FISHMO
Semi-Automated Aquarium Monitoring System

Software Test Report
Revision 2

April 9, 2015

Group 5

Rhett Amin - 1060085
Brandon Da Silva - 1057434
Dan Esteves - 1152328
Michael Liut - 1132938

Prepared for:
Computer Science 4ZP6 Capstone Project
Instructor: Dr. Rong Zheng
Fall/Winter 2014-2015

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide insight into the design and construction of the FISHMO SAAMS product. This is done by simple decomposition of FISHMO SAAMS into its three major sections: The Hardware, The Back End, and The Software. Once explained, further decomposition of each component will be discussed to analyze the installation, implementation, use, design, and all issues with their corresponding solution that arose during the development of the system.

1.2 Design Principles and Structure

The structure of this document is closely following the example provided by Dr. Zheng in lecture. Further more, we are specifically referencing Avenue: Example ¿ DesignDoc.

The design principles applied herein are those of Dr. Donald Norman, the director of The Design Lab at the University of California, San Diego; whose principles include: affordance, consistency, constraints, feedback, mapping and visibility [4][9].

1.3 Definitions, Acronyms, and Abbreviations

Term	**Definition/Acronym**
FISHMO SAAMS	FISHMO Semi-Automated Aquarium Monitoring System
Pi	The Raspberry Pi is a credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word-processing and games.
User	A person who interacts with the FISHMO system
NOOBS	NOOBS (New Out Of the Box Software) is an easy operating system install manager for the Raspberry Pi.
GUI	Graphical User Interface
Camera Board	The Camera Board is a small PCB that connects to the CSI-2 camera port on the Raspberry Pi using a short ribbon cable. It provides connectivity for a camera capable of capturing still images or video recordings.
3D Printer	A process for making a physical object from a three-dimensional digital model, typically by laying down many successive thin layers of a material.

1.4 Revision History

Revision #	**Revision Date**	**Description of Change**	**Author**
0	January 13, 2015	Initial Design Document	All members
1	February 2, 2015	Edited feedback	All members
2	April 9, 2015	Final Design Document Revision	All members

2 Hardware Documentation

2.1 Introduction

The tangible component of FISHMO SAAMS is the hardware end user's receive as part of their "all-in-one" monitoring system. This "all-in-one" system is built from scratch and therefore besides the Pi requires various additional parts which all have to be brought together to create a final product. These additional components are chosen specifically to create a safe, affordable and efficient system. These components along with the Pi are listed below with their specifications.

Hardware Specifications

The construction of the FISHMO SAAMS is only possible with the aid of various hardware add-ons.
Components: Raspberry Pi B, PCB, Camera, Wireless-N Adapter, Resistors

Sensors: Water Flow Sensor, Thermometer

In detail these components include:

- **Raspberry Pi Model B+:**
  - Memory: 1 GB Ram
  - Power: 3.5 W (model B+)
  - Processor: ARM Cortex-A7 Quad-Core 900MHz
  - OS Type: Linux (Raspbian)
  - Storage: MicroSD card, 4GB eMMC
- **Water Flow Sensor: Model SEN02141B:**
  - Mini. Working Voltage: DC 4.5V
  - Max. Working Current: 15mA(DC 5V)
  - Working Voltage: 5V24V
  - Flow Rate Range: 160L/min
  - Load Capacity: 10mA(DC 5V)
  - Operating Temperature: 80
  - Liquid Temperature: 120
  - Water Pressure: 2.0MPa
- **Waterproof LED Light Strip:**
  - Length: 16.4ft./5 Meter
  - Working Voltage: DC 12V
  - Working Power : 4.8 W
  - Number of LEDs: 300 LED lights
  - 3M adhesive backing
  - Life: 50,000 hours
- **Heater:**
  - Voltage: DC 7.5W
  - Optimal Tank Size: 2.5 gallons
- **Water Pump:**
  - Voltage: DC 5W
  - Pump Size: 5 gallons
- **Thermometer: Model DS18B20:**
  - Usable temperature range: -55 to 125C (-67F to +257F)
  - Resolution: 9 to 12 bit selectable
  - Uses 1-Wire interface: requires only one digital pin for communication
  - Query time: less than 750ms
  - Power: 3.0V to 5.5V
  - Unique 64-bit ID
- **Wireless-N Adapter:**
  - External USB WLAN Adapter
  - Wireless Standard: 802.11 a/b/g/n
  - Speed: 150 Mbps

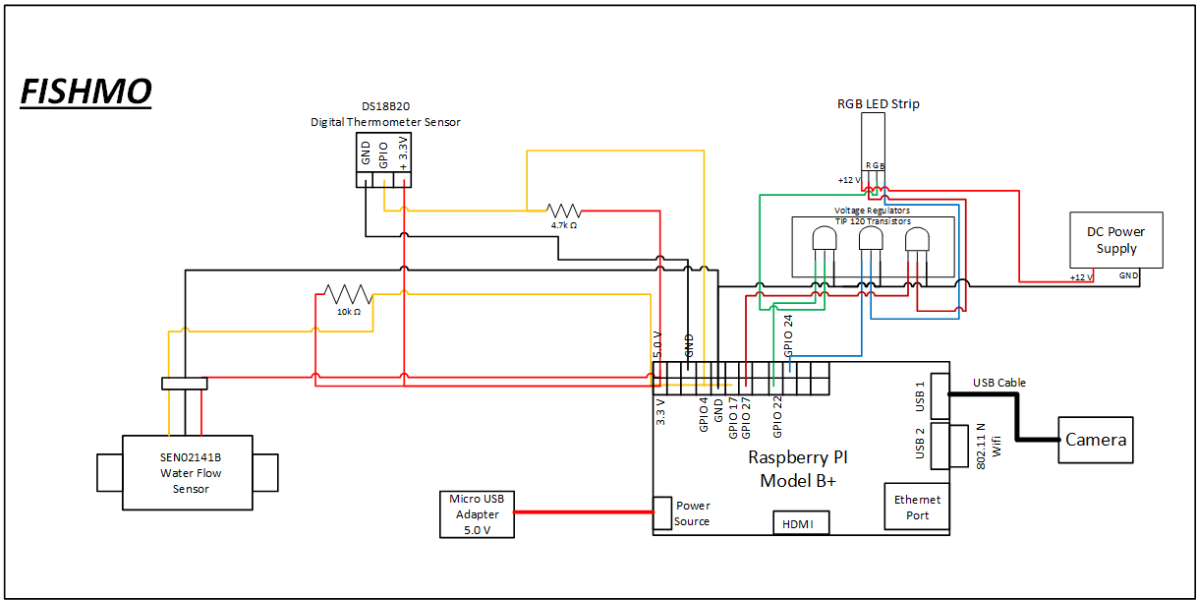


Figure 1: FISHMO Wiring Diagram

2.2 Installation and Configuration

2.2.1 Installation

Any good mechanical system requires thorough testing for accuracy and correctness; FISHMO SAAMS is no exception to this. However, prior to the testing stage, it was crucial to integrate the peripherals to the Pi via some communication channel. In this case, a solderless breadboard with 400 pins was used. This breadboard was used to simplify testing, configuration and finalization of all power, signals and ground lines throughout the circuit. After the circuits passed the testing stage they were deemed to be final, whereby, the circuits were commissioned to transfer onto a permanent printed circuit board. These completed circuits were removed from the solderless breadboard to free space to test the next components in-line for configuration and then manually soldered to the printed circuit board.

2.2.2 Configuration

The configuration stage is the most challenging and important processes in the Hardware section. Aside from the physical construction of the Pi and its enclosure, this stage requires the setup of the Pi and its peripherals to be flawless. This includes: manually editing the Pis network settings, downloading and installing the correct packages for all Pi drivers and peripheral devices, and setup of the administrator accounts. Refer to Appendix H. A to view the manually download of driver package, extraction and compilation. After the appropriate extracted drivers for the hardware go through these step,we then refer to Appendix H.B where they are then installed and an avenue for communication is created.

Once the software and hardware integration was completed, with respect to communication channels between the Pi and the peripherals, information was easily able to travel back and forth between devices. This was accomplished by configuring the Raspberry Pi as a server, where the Pi was able to communicate with the different peripheral components.

A great peripheral example is the Raspberry Pi alongside the water-flow and temperature sensors, whereby they were required send and receive all data through the Pi. This was achieved through downloading and installing package, initiating a connection with the components, and manually scripting the necessary Python code to correctly read the data sent by the sensors.

2.3 Usage

The purpose of the FISHMO SAAMS is to give its customers the ability to easily and effectively monitor the state of their aquarium from anywhere in the world with a computer and an internet connection. The only way to to

achieve this goal is to tend to the aquarium in the same manner that an owner would using a combination of custom hardware and software.

This means that the hardware would need to determine all environmental states of the aquarium that an user would want to know, report all of this data to the user in real time, and provide the opportunity to modify any aquatic settings that the user sees fit. This will eliminate the manual tedious and repetitive process of taking readings and observing the aquarium for daily maintenance.

Through correct implementation, FISHMO SAAMS will provide users with a better alternative to manual maintenance of their aquariums.

