B. TECH. PROJECT REPORT

On

Internet of Things (IoT) enabled Energy Management System for Smart Cities

BY

Kanchan Bhandari

Piyush Dugar

DISCIPLINE OF ELECTRTICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2016

Internet of Things (IoT) enabled Energy Management System for Smart Cities

A PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degrees
of
BACHELOR OF TECHNOLOGY
in
ELECTRICAL ENGINEERING

Submitted by:
Kanchan Bhandari
Piyush Dugar

Guided by:
Dr. Santosh Kumar Vishvakarma,
Assistant Professor,
Electrical Engineering,
Indian Institute of Technology Indore

INDIAN INSTITUTE OF TECHNOLOGY INDORE
December 2016

CANDIDATE'S DECLARATION

We hereby declare that the project **Internet of Things (IoTs) enabled Energy Management System for Smart Cities**submitted in partial fulfillment for the award of the degree of Bachelor of Technology in **Electrical Engineering** completed under the supervision of **Dr. Santosh Kumar Vishvakarma, Assistant Professor, Electrical Engineering, IIT Indore** is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Kanchan Bhandari
130002009
Discipline of Electrical Engineering
Indian Institute of Technology Indore

Piyush Dugar
130003029
Discipline of Electrical Engineering
Indian Institute of Technology Indore

CERTIFICATE

by BTP Guide

It is certified that the declaration made by the student is correct to the best of my knowledge and belief.

Dr. Santosh Kumar Vishvakarma,
Assistant Professor,
Discipline of Electrical and Engineering,
Indian Institute of Technology Indore
(Project Guide)

PREFACE

This report on **Internet of Things (IoTs) enabled Energy Management System for Smart Cities'** is prepared underthe supervision of Dr. Santosh Kumar Vishvakarma, Assistant Professor, ElectricalEngineering, IIT Indore.

Through this report, we have tried to provide a detailed description of the technologies that have been used to design the smart meters for accurate billing process. We have tried and tested the same on different household devices for which electricity usage can be shifted according to the unit prices. Further, we have used Thingspeak to upload the data online so that it will be accessible to both the utility supplier and the customer. We have also tried to implement this smart meter to the best of our abilities.

We have tried our best to explain the proposed design model in detail. The comparison of proposed smart meter with already existing traditional ordinary meters is also discussed.

ACKNOWLEDGEMENTS

We would like to thank our BTP supervisor, **Dr. Santosh Kumar Vishvakarma**, for his constant support in structuring the project and for his valuable feedback which helped us in the course of the project. He gave us the opportunity to discover and work in this domain. He guided us thoroughly and pulled us out of the craters of failures we faced all through the period.

We are especially grateful to **Dipesh Mopngia** sir who guided us, made us familiar with the nuances in various IoT technologies and provided us the pathway for this project in right manner. He simultaneously guided us with useful direction to proceed along whenever necessary.

We are also thankful to all our family members, friends and colleagues who have been a constant source of motivation. Finally, we offer sincere thanks to everyone else who knowingly or unknowingly helped us complete this project.

Kanchan Bhandari
130002009
Discipline of Electrical Engineering
Indian Institute of Technology Indore

Piyush Dugar
130003029
Discipline of Electrical Engineering
Indian Institute of Technology Indore

ENERGY MANAGEMENT USING A SMART-METER

Abstract

One avenue through which today's energy problems can be addressed is through the reduction ofenergy usage in households. The existing utility system only provides feedback at the end of the monthin the form of a bill and consumed kilowatt hours (kWh). A homeowner has no way to track their powerusage on a more immediate basis.

The Arduino-based wireless power meter is a non-invasive current meter for household power readings. Current is measured using split core current transformer. This data is then transmitted over an ESP8266 connection through the Node MCU to the smart data center. The project aims to provide a clear picture of a home's current usage, and through thisdata provide an estimate to power consumption. The project also aims to identify which devices turn onand off by analysis of this current data. The goal of providing such data to a user is that they will optimizeand reduce their power usage.

Contents

List of figures

Chapter 1

Introduction

This chapter highlights the background and motivation for the project. The problem statement has been described of the project and the importance of the results is also clearly portrayed. Towards the end, the objectives are briefly outlines and the future scope is also discussed.

1.1 Background

In the existing power utility set up, consumers are presented with usage information only once a month with their bill. The length of time between updates about power usage is far too long for a consumer to observe a changed behavior's effect on power usage. In addition utility bills can be convoluted in how they present usage information, and a consumer may not be able to decipher changes in their power usage from the last bill. An opportunity to educate customers on power usage is lost because of these realities.

If a person can instantaneously see how much power leaving a device on by accident consumes per minute, they may be more careful in the future about letting devices run when not needed. The goal of creating more awareness about energy consumption would be optimization and reduction in energy usage by the user. This would reduce their energy costs, as well as conserve energy.

The present system of energy metering as well as billing in India uses electromechanical and somewhere digital energy meter. But the traditional ordinary meters consumes more time and labor. One of the prime reasons for developing smart meters is the traditional billing system which is inaccurate. Many times slow, costly and lack in flexibility and reliability.

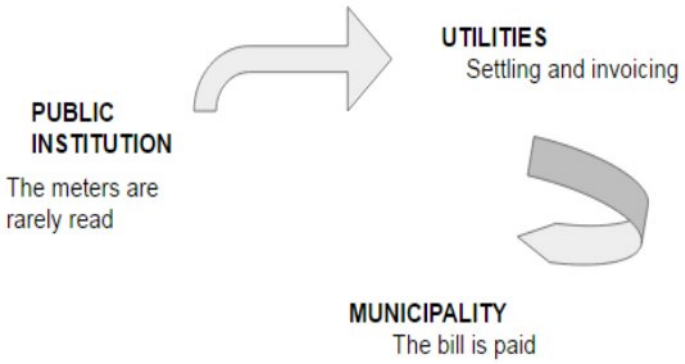

Fig 1: Ordinary meter billing process

Today accuracy in billing is highly recommended leading for a rise of smart meter development. The smart energy meter gives real power consumption as well as accurate billing. Smart energy meter provides real time monitoring of electricity uses in less time and is cost effective. Smart Meters are electronic measurement devices used by utilities to communicate information for billing customers and operating their electric systems. They enable two-way communication between the meter and the central system and unlike home energy-monitors; smart meters can gather data for remote reporting. They give you near real time information on energy use - expressed in rupees. With smart-meters users will only be billed for the energy they use, helping them budget better.

A successful energy management involves a cyclic process as shown:

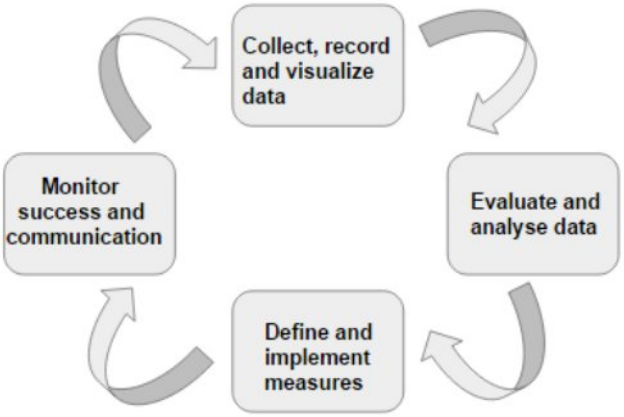

Fig 2: Smart energy management

Smart energy management is as a means to take steps towards sustainable development. Smart meters also satisfy the homeowners' need to track their power usage on a more immediate basis. It ensures Demand side management (DSM). Better accommodate renewable energy sources by flattening of the load curve.

