SCT-013-000 Non-invasive Split-Core Current transformer

Technical Manual Rev 1r0

SCT-013-000 Non-invasive Split Core Current transformer is an AC current sensor based on current transformers, it can transforms the big AC current to little, and then convert to voltage. Split core type makes this sensor suitable for DIY usage such as energy monitoring for house and building, protection of AC motor light equipment, air compressor and so on. Compatible in all gizDuino /Arduino MCU board.

Features:

- Crowtail compatible interface
- Input current: 0~100A
- Output type: 0~50mA
- Non-linearity: +/- 1%
- Turn Ratio: 2000:1
- Resistance Grade: Grade B
- Work Temperature: -25 ~ +70 deg C
- Dielectric Strength
  (between shell and output):
  1000V AC/1min 5mA
- Leading Wire in Length: 1m
- Building sample resistance: ohms

General Specifications:

Core material: Ferrite
Opening size: 13 mm x 13 mm
Mechanical Strength: The number of switching is not less than
1000 times(Test under 20 degrees C)
Safety index: Dielectric strength
(between Shell and output)
1000V AC/1min 5mA

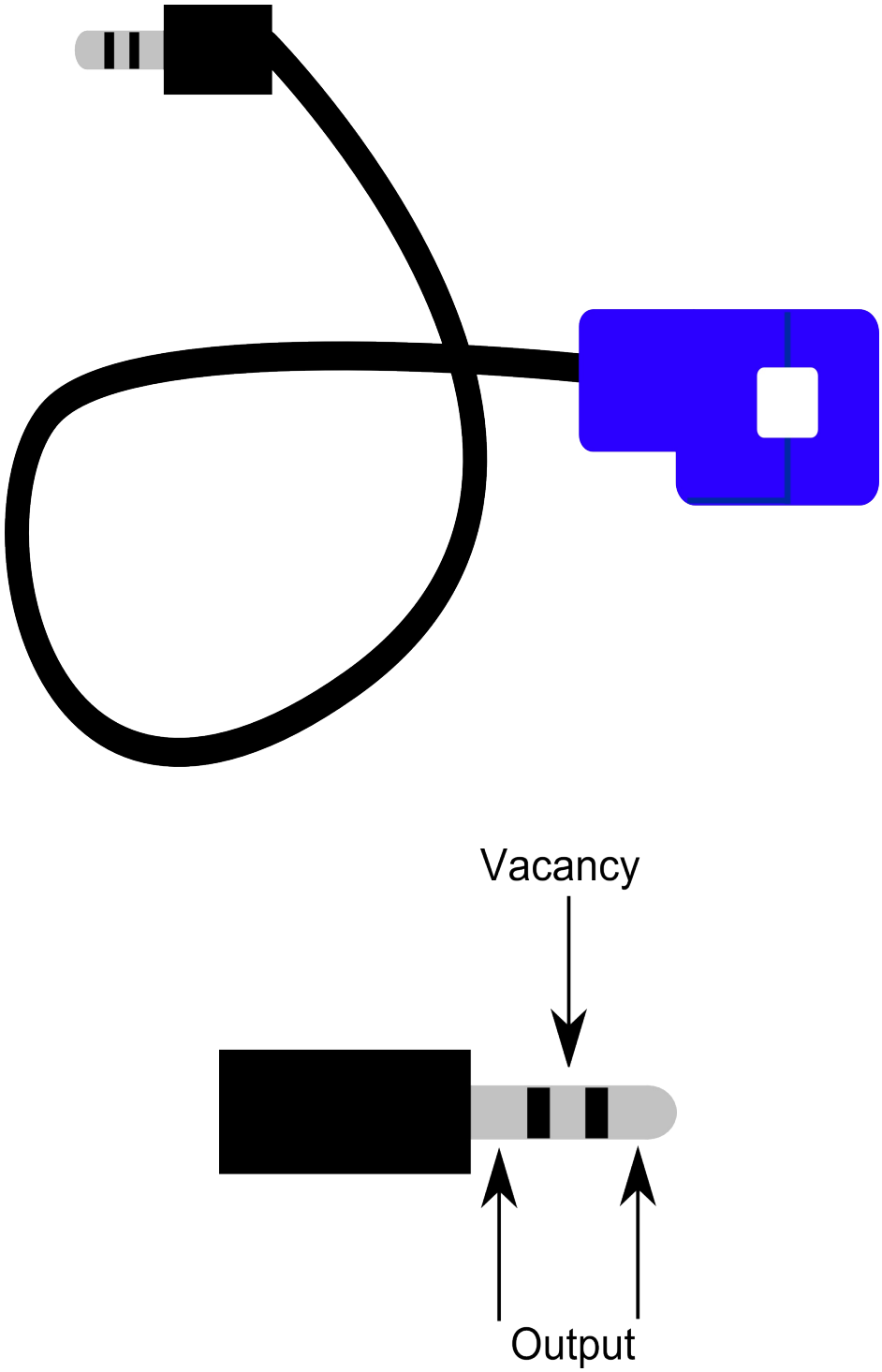

Figure 1: Major Presentation.

