November 3rd, 2014

Dr. Andrew Rawicz
Faculty of Applied Science, School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

Re: ENSC 440W/ENSC 305 Capstone Project: Design specifications for an automated animal feeder to improve laboratory experimentation

Dear Professor Rawicz,

Please allow this report to serve as the design specifications for our ENSC 440w and 305w capstone project. This report will illustrate in detail our design for a capstone project: an automated animal feeder to improve laboratory experimentation.

Our organization, Optimaus, is comprised of four engineering science students: Kevin Killy, Robert Lepine, Kyle Griffiths, and Kenny Woo. We have been in contact with the Circadian Rhythms Laboratory at SFU, and we are designing our feeders for installation in said lab. We have also consulted Teresa Dattalo, a research technician working in the animal research centre, who has met with us multiple times to discuss our product and what would be most beneficial to the labs. Teresa has also given us a tour of the labs such that we could take necessary measurements and notes on available power and wiring.

If you have any questions or concerns about our design specifications, please feel free to contact Rob Lepine at 778-689-8720 or by email at rlepine@sfu.ca.

We hereby confirm that this report was written entirely by the members of Optimaus; we have received no external advice or assistance in writing this report. We also confirm this report has not been previously submitted for academic credit at this or any other institution.

Sincerely,

Robert Lepine
Optimaus - CEO

Optimaus

The Design Specifications for AutoFeed

An Automated Animal Feeder for Laboratory Testing

Prepared for:

Andrew Rawicz - ENSC 440
Steve Whitmore - ENSC 305
Respected staff of
School of Engineering Science
Simon Fraser University

Project Members

Robert Lepine
Kyle Griffith
Kenny Woo
Kevin Killy

Contact:

CEO, Robert Lepine
rlepine@sfu.ca

EXECUTIVE SUMMARY

Optimaus wishes to solve a problem experienced by researchers that involves the routine feeding of laboratory animals. Currently, researchers need to feed rodents on a set schedule. They also need to remove all food that they fed the animals after a set amount of time. Optimaus hopes to make a device which can automatically dispense and retract food from the animal at programmable times. Currently we are working closely with the circadian rhythms lab at Simon Fraser University headed by Dr. Ralph Mistlberger to gather requirements and cages for prototyping. The AutoFeed design poses a solution to automate the feeding of the rats in Dr. Mistlberger's labs.

Optimaus consists of senior engineering students from the School of Engineering Science, dedicated to design and implement a reliable product for the client. There is a need for this type of product in the market. That market consists of labs across North America who feed animals on a strict schedule. Implementation of this product could relieve graduate students from spending hundreds of working hours a semester doing mundane tasks. AutoFeed could also save research labs costs associated to graduate student wages. We have carefully planned to design, develop and test AutoFeed toward a working product. If time permits we plan to make more units to be used in the SFU Animal Research Centre lab.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY OF TERMS

API
Application programming interface. Provides a means of communicating with a particular application using programming functions provided in various programming languages.

CAD
Computer-Aided Design. The use of computer software to assist in generating mechanical design drawings

Google Calendar
Cloud calendar application provided by Google.

Google Drive
Cloud document creation and storage service provided by Google.

Gmail
Email service provided by Google.

I2C
Inter-integrated Circuit. A protocol for bidirectional communication between master(s) and slave(s). Slaves cannot communicate directly between one another. Consists of a 2-wire bus, one for the signal clock and the other for the signal data.

PWM output
Pulse width modulation digital outputs. These outputs will be used to drive the motors required to operate the dispensing apparatus.

Raspberry Pi (RPi)
A single-board computer. This will be used to interface with the Google Calendar API at a high level, before signalling the Arduino of desired actions in the physical world.

Servo driver
A circuit that provides the PWM signal and power to one or more servo motors.

SMBus
System Management Bus. A subset of the operations available in the I2C protocol.

1 INTRODUCTION

At Optimaus, we wish to provide an automated solution for the animal laboratory research industry. Research technicians are regularly forced to perform time consuming, mundane tasks. The time of these technicians would be better spent analyzing data or performing experiments. Automating the feeding of the animals will reduce the number of times the research technicians need to go down to the research facility, saving man-hours. Optimaus wishes to automate the feeding of rodents in an SFU lab. At irregular intervals, students are currently required to enter the labs in the middle of the night to administer food for research purposes. Our project, AutoFeed, will automatically feed the rats in order to relieve the students from coming to the lab at awkward times of the day or night. The AutoFeed design allows for the administration of a bulk of food that will last for up to a week without any required intervention from the laboratory technicians.