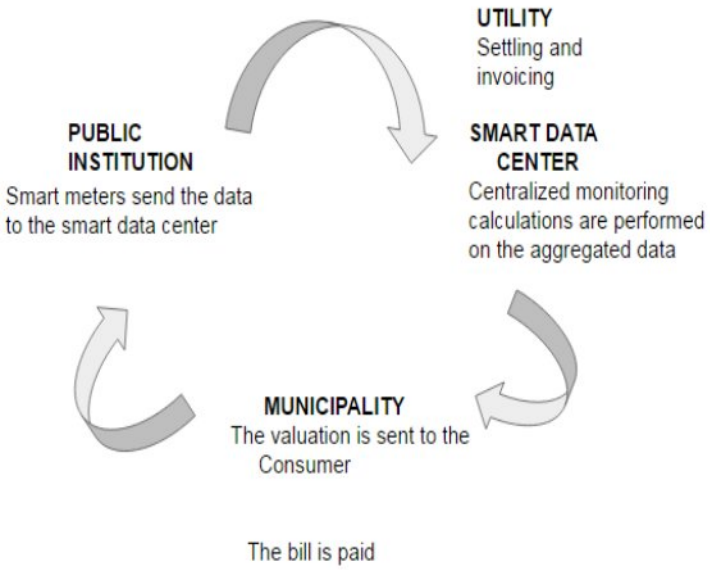

Fig 3: Smart meter billing process

Smart meters are measurement and information capture devices. Although not a common practice at present, smart meters enable demand–responsefor consumers by providing two-way communication. Demand-response is an opportunity provided to the customers to play a significant role in the operation of electric grid by shifting their electricity usage demand during peak periods in response to time-based rates. Smart meters also enable power efficiency programs for customers' benefit.

1.2 Household Power

In most Indian households, power comes into the house through a three wire, split phase connection. Two "hot" wires carry current into the circuit breaker. A neutral wire also provides a connection to ground for the house circuit. Each hot wire has an RMS voltage of 220V +/- 5%. The wires are set up so that the AC waveforms are 180°, so that both lines combined can provide a 440V source for larger household appliances.

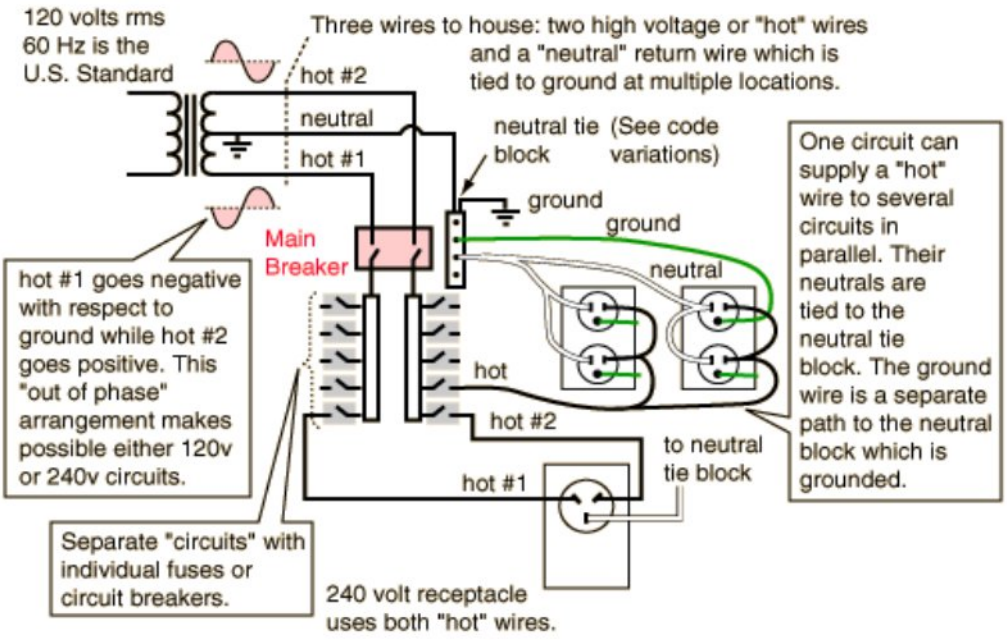

Fig 4: Household 3 phase wire connections.

The apparent power consumed by a household can be found by taking the product of the RMS voltage and total RMS current. The real power can be calculated from samples by taking the average of the product of the voltage and current samples over a specified window. Power factor can then be calculated by dividing real power by the apparent power.

1.3 System Requirements

The goals of the project were as follows:

- Accurately collect a house's total current consumption safely, and with a relatively fast update rate.
- Transmit the information back to Thingspek to be represented visually to a user.
- Identify when devices in the household turn on and off based on changes in the current data.

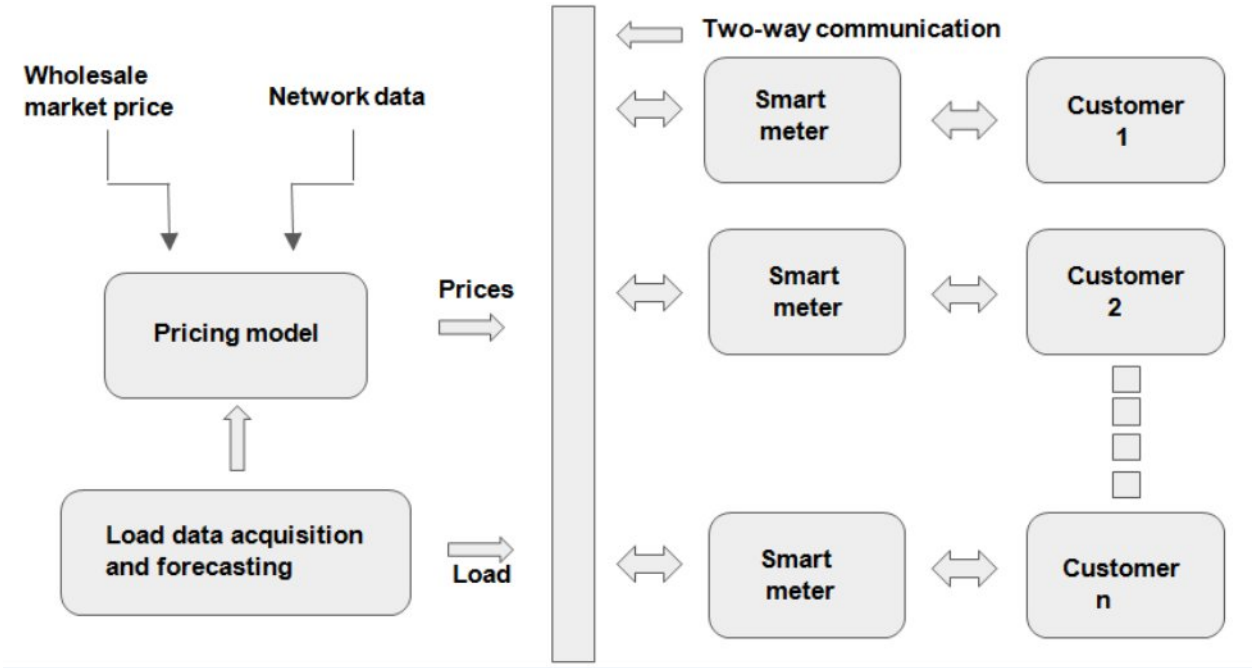

Fig 5: Real-time pricing framework

To ensure the safety of the user and the ease of installation, a non-invasive method of measuring current was required. This meant that the current measurement circuit could not be in series with the mains power line, and could not require the disconnection of the line for any reason. The wireless connection had to be able to integrate into the user's home network, and comply with the IEEE802.11b standard. The data rate of the wireless connection also had to be sufficient to support the required update rate. The overall speed of both the wireless connection protocols had to be sufficiently fast that it would not interfere with obtaining the required amount of samples in a given period.

Chapter 2

ROLE OF SMART METERS IN THE ENERGY MANAGEMENT

2.1 Overview

Internet of Things is essentially an architectural framework which allows integration and data exchange between the physical world and computer systems over existing network infrastructure.

In our project we use an Arduino-based smart power-meter which consists of a **non-invasive current sensor**--thus ensuring the safety of the user and the ease of installation-- to measure household power and then transmit it via a **WiFi module**. The user can then receive feedback from the grid as a form of price signal to better manage his/her consumption. Therefore, the goals of the project are as follows:

- Measure a house's total power consumption safely.
- Transmit the information back to the user so that he/she can make better decisions, and also adjust their schedule with price signals from the grid.

