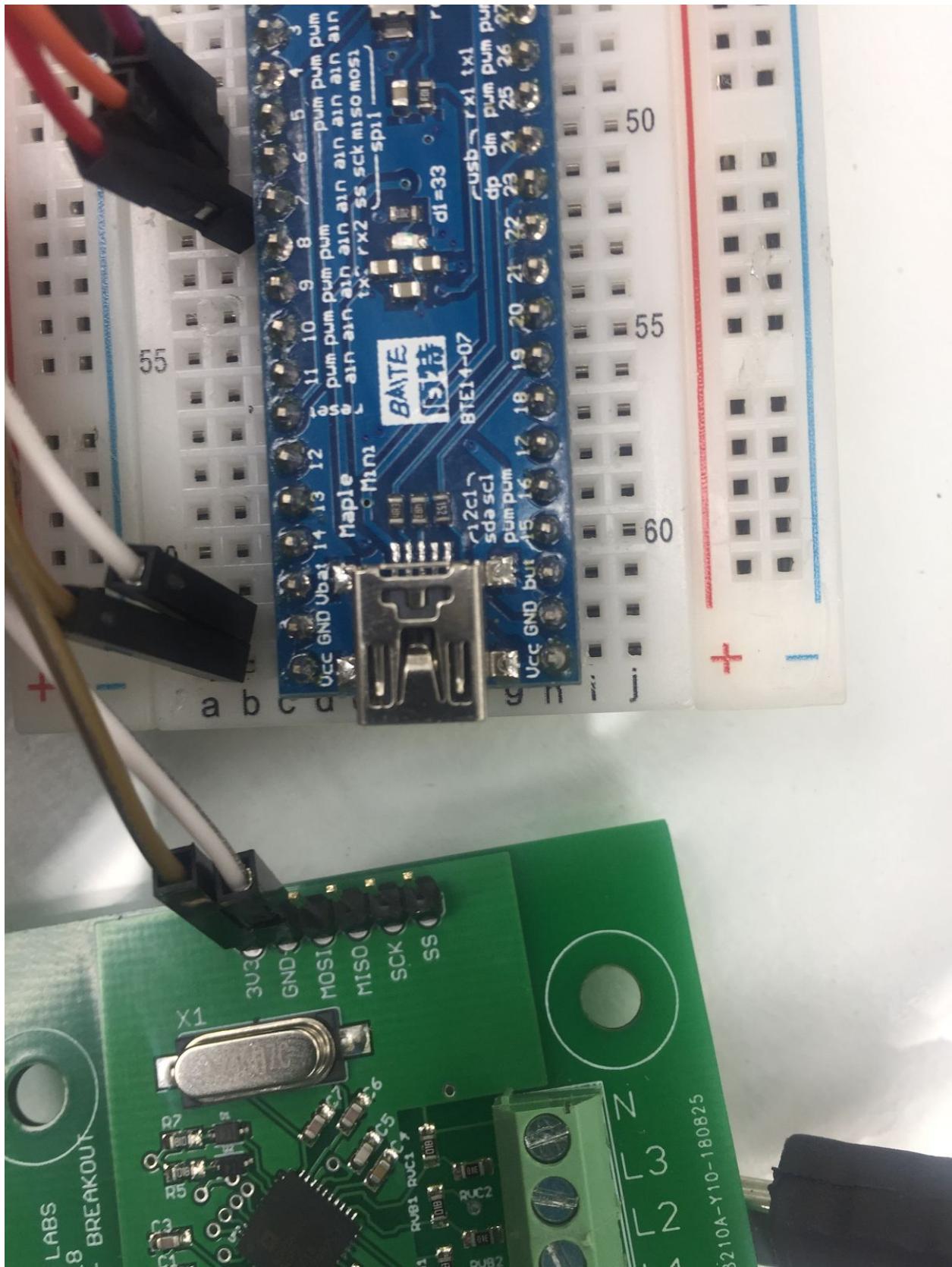
Connect the PM001 to 3.3V power, ground and the SPI1 interface of the maple Mini.

