# ANALYSIS OF ELECTRICITY SUPPLY DATA

John-Paul Blazevic

Bachelor of Engineering
Electronics Engineering Major
Mechatronics Engineering Minor

Department of Electronic Engineering
Macquarie University

November 5, 2017

Supervisor: Professor Graham Town

MACQUARIE UNIVERSITY

DEPARTMENT APPROVAL

of a senior thesis submitted by

John-Paul Blazevic

This thesis has been reviewed by the research advisor, research coordinator, and department chair and has been found to be satisfactory.

| | |
|---|---|
| Date | Supervisor: Professor Graham Town, Advisor |
| Date | Department Ugrad Coordinator , Research Coordinator |
| Date | Department Chair Name, Chair |

## ACKNOWLEDGMENTS

I would like to acknowledge and extend my sincere thanks to my supervisor Professor Graham Town for his help throughout the duration of this thesis. His understanding, guidance and flexibility has allowed me to research into a topic of which I am passionate about and which I believe has significant value.

## STATEMENT OF CANDIDATE

I, John-Paul Blazevic, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: John-Paul Blazevic

Student's Signature: John-Paul Blazevic (Electronic)

Date: 05/11/2017

## ABSTRACT

Determining the power consumption and switching times of a building or households connected loads and appliances is crucial for energy analysis and optimization which is increasingly becoming more relevant in today's society due to the implementation of renewable technologies and soaring energy prices. This project looks to analyse current techniques for determining electrical load characteristics from an aggregate electrical source such as a buildings electrical mains. In this project a test and simulation setup is constructed. Using a non-intrusive power metering device aggregate electrical power data is collected and analysed using the techniques explored to determine individual load characteristics and to autonomously determine when individual loads are switched on and off. This is done so individual appliances power consumption can be estimated or appliance switching times used for automation purposes.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

As renewable technologies such as solar, wind and energy storage solutions are introduced into homes and buildings, coupled with increasing energy prices for base load power there is also an increasing demand and value for determining the power consumption and usage times of different electrical loads or appliances. This is valuable information for users who wish to optimize their energy consumption for environmental or economic reasons. Due to the inefficiency and complexity of attaching power metering equipment to multiple household appliances approaches have been examined to extract unique features of individual loads from an aggregated source such as the households mains supply. This reduces hardware to a single metering device capable of capturing the necessary electrical signals, such a concept is known as Non-intrusive load monitoring (NILM) [2, 5, 10, 11].

This thesis attempts to examine data sets collected of electrical power, current, voltage and power factor over specific periods to build a database of individual appliance's electrical characteristics at switching events (when loads turn on and off) and also determine from a simulated aggregate source (multiple loads connected in parallel) when these loads turn on/off and approximate their consumption over a period by comparing electrical characteristics generated with those formulated in the load database.

## 1.1 Non-Intrusive Load Monitoring

Non-intrusive load monitoring is a method for monitoring electrical loads using current measuring sensors that do not need to be in series with the current flow but rather attached outside the conductor providing both easy installation and safe use [3]. Non-intrusive load monitoring also involves determining various features of a loads electrical characteristics such as current draw for condition monitoring applications [5]. However it is also used in many applications for determining which loads are switching in an aggregate electrical signal source, such an application being applicable to this project.

## 1.2 Project Overview

This section will describe a brief overview of the whole project, the approach taken and structure of this thesis. The thesis builds on previous approaches to NILM to produce a system that can function usefully for the particular application used here.

The project is predominately software based with the main deliverable being a software application possessing the capabilities mentioned in the project goals below. Hardware is used supplementary to this software in order to simulate data expected from the projects application. The main project goals are itemised below;

- The ability to detect load switching events in aggregate electrical consumption data.
- The ability to successfully produce a load characterization (load signature) from different electrical properties at these events.
- To store load signatures of individual loads in a database or array structure.
- To classify the aforementioned signatures formulated at events with those in the database.
- To then present to the user individual load switching times and approximate power consumption of the load over a period.
- The system should be able to use data collected from relatively inexpensive hardware so as to be a feasible solution for households.
- For the system concept to be straight forward and non-convoluted.

For the scope of this project the idealised application for electricity supply data analysis is for the end user in a household. This is the assumed usage case for which the system will be designed due to hardware and software capabilities available as well as system complexity, although similar solutions can be extended to industrial applications.

The approach taken to NILM in this project is analysis of steady-state active, reactive, and apparent power signals between appliance state transitions [1]. This was found to be most effective for the NILM using the capabilities of available power monitoring hardware.

### 1.2.1 Organization of Thesis

The structure of this thesis flows as would be expected of a project requiring testing and experimentation, where each chapter follows the developments and milestones reached throughout the research process.

Chapter two recounts the background research and current approaches to NILM. It describes the various types of load signature formulation methods describing some benefits, drawbacks and challenges. By the end of the chapter an approach is selected to be used for this project and modifications that could be made.

Chapter three looks further at the approach taken and assesses the feasibility of each technique.

Chapter four provides information about the hardware and software tools used in this project. It provides background on the chosen technologies and describes their use in the system.

Chapter five details the design of the system it refers to the approach taken and how it is to be implemented. The chapter then goes on to describe how the system is implemented and the steps that where taken.

Chapter six defines the testing methodology and defines the metrics used to test the system. It goes on to describes each test, the results and discuss conclusions derived from the results, pointing out limitations.

Finally Chapter seven describes the main conclusions from the system tests and of the project. It sets up what future work should be done in regards to the project, its limitations and how to improve the system.

# Chapter 2

# Current Approaches to NILM and Background Information

## 2.1 Load Types

Most NILM implementations differ by their success rates of identifying different types of loads [12]. This difference between loads is evident when the electrical characteristics of the load throughout their operational period is concerned.

Load types can be classified into four main categories;

- Type I
- Type II
- Type III
- Type IV

Type I loads represent those with two basic operational modes, on and off [1]. These loads are the most simple to identify as after state transitions they will draw quite nominal currents and power. Examples of these type of loads may include heaters, or lighting.

Type II loads on the other hand are loads which have multiple modes of operation [1]. One example of this is a washing machine which will operate with different power consumption and current draw over its operational period [1]. First it pumps the water, heats it and then spin the drum. Type II like type I have state changes and thus can be easier to detect then type III, however as operational modes may be numerous and over large periods of time it will prove difficult to formulate loads signatures that represent the load as a whole [1].

Type III loads are those which are continuously varying over time such as the power draw of a computer as CPU load increases [1]. These are surely the most difficult to characterize as there is no clear basis for the steady state power draw of the appliance, however some approaches provide solutions which prove useful.

Finally, type IV loads are those which are constantly consuming power and rarely change state. A good example of this is the alarm system or smoke detectors [1]. The figure shown below demonstrates how each of these load types behave over time.

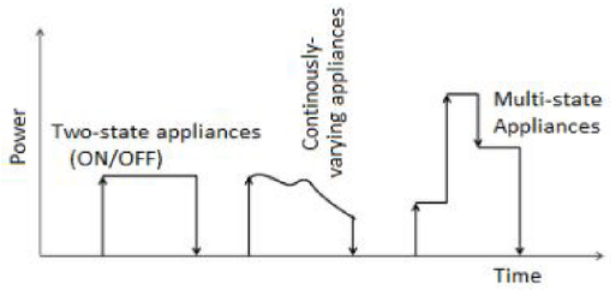


**Figure 2.1:** Load power consumption over time. Operational modes of loads [1].

## 2.2 Transient Methods

Transient methods of load identification refer to the unique response in a loads current draw as the load starts or through the use of load power profiles and their spectral envelope [2]. The one distinct benefit of transient analysis is the lack of overlapping feature sets for some loads as can be found in steady state approaches [1, 5]. Limitations of this approach that make it less suitable for this project application is the inherent need for high sampling rates so as to detect and accurately represent the response [1, 13].

For type I and type II loads it was found that the shape of a loads transient can be used for unique characterization of the load. [1, 2]. Similarly turn on transients also provide distinct features for load feature formulation for these load types. The latter includes determining current or power spikes which occur during some loads turn on state [1, 14].
Such responses are shown in figure 2 where it can clearly be seen the unique turn on transient shape, peaks, and response time. Because it deals with responses at state changes it is clear that it is a useful tool for determining type I and II loads [2, 15].

It is also possible for transient methods to determine continuously varying loads such as variable speed drives (VSD's) found in numerous industrial sites [5]. Approaches such as [2, 5] use spectral envelopes to try to remove the continuously varying power signal from the aggregate real power by subtracting the higher-order harmonic components that contribute to its generation [5]. This approach is not covered in this project.

It should be noted also that transient methods are not ideal for detecting type IV loads as the nature of these loads do not include frequent turn on events and thus transient responses are not displayed [1].

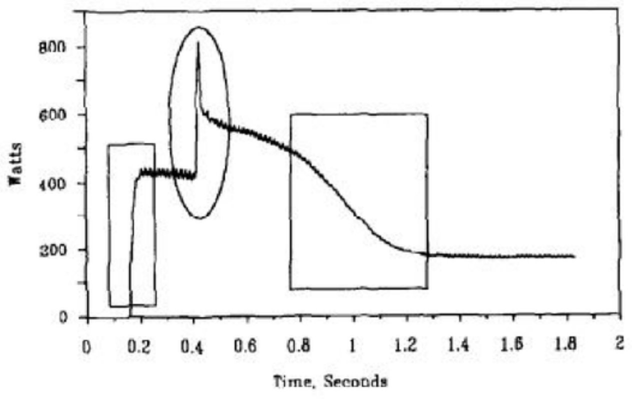


**Figure 2.2:** Load transients of a motor turn on, lamp turn on and motor reaching steady state [2].

## 2.3 Steady-State Methods

Steady state methods are the predominate types used by majority of papers [1, 5, 10, 12, 14, 16]. The nature of steady state allows for lower sampling rates of aggregate power consumption over certain periods. Steady state analysis usually requires periods to be examined "after the fact" thus real time identification is not possible. Near real time identification at resolutions of at least one second is possible through the use of techniques such as current harmonics for load characterization found in [10].

### 2.3.1 Power State Changes

Approaches such as [5] deal with state changes in real $P$ and reactive power $Q$. Real power (active power) is that which is able to generate meaningful work whereas reactive power constitutes a component of the overall power that stores and released energy back to its source through electrical or magnetic fields generated by the device. Apparent power on the other hand is the overall current drawn $I_{RMS}$ multiplied by the overall voltage $V_{RMS}$ and the three are related through the relationship described in equation 1.

$$|S| = \sqrt{P^2 + Q^2} \tag{2.1}$$

Unlike transient study, steady state methods based on these variables take the magnitude of $P$ and $Q$ before and after state change events to determine the power consumption of a particular load [5]. This is seen to be representative of the load with confusion arising when small loads are concerned [3].

The loads types that are most easily determined through state changes in $P$ and $Q$ are type I loads as described by [1]. The conclusions made from [1, 5] describe the approach being suitable for high powered devices that do not overlap in the $P - Q$ plane with such a distribution being shown in figure 3. It is clear from [3,5,17] and figure 3 that lower powered devices present overlap and are hard to discern with simply these two variables.

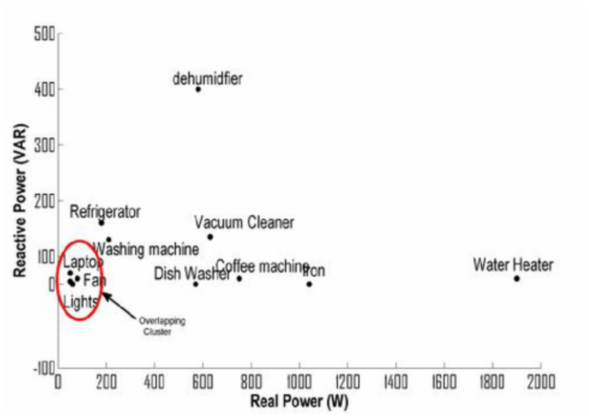


**Figure 2.3:** Distribution of loads in the $P - Q$ signature plane [3].

One important drawback to many of the approaches taken in terms of $P - Q$ analysis is accounting for fluctuations in voltage [3]. As described in [3] the voltage can fluctuate over time by up to 10 percent and for linear devices this means that the power can also fluctuate by 20 percent [3]. The method described in [3] is to use then the admittance $Y(t)$ to account for these fluctuations in voltage given by equation two [3].

$$Y(t) = \frac{P(t)}{V^2(t)} \tag{2.2}$$

The paper goes on to describe that it best to use the admittance in normalized power $P_{norm}$ shown in equation three [3] which will provide power data normalized around a steady supplied voltage which in Australia is $240V$ [3].

$$P_{norm} = \left(\frac{240}{V(t)}\right)^2 P(t) \tag{2.3}$$

This provides a robust solution to improve accuracy of steady state power changes for load signature formulation as will be shown in chapter 3 where the voltage is indeed found to fluctuate during different conditions.

### 2.3.2 Voltage and Current Waveform Analysis

Some loads namely nonlinear loads will often draw currents that are non-sinusoidal, that is they draw currents with higher order harmonics that distort the shape of the current waveform such as that shown in figure 4 [1]. The changes in amplitude of these higher order harmonics can be utilized as additional features as done in [10], however the drawback of this of-course is the higher sampling rate needed to detect harmonic content.

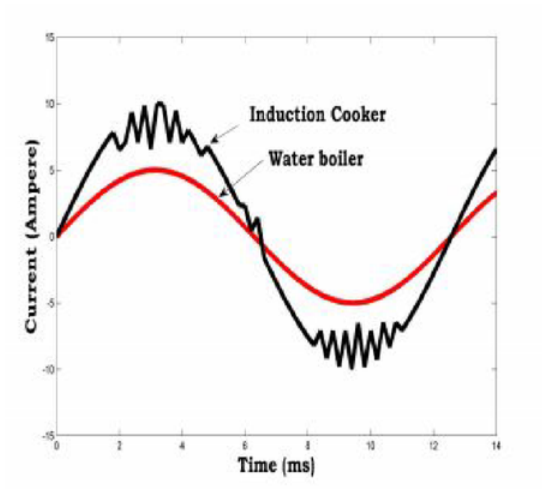


**Figure 2.4:** Current waveform of an induction cooker versus a water heater. Introduced harmonic currents [1,4].

Initial NILM study done by MIT used $P-Q$ analysis for load identification and in [5] has found as previously discussed more load signature features should be added in order to better segregate individual loads [5]. In the case of [5] the third order harmonic component derived from the short time Fourier transform of the current waveform is used.

This is represented best in figure 5 where it can be seen that even though $\Delta P$ and $\Delta Q$ are almost overlapping for the two devices the third order harmonic current is what separates the two [5]. A similar approach of using changes in harmonic currents $\Delta I_{harmonics}$ is used in [10] to create load signatures that are based on the first three odd order harmonics, the fundamental, third and fifth in the current power spectrum [10].

This particular approach applied the FFT to the current waveform captured at one second intervals, thus is described as being capable of near real time identification [10]. By collecting the amplitudes of the first three odd order harmonics $I_{50}$, $I_{150}$, $I_{250}$, the study aims to increase the identification accuracy specifically for nonlinear low voltage loads [10].

This approach does need a high sample rate in the order of at least 500-1000Hz as dictated by the Nyquist Theorem. Like others this study [10] uses a training mode in which 300 samples are taken and standard deviations are calculated in order to produce an acceptable baseline for load signature formulation [10]. The standard deviation allows for the load signature to incorporate flexibility for type II and III loads, those who's operational modes change or vary within five minutes of turning on [10].

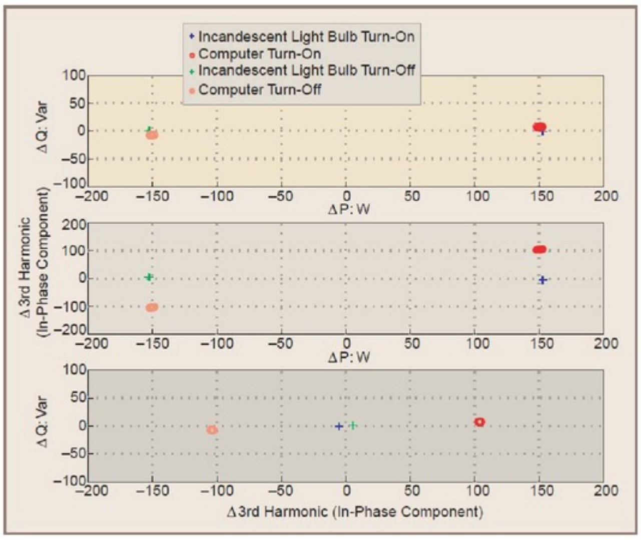


**Figure 2.5:** $\Delta I_{150Hz}$ - $\Delta Q$, $\Delta I_{150Hz}$ - $\Delta P$, and $\Delta P - \Delta Q$ Planes for two loads at turn on and off events [5].

The paper goes on to describe that loads for which operational modes change at times greater than five minutes should be monitored separately and added to the sample space for SD determination [10].

Load signatures are created in the form of $LS_i = [[I_{50}], [I_{150}], [I_{250}]]$ where each component will have a number of samples representing the operation over it's five minute period [10]. After LS formulation for individual loads a similar approach is used for formulation of LS's at turn on and off events however with only one sample needed in the steady state waveform proceeding each event [10].

The conclusion of current harmonics from [10] is that while high identification rates are observed for some load combinations other combinations produce poor results due to the phase angles introduced by different appliances which can cause the overall amplitudes at the higher harmonics to be influenced [10]. The study proposed voltage measurement to be implemented so that the phase of each harmonic can be understood and utilized.

# Chapter 3

# State Changes in Power Space and Power Transient Features

## 3.1 Introduction

Developing a successful system for load disaggregation requires the formulation of appropriate load signatures. These signatures should contain independent features and be unique between different loads as this will make classification significantly more accurate.

As described in the review in chapter two each method of load signature formulation has its distinct benefits and drawbacks, namely the load type best identified by that method and the complexity of implementation. It is determined that the most simple approach is to formulate a load signature based upon the steady state changes in real and reactive power at load switching events as achieved by [18].

The approach examined in this section draws on the benefits of both techniques while still maintaining some significant challenges including:

- Signature space overlap for low voltage appliances. [1]
- Poor recognition of type III appliances. [1]
- Detection errors during simultaneous load switching, assumes steady-state between events.
- Power fluctuations due to voltage drift. [12]

### 3.1.1 State Changes in Power

As reviewed the more features added to a load signature the better the discernment between different loads. For this reason we can focus on using state changes in a multitude of different signals, real and reactive power are chosen however apparent power may also be included in this space as for many non-resistive loads real, apparent and reactive power all give distinct magnitudes. This is evident by the difference between power factors for individual loads. Granted, if a load is purely resistive its power factor approaches unity and real and apparent power become very similar as the phase angle between the current and voltage waveform decreases. This may present a challenge when discerning between resistive loads that also have very similar power consumption, or phase corrected devices.

Since the application chosen for this thesis is specific to households many of the concerns associated with NILM become lesser although still present. Households generally have fewer appliances than large industrial buildings and such appliances are generally quite unique. For example the typical loads one would be most concerned with in terms of power consumption may be air conditioning, televisions, computers and monitors, stoves, ovens, heaters, and washing machines. Each of these devices are quite unique in their consumption magnitudes and load type. This gives reasonable confidence to using $\Delta P$, $\Delta Q$ and even $\Delta S$ as a basis for load identification.