2.2 Hardware

The hardware of the system consists of:

1. **Arduino UNO**
2. **Node MCU ESP8266**
3. **Non-invasive current sensor transformer SCT013-000**
4. **9v AC-AC power adapter.**

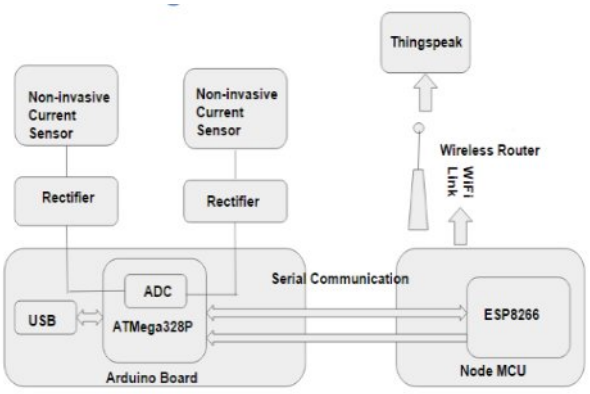

Fig 6: Hardware implementing

The sensor capture hardware consists of the current transducers and rectifier circuit connected to the Analog-to- digital converter (ADC) of the MCU. A split core current transducer was chosen as the current sensor of the project because of its non-invasive nature. The sensor can be clamped onto the mains lines without interrupting power into the circuit breaker. Avoiding interrupting the lines into the circuit breaker makes the installation of the sensors both safer and easier. One sensor is clamped onto each “hot” wire going into the circuit breaker. The total current being drawn by the household is the sum of the measured current in both wires. The output of the current transducer is an AC voltage proportional to the AC current enclosed by the sensor’s ring. The output range of the sensor is determined by the selection of the burden resistance. Selecting a higher burden resistance scales the output higher, providing a greater resolution of currents measurement. A higher resistance also means a smaller range of current measurement, and potential damage to the MCU if the current is higher than intended for the design.

The MCU board is a standalone Arduino Uno development board. The wireless module is an ESP8266 module that is used to establish a network that can help in transmitting the data to a device controlled by the user, ex: Mobile phones.

2.2.1 Arduni UNO

The UNO is a self contained USB development board centered on an ATMega328P. The operating voltage for most of the board is 5V, provided by an onboard voltage regulator. Power for the board can come from the USB connection, the 9V DC connection, or a battery connected to the VIN terminal. For this project, the 9V DC connection was used for power.The current measurement circuit connects to the Arduinoboard through analog pins 0 and 1. This allows the Arduino to sample the output voltage from the current measurement circuit with the ATMega328’s 10-bit ADC. A voltage of 5V was applied from the USB to serial converter chip to the voltage reference pin, AREF, in order to provide the ADC with the required reference voltage. Connections to the ground and VCC pins were also provided for the op amp in the precision rectifier.

The USB connection on the board was used for programming the chip and getting serial output for troubleshooting. The interface between USB and the ATMega328’s UART was provided by the Arduino’s built in USB to serial converter chip. The device showed up as a virtual COM port on the host PC, and could be interfaced with any program capable of reading and writing to a serial port. The board came pre-built from the distributor, so no major hardware assembly was

required for use. Female headers are attached to each of the I/O and power ports, allowing wires to be inserted rather than soldered to each connection. The connections to the AREF pin, the analog pins, and the ground/VCC pins were all wired in this way. No major modifications of the stock hardware were required.

Fig 7: Arduino Uno board

2.2.2 WiFi Module-ESP8266

The ESP8266 WiFi Module is a self-contained SOC (system on chip) with integrated TCP/IP protocol stack that can give any microcontroller access to our WiFi network. The ESP8266 is capable of either hosting an application or offloading all WiFinetworkings functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, we can simply hook this up to our Arduino device and get about as much WiFi-ability as a WiFi Shield offers. The ESP8266 module is an extremely cost effective board with a huge and ever growing community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs (General purpose inputs/outputs) with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, isdesigned to occupy minimal PCB area. The ESP8266 supports APSD (Automatic Power Save Delivery) for VoIP (Voice over Internet Protocol) applications and Bluetooth co-existance interfaces. It contains a self-calibrated RF allowing it to work under all operating conditions and requires no external RF parts.

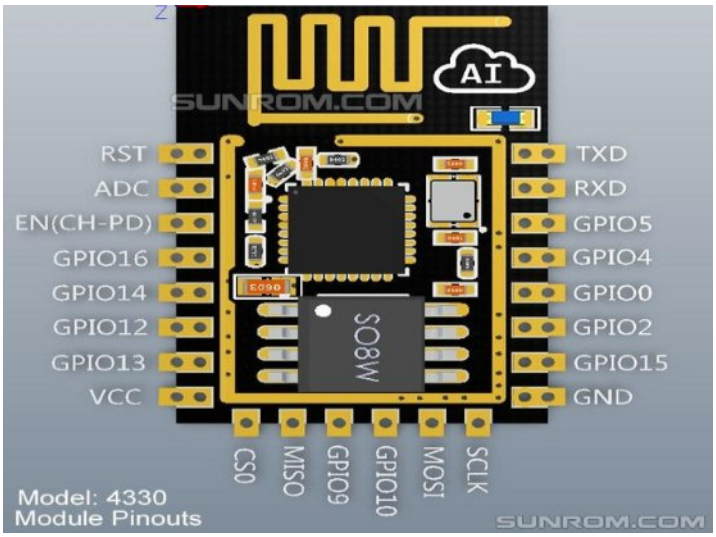

Fig 8: ESP8266 pins

The ESP8266 Module is not capable of 5-3V logic shifting and will require an external Logic Level Converter. This version of the ESP8266 WiFi Module used in our project has increased the flash disk size from 512k to 1MB.

2.2.3 Current Sensor

A split core current transducer was chosen as the current sensor of the project because of its non-invasive nature. The sensor can be clamped onto the mains lines without interrupting power into the circuit breaker. Avoiding interrupting the lines into the circuit breaker makes the installation of the sensors both safer and easier. One sensor is clamped onto each “hot” wire going into the circuit breaker.

Fig 9: Current sensor SCT-013-000

The total current being drawn by the household is the sum of the measuredcurrent in both wires. The output of the current transducer is an AC voltage proportional to the AC current enclosed by the sensor’s ring. The amplitude of the voltage waveform is determined by this equation:

? =?*?/????

Where V is RMS AC voltage across the burden resistor, I is the RMS AC current enclosed by the transducer, and R is the resistance of the burden resistor. This equation means that the output range of the sensor is determined by the selection of the burden resistance R. Selecting a higher burden resistance scales the output higher, providing a greater resolution of currents measurement. A higher resistance also means a smaller range of current measurement, and potentially damage to the MCU if the current is higher than intended for the design. When sizing the burden resistor, the range of linear behavior must also be taken into account.

2.2.4 AC-AC Power Adapter

An AC voltage measurement is needed to calculate real power, apparent power and power factor.This measurement can be made safely (requiring no high voltage work) by using an AC to AC power adaptor. The transformer in the adapter provides isolation from the high voltage mains.

Fig 10: 9V AC-AC power adapter

This briefly covers the electronics required to interface an AC to AC power adapter with an Arduino. As in the case of current measurement with a CT sensor, the main objective for the signal conditioning electronics detailed is to condition the output of the AC power adapter so it meets the requirements of the Arduino analog inputs: a **positive voltage between 0V and the ADC reference voltage** (Usually 5V).

Chapter 3

Measurements Theory

3.1 AC Current Measurements

To connect a CT sensor to an Arduino, the output signal from the CT sensor needs to be conditioned so it meets the input requirements of the Arduino analog inputs, i.e. a **positive voltage between 0V and the ADC reference voltage.**Here we use an Arduino board working at 5 V.