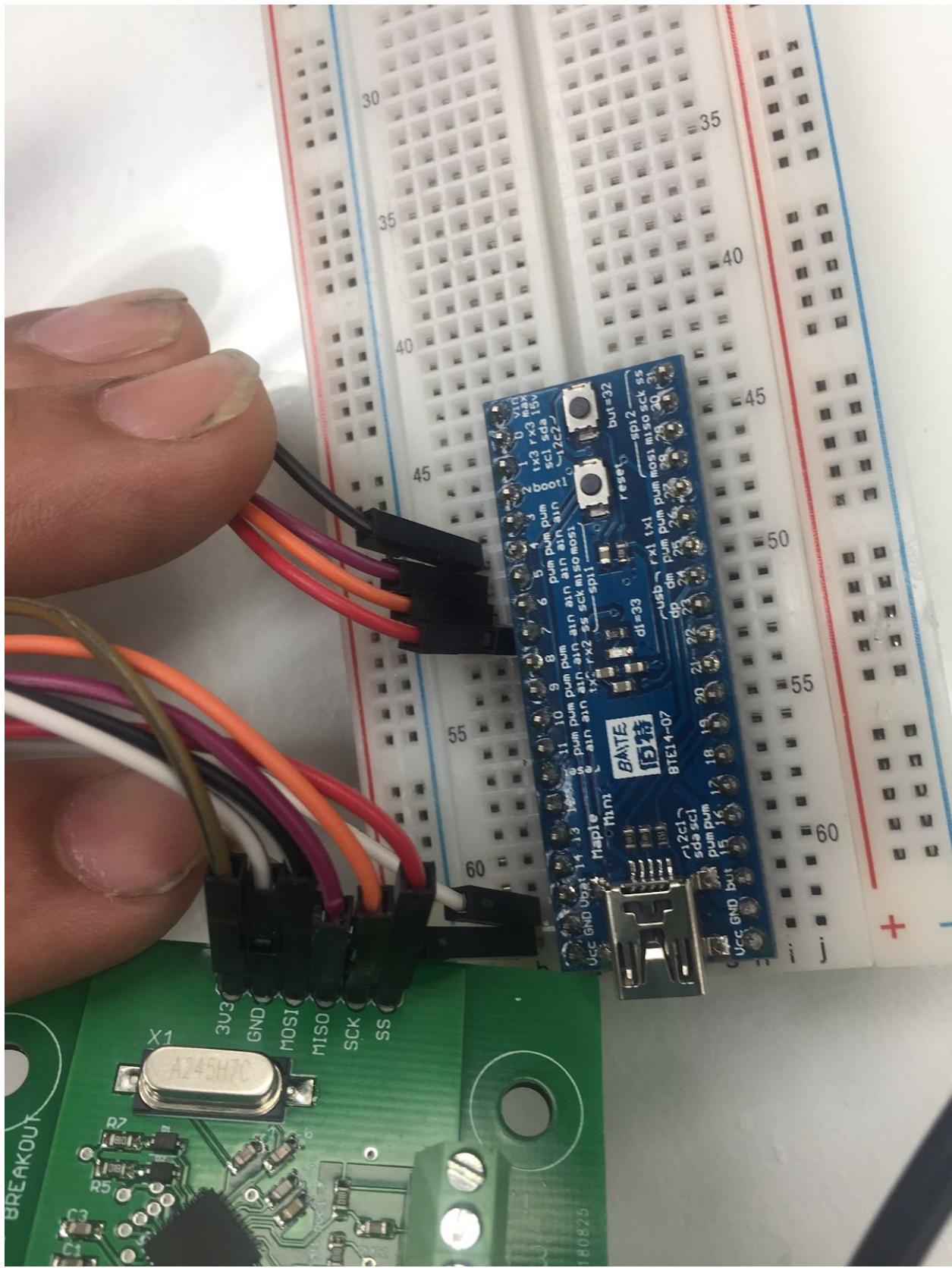
Maple Mini

Pin 7	->	SS
Pin 6	->	SCK
Pin 5	->	MISO
Pin 4	->	MOSI
VCC	->	3V3
GND	->	GND

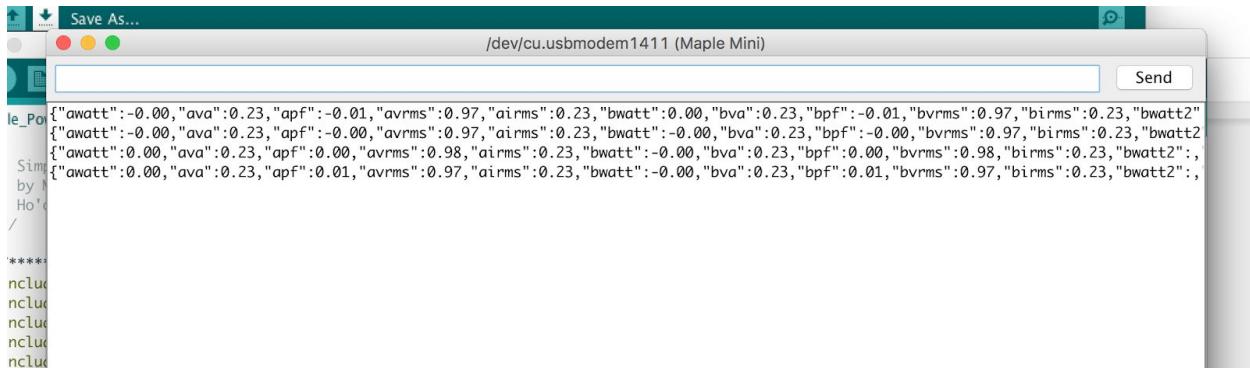
PM001







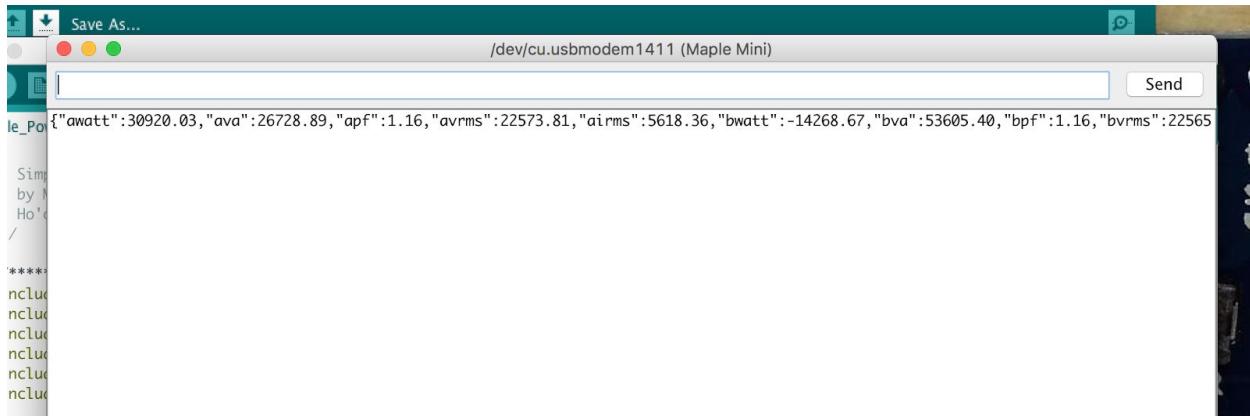
Now test that the chip is working by uploading the sketch to the maple mini. After uploading, press the reset button on the maple mini twice then open the serial monitor.



The screenshot shows a terminal window titled "Save As..." with the path "/dev/cu.usbmodem1411 (Maple Mini)". The window contains several lines of JSON data and some placeholder text. The JSON data includes fields like awatt, ava, apf, avrms, airms, bwatt, bva, bpf, bvrms, and birms. The placeholder text consists of five lines of "nclue".

```
{"awatt": -0.00, "ava": 0.23, "apf": -0.01, "avrms": 0.97, "airms": 0.23, "bwatt": 0.00, "bva": 0.23, "bpf": -0.01, "bvrms": 0.97, "birms": 0.23, "bwatt2": {"awatt": -0.00, "ava": 0.23, "apf": -0.00, "avrms": 0.97, "airms": 0.23, "bwatt": -0.00, "bva": 0.23, "bpf": -0.00, "bvrms": 0.97, "birms": 0.23, "bwatt2": {"awatt": 0.00, "ava": 0.23, "apf": 0.00, "avrms": 0.98, "airms": 0.23, "bwatt": -0.00, "bva": 0.23, "bpf": 0.00, "bvrms": 0.98, "birms": 0.23, "bwatt2": {"awatt": 0.00, "ava": 0.23, "apf": 0.01, "avrms": 0.97, "airms": 0.23, "bwatt": -0.00, "bva": 0.23, "bpf": 0.01, "bvrms": 0.97, "birms": 0.23, "bwatt2": **** nclue nclue nclue nclue nclue
```

If your output looks like this:



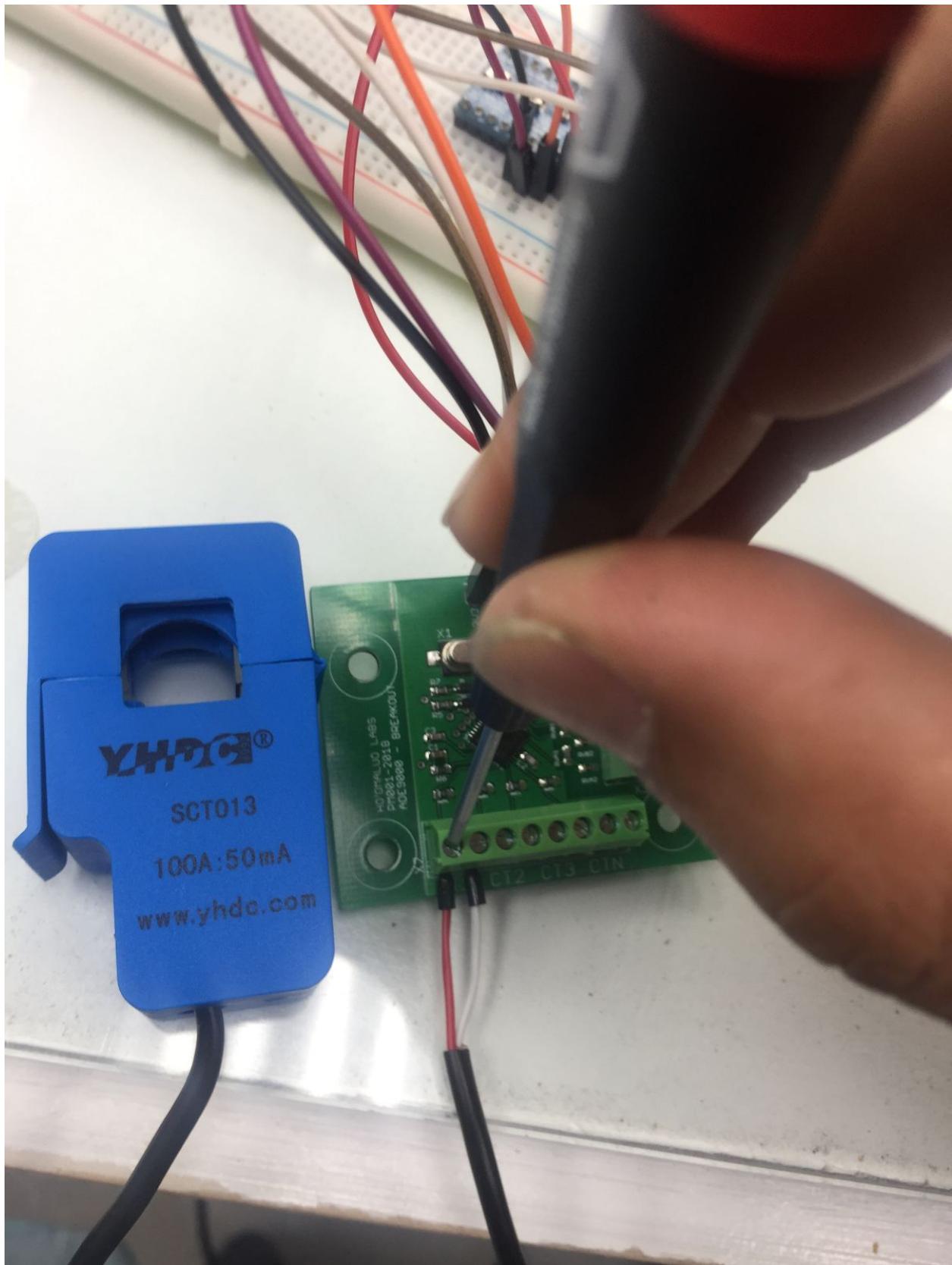
The screenshot shows a terminal window titled "Save As..." with the path "/dev/cu.usbmodem1411 (Maple Mini)". The window contains several lines of JSON data and some placeholder text. The JSON data includes fields like awatt, ava, apf, avrms, airms, bwatt, bva, bpf, bvrms, and birms. The placeholder text consists of five lines of "nclue".

```
{"awatt": 30920.03, "ava": 26728.89, "apf": 1.16, "avrms": 22573.81, "airms": 5618.36, "bwatt": -14268.67, "bva": 53605.40, "bpf": 1.16, "bvrms": 22565 Simple by N Ho' / **** nclue nclue nclue nclue nclue
```

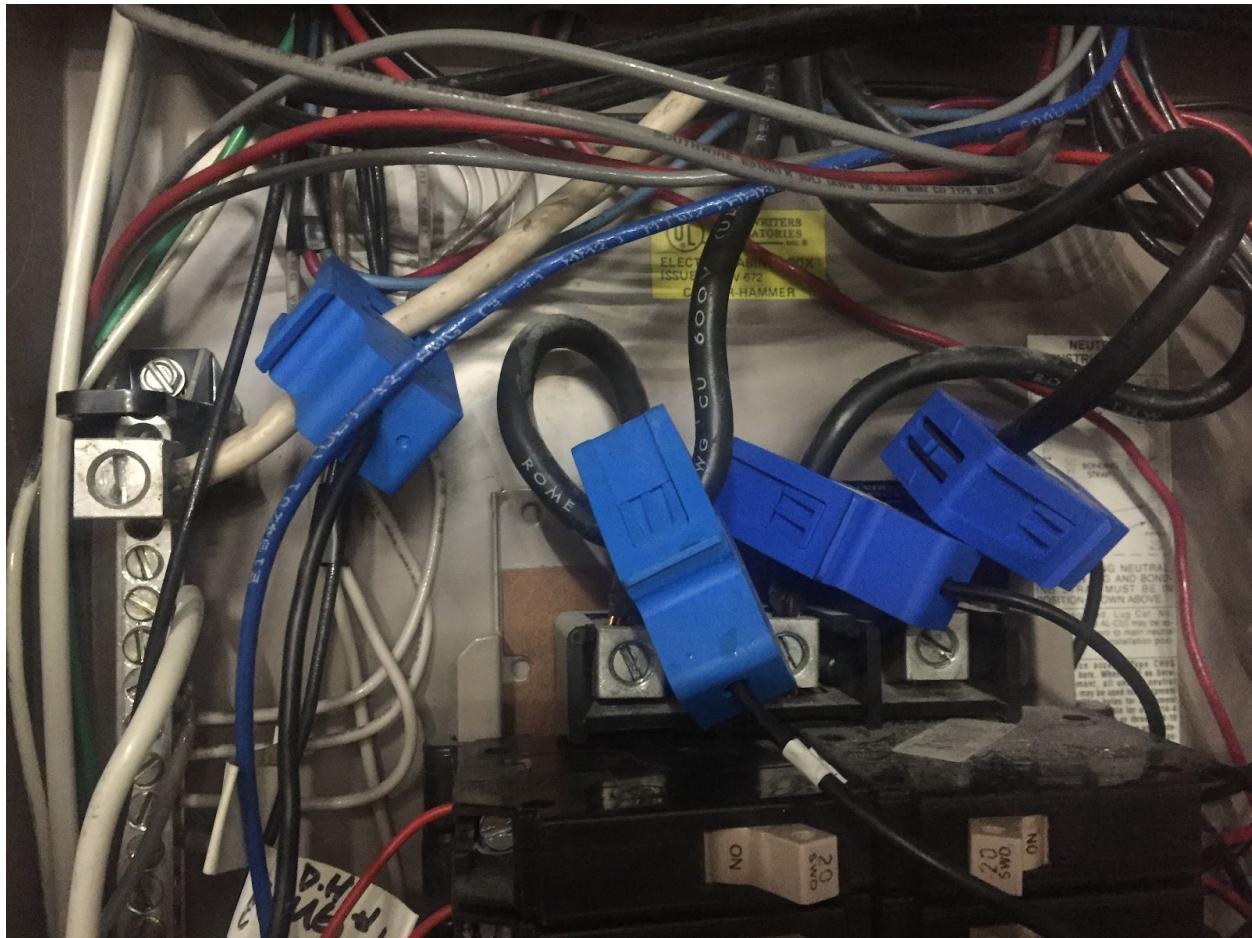
There is a problem with the SPI connection. Check that all of your connections are correct and try again.

CONNECTING CURRENT TRANSDUCERS (CT)

You can use up to 4 CTs to monitor current in and line conductors and the neutral conductor.