The hardware benefits of using these three power features are obvious, high sample rates are not needed and high identification results can be achieved for type I loads [1]. However since start-up transients of devices can last for significant amounts of time it proves difficult to determine when steady-state conditions are met [5].

An example of load state changes in the $\Delta P - \Delta Q$ plane is shown in figure 1. It is evident that for loads that draw similar real and reactive power it is difficult to discern between the two using only these two features.

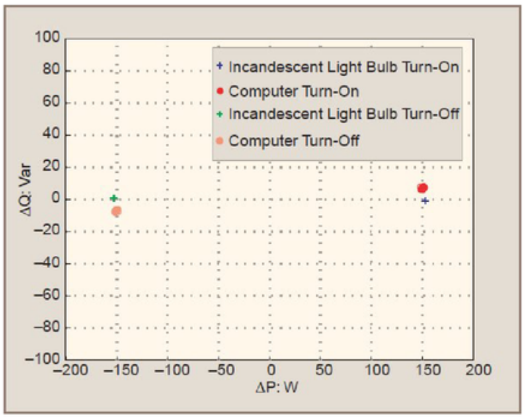


**Figure 3.1:** Magnitudes of real and reactive power for a computer and incandescent lights [5].

### Additional Steady-State Features

Additional steady-state features can be applied to the load signature to increase identification accuracy. One such feature which is currently being examined is the average fundamental current draw or $\Delta I_{50}$. This is the average 50Hz current being drawn by a particular appliance and has proven thus far to be quite static for individual loads even with different aggregate load combinations.

Current used to calculate power values is the overall average current $I_{rms}$ whereas this feature deals strictly with the first harmonic in the current spectrum. This is useful as loads not only draw distinct magnitudes of current but some loads such as computers have been observed to draw current at higher odd harmonics [5] and thus this feature may help to differentiate between loads with similar power consumption.

This is seen to be true in analysis when $I_{rms}$ is compared to $I_{50}$ for a computer monitor shown in figure 2 and 3 respectively where the fundamental current is significantly less although still follows the same power profile.

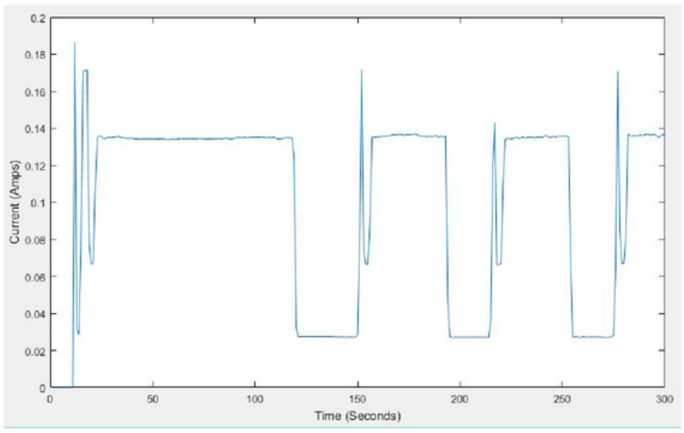


**Figure 3.2:** RMS Current draw of a monitor over a series of switching events.

### 3.1.2 Transient Features

Perhaps the most distinct difference between loads is the way in which they reach steady-state conditions in both current draw and power consumption. The transient response of a load particularly at turn on events is often quite unique. This is shown clearly in figure 4 where the start up transient of first a computer monitor, then a soldering iron, and finally a florescent lamp can be seen. Clearly the shape and response of the start-up transient is unique for these loads even at a sampling rates of one second.

Particular characteristics of transients can be quantified from simple operations. For example, once the beginning and end of steady state periods are identified as displayed by the red and blue markers in figure 4 the settling time, $t_s$, peak value $P_p$ or $I_p$, and even peak time $t_p$ can be determined. These values are observed to be unique as shown in table

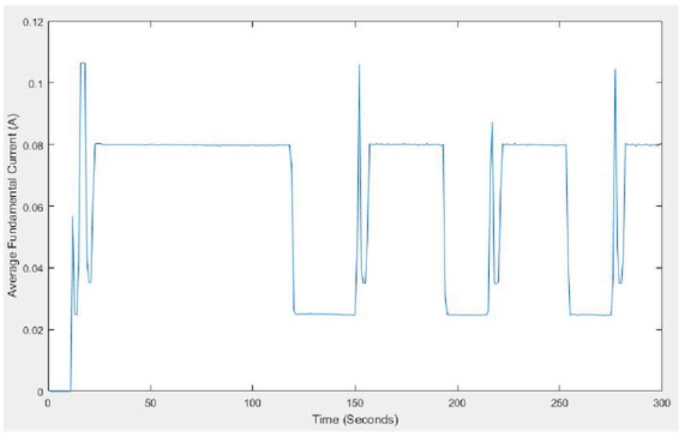


**Figure 3.3:** 50Hz Current draw of a monitor over a series of switching events.

1 below. Comparing these values to the individual load features shown in table 2 highlights that while many values are quite close to those determined by state changes in the aggregate data, the $\Delta Q$ presents a problem of being sensitive to the load configuration.

**Table 3.1:** Transient Settling times and $\Delta P$, $\Delta Q$ values from power data shown in fig 4.

| Load Type | $t_s$ (s) | $\Delta P$ (W) | $\Delta Q$ (VAR) |
|---|---|---|---|
| Monitor | 10 | 17.7 | 19.2 |
| Soldering Iron | 6 | 45.3 | 17.5 |
| Lamp | 7 | 17 | 61.2 |

**Table 3.2:** Transient Settling times and $\Delta P$, $\Delta Q$ for individual load configurations.

| Load Type | $t_s$ (s) | $\Delta P$ (W) | $\Delta Q$ (VAR) |
|---|---|---|---|
| Monitor | 10 | 18.1 | 19.7 |
| Soldering Iron | 6 | 44.45 | 13.8 |
| Lamp | 7.7 | 17.03 | 75.3 |

Once again challenges arise specifically when dealing with these transients at low sampling rates. Since the sampling time here is one second the time responses will generally be at integer values and because transients can often be rapid the likely-hood of similar settling times can be high. Furthermore it is expected as noise is introduced into the system and the aggregate power consumption is significantly higher (kilowatt range) it may prove difficult to determine such peaks depending on the dynamic range and resolution of the metering equipment.

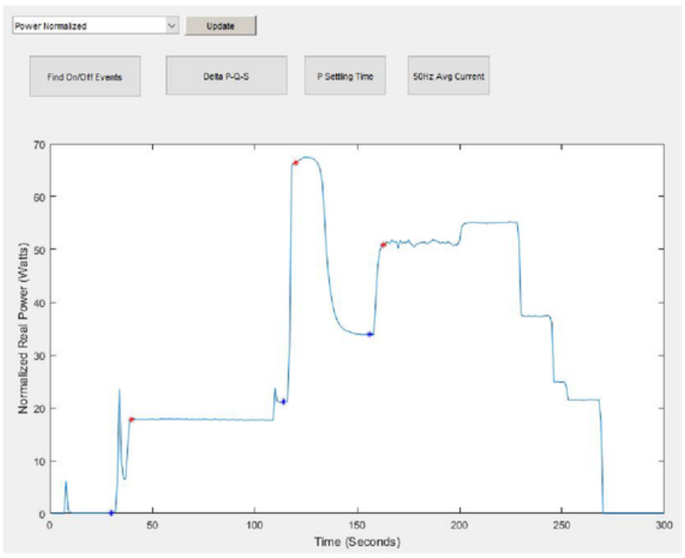


**Figure 3.4:** Aggregate normalized power of a monitor, soldering iron, and fluorescent lamp.

In some cases transients can have significant settling times [5]. This can present a problem when dealing with high event generation as a load signature formulation algorithm would need to wait. This is of particular interest to the event detection function as it must determine an interval between which it will decide is not steady state. Implementation that used Matlab functions to determine when steady state conditions occur and the thresholds concerned where replaced with the designed event detector described in greater detail in chapter 5.

For these reasons it is decided that for such low sample rates using the transient pattern of a load is not feasible, with more sophisticated hardware this problem would be eliminated as responses would be much smoother and well defined. In such a case if the transient settling time is too high it will either be disregarded or a timeout implemented otherwise other state change events would confuse the system. Thus with more sophisticated hardware both steady-state and transient behavior could be implemented.

# Chapter 4

# Technologies and Hardware Used

This project and its implementation requires the use of both hardware technologies and software applications. This chapter will discuss the hardware chosen for data acquisition and the programming paradigms used for the implementation of the system.

## 4.1 Hardware Requirements

Data acquisition hardware is required for this project in order to monitor current and voltage signals from an aggregated source, in most cases this is the fuse box of a household. Furthermore it must be able to record the measurements listed below:

- Real Power
- Reactive Power
- Apparent Power
- Current
- Voltage
- Phase or power factor

While hardware that can achieve this can be purchased this was thought unnecessary as hardware with these capabilities are expensive and the university currently has a range of hardware that has all or at least some of this functionality.

Further more in order to run user interface software a piece of hardware should be chosen that can act as a server for the system. For this task a computer or small embedded device would be suitable. This lead to the selection of a Raspberry Pi.

### 4.1.1 HIOKI PW3360-21 Power Monitor

Ultimately the device used was chosen as it is best suited for this application. This device is capable of measuring all the aforementioned signals and even the ability to measure harmonic components of current up to the 40th order. This is impressive but useless if data acquisition is not possible, thankfully the device is able to store the signals onto an SD memory card in CSV format.

Storing the data is possible over user specified intervals however there is a significant drawback to this piece of equipment, the sampling interval setting between logging is limited to a minimum of one second. This limits the diversity of the project, and additionally only the fundamental harmonic current is logged along with the average, peak and maximum currents every second.

While the ability to monitor high sample rates of electrical signals is lost, accurate and

**Figure 4.1:** HIOKI PW3360-21 Power Quality Meter Used. [6]

relevant data is available which correlates to a large portion of the examined NILM approaches mentioned in chapter two. Furthermore having one second data over specified periods allows for the easy importation of data-sets into the analysis software and a time resolution of events to the nearest second.

The capabilities of the hardware available significantly shapes the direction of the project however the capabilities mentioned do align with the application of the project. Being mainly aimed at household usage and for a relatively small number of specific appliance that a user may want to monitor, keeping in mind that users will not necessarily want to monitor every electrical device installed in the house but rather those which represent a significant portion of their electricity consumption.

Considering these factors it is determined that this hardware should provide the capabilities for the load signature approach chosen in chapter 3. Also it is important to note that while even this hardware is expensive because of the features unused the data available is representative of what we could expect from cheaper and lower grade power monitoring hardware. Thus using this hardware to acquire one second sampled power data simulates real time data acquired from a custom designed monitor or other commer-

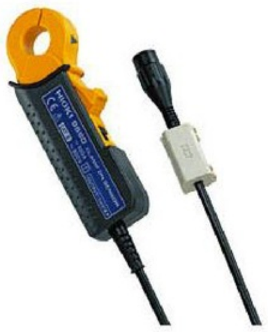


**Figure 4.2:** HIOKI 9660 100Amp Non-Intrusive Current Probe Used. [7]

cially available power monitors.

To summarize the benefits of using this hardware:

- Non-intrusive current monitoring with both 10Amp and 100Amp probes.
- Line voltage monitoring
- Real, Reactive, and Active Power consumption data.
- Average, Maximum, Peak, Minimum and fundamental current data.
- Voltage data
- Power factor, lagging, leading.

This monitor shown in figure 4.1 is battery operated thus can be left to log over extended periods of time to capture household data sets. Furthermore by using a readily available device no budget is needed to purchase hardware.
As stated above the hardware is non-intrusive such that it uses the current sensor shown in figure 4.2 to measure the current flowing through the primary cable. A setup of the system is shown later in chapter 5.2.

### 4.1.2 Raspberry Pi Server

The Raspberry Pi is what is known an embedded single board computer or SBC. It was created by the Raspberry Pi foundation as a platform for learning programming and computing however due to its small size, capable hardware and Linux based operating system it has quickly become an industry standard for rapid prototyping of embedded system applications.
The community surrounding this hardware is extensive and because of this it has become

a standard for home automation and IoT (internet of things) applications among hobbyists and professionals alike.

For these reasons and the forward thinking idea of producing this project into a embedded solution the Raspberry Pi Zero (smallest version) shown in figure 4.3 was chosen to run the server applications described in chapter 4.2. Also this device was on hand thus minimizing project costs further.
It can be powered off a 5V USB power supply and consumes less than 5 Watts of power making it a very efficient computer for a energy monitoring project. It has TCP/IP networking capabilities using a USB to Ethernet adapter which will allow it to communicate to the dissagregation software developed which will be operating on separate computer to simulate a standalone power monitor.

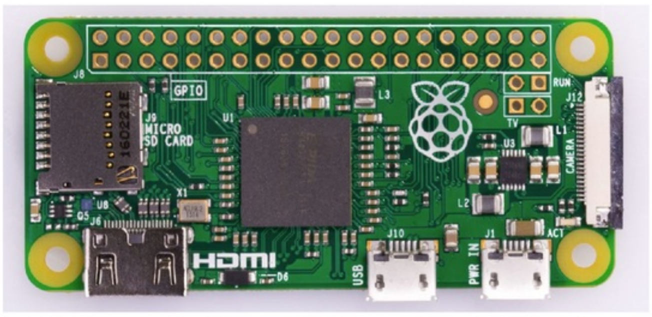


**Figure 4.3:** Raspberry Pi Zero used as a server for this project. [8]

## 4.2 Software Techologies Used

Since the main goal of this project is the development of a software application capable of analyzing electrical signals to determine appliance information, then software technologies must be chosen to achieve this. For this project there are two main software paradigms used, these being;

1. Matlab
   Matlab is a well known mathematical modeling application capable of a diverse range of mathematical tools and algorithms. It was as the main programming environment used for this project due to its intuitive graphical user interface (GUI) design environment called GUIDE and its ability to simply produce plots and figures of all data processed.

2. Node-Red
   Node-Red is a flow based programming paradigm based in Java Script which makes developing web based interfaces and designing automation and IoT systems much more simple and intuitive.

3. Mosquitto MQTT Broker
   MQTT is a messaging protocol used to communicate between IoT devices through a network. For this project MQTT is used to communicate between the Matlab software running on a PC and the Node-Red server.

### 4.2.1 Node-Red Server

Node-Red is flow based programming environment built for IoT. It allows hardware devices, API's and other services to be wired together using modules described as nodes [19]. This is designed to run on a machine such as the Raspberry Pi described and is able to be programmed using a web browser. It does this by setting up a web server on the host device allowing the flow based programming environment to be accessed by any computer.

The Node-Red environment can run on any device whether it be a local computer, an embedded device or even on the cloud. This means that this project can be extended to a cloud service in the future.
Furthermore its event driven which means that it is both light weight and non-blocking in nature, ie. flows are executed as events are detected [19].

Figure 4.4 shows the layout of the environment displaying the node browser on the left hand side where nodes can be selected and added to a flow. The main tabs at the top show the flows where nodes are connected together to perform desired functions. The example in figure 4.4 shows an automation setup developed for web control of lights and devices.
Each node outputs a message object which is then read by the a connected node and the information in the message is acted on. For example using the inject node in figure 4.4

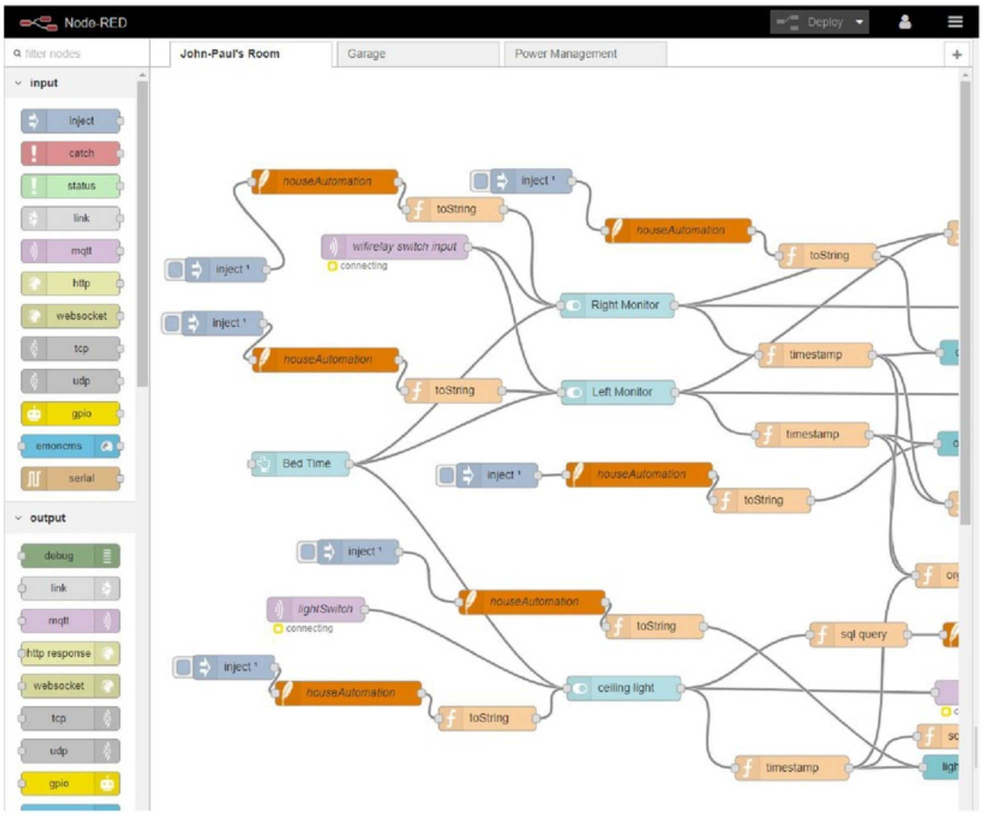


**Figure 4.4:** Node-Red Flow Based Programming Environment. Based in a web browser.

sends a command to the SQLite Database node labeled “houseAutomation” to retrieve information about the state of an element in the database.

Messages in Node-Red for the most part are structured with a topic and a payload. The topic can be a command or address or any arbitrary identifier to which the payload is concerned. The payload can be any data structure such as a string, date time or number which the node connected requires as an input.
Function nodes are used to perform custom functions that may not be available in the node palette. They are often used to input a message object and restructure the data to for use with other nodes or just perform any custom designed function desired and are written in JavaScript.

### 4.2.2 MQTT Communication Protocol

There are two nodes of particular interest for this project they are the MQTT input and output nodes. MQTT stands for Message Queue Telemetry Transport and is a lightweight protocol that runs on top of the TCP/IP protocol used in every computer network [20]. It has become somewhat of a standard in the IoT community due to the little bandwidth needed to send/receive messages and the small code footprint needed [9]. This makes it a perfect candidate for embedded devices and the small packets of data being distributed in IoT applications, this project being no exception.
The protocol uses a publish and subscribe communication method, where there is no real master or slave unless the application is designed so. Similarly to Node-Red topics and payloads are used to send and receive messages.

