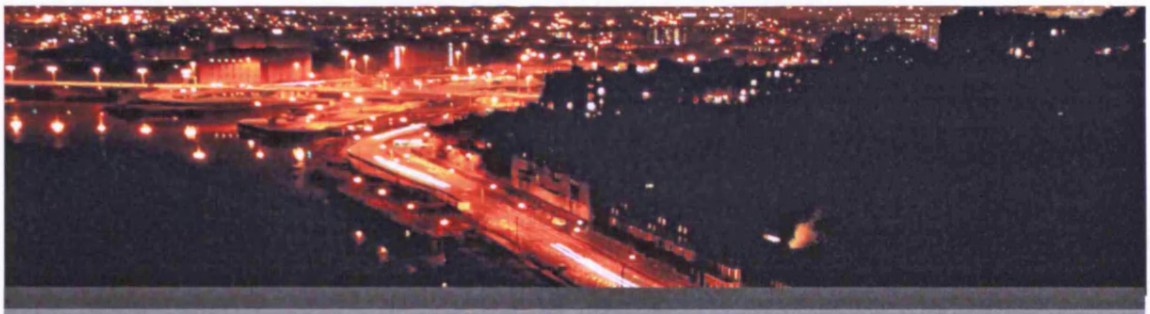


Intelligent Monitoring & Management of Light Sources

Paul Martin Edwards

August 2007

Intelligent Process Monitoring & Management Centre

Cardiff School of Engineering

Cardiff University

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

DECLARATIONS

DECLARATION

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed P. Edwards ..... (candidate) Date 27/9/07

STATEMENT 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD

Signed P. Edwards ..... (candidate) Date 27/9/07

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed P. Edwards ..... (candidate) Date 27/9/07

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed P. Edwards ..... (candidate) Date 27/9/07

ACKNOWLEDGEMENTS

I would like to thank Dr Roger Grosvenor and Mr Paul Prickett for their continual help, enthusiasm and valuable guidance throughout the duration of this work. The help of the Cardiff School of Engineering in providing the facilities and funding for this work is also gratefully acknowledged.

ABSTRACT

A new method for the monitoring of filament lamps and low pressure discharge lamps has been developed. The new technique monitors the electrical characteristics of the lamp to provide real time analysis of the lamp's condition, without the need for additional wires or expensive light sensors. The advent of low-cost microcontrollers developed for electrical metering applications means that not only is this technique technologically practical, it is also financially viable. The deployment of this technology, particularly in the case of UV water sterilizers, would improve safety and save the significant expense and environmental impact of unnecessary replacement lamps.

ACRONYMS

AC	Alternating Current
ADC	Analogue to Digital Converter
BSM	Basic Signalling Module
CFL	Compact Fluorescent Lamp
COM	Component Object Model
DC	Direct Current
DLL	Dynamic Link Library
EEPROM	Electrically Erasable Programmable Read Only Memory
EU	European Union
GE	General Electric
GLS	General Lighting Service
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
HID	High Intensity Discharge
IC	Integrated Circuit
IP	Internet Protocol
LCD	Liquid Crystal Display
LED	Light Emitting Diode
NTC	Negative Temperature Coefficient
PC	Personal Computer
PCB	Printed Circuit Board
OLC	Outdoor Luminaire Controller
Op-Amp	Operational Amplifier
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
RMS	Root Mean Square
RTC	Real Time Clock
SC	Segment Controller
SID	Static Induction Discharge
UML	Unified Modelling Language
USB	Universal Serial Bus
UV	Ultra Violet

INDEX

1 Introduction 1

2 Background Review 7
2.1 Lamp Types and Emerging Lamp Monitoring Technology 8
2.1.1 Filament Lamps 8
2.1.2 Fluorescent Lamps 10
2.1.3 HID (High Intensity Discharge) Lamps 14
2.1.4 SID (Static Induction Discharge) Lamps 18
2.1.5 LED (Light Emitting Diode) Lamps 20
2.2 Emerging Lamp Monitoring Systems 21
2.2.1 Philips Lighting 21
2.2.2 IEI (Integrated Equipment & Instrumentation) 23
2.2.3 Mayrise Systems 23
2.2.4 Harvard Engineering 25
2.2.5 Telensa Ltd 27
2.2.6 Archnet Technology Ltd 29
2.2.7 Fortran Traffic Systems Ltd 29
2.3 Summary 30

3 Filament Lamps - Detecting Failure 32
3.1 Traditional Filament Lamp Testing and Monitoring Techniques 33
3.2 Detecting Defective Lamp Construction 35
3.2.1 Filament and Support Structures 36
3.2.2 Glass Envelope 42
3.2.3 Gas Filling 42
3.2.4 Electrical Connections 44
3.3 Detecting Environmental Failure Causes 46
3.3.1 Over Voltage 46
3.3.2 Vibration 46
3.3.3 Temperature 47
3.3.4 Thermal Shock 47
3.4 Predicting Failure Due to Usage 47

4 Filament Lamp Test Rig 51
4.1 Hardware Design 52
4.1.1 Current and Voltage Monitoring 52
4.1.2 Light Level Monitoring 57
4.1.3 Temperature Monitoring 59
4.1.4 System Integration 62
4.2 Test Rig Software Design 65
4.2.1 Data Recording 66
4.2.2 Data Analysis 70

5 Filament Lamp Monitoring Experiments 74
5.1 Continuous Monitoring 75
5.1.1 Lamp Voltage Monitoring 76
5.1.2 Lamp Current Monitoring 78
5.1.3 Filament Resistance Monitoring 79
5.1.4 Lamp Power Monitoring 80
5.1.5 Lamp Light Output Monitoring 81
5.1.6 Temperature Monitoring 82
5.1.7 Mains Voltage Compensation 86
5.1.8 Results & Discussion 92
5.2 Detecting Incorrectly Filled Lamps 102
5.2.1 Procedure 103
5.2.2 Results 105
5.3 Summary 116

6 UV Discharge Lamps – Detecting Failure 119
6.1 Introduction 120
6.2 Traditional Monitoring Techniques 121
6.2.1 Hour Meters & Scheduled Replacement 121
6.2.2 Failure Detection 122
6.2.3 Light Level Monitoring 124
6.3 Failure Modes and Potential Monitoring Techniques 125
6.3.1 Mechanical 126
6.3.2 Electrode Failure 126
6.3.3 Mercury Loss 130
7 UV Discharge Lamp Experiments 133
7.1 Steady State Electrical Parameter Monitoring 134
7.1.1 Procedure 134
7.1.2 Results 136
7.1.3 Discussion 140
7.2 Striking Voltage Measurement 141
7.2.1 Procedure 141
7.2.2 Discussion 143
7.3 Electrode Monitoring 144
7.3.1 Electrode Temperature Measurement 145
7.3.2 Lamp Rectification Monitoring 152
7.3.3 Discussion 154
7.4 Accelerated Electrode Ageing 156
7.4.1 Procedure 157
7.4.2 Results 159
7.4.3 Discussion 162

8 Discussion 164
8.1 Potential Implementation 165
8.2 Filament Lamps and the Future 168
8.3 UV Lamps 171
8.4 Lamp Monitoring Systems 172

9 Conclusions and Further Work 174
9.1 Filament Lamps 175
9.2 UV Low Pressure Discharge Lamps 177
9.3 Further Work 178
9.3.1 Filament Lamps 178
9.3.2 UV Low Pressure Discharge Lamps 179
9.3.3 Monitoring Systems 179
9.4 Summary 180

References 181

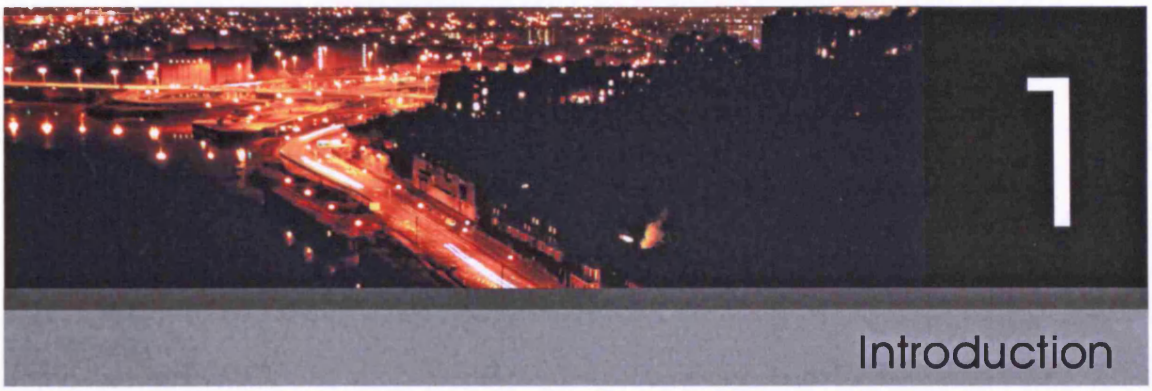
Appendix A – Test Rig Construction 190
PCB Layouts 191
Circuit Diagrams 194

Appendix B – Test Rig Software 196
Result Recording Program Listing 197
Result Analysis Program Listing 223

Appendix C – Filament Lamp Results 242
Continuous Monitoring Results 243
Voltage Compensation Results 252
Initial Power Draw (Over 100 Hours) 254
Incorrectly Filled Lamps 263
Initial Resistance Increase of Unaltered Lamps 10 - 17 276

Appendix D – UV Lamp Results 280
Aged Sample Data 281
Filament Temperature Monitoring 281
Lamp Striking Voltages 283
Accelerated Ageing 283

Appendix E – Publications 284
A Review of Lamp Condition Monitoring Technologies 285
A Method for Detecting Incorrectly Evacuated Filament Lamps 286



1

Introduction

Nearly every aspect of modern day life involves the use of electric lamps; from lamps used to light homes and offices to those used for directing traffic and lighting the road. They are all essential and whilst using completely different technologies, they share one inevitable feature, they will all fail. The time taken for each lamp to fail varies widely, from a few hours up to a few decades for some lamps, such as the recently launched range of induction lamps[1].

Depending on the task a lamp is performing, the consequences of failure can be surprisingly significant. For example, traffic lights failing unexpectedly can cause accidents and traffic congestion, a study in Finland found that accident rates at road junctions were over 3.5 times higher when traffic signals were out of operation[2]; water sterilizers, which rely on UV lamps to remove the risk of illness caused by microbial contamination[3], will not be effective if the lamp is failing or has failed.

It also makes good economic sense to be able to monitor lamps effectively. Often light fittings will be in inaccessible areas where maintenance is difficult, such as above a swimming pool or in the roof of a theatre. Knowing the condition of the lamps in such places means maintenance can be scheduled for the most appropriate time, such as during a period when the pool is drained, or in the case of a theatre, when the lighting rig is being set up for a show.

Currently, the most common way of ensuring continued operation is to schedule routine replacement of the lamp. This is very costly and does not guarantee that a new lamp will not fail unexpectedly as the result of a manufacturing defect. An alternative approach has

been to provide a standby lamp which is switched on when the first lamp fails; however this is not perfect, as there will be a time delay in switch over which, depending on the type of lamp and control gear used, can be significant. This method can significantly add to the size and cost of a light fitting due to the duplication of parts within it and relies upon the requisite reliability of sensing and switching.

A traditional method for estimating the remaining life of a lamp is to measure how long it has been running for, using a device such as an hour meter; however this has a number of significant shortcomings:-

- The initial life expectancy quoted by the manufacturer is inherently inaccurate; some lamps may fail after a very short period of time due to manufacturing defects, whilst others may last double their expected life, which can cause major errors in the predicted replacement time of the lamp. Figure 1 demonstrates the large variability in lamp life, it can be seen that whilst from the same batch, some fluorescent lamps may last as little as 5,000 hours, whilst some may last in excess of 16,000 hours.

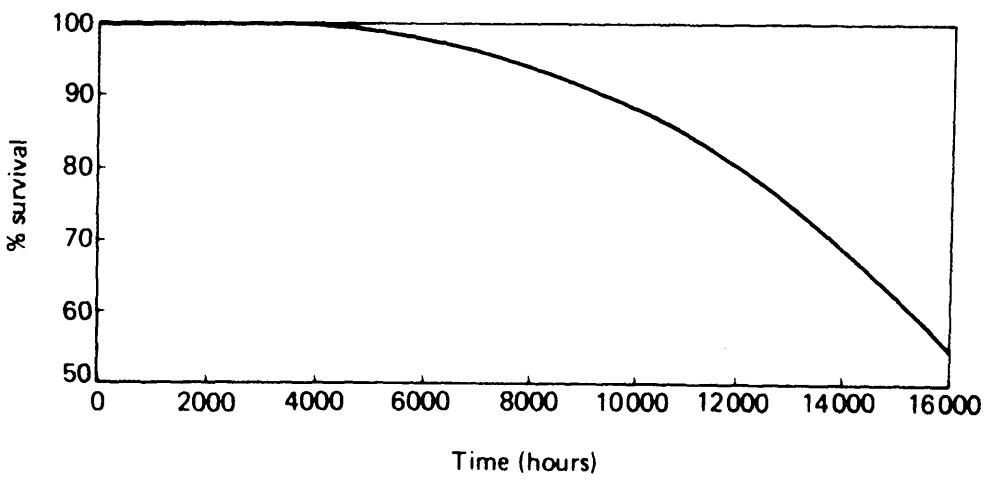


Figure 1 - Typical Fluorescent Lamp Survival Curve[4]

- A lot of lamps are dimmed, which will significantly affect their life expectancy. As the filament is run cooler, it evaporates at a slower rate, which means the lamp will usually last longer than if it were run at full power.

- Environmental factors, such as frequency of switching and vibration, are not accounted for.

A further, well established method of lamp monitoring is to monitor a parameter of the lamp, such as the operating current, so that when the filament breaks or the discharge ceases, an alarm is triggered. Patents using this technique date back decades, one particularly notable US patent was granted in 1987 for a system to monitor all the panel lamps in an elevator for failure, using a single current transducer[5]; although good for speeding up detection of failed lamps, this technique is useless for the purpose of pre-emptive maintenance.

Being able to monitor lighting installations in real time, so that the likelihood of lamp failures can be predicted, and maintenance scheduled accordingly, would be extremely beneficial from a both a safety and economic perspective. If a failure did occur, this could be detected immediately and reported to the end user, so that they are provided with sufficient information to decide whether to take immediate action or to use the information to optimise scheduled maintenance when the failure does not present a major problem.

The following research looks at how predictive lamp maintenance may be achieved for different lamp types. Chapter two begins with a look at the different lamp technologies

that are currently in use and then goes on to discuss the present monitoring techniques and systems, together with some of their shortcomings.

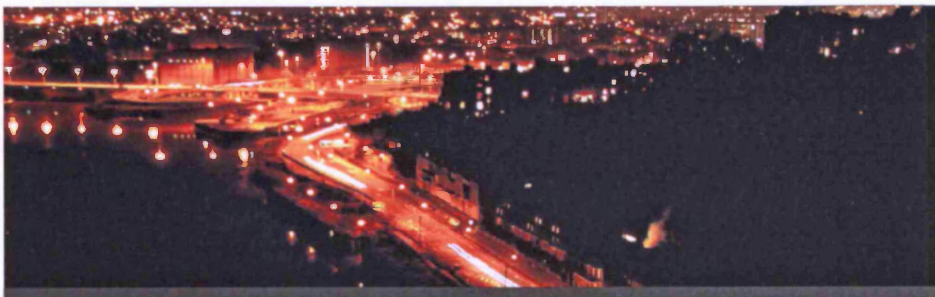
In chapter three, filament lamps are focused upon as being an area where there is potential for improved condition monitoring. The chapter concludes by considering the ways in which impending failure could be detected.

Chapter four details the construction of an automated test rig to enable continuous monitoring of the lamp characteristics as they burned in the lab, in order to try and identify failure indicative behaviour. Chapter five details a further experiment, that attempts to separate inherently faulty lamps before they are put into service, and concludes with a brief discussion of the findings.

Chapter six considers low pressure discharge lamps and focuses primarily on germicidal ultraviolet (UV) lamps as an area where condition monitoring could play a very important role. It concludes with a discussion of how failure could be detected in such lamps.

Chapter seven follows on with details of a series of experiments that were conducted in an attempt to identify impending failure characteristics. The experiments include monitoring the electrical discharge parameters of the lamp, electrode characteristics monitoring and, finally, accelerated ageing of a lamp to analyse its entire life behaviour from new until the point of failure.

The discussion in chapter eight focuses on the general outcomes from all of the experiments and looks at the future possibilities for both lighting technology and its associated monitoring systems. Finally, the conclusions drawn from this work are contained in chapter nine, together with details of what further work is needed to allow comprehensive monitoring of all lamp types.



2

Background Review

2.1 Lamp Types and Emerging Lamp Monitoring Technology

There are a large range of different lighting technologies in use today. The following sections consider how each of the different lamp technologies work, and report on the existing research into monitoring them effectively.

2.1.1 Filament Lamps

A typical filament lamp, as shown in Figure 2, consists of a thin coiled tungsten filament held up by a number of intermediate molybdenum support wires and two nickel or nickel plated[7] wires at either end, which act as the connections to the filament. When current flows through the filament it is heated to around 2200°C[6], causing it to incandess. Tungsten is almost always used due to it having a "high melting point and relatively low rate of evaporation at high temperatures."[7] To stop the filament reacting with oxygen in the air and burning up prematurely, it is enclosed in a glass envelope which is either evacuated of air, or filled with an inert gas such as Argon.

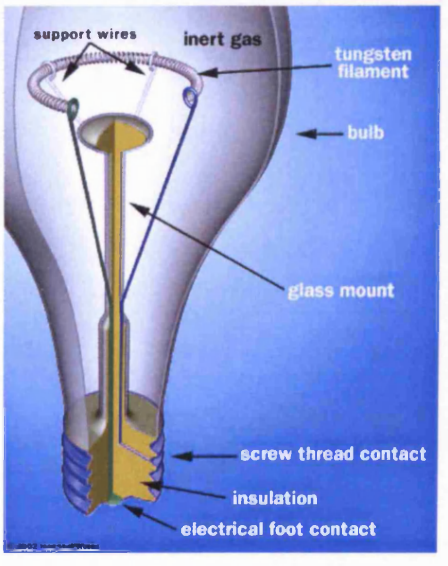


Figure 2 :- *Structure of a typical filament lamp[6]*

In a bulb which is evacuated, tungsten atoms which evaporate from the filament are deposited on the inside of the envelope, leading to reduced light output due to blackening of the glass as well as thinning of the filament due to the loss of Tungsten atoms. By filling the bulb with an inert gas such as Argon, many of the Tungsten atoms which evaporate off the filament collide with Argon atoms and rebound back onto the filament. This reduces the problems of glass blackening and filament thining, and allows the filament to be run at higher temperatures, which increases the radiation efficacy. However, introducing the Argon gas also increases the thermal conduction between the filament and glass envelope, which reduces the overall efficacy of the lamp. "Generally, vacuum is advantageous for lower power lamps and gas filling for higher power lamps. The changeover occurs at about 15W for mains voltage lamps and at about 3W for miniature lamps, such as torch bulbs." "Generally, household lamps are filled to just below 1 atm pressure with 90% argon and 10% nitrogen".[7] "Typical incandescent lamp life ranges from a few hundred to 1500 hours."[8]

One variation of the filament lamp is the Halogen Lamp. Increasing the running temperature of a lamp filament drastically increases the operating efficiency of the lamp, however it also considerably reduces the life expectancy due to increased filament evaporation. To solve this problem, halogen lamps are filled with a gas from the halogen group. Providing the temperature is high enough, the gas combines with the Tungsten vapour and causes the Tungsten atoms to be redistributed on the filament. To keep the temperature inside the lamp high, the glass envelope is made as small as possible, which means it is very close to the filament and consequently gets very hot. If normal glass were

used it would start to melt, so instead Quartz glass is used as it can withstand the very high temperatures involved.[9] "Some tungsten [halogen] lamps last more than 5000 hours."[8]

Nearly all the existing monitoring systems for conventional filament and halogen lamps rely on detecting an interruption in current flow as a means of determining that a lamp has failed completely. These systems are often incorporated into high specification cars, such as the Jaguar XJ-S[10]. There appear to be no dedicated systems or devices for performing predictive lamp failure monitoring. One reason for this maybe that filament lamps are considered to be old technology, with fluorescent lighting and LED's often being used to replace light bulbs; however many applications still rely on them, for example, the majority of stage lights in theatres still use them as the primary source of illumination. It is therefore considered that there would be merit in developing and deploying technology that would monitor such lamps and warn of imminent failure.

2.1.2 Fluorescent Lamps

A typical fluorescent tube of the type commonly found in offices and homes consists of a long thin glass tube, filled with a mixture of gasses and with two Tungsten filaments sealed in at either end of the tube, as shown in Figure 3, (an exception are cold cathode fluorescent lamps, which do not have any heaters as they rely on a higher starting voltage). The filaments are usually covered in a mixture of Barium, Strontium and Calcium oxides to aid thermionic emission at lower temperatures.[11] The gas inside the lamp envelope normally consists of a mixture of Mercury vapour and Argon.[12]

During a normal starting cycle, the two cathodes are usually heated for one to two seconds to release electrons. Once this has happened a large voltage is applied across the tube, which attracts the electrons to the other end of the tube, however due to the presence of large numbers of Argon atoms in the gas they collide with these first. When an electron with high enough velocity collides with an Argon atom, impact ionisation can occur. This results in the release of more electrons, which in turn impact with more atoms, producing avalanche ionisation of the atoms. This leads to a rapidly increasing current flow through the tube which must be limited to prevent damage to the tube.

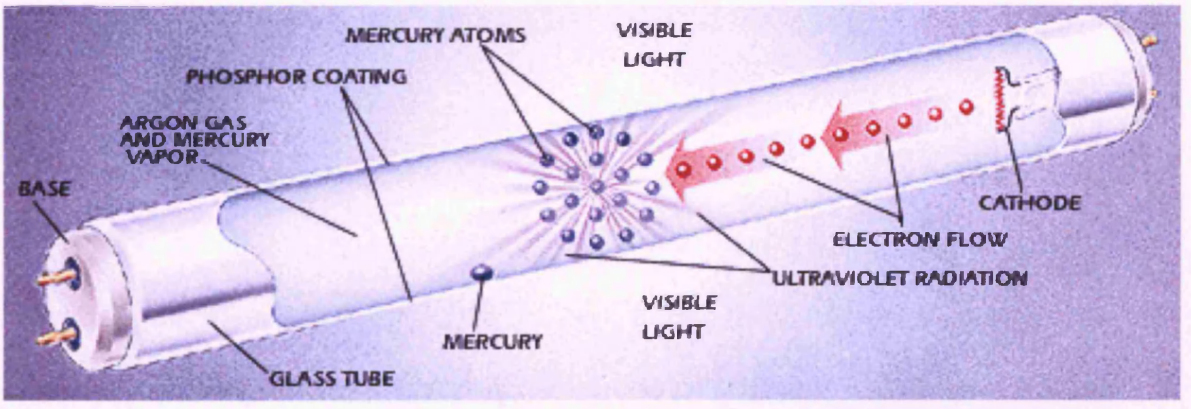


Figure 3 :- Structure and operation of a typical fluorescent tube[13]

Once the tube has 'struck', external control gear reduces the voltage across the tube to keep the current through the tube under control. By reducing the voltage across the tube, the electrons move away from the cathodes at a lower velocity; this means that the majority of the electrons leaving the cathode will have inadequate kinetic energy to cause ionisation of the atoms they collide with and hence the current is stabilised. As the tube warms up, the mercury turns into mercury vapour. When electrons strike the mercury vapour atoms they cause an electron to temporarily move to a higher energy orbit; when the electron returns to its original orbit a photon is emitted, which is how the light is generated.

Approximately 65% of the light generated by the mercury atoms is at a wavelength of 253.7nm while 10-20% is at a wavelength of 185nm[11], this means that the vast majority of light emitted is in the ultraviolet band and not visible to humans. To solve this problem the inside of the glass tube is coated with a thin layer of phosphors; the phosphor coating's atoms absorb the uv light photon, causing an electron to jump to a higher energy orbit. When the electron returns to its original orbit a photon is emitted again; however due to small energy losses during this process the photon emitted is of a lower energy and hence longer wavelength than the one that was absorbed. By using different phosphor combinations, the UV light can be converted to light of a different wavelength; this can be seen in the various colour fluorescent lamps available on the market.

One variation on the fluorescent lamp is the germicidal UV lamp. The operating principle of the lamp is exactly the same as the fluorescent lamp, except that there is no phosphor coating on the inside of the tube. Also the glass tube is made from quartz glass in order to let the UV light out, as normal soda-lime glass will block most UV emission. The result of these alterations is that the majority of the light output is at 253.7nm (UV-C). Although not visible to the human eye, this wavelength is highly effective at deactivating the DNA of bacteria and viruses, preventing them from reproducing and causing disease.[14]

Traditional monitoring techniques for fluorescent lamps have been based around the idea of detecting whether a lamp has failed or not. However in most applications it would be useful to know how long a lamp has left before it needs replacing. Most people have experienced the frustration of an electrician calling to change a flickering fluorescent lamp in the office, only for one nearby to fail as well a day or two later; if the lamps were

monitored better then all the ones near the end of their life could be changed at the same time, saving time and money.

In UV water treatment systems, lamps are routinely replaced to try and ensure they do not fail during use. Most UV treatment unit manufacturers recommend annual replacement[15] and some systems, such as the Aquapro range, even have alarms in case the lamp does fail[16]; however this is not a very satisfactory solution, as following a failure it could be anywhere from a few minutes to a few days before the lamp is changed and in that time harmful bacteria may go untreated. Some systems incorporate a second standby lamp, but this adds significantly to the cost of the system. Knowing when a lamp is reaching the end of its life, based on its electrical characteristics, would enable it to be replaced before it fails altogether.

Considering the large number in use, it is surprising that there seems to be very little commercial activity in condition monitoring of conventional, heated cathode, fluorescent lamps. This may be because fluorescent lamps are inexpensive, as too are the fittings they are used in, so the cost of any monitoring system has to be very low in order to make it economically viable. However the potential market, and hence the money that could be saved in both running costs and lamp costs if they are changed at the right time, means that there is potential for low cost monitoring to become popular. With the increasing concern over the disposal of mercury, as highlighted by an American government article which instructs people to evacuate the room for 15 minutes and open all windows if a fluorescent lamp breaks[17], it also makes sense to minimise the disposal of lamps by using them until

just before they fail. The same monitoring technology could also be adapted for use in compact fluorescent lamps, which operate on the same principle.

In safety critical systems, such as UV sterilizers, the cost of any monitoring system is more of a secondary concern. Despite this, there is still limited commercial activity, which may well be because of the specialist nature of the market and the relatively small numbers in use.

One area of fluorescent lighting where there has been considerable activity is for cold cathode fluorescent lamps, as often used in LCD backlights in devices such as laptops. This is presumably due to the high value of the equipment they are used in. In a white paper for CEYX Technologies[18], a control system for the cold cathode fluorescent lamps used in LCD backlighting is described that monitors the lamp current in order to enable the controller to maintain a consistent light level from the lamp. The controller also uses the lamp current to implement failure prediction using built in diagnostics.

There is clearly a lot of potential for the condition monitoring of traditional heated cathode fluorescent lamps, providing appropriate technology can be developed to enable it to be done simply, cheaply and reliably.

2.1.3 HID (High Intensity Discharge) Lamps

A typical HID lamp consists of two tungsten electrodes housed in a fused quartz tube. Inside the quartz tube there is a mixture of gas and metals. The gas is intended to allow the

starting of the lamp whilst the metals take over light production once the temperature inside the tube is high enough for the metal to vaporise. The quartz tube is usually housed inside an outer glass envelope, which serves to contain any parts which might fly off when the inner quartz tube explodes at the end of its life, which is a fairly common occurance. The glass envelope is also used in most lamps to filter out the UV content of the light given out, which could otherwise cause skin and eye damage.

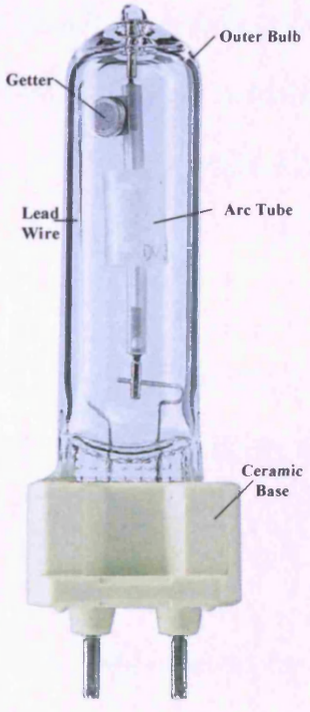


Figure 4 :- Metal Halide Lamp[19]

The starting procedure for the lamp is similar to a cold cathode fluorescent tube. A high voltage is applied across the lamp which causes avalanche ionisation of the gas atoms in the tube; the lamp ballast then drops the voltage across the tube to regulate the current through the lamp. As the gas in the tube heats up, the metal particles vaporise and when the metal atoms are struck by an electron they emit light. This process can take a few

minutes to occur, and can be observed by the changing intensity and colour of the light given off by the lamp as it heats up.

There are many different types of gas and metal combinations available in HID lamps, each producing a different colour of light. One of the earliest HID lamps was the Mercury vapour lamp, which produced a blue/green tinted light output, while today Metal Halide lamps are very popular due to their near natural light colour, a typical example is shown in figure 4. Common applications of HID lamps are in multimedia projectors, car headlights, outdoor lighting and shop lighting, where their bright, efficient, small size and near white light makes them an ideal choice.

Perhaps not surprisingly, there has been a lot of interest in monitoring these types of lamps, primarily due to their use in expensive multimedia projectors, and also due to the possibility of lamp explosion at failure.

In 1996, Osram Sylvania Inc were granted a patent for a "Ballast containing protection circuit for detecting rectification of arc discharge lamp"[20]. This device measures the DC voltage component which develops across the lamp as it approaches the end of its life, and disables the inverter before the cathodes are overheated.

The Hewlett Packard Company were granted a patent in 2002 for an invention which "provides an end-of-life notification signal, indicating that the arc lamp bulb should be replaced, close to the end of the useful life of the arc lamp bulb, but before the bulb

actually burns out. The threshold property value is chosen to correspond to a point in the life of the arc lamp bulb that precedes the actual burn out of the bulb."[21]

Koninkl Philips Electronics were granted a patent in 2003 for a device which monitors gas discharge lamps. "A warning signal is generated when a test value is measured which corresponds to a clear reduction in the lamp voltage, for the purpose of better predicting the life of the lamps and the risk of explosion."[22]

In 2004 a patent application by the BENQ Corporation was published, one of the claims being "A projector capable of detecting remaining lifetime of the light source lamp therein, comprising: an image projection device having a light source lamp with a pair of lamp electrodes; a detection device for detecting a voltage across the lamp electrodes; an analogue-to-digital converter for converting the voltage to a digital value; and a control unit for comparing the digital value with a relational table to calculate the remaining lifetime of the lamp."[23]

From all the patents found and reviewed, it is clear that a lot of work has been done on detecting the imminent end of life of discharge lamps, particularly those used in projectors. However the first three patents do not discuss determining the actual remaining lamp life at any particular time; rather, they concentrate on setting thresholds based on an individual, or combination of, lamp parameters, to warn the user when the lamp is about to expire and shut it off before the lamp explodes. The final patent application (BENQ) does discuss a device for measuring the remaining lamp life at any point in time, and displaying it each time at start up for the user to see. However it is clear that the primary focus of the work

has been for projector lamps, which is normally a stand alone application. It is clear, from these patents, that there is considerable commercial interest in the condition monitoring of HID lamps and although there are some improvements still to be made, developments are being made very quickly in this area of condition monitoring.

2.1.4 SID (Static Induction Discharge) Lamps

The electrodes in discharge lamps are one of the major factors in limiting the life of the lamp. Induction lamps, however, do not have any. As a result of this a typical induction lamp such as the 55W Philips QL lamp has a guaranteed life of 60,000 hours[24]. The lamps work by using an RF power coupler, which consists of a coil wound on a ferrite rod; this rod fits inside a cavity in the bottom of the lamp. The energy is transferred through the glass wall of the bulb by electromagnetic induction, where it excites mercury vapour within the bulb to generate UV light, which is then converted to visible light by the phosphors on the inside surface of the lamp in the same way as in the fluorescent tube.

The lamp requires a special ballast to provide RF power to the RF coupler. The ballast must provide a high voltage to the coil when starting the lamp and then drop the power supplied once the lamp is running.

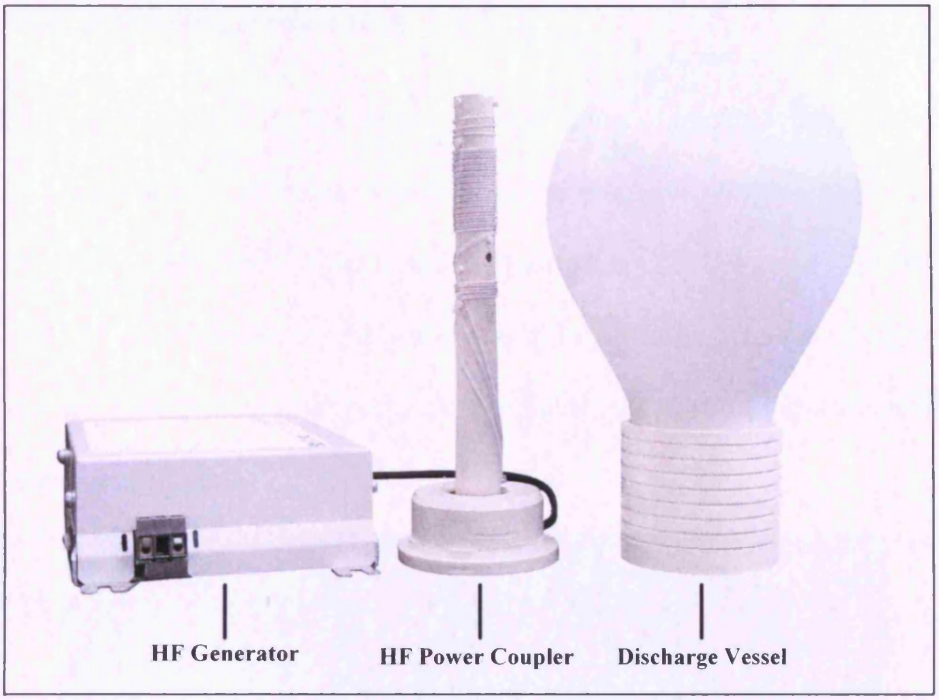


Figure 5 :- Philips QL Induction Lamp System[25]

At the moment there appears to be no information available on condition monitoring techniques for lamps of this type, this is presumably due to two main reasons; firstly the lamps are very new technology and so the numbers installed are low compared with other types of lighting; hence the demand for condition monitoring is low. Secondly the lamp has a very long life, which means obtaining data on the characteristics of the lamp over its lifetime would be very time consuming. As replacement of the lamp is so infrequent it is possible the lamp may not even be the most frequently replaced part of the system, as the electronic ballast used to drive the lamps could cause a significant percentage of failures itself, which would be very hard to predict. Consequently developing a monitoring system for this type of lamp is probably not cost effective or practical until the lamp is in more widespread use, and more data is available about it.

2.1.5 LED (Light Emitting Diode) Lamps

As the name suggests, an LED is a semiconductor diode which emits light when forward biased. A large range of different wavelengths are available, ranging from ultra violet, through the visible spectrum, to infra red. Traditionally LED's were only low power devices, used for user interfaces. However due to major improvements in efficiency and power, they can now be found in torches, stage lights as shown in Figure 6, traffic lights and even some room lights.

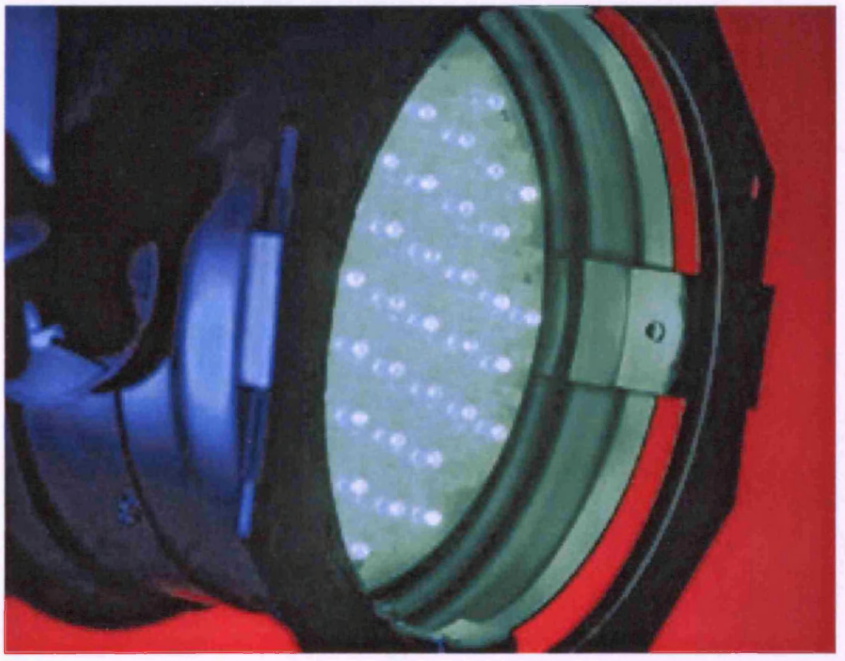


Figure 6 :- RGB LED Colour mixing PAR can [26]

There appears to be little or no existing information in either traditional journal papers or on the internet regarding condition monitoring techniques for LED lamps. Although LED's are not new, their use as major light sources is, and consequently the numbers in use are comparatively few, with most being used for specialist applications such as traffic lights and torches. The reason for there not being much interest in condition monitoring of LED lamps is almost certainly the same as for SID (Static Induction Discharge) lamps.

2.2 Emerging Lamp Monitoring Systems

A few advanced monitoring systems for outdoor lighting installations have recently become commercially available, and are described in the following sections.

2.2.1 Philips Lighting

One of the most recent, and arguably most comprehensive systems, is one advertised by Philips in their "Telemanagement solutions in outdoor lighting" brochure[27]. Philips currently supply two systems, called Starsense and Telesense, which both allow greater control and monitoring of outdoor lighting installations than was previously possible.

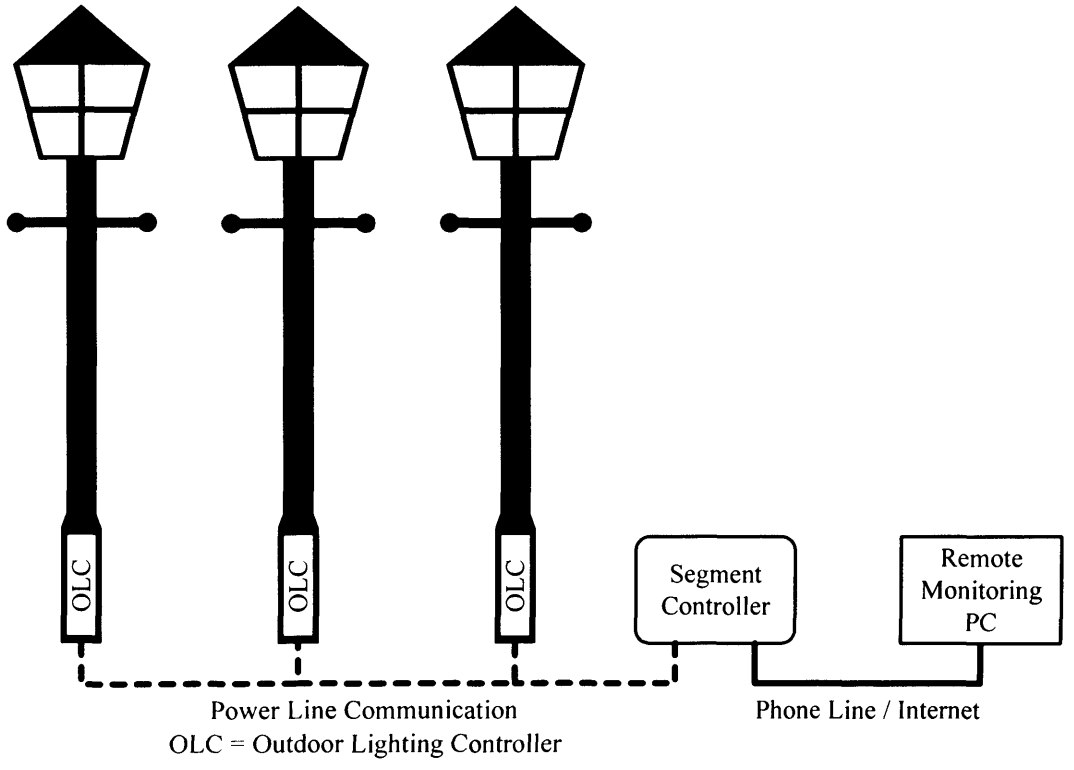


Figure 7 :- Philips Telemanagement Architecture

Both systems revolve around each light being fitted with special electronic control gear which provides dimming control of each discharge lamp. The electronic ballast also monitors the discharge lamp's inner electrical parameters to accurately predict when the lamp is about to fail, thus enabling efficient and timely replacement of the lamp. The systems are claimed to provide significant cost savings by enabling lamps to be dimmed to the required light level depending on the time of day, weather and age of the lamps, etc. They also remove the need to employ people to scout for failed lamps.

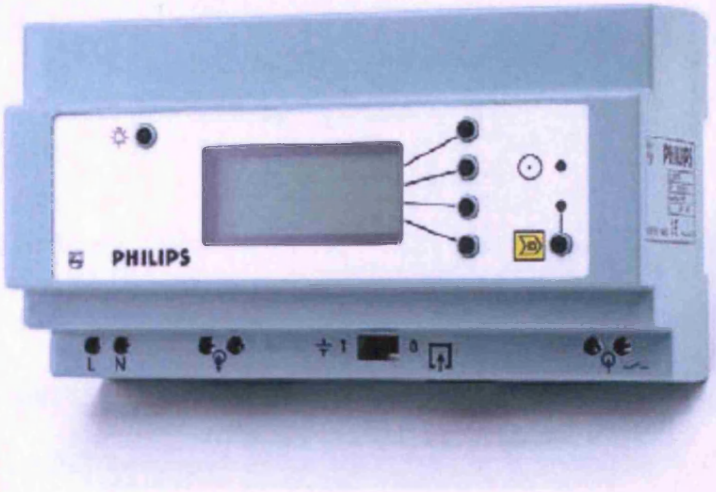


Figure 8 :- Philips LFC7050 Segment Controller [28]

The Starsense system features comprehensive functionality to allow it to interact with traffic management systems and geographical information systems, this makes it ideal for larger installations such as motorways and large interchanges. The Telesense system has a simpler architecture and is intended for stand alone street lighting applications only. With both systems there is a Segment Controller (SC), as shown in figure 8, for each group of lights. The SC communicates via the power lines with the Outdoor Luminaire Controller (OLC) installed in every luminaire. It is the OLC which monitors the lamp and controls its

dimming level under the command of the SC. The SC in turn communicates with the monitoring PC at the control centre via a phone line or the internet.

2.2.2 IEI (Integrated Equipment & Instrumentation)

A very similar system is presented in a paper written by R. Seevaratnam of Integrated Equipment & Instrumentation titled "Street and Public Light Monitoring System (SPL)"[29]. The system incorporates a Basic Signalling Module (BSM) in each luminaire which communicates via the power line with a data logger module housed with the power distribution board; from here data is sent back to a central computer via a telephone line, fibre optic cable or wireless modem.

The system enables remote switching of the luminaire and dimming if the ballast supports it. As well as control, the system also provides feedback on fault conditions, including lamp failure, inefficient lamp running and the door of the lamppost being open. A database on the central computer keeps track of every switching cycle of each lamp, together with any faults that have been detected. The detection of inefficient lamp operation also gives the user the option of replacing lamps before they fail.

2.2.3 Mayrise Systems

One of the products on offer from Mayrise Systems is a software management system designed to help manage the operation and maintenance of street lighting installations. This system has also been integrated with remote monitoring systems from two different manufacturers to provide a complete management solution.

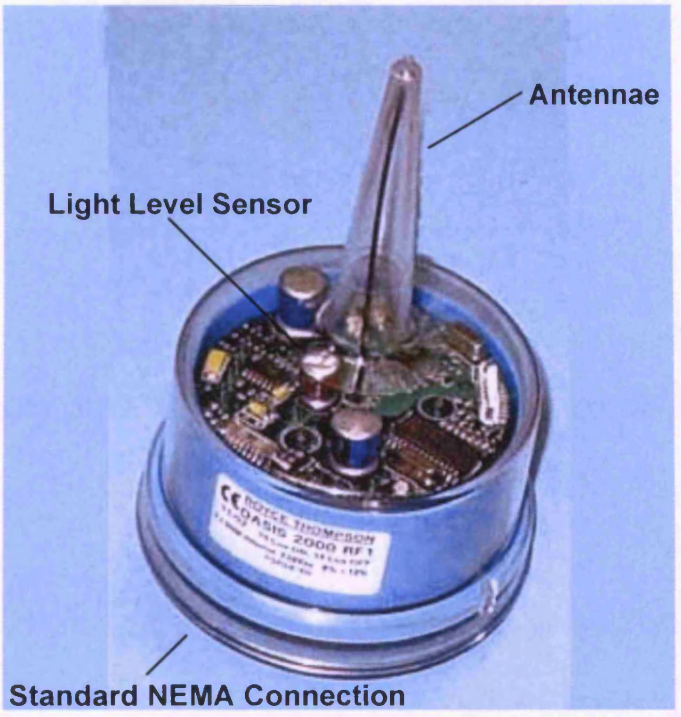


Figure 9 :- Royce Thompson Oasis 2000 Module[30]

The first monitoring system, from Royce Thompson Ltd, uses RF units, shown in figure 9, that replace the photocells in luminaries. "The Oasis 2000 RF can dim electronic ballasts according to timetables or road conditions and also monitors street lighting to give desktop maintenance reports."[30] The system utilizes a street level cluster controller which downloads timetables from the central computer and modifies the dimming levels of the lamps accordingly.

The second monitoring system, called 'Lightmaster', which is produced by Mayflower Intelligent Management Systems, also uses a RF module to replace photocells. This system works by using sub-master controllers to control a number of individual lights each, and then master controllers to control the sub-master controllers, which can control up to 32,000 nodes[31]. The master controllers can be linked to a central computer through radio, existing wide area networks or GSM.[31]

The Mayrise system is reported as currently being implemented and used by a number of local authorities in the UK.

2.2.4 Harvard Engineering

The LeafNut system developed by Harvard Engineering is another wireless street lighting control system. Each luminaire is fitted with a Harvard electronic dimming ballast which is connected to a wireless device called the LeafNode unit, which is designed to replace the photocell on top of the lantern. To control the LeafNode units, a BranchNode unit is required; this is a slightly larger device than the LeafNode units, but is still able to be mounted on top of a lantern. The BranchNode communicates with the LeafNodes via an 868 MHz wireless network and can control up to 256 LeafNode units within a one kilometre range. A main server, called the TrunkNode, communicates with the BranchNode units by GSM mobile phone or by using a GPRS system; it also hosts the web interface, which allows control and monitoring of the system. A system diagram is shown in figure 10.

Up to eight different dimming profiles can be handled by the system to suit different locations; this enables lamps to operate at reduced light levels during the night for example, when fewer cars are on the road. Should communication be lost with a LeafNode unit, it will continue switching and dimming the lamp using the same schedule as from the day before. As well as allowing centralised control using time profiles or a web based solar clock control, the lamps can also be controlled by a photocell installed in every BranchNode.

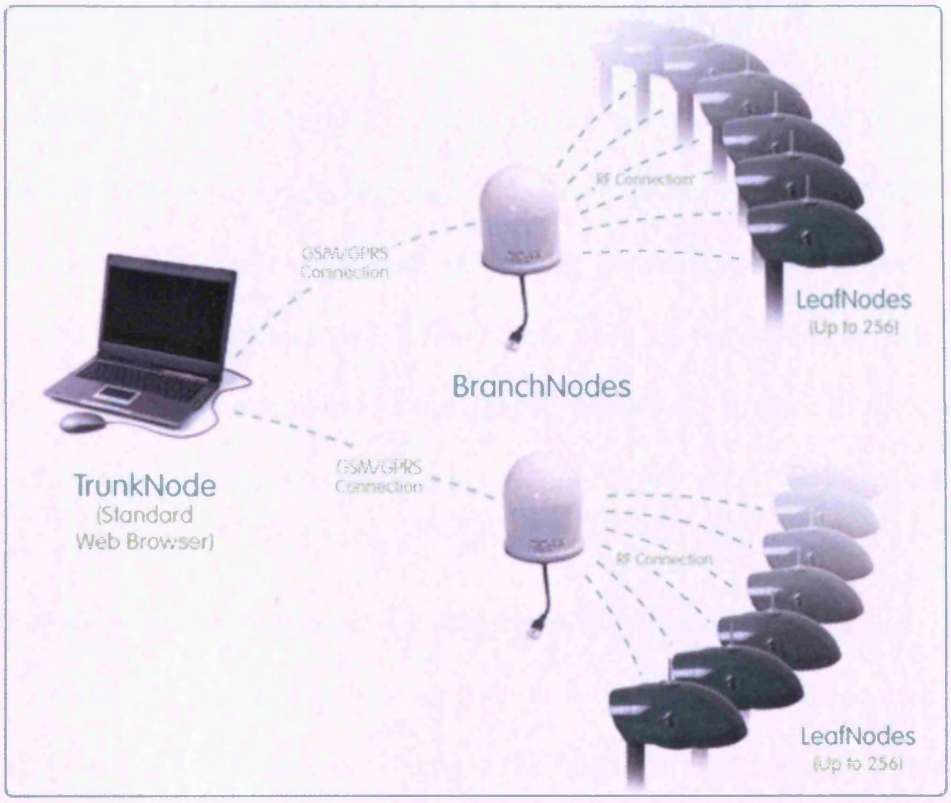


Figure 10:- LeafNut System Overview[32]

As well as controlling lamps, the LeafNut system also provides comprehensive monitoring capabilities; it can monitor each lighting unit for efficiency, predicted lamp life, ballast condition and energy consumption. The system can even stop lamps cycling when they reach end of life, by detecting the cycling and then running the lamp at 75% power until there is opportunity for it to be replaced.

The LeafNut system is believed, by the manufacturer, to be "the only monitoring control system that links directly to the micro-processor in the ballast and can therefore accurately predict lamp failure."[33]

2.2.5 Telensa Ltd

The PLANet™ (Public Lighting Active Network) system by Telensa Ltd is designed to enable the remote control, monitoring and metering of public lighting installations. As shown in figure 11, the system is made up of four key components; Outstations, Base Stations, the Central System and the User Interface. Each light is fitted with a radio device (Outstation) that controls and monitors the light it is attached to. Each of the outstations communicates with the nearest basestation via a Ultra-Narrow Band (UNB) radio link.

A Base Station consist of two parts, a controller module and a radio module. The radio module is mounted at the top of a lighting column or building, while the controller unit is housed in the bottom of a lighting column. The controller unit connects to the Central System via an ADSL telephone line or a GSM/GPRS connection if a phone line is not available. A light meter is mounted next to the radio module, which measures the ambient light at dusk and dawn and the controller then instructs the outstations to switch accordingly.

The Central System server communicates with each of the Base Stations, hosts and administers the system database and acts as a web server for the user interface. As the user interface is web based, there is no client software to install and the system can run securely using a standard web browser. Using the system, lamps can be programmed to switch or dim according to combinations of time and local light level. The system also provides remote monitoring of lamp failures, mains supply metering and lamp electrical parameters such as power factor.

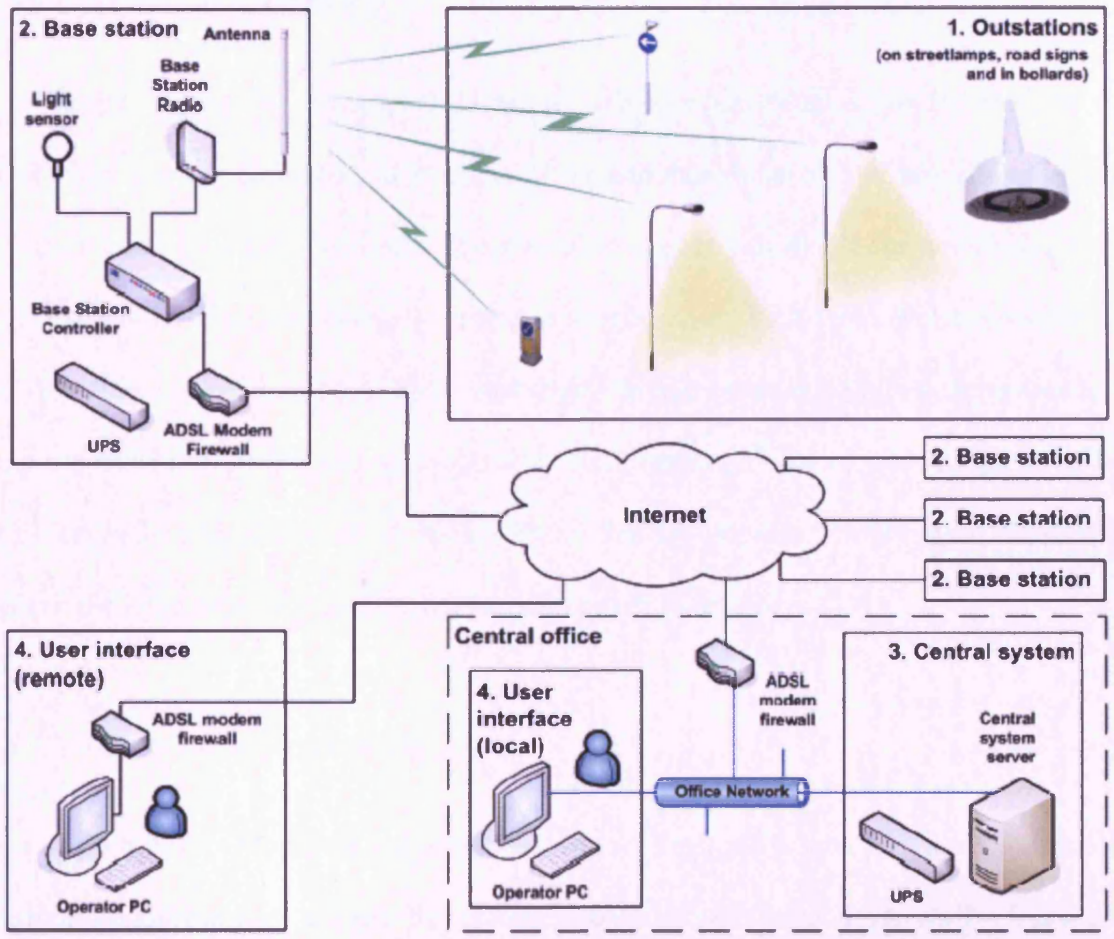


Figure 11:- PLANet^{TM} System Overview[34]

Although similar to the LeafNut system, the manufacturer argues that "It is the first solution of its kind that provides a full range of control, monitoring and metering at a price that makes adoption on all lights economically viable"[34]. This is primarily due to the systems ability to integrate with old lighting hardware via a plug in photocell replacement module. The consequence of this flexibility however is that, unlike LeafNut, the system cannot predict lamp failures.

2.2.6 Archnet Technology Ltd

The system offered by Archnet Technology Ltd[35] operates on a similar basis to the systems previously described, at the time of writing there is no sign of provision for lamp failure prediction, only detection. The system is currently available for monitoring street and campus lighting and features a device connected to each lamp which monitors its current draw to detect when it fails. Each lamp device communicates via the power lines to a power line modem which then conveys the information back to a central pc via either the internet or a GPRS (General Packet Radio Service) modem. Information can also be sent to the lamp module to switch the lamp on or off as desired.

2.2.7 Fortran Traffic Systems Ltd

The Naztec Lamp Monitoring System by Fortran Traffic Systems Ltd[36] uses precision current transformers to monitor the current drawn by each lamp in the traffic lights at a traffic intersection. By detecting the failure of any lamp quickly, a replacement can be arranged before an accident occurs. Although the system can detect an intermittently failing lamp, it appears to have no predictive failure abilities.

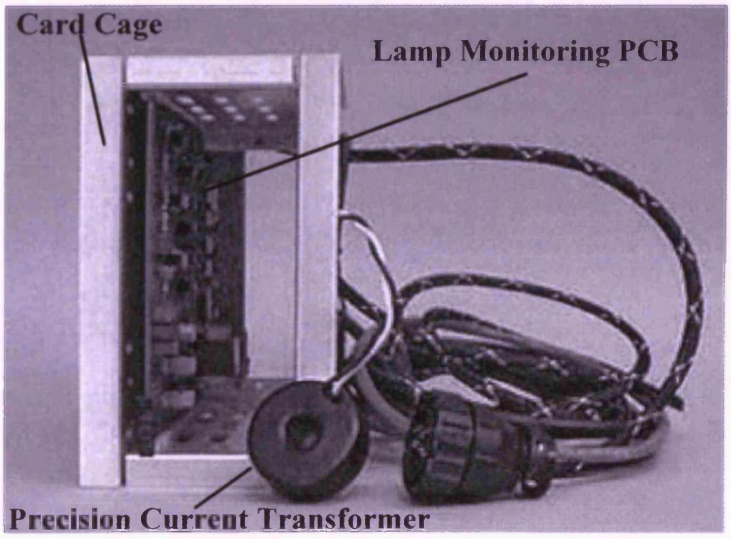


Figure 12 :- Naztec Lamp Monitoring System[36]

2.3 Summary

Following consideration by the author of the current state of monitoring of the different lamp types, together with looking at the systems which are currently available on the market, it is clear that considerable work has been undertaken; however most of this work has been looking at HID lamps and devices for incorporation into projectors to warn the user of impending lamp failure and to protect them from exploding lamps. Very little work has been done on fluorescent lamps and filament lamps. This is not perhaps surprising, as the products which use HID lamps are usually expensive, and hence can accommodate the extra financial burden of a monitoring system. However the full benefit of lamp monitoring has yet to be realised, as HID lamps only make up a small fraction of UK lamp sales, as shown in table 1.

Lamp type	Lamp sales (millions)	Percentage of total sales
Filament (GLS)	323	74.7
Fluorescent tubes	54	12.5
Halogen	26	6.0
Compact fluorescent	24	5.5
High intensity discharge	5.5	1.3

Table 1 : - Lamp sales by UK Lighting Industry Federation members in 2002[37]

It is clear that the work on monitoring HID lamps will have had very little impact on the overall lighting market, since they make up some 1.3% of total UK sales.

With this in mind, it is the authors opinion that further research is required on the condition monitoring of filament lamps (including halogen lamps), which make up 80.7% of the total UK market, and fluorescent lamps (including compact fluorescent lamps), which account

for 18%, so that accurate estimates of the remaining lamp life can be determined. Once this has been achieved, work needs to be done to see how the information gathered from individual lamps can be processed and used to form a collective picture, for the end user, of the condition of an entire lighting system, containing many different types of lamp.

3

Filament Lamps - Detecting Failure

3.1 Traditional Filament Lamp Testing and Monitoring Techniques

Predicting when a filament lamp will fail is a very difficult task. Two approaches can be taken; additional testing could be developed for when the lamp is manufactured, so that the rated life could be adjusted for each individual lamp; alternatively the lamp could be monitored while in use, this method having the added advantage of being able to take into account the conditions of use the lamp is exposed to, but it is likely to be more expensive due to the installation of hardware with every lamp.

Quality control at a filament lamp factory usually relies on sampled lamp testing. Lamps of the type being tested are put on a test rig and run until they burn out. BSEN 60064:1996 specifies that a minimum sample of 50 lamps is required.[38] The arithmetic mean of the times they burned for is then taken as the rated average life.[39] During mass production the quality of the lamps leaving the factory is checked by taking random samples from batches and seeing if on average they last as long as the rated burning hours of the lamp. Often, the lamps are run at abnormally high voltages to shorten their life and provide quick results, the expected life under normal conditions can then be extrapolated from this with reasonable accuracy.[40]

Batch sampling however does not prevent faulty lamps leaving the factory and being installed as, although the average life of the lamps will be tested, that does not mean that every lamp will last that long. The fact that it is an average implies that a substantial number of lamps will fail before then. This also means that scheduling lamp replacement based on the rated lamp life may well lead to lamps failing before they are replaced.

Another test procedure which can be performed after the lamp is manufactured is to burn the lamp in for a set period. This helps reduce the infant mortality rate of the lamps, but it does have problems. The time taken to perform a thorough burn-in is considerable in a production environment and burning the filament for long periods means it is weaker when the lamp has to be shipped out of the factory. It also reduces the lamp life by however long the burn-in period was. For these reasons factory burn-in is seldom done on regular filament lamps.[41]

Some installations make use of hour meters to record how long the lamp has been on for. However hour meters do not take account of whether the lamp has been dimmed or how many times it has been switched on and off, so the reading is inherently flawed. Even ignoring the effects of switching frequency and dimming, the approach is still inaccurate as the actual life of each lamp may vary greatly from the average life stated by the manufacturer due to manufacturing tolerances and possible defects introduced when the lamp was made. In the opinion of the author, a new approach is needed, so that lamp failure can be predicted and replacement can be undertaken before the lamp actually fails.

There are many reasons for why a lamp may fail during use; these are considered, together with possible detection methods, in the following sections.

3.2 Detecting Defective Lamp Construction

The construction of a GLS light bulb can be divided up into four key stages. Fabrication of the filament and support structures (which include the central glass column), fabrication and attachment of the glass bulb envelope, evacuation of the glass envelope followed by the introduction of the final filling gas and finally attachment of the electrical connections cap. The four stages shall now be considered in more detail, together with possible defects that may occur and ways in which they may be detected.

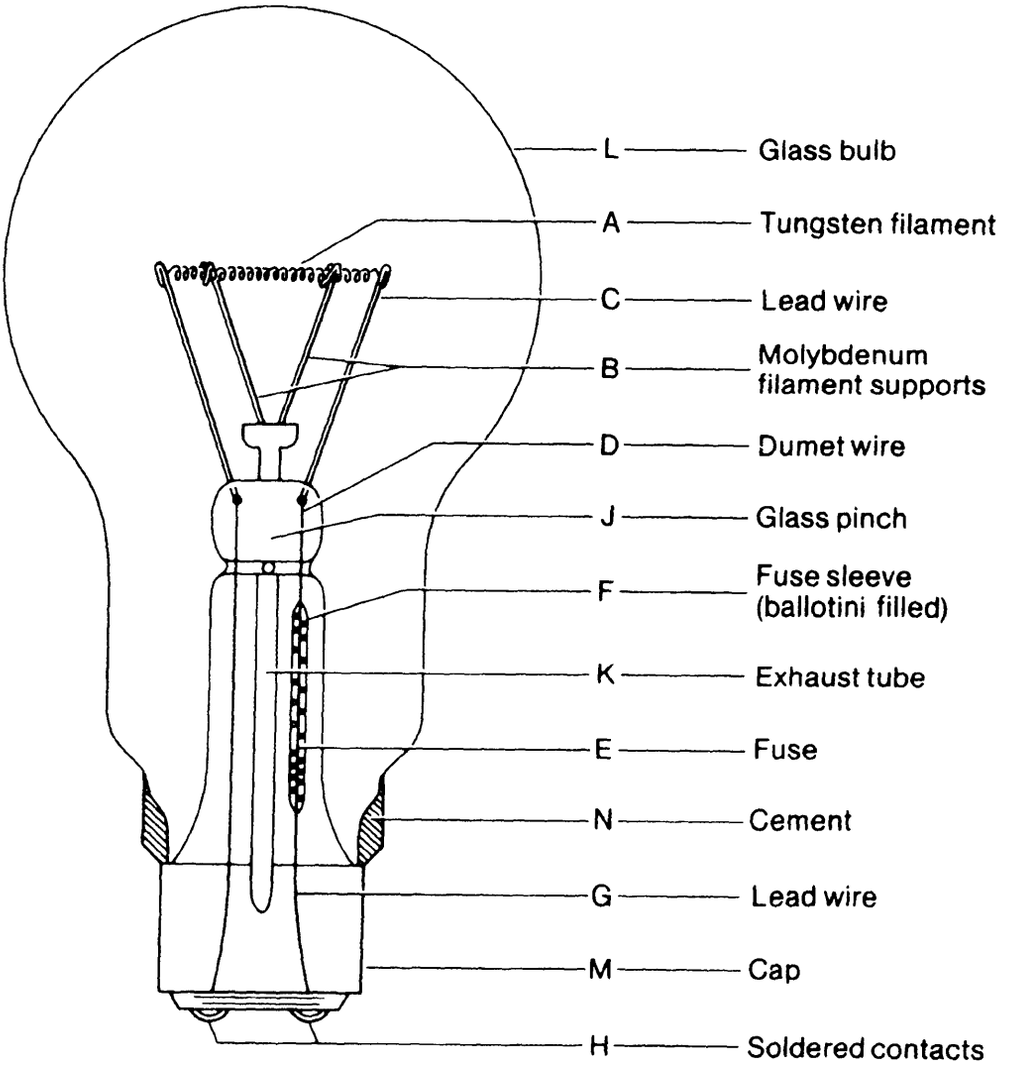


Figure 13:- Diagram of a typical GLS lamp.[42]

3.2.1 Filament and Support Structures

Filament Wire

The filament of virtually every lamp manufactured today is made of tungsten due to its relatively high melting point and low evaporation rate; it also features a higher emissivity in the visible wavelength range than in the infrared range, which for a standard light bulb is desirable. When tungsten was first used for making lamp filaments, it was pressed into shape; however this meant it was brittle and could not be bent afterwards. In 1909 William Coolidge developed a process for making ductile tungsten in wire form.[43]

"Tungsten used for filament manufacture is extracted from ores containing the minerals scheelite, $CaWO_4$, and wolframite $(FeMn)WO_4$. The ore concentrates are chemically treated to produce pure tungsten trioxide, WO_3, from which the metallic powder is obtained by a high temperature reduction with hydrogen."[44] Prior to reduction with hydrogen, small quantities of additives are blended with the tungsten trioxide, these dopants which are usually a combination of potassium silicate and aluminium oxide and are more commonly known as 'AKS' dopants. The dopants are added to promote elongated grain growth when the filament is first heated up above its recrystalization temperature, thus making the filament stronger and less likely to sag. The effect on the grain structure can be seen in Figure 14. "The powder is subsequently pressed into bars and the tungsten densified by sintering at temperatures approaching 3000°C. Further densification occurs during fabrication into rods by hot rolling and swaging operations, after which the tungsten is progressively drawn into wire of the required diameter."[44]

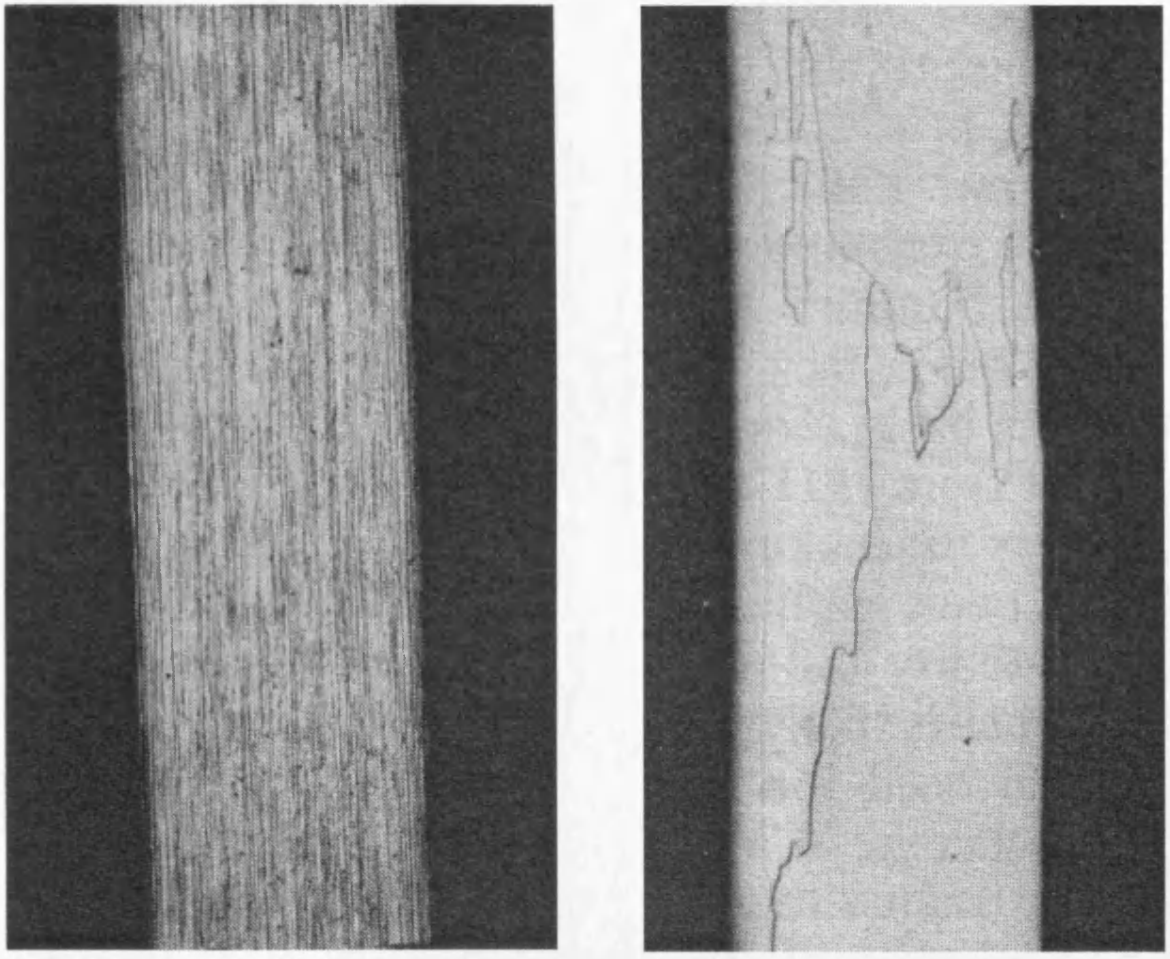


Figure 14: - Microstructure of tungsten wires: (left) after drawing, (right) re-crystallized wire doped with alkali silicate. X 200[45]

It is clear from the description of how the tunsten wire is produced that great care has to be taken to prevent impurities and flaws being introduced into the wire. A flaw in the filament wire such as a nick or stretch or a small localised contamination of an element such as carbon or iron, could lead to a localised hot spot in the filament which would dramatically shorten the lamp's life.

A fault such as this would be very difficult to detect from measuring the electrical characteristics, as the resistance change caused by the nick, stretch or localised contamination would be very small indeed. However a visual inspection whilst the

filament is glowing a dull red could be attempted to reveal any severe hot spots. This sort of testing would have to be undertaken on the production line, as the equipment required to do it automatically would be prohibitively complicated and too expensive to include in the vast majority of lamp monitoring systems.

A mistake during the addition of the doping additives could also leave the filament weak and prone to sagging. Once again this would be very difficult to detect with a monitoring system.

Filament Formation

The primary factors effecting the selection of the length and diameter of a filament wire are the desired operating voltage, current draw, light output, vibration resistance and lamp life. As the filament for mains voltage lamps usually needs to be quite long, the tungsten wire is wound on a mandrel to produce a coiled filament which takes up significantly less space than if the wire were left straight. Another advantage of coiling the filament is efficiency, as thermal loss from the filament is reduced and the overall efficiency of the lamp is greatly improved. Many lamps feature 'coiled coils', where the coiled tungsten wire is wrapped around another mandrel, this second coiling has the effect of reducing the overall filament length of a typical 40W GLS lamp by about 50%[7] while increasing its efficiency by around 20%[7] for the same life. A disadvantage of double coiled coil lamps is that they are more prone to shock and vibration damage than their single coil counterparts. Once the filament has been coiled, it is heat treated to remove internal stresses and to re-crystallize the tungsten wire. "Extreme accuracy in coil geometry and

dimensions is essential since the performance of a filament is very sensitive to parameters such as wire diameter, pitch and the coil length and diameter".[44]

Detecting flaws in filament formation after manufacture would be quite challenging. If the wrong diameter wire has been used, or the filament has been cut too short due to the automated manufacturing process, then it would be likely that the whole batch would be affected and it would be picked up quickly as a result of random sample testing. BSEN60064:1995 recommends a sample size of 100 lamps, and states that no more than 12 lamps may have a wattage in excess of 104% of the rated wattage plus 0.5W.[46] It also sets limits for the minimum allowable luminous flux.[47]

As an alternative to batch testing, a simple resistance test could be performed and the result compared to the average reading obtained from a number of good lamps. However, detecting errors in coil pitch and diameter which did not affect the overall wire length would be very difficult indeed. One way of detecting these errors may be to measure filament resistance while the lamp is powered; if thermal losses are increased due to the filament geometry then in theory the filament should be running cooler and its resistance should be lower. However the change in resistance is likely to be quite small and therefore very hard to detect.

Filament Mounting

The filament is usually mounted using a series of support wires in addition to the two electrical connection wires at either end. The support wires are made of Molybdenum, due

to its very high melting temperature (2623°C[48]). The electrical connections at either end of the filament also usually serve as the end support wires and are usually made of nickel or are nickel plated and are folded over the filament to clamp it. These wires are then welded onto short lengths of Dumet wire which pass through the glass pinch to the outside of the lamp. "Dumet is a composite material consisting of a central core of 42-58% nickel-iron alloy sheathed with copper which constitutes about 25% by weight of the complete wire. The surface of the wire is coated with sodium borate which prevents the formation of spongy cuprous oxide during sealing and also enables the wire to be more easily 'wetted' by the glass. As a result of its composite structure, the thermal expansion coefficient of the wire is different in the axial and radial directions. In the radial direction it matches that of the glass, whilst in the axial direction it is less than that of the glass. Consequently to avoid dangerous axial tensile stress in the glass the seal is not completely annealed, but cooled at a rate such that, at room temperature, it exhibits axial compressive stress. The stress is limited by keeping the wire diameter below 0.8mm."[49]

Occasionally, one of the supporting wires may break off and cause the filament to collapse. This problem was observed on one of the lamp samples during an initial experiment. However, apart from a visual inspection to ensure the support wires protrude well into the glass button, there is no way to test their security once the bulb is assembled as this fault will have no effect on the electrical characteristics of the lamp until it suddenly fails. The only exception to this is if the filament collapses back on itself. In this case the lamp becomes excessively bright and the current draw increases dramatically, which could be picked up relatively easily by a monitoring system. A machine vision system for

inspecting the assembly of light bulb filaments, such as that proposed by Wen C. Lin[50], is unlikely to be able to see how far the supporting wires protrude into the glass button.

The electrical connections to the ends of the filament are usually made by folding over the ends of the lead wires to clamp the filament. If this is not clamped with sufficient force the filament will arc at the connection point, overheat and fail, or just slip out altogether. If the clamping force is too high then the filament may be partially severed and will once again over heat and fail at this point. Detecting this sort of fault once the bulb has been manufactured would be very difficult indeed as it would have no discernable effect on the electrical characteristics if it is over crimped, and if it is under clamped then any arcing is likely to be highly intermittent. A visual inspection at low heat may reveal the problem as a localised hot spot, but this is not guaranteed. Arcing caused by under clamping would be detectable by a lamp monitoring system, as the lamp current would fluctuate significantly over time.

If the Dumet wire was not fused into the glass pinch seal correctly, then one of two problems may occur; either a slow gas leak may develop around the seal, or the glass pinch seal may shatter, causing a rapid loss of gas and immediate failure of the lamp. Virtually any fault with the glass pinch seal is not going to be detectable from the electrical characteristics of the lamp on the production line. A monitoring system based on the electrical characteristics of the lamp is also not going to be able to detect any cracking of the glass seal before the gas starts to escape. A slow leak however will allow small amounts of air into the lamp, which will cause the filament to slowly burn away, which should be detectable from monitoring the current draw of the lamp.

3.2.2 Glass Envelope

The glass envelope of the lamp is made by blowing molten glass into a rotating mould which has been sprayed with water. The resulting steam cushion gives a smooth shiny finish to the lamp, whilst the rotating mould ensures there are no seam marks. The outer envelope is usually made from soda-lime silicate glass, whilst the internal supports and pinch seal are usually made from lead-alkali silicate glass due to its higher electrical resistivity and lower softening temperature, meaning it has a longer working time. The two components are then assembled and fused together. Once the lamp has been evacuated and the final filling gas introduced, the exhaust tube is heated and sealed off.

Failures of the glass envelope caused by manufacturing defects are unusual but can occur. Small defects in the glass envelope such as scratches introduce stresses in the glass when it warms up and cools down, which can lead to the glass envelope cracking or even shattering. This type of failure is particularly serious, as if the lamp is suspended from a ceiling lamp holder then the glass bulb may fall down, leaving broken glass over the floor and two live wires sticking out of the lamp holder. Unfortunately defects like this are very difficult to detect at the factory as the defect is usually very small. Additionally, since it will have no effect on the electrical characteristics of the lamp until the glass envelope breaks, it would be virtually impossible to detect it with a monitoring system. Fortunately failures of this type are rare.

3.2.3 Gas Filling

Once the glass components of the lamp are assembled, the join is fused together and the only route left open to the outside world is via the evacuation tube at the bottom. Initially

a vacuum pump is attached to this pipe to evacuate the air from within the lamp, then the final filling gas is let back in to the lamp. For GLS lamps, argon is usually used as the filling gas, however krypton is often used for small torch bulbs. The filling gas serves to surpress physical processes such as evaporation and sputtering. In general, increasing the gas density decreases the filament surface temperature and increases the number of atomic collisions with atoms trying to leave the filament, which leads to a reduction in the number of atoms leaving the filament. There is a consequence however, as the thermal conduction of the argon gas increases the loss of filament heat, which reduces the overall efficiency of the lamp. Nitrogen, which has good arc-quenching abilities, is often added to the gas mix to prevent arcing between metal structures at different potentials within the lamp. Due to the high operating temperatures of the components within a filament lamp, oxidation and carburisation processes (which would take weeks at room temperature) can occur very quickly. To avoid this, gasses such as "oxygen, carbon monoxide and dioxide, hydrocarbons and water vapour" must be excluded from the lamp[49], which is often achieved through the use of 'getters' after the lamp has been sealed off. 'Getters' are materials which are "used to remove gaseous impurities retained after the lamp bulbs or tubes have been sealed."[51] Red phosphorous is commonly used in filament lamps to remove any traces of oxygen or water vapour after the lamp is sealed off; it is usually applied to the filament, so that when the filament is powered for the first time it is activated and generates phosphorus vapour.

If air is left inside the glass envelope or the glass evacuation point is not correctly sealed allowing air to leak in, then the life of the filament will be dramatically reduced. Typically any leak will quickly lead to the gaseous contents of the lamp being replaced completely

with air and under these circumstances the filament will burn out within a few seconds, which could be detected by any short test cycle. If some air is left inside when the lamp is made however, then depending on how much is inside, the lamp may well survive the initial filament burn as the oxygen inside the lamp is used up by the getter and the burning filament. However this will mean that as the getter has not been fully effective in removing all the oxygen, the filament will be weakened and will probably fail prematurely when the lamp is put into use. In theory this should be detectable by measuring the change in resistance of the filament when the lamp is powered for the very first time.

3.2.4 Electrical Connections

The lamp cap is fitted once the lamp has been gassed and the evacuation tube sealed off. As well as providing a convenient means of connecting power to the lamp, the cap also serves to protect the rather fragile evacuation tube from damage. The cap is made from either brass or aluminium and is cemented onto the bottom of the lamp. GLS lamps usually come with either a bayonett cap or Edison screw cap. The bayonett cap has two (sometimes three) pins which stick out of the side of the cap to hold the lamp in the lampholder. On the bottom of the cap there is an opaque glass disc upon which two brass contacts are mounted. The wires from the lamp are fed through the holes in the glass disc and the contacts are then covered in solder. A similar process happens with Edison screw lamp caps, except there is only one terminal on the bottom as the cap itself forms the other terminal.

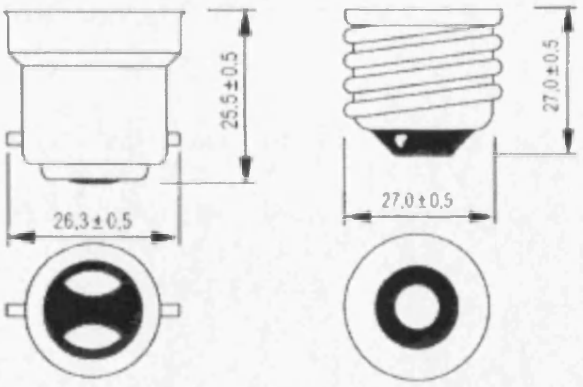


Figure 15: - Bayonet cap (left) Edison screw cap (right) [52]

A fuse is usually fitted in the wires which go from the lamp cap to the bulb itself, this may take the form of two lengths of thin wire or a single Ballotini fuse in one of the wires. A Ballotini fuse is a thin piece of wire housed in a thin glass tube which is filled with tiny arc-quenching glass balls before the two ends are sealed. The fuse is designed to melt if the filament collapses on itself and causes a short, or if a plasma arc occurs when the filament breaks.

Cracking of the glass insulator in the lamp base, the base becoming detached, pins falling out or a break in the wiring inside the cap may stop the lamp working; these are not very common causes of failure, and whilst the wiring would be tested by a short test cycle, cracking of the lamp base and the base falling off will depend a lot on what lamp holder is used and how the lamp is handled, so can not be easily tested. As contact resistance is likely to increase if the base becomes broken or deformed, a monitoring system may detect this problem before the lamp fails as an increase in measured lamp resistance or as a fluctuating lamp current reading if arcing occurs as a result of the damaged contacts or an intermittent wire joint.

3.3 Detecting Environmental Failure Causes

Clearly, external causes of failure cannot be taken into account when the lamp is tested at the factory, as they will not be known. The only way to take account of their effect on lamp life is through a suitable monitoring system.

3.3.1 Over Voltage

Over voltage will cause a higher filament temperature which will cause the light output to increase and the filament to evaporate quicker, shortening the life of the lamp. Extreme over voltage will cause the filament to evaporate immediately. Detecting over voltage should be an easy task for any monitoring system that monitors the lamp supply voltage. However estimating its effect on the life of the lamp would require comparison with data on how the lamp life is affected with voltage for that lamp.

3.3.2 Vibration

Vibration will reduce the life of most lamps, its main effect is to cause metal fatigue of the filament, especially at the points where it is in contact with support wires or lead wires. The filament will then eventually fracture at one of these points, either as a result of further vibration or, alternatively, filament movement when it is next turned on from cold. The point of fatigue may also partially fracture, leading to a hot spot forming and inevitable failure. Lamps which are designed for use in high vibration environments usually have one or many of a number of features: a thicker filament, a single coil filament design, extra supports, a cooler filament design temperature. Detecting vibration in a monitoring system would be very difficult as it is unlikely to have a discernable effect on the electrical

characteristics of the lamp. The use of a separate vibration sensor could provide a rather costly solution.

3.3.3 Temperature

A high ambient temperature will reduce the thermal loss from the filament through the glass wall, which will lead to the filament running hotter than normal which would in turn lead to an increased rate of evaporation; this should be detectable from the increasing filament resistance. If the ambient temperature is much too high, then the glass wall may soften, causing a change in gas pressure within the lamp or even a hole to form. A change in gas pressure may be detectable from the rate of filament evaporation. However the effect is likely to be very small and a hole suddenly forming would be very difficult to anticipate.

3.3.4 Thermal Shock

If water comes into contact with the glass wall of a hot lamp, the glass will usually shatter, allowing air in. Clearly this sort of failure would not be predictable with any monitoring system.

3.4 Predicting Failure Due to Usage

All filament lamps will eventually fail with use, which is currently an unavoidable consequence of using a filament to produce light. The cause of usage-related failure is almost always due to breakage of the filament as a result of excessive thinning, which is

evident from the fact that halogen lamps consistently last longer as a result of their reduced filament evaporation.[54] The rate of filament evaporation is strongly linked to the filament temperature. If an area of the filament is hotter than the rest, then that area will thin quicker, which will increase its resistance and cause it to become hotter still. This run-away situation is called a hot-spot and will eventually cause the filament to fail at that point. "The in-homogeneities usually considered as causes of initial spots are constrictions in the wire and variations in electrical resistivity, in emissivity or in filament geometry."[53] Filament supports and connecting wires conduct heat away from the filament, and consequently cause the areas of the filament in contact with them to run cooler, which can also be a cause of hotspots forming in the middle of unsupported regions of the filament. As well as hotspots, the filament may also fail as a result of overall thinning causing it to be become too weak to withstand small environmental vibrations or the sudden filament movement caused at switch on by thermal expansion and the magnetic field generated by the high inrush current.

By whichever method it happens, failure due to usage is almost always caused by filament thinning, either along the entire length of the filament or at a particular point on the filament. "Most lamp failures are of the 'simple' type, in which the filament simply burns through and then arcs at its weakest point."[55] Any thinning of the filament should lead to an increase in the resistance of the lamp which, although small, should be measurable. Monitoring the resistance of the lamp as it ages should in theory mean it is possible to track the thinning of the filament and therefore get a more accurate prediction of when the lamp needs replacing.

Table 2 shows a summary of the different possible causes of failure identified in this chapter, together with possible detection methods.

Having looked at the ways in which a lamp can fail, it is apparent that although some of the possible causes may not be detectable (such as some of the possible manufacturing defects or some environmental factors), the most frequent cause of failure (i.e. filament thinning) should be able to be monitored by measuring the filament resistance as the lamp ages. The following chapters discuss the experiments which were carried out in order to test this theory and work towards developing improved filament lamp monitoring techniques.

Cause of Failure	Possible Detection Methods
Filament flaws A nick, stretch or small localised contamination	• Visual detection by looking for hot spots whilst glowing dull red. This would have to be done on the production line as the equipment to do it automatically would be too expensive to use in monitoring systems.
Filament geometry Wire diameter, pitch, coil length and diameter	• Simple cold resistance check for incorrect length or gauge wire. • Hot resistance check to reveal if the filament is running cooler as a result of increased thermal loss due to poor coil geometry, the difference is likely to be small and therefore hard to detect.
Filament mounting Molybdenum support wires	• Visual inspection at factory of support wires. • Monitoring for sudden changes in filament resistance during use if filament collapses on itself.
Filament connections	• Monitoring current draw for rapid fluctuations that are indicative of an arcing connection. • Visual detection by looking for hot spots at the connections whilst filament is glowing dull red. This would have to be done on the production line as the equipment to do it automatically would be too expensive to use in monitoring systems.
Glass pinch seal & Glass envelope failures	• Monitoring filament resistance for increases caused by the filament burning as a result of a slow leak. • Impossible to predict a rapid leak caused by cracking.
Gas filling errors	• Monitoring filament resistance closely when the lamp is powered for the first time. The resistance change during first use would indicate how much the filament had burned as a result of air being left inside.
Electrical cap connections	• Check resistance of lamp to confirm continuity. • Monitor current draw for rapid fluctuations that are indicative of an arcing connection. • Cracking of the glass insulator, pins falling out or the cap becoming detached may show up as a bad connection when monitoring the lamp current, but it is more likely this would be undetectable until the point of failure.
Over voltage	• Mains voltage monitoring should make this detectable.
Vibration	• A vibration sensor could be positioned near the lamp, but this would be costly and impractical for most systems.
High ambient temperature	• Monitoring of the ambient temperature would make this detectable.
Thermal shock e.g. Water hitting lamp	• Completely unpredictable.
Usage	• Monitor hot filament resistance for signs of age related thinning leading to an increase in resistance.

Table 2: - Methods for predicting filament lamp failure

4

Filament Lamp Test Rig

One of the most common filament lamps currently in use is the GLS filament lamp, which can be found in virtually every home in Britain. Its popularity is mainly due to its low purchase cost and small size. For this reason it was decided to conduct the filament lamp monitoring experiments on domestic GLS lamps. GLS filament lamps typically have an average life expectancy of 1000 hours[56], which meant that manually monitoring during the experiment would have been a very difficult and time consuming task. What was needed was an automated test rig which would allow regular measurements to be taken and stored on computer. It also became apparent that in order to get enough samples tested in a reasonable time frame, the test rig would have to be able to test multiple lamps at the same time; it would also have to be highly configurable so that it could be adapted for use with other experiments that may follow. The cost of the test rig was not a major concern as only one had to be constructed. However when designing a monitoring system to deploy in the real world it would clearly be of critical importance to keep the cost to a minimum.

4.1 Hardware Design

4.1.1 Current and Voltage Monitoring

It was anticipated that as the filament aged there would be a gradual increase in resistance due to the filament slowly evaporating and becoming thinner. This in theory should cause a decrease in the current drawn by the lamp which would mean less power is being consumed by the lamp; which should also therefore lead to a reduction in overall lamp temperature and light output. It was desirable therefore to have the ability to monitor lamp

current, temperature and light ouput as well as the mains supply voltage, so that the effects of mains voltage fluctuation on the readings could be accounted for.

As the decrease in filament current was likely to be very small, the instrument measuring it needed to be sensitive; it also needed to have a computer interface so that the readings from it could be stored by the computer. The Voltech PM100 Power Analyser (Figure 16) was chosen to fulfil this role; it has a 0.1% basic accuracy and can measure voltage, current, power and power factor, as well as many other measurements. It also has an optional RS232 interface which allows full control of the instrument. Although at this stage it was unnessecary to be able to measure power factor, it was felt that this may be a useful feature when looking at other lamp types such as discharge lamps.

Figure 16:- Voltech PM100 Power Analyser[57]

Due to the power analyser costing over one thousand pounds it would not have been economical to buy one for each lamp that was on test. Instead a method of switching the power analyser between the lamps was required, which would not cause an interruption to the power going to the lamp. This was acheived by using two relays per lamp channel, one

to feed power to the lamp direct from the mains supply, the other to feed power to the lamp via the power analyser. Figure 17 shows the wiring diagram for this.

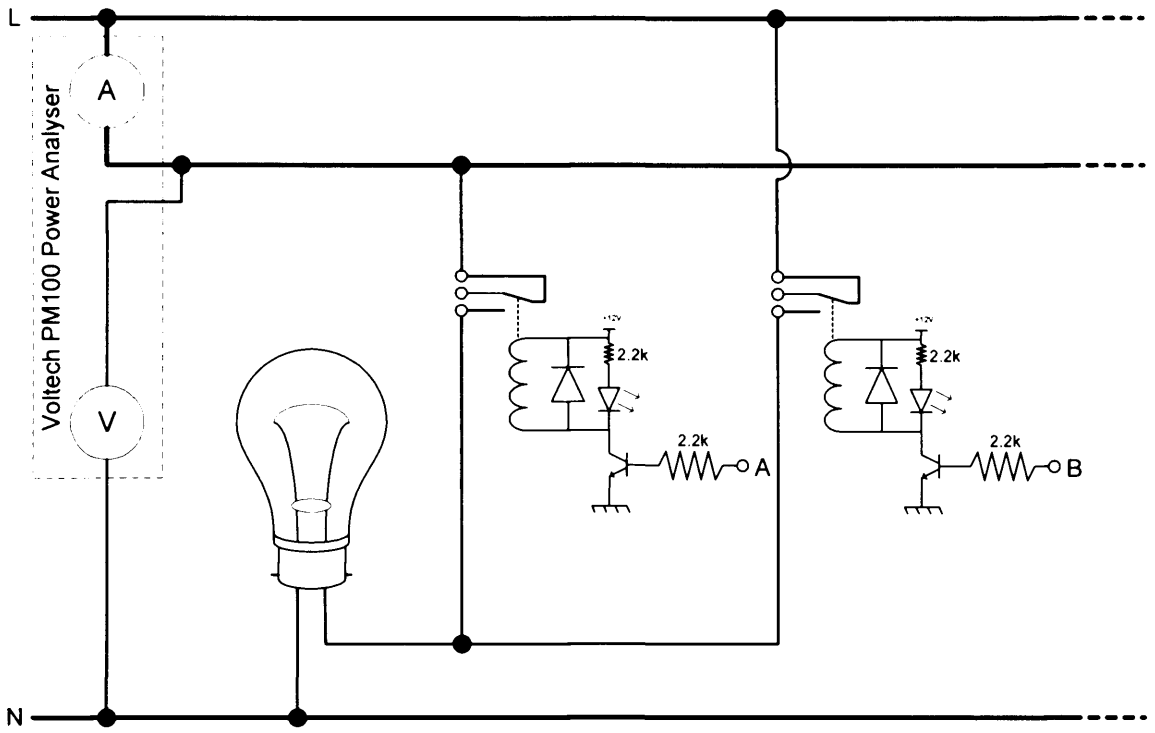


Figure 17:- Wiring Diagram for the power Analyser and one lamp channel

Looking at figure 17, it can be seen that closing either relay will turn the lamp on; however in order to get a current reading for one individual lamp, relay A must be closed and relay B must be open; the A relays on all the other lamp channels must also be open, otherwise an incorrect reading will occur. The normal mode of operation when all the lamps are on is as follows. Relay B is already closed for all the lamps, keeping them powered; relay A is then closed for the channel being measured, connecting the lamp to the power analyser, relay B for that channel then opens so the only source of power to the lamp is through the power analyser. The power reading is then obtained and recorded by the computer. Relay B is then closed to bypass the power analyser before relay A is opened to disconnect the

lamp from the power analyser; the power analyser is then available for use with another channel. Each relay was controlled via a BC182L transistor with a diode fitted across the relay coil to prevent transient voltage damage to the transistor. An LED and current limiting resistor were also fitted across each relay coil to show which relays were on and hence help diagnose any problems. The value of the current limiting resistor connected to the base of the transistor was selected so that the relay could be controlled from a 5V input. Figure 18 shows the completed relay interface board, which was designed and manufactured by the author for this work. It is housed in a plastic enclosure for safety reasons. Further information, including the PCB layout, can be found in appendix A.