A typical situation may look something like this:



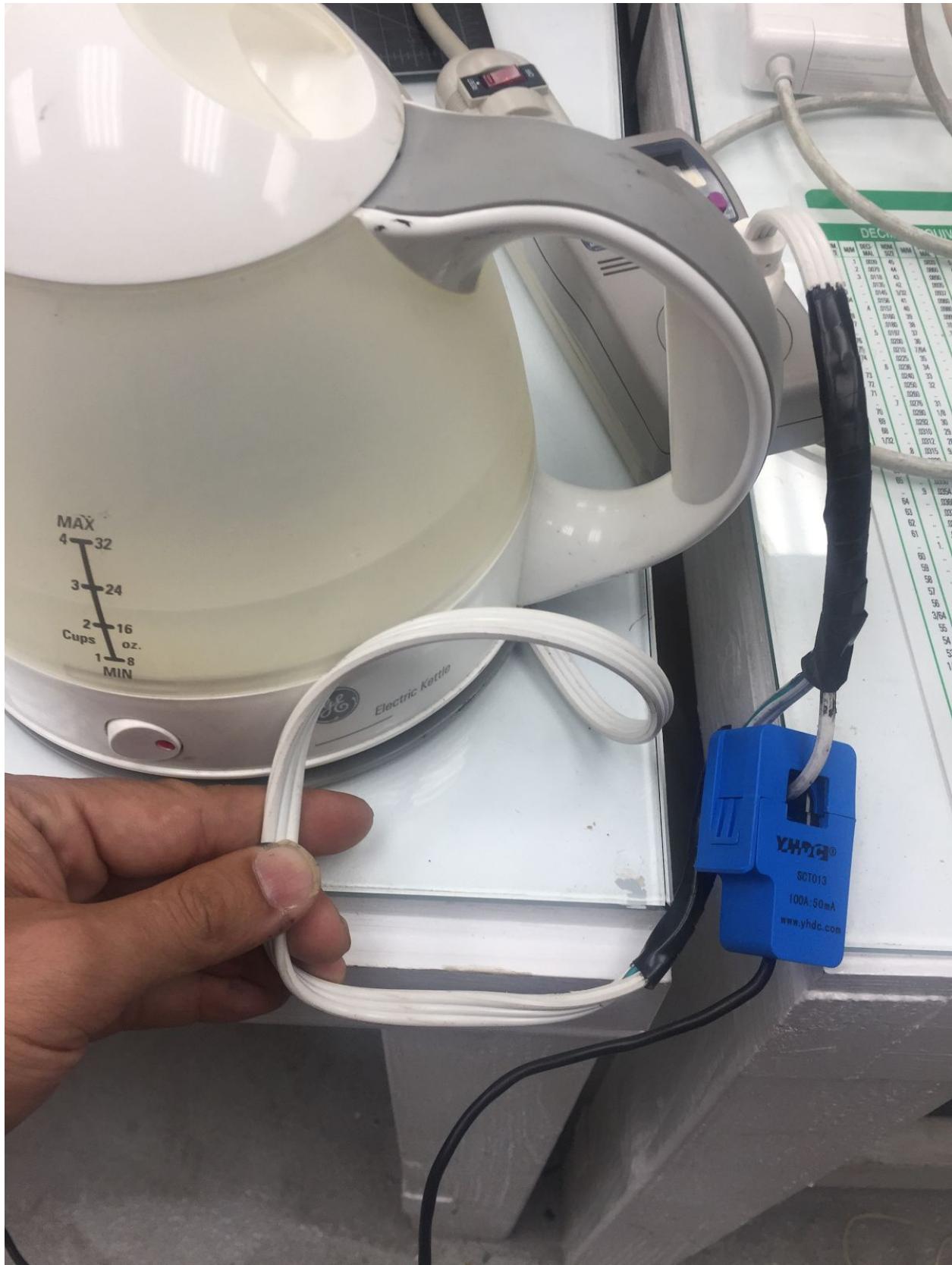
With one CT for each line in and one for the neutral line.

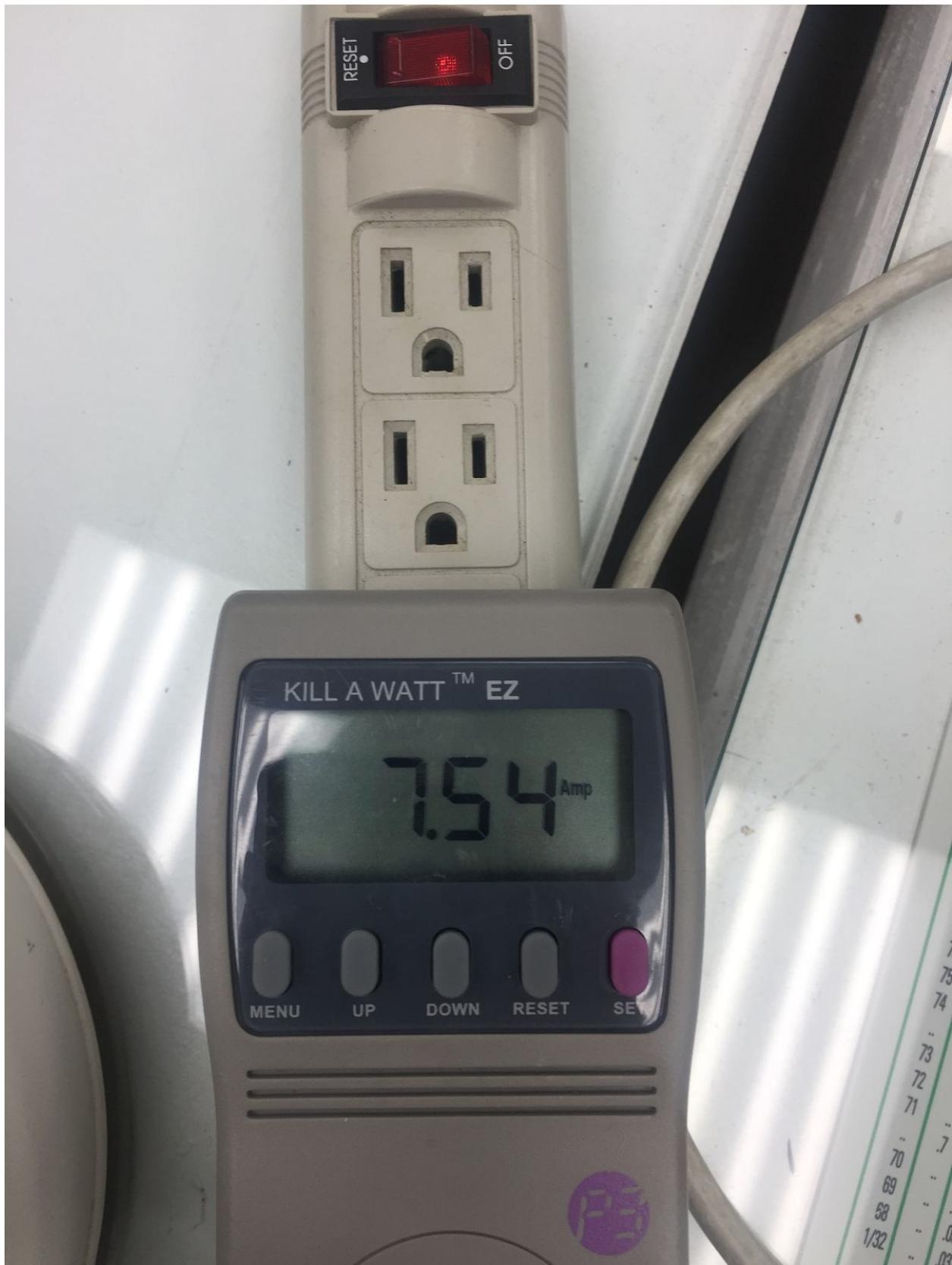
Remember that the burden resistors are 8.2 ohms so if you use your own CTs you need to make sure that the maximum voltage swing at maximum current will be under 500 mV.

$$\text{Peak Current} * \text{CT ratio} * 8.2 < 0.5 \text{ Volts}$$

CALIBRATION

You can calibrate your device by using a quality clamp meter or other current measuring device.