Any clients that are subscribed to particular topic will receive any payload published by any other client about that topic.
It is standard to use the “/” when structuring topics in this way they can be hierarchical such that topics can have sub-topics. For example a client can be subscribed to the topic “bedroom/light/” where it would receive any payload published to that topic, however it can also subscribe to “bedroom/#” this allows the client to listen to all topics and payloads under bedroom.

This kind of usability makes it ideal for connecting network based IoT devices such as the intended NILM system this project aims to simulate though a Matlab application to a server.
Further more this protocol can be password protected and SSL encrypted which is used by most web services today for information privacy, for this project however security is not of significant concern.

Thus to use the MQTT protocol in a system, devices must communicate through an MQTT broker. This is a service running on a server which is responsible for distributing published messages to those subscribed, it is responsible for handling all the connections

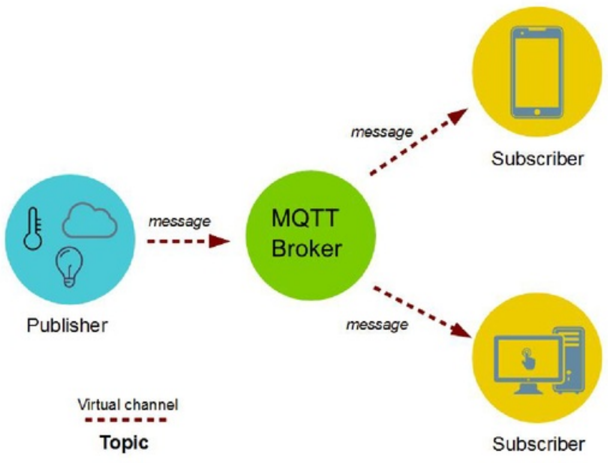


**Figure 4.5:** MQTT Protocol Structure. Messages are published to topics, those subscribed receive message. [9]

and ensure data gets to where it needs to go similar in function to a network switch as shown by figure 4.5 [9].

The broker used in this project is the Mosquitto MQTT broker which runs as a service on the Raspberry Pi server. It can run on any server and there are many freely available internet brokers that can used however for this project it is ideal to keep all subsystems within the same network.
The implementation of these technologies for the system design is shown in more detail in section 5.2.

# Chapter 5

# NILM System Design and Implementation

## 5.1 System Design

This chapter covers the software design and implementation of the proposed NILM system. The design goal for the system is the ability to determine given a sampled mains power signal when particular appliances of interest are switched on and off.
To achieve this the overall system can be divided into five key sub-systems, these being;

1. Event Detection
   For a given power signal imported into the system the events of interest should be isolated so as to formulate the load signature based on the change in real and reactive power at these times. Thus events represent turn on, turn off or change of state of appliances connected to the circuit. This system uses a sliding average change detection algorithm designed for this purpose.

2. Training
   Training data is an essential part of appliance classification. The system requires that load signatures of individual appliances of interest are stored in a database structure to ensure the power features are representative of the load, undisturbed by other electrical loads.

3. Load Signature Formulation
   Once a switching event has been determined the load signature is formulated, this represents the important electrical information about the event which can be used to identify what appliance it is. These features being the real, reactive and apparent power as well as sign of the phase angle (lagging or leading) at the event.

4. Load Signature Classification
   The software takes the detected load signatures and compares it against the database of trained appliances. Using a form of the nearest neighbour classification algorithm the best match is chosen.

5. Power and Appliance Monitoring
   Once the event has been detected and the appliance corresponding to that event identified the information about the appliance and activation time can be sent to the server where users can perform functions with the given information including calculating power consumption or used as rules for automation processes.

### 5.1.1 Event Detection

Using the hardware described in section 4.1 real power, reactive power, apparent power, and RMS voltage signals are acquired and stored in a CSV file at a rate of one sample per second. This rate is adequate for near real time identification as the algorithms used are able to be computed before the next sample is acquired. The underlying assumption here is that the system would be implemented on standalone hardware or acquired in real time by a capable device.
As described in section 2.3.1 the event detection is based on changes in the active power signal and thus was found that using equation 2.3 the normalized power can be determined which helps account for fluctuations in voltage.

Change detection is an essential part of the over system design and thus to determine steady state changes in a signal the system must accurately determine the starting point of an event $t_i$ and finishing point of an event $t_f$. Simplistically an event could be described as the point when the magnitude of power at time $t_{i+1}$ is greater than that at time $t_i$. This however does not lead to a good detection as it assumed steady state is reached at precisely the next sample which is not the case.

Transient responses as described in section 3.1.2 show that appliances generally have a settling time of $t_s > 1s$, this shows that $t_f$ should be the time at which the power after a change at $t_i$ reaches its maximum or minimum value corresponding to an off or on event. Thus, if given initial change time $t_i$ as determined by $P_{t_i} - P_{t_{i+1}} > TH$ where $TH$ is some threshold, the final event time is determined by implementing a counter $k$ which waits for $P_{t_i+k} <= P_{t_i}$ or the inverse for falling power changes. This outputs $t_f = t_{i+k}$ when the aforementioned condition is met.

This is implemented in the Matlab application with an imported data-set producing the output shown in figure 5.1. Shown here is the position of $t_i$ and $t_f$ on the real, reactive and apparent power signals. Any changes below $TH$ are ignored to minimize superfluous events which are not of interest. Thus based on the power consumption of loads of interest $TH$ can be tuned to include such events.

### 5.1.2 Load Signature Formulation

The load signature $LS$ is a vector comprised of electrical features at an event which are to resemble those of individual appliances during its various states of operation. Thus this

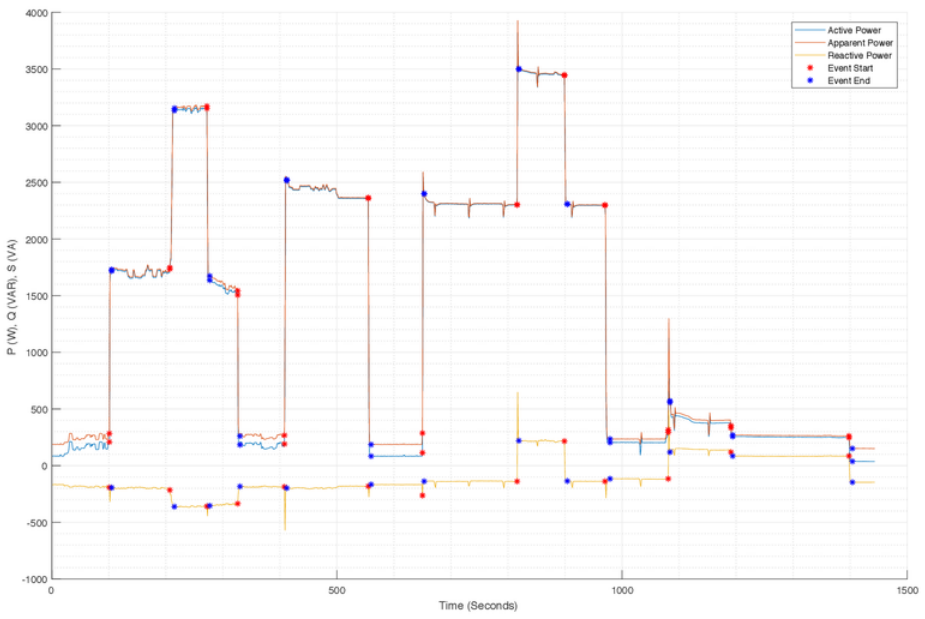


**Figure 5.1:** Output of the event detection algorithm. Red and blue markers indicate $t_i$ and $t_f$ respectively.

vector describes the change in real, reactive and apparent power at event $[t_i, t_f]$. The $t_f$ determined by the event detector is usually the peak value which is not representative of the $LS$ which should contain elements representing the steady state change in electrical energy. To account for this transient and noise in the steady state signal a averaging window is implemented [21, 22].

The averaging window concept is described by equation 5.1 [21] where for any event in our power signal the average of samples over window 1, $w_1$, before $t_i$ is negated from the average during window 2, $w_2$, after $t_f + D_s$ where $D_s$ is a transient delay time [21, 22]. Window 1 and 2 represent how many samples are to be averaged and the output of equation 5.1 being the change in power $\Delta P[t_i, t_f]$ [21, 22].

$$\Delta P[t_i, t_f] = \frac{1}{w_2} \sum_{i=t_f+D_s+1}^{t_f+D_s+w_2} P(i) - \frac{1}{w_1} \sum_{i=t_i-w_1}^{t_i-1} P(i) \tag{5.1}$$

After $\Delta P[t_i, t_f]$ is determined it is compared with $TH$ to evaluate whether it is still a suitable event as during testing it was noticed noise which triggers the event detector but

who's duration is less than $w_2$ should not be included as it assumed not relevant. In this way events such as noise can be rejected so computational time is not taken trying to classify its signature.
Equation 5.1 is also applied to the reactive and apparent power signals at $[t_i, t_f]$ to determine LS features $\Delta Q$ and $\Delta S$.

When calculating $\Delta Q$ based on the algorithm previously described it is important to account for the sign of the phase angle whether the appliance contributes to a lagging or leading reactive power. This is shown in figure 5.2 where it can be seen that at particular events the reactive power crosses zero, shifting between negative and positive reactive power consumption.
The sign of the reactive power describes whether the power factor is lagging or leading at a particular instant. Measures are taken to ensure correct measurements are made shown in the case 5.2.

$$\Delta Q = \begin{cases} -Q_{avg}(t_i), & \text{if } Q_{avg}(t_i) > 0 \text{ and } Q_{avg}(t_f) < 0 \\ Q_{avg}(t_f), & \text{if } Q_{avg}(t_i) < 0 \text{ and } Q_{avg}(t_f) > 0 \\ Q_{avg}(t_f) - Q_{avg}(t_i), & \text{otherwise} \end{cases} \tag{5.2}$$

The above cases handle $\Delta Q$ which can change from negative to positive during an event $[t_i, t_f]$ due to the change of phase angle sign when the appliance is switched.
Furthermore the sign of the phase angle $\phi$ or whether the appliance is lagging or leading is added as a feature to the load signature to further distinguish between appliances. The state of $\phi$ can be represented as a binary bit as shown in equation 5.3. This represents the cases where the appliance has switched and the change in sign of the reactive power is opposite to that of the change is sign of the power.

$$\phi = \begin{cases} 1, & \text{if } \Delta P > 0 \text{ and } \Delta Q < 0 \\ 1, & \text{if } \Delta P < 0 \text{ and } \Delta Q > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

From the state information derived using eq 5.1, 5.2, 5.3 a $LS$ can be formulated in the form of $LS = [t_s, \Delta P, \Delta Q, \Delta S, \phi]$. Information of whether the appliance state is ON or OFF is detailed in the sign of $\Delta P$.

The implementation of this is shown in figure 5.2 where the event distribution is shown in the $P, Q, S$ power space. In the figure marker shapes represent the appliance corresponding to that event, the color in this case represents $\phi$ (phase angle sign). Events situated in the positive power section are turn on events while those in the negative are turn off events.

The separation between events has tremendous effects on the performance of the system which is detailed in chapter 6. However this section has describes thus far the detection of

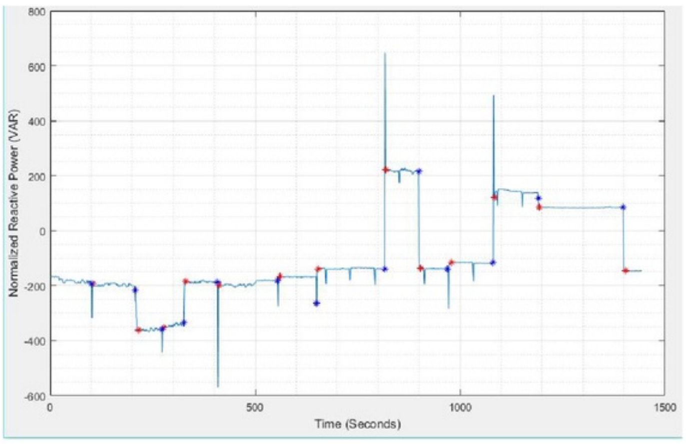


**Figure 5.2:** Reactive power showing the crossing from negative to positive during a switching event.

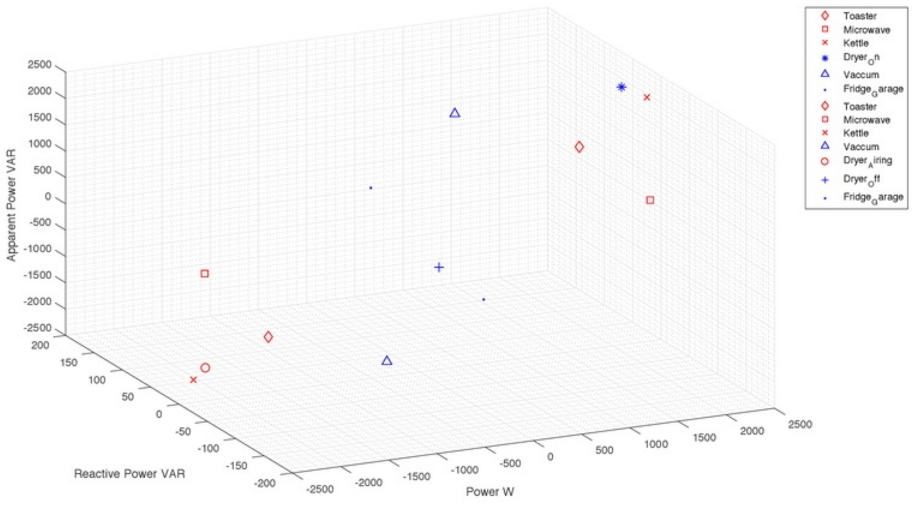


**Figure 5.3:** Turn on and off events for different appliances in the $P, Q, S$ power space.

events and the formulation of a signature vector to describe the electrical characteristics at the event, the next stage of the system design is classification of the signature.

### 5.1.3 Load Signature Classification

Appliance identification in this system is achieved through a classification algorithm known as k-nearest neighbour or KNN. After detecting events in the aggregated electrical signal and producing a load signature vector $LS_{[t_i,t_f]}$ the aim is to determine which class of appliance this signature belongs to.

This presents itself as a typical classification problem similar to determining whether an email is spam or not. It is a prediction which today is becoming increasing more accurate with the aid of machine learning and AI.
There are many algorithms that deal with classification such as the Naive-Bayes classifier [23], and nearest-neighbour [24]. The latter was chosen for three distinct reasons, those being it makes no assumptions about the data, it is relatively robust and it's simple to implement and modify.

Nearest neighbour appliance classification works by treating the load signatures as vectors in a vector space. For any observation $LS_{[t_i,t_f]}$, the euclidean distance between it and every vector in training signature data-set $X$ is calculated using equation 5.4 producing set $D$ corresponding to each element in set $X$.

$$D(LS, X_i) = \sqrt{(\Delta P_{LS} - \Delta P_X)^2 + (\Delta Q_{LS} - \Delta Q_X)^2 + (\Delta S_{LS} - \Delta S_X)^2} \tag{5.4}$$

The distances in $D$ are sorted in ascending order $D_s$, the appliance labels from the class set $L$ associated with the $K$ nearest neighbors from the sorted distance set $D_s$ are selected where $K \geq 1$. From this minimized set the label which occurs most frequently is selected as the most likely appliance thus a prediction is made [24].
The drawbacks of such an algorithm include the inability to exclude matches as for every event including those which are not in our training set $X$ the nearest neighbour is still selected. Another drawback is the computational burden of the algorithm if $X$ gets very large the algorithm becomes increasing slower as it still must go through every element of $X$ [24].

Thus to reduce the computational burden, improve classification accuracy and allow irrelevant events to be excluded some modifications are implemented to the original algorithm. Reducing set $X$ further accomplishes this. By using the signature feature $\phi$ as a condition such that only appliances that are in the same phase angle direction are analyzed. This reduces erroneous predictions where devices have similar power magnitudes but may differ in phase direction.
Removing irrelevant events is accomplished by introducing a threshold on distances $D < TH$ those outside this are labeled as "unclassified". The classification results of this

implementation are best shown in figure 5.3 where the marker shape for each event is associated with the unique appliance classified by this algorithm.

Accuracy can be increased over time by adding more data to set $X$ for appliance labels in $L$ based on known events that present as "unclassified" or are classified incorrectly. KNN is a form of supervised machine learning and over time clusters would be formed for each device around their mean power features and the value of $K$ increased.

### 5.1.4 Classifier Training

Training the KNN classifier is accomplished by adding to training set $X$ and labels $L$ the signature of individual appliances during each of their operational states. This means that for events during an individual non-aggregated appliance power signal a load signature should be formed and added to $X$ with a corresponding label added to $L$ describing the operational mode.

For example looking to figure 5.4 the power signal of a dishwasher during its normal

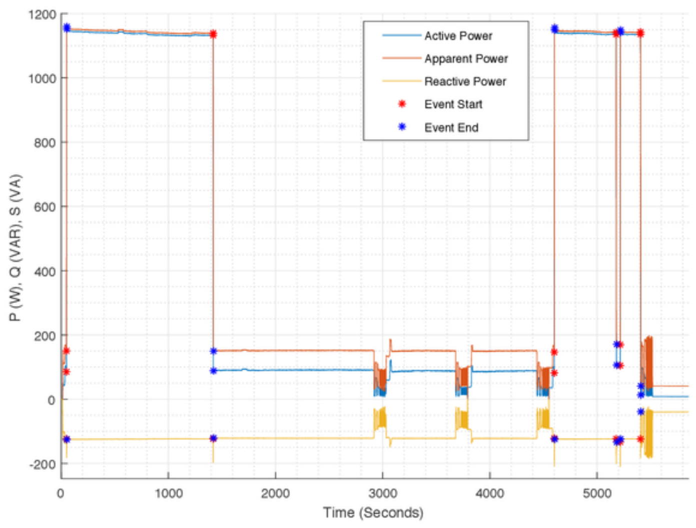


**Figure 5.4:** Power signal of a dishwasher showing operational modes.

operation can be seen. There are multiple states throughout the duration of its operation such as heating water and pumping, however there is only one distinct turn on and turn off event all other modes of operation differ slightly in their power but significantly in reactive power. Nonetheless all events should be identified but have distinct labels for each mode in order to distinguish at the server when the dishwasher was initially turned on and finally turned off.

Thus each event is labeled as $appliance - i$ where $i$ is a number corresponding to each mode. Such labeling can be interpreted at the server to perform whatever function necessary.

Most appliances however are type I in that they turn on and off with distinct power values at these instances. These are much easier to handle as in this case only one signature is needed to represent the operation of the appliance, however in most cases a minimum of two states are added to ensure small variations during ON and OFF states are accounted for. In such cases though only a single label is needed being simply the name of the appliance.
Table 5.1 shows a example of class labels set $L$ while table 5.2 displays and example of training data set $X$. Each row in $X$ corresponds with the same row in $L$ and each column in $X$ corresponds with each feature in the $LS$.