Components:

QTY	*DESCRIPTIONS*
1	18 Ohms, if supply voltage is 3.3V, or 33 Ohms , if supply voltage is 5V
2	10k Ohms (for voltage divider, any matching value resistor pair down to 10k)
1	10uF capacitor

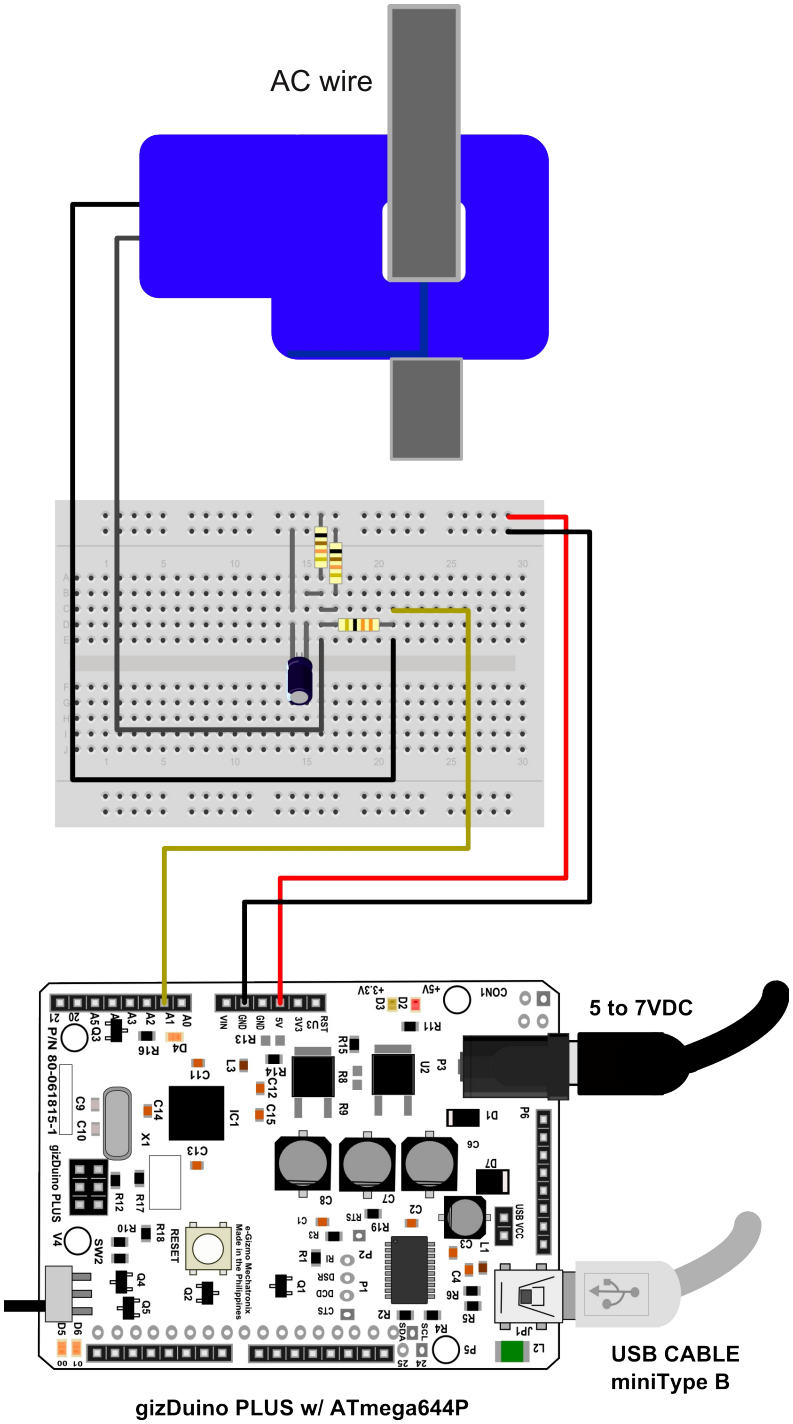

Figure 2: Irms connections with gizDuino PLUS (CT sensor)