This can be achieved with the following circuit which consists of two main parts:

1. The CT sensor and burden resistor
2. The biasing voltage divider (*R1 & R2*)

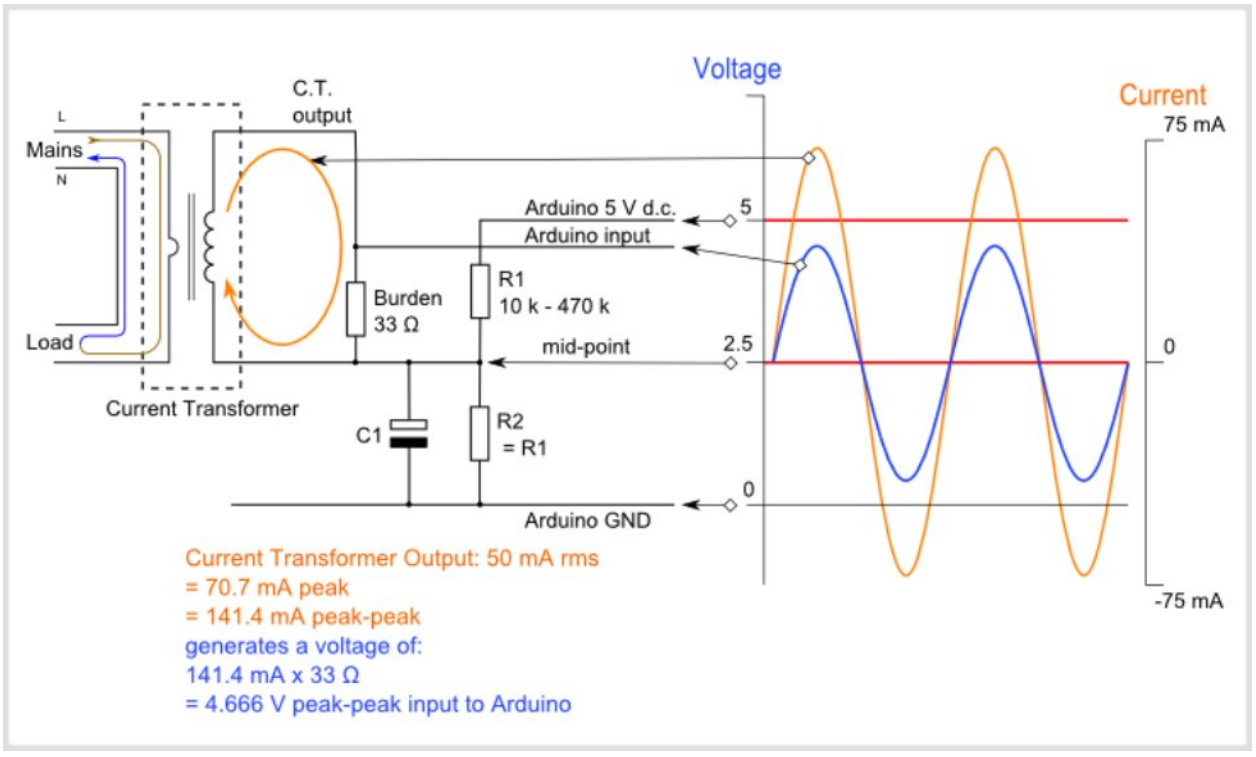

Fig 11: Measuring AC Current with a CT sensor

3.1.1 Calculating a suitable burden resistor size

If the CT sensor is a "current output" type such as the *SCT-013-000*, the current signal needs to be converted to a voltage signal with a burden resistor. If it is a voltage output CT we have skip this step and leave out the burden resistor, as the burden resistor is built into the CT.

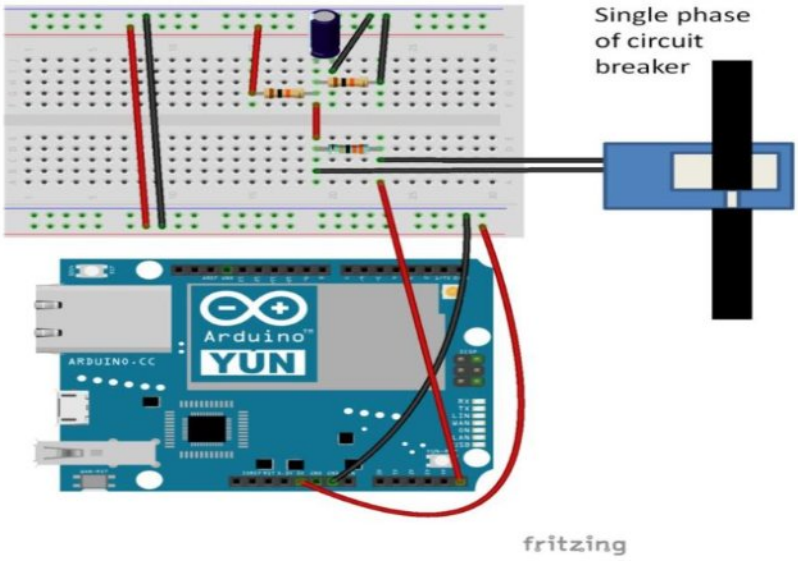

Fig 12: Circuit connections for current measurement

1) Choose the current range you want to measure

The SCT-013-000 CT has a current range of 0 to 100 A. For this example, we choose 100 A as our maximum current.

2) Convert maximum RMS current to peak-current by multiplying by √2.

Primary peak-current = RMS current × √2
= 100A × 1.414
= 141.4A

3) Divide the peak-current by the number of turns in the CT to give the peak-current in the secondary coil.

The SCT-013-000 CT has 2000 turns, so the secondary peak current will be:

Secondary peak-current = Primary peak-current / no. of turns
= 141.4 A / 2000
= 0.0707A

4) To maximize measurement resolution, the voltage across the burden resistor at peak-current should be equal to one-half of the Arduino analog reference voltage. (AREF / 2)

If you're using an Arduino running at 5V: AREF / 2 will be 2.5 Volts. So the ideal burden resistance will be:

Ideal burden resistance = (AREF/2) / Secondary peak-current
= 2.5 V / 0.0707 A
= 35.4 Ω

35 Ω is not a common resistor value. The nearest values either side of 35 Ω are 39 and 33 Ω. Always choose the smaller value, or the maximum load current will create a voltage higher than AREF. We recommend a 33 Ω ±1% burden. In some cases, using 2 resistors in series will be closer to the ideal burden value. The further from ideal the value is, the lower the accuracy will be.

Here are the same calculations as above in a more compact form:

Burden Resistor (ohms) = (AREF * CT TURNS) / (2√2 * max primary current)

3.1.2 Adding a DC Bias

If we want to connect one of the CT wires to ground and measure the voltage of the second wire, relative to ground, the voltage would vary from positive to negative with respect to ground. However, the Arduino analog inputs require a *positive* voltage. By connecting the CT lead we connected to ground, to a source at half the supply voltage instead, the CT output voltage will now swing above and below 2.5 V thus remaining positive.

Resistors R1 & R2 in the circuit diagram above are a voltage divider that provides the 2.5 V source. Capacitor C1 has a low *reactance* - a few hundred ohms - and provides a path for the alternating current to bypass the resistor. A value of 10 µF is suitable.

3.2 AC Voltage measurements

The output signal from the AC voltage adapter is a near-sinusoidal waveform. We have a 9V (RMS) power adapter so the positive voltage peak is 12.7V and the negative peak is -12.7V.

However, due to the poor voltage regulation with this type of adapter, when the adapter is unloaded, the output is often 10-12V (RMS) giving a peak voltage of 14-17V. The voltage output of the transformer is proportional to the AC input voltage.

The signal conditioning electronics needs to convert the output of the adapter to a waveform that has a positive peak that's less than 5V and a negative peak that is more than 0V.

So we need to:

1) **scale down** the waveform and

2) **add an offset** so there is no negative component.

