Low-cost and Real-time Oceanographic Data

Samuel D. Langbridge

University of New South Wales at the Australian Defence Force Academy

The aim of the project was to deploy a semi-permanent and remotely accessible buoy which takes oceanographic observations and distributes these observations via the internet. This project intended to fulfill the need of the remote research platform for the Underwater Communications Group at UNSW@ADFA. The system was designed, developed and constructed so that the need for a remote research platform was addressed which was done by meeting all of the requirements of the Underwater Communications Group. These requirements were either explicitly stipulated or derived from implied needs. The requirements were; a remote research platform that could collect and distribute oceanographic and acoustic channel data, operate Windows XP OS, semi-permanent, deployable to Jervis Bay. The system comprises of a solar powered computer, floating on a buoy, which manages its own power by waking up to perform tasks and then hibernating for a period depending on the charge left in its batteries. This computer is remotely accessible through the ADFA network. The oceanographic data was collected using a thermistor chain design and built by myself. It takes five temperature observations at five depths over the duration of the computer's hibernate period to a resolution of 0.204°C. The system is scheduled for deployment in early November to Jervis Bay. The buoy will host an acoustic payload for underwater communications research. All of the oceanographic data collected by the buoy is made available at http://seit.unsw.adfa.edu.au/research/sites/atlantis/. A complete and operational system which achieved the need of the Underwater Communications Group for a remote research platform and met all of their requirements was successfully handed over to the group on 25 October 2010.

Contents

I. Introduction 2
   A. Oceanography 2
   B. Underwater Communications Research Group 4
II. User Requirements 4
   A. Research Platform 5
   B. Operate Windows XP 5
   C. Semi-permanent Deployment to Jervis Bay 6
   D. Remote System 6
   E. Collect and Distribute Oceanographic Data 7
III. Research Platform 7
   A. Connection Between Buoy and ADFA 7
   B. Buoy to ADFA Communication 8
   C. ADFA to Buoy Communication 9
IV. Remote System 10
   A. Power and Energy 10
   B. Hostile Environment Operations 12
   C. Reliability 13
V. Payload 15
   A. Acoustic Payload 15
   B. Thermistor Chain 15
VI. Software 23
   A. Initial Start up 24

1 OFFCDT, School of Engineering & Information Technology. Electrical Engineering Project A, ZEIT4298 and Electrical Engineering Project B, ZEIT4299.
This project has been set by the Underwater Communications Group at UNSW@ADFA. They require a permanent mooring in an ocean environment which can perform the functions of a research platform for the conduct of tests, with a high-speed internet connection. In addition, the project is to carry a sensor payload for the collection of oceanographic observations. It had been determined at the start of the project that the physical system for the Underwater Communications Group’s communication node would consist of a permanent anchor attached to a buoy which housed the system. The project then consisted of the development of the system inside of the buoy and the sensor payload. The result is a self-managing remotely accessible computer that will be situated in the Jervis Bay approximately 1 km east of Huskisson. A map of the area is included in Appendix G (Tourism Jervis Bay, 2010). The collection of oceanographic data and the remote research platform are two separate functions of the system. The oceanographic data will be displayed and available to the general public. The data generated by the research platform is for use by staff at UNSW@ADFA. The project was designed using engineering rigour from a systems engineering approach. The project was completed and determined successful by the acceptance of the system by the Underwater Communications Group on the 25th October 2010.

A. Oceanography

Oceanographic data, especially observations with a high time resolution close to the coast has a wide range of useful applications in daily life. This includes detailed water condition data for fishing and diving or tidal updates for swimming and surfing. In addition to these applications, which are of interest to the general public.
there is great potential for this system to produce oceanographic observations which are relevant to the global scientific community.

1. Oceanographic Data Standards

Oceanographic data standards are important and effort has been made to address this issue in the project. The intention was to have any of the data produced by the buoy to be available as a resource to the international community of oceanographers. This meant that the information would be made available by the internet and I gave two requirements for this distribution while designing the way in which the buoys data would be generated. First the information should be in a format that displays the data in plain English. This ensures that the data is not only available to everybody to access but can be utilised by anyone because it is simple to understand and needs no technical information about the system and very little instruction on interpreting the publicly displayed format. Secondly the data should be easily transferable into a format that complies with a standard data format for meteorological observations. If this standard format is very detailed then it is to be in addition to the publicly available format. This second consideration was made so that the data could be transferred to a format at a later stage since the actual compliance to a standard was outside the scope of the project. Due to the design of data generation, discussed in Section 3; the software implementation of a standard should be relatively simple. Compliance with a standard means that the information is useful to others who can then easily compile this data with other data of the same format and there are several organisations attempting to achieve. These organisations as well as the usefulness of standard data formats are discussed in Section 3.

2. Australian Oceanographic Data

The Oceanographic Services of the Bureau of Meteorology states the following with reference to oceanography:

“Oceanic conditions are a critical factor in the earth's climate system. They directly influence fisheries and most aquaculture endeavours, while knowledge of them is essential for such diverse applications as coastal construction, maritime safety, marine pollution response and sustainable management of the marine environment”.

(Australian Government, 2010)

The Australian Ocean Data Centre Joint Facility (AODCJF) is a conglomerate of departments in Australian government organisations which aims to provide ocean data as a resource that is managed through a national level network. The AODCJF consists of Australian Institute of Marine Science (AIMS), Australian Antarctic Division (AAD), Bureau of Meteorology (BOM), CSIRO Marine and Atmospheric Research (CMAR), Geoscience Australia (GA) and Department of Defence (RAN Directorate of Oceanography and Meteorology).

The highest profile use for Australia’s off-shore data is the Great Barrier Reef. A mean temperature change of as little as 1°C causes the coral to bleach permanently damaging one of the wonders of the natural world. The CSIRO conducted a study which consisted of both observed ocean data along with simulations based on this data given different scenarios branching out from the present in order to determine the effects of soil erosion and grazing near the Great Barrier Reef. The study involved a graphical plot of water temperatures overlayed with sediment concentrations as they travelled through river systems in the Great Barrier Reef’s catchments to plume across the reef. (CSIRO, 2010). This is an example of the applications of the system and highlights the importance of collecting oceanographic data.

3. Existing Systems

The actual technology used in the project has existed for some time and similar systems which collect and transmit oceanographic data have been operating off the coast for decades. These systems are developed for a very specific task and so capabilities vary between them. The National Oceanic and Atmospheric Administration (NOAA) have buoys for collecting deep sea data some of which were used to monitor conditions during the recent oil spill in the Gulf of Mexico (US Department of Commerce, 2010). OSIL Monitoring Solutions developed a network which monitors environmental conditions in the Arabian Gulf for Kuwait. These collect data in real time and transmit warnings to the Environmental Protection Agency if any thresholds are breached (OSIL, 2010). The CSIRO have robotic buoys deployed off the east coast of Australia which monitor rainfall (ABC News, 2010). Figure 1 shows the system developed by Datawell Directional and deployed to the Celtic Sea, operated by International Ocean Systems (Underwater World Publications Ltd, 2010).

This project differs from other systems because of its research capabilities. All of the existing systems outlined above are controlled with purpose built microcontrollers. Rather than employ a microcontroller, the research platform uses a PC and a desktop operating system. The desktop operating system, which is Windows XP is a requirement set by the Underwater Communications Group. This enables a very
versatile and reconfigurable system to be provided. Despite the increase complexity through the use of a desktop operating system the advantage provided is a deeper level of flexibility which was exploited when designing the software architecture. This additional flexibility is what allows a research payload to be hosted and differentiates this system from current similar systems.

B. Underwater Communications Research Group

The underwater communications research group is researching different forms of communications and sensing underwater. The intersection between underwater communications and low-cost, real-time, oceanographic data is the development of an array of data collecting buoys which do not penetrate the ocean surface and rather sit just under it. These buoys then form a network in which they communicate and share data. All of the data is then returned to a central gateway buoy which does penetrate the surface and is able to distribute this data back to shore. Having a surface penetration is a major disadvantage since it allows moorings to be damaged, leaves the buoys exposed to weathering and risks having the buoy impacted by vessels. In order to work around these disadvantages the communication between the underwater “nodes” needs to be investigated. The project deploys a research platform for that investigation as well as providing a test of function for the network gateway buoy.

1. The Project

The goal of the project is to deploy a semi-permanent and remotely accessible buoy which takes oceanographic observations and distributes these observations via the internet. Achieving this goal will fulfil all of the needs of the Underwater Communications Group. In order to achieve this goal several user requirements need to be met. These requirements are outlined in Section 2. The main constraint for the project placed by the underwater communications research group was the need to utilise a Windows XP operating system and that this software be remotely accessible. The project was scoped so that it could be finished within the timeframe of a single academic year. The project was to consist of the gateway buoy only and would not include the underwater network developed last year by Shaun Reece. A block diagram of the completed system is included in Figure 2. Circuit diagrams of the completed physical system are included in Appendix H and details of the physical components used can be found in Appendix I.

2. Paper Layout

The user requirements will be listed and described in Section 2. How the system is accessed and utilised as a research platform is discussed in Section 3. Section 4 details the considerations made for designing a remote system that is to operate on the surface of the ocean. Section 5 details the payloads attached to the system, both the acoustic research equipment and the thermistor chain designed and built by myself. Section 6 is a summary of the software architecture employed on the buoy. A conclusion and the recommendations for future work is included in Section 7.

II. User Requirements

This project has been assigned with the intention of fulfilling a unique need of the UNSW@ADFA Underwater Communications Research Group. For this reason the customer – being the subject research group and, hereafter referred to as the user, needs to specify a list of requirements. The user requirements are put in place for several different reasons. Firstly the user has detailed knowledge about how and where the buoy system is to be used. They convey this information to the system developer (myself) as a set of requirements. The purpose of the user requirements is to ensure that the project has the functionality and performance to fulfil the user’s need. The user requirements should ensure that enough flexibility in potential solutions is available so that the project is achievable at reasonable cost and in a reasonable timeframe. The user requirements are detailed in this section. As the system was developed, solutions for each of the user requirements were designed, evaluated and then presented to the customer. However some of those solutions were deemed inappropriate or insufficient by the customer who had a wider view of the systems intended use which then generated further user requirements. In other words the user did not stipulate all of the user requirements at the beginning of the project. Some requirements were elicited as
various solutions or sub-systems were proposed or demonstrated. This made it imperative that I elicit these un-stated requirements as early as possible by means of rapid prototyping. An image of the prototype system is shown in Figure 3. The design could then be matured with confidence that critical user requirements had indeed been captured.