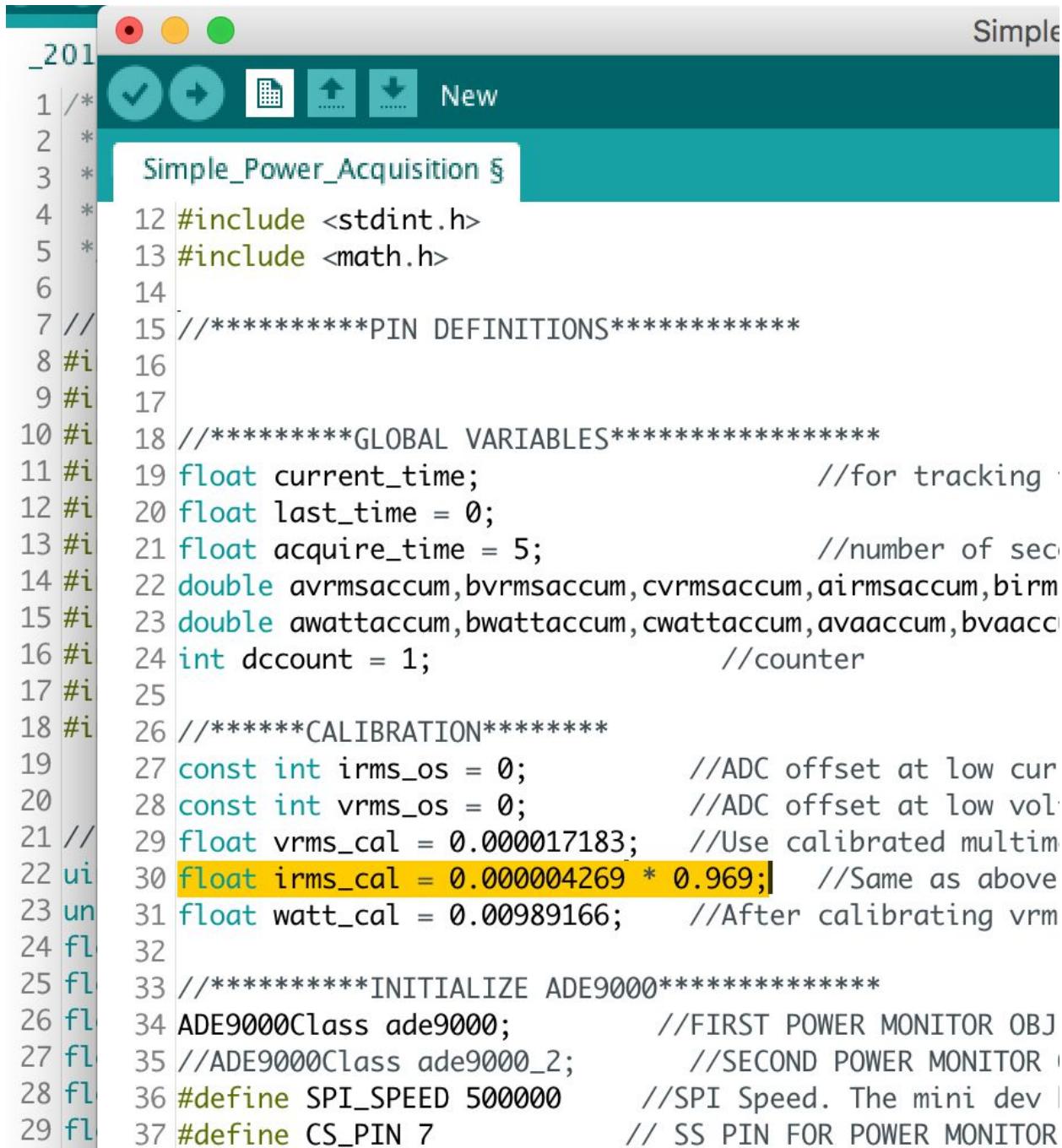


```
/ [{"awatt":0.14,"ava":7.58,"apf":0.02,"avrms":0.97,"airms":7.76,"bwatt":0.00,"bva":0.23,"bpf":0.02,"bvrms":0.98,"birms":0.23,"bwatt2":0.00,"birms2":0.23}, {"awatt":0.09,"ava":7.60,"apf":0.01,"avrms":0.97,"airms":7.78,"bwatt":0.00,"bva":0.23,"bpf":0.01,"bvrms":0.97,"birms":0.23,"bwatt2":0.00,"birms2":0.23}, {"awatt":0.07,"ava":7.61,"apf":0.01,"avrms":0.97,"airms":7.78,"bwatt":0.00,"bva":0.23,"bpf":0.01,"bvrms":0.98,"birms":0.23,"bwatt2":0.00,"birms2":0.23}, {"awatt":0.03,"ava":4.15,"apf":0.01,"avrms":0.97,"airms":4.19,"bwatt":0.00,"bva":0.23,"bpf":0.01,"bvrms":0.97,"birms":0.23,"bwatt2":0.00,"birms2":0.23}],
```

Read the Current from both your serial monitor (airms here) and the measuring device. Notice that here we see a slight difference in readings. Perform the following calculation:

Measuring Device Reading/PM001 Reading = $7.54/7.78 = 0.969$

Now find the following line in the example sketch and update by multiplying the calibration factor `irms_cal` by this number:



```
_201 Simple
1 /*
2 *
3 * Simple_Power_Acquisition §
4 */
5
6 // ****PIN DEFINITIONS*****
7 #include <stdint.h>
8 #include <math.h>
9
10 // ****GLOBAL VARIABLES*****
11 float current_time; //for tracking
12 float last_time = 0;
13 float acquire_time = 5; //number of seconds
14 double avrmsaccum,bvrmsaccum,cvrmsaccum,airmsaccum,birmsaccum;
15 double awattaccum,bwattaccum,cwattaccum,avaaccum,bvaaccum;
16 int dccount = 1; //counter
17
18 // ****CALIBRATION*****
19 const int irms_os = 0; //ADC offset at low current
20 const int vrms_os = 0; //ADC offset at low voltage
21 float vrms_cal = 0.000017183; //Use calibrated multimeter
22 float irms_cal = 0.000004269 * 0.969; //Same as above
23 float watt_cal = 0.00989166; //After calibrating vrm
24
25 // ****INITIALIZE ADE9000*****
26 ADE9000Class ade9000; //FIRST POWER MONITOR OBJ
27 ADE9000Class ade9000_2; //SECOND POWER MONITOR
28 #define SPI_SPEED 500000 //SPI Speed. The mini dev
29 #define CS_PIN 7 // SS PIN FOR POWER MONITOR
```

VOLTAGE MEASUREMENT

Be very careful working with dangerous voltages present in line voltage systems. Connect Fuseholders with 1 Amp fuses to L1, L2 and L3, and white neutral breakout to N.