The waveform can be scaled down using a voltage divider connected across the adapter's terminals, and the offset (bias) can be added using a voltage source created by another voltage divider connected across the Arduino's power supply (in the same way we added a bias for the current sensing circuit). Here's the circuit diagram and the voltage waveforms:

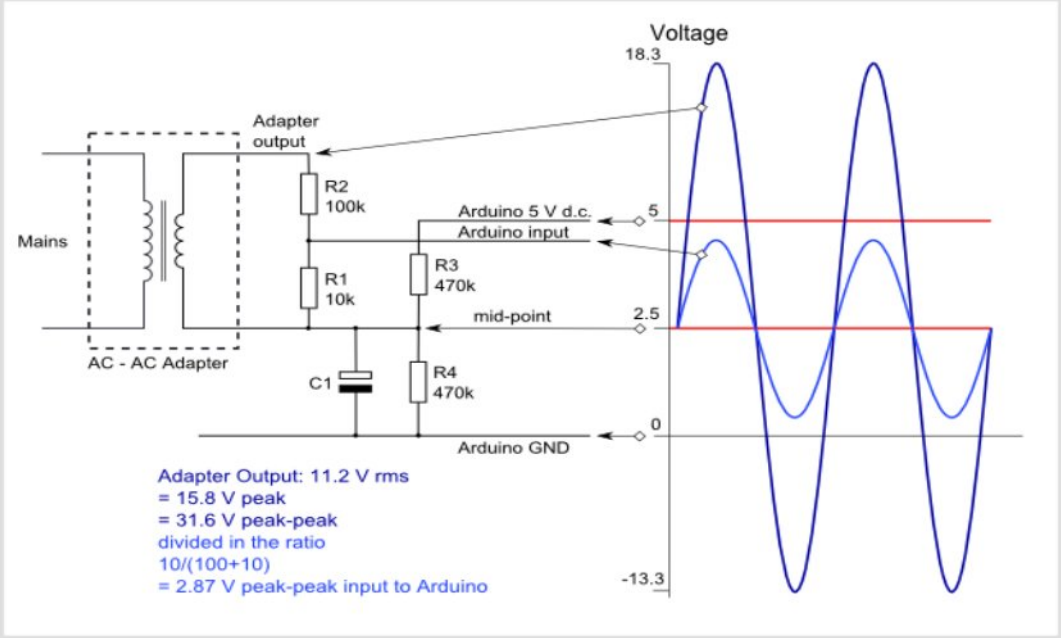

Fig 13: Measuring AC Voltage with an AC to AC power adapter

Resistors **R1** and **R2** form a voltage divider that scales down the power adapter AC voltage. Resistors **R3**and **R4** provide the voltage bias. Capacitor **C1** provides a low impedance path to ground for the AC signal. The value is not critical, between 1 µF and 10 µF will be satisfactory.

R1 and R2 need to be chosen to give a peak-voltage-output of ~1V. For an AC-AC adapter with a 9V RMS output, a resistor combination of 10k for R1 and 100k for R2 would be suitable:

```
peak_voltage_output = R1 / (R1 + R2) x peak_voltage_input
=10k / (10k + 100k) x 12.7V = 1.15V
```

The voltage bias provided by R3 and R4 should be half of the Arduino supply voltage. As such, R3 and R4 need to be of equal resistance. Higher resistance lowers energy consumption

If the Arduino is running at 5V the resultant waveform has a positive peak of 2.5V + 1.15V = 3.65V and negative peak of 1.35V satisfying the Arduino analog input voltage requirements. This also leaves some "headroom" to minimize the risk of over or under voltage.

3.3AC Power Measurements

We make between 50 and 100 measurements every 20 milliseconds. 100 if sampling only current.50, if sampling voltage *and* current.Here we're limited by the Arduino analog read command and calculation speed.

3.3.1 Real power:

```
for (n=0; n<number_of_samples; n++)
{
  // inst_voltage and inst_current calculation from raw ADC input goes here

  inst_power = inst_voltage * inst_current;

  sum_inst_power += inst_power;
}

real_power = sum_inst_power / number_of_samples;
```

3.3.2 Root-Mean-Square (RMS) Current:

```
for (n=0; n<number_of_samples; n++)
{
  // inst_current calculation from raw ADC input goes here.

  squared_current = inst_current * inst_current;

  sum_squared_current += squared_current;
}

mean_square_current = sum_squared_current / number_of_samples;
root_mean_square_current = sqrt(mean_square_current);
```

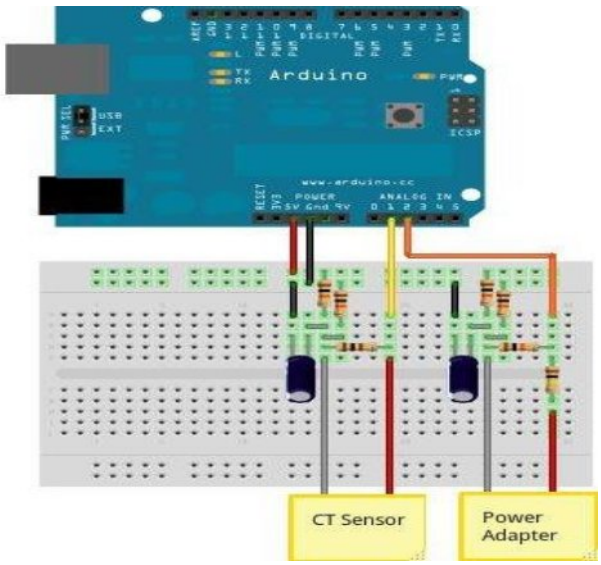

Fig 14: Circuit connections for current and voltage measuremen

3.4 Thingspeak:

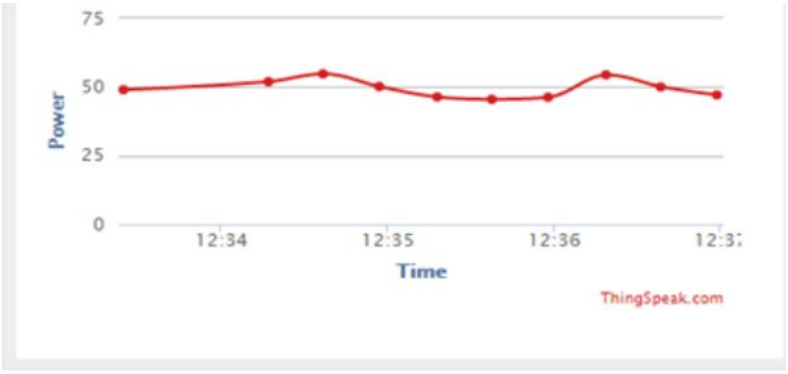

Fig 15: Thingspeak data for power

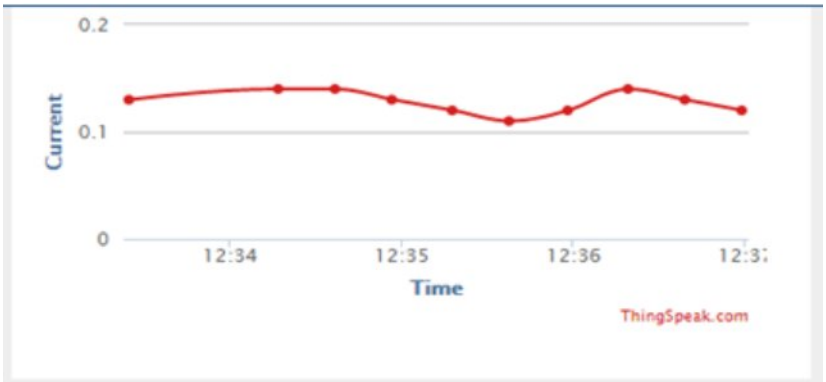

Fig 16: Thingspeak data for current

Using Thingspeak and MATLAB to forecast prices:

Using hourly rates are taken from ISO New England the above screenshot provides the forecasted price(green curve in the left figure) we provide to the user in conjunction with the historical prices over the period of January for years 2012, 2013, 2014 and 2015 acting as datasets to help with the forecast. The forecasting is done with simplistic non-linear regression analysis using **poly-fit** and **polyval**(functions in MATLAB) and using only one predictor variable(previous prices) to make a model that worked the best even by cross-validation after we tested 2015 and 2014's data by adding 2011 and 2010 in the loop respectively.