During the development of the project there were two primary cases where a design change was made which required rework to the system but was caught relatively early through prototyping. These changes were implemented because solutions failed to meet user requirements as a result unforeseen consequences rather than the user reshaping their requirements after a solution was proposed. The rapid prototyping which was employed was a rough system placed on the roof of Building 15 which then performed the primary function of the deployed buoy being waking up, performing tasks and then hibernating again. The first instance of a design change came from the software architecture. The original system collected temperatures and sent this, along with any other logs or schedules relevant to the systems administration as email attachments and to the server. It was observed that the longer the system was operational for, the longer it would spend awake before hibernating. The method of collecting and storing the data was to simply append it to the end of a text file and then email or push to the server the entire file. This was very inefficient and with the amount of data that needed to be transmitted the file sizes were rapidly increasing. The solution was to take a single set of data from the wake up period, this could be voltage values, temperature values or node administration files such as the IP configuration, email or push that file to the server and then concatenate it into a master text file containing previous sets of data into a master text file to be kept on the buoy. This required the entire order of daily operations of the buoy to be re-ordered as well as an overhaul of the methods used to communicate to the server. The second instance came after the need for self-power management was identified. Having a set of wake-up and hibernate cycles which did not alter was observed to be inefficient and so the system required a way to take advantage of large amounts of energy available but be able to reduce power consumption when energy was lacking. This would be achieved by varying the frequency of the computer wake-ups. Up until this point the wake up and hibernate cycles of the PC were controlled through Windows task scheduler. This would not work if the wake/hibernate cycles were to be varied since a new task had to be manually set by the user. To work around this an existing script (Boskoop, 2006), from the website AutoHotkey.com (AutoHotkey, 2010) was modified so that it would be able to be used as a batch file command which meant that the way the computer cycled through wake and sleep states had to be redeveloped from the ground up. This script is discussed in Section 6, Part B.

The user requirements will now be stated and briefly discussed.

A. Research Platform

This is the primary objective for the project which means that this requirement is the motivation behind the proposed solutions for other requirements. The buoy has been described as providing “low-cost, real-time, oceanographic data” however the motive for the users need for the system is an acoustic communications network testbed and gateway. This requirement differentiates the project from current systems available since it is required to host a research payload accessed through a Windows XP operating system which makes an ‘off-the-shelf’ solution unsuitable. The goal of having an operational research platform is assessed as being achieved or not through the testing of the solutions to all of the other requirements. If all of the other requirements are met, with no exception, and the solutions perform to a sufficient standard then the system is deemed to be a suitable research platform.

B. Operate Windows XP

This is a key user requirement that has been specified because of the software required to run the Underwater Communications Group’s payload. It acts as a constraint since it details the operating system to be used which means any software architecture which is to be run must either be hosted on or interface with this operating system. It also adds a substantial layer of complexity to the system. Until now similar systems have employed the use of a microcomputer for data collection and distribution. These have the advantages of being reliable, simple and having low power consumption. The use of a high level desktop operating system, and hence some sort of PC, means that the system will have an inherent and undesirable amount of unreliability, consume a relatively large amount of energy, be complex and contains software which has been designed with the assumption that there is a user physically present to perform a hard reset as a remedy for minor glitches. However this provides a deeper level of flexibility which was utilised to full advantage in the software architecture. The additional flexibility is what allows a research payload to be hosted and differentiates this system from current similar systems.
C. Semi-permanent Deployment to Jervis Bay

This requirement provides all the information required to prepare the system for its operational environment. A map of the area the system will be deployed has been included in Appendix G (Tourism Jervis Bay, 2010). This makes it clear that the system will operate on the surface of the ocean and that it has to be a self sufficient system which leads to the following requirements:

1. **Operate in a hostile environment**

   Deployment to Jervis Bay implies a hostile environment for a system with relatively sensitive components. The need to operate in a hostile environment governs the housing of the system and the interface between the system and the environment. This is because the housing has to be able to withstand the environment but allow the system to operate and the interface needs to allow the system to interact with the environment without exposing it to hostile elements. The hostile environment can be split into the following five environmental conditions.

   - **Saltwater Vapor**: It has severe corroding effects on any metal that is left exposed and sensitive components such as the PC will need to be completely sealed from it.
   - **Water**: The system operates on the surface of the ocean and so the water sensitive components will need to be shielded from the splashing of the ocean as well as rain.
   - **Solar Radiation**: Any of the casings exposed to sunlight, as well as the solar panels should be suited and adequately prepared for continued operation in direct sunlight for an indefinite period of time. There should also be sufficient cooling for any temperature sensitive components which will be heated by solar radiation.
   - **Open-water Conditions**: The structure supporting the system (i.e. the buoy) needs to be able to stay afloat in typical ocean conditions for the area and have a very low risk of capsize. The system must be appropriately secured so as not to move around or fall off in rough seas.
   - **Collisions**: The system must be placed outside of high traffic areas and be appropriately marked for both day and night so that the risk of collision is reduced to a minimum.

2. **Deployable System**

   Much of this is implied in the remote system requirement. However the user requirement for a deployable system reiterates the need for the system to be transportable, self-contained, operate for an indefinite period of time and able to operate normally without any physical interaction.

D. Remote System

The research which is being conducted relates to underwater acoustic channels. The research platform is to facilitate this research and will be operated in Jervis Bay. The user has stipulated that the system be able to operate and conduct this research remotely, without physical interaction or external facilities. A remote system has several implied requirements:

1. **Solar powered**

   As the requirement for a deployable system stipulates, the system must be self-contained. Since no physical interaction is permitted for normal operation and the system is deployed for an indefinite period of time and energy storage used to power the system will have a finite capacity a renewable source of energy is required on-board the system in order to generate power. The requirement was then derived to be a solar powered system.

2. **Manage energy consumption**

   This was a derived requirement in order to increase the efficiency of the system. Because the system incorporates a PC, which consumes a lot of energy, not only would keeping it running be unsustainable but the amount of energy stored can be controlled to a great degree by cycling the usage of the PC. This means that the system would be able to take advantage of situations when a large amount of renewable power is available.

3. **Remotely access Windows XP OS**

   Because the research platform is in a location which is relatively remote, use of the platform or recovery of data at the location of the platform defeats the purpose of the deployment. As a result the research platform itself needs to be remotely accessible. Since the system and the research platform do not necessarily have to be the same thing, just in the same physical location, it is stipulated that the Windows XP OS needs to be remotely accessible since it is the operating system which hosts the software which controls the research payload.

4. **Remotely Controllable**

   While the system is to operate on its own as much as possible, the requirement has been put in for the system to be remotely controllable. This is to allow; changes in the systems operation, software updates and other similar administrative duties to be made without having to physically be at the system. Even though the eventual solution was incorporating the systems main software architecture within the Windows operating system, this has been separated from the previous requirement in order to make the clear distinction to the systems operation and the research capabilities.
5. **Self-recoverable**  
This requirement is a direct result of using the Windows XP operating system. The inherent unreliability of any high level operating system is discussed further however it is clear that the unpredictability of the operating system requires an independent, simple and reliable external circuit which can perform a hard restart of the system should the Windows XP operating system cause a catastrophic failure of the entire structure.

E. **Collect and Distribute Oceanographic Data**  
This requirement has been specified as a proof of concept and functionality of the collection and distribution of oceanographic data. As such, whilst the sensors developed and built are accurate temperature sensors the actual accuracy (compared to the theoretically achievable accuracy as discussed in Section 5) and the required calibration have not been definitively tested and confirmed.

1. **Collect water temperature**  
The choice of sensor and the data which it collects is arbitrary and has not been specified by the user requirement. So in order to fulfil the user requirement of collecting oceanographic data, the derived requirement of collecting water temperature has been specified.

2. **Distribute data**  
The buoy must be able to distribute the data it collects over the internet in a format that is easily understandable. This must be done in a manner which allows the buoy to be treated as a resource by those who have no connection with UNSW@ADFA. Data recovery from the buoy is not to be manual.

**III. Research Platform**

The requirement for the buoy to be a research platform means that the system has to be able to perform three core activities. These are; being able to collect environmental data, have full-duplex communications with ADFA and control a remote research payload through the Windows XP Operating System. The actual collection of the environmental data is covered in Section 5. From the viewpoint of simple environmental data collection communications between the buoy back to ADFA is only what is needed since the system has been designed to be sufficiently self-managing to take care of its own administration. This means the only requirement is for the buoy to make the data available to an external resource. The requirement for duplex communication arises on the research platform in order to increase the flexibility of the research. If a method of communicating to the buoy is unavailable then the research schedule would need to be set before deployment and would be unable to be changed without physical interactions. Communicating to the buoy also allows its day to day operations to be updated and altered.

A. **Connection Between Buoy and ADFA**

Figure 4 shows the method in which the buoy is connect to ADFA and the user. Figure 4 highlights the way in which the user connects to the buoy and that the buoy pushes data to ADFA. The buoys connection to the outside world is through a wireless 3G modem. The telecommunications company that provides the service is Virgin mobile and the service operates on a prepaid basis. 3G was chosen because it has high capacity, low management overhead, is low cost, is a mature technology and there is good 3G coverage at the location of development. The choice of the internet connection is flexible for a more mature system and other connectivity options can be used in locations which are more remote or subject to interference. The requirement for any scheme to access the internet is that it can be connected and disconnected automatically. This is done for the wireless 3G modem using the Windows command line command “rasdial” along with the name of the connection which in this case is “Virgin”. This allows the connection to be turned on and off through a batch file. If the internet access scheme is changed the connection method needs to be taken into account because the connection must be brought up from a command line. An added benefit of 3G is that it provides sufficient data rates for the buoy to execute its operations quickly. When running the scripts there is little notable variability in the time it takes for the tasks to be completed. The biggest variation in the operation comes when the buoy is pushing its data to the server. Due to the concatenation of the data (covered in Section 6) the file sizes remain very small and fairly consistent. However the time the buoy spends accessing the server will change depending on the data rate achieved on the 3G link. If an alternate connection is to be selected, power consumption of the system may increase if the internet access scheme provides low data rates since the computer will be powered-on for longer during each upload cycle.