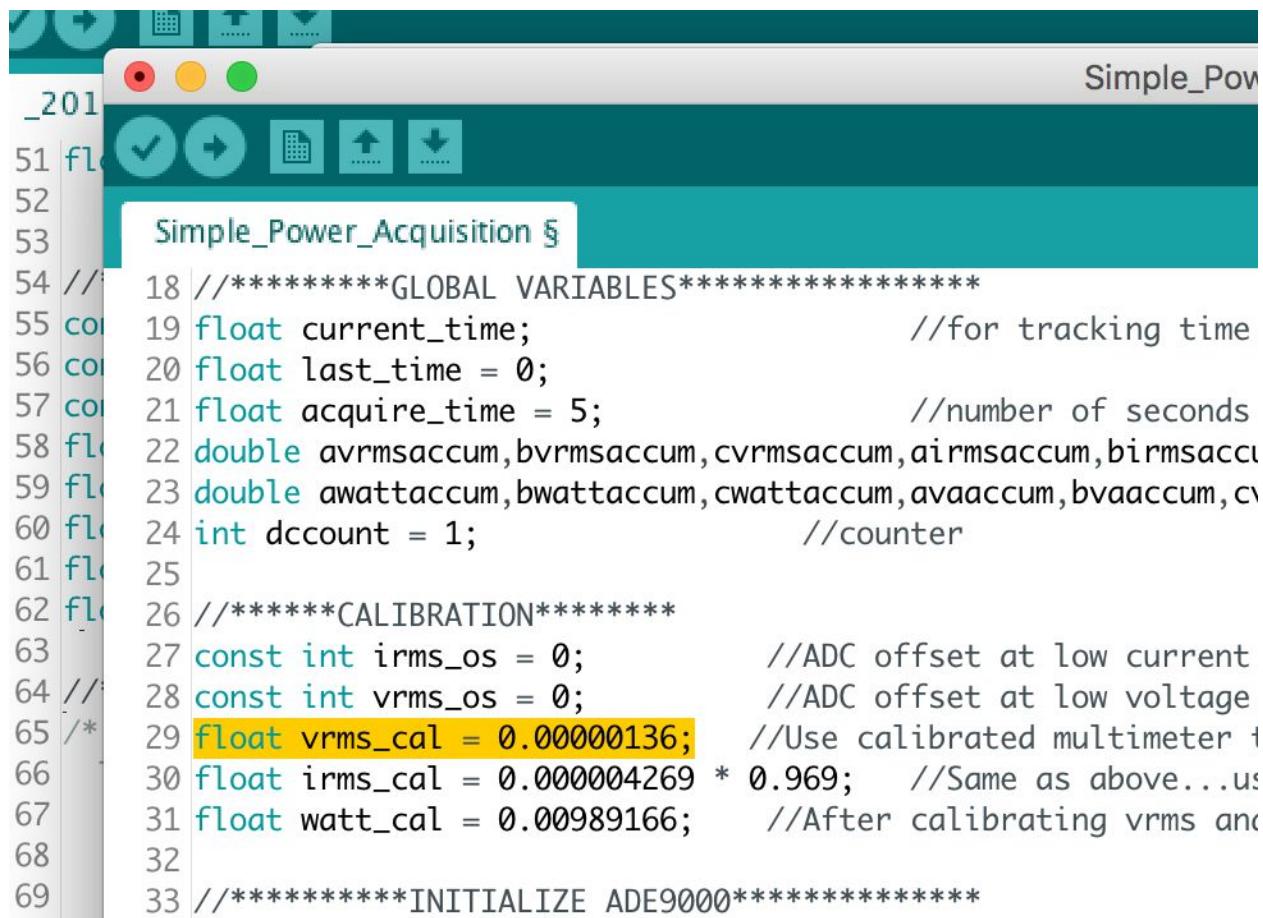


Carefully connect the neutral line to the white wire and L1, L2, and L3 to respective phase legs.

Measure the voltage with a voltmeter and with the PM001. Compare and perform same calibration as before with current:

Measuring Device Reading/PM001 Reading = Calibration coefficient

Multiply by the vrms_cal line:



The screenshot shows a software interface with a code editor window titled "Simple_Pow". The code is a C-like program for power acquisition. It includes comments for global variables, calibration constants (vrms_cal), and initialization of ADE9000. The code editor has syntax highlighting and standard file operations (Save, Open, Find, Replace) at the top.

```
Simple_Pow
_201
51 float
52
53 Simple_Power_Acquisition §
54 //*****
55 const
56 const
57 const
58 float
59 float
60 float
61 float
62 float
63 const int
64 //*
65 /*
66 float vrms_cal = 0.00000136; //Use calibrated multimeter i
67 float irms_cal = 0.000004269 * 0.969; //Same as above...us
68 float watt_cal = 0.00989166; //After calibrating vrms and
69 //*****INITIALIZE ADE9000*****
```

CALIBRATING POWER READINGS

Now you can connect a load to your device and take some readings. Note the values for:

VA
IRMS

VRMS

```
"cwatt":884.39,"cva":885.95,"cpf":1.00,"cvrms":113.21,"cirms":7.55}  
watt":883.90,"cva":885.50,"cpf":1.00,"cvrms":113.18,"cirms":7.54}
```

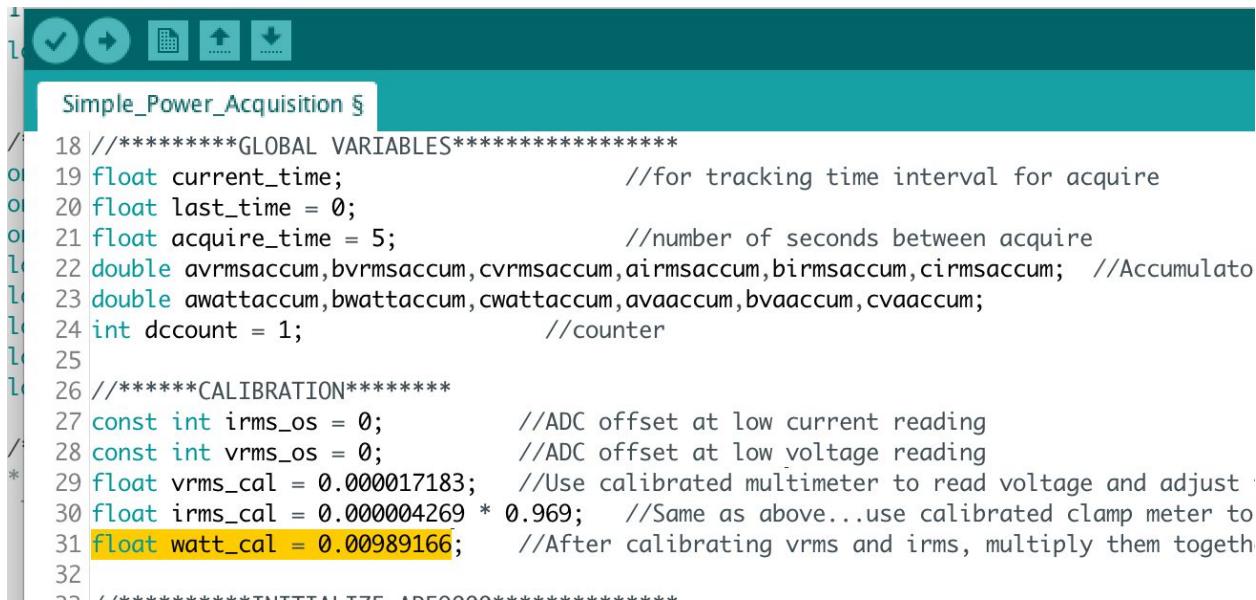
Now multiply :

$$X_{vrms} * X_{irms} = X_{va}$$

Where X is the respective phase you are reading. This number should be the same as Xva which is printed. If not then perform the following calculation.

