Non Intrusive Load Monitoring For Detecting Different Loads in Houses

INTRODUCTION

Non-Intrusive Load Monitoring (NILM) is the process of estimating the energy consumed by individual appliances given just a whole-house power meter reading.

This method dismembers electric meter load curves to watch and analyze the performance of the appliances and systems that use electricity and are embedded in the signal as a virtue of being connected to the mains power of the house. In a sense, each major electricity-using appliance has a distinct “signature” that can be extracted from the load curve of the mains of the home. By parsing out individual appliances like air conditioners, water heaters, refrigerators and other loads, NILM disaggregates smart meter data to let us know of the combination of appliances that are switched on at a moment.

In this project, we aim to detect what appliances are switched on at what point of time in a household. Using the household’s main power supply, analyzing various quantities (voltage, current, power factor, active and reactive powers), we determine which appliance is powered on, with the help of data analytics.

We firstly implemented a Smart Meter circuit consisting of a circuit controlled by a microcontroller. Then we collected the data set comprising V_{rms}, I_{rms}, Power Factor, Active & Reactive powers and Apparent power. Various classification algorithms were trained using this data, and the best algorithm were chosen out of them.

For this we used an *Arduino UNO* to fetch current and voltage measurements from the mains supply and calculate the aforementioned electrical quantities. A *Current Transformer(CT)* and *Voltage Transformer(VT)* were used to measure current and voltage, respectively**.** The data so collected is then analysed and a model is trained to determine the appliances working at any specific time.

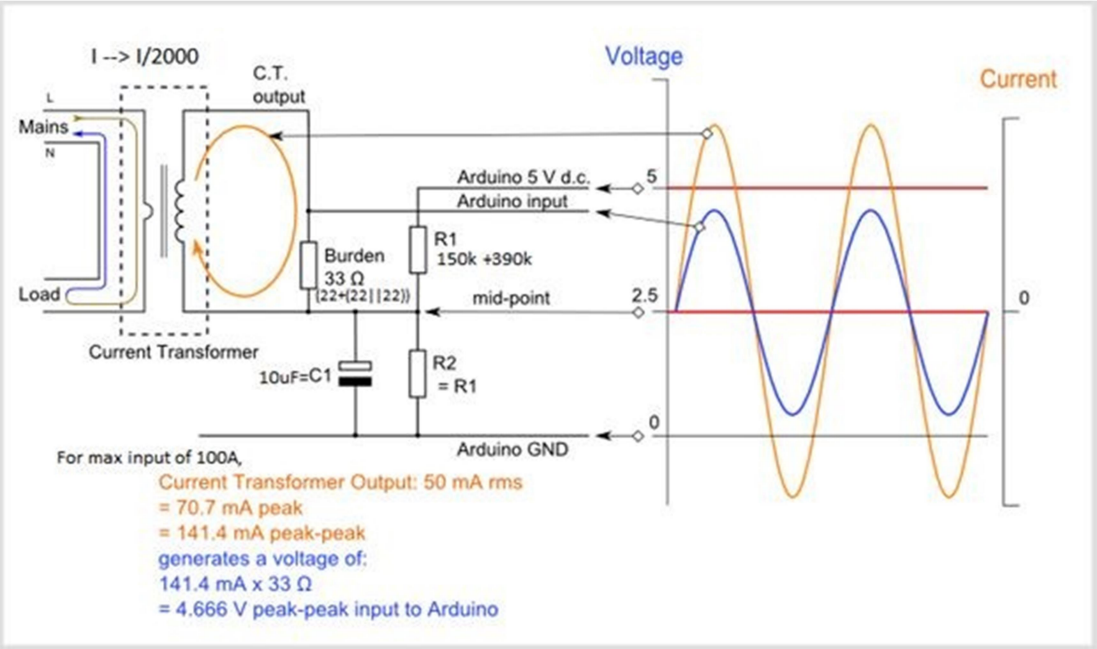
PROCEDURE

We decided to develop a microcontroller based smart meter which would then be used to get the required data set (V_{RMS}, I_{RMS}, Power Factor, etc.) in real time and apply algorithms to find out the components connected and their energy consumption, and hence the appropriate ones as NILM algorithms.