A Virtual Private Network (VPN) is created on the 3G connection. The VPN provides virtual access to the ADFA network at the IP level without the buoy being physically connected to the ADFA network. The VPN has been established by ADFA IT personnel and has already been made sufficiently secure to dramatically broaden the choice for access schemes between ADFA and the buoy.
B. Buoy to ADFA Communications

The buoy to ADFA communication is a part of the buoy's routine reporting of its own status as well as any data it has collected. This direction of communication is specifically for the collection and distribution of oceanographic data and does not include any of the data acquired through the research payload. As mentioned above and will be covered in more detail in Section 6, all of the data that is going to be sent back to ADFA are the last observations so that the files being sent are small. After these last observations are tagged with a time date stamp these are then pushed to a server on the ADFA network, named “Atlantis”. The push is done through a batch file utilising the Windows “move” command. The command “move” is used rather than the command “copy” because the files on the node have already been concatenated so leaving the additional copy on the remote computer is unnecessary.

These files are pushed from the buoy to the server rather than pulled from the server because the latter would require the server having information about the buoy’s wake and sleep cycles. All of the information relevant to the single buoy, specifically its operating cycle and the conversion of the raw data is kept to the buoy. So having the data pushed from the buoy to the server means that the server can display the buoys latest information without knowledge of when the buoy is connected to the network.

The software design and development for the buoy is based around minimising operator load and management overhead and this means employing the concept of compartmentalisation so that the system can operate as independently as possible. Compartmentalisation reduces the number of users required to manage a buoy and has been implemented for when the project is developed into a more complex system. The idea is to have the system as encapsulated as possible and thus separated from the server which enables the server to be “blind”. In the implementation of the Underwater Communications Group network of nodes, there would be one gateway node responsible for a collection of nodes under the surface. If there are tens or hundreds of these gateway nodes in operation they have to manage themselves and their network. Since the server is my solution for the distribution of data, its only task is to display the data that is being produced which means that the website itself can be treated as a simple resource, accessible by anyone. If the distribution of the data is to be expanded than it would be a matter of outputting the buoys data to a different resource.

Sensors collect data as observations however these observations are not easily readable and the raw output from the sensors depend on how the environment is being measured. In this case thermistors are being utilised which means the varying resistance is output as voltage observations. In order to keep with the idea of compartmentalisation all of the data is processed on-board the buoy. When comparing the server and the buoy it is a case of a remote system and a fixed system. The fixed system essentially has unlimited power and computational resources. The remote system has very limited power and this restricts the computer processing ability since the time it can spend on calculations is finite. The computer processing ability is also limited since the hardware has to be small enough to allow the remote system to be deployed and is fixed so it cannot be upgraded easily with improving technologies. It would make sense to have the fixed system to do the calculations of the raw data. However the server needs to be able to display readable, sensible and detailed data without having any information about the buoy that the data is coming from. Having the server change the voltage observations into legible temperatures would require information about the buoy, in this case the resistor values used in the interface circuit between the thermistor and the data logger. This also makes the data less accessible to the wider scientific community since the project is now leaning on a server to recover useful data which means the raw voltage observations have to pass through UNSW@ADFA before it can be distributed. By having all the data processed on board the system is more open ended and more useful to interested parties. The cost in the extra power required to produce readable temperatures on-board the buoy far outweighs the reduction in management overhead on the server as well as the benefit of producing and distributing data which requires no technical information relating to the specific operation of the buoy.

Figure 4: Connection diagram for buoy to ADFA link
Data formatting is very important when an area as large and diverse as the world’s oceans is being mapped since it will require the assistance of several organisations working alone in their area of interest. Collecting this data and turning it into something scientifically useful in order to get some global picture is next to impossible if the organisations are operating in completely independently. Having all the data produced in a common format and having sensors follow a standard will go a long way to achieving complete coastal coverage since data can be shared. The Open Geospatial Consortium (OGC) is working towards providing data as a resource (Open Geospatial Consortium Inc, 2010). It is a consortium of 406 companies, agencies and universities with the aim of developing open standards to achieve geospatial content and services to be integrated into the internet, business and the wider community (Open Geospatial Consortium Inc, 2010). The Group on Earth Observations (GEO) has similar goals. This is a voluntary group of governments, their agencies and international organisations. GEO is currently responsibly for the efforts to build a Global Earth Observation System of Systems (GEOSS) (Group on Earth Observations, 2009). The idea behind GEOSS is to link existing and future observation points and promote common technical standards so data collected from multiple instruments at multiple locations can be collated into a coherent set of data. This is to provide decision makers with an exhaustive range of up-to-date environmental data available on a single network (Group on Earth Observations, 2009). Compliance to a specific standard was outside the scope of the project. The data was displayed in a simple format where the temperatures and the time of the observation are listed in columns. This is generated by compiling the relevant data files containing the temperatures and the times and then arranging it into a readable format. By simply altering the java program which conducts the formatting a recognised data format standard can be met.

Most of the data the buoy generates about the temperatures and itself is made available on the server which is accessible by an operator. Selected information on the server is then displayed through a webpage which is at the address: http://seit.unsw.adfa.edu.au/research/sites/atlantis/. This is accessible outside of the ADFA network. The most relevant file placed on the server is the summary of all of the buoys collected data, which is shown in Figure 5. Figure 5 shows the column headers of the information along with the data below it. The data continues through to the thermostor temperature for sensor e but has not been included in Figure 5. The case temperature isn’t recorded at the time indicated to the left, it is taken using a separate script while the computer is cycling through its standard routine. This is because the cooling system, which uses the thermostor that senses the case temperature, is operated through the administration µChameleon. The administration µChameleon is not able to keep a record of temperature observations, in the form of ADC values, while the computer is in hibernate because it runs the watchdog script. This means that the only time a temperature observation inside the case can be made is when the computer is on and the µChameleon can be queried directly through MATLAB. The text file which contains this information is unique in the software architecture. It is not sent as a last observation and then combined into a master file like the nodes other data. Rather each time the computer wakes up it takes all of the data in the relevant master files and formats it into this single file. This means that the time it takes to format the data and then send it will slowly grow as the collected data grows however given the size of the data it will take a considerable amount of time for any noticeable difference in performance and even so, the logs on the remote computer can be cleared easily using manual or automatic access.

C. ADFA to Buoy Communications

ADFA to buoy communications is provided in two ways; automatic and manual. The instructions for remote and manual access can be found in Appendix E.

1. Automatic Access

Automatic access is simply the reverse of buoy to ADFA communication. The Windows “move” command is used to pull a specific batch file from the server. This file contains the commands a user wishes to execute and once pulled off the server the buoy executes these in sequence. This means that buoy is able to automatically execute any function that can be placed in a Windows batch file making it extremely flexible. Such commands might run specific experiments, reconfigure the buoy software or change the buoy wake/sleep cycle.
also copy additional software from the remote server onto the buoy to expand or otherwise modify its functionality.


Manual access provides the user with a remote desktop to perform interactive operations. By reference to Figure 4 the connection from the remote user block, through the VPN using the remote desktop application TightVNC to the buoy can be seen. Virtual Network Computing (VNC) allows access to the graphical interface of a computer remotely using remote frame buffer (RFB) protocol. This is done over an IP connection; in this case on a VPN through the internet. The need for access to the buoy computer’s graphical interface was deemed necessary because of the Windows software used to operate the research payload. There is a software package (Virtual Desktop) available in Windows but TightVNC was selected in order to adhere to ICTS corporate policy. ICTS is responsible for the creation and maintenance of the VPN and the use of the TightVNC software package is consistent with their current access methods for pre-existing VPN’s. The VPN appears as virtual modem under the heading “PPP adapter ITEE” in IP configuration and TightVNC (or any remote desktop application for that matter) accesses the PC through that. The actual specific manual access is very straightforward. Since the graphical interface is being displayed the user can just treat the TightVNC display as a computer sitting in front of them.

IV. Remote System

The buoy is intended for remote deployment. This means that it must be able to operate in a remote location with no physical interaction or external support. There are three main considerations that were made when specifically addressing the requirement of a remote system. These are; power management, hostile environment operation and reliability. The need for reliability is a direct result of the use of the Windows XP operating system and would not have had the amount of focus and attention given to it if a device with more predictable failure modes, such as a microcontroller was being used instead.

A. Power and Energy

Three items are addressed with reference to power and energy; the actual supply of energy to the system in a remote location for an indefinite period of time, how to manage the limited resource of power on the remote system and how to optimise the power available.

1. Energy supply

The system requires an independent, portable and renewable power supply for the remote system. The solution selected is solar panels with suitable marine packaging to charge a battery. A charging regulator is also included in the system to ensure that the battery is not over charged which would damage the battery (Virtual Technologies, Ltd., 2007). The voltage regulator restricts the battery voltage to 13.5 V. Solar power was assessed as the most economical option since it is relatively cheap and simple to implement. There is insufficient room on the buoy to utilise wind power. Ocean power would seem appropriate since the operating environment is on the ocean however these are not readily available commercially. Solar power is cheap and commercially available off-the-shelf. The solar panels used for the deployment are two 20W panels provided by BP solar. The two panels are mounted on opposite sides of the buoy to ensure that energy is generated each day irrespective of the buoy orientation. During the prototyping phase, an investigation was conducted on supporting the power requirements of the system for an indefinite period of time. Battery voltage observations over a winter period are shown in Figure 6, for the system operating on the roof of Building 15. The system had a hibernation period of 1 hour.

![Figure 6: Battery voltage over time between the periods of 23 Jul and 6 Aug](image-url)
and was awake for 5 minutes when it was conducting its tasks. Here the daily cycles are apparent and the system is shown to support itself. This self-sufficiency was achieved using smaller batteries than those to be used for deployment as well as two solar panels rated at 20 W each. These solar panels would have only been operating at their maximum rating for 2 or 3 hours each day since they were both positioned facing directly upwards next to one another. This means that one of the panels attached to the buoy operating at rated capacity is sufficient to cover the power needs of the entire system in full operation with a hibernate duration of 1 hour which means the batteries would only begin to be drained at night or on overcast days meaning the systems power needs are met. Using the industry rule-of-thumb that systems generate approximately 6 hours at rated capacity of energy per day and assuming that 1 panel is effective at any given orientation, the solar panels should generate of the order of 120 Wh (432kJ) per day (Davis Instruments, 2010). The power budget developed for the system is included in sub-section 3. The battery to be used for the deployment is a 12V, 120 Ah deep-cycle calcium-lead-acid marine battery.