**Table 5.1:** Class Labels, set $L$ for classification.

| Row | Appliance Label |
|---|---|
| 1 | Microwave-1 |
| 2 | Microwave-2 |
| 3 | Dryer-1 |
| 4 | Dryer-2 |
| 5 | Dryer-3 |
| 6 | Vaccum |
| 7 | Vaccum |

**Table 5.2:** Training set $X$ corresponding to labels $L$.

| Row | $\Delta P$ (W) | $\Delta Q$ (VAR) | $\Delta S$ (VA) | $\phi$ |
|---|---|---|---|---|
| 1 | 1510.18 | -258.20 | 1470.90 | 1 |
| 2 | -1399.38 | 238.25 | -1360.05 | 1 |
| 3 | 2271.84 | 55.60 | 2272.52 | 0 |
| 4 | -2117.58 | 20.40 | -2094.08 | 1 |
| 5 | -119.94 | -37.78 | -74.6 | 0 |
| 6 | 1156.00 | 241.00 | 1121.00 | 0 |
| 7 | -1156.00 | -241.00 | -1121.00 | 0 |

The sign of the power has been included to indicate that for loads such as the microwave there is a significant difference in power when turning off similarly during dryer mode 2, airing, the device is an off event as $\Delta P < 0$ however the appliance is not fully off which is indicated by dryer mode 3. It is also important to note that for devices such as the dryer the modes are specific to the event, mode 3 is only an off event it cannot occur as an on event. Thus to minimize error in classification further individual appliance states can be linked to the situation in which they occur effectively stopping a dyer mode 3 from having an ON state classification this however can be done by rules set on the server for that appliance.

## 5.2 System Implementation

The NILM system is implemented based on the design described in the previous section and using the hardware and software tools described in chapter 4. The flow chart in figure 5.5 shows the place of each subsystem in the larger system.

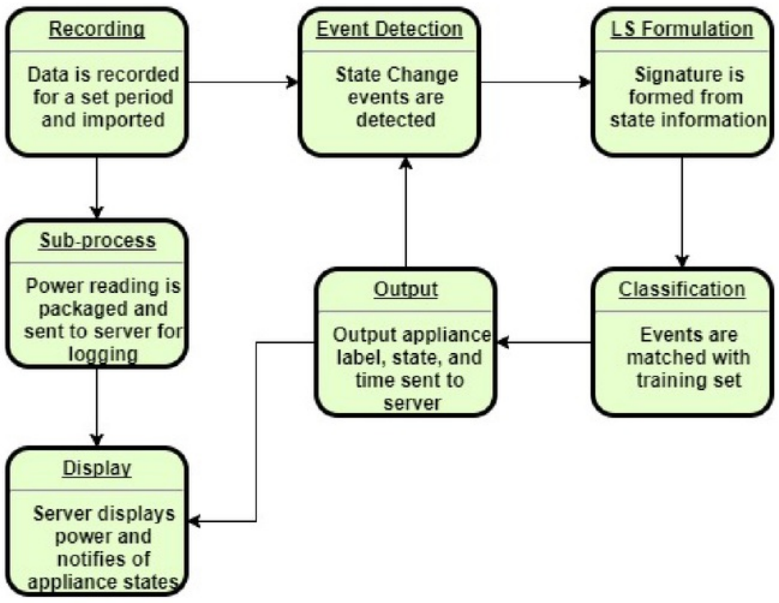
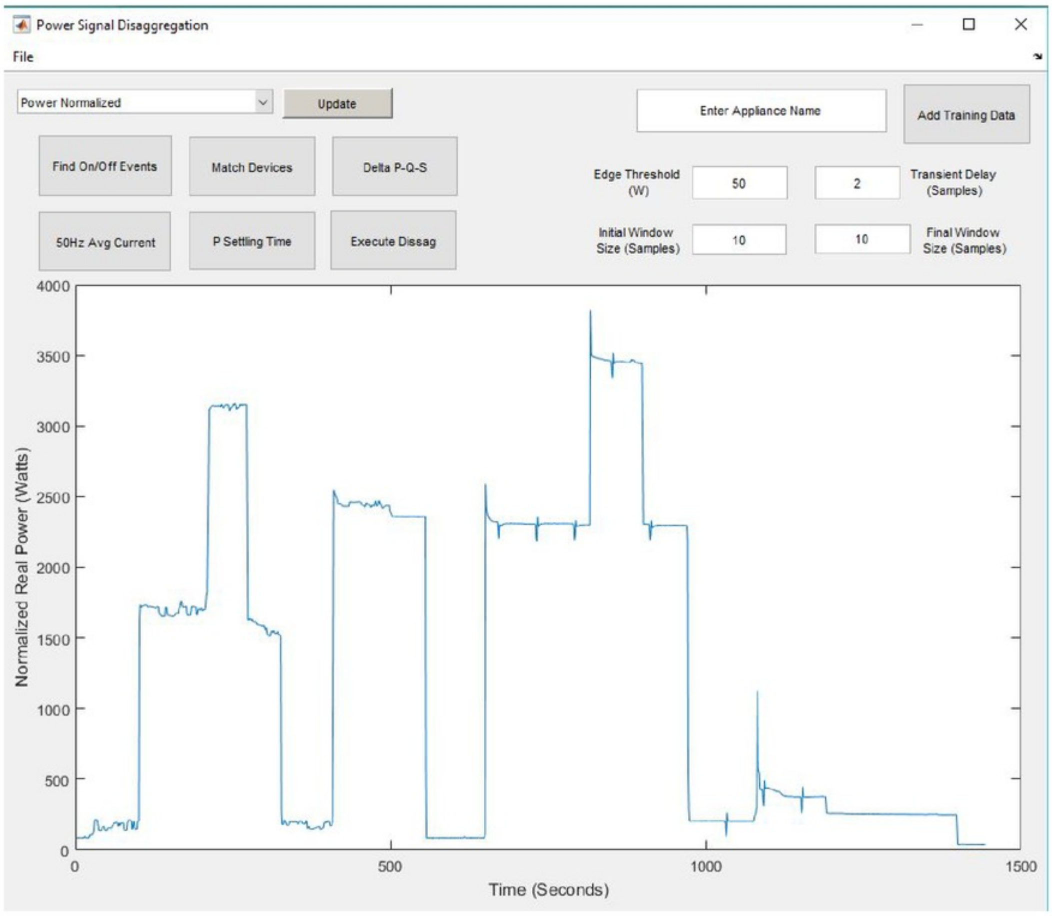

Recording
Data is recorded for a set period and imported
Event Detection
State Change events are detected
LS Formulation
Signature is formed from state information
Sub-process
Power reading is packaged and sent to server for logging
Output
Output appliance label, state, and time sent to server
Classification
Events are matched with training set
Display
Server displays power and notifies of appliance states

**Figure 5.5:** Flow Diagram of System Implementation

Recording of power signals is carried out by the HIOKI meter at a rate of one sample per second into a CSV file which includes the date and time of each sample, the average real, reactive and apparent power, along with voltage and current measurements. This file is downloaded to a pc through USB and is then imported into the Matlab analysis application developed. This is then sent to the node-red server described in chapter 4.2.1 via the MQTT protocol from chapter 4.2.2 for display and notification.

### 5.2.1 Matlab Analysis Application

The Matlab GUI shown in figure 5.6 is constructed using the GUIDE tool which allows for the interface to be custom designed. The interface is set up for this system so that a user can import the generated CSV from the power monitor by selecting 'File → Open CSV', once imported the user has the option of selecting a variety of signals from the drop down box and a variety of buttons each relating to the theory in the previously described design.
This is done so each function can be individually operated and its output individually analyzed. The rest of this chapter will detail how each of the designed subsystems are implemented in this GUI. Once a file is opened equation 2.3 is applied to the real, reactive and apparent power to normalize them as suggested by [3].



Edge Threshold (W): 50
Transient Delay (Samples): 2
Initial Window Size (Samples): 10
Final Window Size (Samples): 10

**Figure 5.6:** Power Signal Dissaggregation GUI.

### Finding Events

Events can be determined using the button 'Find On/Off Events' shown in figure 5.6. This uses the event detection algorithm described in section 5.1.1 and produces the display shown in figure 5.2. The output of this function are two arrays the first being $[initOn]$ and the second being $[steadyOn]$ these contain the indexes of all events in the signal corresponding to $[t_i, t_f]$ from earlier.

The threshold $TH$ for the change detection is set by the value entered in the text box 'Edge Threshold (W)' seen in figure 5.6, this has a default value of 50 Watts, the effects of tuning this threshold will be seen later.

### Forming the Load Signature

Load signatures are formulated as a result of finding the events in the signal as the same main loop is used. This uses the moving averaging window and algorithms first described in section 5.1.2 where two loops calculate the average power magnitudes before $initOn$ for the length of $w_1$ which is entered in the text box 'Initial Window Size' as well as the average after $steadyOn + D_s$ over the length of $w_2$ where $D_s$ is the delay entered in 'Transient Delay' and $w_2$ is entered in 'Final Window Size' shown in figure 5.7.

Conditions are also implemented that ensure the window size is dynamically adjusted if

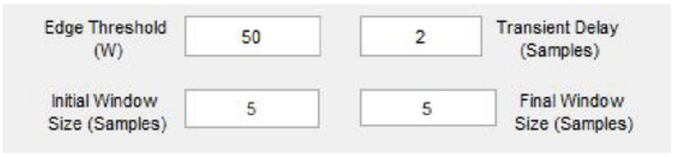

Edge Threshold (W): 50
Transient Delay (Samples): 2
Initial Window Size (Samples): 5
Final Window Size (Samples): 5

**Figure 5.7:** Variable Entry Dialogue Box Allowing Edge Detection Variable to Be Set.

an event is detected at $initOn < w_1$ or $steadyOn > w_2 + D_s$ ensuring the index remains in the range of the signal if events are detected at the beginning or end.

When calculating $\Delta Q$ equation 5.2 is implemented to ensure of the correct phase sign adjusted measurement. The edge threshold value entered is checked against $\Delta P$ once more to ensure that the event was is still valid after averaging. Once the phase sign $\phi$ is found using equation 5.3 the array 'output' is constructed containing variables $[dP, dQ, dS, phase]$ for all elements in $[initOn, steadyOn]$.
By clicking on the button 'Delta P-Q-S' the display in figure 5.3 is created showing all events as points of the events in P-Q-S energy space as well as displaying the event times and $LS$ for each event.

### Creating Training Data Set

As described in section 5.1.4 classifier training is accomplished by adding the load signature for each appliance mode to set $X$ and the corresponding label of the appliance to set

$L$. This is implemented in the GUI by writing the observation array $[dP, dQ, dS, phase]$ to a text file 'training.txt' in a CSV format along with writing the appliance name to file 'group.txt' for each event at $[initOn, steadyOn]$.

This training is achieved by monitoring the appliance for the duration of its operation, importing the file and entering the appliance name into the text box shown in figure 5.8. Once entered clicking 'Add Training Data' will perform the function described above and populate the group and training files.

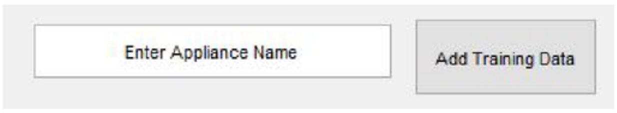


**Figure 5.8:** GUI Interface for adding appliance data to signature definition files.

### K-Nearest Neighbour Classification

The KNN algorithm is used for device classification with modifications to try to reduce the training set in order to increase the possibility for a correct match and decrease the likelihood of similarly powered appliance being mismatched. This algorithm described in section 5.1.3 is implemented in the GUI by using a KNN example originally written in python script from [24] which has been ported to Matlab and tailored for this system. The algorithm is implemented using three functions 'eDist', 'getKNN' and 'match'. These functions are shown in the code below [24].

```
%Calculate the euclidean distance between vectors set1 and set2 up to
%length
function dist = eDist(set1, set2, length)
distance = 0;
for i=1:length
    if i≠3
    distance = distance + (abs(set1(i))-abs(set2(i)))^2;
    end
end
dist = sqrt(distance);

%Find the K neighbors by finding distances if lag or lead and ...
    sorting, selecting the K nearest
function neighbors = getKNN(training,group,observation,k,error)
distances = [];
unspecified = {'unspecified'};
lngth = length(observation)-1;
for i = 1:length(group)
    if observation(4) == training(i,4)
        dist = eDist(observation,training(i,:),lngth);
        if dist≤error
```

```
            distances = [distances; group(i), dist];
        else
            distances = [distances; unspecified, dist];
        end
    end
end
sorted = sortrows(distances,2);
neighbors = [];
for i = 1:k
    neighbors = [neighbors; sorted(i,1)];
end

% Return the match that has the highest vote or occurance.
function match = getMatch(neighbors)
[a,b,c] = unique(neighbors);
count = hist(c,length(a));
if length(count)>1
    [M,I] = max(count);
    match = a(I);
else
    match = a;
end
```

The function ‘match’ takes the output of ‘getKNN’ and returns the most popular appliance label as the most likely appliance to match the load signature of the event [24]. The input of ‘getKNN’ is the training data set, set of appliance labels group, the load signature array observation, the value of $K$ and the error threshold for excluding distant matches.

Function ‘getKNN’ is responsible for getting the distance between each element in training and the observation signature by calling ‘eDist’. The distance is only calculated where observation element $\phi$ is equal to training element $\phi$ thus increasing the likelihood of a correct match. These distances are then sorted forming array ‘neighbors’ containing the sorted list of distances and their associated labels from the ‘group’ array, this is done only for the $K$ closest neighbors.
Selecting the ‘Match Devices’ option outputs the important information for analysis including the appliance, time, state and power figures as shown in table 5.3. The event information can also be displayed my clicking ‘Delta P-Q-S’ which displays figure 5.3 showing the appliances and their event distribution.

### Autonomous Operation

Autonomous operation takes the functions of the system and merges them together to simulate real time identification. This is done by formulating each load signature as an event is detected and transmitting the classification result at that instant to the server via MQTT. It also transmits the power consumption every second so it can be logged in the server for viewing by a user. This simulation is achieved by implementing a simple

**Table 5.3:** Output of Match Devices Function.

| Appliance | State | Time-stamp | $\Delta P$ (W) | $\Delta Q$ (VAR) | $\Delta S$ (VA) | $\phi$ |
|---|---|---|---|---|---|---|
| Toaster | on | 2017-10-31 12:15:52 | 1560 | -15 | 1490 | 1 |
| Microwave | on | 2017-10-31 12:17:42 | 1444 | -160 | 1453 | 1 |
| Toaster | off | 2017-10-31 12:18:44 | -1528 | 7.7 | -1511 | 1 |
| Microwave | off | 2017-10-31 12:19:37 | -1339 | 152 | -1299 | 1 |

delay of one second between each sample analyzed by the program.

In order to communicate with the server an MQTT connection must first be initialized with the MQTT broker running on the Raspberry Pi as described in chapter 4.2.2. This is done using;

```
%Create and instance of MQTT connection to broker
myMQTT = mqtt('tcp://192.168.1.21');
```

This describes the connection type 'TCP' and points the Matlab MQTT object to the ip address of the MQTT broker creating a connection.
Since two different types of information are being sent to the server a way of differentiating between the two is needed. This is done by creating unique topics as mentioned earlier, these topics need to represent the appliance classification data and the one second power data. The topics and payloads are formulated using the standard convention of;

- Appliance = appliance/'applianceLabel'/'dateTime'/'state'/$[\Delta P, \Delta Q, \Delta S, \phi]$
- Power = power/'dateTime'/$Power(i)$

This is implemented using these two lines of code which perform an MQTT publish command to the broker.

```
publish(myMQTT, ...
    'appliance'+'/'+string(KNN)+'/'+string(handles.Date(finish))+'/', ...
    strjoin(string(observation),','));
publish(myMQTT, ...
    'power'+'/'+string(handles.Date(i))+'/',string(Power(i)));
```

Finally this mode is used by clicking the button 'Execute Dissag' from the main display.

# Chapter 6

# Testing and Discussion

## 6.1 Hardware and Setup

In order to test the designed system some hardware needs to be constructed and the HIOKI power monitor needs to be connected. The HIOKI power monitor requires the measurement of the AC current and voltage waveform to calculate and record the desired power measurements. This means that the active wire of a circuit must be isolated in order to install the non-intrusive current probe. This is already available in a household meter box where mains active wiring is available for each circuit within the household, however it presents a problem when individual appliances are to be monitored. Since

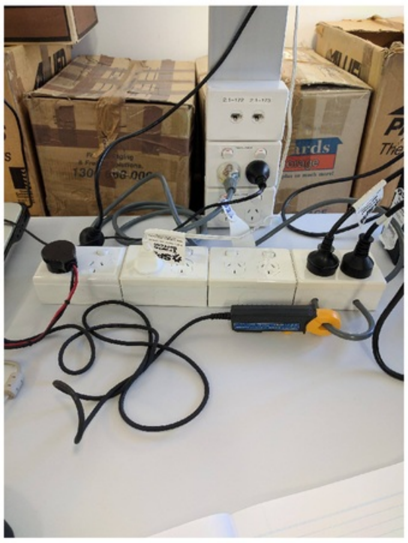


**Figure 6.1:** Setup of experimental test bench.

this system requires the monitoring of individual appliances to formulate the training data sets an inline adapter needs to be made to allow for current measurement. For this a custom power board was built which has an insulated but exposed mains active wire for current measurements and multiple sockets allowing for a controlled testing scenario where appliances can be individually switched on and off.

This designed hardware is shown in figure 6.1 where the board can be seen with the active wire loop. The current clamp is placed around this with care being taken to ensure it is installed in the correct direction, as it functions based on induced magnetic field caused by the AC current and as such if reversed the sensor will read the current as negative.
The voltage measurement is taken by connecting the standard outlet plug on the left of figure 6.1 which connects to the neutral and active connections on the meter. Since this setup is single phase only the one phase of meters three are used.

The equipment is also tested and tagged by a licensed electrician to ensure it can be used safely. This electrician also installed the current probe on the necessary mains circuits in the meter box when testing with household signals. The basic setup of this arrangement is shown by the diagram in figure 6.2.

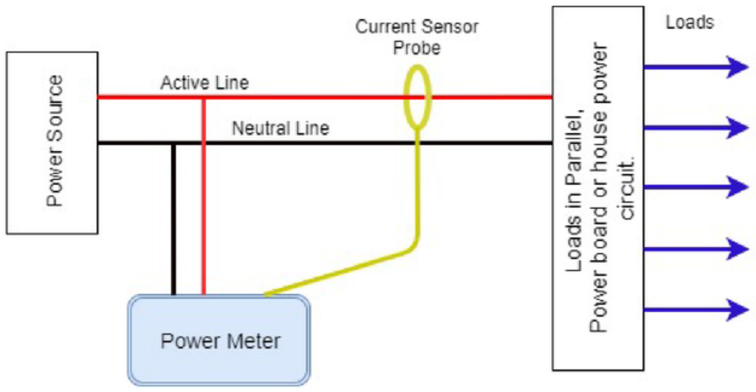


**Figure 6.2:** Diagram of current and voltage connection for measurement of multiple loads.

## 6.2 System Testing

This section is dedicated to testing the system in a number of different configurations and during each the results discussed. Since appliances can differ in load type, that is they can be type I, II, or type III then each type of load should be included in the test combinations. Since the goal of the project is to determine whether individual appliances can be identified from a single source then the main performance measure is the identification rate. This means that for each test scenario and each combination of load type the performance is gauged by the percentage of successful identifications to the actual number of activation's for an appliance over a time period, or put more simply in equation 6.1 and 6.2.

$$\frac{\text{Identified Appliance Duration}}{\text{Actual Appliance Duration}} = \text{Match } \% \tag{6.1}$$

$$\frac{\#\text{ Of Successful Identifications}}{\text{Total }\#\text{ Of Events}} = \text{Success Rate } \% \tag{6.2}$$

### 6.2.1 Testing Methodology

A set of test procedures must be setup to in order to collect data that is meaningful for analysis. There are a few definitions which should be established in relation to data collection, they include;

- The period for which data will be logged.
- The types of loads that will be monitored.
- The combinations of loads that will be monitored.
- The sequence of events of load switching.
- How long each load should stay on in aggregate configurations.
- How many times the load should be switched and for how long in standalone load configurations.