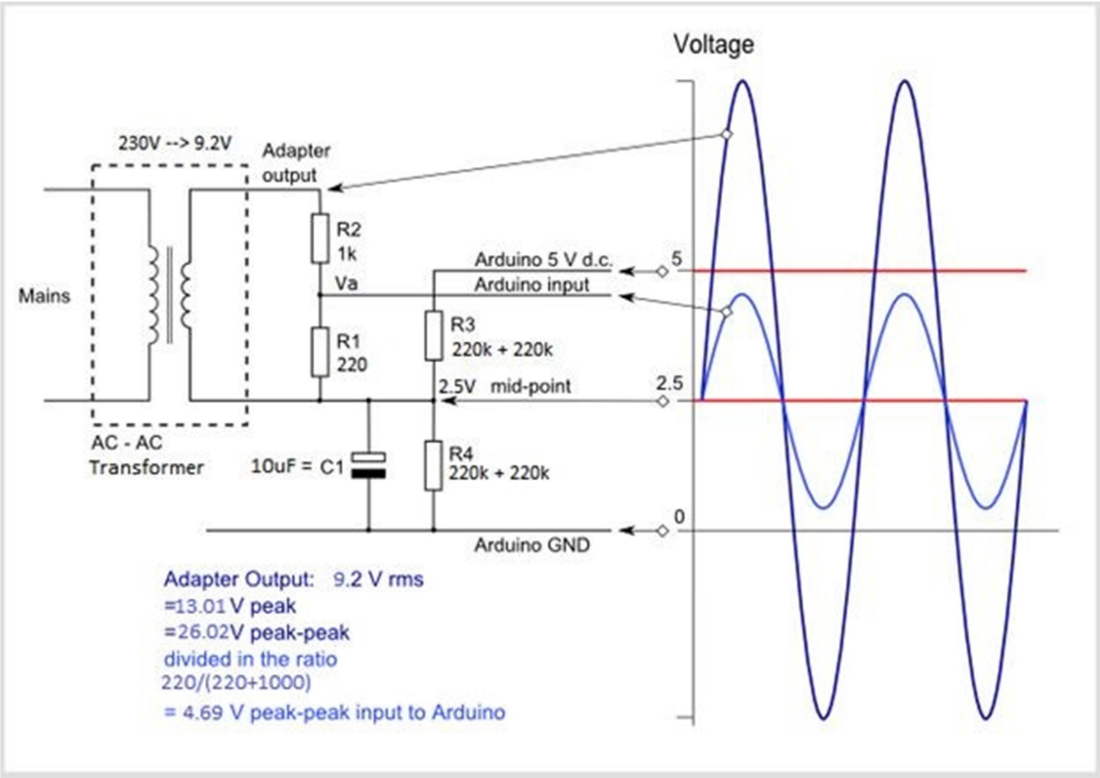
In order to have a marketable product, we envisioned the use of ESP8266, a WiFi enabled SoC which is capable of performing the same operations that the Arduino can, along with transmitting real time data via WiFi to an Android app for better user experience.

The following circuit was implemented to develop the smart meter. *The explanation to this circuit has been provided in a separate file uploaded in our submission folder.*

Current Measurement part of the circuit



Voltage Measurement part of the circuit



ESP8266 & Android App

After the circuit was made, we started off by setting up the ESP8266 for WiFi data transfer with a mobile application. We set up the ESP8266 board support package on the Arduino IDE and burned our first program to the flash memory of the microcontroller inside the ESP. A primitive android app was built to receive incoming data to a socket via WiFi.