2. Energy Management

The solution used for the generation of power cannot support constant operation of the computer and its peripherals, especially during winter. This can be calculated by comparing power produced against energy consumed, see Appendix J, however it was observed during prototyping when the software failed; the batteries would be flattened completely. Since none of the available alternate means of power generation were feasible a method had to be devised to manage the available power. The solution was to cycle the computer through wake and hibernate states. In hibernate the computer draws very little power and the peripherals which would only be usable when the computer was on were not powered. The computer would wake-up periodically, perform its own administration, collect and process temperature data, communicate this externally and then return to the hibernate state. The trade-off to this solution is that the ‘real-time’ component of the oceanographic data is actually near real time. Rather than have the oceanographic data available in real time, the sample rate is reduced to several observations evenly spread over the hibernate period. To limit the impact of this trade-off the software architecture has been designed to allow the system to be reconfigured and provide real time data. However for this to occur the PC would have to remain fully powered.

The energy management scheme was first implemented using Windows Task scheduler and the result was the computer hibernating and waking at regular intervals.

3. Optimised Energy Management

Having the periodic waking and hibernating of the computer allowed the system to run, however there are two additional considerations to be made. Firstly the solar panels are not providing a very consistent power supply. They produce no power at night (for obvious reasons) and the power generated during the day varies with sunlight intensity which is affected by cloud cover and the time of year. Secondly, the use of the system is not necessarily consistent. This could be either using the system as a research platform with the acoustic payload, or when the need arises to produce real-time sensor data for a short period of time (for example the deployment location, Jervis Bay is flushed by eddy currents from the East Australian Current several times a year – this is of interest to oceanographers, as the phenomenon has not been studied in any detail (Marine Parks Authority NSW, 2008)). There was a need to provide variation in the power management. Since the computers use is what can be controlled from a power management perspective a variation in the power management can be achieved by increasing or decreasing the wake/hibernate cycle. For extended periods of use, the batteries should be as charged as possible and will be fairly diminished afterwards so the wake/hibernate cycle is extended before and after the extended use to ensure this. Varying the cycle will also take advantage of high sunlight periods and save power during periods of low-light conditions. Backing off the wake/hibernate cycle also prevents total loss of power to the computer during very long periods of low-light or use. The full details of the power budget are provided in Appendix J. The discussion here serves as a simple operational guide and more accurate calculations based on the information in Appendix J should be made if the system is to be operated to the edge of its limits. The power budget is based on the minimum hibernate period of one hour. This means that over a 24 hour period the system will consume 76 W (the computer is on for 5 minutes, operating at 16 W and asleep for 54 minutes operating at 2 W). This leaves 44Wh per 24 hours for research use. Assuming the acoustic payload has a maximum draw of 2A it should draw no more than 26 W. The computer uses a maximum of 16 W when performing heavy calculations (obtained from observing current draw when starting MATLAB) this leaves just over 1 hour of research work per day available. This will vary depending on the time of year. More sunshine is available during the summer; however the cooling system will need to be operated for longer. The cooling hasn’t been included in the calculations however is uses roughly 8 W when it is operating. The amount of time the cooling system will be on is largely unknown since it is uncertain how much of the heat from the sun and reflections from the water will be trapped under the buoy’s main dome, how well the systems case is insulated from this and how effective the cooling system is at transferring heat. The power budget can be more rigorously defined once data about the buoys power and cooling performance as well as the operating environment temperature has been obtained during the deployment.
Windows Task scheduler did not allow for the hibernate option to be given as an argument when booking a new task through the command line. In order to work around this a more versatile wake/hibernate function was obtained in the form of a script (Boskoop, 2006). The script was modified to take input arguments for the next wakeup time and a new command line argument was generated in a batch file through a java program each time the computer woke up. The script was obtained from AutoHotkey.com (AutoHotkey, 2010) and its details are discussed in Section 6, Part B.

B. Hostile Environment Operation

As stipulated in the user requirements there were several environmental factors which needed to be considered and they were; saltwater vapour, water, solar radiation, open-water conditions and collisions. The system was protected from these factors primarily with two structures; a Pelican case and a Solar Tech buoy.

1. Solar Tech Buoy

The Solar Tech buoy is a proven system and is currently in widespread use, including on the Great Barrier Reef in a marine sensor role. It limits the risk of collisions by being bright yellow, having the appropriate navigation markers for night and is identical to the current marine sanctuary marker buoys employed in Jervis Bay. It has a dome which protects from direct solar radiation as well as majority of rain and wave splashes. Being in use on the Great Barrier Reef its design is suitable for limited open-water conditions. Penetrations through the buoys base allows for equipment to be deployed directly downwards, rather than over the side, enhancing the robustness of the final system. An image of the solar tech buoy is shown in Figure 7.

2. Pelican Case

While the Solar Tech buoy provides a measure of environmental protection, the area under the main dome is not environmentally sealed. It is therefore a humid, salty environment. Inside of the buoy another case is required to ensure that any of the rain, splashing seawater and humid salt air that make it past the buoy cover doesn’t reach the computer or its peripherals. The biggest issue is corrosion from the salt present in the air. As a result the case must be relatively air-tight so that this doesn’t occur. A Pelican case meets rigours standards in waterproofing and air tightness, and such cases are used in similar situations (Pelican Australia, 2010). These standards are; MIL-C-4150J, ATA 300, Def Stan 81-41/STANAG 4280 and Ingress Protection. An image of the pelican case used is shown in Figure 8. Whilst the Pelican case provides the required shielding from moisture and salt in the air it also completely seals the system from the environment which raises two concerns, cooling of the system and interaction of the system with the environment.

Cooling: Cooling the inside of the case was a major concern because the computer is relatively sensitive to temperature having an operating range of 0°C to 50°C (VIA Embedded Platform, 2007). The computer is air cooled and since it generates heat, as well as any solar radiation that is trapped underneath the buoy cover means that the temperature of the case needs to be monitored and controlled. The cooling of the system was done using a computer liquid cooling system and operating it in reverse. Since ventilation is an unacceptable solution (due to the salty air) there were two choices for heat exchange, either through the case itself or through a coolant. Heat exchange through the case is insufficient in air but may be achieved if the case is fully submerged. This was investigated last year. However, even though the pelican case and the environmental connectors are rated for submersion, the testing required to prove that the penetrations in the case are submersible for an extended period of time meant that the option for heat transfer through a coolant was used. Modifying a pre-existing system which is known to work (this refers to the general function, not the ability to cool since the configuration is being altered) to meet the cooling needs saved a great deal of time in the design and development of a cooling system from the ground up. A computer cooling system is suited for operation in a similar sized space and requires relatively little power. A fan in the case circulates the internal air. Cold water is pumped into the case and passes through the radiator, situated behind the fan. The heat exchange occurs in the radiator so the cooled air is now circulated and the warmed water is pumped out. The tubing containing the coolant runs out of the case through the buoy case and into the ocean, where a 10 m length of tubing allows for heat exchange into the
ocean. The water temperature is typically less than 20°C. The tubing goes back up into the case carrying cold water and completing the cycle. Although the length of plastic tubing is not an ideal heat exchange membrane, it has the advantage of reliability and simplicity, there being no underwater joints, and no requirement to fabricate a specialised set of piping. Any inefficiency in heat transfer can be compensated for by using a longer length of tubing. An experiment was conducted in order to determine the effectiveness of the cooling system. The file containing the experiment notes and the interpreted numerical results was corrupted. However the conclusion from the lab experiment was that with the case closed the temperature of the CPU rises and that the cooling system successfully lowers the CPU temperature. The raw data and the experiment set-up can be found in Appendix M. The power to the cooling system is controlled by the administration µChameleon which also has a resistor connected to it. The µChameleon checks the ADC value of the voltage across a thermistor and then turns the cooling system on or off accordingly. In order to monitor the temperature inside the case while the computer is off, the ADC value is checked without being processed into a temperature. This means that the actual temperature at which the cooling system turns on will decrease as the battery voltage decreases. These ADC values and the corresponding maximum temperatures that they represent are as well as the reasons they were chosen are detailed in Appendix F.

Environmental Interface: External connections were required to interface the system with the external environment. In addition to this the size of the case required for the system could be dramatically reduced by having the battery and research payload located separately. The choice for environmental interface is Buccaneer 400 Series Waterproof Connectors. They provide an IP68 environmental rating when connected and allows easy removal of the sensor payload, research payload and power connection at reasonable cost. The IP68 rating means that the connections are dust tight and can be continuously immersed in water beyond 1 m depth. This ensures that the equipment is able to interact with the external environment but still be sealed from salt present in the air as well as rain and wave splashes. The pin diagrams of these connections is included in Appendix K.

C. Reliability

A very large emphasis was placed on reliability. High level operating systems, such as Windows, are designed and developed specifically to assist a human user in communicating with machines. This means that there is the very fair assumption that a user is either physically interacting with the computer running the software or that the computer running the software is at least easily accessible. As a result there are few provisions in the software to fix software glitches in a way that would not require a human user to work around them (either performing a hard restart or clicking cancel etc) because there is absolutely no demand for it in the general market. In the case of this remote system there is not always a user connected to click around prompts and there is never a user physically present to perform the hard restart that is sometimes required. As a result, computer issues which face us everyday have a dramatic effect on the remote system automatic tasks or the users manual remote access.

Given the amount of time and money taken to physically restart the computer the amount of times this is required to be done is to be no more than once a year. Given a minimum sleep period of 1 hour the computer wakes up 8760 times of which there is to be a maximum of one failure. Extensive testing has been done from the start of the project to try and establish this probability of failure, but such reliability is time consuming to demonstrate by simple testing. There are two types of failures as identified, defined and described by myself, the first one being a crash. This usually means the computer has frozen or forced its own shut down. These are easy to diagnose because they tend to be repeatable. In the case of a desktop machine these require the computer to be restarted. The second is a ‘hang’ and refers to the automatic tasks performed by the computer. A hang is anything which prevents the computer from performing its next sequenced task and thus stops it in the middle of a wake cycle. On a desktop machine these require user interaction – often with a pop-up alert box, which might be as simple as choosing to accept the change to daylight savings time or agreeing to update of a software package. Since the solar panels cannot support the computers on state for an extended period of time the computer will remain awake until the batteries are flat. These failures are of the greatest concern because they can be unpredictable and unrepeatable making them difficult to find and debug.
The simple testing isn’t very effective for determining reliability for two reasons. The software was slowly being updated and refined throughout the testing process and was only finalised at a relatively late stage in the project. Figure 10 and Figure 11 were obtained from a paper written by Jiantao Pan on the topic of software reliability (Pan, 1999). The graph in Figure 10 illustrates the observation that even minor changes in the software can create unexpected and unforeseen failures which are not immediately apparent. Secondly the reliability in software cannot be definitively tested, which means the value $\lambda$ cannot be determined with any confidence. The complexity of any desktop operating system means that it would take several multiples of the software’s intended service life to determine with any reasonable confidence the chances of the software failing and why (Pan, 1999). An estimate of the software reliability can be extrapolated from the computer’s performance utilising the final version of software however this is not necessarily an accurate indication of the mean time until failure due to the complexity of a Windows Operating System (Stiffler, 1981). The main advantages of testing, if not to definitively determine reliability, is to test the raw logic of the software and to gain experience on what problems may be expected under certain conditions. This is the reliability for the automated software of the computer however the remote system will also have a user which creates an additional and completely undeterminable amount of uncertainty (Stiffler, 1981).