Our product will allow for a research technique known as temporal restriction, which involves feeding the rats for a given amount of time then restricting the remaining food entirely. Therefore, we have made the retraction of food a priority to satisfy our clients. Traditional rodent feeders do not retract food at all. Our value-add as that we will be restricting the food at remotely programmable times. Our solution will interface with a web application that allows the user to create a feeding schedule. The interface will be based off a calendar application where feeding periods appear as events on the calendar.

At Optimaus we believe that our product is useful and will help the research community test their hypotheses in an easier and more reliable way. Optimaus is composed of four engineering students at Simon Fraser's school of Engineering Science who have made it their goal to take on the automation of animal feeders. With the integration of our skills in hardware, software, and physics, Optimaus will make AutoFeed a success.

1.1 Scope

This document will describe the design requirements for each subsystem of Optimaus's product AutoFeed. This document will cover the following topics in the following order: the mechanical design, the controller design, and the user interface design.

1.2 Intended Audience

This document is intended for the designers at Optimaus. It should be used as a guide to ensure the final AutoFeed product is completed with all key design requirements in mind.

2 MECHANICAL DESIGN

Optimaus will supply a feeding mechanism that can be installed on pre-existing cages. The only alteration to the cages is a small rectangular hole cut out on the side of the cage for the animals to reach through and get the food, and 4 holes to mount the feeding mechanism. If at any point the feeding mechanism needs to be removed from a cage, for cleaning or for a different research project, the technicians can easily cover the rectangular hole with a pre-sized piece of acrylic mounted with the same 4 holes that are used to mount the feeding mechanism. This extra part will be supplied with each feeder. Otherwise the feeding mechanism will remain attached to each animal cage to be used as required.

The mechanism itself is comprised of the following components:

- Aluminum frame
- Controller box
- Clear removable hopper
- Slider door

The exploded view of the 4 parts is shown below in Figure 1. And the assembled parts are shown in Figure 2.