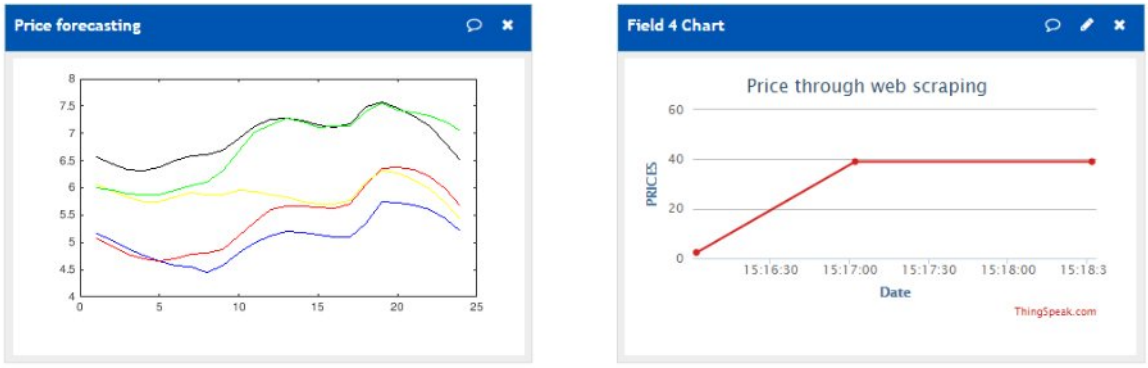

Fig 17: Thingspeak data for power forecasting

Using Thing speak for price scraping:

The screenshot below provides the code implemented for scraping the website of the utility company and obtaining the prices in real-time by uploading it on the Thingspeak Channel.

Fortunately for us retrieving data and writing on Thingspeak barely has any delay and with the Thingspeak twitter option we can notify the user whenever it goes above or below a certain value/threshold.

Name

Scrape prices from a website

MATLAB Code

```
url = 'http://forecast.weather.gov/MapClick.php?lat=42.29&lon=-71.36#.VT5UgiFVhBc';

writeChannelID = [ 180855];

writeAPIKey = 'GCQFO3L1HOEOXE8V';


tempF = urlfilter(url, '<p class="myforecast-current-lrg">');

display(tempF, 'electricity price');

thingSpeakWrite(180855,'Field',[4],'Values',{tempF},'WriteKey','GCQFO3L1HOEOXE8V');
```

My Channels | Documentation

New Channel

Channel Info

Name: Smartmeter
Channel ID: 180855
Access: Public
Read API Key: DFZUWP7FD9STFS38
Write API Key: GCQFO3L1HOEOXE8V
Fields:
1: POWER
2: CURRENT
3: FORECAST PRICES
4: PRICES

3.5 Workflow

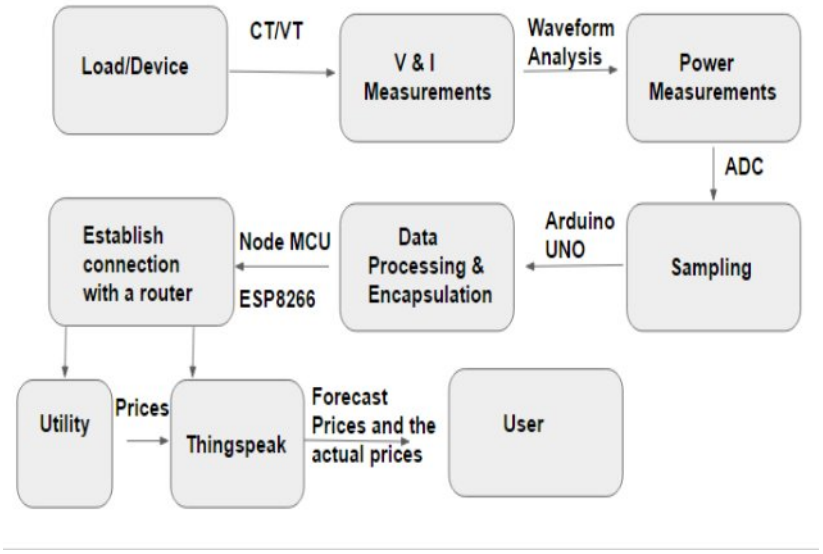

Fig 18: Work flowchart

Chapter 4

Software

The software side of the system was split into the programs running on the Arduino measurements, using the ESP8266 as an access point and using MATLAB to forecast prices and also scrape for prices provided the URL. The Arduino was responsible for both capturing the output of both current sensors, and sending the measured data with timestamps over the wireless connection to Thingspeak via ESP8266.

4.1 Arduino software

4.1.1 Data capture

Sampling of the voltage output of the current measurement circuit was done using the ADC on the Arduino UNO (ATMega328). The Arduino UNO included a family of functions for controlling and retrieving data from the ADC. The initial setup of the ADC was done using the *analogReference(SOURCE)* command, which set which reference voltage the ADC will use in its conversions. The external voltage reference option was set for this project in order to use a reference voltage of 5V. The function *analogRead(port)* returned a value between 0 and 1023, reflecting the level of voltage on the specified analog pin. *AnalogRead()* typically took around 100 μs to execute and return the conversion value.

The smoothing of the half rectified voltage waveform was handled in software, as opposed to with a smoothing circuit or other filter. In each iteration of the main program, *analogRead()* was called 250 times, and the maximum value returned was saved. The voltage waveform had a period of about (1/60Hz), or 16.7ms. Sampling 250 times typically took around 25ms, ensuring that at least a full cycle was sampled. Shorter times were experimented with, but sampling periods lower than this sometimes resulted in "dropped" measurements, where the maximum voltage seen was either during the zero portion of the period, or not near the maximum. The maximum value seen was reset when a packet was sent.

4.1.2 Wi-Fi connectivity and networking

The system uses ESP8266's open source implementation of μIP and a modified version of their example UDP communication app. μIP is an open source TCP/IP stack designed for embedded

systems, with the ability to run a full TCP/IP stack on an 8 or 16 bit system with very limited RAM and instruction memory. A major reason the ESP8266 was selected over other Wifi Modules solutions for embedded systems was because μIP was already ported and released as open source code. This made prototyping of the system far faster and more focus on current and power measurement rather than networking protocols.

The ESP8266'sAT firmware contained all of the included functionality built in. In order to select a configuration, define statements in the header file "apps-conf" were uncommented. The UDP sending function of the system was based off the example UDP endpoint example, and thus the corresponding define statement was added to the configuration header. In addition, the configuration header for the μIP stack was altered to turn on UDP support.

The wireless connection set up was defined by a set of global variables define as the AT Commands. Parameters were recorded in these global variables at compile time and sent to the ESP8266 module by *init()*. These parameters included the SSID, passphrase of the network, security type, IP of the WiFi Module and gateway, and subnet mask. The initialization of the connection was run during *setup()*, and blocked continuation of the program until a connection was established. The system was tested on a WPA2 encrypted network, thus the ESP8266 had to calculate the PSK from the passphrase given, adding substantial time (on the order of 30 seconds) to the set up phase of the program. Using an open or WEP encrypted network would eliminate this, but the added security of WPA2 was considered an acceptable trade off.

Chapter 5

Results

Using this method, the system displayed an error of 0.96 from the output of the clamp meter. The quantization error of the ADC was 0.75 at 1 A, and the two burden resistors had an error of 1.8% and 1.6%. The error displayed in this test can mostly be attributed to these factors. When measuring higher currents, the quantization error would decrease linearly, meaning measuring larger currents would be more accurate. The error rate observed here can be thought of as a worst case scenario in this regard, with a lower error rate during normal operation.

A 20 minute test of measuring current through the main breaker yielded the results. The large initial spike in voltage is possibly an artifact from the current transducer itself reacting to power on. The large changes in current were attributed to the halogen lamp which was on during the testing. The next smallest change in current is thought to be a heating iron, but is not known for certain.