Since the specific logic of the software has been thoroughly tested the main failures to overcome are; the seemingly random ones generated by the computer, logic failures which occur under unforeseen and therefore untested conditions and those introduced by the user and their software. Even if the software logic is proven to work there needs to be enough redundancy in the system to ensure that the computer remains operational even after a failure and without the need for a physical hard reset by a user. The solution to this is to have a watchdog available to perform that hard reset. This allows for some redundancy in the system so that the probability of failure is now the probability of the software failing and the watchdog failing (Lewis, 1994). The reason a hard reset has been identified as a solution to any software glitches is that the software has been configured to automatically recover the computer into normal operation after a hard reset. Having the watchdog means that a failure which requires physical interaction with the buoy will only occur if the software fails and the watchdog fails.

The watchdog being used is a $\mu$Chameleon data logger which is functionally a microcontroller. Since this can be viewed as a solid-state piece of hardware the number of failures we can expect to experience during its useful lifetime is almost zero. The expected failure rate as a function of time is shown in Figure 11. The digital output of the watchdog is connected to a transistor which is acting as a switch. When the digital output is set to high the transistor shorts two pins on the computer which performs the reset. The watchdog uses its own internal background clock as a timer which operates at 20Hz. The 20Hz means the $\mu$Chameleon runs out of memory space before a usable time is reached. To overcome this I wrote the watchdog program so that the clock runs through cycles of 10 minutes and counts each time it reaches 10 seconds before the 10 minute mark, one second before the 10 minute mark and at the 10 minute mark before starting the 10 minute cycle again. These counts act as the timer and keep track of the “virtual” position of the background clock timer if it were allowed to run indefinitely. With reference to Figure 9, there are two types of timers; a timer for when the computer is on and a timer for when the computer is off. The timer for when the computer is on counts down once the computer wakes up. If the computer is on for too long it is reset with a 10 second pulse from the watchdog. The 10 seconds ensures that the computer shuts all the way down ignoring any command prompts. After another period the computer is then started up and the computer enters its initial start mode, discussed below. The computer off timer starts counting down when the computer is asleep for too long and tries to start the computer periodically until it wakes up. These timers are reset when the computer wakes up and go to sleep. This is a flaw in this watchdog design since it is not completely independent of the computer because of these two instances. If an error occurs right in the middle of
transmitting the timer reset the µChameleon behaviour will be unpredictable. However this is deemed relatively unlikely because the error would have to occur mid-transmit.

The initial start mode of the computer simply wakes the computer up, connects to the internet, waits for a user interruption for 30 minutes (in order to restore the computer's normal operation) and if this doesn’t occur, hibernate for 6 hours where the same process is then repeated indefinitely.

### Section V: Payload

The user’s system is required to carry two different payloads one its initial deployment. The two are; a thermistor chain for the collection of oceanographic temperature observations and an acoustic payload for underwater communications research. The physical connectors which allow external interaction whilst sealing the system from the environment's hostile effects have been covered in Section 4. There are six spots for connections on the Pelican case. One spot is taken by the power cable, one spot is taken by an Ethernet port and two spots are taken up by the sensor and research payloads. Two spots have been left empty to allow for future expansion for any additional research payloads or sensors.

#### A. Acoustic Payload

The acoustic equipment included in the research payload is detailed in Appendix L. The system was specified as a sub-assembly, with an interface consisting of a USB connection and a +12V power supply and ground to connect the main system in the Pelican 1450 case to the acoustic payload in the Pelican1300 case. The +12V power supply and the ground were duplicated on two pins on the main system connection because there was a possibility that more than 1A is drawn from the power supply and the pins were only rated to 1A.

Because of the extra load on the power supply generated by the acoustic devices a switch was implemented in order to control the power supply to those devices and save power when they were not in use. I was initially supplied with a solid state relay specifically for switching power, which came in a chip package. However the switch required the power source to cross zero before it would switch off and so would latch into an on state. This switch was to be controlled through the µChameleon whose digital outputs could be accessed directly, without having to send a script through MATLAB by either using the development software provided or hyperterminal. However generating zero-crossings on the DC supply was problematic, so an alternate solution was provided which was a “Super 4 USB Relay” from TCTEC. This is a set of four mechanical relays which are controlled via USB. From a command point of view when the computer is on the USB relays are very versatile since they take binary input in a variety of ways. However the disadvantage of having a relay controlled by USB and not the outputs of the µChameleon is that the acoustic payload cannot be easily switched while the computer is in hibernate. This was not necessarily a problem as it was outlined by the user that the research payload will only be utilised when the computer is on so the software provided with the USB relay board was sufficient for the switching of the power.

#### B. Thermistor Chain

This is the sensor payload attached to be buoy for the collection of oceanographic data, and was designed and constructed by myself as a part of this project. The actual choice of oceanographic data to observe was not explicitly specified by the user. The variable that was chosen to be measured was temperature. It was chosen because it is relatively easy to implement and was developed to be as scientifically interesting as possible by obtaining a cross section of the column of water directly beneath the buoy by measuring temperatures at multiple depths. A block diagram of the entire sensor is shown in Figure 12. With reference to Figure 12 the sensor also includes the µChameleon because it is required to directly interpret the responses from the thermistor chain. In the design and development of the system, two experiments were conducted. The first original experiment was conducted based on a temperature range of 15°C and 25°C. When the temperature range was expanded to between 10°C and 25°C the reference resistor value changed and the experiment was repeated. This range was determined by looking at expected temperatures on the diving website, Dive-Oz (Dive-Oz, 2010). Both the experiments can be found in Appendices B and C. This section displays and discusses only the results for the second experiment because they are the most relevant.

1. Selection of Sensor

Firstly a sensor needs to be selected in order to collect the data. It was determined that the easiest way to collect the data would be with a resistor whose value varied with temperature. This was chosen over a thermocouple, which produces a change in voltage for a change in temperature, due to availability of components at the time of development. A thermistor was chosen since it operated over a narrower and much less extreme temperature range than a resistance temperature detector. A thermistor with a negative temperature coefficient (NTC) would be required as these tend to operate
over lower temperature ranges. A GE NTC thermistor of type MA was chosen as the sensor to collect data for the dummy payload. The decision was based on the need for a thermistor to operate in an average ocean environment. The thermistor was chosen with these relatively few details because the performance characteristics of the thermistor were so wide. For example, it had an operating range of 0°C to 50°C (VIA Embedded Platform, 2007) which are conditions that are never going to be exceeded in the average ocean environment that the buoy would be operating in (Dive-Oz, 2010). The response time for the thermistor is 2.0 seconds (General Electric) this is much quicker than any time resolution that can be achieved while the computer hibernates, due to the memory limitation of the µChameleon, and is more than adequate for a continuous sampling scenario since water temperature is being measured.

2. Interface Sensor with computer

An ADC needed to be selected in order to recover the thermistor resistance value and transfer it to the computer. Once on the computer this resistance value can be converted into a temperature reading. The ADC needed to have multiple inputs to allow additional thermistors to be attached (in order to form a thermistor chain). It should have the option of an external power supply and internal memory so that the ADC can collect values when the computer is asleep. The eventual choice was the µChameleon. It is programmable as well as having a USB interface which automatically switches from USB to external power. It has 8 analogue inputs and an internal clock. This solution was cheap and readily available. Other options where investigated and the alternate was determined to be specific data loggers for sensor observations. Whilst these were more ideally suited their size and expense were not suitable for the project. In addition to this there was a large amount of redundant complexity in these systems since they are designed and built for more detailed applications.

By using a simple voltage divider, Figure 13 consisting of a known resistor value, the thermistor and a known excitation voltage the thermistor value could be determined. This voltage divider was labelled as “the interface”. The interface was chosen to be a voltage divider rather than a Wheatstone Bridge. A Wheatstone Bridge would have provided much greater deflection in voltages across the resistance (and thus temperature) range that I am interested in since it is a differential setup. However the µChameleon measured input voltages with reference to its own ground, so using the µChameleon to measure the voltages across the resistors in a Wheatstone Bridge would have resulted in the same voltage deflection. This is because the ‘differential’ mode of the µChameleon ADC simply takes two ground reference measurements and subtracts them in software.

3. Interface Design

The interface now needs to be designed in order to maximise resolution and accuracy in the circuit. First there were two considerations that needed to be taken into account. To determine the thermistor resistance from the voltage divider the excitation voltage needs to be known. The interface is powered directly from the battery voltages. This voltage can vary by almost 4V between being fully charged and being unable to power the computer and its peripherals so this value would also need to be measured by the µChameleon in order to collect accurate temperature data. In addition to this the input range for the µChameleon is 0V – 5V. This means that the excitation voltage (the battery) would need to be scaled down from its actual voltage using a voltage divider.