There are two main sets of data that are collected for testing, these are standalone and aggregate. Standalone data is collected once and this refers to the data collected for a single load during its operation. This is done for all appliances concerned is achieved by connecting the appliance to the apparatus from figure 6.1. A recording period is then set on the power monitor for a period longer than that of the normal operation of the appliance. For some appliances this period is only a few minutes such as in the case of the kettle however can stretch to two hours for appliances such as the dishwasher which has a long wash cycle in normal operation.

This standalone data for each appliance is imported into the GUI and by typing the appliance name and clicking 'Add Training Data' the state change information for each

event during its operation is added to the training file as previously discussed.
At this stage the 'Edge Threshold', and window sizes can be tuned to produce an accurate representation of the appliance. It was found that using an edge threshold of $50W$ gave good event detection for high powered loads. For lower powered loads this can be decreased. If erroneous events are generated which can be the case in variable load types then these signatures can be removed from the training file. In this case nine signatures are captured and stored into the training file, these are shown in table 6.1. If loads are switched on and off multiple times in this configuration then the average of each similar event is to be used.

**Table 6.1:** Appliance Signatures from individual monitoring.

| Appliance | $\Delta P$ (W) | $\Delta Q$ (VAR) | $\Delta S$ (VA) | $\phi$ |
|---|---|---|---|---|
| Kettle | 2338.42 | -20.20 | 2338.50 | 1 |
| Kettle | -2322.01 | 20.01 | -2322.10 | 1 |
| Dryer-1 | 2271.84 | 55.60 | 2272.51 | 0 |
| Dryer-2 | -2117.58 | 20.40 | -2094.07 | 1 |
| Dryer-3 | -119.94 | -37.78 | -74.6 | 0 |
| Aircon | 788.33 | 998.06 | 1271.79 | 0 |
| Aircon | -861.03 | -851.94 | -1210.09992 | 0 |
| Microwave-1 | 1510.18 | -258.20 | 1470.90 | 1 |
| Microwave-2 | -1399.37 | 238.25 | -1360.04 | 1 |
| Fridge-Garage | 200.12 | 187.55 | 274.27 | 0 |
| Fridge-Garage | -192.01 | -186.18 | -267.43 | 0 |
| Fridge-Garage | 270.70 | 190.88 | 331.19 | 0 |
| Fridge-Kitchen-1 | 74.78 | -63.69 | 95.83 | 1 |
| Fridge-Kitchen-2 | -49.44 | 51.20 | -70.01 | 1 |
| Fridge-Kitchen-3 | 77.85 | -57.01 | 88.57 | 1 |
| Fridge-Kitchen-4 | -62.31 | 48.59 | -76.15 | 1 |
| TV | 123.94 | -41.78 | 130.93 | 1 |
| TV | -128.25 | 43.05 | -135.33 | 1 |
| Vaccum | 1156 | 241 | 1121 | 0 |
| Vaccum | -1156 | -241 | -1121 | 0 |
| Toaster | 1583.02 | -19.69 | 1507.79 | 1 |
| Toaster | -1583.02 | 19.69 | -1507.79 | 1 |

The standalone signal for a fridge is shown in figure 6.3 where it can be seen there are two clear states on and off. It is difficult to capture the full duration of the event because of the extended periods of time and intermittent nature of the fridge compressor turning on. Compare this with figure 6.4 which shows the variable signal of a TV during a four minute period, although the signal varies it has clear on and off states during the small time windows of when it turns on and off. The difficulty here is ignoring false events

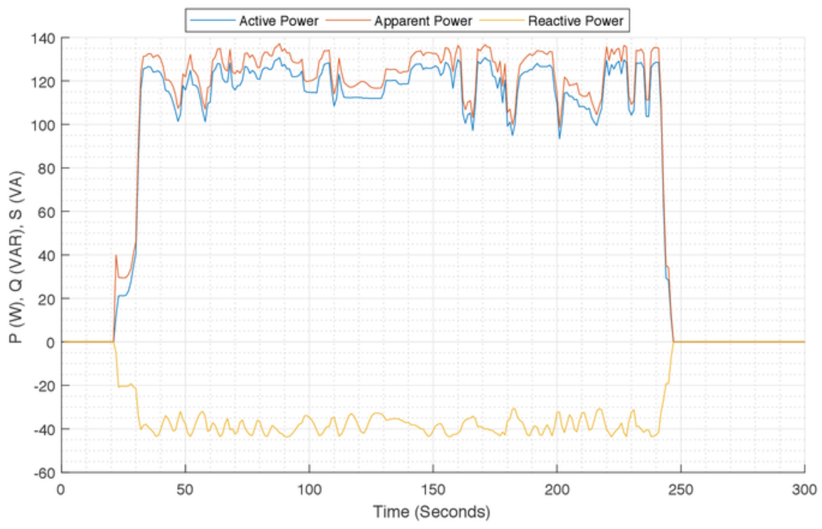


**Figure 6.3:** Power signal of a TV during 4 minutes of operation.

triggered during its operation.

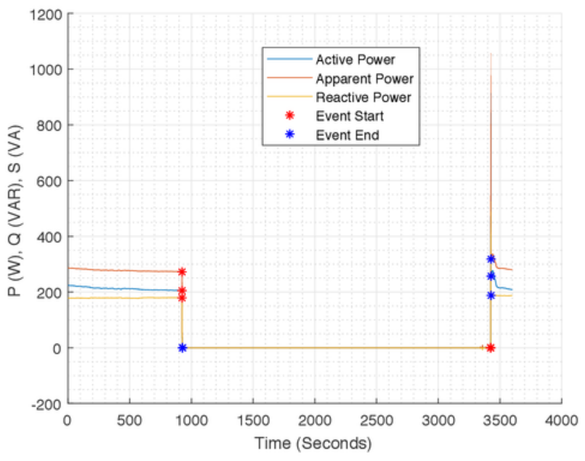


**Figure 6.4:** Power signal of a typical fridge during its cooling cycle.

Aggregate configuration refers to the data sets collected with multiple loads attached, creating a mixed signal with events occurring while others are in operation. This is typical of normal household activity, therefore this data is collected from a typical household and simulated on the test setup as a basic test. In this state different loads can be connected and switched in different configurations so as to determine anomaly's or artifacts that occur which can be expected for some loads and some combinations.

In aggregate data tests these loads where switched on and off in sequence forming unique combinations also with the use of household data there is always a noise floor or background signal which are loads that are always on and the test has no control over. This makes it possible to display real world performance of the system.
Furthermore in some household data there are events which are not known, that is loads switched on elsewhere in the house which are not part of the test and can be ignored.

### 6.2.2 Test Configuration 1

In this test scenario data is collected of a household power circuit for a short period of time in which a set of appliances are turned on and off. This appliance set is combined of a toaster, microwave, kettle, dryer, vacuum cleaner and fridge. Some of these appliance such as the toaster, and kettle are purely resistive while the others are combinations of resistive and inductive such as the dryer and vacuum.

The imported signal is the same as shown back in figure 5.1 with the events defined by the red and blue markers. This is done with the edge threshold set to $60W$ to remove small events which are not of interest.

The distribution of signatures produced at each event is shown in figure 6.5 where the

**Table 6.2:** Output of Match Devices Function.

| Appliance | Time On (s) | Time Off (s) | Identified On (s) | Identified Off (s) | % |
|---|---|---|---|---|---|
| Toaster | 101 | 274 | 101 | 272 | 98.8% |
| Microwave | 208 | 328 | 207 | 326 | 99.2% |
| Kettle | 406 | 557 | 408 | 555 | 97.4% |
| Dryer On-Airing | 650 | 974 | 650 | 970 | 98.8% |
| Dryer Airing-Off | 974 | 1192 | 970 | 1191 | 98.6% |
| Vacuum | 816 | 901 | 816 | 899 | 97.6% |
| Fridge | 1076 | 1404 | 1081 | 1398 | 96.6% |

separation between them gives and idea of the result we can expect. To characterize the performance the time between on and off or the run-time is used. This is the time in seconds in the signal the event is detected and the label that they system classified is shown. Time on and Time off represent the actual times they were activated by a human. Classifying the events yields the results shown in table 6.2.

In terms of identification for this test 100% accuracy is achieved as all appliance events were correctly identified. Table 6.2 shows that the identification times identified by the system are very close to the actual operation times in-fact only varying by at most 11 seconds. This successful identification is due to the fact that all these appliance are substantially different in signatures. Some are very close in terms of power consumption however differ in their reactive power consumption and may be opposing in their lag-lead configuration $\phi$.

Discrepancies in start and finish times can be accounted for by errors in event detection code with regards to the start time where, further more since the off state event detection uses the same algorithm as the on state detection the initial on time and off time are opposite to that of the prior. This means the time off is taken at the top of the falling edge and not the bottom of it.

### 6.2.3 Test Configuration 2

Unlike the first test this configuration consists of more appliances with some switched multiple times during the test period. For this reason the frequency of successful identifications is tested. This test includes the appliances listed in table 6.3 which is ordered by the sequence of events. The most far right column in this instance shows the identified times where '-' represents a miss-identification.

This test uses the household mains power circuit as its source signal. Since the house is three phase each phase services a separate circuit of the house. The power signal and event times are shown in figure 6.6, the signal shows clear power state changes at events.

As can be seen from table 6.3 the second vacuum on event is miss-identified as 'microwave-

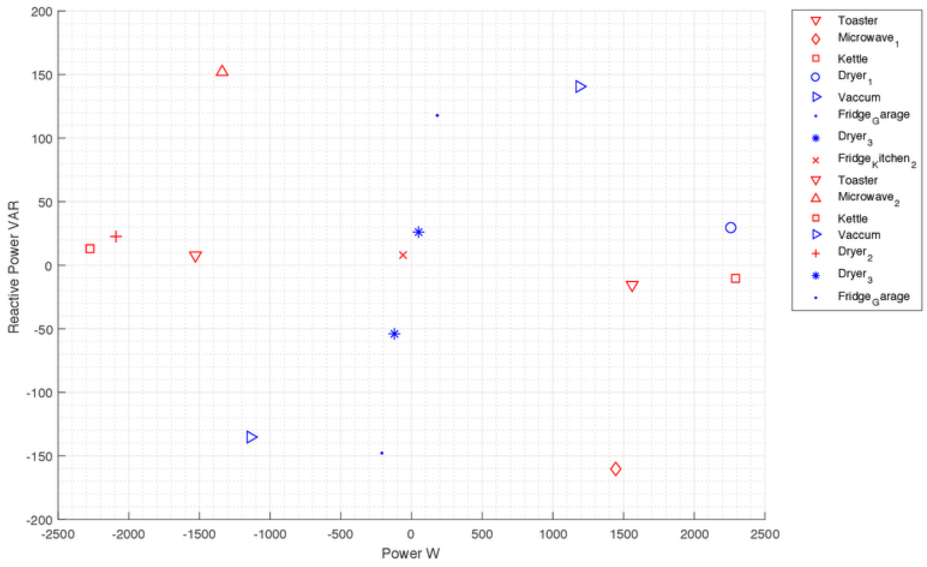


**Figure 6.5:** Test Configuration 1, Load Signature distribution.

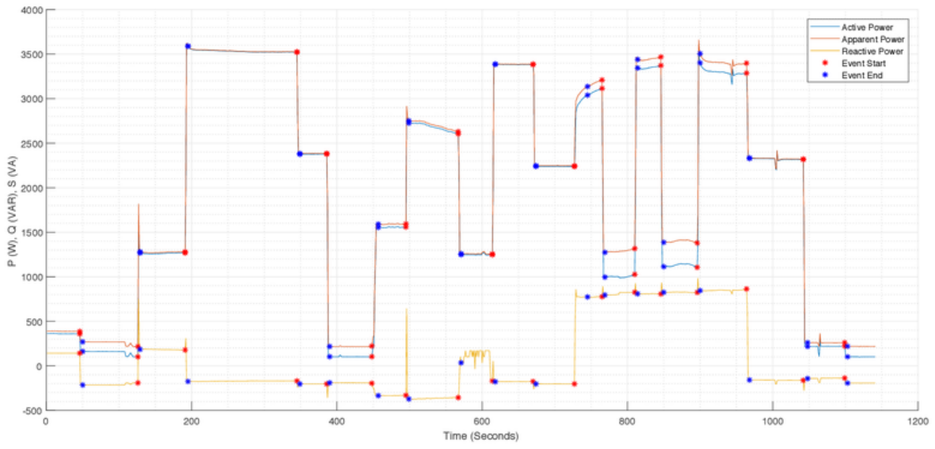


**Figure 6.6:** Test Configuration 2 power signals with event markers.

2' this is because as shown in figure 6.6 and in power signal fig 6.7 the event is displayed by a red marker indicating that it has $\phi = 1$ where the load signature of the vacuum has $\phi = 0$. This anomaly cannot be accounted for in code and is presumed to be caused by the electrical circuit configuration it is in with other appliances. The hair dryer off miss-identification can be linked to the fact that it's off power signature is almost identical to that of the dryer-airing signature, thus the closest match in the moment is selected. This pushes the idea that more signature features are needed to better separate appliances however some alternative measures can be put in place. For example knowing that 'dryer-2' can only occur after 'dryer-1' would have prevented this miss-identification and chosen the next closest candidate.

For this test the results show an identification rate of 88.24% where good performance is achieved for most events which are spaced far enough apart in the P-Q plane.

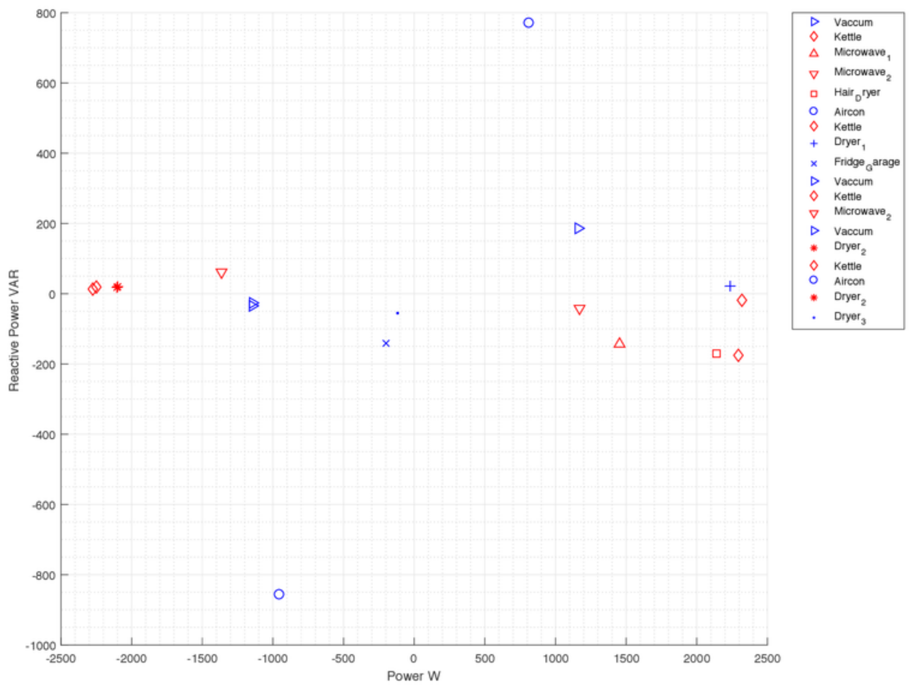


**Figure 6.7:** Test Configuration 2 appliance distribution.

Table 6.3: Event List of test configuration 2.

| Appliance | State | Event Time | Identified Time |
|---|---|---|---|
| Fridge | Off | 1:56:37 | 1:56:37 |
| Vacuum | On | 1:57:56 | 1:57:56 |
| Kettle | On | 1:59:02 | 1:59:02 |
| Vacuum | Off | 2:01:36 | 2:01:36 |
| Microwave | On | 2:03:24 | 2:03:24 |
| Vacuum | On | 2:04:06 | - |
| Microwave | Off | 2:05:18 | 2:05:18 |
| Hair Dryer | On | 2:06:05 | 2:06:05 |
| Vacuum | Off | 2:07:01 | 2:07:01 |
| Aircon | On | 2:08:12 | 2:08:12 |
| Hair Dryer | Off | 2:08:36 | - |
| Kettle | On | 2:09:21 | 2:09:21 |
| Kettle | Off | 2:09:57 | 2:09:57 |
| Dryer | On | 2:10:47 | 2:10:47 |
| Aircon | Off | 2:11:55 | 2:11:55 |
| Dryer - Airing | Off | 2:13:15 | 2:13:15 |
| Dryer | Off | 2:14:10 | 2:14:10 |

### 6.2.4 Test Configuration 3

Another household phase is tested for configuration three, this phase connects to one of the air-conditioner phases and the stove cook-top. The cook-top has multiple options for cooking food each containing a different sized heating element but each heat setting is thermostatically controlled. This mean the actual power magnitude does not change as the heat setting is increased but rather the duration of the element's on state. In such a case training measurement is needed for only one setting per element, in this test two element's are used along with the air-conditioner.

Training the load signature of each of the stove's element sizes in done by the same method used previously, it's power consumption is monitored while each are turned on and off. The system then stores the individual events as separate entries as shown in table 6.4. The air-conditioner is also monitored separately again and added to the load signature database in case of a difference in power on this phase. The table shows how multiple entries are listed for each appliance signature this done to create clusters of points which can increase the likeliness of a correct match by the classifier.

The test protocol involves turning on and off the hotplates, then turning the air-conditioner on and using the hotplates while the air-conditioner is on. The signal is is shown in figure 6.8 while the event signature distribution is shown in figure 6.9. For this test the iden-

tification rate is tested, that is number of correctly identified events divided by the total number of tested events. As can be deduced from the classified output shown in table 6.5, identification rate is 100% in this instance. Take into account this uses the same signature database populated with other signatures therefore this performance figure can be accounted for by the significant separation in the P-Q plane of these signatures.
Event threshold setting was kept the same for this test however averaging window sizes $w_1$ and $w_2$ were set to one sample whereas in previous tests this was kept at a standard five samples. With the larger window sizes some classifications where incorrect due to hotplate duration being less than the window length. Thus adjusting window sizes to the default five results in an identification rate of 77.27%, this is because as described the window size is larger than the event duration and as such the average changes in power are lower resulting in classification being confused between the larger hotplate and the smaller one. This output is shown in table 6.5.

No further tests of this phase where tested due to the limited number of appliances on it and so with that in mind if a three phase meter was used the optimum window size of this phase could be used in conjunction with the optimum window sizes for the other two phases. This way optimal results can be had for each of the phases and their appliances.