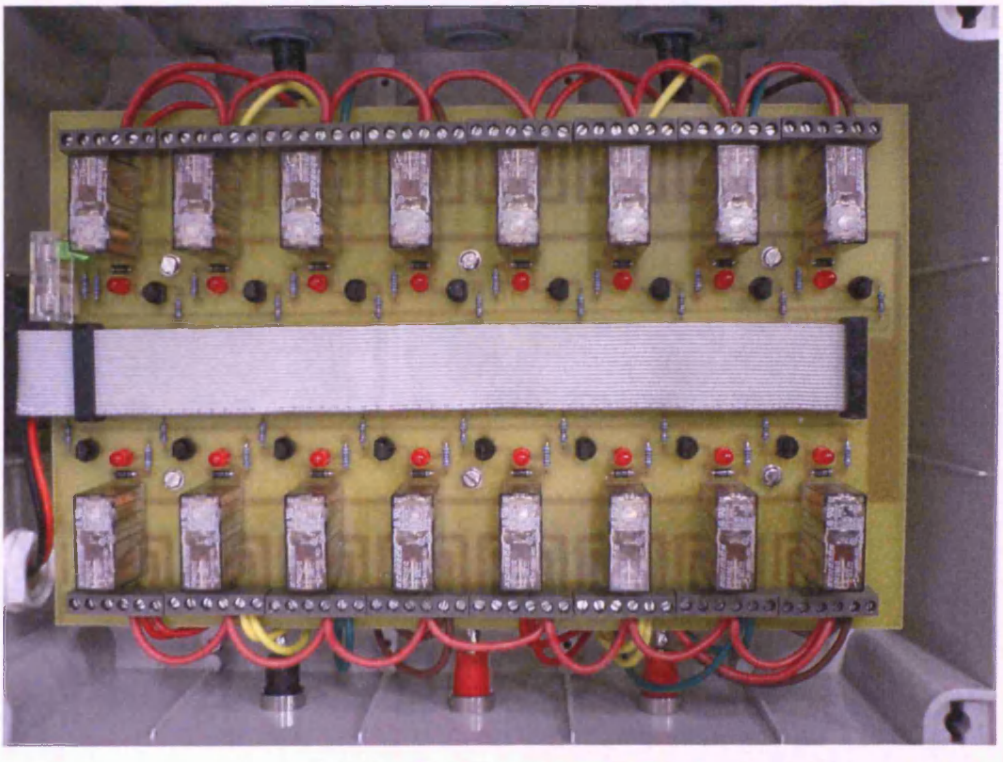


Figure 18:- Photograph of the completed relay interface board

Since the computer needs to be able to control the switching of the relays in order to switch lamps on and off according to the test cycle and to take readings using the power analyser,

a way of connecting the relays to the computer was required. The standard parallel port has eight data pins and four output control pins which could all be used as outputs to control the relays. However this would only allow the operation of six lamps at a time. An alternative solution was adopted which used an external USB interface device called a "LabJack U12", as shown in figure 19.

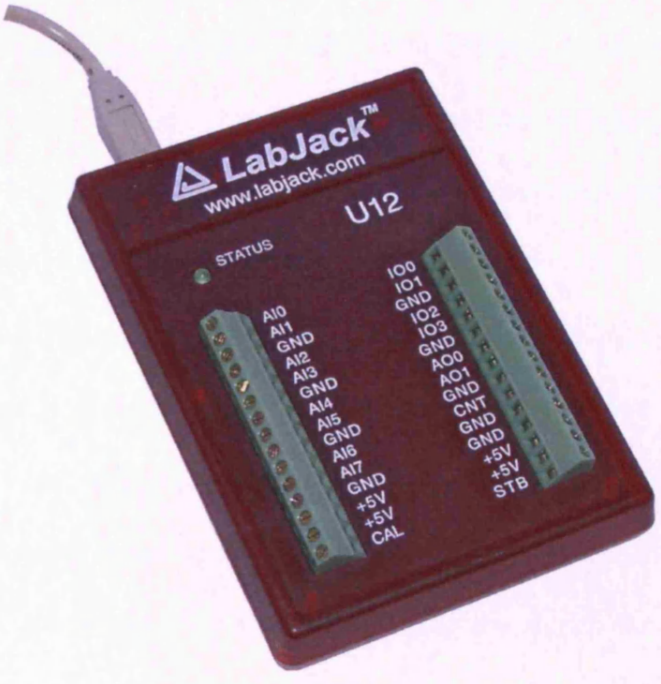


Figure 19:- LabJack U12 USB Data Acquisition Device[58]

The Labjack U12, a product manufactured by the LabJack Corporation, is a USB Data Acquisition Device with eight 12 bit analogue inputs, 2 analogue outputs and 20 digital I/O lines. It costs under one hundred pounds and comes with a well documented Dynamic Link Library (DLL) which can be loaded and used in most computer programming languages. As well as providing many more I/O lines than the parallel port, the analogue inputs also provided a convenient interface for measuring the relative light output and temperature of each lamp.

4.1.2 Light Level Monitoring

In order to measure the light output from each lamp, a light sensor for each lamp was required as the sensor could not easily be moved between the lamps automatically. Calibrated light sensors tend to be very expensive; however the sensor did not need to be calibrated or accurate as it was trying to record a trend in the light level rather than the actual light level itself, so as long as the reading could be compared with the one taken when the lamp was first used, then that would be sufficient. A suitable light sensor was sourced, the IS474 made by Sharp, which had an integrated current amplifier to provide an output current signal which is linearly proportional to the light incident on the sensor. To convert this current signal to a voltage suitable for connecting to the Labjack analogue interface should have just required a simple resistor connected between the output and ground; however the impedance of the labjack analogue inputs was low enough that it would have a significant effect on the level of the signal, so the signal was first passed through a high impedance operational amplifier (op-amp) to ensure that the signal would not be inadvertantly attenuated. As well as buffering the signal, the amplifier was also used to increase its magnitude by a factor of 10, so that the signal would use the full voltage range of the LabJack analogue input and hence improve the resolution of the quantised signal. The circuit for this is shown in Figure 20.

The op-amp chosen was the LM324N quad op-amp due to its high input impedance and its ability to run off a single polarity voltage supply whilst still maintaining an output swing capable of reaching zero volts.

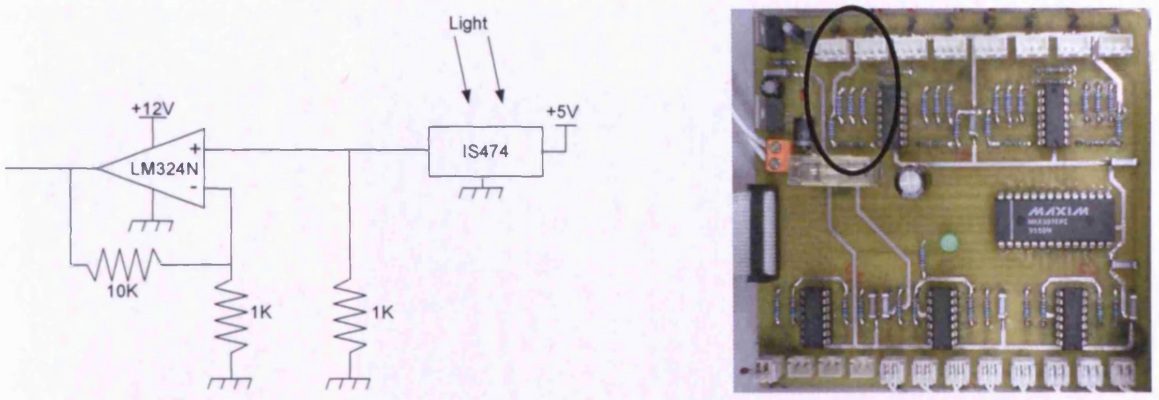


Figure 20:- Circuit for Measuring Lamp Light Output for One Lamp Channel

As well as measuring the light given off by each lamp, the sensor will also pick up background illumination; the effects of this can be minimised by mounting the sensor as close to the lamp as possible or by shielding the sensor so that it only picks up the light coming from the lamp. Since the glass envelopes of filament lamps become very hot during use, and the IS474 sensor is made of plastic, it was not practical to mount the sensor close to the lamp. Instead, the lamps were housed in closed bottom metal cylinders and a glass rod was placed adjacent to the lamp and passed through a small hole made in the bottom of the cylinder so it sat on top of the sensor and acted as a light guide. When the system was tested the light readings were very low, it was found that the readings were higher if the glass rods were omitted and the light allowed to shine directly on the sensor positioned beneath the hole. Consequently, all the glass rods were removed. Figure 21 shows the lamp housing arrangement and positioning of the light sensor.

Figure 21:- Lamp Housing with Light Sensor Circled

Clearly the positioning of the sensor and the shield around the lamp will effect the light level reading obtained, however as long as the sensor is fixed then this factor will remain constant and hence it is not dependent upon absolute values, as the reading is only being compared to the one taken when the lamp was first put in.

4.1.3 Temperature Monitoring

When measuring the temperature of a lamp, there are many different measurements that could be used. The most direct way would be to record the filament temperature; however doing this automatically would be expensive as it would require the use of a device such as

a pyrometer or infra red temperature sensor due to the filament being housed in a vacuum, an approach which would be unlikely to ever be compatible with the ultimate aim of providing low cost monitoring. If the filament temperature cannot be monitored directly, then the next best approach is to measure the temperature of a surface which is being heated by the filament, such as the glass envelope of the lamp or the metal cylinder the lamp is being tested in. It was decided to monitor the temperature of the surrounding metal cylinder as attaching the sensor to the glass envelope of the lamp would have been difficult and it would have had to be re-attached every time a lamp was changed. As the readings are being compared to those attained when the lamp was installed, as long as the location of where the measurement is taken remains constant there should be no experimental errors. Factors which may affect the measurements are the ambient temperature and any air movements; as the test rig is located in a closed room any unexpected air movements would be minimal and the ambient room temperature was recorded with the results using another sensor so that it could be taken into account when considering the results.

Since the temperature sensors needed to be mounted on the metal cylinders surrounding the lamps, one temperature sensor was required for each lamp, plus an additional one to measure the ambient room temperature. Due to the ease of interfacing with the LabJack and their low purchase cost it was decided to use NTC thermistors to record the temperatures of the lamps. As discussed when interfacing the light sensors, the analogue interface input impedance of the LabJack U12 was too low to connect it directly to the output of the thermistor when configured as part of a potential divider network. Instead, the signal had to be fed through a unity gain op amp circuit to ensure that the input impedance of the Labjack did not affect the reading. The circuit can be seen in figure 22.

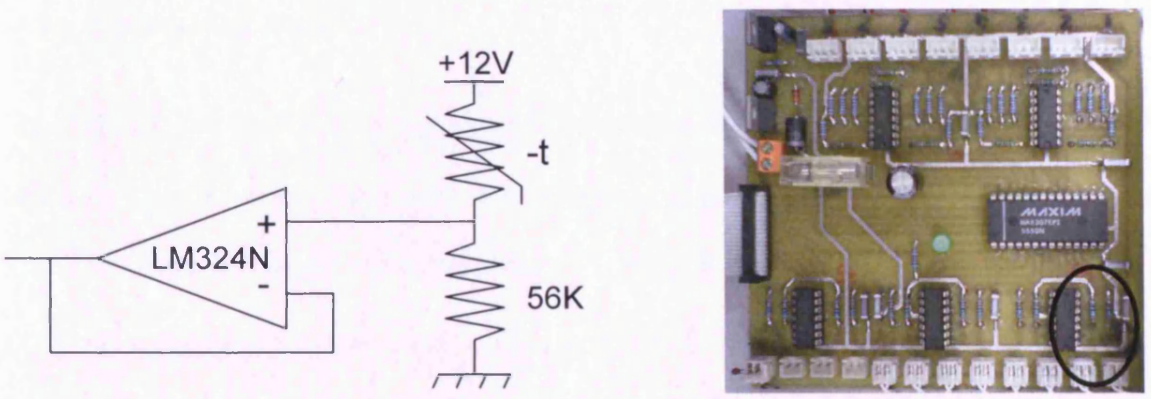


Figure 22:- Circuit for Measuring Temperature for One Lamp Channel

Once the voltage value has been recorded, it is simply a case of using ohms law to calculate the resistance of the thermistor; using this, the current temperature of the thermistor can be read from its characteristic curve, which is shown in figure 23.

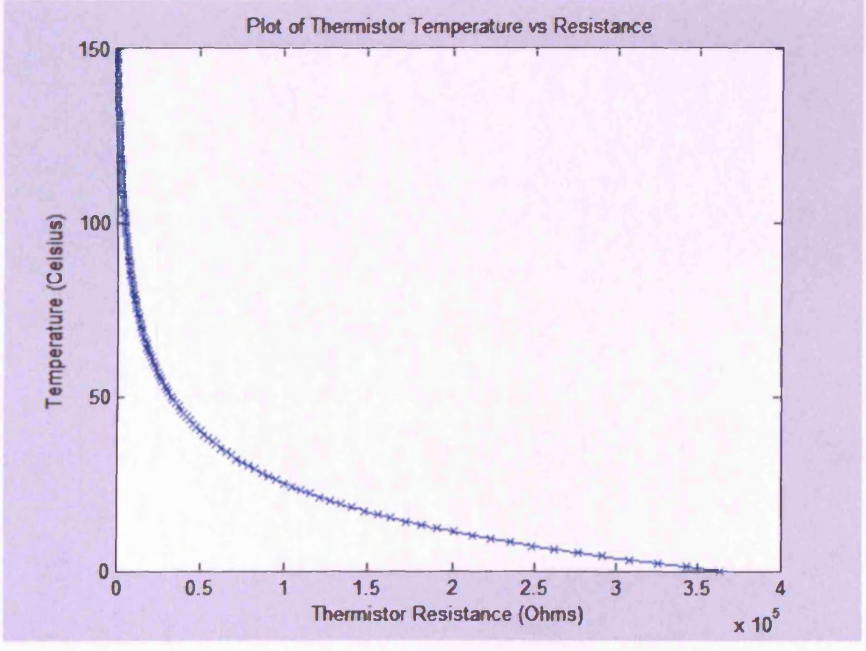


Figure 23:- Characteristic temperature curve for the thermistor used

4.1.4 System Integration

Since the Labjack U12 has 16 digital IO lines on the DB25 connector (the remaining four digital IO lines are on a different connector), using just these allowed for the control of 16 relays and hence 8 lamps. However there are only 8 analogue inputs available, which presented a problem as each lamp needed two analogue inputs, one for the temperature sensor and one for the light level sensor, making a total requirement of 16 analogue inputs. As only one channel is measured at a time due to the power analyser being shared by all the channels, it was decided to do the same with the analogue inputs. A Maxim MAX307 dual channel 8 into 1 analogue multiplexer, which was controlled using the four digital IO lines which had not been used on the front of the Labjack U12, was used to switch between the sensors from each of the lamps. The circuit diagram of the finished design is shown in figure 24.

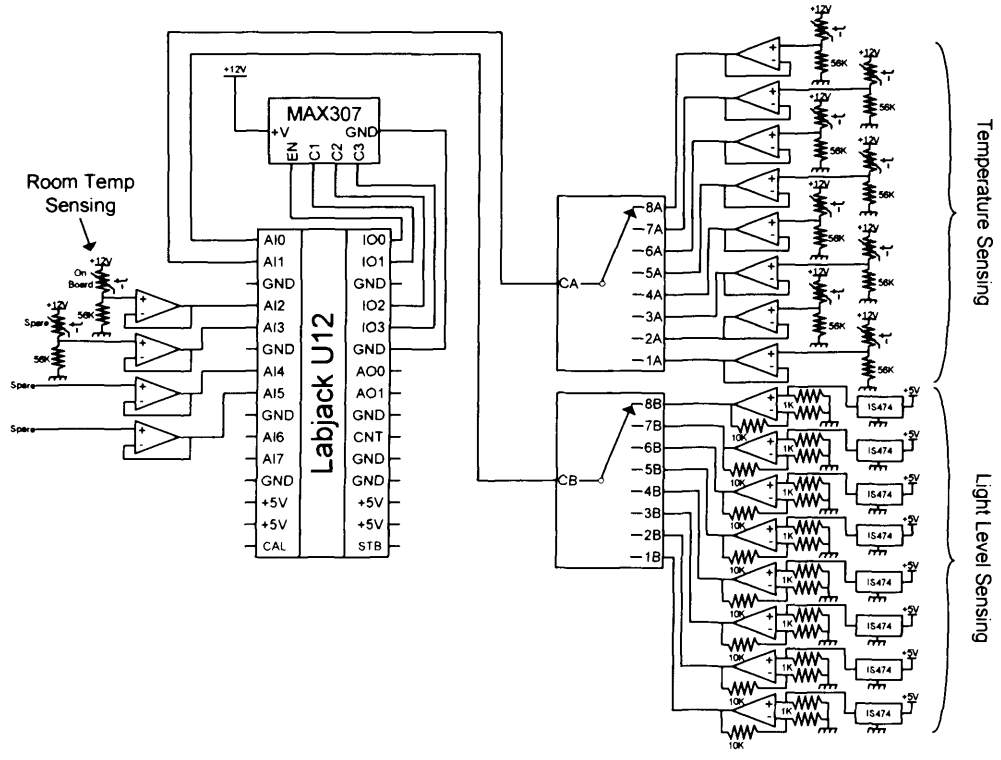


Figure 24:- Complete circuit diagram of the Temperature and Light Sensing.

A photograph of the final circuit board, designed and manufactured by the author for this research, is shown in figure 25. Further construction information including the board layout can be found in appendix A.

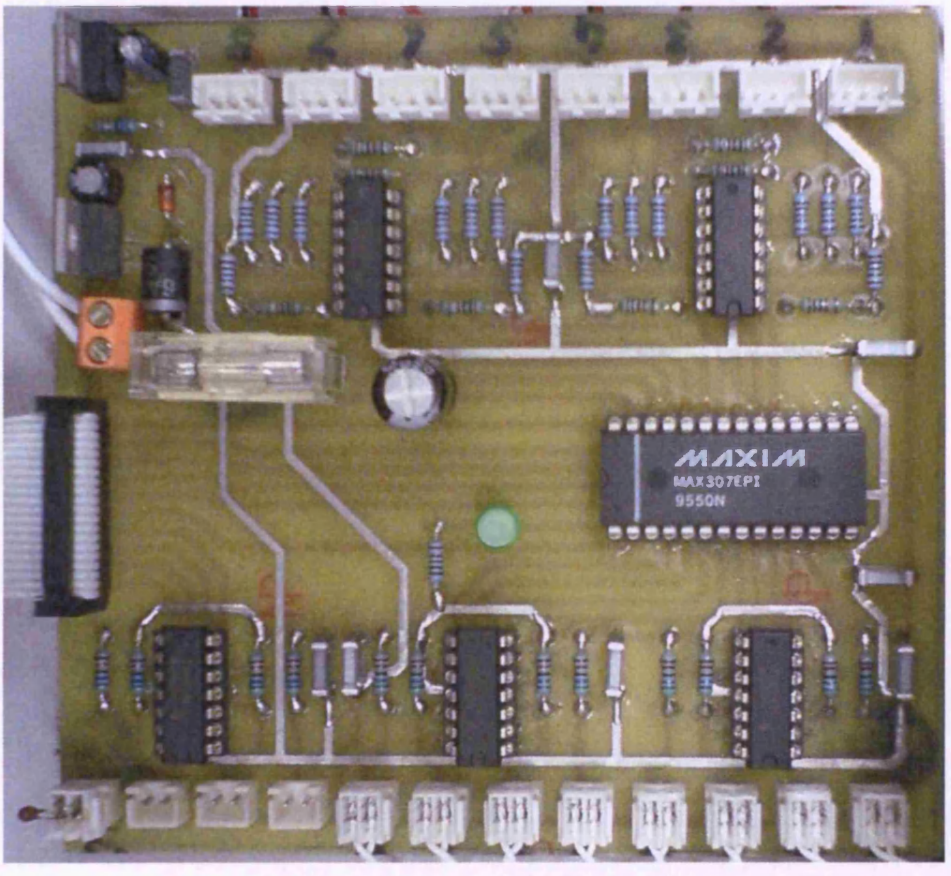


Figure 25:- Photograph of the completed Sensor board

Using this system, any one of the 8 lamps can be turned on or off by the computer and measurements can be taken for the supply voltage, lamp current, indirect lamp temperature, room temperature and light output for each lamp. An overall diagram of the entire system hardware described in this chapter, showing how it interfaces to the computer, is shown in figure 26. A photograph of the completed test rig is shown in figure 27.

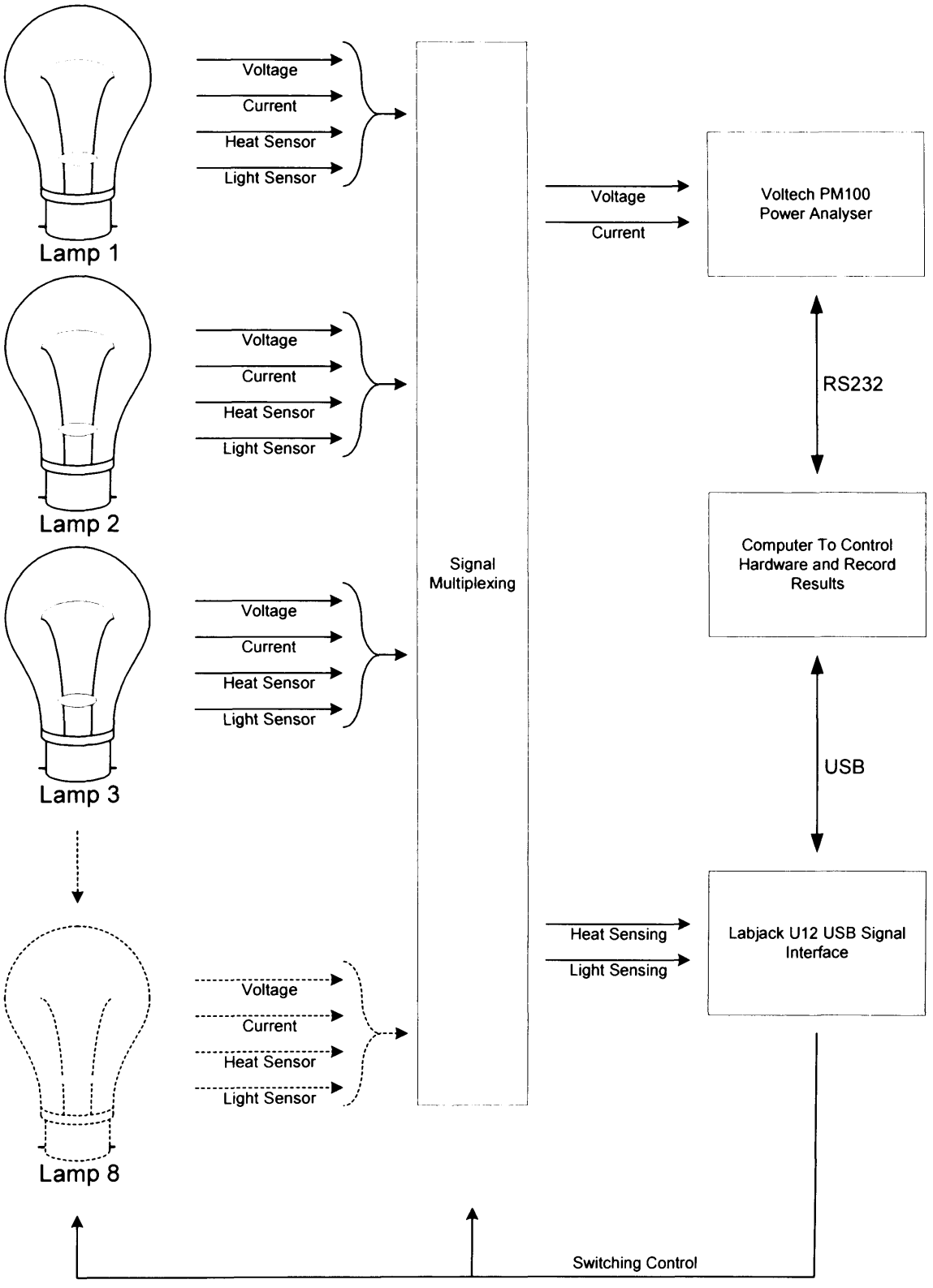


Figure 26:- Block diagram of the complete data capture system hardware.

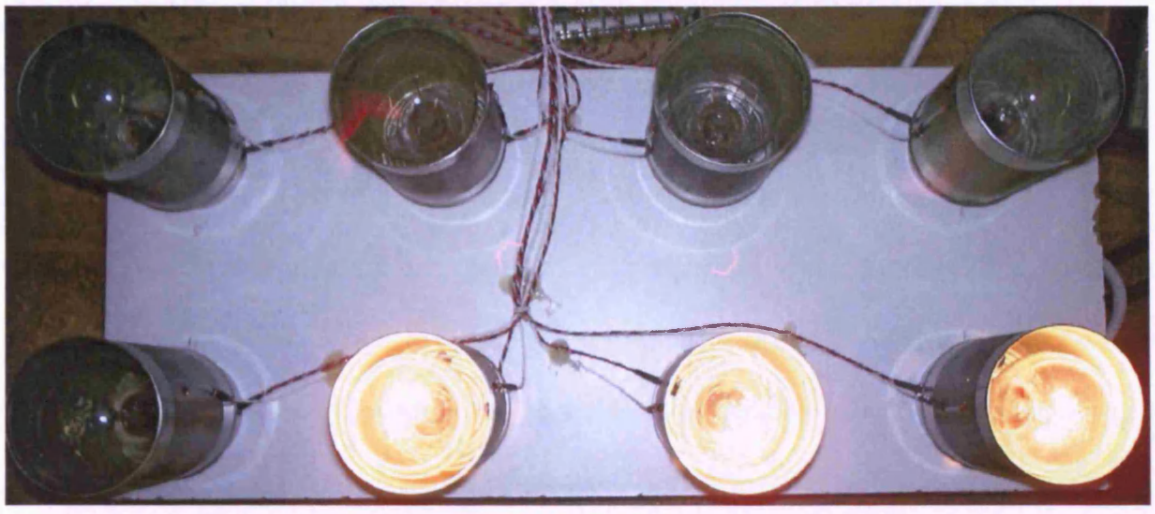


Figure 27:- Photograph of the completed test rig.

4.2 Test Rig Software Design

In order to enable any required data manipulation, it was decided to load the data gathered by the test rig into Microsoft Excel; however since the test rig would be running continuously for long periods of time, it would be inadvisable to store the data directly into Excel or any other large application as a single crash of the application would ruin the experiment. A better solution appeared to be to have two applications, one small application which stored all the data directly to the hard drive and another application which could be run at any time to extract the recorded data and load it into Excel or another application in order to perform operations on the data. Due to the specialised nature of the hardware it was decided that custom written software would be the best way of achieving the flexibility required; C++ was chosen as the programming language due to its support of object orientated programming and the author having had a lot of previous experience in using it.

4.2.1 Data Recording

To allow for more structured programming, C++ incorporates the use of classes and objects. An object is created from a class and can be considered as a grouping of data types and operations which work together. For example, when developing software to control a car, the developer could model the engine as an object. Within that object would be data types, called member variables, such as double numbers for current engine speed and temperature; there would also be functions, called member functions, to control the engine such as start, stop, set speed etc. The exact content of the object is specified by the class which it is created from, and multiple objects can be created from a class. This is useful as a lot of applications will have multiples of the same object. In this example a row of engines could each be controlled by an object created from the one class; although the structure of each object would be identical, the values contained in them would vary for each engine.

One of the benefits of object oriented programming comes from object member protection. Each member variable or function can be declared in the class definition as private or public (there is also a protected option, but for clarity that will be ignored here). A private member variable cannot be accessed by code outside of the class, meaning that protection is afforded against rogue program errors causing serious problems. Returning to the car engine example, it may be damaging for the speed of the engine to exceed 5000rpm. To prevent this the engine speed variable would be declared as private and a public member function added to set the speed. Within the function code there would then be a check to ensure that no value above 5000rpm would be accepted; if the value is acceptable then the funtion would update the member variable and set the speed accordingly, otherwise it

would reject the value. This protection mechanism ensures that an error elsewhere in the code can be isolated and its impact limited. For these reasons, object oriented programming techniques were applied wherever possible when developing the software for the lamp monitoring test rig.

As was mentioned in the first half of this chapter during the discussion of the hardware arrangements, there are two measurement devices the software needs to communicate with in order to record all the data. The LabJack device controls the multiplexing of the measured signals and performs the data capture of the temperature and light signals, while the Voltech PM100 power analyser performs the measurement of the voltage and current supplying the lamp under test.

The LabJack device comes supplied with a dynamic link library (dll), which once loaded will allow communication with the LabJack via a series of function calls. A C++ class was constructed which loaded the dll as soon as an object was instantiated from the class, and implemented all the available function calls as member functions of the class.

A similar approach was adopted for the power analyser; however, as the power analyser did not come with any software all the communication with it had to be achieved by sending command strings through the RS232 port. Once again a C++ class was constructed with all the available function calls, so that the communication with the power analyser and serial port was hidden from the rest of the program, helping to make the program more structured and understandable.

One of the simplest methods of storing the data to the harddrive is in the form of a text file. The format decided upon is demonstrated in figure 28, every reading being stored exactly as it was read in the text file along with the date and time of when it was recorded.

```
21Feb2005 23:02 57  Result:- ----- Lamp Turned On, Inrush Current; 1.4269
21Feb2005 23:03 00  Result:- Measurement Cycle Start, Minutes Burned:- ; 0
21Feb2005 23:03 00  Result:- Lamp Temp Sens Voltage; 3.19336
21Feb2005 23:03 00  Result:- Lamp Light Sens Voltage; 0.512695
21Feb2005 23:03 00  Result:- Room Temp Sens Voltage; 2.96875
21Feb2005 23:03 00  Result:- Volts AC; 248.7
21Feb2005 23:03 01  Result:- Amps AC; 0.10835
21Feb2005 23:03 02  Result:- Watts; 26.94
21Feb2005 23:03 02  Result:- Power Factor; 0.9998
21Feb2005 23:03 03  Result:- ----- End of Measurement Cycle -----; 0
21Feb2005 23:12 45  Result:- Measurement Cycle Start, Minutes Burned:- ; 10
21Feb2005 23:12 45  Result:- Lamp Temp Sens Voltage; 6.90918
21Feb2005 23:12 45  Result:- Lamp Light Sens Voltage; 0.576172
21Feb2005 23:12 45  Result:- Room Temp Sens Voltage; 2.97363
21Feb2005 23:12 45  Result:- Volts AC; 250.4
21Feb2005 23:12 46  Result:- Amps AC; 0.1072
21Feb2005 23:12 46  Result:- Watts; 26.77
21Feb2005 23:12 47  Result:- Power Factor; 0.9998
21Feb2005 23:12 48  Result:- ----- End of Measurement Cycle -----; 0
```

Figure 28:- Extract from a results text file.

The file format was designed so that it could be easily read by humans as well as by computer software; this meant that should any spurious results occur, their origin could be thoroughly investigated by looking at the original raw results.

A class called "result handling" was created to deal with the opening and closing of the file as well as time and date stamping of all the results. As there were 8 lamps in total, 8 instances of the result handling class were instantiated so that each lamp had its own results file.

The main timing and control of the test rig were dealt with by two classes. The "lamp class" objects serve to keep track of how long the lamp has been on and how long before it needs to be turned off. Again each lamp has its own lamp class object. The class also has member functions which communicate with the LabJack class to turn the lamp on and off and to trigger a measurement sequence for that lamp. The main timing control however is dealt with by the "scheduler class" which uses a windows system timer to run a member function every minute. This function call is used to query each lamp object in turn to check whether any measurement cycle needs to occur or whether a lamp needs to be switched on or off; if something needs doing then the lamp object will carry out the necessary task. Upon start up, the scheduler also ensures that the lamp turn on times are staggered so that the shared measurement equipment is never in use by two lamps at the same time.

The final class created during the development of the software was the error handling class. This class serves to record any errors that occur, such as communication errors to a text file, so that any effect on the results can be taken into account later.

A Unified Modelling Language (UML) class diagram of the complete data recording software is shown in figure 29; full code listings are included in appendix B. The entirety of the software for recording the measurements was written by the author.

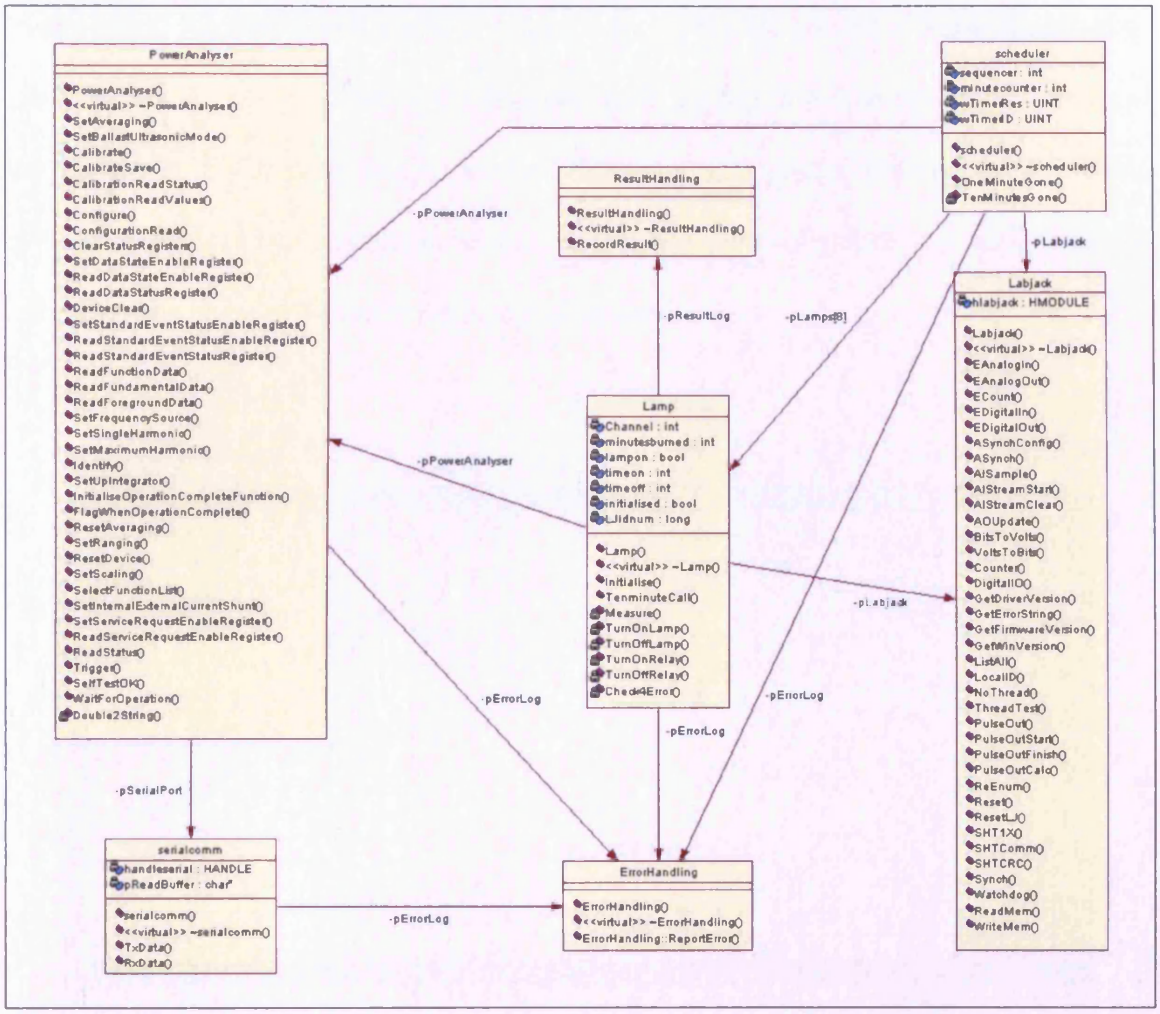


Figure 29:- UML Class Diagram of the Data Recording Software.

4.2.2 Data Analysis

In order to analyse the data obtained from the experiments, it was clear that plots would need to be prepared to show the behaviour of the different lamp parameters over time. Microsoft Excel was chosen to generate the plots due to its comprehensive plotting features as well as its easy to use COM (Component Object Model) programming interface. A C++ program was developed to read in the results file generated by the program discussed in section 4.2.1, trim off any zero results that were measured after the

lamps failed, load Microsoft Excel and then copy all the results into a blank spreadsheet. After the results are transferred, the program adds appropriate headings to the table and then generates XY scatter plots for each of the lamp parameters. Automating the process in this way ensures that large amounts of data can be processed quickly and easily and also minimises the chance of data errors being introduced.

	B	C	D	E	F	G	H
1	Voltage (V)	Current (A)	V/I (Ohms)	Power (W)	Lamp Temp (V)	Room Temp (V)	Light (V)
2	248.7	0.10635	2295.339179	26.94	3.19336	2.96875	0.512695
3	250.4	0.1072	2335.820896	26.77	6.90918	2.97363	0.576172
4	247.9	0.10665	2324.425692	26.44	7.03125	2.96387	0.551758
5	248.8	0.10685	2328.497894	26.57	7.14355	2.95898	0.566406
6	249.6	0.10707	2331.185206	26.72	7.1582	2.95898	0.571289
7	249.2	0.10681	2333.114877	26.56	7.13379	2.93457	0.751953
8	248.1	0.10659	2327.61047	26.43	7.0459	2.94434	0.742188
9	247.1	0.10641	2322.150174	26.27	7.0459	2.95898	0.600586
10	248.3	0.10677	2325.559614	26.52	7.08984	2.9834	0.712891
11	247.5	0.10655	2322.853121	26.37	7.14355	2.96875	0.556641
12	247.9	0.10665	2324.425692	26.43	7.12891	2.96875	0.634766
13	247.8	0.10659	2324.795947	26.36	7.12402	2.98828	0.517578
14	247.8	0.10667	2323.052405	26.44	7.08496	2.96875	0.576172
15	248.8	0.10682	2329.151844	26.56	7.11914	2.98828	0.507813
16	249.2	0.10701	2328.754322	26.66	7.08984	2.97363	0.654297
17	249.7	0.10712	2331.03062	26.69	7.1582	2.96875	0.703125
18	248.6	0.10685	2326.626111	26.56	7.1582	2.9834	0.664063
19	248.2	0.10667	2326.802287	26.42	7.10938	2.97363	0.512695
20	249.2	0.10699	2329.189644	26.64	7.09473	2.97363	0.673828
21	250.3	0.10732	2332.277302	26.86	3.45703	2.97852	0.65918
22	249.9	0.10714	2332.462199	26.77	6.96777	2.98828	0.693359
23	250.2	0.10718	2334.390745	26.79	7.1582	3.00781	0.551758
24	249.5	0.10705	2330.686595	26.7	7.11426	2.99316	0.59082
25	249.4	0.10716	2327.360956	26.77	7.12402	2.96875	0.693359
26	250.5	0.10732	2334.140887	26.87	7.19727	2.96387	0.634766
27	250.7	0.10743	2333.612585	26.94	7.09961	2.97852	0.664063
28	249.7	0.10711	2331.248249	26.73	7.14355	2.97852	0.522461
29	249.7	0.10717	2329.943081	26.75	7.1875	2.9834	0.561523
30	250.1	0.10727	2331.499953	26.82	7.12891	2.94922	0.576172
31	251.8	0.10763	2339.496423	27.1	7.14844	2.95898	0.556641
32	250.9	0.10747	2334.605006	26.98	7.19727	2.98828	0.722656
33	251.6	0.10765	2337.203902	27.09	7.0752	2.97363	0.620117
34	245.8	0.1063	2312.323612	26.18	7.0459	2.9834	0.488281
35	249.8	0.10721	2330.006529	26.8	7.20703	2.97363	0.649414
36	249.7	0.1071	2331.46592	26.73	7.09473	2.96387	0.512695
37	248	0.10674	2323.402661	26.47	7.12891	2.97852	0.625
38	247.9	0.10662	2325.079722	26.41	7.11426	2.97363	0.561523
39	248.7	0.10689	2326.690991	26.62	7.08984	2.97852	0.581055
40	248.1	0.10679	2323.251241	26.48	3.42285	2.96875	0.50293
41	248.4	0.10675	2326.932084	26.49	6.95313	2.9834	0.634766
42	248.2	0.1069	2321.796071	26.56	6.97266	2.98828	0.625
43	247.1	0.10655	2319.099015	26.34	7.06055	2.98828	0.678711
44	246.7	0.10645	2317.519962	26.28	7.14844	2.9834	0.649414
45	247.3	0.10662	2319.45226	26.32	7.05078	2.96875	0.527344
46	245.4	0.10612	2312.476442	26.04	7.08984	2.96387	0.483398
47	245.9	0.10627	2313.91738	26.15	7.09961	2.96875	0.498047

Voltage | Current | Resistance | Power | Lamp Temp | Room Temp | Lamp Light | Power Error | Sheet1 | Sheet2

Figure 30:- An example of one of the automatically generated spreadsheets.

The program also automatically titles each of the plots that are generated with the lamp number that appears in the filename of the recorded results text file, this feature was included to ensure that the results from each lamp could not get mixed up.

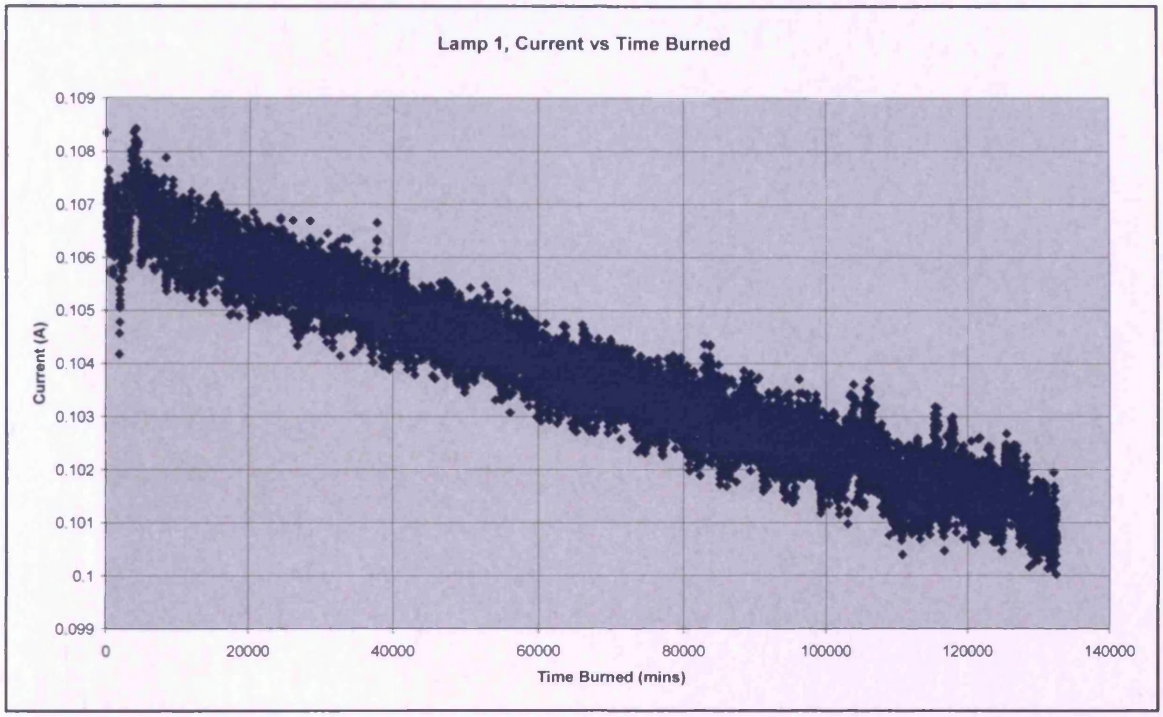


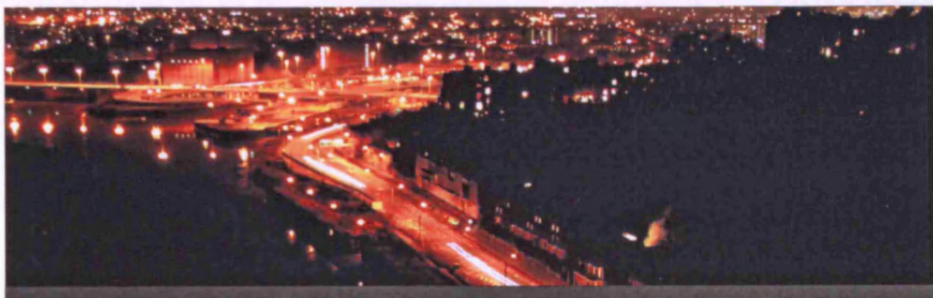
Figure 31:- An example of one of the automatically generated plots.

With the software complete, it was now possible to run an experiment simultaneously on upto 8 filament lamps at a time for as long as the lamps lasted, something that would not have been practical by hand.

Full code listings for the data analysis software are included in appendix B. Apart from the standard compiler additions for an MFC (Microsoft Foundation Class) application and the standard library files "excel.h" and "excel.cpp", which are provided by Microsoft for

interfacing with Excel, the entirity of the analysis software was written by the author for this application.

In the next chapter, the first filament lamp experiments conducted using these software tools are discussed, together with the results that were obtained.



5

Filament Lamp Monitoring Experiments

5.1 Continuous Monitoring

Using the automated test rig that was discussed in chapter four, eight GLS lamps made by General Electric were put on test. Four 25W lamps and four 40W lamps were used (lamp number 9 was added in place of lamp 5 after it failed within the first hour); these were followed by another batch of eight 40W lamps which were all branded as Tesco lamps. Each lamp was turned on for a period of 180 minutes (3 hours), and then turned off for 20 minutes to allow the lamp to cool; this cycle was repeated until each lamp failed. This is a standard test cycle used by manufacturers.[8] Immediately after each lamp was turned on a set of readings were taken which included lamp voltage, current, power, temperature and light output, the same readings were also taken every 10 minutes that the lamp was on. From Tables 3 & 4, it can be seen that despite the 40W lamps having a 1000 hour rated life, the actual life of the lamps varied greatly between the samples.

Lamp No.	Power (W)	Time to Failure (minutes)	Time to Failure (hours)
3	25	475,900	7932
2	25	136,000	2267
1	25	132,590	2210
8	40	65020	1084
4	25	50,840	847
7	40	49550	826
6	40	48120	802
9	40	36430	607
5	40	50	<1

Table 3:- Time to Failure for the first batch of tested lamps.

Lamp No.	Power (W)	Time to Failure (minutes)	Time to Failure (hours)
15	40	57950	966
10	40	55710	929
13	40	50070	835
11	40	48130	802
12	40	44690	745
16	40	44250	738
14	40	43390	723
17	40	32970	550

Table 4:- Time to Failure for the second batch of tested lamps.

For each of the lamps tested, the results were automatically loaded into an excel spreadsheet and plots produced for the different lamp parameters. Most of the lamps showed very similar trends for the different parameters. It was noticed that most of the lamps had multiple fractures in the filament after failure, suggesting that the filaments had thinned significantly over their entire length, rather than just in one place.

5.1.1 Lamp Voltage Monitoring

The lamp voltage was recorded, as although this is a result of the mains voltage rather than a lamp parameter, it is important to have a record of it so that it can be taken into account when explaining fluctuations in the other readings. A plot of the Voltage across lamp one is shown in figure 32.

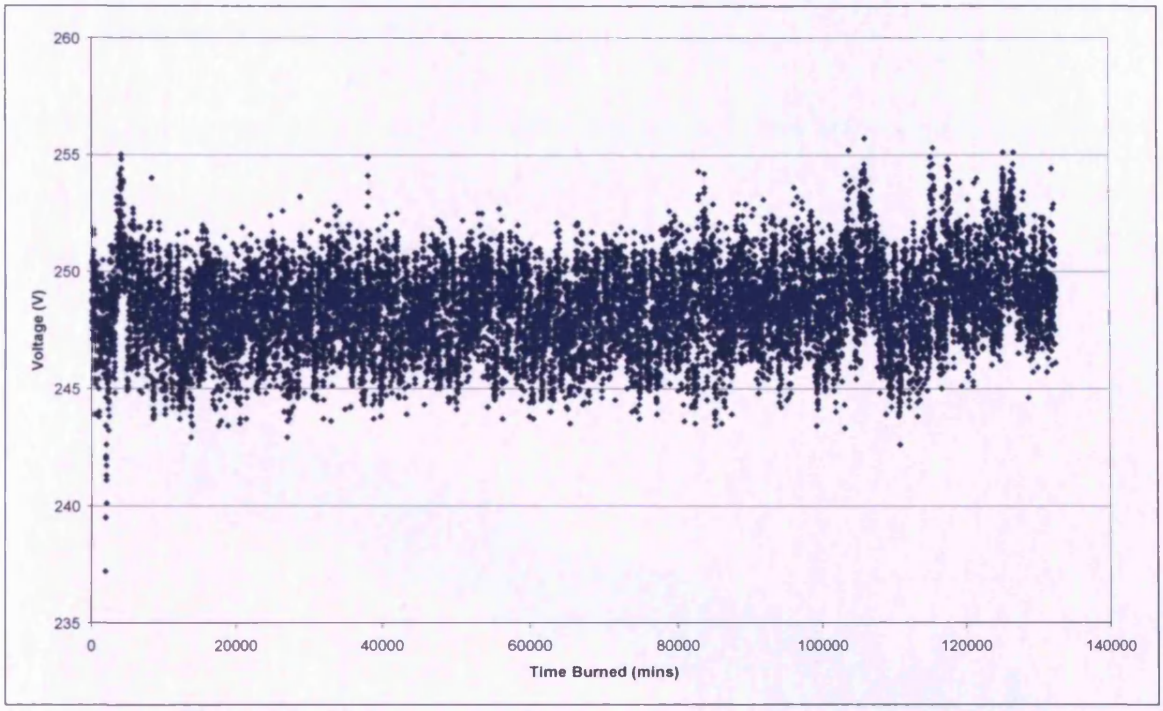


Figure 32:- Voltage across lamp number 1 while it was burning

13,996 readings were taken in total, during which time the lowest recorded voltage was 237.2V and the highest was 255.7V. The average of all the readings was 248.5V.

From figure 32, it can be seen that throughout the experiment the mains voltage mostly varied between 245V and 252V, a fluctuation of 7V, this is most probably due to the varying energy use of the building during the day and night. During the last 30,000 minutes of the experiment the average mains voltage appears to have risen very slightly, this change occurred in the middle of May, which suggests this is also the result of a change in energy use patterns, due to the summer break. These small changes are unlikely to have a large effect on the other lamp parameters. However they were recorded so they could be taken into account when analysing the other results.

5.1.2 Lamp Current Monitoring

The next measurement taken was lamp current and the plot for lamp number one is shown in figure 33.

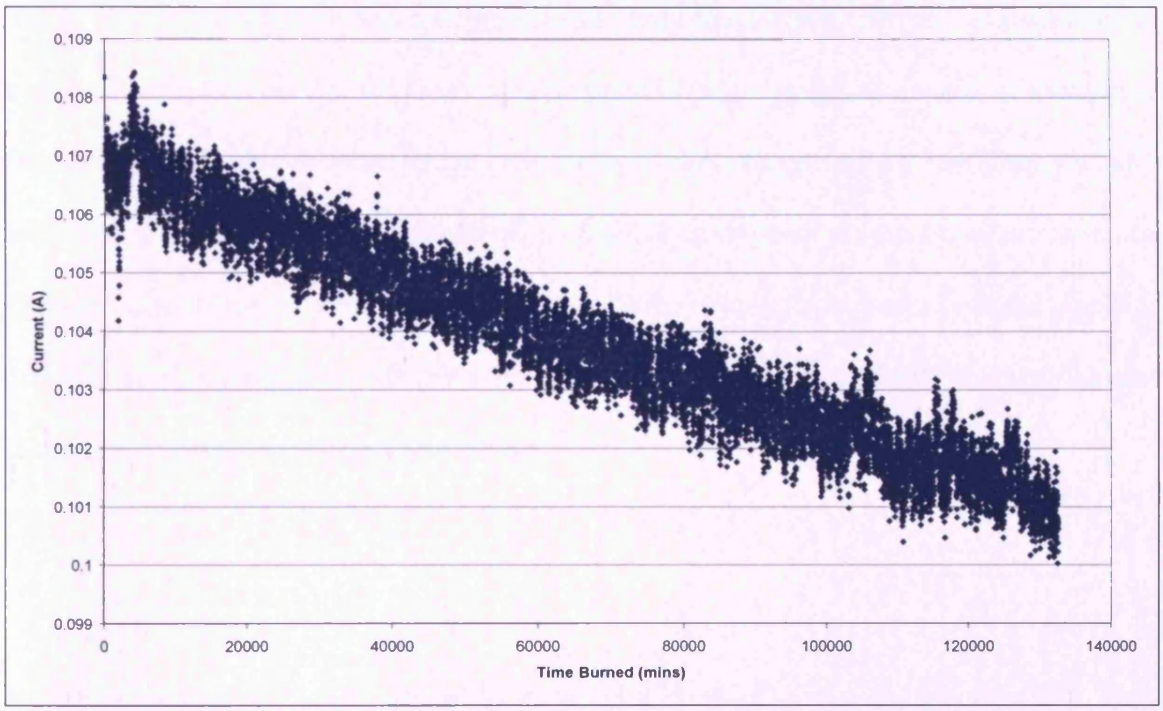


Figure 33:- Current through lamp number one while it was burning

Looking at figure 33, it can be seen that the lamp current showed a general downwards trend while the lamp was burning, coinciding with what was expected as the filament will thin during use due to evaporation, which causes it to increase in resistance and hence the current through it to decrease.

The current decrease of 0.006A is very small and the affect of the mains voltage fluctuations on the filament current can be seen in the variation of the readings (approximately ±0.001A).

5.1.3 Filament Resistance Monitoring

It is clearly important that the mains voltage fluctuations are either kept under control or compensated for in the results in order to improve the accuracy. A simple way of doing this is to plot the filament hot resistance rather than the filament current, as resistance is equal to voltage divided by current and filament current increases as a result of increasing filament voltage, the fluctuations in both current and voltage should therefore partially cancel each other. The fluctuations will not cancel completely, as the filament resistance changes with filament temperature, which will depend on the mains voltage applied, however some cancellation will be achieved. Figure 34 shows a plot of the hot filament resistance of lamp one while it was on test.

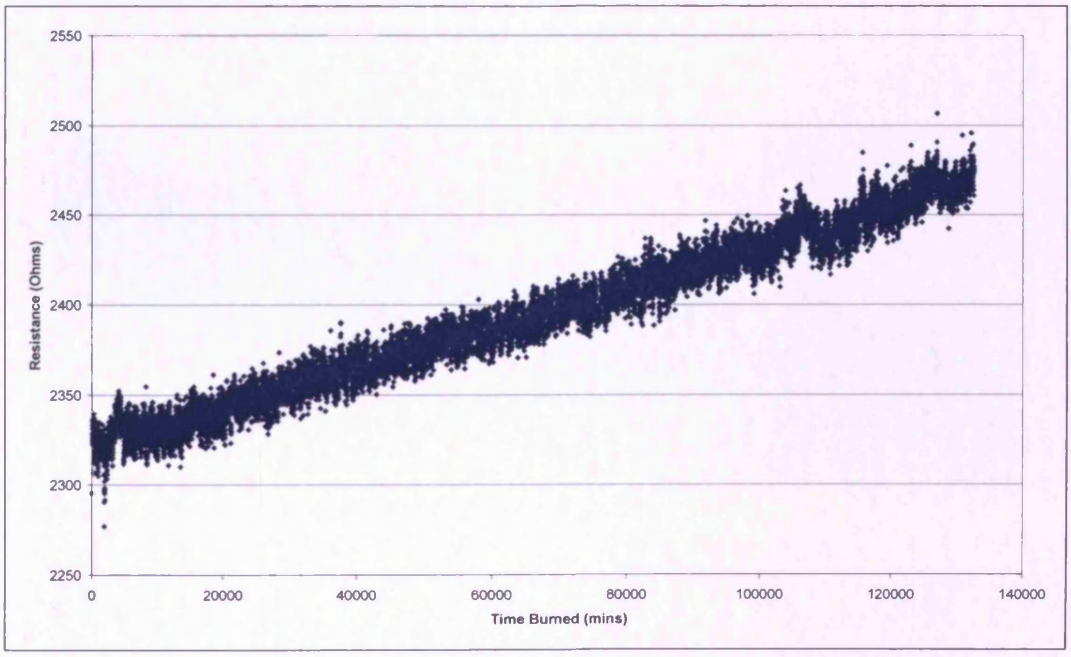


Figure 34:- Resistance of lamp number one while it was burning

It can be seen from figure 34 that although some fluctuation is still visible on the resistance plot, it is proportionally smaller than the fluctuation visible on the current plot.

5.1.4 Lamp Power Monitoring

Another parameter which was measured was the lamp power. Although this is essentially the same as the voltage and current multiplied together, due to the lamp being resistive and having near unity power factor, it was recorded anyway as a secondary check for the results. Once again, the plot which can be seen in figure 35 shows a general decrease in power consumption as the lamp ages, presumably due to the increasing filament resistance caused by tungsten evaporation. The mains voltage fluctuations can also be seen causing fluctuations in the power consumption.

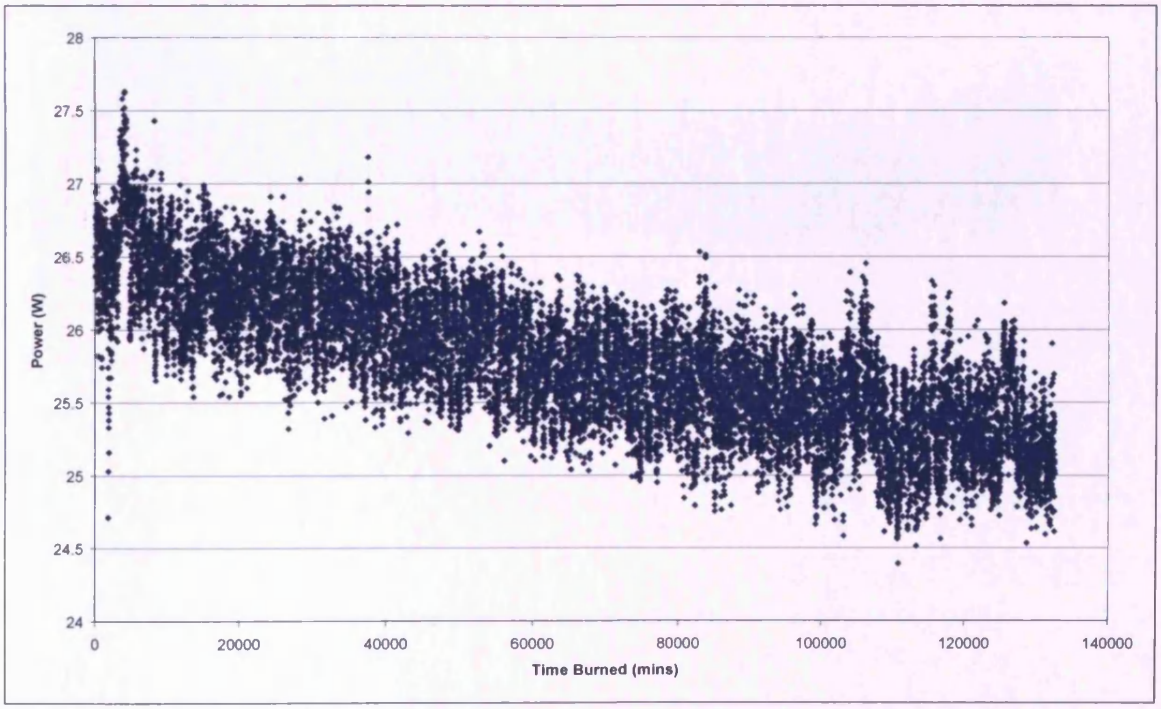


Figure 35:- Power consumption of lamp number one while it was burning.

5.1.5 Lamp Light Output Monitoring

As well as affecting the electrical characteristics of the lamp, the light output and temperature of the lamp should also reduce as the filament evaporates due to the increasing filament resistance and corresponding decrease in input electrical power. To test this theory, the light output was recorded together with the temperature of the casing surrounding each lamp every time a set of readings were taken. The room temperature was also recorded so that the effects of the fluctuating air temperature could be accounted for.

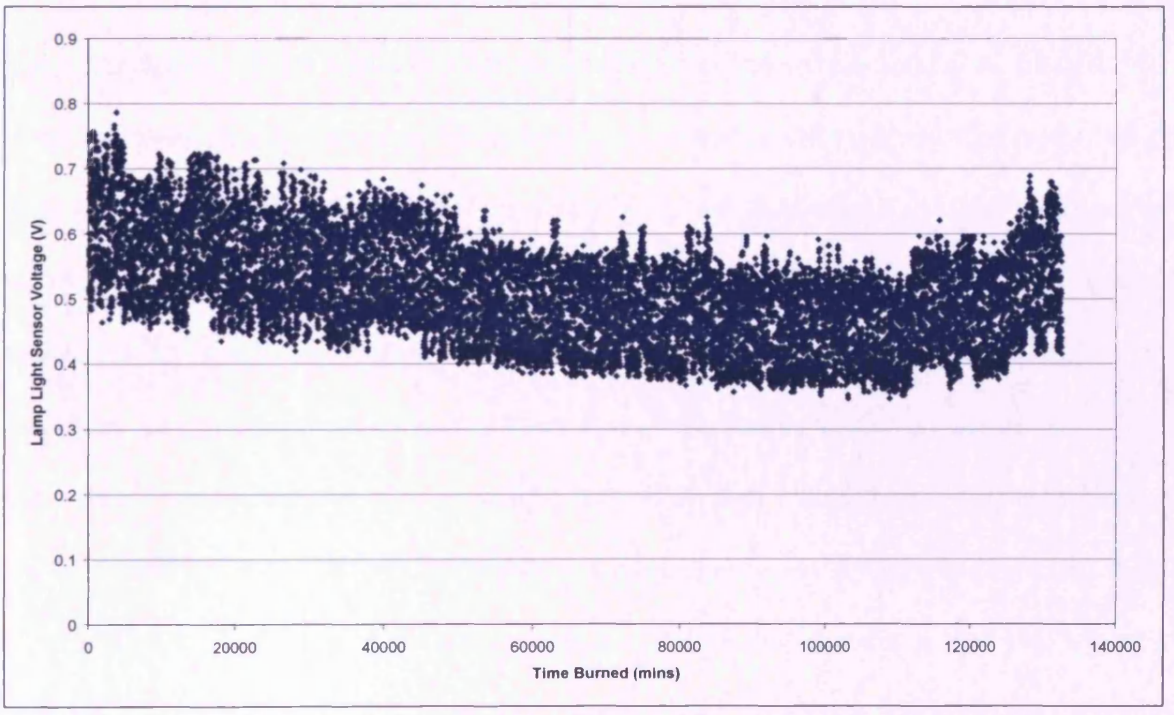


Figure 36:- Light output of lamp number one while it was burning.

The light sensor used was designed to produce a constant current which is linearly proportional to the intensity of the light on the sensor. This was converted to a measurable voltage signal by using a resistor to form a potential divider. As the experiment was

looking for a measurable trend rather than trying to measure the actual light output, the results were left as voltages.

Once again, there is a general downward trend shown by the plot in figure 36. During the last 30000 minutes however there is a significant rise in the light output, which coincides with the rise in mains voltage observed on the voltage plot.

5.1.6 Temperature Monitoring

The temperature results required further manipulation before they could be plotted, the lamp and room temperatures were measured by thermistors, with the recorded result being the voltage across the 56k resistor in series with the thermistor. As the relationship between thermistor resistance and temperature was not linear, plotting the measured voltage trend would have been misleading. Instead the measured voltage had to be converted to a temperature using the manufacturer's datasheet, so that the effect of the thermistor's non-linear response would be removed. The manufacturer's datasheet only gave a table of 206 discrete conversion points, so the computer software package MATLAB was used to plot the characteristic curve of the thermistor and the "spline" function within MATLAB was used to perform "cubic spline interpolation" between the points, which is shown in figure 37.

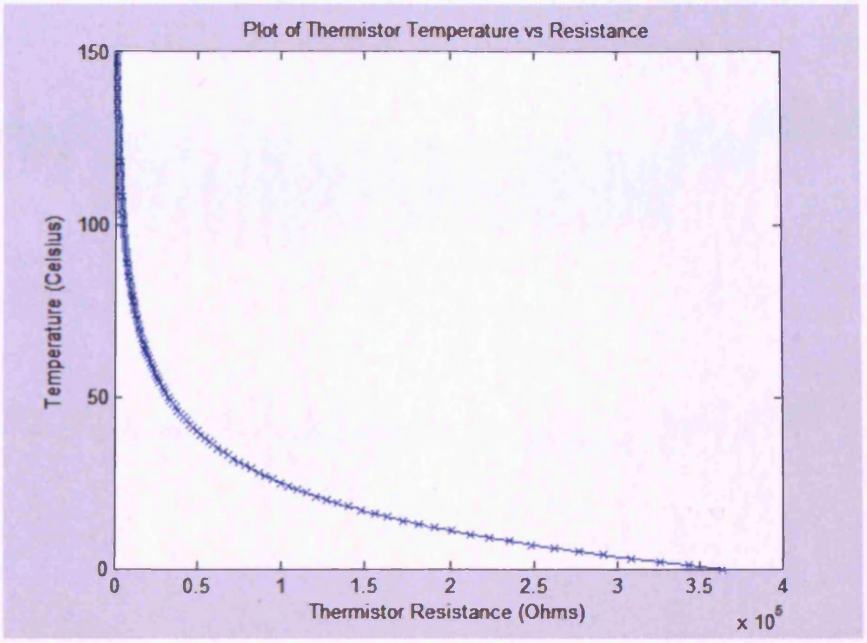


Figure 37:- Thermistor Temperature vs. Resistance with interpolated points

Using MATLAB, the voltage measured across the 56k resistor in series with the thermistor was used to calculate the thermistor's resistance for each measurement using the following equation:-

$$R(\Omega) = \frac{10 - Vin(V)}{\frac{Vin(V)}{56000}} = \frac{560{,}000}{Vin(V)} - 56{,}000$$

Equation 1:- Calculating thermistor resistance from input voltage

where Vin is the voltage measured across the 56k resistor, R is the resistance of the thermistor in ohms and the "10" is the voltage across the potential divider, which was kept constant using a voltage regulator. The MATLAB "spline" function was then used again to convert each thermistor resistance reading into a temperature in degrees Celsius. Once converted, the lamp temperatures were plotted in Excel, as shown in figure 38.

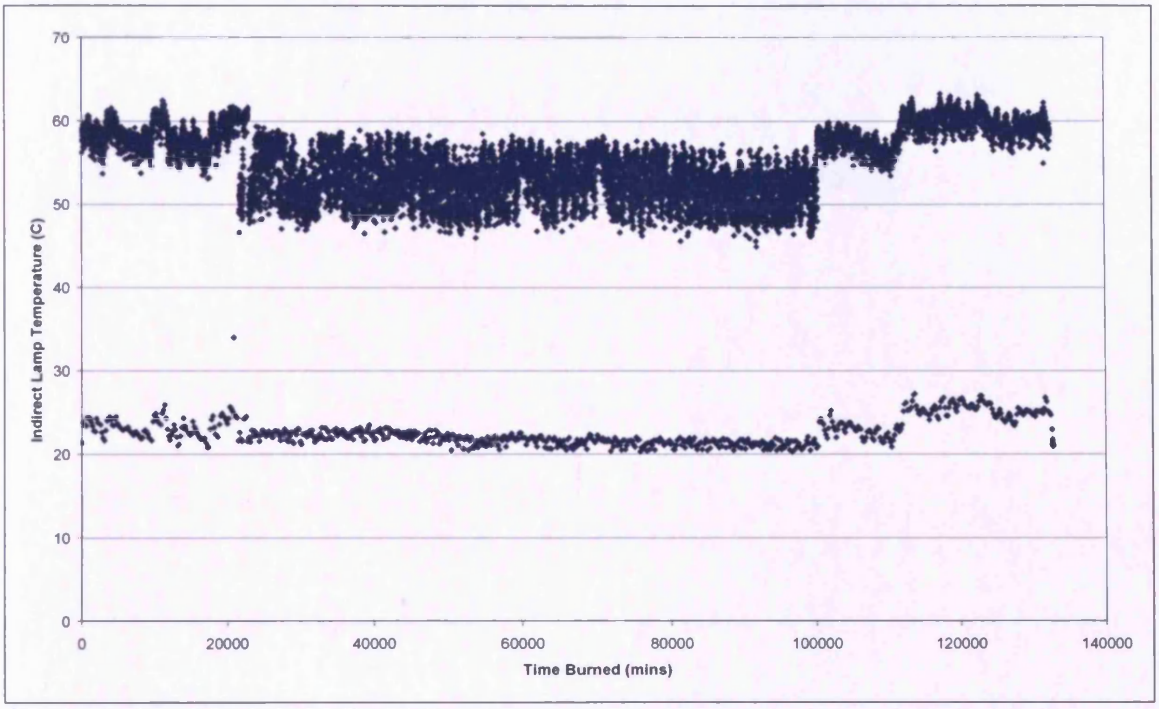


Figure 38:- Indirect temperature of lamp number one while it was burning.

Looking at figure 38, it can be seen that there are two discrete bands of readings. All but one of the readings are either between 20-30°C or between 45-65°C. The reason for this is that the first reading is taken a few seconds after the lamp has been turned on, which means it is virtually at room temperature.

There are also two regions on the plot where the readings are more erratic than normal; these are during the first 2,000 minutes of burning and the last 30,000 minutes. These regions can be explained by looking at the room temperature plot which is shown in figure 39.

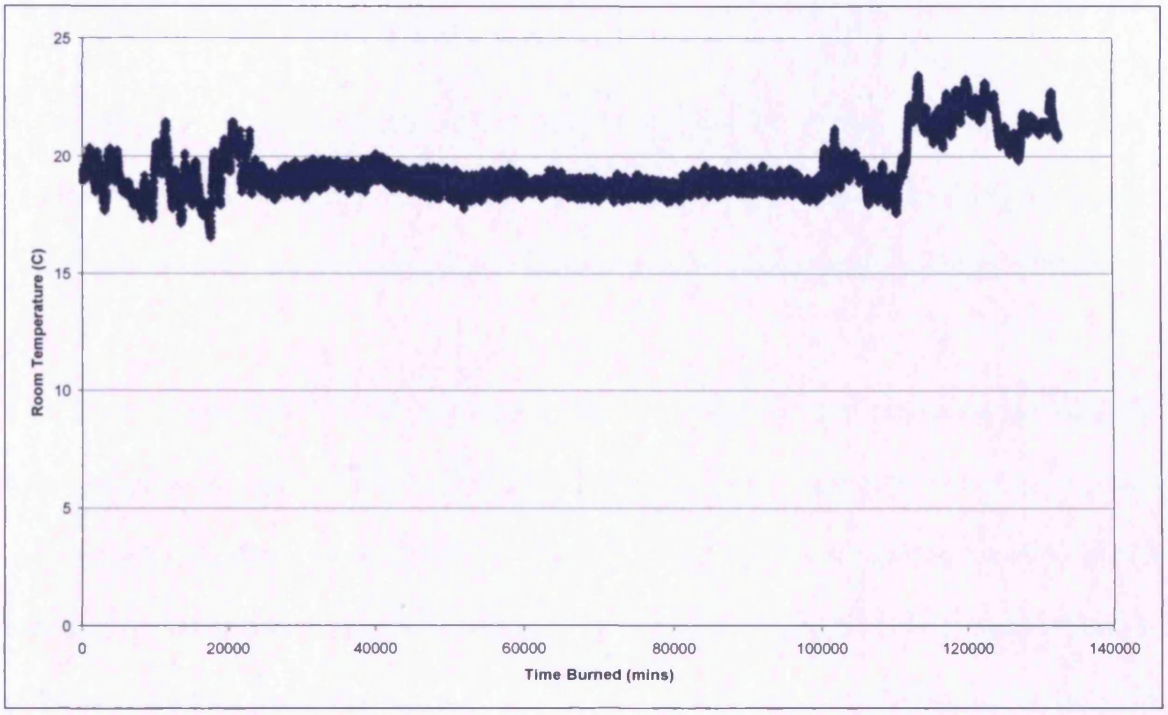


Figure 39:- Room temperature while lamp number one was burning.

From figure 39, it can be seen that the room temperature fluctuations are greatest at the same time as the lamp temperature fluctuations; this indicates that the fluctuations in lamp temperature have been caused by the room temperature changing during the different seasons of the year, rather than any change in the lamp characteristics.

It is very difficult to identify any trend in the plots of lamp temperature. It appears that if there is a trend there then it is obscured by room temperature fluctuations. Considering the experiment was conducted in a room where the temperature varied by less than 10°C, it seems very unlikely that measuring lamp temperature would be a practical proposition for a real monitoring system, where ambient temperature would be much more varied.

5.1.7 Mains Voltage Compensation

In the majority of plots looked at so far, each lamp parameter showed a general downward or upward trend as the lamp aged; however the effect of the fluctuating mains voltage has meant that the trend is often disrupted, which makes predicting lamp life rather difficult.

The primary problem with a fluctuating mains voltage is that it causes the filament temperature to fluctuate which in turn affects the filament resistance and hence lamp power and current. It would be difficult and relatively expensive to regulate the AC voltage reaching the lamp, which ruled this out as an appropriate approach for low cost monitoring systems. The only other approach is to try and compensate for the fluctuating voltage. A simple approach was shown in section 5.1.3, where the lamp resistance was plotted rather than lamp current, the fluctuations in lamp voltage and current partially cancel each other when the resistance is calculated. However this approach only works fully if the resistance of the filament does not vary with mains voltage, which for a filament lamp is not the case.

To compensate more fully for mains voltage fluctuations, the fluctuation of filament resistance with mains voltage needs to be taken into account. As the filament resistance will not vary linearly with mains voltage, an experiment was conducted to determine how the filament resistance changes with mains voltage. This relationship could have been determined theoretically, however it was done experimentally to take account of heat loss to the environment, which would have been very difficult to calculate.

A filament lamp was connected to a variac and its voltage, current and power monitored using the Voltech PM100 power analyser which had been used in the first experiment. A diagram of the experiment setup is shown in figure 40.

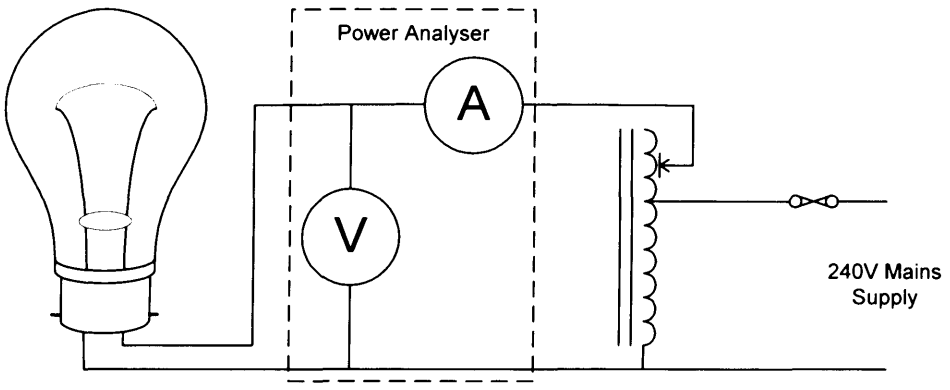


Figure 40:- Experimental setup for investigating filament resistance variation.

Four different wattage lamps were used, each was connected to the circuit shown above and the voltage applied to it was increased from 0 to 265V in 5V increments using the Variac. At each voltage, the lamp was given 5 minutes for the filament temperature to stabilise, before the current was recorded. The current and power readings were then plotted against the applied voltage to produce the plots shown in figures 40 and 41.

From the plots it can be seen that the filament resistance increases as the voltage across the lamp rises. The increase is not linear, so the next step was to try and obtain an expression for the relationship so that it could be used to correct for mains voltage fluctuations in the readings obtained from the other experiments.

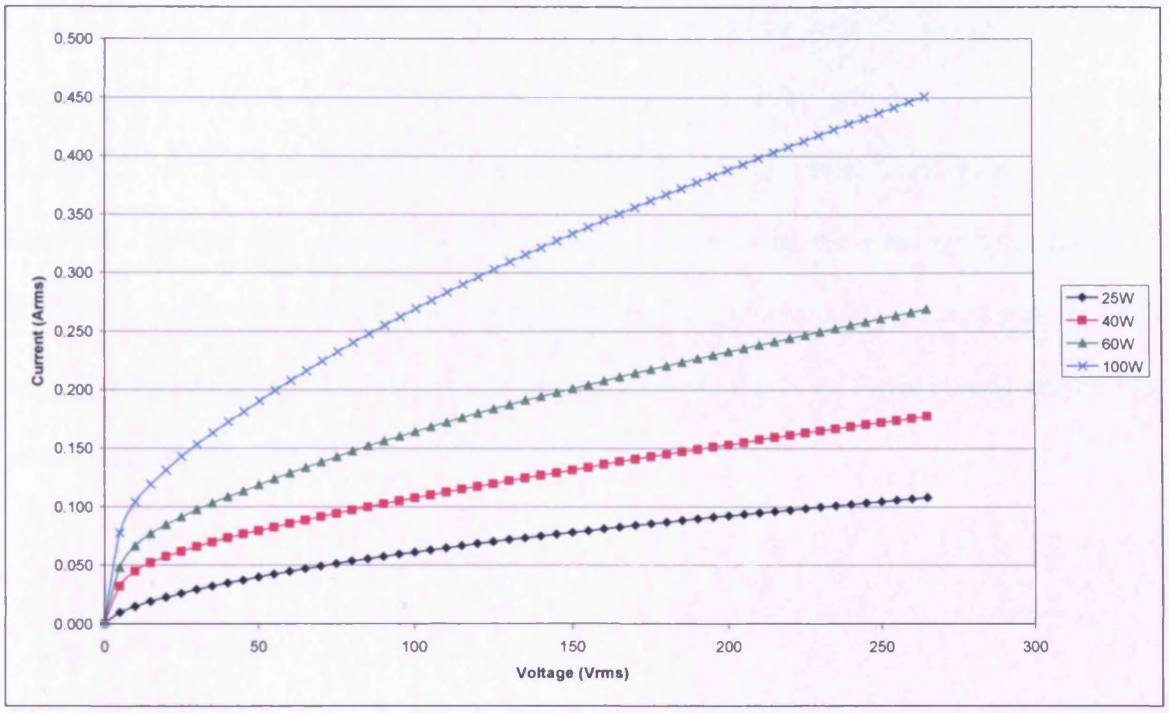


Figure 41:- Current vs. Voltage for each GLS lamp power rating

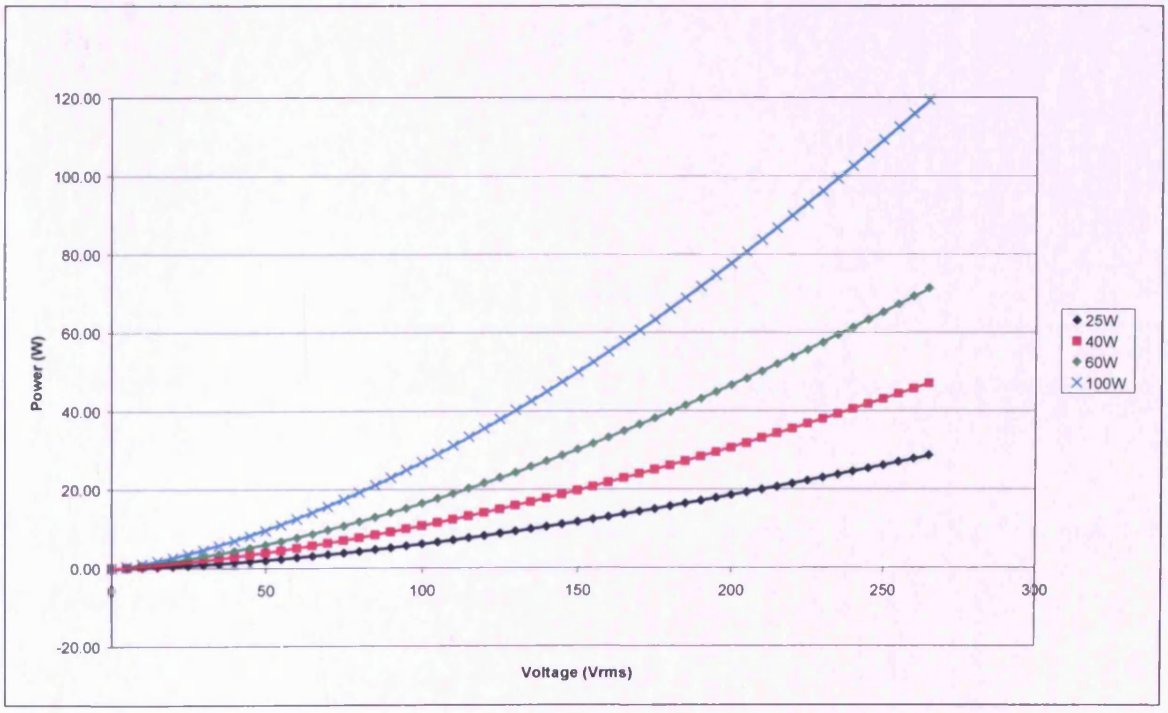


Figure 42:- Power vs. Voltage for each GLS lamp power rating

Looking at figure 42, it can be seen that although the trend lines do separate from each other, their shape is very similar. It was found that if the power of each lamp was converted into a percentage of its value at 240V then the trend lines were practically identical. So that one equation could be used to compensate the readings from lamps of different powers, the trend lines of the four lamps were converted to percentages and then averaged to remove the very small variations present; a plot of the averaged percentage power against voltage is shown in figure 43.

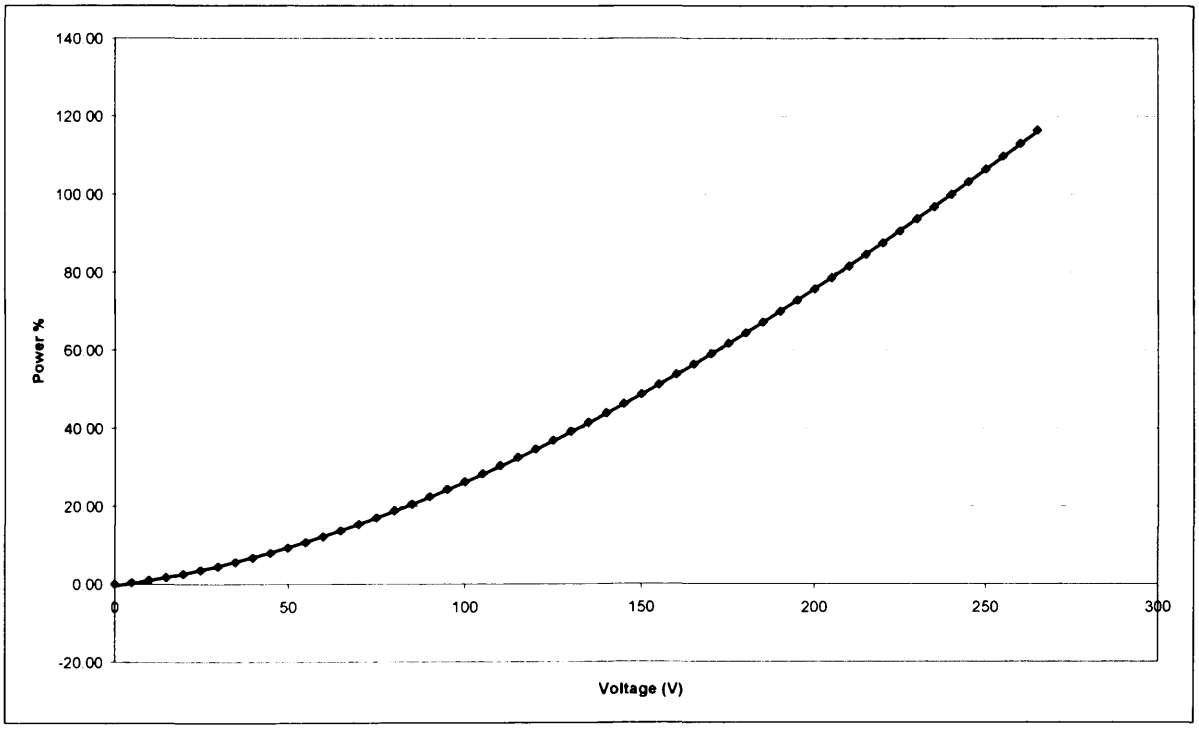


Figure 43:- Averaged percentage power of the four lamps versus supply voltage

Using Microsoft Excel, a third order polynomial trend line was fitted to the data and this was used to generate an equation for the curve.

$$y = -1.472\times10^{-6}x^3 + 1.596\times10^{-3}x^2 + 0.121x - 0.450$$

Equation 2:- Calculating percentage power from mains voltage

Using this equation, it was possible to calculate the expected power draw of a lamp for each value of mains voltage recorded during the experiment. To do this, the power drawn by the lamp when it was powered up for the first time was divided by the percentage expected for the mains voltage at the time; the result was then multiplied by each calculated percentage value to give a predicted power for each value of mains voltage measured during the experiment. The difference between the measured power and predicted power was then calculated for each of the lamps and plotted against minutes burned; figure 44 shows the result of this for lamp number one. Plots of the power difference for all the tested lamps are included in appendix C.

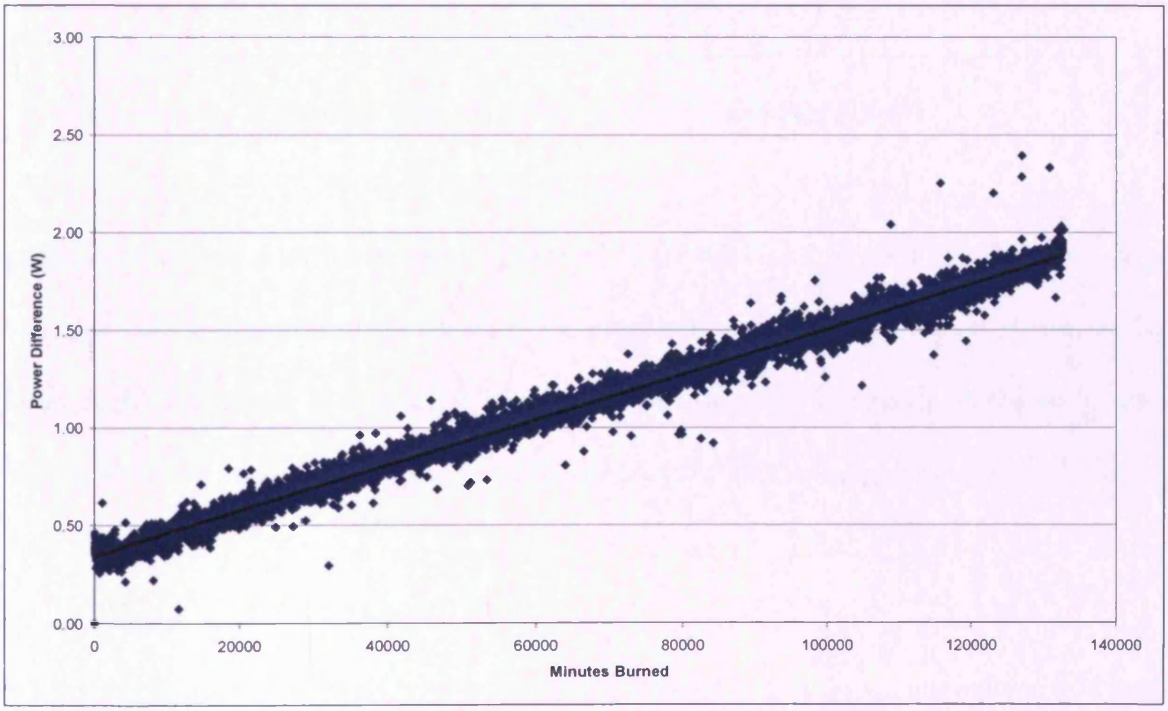


Figure 44:- Lamp 1 (25W) power difference (expected – actual) vs. minutes burned

By comparing the plot of power difference shown in figure 44 with the plot of recorded lamp power shown in figure 45, it can be seen that the mains voltage compensation has significantly reduced the effect of the mains voltage fluctuations on the readings.

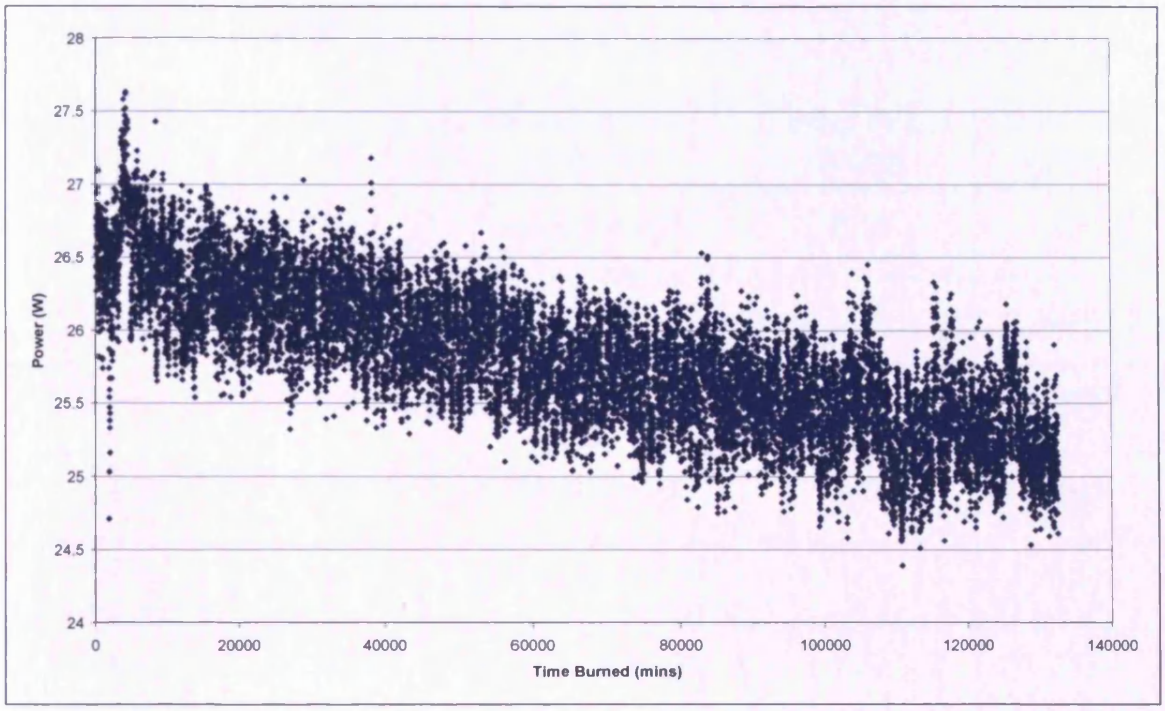


Figure 45:- Lamp 1 (25W) power vs. minutes burned

Looking at the power difference plot shown in figure 44, it can be seen that there is a very clear increase in the difference between the predicted power and the actual power as the lamp ages. This shows that the real lamp power is gradually decreasing as the lamp ages and the filament thins. Plots of the power difference for all the tested lamps are included in appendix C.

The rate of increase in the difference between predicted power and actual power was then calculated for the remaining lamps and tabulated, together with how long they actually lasted.