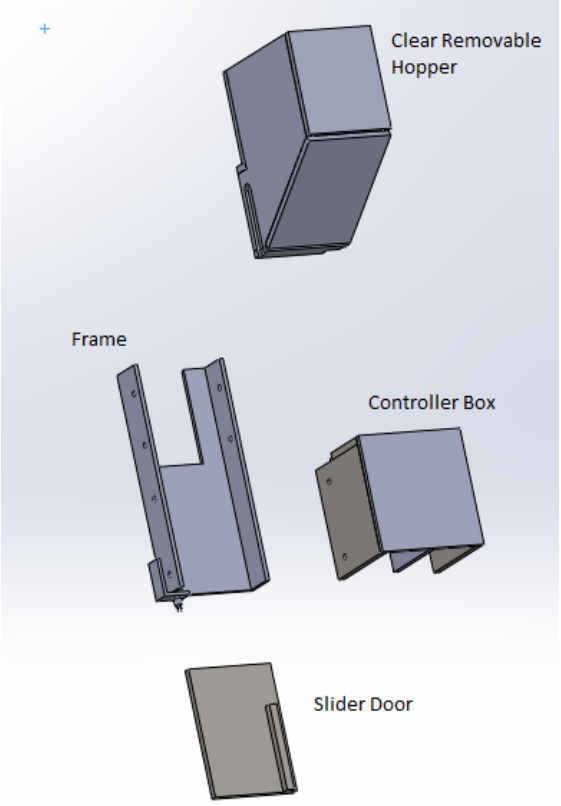

Figure 1: Four parts of AutoFeed

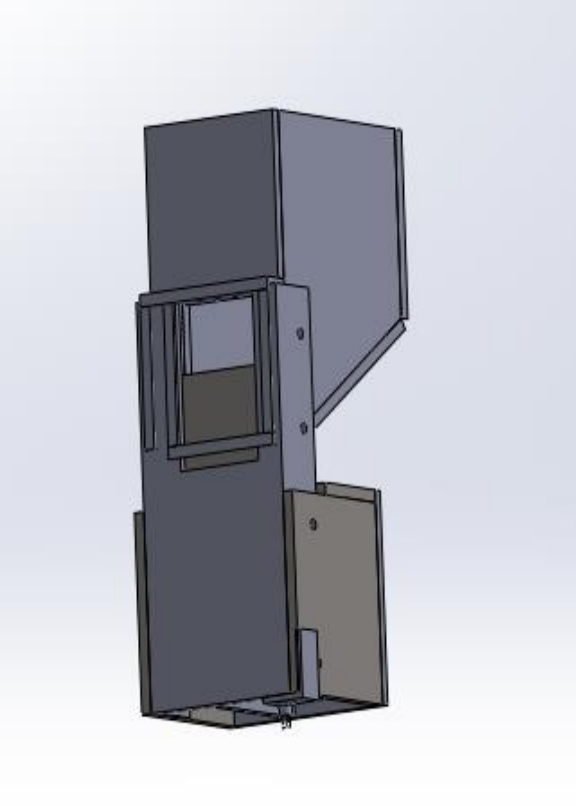

Figure 2: AutoFeed assembly

The integration of all 4 parts is designed to be easily assembled and disassembled. This is to allow the apparatus to be cleaned efficiently and allow for parts to be interchanged between cages if necessary. The final assembly is also expected to be very durable, as there are no fragile pieces besides the Hall Effect sensors which will be protected inside the controller box. The acrylic itself is 0.22 inches thick, bonded by methylene chloride to produce a single solid part.

2.1 The Aluminum Frame

The frame itself is made out of an architectural channel, which will be attached directly to the existing cage with wing nuts for easy removal and/or installation. The remaining elements of the feeding mechanism are attached to this frame, either by screws or easily accessible sliding joints. In the prototyping stage of our design, this frame will be cut using basic machine tools. When AutoFeed is pushed into production, the frame will be cut using an N-mill for precision cutting and also to add chamfer to the edges to reduce burs and sharp points. A CAD model of the frame is shown below in Figure 1.

2.2 Controller Box

The controller box will be made out of coloured acrylic or a material similar in strength and machinability. Inside it will house the servo motor (found in Figure 3), which controls the slider door through a gear rack and pinion. The gears will be made out of a hard plastic to avoid broken teeth if for any reason the gear begins to skip. The controller box will also have two Hall Effect sensors (found in Figure 4) to supply feedback to the microcontroller about the position of the slider door; specifically when it is either fully open or closed. The Hall Effect sensors will be mounted roughly two inches apart to account for the door slider moving, and avoid the effect of magnets that are not meant to trigger the sensors. Also included in the controller box is a cat5e cable connection to communicate with the microcontroller. The cat5e cable and connectors will simplify the procedure for moving the feeding mechanisms between different cages and will help avoid any tangling of multiple long wires.

Figure 3: Servo motor

Figure 4: Hall Effect sensor

2.3 Clear Removable Hopper

The hopper is attached to the frame by an easily removable sliding mechanism to allow the technicians to refill the hopper without the need of a screwdriver. The hopper was designed to store enough food to feed a single rat for a week (roughly 300g of pelleted food). The clear removable hopper will also be made out of acrylic, which can be cut by a laser cutter for simplified manufacturability during production. The laser cutter will also ensure identical products between mechanisms which allows for parts to be interchangeable in the event of a malfunction. The CAD design can easily be exported onto a laser cutter friendly program on a 2D sheet (shown in Figure 5). The feeder mechanism can then be assembled after the pieces are cut to resemble the CAD design. Refer to Figure 6 for an image of the clear removable hopper that has been cut on the laser cutter and then assembled.