Calculated VA/reading from PM001 = calibration factor

Then multiply this factor by watt_cal to calibrate the power readings.



```
1 //*****GLOBAL VARIABLES*****  
2 float current_time; //for tracking time interval for acquire  
3 float last_time = 0;  
4 float acquire_time = 5; //number of seconds between acquire  
5 double avrmsaccum,bvrmsaccum,cvrmsaccum,irmsaccum,birmsaccum,cirmsaccum; //Accumulator  
6 double awattaccum,bwattaccum,cwattaccum,avaaccum,bvaaccum,cvaaccum;  
7 int dccount = 1; //counter  
8  
9 //*****CALIBRATION*****  
10 const int irms_os = 0; //ADC offset at low current reading  
11 const int vrms_os = 0; //ADC offset at low voltage reading  
12 float vrms_cal = 0.000017183; //Use calibrated multimeter to read voltage and adjust  
13 float irms_cal = 0.000004269 * 0.969; //Same as above...use calibrated clamp meter to  
14 float watt_cal = 0.00989166; //After calibrating vrms and irms, multiply them together  
15  
16 //*****ACQUISITION*****  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32
```

That would be:

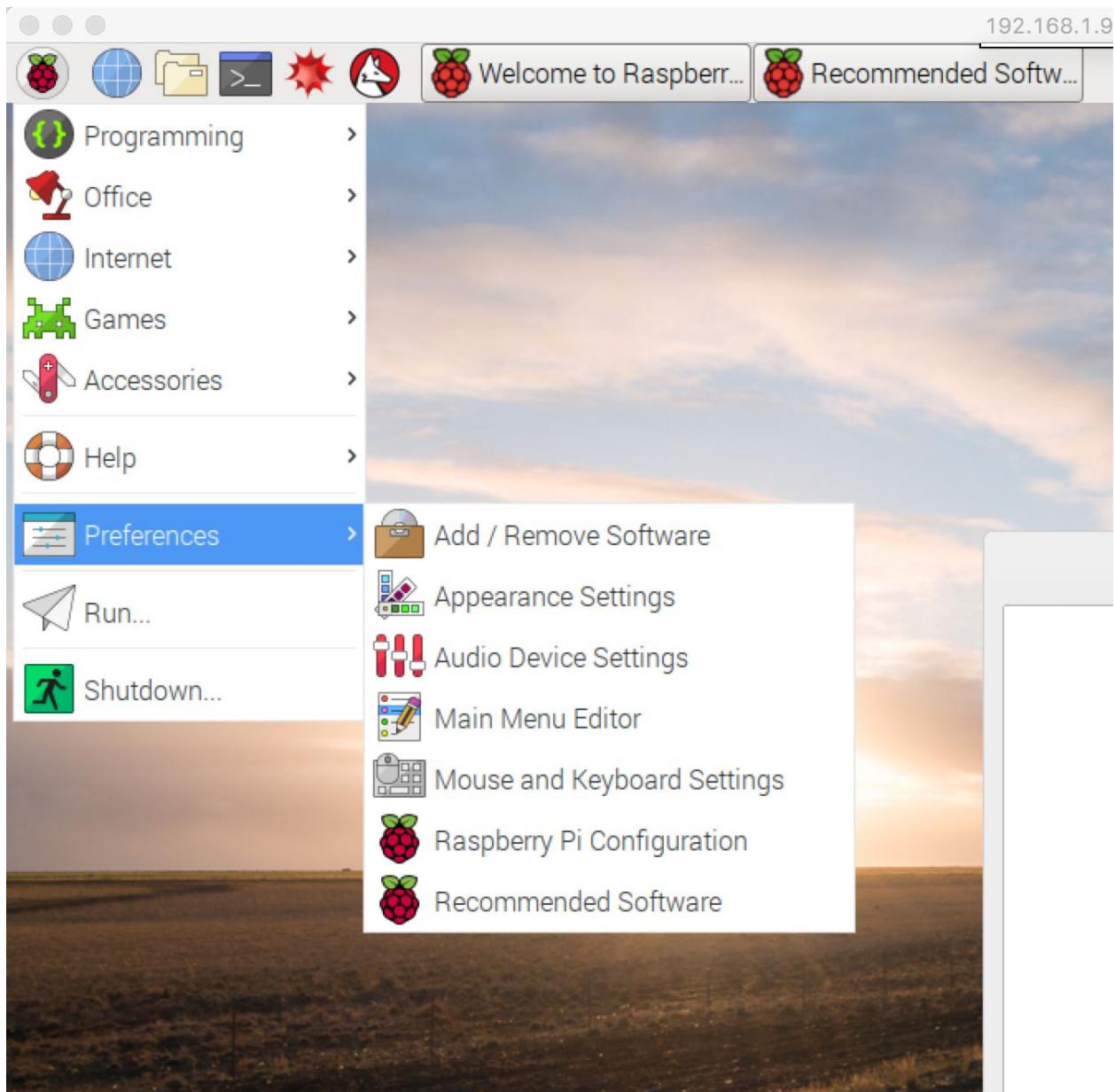
$$\text{float watt_cal} = 0.00989166 * \text{calibration factor}$$