2.4 Error Handling

While designing FISHMO SAAMS it was important to take into consideration the diversity of users considering utilizing our product. Many aquarium owners do not have any experience with hardware and/or software to be able to handle errors on their own. For this reason, we aimed at creating an error free experience. With regards to the hardware, there has to be precautions in place to account for any error that could interrupt the users experience.

For example, there is a very good chance that cables do not get plugged-in properly or get disconnected, if this is the case then the data will not be transmitted to website. We have caught exceptions by notifying users that there is no connection between the Pi and that peripheral; prompting them to check the connections manually. The system runs independent of all peripherals, therefore it is not dependent on all connections being enabled so if the user accidentally disconnects an external device, the system will continue to run and allow the user to modify settings. This is the most effective approach as it does not compromise the system and concurrently provides feedback to users, giving them the utmost control.

2.5 Problems Encountered and Resolutions

2.5.1 External Power Supply

With the Raspberry Pis limited powering capabilities, external power was required for the water pump, heater and RGB LEDs. It was observed that as additional components were added to the circuit, the performance of each individual device deteriorated. This issue was directly related to the Pis inability to increase the outputted power per channel. As only a small fraction of the power was being provided to each device, they were unable to perform at their optimal levels.

At this point in time we began to research external power supplies as an alternate solution to the issue. However, we soon began to realize the conundrum of using these power supplies as configuring them with the Pi and bridging the already incoming current was difficult to say the least. Our solution at this point consisted of the following two stages: one, separating the power, two, separating the signal lines from each of the devices. This provided us with the capabilities to directly connect and control each of the devices to and from the Pi, while supplying the peripherals with the proper amount of power.

2.5.2 Unconventional Wiring

While trying to add the RGB LEDs to FISHMO SAAMS, an issue arose while trying to complete the installation of the product. As the LEDs drew far too much power from the Raspberry Pi, the current wiring scheme for the LEDs soon became a cause for concern. In standard practice, for these devices it is usually safe to assume that black coloured wires are ground, while red would imply power. However, the device that was sent to us was not constructed in this manner.

While affixing the LEDs to the Raspberry Pi according to our assumed correct configurations, within five seconds of powering the device sent a 12V surge of power directly to the Raspberry Pi board, short circuiting the unit.

It was not until this point in time did we discover that black was power while red was one of 3 signals used to control the red component of the RGB light spectrum. At this point in time, a replacement Raspberry Pi was ordered and extensive testing was completed on the RGB LED strip at McMasters IEEE laboratory. Furthermore, the schematics for the wiring were re-written to the correct specifications used in the current diagram and setup of FISHMO SAAMS today.

2.5.3 Signal Edges

A major issue encountered while configuring the Pi with the water flow sensor was correctly interpreting the data sent to the Pi. As the water flow sensor does not send numerical data, instead it constantly sends pulses (waves) to

the Pi. This caused serious problems when trying to read and interpret the data, as the pulse would send a signal every half second, communicating the current position of the sensor, rather than the water current. This sensor in particular would be in one of two states; 1 (high) or 0 (low). By receiving 0's and 1's even when there was no rotation (as it automatically returned the state is was in every half second) we could not use the data as a method of counting full rotations.

This was solved by determining the number of full spins that the sensor recorded, by using a process of detecting rising and falling edges. A rising edge is the transition in the signal from low to high. By detecting this, it is clear that it would represents half a rotation in the sensor and therefore can be ignored. A falling edge is the transition in the signal from high to low. When a falling edge occurs, the rotation completes itself. When a falling edge is triggered the clock signal changes and we record a rotation. Using this algorithm, the Pi is able to calculate the number of rotations in a short intervals to produce an instantaneous velocity of the water flow.

2.5.4 Waterproofing Epoxy

Another problem faced while finalizing certain hardware components of FISHMO SAAMS was giving the system the ability to be submerged in water. This would provide our users the ability to make the system more visually appealing and allow larger aquarium owners to use this device.

Water is detrimental to FISHMO SAAMS, therefore having a system which can be completely destroyed by the environment it was built to monitor is extremely counter productive. In order to increase the longevity of the system it was determined that FISHMO SAAMS must be waterproofed as it is prone to water damage.

The solution to this was a waterproof epoxy. This waterproof epoxy would be applied to all exposed components and all other regions prone to water exposure, protecting the Peripherals and the FISHMO SAAMS system.

2.6 Features to be Implemented

All features set out in initial scope have been implemented into the FISHMO SAAMS system at this point in time. There are currently no features which are scheduled to be implemented.

3 Back-End Documentation

3.1 Introduction

For this project, a server was assembled and configured to host the web application and database for FISHMO. With this, there had to be some configuration of the server to fit the needs of this project. The server had to be able to host the web application, database and be able for the group members to upload and download to and from the server to work on it. This needed to be done with the ability to upload to the server from the group member's personal computers, as well as keeping a log as to who uploaded the content for submission on the Github repository.

The following sections provide an in-depth look into how this functionality was installed, configured and how it is used by the team to get things working. There are also descriptions of the types of software and the specifications of the hardware to gauge an idea of what we are using and how our back-end is being implemented.

3.1.1 Server Specifications

The server was built using an old desktop computer with the specifications outlined below. The previous desktop was used for personal use and was running Windows XP. Since it was not being used, the hard drive was formatted and Ubuntu Server was installed along with the specific packages needed for the purpose of our application which will be outlined further in this design document.

- **Hardware Information:**
  - Memory - 1003.3 MiB DDR2
  - Processor - AMD Athlon(tm) XP 2600+
  - OS Type - 32-bit
  - Hard Drive - 80GB capacity
- **Software Information:**
  - OS - Ubuntu Server 14.10
- **Installed Packages:**
  - LAMP Server
  - OpenSSH Server

3.2 Installation and Configuration

When installing Ubuntu Server, the installation disk includes all of these extra packages that can be installed at the same time as the operating system. When installing the operating system, a prompt during the installation asks which packages that are included in the operating system disk image are to be installed alongside the operating system. Here we are able to install the LAMP Server and OpenSSH which are the two main packages that are being used for this project.

Other setup and configuration that was not included in the initial installation of the server is also outlined in this section including SFTP and port mapping for group members to gain access over the internet and for hosting the web application over the domain. This additional setup is necessary to gain access to our server over the internet with added ease of use.

3.2.1 LAMP Server

In order to host a website on Ubuntu Server, the LAMP Stack must be installed. The LAMP Stack represents an acronym for a model of web service stacks: Linux, the Apache HTTP Server, the MySQL relational database management system, and the PHP programming language [7]. With this installed, this gives the server the capability to host a web application which is within the scope of our project.

When installing the operating system, the prompt to install other packages allowed us to install the LAMP Server to the server. This avoided having to install it after the operating system installed on the server. When this stack was installing, the MySQL root user and password was configured and the database was set up via prompts displaying on the screen during installation.

The Ubuntu help community also provides a step-by-step guide for installing and configuring the LAMP Server on your own server (`https://help.ubuntu.com/community/ApacheMySQLPHP`). Using the guide provided by Ubuntu in the link will show how to install the standard LAMP Server but will also go in depth into each of the components if you require a more advanced configuration. The guide is for configuring the LAMP Server if it wasn't done when installing the OS initially.

3.2.2 OpenSSH

OpenSSH is a set of computer programs that provides encrypted communication sessions over a network using the SSH protocol [11]. This allows for group members to communicate securely with the server over the network.

With this the group members can securely connect to the server to navigate and operate it. It also provides the file transfer protocol SFTP to securely transfer files to and from the server which is explained below.

3.2.3 SFTP (Secure File Transfer Protocol

In order for the group members to access the server remotely, OpenSSH was installed, which has already been discussed. Within OpenSSH, it includes SFTP (Secure File Transfer Protocol) which allows for users with access to the server to upload their files securely to the server over SSH. With installing OpenSSH, this functionality is already included, it's just a matter of configuring it.

In order for multiple users to have access to the server via SFTP, user permissions need to be set in order for the group members to access the server. The command arguments in Figure 2 allows to add users and give them permissions to upload their files to the server. The way this does it is by having a folder on the user's account on the server point to the roots "/var/www/" folder [3]. In order to assign these privileges, it must be done as the root user.

```
useradd someuser
mkdir -p /home/someuser/www
mount --bind /var/www /home/someuser/www
chmod g+s /home/someuser/www
chown -R someuser:www-data /home/someuser/www
setfacl -d -m g::rwx /home/someuser/www
```

Figure 2: Command arguments to give user permissions to edit the /var/www/ folder on the server

In order for the folder to stay mounted on the server when you restart it, a file must be edited to keep the mounts persistent. In Figure 3, it shows the necessary changes to make to the /etc/fstab in order to keep the mounts persistent for all users (replacing "someuser" with the username of the group members).

```
/var/www /home/someuser/www none bind 0 0
```

Figure 3: Code to add to /etc/fstab to keep mounts persistent

There is an add-on for Sublime Text that makes it easy to have files uploaded to the server via SFTP by connecting to the server through Sublime. This is outlined in Appendix B.