Components:

QTY	*DESCRIPTIONS*
1	100k Ohms (for step down voltage divider)
2	10k Ohms (for biasing voltage divider)
1	10k Ohms (for step down voltage divider)
1	10uF capacitor

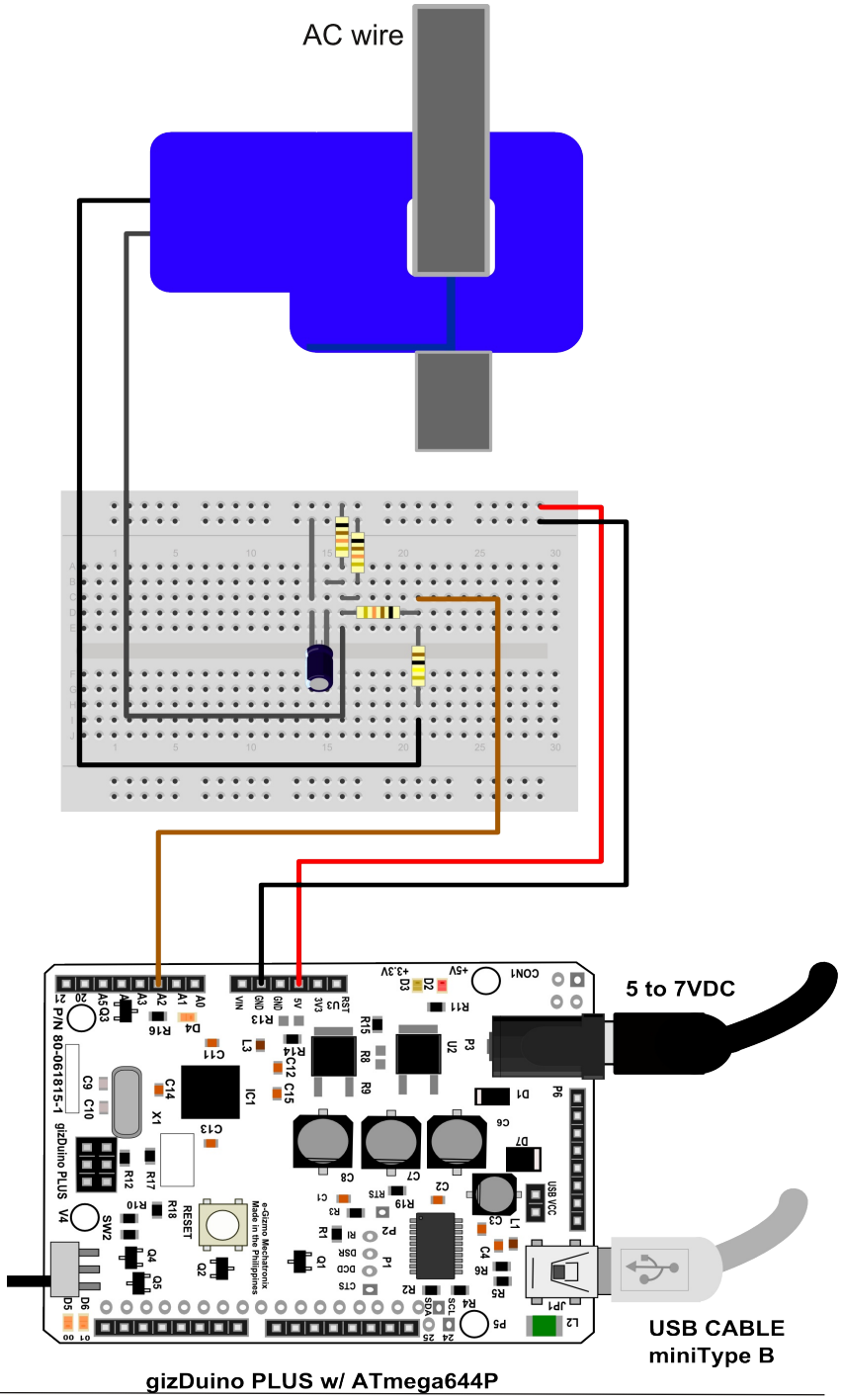

Figure 3: Vrms connections with gizDuino PLUS (Power Adapter)

A sample sketch to converts the raw data from its analog input value and outputs them to serial.