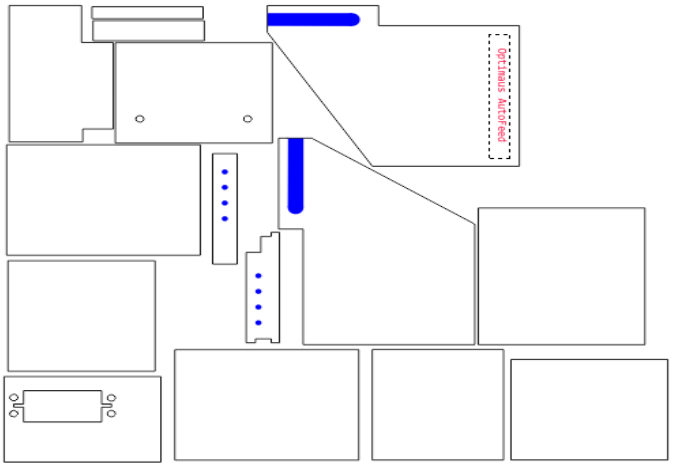

Figure 5: CorelDRAW drawing of an AutoFeed unit

Figure 6: Hopper assembly after laser cutting

At the base of the hopper is a set of bars which the food will rest against. These bars will be spaced such that the rat can eat between them but may not remove entire pellets and store them elsewhere in the cage. The hopper will be see-through so that the technicians can see the food level in each hopper and can refill when necessary.

2.4 Slider Door

The sliding door will be braced inside of the architectural channel, laser cut to fit with highest tolerance. Imbedded inside the slider door there will be a small magnet that will trigger the Hall Effect sensor for immediate feedback when the door is either fully opened or fully closed. The slider door will also be rounded on its edges to avoid injuring the rats when the door is closing.

3 USER INTERFACE DESIGN

We are designing AutoFeed for laboratory technicians and researchers who do not have technical experience working with computers. Keeping our target users in mind, we have chosen to implement a Google Calendar user interface. Google Calendar allows a user to create different calendars and name them to whatever they desire. We will use this feature to create a different calendar for each cage that AutoFeed is controlling. In this way, our user will be able to customize feeding schedules individually for each cage.

We have chosen to design our user interface with Google Calendar for two reasons. Firstly, as previously mentioned, we have made the assumption that our users do not have technical computer experience. Therefore, it is important that we design our user interface to keep the command line and code hidden. We want our product to be desirable out of the box; not intimidating with a steep learning curve. Online calendar applications such as Google Calendar are already intuitive and pleasant to exploit for first time users. By coupling AutoFeed directly with Google Calendar, our users will quickly realize its ease of use and will be more receptive to integrating it into their work practices. Secondly, our clients at the SFU Animal Research Centre have a Google Drive account to which they upload their rat activity data. Therefore, a Google Calendar solution for programming feeding schedules will fit seamlessly into their current work practices.

Google is going through API version changes and the new base version is API v3 (upgraded from API v2). The long standing API v2 will become obsolete as of November 2014, thus we will need to adopt API v3 in our design. The biggest difference between the versions, aside from data structure details and the naming of the functions, is the authentication method. Google requires authentication to pull data from their servers and the authentication protocol implemented in API v3 is called OAuth 2.0

With the earlier versions of the Google API, the developer account that is hosting an application must embed their Google account credentials directly into the code in order to authenticate that a call to the API is coming from a trusted source. OAuth 2.0 uses a Client ID and a Client Secret as credentials to validate that a trusted source is pulling data from the Google API. In this way, we are not required to include our user's Google account credentials in the code, and will embed the Client ID and Secret instead. Because we will be using this method of authentication, our design will not compromise the security of our user's data by having sensitive account credentials embedded in our code.

Below, in Figure 7, is a screenshot from Google Calendar. On the left hand side of Figure 7, there is a list of the calendars associated with the user's Google account. Lines are drawn connecting the calendar to their respective events. It can be seen by this image that Google Calendar implements color coding to relate events to their home calendar. The user can also select or deselect calendars such that they can control which events are visible. In our implementation, an event will correspond to the times that the rats will have access to the food. When an event for a starts the food door will open, and when an event ends the food door will close.