The maximum voltage being provided by the batteries will be 13.5 in a worst case scenario due to the regulator attached to the solar panels which is charging them. Knowing that 5V is the maximum allowable input for the µChameleon and that the maximum battery voltage will be 13.5V the values for \(R_a\) and \(R_b\) which will make up the voltage divider to scale the excitation voltage was determined as below:

\[
\frac{V_{IN}}{V_{IN}^*} = \frac{R_a + R_b}{R_b}
\]

(Equation 1.1)

For a maximum input voltage of 13.5 which is to be scaled down to a voltage no greater than 4.5:

\[
\frac{V_{IN}}{V_{IN}^*} = \frac{13.5}{4.5} \geq 3
\]

\[
\therefore A \geq 3
\]

(Equation 1.2)
With reference to Equation 1.3 this is only true if A is equal to 3. Equation 1.1 should be used to ensure that the resistor value combination keeps the value of A \((V_{IN}/V'_{IN})\) greater than 3. From this the values of 2 k\(\Omega\) for \(R_a\) and 1 k\(\Omega\) for \(R_b\) were chosen. A detailed derivation can be found in Appendix A. The next design consideration is the value of the known resistor for the different thermistors which will determine the range over which the measured voltage will vary. Since the temperature range for each of the thermistors is very similar only the one reference value is needed to be calculated and this is designated \(R_R\) in the equations. To optimise \(R_R\) a value needs to be chosen which produces the biggest difference between expected maximum and minimum voltage values while remaining below 5V. Dive-Oz (Dive-Oz, 2010) was used a reference for the expected water temperatures in Jervis Bay. Throughout the year the water temperature on the surface is expected to vary between 16\(^\circ\)C and 22\(^\circ\)C. Based on this a range of between 10\(^\circ\)C and 25\(^\circ\)C was chosen for the sensor system. This meant that \(R_T\) would vary between 19.8983 k\(\Omega\) and 10.0000 k\(\Omega\). Since the thermistor has a NTC the resistance increases with a decrease in temperature. From the voltage divider, the expression which determines the thermistor resistance, designated \(R_a\) through to \(R_e\) but expressed as \(R_T\) in the equations, is shown in Equation 2.1. This can then be rearranged to express the voltage drop across the thermistor as shown in Equation 2.2.

\[
R_T = \frac{R_R V_T}{V_{IN} - V_T}
\]

(Equation 2.1)

\[
V_T = \frac{R_T V_{IN}}{R_R + R_T}
\]

(Equation 2.2)

Figure 14: Plot determining optimal reference resistance

The plot Figure 14 shows the variation of the measured voltage \(V_T\) as the thermistor varies between 19.8983 k\(\Omega\) and 10.0000 k\(\Omega\) in the “worst-case” scenario of \(V_{IN}\) having a value of 13.5V. This appears as single line trace. This was then repeated for several different values of \(R_R\). With reference to Figure 14 the higher the value of the reference resistor the steeper the curve. This means that a plot as far to the right as possible would be ideal, provided that the maximum voltage isn’t more than 5V in order to provide maximum deflection across the temperature range I am interested in. From the plot a reference resistor of 34 k\(\Omega\) was selected. In order to save space on the board which carries the interface circuit, only one reference resistor was to be used. Since the exact resistance value of 34 k\(\Omega\) was unavailable and rather than trying to achieve it with resistors in parallel or series, the next best available resistor value of 39 k\(\Omega\) was chosen. This value was higher than the originally selected resistor in order to ensure that the maximum expected voltage isn’t more than 5V.

An experiment was conducted, found in Appendix D, in order to determine the functionality of the sensor. The results are shown below in Figure 15. Figure 15 shows a plot of the output voltage from the chameleon’s ADC over time. A source of heat was applied as consistently as possible and the voltage value recorded at certain times. The thermistor was then allowed to stabilise and the same was done using a cold source. The
thermistor was found to be very responsive and even a small change in the amount of heating or cooling being applied resulted in an observed trend change.

4. Characterising the System

In order to determine the resolution and accuracy of the system a relationship between the change in resistance and the corresponding change in temperature and the relationship between the change in $V_T$ and the corresponding change in temperature.

The change in temperature against the change in resistance obtained from the thermistor data sheet (Appendix N) was plotted and then an expression for the relationship was found in MATLAB over the relevant temperature range using the command POLYFIT(X,Y,N). The result can be seen in Figure 16. The relationship between temperature and resistance was found to be:

$$k\Omega = 0.0160. temp^2 - 1.2177. temp + 30.4040$$

This means that for a 1°C change in temperature results in a varying change in resistance values since the relationship is not linear, with reference to Figure 17. For resolution and error calculations the extreme values are relevant. So at most there is a 0.9128 kΩ (912.8 Ω) change in resistance and at least a 0.3994 kΩ (399.4 Ω) over the temperature range I am interested in.

$V_T$, which is being used to determine $R_T$ is being registered by the µChameleon which uses an 8-bit ADC. The full range of the ADC is 256 discrete levels over the voltage range of 0V to 5V. This means that a voltage change of 0.0195 can be detected by the ADC. From the experiment we can determine the relationship between $V_T$ and $R_T$, reference Figure 18. Using polyfit again, this time to find a linear relationship, the equation of the line in Figure 18 was found to be:

$$R_T = 5.0063 \times V_T - 6.5641$$

This has been plotted over the experimental results, reference Figure 18. A change of 1V across $V_T$ means the thermistor resistance has changed by 5.0063 kΩ.
5. **Resolution of the Sensor System**

When asked what resolution desired for a thermistor chain Oceanographers replied that better resolution is always preferred, the question then, is useful resolution provided by a low-cost sensor? Observing a system collecting oceanographic temperature readings in Manly for the Many Hydraulics Laboratory (Department of Environment and Climate Change, 2010) and using their data as a reference a temperature resolution of 0.2°C is deemed sufficient for this project since that is the resolution they are working on, and such a resolution clearly shows the temperature cycles in the ocean.

Looking at the characteristics of the sensor system the relationship between $V_T$ and $R_T$ and the relationship between $R_T$ and temperature was established. Combining the two relationships and using the known resolution in volts the resolution of the system in degrees Celsius can be determined. Given a change of $97.62285 \, \Omega$ at the thermistor there is a 0.0195 V change in $V_T$, the resolution of the system in volts. This corresponds to a temperature change (taking the maximum value for changes in temperature for resistance) of 0.244°C. This is just over the required resolution of 0.2°C and it is clear that any change in resolution would be achieved by increasing the number of bits in the ADC.

![Figure 17: Change of resistance for change in temperature over possible resistance values](image1)

![Figure 18: Change in resistance for change in voltage](image2)
6. Error of the Sensor System

Taking the change in Ω for a change in 1°C determined above, this translates to a temperature change of between 0.0011°C and 0.0025°C for an increase of 1 Ω. In order to determine the error of the system, several resistors with measured resistance were attached to the circuit in place of a thermistor. Using the readings from the ADC the resistances were then calculated and the difference between the true resistor value and the resistor value as determined by the μChameleon were recorded. The maximum difference was found to be 83.3 Ω. Using the known resistance to calculate the expected temperature and comparing that to the temperature obtained using the determined resistance the difference was found to be 0.2046°C. This is less than the resolution of the sensor which means the error would have very little effect on observed temperature. This error would have to be reduced to half the resolution of the system for it to be negligible. This error would have to be reduced to half the resolution of the system for it to be negligible. This is the maximum error found in three tested resistance with the other two errors being 58 Ω and 34.3 Ω which produce a difference between the expected and determined temperature of 0.0849°C and 0.0623°C. These errors and the comparison between the expected temperatures are shown Figure 19. A fourth resistor was tested, however the error was found to be 1.3 kΩ. Upon further investigation it was found that the ADC on the particular μChameleon being used didn’t register inputs all the way up to 5V. The ADC output was 255 at an input voltage of 4.69 V. This seemed to be specific to the particular μChameleon and means that the range of the ADC shouldn’t be assumed to be 0 V to 5 V and is something that should be checked in order to achieve the required temperature range.

This experiment served as only an indication of the general error to be expected from the sensor. To determine the maximum and average error over the entire range of temperatures the experiment should be repeated using resistors of values incrementing by the resolution of the system.

7. Environmental Temperature Readings

During the software testing temperature observations were also collected. Figure 20 are the temperature observations between 23 July and 6 August from the roof of Building 15. The temperatures are obviously not consistent with the time of year, as shown in Figure 20. Whilst the error between the actual resistance and the resistance determined by the sensor system is relatively small, there was an error in the interpretation of temperatures from those resistances. This means that there may be additional error associated with determining temperatures from resistance however there is no means of determining this without some form of calibration.

![Figure 19: Comparing the temperatures using the true resistance and the determined resistance](image-url)
The latest set of temperatures taken has been plotted in Figure 21 and shows the temperatures as recorded from each of the five sensors over the period indicated on the axis. The unusually high temperatures shown in Figure 20 were a result of a calculation error which was then rectified. The last set of measurements, plotted in Figure 21, was then overlayed with Bureau of Meteorology observations from Canberra airport in Figure 22. This comparison is a test of general trends since the two locations for the observations are different. Figure 22 has the BOM data adjusted slightly so that a better comparison can be made between the two sets of measurements. The actual temperatures are irrelevant since the two locations are different and have different factors influencing the temperature. For example, the weather station at the airport is open and these temperatures were taken from a covered area at the back of the Civil Engineering lab. The BOM data was shifted upwards by 4°C simply to create a better visual comparison of the two data sets. Referencing Figure 22 the general movement of the temperatures as well as the subtler features of the two plots are very similar, which is verification that the buoy’s temperature sensors operate as designed. Because the temperature sensors on the buoy have not been fully calibrated the observed temperatures at the back of the Civil Engineering building may or may not be 4°C warmer at the warmest time of day than the Canberra airport observations. Calibration was not done in this project because the sensors primary purpose was a test of functionality, which Figure 22 validates, and a sufficiently accurate and appropriate weather station or similar was not available.

![Figure 20: Observed temperatures over 23 Jul – 6 Aug](image-url)
Figure 21: Collected temperature data

Figure 22: Collected temperature data compared to BOM observations over the same period
VI. Software

The software architecture was specifically designed to address the user requirement of remote access to the Windows XP OS. It was also used to implement the power management scheme as well as the collection and distribution of data. This section provides a summary on the software architecture used on the gateway buoy. Each of the functional requirements that needed to be meet by the software could have been fulfilled using externally developed programs. However this would have raised complexity by having so many different programs trying to communicate with each other as well as attempting to time their interaction with the computer. For this reason batch files were used as the main structure for the program architecture. For specific needs the languages of Java, AutoHotKey, Visual Basic, Basic and MATLAB were also used. The develop of the software, utilising MATLAB and Windows Command Line was greatly assisted by the information provided on the Microsoft support page (Microsoft) and the MATLAB MATHWORKS forums (MathWorks, 2010).

The file structure on the host node computer is as follows and is displayed visually in Figure 21; all files relating to the operation of the node are located on the C: drive under the name of the node with the exception of the initial script which is located in the Startup folder of the computer so it can run when power is first applied. In the case of the developed system it has been named MasterNode so the location is C:\MasterNode. In this folder are five sub-directories and the MATLAB licence file. The CommandFiles folder contains all of the batch files, Visual Basic (.vbs), MATLAB files (.m) and java class files (.class) for normal operation. The StartUp folder which is a sub-directory of the CommandFiles folder (different to Startup) contains all of the scripts and log files specific to the initial start up of the computer. The Sensor folder contains the MATLAB .m scripts and .txt files necessary to communicate to the µChameleon and process the recovered data. The two folders NodeData and NodeLogs contain the collected data and log files of the node. Each are split into three folders; CoreLogs, DailyLogs and Outbox. CoreLogs contains the entire history of the data and log entries. DailyLogs is a temporary folder which contains the last entry for each log or data set and is concatenated into the CoreLogs folder each time they are generated. The Outbox folder contains any text files from the DailyLogs folder which
couldn’t be posted to the server. Inside DailyLogs and CoreLogs for NodeData they are further split into three folders; AdminData, RawData and SiteData which each contain specific information relevant to their category. The location of the node’s data on the server is under the name of the node so for a node named MasterNode the location is \MasterNode. Inside the node’s folder on the server are four sub-directories. The folder NodeAdministration contains administration data for the node as specified by the user. The folders NodeData and NodeLogs contain the log text files from those folders on the buoys computer. These text files are arranged in folders based on the date they were posted. The commands required for the computer to execute a specific task are all put into a single batch file, java script etc and are referred to as functions. Each function is called in sequence by the MasterControlProgram, except for the initial start up. The order in which these functions are called and executed is shown in Figure 24. Log-off is not a part of the structure because it is user initiated and simply returns the software to the hibernate state at the end of its execution.