5.1.8 Results & Discussion

Lamp Number	Rated Power (W)	Corrected Power Drop (W)	Time to Failure (hours))	Hours / Watt decreased
1	25	1.90 - 0.34 = 1.56	2210	1417
2	25	1.29 - 0.27 = 1.02	2267	2223
3	25	3.72 – 2.63 = 1.09	7932	7277
4	25	1.20 - 0.39 = 0.81	847	1045
5	40	4.19 – 0.00 = 4.19	<1	0.20
6	40	1.12 – 0.18 = 0.94	802	853
7	40	1.50 – 0.17 = 1.33	826	621
8	40	1.72 – 0.10 = 1.62	1084	669
9	40	1.26 – 0.46 = 0.80	607	759
10	40	1.23 – (-0.10) = 1.33	929	699
11	40	1.21 – (-0.04) = 1.25	802	642
12	40	1.16 – (-0.08) = 1.24	745	601
13	40	1.24 – (-0.08) = 1.32	835	633
14	40	1.21 – 0.08 = 1.13	723	640
15	40	1.15 – (-0.07) = 1.22	966	792
16	40	1.07 – (-0.18) = 1.25	738	590
17	40	0.70 – (-0.13) = 0.83	550	663

Table 5:- Results table for all tested filament lamps

Looking at the results table, it can immediately be seen that there is a significant difference between the behaviour of the 25W lamps and the 40W lamps. The 25W lamps on average lasted 3314 hours, significantly longer than the 40W lamps, which on average lasted only 739 hours. The average time taken for a drop of one Watt in electrical power was 628 hours for the 40W lamps and 2991 hours for the 25W lamps. The probable explanation for this is that the filament of a 25W lamp runs at a lower temperature than the filament of a 40W lamp, therefore it evaporates at a slower rate and lasts for longer, but at the cost of light output. Due to the different behaviour of the lamps, the results for the 40W and 25W lamps need to be considered seperately.

Lamp number five was also excluded from the comparisons as it only lasted for 50 minutes, which meant there were very few measurement results for it and so it will be considered separately later on.

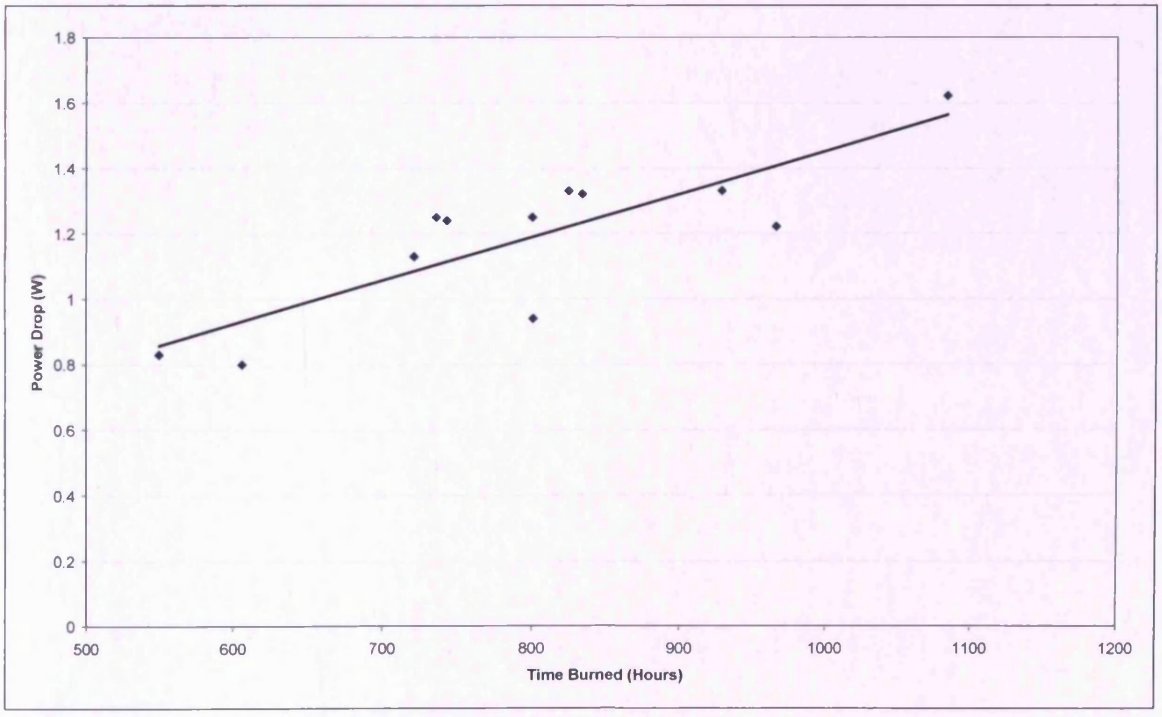


Figure 46:- Power drop at failure vs time burned until failure for the 40W lamps

From figure 46, a trend is clearly visible, the decrease in power of the lamp by the time it fails is approximately proportional to the length of time it burned before it failed. This is a disappointing result for lamp condition monitoring, as it would appear that the filament power decreases linearly with the age of the lamp and although the rate of decrease varies between the lamps, it appears to have little observable effect on when the point of failure will occur.

The results from the 25W lamps however showed a rather different trend, as can be seen in figure 47. The projected trend line shown was fitted using the first three points and the origin of the plot which was set as the intercept. The fourth point was omitted when fitting the trend line as its behaviour was clearly different to the other points on the plot and the results observed for the 40W lamps. Although there are very few data points, due to test rig and time limitations, the dotted line represents a reasonable prediction based on the similar trend observed for the 40W lamps.

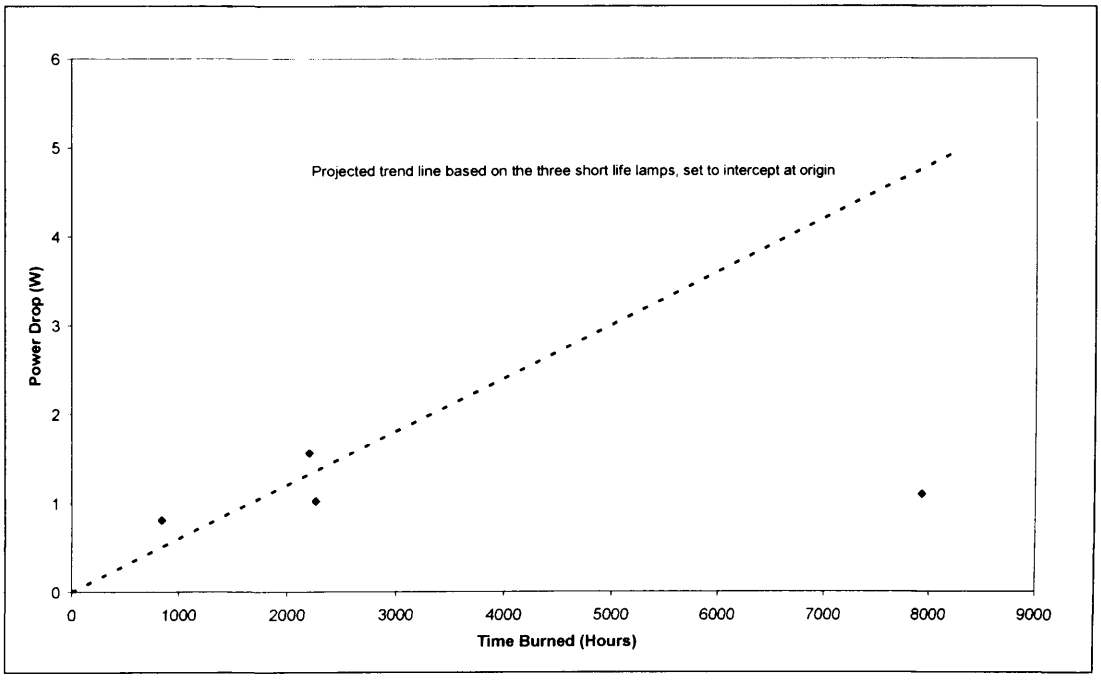


Figure 47:- Power drop at failure vs time burned until failure for the 25W lamps

From figure 47, it can be seen that the three points on the left appear to point to the same behaviour as was observed for the 40W lamps; the longer the lamp burned, the higher the power drop it experienced. One of the four lamps however, shown on the right of the graph, behaved very differently to the others; this lamp burned for 7277, hours which was seven times longer than its 1000 hour rated life and almost four times longer than any of the other lamps. From the projection based on the behaviour of the other lamps tested, had any of them gone on to last this long, then due to their rate of filament evaporation their power consumption would have dropped by 4.75 Watts in this time, rather than just over one Watt as was observed for this lamp. Something was clearly causing the filament of this lamp to degrade at a much slower rate than the other lamps.

Since the predominant cause of filament failure is the formation of hotspots, which is accelerated by a high filament temperature, it seemed reasonable that a filament which lasted almost four times longer than any of the other lamps may have been running cooler. To determine if there was any sign of this in the recorded lamp measurements, the recorded lamp power was examined for each lamp, rather than just the relative power dropped as the lamp burned, which had been considered previously. To make the comparison valid, the power readings for each lamp were adjusted for mains voltage compensation in the same way as discussed for the previous experiment and the power readings for the first 100 hours of burning were used to extrapolate an initial power reading so that the effects of spurious readings would be reduced. A plot of this for lamp 1 is shown in figure 48, plots for all the lamps are shown in appendix C. The point where the line of best fit intercepts the y axis was taken as the initial starting power. The actual initial starting power of the lamp would have been slightly higher as the lamp power drops in the first few seconds of

use; however using this transient value would have been misleading as the drop is very quick and the first measurement time from switch on may have varied by a few seconds for each lamp, which would have made measurements for this reading unreliable.

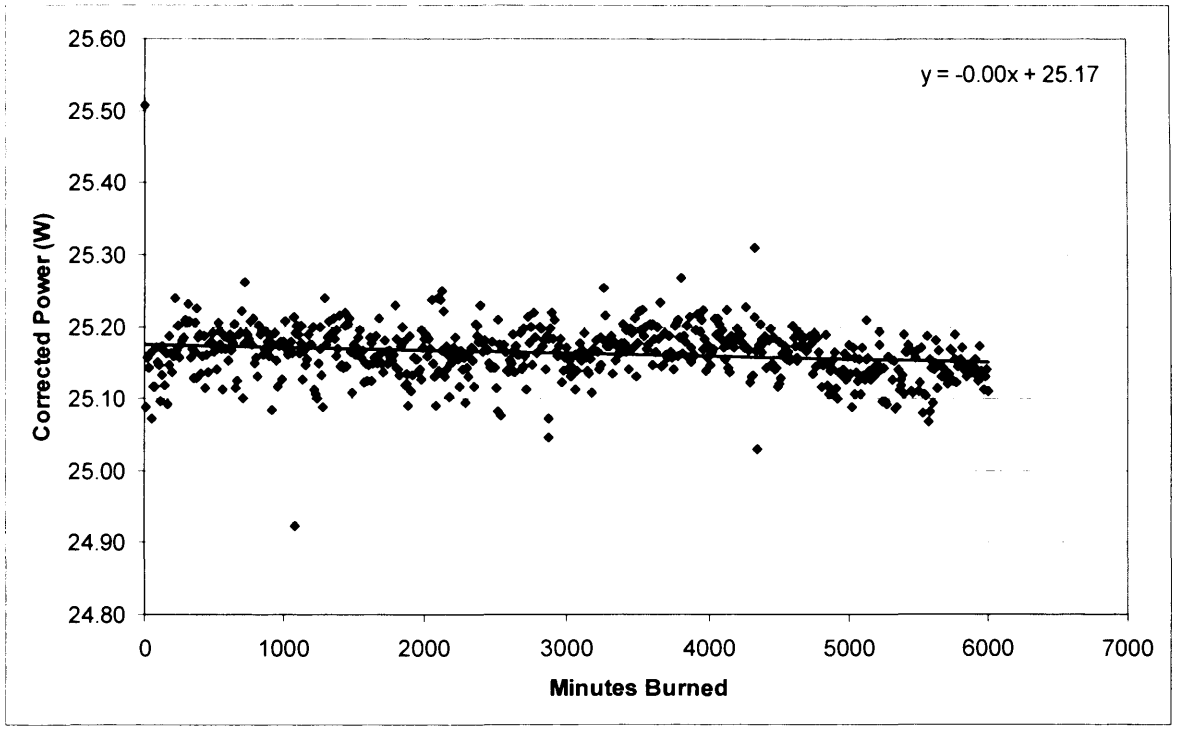


Figure 48:- Voltage compensated power draw of lamp 1 (25W) during the first 100 hours.

Once the initial running power was obtained for all lamps, it became apparent that there was indeed a plausible explanation for the long life of lamp three; its running power was over one Watt lower than any of the other 25 W lamps, as can be seen in table 6.

Lamp Number	Rated Power (W)	Initial Running Power (W) (Extrapolated Over 100hrs)	Time to Failure (hours))
1	25	25.17	2210
2	25	24.23	2267
3	25	23.00	7932
4	25	25.97	847
5	40	-	1
6	40	39.86	802
7	40	39.77	826
8	40	39.56	1084
9	40	39.81	607
10	40	40.06	929
11	40	39.88	802
12	40	40.26	745
13	40	40.26	835
14	40	40.68	723
15	40	39.84	966
16	40	40.48	738
17	40	40.22	550

Table 6:- Initial Lamp Running Power

Looking at the values in the table, it is apparent that there is a relationship between the initial running power of the lamp and how long it lasted. To explore this further, plots were produced of lamp life against the initial running power of the lamp. Each of the different lamp types were plotted separately, as the results varied with the design of the lamp. Lamp 5 was excluded from the plots as it did not last long enough to average.

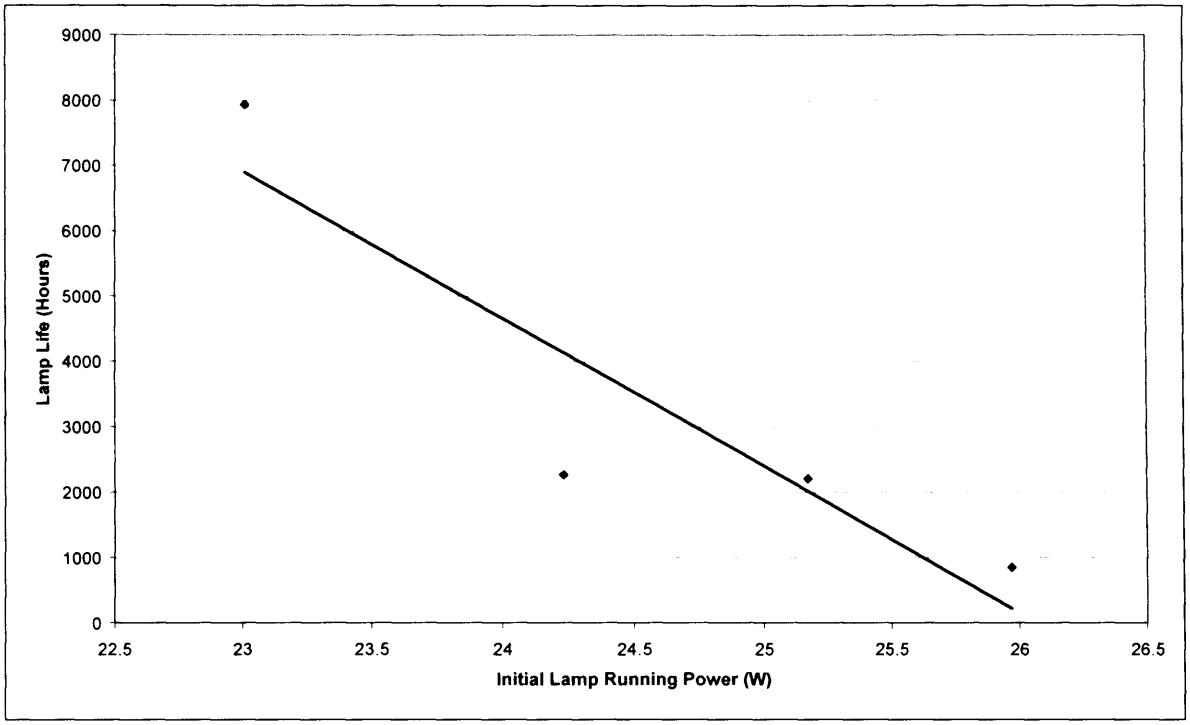


Figure 49:- Initial running power vs life for 25W GE lamp samples (lamps 1-4).

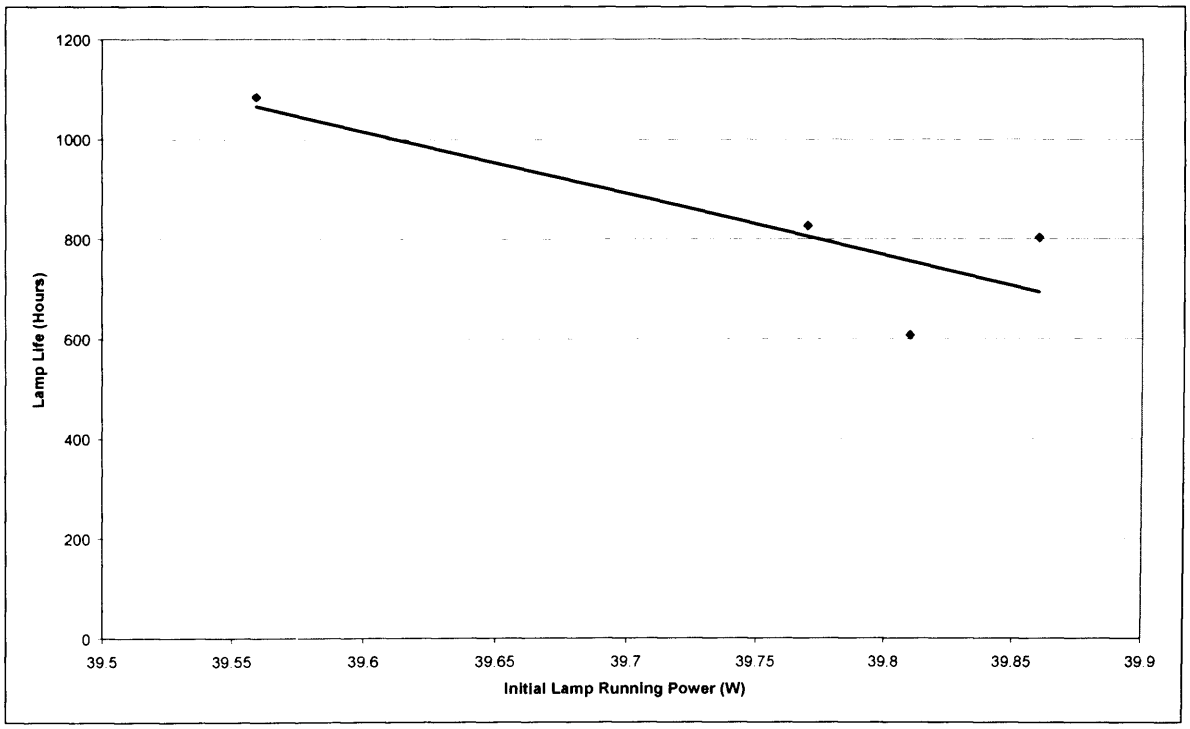


Figure 50:- Initial running power vs life for 40W GE lamp samples (lamps 6-9).

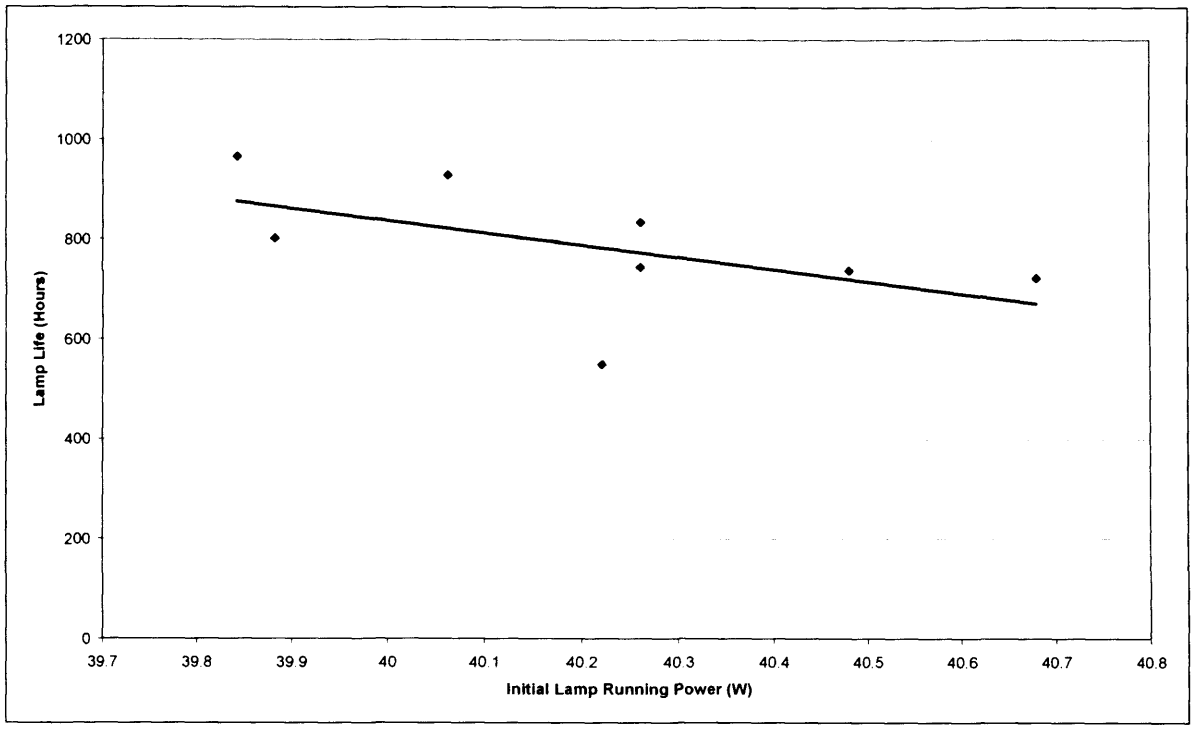


Figure 51:- Initial running power vs life for 40W Tesco lamp samples (lamps 10 – 17).

Considering figures 48 to 50, it can be seen that all three indicate the same trend of decreasing lamp life with running power. This is not surprising as it is common knowledge that running a lamp at an increased voltage and therefore power level will dramatically decrease its life; however it appears that the same principle is also true on a smaller scale when the lamp is manufactured. If the filament is manufactured slightly shorter or of a thicker gauge than usual, then the filament will have a lower resistance than normal and therefore will run at a higher temperature and consequently fail sooner due to the accelerated formation of hot spots and increased filament evaporation.

Based on these results, it appears that it would be possible to approximately predict the lifespan of a regular filament lamp based on the running power of it when it is first put into use. Clearly more experimentation would need to be done to collect data from a large number of samples so that a meaningful relationship could be formed between initial

filament power and expected lamp life. One limitation of this method is that it in effect analyses the filament when the lamp is first put into use; it does not take into account any flaws in the gassing of the lamp, for example, that might lead to an accelerated evaporation of the filament as the lamp ages. This flaw becomes evident when considering the results of lamp number 5.

Lamp number 5 exhibited some extremely strange behaviour. It failed after only 50 minutes of use, and failed while it was already on as opposed to all the other lamps which failed at switch on. A closer look at the power consumption of the lamp during its short life reveals some very interesting behaviour.

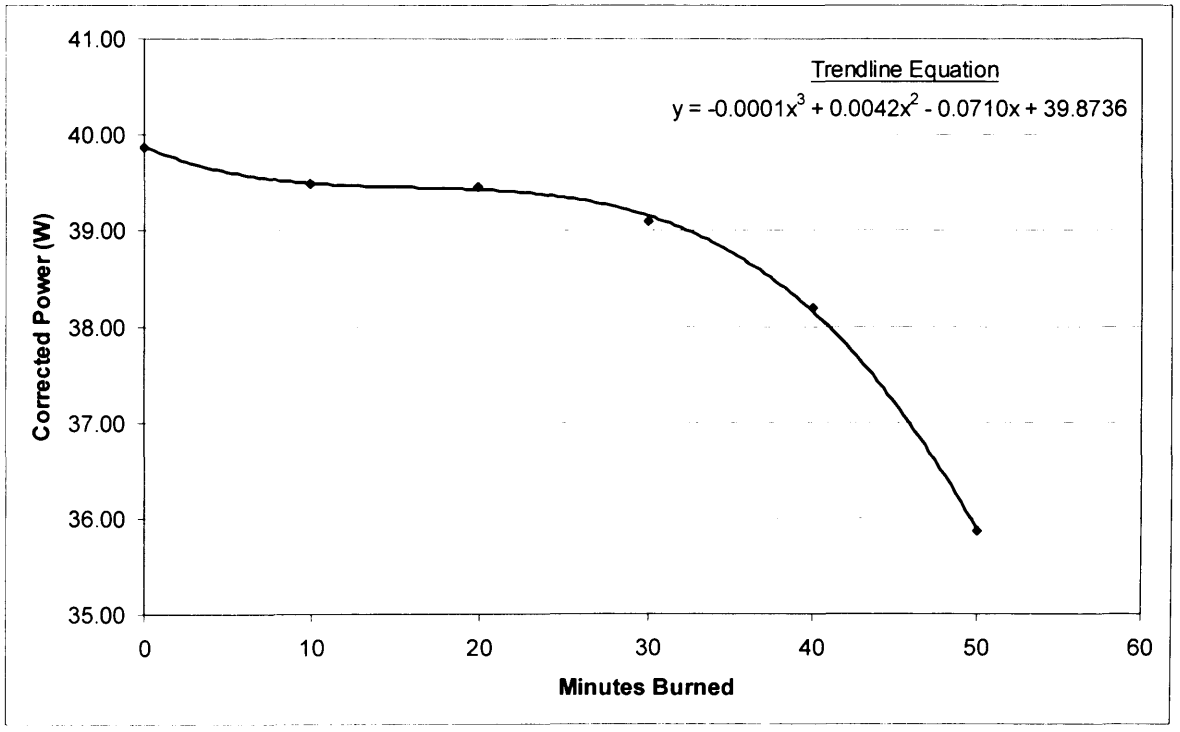


Figure 52:- Voltage compensated power draw of lamp 5 (40W GE lamp sample).

Using the principles established from the previous experiment, the initial power reading of under 40W would suggest that the filament is running at the correct power rating and therefore a lamp life similar to the other lamps should have been expected. However by the time of the next reading ten minutes later there has been a filament power drop of half a Watt. This is not entirely unexpected, as the first reading was taken almost immediately after the lamp was switched on for the very first time, and as mentioned earlier a drop in power was noted for all the lamps when the filament was lit for the very first time; however most of the other lamps only featured a drop of around a quarter of one Watt, so a drop of almost half a Watt was a little concerning.

Twenty minutes into the test, and the next reading indicates that the dropping power has stabilized; however ten minutes later it has started dropping again and it then carries on dropping until it fails sometime between 50 and 60 minutes of use. Further clues as to the cause of failure were evident upon visual inspection of the lamp.

Figure 53:- Photograph of lamp number five after failure.

All of the other lamps exhibited some glass blackening as they aged, as expected, due to the filament evaporating and depositing on the glass envelope. The difference with lamp five was the speed with which the deposits appeared on the envelope and the colour of them. Within the 50 to 60 minutes it was burning, the inside surface of the glass envelope of lamp five became coated with a misty white deposit. Combined with the evidence of the falling electrical power consumption, this suggests that too much oxygen was inside the lamp and the filament was simply burning away with the oxygen and forming tungsten oxide which was being deposited as a white mist on the inside of the envelope. The swirling patterns, evident in the deposit, would have been caused by the thermal convection currents caused by the heat of the filament.

In the opinion of the author, the reason that lamp five behaved differently to the other lamps was that oxygen was inside the lamp; it had either been filled incorrectly during manufacture or a slow leak had developed during use allowing air inside. While measuring the initial power draw may enable lamp life prediction for correctly gassed lamps, it clearly will be of no use on an incorrectly gassed lamp, as it will take time for the effects to become apparent. What is needed is a way of predicting lamp life which can take into account gassing faults as well.

5.2 Detecting Incorrectly Filled Lamps

During a typical gassing, the bulb is evacuated through the exhaust tube and filled with nitrogen/argon gas. The bulb is then partially re-evacuated and the lower end of the tube

heat-sealed.[59] As with any such process, not all the oxygen will be removed from the lamp, so red phosphorous is usually applied to the filament[51] to act as a 'getter'; it is activated by heat and so is fired by the filament heating up for the first time.

During the first few seconds of the lamp being turned on, the tungsten filament will react with the oxygen and burn. This will stop as soon as all the oxygen in the lamp has been used up, either as a result of the action of the getter or due to the burning of the tungsten filament. While the filament is burning, it becomes thinner and the tungsten-oxide which is formed deposits as a white coating on the inside of the glass envelope.

The longer the filament burns the thinner it will become and the more likely it is that the filament will fail prematurely. In the opinion of the author, by measuring the length of the initial filament burn, it should be possible to detect lamps which are likely to fail prematurely due to incorrect gassing.

5.2.1 Procedure

In order to record the start-up power drop, a higher sampling rate was required than the ten minute recording intervals used in the previous experiment. The Voltech PM100 power analyser was again used for this experiment, together with the software developed in chapter 4. The power analyser was connected as shown in figure 54.

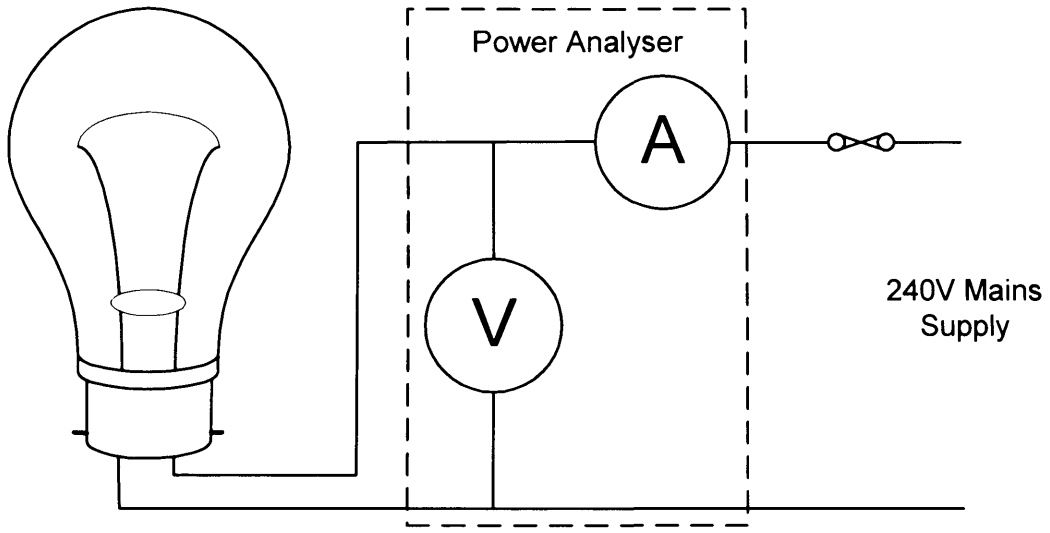


Figure 54:- Incorrect Gassing Detection Experiment Setup

The software was modified so that it still recorded the same data, but took measurements every second. A sampling rate of 0.2 seconds had been tried initially, but it was found that this did not always allow enough time to read the data from the power analyser and record it to the text file before the next reading was due to take place; it was also observed from tests on a sample lamp that recording the data every second was sufficient for the trend being recorded.

Using this experimental setup, the initial burn period of a healthy lamp can be seen in figure 55, which shows the start up current draw of a small envelope 40W Tesco GLS lamp.

5.2.2 Results

The following figures show start-up electrical parameters of a previously unused Tesco 40W small GLS lamp.

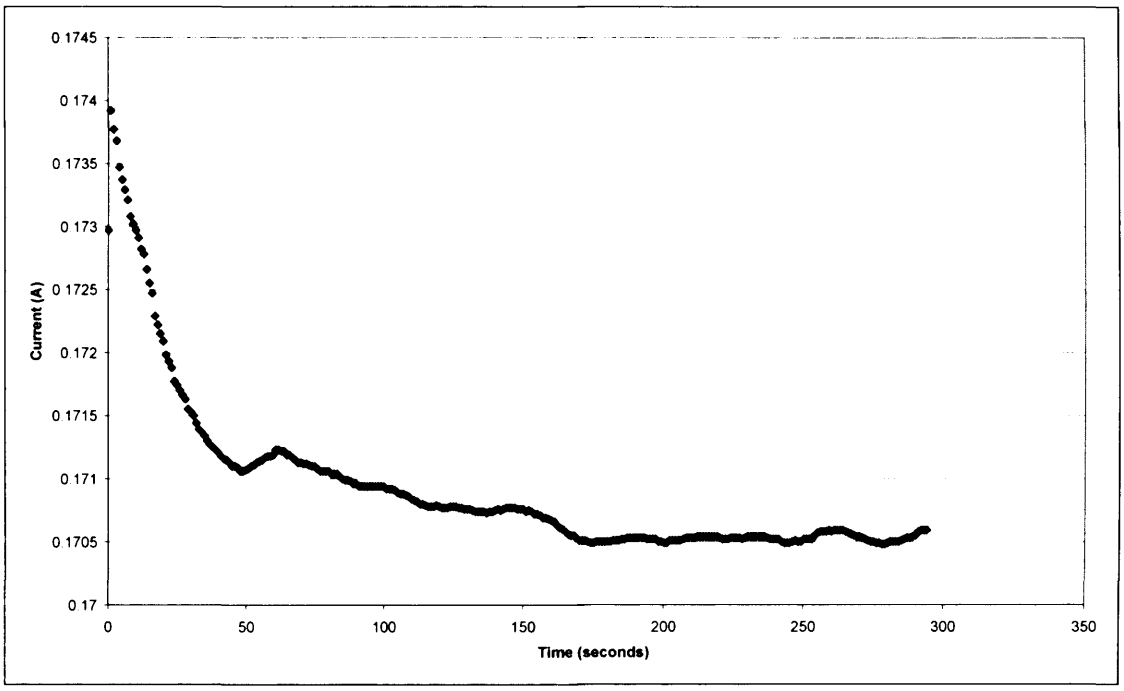


Figure 55:- Current vs. time for the first start-up of a Tesco 40W small GLS lamp.

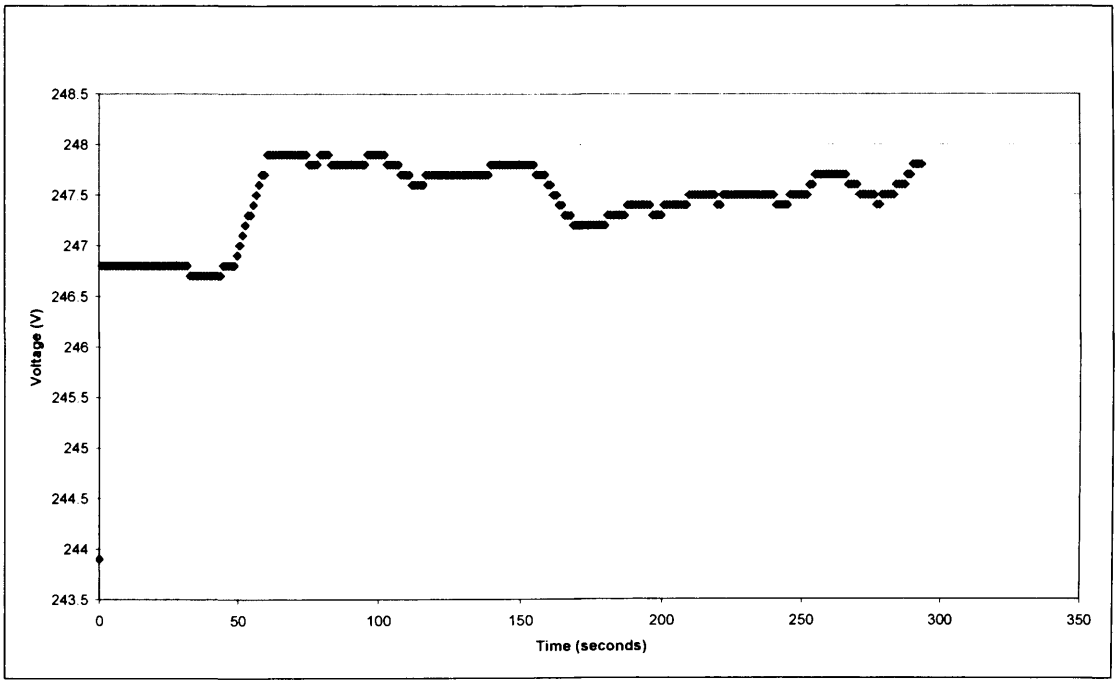


Figure 56:- Voltage vs. time for the first start-up of a Tesco 40W small GLS lamp.

From figure 55, it can be seen that the initial current draw of the lamp drops rapidly when it is first turned on; this drop continues for approximately the first 40 seconds the lamp is on, before the current then stabilizes as all the oxygen within the bulb has been used up.

The small fluctuations in the steady state current value are due to small fluctuations in the mains supply voltage, shown in figure 56. Although the drop in current shown in figure 55 is very small compared to the current draw of the lamp, it does show a clear trend. As the lamp was a healthy one it was expected that the current drop due to the initial power up burn would be very small.

To reduce the effect of voltage fluctuations, the resistance of the lamp was also plotted during start-up, this is shown in figure 57.

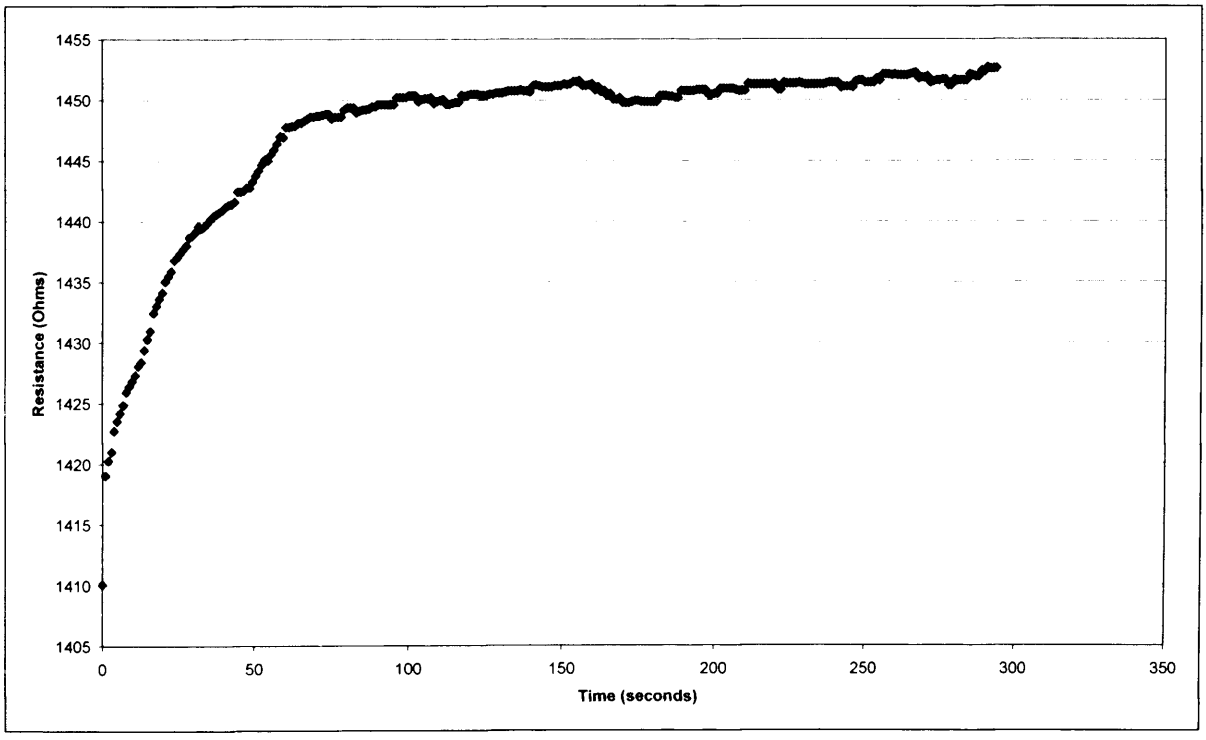


Figure 57:- Resistance vs. time for the first start-up of a Tesco 40W small GLS lamp.

To prove that this drop in lamp current was a characteristic unique to the first start up of the lamp and therefore linked to the burning in of the filament rather than a start up transient as a result of the filament heating up each time, the same lamp was tested again when it was turned on for a second time. The result of this test is shown in figure 58, overlaid with the plot of the first start up for comparison purposes. This time the recorded mains voltage was divided by the recorded lamp current to give the hot resistance of the lamp filament; this was done to minimise the effect of mains voltage fluctuations so that the results could be more easily compared. Clearly the results will still be affected by mains voltage fluctuations to some degree as the lamp resistance is not linearly related to the mains voltage. However as both the mains voltage and lamp current will rise together, calculating the equivalent resistance helps minimise this error.

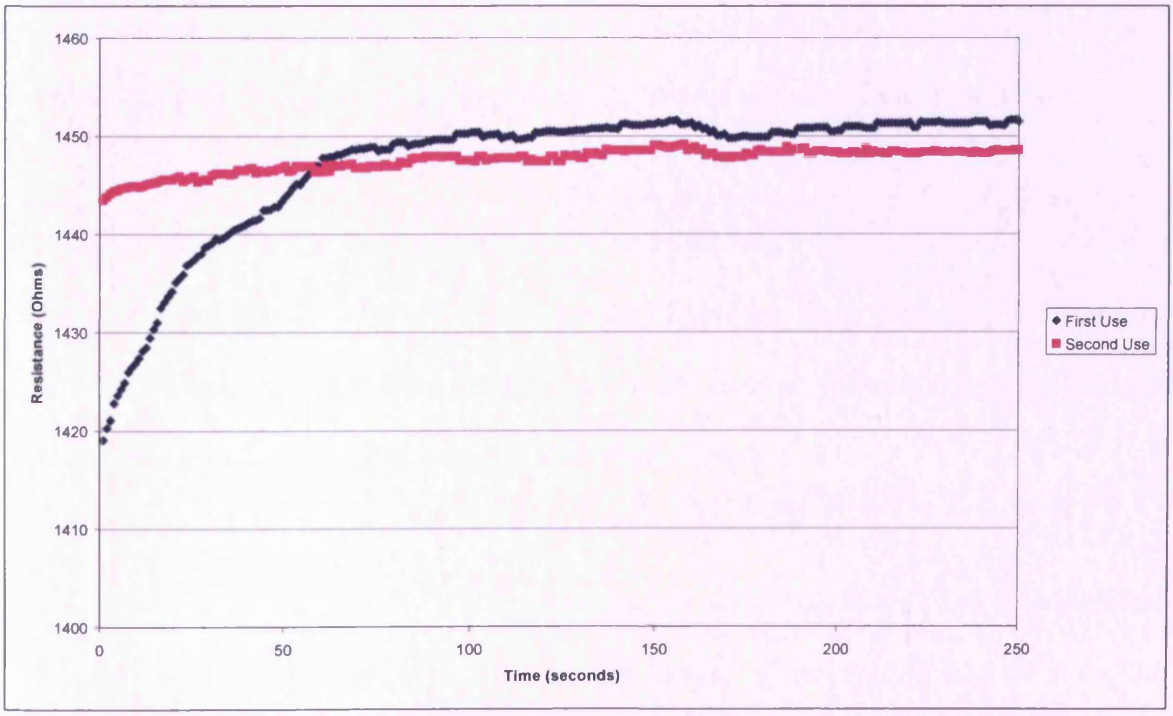


Figure 58:- Filament resistance of a 40W Tesco lamp during first and second use.

Figure 58 shows an increase in lamp resistance of 3Ω after the second switch on, due to the lamp taking time to reach a steady temperature. However it is significantly smaller than the resistance increase of 31Ω during the first use of the lamp. This shows that a significant change is occurring within the lamp when it is powered up for the first time. Monitoring how long the filament resistance keeps rising when the lamp is first powered will, in the opinion of the author, provide an indication of the damage the filament has suffered during its first burn and for incorrectly gassed lamps may provide a usable indicator of how long the lamp will last as a result of this damage.

In order to conduct the necessary experiments and measure the start up current of incorrectly gassed lamps, a supply of faulty lamps was required. Since identifying faulty lamps would require testing every lamp and waiting to see if it was a faulty one before using the results, this was ruled out as a way of obtaining results in a reasonable time scale. Instead, a method of simulating such affects was devised and developed as follows:

Brand new bulbs were purchased and a small 1mm hole was drilled in the neck of the lamp using a tungsten carbide drill. Air then quickly filled the lamp, replacing the previous contents; however it proved difficult to evacuate the lamp and then successfully seal the hole again and it was decided to adopt a different approach.

Being an inert gas, Argon will not react with the hot tungsten filament and it is therefore commonly used as a filling gas in incandescent lamps. Thus after drilling the hole, the lamp was filled with Argon welding gas (plus an inevitable small amount of air which remained inside). The exact composition of the gas mixture inside the lamp was not

known and was deemed not to be important as this would not be known when testing a real lamp either. The hole was finally sealed with 60 second curing epoxy resin glue.

In all a total of 25 lamps were filled and successfully tested; many were wasted due to the glass cracking as a result of the hole being drilled, or the Argon gas escaping completely as the hole was sealed. Of the 25 lamps successfully filled, 14 were 40W lamps and 11 were 60W lamps (it had been intended to have equal numbers of both).

Each of the faulty lamps was subjected to the same test as the healthy lamp shown in figure 57. Resistance plots for all 25 faulty lamps can be seen in appendix C. A plot of one of the faulty lamps is shown in figure 59. Attention is drawn to the change of axis scale, which confirms the significantly larger increase in the resistance of the faulty lamp.

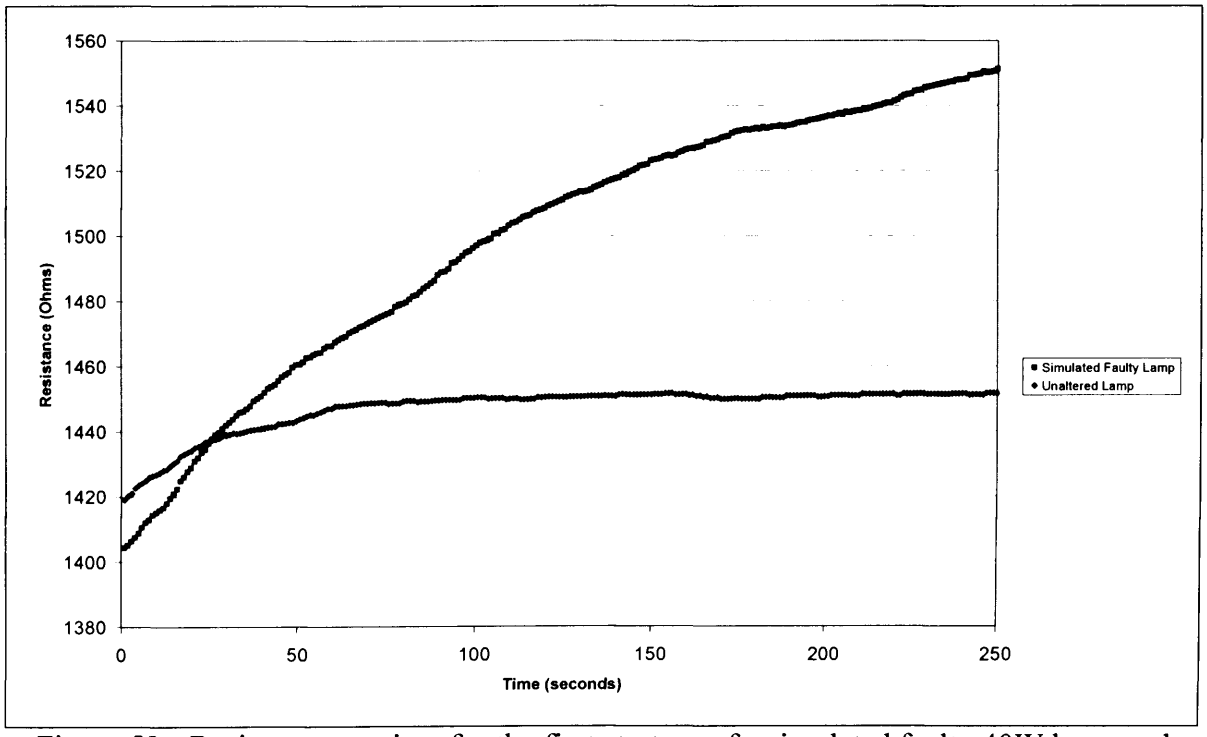


Figure 59:- Resistance vs. time for the first start-up of a simulated faulty 40W lamp and an unaltered 40W lamp

When comparing the example results shown in figure 59 there are two significant differences in the resistance profiles. The rate of increase in lamp resistance at switch on is significantly larger than for the unaltered lamp and the erosion of the filament also continued for longer than the typical 60 seconds observed for an unaltered lamp. Indeed the latter effect continued and did not stabilise before the lamp failed. The particular lamp shown in the plot lasted less than 10 minutes. Most of the other lamps tested lasted longer and did stabilise after the initial filament burning period, as shown in figure 60; the longest burned continuously for 33.5 hours.

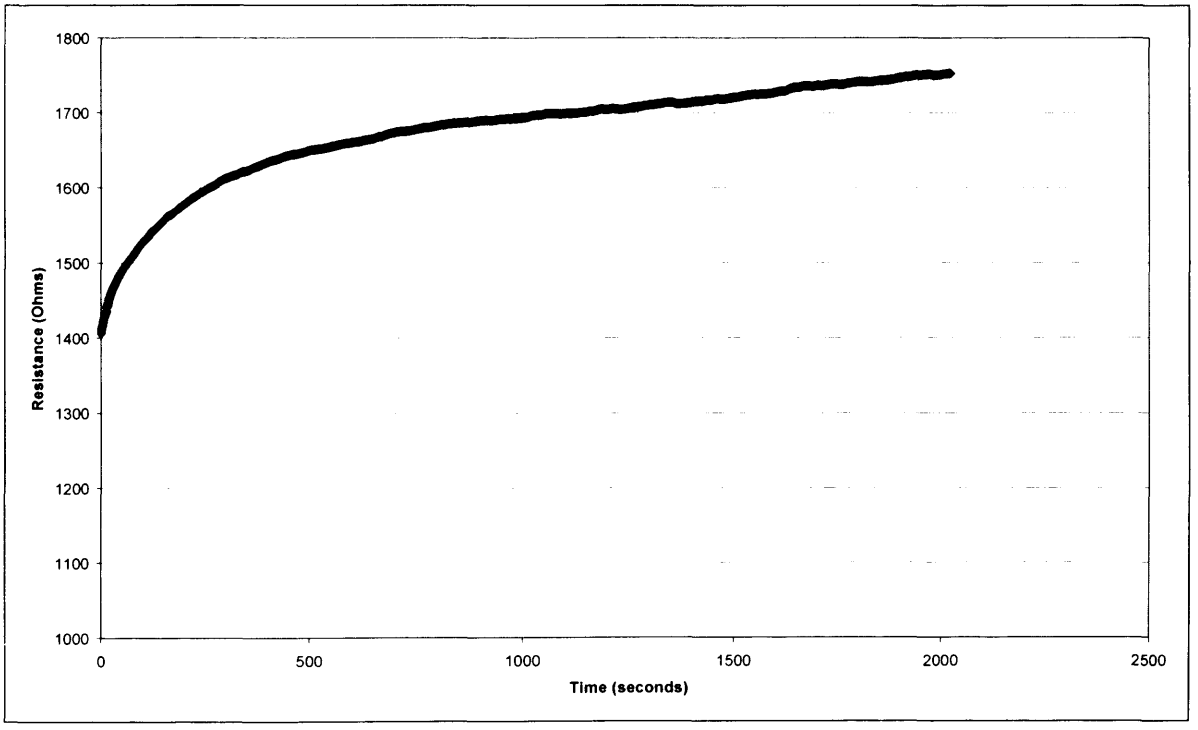


Figure 60:- Start up resistance of a 40W GLS lamp when filled with Air and Argon

The results obtained were from lamps filled with a mixture of Air and Argon; one consequence of this was that the filament evaporated at a quicker rate than usual, which could be observed by watching the envelope of the lamp. Initially when the lamp was first

turned on a white deposit formed at the top of the glass envelope, which is due to the filament burning as a result of the oxygen from the atmosphere present inside the envelope and caused the initial sharp increase in resistance. After the oxygen had been used up, the formation of the white deposit stopped and a dark grey deposit started forming due to the filament evaporating.

Normally there is a very small amount of Argon and nitrogen in the lamp envelope, as the lamp envelope is mostly evacuated; however as the lamps in this experiment were filled with air and argon there was no vacuum at all. In the opinion of the author, the grey deposit was therefore probably due to increased thermal currents within the lamp which lead to the increased rate of filament evaporation; this theory was reinforced by the swirling patterns evident in the black deposit. This pattern of rapid burning followed by slow evaporation also explains the trend visible in the lamp resistance plot shown in figure 60, with an initial rapid increase in filament resistance when it is turned on and then a slow linear rise during the remaining time that the lamp is burning for.

As a potential diagnostic and in order to identify any relationship between the initial resistance increase and the life of the faulty lamp, the change in resistance between when the lamp was first turned on and after 30 seconds of use ($R_{30} - R_0$) was calculated for the 25 data sets. These values were then correlated to the lamp survival times and the results are presented in table 7 and the subsequent figures, presented separately for the 40W and 60W lamps.

Lamp No.	$R_{30} - R_0$ (Ω)	Life (Seconds)
40 Watt Lamps		
F17 (40W)	N/A	7
F19 (40W)	N/A	28
F14 (40W)	687.74	35
F2 (40W)	426.58	59
F12 (40W)	188.46	127
F20 (40W)	141.82	493
F1 (40W)	175	564
F11 (40W)	121.37	578
F18 (40W)	55.72	654
F16 (40W)	127.24	818
F15 (40W)	106.60	901
F4 (40W)	149.10	1474
F13 (40W)	60.54	2023
F3 (40W)	38.49	2497
60 Watt Lamps		
F24 (60W)	618.81	31
F21 (60W)	45.65	449
F25 (60W)	33.45	548
F23 (60W)	75.20	851
F6 (60W)	23.45	1045
F22 (60W)	72.68	1355
F5 (60W)	44.43	2038
F7 (60W)	28.16	5031
F8 (60W)	22.82	8207
F10 (60W)	21.97	43440
F9 (60W)	24.24	120640

Table 7:- Results for the 25 simulated faulty lamps

Looking at the results, it can be seen that some of the 60W lamps lasted significantly longer than the 40W lamps, this is believed to be due to a quirk of the filling process.

Figure 61 shows the plot for 12 of the 40W Argon and Air filled lamps. (The two lamps which didn't last 30 seconds were omitted the results).

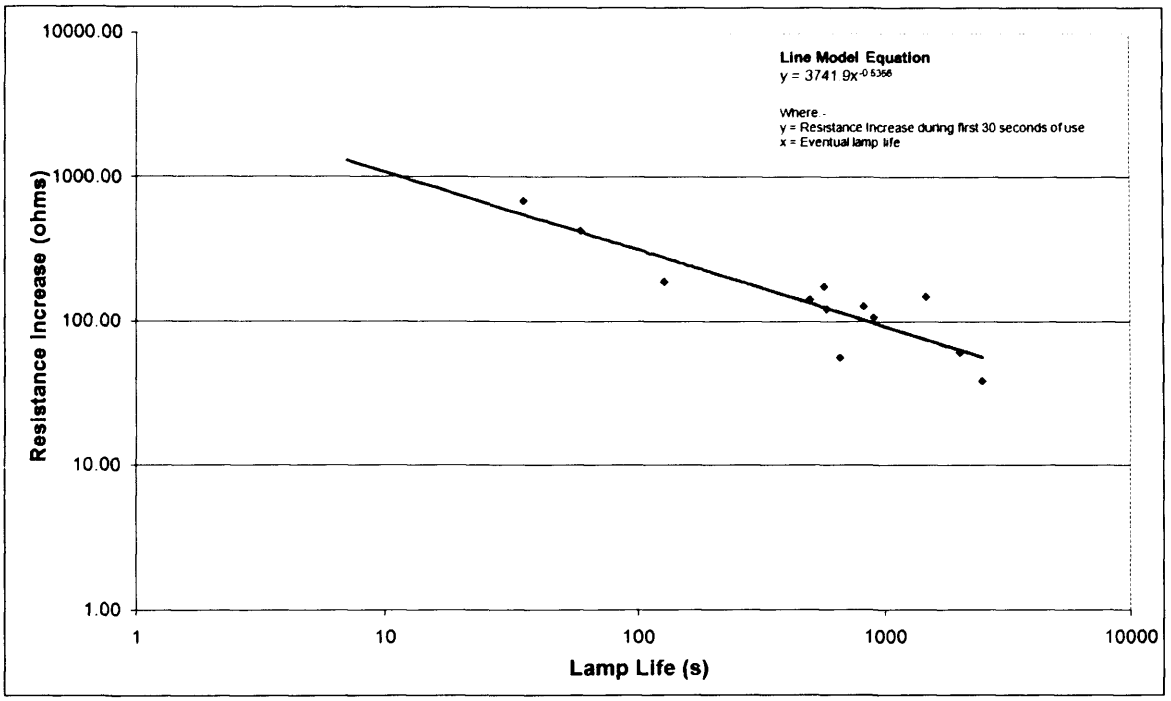


Figure 61:- 40W lamp ($R_{30} - R_0$) values vs. how long the lamp lasted

From figure 61, it can be seen that there is a strong correlation between the calculated ($R_{30} - R_0$) values and the filament survival times.

Figure 62 shows the same information as figure 61, except this time the experiments were carried out on the 11 60W lamps. Once again the trend is similar to that obtained for the 40W lamps, except the resistance increases are lower due to the higher power lamps.

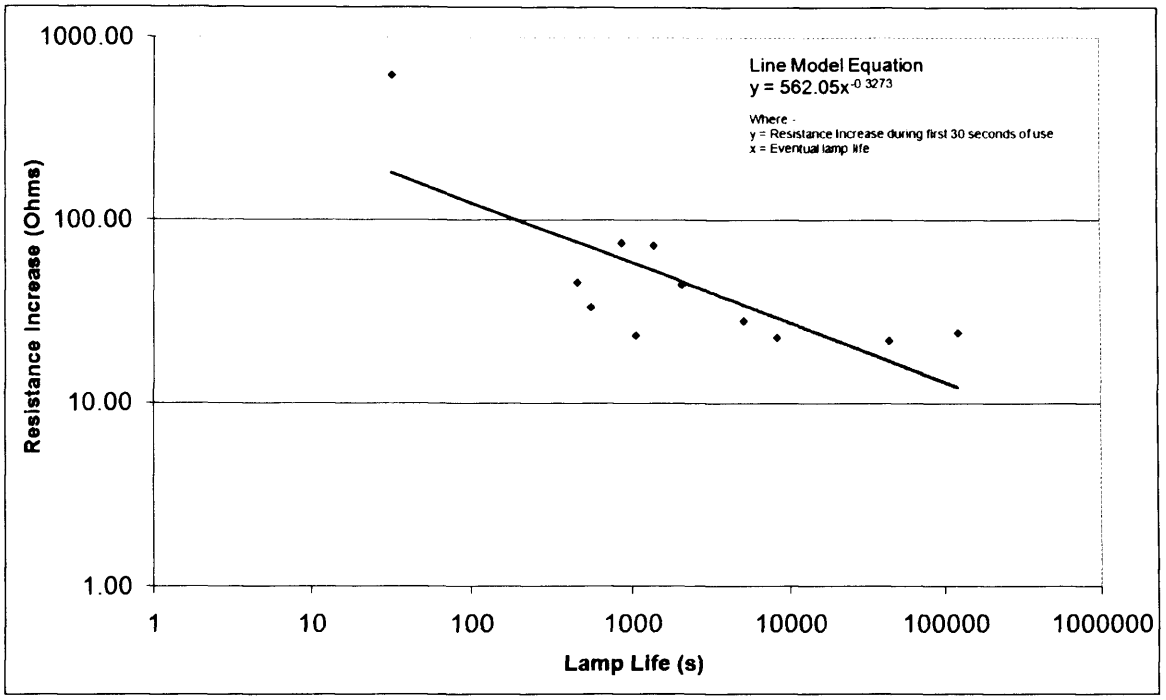


Figure 62:- 60W lamp ($R_{30} - R_0$) values vs. how long the lamp lasted

The ($R_{30} - R_0$) value was also recorded for the second batch of healthy 40W lamps (lamps 10-17) used in the continuous monitoring experiment in section 5.1.8. The values obtained were as follows:-

Lamp No.	$R_{30} - R_0$ (Ω)	Life (Hours)
10	27.68	929
11	18.92	802
12	18.03	745
13	20.34	835
14	23.76	723
15	19.39	966
16	16.38	738
17	20.70	550

Table 8:- Results for a batch of 8 unaltered lamps.

There is no noticeable correlation in the results between how long the healthy lamps lasted and their value of ($R_{30} - R_0$); however the purpose of the test was to detect incorrectly gassed lamps that are likely to fail early. The healthy lamps, all of which lasted at least 500 hours, had a ($R_{30} - R_0$) value of less than 28Ω, while the simulated faulty 40W lamps all had readings in excess of 38Ω, with the majority of the lamps having readings above 100Ω. This shows that the technique is able to distinguish incorrectly filled lamps from normal healthy lamps

Considering again how most lamps are filled, the bulb is evacuated through the exhaust tube and filled with nitrogen/argon gas. The bulb is then partially re-evacuated and the lower end of the tube heat-sealed.[59] It becomes evident that there are two problems that could occur: the nitrogen/argon gas may not be added correctly or the lamp may not be evacuated fully. It is clear that this testing technique has the potential to diagnose both of these problems, as a large initial resistance increase indicates too much oxygen in the lamp as a result of poor evacuation and the rate of resistance increase after this initial period will vary depending on the amount of Argon/Nitrogen left in the lamp, due to evaporation and thermal convection. To develop this technique further a large scale trial is required so that the behaviour of normal and genuinely faulty lamps can be recorded and thresholds determined for reliably identifying the faulty lamps.

5.3 Summary

In order to assess lamp condition, lamp power or current and lamp voltage need to be monitored. The key phases in the life of the lamp that need careful monitoring (as they provide significant information on the condition of the lamp) are as follows:

The rate of power drop during the first few seconds use of a new lamp signifies the quality of the gassing of the lamp. The results of the experiments reported in this chapter show that the quicker the drop in lamp power and the longer it continues for, the more the filament is eroded, which means the lamp is more likely to fail prematurely.

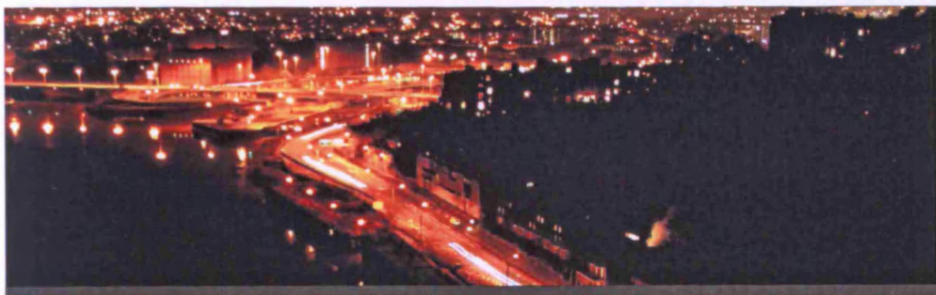
Once the lamp has stabilised after its initial burn-in period (no more than a few minutes), the initial running power of the lamp needs to be measured as it provides crucial information on how long the lamp is likely to last. The continuous monitoring experiments at the beginning of this chapter showed the significance of the initial running power on how long the lamp lasted; lamp three which had a running power of 23W, lasted 7932 hours, which was just over nine times longer than lamp four, which had a running power of 25.97W and lasted just 847 hours. Since a variation of three Watts is a relatively small signal to noise ratio, it is essential that the mains voltage is monitored as well, so that it can be used to compensate the power readings. It is also important that sufficient readings are taken and averaged to minimise any error in the reading.

Finally, the descent profile of the lamp power needs to be monitored during the entire life of the lamp. This is for two reasons: firstly, a fault may develop in the lamp while it is in use, such as a small crack forming in the glass envelope that causes a slow gas leak, which could be detected as an increase in the rate of the power drop of the lamp; secondly, the experiments discussed in chapter four showed that lamp power decreases linearly with age although at a different rate for each lamp. By determining the normal rate of power drop for the lamp being monitored during its early life, the lamp's age can subsequently be determined by measuring its power and comparing it with what it was when the lamp was new. This is important as it means the age can be monitored without resorting to measuring the time it has been switched on for.

Another possible benefit of measuring the age of a lamp by its declining power draw is that the declining power draw should decrease more slowly if the lamp is not run at full power, due to the filament running at a lower temperature and therefore evaporating more slowly. Whereas an hour meter becomes inaccurate if the lamp is not run at full power, such as when it is dimmed or run from a slightly lower voltage than it is designed for, this method should automatically compensate. Clearly, further experimentation is required to prove this.

Whilst the experiments have clearly shown that monitoring filament lamp condition more accurately by monitoring the lamp's electrical characteristics during use is possible, more data needs to be gathered in order to establish normal lamp behaviour patterns. A large scale trial of a few hundred lamps is now required in order to gather enough data to reliably predict lamp behaviour. The affect of other factors also needs to be researched,

such as: running the lamps at lower power levels; using different switching cycles and how orientation of the lamp whilst running may affect its behaviour. Unfortunately, testing hundreds of lamps on an 8 lamp test rig would take a long time and was outside the timeframe of this work. In order to be practical the implementation technology needs to be cost effective and compact, a topic which is discussed in greater depth in chapter 8.



6

UV Discharge Lamps - Detecting Failure

6.1 Introduction

UV sterilisation lamps present particular challenges for monitoring systems. The lamp is housed inside a close fitting quartz glass sleeve, which is surrounded by flowing water. As the lamp is effectively enclosed, monitoring it by electrical characteristics alone is particularly attractive.

Figure 63 shows the inside of a typical stainless steel UV water steriliser, the model shown also features an automatic sleeve cleaning mechanism, which can be seen to the left of the tube.

Figure 63:- Inside of a UV water treatment chamber.[60]

6.2 Traditional Monitoring Techniques

Traditional techniques for monitoring low pressure discharge lamps such as UV sterilisation lamps and fluorescent lamps are broadly similar to those used for monitoring filament lamps and are described in the following sections.

6.2.1 Hour Meters & Scheduled Replacement

Hour meters monitor how long the lamp has been burning and the lamp is then replaced after a specified number of hours of use. This technique is often used for germicidal lamp installations, where lamp failure can present a significant hazard. Drawbacks with this technique are that it does not take into account the number of lamp starts, which can have a major effect on its life expectancy.[61] In addition, each lamp will last different amounts of time, so to avoid the risk of it failing the lamp is usually changed well before its rated life. As most germicidal lamps are left burning continuously, smaller installations without hour meters usually have their lamp changed every year as a matter of routine.

Changing lamps well before they are likely to fail may seem a safe option for guaranteeing continued operation; however not every lamp will last as long as its rated design life, so some may well fail without warning before they are routinely changed. Another problem with relying on routine replacement is that the majority of lamps will be replaced long before they need to be. As each lamp contains mercury, it does not make sense environmentally to scrap lamps before they need replacement. UV sterilisation lamps are also very expensive compared to regular fluorescent lamps (a typical 55W lamp is around

£50+vat), so changing them before they need replacing is an expensive way of trying to ensure their continued operation.

6.2.2 Failure Detection

Another common approach to monitoring lamp failure is to install basic monitoring circuitry to alert the user when a lamp has failed. This can take a variety of different forms. On smaller lamps (around 15W) that use 240V magnetic ballasts, the voltage across the lamp when it has struck drops from line voltage to around 40V, which can be utilised by simply connecting a neon lamp (with series resistor) across the lamp. The warning lamp will illuminate when the UV lamp is not lit, a diagram of which is shown in figure 64.

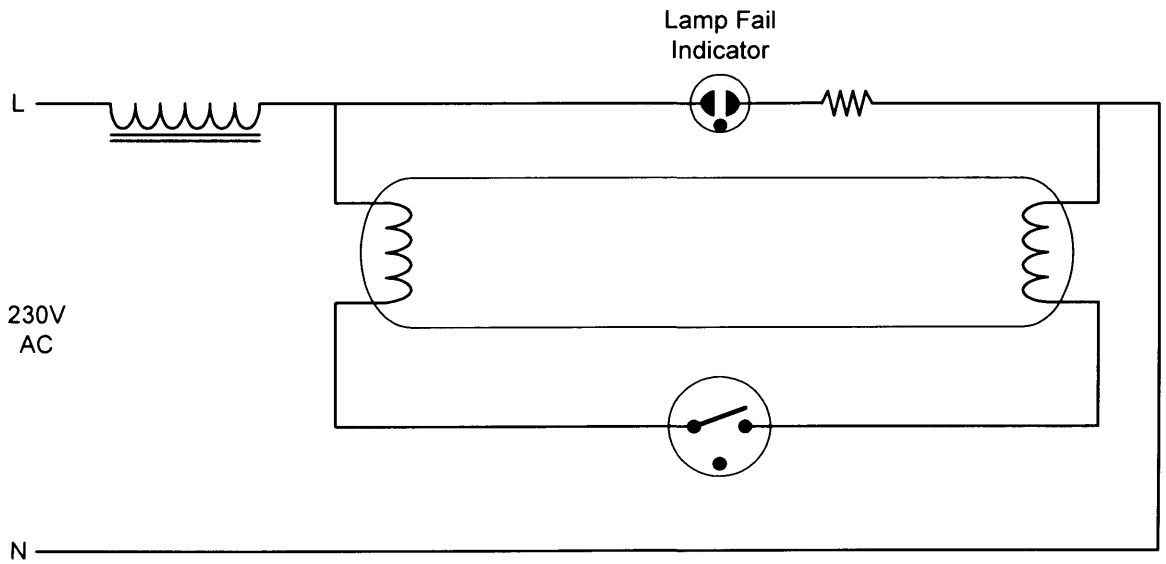


Figure 64:- Circuit diagram of a very simple lamp monitoring setup using a neon lamp.

A more robust approach that is commonly used is to apply current monitoring to the lamp so that if no current flows through the lamp then a buzzer sounds to warn the user that the

lamp has failed. A typical circuit diagram for acheiving this is shown in figure 65, the overall component count is low and, although more complicated than using a simple neon lamp as in the previous method, the completed board would cost no more than a few pounds to build.

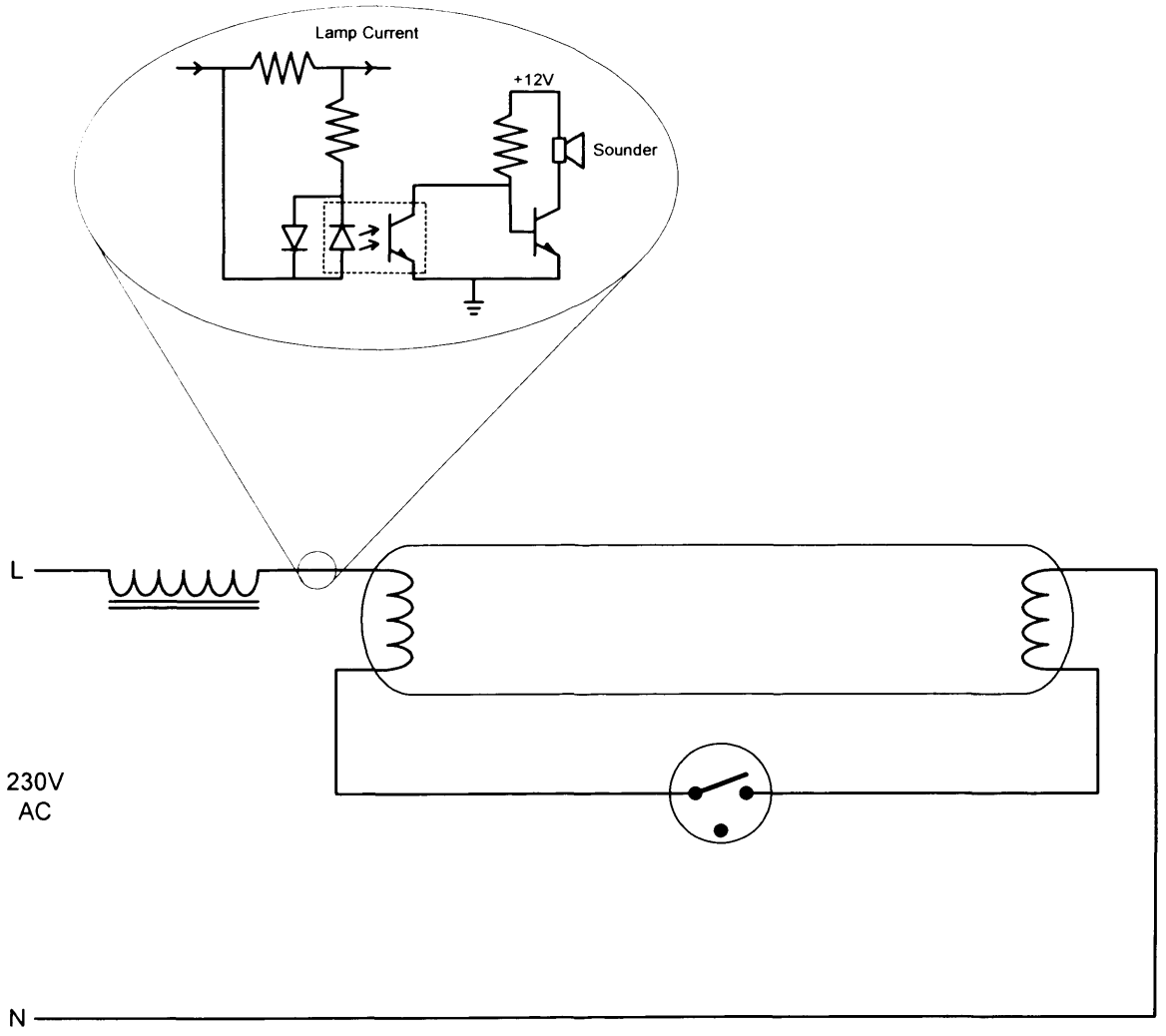


Figure 65:- Circuit diagram of a typical lamp failure detector.

By its nature, any failure detection system will only work after the lamp has failed. To make the system fail-safe, the failure detection can be linked to an electric solenoid valve

to automatically cut off the water supply until the lamp is replaced. Although this prevents untreated water passing through, it leaves the user without water while a new lamp is sourced, which in most cases is not an acceptable solution.

6.2.3 Light Level Monitoring

The most robust way to verify UV light output is to measure it using a UV meter; however due to the hazardous nature of UV-C radiation to humans and animals, the lamps are always enclosed within the water treatment unit. This means that the only way to safely get a consistent reading from a lamp when it is in use is to install a sensor within the lamp housing and then link this sensor to the monitoring system being used. A number of sensors designed for this task are available, an example of one supplied by Cole Parmer is shown in figure 66. Although systems using sensors such as this are more robust as they measure the actual light level delivered rather than some other parameter that is related to light output, they have the significant disadvantage of being very expensive to implement. The UV-C sensor alone costs £192, which for small installations would be completely impractical. When purchased with a corresponding lamp monitoring unit the total price rises to £373, which for most domestic installations would be more than the UV disinfection unit.

Figure 66:- Cole Parmer UV Disinfection Sensor[62]

6.3 Failure Modes and Potential Monitoring Techniques

As previously discussed in chapter one, a typical UV lamp consists of two heater filaments enclosed at either end of a quartz glass tube. The tube is usually filled with argon gas and a very small amount of mercury which turns into mercury vapour when the lamp warms up. To light the lamp, the two heaters are usually warmed up for a few seconds before a high voltage is applied across the length of the tube to start the discharge; once struck the current through the tube has to be limited.

6.3.1 Mechanical

Any leak in the quartz glass envelope will quickly lead to the gas within the lamp escaping and air being allowed in, which will cause the lamp to immediately fail, if the heaters are powered separately then they will burn out and leave a white deposit on the glass; if the heaters are kept hot by only the discharge then the discharge will just cease and the lamp will be unable to strike again.

There are several reasons why a leak may develop in the quartz glass envelope. Minute flaws in the glass wall or around the wire seals may cause a stress fracture to develop when the lamp warms up or cools down; alternatively thermal shock, such as water dripping on the lamp, may cause the glass to contract suddenly and fracture; physical handling may also lead to breakage of the glass envelope. Whatever the cause, detection of a mechanical failure is virtually impossible for any monitoring system, the only exception to this being the development of a slow leak where some changes in the operating characteristics of the lamp may be evident before it fails completely.

6.3.2 Electrode Failure

In a paper titled "Reducing Barriers to Use of High Efficiency Lighting Systems"[63] by the Lighting Research Centre, it was stated that "The failure of fluorescent lamps is caused mainly by the loss of the electron emissive coating of the lamp electrodes. Under certain circumstances, such as high frequency operation and frequent starting on instant start

ballasts ("cold ignition"), fracture of the tungsten coil is also observed, which causes the lamp to fail."[63]

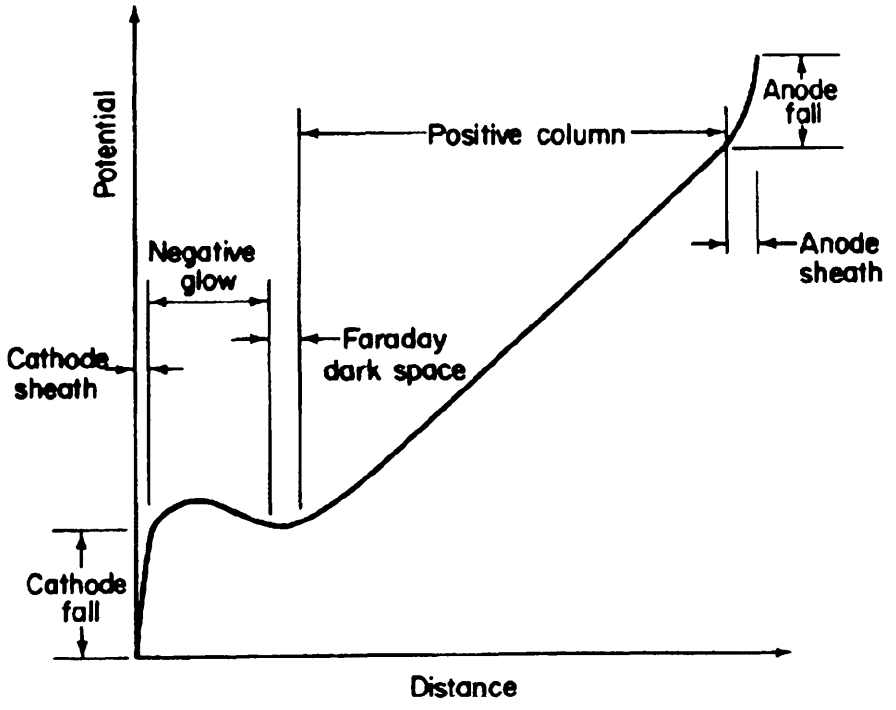


Figure 67:- A sketch of the main discharge regions[64]

The discharge within the lamp is made up of a number of different regions, as shown in figure 67. The cathode sheath and negative glow regions generate the supply of electrons for the positive column and hence sustain the discharge. The length of this region is determined mainly by the gas pressure, and is independent of the length of the tube. The positive column is the main part of the discharge and generates the majority of the light output. The anode sheath completes the path between the positive column and the anode, and is typically significantly shorter than the cathode sheath and negative glow regions.

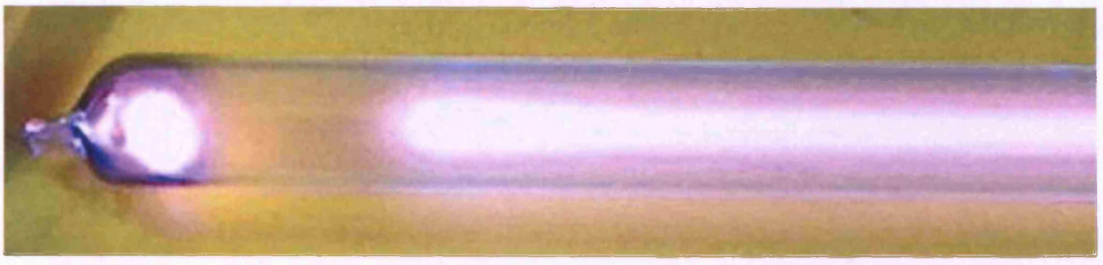


Figure 68:- Negative glow and Faraday dark space regions in a Geissler tube[65]

Figure 68 shows the negative glow and Faraday dark space regions in a DC driven Geissler tube. These regions are not visible in an everyday fluorescent lamp as the gas pressure is higher and so the Faraday Dark Space is significantly smaller; also most lamps are operated on AC, so the anode and cathode reverse fifty times a second, which means the light generated when it is acting as an anode masks the dark space present when it is acting as a cathode.

An important factor in lamp operation and life is the electrode temperature. Most electrons emitted by the cathode are as a result of thermionic emission, where thermally excited electrons break free from the material. Fluorescent lamp electrode filaments are coated with an emission mix, made from calcium, barium, and strontium oxides to improve thermionic emission. For coated filaments, temperatures of about 900˚C are high enough to create thermionic emission of electrons sufficient to maintain the discharge current. Without the emissive coating, thermionic emission is insufficient for the discharge current; maintaining the discharge by increasing the voltage across the tube would then lead to the destruction of the electrode and lamp failure.[63]

The reason for the loss of electron emissive coating can be divided into two causes, evaporation and erosion due to sputtering. A very high electrode temperature (greater than 1000°C) reduces lamp life due to evaporation of the emitting material, while a low electrode temperature (less than 700°C) reduces lamp life due to erosion of the emitting material by sputtering.[63] Electrode temperature is critical to the life expectancy of the lamp.

Cathode fall voltage is the drop in electric potential from the cathode surface to the end of the cathode sheath region, over a distance of approximately 0.1mm. It serves to accelerate the electrons emitted from the electrode toward the lamp arc stream and enables ion generation when these accelerated electrons collide with the mercury and argon atoms in the gas atmosphere of the lamp. Once produced, the ions are accelerated by the cathode fall voltage and strike the cathode. The ion bombardment heats the cathode surface, raising the electrode temperature and increasing the emission of electrons. [63]

The cathode fall voltage is an important parameter of a discharge as it indicates the level of electrode sputtering. A high cathode fall voltage causes ions to strike the cathode so forcefully that significant sputtering of the emissive coating occurs. Alternately, a low cathode fall voltage indicates an abundance of free electrons, which implies that the electrode is operating at an excessively high temperature and consequently lamp life will be drastically shortened as the result of a high electrode-evaporation rate.[63]

Based on these observations, a cathode with a worn or damaged emissive coating should correspond to an increased cathode fall voltage. A typical acceptable range of peak

cathode fall voltage is between 11 and 14.5V[63], although 7 to 18V is not uncommon, which means a change in cathode fall voltage should be detectable as a change in the normal operating voltage of the lamp.

6.3.3 Mercury Loss

In a low pressure UV germicidal lamp, the UV output is generated as a result of collisions between electrons and mercury atoms. If mercury atoms are not present for the electrons to colide with then a discharge still takes place through the argon gas, but very little (if any) UV light is given out.

As the quartz tube is completely sealed, it would seem that the mercury could not escape. However, over time the mercury can bond with other materials in the lamp, particularly evaporated electrode material that gets deposited on the lamp wall, leaving less free mercury to generate the UV output. Due to environmental reasons, some new lamp designs are manufactured with near the minimum amount of mercury needed for an efficient discharge. Mercury absorption is not an operational problem in lamps with excess mercury added because the lamp electrodes limit life before the excess runs out; however, for lamps with marginal amounts of mercury dosing and severely sputtered and/or evaporated electrodes, different end-of-life behaviour has been reported.[63]

Mercury loss represents a very dangerous mode of failure for germicidal lamps, as the argon within the lamp will maintain the discharge through the lamp even with no mercury present. To the user and any basic electrical lamp monitoring equipment, it appears that

the lamp is operating satisfactorily; the lamp will even give off the same visible blueish glow as normal, but the output of invisible UV radiation which kills the harmful bacteria in the water may be significantly reduced.

A loss of mercury atoms should coincide with a reduction in the number of collisions taking place as the electrons pass through the tube, meaning that the electrons should travel further through the gas before colliding with a mercury atom.

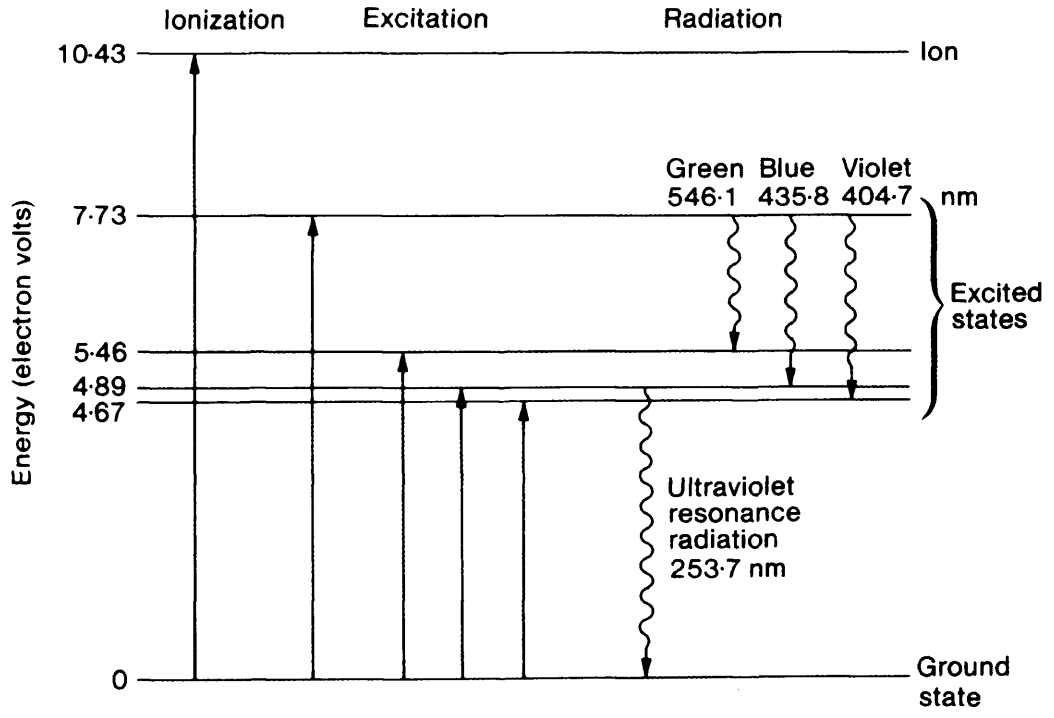
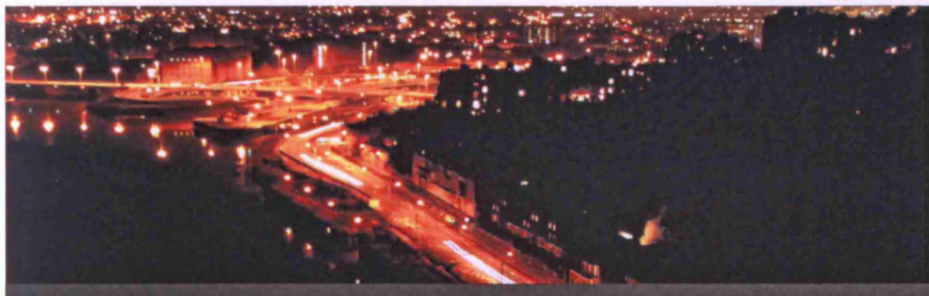


Figure 69:- Simplified transition diagram for mercury[66]

The transition diagram in figure 69 shows that to generate UV light, the incident electron needs to have a kinetic energy of at least 4.89eV and to ionise the atom it has to have an energy of at least 10.43eV. Since on average the electrons will be travelling further before

colliding with an atom, they should have gained more energy. This means the likelihood of ionising a mercury atom rather than just exciting it would be higher. Ionising more mercury atoms would mean an increase in electron density through the tube which means an increase in lamp current. The control gear would then decrease the lamp voltage to control the lamp current. In theory then, a loss of mercury should result in a decrease in lamp operating voltage as well as a decrease in light output.

The next chapter discusses the different experiments that were conducted to try and identify ways of monitoring for lamp electrode erosion and mercury loss.



7

UV Discharge Lamp Experiments

7.1 Steady State Electrical Parameter Monitoring

An initial set of experiments were performed in which the voltage across and current through 15 different lamps were recorded, together with the measured UV output of each lamp to determine if the relationship between the measurements varied with lamps of different ages. The procedure and results of these experiments are discussed next.

7.1.1 Procedure

All 15 lamps were made by Philips and were type G30T8. Two of the lamps were brand new and had never been used before, the remaining 13 were obtained from a local water treatment company and were all removed from working installations during routine services and would have been run continuous for at least one year (approximately 9000 hours) so were all beyond their stated useful design life of 8000 hours.

A test rig was constructed using a tube to cover the lamp, with a hole drilled in the middle of it to allow light onto the UV sensor which was secured over the hole in the tube. The lamp was then connected to a standard 30W magnetic ballast and wired up in the configuration shown in figure 70. Rather than use a standard switched starter, a push switch was used instead to allow for controllable starting conditions and to ensure that any current that might otherwise flow through the gas in the starter switch could not corrupt the readings for the lamp.

The Voltech PM100 power analyser which was used for the previous filament lamp experiments was used for the voltmeter and ammeter shown in the diagram. As well as

measuring voltage and current it also gave the power factor and power reading for the lamp being tested.

To measure the UV output of the lamp, a UVX radiometer made by UV Products Ltd was used together with the UVX-25 sensor which is calibrated at a wavelength of 254nm and is designed to function in the 250-290nm wave band (upper region of UV-C band) at intensities upto $20mW/cm^2$[67]. The lamps being tested produce peak output at 254nm.

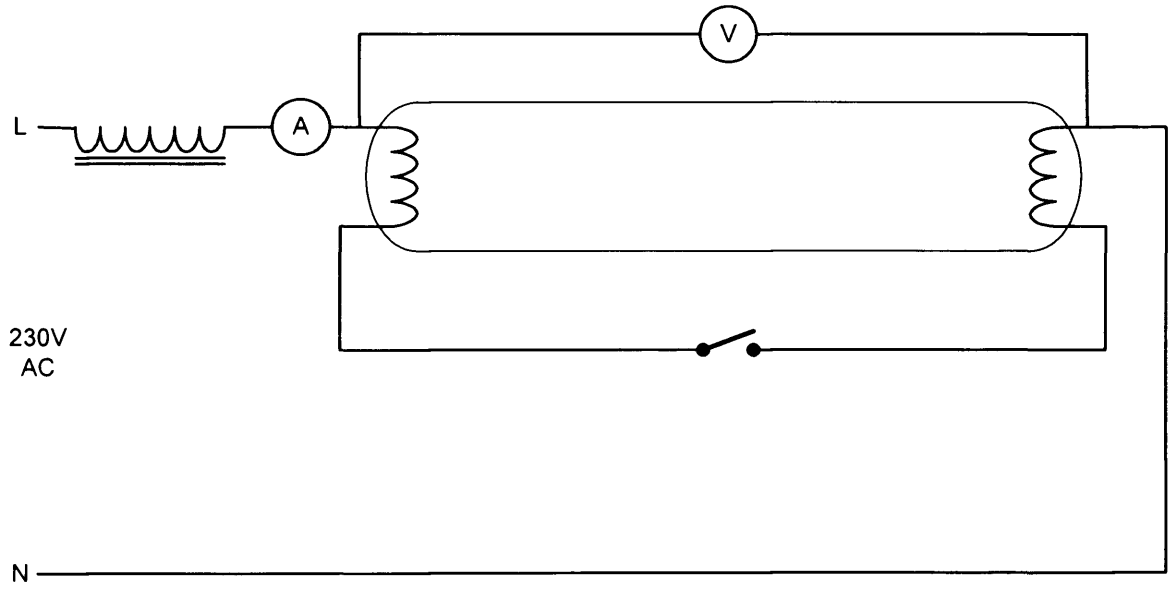


Figure 70:- UV Lamp Test Circuit Diagram

Each lamp was connected to the test circuit shown in figure 70 and run for 2 hours to allow time for it to warm up and stabilize; at the end of this period, the lamp voltage, current, power factor and real power were recorded together with the UV light output reading obtained from the radiometer.

7.1.2 Results

The following figures show one of the measured lamp parameters plotted against the UV light output to identify if there is any relationship between any of the parameters and light output. The values used to generate the plots are presented in tabular form in appendix D.