**Table 6.4:** Individual Signature's for stove and air-conditioner.

| Appliance | $\Delta P$ (W) | $\Delta Q$ (VAR) | $\Delta S$ (VA) | $\phi$ |
|---|---|---|---|---|
| Aircon | 1118.90 | -202.69 | 1136.87 | 1 |
| Aircon | -1212.47 | 350.83 | -1262.09 | 1 |
| Hotplates-1 | 935.14 | -1660.32 | 1905.70 | 1 |
| Hotplates-1 | 931.82 | -1651.71 | 1896.58 | 1 |
| Hotplates-1 | -929.82 | 1649.33 | -1893.27 | 1 |
| Hotplates-2 | 628.34 | -1128.50 | 1291.49 | 1 |
| Hotplates-2 | -623.11 | 1120.03 | -1281.99 | 1 |
| Hotplates-2 | 632.02 | -1132.132 | 1296.54 | 1 |
| Hotplates-2 | -626.30 | 1121.97 | -1285.03 | 1 |

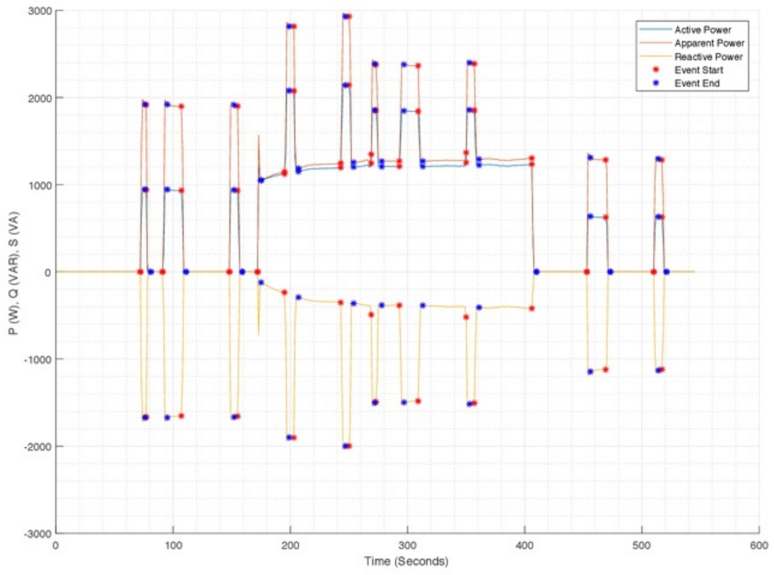


**Figure 6.8:** Test Configuration three, kitchen stove and air-conditioner phase.

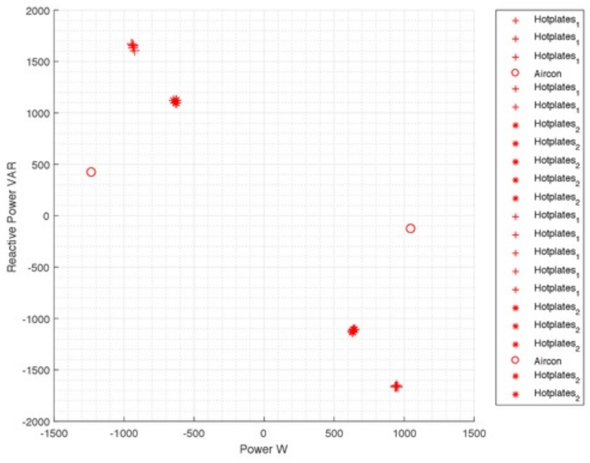


**Figure 6.9:** Test configuration three, event sequence with activation times and correctly identified times.

**Table 6.5:** Event List of test configuration 2. Averaging window size set to one and five.

| Appliance | State | Event Time | Appliance $w_{1,2} = 1$ | Appliance $w_{1,2} = 5$ |
|---|---|---|---|---|
| Hotplate-1 | On | 1:06:00 | Hotplate-1 | Fridge-Kitchen-1 |
| Hotplate-1 | Off | 1:06:05 | Hotplate-1 | Hotplate-2 |
| Hotplate-1 | On | 1:06:19 | Hotplate-1 | Hotplate-1 |
| Hotplate-1 | Off | 1:06:35 | Hotplate-1 | Hotplate-1 |
| Hotplate-1 | On | 1:07:16 | Hotplate-1 | Hotplate-2 |
| Hotplate-1 | Off | 1:07:23 | Hotplate-1 | Hotplate-1 |
| Aircon | On | 1:07:39 | Aircon | Aircon |
| Hotplate-1 | On | 1:08:03 | Hotplate-1 | Hotplate-1 |
| Hotplate-1 | Off | 1:08:11 | Hotplate-1 | Hotplate-1 |
| Hotplate-1 | On | 1:08:51 | Hotplate-1 | Hotplate-2 |
| Hotplate-1 | Off | 1:08:58 | Hotplate-1 | Hotplate-1 |
| Hotplate-2 | On | 1:09:16 | Hotplate-2 | Fridge-Kitchen-1 |
| Hotplate-2 | Off | 1:09:22 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | On | 1:09:41 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | Off | 1:09:57 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | On | 1:10:37 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | Off | 1:10:45 | Hotplate-2 | Hotplate-2 |
| Aircon | Off | 1:11:34 | Aircon | Aircon |
| Hotplate-2 | On | 1:12:20 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | Off | 1:12:37 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | On | 1:13:18 | Hotplate-2 | Hotplate-2 |
| Hotplate-2 | Off | 1:13:25 | Hotplate-2 | Hotplate-2 |

### 6.2.5 Test Configuration 4

Another test configuration of the household power circuit is examined this time however it contains a considerable amount of noise or variation influenced by the operation of computers and TV's, appliances which vary in power over time. The signal shown in figure 6.10 was captured over a five hour period and the appliances monitored include a kettle, air-conditioner and fridge.

It is evident from the event distribution space in figure 6.11 that besides the above appliances there are events which are incorrectly identified these being the dryer off mode and the TV. This is entirely due to the fact that these unknown events are closely matched to these appliances and demonstrates a weakness of the system in identifying appliances with smaller power magnitudes. Multi-state appliances such as the dryer could be ignored this case since a dryer-airing has not occurred before the dryer-off, thus it is very unlikely this is a good match.
However generally lower powered appliances, those which have power magnitudes less

than 200 Watts tend to occur frequently throughout the day and there is usually a large number of these appliances in the household. Thus this can confuse the system unless special rules are set per appliance or more features are included to characterize the appliance type.

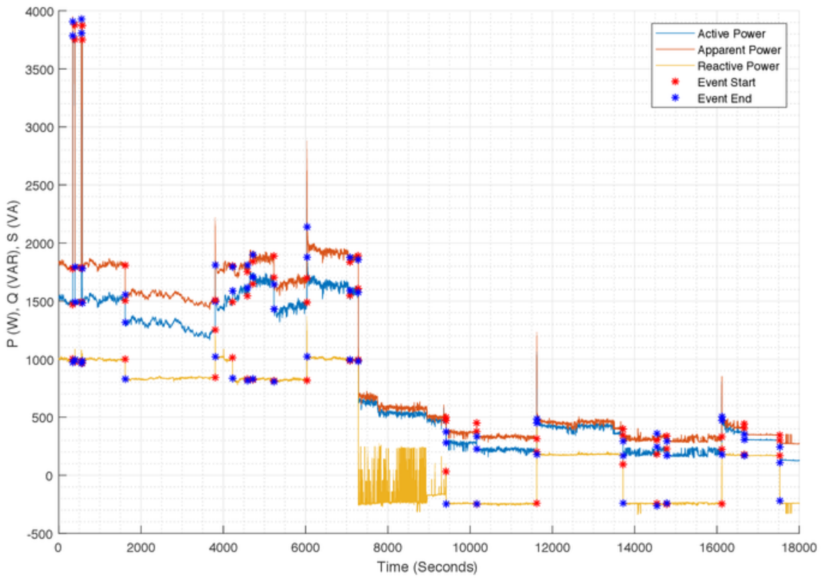


**Figure 6.10:** Test four power signals with considerable noise in the signal.

From the table 6.6 below it can be seen that most of the events are identified thus high accuracy for the higher powered appliances, however there are miss-identified moments where variation in the power signal caused by unknown appliances is identified as the TV and Dryer-Off as described above. The miss-identified fridge off event is due to the fact that the change in reactive power is not close to that of the fridge, again this a presumed issue caused by the circuit configuration with it and an appliance attached.

In terms of the three appliances monitored the overall identification success was 92.85% however the extraneous events detected particularly for the TV could have been true events however the order of states transitions (on-off, off-on) for this appliance suggest they are incorrect thus it can be presumed the actual identification rate is lower that shown.

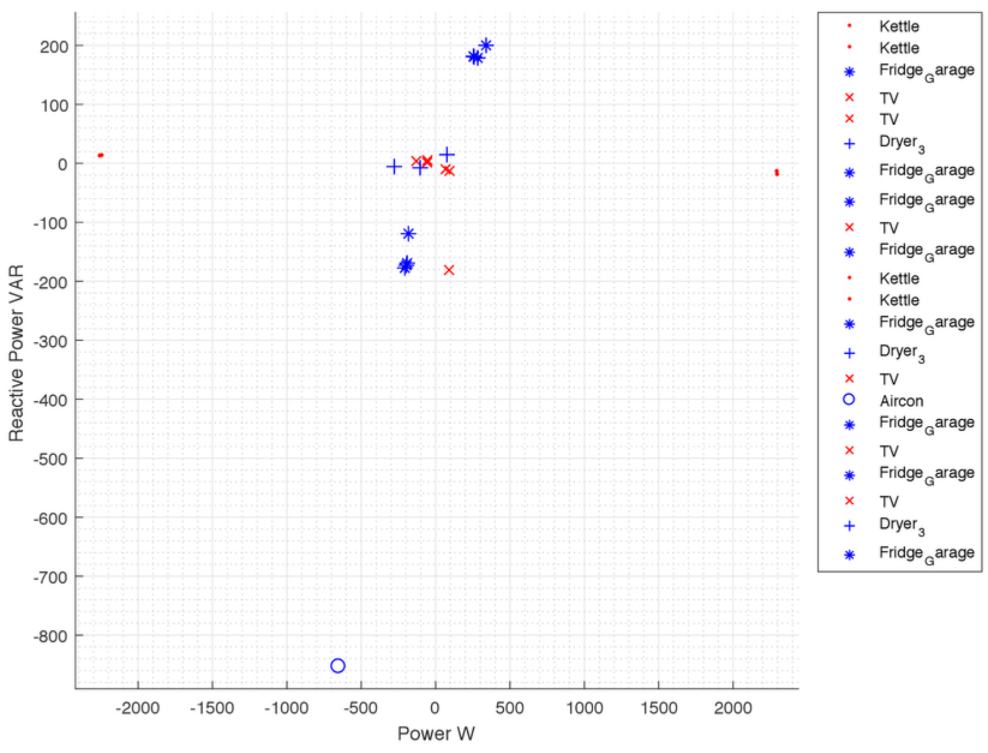


**Figure 6.11:** Test four event distribution in power space.

**Table 6.6:** Event List of test configuration 4.

| Appliance | State | Event Time | Identified Appliance |
|---|---|---|---|
| Kettle | On | 16:01:27 | Kettle |
| Kettle | Off | 16:02:23 | Kettle |
| Kettle | On | 16:05:00 | Kettle |
| Kettle | Off | 16:05:21 | Kettle |
| Fridge | Off | 16:22:45 | Fridge |
| Fridge | On | 16:59:13 | Fridge |
| - | On | 17:06:15 | TV |
| - | On | 17:12:16 | TV |
| - | On | 17:14:28 | Dryer-3 |
| Fridge | Off | 17:23:00 | Dryer-3 |
| Fridge | On | 17:36:24 | Fridge |
| - | Off | 17:53:47 | TV |
| Aircon | Off | 17:57:05 | Aircon |
| Fridge | Off | 18:32:44 | Fridge |
| - | Off | 18:45:09 | TV |
| Fridge | On | 19:09:29 | Fridge |
| Fridge | Off | 19:44:27 | Fridge |
| - | On | 19:58:08 | TV |
| - | Off | 20:02:08 | TV |
| Fridge | On | 20:24:29 | Fridge |
| - | Off | 20:33:35 | Dryer-3 |
| Fridge | Off | 20:47:56 | Fridge |

## 6.3 Discussion Review

Although the results of each test have been discussed during their associated sections a review of these is done here. As discussed, in each of the test scenarios the results of this NILM approach vary depending primarily on the load type monitored and the particular load combination they are in. Test one demonstrates that 100% identification is possible for appliances that have significant enough separation in the $\Delta P - \Delta Q$ power plane when not interfered by large amounts of frequently varying loads.
It shows also that the identified duration or response time can be almost identical to reality, this means when calculating power consumption over periods relatively accurate approximations can be made.

From test two it is shown that appliances with very close load signatures are difficult to distinguish as load signatures formulated from an aggregate source will never be identical to those of appliances on their own. This can be due to the influence of varying type loads and also because of physical properties of appliances such as motors or heating elements which can change depending on conditions. Nonetheless good identification is still achieved for these higher powered appliances.

The results of test three demonstrate limitations of the system where window size can drastically determine the identification accuracy. This means that appliances with very short duration's within the seconds range are difficult for the system to identify correctly. This was observed with the hotplates which only turn on for a short period of time at some heat settings, thus these were difficult to correctly identify due to the size of the averaging windows overlapping the rising and falling edges at $t_i$ and/or $t_f$. This can be improved by altering the window size however a dynamic implementation would prove to be more useful and robust. This could be implemented by performing a check for a significant change within the window lengths $w_1$ and $w_2$, the size of these windows could then dynamically be altered to a shorter length.

Since this system uses a sample rate of one sample per second transient effects at $t_i$ are often observed as described in chapter 3 and can be seen in various appliances throughout the test configurations. Due to the slow sample rate and often short lived transients of the appliances it is not useful to include their features in the load signature as during experimentation with noisy data such as that shown above, the transient features are never repeatable for the same load at low sample rates.
Taking the case of the fridge shown by the peaks in signal figure 6.10, these peak values can alter in magnitude quite significantly and in duration. Thus at higher sample rates it would be possible to capture the responses such as that of the fridge within the length of $w_1$ and $w_2$ since a high enough sampling rate would be used upwards of $2k$ samples per second, this way a filter such as the median filter [25] could be implemented within the transient window to reduce noise from the captured transient and deduce features such as peak power, and settling time. In such a case discrete time convolution could be used to determine the similarities between the observed transient pattern and those stored in the database.

Finally test scenario four shows that with the influence of varying and noisy non-linear appliances on the power signal the system can tend to report false identifications, events which are not of interest and incorrectly identified.
The system thus overall works quite well at determining two state and multi-state household appliances. The main conclusion being that for better identification results and recognition of variable loads more signature features are necessary which could be found by start-up transient features previously described.

# Chapter 7

# Conclusions and Future Work

The main objective of this thesis was to study electricity supply data and determine whether information about particular appliances attached to the circuit is able to the used in order to identify which appliances are in use. From background study in chapter 2 it was found this particular area of research is referred to as Non-intrusive load monitoring and from using data gathered from a mains metering device certain electrical characteristics can be utilized to identify when appliances are turned on and off in the source power signal.

To achieve this it was found that load signatures could be formed from state changes in power in the source power signal and that these formed signatures could be compared to those in a database of known appliance signatures to determine their on and off times. The concepts of load signature formulation from chapter 2 were implemented through the use of a Matlab GUI program which allowed analysis of electrical power signals, detection of changes in this signal, formulation of signatures and appliance classification through the use of kNN classification methods.

The design and implementation demonstrated in this thesis explores the possibility of creating a complete solution with the incorporation of internet of things protocol MQTT allowing for a scale-able internet based server solution using Node-Red. Such a solution can have wide use in both the home and industry reducing the reliance on costly smarter appliances and allowing for more intricate home automation processes by knowing the state of appliances and when they change state. From this power consumption can be estimated and rules set for automation processes. Furthermore this knowledge allows household residents to be more energy conscious and provide a basis for optimizing their usage times to minimize cost and consumption. Such use cases become increasingly relevant as home solar and battery systems become more prevalent and energy in the home becomes more complicated to balance.

## 7.1 Conclusions

From testing of the system in a variety of household usage cases shown in chapter 6 it was found that while the system is quite effective at identifying when large powered appliances are in use with up to 100% accuracy for small numbers of appliances, poor performance is observed for lesser power consuming appliances especially as the number of these appliances increases. Thus it was determined that the changes in power signals alone are not unique enough in some cases for successful identification and that additional features should also be implemented in the load signature formulation. This was found to be due to the fact that multiple loads use similar electrical elements for their operation, for example heating elements used in various appliances are found to produce almost identical magnitudes in both real and reactive power during some appliance states.

It was shown that while identification of dual and multi-state appliances is good poor performance is observed when constantly varying loads are introduced and recognition of these appliances using this method is not ideal. These appliances almost continually change in power draw and as such state changes in power are often small and clustered together and overlap making them hard to accurately discern.

The conclusion from this is that further optimization of event detection algorithms should be implemented with the use of high sample rate data and using more advanced filtering methods a change window can be captured where transient information about the appliance can be used to further distinguish between appliances.
Also through the use of more advanced machine learning algorithms which remodel appliance signatures over time would provide an interesting solution to the effects of inaccuracies in measurements caused by the combination of appliances on in the circuit.

## 7.2 Future Work

This project signifies a first step into the development of a complete NILM system and it is encouraged for future students or academics to use and continue to develop ideas and methods produced in this project to better improve identification accuracy particularly for low powered and noisy devices.
Below are some suggestions for future development in this field.

#### Higher Sampled Rate Data Acquisition

As mentioned previously the use of higher sampled rate data would allow for a signal that has smoother responses and the detection of transient signature features during changes. This would further increase the accuracy as load transient patterns are often unique. Similarly advanced digital filters can be used with higher sampled data which will lead to less miss-identifications.
Besides the ability to monitor transients, high sampled data would provide analysis of the

AC current waveforms harmonic components which can be determined using the Fourier transformation which as discussed in [10] helps to determine between non-linear loads. Implementing this at low cost could be done through the development of an IoT data acquisition unit capable of high sample rate analogue to digital conversion.

### Optimized Event Detection

By optimizing the event detection algorithm specifically the transient delay time $D_s$ and windows $w_1$, $w_2$ to be adjusted dynamically based on the response of the signal would improve the detection of short duration events and increase the identification accuracy.

### Cloud Based Classification

This would be a significant improvement as classification done on a server would be both accessible anywhere as well as having access to much more advanced machine learning capabilities including the use of big data to perform more accurate identification based on more rules and decisions as well as more advanced and useful analytic and data representation abilities.