A. Initial Start up

The collection of scripts; EngageThrusters.bat, FirstThingsFirst.bat, StartUpStatus.vbs, TimeUpAlert.vbs, TheWaitingGame.class, WaitCycle.bat and moddedWakeUp.exe all enable the user to remotely access the computer and conduct the initial set-up before the commencement of normal operations. This is achieved without physical contact with the device after the momentary action switch is first pressed. Instructions for the deployment of the system are found in Appendix E. These scripts start once the computer is turned on from the off state, connects to the internet and VPN, alert a user and wait 30 minutes after which another alert is sent, internet and VPN connections are disabled and the computer goes to sleep for 6 hours. Upon wake-up the exact same process starts again and will repeat until there is user intervention. The 6 hour sleep is an important feature since it means that the computer is not waiting for user intervention at the same time everyday, avoiding the need to greet the buoy at say 2 am. The user can just wait until later in the morning, when the buoy will wake-up at the more civilised time of 8 am. Figure 24 shows the order in which the files are executed when the PC starts from the off state.

1. EngageThrusters.bat

This .bat file is simple pointer to FirstThingsFirst.bat and will only run when the computer is turned on from an off state, not a sleep or hibernate state. It therefore only runs on initial deployment or recovery from a watchdog initiated power down as with all files under this Initial Start up heading.

2. FirstThingsFirst.bat

This .bat file establishes a connection to the internet, sends an alert to an email account and then starts the sequence of scripts to allow the user to access the computer for initial setup. It is similar to EstablishConnection.bat with three main differences. It resets the watchdog timer when it wakes up, it incorporates the 20 second delay to allow the Virgin Broadband application to open and it calls the series of .bat files rather than simply exiting.

3. StartUpStatus.vbs and TimeUpAlert.vbs

These two visual basic scripts send emails that alert a user that the buoy has turned on for the first time and that it was finished waiting and will be awake again 32 hours from the receipt of
the TimeUpAlert.vbs email. The start up email contains four attachments. The network connection and IP address information for remotely connecting to the computer, the WaitCycle.bat script so that the exact wake up time is known and the amount of time the computer will be asleep for, which is found in the AwaitTime.txt file.

4. **TheWaitingGame.class**

The WaitingGame java program generates the next wake-up time for the computer. It reads in the Hibernate.bat file and changes the line which executes the program moddedWakeUp.exe so that it will wake the computer 32 hours after it has gone to sleep.

5. **WaitCycle.bat**

This batch file has the computer wait 30 minutes before it executes TimeUpAlert.vbs and then hibernates the computer for 32 hours using moddedWakeUp.exe. Before hibernation this .bat file also resets the watchdog clock and disconnects all of the active connections. The easiest was to implement a pause in a batch file is to have the computer ping itself. The timeout is set to 1000 milliseconds so that the time to wait in seconds can be given under the number of echoes option. However it was thought that this may be causing connection issues when using VNC software. To avoid any possible complications the pause command was then implemented using a MATLAB script which accepted a pause in seconds as an input argument and then paused for that amount of time before exiting. The MATLAB file used for this is delay.m

6. **moddedWakeUp.exe**

This code was modified from the script wakeywakey.ahk and then complied into an executable file. The modification was the program run on wakeup which was changed from MasterControlProgram.bat to FirstThingsFirst.bat. This enables the computer to continuously wakeup, wait and then hibernate again indefinitely until there is some user intervention.

The script wakeywakey.ahk was modified from a script on forum (Boskoop, 2006) on the website AutoHotkey, which is a free, open-source utility for Windows (AutoHotkey, 2010). The code was modified from its original form by changing the time/date format into something more familiar and editing the program so that it runs on wakeup to the MasterControlProogram.bat. The script was then converted into an executable file using software provided on the AutoHotkey website. The purpose of the wakeywakey executable file is to put the computer into hibernate and then wake it up at the specified time and then run the specified .bat file. wakewakey takes a time and date in a specific format as its input arguments. It then hibernates the computer and wakes it up at the specified time and date. The script works by converting the input in to Coordinated Universal Time (UTC) and then setting a timer. If the computer is hibernated when the timer expires then it is woken up by calling the relevant .txt file. The script was compiled and is being used as an executable file; however the modified .ahk script has been corrupted.

B. **Wake and Hibernate Cycle**

The wake/hibernate cycle of the computer is achieved through the files; WakeUp.bat, Hibernate.bat, GenHibSamp.class and wakeywakey.exe. The computer wakes up, conducts its usual operation, generates itself a new wakeup time and then puts itself to sleep.

1. **WakeUp.bat**

This batch file was required for when the wake/hibernate cycle was being handled by Windows Task Scheduler. It was been retained for logging and debugging purposes so the time the computer has woken up can be recorded. It also has potential in the future for indicating the status of the node to the website.

2. **GenHibSamp.class (Hibernate Portion)**

The function of the java .class file outlined here is the setting of the new hibernate time. The program looks at two files. The first file contains the latest battery state which has been generated by the previous function batState. The second file contains three values which are referred to as long sleep, short sleep and battery threshold. The long sleep is the extended hibernate time for the computer to save power, short sleep is the computer’s hibernate time under normal operations and battery threshold is the battery voltage which determines when to switch from short to long. So the java program compares the last battery voltage to the battery threshold voltage and decides on a short or long sleep based on this. It would be a simple case of applying some sort of relationship between the battery charge state (approximated using the battery voltage) and the amount of time that should be spent asleep which would provide a more dynamic sleeping time, rather than the “decision” of either a short or long sleep. A more sophisticated program would allow the sleep time to be determined based on amount of sunshine, time of year, time of day etc. If the .txt file with the three variables is corrupted or incorrectly edited then the program defaults to long sleep of 8 hours, short sleep of 1 hour and battery threshold of 12V. Once the hibernate time is determined the corresponding data and time for the wake up is then produced. The Hibernate.bat file is read in and the line which runs wakeywakey.exe is replaced with the new input arguments and the new Hibernate.bat file with the revised wakeup time is read out.

3. **Hibernate.bat**

This is the batch file which puts the computer into its hibernate state. It ensures the internet connections are still active and reconnects them if they aren’t, sends and email alert, uploads text files to the server, initiates the
sampling, disconnects active connections, resets the watchdog timer and then hibernates the computer with a set wakeup time. The /wait command is used with each externally called function in order to ensure that they don’t run into each other. The batch file won’t actually exit until wakeywakey.exe has finished executing which is after the computer has woken up again.

4. wakeywakey.exe

Functionally the same as moddedWakeup.exe it runs MasterControlProgram.bat when the computer comes out of hibernate instead.

C. Data Upload

In order to externally distribute any information the buoy generates itself or any of the data it acquires there needs to be a means to make this information available remotely. This is done through the functions EstablishConnection.bat, CheckConnection.bat, ServerUploadDaily.bat, CatAdmin.bat, CatLogs.bat, CatRaw.bat and CatSite.bat.

1. EstablishConnection.bat

This code is a modified version of the code produced by Sean Reece. The code has two attempts at connecting to Virgin Mobile Broadband Internet and two attempts at connecting to the ADFA VPN. This is necessary because if either of these services are unavailable or there is insufficient signal then the computer needs to be able to stop trying to connect and carry on with normal operations. Unfortunately if there is no connection to either of these services the user has no way of knowing the status of the buoy. After a connection is established the network drive of the server is mounted. The line “net use” was an attempt to step around a windows glitch where a mapped network drive would appear unavailable to the computer (notably when navigating through the command line) until a user manually double clicked on the icon in My Computer and entered their credentials. Since this solution didn’t work and no other solution has been found the manual login of a user is a requirement of the initial setup. An email alert is sent letting the user know that the node is awake.

2. CheckConnection.bat

Performs a similar function to EstablishConnection except that it checks whether or not a connection is active first and then attempts a reconnect. Doesn’t send any email status to a user and only attempts a single reconnect if no connection exists. Performed in the event that either the internet or VPN connection drops out while the computer is on and is done to confirm an active connection before an alert email is sent.

3. ServerUploadDaily.bat

This is the .bat file which pushes the nodes files onto the server. . ServerUploadDaily uses xcopy to move specific files to the mapped network drive on the computer. It then calls several other functions which concatenate the computers logs and data files before sending them to the server.

4. CatLogs.bat, CatAdmin.bat, CatRaw.bat, CatSite.bat

All of these functions are exactly the same; there is just one for each directory. The only difference in code are the file paths. First the text files are concatenated with those files in the core directory using a variation of the Windows copy command. They get each of the text files in the directory and append to the end of the file name a time stamp. It then moves these out onto the server in a folder which is labelled with a date. If contact with the server is lost, for whatever reason, then any files which failed to get moved are then shifted into a folder named outbox. At this stage there is no provision for moving those text files in the outbox the next time the connection is restored since dropping a set of data or log files has no huge impact on performance especially if they are still available on the computers hard drive. The code in each of these files does this once for a text file with a given input (the file name). The ServerUploadDaily.bat function is what iterates through all of the text files in the directory. No exit command is required since it is being called from another .bat file.

D. Data Acquisition and Processing

The computer is also responsible for collecting and disseminating sensor data. In this initial system the computer receives data from sensor devices attached to the µChameleon data logger. The µChameleon has limited data storage and some programming constraints – it lacks loops for example. The µChameleon is programmed using code dynamically generated in a MATLAB script to evenly allocate the available memory storage across the dynamically chosen hibernate time.