CONNECTING TO A RASPBERRY PI THROUGH NODE RED

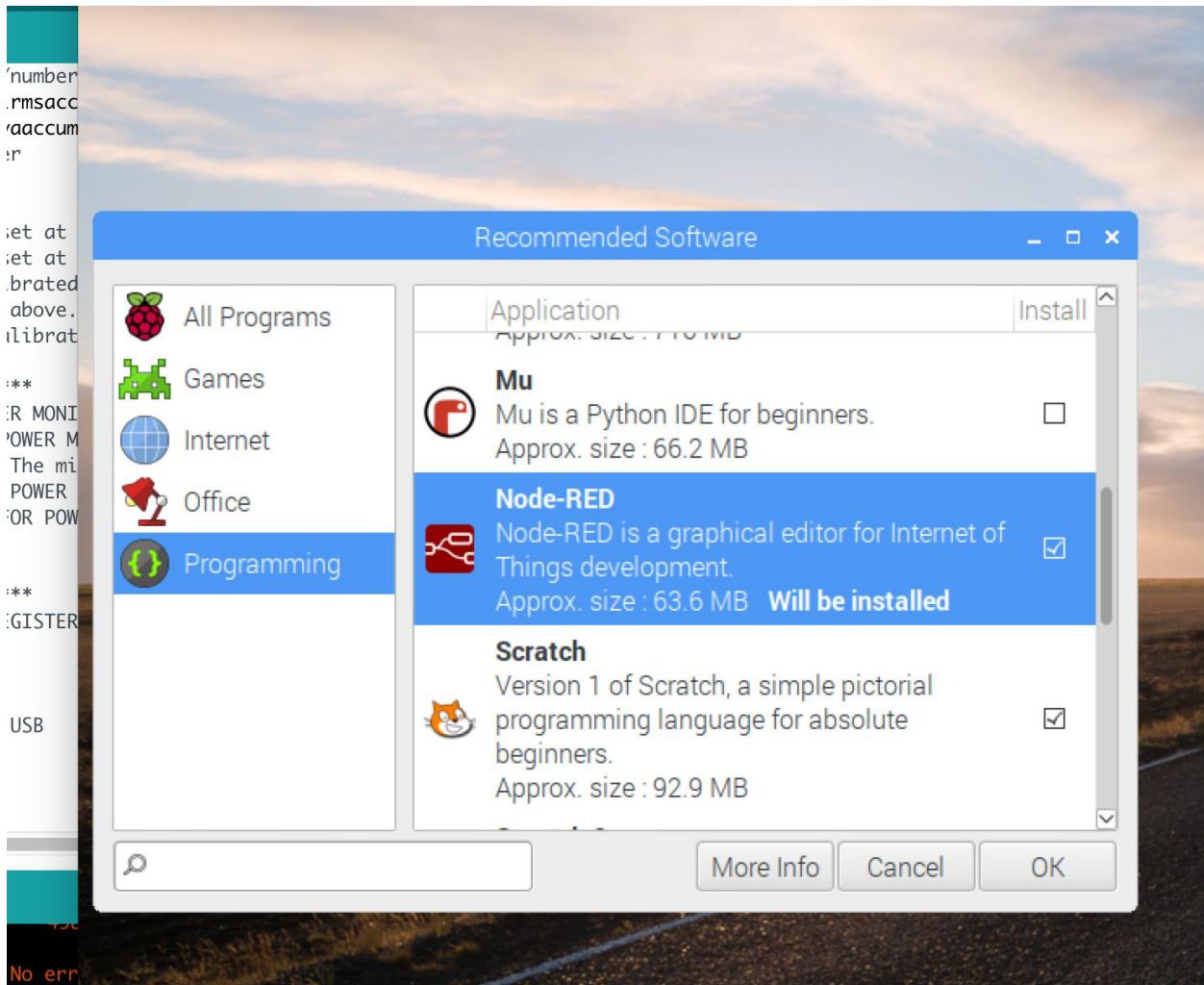
The Raspberry PI comes with a fun but powerful program called NodeRed which we love to use because it makes building programs fast and easy. It is based off of the nodeJS framework which uses javascript as its primary programming language. Some of you may have noticed that we have framed our data outputs from the Maple Mini as a JSON object. This makes it easier for NodeRed to parse the data.

On the older versions of Raspbian Jessie, NodeRed comes pre-installed. If you are using the new Stretch distribution use the following procedure to install NodeRed:

Open the recommended software tab:



And install the NodeRed application:



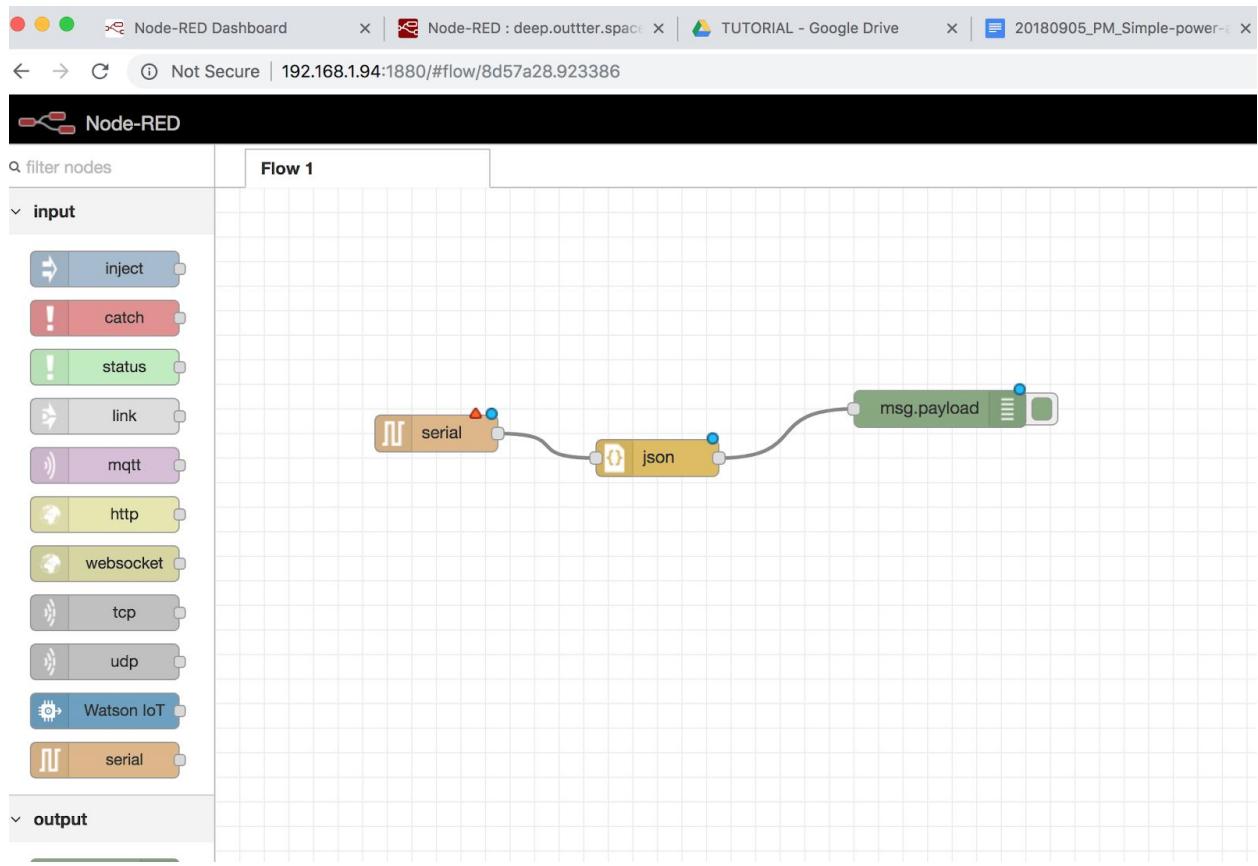
Open the terminal and type this command to have nodered autostart when you boot the PI.

```
sudo systemctl enable nodered.service
```

Now open your browser on the PI and type in localhost:1880 in the address bar. You should be able to pull up the NodeRed interface at this point.

Plug the USB cable from the maple mini and your power monitor into the USB port on the PI.

Drag the following nodes into the flow:



Connect and configure the serial port by double clicking on the node:

Edit serial in node

node properties

Serial Port: Add new serial-port...

Name: Name

Information

Node: "99f10067."
Type: serial in

Node Help

Reads data from a local serial port.

Can either

- wait for a "split" character (accepts hex notation (0x...))
- Wait for a timeout in milliseconds (first character received)
- Wait to fill a fixed sized buffer

It then outputs `msg.payload` (UTF8 ascii string or a binary)

If `msg.port` is set to the name of the selected serial port.

Edit serial in node > Add new serial-port config node

Serial Port: for example: /dev/ttyUSB0/

Settings: /dev/ttyAMA0 Bits:

/dev/ttyACM0

Input

Split input: on the character \n

and deliver: ascii strings

Output

add split character to output messages

Information

Node: "1f73c..."
Type: serial-port

Node Help

Provides configuration for serial ports.

The search button shows available serial ports to type in the location if known.

The input can be split on a character, a timeout, or after a fixed number of bytes.

If using a character, it can be escaped using the hex code (e.g. 0x0c).

Deploy your flow and there you go!!! A working power monitor. From here you can decide to do what you want with your data...send it over MQTT, store it in a database, make an HTTP dashboard...you name it!