Download the ***EmonLib.h library***

Real power, Apparent power, power factor, rms voltage, rms current and power factor.

```
// EmonLibrary examples openenergymonitor.org, Licence GNU GPL V3

#include "EmonLib.h"             // Include Emon Library
EnergyMonitor emon1;             // Create an instance

void setup()
{
  Serial.begin(9600);

  emon1.voltage(2, 234.26, 1.7);  // Voltage: input pin, calibration, phase_shift
  emon1.current(1, 111.1);       // Current: input pin, calibration.
}

void loop()
{
  emon1.calcVI(20,2000);         // Calculate all. No.of half wavelengths (crossings), time-out
   //emon1.serialprint();           // Print out all variables (realpower, apparent power, Vrms, Irms, power factor)

  float realPower       = emon1.realPower;       //extract Real Power into variable
  float apparentPower   = emon1.apparentPower;   //extract Apparent Power into variable
  float powerFActor     = emon1.powerFactor;     //extract Power Factor into Variable
  float supplyVoltage   = emon1.Vrms;            //extract Vrms into Variable
  float Irms            = emon1.Irms;            //extract Irms into Variable

Serial.print("RP= ");
Serial.print(realPower);
Serial.print(" AP= ");
Serial.print(apparentPower);
Serial.print(" PF= ");
Serial.print(powerFActor);
Serial.print(" SV= ");
Serial.print(supplyVoltage);
Serial.print(" Irms= ");
Serial.println(Irms);
}
```

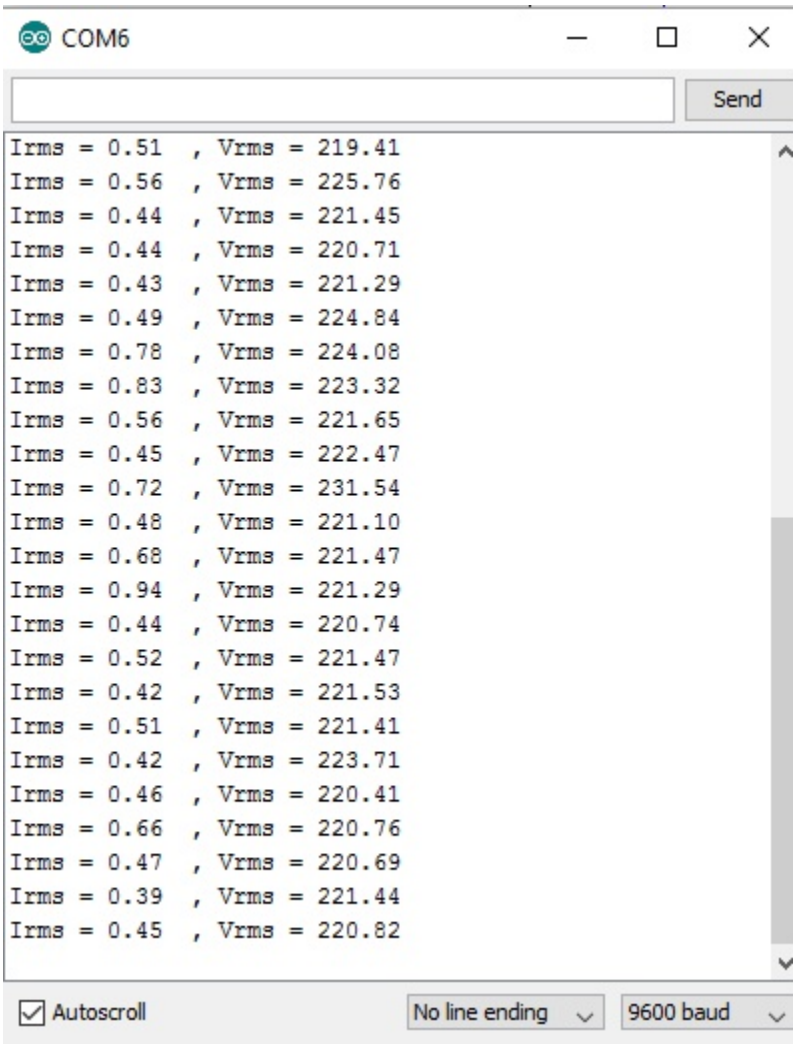

Figure 4: Serial Monitor

REFERENCE:
https://learn.openenergymonitor.org/electricity-monitoring/ctac/how-to-build-an-arduino-energy-monitor