For the complete data process, the computer must first send a script to the data logger which tells it what data to save and when. Then there is the recovery of this data from the data logger, which is necessary since the computer is hibernating as the data logger takes samples, storing the values on its own internal memory. Then there is the preliminary data process which is turning the values from the loggers analogue to digital converter back into actual voltages. The main data process is then converting these voltage values into relevant and useful data, in this case temperature readings. The final data process is then placing these data samples, along with any other relevant information, into a predetermined format. At this stage it is simply columns and rows however there should be a migration towards a recognised data format. There are two functions which achieve this, EvaluateTemp-MATLAB.bat and InitiateLoggerSample-MATLAB.bat however these simply call and run the
MATLAB .m files temp.m and sample.m respectively. The sampling schedule is generated by GenHibSamp.class

1. EvaluateTemp-MATLAB.bat and InitiateLoggerSample-MATLAB.bat

These are simple one line .bat files which start up MATLAB in order to run their respective .m files. When MATLAB is being run from the command line options are added so that; the desktop is not started, all java is turned off, there is no start up screen and so that the .bat file waits until the MATLAB code has finished executing before proceeding.

2. sample.m

This sends the text file LoggerCMD.txt to the µChameleon. The most important part of this code is closing the serial port once the command has been sent because otherwise that port becomes unavailable until the computer has a hard restart. The buffer size is set to always be twice the size of the script. If each of the characters are 1 byte each, which would be typical, then the buffer length would be only need to be the length of the script. Multiplying it by two ensures the buffer input is long enough for different encoding schemes and if there are several characters that are represented with two bytes.

3. temp.m

This script recovers the values of the data loggers ADC from its internal memory, converts these values into proper voltages and then uses these voltages to obtain the temperature at the time of the sample

4. GenHibSamp.class (Sample Schedule portion)

The sample portion of this java program generates the file LoggerCMD.txt which is a script for the data logger to execute. The reason this is generated in a java program is so that the data samples are taken at regular intervals across the sleep cycle. This is satisfactory for the current power management scheme (a binary decision of a short or long sleep based on the battery voltage) however will be extremely beneficial when a more dynamic approach is taken since this allows for the equal spacing of samples given any sleep time. The µChameleon basic language is very rudimentary. Simple things like a white space at the end of the line cause the sent command to fail. Each line of the text file is outputted using .println in java. This is to enable a carriage return to be present in the text file after each command, which is necessary for the µChameleon to execute those commands correctly.

5. DataProcess.class

This java file reads in all of the processed measurements from the nodes core files and then outputs them in a more readable format. It then outputs them as the file allTheNumbers.txt which is the file that will be viewed by the public. The information contained within allTheNumbers.txt and the way in which it is displayed can be altered in this java program, however in order to this will require the original java code as well as means to edit and compile it.

E. Automatic User Control and Computer Self Management

These are a group of files which allow a user to remotely control the computer, either directly or through a script or which enables the computer to manage itself. These are MasterNodeStatus.vbs, MasterNodeHibernateStatus.vbs, UserCmd.bat, LogOff.bat, IdleTime.bat, AdminUpload.bat, computerON.txt, computerOFF.txt, adminON.m, adminOFF.m, adminON-MATLAB.bat, adminOFF-MATLAB.bat

1. adminON.m and adminOFF.m with adminON-MATLAB.bat and adminOFF-MATLAB.bat

These MATLAB scripts send the command files computerON.txt and computerOFF.txt (respectively) to the administration µChameleon and are run using the batch files adminON-MATLAB.bat and adminOFF-MATLAB.bat.

2. MasterNodeStatus.vbs and MasterNodeHibernateStatus.vbs

These two visual basic scripts send emails to a particular email address. MasterNodeStatus.vbs is to alert that the computer has woken up and is performing its normal operations. It includes several attachments including the IP configuration data and network connection details so that a user would be able to access the computer remotely. The MasterNodeHibernateStatus.vbs is a simple text alert that the computer has begun hibernation again.

3. AdminUpload.bat

This is the logistical handling of the remote user function found in UserCmd.bat. Basically it fetches the UserCmd.bat file from the server, executes that .bat file, deletes it then recreates a default UserCmd.bat file. It is necessary to delete it once the file is executed, so that it is only executed once. The default file is generated because if there are no user commands to be executed then there should be no UserCmd.bat file present on the server and the default file allows MasterControlProgram.bat to continue running without throwing up a file not found error.

4. UserCmd.bat

This is the user’s commands for the computer. Simply place the named .bat file in the correct directory (\MasterNode\NodeAdministration\) on the server and the computer will execute it the next time it wakes up. The worst thing you can do is not provide an exit command, even if the computer is to wait for human
intervention there should be a timeout. Any UserCmd.bat scripts should be thoroughly tested onshore beforehand.

5. **Idle.bat**
   
   Was used to have the computer sit idle for a time but is now does nothing. May prove useful in the future since it is a spot for batch file commands in the MasterControlProgram.bat ‘tree’ and the MasterControlProgram.bat tree cannot be modified automatically.

6. **LogOff.bat**
   
   A .bat file the user runs when they are finished with any remote access. This file basically runs the last two commands of the MasterControlProgram.bat file which would have been skipped since if manual remote access is being utilised then this is where the computer would have stepped out of the systems usual sequence of events.

7. **computerON.txt and computerOFF.txt**
   
   These two text files contain the watchdog timers and also monitor the cooling for the computer. Descriptions of these two functions can be found in Section 4.

8. **caseTEMP-MATLAB.bat and batState-MATLAB**
   
   These are simple one line .bat files which start up MATLAB in order to run their respective .m files. When MATLAB is being run from the command line options are added so that; the desktop is not started, all java is turned off, there is no start up screen and so that the .bat file waits until the MATLAB code has finished executing before proceeding.

9. **batState.m**
   
   This file takes the ADC value from a µChameleon and converts this into a voltage. It then outputs this voltage as a single value in a file which is then used by GenHibSamp.class to determine how long the PC will hibernate for.

10. **caseTEMP.m**
    
    This file takes an ADC value from the thermister positioned inside of the case and converts this to a temperature value, saving it to a text file. This is simply for the interest of the user and is not connected in any way to the cooling system.

**VII. Conclusion**

The system was designed and developed within the required timeframe, meeting all of the stipulated user requirements. The system was completed, fully tested, documented and officially handed over to the user on 25 Oct 2010 for their deployment and use. The system met all of the user requirements. The major achievement was the ability to have a remote desktop PC deployed to an environment as hostile as the surface of the ocean. This is attributed to the manual remote access from any computer with access to the internet as well as a functioning watchdog circuit which can ensure that the system will continue to run after software crashes. The planned deployment date for the system is early November pending final testing of the acoustic buoy. Figure 26 and Figure 25 show the completed system and how it positioned on the buoy.

![Figure 26: Completed physical system](Langbridge, 2010)

![Figure 25: Pelican case positioned on the buoy](Langbridge, 2010)
A. Future work

The system was designed, developed, prototyped and assembled with the timeframe of one academic year. Whilst the system has been thoroughly tested, there is still ample scope for further maturity.

1. File Transfer

The current system employs Windows “move” commands for the upload of its data. A more dynamic and elegant way of achieving this, especially for cross-platform considerations, would be Secure File Transfer Protocol using a program such as Putty. This was unachievable during the project due to ICTS regulations. Security is not an issue for this method since the VPN accounts for that.

2. More Dynamic Hibernate/Wake Cycles

The current system uses a “hard” decision of either a long or short hibernate cycle for power management. Ideally the system should estimate the amount of charge left in the battery (possibly by also measuring current from the battery although this would need to be physically added) and then adjust the hibernate cycle according to that determined value. The system would then be able to give the user an estimate amount of continued use available before the system needs to be placed into hibernate. If this is implemented the system should also take a user specified hibernate time in order to facilitate a precise wake-up time.

3. Executable files

The MATLAB scripts are currently being run through MATLAB by calling them from a command in a batch file. Opening MATLAB, even as a console requires a large amount of unnecessary computational power. In order to avoid this the MATLAB .m files should be compiled into executable C programs.

4. Data Format

The java file which outputs the data should be altered to output the data in a recognized format as well as the current format. A suitable standard needs to be selected and then followed. The organizations discussed in this paper are a good indication of where to begin researching possible standards.

5. Sensor Standard

Investigation needs to be done into possible standards to follow with regards to sensors. The μChameleon has the option to act as a serial interface and this should be utilised. Once a sensor standard is met by the system, different types of sensors can be bought commercially off the shelf and integrated into the system with relative ease, increasing the payload of the buoy.

6. Communication Options

A wireless device should be attached and implemented so that the buoy can be communicated with from a boat pulled up alongside. In addition to this a GPS should be installed and be made to transmit its data. This is to allow easy management of multiple buoys more than one is deployed and can be included with the oceanographic data collected. It also allows tracking of the buoy it comes adrift from its anchor.

7. Calibration of Sensor

The sensor payload of the buoy should be calibrated so that the information produced is not just an indication of temperature trends but can be used as usefully and accurate data. This may require correspondence with an accredited testing laboratory.

8. Improved cooling management

A profile on the temperature inside the buoy and the case as well as the effect of the cooling system on the internal case temperature needs to be established from data obtained during operation. This will allow more suitable values to be chosen for the turning on and off of the cooling system. This will require a thermistor outside the case but inside the buoy and more frequent temperature samples to be taken and a record of when the cooling system turns on and off. Secondly the control of the cooling system should be related to a temperature, not just the ADC value as it is at the moment, without the computer being on to process the results.

9. Improved data handling

The saving of data in the μChameleon should use all 16 bits of each variable. This will require bitwise operations in MATLAB. This will allow the sampling rate to be doubled.

Acknowledgements

The success of delivering an accepted project which meet the original need and fulfilled all of the requirements of the Underwater Communications Group would not have been possible without the assistance of several individuals for which I offer my sincere gratitude. To my favourite engineering colleagues (you know who you are); your support, advice, distractions and general humour kept me sane throughout the year. To Shaun Reece; for convincing me to take up this project in the first place. To the occupants of the 4th year thesis room; for tolerating the woodwork (and accompanying sawdust) and minor flooding with only verbal abuse. To my family; for nodding and agreeing with all my ideas, plans, problems and gripes without fully knowing what I was talking about. And finally to Craig Benson, my thesis supervisor; it cannot be understated the value of your supervision. Your technical expertise and industry knowledge is the reason this project was finished and to such a high standard. In addition to this working with you has been very enjoyable, I hope my future working relationships are as half as rewarding as this one was.
Bibliography


General Electric. *NTC Thermistors: Type MA*.


Marine Parks Authority NSW. (2008). *Natural values of the Jervis Bay Marine Park*. Marine Parks Authority NSW.