	Halogen Lamp	Heating Iron	Laptop Charger
Current(Measured)	1.63 A	0.87 A	0.18 A
Actual Current	1.6 A	0.9 A	0.2 A
Power(estimated via the device)	17.44W	213W	58.8W
Rated Power	20W	250W	65W

Table 1: Our readings data

Chapter 6

Conclusion

The system was successful in measuring current within an acceptable range of error, and sending that information at a higher update rate. The project was a valuable experience in the design, implementation, and testing of a system that involved several discrete hardware and software components. The use of an open source project for such a central function as the access point and energy measurement in the project ended up greatly accelerating the design of the meter.

Ultimately the system accomplished its primary goal of presenting energy information to a user in a clear way. An individual could watch the interface and visibly see the effect of leaving an appliance or device on, or see the difference in remembering to turn the lights off when they are not needed. More functionality in this regard would have been helpful, specifically long term averages and either numerical or visual comparisons between older data and present data.

Future Scope

Smart meters provide us huge volumesof data via generation and collection. With smart-meters, this could be time-interval consumption data as well as aggregated values for billing purposes. This enables profiling of consumer behavior and relates consumption to extrinsic parameters like temperature changes, humidity, time of the week, major holidays etc. The major requirement for creating such a model is taking help of neural networks and creating more comprehensive models all the while keeping in mind that over-fitting would lose the essence of forecasting. It is this fine balance forecasting must accomplish, and only constant cross-validation can help in that regard. Also, establishing a more direct link from the utilities to the user instead of using the Thingspeak code to scrape data is recommended. All these would contribute hugely in ensuring better awareness and bridging the supply demand-gap at peak hours, thereby flattening the load curve—the essence of Smart energy management.

References

1. M.P.Selvan, AshaRadhakrishnan. “Load scheduling for smart energy management in residential buildings with renewable sources.” IEEE2014
2. Martin Liska, Marian Ivanic, Vladimir Volcko, Peter Janiga. “Research on Smart Home Energy Management system.” IEEE 2015
3. DammindaAlahakoon, Xinghuo Yu. “Smart electricity meter data intelligence for future energy systems.” IEEE 2016
4. X. Yu, C. Cecati, T. Dillon, and M. G. Simoes. “New frontier of smart grids.” IEEE 2011
5. Ja.Sruthi, R.L.Helen Catherine. “A Review on Electrical Load Forecasting in Energy Management.”IJISET 2015
6. Smart Meters and Smart Meter Systems: “A Metering Industry Perspective.”EEI-AEIC-UTC White Paper.
7. P. Palensky and D. Dietrich, “Demand side management: Demand response, intelligent energy systems, and smart loads.” IEEE
8. https://openenergymonitor.org/emon/Overview

Appendix

Code:

```
#include <SoftwareSerial.h>
#include "EmonLib.h"
EnergyMonitor emon1;
intsensor_current = A1;
intvalue_current = A2;
#define DEBUG FALSE                //comment out to remove debug msgs //*-- Hardware
Serial
#define _baudrate 9600              //*-- Software Serial//
#define _rxpin      2
#define _txpin      3
SoftwareSerialdebug( _rxpin, _txpin );              // RX, TX

#define SSID "IIT indore(tC 151f)"
#define PASS "jio1234!@#$"
#define IP "184.106.153.149"                  // ThingSpeak IP Address: 184.106.153.149

                //GET/update?key=[THINGSPEAK_KEY]&field1=[data1]&field2=[data 2]...;
String GET = "GET /update?key=GCQFO3L1HOEOXE8V";
//----- update the  Thingspeak string with 3 values

void setup()
{
  Serial.begin( _baudrate );
  debug.begin( _baudrate );

  sendDebug("AT");
  delay(5000);
  if(Serial.find("OK"))
  {
    debug.println("RECEIVED: OK\nData ready to send!");
    connectWiFi();
  }
   // emon1.voltage(A2, 234.26, 1.7);  // Voltage: input pin, calibration, phase_shift
  emon1.current(A1,10);       // 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 RP      = emon1.realPower;        //extract Real Power into variable
  float AP   = emon1.apparentPower;    //extract Apparent Power into variable
```

```
  floatpf    = emon1.powerFactor;      //extract Power Factor into Variable
  float v   = emon1.Vrms;             //extract Vrms into Variable
  float c  = emon1.Irms;             //extract Irms into Variable
  String POWER =String(RP);// turn integer to string
  String CURRENT = String(c);
  updateTS(POWER,CURRENT);
  delay(3000);
}

voidupdateTS(String C,String P)
{
  // ESP8266 Client
  String cmd = "AT+CIPSTART=\"TCP\",\"";// Setup TCP connection
  cmd += IP;
  cmd += "\",80";
  sendDebug(cmd);
  delay(2000);
  if(Serial.find( "Error" ) )
  {
   debug.print( "RECEIVED: Error\nExit1" );
   return;
  }

  cmd = GET + "&field1=" + C + "&field2="+ P + "\r\n";
  Serial.print( "AT+CIPSEND=" );
  Serial.println(cmd.length() );
  if(Serial.find( ">" ) )
  {
   debug.print(">");
   debug.print(cmd);
   Serial.print(cmd);
  }
  else
  {
   sendDebug( "AT+CIPCLOSE" );//close TCP connection
  }
  if(Serial.find("OK") )
  {
   debug.println( "RECEIVED: OK" );
  }
  else
  {
   debug.println( "RECEIVED: Error\nExit2" );
  }
}

voidsendDebug(String cmd)
{
```

```
  debug.print("SEND: ");
  debug.println(cmd);
  Serial.println(cmd);
}

booleanconnectWiFi()
  {
   Serial.println("AT+CWMODE=1");//WiFi STA mode - if '3' it is both client and AP
  delay(2000);
  //Connect to Router with AT+CWJAP="SSID","Password";
  // Check if connected with AT+CWJAP?
  String cmd="AT+CWJAP=\""; // Join accespoint
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  sendDebug(cmd);
  delay(5000);
  if(Serial.find("OK"))
  {
   debug.println("RECEIVED: OK");
   return true;
  }
  else
  {
   debug.println("RECEIVED: Error");
   return false;
  }

  cmd = "AT+CIPMUX=0";// Set Single connection
  sendDebug(cmd );
  if(Serial.find( "Error") )
  {
   debug.print( "RECEIVED: Error" );
   return false;
  }
       }
```