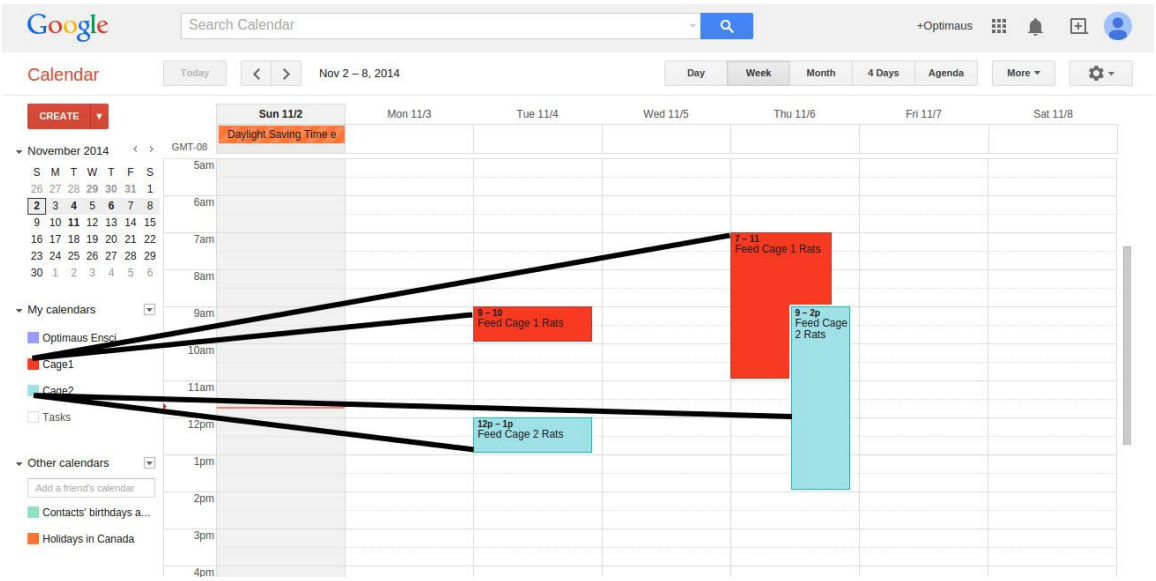

Figure 7: Google Calendar User Interface

4 CONTROLLER DESIGN

The controller design section has been broken up into four subcategories, each pertaining to a critical function of the controller. These four subcategories are as follows: *Obtaining Event Data*, *Door Open/Close: Hall Effect Sensors*, *PWM Signaling: From RPi to Servo Driver*, and *Email Alert for Error Handling*.

4.1 Obtaining Event Data

Firstly, we need to obtain the Google Calendar data, interpret it, and execute the appropriate action at the appropriate time. The Google Calendar API supports various programming languages and types of applications. In all cases, we require some sort of high level application in order to utilize the API.

We have chosen to use a Raspberry Pi (RPi), running the Raspbian operating system, to access the calendar through the API's Python bindings. The RPi is a low-cost, credit card size computer that will provide us with the high level applications layer we require in order to utilize the calendar data. The RPi will poll the calendar once every 20 seconds to check for new data.

We will need to loop through each calendar that is controlling a cage for whether there is an event with the start time or end time that is equal to the present time (down to the minute). Therefore, we will need to be able to compare against the current time. This also means that, for sixteen cages, we will be requesting data from Google servers sixteen times every twenty seconds. At these rates, we will be making a total of 69,120 requests to Google servers every day, which is less than the allowable 100,000 request per day limit for Google's Calendar API (4).

If a calendar has an event starting at the current time, we will communicate with the servo driver and open the door to the corresponding cage. If a calendar has an event ending at the current time, we will then communicate with the servo driver and instead close the door to the corresponding cage.

4.2 Door Open/Close: Hall Effect Sensors

Each feeder door will utilize 2 Hall Effect sensors acting as limit switches to stop the door when opening and closing. For a full block diagram of our control system with feedback from our Hall Effect sensors, please see Figure 8.