With standard SFTP, it does not automatically keep a log of user activity. In Appendix C, it is outlined as to how a log has been kept to keep track of this activity.

3.2.4 Port Mapping/Forwarding

Having all of this functionality installed on the server is very beneficial for the purpose of the application but it can also be useless if the server is not connected to the internet for access from anywhere. In order to account for this and have the various services provided by the server available from anywhere, the IP address that the server is hosted on must map the ports accordingly for the features. The tutorial found in [12] provides a how-to for configuring port mapping with previous Apple Airport Firmware Versions but the same steps can be taken with the new Apple Airport Utility application and the new firmware.

The purpose for this is to have access via SSH and the web through the domain name we acquired for this project (`fishmo.ddns.net`). In order for this domain to be routed to the IP address of the server, the dynamic global hostname on the main router connected to the router needs to be set. Since we acquired the domain name from `noip.com`, the credentials from the account holding the domain name will be the credentials used for the dynamic global hostname login on Airport Utility (see Appendix D and E for how to set up this on Airport Utility)

An alternative to assigning a domain name to an IP address by signing in with a dynamic global hostname like previously explained, it is possible to purchase a domain from a provider like GoDaddy and routing the DNS services using their online interface. With GoDaddy, it allows a user to route the domain to your IP address which is ideal if the user is using their own server to host or have other hosting options that they prefer.

From here all that needs to be set up on the local network is physically mapping the ports to the server. Since we are only using SSH and web on the server, only two ports need to be mapped. Apple's Airport Utility provides an easy solution for configuring these ports by selecting which service you want to configure as shown in Appendix F. By selecting which service you want to port to a specific IP address, Airport Utility automatically configures the appropriate ports for the service that the user wishes to configure. All that needs to be determined is the local IP address of the server in the IP field [1].

Table 1 - Settings to set up port forwarding for SSH and the Web Server

SSH Port Forwarding	**Web Server Port Forwarding**
From dropdown, select "Remote Login - SSH"	From dropdown, select "Personal Web Sharing"
Specify the local "Private IP Address" that we have made static	Specify the local "Private IP Address" that we have made static

Restart the Airport Extreme in order for changes to take effect

In order for the forwarded ports to stay consistent with the IP address of the server at all times, the IP address of the server needs to be made static on the local network. This ensures that when accessing the domain of the application or when using SSH with the domain you are directed to the server on that local network. To configure the server to have a static IP address, the "/etc/network/interfaces" file must be edited to account for this. The following steps outline how to accomplish this:

- Make a backup of the interfaces file that contains the IP settings
  - `sudo cp /etc/network/interfaces /etc/network/interfaces.backup`
- Next edit the file in order to specify the static IP address
  - `sudo nano /etc/network/interfaces`
  - ```
    auto eth0
    iface eth0 inet static
    address 10.0.1.6
    netmask 255.255.255.0
    network 10.0.1.0
    broadcast 10.0.1.255
    gateway 10.0.1.1
    ```
- Restart the server for the changes to take place

Since the server is connected to the network via Ethernet and there is not a wireless network card installed, only the ethernet settings were changed to accommodate the static IP. Also, only the root user was able to configure the file.

With the appropriate ports mapped and using either method for DNS (depending on the service used), the server will be able to be accessed from the internet accurately. The networking hierarchy of the server used for our application is shown in Figure 4. This shows the current state of the server on the local network it is set up on as well as showing the appropriate information with the routes someone would take to access certain services of the server.

Table 2 lists the current hardware and software currently in use and what was used to configure the server to the network.
    ```

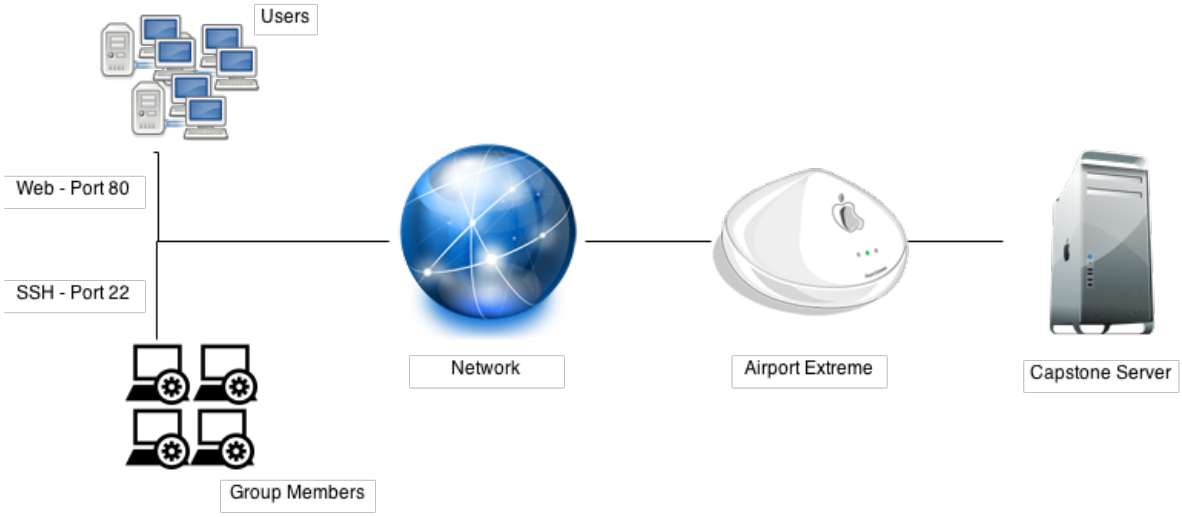


Figure 4: Networking Hierarchy

Table 2 - Hardware and software used for configuring and maintaining the server on the network

Hardware	**Software**
Apple Airport Extreme - 802.11n - Firmware Version: 7.6.4 Cogeco Modem	Terminal on Ubuntu Apple Airport Utility

3.2.5 MySQL Database

The database was originally installed and configured during the installation of the operating system when prompted to install the LAMP Stack which MySQL was bundled with. Besides setting the root user credentials for the database, it isn't until you start populating the database with content that you are really configuring the database.

There are four tables needed in order for the database to hold all of the necessary information for the application. These tables are for the users (i.e. User), fish (i.e. Fish), the fish tanks (i.e. Tanks), and the virtual fish tanks stored on the database to associate the fish inside the unique fish tanks (i.e. InTank). These tables are shown in Appendix I with the statements to create them and all of the columns related to them.
Some features that will require queries into the database:

- Retrieving user information about their aquatic environment including tank information and the fish in their environment
- Must be able to retrieve the user's fish in their respective tank and all of the optimal settings for those specific fish in order for the web application to calculate the optimal settings for the aquatic environment as a whole
- Retrieving the current settings of the FISHMO device on the user's tank at any given point in time
- Determining the compatibility of the fish regarding to their set type

3.3 Usage

The features that were all installed are used for various services that make the project be able to communicate effectively between hardware, software and the internet. With this collection of services operating at the same time in order to keep the application running, the backend has to be a reliable resource to support the many moving parts of the application.

The group members are the ones who need virtually unrestricted access to the server in order to modify and maintain the backend and web development of the project. The require special permission to write to the root folders that was given by the backend administrator. In order for group members to modify and add their own files, they would use SSH and its built-in feature of SFTP to transfer files to and from the server over the internet.

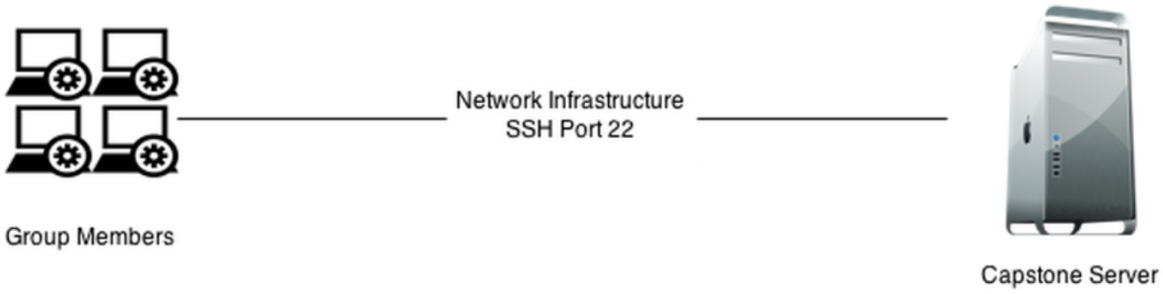


Figure 5: Networking Hierarchy

Users communicate with the server through accessing the web application that connects to the FISHMO SAAMS through a standard web browser. Besides this, the user is not concerned with communicating with the server explicitly and in most cases, the user is completely unaware of the existence of the server and it's function.

Figure 6: Networking Hierarchy

The Raspberry Pi controller for the FISHMO SAAMS device communicates to the server through the web application over the internet. The subject matter of the information being exchanged includes the camera feed and the readings of the various sensors determining the conditions of the environment.

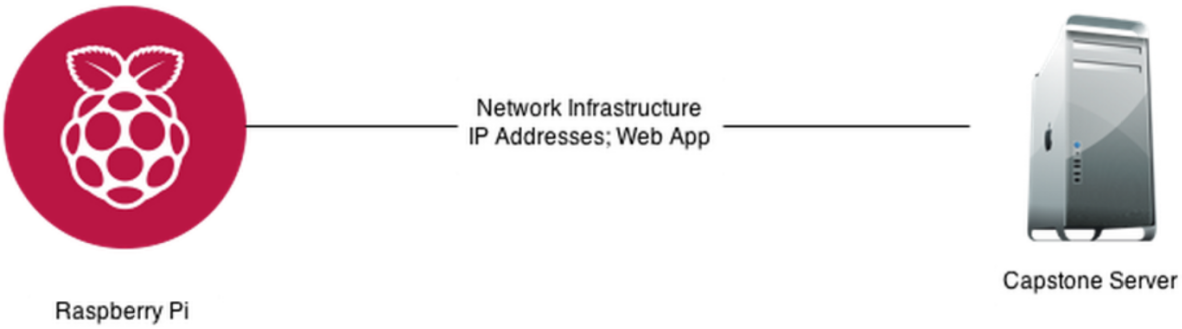


Figure 7: Networking Hierarchy

In all, every component from used from users to developers communicates with the back end of this application. With everything that went into installation and configuration of the server, it makes the application run smoothly. But that didn't come without any challenges as outlined in the next section with their solutions.

3.4 Error Handling

After the initial file upload of fish and their optimal recommended settings into the database it is imperative that the data being loaded is correct. To handle errors correctly, the queries used to push data onto the database and pull data for the web application must be verified. This verification procedure requires the use of scripts to check

the accuracy between the pulled data and the current data in the database. Furthermore, it would be necessary to authenticate and all changes to the database via pre-established conditions. This will ensure the utmost most accuracy and avoid data corruption.

3.5 Problems Encountered and Resolutions

3.5.1 Port Mapping and Dynamic DNS

Trying to install a script provided by no-ip to support changing IP addresses should that happen. When installing the script, a file had to be created and edited in one of the root folders. The changes made to the file were done incorrectly, corrupting the file. When the server restarted, it was unable to fully boot up resulting in the operating system to just be reinstalled.

Upon reinstalling the operating system, it was discovered that by making the IP static and using port forwarding would be sufficient instead of having a script run on startup. Also, on the Apple Airport, it allows you to login using a "dynamic global hostname" in order for the DDNS provider to be associated with the IP address of the server and the hostname URL.

In order to make this happen, we just went by what was discussed in the subsection for Port Mapping/Forwarding.

An alternative that is possible is when purchasing a domain from a provider like GoDaddy. With GoDaddy they provide a management interface for managing the DNS of the domain. From here, a user is able to set the IP address that the domain should be routed to. This is ideal if the user is using their own server to host of having other hosting options.

3.6 Features to be Implemented

All features set out in initial scope have been implemented into the FISHMO SAAMS system at this point in time. There are currently no features which are scheduled to be implemented.

4 Software Documentation

4.1 Introduction

FISHMO SAAMS provides users with the ability to modify and monitor their aquatic system via a web interface. On first load of the `http://fishmo.ddns.net` the user must register using the Product ID given with the physical system before being able to login. Once logged in, the website provides an interface to control the FISHMO system as well as accessing the tank's camera.

4.1.1 Software Specifications

The website will be created using the industry standard web programming languages: HTML, CSS, Javascript, and PHP. The code will be tested locally on a XAMPP/MAMP server running the latest version of Apache before being pushed to the Linux server to work on the LAMP Stack detailed above. The website will be usable from any popular web browser.

Regarding the CSS specifications, the Materialize framework was used for the majority of the styling with other adjustments made ourselves.

4.1.2 Software Design

Since the website will be the primary channel for human computer interaction, our team has focused on creating an enjoyable smooth experience in which the user will feel full satisfaction with each use. The base design for the website is focused on providing the user with all the information they want and need in a convenient format without flooding the user with information; simplification is key. The website design is created with two goals in mind: functional and accessible. The first goal is to make all controls going into the system easy to use while maintaining good visibility, the second goal is to make all sensor data easily accessible and readable to increase the user experience. This is be accomplished while following the main design principles for graphical user interfaces. Principals including:

- Following proper relationships between controls and their software
- Providing visual feedback to actions performed
- Applying constraints to avoid incorrect usage of the website this includes semantic constraints and logical constraints
- Ensure a high level of function visibility to ease the user through navigating the website
- Simplicity of learnability, allowing the user to grasp the site easily

These all help to build a conceptual model of what the user believes the system should look like in order to lower the chance of interaction error between the human and the system.

4.1.3 Design Rationalization

As per any software project, requirements will always influence the design of software. For example, the important requirement of security has influenced the layout of the website to best optimize user access. Furthermore, we modified the SSL encryption link between the website and the database for utmost security. Following our requirements: the GUI must be simple and easy to navigate, while containing all relevant information, without overwhelming the user. From these requirements the website is designed as a 3-page pamphlet of sorts. The main center page will contain the most relevant information, which is the user's aquarium, where they are able to customize their tank's settings and view the aquatic life in their tank via live camera feed. Auxiliary pages (The Fish Database and the Account pages) are linked from the homepage at the top, and will pop out a side menu when clicked(depicted in Appendix J).

4.1.4 Website and Database Functionality

Using PHP and MySQL, we are able to pull information from the database to populate our website. With this, we are able to store our users information as well as have our fish database on the web application.

4.1.5 User Interface Elements

As previously mentioned in section 4.1.2, the GUIs design principles applied are those of Dr. Donald Norman, whose principles include: affordance, consistency, constraints, feedback, mapping and visibility [4][9]. These principles have been thought-out and are currently being incorporated into the website (depicted in Appendix J). The main interaction the user will have with the FISHMO SAAMS system is through the website. The current design puts a big emphasis on the the user's tank environment, as such any information regarding the user's tank is primarily display. Extra information can be accessed with a simple button click.

As per Norman's rules, there are also constraints applied to the website to ensure that the temperature and water flow levels remain within appropriate ranges for the fish inhabiting in the environment.

Moreover, there will be the options to access the live-feed camera and modify the RGB LEDs which will be accessible via buttons. These buttons will provide feedback after being clicked so the user is constantly aware of what state the peripheral is in.

4.2 Installation and Configuration

There is no installation or configuration necessary to use the FISHMO website, though however the user has the freedom to install any browser to access the web page.

4.3 Usage

The website can be accessed from any computer with access to the internet. The website will be mainly developed for use on a desktop or laptop computer with development in the future for mobile devices. The user, when first visiting the website, will login which takes them to the main page for the website. On this page it will show the user's tank settings, which they can manipulate as they deem necessary. It will also show the camera feed, and the recommended settings for each fish in their tank. The user can add more fish to their tank on this screen, which will update the recommended settings automatically. Users can also access the account information through the account tab, and change any personal info as they deem necessary.

4.4 Error Handling

The website itself will have a lot of constraints to ensure the user does not input or try to input values that would cause errors with the software. The following are the constraints we will put in place to minimize the amount of errors to handle:

- **Temperature Level**
  - The temperature levels will update every time a different fish is added to the tank. These levels cannot be adjust past a certain threshold, as such the user cannot take the temperature level out of an appropriate range.
- **Water Flow**
  - The same design will be implemented here as the temperature level. We will constrain the levels to ensure proper functionality of the system.

Following these constraints there might be a few areas that can cause errors, and as such we will detail what they could be and how they will be resolved.

- **Error Connecting to the Pi**
  - This can be either caused by a faulty network card on the PI or the user's internet connection has issues with it. If the former is the case, the user will have to purchase a new tank unit to replace the faulty card. If the latter then the user will have to resolve their connections issue. We can offer a set of steps to help the user with connection issues on the website.
- **Error connecting to the server**
  - If this occurs it will be because the server has stopped working for an unknown reason. If this is the case we will work to figure out the problem and fix it to get the server back up and running.

4.5 Problems Encountered and Resolutions

There have been no problems up until the current state of the website. Further development will be required once the Hardware and Software integration commences.

4.6 Features to be Implemented

All features set out in initial scope have been implemented into the FISHMO SAAMS system at this point in time. There are currently no features which are scheduled to be implemented.

5 Appendix

5.1 Appendix A - Server Content on Github Repository

Due to the submissions for this projects work are through a Github repository, it posed an issue with trying to show the work done on the server for submission. In order to account for this, we have created a folder in the Github repository named "Server". This folder includes two sub-folders for the server content and a folder for the logs of SFTP uploads (the setup of SFTP logs is elaborated in Appendix C).

Within the "Server Content" folder, this is an exact replica of the content uploaded on the servers "/var/www/" folder to host the web application. Also within this folder is the "sftp-config.json" file that is discussed in Appendix B for SFTP setup on Sublime Text 2. With this file in the working directory of the server on the Github repository, every time a group member saves a file, the file is automatically updated on the server. And since this working directory and the server are always in sync, when this folder is committed to Github after a group member is finished working, Github will then have an updated version of the server content as well for submission.

An issue that arose during the syncing of the content between Github and the server is who is to use the "working directory" within the Github repository for the server. The reason for this is that the JSON file (in the repo) is configured for the group member that is in charge of the back-end. The other group members would have to work on separate folders on their own computers. The reason for this is so they can have their own JSON files on their working folders on their local machines and upload to the server using their username and password so the log can be accurate to the work they did. If they just worked off of the working directory within the repository, all other their changes would show up in the log as a different user considering they wouldn't be using their own username and password. The group member that is in charge of back-end, once the other group members have finished uploading their work to the server, would then notify the head of back-end who would then sync the content from the server to his local working directory, then proceed to commit the changes to the Github repository.

The purpose of this is to submit a copy of the contents being uploaded to the server to show the Professor and TA the work done on the server. With the server content alongside the logs will give the Professor and TA an idea of who was uploading content to the server.

5.2 Appendix B - Sublime Test SFTP Setup

An advanced text editor like Sublime Text 2 provides a significant amount of power and functionality for a user to take advantage of for whatever programming needs they require. There are even third party developers that make extensions for Sublime like SFTP functionality like we are using. This allows for users to have a folder on their local machine as a "working repository" so to speak, and once the user saves in Sublime, the changes get "pushed" instantly to the server.

In order to get this package, visit the developer's website (`http://wbond.net/sublime_packages/sftp`) and select "Install Free Trial" which will bring you to the install page. It then will ask you to install their "Package Control". In order to do this, paste the following code into the Console of Sublime and restart it.