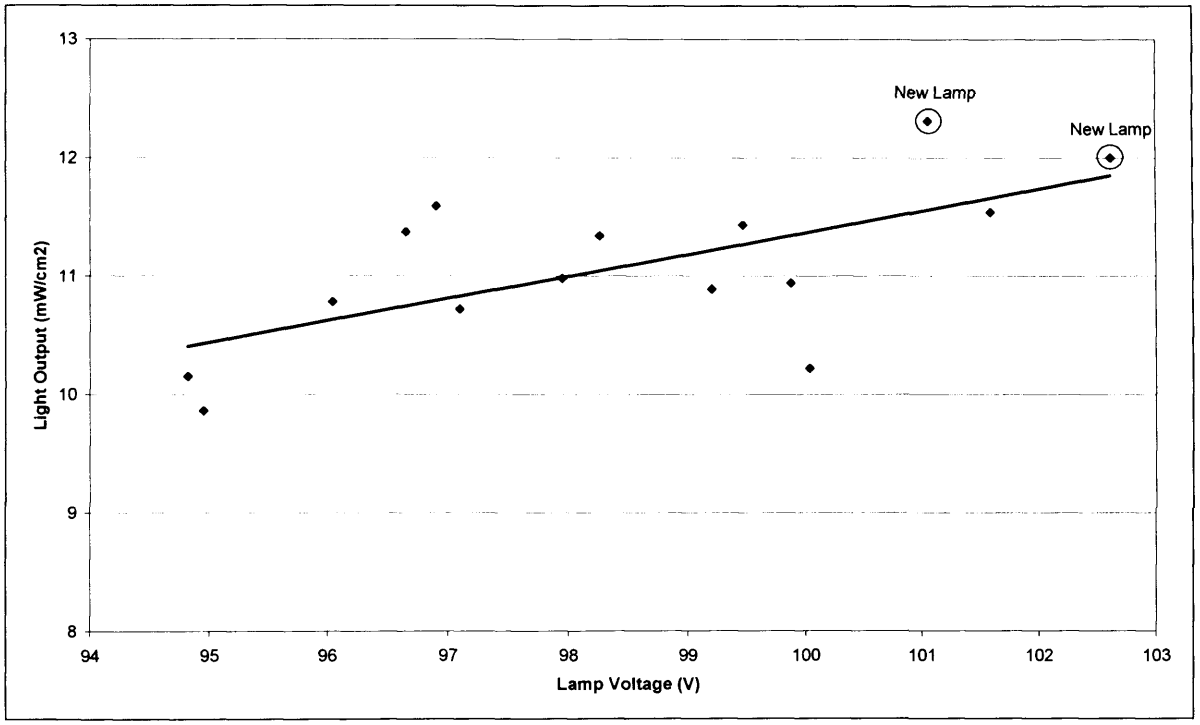


Figure 71:- Light Output vs. Lamp Voltage for the 15 sample lamps.

From figure 71, it can be seen that there is a possible correlation between light output and lamp voltage. The lamps with a larger voltage drop across them had a higher light output. The brand new lamps are the two highest points on the plot as they had the highest light outputs at 12.31 and 12.00mW/cm^2; they also had the highest and third highest lamp voltage drops of 101.05 and 102.61V, respectively. From this it would appear that the lamp voltage and light output fall as the lamp ages.

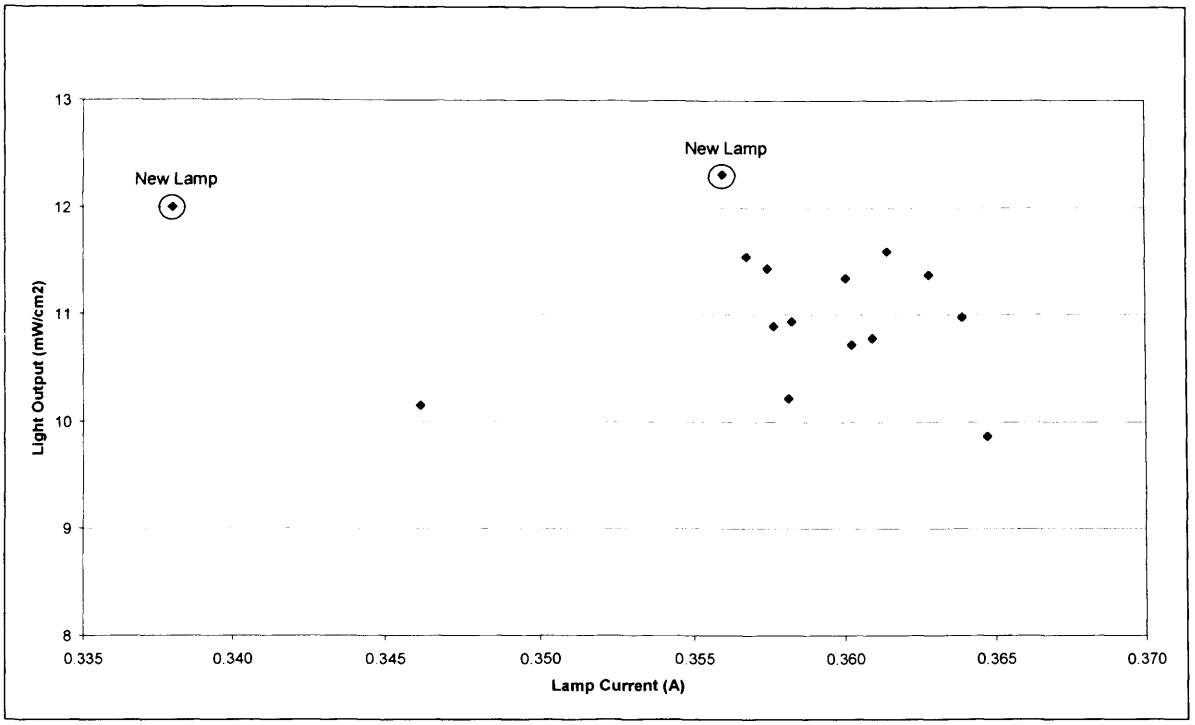


Figure 72:- Light Output vs. Lamp Current for the 15 sample lamps.

From figure 72, it can be seen that there is little (if any) relationship between lamp current and light output; this is a little misleading as the magnetic ballast will have had the effect of adjusting the lamp voltage and current to try and maintain a near constant power to the lamp. This explains why 13 of the lamps have nearly the exact same current (±0.005A) through them, despite one of them being brand new and the others being near the end of their life.

To try and minimise the effects of the ballast, the lamp voltage was divided by the lamp current to give the equivalent lamp running resistance. This is shown in figure 73.

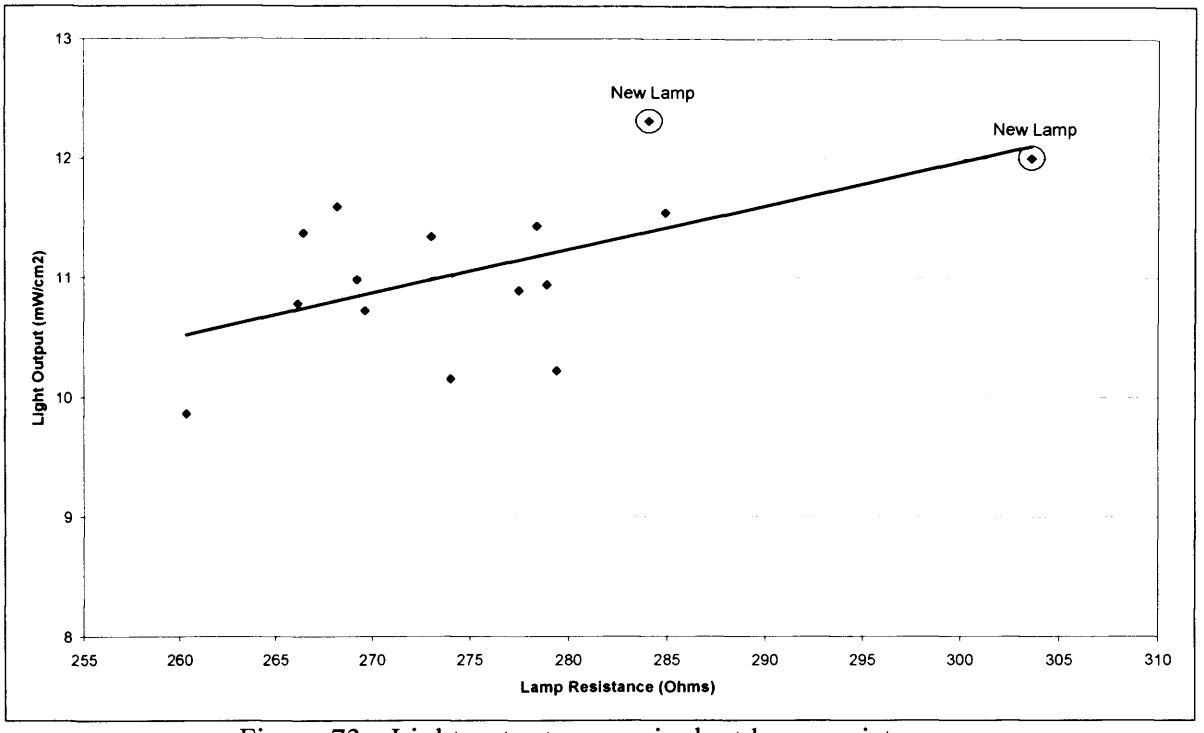


Figure 73:- Light output vs. equivalent lamp resistance

From figure 73, it can be seen that there is a possible correlation between light output and the equivalent lamp resistance. The trend is a little blurred, possibly due to the fact that the readings came from different lamps of different ages, rather than one lamp at different times during its life. Unfortunately, due to the long life of these lamps, this method was the only way to get readings within a reasonable time scale.

In order to complete the analysis, the power and power factor of the 15 lamps were plotted against light output, and are shown in figures 72 and 73, respectively. Although the values for the lamps vary, there does not appear to be any correlation between these and the light output of the lamps.

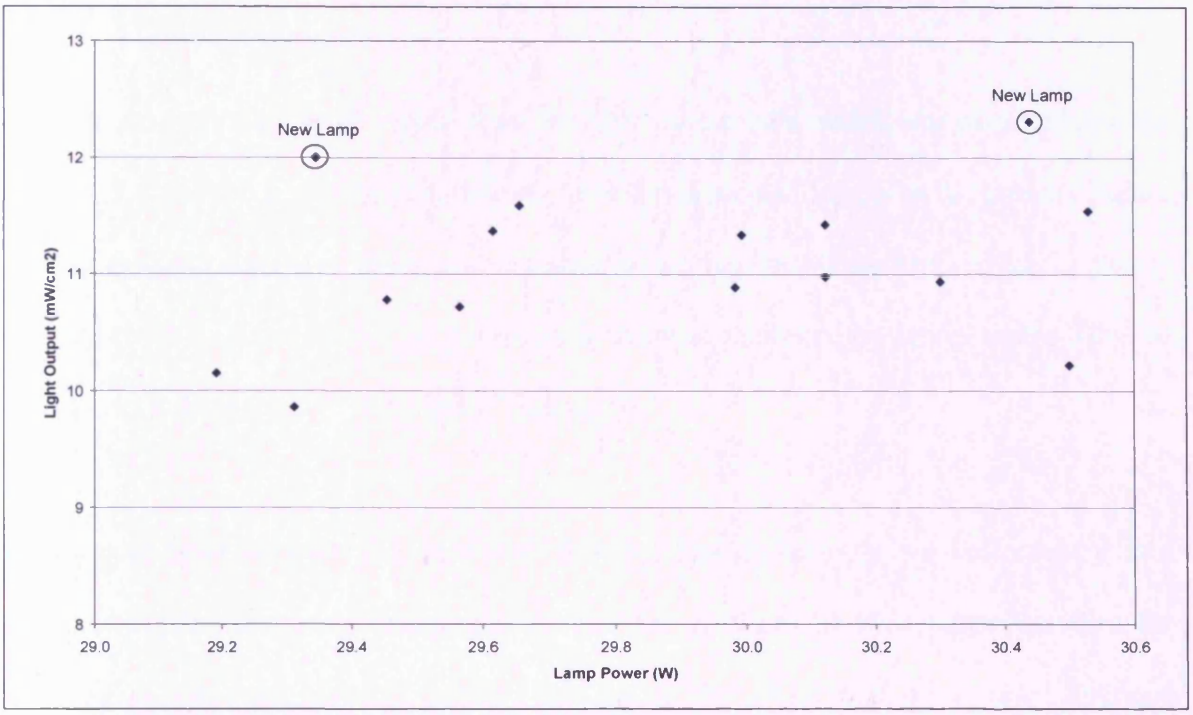


Figure 74:- Light Output vs. Electrical Power Input to Lamp

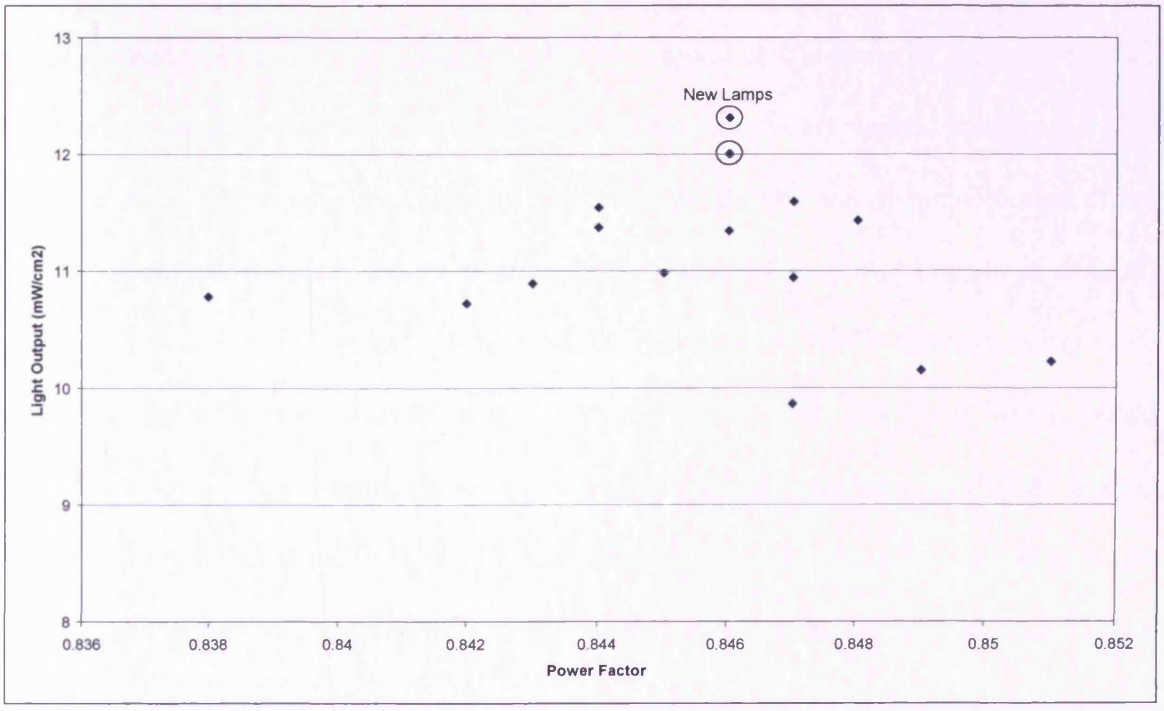


Figure 75:- Light Output vs. Power Factor of Lamp

7.1.3 Discussion

The results indicate that as a lamp ages, the light output falls, which was expected and the gas discharge becomes more conductive. At first this seems illogical as the primary failure mechanism is erosion of the electron emissive coating on the heaters, which in theory should result in a rise of the lamp voltage in order to maintain the same current (flow of electrons) from the now less emissive electrodes.

In the opinion of the author, if the current flowing through the lamp is not affected by light output and therefore age, which appears from the graph to be what happened, then the number of electrons flowing through the gas in the glass tube must have remained relatively constant as the lamp aged, yet the light output appears to have fallen with the older lamps.

As discussed in the previous chapter, the UV output of the lamp is generated when electrons flowing through the tube collide with atoms of mercury vapour which are mixed with the argon gas in the tube. The incident electron excites one of the electrons of the mercury atom into a higher energy orbit, and when this electron drops back to its original orbit it emits energy in the form of ultra violet radiation. If fewer mercury atoms were present in the tube then fewer collisions with electrons would take place, which would result in a lower light output; however the current through the lamp would be largely unaffected as the electrons would still pass through the tube. As fewer collisions take place, the electrons will acquire more kinetic energy before colliding with an atom and so the chance of ionisation when a collision occurs will be higher; with a constant voltage across the tube this would lead to an increase in current through the tube. However to stop

the tube being damaged, the ballast serves to limit the current through the tube; therefore this increase in electron liberation leads to a decrease in lamp voltage. Based on these results, it is proposed by the author that a decrease in light output due to mercury loss also leads to a decrease in the normal operating voltage of the lamp.

7.2 Striking Voltage Measurement

In order to strike a low pressure discharge lamp, a high voltage is applied across the ends of the lamp. In a conventional magnetic ballast system this is achieved by momentarily shorting out the lamp with a glow switch starter via the tube filaments. This warms up the filaments, and when the starter opens, the ballast releases a high voltage impulse which breaks down the gas in the tube. As the lamps age, it would seem logical to propose that a higher voltage would be required to strike them; however this experiment shows that this is not necessarily the case.

7.2.1 Procedure

The experiment was set up as shown in figure 76.

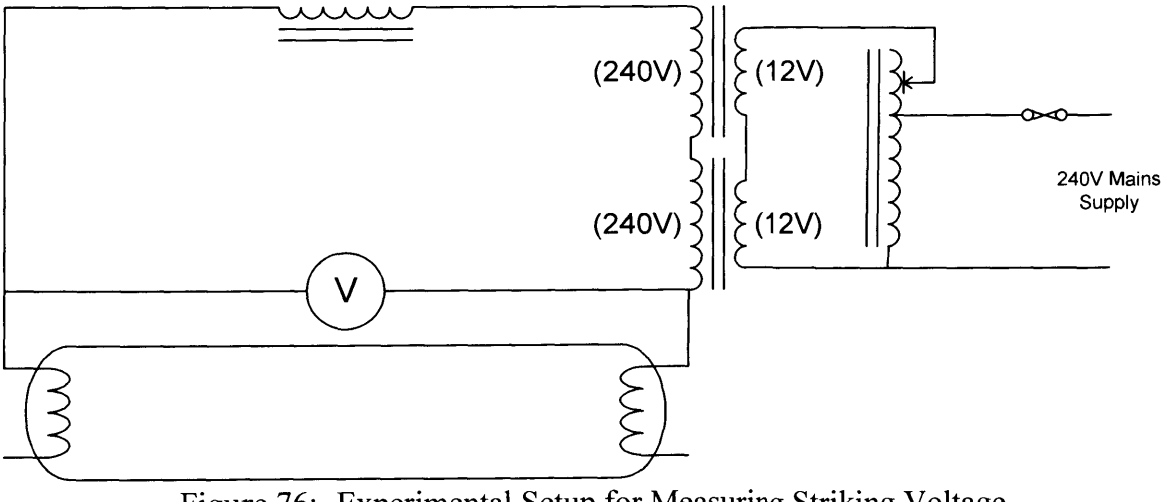


Figure 76:- Experimental Setup for Measuring Striking Voltage

To keep the number of variables to a minimum, the heaters were left cold and the lamps were left overnight in the room so that they were all at the same temperature. To generate a high enough voltage to start the discharge, two 240V-12V transformers were connected in series, with the 12V windings fed from a mains variac. By overloading the inputs to the transformers for very brief periods, it was possible to generate around 700V using this setup. The lamp ballast was left in the circuit to limit the current through the lamp once it had struck.

The lamp was put inside a thick walled plastic tube, firstly to block the harmful UV-C radiation given off and secondly to stop the light in the room liberating electrons from the cathodes via the photoelectric effect, which would have caused the lamp to strike at a lower voltage than it would have otherwise. Once the lamp was connected, the circuit was energised and the variac was very slowly turned up from zero volts. As the lamp was being supplied from an abnormally high mains supply, once struck the lamp current would be abnormally high, even with the ballast in circuit; for this reason, as well as avoiding warming the lamp up, the circuit was turned off as soon as the lamp struck.

The same test procedure was repeated three times for each lamp, with 10 minutes allowed between each strike for it to return to room temperature. As found in an experiment by Yunfen Ji, the resistance and hence temperature of a T8 lamp electrode takes at least 5 minutes to return to cold values after being on[68]. The highest breakdown voltage was taken from the three readings for each lamp, as it represents the minimum required voltage to reliably strike the lamp and hence the worst case scenario.

The same lamps were used as per the previous experiment, apart from lamp 5 which had to be left out of the experiment due to breakage. As the light output is indicative of lamp age, the highest striking voltage of each lamp was plotted against the light output, which was obtained for each lamp from the results of the previous experiment. The plot is shown in figure 77, while the full tabulated results are presented in appendix D.

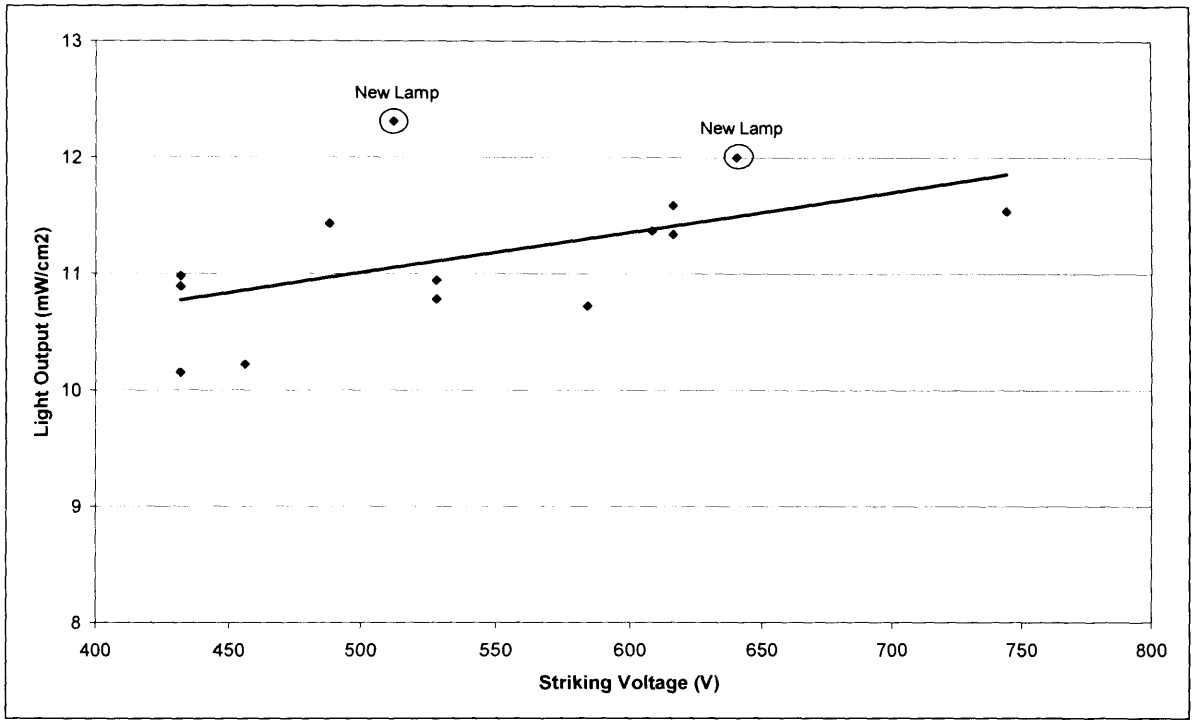


Figure 77:- Light Output vs. Striking Voltage of the 14 UV Lamps

7.2.2 Discussion

From figure 77, it can be seen that the striking voltage appears to decrease with the light output, which implies that the striking voltage decreases as the lamp ages. This is contrary to what would be expected if the electrodes degrading were the cause of the changing striking voltage. The author therefore proposes that the cause of the decreasing striking

voltage is most likely to be the same as the cause of the decreasing steady state lamp voltage observed in the previous experiment. As the lamp ages and the mercury content depletes, there are fewer atoms in the gas for the electrons to collide with, which means that the electrons accelerate further before colliding with an atom. As the average distance before collision is further as the lamp ages, the electrons do not need to be accelerated as quickly through the tube and consequently a lower voltage is required to trigger ionisation breakdown. In summary, as the mercury in the lamp depletes with age, the striking voltage of the tube decreases too; presumably at some further point in the life of the lamp electrode failure will start to reverse this trend.

7.3 Electrode Monitoring

As electrodes age, their electron emissive coating is eroded by both sputtering and evaporation. Once the coating has been damaged, it is harder to draw electrons from the cathode and a higher electric field is required to maintain the discharge current; this leads to an increased cathode fall voltage. The increased cathode fall voltage leads to increased sputtering as the mercury and argon ions are positively charged and so accelerate at higher speed into the cathode. Their impacts sputter more of the electrode material onto the walls of the lamp and at the same time warm up the electrode; however the increased electrode temperature reduces the cathode fall voltage and the rate of sputtering is self limiting to an extent.[69] In smaller lamps such as compact fluorescent lamps, this phenomenon can cause significant problems. It is reported[70] that at the end of life of the lamp, continued operation can lead to a shattered glass envelope or melting of the lamp ends, as shown in figure 78.

Figure 78:- A compact fluorescent lamp at end of life[70]

In the previously reported experiments undertaken in this work, there was no evidence of a rise in lamp operating voltage (or current fall), this could be because the lamp electrodes were running hotter and so compensating for their expected inefficiency due to aging. If the effects of electrode ageing on lamp performance are being obscured by a gradually increasing electrode temperature, it follows that the electrode temperature could be an indicator of their condition. The following experiments introduce electrode temperature monitoring to identify if there is any correlation with lamp age.

7.3.1 Electrode Temperature Measurement

In most UV filtration units the lamp is housed inside a close fitting quartz glass sleeve. Thus attempts to measure the filament temperature indirectly, by measuring the wall temperature of the lamp near the electrode, would not be possible, since this glass sleeve is also in contact with the water flowing through the unit and this would significantly affect

the lamp wall temperature. Further, as the lamp is always hidden from view, optical based systems such as infrared pyrometers are not possible, and any practical system would have to determine the electrode temperature from the electrical characteristics of the heater.

In order to relate the filament voltage to filament temperature, an experiment was conducted where the filament voltage of a new lamp was varied and the colour compared, by eye, with a steel wire supported inside a temperature controlled kiln. The kiln temperature was varied in 50°C steps and the filament voltage required to match the colour temperature of the kiln recorded.

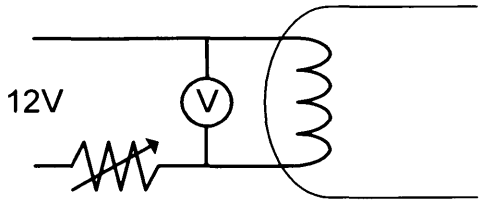


Figure 79:- Circuit configuration for varying filament temperature

A pyrometer could have been used to measure the filament temperature instead of using the kiln, however it proved difficult to find one as for most applications the infrared thermometer has taken over; although one of these was tried, it proved unsuitable as it measured the glass wall temperature rather than the filament. Figure 80 shows a plot of the results for this experiment and tabulated results are presented in appendix D. The trend line is extrapolated to intercept the y-axis at 20°C, which was the recorded room temperature and hence would be the filament temperature when no voltage is applied to it.

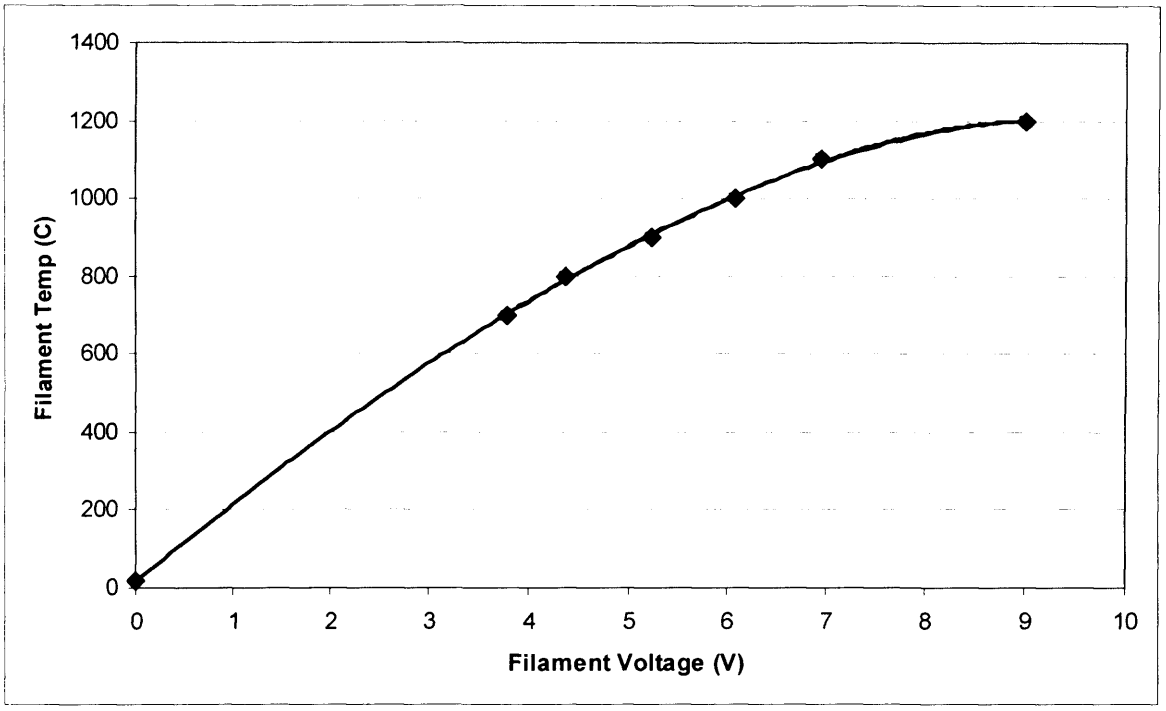


Figure 80:- Filament temperature vs. voltage applied to filament

This relationship assumes that the filament design remains constant between the lamp samples. As all the lamps used in the experiments were Philips G30T8 lamps, the filaments were nominally identical; however if the study were extended to lamps made by other manufacturers then this experiment would need to be repeated to obtain trend information for each lamp type.

The trend line equation fitted to the data from this experiment, which is shown in appendix D, was subsequently used by the next experiment to convert the filament voltage readings into temperatures.

Figure 81 shows the circuit diagram of the experimental setup used for monitoring the running temperature of the filaments in each lamp. Prior to the lamp being struck the filaments were warmed up from the two windings of the 12V transformer via two 22Ω power resistors. The power resistors served to limit the power to the filaments so that they reach approximately 600°C before the lamp is struck. Since any increase in filament temperature will lead to an increase in filament resistance, the voltage across the filament will vary with the filament temperature, due to the series resistance of the power resistors, up to a maximum of the filament supply voltage, which was 12V. The first temperature reading was taken after 30 minutes, i.e. once the filament temperature had stabilized but before the lamp was struck. The reading was seen to stabilise after ten minutes, but a further twenty minutes were allowed for good measure.

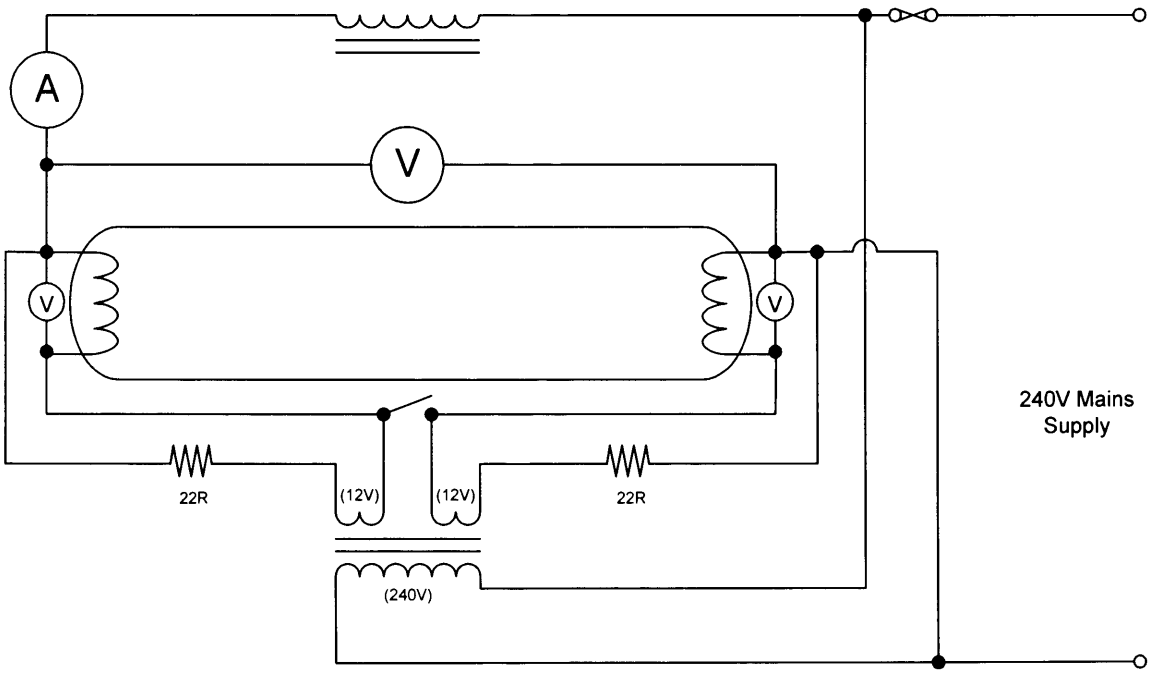


Figure 81:- Experimental setup for measuring filament temperature increase

Once the filaments were stabilized, the starter switch was closed and then opened again to strike the lamp. The filament voltage was then recorded after 30 minutes of being struck. The two readings were then converted to temperatures and the difference between them taken as the rise in electrode temperature due to discharge operation. There are a number of potential flaws in this technique. Firstly, the filaments are likely to be running at a higher temperature than usual due to the extra power introduced by the heater supply. However since this will be the same for all the lamps tested, it should not affect comparisons between the lamps. Secondly, the discharge through the lamp may not leave the filament at the same point in each lamp. Investigations at the Lighting Research Centre[63] have confirmed the theory that the discharge actually moves along the filament like a "burning candle"; the results of one of their experiments can be seen in figure 82.

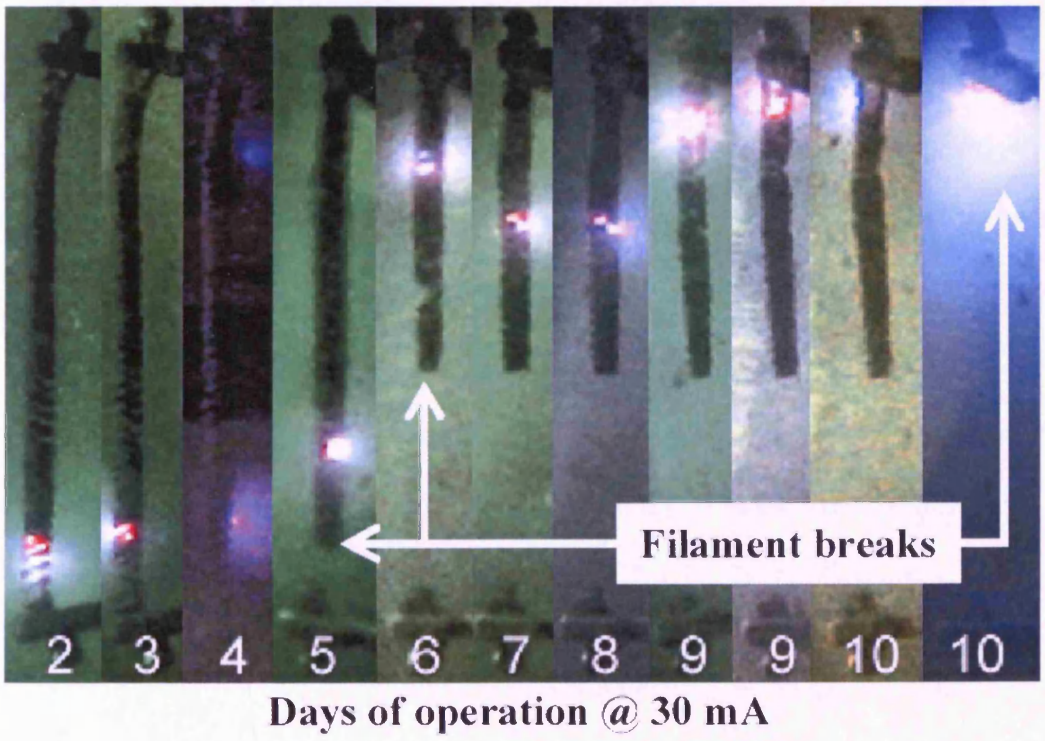


Figure 82:- Movement of hot spot for instant start fluorescent lamp under dimmed condition[63]

The problem with the discharge leaving the filament at different places is that the discharge current, which is usually around 100mA, may flow through the entire length of the filament before exiting the tube to the mains connection, or it may flow through a very small section of filament before exiting. Since the discharge current is supplied from the ballast, it is likely that it could be out of phase with the heater supply current, meaning that the total current through the filament could be 100mA greater or 100mA less than in another lamp where the lamp discharge does not flow through the filament. This difference in current flow would translate to an error in the filament voltage reading of around 1.5V, which would affect the temperature reading by approximately 100°C. To avoid this error, the lamp current was turned off and the reading taken immediately after the light went out, before the filament had time to cool.

Another consequence of the lamp discharge current leaving the filament in different places is that the section of filament it flows through will be hotter than the other part of the filament. This means the voltage (and therefore temperature) reading obtained from this experiment will reflect the average temperature of the filament, rather than the temperature of the discharge point. Since the damage caused by electrode overheating in small diameter lamps is so extreme, as observed in figure 78, it is evident that there is a large scale change in electrode temperature, which should be easily observable from the average filament temperature.

To generate the plot shown in figure 83, the filament voltages from both ends of the lamp were measured while the lamp was struck and converted into temperatures using the

formulae of the best fit line shown in figure 80. The starting temperature of the filaments, which were calculated in the same way, were then subtracted from these values to give the temperature rise of each filament due to the effect of the discharge. As both electrodes alternate between acting as cathode and anode, both are equally responsible for maintaining the electron flow through the lamp; therefore the two figures for electrode temperature rise were added together and divided by two to give the average electrode temperature rise for the lamp.

To measure the UV output of the lamp, a UVX radiometer made by UV Products Ltd was used together with the UVX-25 sensor which was secured over a hole in the middle of the plastic tube used to cover the lamp.

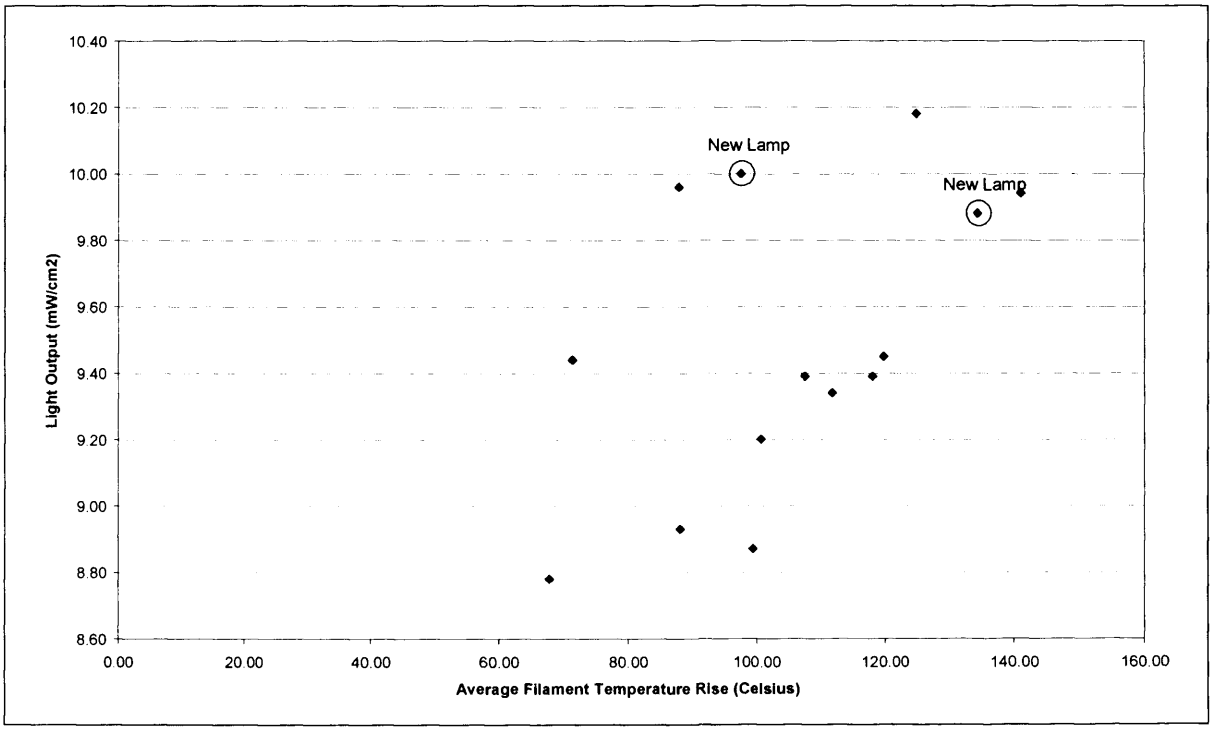


Figure 83:- Light output vs average filament temperature.

From figure 83, it appears that there is no correlation at all between the average filament temperature rise and the light output of the lamp. Since light output is indicative of age, this also implies that there is no relationship between filament temperature and age.

All of the filaments were run at approximately 800°C (± 75°C) before the lamps were struck. From figure 83, it can be seen that all the lamps experienced an electrode temperature rise of between 65 and 145°C. This means that there was a maximum of 80°C difference in temperature rise between each of the lamp samples, which is not a large difference for a filament that is operating at around 900°C. Closer examination of the figures, which are tabulated for inspection in appendix D, revealed that the second highest electrode temperature rise occurred in lamp 14, which before the experiments started was a brand new lamp and had only been used for an hour or so during the first experiment. The results seem to suggest that monitoring the electrode temperature has revealed no evidence of the electrodes running any hotter in the older lamps.

7.3.2 Lamp Rectification Monitoring

As the mains supply is AC, the direction of current reverses twice every second. This means that for the first half of a mains cycle the one electrode acts as the cathode and during the second half the other electrode acts as the cathode. If one electrode is worse at emitting electrons than the other, then more current will flow through the tube in one direction than in the other; this means that the tube acts as a partial rectifier and a DC component develops across the tube. As no two electrodes will degrade at exactly the same rate, the development of a DC potential across the tube can be used to indicate that the electrodes in the tube are reaching their end of life. This technique is deployed in the

Infineon ICB1FL01G ballast control $IC_{[71]}$, where the controller shuts the lamp off if the ratio of the positive and negative peaks of the lamp voltage is above 1.15 or below 0.85 for longer than 500mS. This ratio equates to a DC level of approximately 0.075 times the peak lamp voltage. The lamp voltages were measured in the previous experiments and were all around 100V RMS, which would equate to a DC cut off voltage of approximately 10V. In this experiment, the DC component of the lamp voltage is measured to find out if any of the lamps are exhibiting signs of assymetrical operation due to a failing electrode. Each lamp was connected up in turn to the aparatus shown in figure 84. The lamp was run under normal operating conditions for 30 minutes and the DC component was then measured using a true RMS Fluke 87 Series Multimeter.

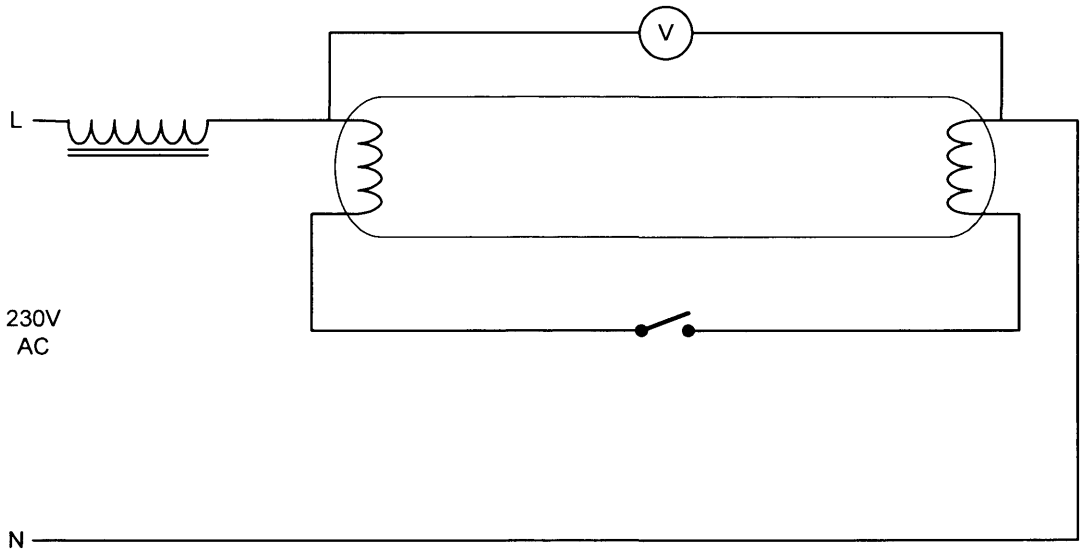


Figure 84:- Experiment configuration for measuring rectification

The measurements were then plotted against the light output previously measured for each lamp, as shown in figure 85.

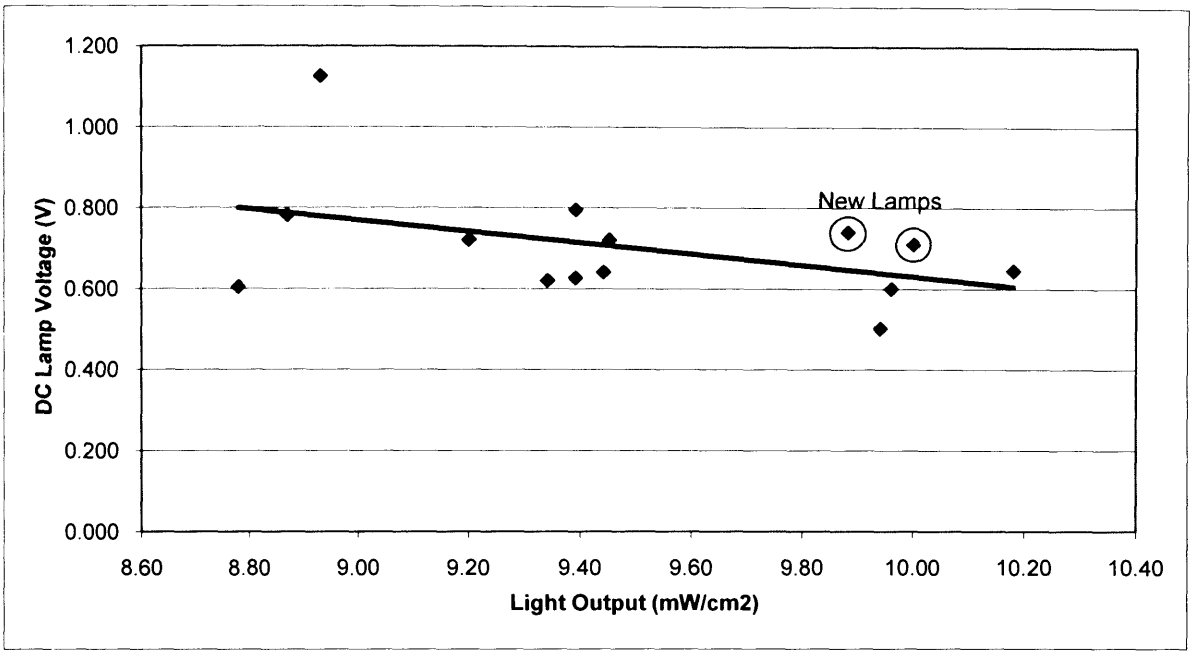


Figure 85:- DC component of lamp voltage vs light output

From figure 85, it is apparent that as the lamp ages and its light output falls, the dc component of the lamp voltage increases. This is as expected, as a reduction in the emission rate of one of the electrodes will lead to asymmetrical behaviour. However the level of DC voltage is exceedingly small (approximately 1% of the ac voltage across the lamp). The Infineon ballast control $IC_{[71]}$ would have required a DC component of approximately 10V to shut down. This shows that the DC component measured on these lamps is very small compared with that of a faulty lamp and the electrodes in each of these lamps are operating virtually in balance with each other.

7.3.3 Discussion

The electrodes in a fluorescent lamp transfer the discharge current from the incoming conductor to the discharge column and then back out again on the other side. During the half cycle that the electrode acts as a cathode, it is essential that the electrode has a high

electron emissivity, and for this reason each electrode is coated with an emissive coating. Without the emissive coating, thermionic emission is insufficient for the discharge current.[63] Therefore, once the coating has worn off the lamp current should start to decrease, causing the electrode temperature to decrease and the lamp current to drop further still until the lamp goes out.

In this chapter, experiments were carried out to measure the lamp voltage, current, filament temperature, striking voltage and light output. The samples varied greatly in age, two of the lamps were bought brand new for the experiments and so had a running time of only a few hours by the end, the others were all removed from use from water filtration installations around Wales, and had been in constant use for between one and two years (8760 - 17520 hours), which is well beyond their rated life. Despite this large difference in sample ages, the UV light output varied very little, in fact one of the two new lamps had a lower UV output than many of the older lamps. It was also found that the electrical characteristics of the lamps varied only slightly, the lamp voltage changed only slightly and rather than rising with age, which would have been consistent with electrode failure, the voltage appeared to fall with age, indicating mercury loss. The filament temperature did vary between the samples by up to 80°C, but with a filament temperature of around 900°C, this again is not a very large variation and showed no sign of a correlation with light output.

The only electrode monitoring parameter which did show any sign of a correlation with light output was the DC lamp voltage, which is generated as a result of assymetric lamp operation due to an under performing electrode. The DC voltage present across the lamps

varied between 0.50 and 1.13V, which compared with the AC lamp voltage of approximately 100V is a fluctuation of less than 1%. The Infineon Ballast Control IC[71] mentioned earlier in this chapter would deem the lamp to have failed and shut it off when its DC component reached 10V, which is nearly a factor of 10 greater than any of the sample lamps measured in these experiments.

From these results, it is concluded by the author that although the lamp samples were all older than their rated life, the electrodes and electrode coatings were still in good operating condition despite their age. In order to determine what happens when the electrodes fail, the experiment in the next section attempts to damage the electrodes of one of the new lamps by over running the tube heaters to see what effect it has on the running characteristics of the lamp.

7.4 Accelerated Electrode Ageing

Failure due to electrode erosion occurs when the emissive coating on the cathode has decreased to the point where not enough electrons are emitted to maintain the discharge current. As discovered in the previous experiments, the time taken for an electrode to fail can extend into years, which meant it would have been impractical to test for this failure mode under normal operating conditions. The two primary causes of electrode erosion are sputtering and evaporation. Sputtering occurs (mainly) when the cathode is too cold and the subsequent high cathode fall voltage acts to accelerate the mercury ions into the cathode at high speed, causing bulk displacement of cathode material. As the cathode is

generally kept warm by the discharge current alone, cooling it to speed up the sputtering process would have been impractical; evaporation, on the other hand, occurs slowly during normal use and much quicker when the cathodes are run too hot. As the cathodes are filaments, it is relatively easy to run the cathodes too hot by passing an additional electrical current through the filaments while the lamp is in use. This causes the filaments to evaporate and deposit on the inside of the glass tube at a much faster rate than normal, rapidly eroding their coating.

7.4.1 Procedure

To overrun the electrodes, a voltage had to be applied to both of the filaments while the lamp was in use. It was found through simple experiments that approximately 12V caused the filaments to glow bright white and it appeared that increasing it any further may cause the filament to melt, so 12V was adopted. During the previous experiments, the filament reached its normal operating temperature of around 900°C when a voltage of the order of 6V was applied. Running the filament at 12V, double its normal voltage, will, in the opinion of the author, significantly increase the rate of electrode evaporation and seriously reduce the life of the lamp.

In order to apply voltage to the filaments while the lamp was in use, two electrically isolated 12V sources were required; as during normal lamp operation there is a voltage difference of around 100V between the two filaments. A single transformer was used with two 12V windings on the secondary, which were connected directly across the filaments,

as shown in figure 86. Switches were inserted to allow switching between the accelerated testing and normal operating mode when gathering results.

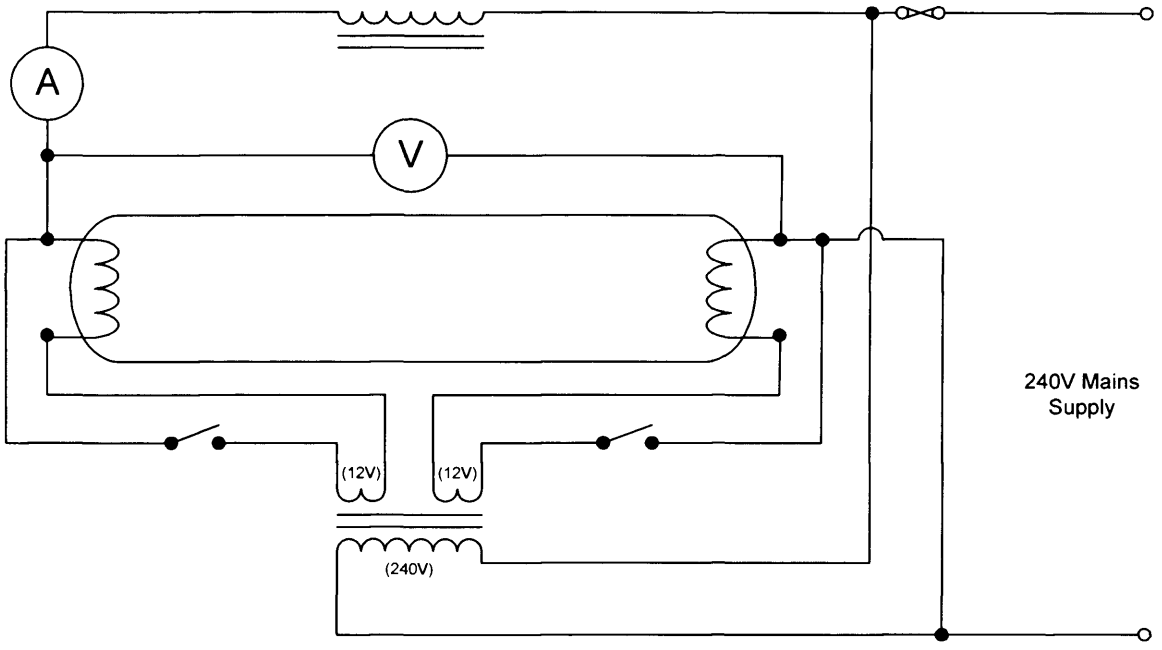


Figure 86:- Experimental Setup

The Philips 30W G8 lamp was left on continuously for 6 days each week with the heaters being overrun, and on the seventh day the switches were opened to disconnect the power to the filaments and allow the lamp to run normally. At the end of the seventh day the readings for the lamp were recorded and the lamp filaments were then reconnected to the 12V supplies for another 6 days. This process, which is illustrated in figure 87, was repeated until the lamp eventually failed.

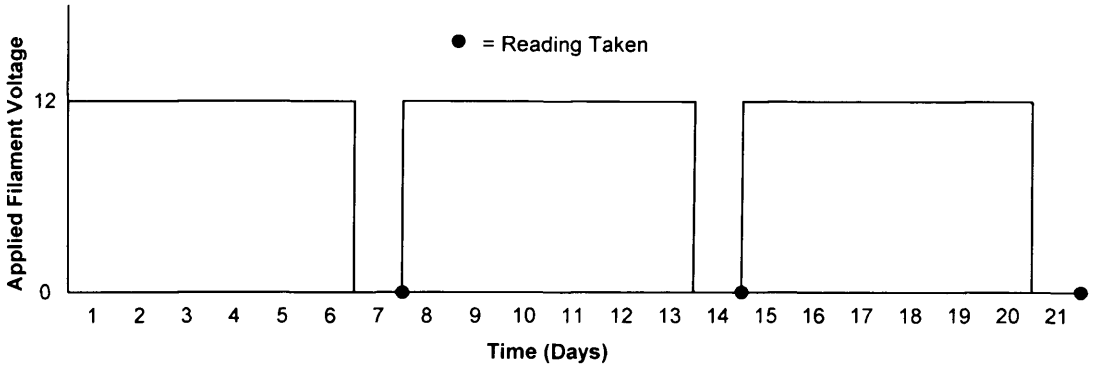


Figure 87:- Filament voltage timing profile

7.4.2 Results

The following figures show the measured characteristics, i.e. light output, lamp voltage, current and power as the accelerated ageing test progressed. The lamp tested survived for 13 weeks in total, although by the time the measurements were taken on the 13th week, it was effectively useless as it would not stay alight for more than 10 minutes without restriking.

Figure 88 shows the light output for the lamp during its entire life. It can be seen that the light output falls steadily with age; there is a small platteau around week 8, before it starts falling more sharply again after week 9. By the end of week 12, when the lamp was still running unaided, the light output had fallen to less than half its initial value. During the 13th week the lamp failed, its light output fell to a quarter of its original value and it required frequent restriking.

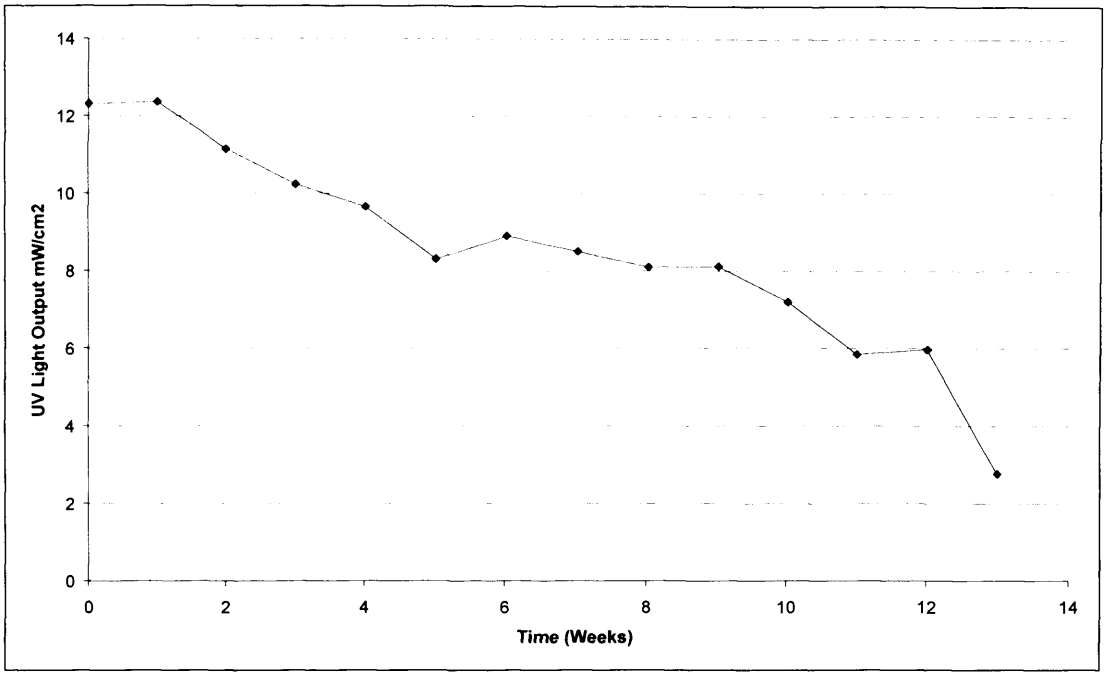


Figure 88:- Light Output vs. time with accelerated aging

The lamp power, shown in figure 89, dropped steadily with time as the lamp aged until week 13, when the power dropped to approximately a third of its original value.

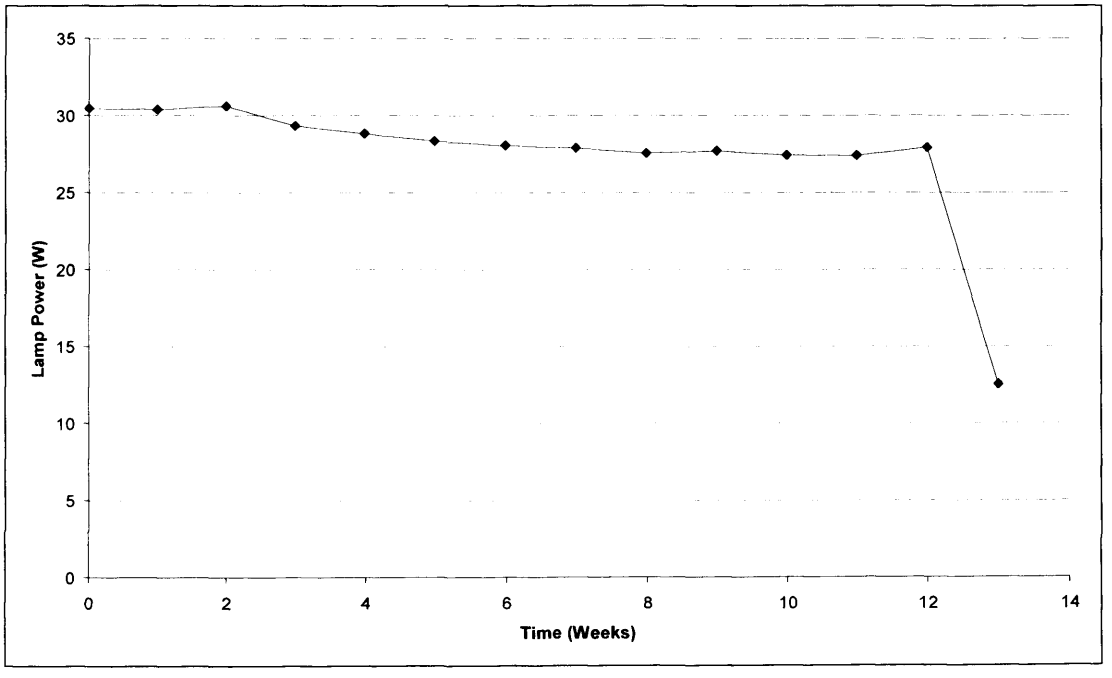


Figure 89:- Lamp Power vs. time with accelerated aging

The lamp current also dropped steadily with time until week 13, as shown in figure 90.

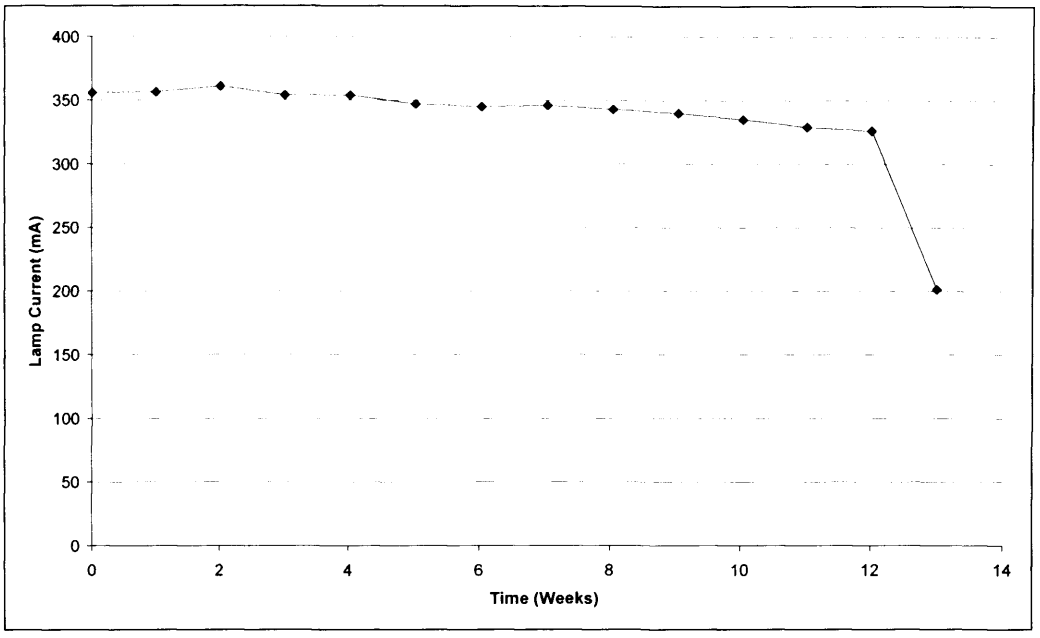


Figure 90:- Lamp Current vs. time with accelerated aging

The lamp voltage decreased steadily with age until the lamp was 8 weeks old and then started increasing again. The plot for this can be seen in figure 91.

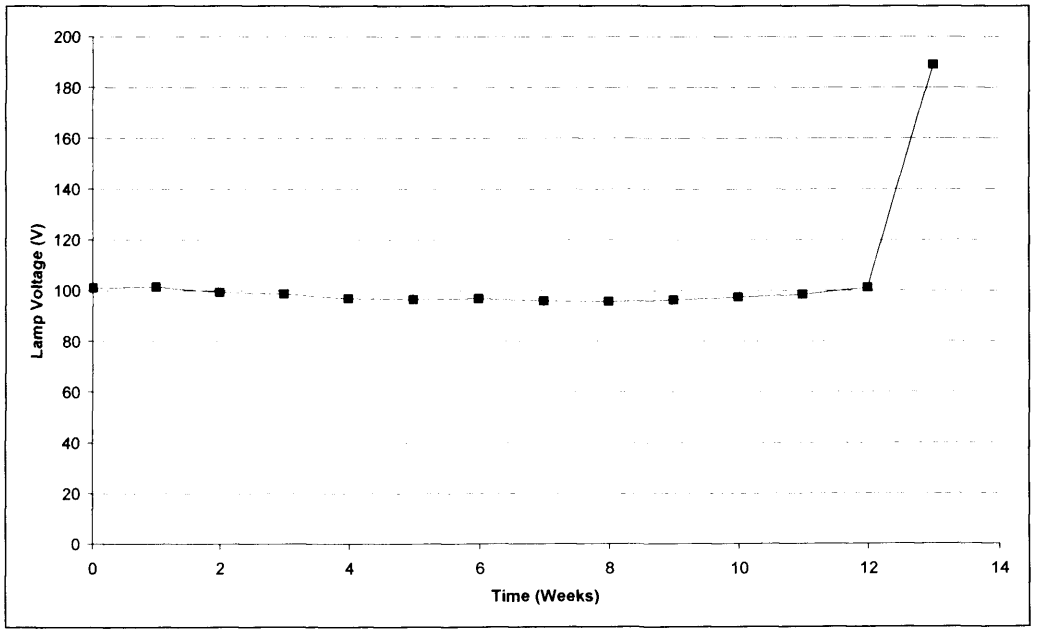


Figure 91:- Lamp Voltage vs. time with accelerated ageing

7.4.3 Discussion

After week 13, when the lamp had failed, it would start normally but extinguish after around 10 minutes of use. Looking at the results for lamp current and power, this is not surprising as the lamp power when it was running was nearly a third of the normal lamp power. Reduced lamp running power leads to reduced electrode temperatures, which reduces the lamp current further until the discharge eventually extinguishes.

Up until week 13, the lamp power and current showed a gradual decrease over the life of the lamp. The lamp voltage however showed a gradual decrease during the first 8 weeks of use and then started increasing again, so that at the point of failure the lamp voltage was nearly double what it was when the lamp was new. In section 7.1 it was shown that the lamp voltage decreased with the light output. It was proposed that this was due to the loss of mercury within the lamp enclosure; it is further proposed here that this mechanism explains the observed decrease in lamp voltage observed up until week 8.

After week 8, the lamp voltage started rising again and continued rising until the point of failure; in the author's opinion this is indicative of electrode failure, as the emissivity of the electrodes decreases due to electrode erosion, the ballast increases the voltage across the lamp to try and maintain the current through the lamp. The eroded electrode material could be seen deposited on the inside of the glass tube; a photograph of this, which was taken after the experiment was completed, is shown in figure 92. The change in prevalent failure mechanism around week 8 also coincides with and explains the resumation of the decrease in light output observed after week 8.

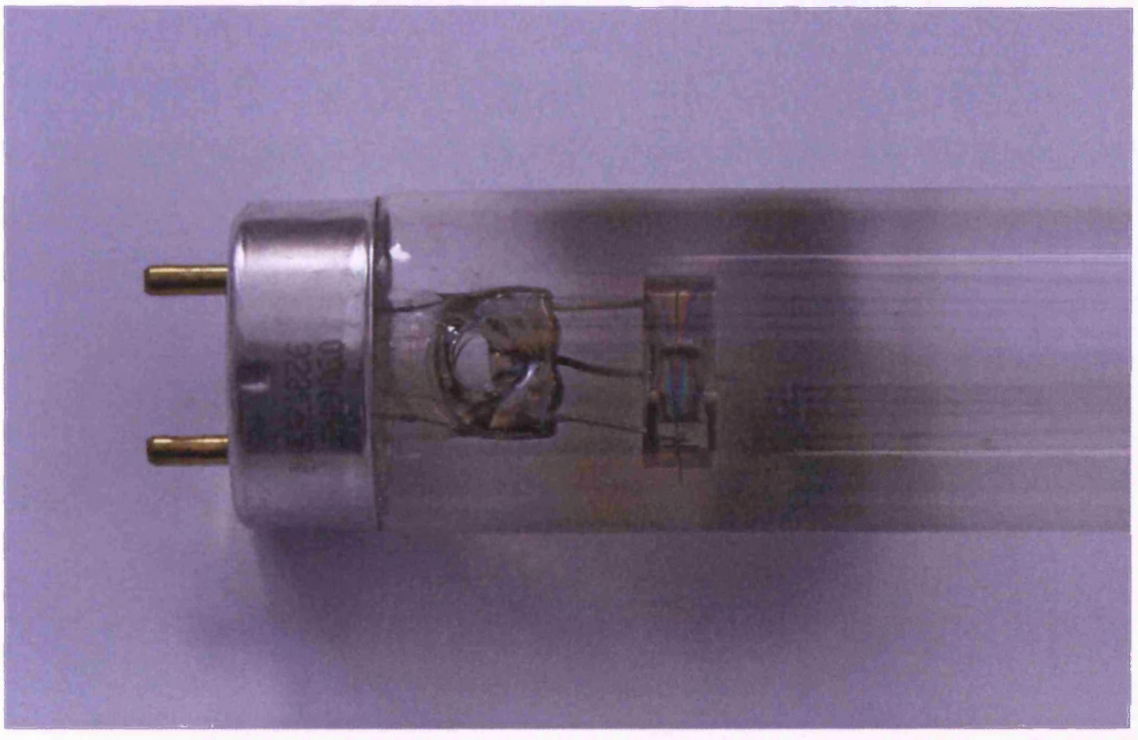
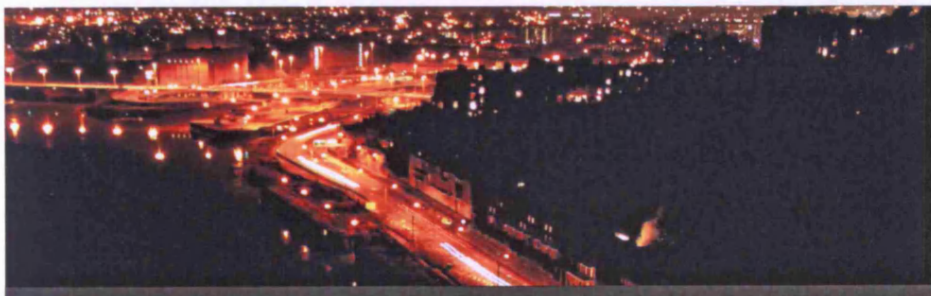


Figure 92:- Photograph of the lamp end post failure

Based on the results obtained, it is possible to monitor the voltage across the UV lamps to determine the extent of mercury loss and corresponding light reduction based on the reduction in lamp operating voltage during the first phase of the lamps life. Electrode failure can also be detected when the lamp voltage begins to rise.

To build on the findings of this experiment and establish suitable fault detection thresholds, a large scale trial involving a few hundred lamps being run at regular power levels until they fail is required. Unfortunately this would take several years to complete and was therefore outside the possibilities of this work. However, these findings do show that the technique works and would be practical once the appropriate functional thresholds have been determined.



8

Discussion

8.1 Potential Implementation

With both filament lamp monitoring and UV lamp monitoring, it was found that the changes in electrical characteristics of the lamp are relatively small (typically less than 1Watt) and will require accurate measurement to detect them. However, as well as being accurate, any implementation has to be low cost (in comparison to the lamp it is monitoring) and compact (approximately as small as the lamp it is trying to monitor). Fortunately, precision power measurement is used extensively for electricity metering and as a result of technological advances in remote reading of electricity meters, manufacturers such as Maxim have developed purpose-made integrated circuits for use in such applications. One such device is the Maxim MAXQ3120 High Precision Analaogue to Digital Convertor Mixed-Signal Microcontroller, which is primarily designed for the single phase electricity metering market, but is also suitable for any application requiring high precision analogue to digital conversion and fast processing.

"The MAXQ3120 microcontroller is a high-performance, 16-bit microcontroller that incorporates dual, true-differential, 16-bit sigma-delta analogue-to-digital converters (ADCs), a liquid-crystal display (LCD) interface that can drive up to 112 segments, and a real-time clock (RTC) module with a dedicated battery-backup supply."[72] Although some parts of this device are redundant for this application, all of the requirements needed for lamp monitoring are incorporated. The device is listed on Maxim's website as costing $4.83 for 1000+ quantities, which is approximately equivalent to £2.50 in Britain. As there are very few external components required with this device, it offers an extremely cost effective and compact way of implementing lamp monitoring, while the 16,000 word

flash program memory allows complete customisation of the monitoring system. A block diagram of the MAXQ3120 is shown in figure 93 and a possible way of using it is shown in figure 94.

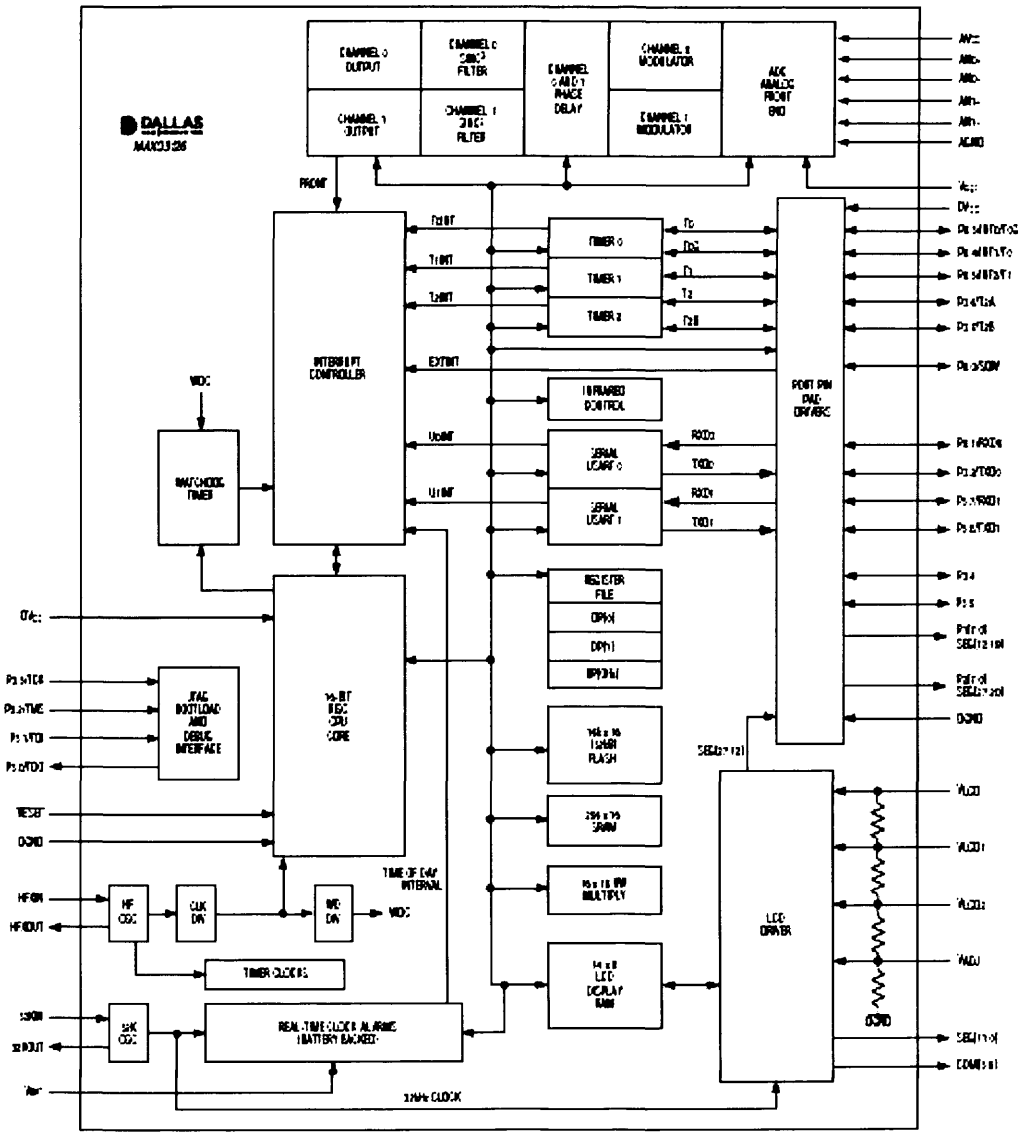


Figure 93:- Block Diagram of the Maxim MAXQ3120 IC[72]

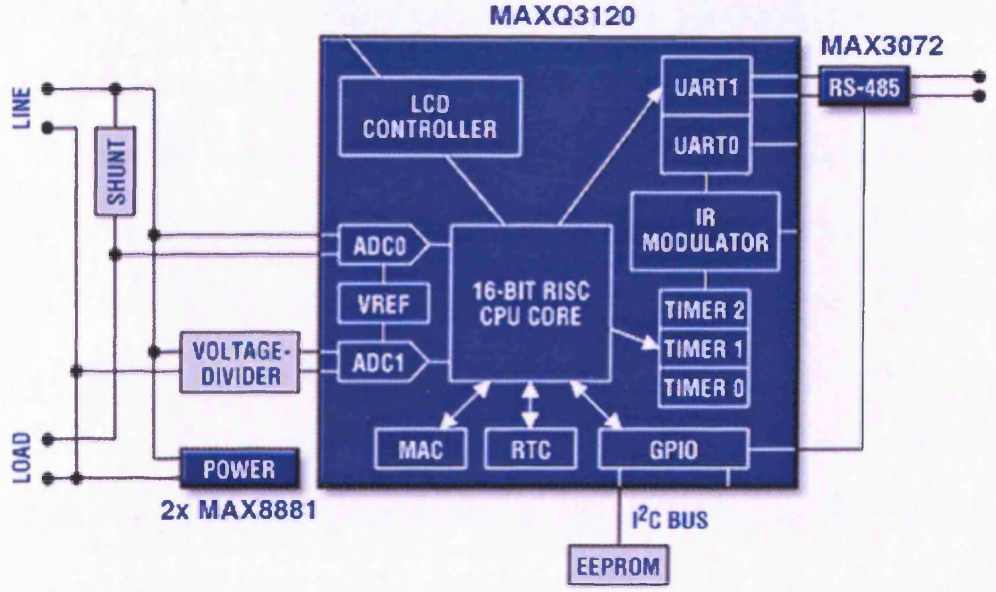


Figure 94:- Block diagram of how the MAXQ3120 could be used for lamp monitoring [adapted from 66]

From figure 94, it can be seen that very few external components would be required in order to use the MAXQ3120 for lamp monitoring applications. The EEPROM is required so that readings taken from the lamp while it is in use are not lost on power down, as a significant number of readings will usually be required in order to accurately calculate the rate of filament evaporation. The RS-485 transceiver enables the lamp being monitored to be connected along with other lamp modules to a central monitoring station. An alternative to this would be to use power-line communication to link local lamp modules together without the addition of extra wires and then link the local groups to a central monitoring station using another method such as an RS-485 cabled link, a GSM radio link or even an IP based link across the internet.

Any practical system is likely to feature a large number of lamps, so for this reason it is essential that all data is dealt with at the source and only high level information is passed back to the main controller. For example, every lamp will produce a large number of readings for voltage, current etc. Generally speaking, the main controller has no need to receive all this data; the only data the main controller needs is feedback on lamp condition, which could be in the form of a percentage life remaining. This level of abstraction not only significantly reduces data transfer, but enables the main controller to monitor many lamps of different types without needing detailed information about each and every lamp. The consequence of dealing with the data at a low level is that processing power is required at each lamp, which is also catered for by the 16-bit RISC core of the Maxim MAXQ3120 IC.

Having developed practical techniques for monitoring filament and UV discharge lamps, consideration has to be given to what the future holds for such technology.

8.2 Filament Lamps and the Future

A valid point which could be raised about this technology is why would anyone install filament lamp monitoring equipment when the British government, along with other governments around the world, are proposing banning filament lamps within the next four years[73]. What sounds like a simple bold government initiative however is far from simple and many barriers stand in the way of this objective.

The commonly cited reasons for replacing incandescent light bulbs with Compact Fluorescent Lamps (CFL's) include approximately five times lower power usage and a longer life expectancy. These advantages however are not as straightforward as they seem at first sight; although a fairly inefficient light generator, incandescent lamps are very efficient at generating heat, but this is discounted as not being of any use when the lamp efficiency is calculated. Since most homes use lighting more in the winter than during the summer, the heat from the lamps is not always wasted and simply serves to heat the home as well as light it, reducing the amount of time the thermostatically controlled room heating is on for. In his article for BBC News[74], Dr Matt Prescott makes the valid point that although the heat given off by light bulbs reduces the amount of other heating required, gas central heating is more efficient than the conversion process used to generate the electricity to power the lamp in the first place; however a lot of modern apartments and flats are heated solely by electricity and in these cases the argument that the heat from a light bulb is not wasted can still hold true. The other argument for the use of CFL's is their longer life expectancy. Once again this is a sweeping generalisation; CFL's do not respond well to frequent starting, which is why the packets usually display a warning to not use them in circuits controlled by occupancy or PIR sensors. If the lamp is used in an automatic security light, for example, where it may be on for less than a minute at a time, then its life span may be less than that of a standard incandescent lamp in the same role. Considering the higher purchase cost and hazardous substances contained in CFL's, it makes no sense to use CFL's in locations such as these.

The major problem for homeowners trying to change over to compact fluorescent lamps is compatability with existing light fittings. The vast majority of CFL's cannot be dimmed

and will be seriously damaged if they are put in a circuit which features a conventional dimmer. CFL's also require good ventilation to keep the lamp and electronics cool, which makes them unsuitable for most enclosed light fittings, such as those found in bathrooms etc; use in such fittings can result in a 15% to 20%[75] reduction in light output and potential early failure.

Outside of the domestic environment there are many specialist applications where incandescent lamps are the only practical choice, these include examples like stage lighting where LED lighting has had some impact, but the majority of luminaires are still based on incandescent lamps due to their high light output, better colour rendering and ease of dimming.

In summary, although CFL's can in many cases be substituted for standard GLS lamps, banning incandescent lamps completely would be impractical as there are many applications where CFL's or LED lamps simply cannot compete with incandescent lamps in terms of light output or endurance. Although it could be easily missed amongst the sensationalist headlines regarding the banning of incandescent lamps, the European Comission has recognised the nessessity of incandescent lamps in certain applications and said that 15% of lamps sold will be exempt from the minimum efficiency standards. In the words of Dr Matt Prescott, "none of the attention grabbing proposals to ban incandescents, made anywhere in the world, have yet made it into binding legislation and there is a danger that light bulb bans have been announced in order to silence critics rather than drive meaningful change. Places such Australia, Canada, California and the EU have all

announced light bulb bans, to great applause, but none have said how their bans will be implemented"[76]; it appears for now at least that the incandescent light bulb is far from extinction, and is likely to continue to be used in the more specialist applications for years to come. It is these specialist applications, such as theatre lighting, that are particularly likely to benefit from comprehensive lamp monitoring systems.

8.3 UV Lamps

Compared with filament lamps, very few germicidal UV lamps are manufactured each year; however the techniques developed for monitoring them during the course of this work would be equally applicable to standard office fluorescent lamps, which provide a much larger potential market for any lamp monitoring system. UV lamps present significant monitoring challenges to the person maintaining them; not only is their continued operation critical for the safety of the water they are being used to treat, but detecting when they are failing can be very challenging indeed. As was shown in chapter 7, just because the same power is being delivered to the lamp does not mean that its UV light output is correct. Since the main wavelength of their light output is invisible to the human eye, it is impossible to tell from looking at the lamp whether it is producing the correct light intensity or not, and purchasing dedicated UV sensors is a very expensive way of monitoring each lamp. Based on the findings of this research, monitoring the electrical characteristics of the lamp more closely over time could provide a very cost effective way of implementing automated lamp monitoring.

Most UV lamp installations feature a very small number of lamps at a specific location on site, as they are often used to treat drinking water in locations where mains water is not available. Their location can be well away from main roads and other accessible routes and hence automated monitoring presents enormous benefits to the companies and people whose job it is to maintain the UV germicidal units. Linking the findings of this work to existing distributed monitoring systems, such as those discussed in the first chapter for monitoring street lighting, could produce a system whereby a company can remotely monitor all its installations and visit customers as their systems require servicing.

If the monitoring hardware was mass produced, its highly likely that it would find new applications in monitoring fluorescent lighting throughout entire buildings, so that maintenance could be optimised and lamps changed before they start flickering and wasting energy through inefficient operation.

8.4 Lamp Monitoring Systems

Whatever the type of lamp being monitored, the potential for distributed monitoring systems is clear. The hurdles at the moment are the succesful monitoring of individual lamp types and simplifying the technology required so that it remains reliable and cost effective in relation to the system it is monitoring. As discussed in chapter two, street light monitoring systems are already being installed by several councils, and although the initial cost is high, it is clearly hoped that the reduced maintenence costs and improved service will justify them. The challenge now is to tune the techniques discovered during this work on low pressure discharge and filament lamps, so that the hardware required can be

minimised and the associated cost of installing the system will then make it practical for installation in theatre and stage lights etc, as well the lower cost light fittings found in offices and commercial premises around the world.

References

[1] Wikipedia "Electrodeless Lamp"
[Online]
http://en.wikipedia.org/wiki/Electrodeless_lamp [2007, August 4th]

[2] Finnish National Road Administration, "Traffic Safety at Signal Controlled Junctions" (1996), ISBN 951-726-289-2.
http://virtual.vtt.fi/fits/impacts/tl6796en.pdf [2008, January 19th]

[3] Aqua-Win Watertec Co Ltd, "UVC Lighting"
[Online]
http://www.hoyi.com.tw/index.htm [2007, January 19th]

[4] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 11 – Fluorescent lamps, page 192

[5] Alan F Mandel, Kenneth M Eichler & William H Moore, of Westinghouse Electric Corporation, "Elevator System With Lamp Failure Monitoring"
Patent Number US4646058A

[6] HowStuffWorks "How Light Bulbs Work"
[Online]
http://science.howstuffworks.com/light-bulb2.htm [2006, May 21st]

[7] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 9 – Incandescent lamps, pages 155-157

[8] John D. Bullough, "la vita è bella",
"Lighting Futures", Rensselaer Polytechnic Institute,
Volume 4, Number 3, Page 2 (2000)
[Online]
http://www.lrc.rpi.edu/programs/futures/LF-LampLife [2007, September 7th]

[9] How Stuff Works, "How does a halogen light bulb work?"
[Online]
http://home.howstuffworks.com/question151.htm [2007, September 23rd]

[10] The Jag-lovers Web, "Jaguar XJ-S Bulb Failure Sensors"
[Online]
http://www.jag-lovers.org/xj-s/book/BulbFailureSensor.html
[2007, September 23rd]

[11] Wikipedia "Fluorescent Lamp"
[Online]
http://en.wikipedia.org/wiki/Fluorescent_lamp [2006, May 24th]

[12] Encyclopedia Britannica Online, "Fluorescent lamp" [Online] http://www.britannica.com/eb/article-9034676/fluorescent-lamp [2007, September 23rd]

[13] Merle Henkenius PM, Illustration by Eugene Thompson "Fluorescent Lamp" [Online] http://www.ustr.net/electronics/fluorescent.shtml [2006, May 24th]

[14] Atlantic Ultraviolet Corporation "What is germicidal ultraviolet?" [Online] http://www.ultraviolet.com/whatis.htm

[15] Ashok Gadgil, "UV Waterworks 2.0" [Online] http://www.lbl.gov/Education/ELSI/Frames/Sustain21-f.html [2007, September 7th]

[16] AquaPro Industrial Co. Ltd. "AquaPro UV6GPM-H" [Online] http://www.aquapro.com.tw/images/PageB10-2.htm [2008, January 19th]

[17] U.S. Environmental Protection Agency, "Spills, Disposal and Site Cleanup" [Online] http://www.epa.gov/mercury/spills/index.htm#fluorescent [2007, September 23rd]

[18] CEYX Technologies Inc (2003, Feb 6th). "White Paper for CEYX Technologies LCD Backlighting Control" [Online] http://www.ceyx.com/web/WhitepaperLCDBacklightingControl.pdf [2005, April 5th]

[19] Royal Philips Electronics (Lighting). "Master Colour CDM-T" [Online] http://www.lighting.philips.com [2006, May 26th]

[20] Yiyoung Sun (US) of Osram Sylvania Inc (US), Patent No. EP0696157, "Ballast containing protection circuit for detecting rectification of arc discharge lamp", Published 02 July 1996

[21] Helbing Rene (US), Nishimura Ken A (US), Tullis Barclay J (US) of Hewlett Packard Co (US), Patent No. US6362573, "Apparatus and method for monitoring the life of arc lamp bulbs", Published 26 March 2002

[22] Klinkenberg Klaus (DE) of Koninkl Philips Electronics NV (NL), Patent No. US6534932, "Method and arrangement for monitoring a gas discharge lamp", Published 18 March 2003

[23] Wei-Chun Chang of BENQ Corporation, Patent Application Publication No. US20040032225, "Method and apparatus for detecting remaining lamp lifetime", Published 19 Feb 2004

[24] Royal Philips Electronics (Lighting) "BIG BEN and little details" [Online] http://www.lighting.philips.com/gl_en/feature/bigben.php?main=global&parent=4390&id=gl_en_feature_stories&lang=en [2006, May 26th]

[25] Royal Philips Electronics (Lighting). "Philips QL Lamp Systems, Information for Original Equipment Manufacturers" [Online] http://www.lighting.philips.com [2006, May 26th]

[26] American DJ. "P64 LED" [Online] http://www.americandj.com/product.asp?ProductIDNumber=1558&cat= [2006, May 26th]

[27] Koninklijke Philips Electronics N.V. (No date). "Keeping an eye on your future, Telemanagement solutions in outdoor lighting by Philips" [Online] http://www.lighting.philips.com/gb_en/architect/luminaire/controls/tech_data/Outdoor/telemanagement_brochure.pdf [2005, April 5th]

[28] Philips Lighting UK. (No date). "LFC7050/00 Segment Controller" [Online] http://www.lightingcontrols.philips.co.uk/gb_en/architect/luminaire/controls/tech_data/Outdoor/outdoor_datasheet.pdf [2005, April 15th]

[29] R.Seevaratnam of Integrated Equipment & Instrumentation (2001). "Street and Public Light Monitoring System (SPL)". In MonoSys Guide to Monitoring [Online Journal] Pages 12-16, 3rd Quarter edition 2001. http://www.aguide.net/magazine/Q3-2001/pdf/Q3-2001.pdf [2005, April 5th]

[30] Royce Thompson Ltd (2003) "Oasis 2000 RF Radio Frequency Module" [Online] http://www.roycethompson.com [2005, April 5th]

[31] TEC Product Review (2003) "FVD data ripe for transport plans, Integrated Systems" [Online] http://www.itisholdings.com/pdf/fvd_ripe.pdf [2005, April 5th]

[32] Harvard Engineering Plc (No date). "How LeafNut works" [Online]. http://www.leafnut.com/how-it-works.asp [2006, October 31st]

[33] The Lighting Journal (2006) "LeafNut: Remote Monitoring and Dimming via the Web" [Journal] Pages 28-30 May/June edition 2006.

[34] Telensa Ltd. (2006) "PLANet™ Public Lighting Active Network" [Brochure]

[35] Archnet Technology Ltd (No date). "Street Public Lighting Control and Monitoring" [Online].
http://www.archnetco.com/english/product/lamp.htm [2005, April 5th]

[36] Fortran Traffic Systems Ltd (No date). "Naztec Lamp Monitoring System" [Online].
http://www.fortrantraffic.com/naztec/lampmon.htm [2005, April 5th]

[37] House of Commons Hansard Written Answers (2003, June 20th), Column 453W, "Light Bulbs" – Estimated figures from the Lighting Industry Federation in 2002 [Online]
http://www.publications.parliament.uk/pa/cm200203/cmhansrd/vo030620/text/30620w03.htm [2005, April 5th]

[38] British Standard EN60064:1996 including amendments No's 1 & 2
"Tungsten filament lamps for domestic and similar general lighting purposes – Performance requirements"
Page 16, Section 5.3.3

[39] British Standard EN60064:1996 including amendments No's 1 & 2
"Tungsten filament lamps for domestic and similar general lighting purposes – Performance requirements"
Page 8, Section 1.5.20

[40] How Products Are Made, "Light Bulb – Quality Control"
[Online]
http://www.madehow.com/Volume-1/Light-Bulb.html [2006, January 31st]

[41] Steve Guch, Litton Systems Inc., "What are the experiments done to test the properties of the light bulb?"
[Online]
http://www.madsci.org/posts/archives/feb99/917962248.Ph.r.html
[2005, August 8th]

[42] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 9 – Incandescent lamps, Figure 9.1, page 156.