# Chapter 8

# Abbreviations

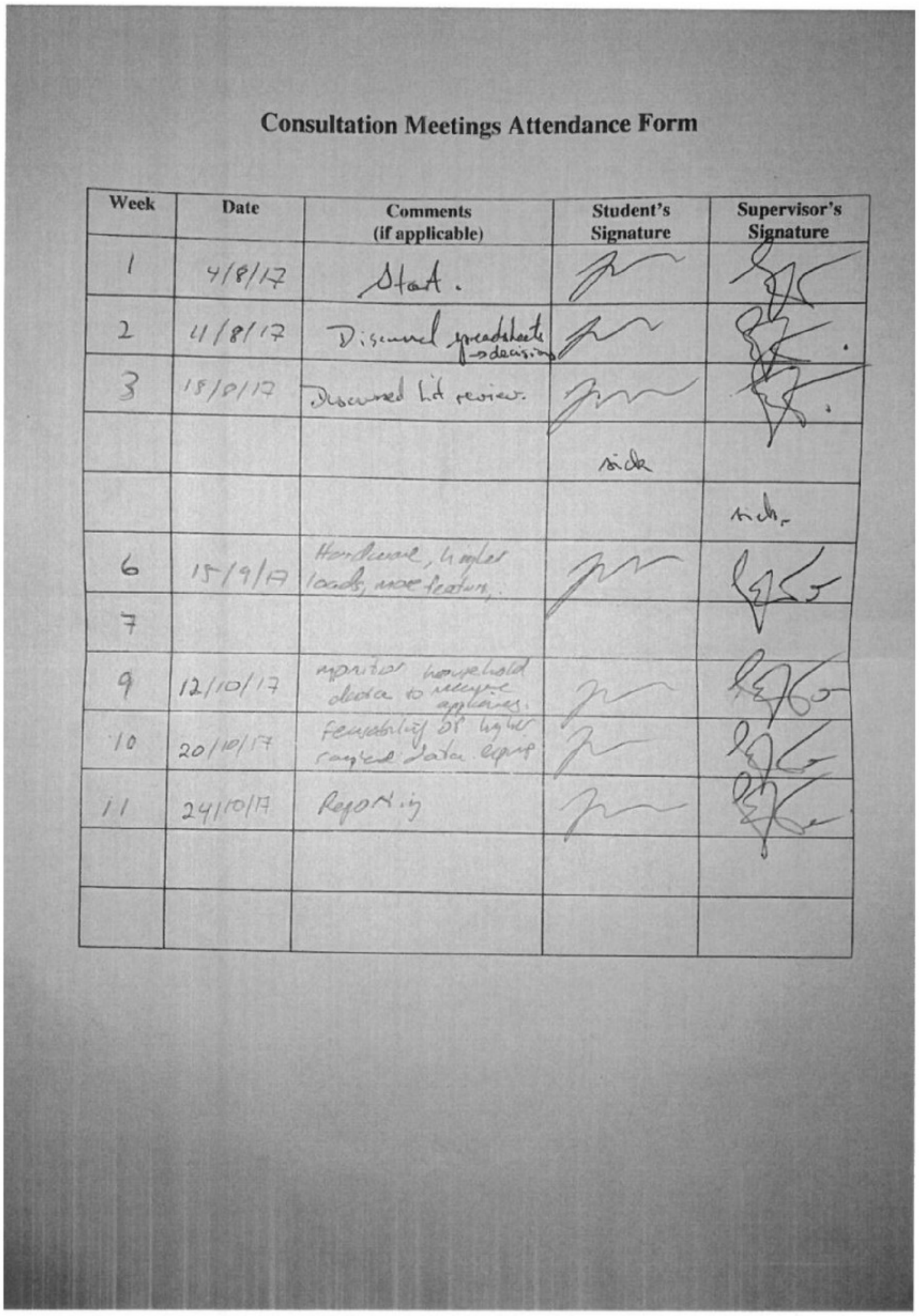
| | |
|---|---|
| NLIM | Non-Intrusive Load Monitoring |
| LS | Load Signature |
| SD | Standard Deviation |
| VSD | Variable Speed Drives |
| FFT | Fast Fourier Transform |
| PF | Power Factor |
| CSV | Comma-Delimited file format |
| AC | Alternating Current |
| MQTT | Message Queue Telemetry Transport |
| TCP | Transmission Control Protocol |
| GUI | Graphical User Interface |
| API | Application Programming Interface |
| IoT | Internet Of Things |
| IP | Internet Protocol |
| KNN | K Nearest Neighbour |
| RMS | Root Means Squared |

# Appendix A

# Supplementary Documentation

## A.1 Attendance Form

This section contains the project consultation attendance form.



**Consultation Meetings Attendance Form**

| Week | Date | Comments (if applicable) | Student's Signature | Supervisor's Signature |
|---|---|---|---|---|
| 1 | 4/8/17 | Start. | [illegible] | [illegible] |
| 2 | 11/8/17 | Discussed spreadsheets → decision | [illegible] | [illegible] |
| 3 | 18/8/17 | Discussed lit review. | [illegible] | [illegible] |
| | | | sick | |
| | | | | sick. |
| 6 | 15/9/17 | Hardware, higher loads, more features. | [illegible] | [illegible] |
| 7 | | | | |
| 9 | 12/10/17 | monitor household device to [illegible] appliances. | [illegible] | [illegible] |
| 10 | 20/10/17 | feasibility of higher sampled data. equip | [illegible] | [illegible] |
| 11 | 24/10/17 | Reporting | [illegible] | [illegible] |
| | | | | |
| | | | | |