```
import urllib2,os,hashlib; h = '7183a2d3e96f11eeadd761d777e62404' +
'e330c659d4bb41d3bdf022e94cab3cd0'; pf = 'Package Control.sublime-package'; ipp =
sublime.installed_packages_path(); os.makedirs( ipp ) if not os.path.exists(ipp) else
None; urllib2.install_opener( urllib2.build_opener( urllib2.ProxyHandler()) ); by =
urllib2.urlopen( 'http://packagecontrol.io/' + pf.replace(' ', '%20')).read(); dh =
hashlib.sha256(by).hexdigest(); open( os.path.join( ipp, pf), 'wb' ).write(by) if dh == h
else None; print('Error validating download (got %s instead of %s), please try manual
install' % (dh, h) if dh != h else 'Please restart Sublime Text to finish installation')
```

Figure 8: Script to install package control

The developer provides a step-by-step process for installing the extension on their website (`http://wbond.net/sublime_packages/sftp/installation`).

In order for the group member to have SFTP working properly for the appropriate directory they want locally, they must create a file with the appropriate settings in order for Sublime to connect to the server for SFTP. Wherever the group member is creating their folder to hold the contents of the server, the configuration file for the SFTP would go in the root of the folder to ensure that the contents of that folder get uploaded. In order to create this new file from Sublime, go to "File–SFTP–Setup Server" and a similar file as shown in Figure 9 will be

displayed. Look to Figure 9 for the appropriate settings you should have on the file to ensure that the connection is made by replacing the username and password with the group member's credentials and adding the additional lines of code. These extra lines of code ensure that every time a change is saved to the file, it gets automatically uploaded to the server.

```
{
    // The tab key will cycle through the settings when first created
    // Visit http://wbond.net/sublime_packages/sftp/settings for help

    // sftp, ftp or ftps
    "type": "sftp",

    "upload_on_save": true,
    "sync_down_on_open": true,
    "sync_same_age": true,

    "host": "fishmo.ddns.net",
    "user": "brandondasilva",
    "password": "msleslie",
    "port": "22",