[43] "William David Coolidge" (2003, April 23rd)
[Online]
http://www.geocities.com/bioelectrochemistry/coolidge.html [2006, Nov 20th]

[44] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 7 – Lamp Materials, page 131.

[45] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 7 – Lamp Materials, Figure 7.3, page 132.

[46] British Standard EN60064:1996 including amendments No's 1 & 2 "Tungsten filament lamps for domestic and similar general lighting purposes – Performance requirements"
Page 11, Section 3.4.1

[47] British Standard EN60064:1996 including amendments No's 1 & 2 "Tungsten filament lamps for domestic and similar general lighting purposes – Performance requirements"
Page 14, Section 4.2.4

[48] Wikipedia, the free encyclopedia (2006, Nov 13th), "Molybdenum" [Online]
http://en.wikipedia.org/wiki/Molybdenum [2006, Nov 20th]

[49] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 7 – Lamp Materials, page 136 - 137.

[50] Wen C. Lin & Chi-Foon Chan, "Feasibility Study of Automatic Assembly and Inspection of Light Bulb Filaments"
Proceedings of the IEEE, Vol.63, No.10, October 1975, pages 1437-1445

[51] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 7 – Lamp Materials, page 134.

[52] Tony Davis (2006), "Letsfixit.co.uk" [Online]
http://www.letsfixit.co.uk/html/lamps_faq.html [2006, Nov 21st]

[53] Fisher E, Fitzgerald J, Lechner W, and Lems W, (1975) Philips Tech. Rev., 35, 296-302: Research on incandescent lamps. "Part II. Transport and burn-out in incandescent lamps"

[54] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9
Chapter 10 – Tungsten Halogen Lamps, page 174.

[55] R M Marston, "Newnes Passive and Discrete Circuits" Published 2000, ISBN 0750641924, Page 369

[56] British Standard EN60064:1996 including amendments No's 1 & 2 "Tungsten filament lamps for domestic and similar general lighting purposes – Performance requirements", Page 9, Section 2.1.3

[57] Avalon Equipment Corporation (No date). "Voltech PM100 AC Power Analyzer" [Online] http://avalontest.com/e-store/prod_details.asp?pid=1016&pcid=12157493871983 [2006, October 7th]

[58] LabJack Corporation. "Labjack U12" [Online] http://www.labjack.com/images/lju12.jpg [2006, October 7th]

[59] Centre for Solid State Science, Arizona State University. "Why does a light bulb burn out?" [Online] http://invsee.asu.edu/nmodules/lightbulbmod/comp.html [2007, August 19th]

[60] Hanovia UV, "Automatic sleeve cleaning system inside UV treatment chamber" [Online] http://www.hanovia.com/information/newsUK.aspx [2008, January 27th]

[61] Lighting Industry Federation Ltd, "Use of PIR detectors with electronic ballasts and fluorescent lamps" LIF Technical Statement No.17, Issue No.4. (April 2007) page 1.

[62] Cole Palmer. "EW-97650-64 Germicidal Detector and Lamp Monitor" [Online] http://www.coleparmer.com/Catalog/product_view.asp?sku=9765064&pfx=EW [2007, April 8th]

[63] Lighting Research Centre, Rensselaer Polytechnic Institute, "Investigation of the Effects of Dimming on Fluorescent Lamp Life" [Online] http://www.lrc.rpi.edu/researchTopics/reducingBarriers/pdf/investigateEffectsOfDimming.pdf [2007, April 9th]

[64] Waymouth, J.F. "Electric discharge lamps" Cambridge, MA and London, England: M.I.T. Press, 1971).

[65] Peter Terren, "Tesla Down Under" [Online] http://tesladownunder.com/VactubeCrookes.jpg [2007, May 7th]

[66] M A Cayless and A M Marsden, "Lamps and Lighting" Third Edition 1983, Edward Arnold (Publishers) Ltd, ISBN 0-7131-3487-9 Chapter 6 – Production of incoherent radiation, Figure 6.4, page 109.

[67] Ultra Violet Products, "UVX Radiometers"
[Online]
http://www.uvp.com
[2007, Aug 22nd]

[68] Yunfen Ji, Robert Davis, Conan O'Rourke, Edmond Wai Mun Chui
"Compatibility Testing of Fluorescent Lamp and Ballast Systems"
IEEE Transactions on industry applications
Vol. 35, No. 6 (Nov/Dec 1999) page 1272

[69] A .Hilscher (Osram), "Determination of the cathode fall voltage in fluorescent lamps by measurement of the operating voltage"
Journal of Physics D: Applied Physics 35 (2002) page 1707.

[70] "The electrical safety authority responds to consumer concerns about compact fluorescent lamps"
[Online]
http://www.ofm.gov.on.ca/ENGLISH/Fire%20Safety%20&%20Public%20Education/Recalls/2007/ESA_CFLamps_March22.asp
[2007, May 19th]

[71] Infineon Application Note, "ICB1FL02G Smart Ballast Control IC for Fluorescent Lamp Ballasts" (June 2007)
[Online]
http://www.infineon.com/dgdl/AN_icb1+070612.pdf?folderId=db3a304313719f4f01137358642500ee&fileId=db3a304313719f4f01137359052c00ef
[2007, August 25th]

[72] Maxim Semiconductor datasheet, "MAXQ3120 High-Precision ADC Mixed-Signal Microcontroller"
[Online]
http://pdfserv.maxim-ic.com/en/ds/MAXQ3120.pdf
[2007, August 25th]

[73] Jonathan Duffy, BBC News, "Age of enlightenment" (March 2007)
[Online]
http://news.bbc.co.uk/1/hi/magazine/6457991.stm [2007, August 25th]

[74] Matt Prescott, BBC News, "Shedding light on call to ban bulb" (April 2006)
[Online]
http://news.bbc.co.uk/1/hi/sci/tech/4922496.stm [2007, August 25th]

[75] "Not Cool to Be Hot"
IAEEL newsletter 2/93
http://www.iaeel.org/iaeel/news/1993/tva1993/LiTech_2_93.html
[2007, September 7th]

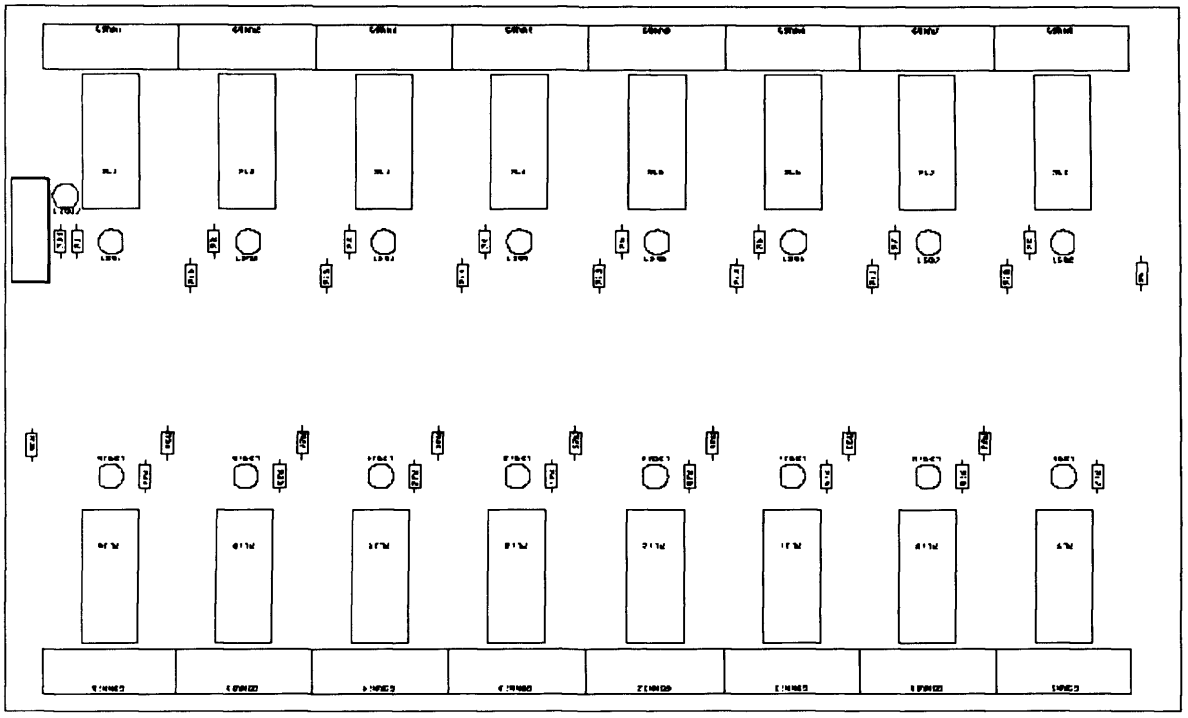
[76] Matt Prescott, "Ban the bulb" On-line campaign. [Online] http://www.banthebulb.org/ [2007, August 25th]

A

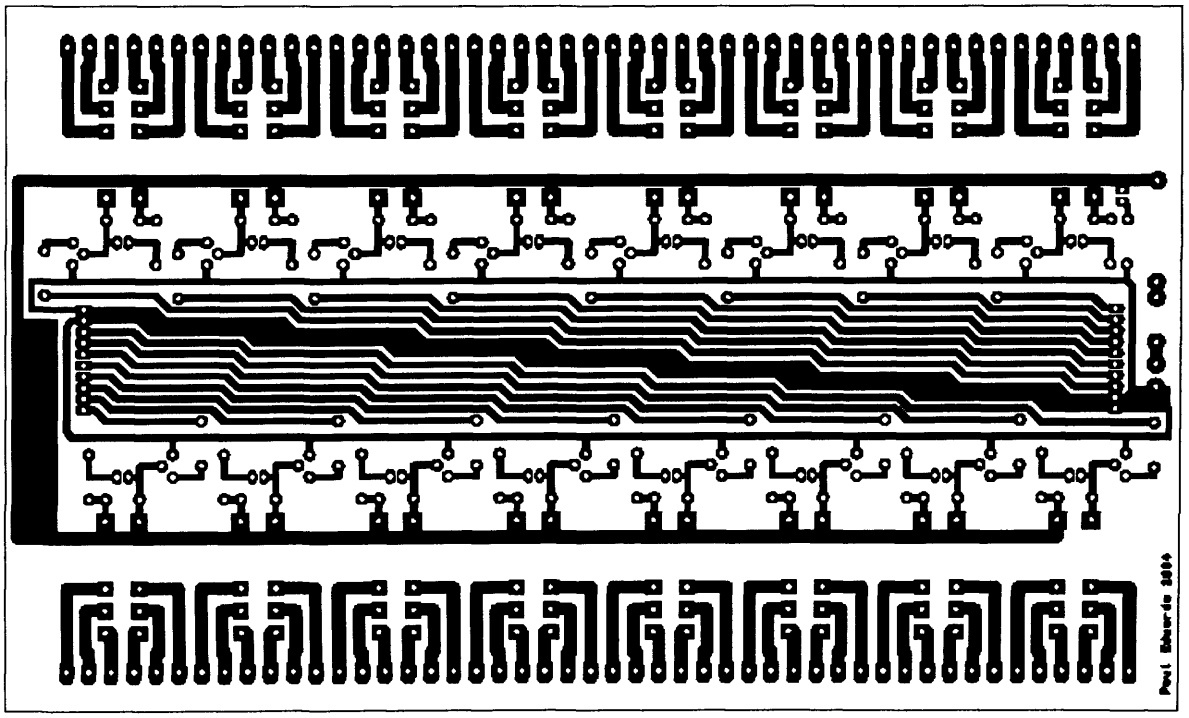
Appendix A - Test Rig Construction

PCB Layouts

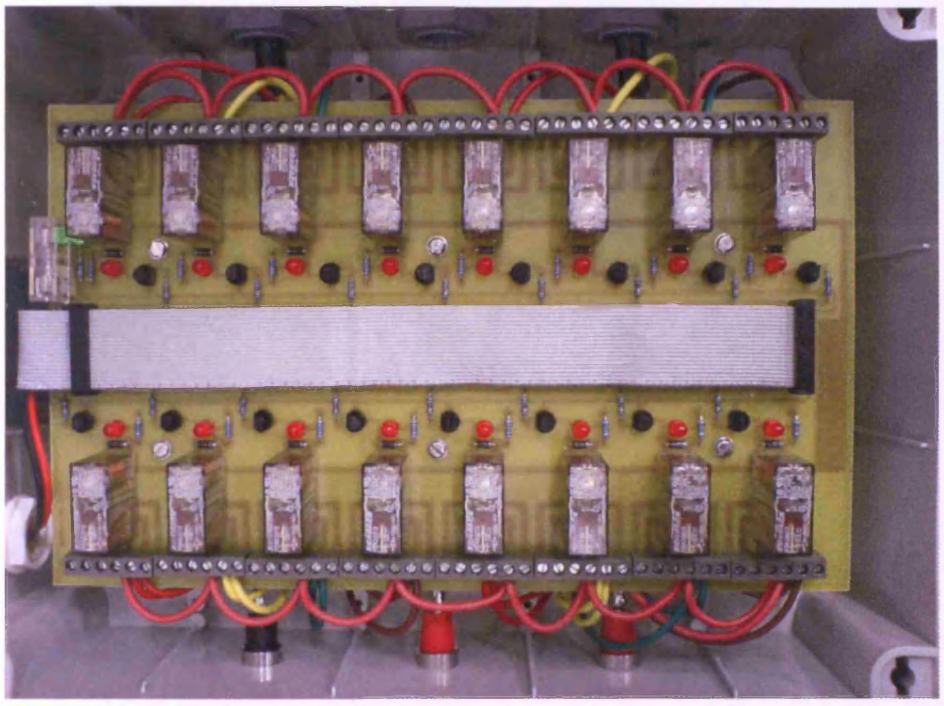
Relay Board Silk Side



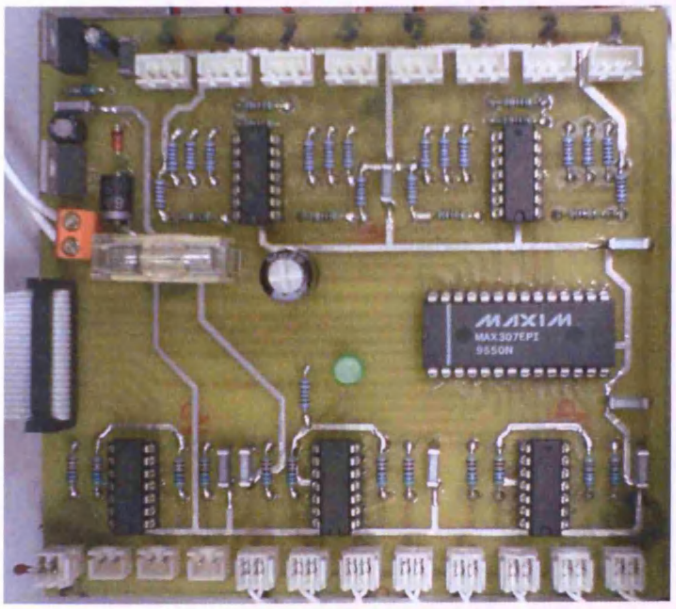
Relay Board Track Side



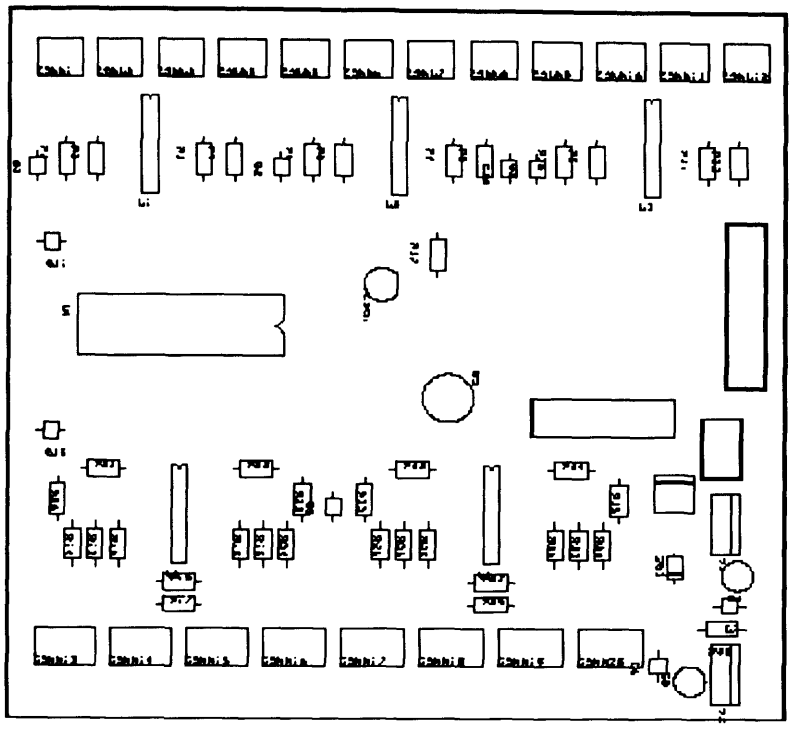
Relay Board Photograph



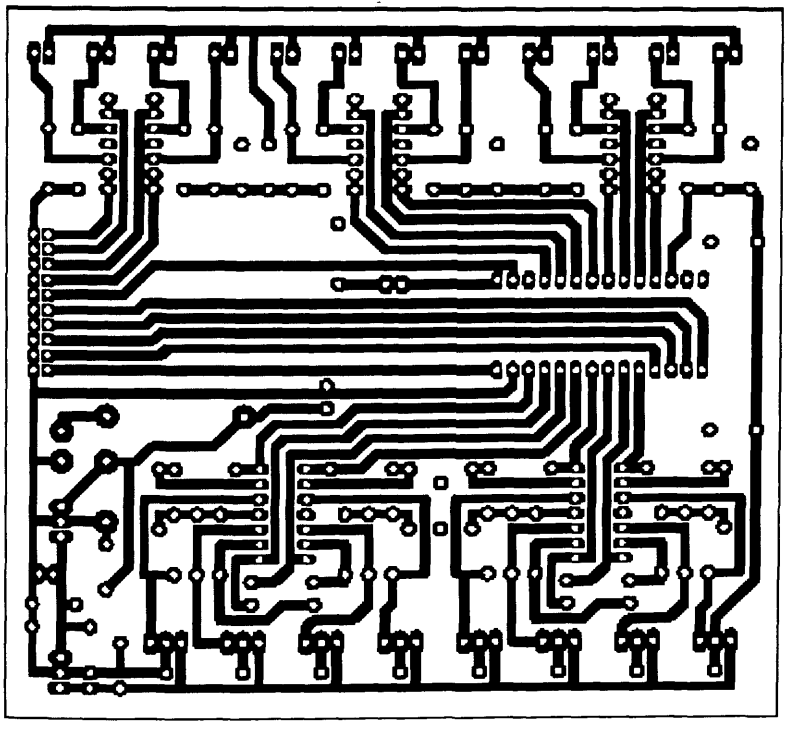
Sensor Board Photograph



Sensor Board Silk Side

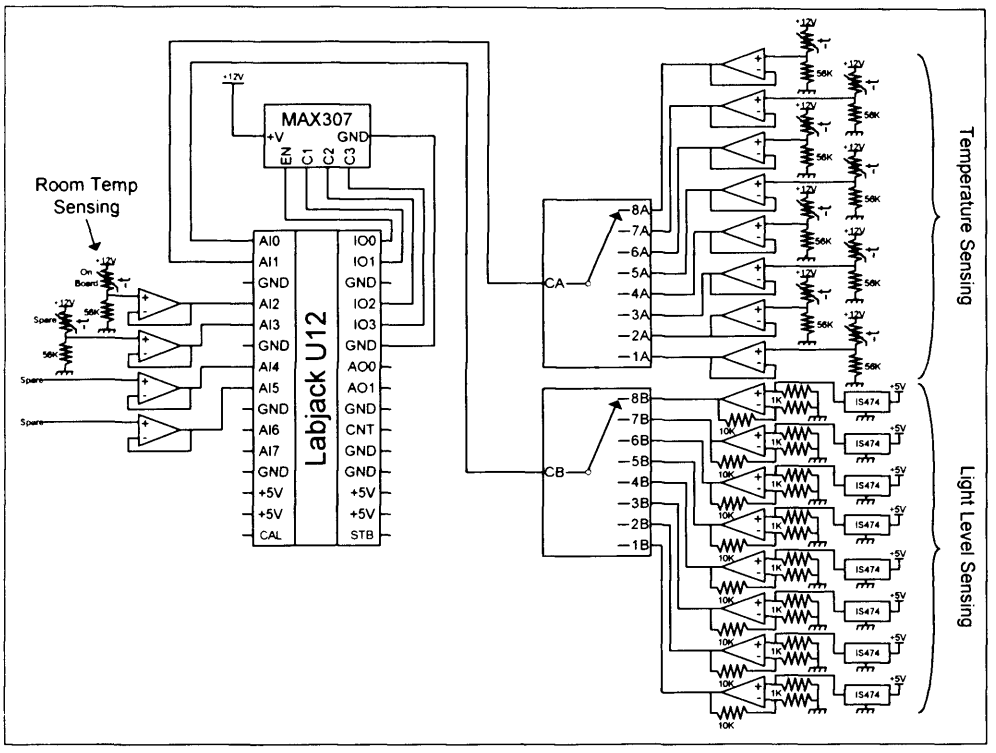


Sensor Board Track Side

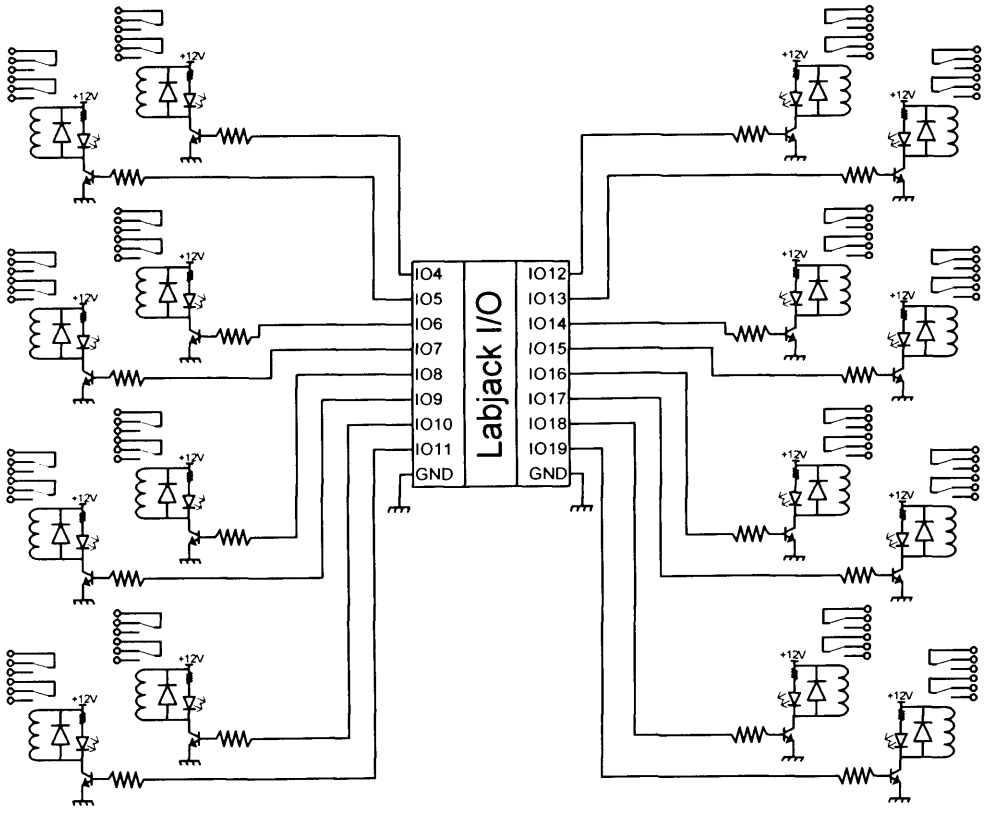


Circuit Diagrams

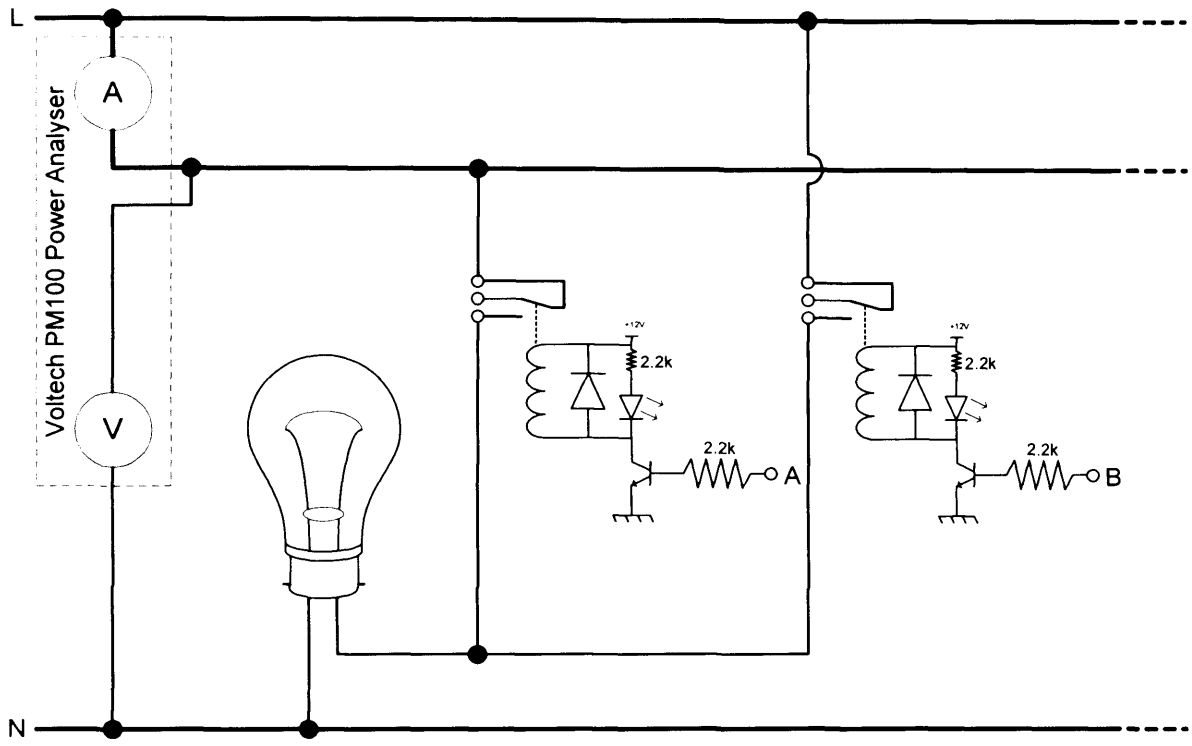
Sensor Board



Relay Board



Mains Wiring

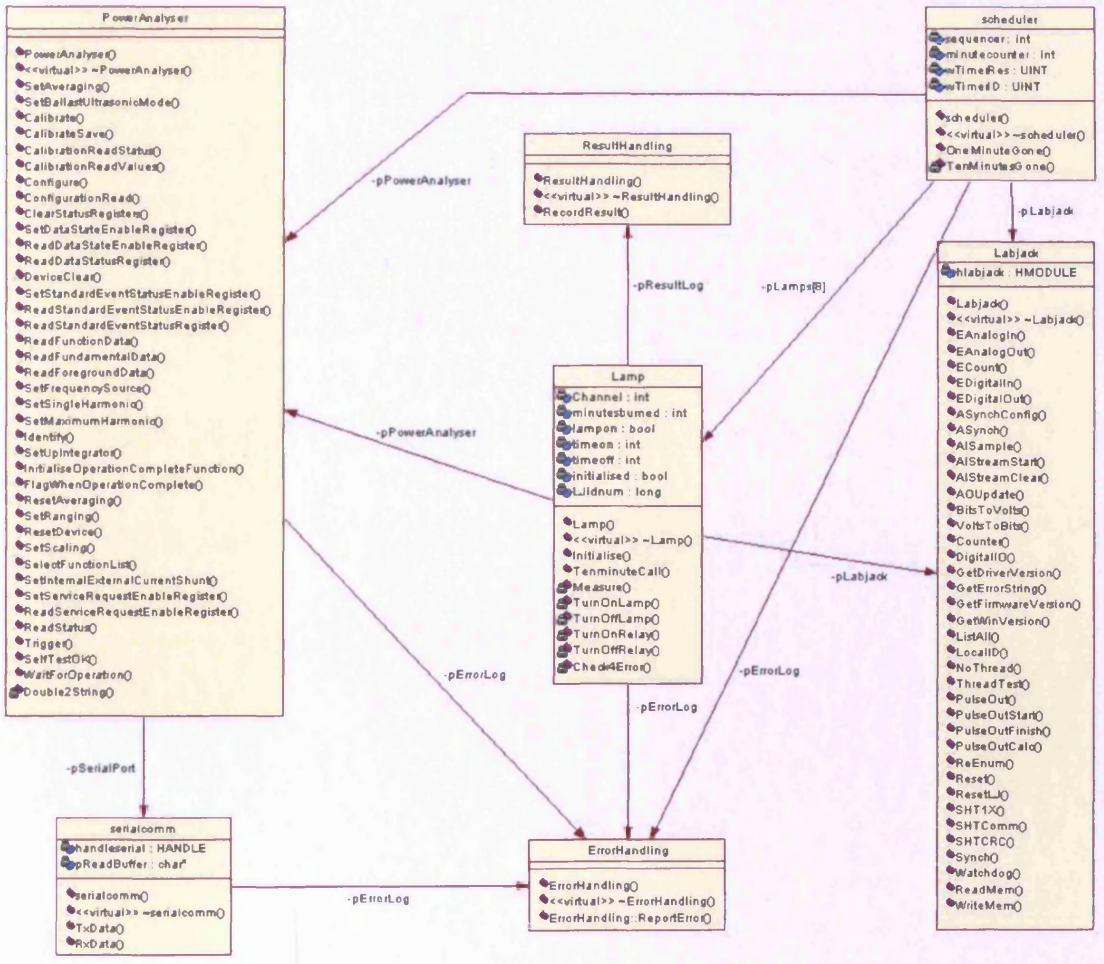


B

Appendix B - Test Rig Software

Result Recording Program Listing

UML Model



Scheduler.cpp

```
//////////////////////////////////////////////////////////////////////
// Scheduler.cpp: implementation of the Scheduler class.
//
//////////////////////////////////////////////////////////////////////

#include "Scheduler.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

// Call back function for timer
void CALLBACK OneMinuteUp( UINT wTimerID, UINT msg,
        DWORD dwUser, DWORD dw1, DWORD dw2 )
{
        scheduler* pScheduler;
        pScheduler = (scheduler*)dwUser;
        pScheduler->OneMinuteGone();
}

scheduler::scheduler(ErrorHandling* pErrorHandler)
{
        pErrorLog = pErrorHandler;
        minutecounter = 0;
        sequencer = 0;
        TIMECAPS  timecaps;  // needed by timeGetDevCaps

        if ( timeGetDevCaps( &timecaps, sizeof( TIMECAPS ) ) != TIMERR_NOERROR ) // Get max & min of sys timer
        {
                pErrorLog->ReportError(true, "Cannot find limits of timer", 0);
        }

        wTimerRes = max( timecaps.wPeriodMin, 10 );  // Set minimum resolution of 10ms
        if( timeBeginPeriod( wTimerRes ) != TIMERR_NOERROR )  // set minimal res for our timer
        {
                pErrorLog->ReportError(false, "Cannot set min timer res", 0);
        }

        // Now, run the timer
        wTimerID = timeSetEvent(60000, wTimerRes, OneMinuteUp, (DWORD)this, (TIME_PERIODIC | TIME_CALLBACK_FUNCTION));
                                                        // (delay in ms, resolution, callback function, user data, periodic timer event)

        if( wTimerID == NULL )
        {
                pErrorLog->ReportError(true, "Cannot start timer", 0);
        }

        // Create new labjack object
        long LJidnum = -1;
        pLabjack = new Labjack(pErrorLog);

        // Create new power analyser object
        pPowerAnalyser = new PowerAnalyser("COM5", "19200,n,8,1", pErrorLog);

        // Create 8 lamp objects
        for(int i = 0; i <= 7; i++)
        {
                pLamps[i] = new Lamp((i+1), pLabjack, pPowerAnalyser, pErrorLog);
        }

}


scheduler::~scheduler()
{
        timeKillEvent( wTimerID );     // Kill the timer
        wTimerID = 0;
        timeEndPeriod( wTimerRes );    // return previous setings

        if (pLabjack != NULL) delete pLabjack;
        pLabjack = NULL;

        if (pPowerAnalyser != NULL) delete pPowerAnalyser;
        pPowerAnalyser = NULL;

        // Delete the 8 lamp objects
        for(int i = 0; i <= 7; i++)
        {
                if (pLamps[i] != NULL) delete pLamps[i];
                pLamps[i] = NULL;
        }
}


// Function called every 60 seconds
void scheduler::OneMinuteGone(void)
{
        minutecounter++;
        if(minutecounter == 10)
        {
                TenMinutesGone();
                minutecounter = 0;
        }
}
```

```
void scheduler::TenMinutesGone(void)
{
	// Stagger the start of the lamps
	if (sequencer < 8)
	{
		pLamps[sequencer]->Initialise();
		sequencer++;
	}

	// Update the 8 lamp objects
	for(int i = 0; i <= 7; i++)
	{
		pLamps[i]->TenminuteCall();
	}
}
```

Scheduler.h

```
#include "ErrorHandling.h"
#include "mmsystem.h"
#include "lamp.h"

class scheduler
{
public:
	scheduler(ErrorHandling* pErrorHandler);
	virtual ~scheduler();
	void OneMinuteGone(void);


private:
	void TenMinutesGone(void);
	int sequencer;				// used to stagger the first start of the lamps
	int minutecounter;			// used to divide down to ten minutes
	UINT wTimerRes;				// timer resolturion
	UINT wTimerID;				// timer ID
	ErrorHandling* pErrorLog;
	PowerAnalyser* pPowerAnalyser;
	Labjack* pLabjack;
	Lamp* pLamps[8];			// Array of pointers to the lamps

};
```

Labjack.cpp

```
//////////////////////////////////////////////////////////////////////
// Labjack.cpp: implementation of the Labjack class.
//
//////////////////////////////////////////////////////////////////////

#include "Labjack.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

Labjack::Labjack(ErrorHandling* pErrorLog)
{
	hlabjack = NULL;
	hlabjack = LoadLibrary("ljackuw");		// Load labjack dll
	if (hlabjack == NULL) pErrorLog->ReportError(true, "Unable to load ljackuw.dll", 0); // Throw error if unable to load dll
}

Labjack::~Labjack()
{
	FreeLibrary(hlabjack);
}

long Labjack::EAnalogIn(long* idnum, long demo, long channel, long gain, long* overVoltage, float* voltage)
{
	EANALOGINDLL EAnalogInDll;
	EAnalogInDll = (EANALOGINDLL)GetProcAddress(hlabjack, "EAnalogIn");
	return EAnalogInDll(idnum, demo, channel, gain, overVoltage, voltage);
}

long Labjack::EAnalogOut(long* idnum, long demo, float analogOut0, float analogOut1)
{
	EANALOGOUTDLL EAnalogOutDll;
	EAnalogOutDll = (EANALOGOUTDLL)GetProcAddress(hlabjack, "EAnalogOut");
	return EAnalogOutDll(idnum, demo, analogOut0, analogOut1);
}

long Labjack::ECount(long* idnum, long demo, long resetCounter, double* count, double* ms)
{
	ECOUNTDLL ECountDll;
	ECountDll = (ECOUNTDLL)GetProcAddress(hlabjack, "ECount");
	return ECountDll(idnum, demo, resetCounter, count, ms);
}

long Labjack::EDigitalIn(long* idnum, long demo, long channel, long readD, long* state)
{
	EDIGITALINDLL EDigitalInDll;
	EDigitalInDll = (EDIGITALINDLL)GetProcAddress(hlabjack, "EDigitalIn");
	return EDigitalInDll(idnum, demo, channel, readD, state);
}

long Labjack::EDigitalOut(long* idnum, long demo, long channel, long writeD, long state)
{
	EDIGITALOUTDLL EDigitalOutDll;
	EDigitalOutDll = (EDIGITALOUTDLL)GetProcAddress(hlabjack, "EDigitalOut");
	return EDigitalOutDll(idnum, demo, channel, writeD, state);
}

long Labjack::ASynchConfig(long* idnum, long demo, long timeoutMult, long configA, long configB, long configTE, long fullA, long fullB, long fullC, long halfA, long halfB, long
halfC)
{
	ASYNCHCONFIGDLL ASynchConfigDll;
	ASynchConfigDll = (ASYNCHCONFIGDLL)GetProcAddress(hlabjack, "ASynchConfig");
	return ASynchConfigDll(idnum, demo, timeoutMult, configA, configB, configTE, fullA, fullB, fullC, halfA, halfB, halfC);
}

long Labjack::ASynch(long* idnum, long demo, long portB, long enableTE, long enableTO, long enableDel, long baudrate, long numWrite, long numRead, long* data)
{
	ASYNCHDLL ASynchDll;
	ASynchDll = (ASYNCHDLL)GetProcAddress(hlabjack, "ASynch");
	return ASynchDll(idnum, demo, portB, enableTE, enableTO, enableDel, baudrate, numWrite, numRead, data);
}

long Labjack::AISample(long* idnum, long demo, long* stateIO, long updateIO, long ledOn, long numChannels, long* channels, long* gains, long disableCal, long* overVoltage,
float* voltages)
{
	AISAMPLEDLL AISampleDll;
	AISampleDll = (AISAMPLEDLL)GetProcAddress(hlabjack, "AISample");
	return AISampleDll(idnum, demo, stateIO, updateIO, ledOn, numChannels, channels, gains, disableCal, overVoltage, voltages);
}

long Labjack::AIBurst(long* idnum, long demo, long stateIOin, long updateIO, long ledOn, long numChannels, long* channels, long* gains, float* scanRate, long disableCal, long
triggerIO, long triggerState, long numScans, long timeout, float* (voltages)[4], long* stateIOout, long* overVoltage, long transferMode)
{
	AIBURSTDLL AIBurstDll;
	AIBurstDll = (AIBURSTDLL)GetProcAddress(hlabjack, "AIBurst");
	return AIBurstDll(idnum, demo, stateIOin, updateIO, ledOn, numChannels, channels, gains, scanRate, disableCal, triggerIO, triggerState, numScans, timeout, voltages,
stateIOout, overVoltage, transferMode);
}
```

```
long Labjack AIStreamStart(long* idnum, long demo, long stateIOin, long updateIO, long ledOn, long numChannels, long* channels, long* gains, float* scanRate, long disableCal, long reserved1, long readCount)
{
	AISTREAMSTARTDLL AIStreamStartDll;
	AIStreamStartDll = (AISTREAMSTARTDLL)GetProcAddress(hlabjack, "AIStreamStart");
	return AIStreamStartDll(idnum, demo, stateIOin, updateIO, ledOn, numChannels, channels, gains, scanRate, disableCal, reserved1, readCount);
}

long Labjack AIStreamRead(long localID, long numScans, long timeout, float* (voltages)[4], long* stateIOout, long* reserved, long* ljScanBacklog, long* overVoltage)
{
	AISTREAMREADDLL AIStreamReadDll;
	AIStreamReadDll = (AISTREAMREADDLL)GetProcAddress(hlabjack, "AIStreamRead");
	return AIStreamReadDll(localID, numScans, timeout, voltages, stateIOout, reserved, ljScanBacklog, overVoltage);
}

long Labjack AIStreamClear(long localID)
{
	AISTREAMCLEARDLL AIStreamClearDll;
	AIStreamClearDll = (AISTREAMCLEARDLL)GetProcAddress(hlabjack, "AIStreamClear");
	return AIStreamClearDll(localID);
}

long Labjack AOUpdate(long *idnum, long demo, long trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long resetCounter, unsigned long *count, float analogOut0, float analogOut1)
{
	AOUPDATEDLL AOUpdateDll;
	AOUpdateDll = (AOUPDATEDLL)GetProcAddress(hlabjack, "AOUpdate");
	return AOUpdateDll(idnum, demo, trisD, trisIO, stateD, stateIO, updateDigital, resetCounter, count, analogOut0, analogOut1);
}

long Labjack BitsToVolts (long chnum, long chgain, long bits, float *volts)
{
	BITSTOVOLTSDLL BitsToVoltsDll;
	BitsToVoltsDll = (BITSTOVOLTSDLL)GetProcAddress(hlabjack, "BitsToVolts");
	return BitsToVoltsDll(chnum, chgain, bits, volts);
}

long Labjack VoltsToBits (long chnum, long chgain, float volts, long *bits)
{
	VOLTSTOBITSDLL VoltsToBitsDll;
	VoltsToBitsDll = (VOLTSTOBITSDLL)GetProcAddress(hlabjack, "VoltsToBits");
	return VoltsToBitsDll(chnum, chgain, volts, bits);
}

long Labjack Counter(long *idnum, long demo, long *stateD, long *stateIO, long resetCounter, long enableSTB, unsigned long *count)
{
	COUNTERDLL CounterDll;
	CounterDll = (COUNTERDLL)GetProcAddress(hlabjack, "Counter");
	return CounterDll(idnum, demo, stateD, stateIO, resetCounter, enableSTB, count);
}

long Labjack DigitalIO(long *idnum, long demo, long *trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long *outputD)
{
	DIGITALIODLL DigitalIODll;
	DigitalIODll = (DIGITALIODLL)GetProcAddress(hlabjack, "DigitalIO");
	return DigitalIODll(idnum, demo, trisD, trisIO, stateD, stateIO, updateDigital, outputD);
}

float Labjack GetDriverVersion(void)
{
	GETDRIVERVERSIONDLL GetDriverVersionDll;
	GetDriverVersionDll = (GETDRIVERVERSIONDLL)GetProcAddress(hlabjack, "GetDriverVersion");
	return GetDriverVersionDll();
}

void Labjack GetErrorString(long errorcode, char *errorString)
{
	GETERRORSTRINGDLL GetErrorStringDll;
	GetErrorStringDll = (GETERRORSTRINGDLL)GetProcAddress(hlabjack, "GetErrorString");
	GetErrorStringDll(errorcode, errorString);
	return;
}

float Labjack GetFirmwareVersion(long *idnum)
{
	GETFIRMWAREVERSIONDLL GetFirmwareVersionDll;
	GetFirmwareVersionDll = (GETFIRMWAREVERSIONDLL)GetProcAddress(hlabjack, "GetFirmwareVersion");
	return GetFirmwareVersionDll(idnum);
}

long Labjack GetWinVersion(unsigned long *majorVersion, unsigned long *minorVersion, unsigned long *buildNumber, unsigned long *platformID, unsigned long *servicePackMajor, unsigned long *servicePackMinor)
{
	GETWINVERSIONDLL GetWinVersionDll;
	GetWinVersionDll = (GETWINVERSIONDLL)GetProcAddress(hlabjack, "GetWinVersion");
	return GetWinVersionDll(majorVersion, minorVersion, buildNumber, platformID, servicePackMajor, servicePackMinor);
}

long Labjack ListAll(long *productIDList, long *serialnumList, long *localIDList, long *powerList, long (*calMatrix)[20], long *numberFound, long *fcddMaxSize, long *hvcMaxSize)
{
	LISTALLDLL ListAllDll;
	ListAllDll = (LISTALLDLL)GetProcAddress(hlabjack, "ListAll");
	return ListAllDll(productIDList, serialnumList, localIDList, powerList, calMatrix, numberFound, fcddMaxSize, hvcMaxSize);
}
```

```
long Labjack  LocalID(long *idnum, long localID)
{
        LOCALIDDLL LocalIDDll;
        LocalIDDll = (LOCALIDDLL)GetProcAddress(hlabjack, "LocalID");
        return LocalIDDll(idnum, localID);
}

long Labjack  NoThread(long *idnum, long noThread)
{
        NOTHREADDLL NoThreadDll;
        NoThreadDll = (NOTHREADDLL)GetProcAddress(hlabjack, "NoThread");
        return NoThreadDll(idnum, noThread);
}

long Labjack  ThreadTest(long timeoutms)
{
        THREADTESTDLL ThreadTestDll;
        ThreadTestDll = (THREADTESTDLL)GetProcAddress(hlabjack, "ThreadTest");
        return ThreadTestDll(timeoutms);
}

long Labjack  PulseOut(long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2)
{
        PULSEOUTDLL PulseOutDll;
        PulseOutDll = (PULSEOUTDLL)GetProcAddress(hlabjack, "PulseOut");
        return PulseOutDll(idnum, demo, lowFirst, bitSelect, numPulses, timeB1, timeC1, timeB2, timeC2);
}

long Labjack  PulseOutStart(long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2)
{
        PULSEOUTSTARTDLL PulseOutStartDll;
        PulseOutStartDll = (PULSEOUTSTARTDLL)GetProcAddress(hlabjack, "PulseOutStart");
        return PulseOutStartDll(idnum, demo, lowFirst, bitSelect, numPulses, timeB1, timeC1, timeB2, timeC2);
}

long Labjack  PulseOutFinish(long *idnum, long demo, long timeoutMS)
{
        PULSEOUTFINISHDLL PulseOutFinishDll;
        PulseOutFinishDll = (PULSEOUTFINISHDLL)GetProcAddress(hlabjack, "PulseOutFinish");
        return PulseOutFinishDll(idnum, demo, timeoutMS);
}

long Labjack  PulseOutCalc(float *frequency, long *timeB, long *timeC)
{
        PULSEOUTCALCDLL PulseOutCalcDll;
        PulseOutCalcDll = (PULSEOUTCALCDLL)GetProcAddress(hlabjack, "PulseOutCalc");
        return PulseOutCalcDll(frequency, timeB, timeC);
}

long Labjack  ReEnum(long *idnum)
{
        REENUMDLL ReEnumDll;
        ReEnumDll = (REENUMDLL)GetProcAddress(hlabjack, "ReEnum");
        return ReEnumDll(idnum);
}

long Labjack  Reset(long *idnum)
{
        RESETDLL ResetDll;
        ResetDll = (RESETDLL)GetProcAddress(hlabjack, "Reset");
        return ResetDll(idnum);
}

long Labjack  ResetLJ(long *idnum)
{
        RESETLJDLL ResetLJDll;
        ResetLJDll = (RESETLJDLL)GetProcAddress(hlabjack, "ResetLJ");
        return ResetLJDll(idnum);
}

long Labjack  SHT1X(long *idnum, long demo, long softComm, long mode, long statusReg, float *tempC, float *tempF, float *rh)
{
        SHT1XDLL SHT1XDll;
        SHT1XDll = (SHT1XDLL)GetProcAddress(hlabjack, "SHT1X");
        return SHT1XDll(idnum, demo, softComm, mode, statusReg, tempC, tempF, rh);
}

long Labjack  SHTComm(long *idnum, long softComm, long waitMeas, long serialReset, long dataRate, long numWrite, long numRead, unsigned char *datatx, unsigned char
*datarx)
{
        SHTCOMMDLL SHTCommDll;
        SHTCommDll = (SHTCOMMDLL)GetProcAddress(hlabjack, "SHTComm");
        return SHTCommDll(idnum, softComm, waitMeas, serialReset, dataRate, numWrite, numRead, datatx, datarx);
}

long Labjack  SHTCRC(long statusReg, long numWrite, long numRead, unsigned char *datatx, unsigned char *datarx)
{
        SHTCRCDLL SHTCRCDll;
        SHTCRCDll = (SHTCRCDLL)GetProcAddress(hlabjack, "SHTCRC");
        return SHTCRCDll(statusReg, numWrite, numRead, datatx, datarx);
}
```

```
long Labjack::Synch(long *idnum, long demo, long mode, long msDelay, long husDelay, long controlCS, long csLine, long csState, long configD, long numWriteRead,
        long *data)
{
        SYNCHDLL SynchDll;
        SynchDll = (SYNCHDLL)GetProcAddress(hlabjack, "Synch");
        return SynchDll(idnum, demo, mode, msDelay, husDelay, controlCS, csLine, csState, configD, numWriteRead, data);
}

long Labjack::Watchdog(long *idnum, long demo, long active, long timeout, long reset, long activeD0, long activeD1, long activeD8, long stateD0, long stateD1, long stateD8)
{
        WATCHDOGDLL WatchdogDll;
        WatchdogDll = (WATCHDOGDLL)GetProcAddress(hlabjack, "Watchdog");
        return WatchdogDll(idnum, demo, active, timeout, reset, activeD0, activeD1, activeD8, stateD0, stateD1, stateD8);
}

long Labjack::ReadMem(long *idnum, long address, long *data3, long *data2, long *data1, long *data0)
{
        READMEMDLL ReadMemDll;
        ReadMemDll = (READMEMDLL)GetProcAddress(hlabjack, "ReadMem");
        return ReadMemDll(idnum, address, data3, data2, data1, data0);
}

long Labjack::WriteMem(long *idnum, long unlocked, long address, long data3, long data2, long data1, long data0)
{
        WRITEMEMDLL WriteMemDll;
        WriteMemDll = (WRITEMEMDLL)GetProcAddress(hlabjack, "WriteMem");
        return WriteMemDll(idnum, unlocked, address, data3, data2, data1, data0);
}
```

Labjack.h

```
//////////////////////////////////////////////////////////////////////
// Labjack.h: interface for the Labjack class.
//
//////////////////////////////////////////////////////////////////////
#include "afx.h"
#include "ErrorHandling.h"

#if !defined(AFX_LABJACK_H__B887B4F9_9FE3_4DC9_B513_695F21F7BBF2__INCLUDED_)
#define AFX_LABJACK_H__B887B4F9_9FE3_4DC9_B513_695F21F7BBF2__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif

typedef long (WINAPI *EANALOGINDLL) (long* idnum, long demo, long channel, long gain, long* overVoltage, float* voltage);
typedef long (WINAPI *EANALOGOUTDLL) (long* idnum, long demo, float analogOut0, float analogOut1);
typedef long (WINAPI *ECOUNTDLL) (long* idnum, long demo, long resetCounter, double* count, double* ms);
typedef long (WINAPI *EDIGITALINDLL) (long* idnum, long demo, long channel, long readD, long* state);
typedef long (WINAPI *EDIGITALOUTDLL) (long* idnum, long demo, long channel, long writeD, long state);
typedef long (WINAPI *ASYNCHCONFIGDLL) (long* idnum, long demo, long timeoutMult, long configA, long configB, long configTE, long fullA, long fullB, long fullC, long
halfA, long halfB, long halfC);
typedef long (WINAPI *ASYNCHDLL) (long* idnum, long demo, long portB, long enableTE, long enableTO, long enableDel, long baudrate, long numWrite, long numRead, long*
data);
typedef long (WINAPI *AISAMPLEDLL) (long* idnum, long demo, long* stateIO, long updateIO, long ledOn, long numChannels, long* channels, long* gains, long disableCal,
long* overVoltage, float* voltages);
typedef long (WINAPI *AIBURSTDLL) (long* idnum, long demo, long stateIOin, long updateIO, long ledOn, long numChannels, long* channels, long* gains, float* scanRate, long
disableCal, long triggerIO, long triggerState, long numScans, long timeout, float* (voltages)[4], long* stateIOout, long* overVoltage, long transferMode);
typedef long (WINAPI *AISTREAMSTARTDLL) (long* idnum, long demo, long stateIOin, long updateIO, long ledOn, long numChannels, long* channels, long* gains, float*
scanRate, long disableCal, long reserved1, long readCount);
typedef long (WINAPI *AISTREAMREADDLL) (long localID, long numScans, long timeout, float* (voltages)[4], long* stateIOout, long* reserved, long* ljScanBacklog, long*
overVoltage);
typedef long (WINAPI *AISTREAMCLEARDLL) (long localID);
typedef long (WINAPI *AOUPDATEDLL) (long *idnum, long demo, long trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long resetCounter, unsigned long
*count, float analogOut0, float analogOut1);
typedef long (WINAPI *BITSTOVOLTSDLL) (long chnum, long chgain, long bits, float *volts);
typedef long (WINAPI *VOLTSTOBITSDLL) (long chnum, long chgain, float volts, long *bits);
typedef long (WINAPI *COUNTERDLL) (long *idnum, long demo, long *stateD, long *stateIO, long resetCounter, long enableSTB, unsigned long *count);
typedef long (WINAPI *DIGITALIODLL) (long *idnum, long demo, long *trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long *outputD);
typedef float (WINAPI *GETDRIVERVERSIONDLL) (void);
typedef void (WINAPI *GETERRORSTRINGDLL) (long errorcode, char *errorString);
typedef float (WINAPI *GETFIRMWAREVERSIONDLL) (long *idnum);
typedef long (WINAPI *GETWINVERSIONDLL) (unsigned long *majorVersion, unsigned long *minorVersion, unsigned long *buildNumber, unsigned long *platformID, unsigned
long *servicePackMajor, unsigned long *servicePackMinor);
typedef long (WINAPI *LISTALLDLL) (long *productIDList, long *serialnumList, long *localIDList, long *powerList, long (*calMatrix)[20], long *numberFound, long
*fcddMaxSize, long *hvcMaxSize);
typedef long (WINAPI *LOCALIDDLL) (long *idnum, long localID);
typedef long (WINAPI *NOTHREADDLL) (long *idnum, long noThread);
typedef long (WINAPI *THREADTESTDLL) (long timeoutms);
typedef long (WINAPI *PULSEOUTDLL) (long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2);
typedef long (WINAPI *PULSEOUTSTARTDLL) (long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2);
typedef long (WINAPI *PULSEOUTFINISHDLL) (long *idnum, long demo, long timeoutMS);
typedef long (WINAPI *PULSEOUTCALCDLL) (float *frequency, long *timeB, long *timeC);
typedef long (WINAPI *REENUMDLL) (long *idnum);
typedef long (WINAPI *RESETDLL) (long *idnum);
typedef long (WINAPI *RESETLJDLL) (long *idnum);
typedef long (WINAPI *SHT1XDLL) (long *idnum, long demo, long softComm, long mode, long statusReg, float *tempC, float *tempF, float *rh);
typedef long (WINAPI *SHTCOMMDLL) (long *idnum, long softComm, long waitMeas, long serialReset, long dataRate, long numWrite, long numRead, unsigned char *datatx,
unsigned char *datarx);
typedef long (WINAPI *SHTCRCDLL) (long statusReg, long numWrite, long numRead, unsigned char *datatx, unsigned char *datarx);
typedef long (WINAPI *SYNCHDLL) (long *idnum, long demo, long mode, long msDelay, long husDelay, long controlCS, long csLine, long csState, long configD, long
numWriteRead, long *data);
```

```
typedef long (WINAPI *WATCHDOGDLL) (long *idnum, long demo, long active, long timeout, long reset, long activeD0, long activeD1, long activeD8, long stateD0, long stateD1, long stateD8);
typedef long (WINAPI *READMEMDLL) (long *idnum, long address, long *data3, long *data2, long *data1, long *data0);
typedef long (WINAPI *WRITEMEMDLL) (long *idnum, long unlocked, long address, long data3, long data2, long data1, long data0);


class Labjack
{
public
        Labjack(ErrorHandling* pErrorLog);
        virtual ~Labjack();
        long EAnalogIn(long* idnum, long demo, long channel, long gain, long* overVoltage, float* voltage);
        long EAnalogOut(long* idnum, long demo, float analogout0, float analogout1);
        long ECount(long* idnum, long demo, long resetcounter, double* count, double* ms);
        long EDigitalIn(long* idnum, long demo, long channel, long readd, long* state);
        long EDigitalOut(long* idnum, long demo, long channel, long writed, long state);
        long ASynchConfig(long* idnum, long demo, long timeoutmult, long configa, long configb, long configte, long fulla, long fullb, long fullc, long halfa, long halfb, long halfc);
        long ASynch(long* idnum, long demo, long portb, long enablete, long enableto, long enabledel, long baudrate, long numwrite, long numread, long* data);
        long AISample(long* idnum, long demo, long* stateio, long updateio, long ledon, long numchannels, long* channels, long* gains, long disablecal, long* overvoltage, float* voltages);
        long AIBurst(long* idnum, long demo, long stateioin, long updateio, long ledon, long numchannels, long* channels, long* gains, float* scanrate, long disablecal, long triggerio, long triggerstate, long numscans, long timeout, float* (voltages)[4], long* stateioout, long* overvoltage, long transfermode);
        long AIStreamStart(long* idnum, long demo, long stateioin, long updateio, long ledon, long numchannels, long* channels, long* gains, float* scanrate, long disablecal, long reserved1, long readcount);
        long AIStreamRead(long localid, long numscans, long timeout, float* (voltages)[4], long* stateioout, long* reserved, long* ljscanbacklog, long* overvoltage);
        long AIStreamClear(long localID);
        long AOUpdate(long *idnum, long demo, long trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long resetCounter, unsigned long *count, float analogOut0, float analogOut1);
        long BitsToVolts (long chnum, long chgain, long bits, float *volts);
        long VoltsToBits (long chnum, long chgain, float volts, long *bits);
        long Counter(long *idnum, long demo, long *stateD, long *stateIO, long resetCounter, long enableSTB, unsigned long *count);
        long DigitalIO(long *idnum, long demo, long *trisD, long trisIO, long *stateD, long *stateIO, long updateDigital, long *outputD);
        float GetDriverVersion(void);
        void GetErrorString (long errorcode, char *errorString);
        float GetFirmwareVersion (long *idnum);
        long GetWinVersion(unsigned long *majorVersion, unsigned long *minorVersion, unsigned long *buildNumber, unsigned long *platformID, unsigned long *servicePackMajor, unsigned long *servicePackMinor);
        long ListAll(long *productIDList, long *serialnumList, long *localIDList, long *powerList, long (*calMatrix)[20], long *numberFound, long *fcddMaxSize, long *hvcMaxSize);
        long LocalID(long *idnum, long localID);
        long NoThread(long *idnum, long noThread);
        long ThreadTest(long timeoutms);
        long PulseOut(long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2);
        long PulseOutStart(long *idnum, long demo, long lowFirst, long bitSelect, long numPulses, long timeB1, long timeC1, long timeB2, long timeC2);
        long PulseOutFinish(long *idnum, long demo, long timeoutMS);
        long PulseOutCalc(float *frequency, long *timeB, long *timeC);
        long ReEnum(long *idnum);
        long Reset(long *idnum);
        long ResetLJ(long *idnum);
        long SHT1X(long *idnum, long demo, long softComm, long mode, long statusReg, float *tempC, float *tempF, float *rh);
        long SHTComm(long *idnum, long softComm, long waitMeas, long serialReset, long dataRate, long numWrite, long numRead, unsigned char *datatx, unsigned char *datarx);
        long SHTCRC(long statusReg, long numWrite, long numRead, unsigned char *datatx, unsigned char *datarx);
        long Synch(long *idnum, long demo, long mode, long msDelay, long husDelay, long controlCS, long csLine, long csState, long configD, long numWriteRead, long *data);
        long Watchdog(long *idnum, long demo, long active, long timeout, long reset, long activeD0, long activeD1, long activeD8, long stateD0, long stateD1, long stateD8);
        long ReadMem(long *idnum, long address, long *data3, long *data2, long *data1, long *data0);
        long WriteMem(long *idnum, long unlocked, long address, long data3, long data2, long data1, long data0);

private
        HMODULE hlabjack;
};

#endif
```

PowerAnalyser.cpp

```
//////////////////////////////////////////////////////////////////////
// PowerAnalyser.cpp: implementation of the PowerAnalyser class.
//
//////////////////////////////////////////////////////////////////////

#include "PowerAnalyser.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

PowerAnalyser::PowerAnalyser(char* gszPort, char* settings, ErrorHandling* pErrorHandler)
{
	pErrorLog = pErrorHandler;
	// Constructor for Power analyser class, indirectly opens up serial port
	pSerialPort = new serialcomm(gszPort, settings, pErrorLog);
}

PowerAnalyser::~PowerAnalyser()
{
	// Destructor for Power analyser class, indirectly closes serial port
	if(pSerialPort != NULL) delete pSerialPort;
	pSerialPort = NULL;
}

void PowerAnalyser::SetAveraging(char option[3], int NoOfMeasurements)
{
	//////////////////////////////////////////////////////////////////////
	// Set to average several measurements
	// option = AUT for Auto Range
	//
	// FIX for Fixed Range
	// Number of measurements = 0 to 16
	//////////////////////////////////////////////////////////////////////
	ASSERT((NoOfMeasurements <= 16) && (NoOfMeasurements >= 1));
	CString command = "";
	CString NoOfMeas = "";
	if(option == "AUT")		command += ":AVG:AUT ";		// Set to auto range
	if(option == "FIX")		command += ":AVG:FIX ";		// Set to fixed range
	if(command == "")
	{
		ASSERT(false);
		return;
	}
	NoOfMeas.Format("%d", NoOfMeasurements);
	command += NoOfMeas;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SetBallastUltrasonicMode(char option[3])
{
	//////////////////////////////////////////////////////////////////////
	// Set Meter to ballast and ultrasonic mode
	// option = H50 for 50Hz operation
	// H60 for 60Hz operation
	//////////////////////////////////////////////////////////////////////
	CString command = "";
	if (option == "H50") command = ":BAL:H50\r";
	if (option == "H60") command = ":BAL:H60\r";
	if (command == "")
	{
		ASSERT(false);
		return;
	}
	pSerialPort->TxData(command);
}


void PowerAnalyser::Calibrate(char option[3], int range, double value)
{
	//////////////////////////////////////////////////////////////////////
	// Perform calibration.
	//
	// Option	= VLT Calibrate voltage.
	//		= AMP Calibrate current
	//		= EXT Calibrate external shunt
	// Range	= (1 - 8)
	//
	// Value	= Double value, rms Voltage for VLT and EXT, rms Amps
	//		for AMP
	//////////////////////////////////////////////////////////////////////
	ASSERT((range <= 8) && (range >= 1));
	CString command = "";
	CString strrange = "";
	strrange.Format("%d", range);
	command = ":CAL:";
	command += option;
	command += strrange;
	command += ",";
	command += Double2String(value);
```

```
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::CalibrateSave(int password)
{
	//////////////////////////////////////////////////////////////////////
	// Save calibration settings, password = (0-9999)
	//////////////////////////////////////////////////////////////////////
	ASSERT((password <= 9999) && (password >= 0));
	CString command = "";
	CString StrPassword = "";
	command = ":CAL:END ";
	StrPassword.Format("%d", password);
	command += StrPassword;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::CalibrationReadStatus(bool& Vflag, bool& Aflag, bool& Xflag)
{
	//////////////////////////////////////////////////////////////////////
	// Function will return true for each measurement that is
	// calibrated, false if not calibrated.
	//
	// Vflag refers to the voltage input.
	// Aflag refers to the current input.
	// Xflag refers to the external shunt input.
	//////////////////////////////////////////////////////////////////////
	int posa = 0;
	int posb = 0;
	int posc = 0;
	int i = 0;
	CString command = "";
	CString values = "";
	CString strVflag = "";
	CString strAflag = "";
	CString strXflag = "";

	command = ":CAL?\r";
	pSerialPort->TxData(command);
	values = pSerialPort->RxData();

	posa = values.Find(",");				// Find first comma

	for (i = 0; i<posa; i++)
	{
		strVflag += values.GetAt(i);		// Copy first flag to Voltage
	}
	values.SetAt(posa, ' ');				// Remove comma so that next one is found
	posb = values.Find(",");				// Find second comma

	for (i = (posa+1); i<posb; i++)
	{
		strAflag += values.GetAt(i);		// Copy second flag to Current
	}
	values.SetAt(posb, ' ');				// Remove comma so that next one is found
	posc = values.GetLength();				// Find last character

	for (i = (posb+1); i<posc; i++)
	{
		strXflag += values.GetAt(i);		// Copy third flag to external Shunt
	}

	Vflag = false;
	Aflag = false;
	Xflag = false;

	if(strVflag == "255") Vflag = true;
	if(strAflag == "255") Aflag = true;
	if(strXflag == "255") Xflag = true;
}

double PowerAnalyser::CalibrationReadValues(char option[3], int range)
{
	//////////////////////////////////////////////////////////////////////
	// Returns the calibration correction value for the given channel
	// Option	= VLT Read voltage calibration
	//		= AMP Read current calibration
	//		= EXT Read external shunt calibration
	// Range can take the values 1-8
	//////////////////////////////////////////////////////////////////////
	ASSERT((range <= 8) && (range >= 1));
	CString result = "";
	CString command = ":CAL:";
	CString strRange = "";
	strRange.Format("%d", range);
	command += option;
	command += "? ";
	command += strRange;
	command += "\r";
	pSerialPort->TxData(command);
	result = pSerialPort->RxData();
	return strtod(result, NULL);
```

```
}

void PowerAnalyser::Configure(int prog, int data)
{
	//////////////////////////////////////////////////////////////////////
	// Configure operating modes of analyser
	// Prog		= integer (0 - 49)
	//
	// Mode = Appropriate integer or floating point data
	//////////////////////////////////////////////////////////////////////
	ASSERT((prog <= 49) && (prog >= 0));
	CString command = "";
	CString strprog = "";
	CString strdata = "";
	strprog.Format("%d", prog);
	strdata.Format("%d", data);
	command = ":CFG ";
	command += strprog;
	command += ", ";
	command += strdata;
	command += "\r";
	pSerialPort->TxData(command);
}

double PowerAnalyser::ConfigurationRead(int prog)
{
	//////////////////////////////////////////////////////////////////////
	// Returns configuration as double
	// prog		= Integer number (0 - 49)
	//////////////////////////////////////////////////////////////////////
	ASSERT((prog <= 49) && (prog >= 0));
	CString command = "";
	CString strprog = "";
	CString data = "";
	strprog.Format("%d", prog);
	command = ":CFG? ";
	command += strprog;
	command += "\r";
	pSerialPort->TxData(command);
	data = pSerialPort->RxData();
	return strtod(data, NULL);
}

void PowerAnalyser::ClearStatusRegisters(void)
{
	//////////////////////////////////////////////////////////////////////
	// Clear Standard Status Register and Data Status Register
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*CLS\r");
}

void PowerAnalyser::SetDataStateEnableRegister(int DSEData)
{
	//////////////////////////////////////////////////////////////////////
	// Set data status enable register (0-255)
	//////////////////////////////////////////////////////////////////////
	ASSERT((DSEData <= 255) && (DSEData >= 0));
	CString command = "";
	CString number = "";
	number.Format("%d", DSEData);
	command = ":DSE ";
	command += number;
	command += "\r";
	pSerialPort->TxData(command);
}

int PowerAnalyser::ReadDataStateEnableRegister(void)
{
	//////////////////////////////////////////////////////////////////////
	// Retrieve current value held in data status enable register(0-255)
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData(":DSE?\r");
	CString DSEData;
	DSEData = pSerialPort->RxData();
	int DataValue = _ttoi(DSEData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}

int PowerAnalyser::ReadDataStatusRegister(void)
{
	//////////////////////////////////////////////////////////////////////
	// Retrieve current value held in data status register (0-255).
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData(":DSR?\r");
	CString DSRData;
	DSRData = pSerialPort->RxData();
	int DataValue = _ttoi(DSRData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}
```

```
void PowerAnalyser::DeviceClear(void)
{
	//////////////////////////////////////////////////////////////////////
	// Resets the PM100 analyser.
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData(":DVC\r");
}

void PowerAnalyser::SetStandardEventStatusEnableRegister(int ESEData)
{
	//////////////////////////////////////////////////////////////////////
	// Load int value into the standard event status enable register,
	// Accepts int (0-255)
	//////////////////////////////////////////////////////////////////////
	ASSERT((ESEData <= 255) && (ESEData >= 0));
	CString command = "";
	CString number = "";
	number.Format("%d", ESEData);
	command = "*ESE ";
	command += number;
	command += "\r";
	pSerialPort->TxData(command);
}

int PowerAnalyser::ReadStandardEventStatusEnableRegister(void)
{
	//////////////////////////////////////////////////////////////////////
	// Retrieve current value in standard event status enable register.
	// Returns int (0-255)
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*ESE?\r");
	CString ESEData;
	ESEData = pSerialPort->RxData();
	int DataValue = _ttoi(ESEData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}

int PowerAnalyser::ReadStandardEventStatusRegister(void)
{
	//////////////////////////////////////////////////////////////////////
	// Retrieve current value held in standard event status register
	// Returns the integer value held in the register (0 - 255)
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*ESR?\r");
	CString ESRData;
	ESRData = pSerialPort->RxData();
	int DataValue = _ttoi(ESRData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}

double PowerAnalyser::ReadFunctionData(char option[3])
{
	//////////////////////////////////////////////////////////////////////
	// Read Function data
	// Option	= WAT Watts
	//		= VAS VA
	//		= VAR VAr
	//		= VLT V rms
	//		= AMP A rms
	//		= PWF Power Factor
	//		= VPK V Peak
	//		= APK A Peak
	//		= VCF V Crest Factor
	//		= ACF A Crest Factor
	//		= WHR Watt Hour
	//		= VAH VA Hour
	//		= VRH VAr Hour
	//		= AHR Amp Hour
	//		= APF Average Power Factor
	//		= VHM V harm magnitude
	//		= AHM A harm magnitude
	//		= WHM W harm magnitude
	//		= VDF V THD
	//		= ADF A THD
	//		= FRQ Frequency
	//		= VDC V dc
	//		= ADC A dc
	//		= VHA V harm angle
	//		= AHA A harm angle
	//		= RES Resistance
	//		= IMP Impedance
	//		= REA Reactance
	// Returns one double result.
	//////////////////////////////////////////////////////////////////////
	CString data = "";
	CString command = ":FNC:";
	command += option;
	command += "?\r";
	pSerialPort->TxData(command);
	data = pSerialPort->RxData();
```

```
	return strtod(data, NULL);
}

double PowerAnalyser::ReadFundamentalData(char option[3])
{
	////////////////////////////////////////////////////////////////////
	// Returns one double number for the value requested.
	// Option	= WAT, VAS, VAR, VLT, AMP, PWF, WHR, VAH, VRH, AHR, APF
	// Data read can be synchronised to new data values by using the
	// NDV bit in the data status register
	////////////////////////////////////////////////////////////////////
	CString data = "";
	CString command = ":FND:";
	command += option;
	command += "?\r";
	pSerialPort->TxData(command);
	data = pSerialPort->RxData();
	return strtod(data, NULL);
}

double PowerAnalyser::ReadForegroundData(void)
{
	////////////////////////////////////////////////////////////////////
	// Data sent is determined by the previously stored selection using
	// SelectFunctionList().
	//
	// Repeated use returns subsequent measurements and do not repeat
	// the same data.
	// Returns one double number.
	////////////////////////////////////////////////////////////////////
	CString data = "";
	CString command = ":FRD?\r";
	pSerialPort->TxData(command);
	data = pSerialPort->RxData();
	return strtod(data, NULL);
}

void PowerAnalyser::SetFrequencySource(char option[3])
{
	////////////////////////////////////////////////////////////////////
	// Set Frequency Source
	//
	// When the input signal is complex, the PM100 analyser locks onto
	// the lowest frequency.
	//
	// Option	= AUT Set auto frequency source
	//		= VLT Set voltage frequency source
	//		= AMP Set current frequency source
	////////////////////////////////////////////////////////////////////
	CString command = ":FSR:";
	if (option == "AUT") command += "AUT\r";
	if (option == "VLT") command += "FIX:VLT\r";
	if (option == "AMP") command += "FIX:AMP\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SetSingleHarmonic(int harmonic)
{
	////////////////////////////////////////////////////////////////////
	// Set single harmonic
	//
	// Harmonic	= integer harmonic number (0 - 50)
	////////////////////////////////////////////////////////////////////
	ASSERT((harmonic <= 50) && (harmonic >= 0));
	CString command = "";
	CString number = "";
	number.Format("%d", harmonic);
	command = ":HRM ";
	command += number;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SetMaximumHarmonic(char option[3], int harmonic)
{
	////////////////////////////////////////////////////////////////////
	// Set Maximum Harmonic For Series
	// Both ODD and ALL series start with harmonic 0, the DC component.
	// Therefore the ODD series is 0,1,3,5.. and the ALL is 0,1,2,3..
	// Option	= ODD Only odd harmonics
	//		= ALL Use both odd and even harmonics
	// Harmonic = Integer maximum harmonic (1 - 50)
	////////////////////////////////////////////////////////////////////
	ASSERT((harmonic <= 50) && (harmonic >= 1));
	CString strHarmonic = "";
	strHarmonic.Format("%d", harmonic);
	CString command = ":HMX:";
	command += option;
	command += strHarmonic;
	command += "\r";
	pSerialPort->TxData(command);
}
```

```
CString PowerAnalyser::Identify(void)
{
	//////////////////////////////////////////////////////////////////
	// Returns CString containing a description of the analyser
	//////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*IDN?\r");
	return pSerialPort->RxData();
}

void PowerAnalyser::SetUpIntegrator(char option[3], int time)
{
	//////////////////////////////////////////////////////////////////
	// Set up integrator and start running
	// Options	= RUN, time		Run time in integer minutes
	//		= ENB			Start integrator with previous time
	//		= DIS			Stop integrator
	//////////////////////////////////////////////////////////////////
	CString strTime = "";
	strTime.Format("%d", time);
	CString command = ":INT:";
	command += option;
	if (option == "RUN") command += strTime;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::InitialiseOperationCompleteFunction(void)
{
	//////////////////////////////////////////////////////////////////
	// Clears the NDV bit, NDV bit will be set again when new data is
	// available, use FlagWhenOperationComplete() to check.
	//////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*OPC?\r");
}

bool PowerAnalyser::FlagWhenOperationComplete(void)
{
	//////////////////////////////////////////////////////////////////
	// Use with InitialiseOperationCompleteFunction(), will return true
	// when new data is available.
	//////////////////////////////////////////////////////////////////
	if (pSerialPort->RxData() == "1")
	{
		return true;
	}
	else
	{
		return false;
	}
}

void PowerAnalyser::ResetAveraging(void)
{
	//////////////////////////////////////////////////////////////////
	// This command can be used to speed up the response of the
	// instrument to step changes especially when in fixed averaging.
	//////////////////////////////////////////////////////////////////
	pSerialPort->TxData(":RAV\r");
}

void PowerAnalyser::SetRanging(char option1[3], char option2[3], int range)
{
	//////////////////////////////////////////////////////////////////
	// Set fixed or auto ranging for each input
	// Option 1 =	VLT Set Voltage ranging
	//		AMP Set Current ranging
	// Option 2 =	FIX Fixed Range
	//		AUT Auto Range
	// Range =	1 - 8  (Not used if auto range specified)
	//////////////////////////////////////////////////////////////////
	ASSERT((range <= 8) && (range >= 1));
	CString strRange = "";
	strRange.Format("%d", range);
	CString command = ":RNG:";
	command += option1;
	command += ":";
	command += option2;
	if (option2 == "FIX") command += strRange;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::ResetDevice(void)
{
	//////////////////////////////////////////////////////////////////
	// Reset Power Analyser
	//////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*RST\r");
}
```

```
void PowerAnalyser::SetScaling(char option[3], double scale)
{
	//////////////////////////////////////////////////////////////////////
	// Set scaling for each input
	// Option	= VLT Set Voltage Scaling
	//		= AMP Set Current Scaling
	// Scale	= Number with decimal point, e.g. 0.1, 1.0 etc
	//////////////////////////////////////////////////////////////////////
	CString strScale;
	strScale.Format("%f", scale);
	strScale.TrimRight("0");
	CString command = ":SCL:";
	command += option;
	command += " ";
	command += strScale;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SelectFunctionList(char option[3])
{
	//////////////////////////////////////////////////////////////////////
	// Select the list of parameters for return by ReadForegroundData()
	// Option	= CLR Clear Entire Selection
	//		= CH1 Channel 1
	//		= WAT Watts
	//		= VAS VA
	//		= VAR VAr
	//		= VLT V rms
	//		= AMP A rms
	//		= PWF Power Factor
	//		= VPK V Peak
	//		= APK A Peak
	//		= VCF V Crest Factor
	//		= ACF A Crest Factor
	//		= WHR Watt Hour
	//		= VAH VA Hour
	//		= VRH VAr Hour
	//		= AHR Amp Hour
	//		= APF Average Power Factor
	//		= VHM V harm magnitude
	//		= AHM A harm magnitude
	//		= WHM W harm magnitude
	//		= VDF V THD
	//		= ADF A THD
	//		= FRQ Frequency
	//		= VDC V dc
	//		= ADC A dc
	//		= VHA V harm angle
	//		= AHA A harm angle
	//		= FND Include Fundamentals in the list
	//		= SER Include the harmonic series defined by SetMaxHarm
	//		= RES Resistance
	//		= IMP Impedance
	//		= REA Reactance
	//////////////////////////////////////////////////////////////////////
	CString command = ":SEL:";
	command += option;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SetInternalExternalCurrentShunt(char option[3])
{
	//////////////////////////////////////////////////////////////////////
	// Set whether using an external or the internal current shunt
	// Option	= INT Use Internal Current Shunt
	//		= EXT Use External Current Shunt
	//////////////////////////////////////////////////////////////////////
	CString command = ":SHU:";
	command += option;
	command += "\r";
	pSerialPort->TxData(command);
}

void PowerAnalyser::SetServiceRequestEnableRegister(int SREData)
{
	//////////////////////////////////////////////////////////////////////
	// Set service request enable register with the integer value
	// (0-255)
	//////////////////////////////////////////////////////////////////////
	ASSERT((SREData <= 255) && (SREData >= 0));
	CString command = "*SRE ";
	CString number = "";
	number.Format("%d", SREData);
	command += number;
	command += "\r";
	pSerialPort->TxData(command);
}

int PowerAnalyser::ReadServiceRequestEnableRegister(void)
{
	//////////////////////////////////////////////////////////////////////
```

```
	// Retrieves current value held in the service request enable
	// register. Returns integer in range 0-255
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*SRE?\r");
	CString SREData;
	SREData = pSerialPort->RxData();
	int DataValue = _ttoi(SREData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}

int PowerAnalyser::ReadStatus(void)
{
	//////////////////////////////////////////////////////////////////////
	// Returns the status register as an integer (0-255)
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*STB?\r");
	CString STBData;
	STBData = pSerialPort->RxData();
	int DataValue = _ttoi(STBData);
	ASSERT((DataValue <= 255) && (DataValue >= 0));
	return DataValue;
}

void PowerAnalyser::Trigger(void)
{
	//////////////////////////////////////////////////////////////////////
	// Resets the averaging, and starts the new measurement
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*TRG\r");
}

bool PowerAnalyser::SelfTestOK(void)
{
	//////////////////////////////////////////////////////////////////////
	// Query status of self test, true = OK, false = Error
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*TST?\r");
	CString data = pSerialPort->RxData();
	if ((data == "1") || (data.GetLength == 0))
	{
		return false;
	}
	else
	{
		return true;
	}
}

void PowerAnalyser::WaitForOperation(void)
{
	//////////////////////////////////////////////////////////////////////
	// The operation complete flag is set when new data is available,
	// this function will then effect a delay until data is available.
	//////////////////////////////////////////////////////////////////////
	pSerialPort->TxData("*WAI\r");
}


CString PowerAnalyser::Double2String(double number)
{
	//////////////////////////////////////////////////////////////////////
	// Accepts a double number, returns CString in the following format
	// d.dddEdd
	//////////////////////////////////////////////////////////////////////
	int power = 0;
	int dec = 0;
	int sign = 0;
	CString strnumber = "";
	CString strpower = "";
	CString strsign = "";
	CString stranswer = "";

	//Convert number to positive if negative, whilst storing sign
	if (number < 0.0)
	{
		strsign = "-";
		number = -number;
	}


	//Find power for large numbers
	if (number > 1.0)
	{
		while (number >= 10.0)
		{
			number = number / 10.0;
			power = power + 1;
		}
	}
	//Find power for small numbers
	else
	{
		while ((number < 1.0) && (number > 0))
		{
			number = number * 10.0;
			power = power - 1;
```

```
                }
        }

        ASSERT (power < 100);

        // Convert the remaining number to a string, round to 4d.p.'s
        strnumber = _ecvt(number, 5, &dec, &sign);
        // Insert decimal point which was lost during conversion into string
        strnumber.Insert(1, ".");

        // Convert power to string format

        strpower.Format("%d", power);
        stranswer = strsign;
        stranswer += strnumber;
        stranswer += "E";
        stranswer += strpower;

        return stranswer;
}
```

PowerAnalyser.h

```
#include "SerialIO.h"
#include "ErrorHandling.h"
#include "strstrea.h"
#include "math.h"

class PowerAnalyser
{
public:
        PowerAnalyser(char* gsvPort, char* settings, ErrorHandling* pErrorHandler);
        virtual ~PowerAnalyser();
        void SetAveraging(char option[3], int NoOfMeasurments);
        void SetBallastUltrasonicMode(char option[3]);
        void Calibrate(char option[3], int range, double value);
        void CalibrateSave(int password);
        void CalibrationReadStatus(bool& Vflag, bool& Aflag, bool& Cflag);
        double CalibrationReadValues(char option[3], int range);
        void Configure(int prog, int data);
        double ConfigurationRead(int prog);
        void ClearStatusRegisters(void);
        void SetDataStateEnableRegister(int DSEData);
        int ReadDataStateEnableRegister(void);
        int ReadDataStatusRegister(void);
        void DeviceClear(void);
        void SetStandardEventStatusEnableRegister(int ESEData);
        int ReadStandardEventStatusEnableRegister(void);
        int ReadStandardEventStatusRegister(void);
        double ReadFunctionData(char option[3]);
        double ReadFundamentalData(char option[3]);
        double ReadForegroundData(void);
        void SetFrequencySource(char option[3]);
        void SetSingleHarmonic(int harmonic);
        void SetMaximumHarmonic(char option[3], int harmonic);
        CString Identify(void);
        void SetUpIntegrator(char option[3], int time);
        void InitialiseOperationCompleteFunction(void);
        bool FlagWhenOperationComplete(void);
        void ResetAveraging(void);
        void SetRanging(char option1[3], char option2[3], int range);
        void ResetDevice(void);
        void SetScaling(char option[3], double scale);
        void SelectFunctionList(char option[3]);
        void SetInternalExternalCurrentShunt(char option[3]);
        void SetServiceRequestEnableRegister(int SREData);
        int ReadServiceRequestEnableRegister(void);
        int ReadStatus(void);
        void Trigger(void);
        bool SelfTestOK(void);
        void WaitForOperation(void);


private:
        serialcomm* pSerialPort;
        ErrorHandling* pErrorLog;
        CString Double2String(double number);
};
```

SerialIO.cpp

```
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

serialcomm::serialcomm(char* gszPort, char* settings, ErrorHandling* pErrorHandler)
{
	pErrorLog = pErrorHandler;
	// Buffer for data read in
	pReadBuffer = new char[500];

	DeviceControlBlock = NULL;
	handleserial = NULL;

	// Open COM port
	handleserial = CreateFile(gszPort, GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING, NULL, 0);
	if(handleserial == INVALID_HANDLE_VALUE)
	{
		// Unable to obtain handle on serial port, report and terminate
		pErrorLog->ReportError(true, "Unable to open COM port", GetLastError());
	}

	// Define a Device Control Block for the COM port and set baud rate etc.
	DeviceControlBlock = new DCB;
	FillMemory(DeviceControlBlock, sizeof(*DeviceControlBlock), 0);
	DeviceControlBlock->DCBlength = sizeof(*DeviceControlBlock);
	if(!BuildCommDCB(settings, DeviceControlBlock))
	{
		// DCB could not be built, report and terminate
		pErrorLog->ReportError(true,"COM Port DCB could not be built", GetLastError());
	}

	// Specify timeouts for send and receive
	timeout = new COMMTIMEOUTS;
	timeout->ReadIntervalTimeout = 1;
	timeout->ReadTotalTimeoutConstant = 500;
	timeout->ReadTotalTimeoutMultiplier = 1;
	timeout->WriteTotalTimeoutConstant = 500;
	timeout->WriteTotalTimeoutMultiplier = 1;
	if(!SetCommTimeouts(handleserial, timeout))
	{
		// Timeout values could not be set
		pErrorLog->ReportError(false, "Unable to set COM port timeout values", GetLastError());
	}
}


serialcomm::~serialcomm()
{
	// Close COM port
	if(!CloseHandle(handleserial))
	{
		// Handle on serial port could not be closed
		pErrorLog->ReportError(false, "Unable to close handle on serial port", GetLastError());
	}
	// Delete DCB
	if(DeviceControlBlock != NULL) delete DeviceControlBlock;
	DeviceControlBlock = NULL;
	if(timeout != NULL) delete timeout;
	timeout = NULL;
	if(pReadBuffer != NULL) delete pReadBuffer;
	pReadBuffer = NULL;
}


void serialcomm::TxData(CString data)
{
	unsigned long byteswritten;
	long length = data.GetLength();

	if(!WriteFile(handleserial, data, length, &byteswritten, NULL))
	{
		// Write to file failed
		pErrorLog->ReportError(false, "Warning:- Unable to write to COM port", GetLastError());
		return;
	}

	if((long)byteswritten != data.GetLength())
	{
		// Warning - data sent to port was more than what was transferred
		pErrorLog->ReportError(false, "Data written to COM port was less than that sent", 0);
		return;
	}
	return;
}


CString serialcomm::RxData()
{
	CString DataRead;
	unsigned long BytesRead;
	// Read bytes in synchronously
	if(!ReadFile(handleserial, pReadBuffer , 500, &BytesRead, NULL))
	{
		pErrorLog->ReportError(false, "Failure reading from COM Port", GetLastError());
```

```
        }

        for(int i = 0; i<((int)BytesRead); i++)
        {
                DataRead += pReadBuffer[i];
        }
        return DataRead;
}
```

SerialIO.h

```
#include "afx.h"
#include "ErrorHandling.h"


class serialcomm
{
public:
        serialcomm(char* gsvPort, char* settings, ErrorHandling* pErrorHandler);
        virtual ~serialcomm();
        void TxData(CString data);
        CString RxData();

private:
        HANDLE handleserial;                    // File Handle for opened COM Port
        DCB* DeviceControlBlock;                // Pointer to DCB block for COM Port
        COMMTIMEOUTS* timeout;                  // Pointer to timeout limits on communication
        ErrorHandling* pErrorLog;               // Pointer to object for dealing with errors
        char* pReadBuffer;                      // Pointer to reading buffer
};
```

ErrorHandling.cpp

```
//////////////////////////////////////////////////////////////////////
// ErrorHandling.cpp: implementation of the ErrorHandling class.
//
//////////////////////////////////////////////////////////////////////

#include "ErrorHandling.h"
#include "fstream.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

ErrorHandling::ErrorHandling()
{
	// Create or open new file object for logging errors
	char* pFileName = "errorlog.txt";
	pErrorFile = NULL;
	pErrorFile = new ofstream;

	// Retrieve current time and date
	char* pTimeString = NULL;
	CTime thetime = CTime::GetCurrentTime();
	CString timestring;
	timestring = thetime.Format( "%d%b%Y %H:%M  " );

	// Open Error log file for writing by this process only, if not in existance create new file
	pErrorFile->open(pFileName, ios::app | ios::in | ios::out, 0);//filebuf::sh_read );

	if(pErrorFile->fail())
	{
		// File not opened, delete nearby objects and terminate immediately
		cout << timestring << "Error:- Unable to open or create Log File" << endl;
		delete pErrorFile;
		pErrorFile = NULL;
		exit(1);
	}
	else
	{
		// File opened successfully, carry on
		cout << "************** " << timestring << "Log File Opened Successfully **************" << endl;
		*pErrorFile << "************** " << timestring << "Log File Opened Successfully **************" << endl;
	}
}

ErrorHandling::~ErrorHandling()
{
	//Close file etc
	if (pErrorFile != NULL)
	{
		pErrorFile->flush();
		pErrorFile->close();
		delete pErrorFile;
		pErrorFile = NULL;
	}
}

ErrorHandling::ReportError(bool terminate, CString errordescription, long errorcode)
{
	// Retrieve current time and date
	CTime thetime = CTime::GetCurrentTime();
	CString timestring;
	timestring = thetime.Format( "%d%b%Y %H:%M %S  " );

	if (terminate == true)
	{
		// Write error to log file
		*pErrorFile << timestring << "Error:- " << errordescription << ", Code:- "<< errorcode << endl;
		// Copy error to screen
		cout << timestring << "Error:- " << errordescription << ", Code:- "<< errorcode << endl;
		exit(1);
	}
	else
	{
		// Write error to log file
		*pErrorFile << timestring << "Warning:- " << errordescription << ", Code:- "<< errorcode << endl;
		// Copy error to screen
		cout << timestring << "Warning:- " << errordescription << ", Code:- "<< errorcode << endl;
	}
}
```

ErrorHandling.h

```
//////////////////////////////////////////////////////////////////////
// ErrorHandling.h: interface for the ErrorHandling class.
//
//////////////////////////////////////////////////////////////////////
#include "afx.h"
#include "fstream.h"


#if !defined(AFX_ERRORHANDLING_H__E882C7E3_847E_4B49_95B2_D294E221AD25__INCLUDED_)
#define AFX_ERRORHANDLING_H__E882C7E3_847E_4B49_95B2_D294E221AD25__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class ErrorHandling
{
public:
		ErrorHandling();
		virtual ~ErrorHandling();
		ErrorHandling::ReportError(bool terminate, CString errordescription, long errorcode);
private:
		ofstream* pErrorFile;
};

#endif // !defined(AFX_ERRORHANDLING_H__E882C7E3_847E_4B49_95B2_D294E221AD25__INCLUDED_)
```