**Figure A.1:** Meeting attendance sheet.

## A.2 Matlab Code

For reference and for anyone who may wish to read below is the Matlab code for main program GUI which includes the functions used. What is not included are the importation scripts which select the necessary data from the signal CSV and training text files.

```
function varargout = gui(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)
% Choose default command line output for gui
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        plot(handles.I1_AvgA);
        xlabel('Time (Seconds)');
        ylabel('Current (Amps)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.I1_AvgA(handles.steadyOn),'r*');
            plot(handles.initOn,handles.I1_AvgA(handles.initOn),'b*');
            hold off;
            grid on;
            grid minor;
        end
    case 2
        plot(handles.U1_AvgV);
        xlabel('Time (Seconds)');
        ylabel('Voltage (Volts)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.U1_AvgV(handles.steadyOn),'r*');
            plot(handles.initOn,handles.U1_AvgV(handles.initOn),'b*');
            hold off;
            grid on;
            grid minor;
        end
    case 3
        plot(handles.P1_AvgW);
        xlabel('Time (Seconds)');
        ylabel('Real Power (Watts)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.P1_AvgW(handles.steadyOn),'r*');
            plot(handles.initOn,handles.P1_AvgW(handles.initOn),'b*');
            hold off;
            grid on;
            grid minor;
        end
    case 4
        plot(handles.S1_AvgVA);
        xlabel('Time (Seconds)');
        ylabel('Apparent Power (VA)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.S1_AvgVA(handles.steadyOn),'r*');
            plot(handles.initOn,handles.S1_AvgVA(handles.initOn),'b*');
            hold off;
```

```
                grid on;
                grid minor;
            end
        case 5
            plot(handles.Q1_Avgvar);
            xlabel('Time (Seconds)');
            ylabel('Reactive Power (VAR)');
            if handles.onoff==1
                hold on;
                plot(handles.steadyOn,handles.Q1_Avgvar(handles.steadyOn),'r*');
                plot(handles.initOn,handles.Q1_Avgvar(handles.initOn),'b*');
                hold off;
                grid on;
                grid minor;
            end
        case 6
            plot(handles.PF1_Avg);
            xlabel('Time (Seconds)');
            ylabel('Power Factor');
            if handles.onoff==1
                hold on;
                plot(handles.steadyOn,handles.PF1_Avg(handles.steadyOn),'r*');
                plot(handles.initOn,handles.PF1_Avg(handles.initOn),'b*');
                hold off;
                grid on;
                grid minor;
            end
        case 7
            plot(handles.activeP);
            xlabel('Time (Seconds)');
            ylabel('Normalized Real Power (Watts)');
            if handles.onoff==1
                hold on;
                plot(handles.steadyOn,handles.activeP(handles.steadyOn),'r*');
                plot(handles.initOn,handles.activeP(handles.initOn),'b*');
                hold off;
                grid on;
                grid minor;
            end
        case 8
            plot(handles.reactiveP);
            xlabel('Time (Seconds)');
            ylabel('Normalized Reactive Power (VAR)');
            if handles.onoff==1
                hold on;
                plot(handles.steadyOn,handles.reactiveP(handles.steadyOn),'r*');
                plot(handles.initOn,handles.reactiveP(handles.initOn),'b*');
                hold off;
                grid on;
                grid minor;
            end
        case 9
        plot(handles.apparentP);
        xlabel('Time (Seconds)');
        ylabel('Normalized Apparent Power (VAR)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.apparentP(handles.steadyOn),'r*');
            plot(handles.initOn,handles.apparentP(handles.initOn),'b*');
            hold off;
            grid on;
            grid minor;
        end
    case 10
        plot(handles.Ifnd1_AvgA);
        xlabel('Time (Seconds)');
        ylabel('Average Fundamental Current (A)');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.Ifnd1_AvgA(handles.steadyOn),'r*');
            plot(handles.initOn,handles.Ifnd1_AvgA(handles.initOn),'b*');
            hold off;
            grid on;
            grid minor;
        end
    case 11
        clf('reset');
        hold on;
        plot(handles.activeP);
        plot(handles.apparentP);
        plot(handles.reactiveP);
        xlabel('Time (Seconds)');
        ylabel('P (W), Q (VAR), S (VA)');
        legend('Active Power','Apparent Power','Reactive Power');
        if handles.onoff==1
            hold on;
            plot(handles.steadyOn,handles.Ifnd1_AvgA(handles.steadyOn),'r*');
            plot(handles.initOn,handles.Ifnd1_AvgA(handles.initOn),'b*');
            grid on;
            grid minor;
        end
        hold off;
end



% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents ...
   as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from ...
    popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'Average Current', 'Voltage', 'Real Power', ...
    'Apparent Power', 'Reactive Power', 'Power Factor', 'Power ...
    Normalized', 'Reactive Power Normalized', 'Apparent Power ...
    Normalized', 'Average Fundamental Current', 'P,Q,S'});



% --------------------------------------------------------------------
function File_Callback(hObject, eventdata, handles)


% --------------------------------------------------------------------
function openCSV_Callback(hObject, eventdata, handles)
file = uigetfile('*.csv','Select the CSV data file');
if file ≠ 0
    handles.file = file;
    dataSet = importfile(file);
    handles.I1_AvgA = dataSet.I1_AvgA;
    handles.U1_AvgV = dataSet.U1_AvgV;
    handles.P1_AvgW = dataSet.P1_AvgW;
    handles.Etime = dataSet.Etime;
    handles.Date = dataSet.Date;
    handles.S1_AvgVA = dataSet.S1_AvgVA;
    handles.Q1_Avgvar = dataSet.Q1_Avgvar;
    handles.PF1_Avg = abs(dataSet.PF1_Avg);
    handles.Ifnd1_AvgA = dataSet.Ifnd1_AvgA;
    Snorm = [];
    activeP = [];
    apparentP = [];
    reactiveP = [];
    for i=1:length(dataSet.Etime)
        activeP = ...
            [activeP,((240/dataSet.U1_AvgV(i))^2)*dataSet.P1_AvgW(i)];
        reactiveP = ...
            [reactiveP,((240/dataSet.U1_AvgV(i))^2)*dataSet.Q1_Avgvar(i)];
        apparentP = ...
            [apparentP,((240/dataSet.U1_AvgV(i))^2)*dataSet.S1_AvgVA(i)];
        Snorm = ...
            [Snorm,((240/dataSet.U1_AvgV(i))^2)*(dataSet.U1_AvgV(i)*dataSet.I1_Av
    end
    handles.activeP = activeP;
    handles.reactiveP = reactiveP;
    handles.apparentP = apparentP;
    handles.Snorm = Snorm;
    handles.onoff = 0;
    guidata(hObject, handles);
end




function edit2_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
handles.currentTh = str2double(get(hObject,'String'));


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% Moving window event detector
%Get variables from gui
threshold = str2double(get(handles.threshold,'String'));
window1 = str2double(get(handles.window1,'String'));
window2 = str2double(get(handles.window2,'String'));
trans = str2double(get(handles.trans,'String'));
ΔP = [];
ΔS = [];
ΔQ = [];
initOn = [];
steadyOn = [];
phaseSign = [];
output = [];
finish = 0;
start = 0;
change = false;
i=2;
%Go through the signal find significant changes and determine the changes
%in power.
while i ≤ (length(handles.activeP)-1)
    %Positive State Change Events
    if handles.activeP(i+1)-handles.activeP(i) > threshold
        change = true;
        start = i;
        count = 0;
        rising = true;
        while rising == true
```

```matlab
                if handles.activeP(start+count+1) > ...
                    handles.activeP(start+count)
                    count = count + 1;
                else
                    count = count + 1;
                    rising = false;
                end
            end
            finish = start+count+1;
            Pi = 0;
            Pf = 0;
            Qi = 0;
            Qf = 0;
            Si = 0;
            Sf = 0;
            if start≤window1
                window1 = start-1;
            end
            for j=(start-window1):start-1
                Pi = Pi + handles.activeP(j);
                Qi = Qi + handles.reactiveP(j);
                Si = Si + handles.apparentP(j);
            end
            Pi = Pi/window1;
            Qi = Qi/window1;
            Si = Si/window1;
            if (length(handles.activeP)-finish)<(window2+trans)
               window2 = (length(handles.activeP)-finish)-trans;
            end
            for j=(finish+trans):(finish+trans+window2)-1
                Pf = Pf + handles.activeP(j);
                Qf = Qf + handles.reactiveP(j);
                Sf = Sf + handles.apparentP(j);
            end
            Pf = Pf/(window2);
            Qf = Qf/(window2);
            Sf = Sf/(window2);
            dP = Pf-Pi;
            dS = Sf-Si;
            if Qi > 0 && Qf <0
                dQ = Qf;
            elseif Qi < 0 && Qf > 0
                dQ = Qf;
            else
                dQ = Qf-Qi;
            end
        %Negative State Change Events
        elseif abs(handles.activeP(i+1)-handles.activeP(i)) > threshold
            change = true;
            start = i;
            count = 0;
            falling = true;
        while falling == true
            if handles.activeP(start+count+1) < ...
                handles.activeP(start+count)
                count = count + 1;
            else
                count = count + 1;
                falling = false;
            end
        end
        finish = start+count+1;
        Pi = 0;
        Pf = 0;
        Qi = 0;
        Qf = 0;
        Si = 0;
        Sf = 0;
        if start≤window1
            window1 = start-1;
        end
        for j=(start-window1):start-1
            Pi = Pi + handles.activeP(j);
            Qi = Qi + handles.reactiveP(j);
            Si = Si + handles.apparentP(j);
        end
        Pi = Pi/window1;
        Qi = Qi/window1;
        Si = Si/window1;
        if (length(handles.activeP)-finish)<(window2+trans)
            window2 = (length(handles.activeP)-finish)-trans;
        end
        for j=(finish+trans):(finish+trans+window2)-1
            Pf = Pf + handles.activeP(j);
            Qf = Qf + handles.reactiveP(j);
            Sf = Sf + handles.apparentP(j);
        end
        Pf = Pf/(window2);
        Qf = Qf/(window2);
        Sf = Sf/(window2);
        dP = Pf-Pi;
        dS = Sf-Si;
        if Qi > 0 && Qf <0
            dQ = 0-Qi;
        elseif Qi < 0 && Qf > 0
            dQ = Qf;
        else
            dQ = Qf-Qi;
        end
    end
    %Check to see if change in power is significant enough to be an event
    if (change == true)&&(abs(dP) ≥ threshold)
        %Check for lagging or leading phase angles, 1 = leading, 0 = ...
           lagging
```

```
            i = finish;
            if dP>0 && dQ<0 || dP<0 && dQ>0
                phase = 1;
            else
                phase = 0;
            end
            initOn = [initOn; start];
            steadyOn = [steadyOn; finish];
            ΔP = [ΔP; dP];
            ΔQ = [ΔQ; dQ];
            ΔS = [ΔS; dS];
            phaseSign = [phaseSign; phase];
            observation = [dP, dQ, dS, phase];
            output = [output; observation];
        else
            i = i+1;
        end
        change = false;
    end
    handles.initOn = initOn;
    handles.steadyOn = steadyOn;
    handles.output = output;
    format short g;
    disp([initOn, steadyOn, output]);
    figure
    hold on;
    plot(handles.activeP);
    plot(handles.apparentP);
    plot(handles.reactiveP);
    plot(initOn,handles.activeP(initOn),'r*');
    plot(steadyOn,handles.activeP(steadyOn),'b*');
    plot(initOn,handles.apparentP(initOn),'r*');
    plot(steadyOn,handles.apparentP(steadyOn),'b*');
    plot(initOn,handles.reactiveP(initOn),'r*');
    plot(steadyOn,handles.reactiveP(steadyOn),'b*');
    xlabel('Time (Seconds)');
    ylabel('P (W), Q (VAR), S (VA)');
    legend('Active Power','Apparent Power','Reactive Power','Event ...
        Start', 'Event End');
    grid on;
    grid minor;
    hold off;
    handles.onoff = 1;
    guidata(hObject, handles);


    % --- Executes on button press calculates the state changes the ...
        signal at
    % the change points
    function pushbutton4_Callback(hObject, eventdata, handles)
    appliances = unique(handles.out(:,1));
    markers = ['o','+','*','.','x','s','d','^','v','>','<','p','h'];
```

```
stylesOn = [];
stylesOff = [];
On = [];
Off = [];
for i=1:size(handles.out,1)
    if handles.out(i,2)=="on"
        On = [On;handles.out(i,:)];
    else
        Off = [Off;handles.out(i,:)];
    end
    for j=1:length(appliances)
        if handles.out(i,1)==appliances(j)
            if handles.out(i,2)=="on"
                stylesOn = [stylesOn;markers(j)];
            else
                stylesOff = [stylesOff;markers(j)];
            end
        end
    end
end
Pon = str2double(On(:,4));
Qon = str2double(On(:,5));
Son = str2double(On(:,6));
Poff = str2double(Off(:,4));
Qoff = str2double(Off(:,5));
Soff = str2double(Off(:,6));

%Plot the state changes in three dimentional space
figure
hold on;
for i=1:length(Pon)
if On(i,7) == "1"
    c = 'r';
else
    c = 'b';
end
f1 = scatter3(Pon(i),Qon(i),Son(i),60,c,stylesOn(i));
end
for i=1:length(Poff)
if Off(i,7) == "1"
    c = 'r';
else
    c = 'b';
end
f2 = scatter3(Poff(i),Qoff(i),Soff(i),60,c,stylesOff(i));
end
legend([On(:,1);Off(:,1)]);
set(legend,'Location','northeastoutside');
xlabel('Power W');
ylabel('Reactive Power VAR');
zlabel('Apparent Power VAR');
grid on;
grid minor;
hold off;


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
plot(medfilt1(handles.activeP,10));
% Ts = [];
% Ts = [Ts; (handles.steadyOn-handles.initOn)];
% handles.Ts = Ts;
% disp('Settling Time');
% disp(Ts);
guidata(hObject, handles);


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
∆Ifnd = [];
∆Irms = [];
for i=1:length(handles.steadyOn)
    ∆Ifnd = [∆Ifnd; ...
        (handles.Ifnd1_AvgA(handles.steadyOn(i))-handles.Ifnd1_AvgA(handles.initC
    ∆Irms = [∆Irms; ...
        (handles.I1_AvgA(handles.steadyOn(i))-handles.I1_AvgA(handles.initOn(i)))
end
handles.∆Ifnd = ∆Ifnd;
handles.∆Irms = ∆Irms;
disp('Fundamental Average Current');
disp(∆Ifnd);
disp('RMS Current');
disp(∆Irms);

scatter(∆Irms,∆Ifnd);
xlabel('RMS Current A');
ylabel('50Hz Component of Current A');
axis([-15 15 -15 15]);
grid on;
grid minor;
guidata(hObject, handles);

function groupBox_Callback(hObject, eventdata, handles)
% hObject    handle to groupBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of groupBox as text
%        str2double(get(hObject,'String')) returns contents of ...
   groupBox as a double

% --- Executes during object creation, after setting all properties.
function groupBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to groupBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns ...
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in matchDevices.
function matchDevices_Callback(hObject, eventdata, handles)
% This functon reads the training database and group database and ...
    performs
% K-nearest neighbour to determine which is the closest match at each ...
    of the
% on events. To make this better load vector will need to improved as ...
    well
% as training data array.
%Classifier Datasets
training = readTraining('training.txt');
group = readGroup('group.txt');
out = [];
for i=1:size(handles.output,1)
    if handles.output(i,1) > 0
        state = "on";
    else
        state = "off";
    end
    out = ...
        [out;getMatch(getKNN(training,group,handles.output(i,:),2,500)), ...
        state, handles.Date(handles.steadyOn(i)), handles.output(i,:)];
end
handles.out = out;
guidata(hObject, handles);
disp(out);





% --- Executes on button press in executeDisagg.
function executeDisagg_Callback(hObject, eventdata, handles)
%Executes the iterative dissaggregation algorithm, here the three ...
    datasets
%are incremented and compared to the last changes after each second or
%sample are classified and if a close match is found then it is accepted
%and posted.
PmedFil = medfilt1(handles.activeP,10);
QmedFil = medfilt1(handles.reactiveP,10);
SmedFil = medfilt1(handles.apparentP,10);
```

```
Power = handles.activeP;

%Classifier Variables
training = readTraining('training.txt');
group = readGroup('group.txt');

%Instanciate MQTT connection to broker
myMQTT = mqtt('tcp://192.168.1.21');

% Moving window event detector
%Get variables from gui
threshold = str2double(get(handles.threshold,'String'));
window1 = str2double(get(handles.window1,'String'));
window2 = str2double(get(handles.window2,'String'));
trans = str2double(get(handles.trans,'String'));
ΔP = [];
ΔS = [];
ΔQ = [];
initOn = [];
steadyOn = [];
phaseSign = [];
output = [];
finish = 0;
start = 0;
change = false;
i=2;
%Go through the signal find significant changes and determine the changes
%in power.
while i ≤ (length(Power)-1)
    publish(myMQTT, "power"+"/",string(Power(i)));
    pause(0.2);
    %Positive State Change Events
    if Power(i+1)-Power(i) > threshold
        change = true;
        start = i;
        count = 0;
        rising = true;
        while rising == true
            if Power(start+count+1) > Power(start+count)
                count = count + 1;
            else
                count = count + 1;
                rising = false;
            end
            publish(myMQTT, "power"+"/",string(Power(count)));
            pause(0.2);
        end
        finish = start+count+1;
        Pi = 0;
        Pf = 0;
        Qi = 0;
        Qf = 0;
        Si = 0;
        Sf = 0;
        if start≤window1
            window1 = start-1;
        end
        for j=(start-window1):start-1
            Pi = Pi + Power(j);
            Qi = Qi + handles.reactiveP(j);
            Si = Si + handles.apparentP(j);
        end
        Pi = Pi/window1;
        Qi = Qi/window1;
        Si = Si/window1;
        if (length(Power)-finish)<(window2+trans)
           window2 = (length(Power)-finish)-trans;
        end
        for j=(finish+trans):(finish+trans+window2)-1
            Pf = Pf + Power(j);
            Qf = Qf + handles.reactiveP(j);
            Sf = Sf + handles.apparentP(j);
            publish(myMQTT, "power"+"/",string(Power(j)));
            pause(0.2);
        end
        Pf = Pf/(window2);
        Qf = Qf/(window2);
        Sf = Sf/(window2);
        dP = Pf-Pi;
        dS = Sf-Si;
        if Qi > 0 && Qf <0
            dQ = Qf;
        elseif Qi < 0 && Qf > 0
            dQ = 0-Qi;
        else
            dQ = Qf-Qi;
        end
    %Negative State Change Events
    elseif abs(Power(i+1)-Power(i)) > threshold
        change = true;
        start = i;
        count = 0;
        falling = true;
        while falling == true
            if Power(start+count+1) < Power(start+count)
                count = count + 1;
            else
                count = count + 1;
                falling = false;
            end
            publish(myMQTT, "power"+"/",string(Power(count)));
            pause(0.2);
        end
        finish = start+count+1;
        Pi = 0;
        Pf = 0;
        Qi = 0;
        Qf = 0;
        Si = 0;
        Sf = 0;
        if start≤window1
            window1 = start-1;
        end
        for j=(start-window1):start-1
            Pi = Pi + Power(j);
            Qi = Qi + handles.reactiveP(j);
            Si = Si + handles.apparentP(j);
        end
        Pi = Pi/window1;
        Qi = Qi/window1;
        Si = Si/window1;
        if (length(Power)-finish)<(window2+trans)
            window2 = (length(Power)-finish)-trans;
        end
        for j=(finish+trans):(finish+trans+window2)-1
            Pf = Pf + Power(j);
            Qf = Qf + handles.reactiveP(j);
            Sf = Sf + handles.apparentP(j);
            publish(myMQTT, "power"+"/",string(Power(j)));
            pause(0.2);
        end
        Pf = Pf/(window2);
        Qf = Qf/(window2);
        Sf = Sf/(window2);
        dP = Pf-Pi;
        dS = Sf-Si;
        if Qi > 0 && Qf <0
            dQ = Qf;
        elseif Qi < 0 && Qf > 0
            dQ = 0-Qi;
        else
            dQ = Qf-Qi;
        end
    end
    %Check to see if change in power is significant enough to be an event
    if (change == true)&&(abs(dP) ≥ threshold)
        %Check for lagging or leading phase angles, 1 = leading, 0 = ...
            lagging
        i = finish+trans+window1;
        if dP>0 && dQ<0 || dP<0 && dQ>0
            phase = 1;
        else
            phase = 0;
        end
        initOn = [initOn; start];
        steadyOn = [steadyOn; finish];
        ΔP = [ΔP; dP];
        ΔQ = [ΔQ; dQ];
        ΔS = [ΔS; dS];
        phaseSign = [phaseSign; phase];
        observation = [dP, dQ, dS, phase];
        if dP<0
            state = 'off';
        else
            state = 'on';
        end
        KNN = getMatch(getKNN(training,group,observation,2,200));
        output = [output;KNN, state, handles.Date(finish), observation];
        publish(myMQTT, ...
            "appliance"+"/"+string(KNN)+"/"+string(handles.Date(finish))+"/", ...
            strjoin(string(observation),','));
    else
        i = i+1;
    end
    change = false;
end

handles.loadSignatures = training;
guidata(hObject, handles);
disp(output);

figure
hold on;
plot(Power);
plot(handles.apparentP);
plot(handles.reactiveP);
xlabel('Time (Seconds)');
ylabel('P (W), Q (VAR), S (VA)');
legend('Active Power','Apparent Power','Reactive Power');
grid on;
grid minor;
hold off;

figure
hold on;
scatter3(ΔP,ΔQ,ΔS,20,phaseSign);
xlabel('Power W');
ylabel('Reactive Power VAR');
zlabel('Apparent Power VA');
axis([-3000 3000 -3000 3000 -3000 3000]);
grid on;
grid minor;
hold off;

%Calculate the euclidean distance between vectors set1 and set2 up to
%length
function dist = eDist(set1, set2, length)
distance = 0;
```

```
for i=1:length
    if i≠3
    distance = distance + (abs(set1(i))-abs(set2(i)))^2;
    end
end
dist = sqrt(distance);

%Find the K neighbors by finding distances if lag or lead and ...
    sorting, selecting the K nearest
function neighbors = getKNN(training,group,observation,k,error)
distances = [];
unspecified = {'unspecified'};
lngth = length(observation)-1;
for i = 1:length(group)
    if observation(4) == training(i,4)
        dist = eDist(observation,training(i,:),lngth);
        if dist≤error
            distances = [distances; group(i), dist];
        else
            distances = [distances; unspecified, dist];
        end
    end
end
sorted = sortrows(distances,2);
neighbors = [];
for i = 1:k
    neighbors = [neighbors; sorted(i,1)];
end

% Return the match that has the highest vote or occurance.
function match = getMatch(neighbors)
[a,b,c] = unique(neighbors);
count = hist(c,length(a));
if length(count)>1
    [M,I] = max(count);
    match = a(I);
else
    match = a;
end

% --- Executes on button press in addTrainingData.
function addTrainingData_Callback(hObject, eventdata, handles)
%This function takes the text inputed and when Add training data is ...
    clicked
%the function will take a training dataset imported and find the average
%values for that appliance and add them to the training and group
%matricies.
Power = handles.activeP;
figure
hold on;
plot(Power);
xlabel('Time S');
```

```
ylabel('Power W');
grid on;
grid minor;
hold off;
%Set variables via gui
threshold = str2double(get(handles.threshold,'String'));
window1 = str2double(get(handles.window1,'String'));
window2 = str2double(get(handles.window2,'String'));
trans = str2double(get(handles.trans,'String'));
ΔP = [];
ΔS = [];
ΔQ = [];
initOn = [];
steadyOn = [];
phaseSign = [];
finish = 0;
start = 0;
change = false;
i=2;
%Go through the signal find significant changes and determine the changes
%in power.
while i ≤ (length(Power)-1)
    %Positive State Change Events
    if Power(i+1)-Power(i) > 5
        change = true;
        start = i;
        count = 0;
        rising = true;
        while rising == true
            if Power(start+count+1) > Power(start+count)
                count = count + 1;
            else
                count = count + 1;
                rising = false;
            end
        end
        finish = start + count+1;
        i = finish;
        Pi = 0;
        Pf = 0;
        Qi = 0;
        Qf = 0;
        Si = 0;
        Sf = 0;
        if start≤window1
            window1 = start-1;
        end
        for j=(start-window1):start-1
            Pi = Pi + Power(j);
            Qi = Qi + handles.reactiveP(j);
            Si = Si + handles.apparentP(j);
        end
        Pi = Pi/window1;
        Qi = Qi/window1;
        Si = Si/window1;
        if (length(Power)-finish)<(window2+trans)
           window2 = (length(Power)-finish)-trans;
        end
        for j=(finish+trans):(finish+trans+window2)-1
            Pf = Pf + Power(j);
            Qf = Qf + handles.reactiveP(j);
            Sf = Sf + handles.apparentP(j);
        end
        Pf = Pf/(window2);
        Qf = Qf/(window2);
        Sf = Sf/(window2);
        dP = Pf-Pi;
        dS = Sf-Si;
        if Qi > 0 && Qf <0
            dQ = Qf;
        elseif Qi < 0 && Qf > 0
            dQ = 0-Qi;
        else
            dQ = Qf-Qi;
        end
    %Negative State Change Events
    elseif abs(Power(i+1)-Power(i)) > 5
        change = true;
        start = i;
        count = 0;
        falling = true;
        while falling == true
            if Power(start+count+1) < Power(start+count)
                count = count + 1;
            else
                count = count + 1;
                falling = false;
            end
        end
        finish = start + count+1;
        i = finish;
        Pi = 0;
        Pf = 0;
        Qi = 0;
        Qf = 0;
        Si = 0;
        Sf = 0;
        if start≤window1
            window1 = start-1;
        end
        for j=(start-window1):start-1
            Pi = Pi + Power(j);
            Qi = Qi + handles.reactiveP(j);
            Si = Si + handles.apparentP(j);
            end
            Pi = Pi/window1;
            Qi = Qi/window1;
            Si = Si/window1;
            if (length(Power)-finish)<(window2+trans)
                window2 = (length(Power)-finish)-trans;
            end
            for j=(finish+trans):(finish+trans+window2)-1
                Pf = Pf + Power(j);
                Qf = Qf + handles.reactiveP(j);
                Sf = Sf + handles.apparentP(j);
            end
            Pf = Pf/(window2);
            Qf = Qf/(window2);
            Sf = Sf/(window2);
            dP = Pf-Pi;
            dS = Sf-Si;
            if Qi > 0 && Qf <0
                dQ = Qf;
            elseif Qi < 0 && Qf > 0
                dQ = 0-Qi;
            else
                dQ = Qf-Qi;
            end
        end
        %Check to see if change in power is significant enough to be an event
        if (change == true)&&(abs(dP) ≥ threshold)
            %Check for lagging or leading phase angles, 1 = leading, 0 = ...
                lagging
            if dP>0 && dQ<0 || dP<0 && dQ>0
                phase = 1;
            else
                phase = 0;
            end
            initOn = [initOn; start];
            steadyOn = [steadyOn; finish];
            ΔP = [ΔP; dP];
            ΔQ = [ΔQ; dQ];
            ΔS = [ΔS; dS];
            phaseSign = [phaseSign; phase];
            observation = [start,finish,dP, dQ, dS, phase];
        else
            i = i+1;
        end
        change = false;
    end

    deviceLabel = get(handles.groupBox,'String');
    if handles.file ≠ 0
        for i=1:length(ΔP)
          training = [ΔP(i) ΔQ(i) ΔS(i) phaseSign(i)];
          fileID = fopen('training.txt','at');
        fprintf(fileID,'%f,%f,%f,%f;\n',training);
        fclose(fileID);
        fileID = fopen('group.txt','at');
        fprintf(fileID,'%s;\n',deviceLabel);
        fclose(fileID);
    end
  end








function threshold_Callback(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of threshold as text
%        str2double(get(hObject,'String')) returns contents of ...
    threshold as a double


% --- Executes during object creation, after setting all properties.
function threshold_CreateFcn(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns ...
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function window2_Callback(hObject, eventdata, handles)
% hObject    handle to window2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of window2 as text
%        str2double(get(hObject,'String')) returns contents of ...
    window2 as a double


% --- Executes during object creation, after setting all properties.
```

```
function window2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to window2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns ...
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function window1_Callback(hObject, eventdata, handles)
% hObject    handle to window1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of window1 as text
%        str2double(get(hObject,'String')) returns contents of ...
    window1 as a double


% --- Executes during object creation, after setting all properties.
function window1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to window1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns ...
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function trans_Callback(hObject, eventdata, handles)
% hObject    handle to trans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of trans as text
%        str2double(get(hObject,'String')) returns contents of trans ...
    as a double


% --- Executes during object creation, after setting all properties.
```

```
function trans_CreateFcn(hObject, eventdata, handles)
% hObject    handle to trans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns ...
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
   get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

# References

[1] A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *Sensors*, vol. 12, no. 12, p. 1683816866, Dec 2012.

[2] L. K. Norford and S. B. Leeb, "Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms," *Energy and Buildings*, vol. 24, no. 1, pp. 51 – 64, 1996.

[3] E. Aladesanmi and K. Folly, "Overview of non-intrusive load monitoring and identification techniques," *IFAC-PapersOnLine*, vol. 48, no. 30, pp. 415 – 420, 2015, 9th IFAC Symposium on Control of Power and Energy Systems CPES 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896315030566

[4] J. Liang, S. Ng, G. Kendall, and J. Cheng, "Load signature study v part i: Basic concept, structure and methodology," in *IEEE PES General Meeting*, July 2010, pp. 1–1.

[5] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *IEEE Power and Energy Magazine*, vol. 1, no. 2, pp. 56–63, Mar 2003.

[6] (2017) Hioki-pw3360-20 power analyzer. [Online]. Available: http://www.electrotest.co.nz/shop/Electrical+Test/PQA++Power+Loggers/Power+Analyser/HIOKI+PW3360-20-HIO+PW3360-20.html

[7] (2017) Hioki-9660-current probe. [Online]. Available: http://www.testequipmentdepot.com/hioki/images/9660.jpg

[8] R. P. Foundation. (2017) Raspberry pi zero. [Online]. Available: https://www.raspberrypi.org/app/uploads/2017/05/Raspberry-Pi-Zero-Overhead-1-1748x1080.jpg

[9] F. Azzola. (2017) Mqtt protocol tutorial: Step by step guide. [Online]. Available: https://www.survivingwithandroid.com/2016/10/mqtt-protocol-tutorial.html

[10] A. S. Bouhouras, P. A. Gkaidatzis, K. C. Chatzisavvas, E. Panagiotou, N. Poulakis, and G. C. Christoforidis, "Load signature formulation for non-intrusive load

monitoring based on current measurements," *Energies*, vol. 10, no. 4, 2017. [Online]. Available: http://www.mdpi.com/1996-1073/10/4/538

[11] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870 – 1891, Dec 1992.

[12] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, pp. 76–84, February 2011.

[13] M. B. Figueiredo, A. de Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems," in *Adaptive and Natural Computing Algorithms: 10th International Conference, ICANNGA 2011, Ljubljana, Slovenia, April 14-16, 2011, Proceedings, Part II*, A. Dobnikar, U. Lotrič, and B. Šter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 31–40.

[14] A. I. Cole and A. Albicki, "Data extraction for effective non-intrusive identification of residential power loads," in *IMTC/98 Conference Proceedings. IEEE Instrumentation and Measurement Technology Conference. Where Instrumentation is Going (Cat. No.98CH36222)*, vol. 2, May 1998, pp. 812–815 vol.2.

[15] H.-H. Chang, "Non-intrusive demand monitoring and load identification for energy management systems based on transient feature analyses," *Energies*, vol. 5, no. 11, pp. 4569–4589, 2012. [Online]. Available: http://www.mdpi.com/1996-1073/5/11/4569

[16] T. D. Huang, W. S. Wang, and K. L. Lian, "A new power signature for nonintrusive appliance load monitoring," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1994–1995, July 2015.

[17] N. F. Esa, M. P. Abdullah, and M. Y. Hassan, "A review disaggregation method in non-intrusive appliance load monitoring," *Renewable and Sustainable Energy Reviews*, vol. 66, pp. 163–173, 2016.

[18] V. Amenta and G. M. Tina, "Load demand disaggregation based on simple load signature and user's feedback," vol. 83, 2015, pp. 380 – 388, sustainability in Energy and Buildings: Proceedings of the 7th International Conference SEB-15.

[19] (2017) Node red website. [Online]. Available: https://nodered.org/

[20] Mqtt homepage. [Online]. Available: http://mqtt.org/

[21] K. D. Anderson, M. E. Bergs, A. Ocneanu, D. Benitez, and J. M. F. Moura, "Event detection for non intrusive load monitoring," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 3312–3317.

[22] C. C. Yang, C. S. Soh, and V. V. Yap, "Comparative study of event detection methods for non-intrusive appliance load monitoring," *Energy Procedia*, vol. 61, no. Supplement C, pp. 1840 – 1843, 2014, international Conference on Applied Energy, ICAE2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1876610214032548

[23] StatSoft. (2013) Electronic statistics textbook. [Online]. Available: http://www.statsoft.com/textbook/

[24] J. Brownlee. (2014, sep) Tutorial to implement k-nearest neighbors in python from scratch. [Online]. Available: https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

[25] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, Feb 1979.