    "remote_path": "/var/www/",
    "connect_timeout": 30,
    "ssh_key_file": "~/.ssh/id_rsa",
    "remote_time_offset_in_hours": 0,
}
```

Figure 9: sftp-config.json file for configuring SFTP on Sublime Text 2

5.3 Appendix C - SFTP Log Setup

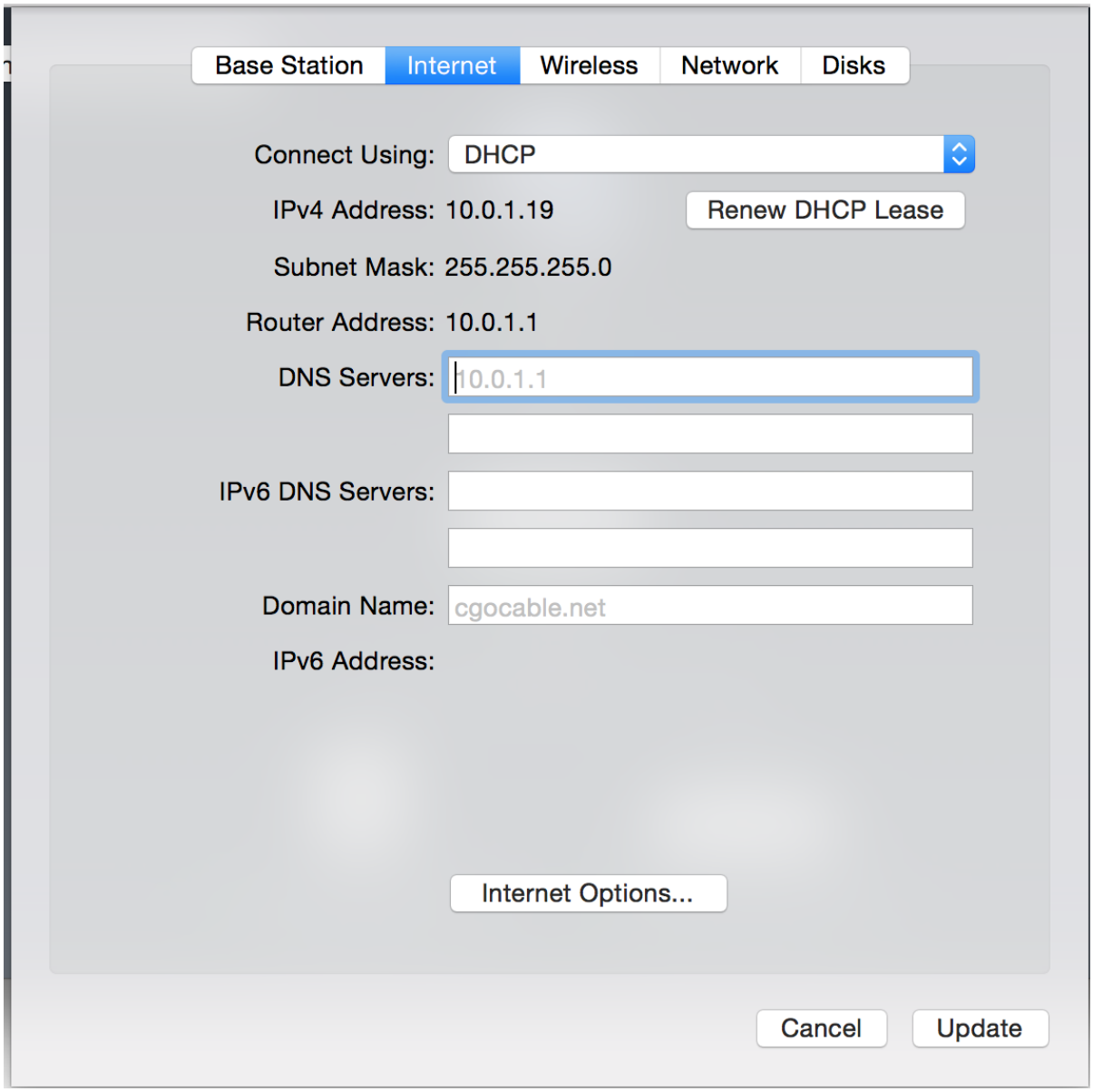
As discussed in Appendix A for keeping the Github repository up-to-date for submission purposes, a log for the SFTP uploads was necessary to keep track of which group member contributed what content. Since standard SFTP does not come equipped with the capability to log the changes made by users via SFTP, some custom configuration of the SFTP logging was in order.

This functionality could all be implemented from command line args and editing a text file within the server. The following steps were taken to implement SFTP logging [10]:

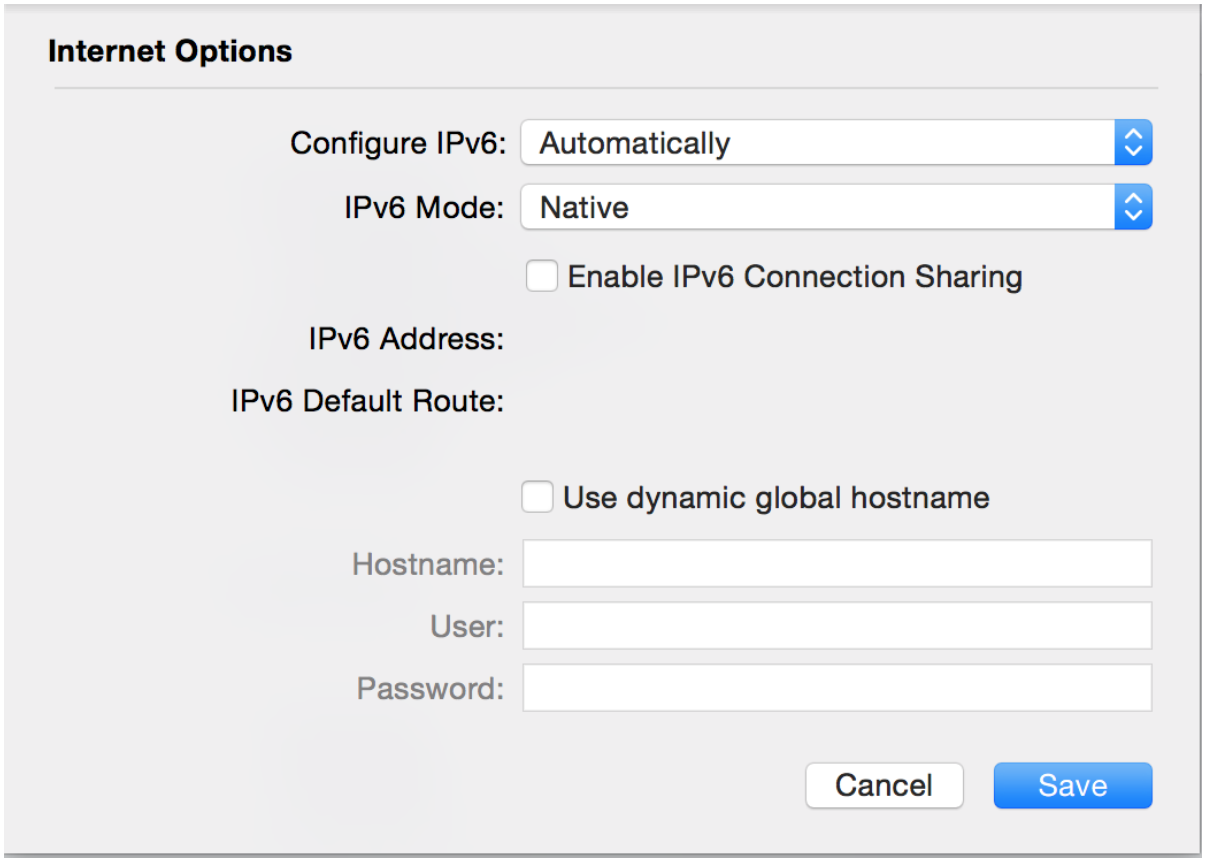
- Replace the "Subsystem" line in the "/etc/ssh/sshd_config" file with:
  - `Subsystem sftp /usr/libexec/openssh/sftp-server -f LOCAL5 -l INFO`
- Then add the following to "/etc/syslog.conf"
  - `local5.* /var/log/sftpd.log`
  - This code creates a log file for all of the SFTP changes done on the server upon restarting
- Restart the server and the changes will be implemented
  - To view the log, open the log file at "/var/log/sftpd.log" as the root user

By implementing the log using the directions listed, this allows to acquire the log files to track the group member's interactions for submission.

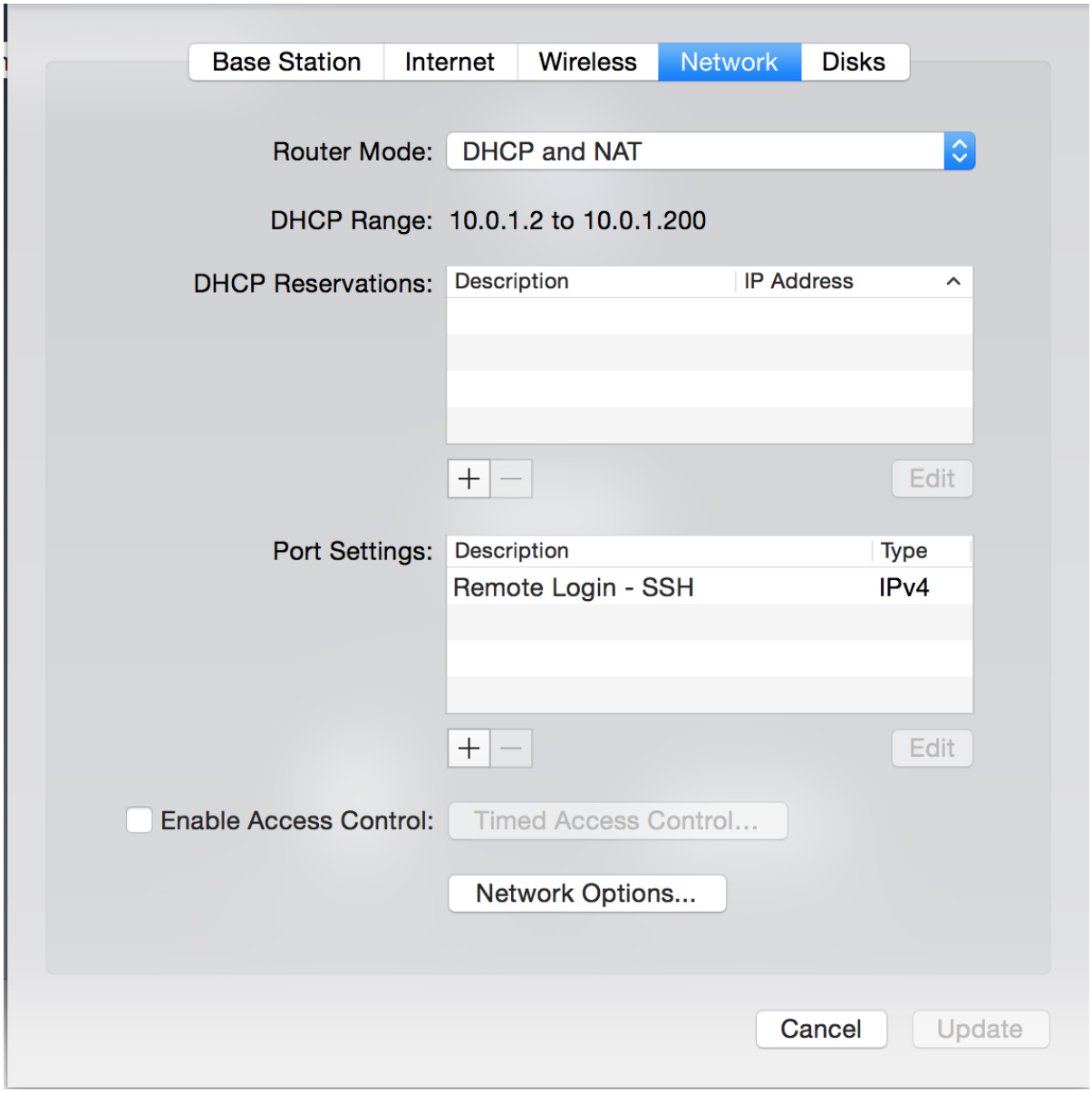
5.4 Appendix D - Menu to Set the Dynamic Global Hostname



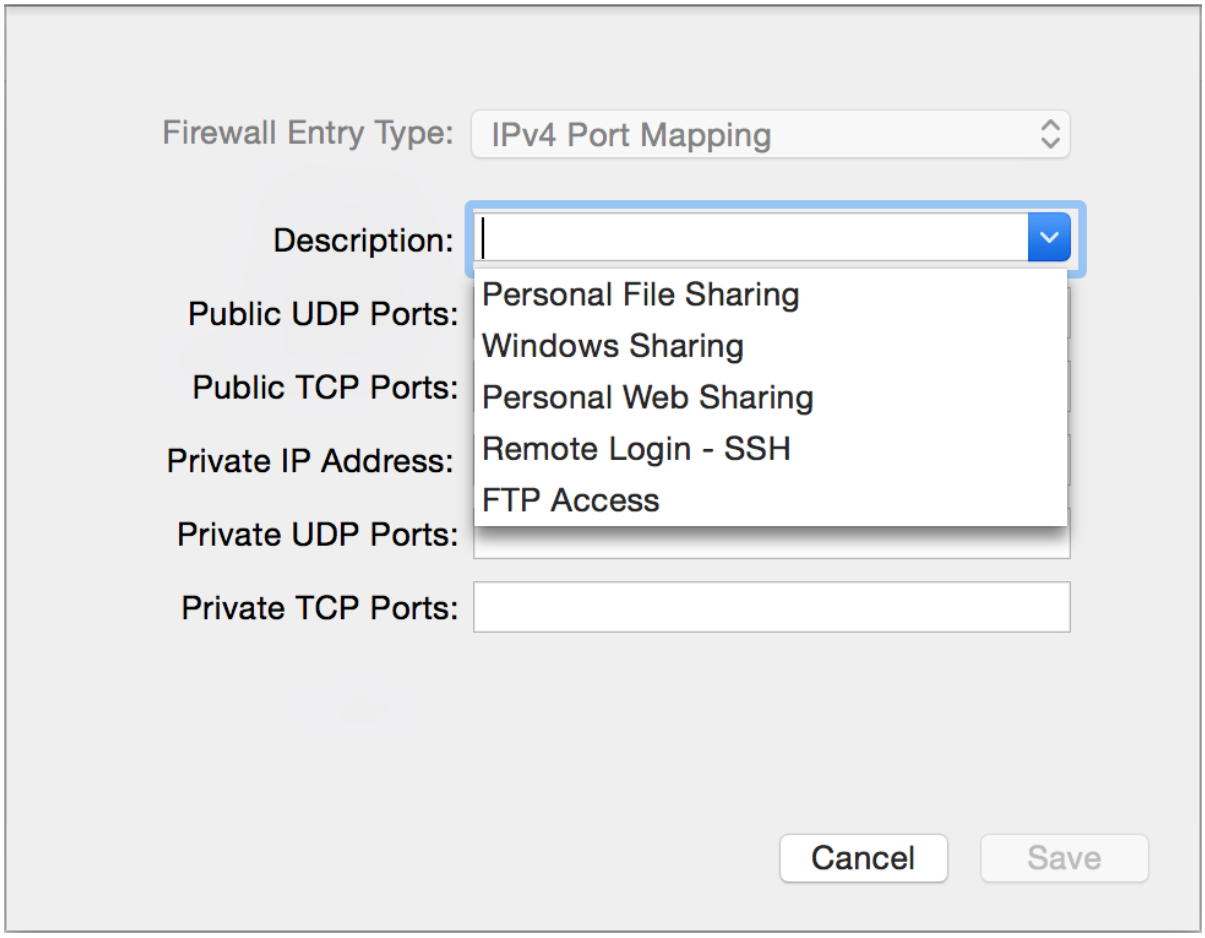
5.5 Appendix E - Menu for Setting the Dynamic Global Hostname



5.6 Appendix F - Menu for Setting up the Ports on Airport Utility

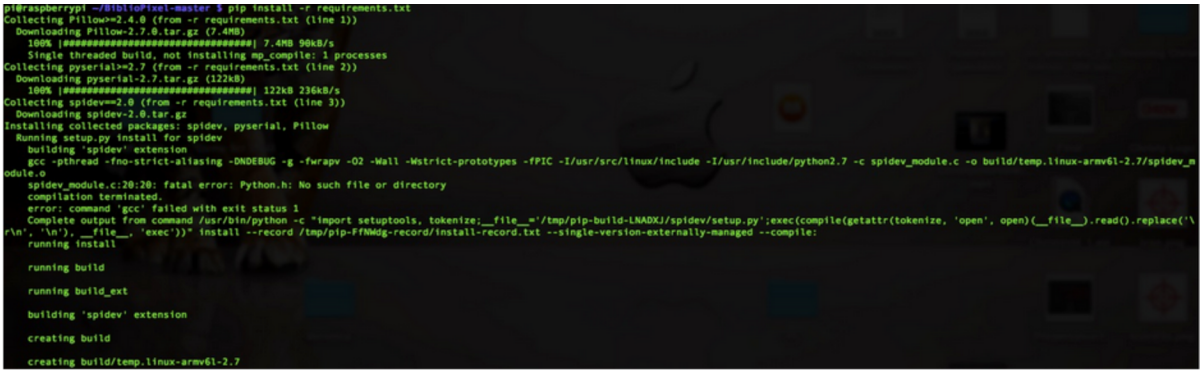


5.7 Appendix G - Setting up the Ports on Airport Utility

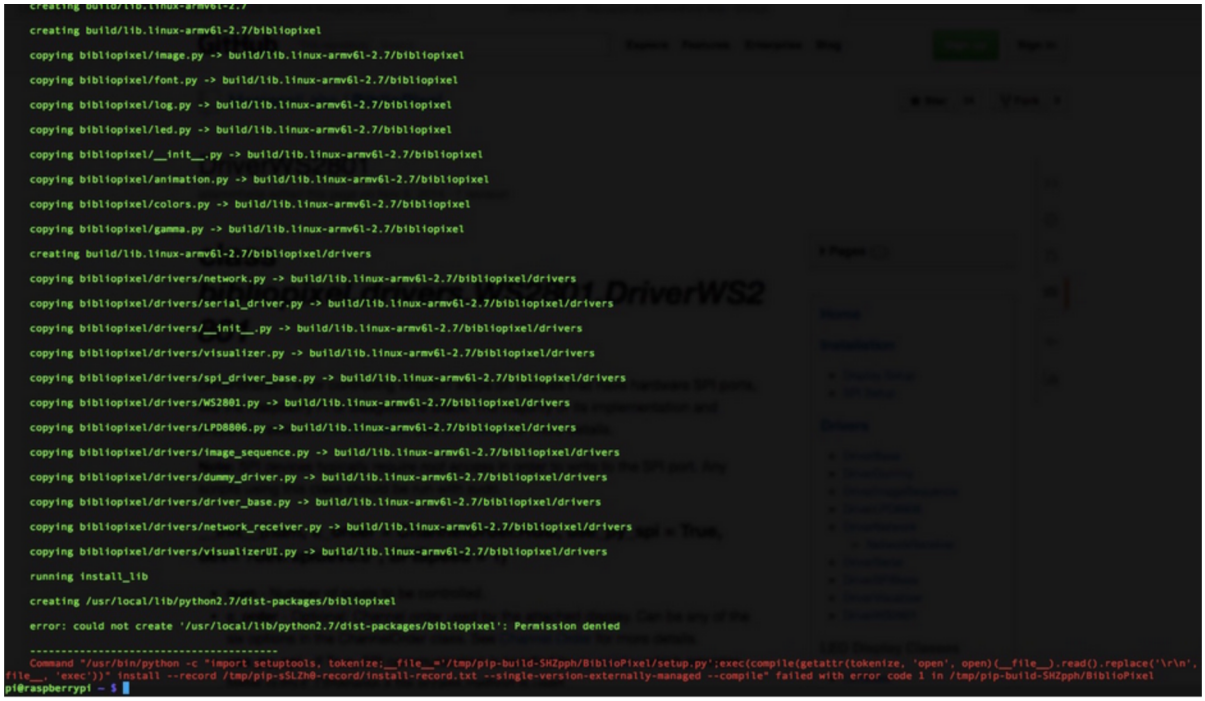


5.8 Appendix H - Manual Installation of Drivers

5.8.1 Appendix H.A - Download of Driver Package, Extraction and Compilation



5.8.2 Appendix H.B - Installation of Extracted Driver



5.9 Appendix I - SQL Commands for Setting the Tables