Lamp.cpp

```
//////////////////////////////////////////////////////////////////////
// Lamp.cpp: implementation of the Lamp class.
//
//////////////////////////////////////////////////////////////////////

#include "lamp.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

Lamp::Lamp(int chnum, Labjack* pLabj, PowerAnalyser* pPowerMeter, ErrorHandling* pErrorHandler)
{
	ASSERT((chnum >= 1) && (chnum <= 8));
	Channel = chnum;                                    // Initialise known variables
	pLabjack = pLabj;
	pPowerAnalyser = pPowerMeter;
	pErrorLog = pErrorHandler;
	LJidnum = -1;                                       // Set labjack id to first found

	TurnOffRelay(Channel);                              // Turn off lamp
	TurnOffRelay(Channel+8);
	minutesburned = 0;                                  // Initialize all yet to be used variables
	timeoff = 0;
	timeon = 0;
	lampon = false;
	initialised = false;


	// Generate filename for the result file for this channel
	CString strChannel = "";
	CString	filename = "LampChannel";
	strChannel.Format("%d", Channel);
	filename += strChannel;
	filename += ".txt";

	// Create new result handler object and file for this lamp channel
	pResultLog = new ResultHandling(filename.GetBuffer(20));
	filename.ReleaseBuffer();

}

Lamp::~Lamp()
{
	if(pResultLog != NULL) delete pResultLog;
	pResultLog = NULL;                                  // Destroy the corresponding file object
}


void Lamp::Initialise(void)
{
	if (initialised = true) return;                     // Skip this step if already initialised
	initialised = true;
	TurnOnLamp();
}


void Lamp::TenminuteCall(void)
{
	if (!initialised) return;                           // Check channel is up and running
	minutesburned = minutesburned + 10;                 // Update number of minutes lamp has been on for
	if (lampon == true)
	{
		timeon = timeon + 10;
		if (timeon >= 180)                              // Check to see if it is time yet to turn off lamp
		{
			TurnOffLamp();                          // Turn off lamp
		}
		else
		{
			Measure();                              // If execution reaches here lamp is still on, take routine measurement
		}
	}
	else
	{
		timeoff = timeoff + 10;
		if (timeoff >= 10)                              // Check to see if it is time yet to turn on lamp
		{
			TurnOnLamp();                           // Turn on lamp
		}
	}

}


void Lamp::Measure()
{
	long overvoltage = 0;
	float voltage = 0;

	// Switch ammeter to channel of interest
	TurnOnRelay(Channel);
```

```
		TurnOffRelay(Channel+8);

		// Set bits of multiplexer to select input channel according to channel number
		pLabjack->EDigitalOut(&LJidnum, 0, 0, 0, 0);
		pLabjack->EDigitalOut(&LJidnum, 0, 1, 0, 0);
		pLabjack->EDigitalOut(&LJidnum, 0, 2, 0, 0);
		pLabjack->EDigitalOut(&LJidnum, 0, 3, 0, 1);		// Enable output of analogue multiplexer
		if((Channel - 1) & 4) pLabjack->EDigitalOut(&LJidnum, 0, 2, 0, 1);  //MSB
		if((Channel - 1) & 2) pLabjack->EDigitalOut(&LJidnum, 0, 1, 0, 1);
		if((Channel - 1) & 1) pLabjack->EDigitalOut(&LJidnum, 0, 0, 0, 1); //LSB

		pResultLog->RecordResult("Measurement Cycle Start, Minutes Burned:- ", (double)minutesburned);

		// Read temperature on analogue channel 0
		Check4Error(pLabjack->EAnalogIn(&LJidnum, 0, 0, 0, &overvoltage, &voltage));
		if(overvoltage > 0) pErrorLog->ReportError(false, "LJ Overvoltage on AIChannel 0", Channel);
		pResultLog->RecordResult("Lamp Temp Sens Voltage", (double)voltage); // Record Result

		// Read light level on analogue channel 1
		Check4Error(pLabjack->EAnalogIn(&LJidnum, 0, 1, 0, &overvoltage, &voltage));
		if(overvoltage > 0) pErrorLog->ReportError(false, "LJ Overvoltage on AIChannel 1", Channel);
		pResultLog->RecordResult("Lamp Light Sens Voltage", (double)voltage); // Record Result

		// Read ambient temperature on analogue channel 2
		Check4Error(pLabjack->EAnalogIn(&LJidnum, 0, 2, 0, &overvoltage, &voltage));
		if(overvoltage > 0) pErrorLog->ReportError(false, "LJ Overvoltage on AIChannel 2", Channel);
		pResultLog->RecordResult("Room Temp Sens Voltage", (double)voltage); // Record Result

		// Record Lamp Voltage
		double volts = pPowerAnalyser->ReadFunctionData("VLT");
		pResultLog->RecordResult("Volts AC", volts);

		// Record Lamp Current
		double amps = pPowerAnalyser->ReadFunctionData("AMP");
		pResultLog->RecordResult("Amps AC", amps);

		// Record Lamp Power
		double power = pPowerAnalyser->ReadFundamentalData("WAT");
		pResultLog->RecordResult("Watts", power);

		// Record Lamp Power Factor
		double PFactor = pPowerAnalyser->ReadFundamentalData("PWF");
		pResultLog->RecordResult("Power Factor", PFactor);

		// Switch channel back to bypass
		TurnOnRelay(Channel+8);
		TurnOffRelay(Channel);

		// Mark end of measurements
		pResultLog->RecordResult("----- End of Measurement Cycle -----", 0.0);

}

void Lamp::TurnOnLamp(void)                                     // Switch on lamp
{
		if (lampon == true) return;                             // Return if lamp already on

		pPowerAnalyser->Configure(1,1);                         // Set analyser to inrush current mode
		pPowerAnalyser->SetRanging("AMP","FIX",6);              // Set analyser to fixed current range 24.3A max

		TurnOnRelay(Channel);

		double amps = pPowerAnalyser->ReadFunctionData("AMP");  // Read inrush current
		pResultLog->RecordResult("----- Lamp Turned On, Inrush Current", amps);

		pPowerAnalyser->SetRanging("AMP","AUT",8);              // Return analyser to auto range mode

		TurnOnRelay(Channel+8);                                 // Put channel onto bypass
		TurnOffRelay(Channel);
		lampon = true;                                          // Update variables
		timeoff = 0;
		timeon = 0;

		Measure();                                              // Take measurements imediately after startup
}

void Lamp::TurnOffLamp(void)                                    // Turn off lamp for rest
{
		if (lampon == false) return;                            // Return if lamp already off
		Measure();                                              // Take last measurements before turning off
		TurnOffRelay(Channel+8);
		lampon = false;                                         // Update status variables
		timeon = 0;
		timeoff = 0;
		pResultLog->RecordResult("----- Lamp Turned Off -----", 0.0);
}

void Lamp::TurnOffRelay(int RelayNo)
{
		ASSERT((RelayNo >= 1) && (RelayNo <= 16));
		Check4Error(pLabjack->EDigitalOut(&LJidnum, 0, (RelayNo - 1), 1, 0));
}

void Lamp::TurnOnRelay(int RelayNo)
{
		// Turn on relay Ammeter relay's = 1-8, Bypass relay's = 9-16.
		ASSERT((RelayNo >= 1) && (RelayNo <= 16));
```

```
		Check4Error(pLabjack->EDigitalOut(&LJidnum, 0, (RelayNo - 1), 1, 1));
}

void Lamp::Check4Error(long errorcode)
{
		if (errorcode != 0)
		{
			CString errorstring;
			CString errorstring1 = "LJack ";
			pLabjack->GetErrorString(errorcode, errorstring.GetBuffer(200));
			errorstring.ReleaseBuffer();
			errorstring1 += errorstring;
			pErrorLog->ReportError(false, errorstring1, errorcode);
		}
}
```

Lamp.h

```
#include "ErrorHandling.h"
#include "PowerAnalyser.h"
#include "Labjack.h"
#include "ResultHandling.h"

class Lamp
{
public:
		Lamp(int chnum, Labjack* pLabj, PowerAnalyser* pPowerMeter, ErrorHandling* pErrorHandler);
		virtual ~Lamp();
		void Initialise(void);
		void TenminuteCall(void);

private:
		int Channel;                                    // Lamp number
		int minutesburned;                              // Total time lamp has run
		bool lampon;                                    // Flag to show if lamp on
		int timeon;                                     // Used to keep track of switching times
		int timeoff;                                    // Used to keep track of switching times
		bool initialised;                               // Used to monitor startup of channel
		void Measure(void);
		void TurnOnLamp(void);
		void TurnOffLamp(void);
		void TurnOnRelay(int RelayNo);
		void TurnOffRelay(int RelayNo);
		void Check4Error(long errorcode);
		Labjack* pLabjack;
		long LJidnum;
		PowerAnalyser* pPowerAnalyser;
		ErrorHandling* pErrorLog;
		ResultHandling* pResultLog;
};
```

ResultHandling.cpp

```
//////////////////////////////////////////////////////////////////////
// ResultHandling.cpp: implementation of the ResultHandling class.
//
//////////////////////////////////////////////////////////////////////

#include "ResultHandling.h"
#include "fstream.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

ResultHandling::ResultHandling(char* pFileName)
{
	// Create or open new file object for logging Results
	pResultFile = NULL;
	pResultFile = new ofstream;

	// Retrieve current time and date
	char* pTimeString = NULL;
	CTime thetime = CTime::GetCurrentTime();
	CString timestring;
	timestring = thetime.Format( "%d%b%Y %H:%M  " );

	// Open Result log file for writing by this process only, if not in existance create new file
	pResultFile->open(pFileName, ios::app | ios::in | ios::out, 0);//filebuf::sh_read );

	if(pResultFile->fail())
	{
		// File not opened, delete nearby objects and terminate immediately
		cout << timestring << "Error:- Unable to open or create Results File" << endl;
		delete pResultFile;
		pResultFile = NULL;
		exit(1);
	}
	else
	{
		// File opened successfully, carry on
		cout << "*********** " << timestring << "Results File Opened Successfully ***********" << endl;
		*pResultFile << "*********** " << timestring << "Results File Opened Successfully ***********" << endl;
	}
}

ResultHandling::~ResultHandling()
{
	//Close file etc
	pResultFile->flush();
	pResultFile->close();
	delete pResultFile;
	pResultFile = NULL;
}

ResultHandling::RecordResult(CString Resultdescription, double Result)
{
	// Retrieve current time and date
	char* pTimeString = NULL;
	CTime thetime = CTime::GetCurrentTime();
	CString timestring;
	timestring = thetime.Format( "%d%b%Y %H:%M %S  " );

	// Write Result to log file
	*pResultFile << timestring << "Result:- " << Resultdescription << ", "<< Result << endl;
}
```

ResultHandling.h

```
//////////////////////////////////////////////////////////////////////
// ResultHandling.h: interface for the ResultHandling class.
//
//////////////////////////////////////////////////////////////////////
#include "afx.h"
#include "fstream.h"

class ResultHandling
{
public:
	ResultHandling(char* pFileName);
	virtual ~ResultHandling();
	RecordResult(CString Resultdescription, double Result);
private:
	ofstream* pResultFile;
};
```

Main.cpp

```
#include "iostream.h"
#include "ErrorHandling.h"
#include "Scheduler.h"


void main()
{
	char text[200] = "";

	ErrorHandling* pErrorLog;
	pErrorLog = new ErrorHandling;				// Create error handling object ready to receive any errors

	scheduler* pScheduler;
	pScheduler = new scheduler(pErrorLog);			// Create scheduler object which will start test cycle

	while ((text != "exit") && (text != "EXIT") && (text != "quit") && (text != "QUIT"))
	{
		cin >> text;				// Halt main program thread until exit typed
	}

	delete pScheduler;				// Clean up scheduler object
	pScheduler = NULL;

	delete pErrorLog;				// Clean up error handling object
	pErrorLog = NULL;
}
```

Result Analysis Program Listing

As well as the following files, excel.cpp and excel.h were used, however these are generic files that are supplied as part of the Excel C++ type library, and so are not included here. The result analysis program is an MFC (Microsoft Foundation Class) program, consequently the compiler generates the basic framework of the program required for it to operate as a program in Windows, (i.e. the window and ok/cancel buttons)

FileRead.cpp

```
// FileRead.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "FileRead.h"
#include "FileReadDlg.h"
#include <afxdisp.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFileReadApp

BEGIN_MESSAGE_MAP(CFileReadApp, CWinApp)
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFileReadApp construction

CFileReadApp::CFileReadApp()
{
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CFileReadApp object

CFileReadApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CFileReadApp initialization

BOOL CFileReadApp::InitInstance()
{
  if(!AfxOleInit())
  {
    AfxMessageBox("Could not initialize COM dll");
    return FALSE;
  }
  AfxEnableControlContainer();


#ifdef _AFXDLL
        Enable3dControls();         // Call this when using MFC in a shared DLL
#else
        Enable3dControlsStatic();   // Call this when linking to MFC statically
#endif

        CFileReadDlg dlg;
        m_pMainWnd = &dlg;
        int nResponse = dlg.DoModal();
        if (nResponse == IDOK)
        {
                // dismissed with OK
        }
        else if (nResponse == IDCANCEL)
        {
                // dismissed with Cancel
        }

        // Since the dialog has been closed, return FALSE so that we exit the
        //  application, rather than start the application's message pump.
        return FALSE;
}
```

FileRead.h

```
// FileRead.h : main header file for the FILEREAD application
//

#if !defined(AFX_FILEREAD_H__073F182F_FAAB_43B5_A125_096348EB87C2__INCLUDED_)
#define AFX_FILEREAD_H__073F182F_FAAB_43B5_A125_096348EB87C2__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// main symbols

/////////////////////////////////////////////////////////////////////////////
// CFileReadApp:
// See FileRead.cpp for the implementation of this class
//

class CFileReadApp : public CWinApp
{
public:
	CFileReadApp();

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CFileReadApp)
	public:
	virtual BOOL InitInstance();
	//}}AFX_VIRTUAL

	DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_FILEREAD_H__073F182F_FAAB_43B5_A125_096348EB87C2__INCLUDED_)
```

FileReadDlg.cpp

```
// FileReadDlg.cpp : implementation file
//

#include "stdafx.h"
#include "FileRead.h"
#include "FileReadDlg.h"
#include "spreadsheet.h"
#include "fileio.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
		CAboutDlg();

		// Dialog Data
		//{{AFX_DATA(CAboutDlg)
		enum { IDD = IDD_ABOUTBOX };
		//}}AFX_DATA

		// ClassWizard generated virtual function overrides
		//{{AFX_VIRTUAL(CAboutDlg)
		protected:
		virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
		//}}AFX_VIRTUAL

protected:
		//{{AFX_MSG(CAboutDlg)
		//}}AFX_MSG
		DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
		//{{AFX_DATA_INIT(CAboutDlg)
		//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
		CDialog::DoDataExchange(pDX);
		//{{AFX_DATA_MAP(CAboutDlg)
		//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
		//{{AFX_MSG_MAP(CAboutDlg)
				// No message handlers
		//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFileReadDlg dialog

CFileReadDlg::CFileReadDlg(CWnd* pParent /*=NULL*/)
		: CDialog(CFileReadDlg::IDD, pParent)
{
		//{{AFX_DATA_INIT(CFileReadDlg)
		m_Filename = _T("");
		//}}AFX_DATA_INIT
		// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
		m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CFileReadDlg::DoDataExchange(CDataExchange* pDX)
{
		CDialog::DoDataExchange(pDX);
		//{{AFX_DATA_MAP(CFileReadDlg)
		DDX_Text(pDX, IDC_EDIT1, m_Filename);
		//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CFileReadDlg, CDialog)
		//{{AFX_MSG_MAP(CFileReadDlg)
		ON_WM_SYSCOMMAND()
		ON_WM_PAINT()
		ON_WM_QUERYDRAGICON()
		ON_BN_CLICKED(IDRUN, OnRun)
		//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFileReadDlg message handlers

BOOL CFileReadDlg::OnInitDialog()
{
```

```
	m_Filename = "Enter File Name, NO Directories (run program in same folder)";
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon


	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CFileReadDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}


void CFileReadDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

HCURSOR CFileReadDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CFileReadDlg::OnRun()
{
	UpdateData(TRUE);
	spreadsheet myspread;
	// Label first sheet
	myspread.celldata(1,"A1","Mins Burned");
	myspread.celldata(1,"B1","Voltage (V)");
	myspread.celldata(1,"C1","Current (A)");
	myspread.celldata(1,"D1","V/I (Ohms)");
	myspread.celldata(1,"E1","Power (W)");
	myspread.celldata(1,"F1","Lamp Temp (V)");
	myspread.celldata(1,"G1","Room Temp (V)");
	myspread.celldata(1,"H1","Light (V)");
	// Label second sheet
	myspread.celldata(2,"A1","Mins Burned");
	myspread.celldata(2,"B1","Power (W)");
	myspread.celldata(2,"C1","Predicted Power %");
	myspread.celldata(2,"D1","Predicted Power (W)");
	myspread.celldata(2,"E1","Predicted - Actual");
	myspread.celldata(2,"F1","Filtered");

	fileio myfile(m_Filename);
	CString line = "";
	CString num = "";
	CString cell = "";
	CString row = "";
```

```
CString first = "";
CString second = "";
CString formula = "";
CString source = "";
CString source1 = "";

int rowcount = 2;
int fail = 0;

while(myfile.Getnextline(&line))
{
        // Read next line from the file
        if((line.Find("Results File Opened Successfully **********")) != -1)
        {

        }

        if((line.Find("Result - ----- Lamp Turned On, Inrush Current,")) != -1)
        {
                /*
                cell = "I";
                row.Format("%d", rowcount);
                cell += row;
                num = line.Mid(67);
                myspread.celldata(cell,num);
                */ // Not used for filament lamps

                // Shade cells green at switch on for sheet 1
                row.Format("%d", rowcount);
                first = "A";
                first += row;
                second = "H";
                second += row;
                myspread.setcolour(1,first,second);

                // Shade cells green at switch on for sheet 2
                row.Format("%d", rowcount);
                first = "A";
                first += row;
                second = "F";
                second += row;
                myspread.setcolour(2,first,second);
        }

        if((line.Find("Result:- Measurement Cycle Start, Minutes Burned:-")) != -1)
        {
                cell = "A";
                row.Format("%d", rowcount);             // Convert integer to string
                cell += row;
                num = line.Mid(73);
                myspread.celldata(1,cell,num);          // Store minutes burned on sheet 1
                myspread.celldata(2,cell,num);          // Store minutes burned on sheet 2
                cell = "D";
                cell += row;
                num = "=B";
                num += row;
                num += "/C";
                num += row;
                myspread.celldata(1,cell, num);         // Load formulae for V/I on sheet 1


        }

        if((line.Find("Result:- Lamp Temp Sens Voltage;")) != -1)
        {
                cell = "F";
                row.Format("%d", rowcount);
                cell += row;
                num = line.Mid(53);
                myspread.celldata(1,cell,num);
        }

        if((line.Find("Result:- Lamp Light Sens Voltage,")) != -1)
        {
                cell = "H";
                row.Format("%d", rowcount);
                cell += row;
                num = line.Mid(54);
                myspread.celldata(1,cell,num);
        }

        if((line.Find("Result:- Room Temp Sens Voltage;")) != -1)
        {
                cell = "G";
                row.Format("%d", rowcount);
                cell += row;
                num = line.Mid(53);
                myspread.celldata(1,cell,num);
        }

        if((line.Find("Result:- Volts AC,")) != -1)
        {
                cell = "B";
                row.Format("%d", rowcount);
                cell += row;
                num = line.Mid(39);
                myspread.celldata(1,cell,num);
```

```
}

if(line.Find("Result:- Amps AC,")) != -1)
{
		cell = "C";
		row.Format("%d", rowcount);
		cell += row;
		num = line.Mid(38);
		myspread.celldata(1,cell,num);
}

if(line.Find("Result:- Watts,")) != -1)
{
		cell = "E";
		row.Format("%d", rowcount);
		cell += row;
		num = line.Mid(36);
		myspread.celldata(1,cell,num);
		double power = strtod(num, NULL);
		if(power >= 10.0)
		{
			fail = rowcount;
		}
		cell = "B";
		row.Format("%d", rowcount);
		cell += row;
		num = line.Mid(36);
		myspread.celldata(2,cell,num);
}

if(line.Find("Result:- Power Factor,")) != -1)
{
		// Not used for filament lamps
		/*cell = "H";
		row.Format("%d", rowcount);
		cell += row;
		num = line.Mid(43);
		myspread.celldata(cell,num);*/

		// Fill in equation for predicted % power on sheet 2
		cell = "C";
		row.Format("%d", rowcount);
		cell += row;

		source = "(Sheet1!B";
		source += row;
		source += ")";

		formula = "=(-1.47203332464055E-06*(";
		formula += source;
		formula += "^3)) + (0.001595739518379 66*(";
		formula += source;
		formula += "^2)) + (0.120660948856852*";
		formula += source;
		formula += ") - 0.449534812519687";

		myspread.celldata(2,cell,formula);


		// Fill in equation for predicted power in watts on sheet2
		cell = "D";
		row.Format("%d", rowcount);
		cell += row;

		formula = "=((";
		source = "C";
		source += row;
		formula += source;
		formula += "/C2)*B2)";
		myspread.celldata(2,cell,formula);

		// Fill in equation for predicted power - actual power
		cell = "E";
		row.Format("%d", rowcount);
		cell += row;

		formula = "=(";
		source = "D";
		source += row;
		formula += source;
		formula += "-";
		source = "B";
		source += row;
		formula += source;
		formula += ")";
		myspread.celldata(2,cell,formula);

		source1 = "F";
		row.Format("%d", (rowcount-1));
		source1 += row;

		cell = "F";
		source = "E";
		row.Format("%d", rowcount);
		source += row;
		cell += row;
```

```
                        formula = "=IF((ABS(",
                        formula += source,
                        formula += "-",
                        formula += source1,
                        formula += ")>0.6), ",
                        formula += source1,
                        formula += ", ",
                        formula += source,
                        formula += ")",
                        myspread celldata(2,cell,formula),
                }

                if(line Find("Result - ----- End of Measurement Cycle -----,")) != -1)
                {
                        rowcount ++,
                }

                if(line Find("Result - ----- Lamp Turned Off -----,")) != -1)
                {

                }

}

row Format("%od", fail),
cell = "F",
cell += row,
formula = "=INDEX(LINEST(F2:",
formula += cell,
formula += ",A2:",
cell = "A",
cell += row,
formula += cell,
formula += ",TRUE,FALSE),1)",
myspread celldata(2, "H2", formula),

row Format("%od", fail),
cell = "F",
cell += row,
formula = "=INDEX(LINEST(F2:",
formula += cell,
formula += ",A2:",
cell = "A",
cell += row,
formula += cell,
formula += ",TRUE,FALSE),2)",
myspread celldata(2, "I2", formula),

formula = "=abs(((H2*A2)+I2)-((H2*",
formula += cell,
formula += ")+I2))",
myspread celldata(2, "J2", formula),

myspread celldata(2, "H1", "Gradient"),
myspread celldata(2, "I1", "Intercept"),
myspread celldata(2, "J1", "Difference"),


formula Empty(),
cell Empty(),
num Empty(),
line Empty(),
row Empty(),
first Empty(),
second Empty(),

myspread celldata(2,"F2", "0"),
myspread settableborders(1,rowcount-1),
myspread settableborders(2,rowcount-1),
myspread graph(fail, m_Filename),
myspread save(),
exit(0),
}
```

FileReadDlg.h

```
// FileReadDlg.h : header file
//

#if !defined(AFX_FILEREADDLG_H__51F8BC86_0BF3_46B8_9A20_B772C4AE528D__INCLUDED_)
#define AFX_FILEREADDLG_H__51F8BC86_0BF3_46B8_9A20_B772C4AE528D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CFileReadDlg dialog

class CFileReadDlg : public CDialog
{
// Construction
public:
	CFileReadDlg(CWnd* pParent = NULL);	// standard constructor

// Dialog Data
	//{{AFX_DATA(CFileReadDlg)
	enum { IDD = IDD_FILEREAD_DIALOG };
	CString	m_Filename;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CFileReadDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CFileReadDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnRun();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_FILEREADDLG_H__51F8BC86_0BF3_46B8_9A20_B772C4AE528D__INCLUDED_)
```

FileIO.cpp

```
// Read in data from textfile
#include "fileio.h"

fileio::fileio(CString Filename)
{
	pLampFile = new ifstream;
	pLampFile->open(Filename, ios::in | ios::nocreate, filebuf::sh_read);
	if(pLampFile->fail())
	{
		AfxMessageBox("File could not be opened");
		exit(1);
	}

}

fileio::~fileio()
{
	pLampFile->close();
	delete pLampFile;
	pLampFile = NULL;
}

bool fileio::Getnextline(CString* line)
{
	char temp[100];
	pLampFile->getline(temp, 100);
	*line = temp;
	if (pLampFile->eof() != 0)
	{
		return false;
	}
	else
	{
		return true;
	}
}
```

FileIO.h

```
#include "stdafx.h"
#include "fstream.h"

class fileio
{
public:
	fileio(CString Filename);
	virtual ~fileio();
	bool Getnextline(CString* line);

private:
	ifstream* pLampFile;


};
```

Spreadsheet.cpp

```
// spreadsheet.cpp : implementation file
//

#include "stdafx.h"
#include "FileRead.h"
#include "spreadsheet.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////////////////////////////////////
// spreadsheet

spreadsheet::spreadsheet()
{
	try
	{
		LPDISPATCH lpDisp;			// idispatch pointer

		COleVariant
					covTrue((short)TRUE),
					covFalse((short)FALSE),
					covOptional((long)DISP_E_PARAMNOTFOUND, VT_ERROR);


		// Start Excel and get Application object.
		if(!app.CreateDispatch("Excel.Application"))
		{
			AfxMessageBox("Couldn't start Excel.");
			return;
		}

		//Make Excel Visible
		app.SetVisible(TRUE);
		// Get IDispatch pointer and attach to the books object
		lpDisp = app.GetWorkbooks();
		ASSERT(lpDisp);
		books.AttachDispatch(lpDisp);
		// Create new workbook
		lpDisp = books.Add(covOptional);
		ASSERT(lpDisp);
		book.AttachDispatch(lpDisp);

		// Get sheets
		lpDisp = book.GetSheets();
		ASSERT(lpDisp);
		sheets.AttachDispatch(lpDisp);

		// Get sheet
		lpDisp = sheets.GetItem(COleVariant((short)(1)));
		ASSERT(lpDisp);
		sheet.AttachDispatch(lpDisp);
		// Get second sheet
		lpDisp = sheets.GetItem(COleVariant((short)(2)));
		ASSERT(lpDisp);
		sheet2.AttachDispatch(lpDisp);
		// Set column width for A1-I1 on sheet 1
		lpDisp = sheet.GetRange(COleVariant("A1"), COleVariant("I1"));
		ASSERT(lpDisp);
		range.AttachDispatch(lpDisp);
		range.SetColumnWidth(COleVariant("13"));
		// Set column width for A1-F1 on sheet 2
		lpDisp = sheet2.GetRange(COleVariant("A1"), COleVariant("F1"));
		ASSERT(lpDisp);
		range.AttachDispatch(lpDisp);
		range.SetColumnWidth(COleVariant("18"));
	}

	catch(COleException *e)
	{
		char buf[1024];

		sprintf(buf, "COleException. SCODE: %08lx.", (long)e->m_sc);
		::MessageBox(NULL, buf, "COleException", MB_SETFOREGROUND | MB_OK);
	}

	catch(COleDispatchException *e)
	{
		char buf[1024];
		sprintf(buf,
			"COleDispatchException. SCODE: %08lx, Description: \"%s\".",
			(long)e->m_wCode,
			(LPSTR)e->m_strDescription.GetBuffer(1024));
			::MessageBox(NULL, buf, "COleDispatchException",
			MB_SETFOREGROUND | MB_OK);
	}

	catch(...)
```

```
	{
		::MessageBox(NULL, "General Exception caught.", "Catch-All", MB_SETFOREGROUND | MB_OK);
	}
}



spreadsheet::~spreadsheet()
{
	app.ReleaseDispatch();
	book.ReleaseDispatch();
	books.ReleaseDispatch();
	borders.ReleaseDispatch();
	chart.ReleaseDispatch();
	chartobject.ReleaseDispatch();
	chartobjects.ReleaseDispatch();
	interior.ReleaseDispatch();
	range.ReleaseDispatch();
	series.ReleaseDispatch();
	sheet.ReleaseDispatch();
	sheet2.ReleaseDispatch();
	sheets.ReleaseDispatch();
	trendline.ReleaseDispatch();
	trendlines.ReleaseDispatch();

}


////////////////////////////////////////////////////////////////////////////
// spreadsheet message handlers

void spreadsheet::celldata(int sheetnum, CString cellname, CString data)
{
	try
	{
		LPDISPATCH lpDisp;	// idispatch pointer
		// get cell
		if(sheetnum==1)
		{
			lpDisp = sheet.GetRange(COleVariant(cellname), COleVariant(cellname));
			ASSERT(lpDisp);
			range.AttachDispatch(lpDisp);
			range.SetValue2(COleVariant(data));
		}
		if(sheetnum==2)
		{
			lpDisp = sheet2.GetRange(COleVariant(cellname), COleVariant(cellname));
			ASSERT(lpDisp);
			range.AttachDispatch(lpDisp);
			range.SetValue2(COleVariant(data));
		}

	}

	catch(COleException *e)
	{
		char buf[1024];

		sprintf(buf, "COleException. SCODE: %08lx.", (long)e->m_sc);
		::MessageBox(NULL, buf, "COleException", MB_SETFOREGROUND | MB_OK);
	}

	catch(COleDispatchException *e)
	{
		char buf[1024];
		sprintf(buf,
			"COleDispatchException. SCODE: %08lx, Description: \"%s\".",
			(long)e->m_wCode,
			(LPSTR)e->m_strDescription.GetBuffer(1024));
			::MessageBox(NULL, buf, "COleDispatchException",
			MB_SETFOREGROUND | MB_OK);
	}

	catch(...)
	{
		::MessageBox(NULL, "General Exception caught.", "Catch-All", MB_SETFOREGROUND | MB_OK);
	}
}

void spreadsheet::setcolour(int sheetnum, CString first, CString second)
{
	try
	{
		LPDISPATCH lpDisp;	// idispatch pointer

		COleVariant
					covTrue((short)TRUE),
					covFalse((short)FALSE),
					covOptional((long)DISP_E_PARAMNOTFOUND, VT_ERROR);

		if(sheetnum == 1)
		{
			// Select cells
			lpDisp = sheet.GetRange(COleVariant(first), COleVariant(second));
			ASSERT(lpDisp);
```

```
                range.AttachDispatch(lpDisp);
                // Colour cells green
                lpDisp = range.GetInterior();
                ASSERT(lpDisp);
                interior.AttachDispatch(lpDisp);
                interior.SetColor(COleVariant("255,255"));
        }
        if(sheetnum == 2)
        {
                // Select cells
                lpDisp = sheet2.GetRange(COleVariant(first), COleVariant(second));
                ASSERT(lpDisp);
                range.AttachDispatch(lpDisp);
                // Colour cells green
                lpDisp = range.GetInterior();
                ASSERT(lpDisp);
                interior.AttachDispatch(lpDisp);
                interior.SetColor(COleVariant("255,255"));
        }

    }

    catch(COleException *e)
    {
        char buf[1024];

        sprintf(buf, "COleException. SCODE: %08lx.", (long)e->m_sc);
        ::MessageBox(NULL, buf, "COleException", MB_SETFOREGROUND | MB_OK);
    }

    catch(COleDispatchException *e)
    {
        char buf[1024];
        sprintf(buf,
            "COleDispatchException. SCODE: %08lx, Description: \"%s\".",
            (long)e->m_wCode,
            (LPSTR)e->m_strDescription.GetBuffer(1024));
            ::MessageBox(NULL, buf, "COleDispatchException",
            MB_SETFOREGROUND | MB_OK);
    }

    catch(...)
    {
        ::MessageBox(NULL, "General Exception caught.", "Catch-All", MB_SETFOREGROUND | MB_OK);
    }
}

void spreadsheet::settableborders(int sheetnum, int noofrows)
{
    try
    {
        LPDISPATCH lpDisp;      // idispatch pointer


        if(sheetnum == 1)
        {
            CString bottomright = "H";
            CString row;
            row.Format("%d", noofrows);
            CString bottomleft = "A";
            bottomleft += row;
            bottomright += row;
            // Select cells
            lpDisp = sheet.GetRange(COleVariant("A1"), COleVariant(bottomright));
            ASSERT(lpDisp);
            range.AttachDispatch(lpDisp);
            // Put on thin border around all cells
            borders=range.GetBorders();
            borders.SetWeight(COleVariant((short)2));      //xlThin = 2
            // Put medium line around mins column
            lpDisp = sheet.GetRange(COleVariant("A1"), COleVariant(bottomleft));
            ASSERT(lpDisp);
            range.AttachDispatch(lpDisp);
            range.BorderAround(COleVariant((short)1), -4138, 0, COleVariant((short)0)); // Solid line = 1, -4138 = Thick
            // Put medium line around heading
            lpDisp = sheet.GetRange(COleVariant("A1"), COleVariant("H1"));
            ASSERT(lpDisp);
            range.AttachDispatch(lpDisp);
            range.BorderAround(COleVariant((short)1), -4138, 0, COleVariant((short)0)); // Solid line = 1, -4138 = Medium
            // Put double line border around entire table
            lpDisp = sheet.GetRange(COleVariant("A1"), COleVariant(bottomright));
            ASSERT(lpDisp);
            range.AttachDispatch(lpDisp);
            range.BorderAround(COleVariant((short)1), 4, 0, COleVariant((short)0)); // Solid line =1, 4 = Thick
        }
        if(sheetnum == 2)
        {
            CString bottomright2 = "F";
            CString row2;
            row2.Format("%d", noofrows);
            CString bottomleft2 = "A";
            bottomleft2 += row2;
            bottomright2 += row2;
            // Select cells
            lpDisp = sheet2.GetRange(COleVariant("A1"), COleVariant(bottomright2));
            ASSERT(lpDisp);
            range.AttachDispatch(lpDisp);
```

```
                    // Put on thin border around all cells
                    borders=range.GetBorders();
                    borders.SetWeight(COleVariant((short)2));      //xlThin = 2
                    // Put medium line around mins column
                    lpDisp = sheet2.GetRange(COleVariant("A1"), COleVariant(bottomleft2));
                    ASSERT(lpDisp);
                    range.AttachDispatch(lpDisp);
                    range.BorderAround(COleVariant((short)1), -4138, 0, COleVariant((short)0)); // Solid line = 1, -4138 = Thick
                    // Put medium line around heading
                    lpDisp = sheet2.GetRange(COleVariant("A1"), COleVariant("F1"));
                    ASSERT(lpDisp);
                    range.AttachDispatch(lpDisp);
                    range.BorderAround(COleVariant((short)1), -4138, 0, COleVariant((short)0)); // Solid line = 1, -4138 = Medium
                    // Put double line border around entire table
                    lpDisp = sheet2.GetRange(COleVariant("A1"), COleVariant(bottomright2));
                    ASSERT(lpDisp);
                    range.AttachDispatch(lpDisp);
                    range.BorderAround(COleVariant((short)1), 4, 0, COleVariant((short)0)); // Solid line =1, 4 = Thick
                }
        }

        catch(COleException *e)
        {
                char buf[1024];

                sprintf(buf, "COleException. SCODE: %08lx.", (long)e->m_sc);
                ::MessageBox(NULL, buf, "COleException", MB_SETFOREGROUND | MB_OK);
        }

        catch(COleDispatchException *e)
        {
                char buf[1024];
                sprintf(buf,
                        "COleDispatchException. SCODE: %08lx, Description: \"%s\".",
                        (long)e->m_wCode,
                        (LPSTR)e->m_strDescription.GetBuffer(1024));
                        ::MessageBox(NULL, buf, "COleDispatchException",
                        MB_SETFOREGROUND | MB_OK);
        }

        catch(...)
        {
                ::MessageBox(NULL, "General Exception caught.", "Catch-All", MB_SETFOREGROUND | MB_OK);
        }

}

void spreadsheet::graph(int noofrows, CString lampnum)
{
        try
        {
                LPDISPATCH lpDisp;          // idispatch pointer
                VARIANT var;
                var.vt = VT_DISPATCH;
                COleVariant covTrue((short)TRUE), covFalse((short)FALSE),
                                                        covOptional((long)DISP_E_PARAMNOTFOUND, VT_ERROR);

                lpDisp=sheet.ChartObjects(COleVariant(covOptional));
                ASSERT(lpDisp);
                chartobjects.AttachDispatch(lpDisp);
                CString row;
                row.Format("%d", noofrows);
                CString column;
                CString cola = "A";
                cola += row;
                CString title;
                int position = lampnum.Find(".txt",0);
                lampnum.Delete(position,4);

//*************************************************************************************************
// Plot Voltage vs Time

                // Generate name of last cell to be plotted
                column = "B";
                column += row;

                // Generate Title from filename
                title = lampnum;
                title += ", Voltage vs Time Burned";

                // Find initial site for chart
                chartobject = chartobjects.Add(0,0,500,500);
                chart.AttachDispatch(chartobject.GetChart());

                // Get y-axis range and store as range object
                lpDisp = sheet.GetRange(COleVariant("B2"), COleVariant(column));
                range.AttachDispatch(lpDisp);
                // Package range object as a Variant
                var.pdispVal = lpDisp;
                // Set chart data values to this range of values
                series.SetValues(COleVariant(var));

                // Generate and label chart
                chart.ChartWizard(var,                                  // Source.
                COleVariant((short)-4169),                              // Gallery: xyscatter.
                covOptional,                                            // Format, use default.
```

```
COleVariant((short)2),                          // PlotBy: xlColumns.
COleVariant((short)FALSE),                      // CategoryLabels.
COleVariant((short)FALSE),                      // SeriesLabels.
COleVariant((short)FALSE),                      // HasLegend.
COleVariant(title),                             // Title.
COleVariant("Time Burned (mins)"),              // CategoryTitle.
COleVariant("Voltage (V)"),                     // ValueTitles.
covOptional                                     // ExtraTitle.
);
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));

// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Voltage"));

//*****************************************************************************************
// Plot Current Vs Time

// Generate name of last cell to be plotted
column = "C";
column += row;

// Generate Title from filename
title = lampnum;
title += ", Current vs Time Burned",

// Find initial site for chart
chartobject = chartobjects.Add(0,0,500,500);
chart.AttachDispatch(chartobject.GetChart());

// Get y-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("C2"), COleVariant(column));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart data values to this range of values
series.SetValues(COleVariant(var));

// Generate and label chart
chart.ChartWizard(var,                          // Source.
COleVariant((short)-4169),                      // Gallery: xyscatter.
covOptional,                                    // Format, use default.
COleVariant((short)2),                          // PlotBy: xlColumns.
COleVariant((short)FALSE),                      // CategoryLabels.
COleVariant((short)FALSE),                      // SeriesLabels.
COleVariant((short)FALSE),                      // HasLegend.
COleVariant(title),                             // Title.
COleVariant("Time Burned (mins)"),              // CategoryTitle.
COleVariant("Current (A)"),                     // ValueTitles.
covOptional                                     // ExtraTitle.
);
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));

// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Current"));

//*****************************************************************************************
// Plot Resistance Vs Time

// Generate name of last cell to be plotted
column = "D";
column += row;

// Generate Title from filename
title = lampnum;
title += ", Resistance vs Time Burned",

// Find initial site for chart
chartobject = chartobjects.Add(0,0,500,500);
chart.AttachDispatch(chartobject.GetChart());

// Get y-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("D2"), COleVariant(column));
range.AttachDispatch(lpDisp);
```

```
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart data values to this range of values
series.SetValues(COleVariant(var));

// Generate and label chart
chart.ChartWizard(var,                                  // Source.
COleVariant((short)-4169),                              // Gallery: xyscatter.
covOptional,                                            // Format, use default.
COleVariant((short)2),                                  // PlotBy: xlColumns.
COleVariant((short)FALSE),                              // CategoryLabels.
COleVariant((short)FALSE),                              // SeriesLabels.
COleVariant((short)FALSE),                              // HasLegend.
COleVariant(title),                                     // Title.
COleVariant("Time Burned (mins)"),                      // CategoryTitle.
COleVariant("Resistance (Ohms)"),                       // ValueTitles.
covOptional                                             // ExtraTitle.
);
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));

// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Resistance"));

//*****************************************************************************************
// Plot Power Vs Time

// Generate name of last cell to be plotted
column = "E";
column += row;

// Generate Title from filename
title = lampnum;
title += ", Power vs Time Burned";

// Find initial site for chart
chartobject = chartobjects.Add(0,0,500,500);
chart.AttachDispatch(chartobject.GetChart());

// Get y-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("E2"), COleVariant(column));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart data values to this range of values
series.SetValues(COleVariant(var));

// Generate and label chart
chart.ChartWizard(var,                                  // Source.
COleVariant((short)-4169),                              // Gallery: xyscatter.
covOptional,                                            // Format, use default.
COleVariant((short)2),                                  // PlotBy: xlColumns.
COleVariant((short)FALSE),                              // CategoryLabels.
COleVariant((short)FALSE),                              // SeriesLabels.
COleVariant((short)FALSE),                              // HasLegend.
COleVariant(title),                                     // Title.
COleVariant("Time Burned (mins)"),                      // CategoryTitle.
COleVariant("Power (W)"),                               // ValueTitles.
covOptional                                             // ExtraTitle.
);
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));
// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Power"));

//*****************************************************************************************
// Plot Lamp Temp Vs Time

// Generate name of last cell to be plotted
column = "F";
column += row;

// Generate Title from filename
title = lampnum;
title += ", Lamp Temperature vs Time Burned";
```

```
// Find initial site for chart
chartobject = chartobjects.Add(0,0,500,500);
chart.AttachDispatch(chartobject.GetChart());

// Get y-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("F2"), COleVariant(column));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart data values to this range of values
series.SetValues(COleVariant(var));

// Generate and label chart
chart.ChartWizard(var,                                  // Source.
COleVariant((short)-4169),                              // Gallery: xyscatter.
covOptional,                                            // Format, use default.
COleVariant((short)2),                                  // PlotBy: xlColumns.
COleVariant((short)FALSE),                              // CategoryLabels.
COleVariant((short)FALSE),                              // SeriesLabels.
COleVariant((short)FALSE),                              // HasLegend.
COleVariant(title),                                     // Title.
COleVariant("Time Burned (mins)"),                      // CategoryTitle.
COleVariant("Lamp Temperature Sensor Voltage (V)"),     // ValueTitles.
covOptional                                             // ExtraTitle.
 );
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));

// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Lamp Temp"));

//*****************************************************************************************
// Plot Room Temp Vs Time

// Generate name of last cell to be plotted
column = "G";
column += row;

// Generate Title from filename
title = lampnum;
title += ", Room Temperature vs Time Burned";

// Find initial site for chart
chartobject = chartobjects.Add(0,0,500,500);
chart.AttachDispatch(chartobject.GetChart());

// Get y-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("G2"), COleVariant(column));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart data values to this range of values
series.SetValues(COleVariant(var));

// Generate and label chart
chart.ChartWizard(var,                                  //Source.
COleVariant((short)-4169),                              // Gallery: xyscatter.
covOptional,                                            // Format, use default.
COleVariant((short)2),                                  // PlotBy: xlColumns.
COleVariant((short)FALSE),                              // CategoryLabels.
COleVariant((short)FALSE),                              // SeriesLabels.
COleVariant((short)FALSE),                              // HasLegend.
COleVariant(title),                                     // Title.
COleVariant("Time Burned (mins)"),                      // CategoryTitle.
COleVariant("Room Temp Sensor Voltage (V)"),            // ValueTitles.
covOptional                                             // ExtraTitle.
);
// The return is void

// Get hold of the first and only plot series and turn into object
lpDisp = chart.SeriesCollection(COleVariant((short)1));
series.AttachDispatch(lpDisp);

// Get x-axis range and store as range object
lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
range.AttachDispatch(lpDisp);
// Package range object as a Variant
var.pdispVal = lpDisp;
// Set chart x axis to this range of values
series.SetXValues(COleVariant(var));

// Display chart as a separate sheet in the workbook
chart.Location((long(1)),COleVariant("Room Temp"));

//*****************************************************************************************
```

```
// Plot Lamp Light Vs Time

                // Generate name of last cell to be plotted
                column = "H";
                column += row;

                // Generate Title from filename
                title = lampnum;
                title += ", Light Output vs Time Burned";

                // Find initial site for chart
                chartobject = chartobjects.Add(0,0,500,500);
                chart.AttachDispatch(chartobject.GetChart());

                // Get y-axis range and store as range object
                lpDisp = sheet.GetRange(COleVariant("H2"), COleVariant(column));
                range.AttachDispatch(lpDisp);
                // Package range object as a Variant
                var.pdispVal = lpDisp;
                // Set chart data values to this range of values
                series.SetValues(COleVariant(var));

                // Generate and label chart
                chart.ChartWizard(var,                                  //Source.
                COleVariant((short)-4169),                              // Gallery: xyscatter.
                covOptional,                                            // Format, use default.
                COleVariant((short)2),                                  // PlotBy: xlColumns.
                COleVariant((short)FALSE),                              // CategoryLabels.
                COleVariant((short)FALSE),                              // SeriesLabels.
                COleVariant((short)FALSE),                              // HasLegend.
                COleVariant(title),                                     // Title.
                COleVariant("Time Burned (mins)"),                      // CategoryTitle.
                COleVariant("Lamp Light Sensor Voltage (V)"),           // ValueTitles.
                covOptional                                             // ExtraTitle.
                );
                // The return is void

                // Get hold of the first and only plot series and turn into object
                lpDisp = chart.SeriesCollection(COleVariant((short)1));
                series.AttachDispatch(lpDisp);

                // Get x-axis range and store as range object
                lpDisp = sheet.GetRange(COleVariant("A2"), COleVariant(cola));
                range.AttachDispatch(lpDisp);
                // Package range object as a Variant
                var.pdispVal = lpDisp;
                // Set chart x axis to this range of values
                series.SetXValues(COleVariant(var));

                // Display chart as a separate sheet in the workbook
                chart.Location((long(1)),COleVariant("Lamp Light"));


                //*************************************************************************************************
                // Plot Power Error Signal vs time (mins)
                // Generate name of last cell to be plotted
                column = "F";
                column += row;

                // Generate Title from filename
                title = lampnum;
                title += ", Power Error Signal vs Time Burned";

                // Find initial site for chart
                chartobject = chartobjects.Add(0,0,500,500);
                chart.AttachDispatch(chartobject.GetChart());

                // Get y-axis range and store as range object
                lpDisp = sheet2.GetRange(COleVariant("F2"), COleVariant(column));
                range.AttachDispatch(lpDisp);
                // Package range object as a Variant
                var.pdispVal = lpDisp;
                // Set chart data values to this range of values
                series.SetValues(COleVariant(var));

                // Generate and label chart
                chart.ChartWizard(var,                                  //Source.
                COleVariant((short)-4169),                              // Gallery: xyscatter.
                covOptional,                                            // Format, use default.
                COleVariant((short)2),                                  // PlotBy: xlColumns.
                COleVariant((short)FALSE),                              // CategoryLabels.
                COleVariant((short)FALSE),                              // SeriesLabels.
                COleVariant((short)FALSE),                              // HasLegend.
                COleVariant(title),                                     // Title.
                COleVariant("Time Burned (mins)"),                      // CategoryTitle.
                COleVariant("Power Error Signal (W)"),                  // ValueTitles.
                covOptional                                             // ExtraTitle.
                );
                // The return is void

                // Get hold of the first and only plot series and turn into object
                lpDisp = chart.SeriesCollection(COleVariant((short)1));
                series.AttachDispatch(lpDisp),

                // Get x-axis range and store as range object
                lpDisp = sheet2.GetRange(COleVariant("A2"), COleVariant(cola));
```

```
                range.AttachDispatch(lpDisp);
                // Package range object as a Variant
                var.pdispVal = lpDisp;
                // Set chart x axis to this range of values
                series.SetXValues(COleVariant(var));

                // Display chart as a separate sheet in the workbook
                chart.Location((long(1)),COleVariant("Power Error"));

                lpDisp = series.Trendlines(covOptional);    //xlLinear -4132
                ASSERT(lpDisp);
                trendlines.AttachDispatch(lpDisp);
                lpDisp = trendlines.Add(
                                        ((long)-4132),          // Line type = xlLinear -4132
                                        covOptional,            // Order
                                        covOptional,            // Period
                                        covOptional,            // Forward Prediction units
                                        covOptional,            // Backward Prediction units
                                        covOptional,            // Intercept
                                        covTrue,                // Display equation
                                        covTrue,                // Display R Squared
                                        covOptional);           // Name
                ASSERT(lpDisp);
                trendline.AttachDispatch(lpDisp);
                trendline.SetNameIsAuto(true);
                trendline.SetInterceptIsAuto(true);
                trendline.SetType((long)-4132);
                trendline.SetForward((long)0);
                trendline.SetBackward((long)0);
                // Location MUST be set last, changes after this do not take effect
                chart.Location((long(1)),COleVariant("Power Error"));
                double intercept = trendline.GetIntercept();
        }

                catch(COleException *e)
        {
                char buf[1024];

                sprintf(buf, "COleException. SCODE: %08lx.", (long)e->m_sc);
                ::MessageBox(NULL, buf, "COleException", MB_SETFOREGROUND | MB_OK);
        }

        catch(COleDispatchException *e)
        {
                char buf[1024];
                sprintf(buf,
                        "COleDispatchException. SCODE: %08lx, Description: \"%s\".",
                        (long)e->m_wCode,
                        (LPSTR)e->m_strDescription.GetBuffer(1024));
                        ::MessageBox(NULL, buf, "COleDispatchException",
                        MB_SETFOREGROUND | MB_OK);
        }

        catch(...)
        {
                ::MessageBox(NULL, "General Exception caught.", "Catch-All", MB_SETFOREGROUND | MB_OK);
        }
}


void spreadsheet::save()
{
        try
        {
                app.SaveWorkspace(COleVariant(""));

        }
        catch(...)
        {

        }
}
```

Spreadsheet.h

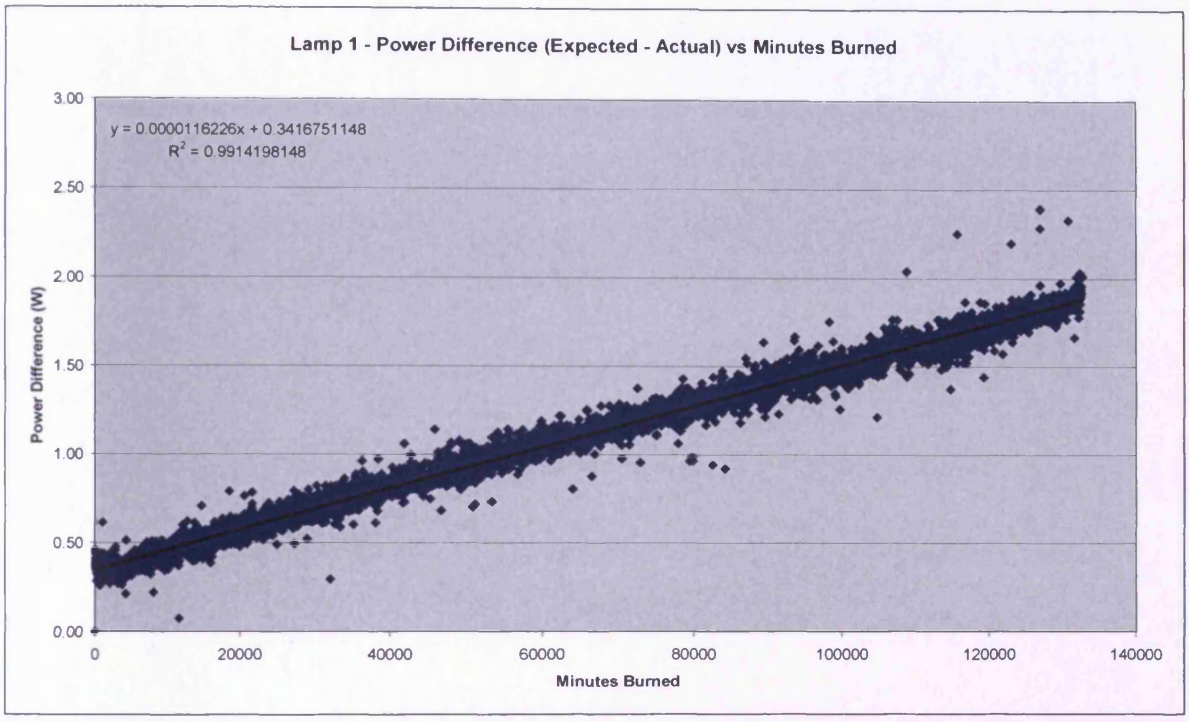
```
#include "excel.h"

#if !defined(AFX_SPREADSHEET_H__26F4F7BD_9BD6_4EE9_A32F_71D20EDADA19__INCLUDED_)
#define AFX_SPREADSHEET_H__26F4F7BD_9BD6_4EE9_A32F_71D20EDADA19__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// spreadsheet.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// spreadsheet window

class spreadsheet
{
// Construction
public:
		spreadsheet();

// Attributes
public:

// Operations
public:

private:
		_Application app;
		_Workbook book;
		_Worksheet sheet;
		_Worksheet sheet2;
		_Chart chart;
		Trendline trendline;
		Trendlines trendlines;
		Workbooks books;
		Worksheets sheets;
		Range range;
		Interior interior;
		Borders borders;
		ChartObjects chartobjects;
		ChartObject chartobject;
		Series series;
// Overrides
		// ClassWizard generated virtual function overrides
		//{{AFX_VIRTUAL(spreadsheet)
		//}}AFX_VIRTUAL

// Implementation
public:
		void celldata(int sheetnum, CString cellname, CString data);
		void setcolour(int sheetnum, CString first, CString second);
		void settableborders(int sheetnum, int noofrows);
		void graph(int noofrows, CString lampnum);
		void save();
		virtual ~spreadsheet();

		// Generated message map functions
protected:
		//{{AFX_MSG(spreadsheet)
			// NOTE - the ClassWizard will add and remove member functions here.
		//}}AFX_MSG
		//DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_SPREADSHEET_H__26F4F7BD_9BD6_4EE9_A32F_71D20EDADA19__INCLUDED_)
```

C

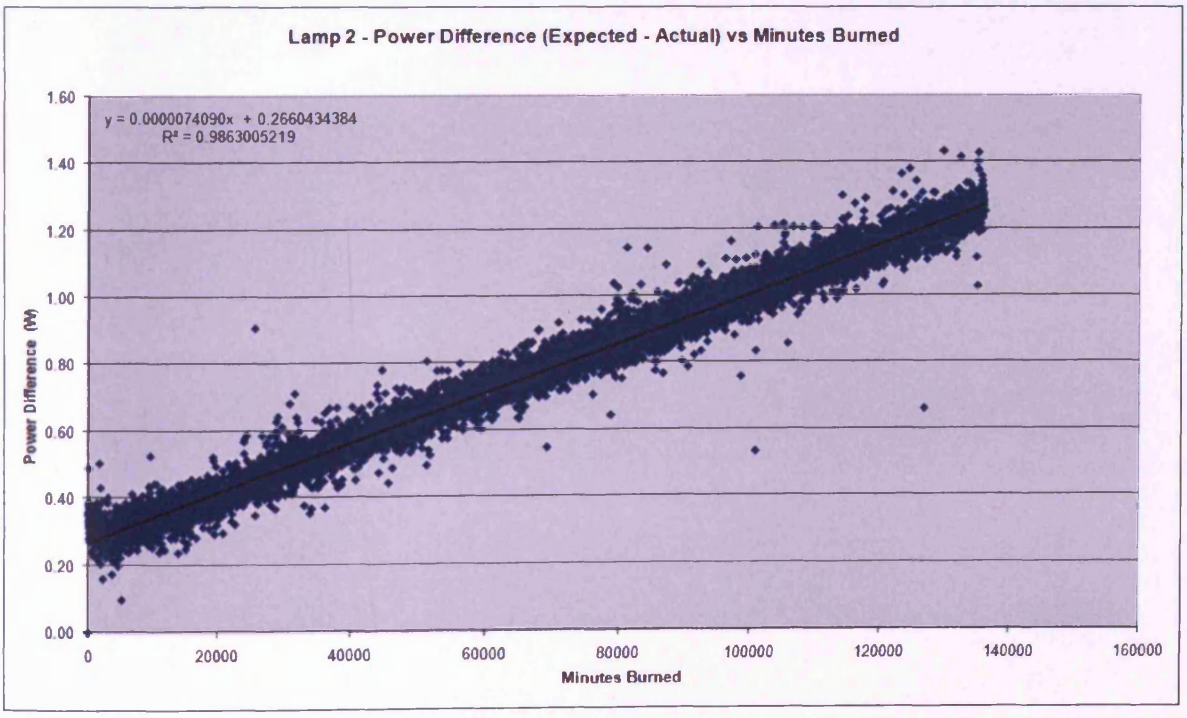
Appendix C - Filament Lamp Results

Continuous Monitoring Results

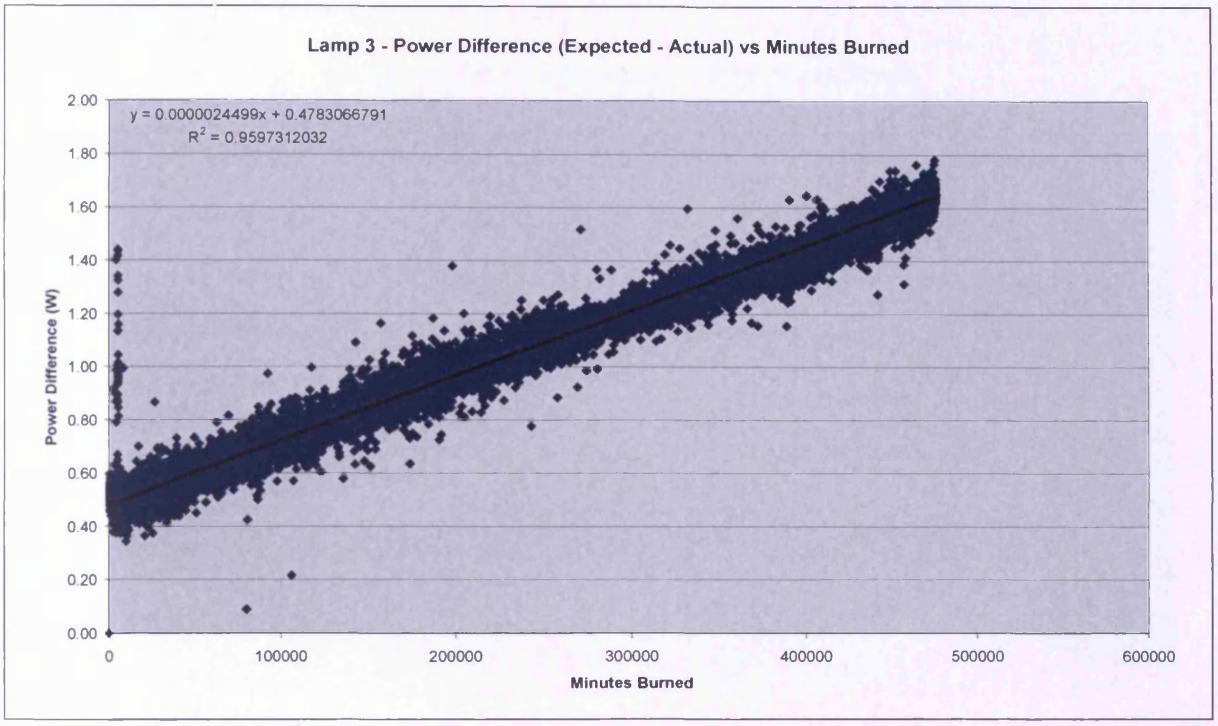
Lamp 1



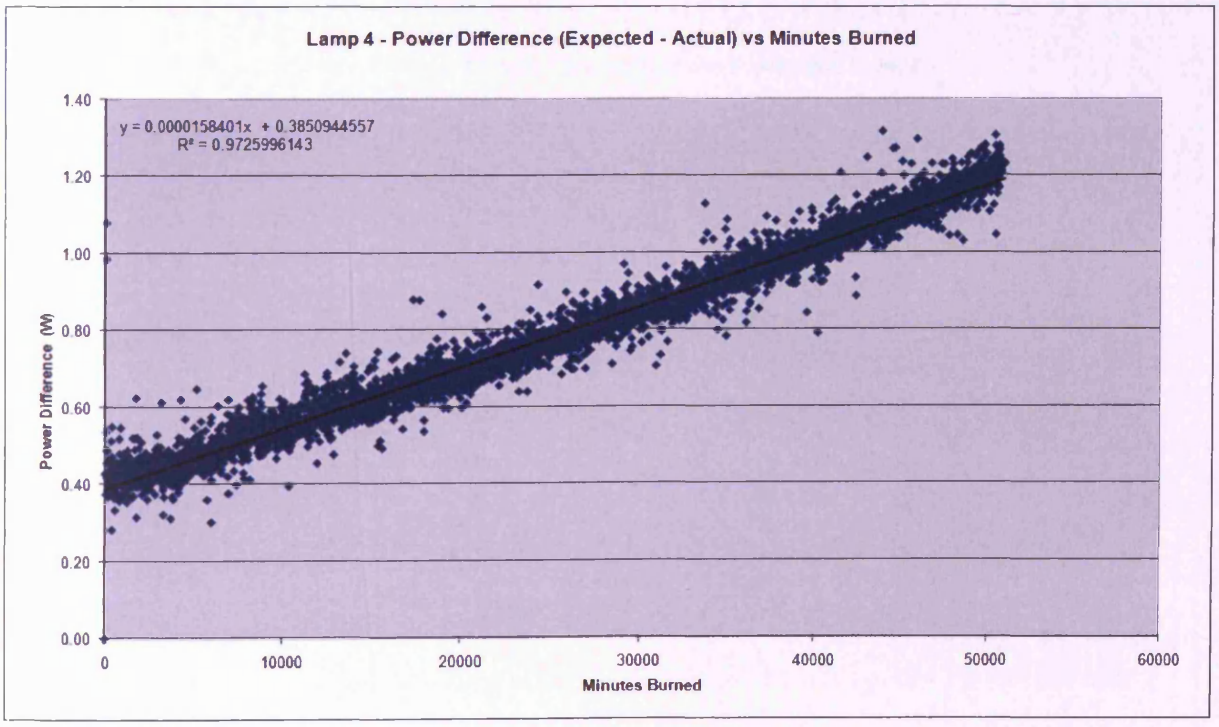
Lamp 2



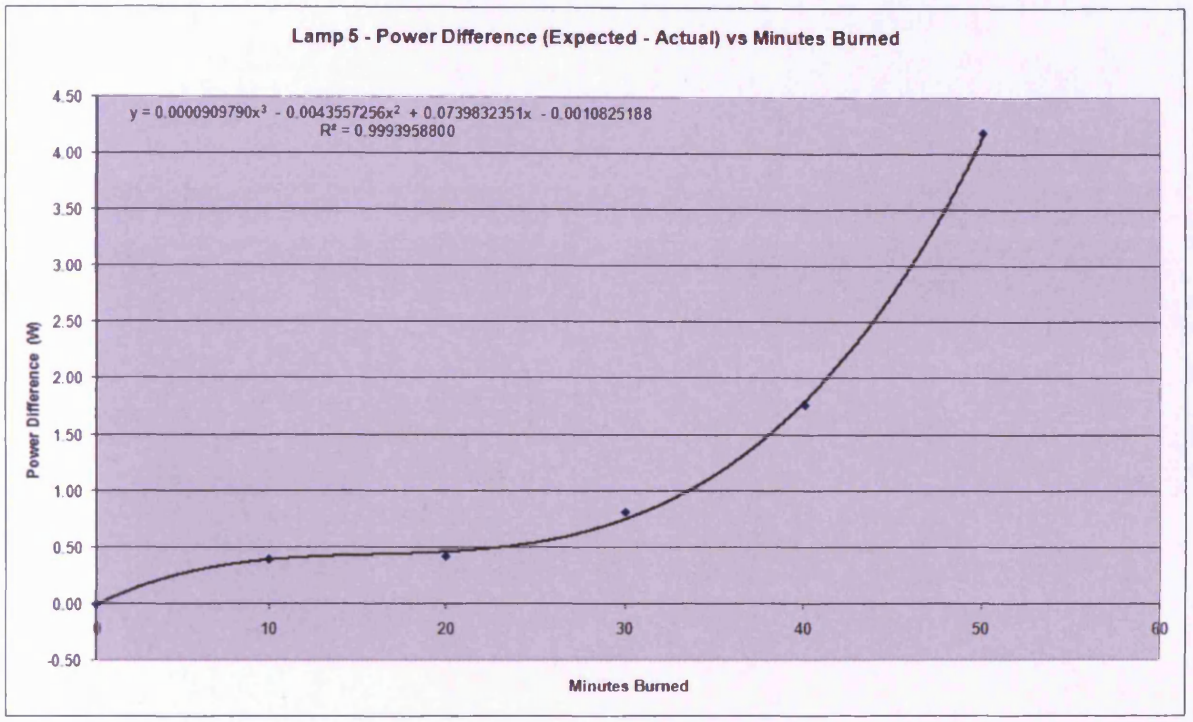
Lamp 3


Lamp 3 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000024499x + 0.4783066791
R² = 0.9597312032
Power Difference (W)
Minutes Burned
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80
2.00
0
100000
200000
300000
400000
500000
600000

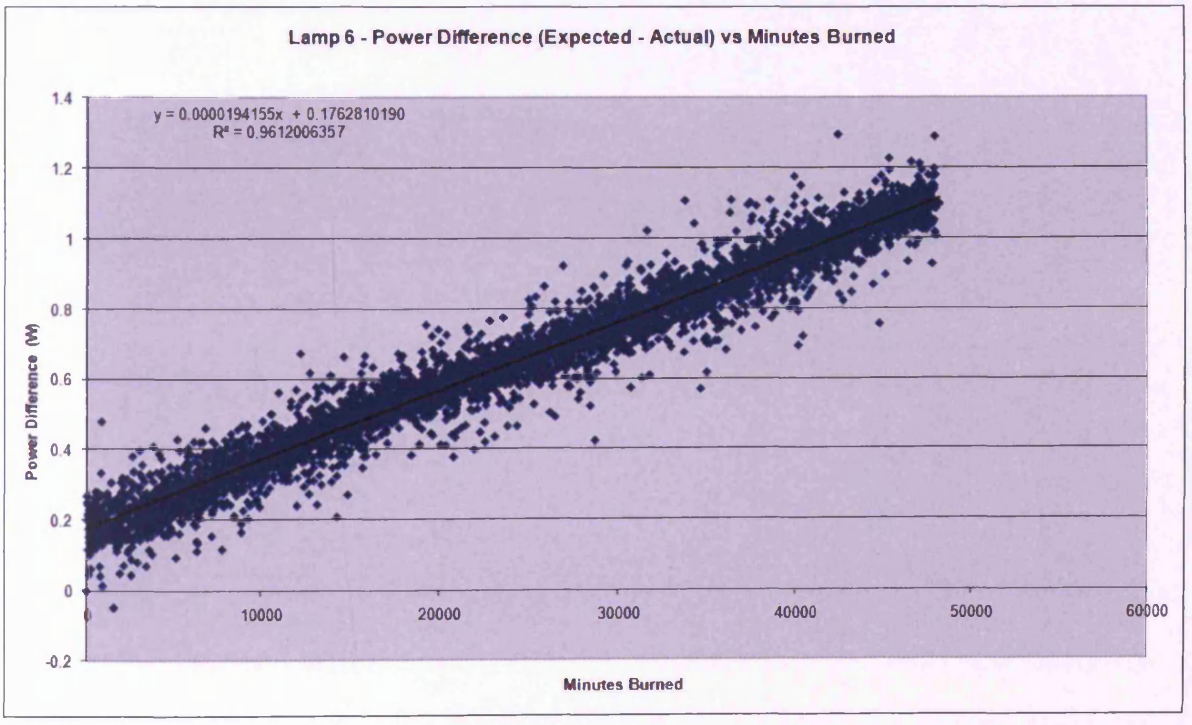
Lamp 4


Lamp 4 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000158401x + 0.3850944557
R² = 0.9725996143
Power Difference (W)
Minutes Burned
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
0
10000
20000
30000
40000
50000
60000

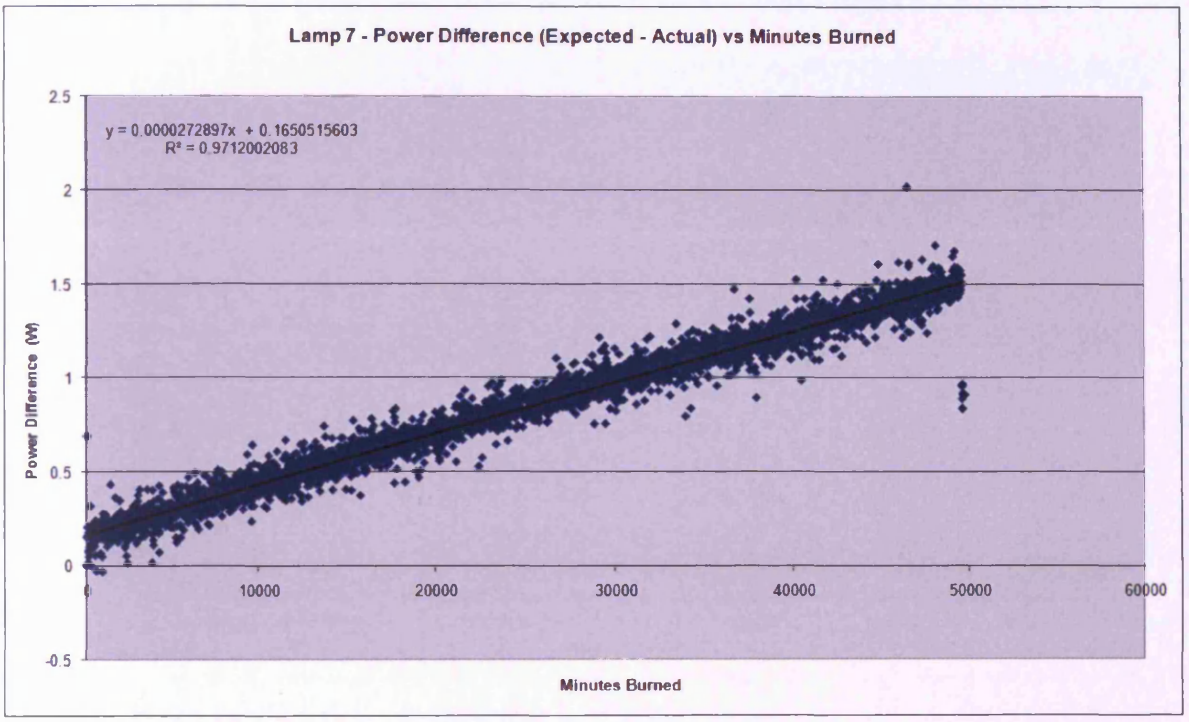
Lamp 5


Lamp 5 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000909790x³ - 0.0043557256x² + 0.0739832351x - 0.0010825188
R² = 0.9993958800
Power Difference (W)
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
-0.50
0
10
20
30
40
50
60
Minutes Burned

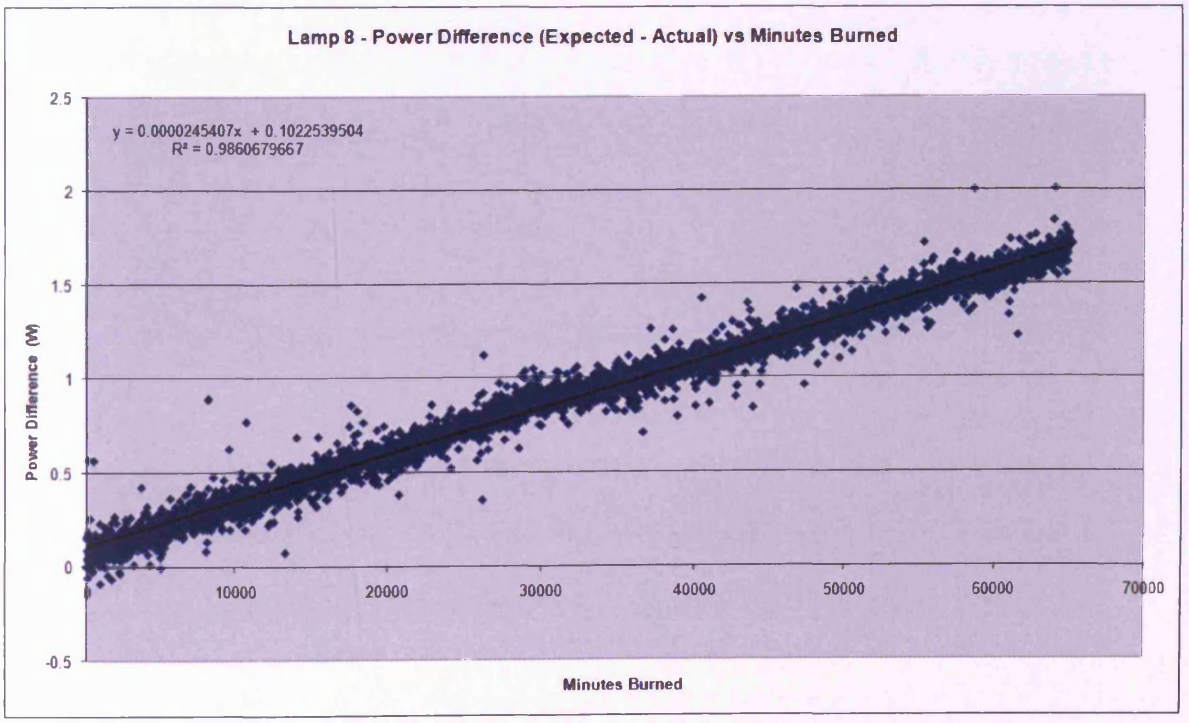
Lamp 6


Lamp 6 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000194155x + 0.1762810190
R² = 0.9612006357
Power Difference (W)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
10000
20000
30000
40000
50000
60000
Minutes Burned

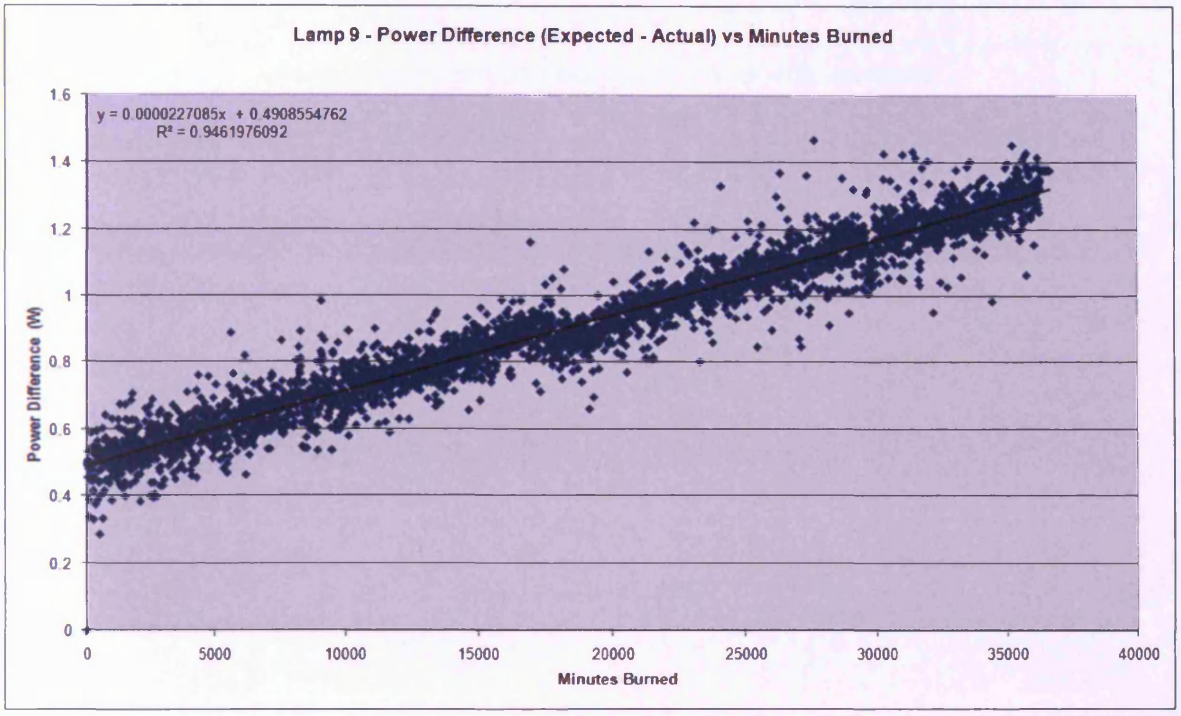
Lamp 7


Lamp 7 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000272897x + 0.1650515603
R² = 0.9712002083
Power Difference (W)
Minutes Burned
2.5
2
1.5
1
0.5
0
-0.5
10000
20000
30000
40000
50000
60000

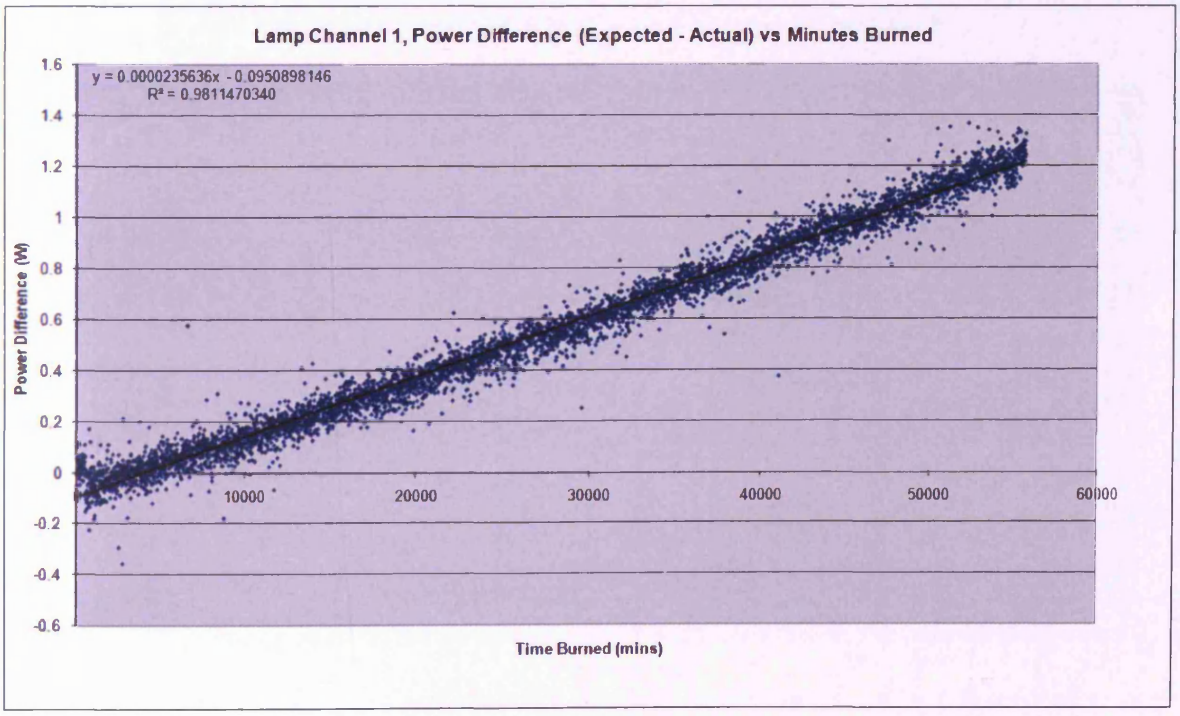
Lamp 8


Lamp 8 - Power Difference (Expected - Actual) vs Minutes Burned
y = 0.0000245407x + 0.1022539504
R² = 0.9860679667
Power Difference (W)
Minutes Burned
2.5
2
1.5
1
0.5
0
-0.5
10000
20000
30000
40000
50000
60000
70000

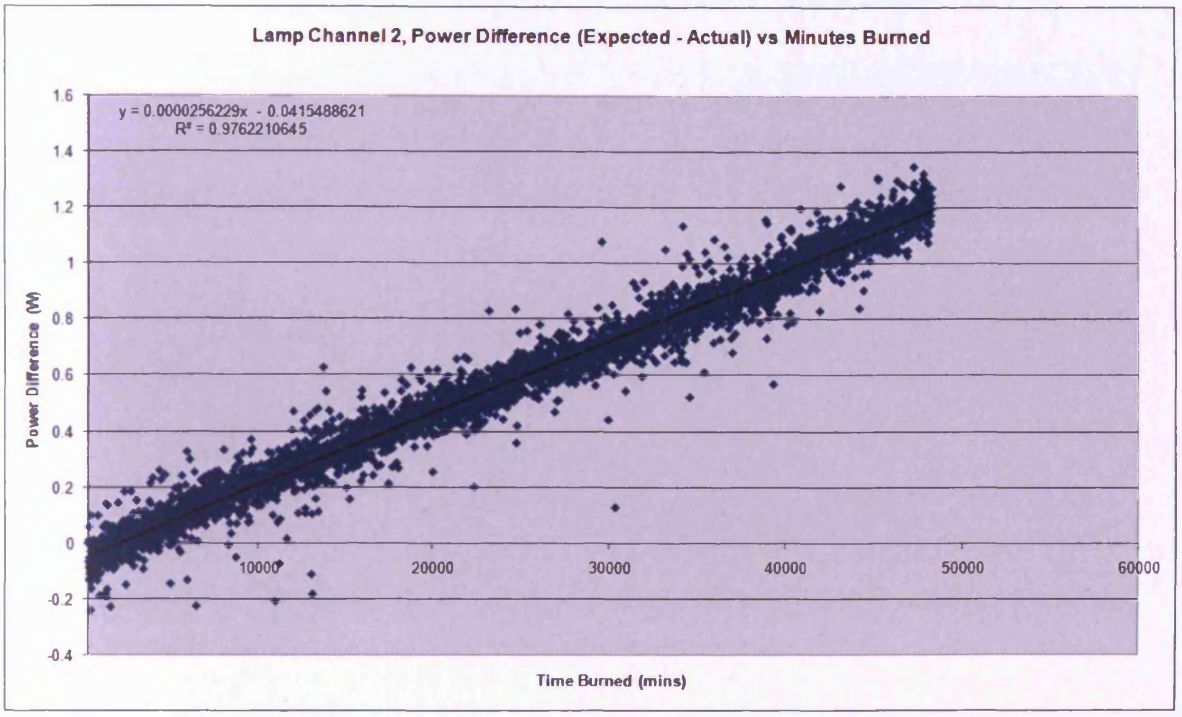
Lamp 9 (Channel 5)



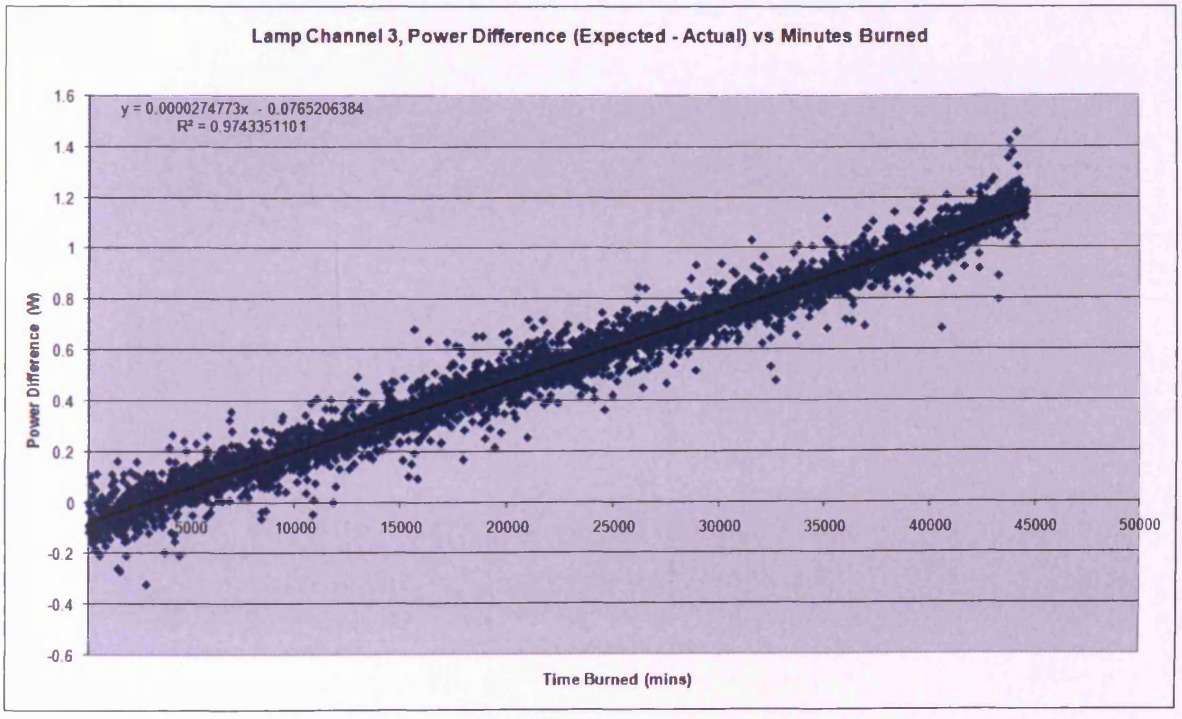
Lamp 10 (Channel 1)



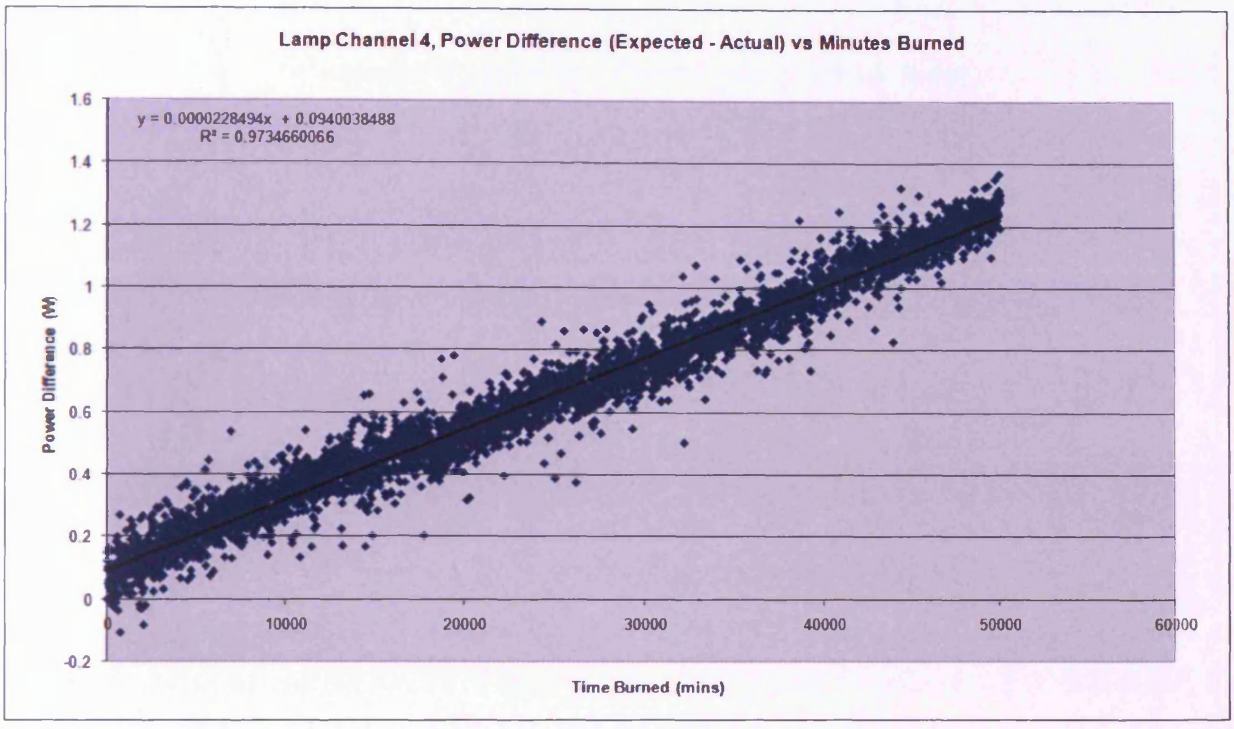
Lamp 11 (Channel 2)



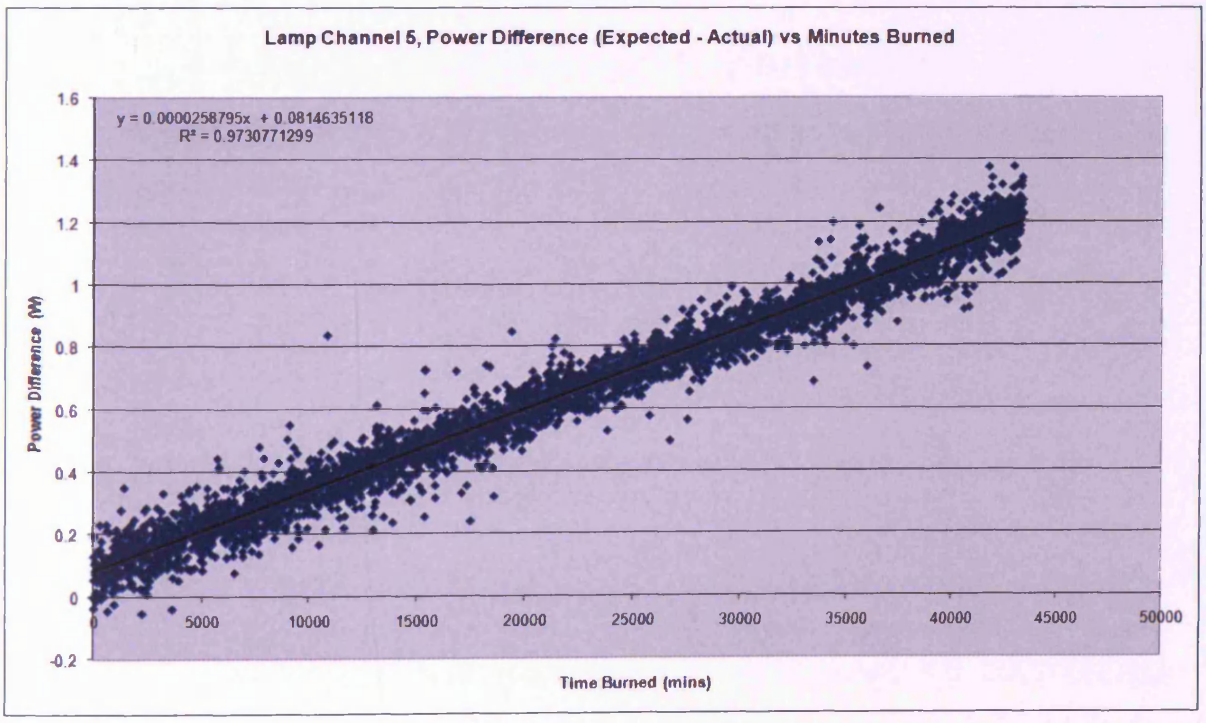
Lamp 12 (Channel 3)



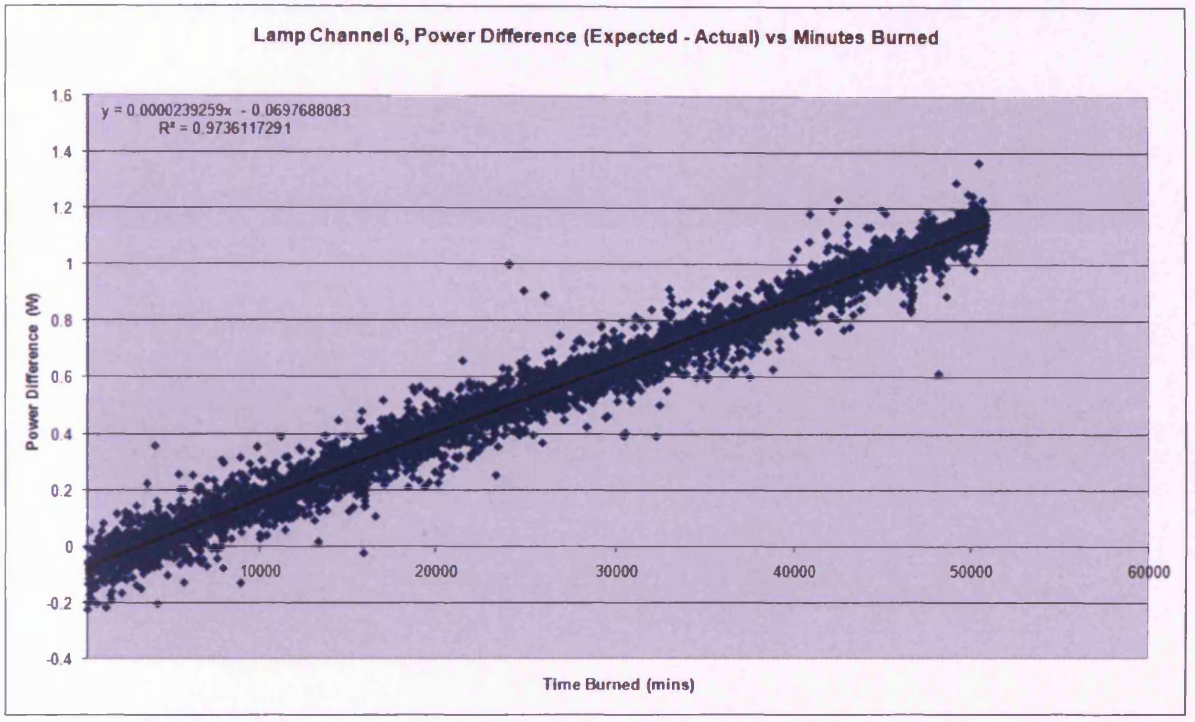
Lamp 13 (Channel 4)



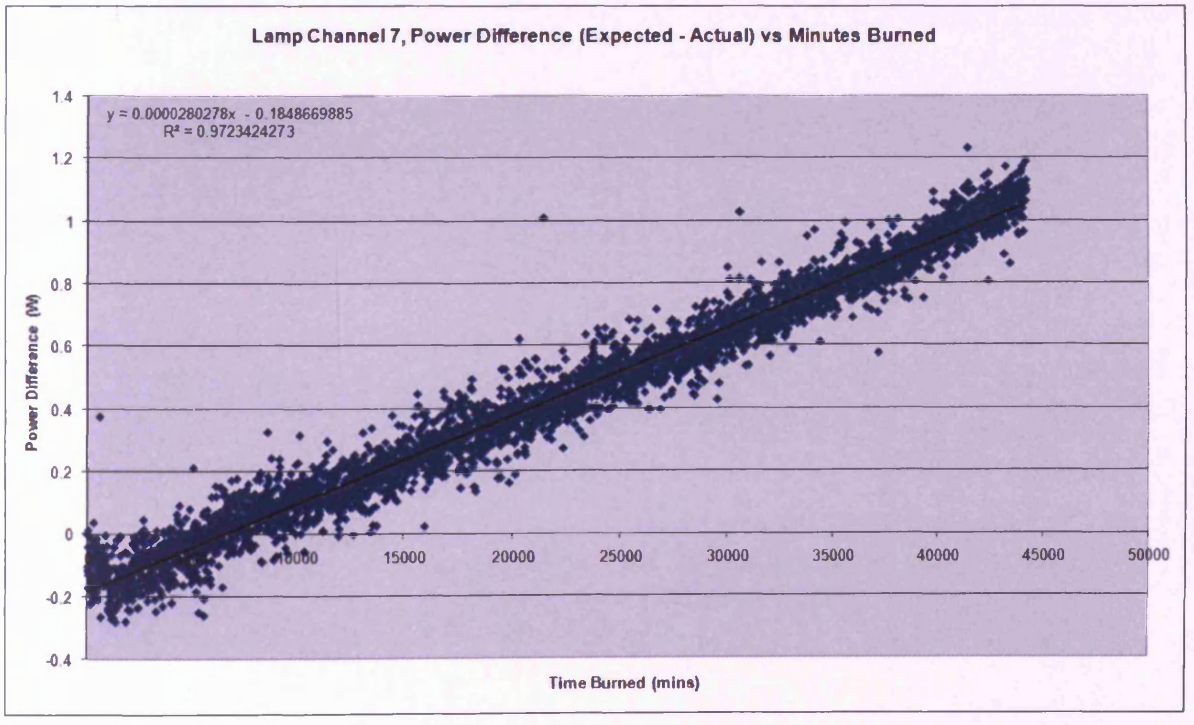
Lamp 14 (Channel 5)