Data Verification - Problem Encountered

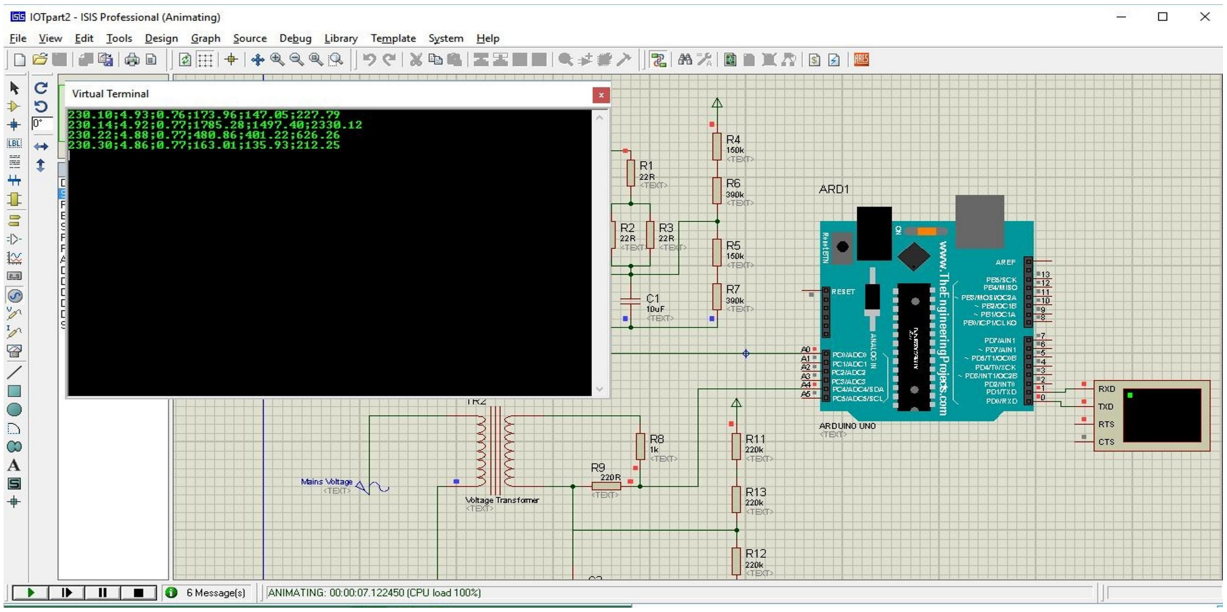
We verified the data being obtained the Arduino circuit first, before finally sending it to the Android app for processing via the ESP. We did this by comparing it with expected wave characteristics of the values as well as usual trends observed with applianced. We realised that the value of current that the Arduino was receiving were faulty. The current values from the CT sensor were constant, while we know that the current is supposed to be a sinusoidal wave. This made us stop in our tracks and take on a different route, towards simulation of the circuit.

Circuit Simulation

At this stage, there could have been two reasons for the error - either the code was wrong or it was some component that was causing this failure. Simulation would help us determine just that.

We used the Proteus Design Suite which is an Electronic Design Automation tool including schematic capture, simulation and PCB Layout modules.

As the below screenshot(along with others uploaded in the folder) shows, the codes that we used were fine and it was the component that was causing this error. But due to constraints of time and money, we could not get a component and had to rely on the simulation itself.



Proteus Simulation producing required data set

In order to get data equivalent to that out of combinations of appliances running, we created signal generators in *EasyHDL* in Proteus, which would add the sinusoidal current signals of each appliance according to amplitude and phase of each. *The code file is attached in the folder.*The obtained data was found to be well within expected range of values for the appliances that we chose. This data was then processed for performing NILM.

IMPLEMENTATION of NILM Algorithms

The implementation of NILM was done using the machine learning algorithm - The DECISION TREE. A **decision tree** is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. The algorithm was used because this algorithm is efficient, easy to implement and also easy to analyse.

Fan, Microwave and **Television** were the appliances used to collect the input data for the algorithm. Thes three were used in seven different combinations. The reason for choosing such appliances was that not only are they used in nearly every household, but also unlike refrigerators, ACs, and geysers; they consume electricity at a constant rate, while the latter few have internal systems like compressors and heaters that switch on and off from time to time. Therefore, complications were avoided.The data generated was divided into two
different categories - the training data set and the testing data set. The features used $\mathbf{V_{rms}}$, $\mathbf{I_{rms}}$, **Power Factor , Active Power , Reactive Power** and **Apparent Power.** The Classification States were corresponding to the seven different combination listed below
(according to when the states were ON) -

- Fan
- Microwave
- TV
- Fan & Microwave
- TV & Microwave
- Fan & TV
- Tv and Fan and Microwave

The training data set consisted of 651 data entries. The testing data set consisted of 156 data entries. Both the files are attached to the folder. **Python** was used to implement the algorithm. The accuracy of the model is **80.77%**. Therefore out of the 156 test entries, the model could correctly predict 126 entries.

The python code, the tree formed and the other related files are attached in the folder.