```
CREATE TABLE User (
        ID SMALLINT AUTO_INCREMENT NOT NULL UNIQUE,
        Username VARCHAR(10) NOT NULL UNIQUE,
        FirstName VARCHAR(20) NOT NULL,
        LastName VARCHAR(20) NOT NULL,
        Password VARCHAR(20) NOT NULL,
        PRIMARY KEY (ID, Username)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB;

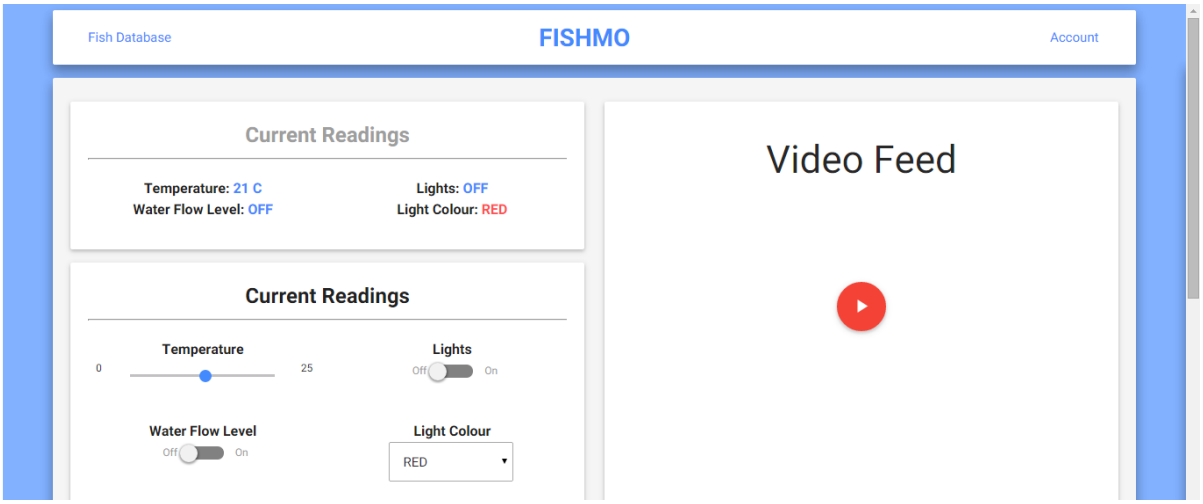
CREATE TABLE Tank (
        ID SMALLINT AUTO_INCREMENT NOT NULL,
        UserID SMALLINT NOT NULL,
        CurrTemp REAL NOT NULL,
        TargetTemp REAL NOT NULL,
        WaterFlow BOOLEAN NOT NULL,
        Lights BOOLEAN NOT NULL,
        LightColour VARCHAR(10) NOT NULL
                CHECK (LightColour IN ('red', 'green', 'blue', 'white', 'yellow', 'purple', 'multi')),
        Name VARCHAR(20),
        Model VARCHAR(20),
        PRIMARY KEY(ID, UserID),
        FOREIGN KEY (UserID) REFERENCES User(ID)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB;