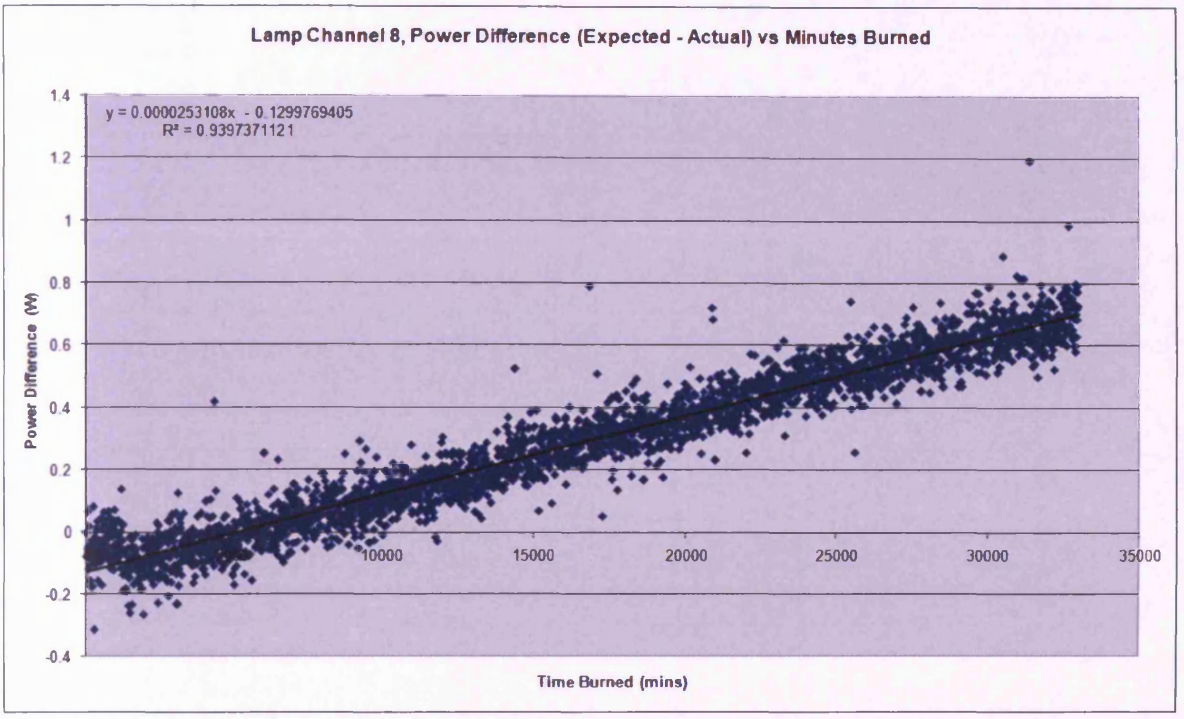
Lamp 15 (Channel 6)



Lamp 16 (Channel 7)



Lamp 17 (Channel 8)



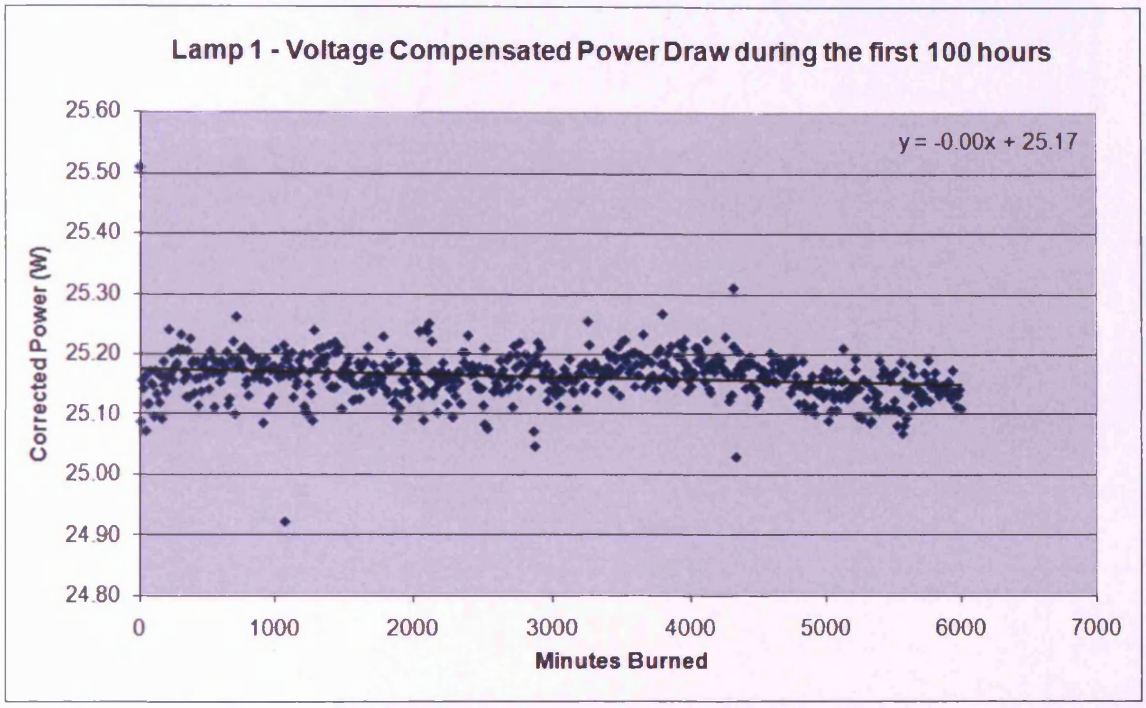
Voltage Compensation Results

AC Voltage RMS	Power (W)			
Volts +/- 0.1(V)	25W	40W	60W	100W
0	0.00	0.00	0.00	0.00
5	0.05	0.16	0.24	0.39
10	0.15	0.45	0.67	1.04
15	0.28	0.78	1.16	1.79
20	0.45	1.15	1.69	2.62
25	0.65	1.55	2.28	3.57
30	0.87	1.98	2.92	4.60
35	1.12	2.45	3.61	5.70
40	1.39	2.94	4.34	6.90
45	1.67	3.46	5.10	8.15
50	1.99	3.98	5.94	9.51
55	2.32	4.54	6.80	10.95
60	2.67	5.14	7.72	12.47
65	3.04	5.77	8.67	14.05
70	3.44	6.41	9.67	15.71
75	3.85	7.08	10.70	17.42
80	4.27	7.78	11.80	19.22
85	4.71	8.50	12.91	21.05
90	5.17	9.23	14.05	22.96
95	5.63	10.00	15.20	24.92
100	6.11	10.78	16.43	26.94
105	6.60	11.58	17.67	29.03
110	7.12	12.40	18.94	31.17
115	7.67	13.25	20.26	33.35
120	8.21	14.10	21.60	35.57
125	8.76	14.98	22.95	37.88
130	9.35	15.93	24.34	40.21
135	9.91	16.83	25.79	42.57
140	10.50	17.78	27.19	45.00
145	11.11	18.72	28.65	47.46
150	11.72	19.73	30.15	50.00
155	12.34	20.72	31.67	52.53
160	12.98	21.84	33.22	55.17
165	13.61	22.89	34.80	57.80
170	14.28	23.94	36.40	60.50
175	14.95	25.01	38.03	63.25
180	15.62	26.10	39.67	66.01
185	16.32	27.20	41.35	68.82
190	17.02	28.33	43.04	71.67
195	17.75	29.46	44.75	74.55
200	18.44	30.58	46.50	77.52
205	19.17	31.75	48.24	80.48
210	19.89	33.08	50.02	83.54
215	20.62	34.27	51.84	86.58

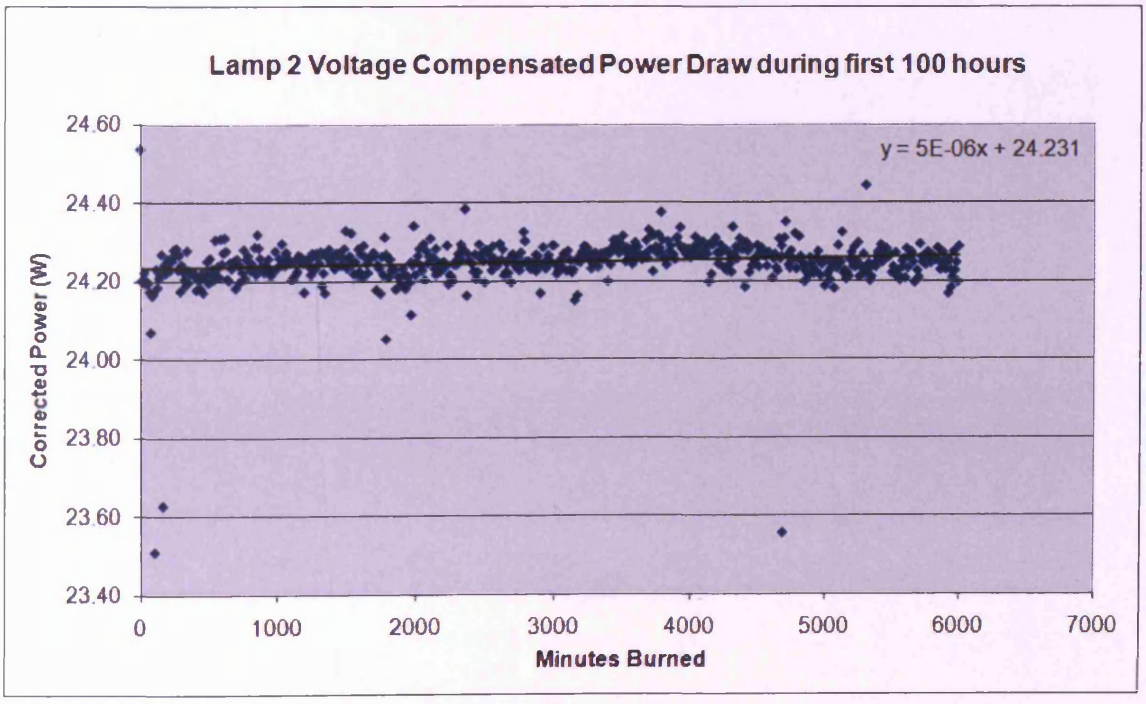
220	21.36	35.46	53.68	89.67
225	22.12	36.72	55.53	92.81
230	22.91	37.95	57.41	96.00
235	23.69	39.20	59.31	99.24
240	24.48	40.46	61.25	102.50
245	25.28	41.75	63.19	105.77
250	26.10	43.05	65.18	109.13
255	26.93	44.37	67.17	112.51
260	27.74	45.68	69.19	115.96
265	28.59	47.04	71.23	119.38

Initial Power Draw (Over 100 Hours)

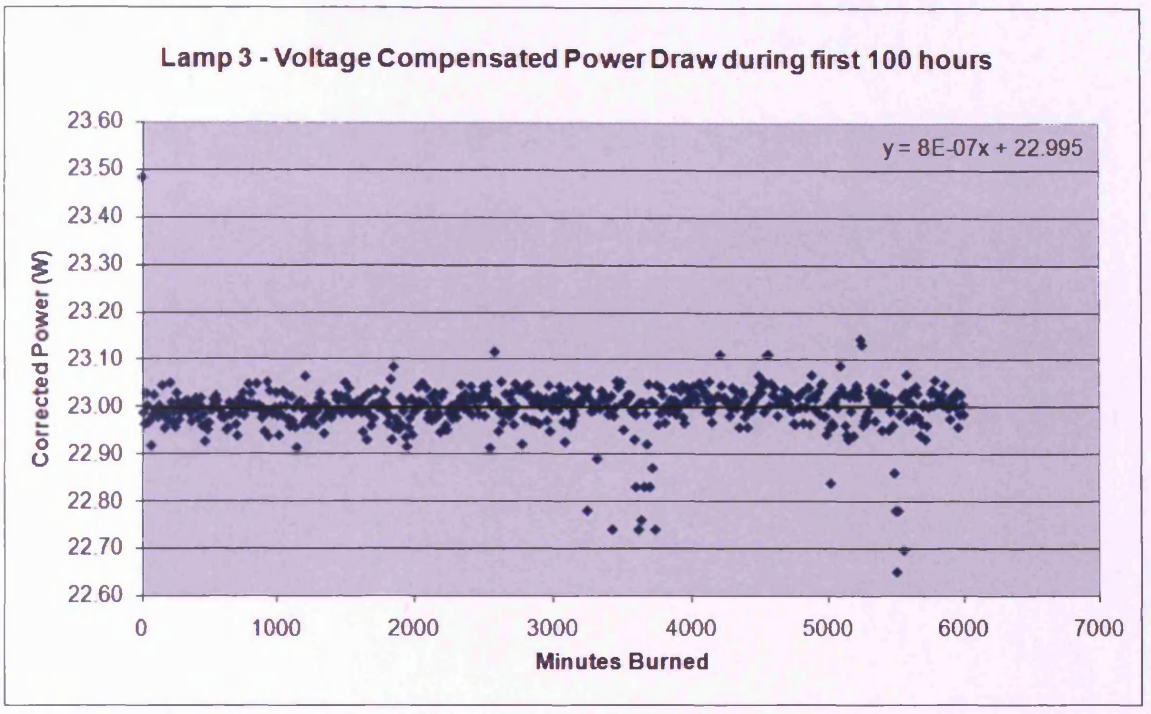
Lamp 1



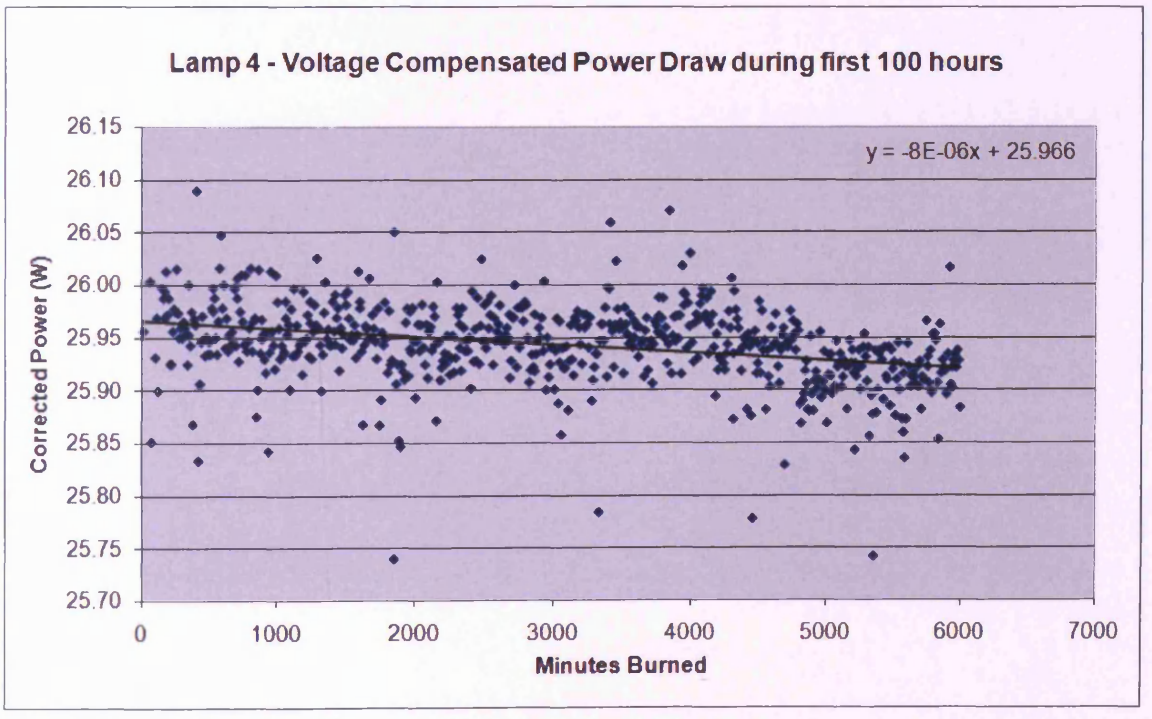
Lamp 2



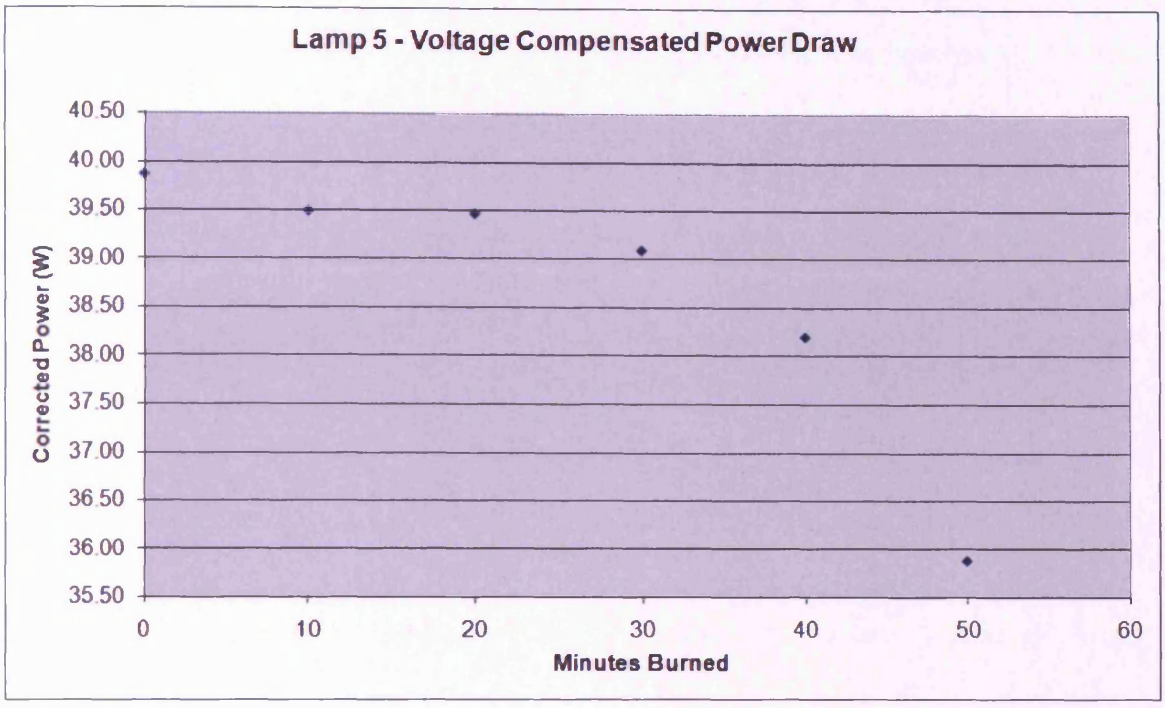
Lamp 3


Lamp 3 - Voltage Compensated Power Draw during first 100 hours
y = 8E-07x + 22.995
Corrected Power (W)
23.60
23.50
23.40
23.30
23.20
23.10
23.00
22.90
22.80
22.70
22.60
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

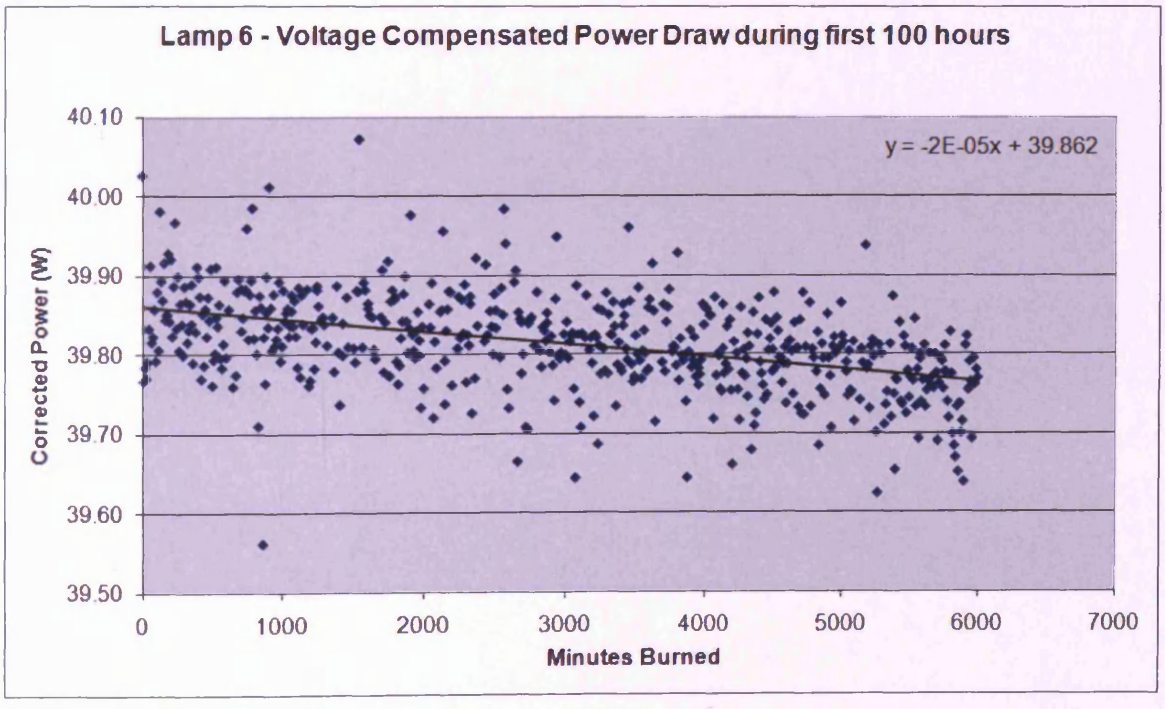
Lamp 4


Lamp 4 - Voltage Compensated Power Draw during first 100 hours
y = -8E-06x + 25.966
Corrected Power (W)
26.15
26.10
26.05
26.00
25.95
25.90
25.85
25.80
25.75
25.70
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

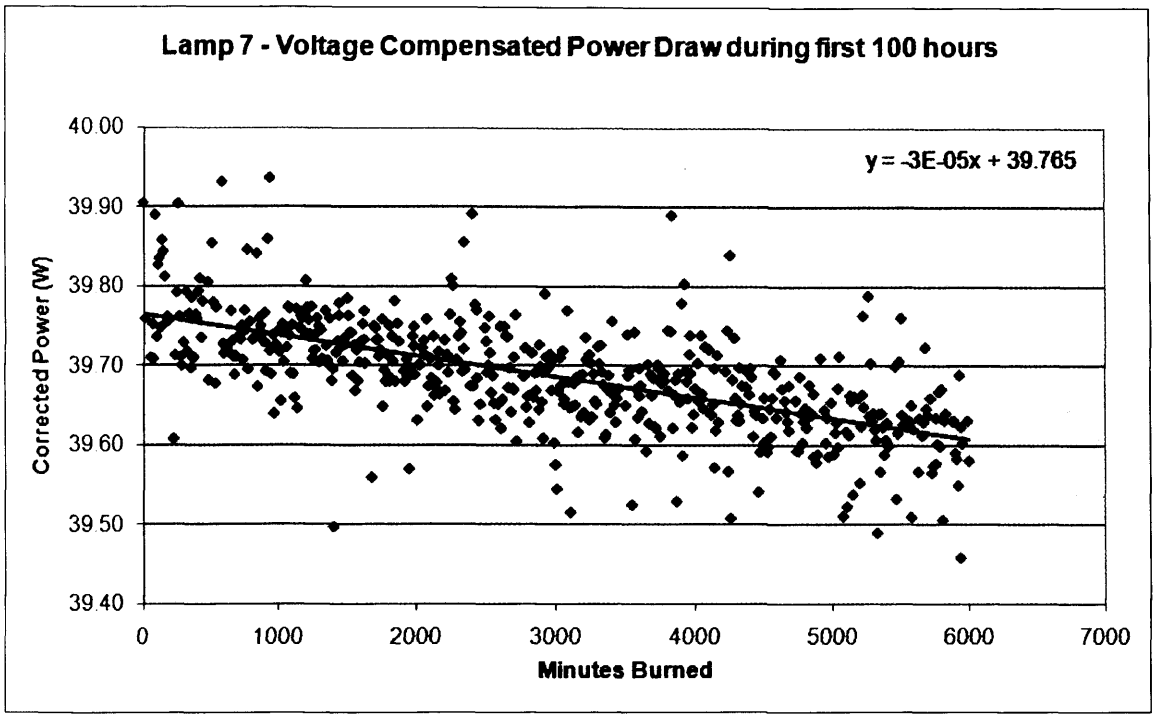
Lamp 5


Lamp 5 - Voltage Compensated Power Draw
Corrected Power (W)
40.50
40.00
39.50
39.00
38.50
38.00
37.50
37.00
36.50
36.00
35.50
0
10
20
30
40
50
60
Minutes Burned

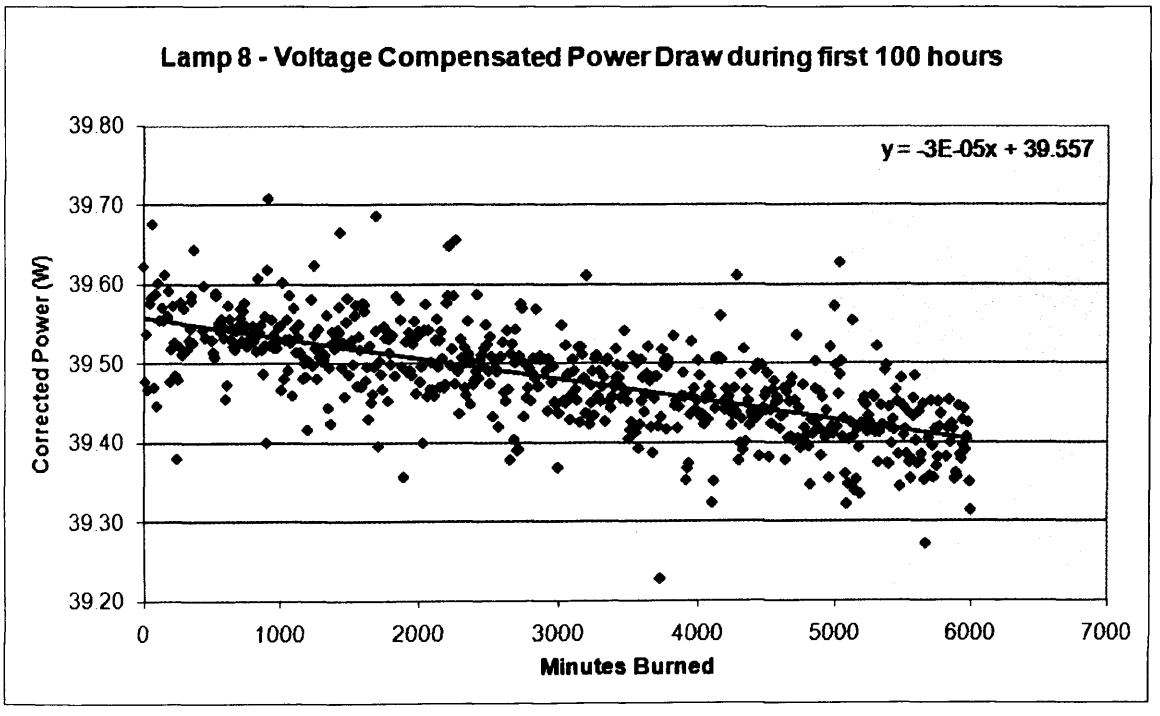
Lamp 6


Lamp 6 - Voltage Compensated Power Draw during first 100 hours
y = -2E-05x + 39.862
Corrected Power (W)
40.10
40.00
39.90
39.80
39.70
39.60
39.50
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

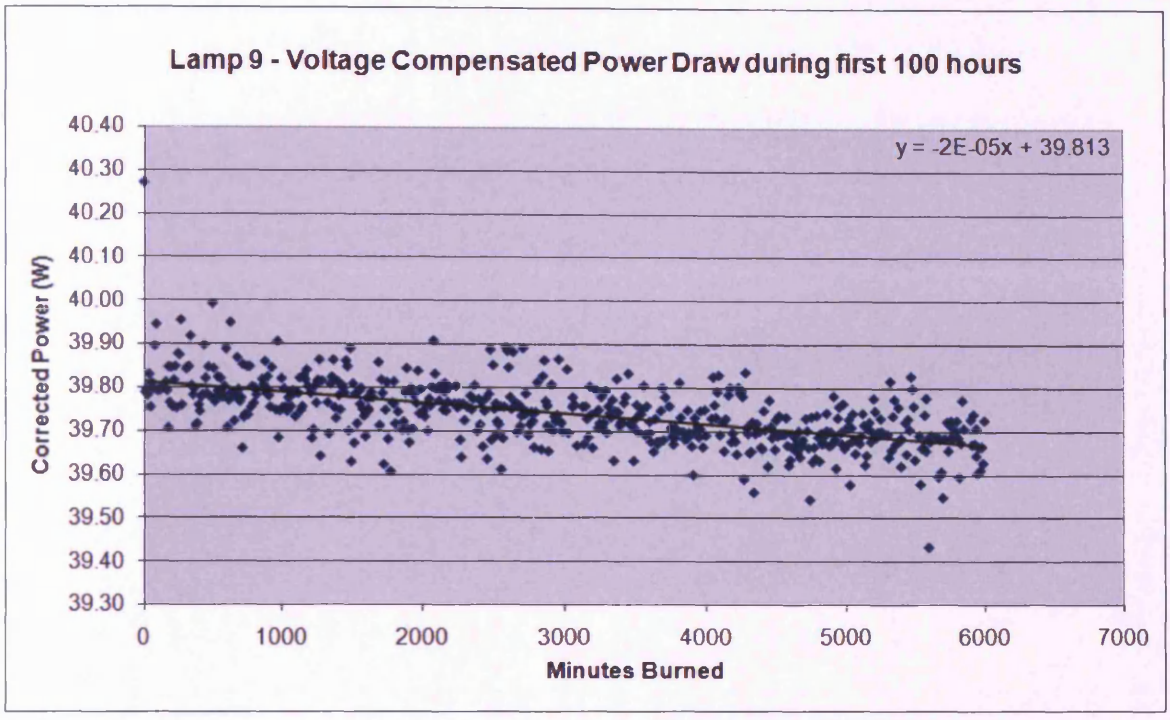
Lamp 7


Lamp 7 - Voltage Compensated Power Draw during first 100 hours
y = -3E-05x + 39.765
Corrected Power (W)
40.00
39.90
39.80
39.70
39.60
39.50
39.40
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

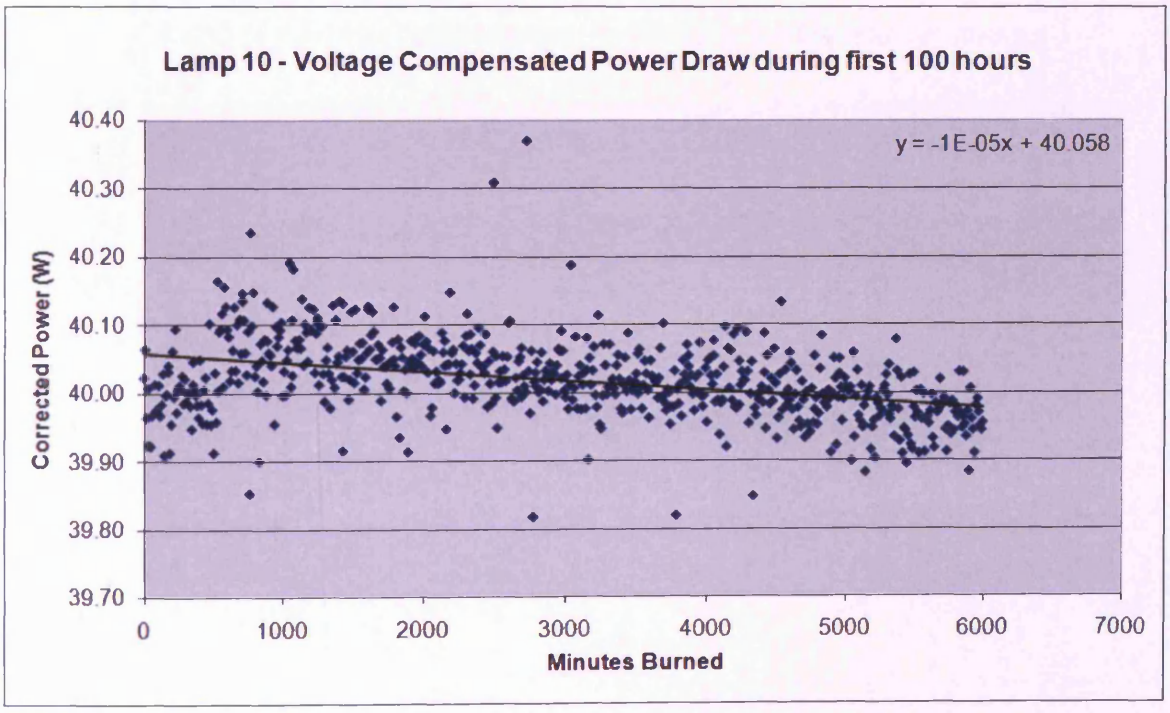
Lamp 8


Lamp 8 - Voltage Compensated Power Draw during first 100 hours
y = -3E-05x + 39.557
Corrected Power (W)
39.80
39.70
39.60
39.50
39.40
39.30
39.20
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

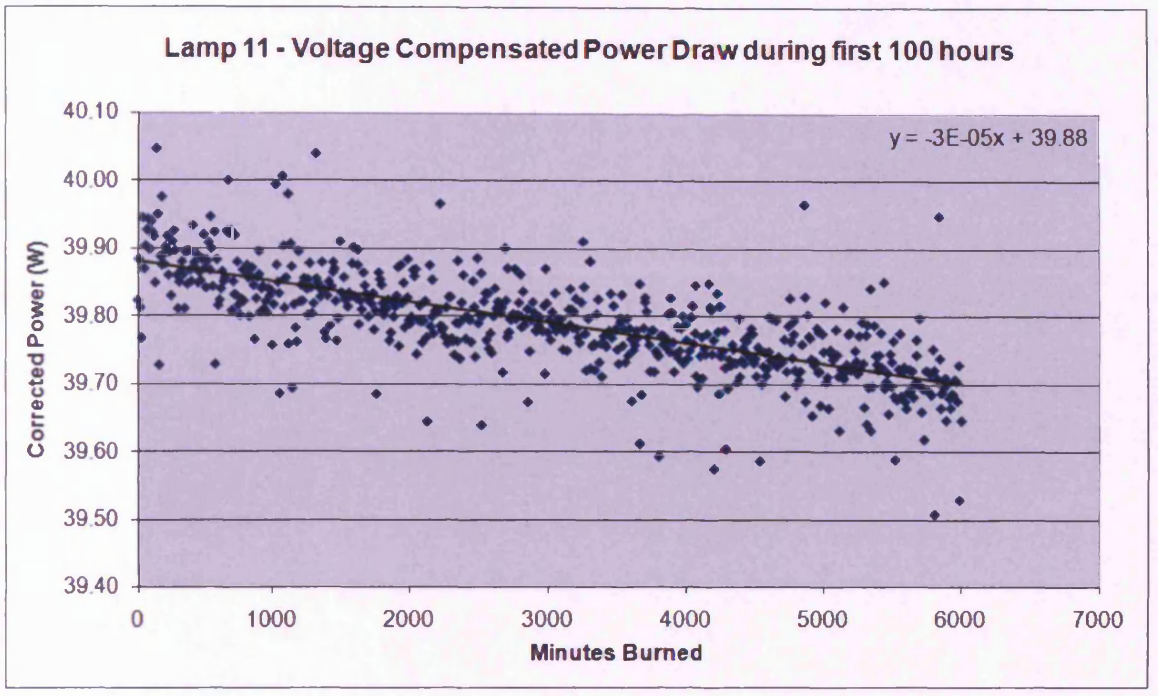
Lamp 9



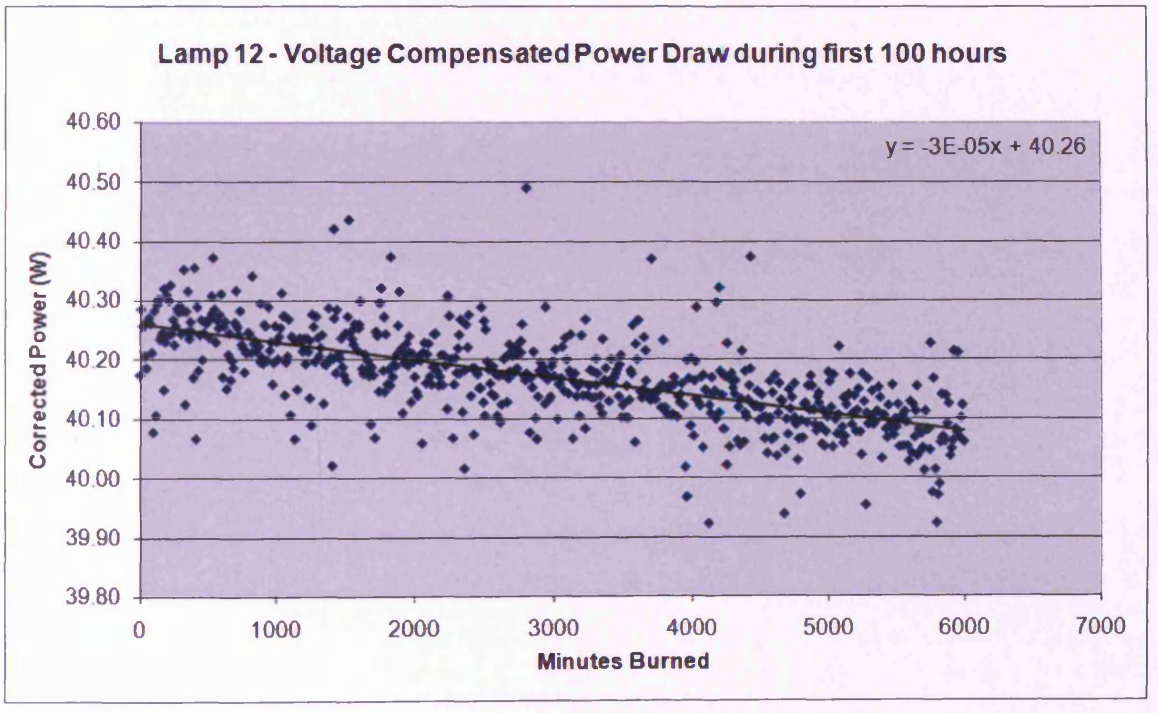
Lamp 10



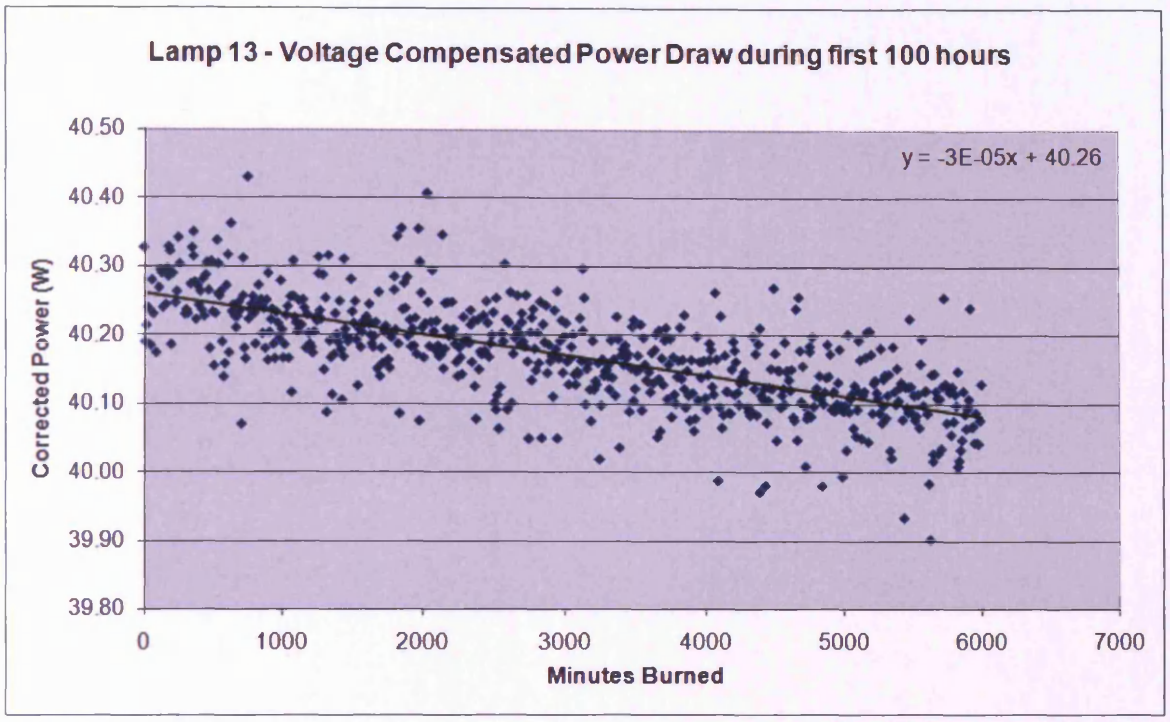
Lamp 11



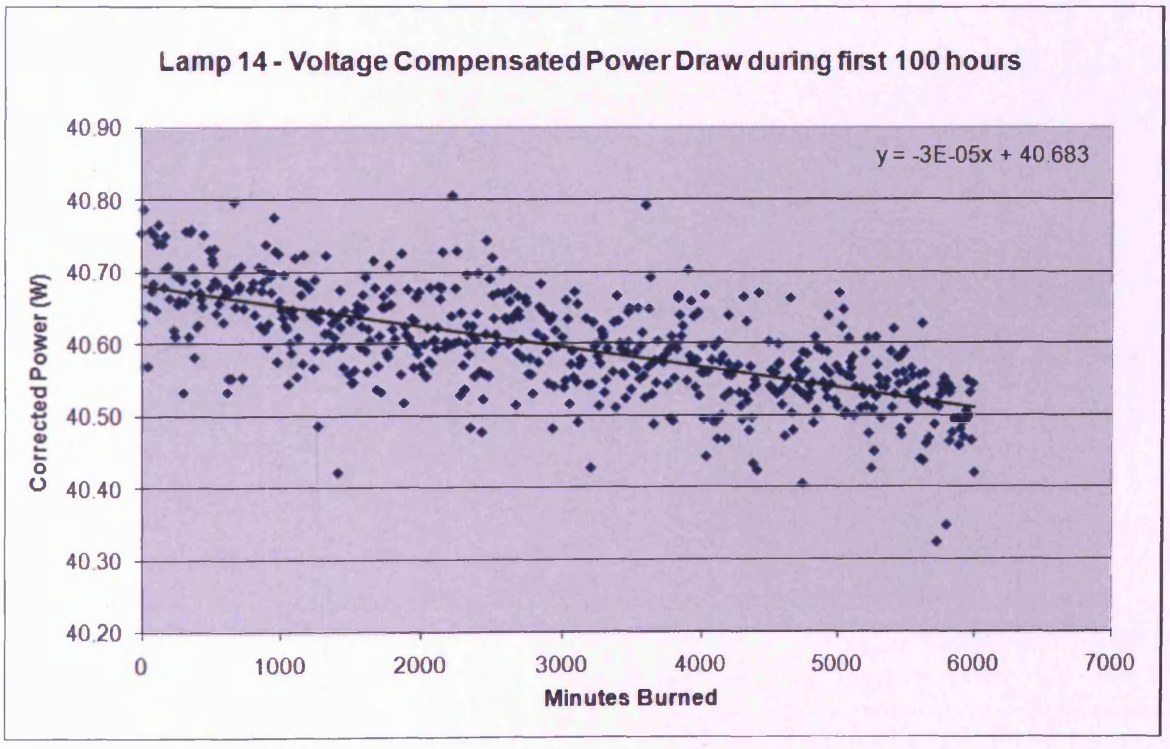
Lamp 12



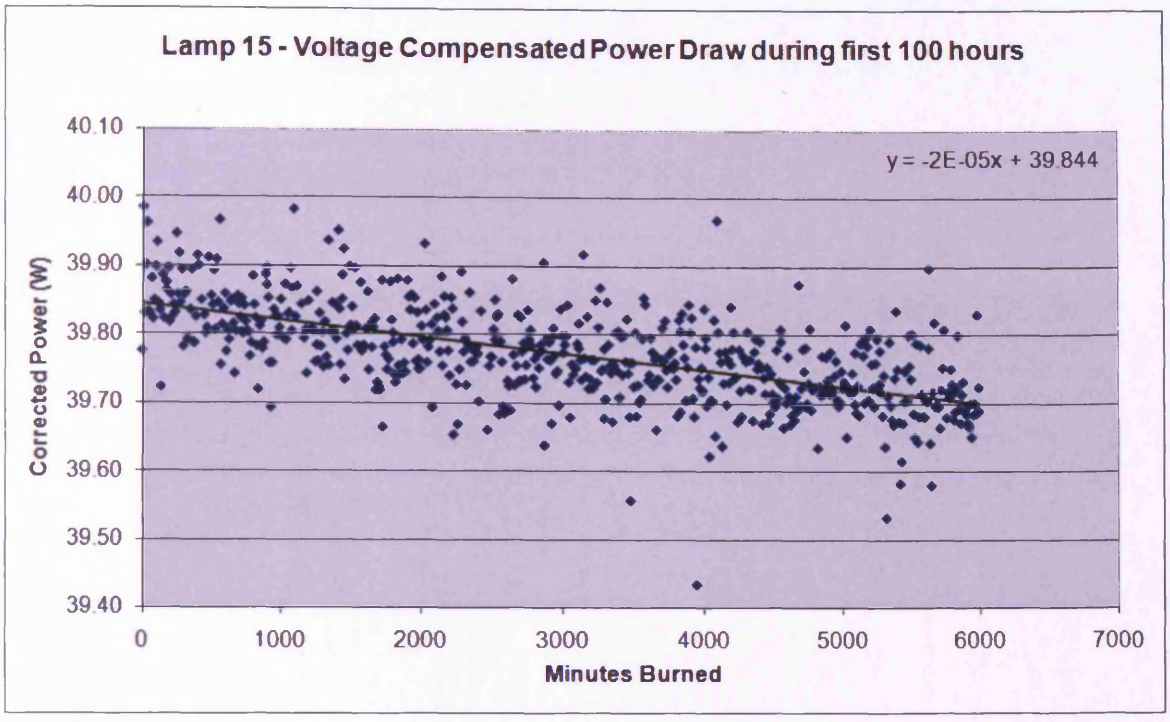
Lamp 13



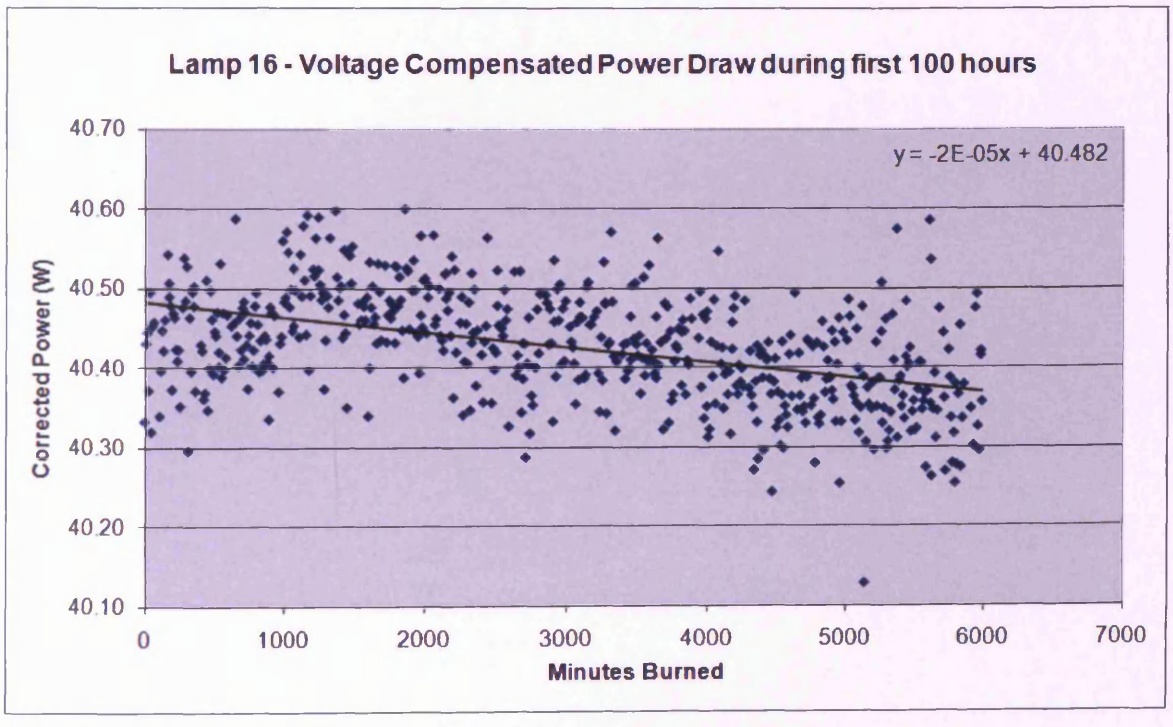
Lamp 14



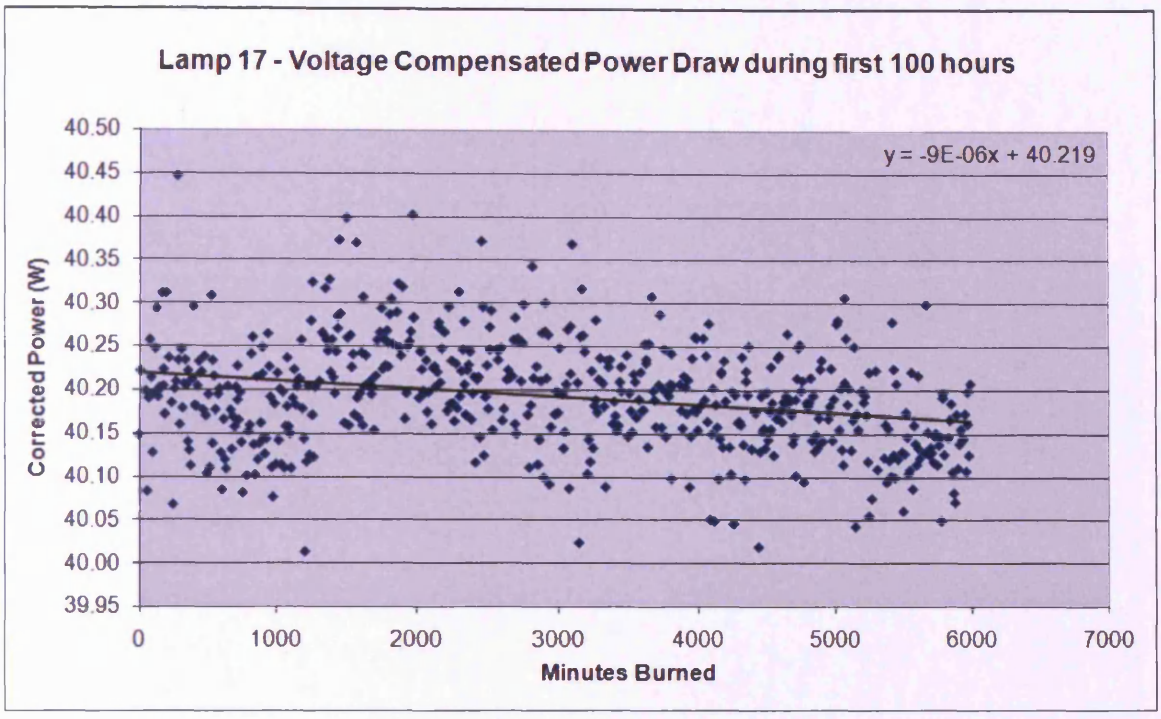
Lamp 15



Lamp 16

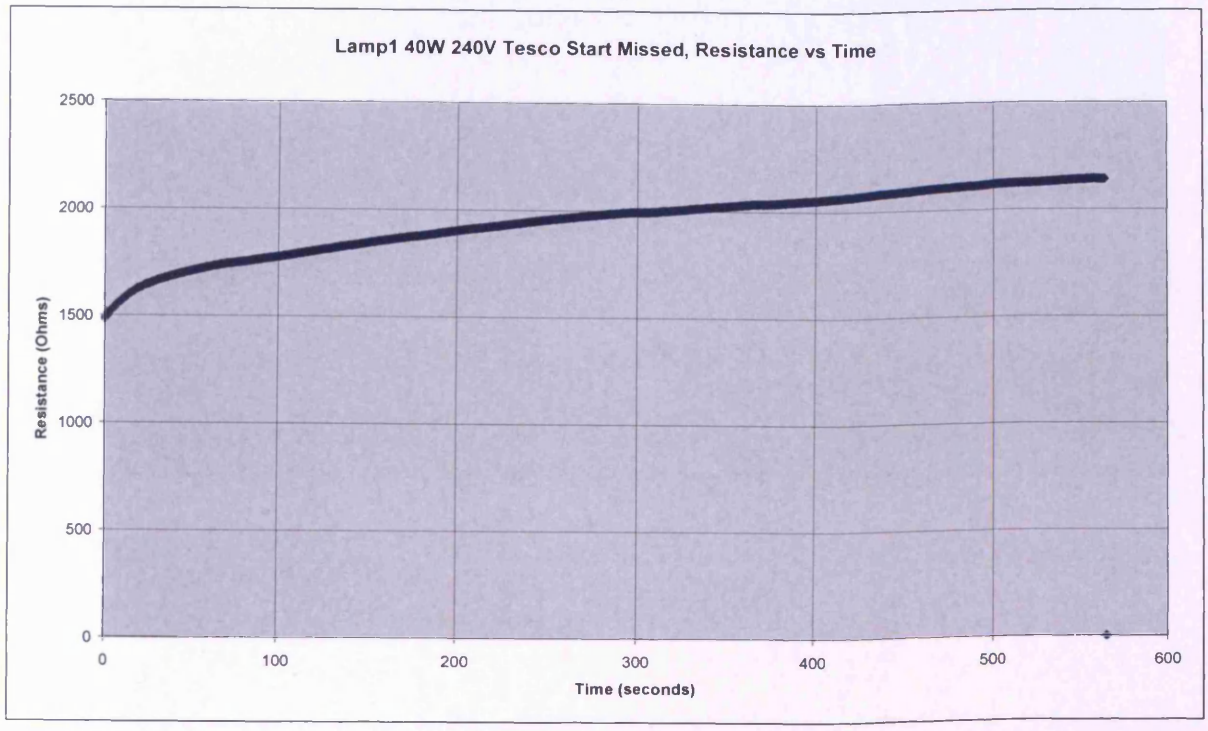


Lamp 17

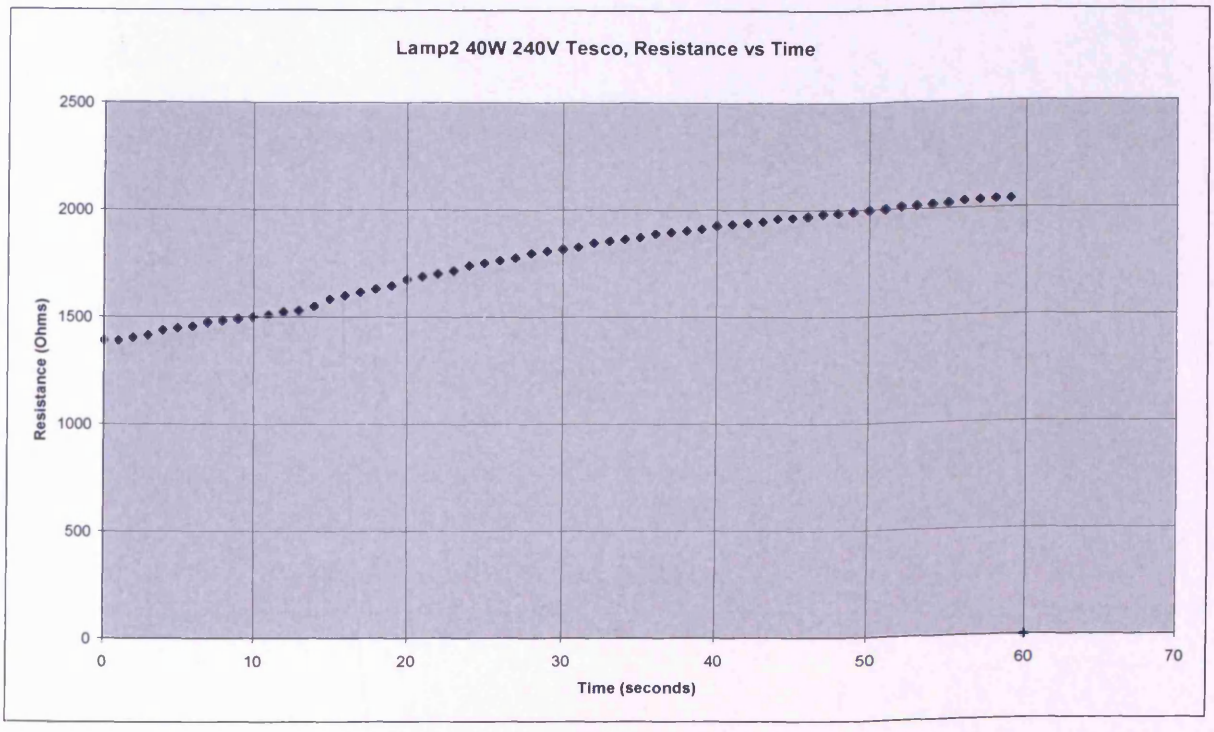

Lamp 17 - Voltage Compensated Power Draw during first 100 hours
y = -9E-06x + 40.219
Corrected Power (W)
40.50
40.45
40.40
40.35
40.30
40.25
40.20
40.15
40.10
40.05
40.00
39.95
0
1000
2000
3000
4000
5000
6000
7000
Minutes Burned

Incorrectly Filled Lamps

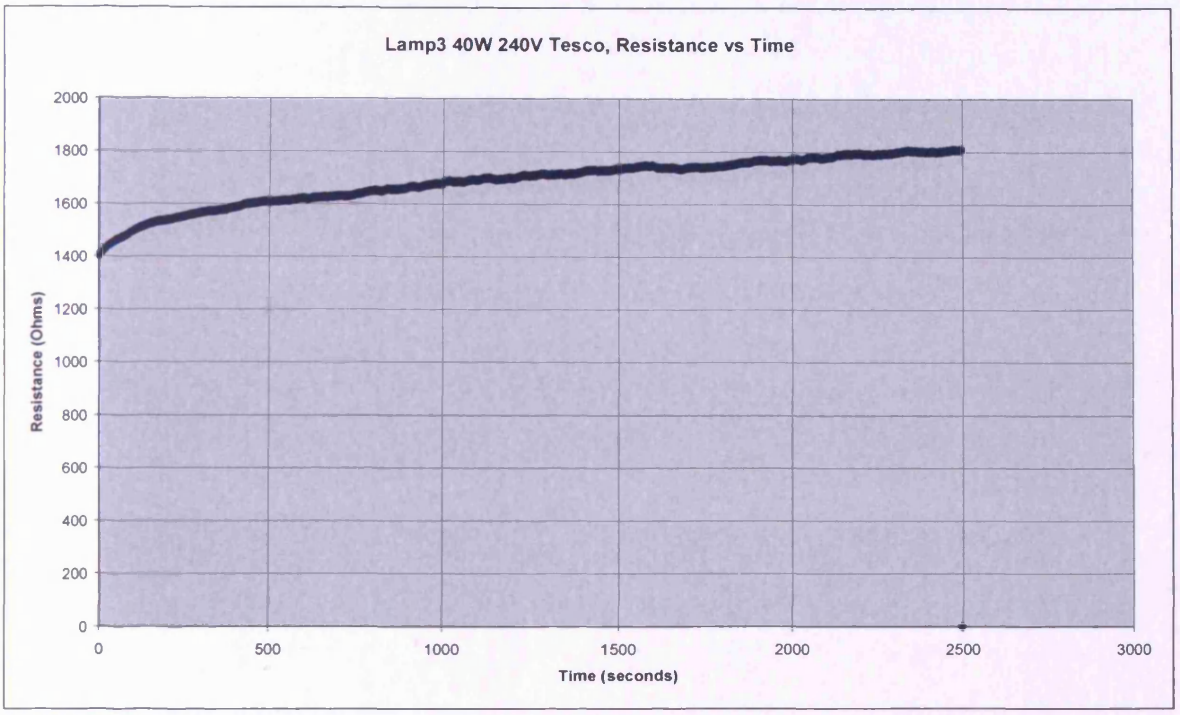
Lamp 1



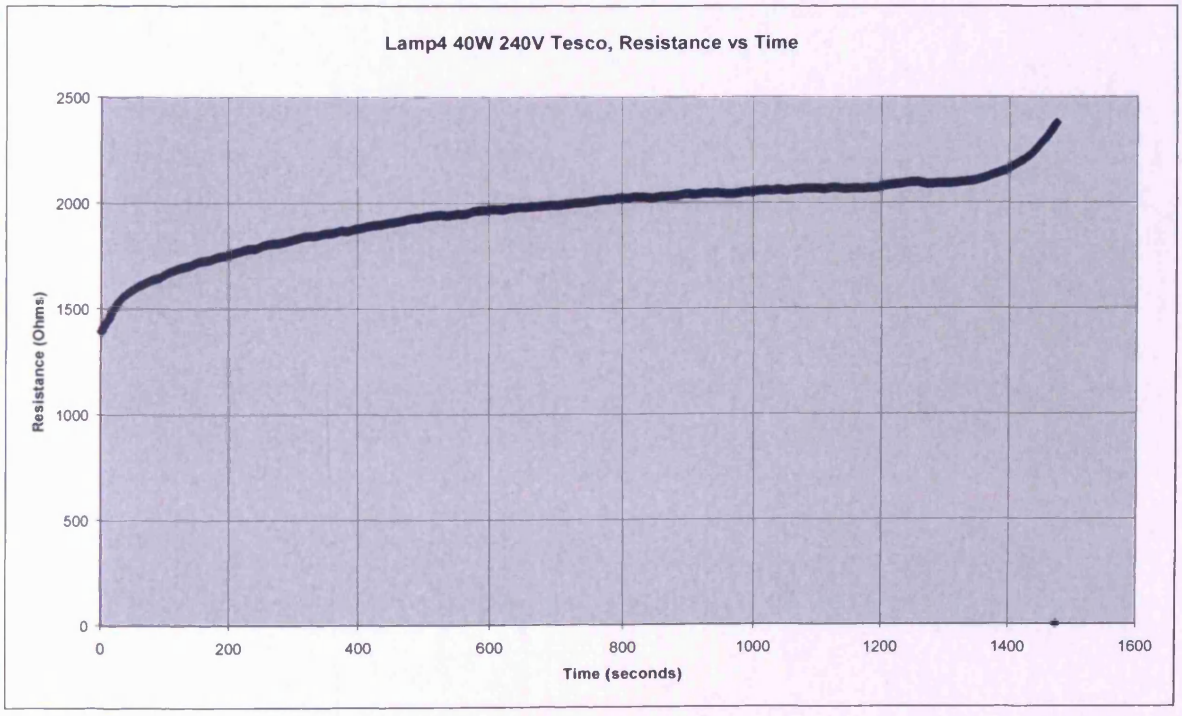
Lamp 2



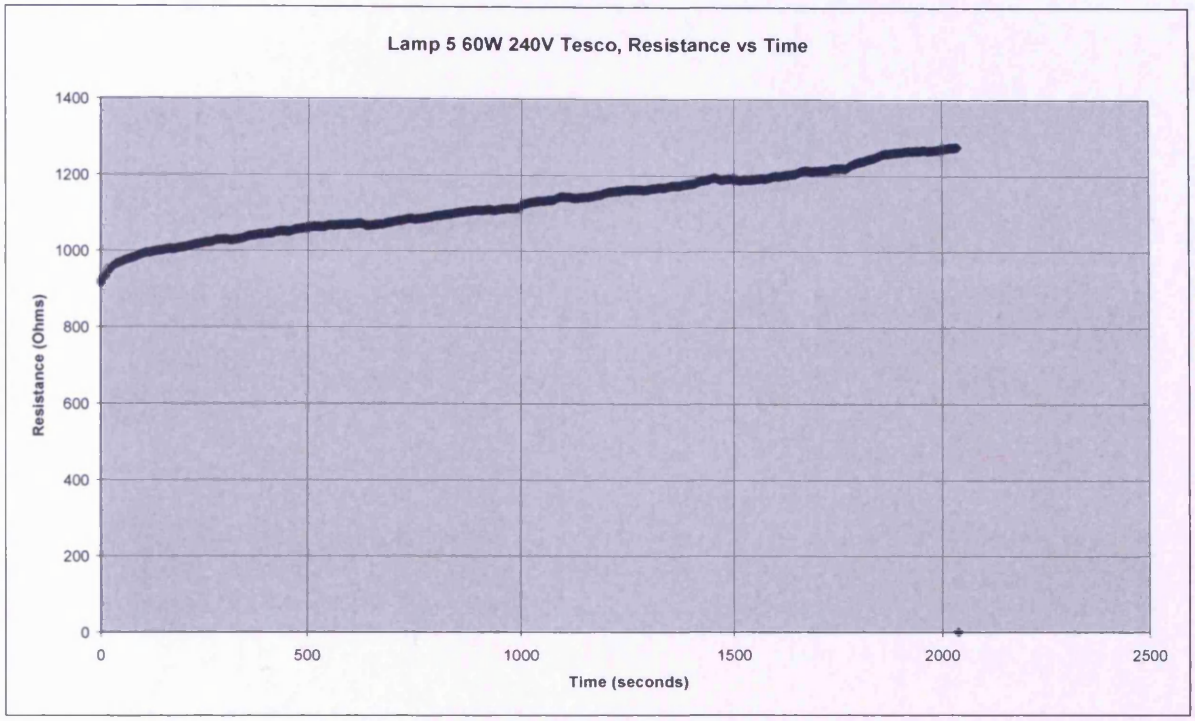
Lamp 3


Lamp3 40W 240V Tesco, Resistance vs Time
Resistance (Ohms)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
3000
Time (seconds)

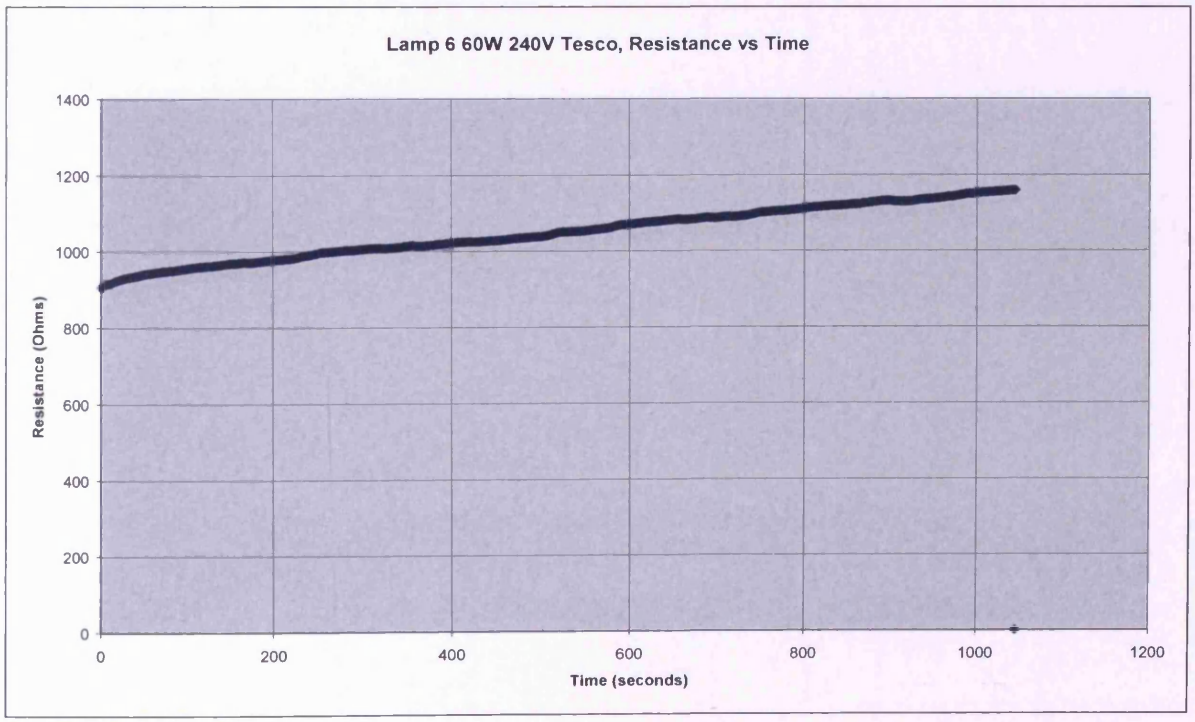
Lamp 4


Lamp4 40W 240V Tesco, Resistance vs Time
Resistance (Ohms)
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200
1400
1600
Time (seconds)

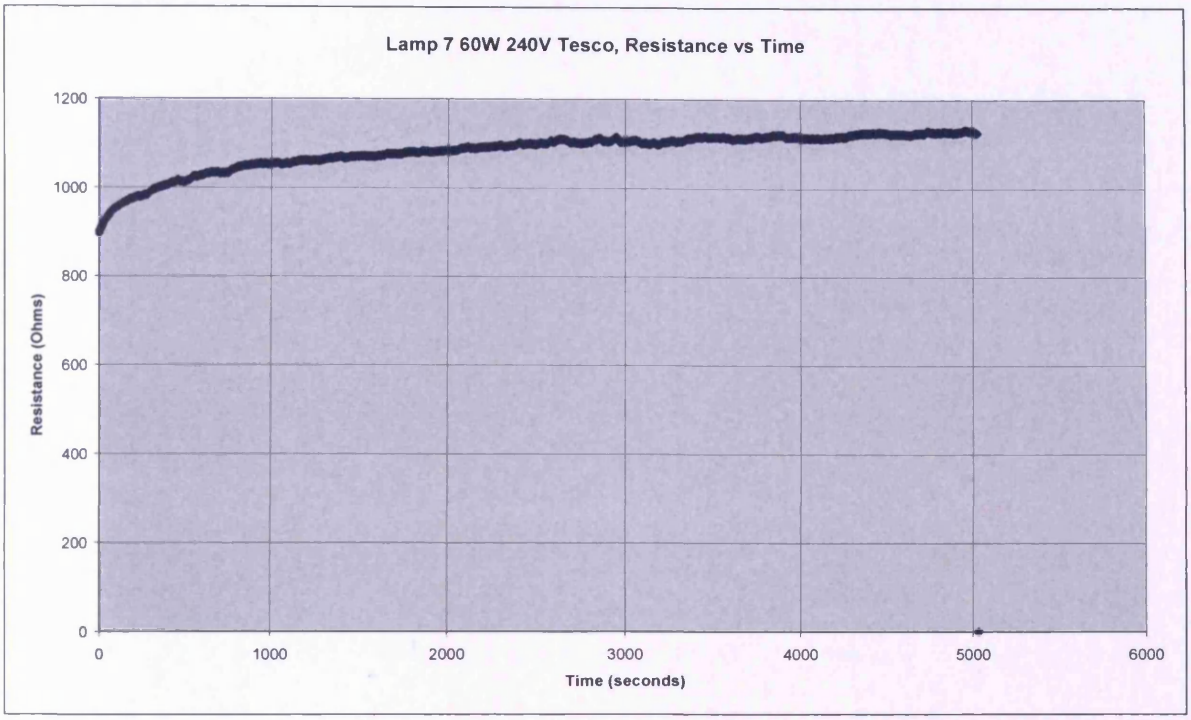
Lamp 5


Lamp 5 60W 240V Tesco, Resistance vs Time
Resistance (Ohms)
1400
1200
1000
800
600
400
200
0
0
500
1000
1500
2000
2500
Time (seconds)

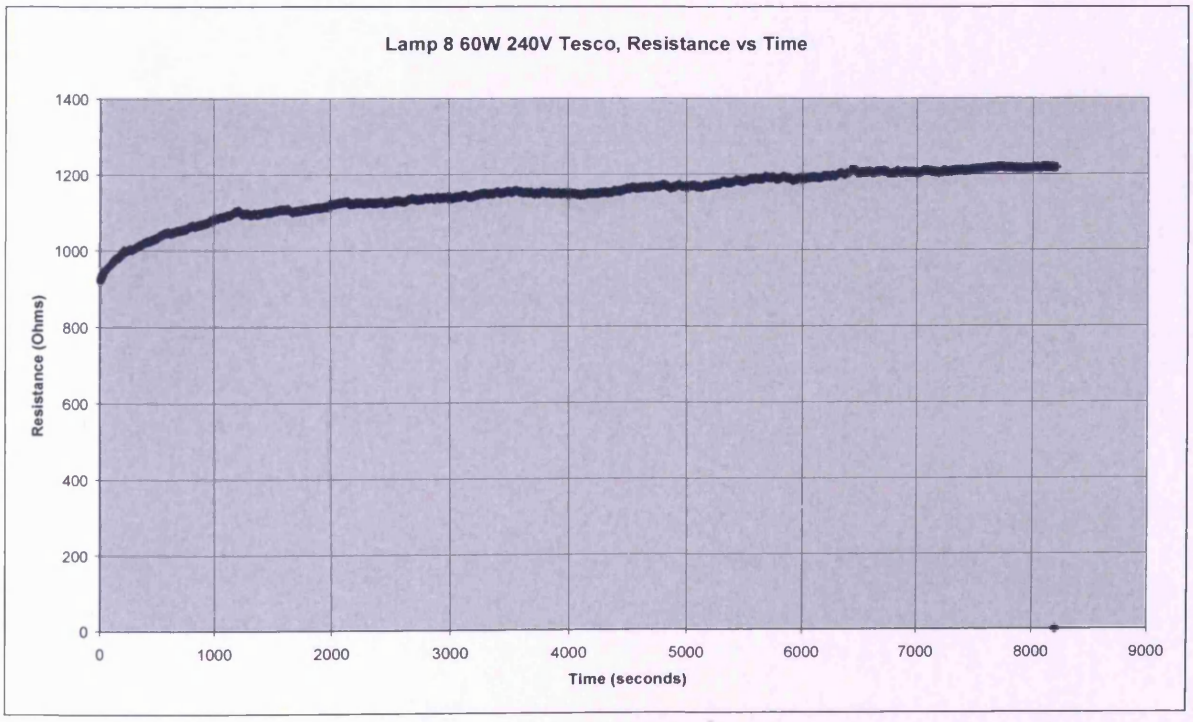
Lamp 6


Lamp 6 60W 240V Tesco, Resistance vs Time
Resistance (Ohms)
1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
Time (seconds)

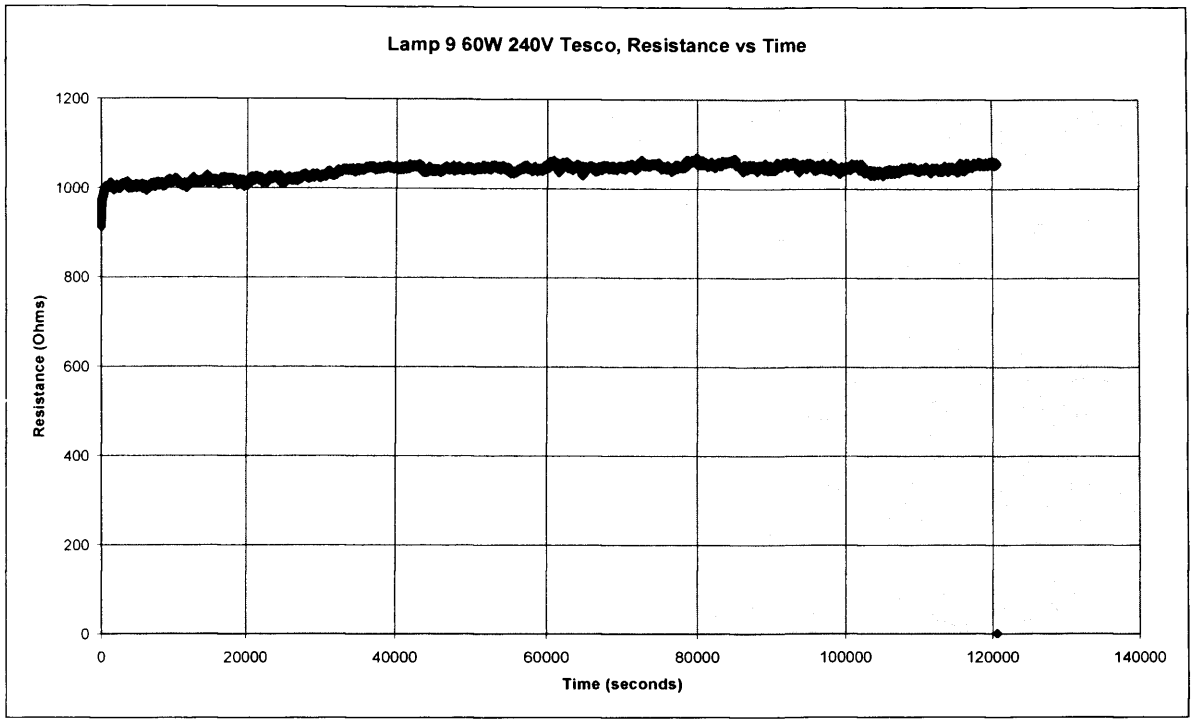
Lamp 7


Lamp 7 60W 240V Tesco, Resistance vs Time
Resistance (Ohms)
0
200
400
600
800
1000
1200
0
1000
2000
3000
4000
5000
6000
Time (seconds)

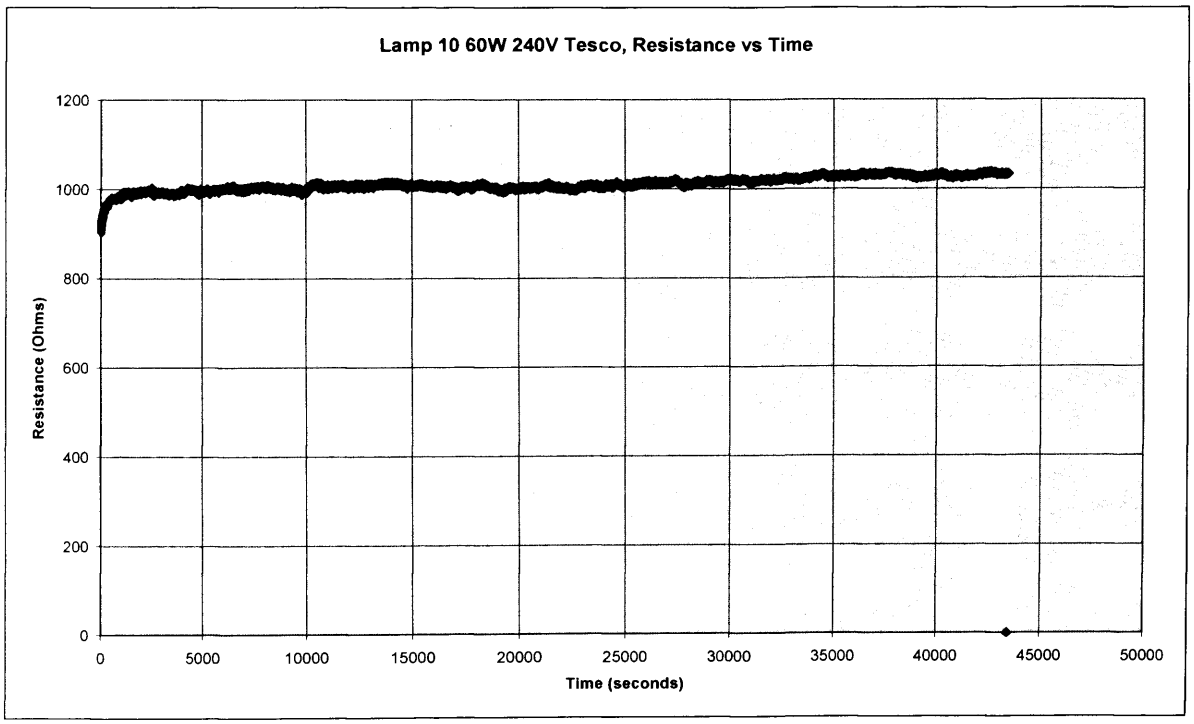
Lamp 8


Lamp 8 60W 240V Tesco, Resistance vs Time
Resistance (Ohms)
0
200
400
600
800
1000
1200
1400
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
Time (seconds)