Figures

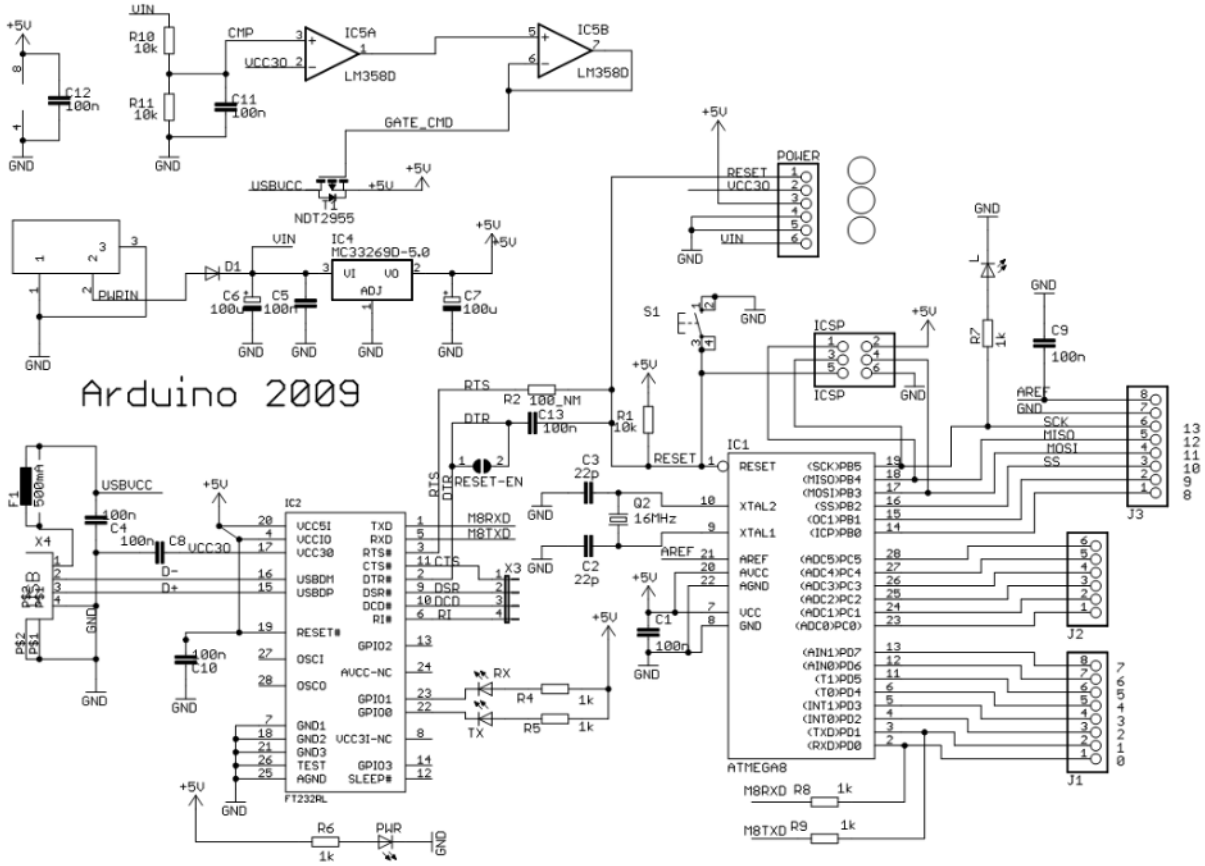

Fig 19: Schematic diagram of Arduino Uno

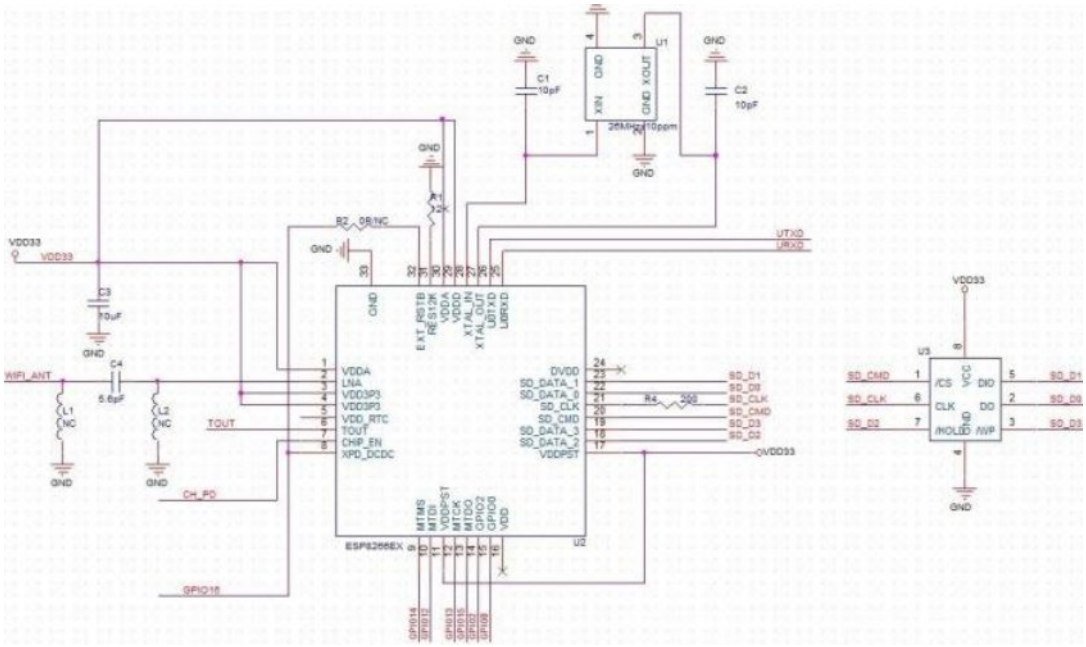

Fig 20: Schematic diagram of ESP8266

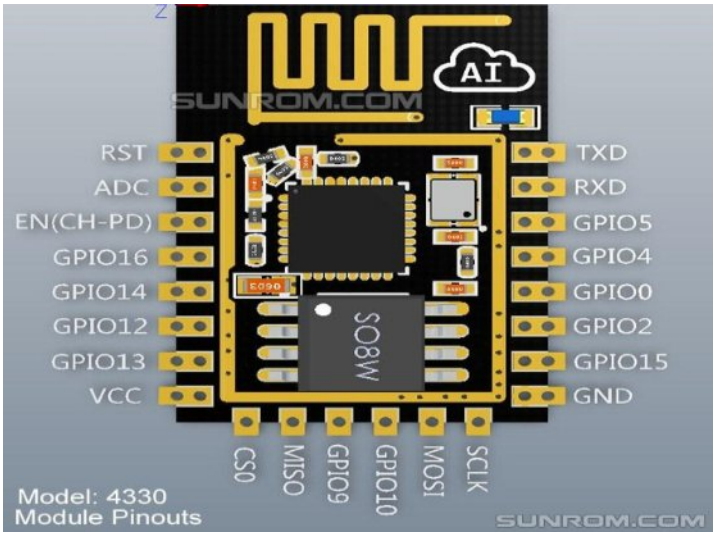

Fig 21: ESP8266 pins

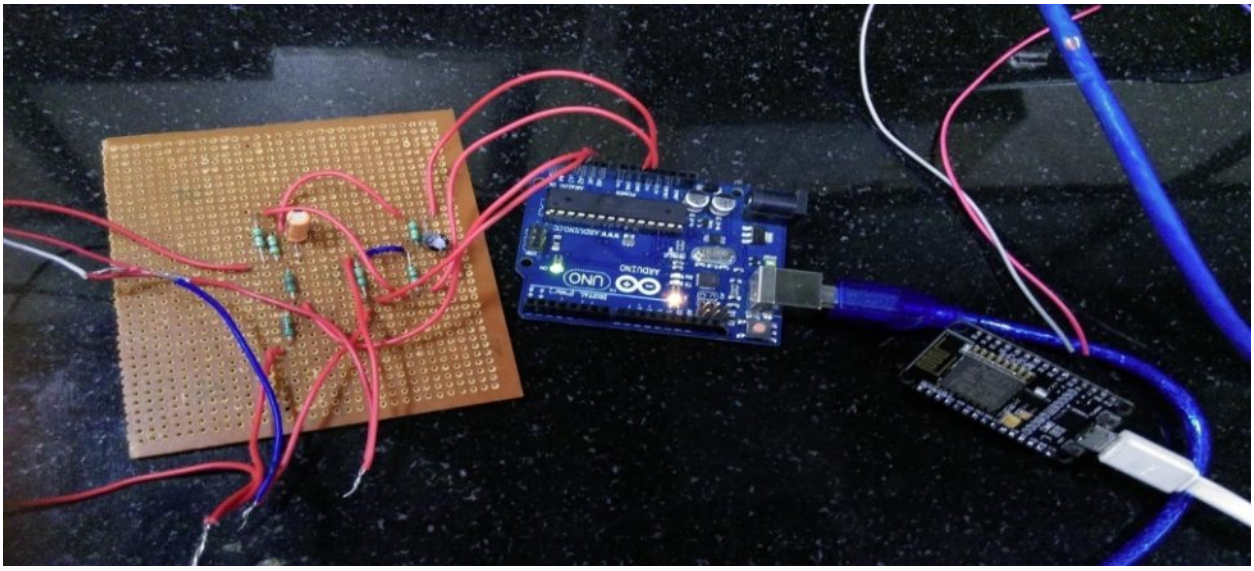

Fig 22: Executed circuit