CREATE TABLE Fish (
        ID SMALLINT AUTO_INCREMENT NOT NULL UNIQUE,
        Name VARCHAR(20) NOT NULL,
        Picture VARCHAR(30) NOT NULL,
        TempLow REAL,
        TempHigh REAL,
        Compatible VARCHAR(255),
        Description VARCHAR(120),
        Type INT,
        PRIMARY KEY (ID, Name)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB;

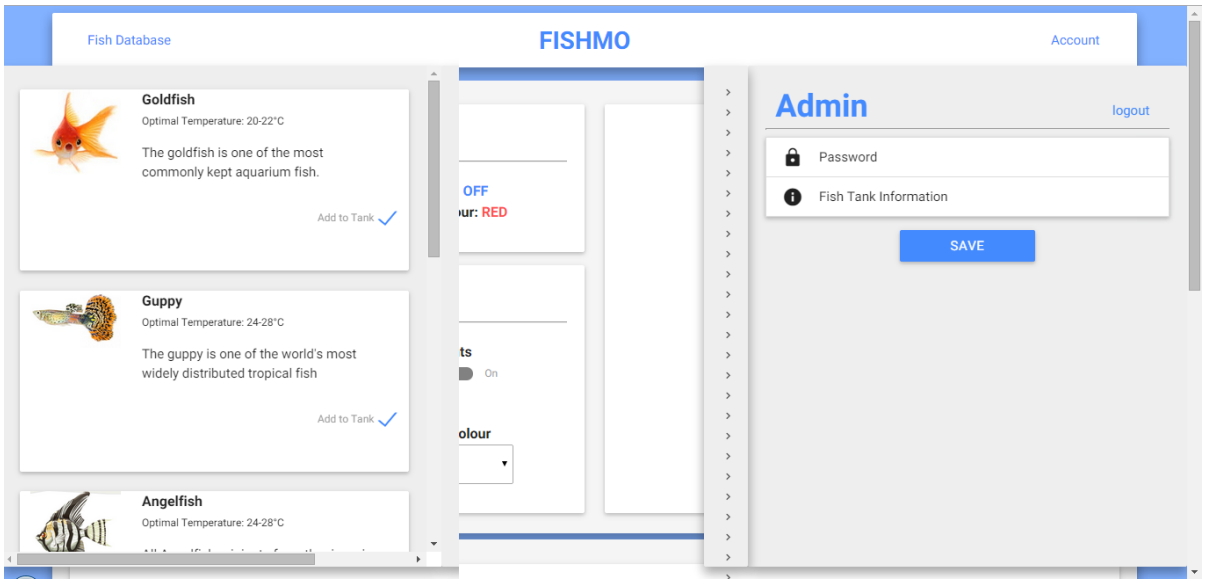
CREATE TABLE InTank (
        TankID SMALLINT NOT NULL,
        FishID SMALLINT NOT NULL,
        PRIMARY KEY (TankID, FishID),
        FOREIGN KEY (TankID) REFERENCES Tank(ID),
        FOREIGN KEY (FishID) REFERENCES Fish(ID)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB;
```

5.10 Appendix J - Website Design

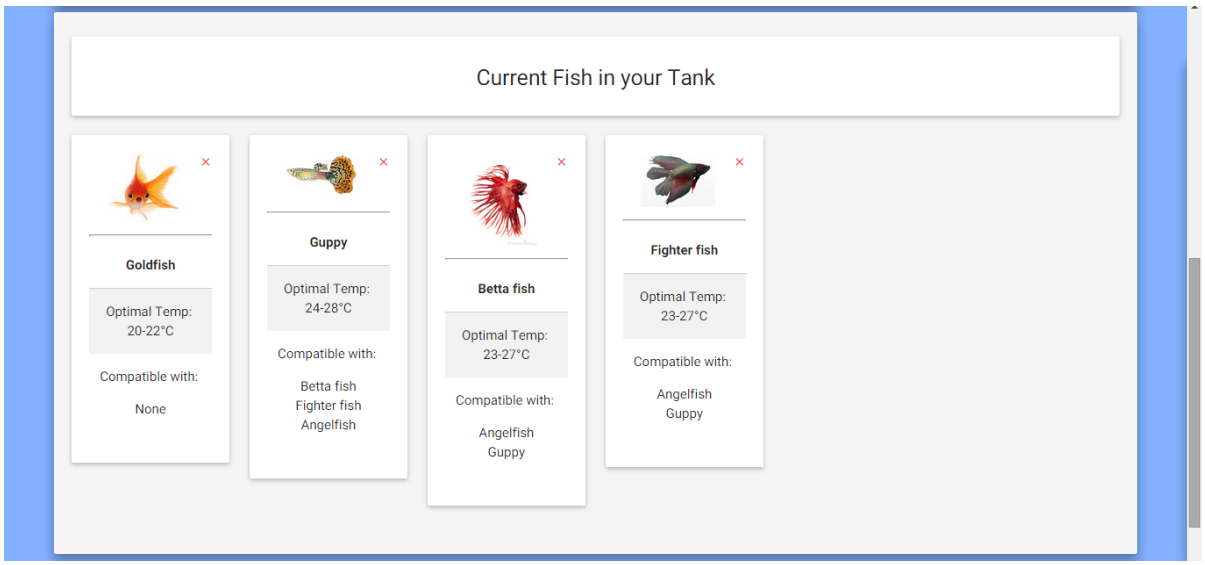
5.10.1 Appendix J.A - FISHMO Website Dashboard Design



5.10.2 Appendix J.B - FISHMO Dashboard Side-Menu Design



5.10.3 Appendix J.C - FISHMO Dashboard Virtual Tank Design



6 References

1. "AirPort and Time Capsule with Dynamic DNS." Dyn. N.p., n.d. Web. 14 Jan. 2015. http://dyn.com/support/airport-time-capsule-with-dynamic-dns/.
2. "Bootstrap." Bootstrap. Bootstrap, n.d. Web. 14 Jan. 2015. http://getbootstrap.com/.
3. "Chrooted Sftp User with Write Permissions to /var/www." Superuser. Stack Exchange, n.d. Web. 14 Jan. 2015. http://superuser.com/questions/669690/chrooted-sftp-user-with-write-permissions-to-var-www.
4. "Don Norman." Wikipedia. Wikimedia Foundation, n.d. Web. 14 Jan. 2015. http://en.wikipedia.org/wiki/Don_Norman.
5. "G3/4 Water Flow Sensor." Wikipedia. Wikimedia Foundation, n.d. Web. 14 Jan. 2015. http://www.seeedstudio.com/wiki/G3/4_Water_Flow_sensor.
6. Hawkins, Matt. "Raspberry Pi 1- Wire Digital Thermometer Sensor." Raspberry Pi Spy. N.p., n.d. Web. 14 Jan. 2015. http://www.raspberrypi-spy.co.uk/2013/03/raspberry-pi-1-wire-digital-thermometer-sensor/.
7. "How to Assign an IP Address on a Linux Computer." WikiHow. WikiHow, n.d. Web. 14 Jan. 2015. http://www.wikihow.com/Assign-an-IP-Address-on-a-Linux-Computer/.
8. "LAMP (Software Bundle)." Wikipedia. Wikimedia Foundation, n.d. Web. 14 Jan. 2015. http://en.wikipedia.org/wiki/LAMP_(software_bundle).
9. Matz, Kevin. "Donald Normans Design Principles for Usability." Architecting Usability. N.p., 28 June 2012. Web. 14 Jan. 2015. http://architectingusability.com/2012/06/28/donald-normans-design-principles-for-usability/.
10. Mehta, Shashank. "Enable Logging in the SFTP Subsystem." Enable Logging in the SFTP Subsystem -. Oneiroi, n.d. Web. 14 Jan. 2015. http://blog.oneiroi.co.uk/linux/enable-logging-in-the-sftp-subsystem/.
11. "OpenSSH." Wikipedia. Wikimedia Foundation, n.d. Web. 14 Jan. 2015. http://en.wikipedia.org/wiki/OpenSSH.
12. "Port Forwarding the Apple AirPortExtreme Router." Port Forward. Port Forward, n.d. Web. 14 Jan. 2015. http://portforward.com/english/routers/port_forwarding/Apple/AirPortExtreme/.
13. "Raspberry Pi." Wikipedia. Wikimedia Foundation, n.d. Web. 14 Jan. 2015. http://en.wikipedia.org/wiki/Raspberry_Pi.
14. Wang, Alvin, Alan Chang, Alex Mark, and Kevin Louie. "Materialize."Materializecss. N.p., n.d. Web. 22 Jan. 2015. http://materializecss.com/.
15. "Waterproof DS18B20 Digital Temperature Sensor + Extras." Adafruit. N.p., n.d. Web. 14 Jan. 2015. http://www.adafruit.com/product/381.