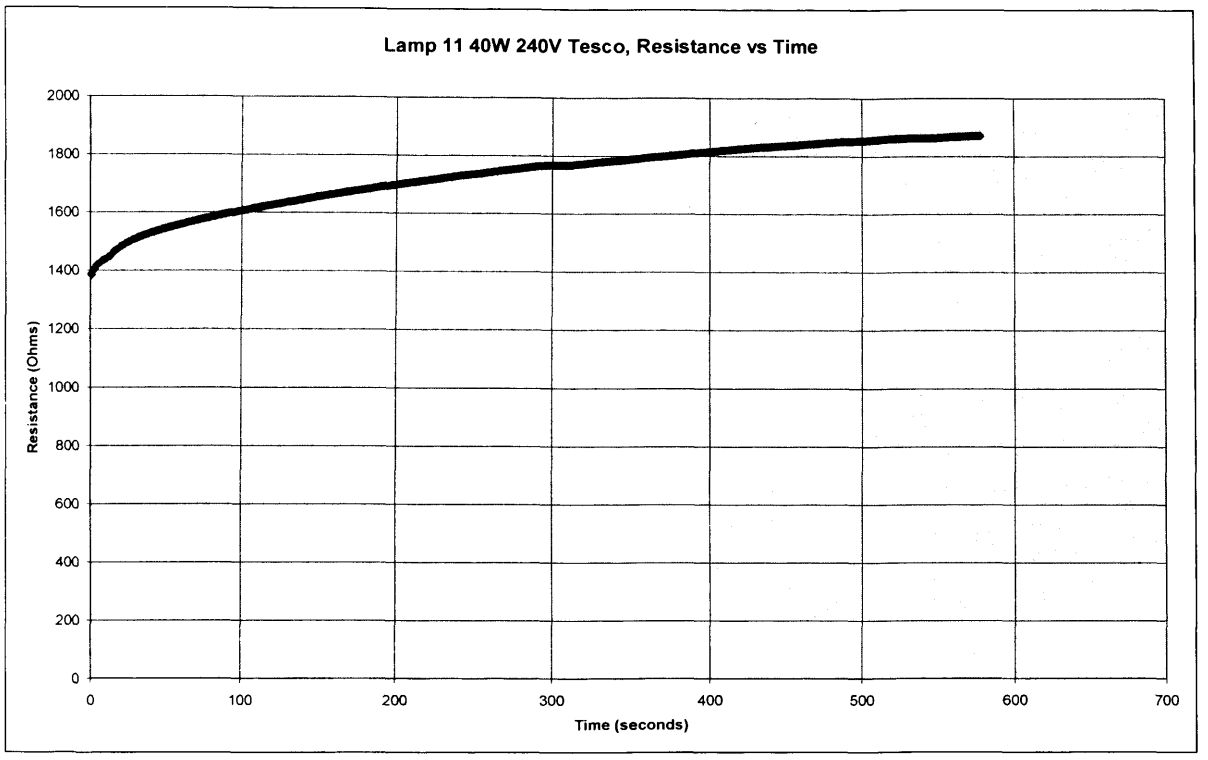
Lamp 9



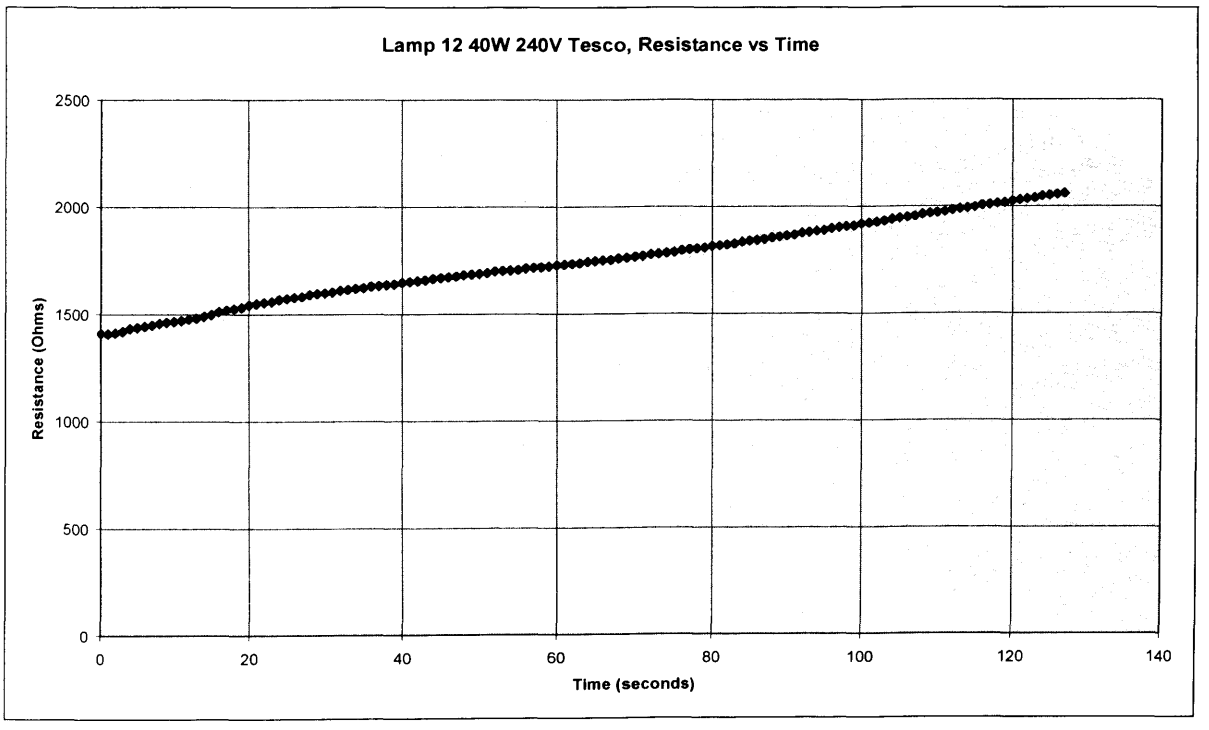
Lamp 10



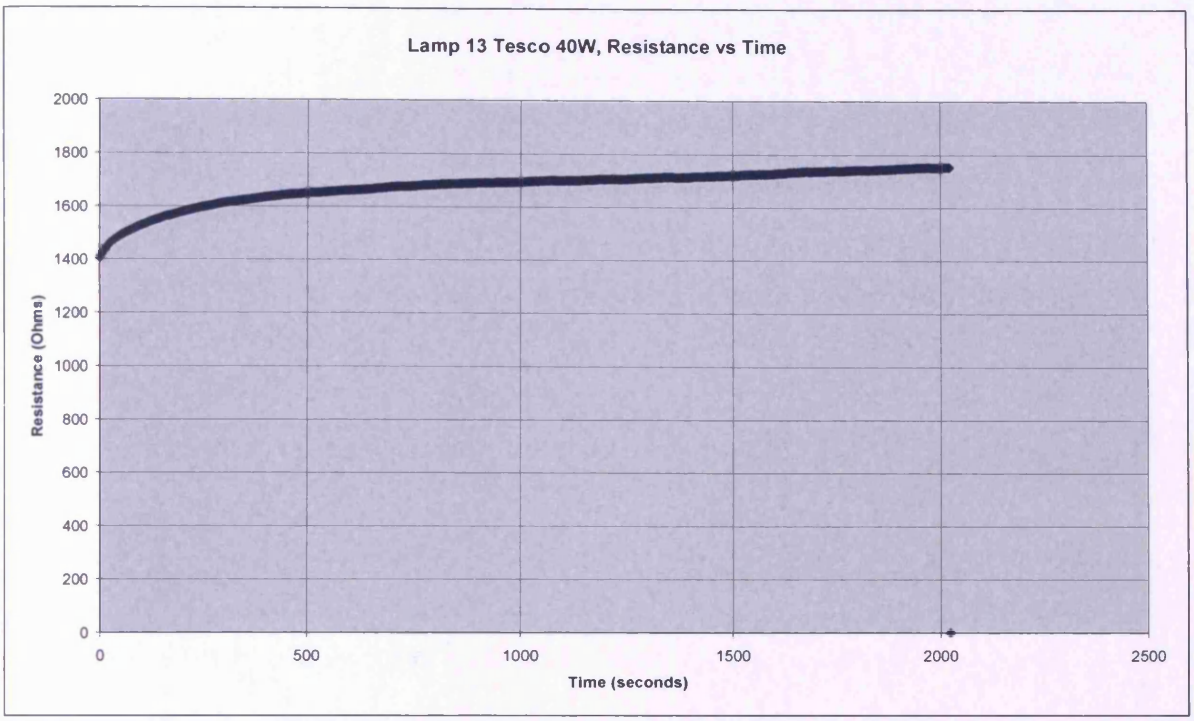
Lamp 11



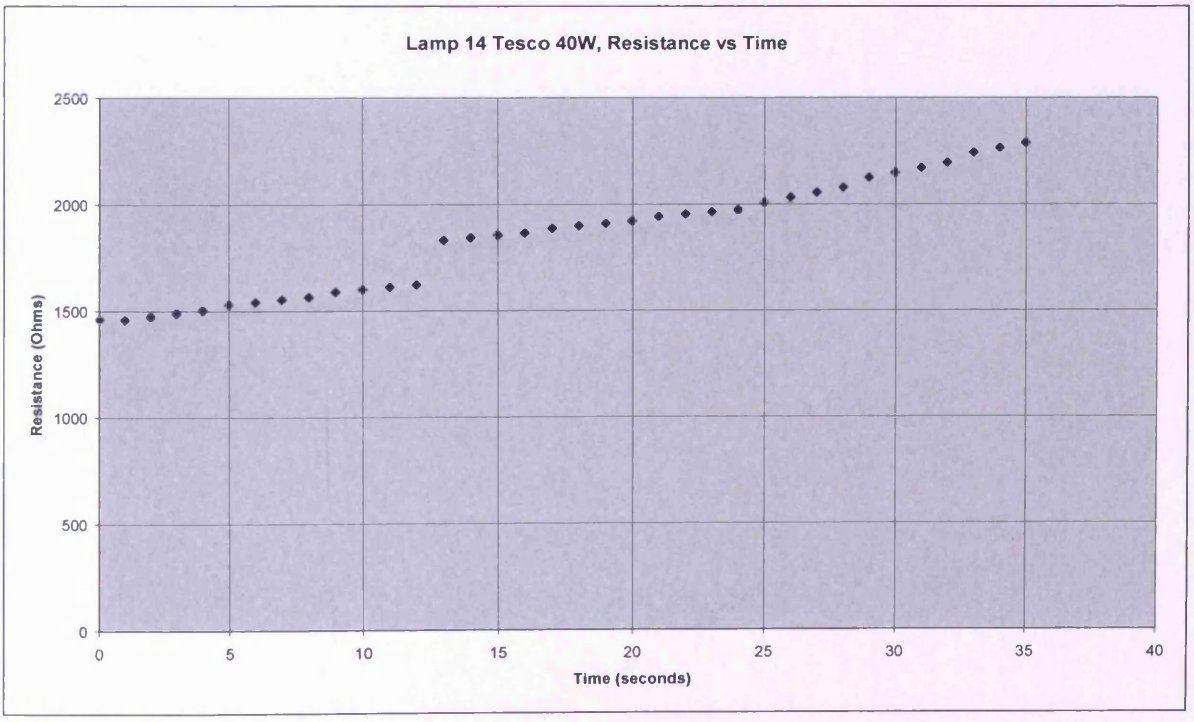
Lamp 12



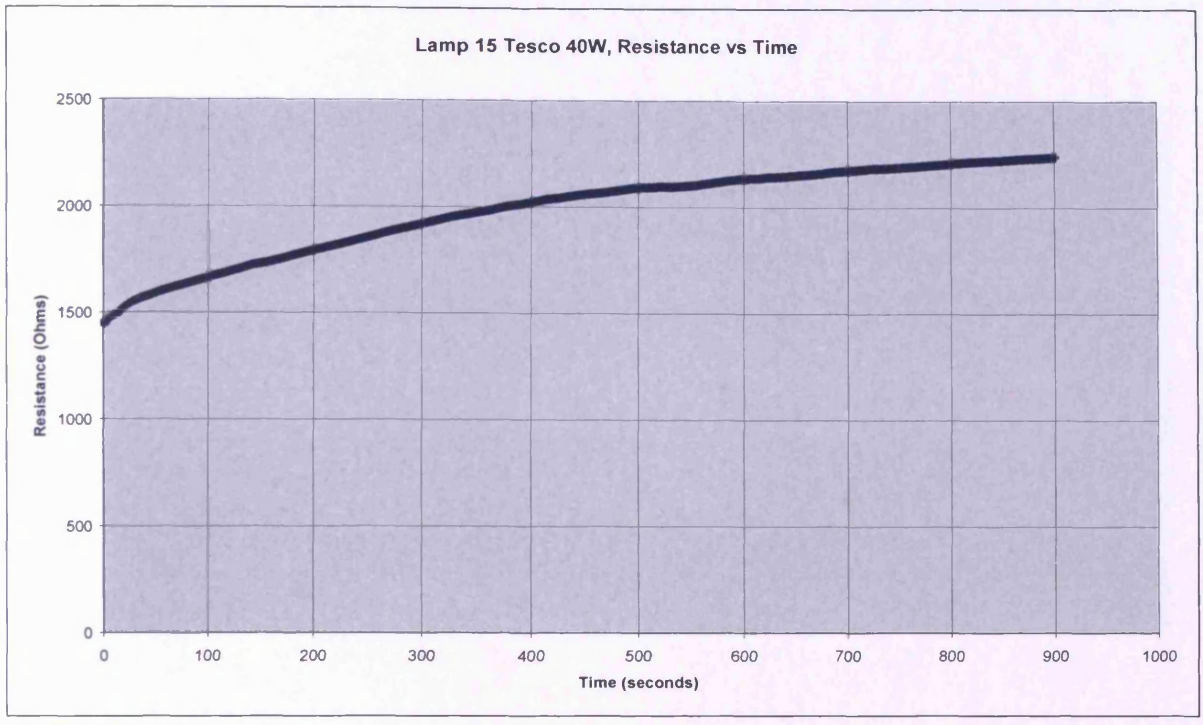
Lamp 13



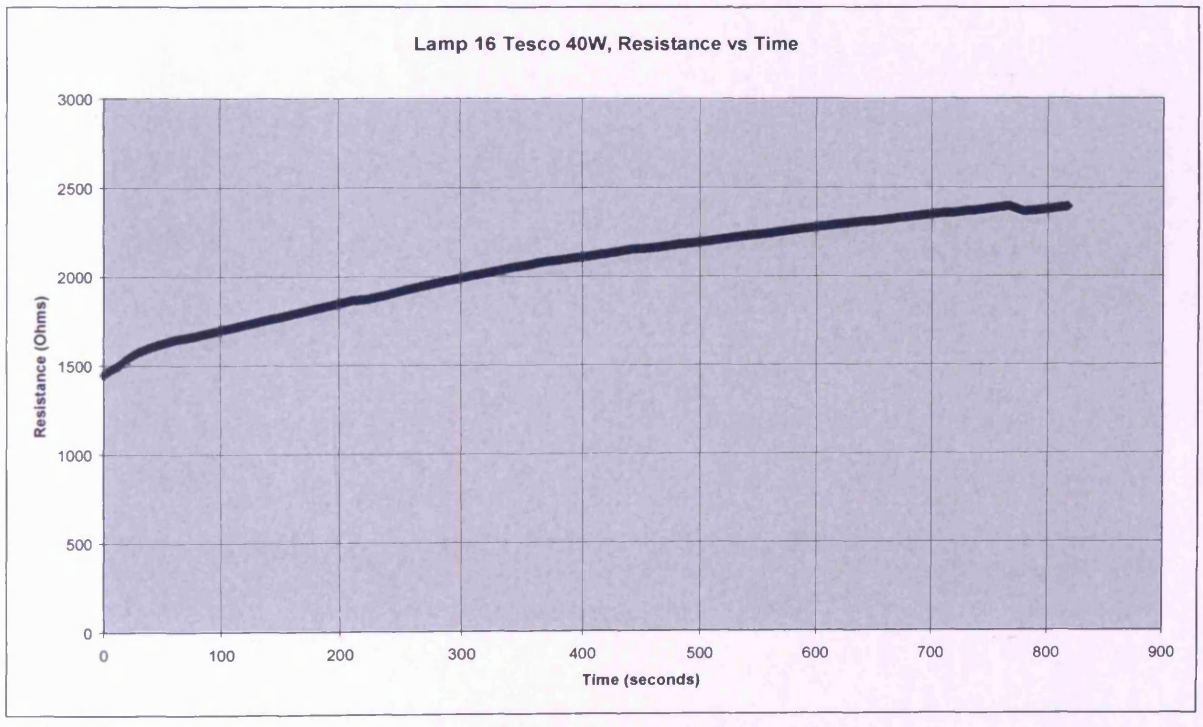
Lamp 14



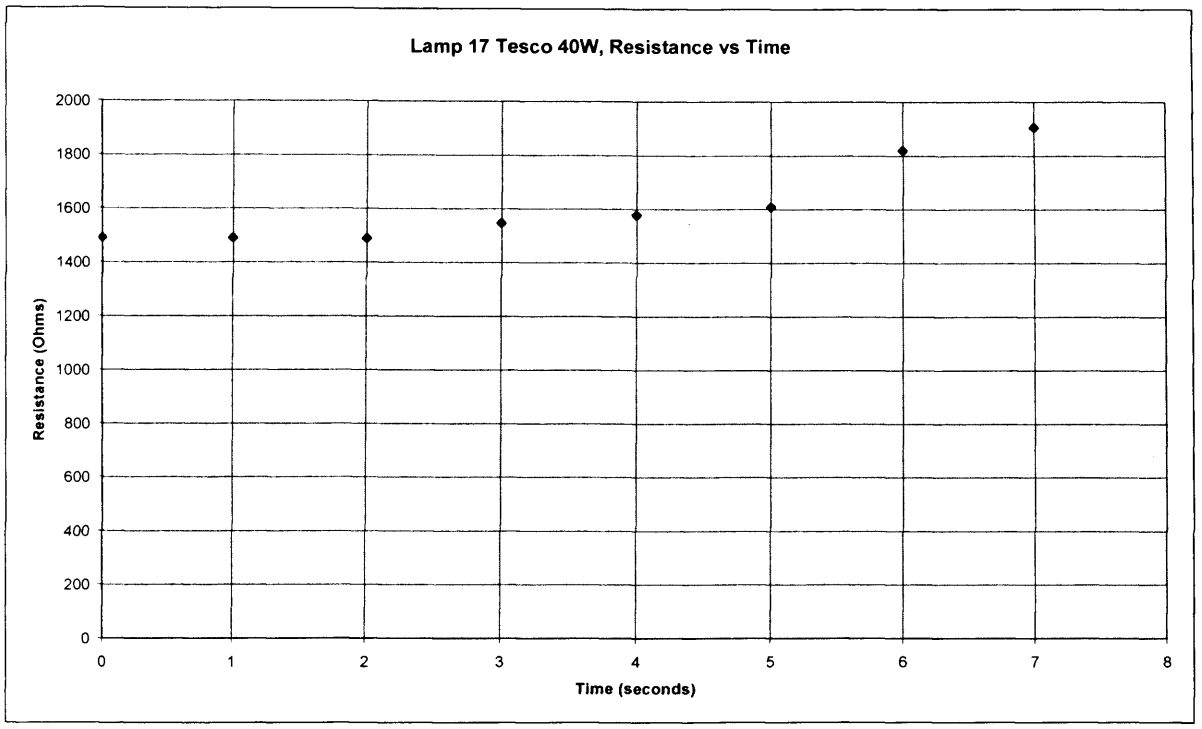
Lamp 15



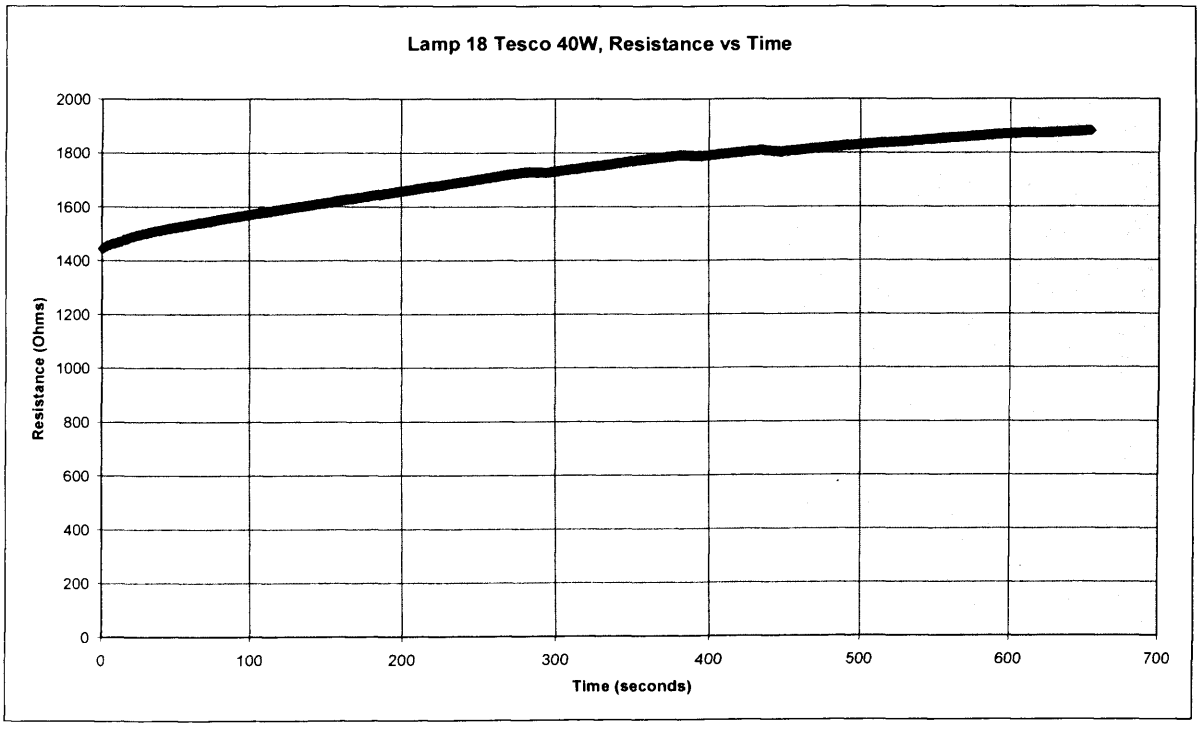
Lamp 16



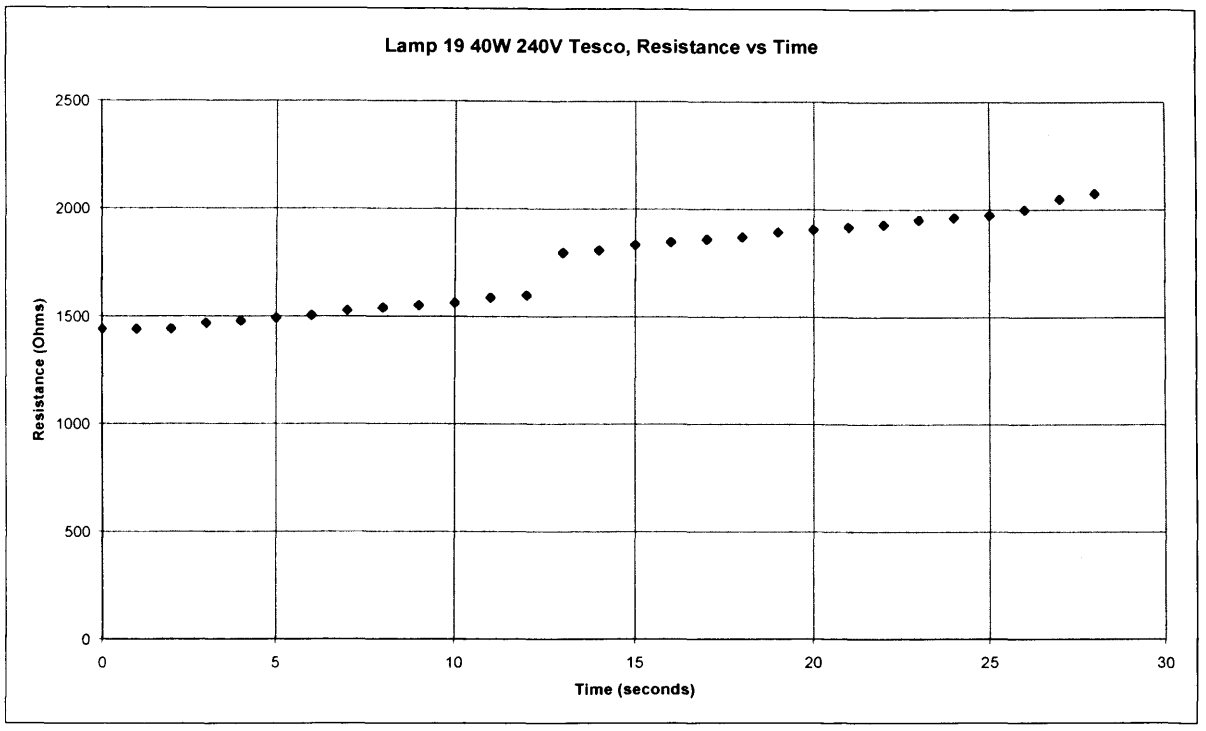
Lamp 17



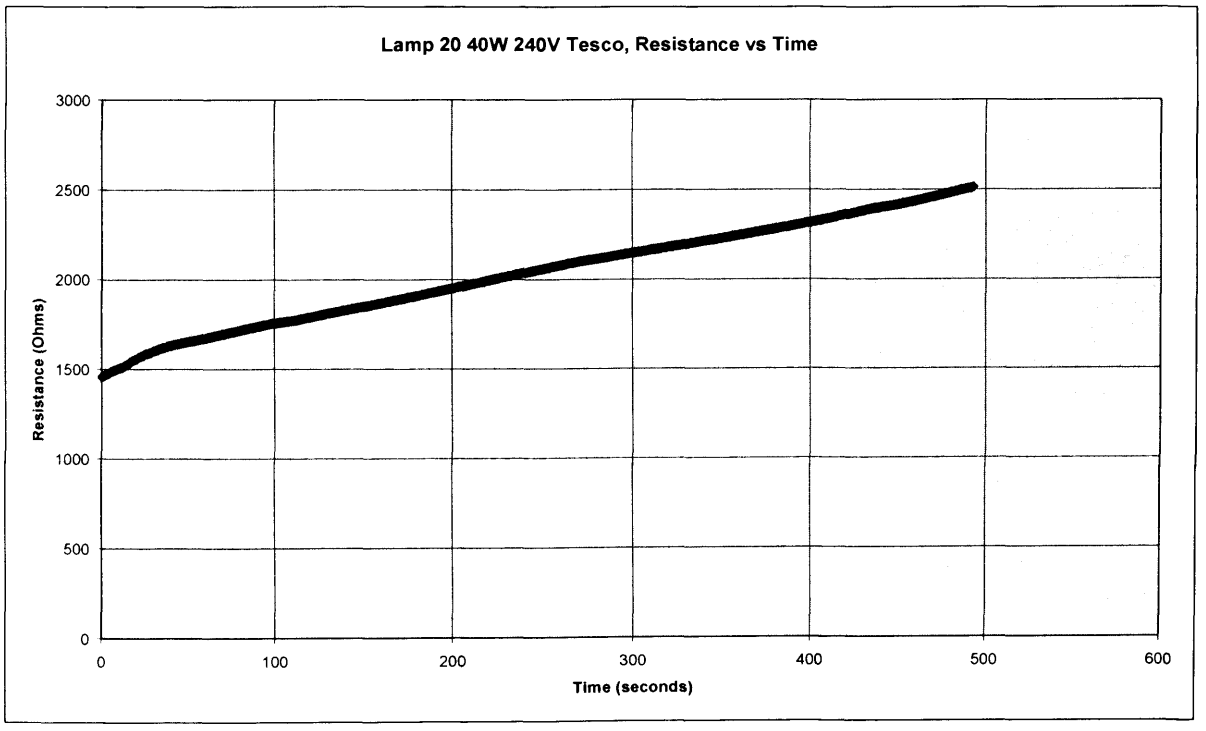
Lamp 18



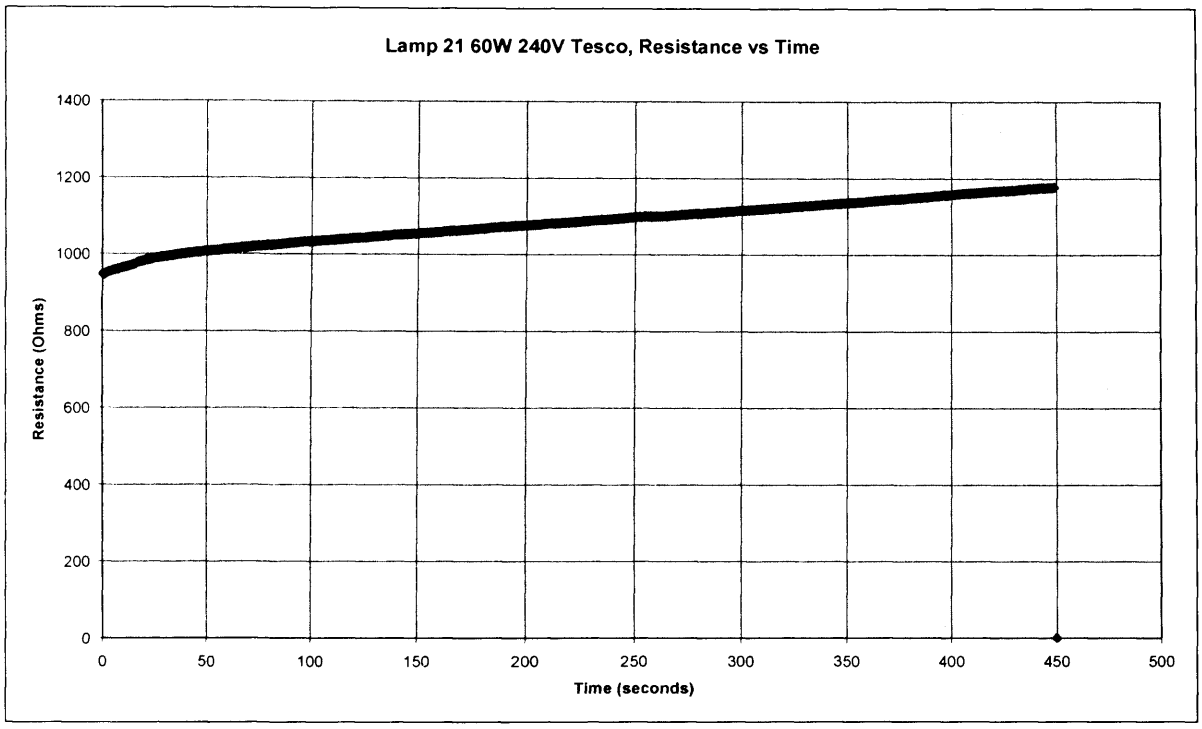
Lamp 19



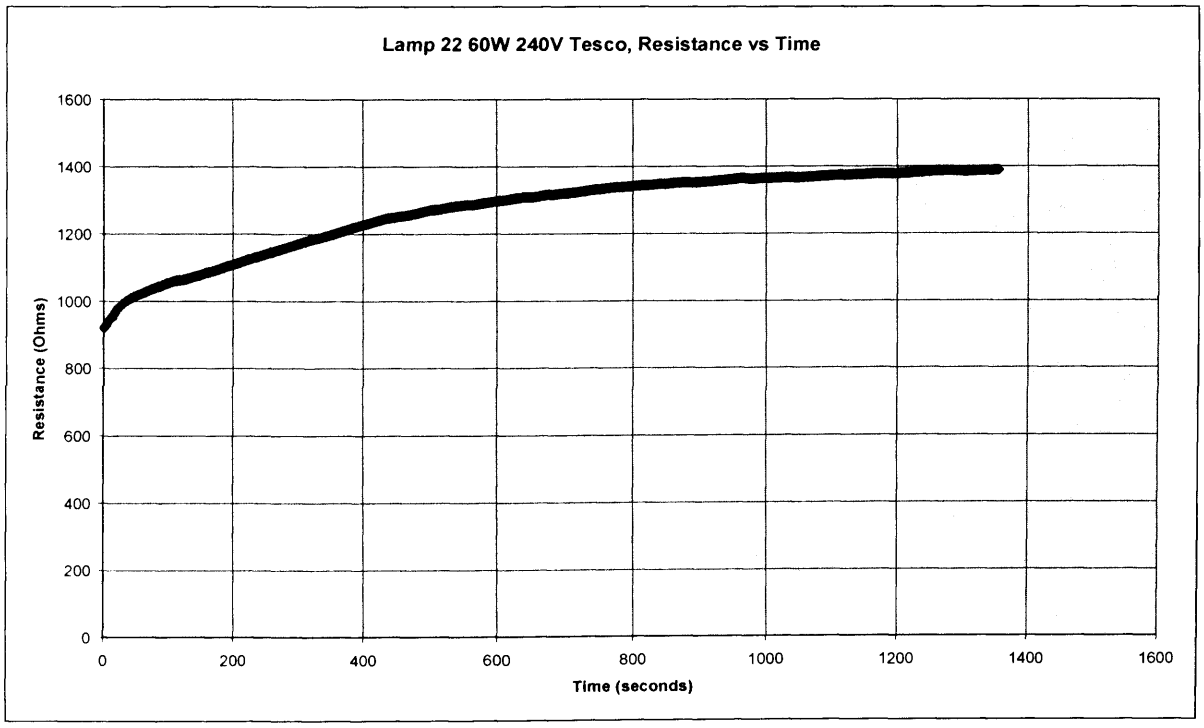
Lamp 20



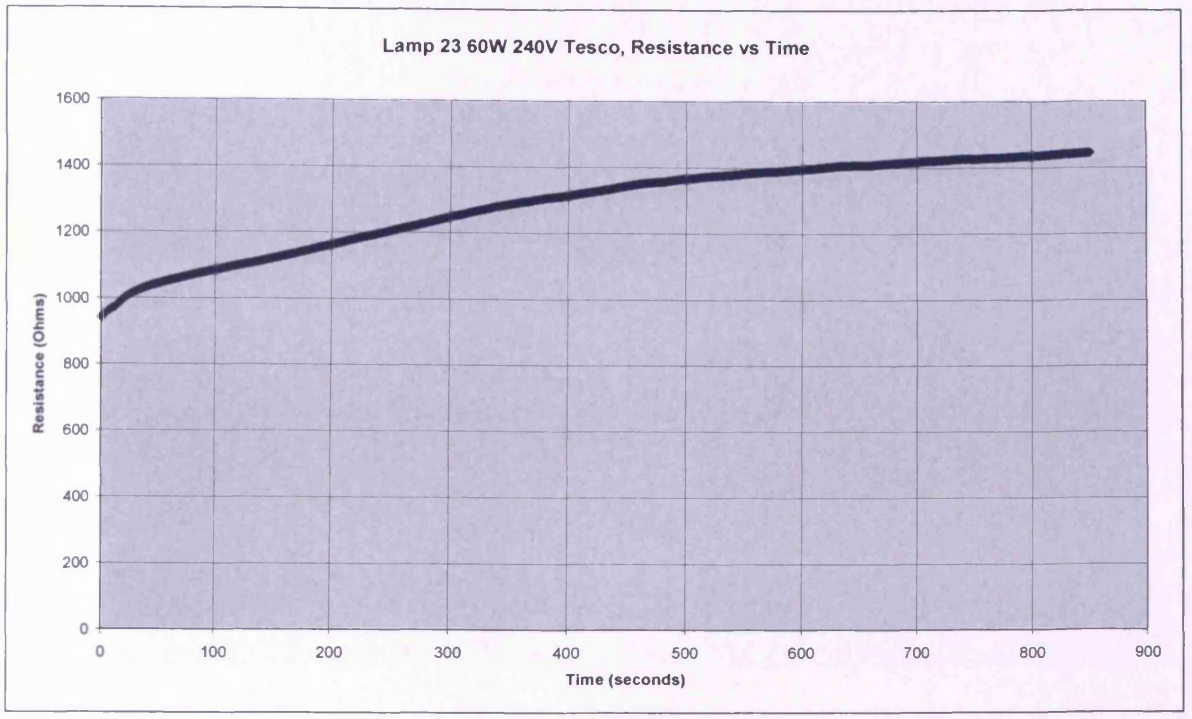
Lamp 21



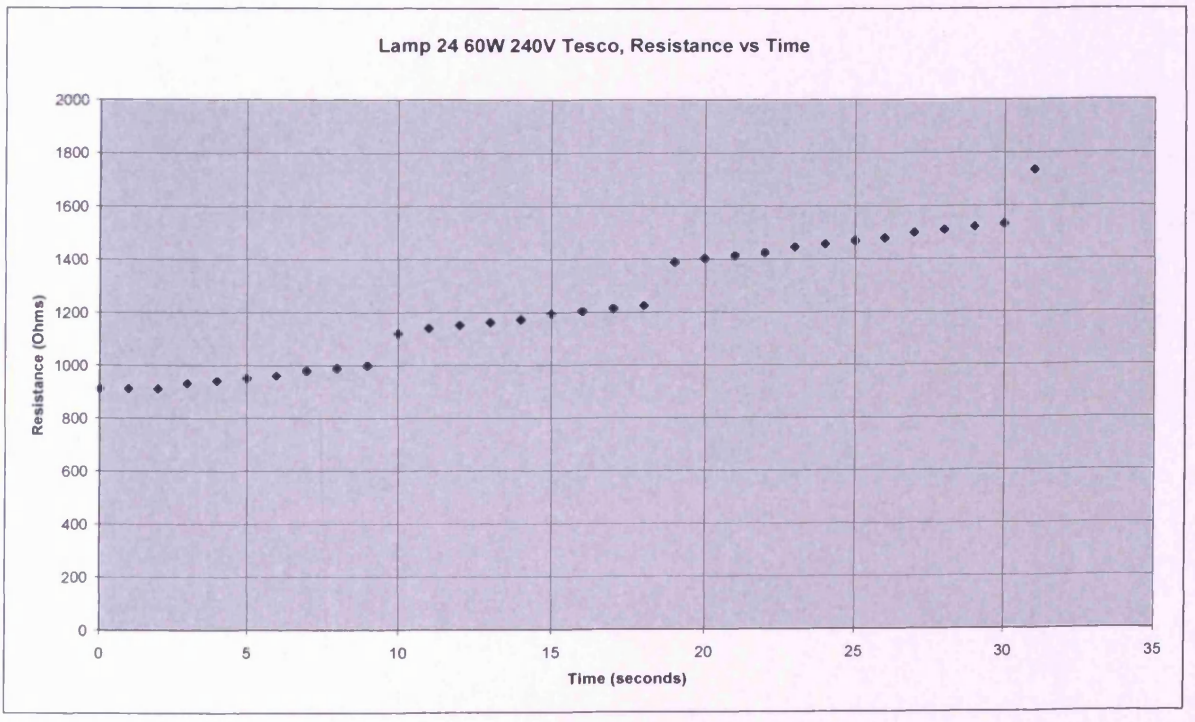
Lamp 22



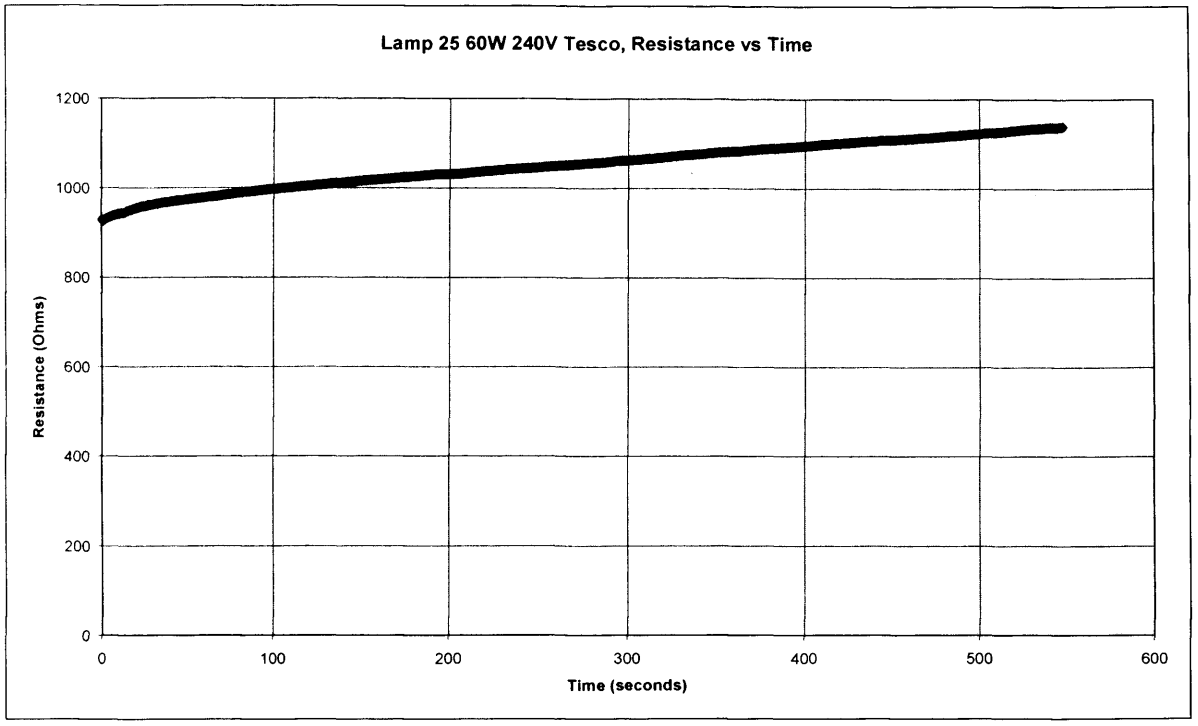
Lamp 23



Lamp 24

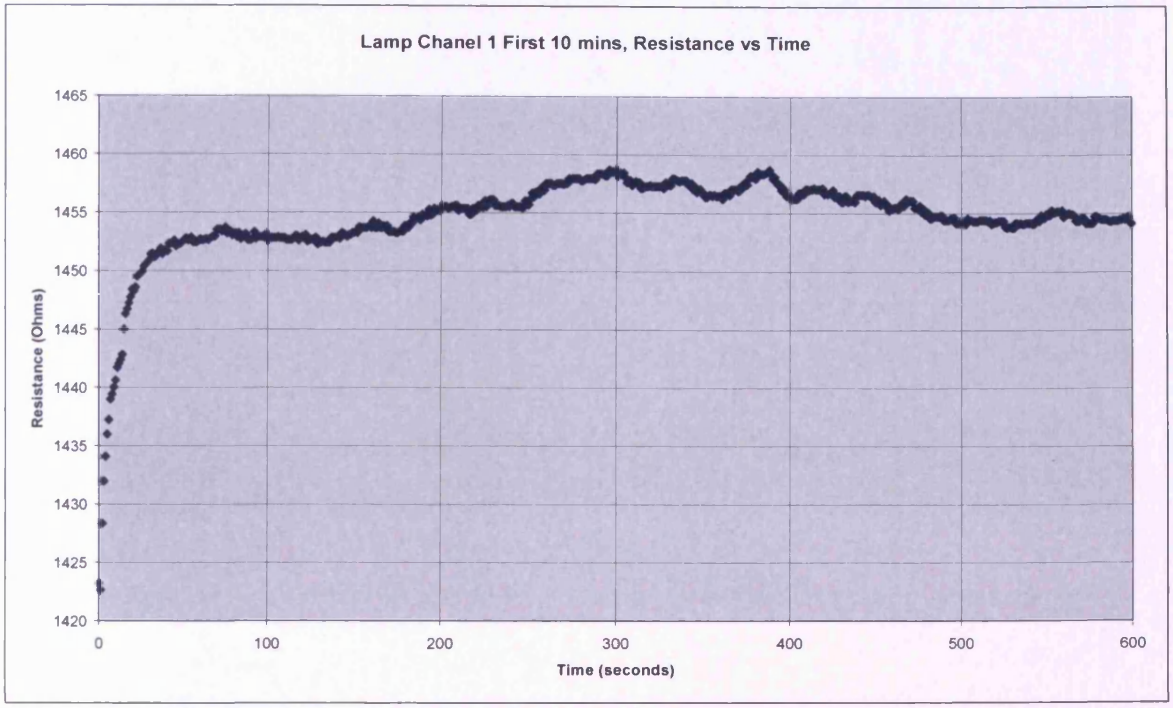


Lamp 25

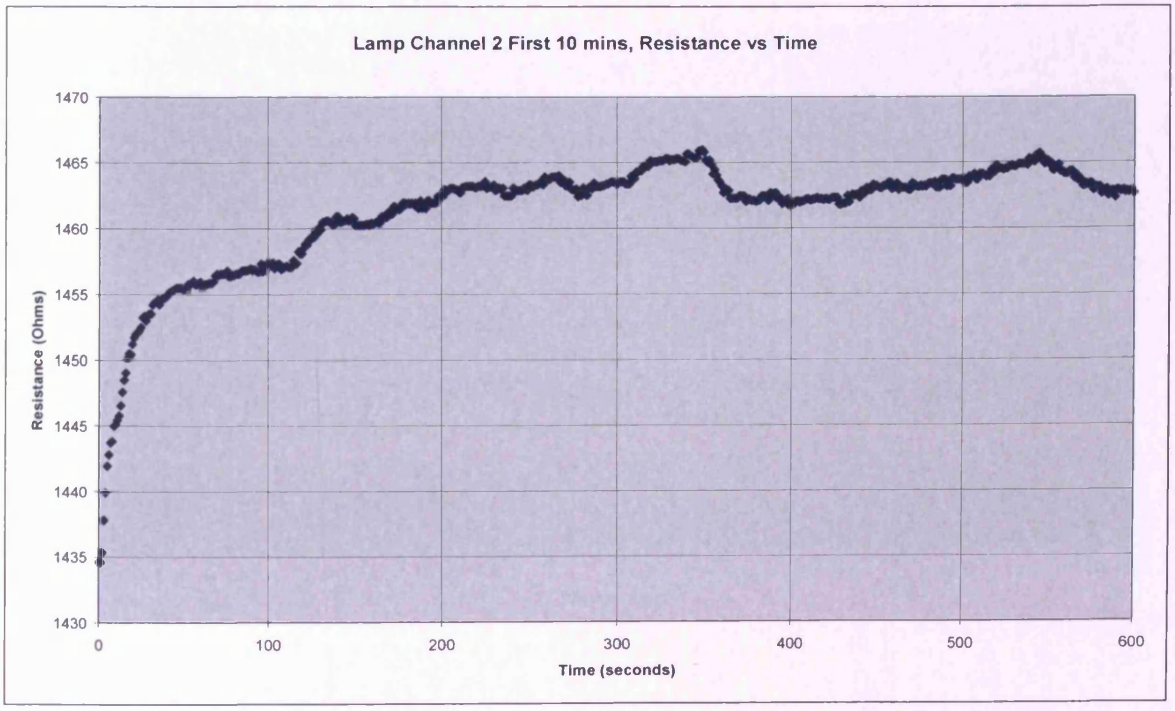

Lamp 25 60W 240V Tesco, Resistance vs Time
Resistance (Ohms)
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
Time (seconds)

Initial Resistance Increase of Unaltered Lamps 10 - 17

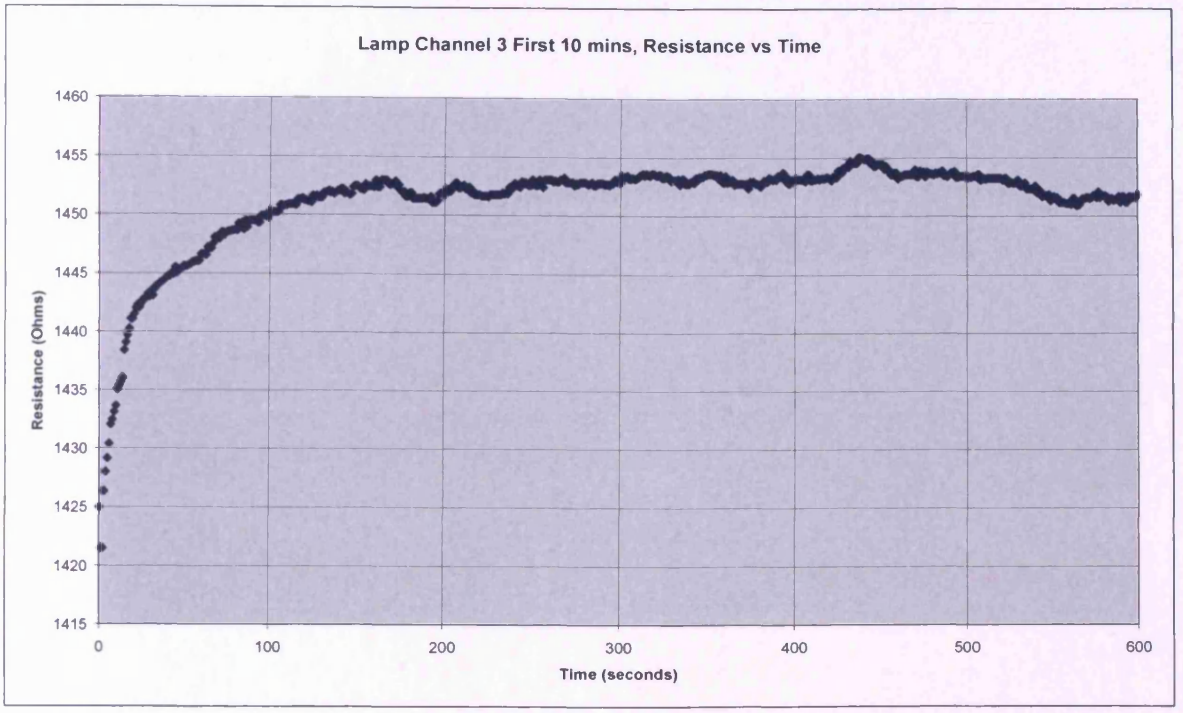
Lamp 10 (Channel 1)



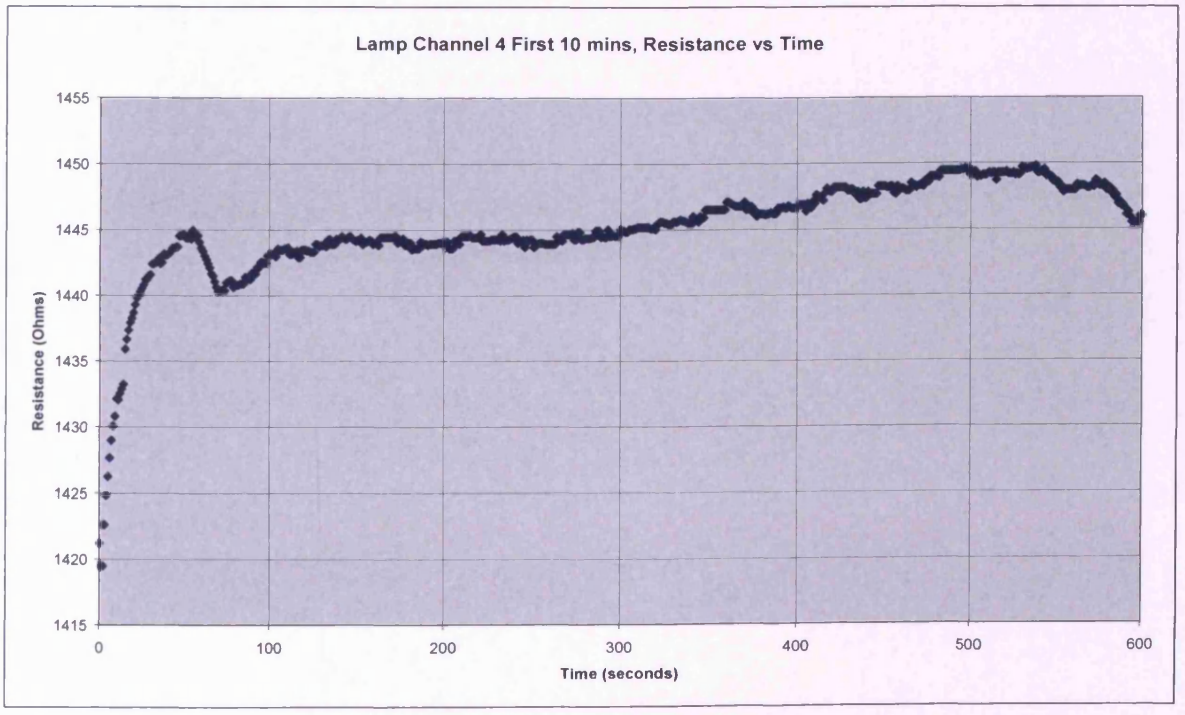
Lamp 11 (Channel 2)



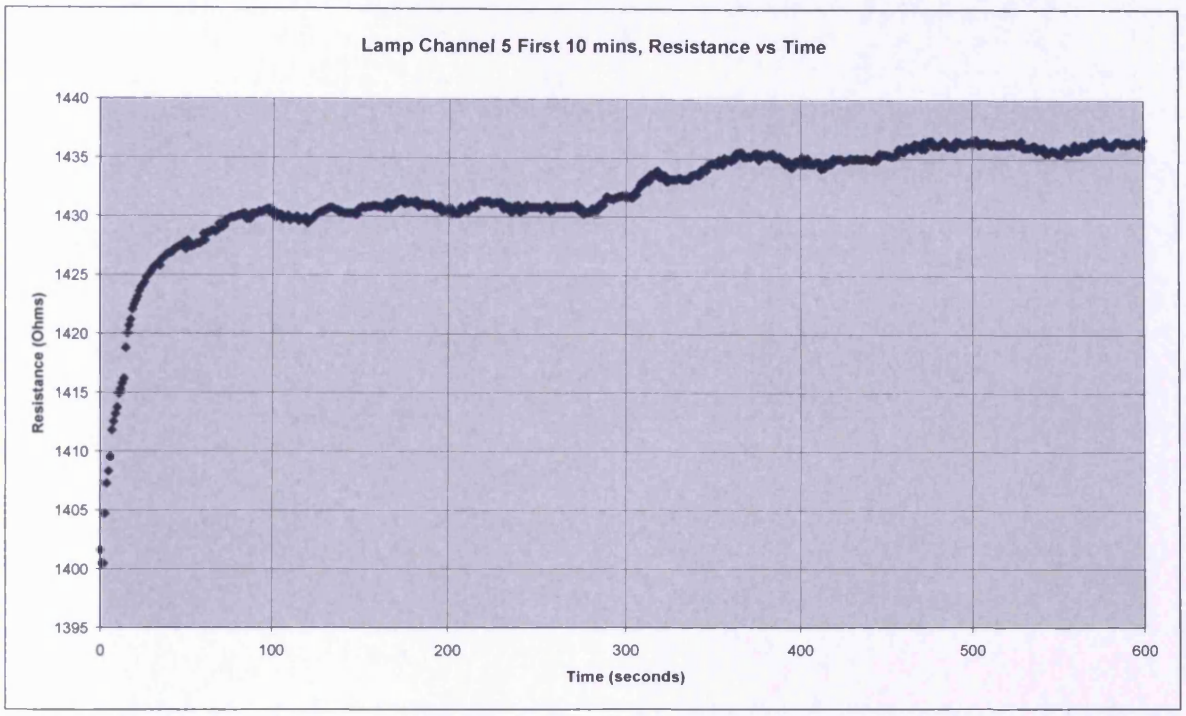
Lamp 12 (Channel 3)



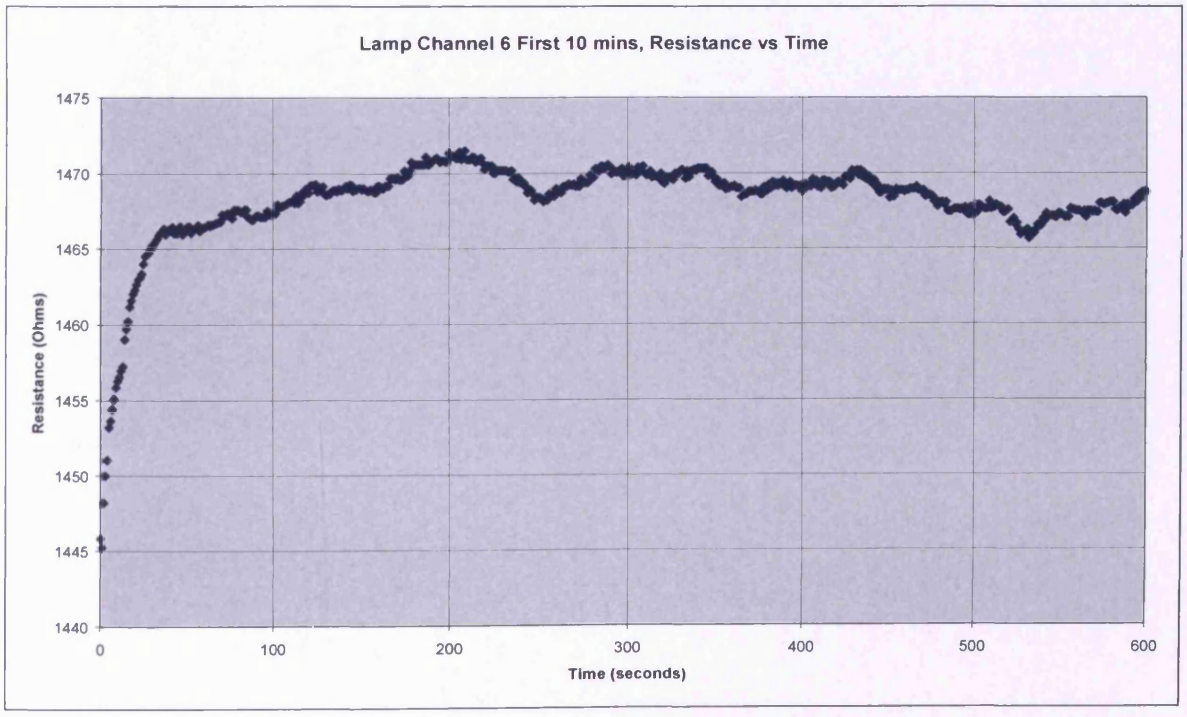
Lamp 13 (Channel 4)



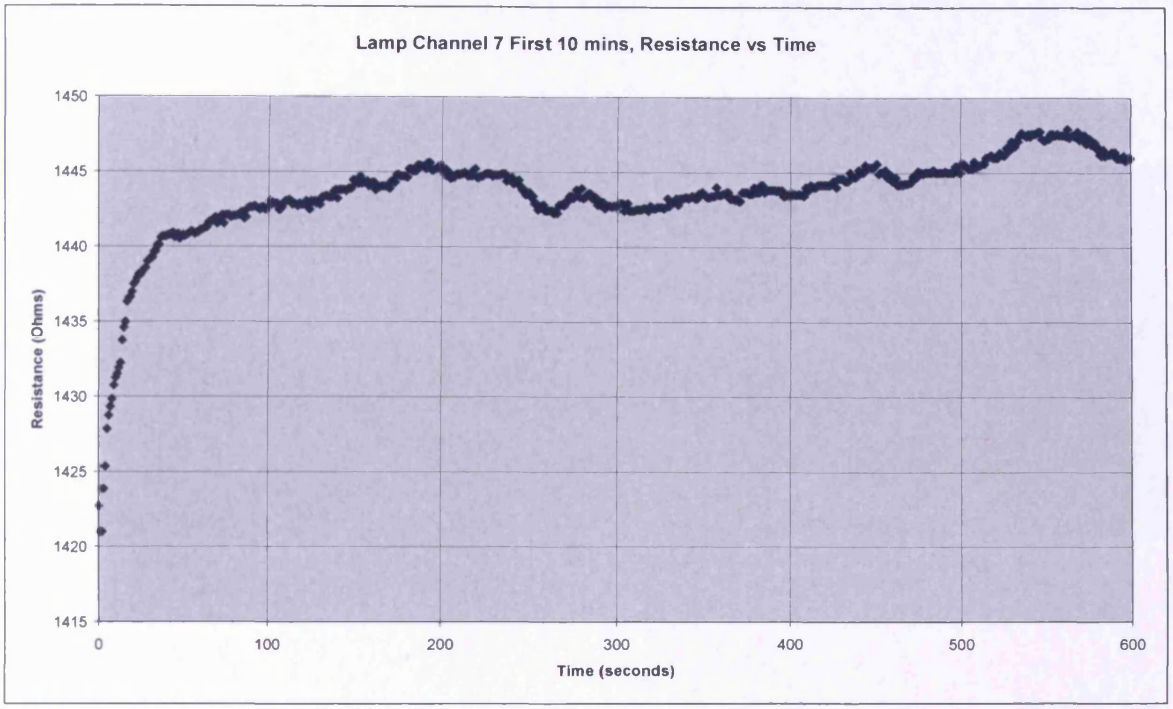
Lamp 14 (Channel 5)



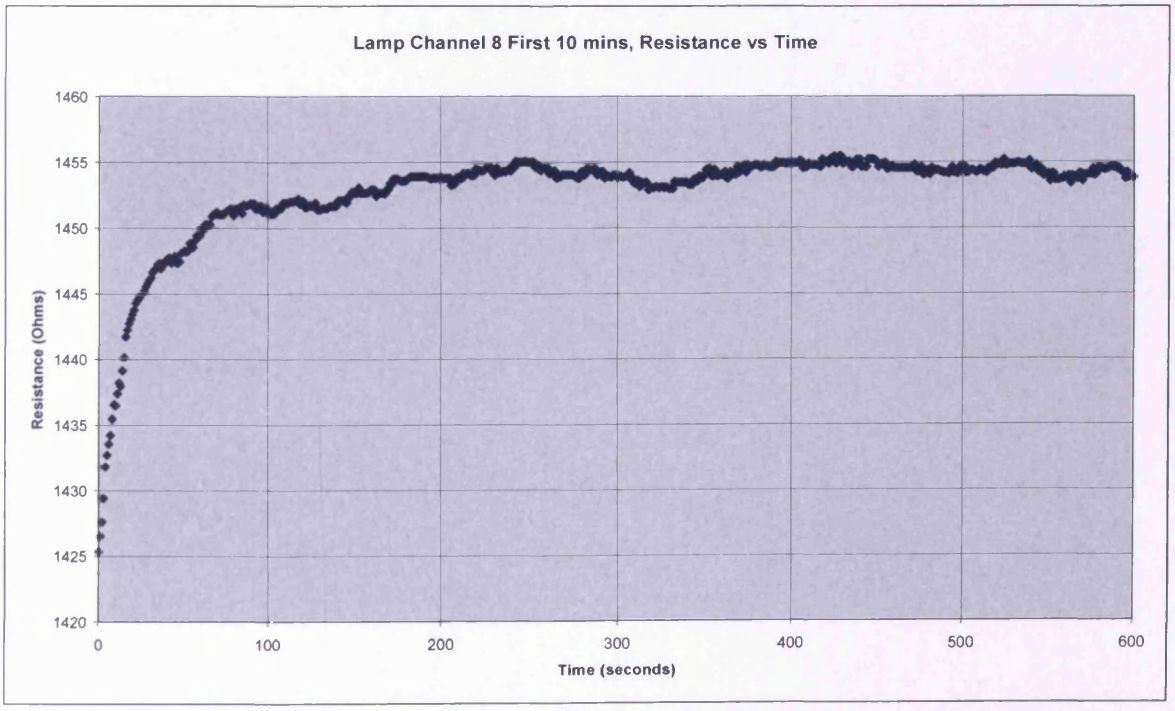
Lamp 15 (Channel 6)



Lamp 16 (Channel 7)



Lamp 17 (Channel 8)



D

Appendix D - UV Lamp Results

Aged Sample Data

Lamp No	Lamp Voltage (VAC)	Lamp Current (mA)	Lamp V/I (Ω)	Lamp Power (W)	UV (mW/cm2)	Lamp Voltage (VDC)
1	95.74	368	260.16	29.55	9.45	0.72
2	97.66	367	266.10	30.1	9.94	0.50
3	96.88	361	268.37	29.41	9.96	0.60
4	93.73	375	249.95	29.43	8.87	0.78
6	93.9	378	248.41	29.74	8.93	1.13
7	94.63	373	253.70	29.69	9.44	0.64
8	94.58	377	250.88	29.88	9.34	0.62
9	97.69	371	263.32	30.45	10.18	0.65
10	96.95	369	262.74	30.06	9.39	0.80
11	95	358	265.36	28.5	8.78	0.60
12	98.28	350	280.80	29.01	9.2	0.72
13	97.41	362	269.09	29.59	9.39	0.63
14	98.79	365	270.66	30.36	9.88	0.74
15	97.08	361	268.92	29.42	10	0.71

Filament Temperature Monitoring

Conversion Scale

Filament Voltage	Temperature
3.78	700
4.35	800
5.21	900
6.04	1000
6.93	1100
8.99	1200

Trend Line Equation

$$\text{Temp (°C)} = -1.239389154620x^3 + 10.574090956233x^2 + 119.882869143828x + 168.753050283471$$

Where x = Filament Voltage (V)

Results

	Cold Readings				Hot Readings							
	Left		Right		Left		Right		Left	Right	Average	UV
Lamp No	Volts AC	Temp °C	Volts AC	Temp °C	Volts AC	Temp °C	Volts AC	Temp °C	Diff °C	Diff °C	Diff °C	(mW/cm2)
1	4.47	805.21	4.34	786.90	5.22	906.38	5.37	925.52	101.17	138.63	119.90	9.45
2	4.34	786.90	4.26	775.53	5.30	916.64	5.39	928.04	129.74	152.51	141.13	9.94
3	4.51	810.81	4.32	784.06	5.13	894.70	4.99	876.27	83.90	92.21	88.05	9.96
4	4.73	841.22	4.87	860.22	5.42	931.81	5.72	968.50	90.60	108.28	99.44	8.87
6	4.74	842.58	4.91	865.59	5.36	924.26	5.65	960.10	81.68	94.51	88.10	8.93
7	4.94	869.61	4.94	869.61	5.49	940.53	5.50	941.77	70.93	72.16	71.55	9.44
8	4.50	809.41	4.67	832.99	5.44	934.31	5.42	931.81	124.90	98.82	111.86	9.34
9	4.47	805.21	4.16	761.23	5.35	923.00	5.12	893.40	117.78	132.17	124.97	10.18
10	4.52	812.20	4.42	798.19	5.25	910.24	5.29	915.37	98.04	117.17	107.61	9.39
11	4.81	852.11	4.77	846.67	5.24	908.96	5.37	925.52	56.85	78.85	67.85	8.78
12	4.20	766.96	4.43	799.60	4.91	865.59	5.19	902.51	98.63	102.91	100.77	9.20
13	4.21	768.39	4.37	791.14	5.12	893.40	5.19	902.51	125.00	111.36	118.18	9.39
14	4.52	812.20	4.53	813.60	5.80	977.97	5.30	916.64	165.76	103.04	134.40	9.88
15	4.48	806.61	4.27	776.96	5.15	897.31	5.03	881.57	90.70	104.61	97.65	10.00

Lamp Striking Voltages

Lamp	1st Strike Voltage	2nd Strike Voltage	3rd Strike Voltage	Max	Light
1	608	576	552	608	11.37
2	480	488	488	488	11.43
3	616	608	608	616	11.59
4	432	408	392	432	10.98
5					9.86
6	432	384	368	432	10.15
7	520	528	520	528	10.78
8	584	568	560	584	10.72
9	616	600	592	616	11.34
10	448	456	448	456	10.22
11	432	408	416	432	10.89
12	728	744	744	744	11.54
13	528	528	520	528	10.94
14	640	624	584	640	12.00
15	512	504	504	512	12.31

Accelerated Ageing

Weeks	Voltage (V)	Current (mA)	Power (W)	UV(mW/cm2)
0	101.05	356	30.44	12.31
1	101.37	357	30.38	12.35
2	99.31	361	30.61	11.14
3	98.60	354	29.32	10.23
4	96.90	354	28.81	9.65
5	96.50	347	28.33	8.30
6	96.80	345	28.05	8.90
7	95.90	346	27.87	8.50
8	95.60	343	27.54	8.10
9	96.27	340	27.69	8.12
10	97.40	335	27.41	7.20
11	98.38	329	27.39	5.85
12	101.16	326	27.91	5.97
13	188.96	201	12.52	2.76

E

Appendix E - Publications

A Review of Lamp Condition Monitoring Technologies

Published In

Lighting Journal, The Official Bi-monthly Publication of the Institution of Lighting Engineers. Volume 71/No 5

Publication Date

October 2006

Authors

Paul Edwards, Roger Grosvenor, Paul Prickett

A Review of Lamp Condition Monitoring Technologies

As remote monitoring systems for commercial and public lighting proliferate almost monthly, Paul Edwards, Roger Grosvenor and Paul Prickett survey the various systems and technologies on offer – and look at future monitoring needs

Paul Edwards M Eng (Hons) MIEE, Roger Grosvenor B Eng (Tech) M Eng PhD C Eng MInst MC and Paul Prickett B Eng (Tech) M Eng C Eng MIMechE, work at the Condition Monitoring Group in the School of Engineering, Cardiff University

Abstract

This paper considers the monitoring techniques for determining lamp condition and discusses some of the world-wide research currently under way on the monitoring of different lamp types. Having identified areas of lighting technology where monitoring technology is still inadequate, the paper discusses the work required to enable informed pre-emptive lamp replacement to become a routine part of maintenance.

1. Introduction

Nearly every aspect of modern day life involves the use of electric lamps. Lamps used to light homes, offices, traffic signals and roads are all essential to modern life. While they may use completely different technologies, they share one inevitable feature – they will all fail at some time or another. The time taken for each lamp to fail varies widely, from a few hours up to a few decades for some lamps, such as the recently launched range of induction lamps.

Street lighting presents a major potential growth area for lamp monitoring systems, due to the large number of lamps involved and their diverse locations (photo: Simon Rowe)

Depending on the task a lamp is performing, the consequences of failure can be surprisingly significant. For example, traffic lights failing unexpectedly can cause accidents and traffic congestion – and drinking water sterilisers which rely on UV lamps to kill harmful will not be effective if the lamp is failing or has failed. This could lead to illness or even death for anyone drinking the water.

It also makes good economic sense to monitor lamps effectively, where light fittings are in inaccessible, difficult-to-maintain areas, such as above a swimming pool or in the roof of a theatre. Knowing the condition of lamps in such places means maintenance can be scheduled for the most appropriate time, such as during a period when the pool is drained – or in the case of a theatre, when the lighting rig is being set up for a show.

Currently the most popular way of ensuring a light fitting's continued operation is to schedule routine replacement of the lamps at set periods. This is very costly and still does not guarantee one of the new lamps will not fail unexpectedly at an early stage. An alternative approach has been to provide a second lamp which is switched on when the first lamp fails. However this is not perfect, as it is adds cost and bulk to the lighting system. and there will always be a time delay in switch over. Depending on the type of lamp and control gear this can be significant.

Clearly, there is a growing requirement to be able to determine the remaining life expectancy of lamps used in everyday life. The Intelligent Process Monitoring & Management (IPMM) Centre based at Cardiff University has carried out a number of successful research projects during the last 20 years based on machine tool condition monitoring. During these projects, different monitoring techniques were explored and developed in collaboration with numerous industrial partners, with the aim of providing fast fault detection and informed maintenance scheduling.

In this latest research project, the IPMM Centre is considering the condition monitoring of lighting systems and how they may be improved to provide 'real time' detection of impending lamp failures. This could provide the end user with sufficient information to take immediate action if required – or to optimise scheduled maintenance when the imminent failure does not present an immediate problem.

2. Traditional Monitoring Techniques

The traditional method of measuring the remaining life of a lamp is to measure how long it has been running for, using a device such as an hour meter. However, this has a number of major drawbacks:

The initial life expectancy quoted by the manufacturer is inherently inaccurate. Some lamps may fail after a very

short period of time due to infant mortality, whilst others may last twice their expected life – this can cause major errors in the predicted replacement time of the lamp.

A lot of lamps are dimmed, which will significantly affect their life expectancy.

Environmental factors such as switching frequency are not accounted for.

Another well established method of lamp monitoring is to monitor a parameter of the lamp, such as the operating current, so that when the filament breaks or the discharge ceases, an alarm is triggered. However, this is useless for the purpose of pre-emptive maintenance.

3. Emerging Lamp Condition Monitoring Technology

An emerging approach to lamp monitoring, around which this paper is based, is to measure a characteristic, or combination of characteristics, of a lamp to accurately determine the remaining life. When perfected, this will provide a continuously updated estimate of the remaining life, based on its current usage patterns. This work is still in its infancy and progress with the technique and the enabling technologies varies greatly with lamp type.

3.1. Filament Lamps: Little, if any, work appears to have been published on condition monitoring for conventional filament or halogen lamps. Nearly all existing monitoring systems rely on detecting an interruption in current flow as a means of determining that a lamp has failed. These systems are often incorporated into high specification cars. There appear to be no dedicated systems or devices for performing predictive lamp failure monitoring.

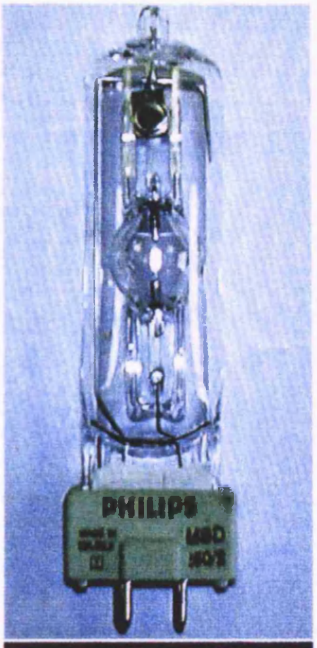
3.2. HID (High Intensity Discharge) Lamps: There has been a lot of interest in monitoring these types of lamps, primarily due to their use in expensive multi-media projectors and also due to the possibility of lamp explosion at failure.

In 1996, Osram Sylvania Inc. was granted a patent for a 'Ballast containing protection circuit for detecting rectification of arc discharge lamp'[1]. This device measures the DC voltage component, which develops across the lamp as it approaches the end of its life, and disables the inverter before the cathodes are overheated.

The Hewlett Packard Company was granted a patent in 2002 for an invention which 'provides an end-of-life notification signal, indicating that the arc lamp bulb should be replaced, close to the end of the useful life of the arc lamp bulb, but before the bulb actually burns out. The threshold property value is chosen to correspond to a point in the life of the arc lamp bulb that precedes the actual burn out of the bulb.' [2]

Koninkl Philips Electronics was granted a patent in 2003 for a device which monitors gas discharge lamps. 'A warning signal is generated when a test value is measured which corresponds to a clear reduction in the lamp voltage, for the purpose of better predicting the life of the lamps and the risk of explosion.' [3]

In 2004 a patent application by the BENQ Corporation was published, one of its claims being 'A projector capable of detecting remaining life time of the light source lamp therein,



Typical High Intensity Discharge Lamp

comprising: an image projection device having a light source lamp with a pair of lamp electrodes; a detection device for detecting a voltage across the lamp electrodes; an analogue-to-digital converter for converting the voltage to a digital value; and a control unit for comparing the digital value with a relational table to calculate the remaining life time of the lamp.'[4]

From all the patents found, it is clear that quite a lot of work has been done on detecting the imminent end of life of discharge lamps, particularly those used in projectors. However, the first three patents do not discuss determining the actual remaining lamp life at any particular time – they concentrate on setting thresholds based on an individual, or combination of, lamp parameters, in order to warn the user when the lamp is about to expire and shut it off before the lamp explodes. The final patent application

does discuss a device for measuring the remaining lamp life at any point in time, and displaying it each time at start up for the user to see, but it is clear that the primary focus of the work has been for projector lamps, which is primarily a stand-alone application

3.3. Fluorescent Lamps: In a white paper for CEYX Technologies[5], a control system for the cold cathode fluorescent lamps used in LCD backlighting is described that monitors the lamp current in order to enable the controller to maintain a consistent light level from the lamp. The controller also uses the lamp current to implement failure prediction, using built in diagnostics.

At present, there appears to be very little interest in condition monitoring conventional, heated cathode, fluorescent lamps. Unlike HID lamps, fluorescent lamps are inexpensive, as too are the fittings they're used in. Consequently the cost of any monitoring system has to be low in order to make it economically viable, unless the lamp happens to be part of a safety critical system where cost is then a secondary concern. One factor which helps make condition monitoring of fluorescent lamps a favourable option is the large number in use. Huge savings in both running costs and lamp costs could be achieved if they were changed at the right time. The same monitoring technology could also probably be used on compact fluorescent lamps.

3.4. SID (Static Induction Discharge) Lamps: There seems to be no information available on condition monitoring techniques for lamps of this type. This is presumably due to the fact

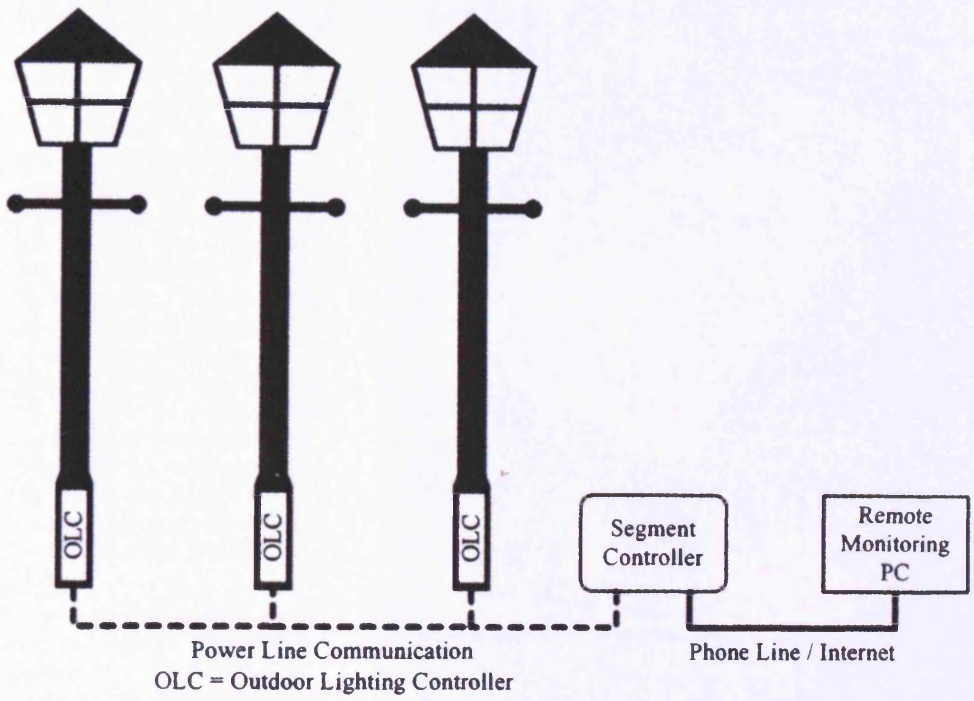


Figure 1: the Philips Telemanagement Architecture (see Section 4.1)

that they are, firstly, a very new technology with relatively few installed – and the fact that the lamps have a typical life of around 60,000 hours[6], which means obtaining data on the characteristics of the lamp over its lifetime would be very time consuming. As replacement of the lamp is so infrequent, it is possible the lamp may not even be the most frequently replaced part of the system. The electronic ballast used to drive the lamps could cause a significant percentage of failures itself, which would be very hard to predict. Consequently developing a monitoring system for this type of lamp may not turn out to be cost-effective until the lamp is in more widespread use and more data is available.

3.5. LED (Light Emitting Diode) Lamps: There seems to be little or no existing

information on condition monitoring techniques for LED lamps. Although LEDs are not new, their use as major light sources has only been facilitated by recent improvements in their output power; consequently the numbers in use are comparatively few, with most being used for specialist applications such as traffic lights and torches. The reason for there not being much interest in condition monitoring of LED lamps is almost certainly the same as for SID (Static Induction Discharge) lamps.

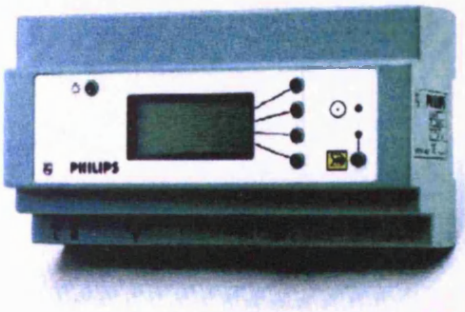
4. Emerging Lamp Monitoring Systems

A few advanced monitoring systems for outdoor (mainly street and road) lighting installations have recently become commercially available.

4.1. Philips Lighting: One of the most recent, and arguably most comprehensive systems, is one advertised by Philips in their *Telemanagement Solutions in Outdoor Lighting* brochure[7]. Philips currently supplies two systems, called Starsense and Telesense, which both allow greater control and monitoring of outdoor lighting installations than was previously possible.

Both systems revolve around each light being fitted with special electronic control gear which provides dimming control of each discharge lamp. The electronic ballast also monitors the discharge lamp's inner electrical parameters, to accurately predict when the lamp is about to fail, thus enabling efficient and timely replacement of the lamp. The systems are claimed to provide significant cost savings, by enabling lamps to be dimmed to the required light level depending on the time of day, weather, age of the lamps etc. They also remove the need to employ people to scout for failed lamps.

The Starsense system has comprehensive functionality, to allow it to interact with traffic management systems and geographical information systems, which makes it ideal for larger installations such as motorways and large interchanges. The Telesense system has a simpler architecture and is intended for stand-alone street lighting applications only (see Fig.1 on page 33). With both systems there is a SC (Segment Controller) for each group of lights – this communicates via the power lines with the OLC (Outdoor Luminaire Controller) installed in every luminaire. The OLC monitors the lamp and controls its dimming level under the command of the SC. The SC in turn communicates with the monitoring PC at the control centre via a phone line or the internet.



In Starsense, the lamp monitoring occurs in the OLC (Outdoor Lighting Controller) in each street light. The Segment Controller acts as a network component that manages the passing of information and also checks that the OLCs are healthy.

The Royce Thompson Cluster Controller scans the individual radio modules on each light to provide supervision of the street lighting system.

4.2. IEI (Integrated Equipment & Instrumentation): A very similar system is presented in a paper written by R. Seevaratnam of Integrated Equipment & Instrumentation entitled 'Street and Public Light Monitoring System (SPL)'[8]. The system incorporates a Basic Signalling Module (BSM) in each luminaire which communicates via the power line with a data logger module housed within the power distribution board; from here data is sent back to a central computer via a telephone line, fibre optic cable or wireless modem. The system enables remote switching of the luminaire, plus dimming if the ballast supports it. As well as control, the system also provides feedback on fault conditions, including lamp failure, inefficient lamp running and the door of the lamp column being open. A database on the central computer keeps track of the switching cycles of each lamp, together with any faults that have been detected. The detection of inefficient lamp operation also gives the user the option to replace lamps before they fail.

4.3. Mayrise Systems: One of the products on offer from Mayrise Systems is a software management system designed to help manage the operation and maintenance of street lighting installations. This system has also been integrated with remote monitoring systems from two different manufacturers to provide a complete management solution.

The first monitoring system, from Royce Thompson Ltd, uses RF units that replace the photocells in luminaries. 'The Oasis 2000 RF can dim electronic ballasts according to timetables or road conditions and also monitors street lighting to give desktop maintenance reports.'[9]. The system utilises a street level cluster controller which downloads timetables from the central computer and modifies the dimming levels of the lamps accordingly.

The second monitoring system, called 'Lightmaster', which is produced by Mayflower Intelligent Management Systems, also uses a RF module to replace photocells. This system works by using sub-master controllers to control a number of individual lights each, and then master controllers to control the sub-master controllers, which can control up to 32,000 nodes[10]. The master controllers can be linked to a central computer through radio, existing wide area networks or GSM. The Mayrise system is currently being implemented and used by a number of local authorities in the UK.

4.4 Harvard Engineering: The LeafNut system developed by Harvard Engineering is another wireless street lighting control system[11]. Each luminaire is fitted with a Harvard electronic dimming ballast which is connected to a wireless device called the LeafNode unit, which is designed to replace the photocell on top of the lantern. To control the LeafNode units, a BranchNode unit is required; this is a slightly larger device than the LeafNodes units, but is still able to be mounted on top of a lantern. The BranchNode communicates with the LeafNodes via an 868Mhz wireless network and can control up to 256 LeafNode units within a one kilometre range. A main server, called the TrunkNode, communicates with the BranchNode units by GSM mobile phone or by using a GPRS system. It also hosts the web interface, which allows control and monitoring of the system.

Up to eight different dimming profiles can be handled by the system to suit different locations, to enable lamps to operate at reduced light levels during the night, for example, when fewer cars are on the road. Should communication be lost with a LeafNode unit, it will continue switching and dimming the lamp using the same schedule as from the day before. As well as allowing centralised control using time profiles or a web-based solar clock control, the lamps can also be controlled by a photocell installed in every BranchNode.

The Leafnut system's Branchnode, which controls and monitors up to 256 street lights via wireless network

As well as controlling lamps, the LeafNut system also provides comprehensive

onitoring capabilities; it can onitor each lighting unit for fficiency, predicted lamp life, llast condition and energy nsumption. The system can en stop lamps cycling when y reach end of life, by tecting the cycling and then ming the lamp at 75% wer, until there is portunity for it to be placed. The LeafNut system believed, by the anufacturer, to be 'the only onitoring control system that ks directly to the micro-ocessor in the ballast and n therefore accurately edict lamp failure' [11].

5 ***Archnet Technology Ltd***: nother system by Archnet chnology Ltd.[12] operates a similar basis to the stems previously described sections 4.1 to 4.3, except at at the time of writing re is no sign of provision lamp failure prediction, ly detection. The system is rrently available for onitoring street and campus hting and features a device nnected to each lamp ich monitors its current w to detect when it fails. ch lamp device mmunicates via the power s to a power line modem, ich then conveys the rmation back to a central via either the internet or a PRS (General Packet Radio Service) modem. Information can also be sent to the lamp module to switch the lamp on or off as desired.

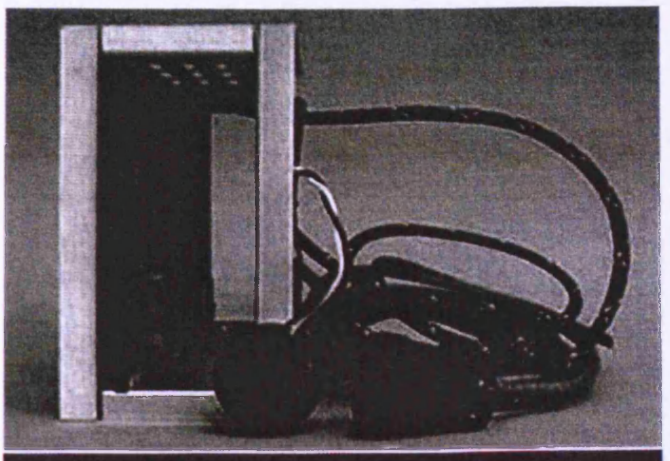
4.6 Fortran Traffic Systems Ltd: The Naztec Lamp Monitoring System by Fortran Traffic Systems Ltd.[13] uses precision current transformers to monitor the current drawn by each lamp in the traffic lights at a traffic intersection. By detecting the failure of any lamp quickly, replacement can be arranged before an accident occurs. Although the system can detect an intermittently failing lamp, it appears to have no predictive failure abilities.

5. Conclusions

Following consideration of the current state of monitoring of the different lamp types, together with a survey of the systems currently available on the market, it is clear that a lot of work has been done. However, most of the work has involved HID lamps and devices for incorporation into projectors, to warn the user of impending lamp failure and to protect them from exploding lamps. Very little work has been done on fluorescent lamps and filament lamps. This is not perhaps surprising, as the products which use HID lamps are usually expensive and hence can accommodate the extra financial burden of a monitoring system. However, the full benefit of lamp monitoring has yet to be realised, as HID lamps only make up a small fraction of UK lamp sales, as shown in Table 1.

amp type	Lamp sales (millions)	Percentage of total sales
lament (GLS)	323	74.7
uorescent tubes	54	12.5
alogen	26	6.0
ompact fluorescent	24	5.5
igh intensity discharge	5.5	1.3

ble 1: Lamp sales by UK Lighting Industry Federation members in 2002 [13]



The Naztec lamp monitoring system by Fortran

Looking at the table, it is clear that while a lot of work has been done on monitoring HID lamps, which make up around 1.3% of UK sales, this will have had very little impact on the overall lighting market. With this in mind further research needs to be carried out into condition monitoring of filament lamps (including halogen lamps), which make up an overwhelking 80.7% of the total UK market, and fluorescent lamps (including compact fluorescent lamps), which account for 18%. Once this has been achieved, work needs to be done to see how the information gathered from individual lamps can be processed and used to form a collective picture, for the end user, of the condition of an entire lighting system, containing many different types of lamp. The total cost and complexity of such a system must be kept to a minimum, in order that it may be commercially viable for use with the more common, lower cost types of everyday lamps.

6. References

[1] Yiyoung Sun (US) of Osram Sylvania Inc (US). Patent No. EP0696157, 'Ballast containing protection circuit for detecting rectification of arc discharge lamp'. Published July 2, 1996.

[2] Helbing Rene (US), Nishimura Ken A (US), Tullis Barclay J (US) of Hewlett Packard Co (US). Patent No. US6362573, 'Apparatus and method for monitoring the life of arc lamp bulbs', Published March 26, 2002.

[3] Klinkenberg Klaus (DE) of Koninkl Philips Electronics NV (NL), Patent No. US6534932, 'Method and arrangement for monitoring a gas discharge lamp', Published March 18, 2003.

[4] Wei-Chun Chang of BENQ Corporation, Patent Application Publication No. US20040032225, 'Method and apparatus for detecting remaining

lamp lifetime', Published Feb 19, 2004.

[5] CEYX Technologies Inc (Feb 6, 2003): 'White Paper for CEYX Technologies LCD Backlighting Control' [Online] http://www.ceyx.com/web/WhitepaperLCDBacklightingControl.pdf [April 5, 2005]

[6] Motorola Semiconductor Application Note AN1543 (1995): 'Electronic Lamp Ballast Design', Page 26 [Online]. http://www.wb-power.com/bbs/u/18/1087480661.pdf [April 5, 2005]

[7] Koninklijke Philips Electronics N.V. (No date): 'Keeping an Eye on your Future: Telemanagement Solutions in Outdoor Lighting by Philips' [Online] http://www.lighting.philips.com/gb_en/architect/luminaire/controls/tech_data/Outdoor/telemanagement_brochure.pdf [April 5, 2005]

[8] R.Seevaratnam of Integrated Equipment & Instrumentation (2001): 'Street and Public Light Monitoring System (SPL)'. In MonoSys Guide to Monitoring [Online Journal] Pages 12-16, 3rd Quarter edition, 2001. http://www.aguide.net/magazine/Q3-2001/pdf/Q3-2001.pdf [April 5, 2005]

[9] Royce Thompson Ltd. (2003): 'Oasis 2000 RF Radio Frequency Module' [Online] http://www.roycethompson.com [April 5, 2005]

[10] TEC Product Review (2003): 'FVD data ripe for Transport plans, Integrated Systems' [Online] http://www.itisholdings.com/pdf/fvd_ripe.pdf [April 5, 2005]

[11] J. McDonnell: 'Leafnut: Remote Dimming and Monitoring via the Web', *LJ* , May-June 2006, p. 28

[12] Archnet Technology Ltd (No date): 'Street Public Lighting Control and Monitoring' [Online]. http://www.archnetco.com/english/product/lamp.htm [April 5, 2005]

[13] Fortran Traffic Systems Ltd (No date): 'Naztec Lamp Monitoring System' [Online]. http://www.fortrantraffic.com/naztec/lampmon.htm [April 5, 2005]

[14] House of Commons Hansard Written Answers (June 20, 2003), Column 453W, 'Light Bulbs' – Estimated figures from the Lighting Industry Federation in 2002 [Online] http://www.publications.parliament.uk/pa/cm200203/cmhansrd/vo030620/text/30620w03.htm [April 5, 2005

A Method for Detecting Incorrectly Evacuated Filament Lamps

Accepted For Publication In

Lighting Research & Technology,

Published in association with The Society of Light and Lighting

Publication Date

Accepted for publication in August 2007, publication date TBC

Authors

Paul Edwards, Roger Grosvenor, Paul Prickett

Condition Monitoring Group
School of Engineering
Cardiff University
Newport Road
Cardiff
Wales
CF24 3AA

A Method for Detecting Incorrectly Evacuated Filament Lamps

P.M. Edwards MEng(Hons) MIEE
R.I. Grosvenor BEng(Tech) MEng PhD CEng MInstMC
P.W. Prickett BEng(Tech) MEng CEng MIMechE

Abstract

This research note describes a novel yet simple way to quickly detect incorrectly evacuated lamps from the electrical characteristics of the lamp when it is turned on for the very first time.

1. Introduction

Filament lamps still make up a large proportion of total lamp sales; in 2002 General Lighting Service (GLS) filament lamps made up 74.7% of total UK lamps sales whilst halogen lamps made up a further 6% of total sales[1]. This means that over 80% of lamps sold in the UK in 2002 used a filament as their main source of illumination.

During the current research project, which follows on from previous research on monitoring systems based on low-cost eight bit microcontrollers, the Intelligent Process Monitoring & Management (IPMM) centre, which is based at Cardiff University has been considering how informed maintenance scheduling for lighting systems can be achieved through the detection or prediction of impending lamp failures before they occur. The latest part of this work has revolved around trying to pre-empt lamp failures by detecting flawed lamps before they are used. The main focus of the work so far has been looking at how incorrectly gassed lamps can be detected before they are put into use.

2. Detecting Incorrectly Filled Lamps

During a typical gassing, "the bulb is evacuated through the exhaust tube and filled with nitrogen/argon gas. The bulb is partially re-evacuated and the lower end of the tube heat-sealed."[2] As with any process like this not all the oxygen will be removed from the lamp, so the filament is coated with "red phosphorus, which is very effective in removing traces of oxygen and moisture from the inert gases commonly used in lamps."[3] The red phosphorous on the filament is activated during the first lighting of the lamp, and along with the recrystallisation of the tungsten wire during first use, contributes to the electrical phenomena shown in figure 1, which shows the filament resistance of a normal GLS lamp during its very first and then second start up.

The small fluctuations in the steady state resistance value are due to small fluctuations in the mains supply voltage which was also recorded. The measurements were taken using a Voltech PM100 single phase power analyser interfaced to a computer which recorded all the measurements using custom written C++ software.

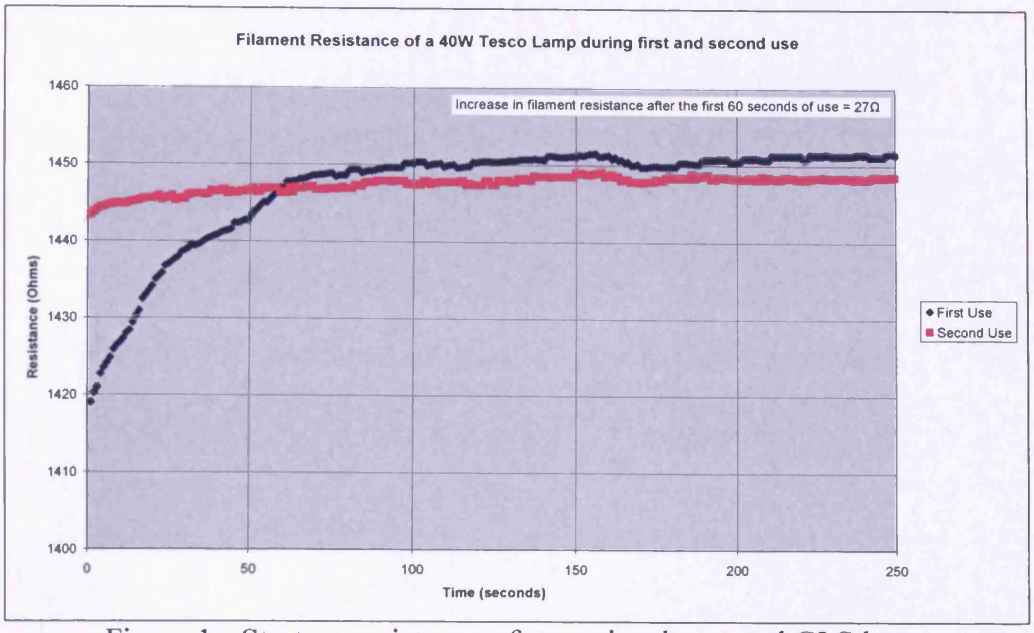


Figure 1:- Start up resistance of a previously unused GLS lamp

If excess air is left inside when the lamp is made, then depending on how much is inside, the lamp may well survive the initial filament burn as the oxygen inside the lamp is used up by the action of the getter and the burning filament, however this will mean that the filament is weakened and will probably fail prematurely when the lamp is put into use. A solution for detecting this problem has been the main outcome of the work so far and is discussed in the next section.

3. A New Approach

In order to conduct experiments and measure the start up current of incorrectly gassed lamps, a supply of faulty lamps was required. Since identifying faulty lamps would require testing every lamp and waiting to see if it was a faulty one before using the results, this was ruled out as a way of obtaining results in a reasonable time scale. Instead brand new bulbs were purchased and a small 1mm hole was drilled in the neck of the lamp using a tungsten carbide drill. This meant that air quickly filled the lamp, replacing the previous contents. As it proved very difficult to evacuate the lamp and then successfully seal the hole again it was decided to adopt a different approach. Being an inert gas, Argon will not react with the hot tungsten filament, therefore the lamp was filled with Argon welding gas plus a small amount of air which remained inside. The exact quantities of the gas inside the lamp did not matter as this would not be known when testing a real lamp either. The hole was then finally sealed with epoxy resin glue.

Each of the faulty lamps was then subjected to the same test as the healthy lamp shown in figure 1, except this time they were left on until they burned out. A plot of one of the faulty lamps is shown in figure 2. Note the change of the Y axis scale, which is required due to the significantly larger increase in the resistance of the faulty lamp.

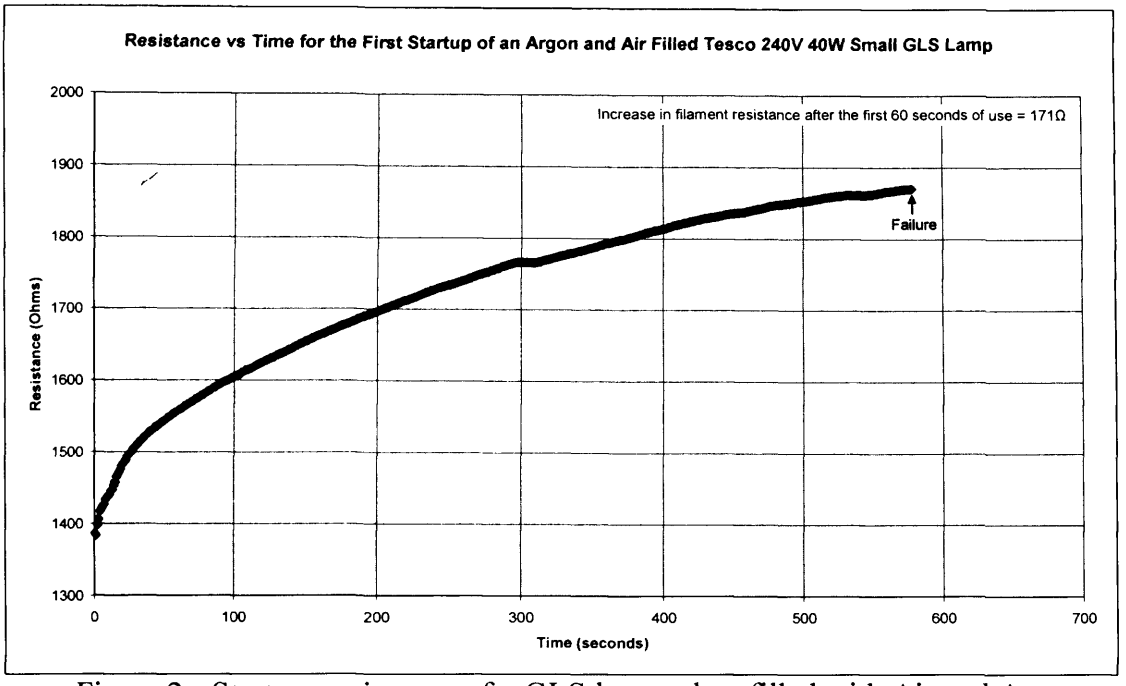


Figure 2:- Start up resistance of a GLS lamp when filled with Air and Argon

Looking at figure 2 it can be seen that the increase in resistance during the first 60 seconds of being on is significantly larger than for the unaltered lamp. The particular lamp shown in the plot lasted less than 10 minutes, however a lot lasted longer than this and some lamps burned for a couple of days.

In all a total of 13 40W lamps were filled and successfully tested. Figure 3 shows a plot of the increase in resistance for the lamps after being on for 30 seconds against how long the lamps lasted when they were left on continuously.

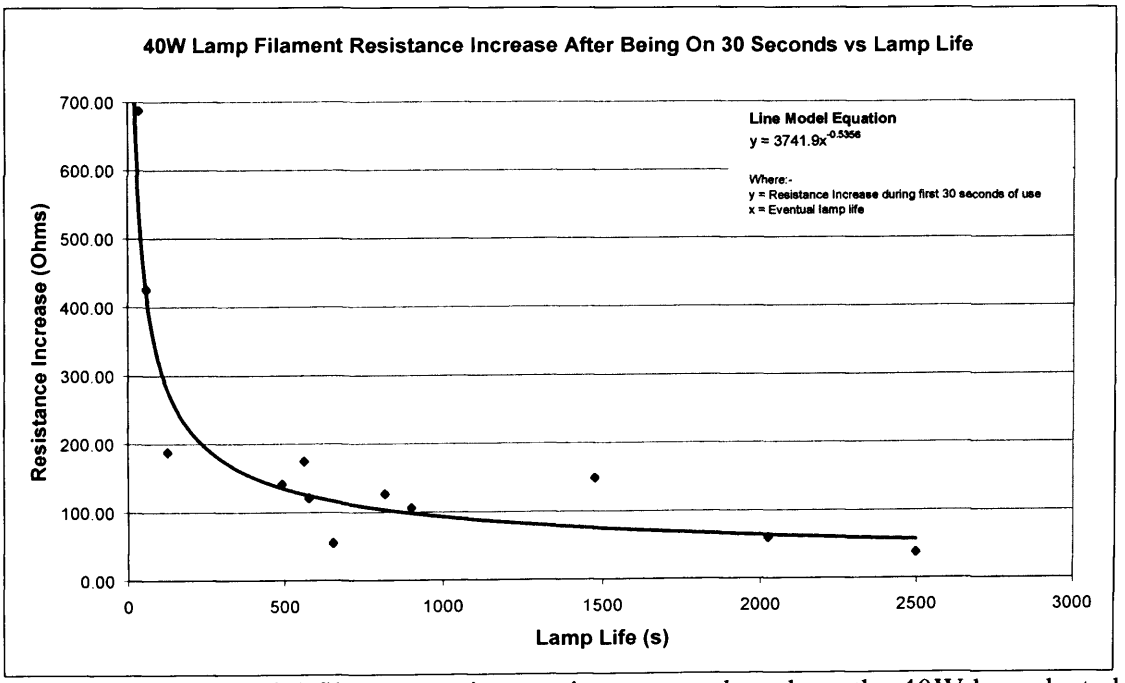


Figure 3:- Plot of initial filament resistance increase vs how long the 40W lamp lasted

From figure 3 it can be seen that there is a good correlation between the increase in resistance when the lamp is turned on for the very first time and how long the filament is likely to last as a result of incorrect gassing.

Figure 4 shows the deposit patterns on one of the lamps after the test was completed. Normally there is only a very small amount of Argon and nitrogen in the lamp envelope, as the lamp envelope is mostly evacuated; however as the lamps in this experiment were filled with air and argon there was no vacuum at all. The grey deposit, which formed after the white deposit from the filament burning ceased forming, was therefore probably due to increased thermal currents within the lamp which lead to the increased rate of filament evaporation; this was backed up by the swirling patterns evident in the grey deposit.

Figure 4:- Photograph of one of the modified lamps after testing

By considering again how most lamps are filled, "the bulb is evacuated through the exhaust tube and filled with nitrogen/argon gas. The bulb is partially re-evacuated and the lower end of the tube heat-sealed.";[2] it becomes evident that there are two problems that could occur, the nitrogen/argon gas may not be added correctly or the lamp may not be evacuated fully. Although more research work needs to be done to establish how the results correspond to real life lamps, it is clear that this testing technique has the potential to quickly diagnose both of these problems, as a large initial increase in resistance indicates too much oxygen in the lamp as a result of poor evacuation and the rate of resistance increase after this initial period will vary depending on the amount of Argon/Nitrogen left in the lamp, due to evaporation and thermal convection.

Although this testing technique will not detect all possible faults with a lamp, its main benefit is the simplicity of the test. All that is required to detect incorrect gassing in a

previously unused light bulb is to illuminate it for around 30 seconds by connecting it to a stable voltage source and monitor how quickly its current draw decreases (resistance increases) during the first few seconds of use. The test does not harm the light bulb and can be conducted on the actual lamps that will be put into service; with the use of computer interfaced test equipment it could be performed completely automatically. To develop this technique further a larger scale trial is required so that the behaviour of normal and genuinely faulty lamps can be recorded and thresholds determined for reliably identifying the faulty lamps in the shortest test period possible.

4 References

[1] House of Commons Hansard Written Answers (2003, June 20th), Column 453W, "Light Bulbs" – Estimated figures from the Lighting Industry Federation in 2002 [Online] http://www.publications.parliament.uk/pa/cm200203/cmhansrd/vo030620/text/30620w03.htm [2006, February 12th]

[2] Centre for solid state science, Arizona State University, "Why does a light bulb burn out" [Online] http://invsee.eas.asu.edu/nmodules/lightbulbmod/comp.html [2006, February 12th]

[3] Cayless and Marsden, 1983. Lamps and Lighting, Third Edition. Page 134