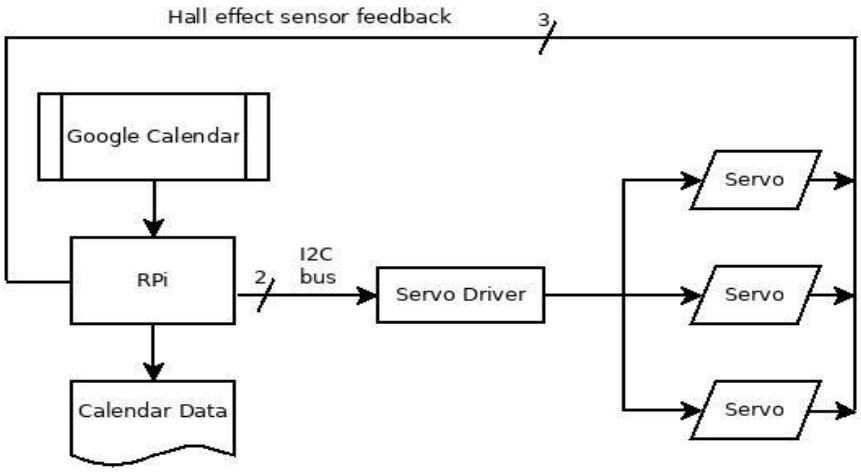

Figure 8: Controller block diagram with Hall Effect feedback

Hall Effect sensors will be used to stop the servo motors once a feeder door is fully opened or closed. Hall Effect sensors have a very low activation range and are polarity-sensitive. This will minimize the chance of spurious magnetic field changes affecting our sensors and give us the ability to obtain feedback regarding the position of the feeder door.

4.3 PWM Signaling: From RPi to Servo Driver

The Adafruit 16-Channel 12-bit PWM/Servo Driver was chosen as the servo driver. This servo driver can drive up to 16 servo motors, and has the ability to be "chained" together with more drivers, allowing the number of supported servo motors to be scaled up with ease. The servo driver uses the I2C (Inter-integrated Circuit) protocol to receive its PWM signals. For a diagram of the necessary components for 12C communication and the wiring between these components, please refer to Figure 9.

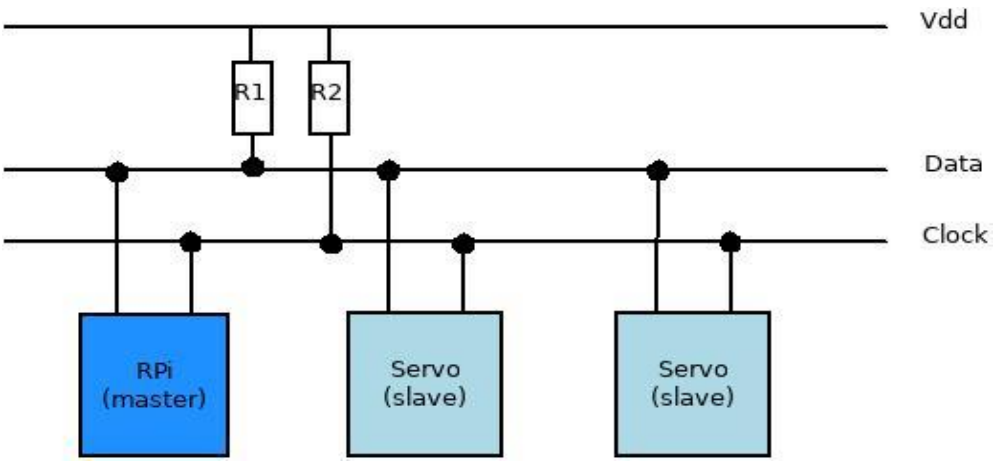

Figure 9: I2C Protocol

The master uses the unique address of the desired slave and initiates communication between the slave via a 2-wire bus to either read or write data. The bus is used to write PWM signals to the slaves (i.e. servo motors). The RPi I2C hardware supports 7-bit addresses, allowing up to 128 unique addresses. This, along with the fact that our chosen servo driver can be chained with more drivers, makes using the I2C protocol very scalable for our purposes. With our solution, we can automate the feeding schedule of up to 128 individual cages, each cage having its own schedule customizable via Google Calendar.

The Python module 'python-smbus', containing Python bindings for a subset of the I2C protocol, will be used in order to communicate with the servo driver. With this module, the Python script setPWMFreq(freq) can be used to adjust the PWM frequency to determine how many full 'pulses' per second are generated. The function setPWM(channel, on, off) controls when to turn the motor at the specified channel on and off. The 'on' and 'off' parameters can be values

between 0 and 4095, which represent how far into the pulse we well firstly turn the motor on, and then subsequently turn it off (5).

In the case that the Hall Effect sensor fails to stop the motor, we still need the motor to eventually stop rotating anyways. Therefore, five seconds after we turn the motor on, we will turn the motor off regardless of whether we have received feedback from the Hall Effect sensor. For example, if we want to activate a motor on channel 3 of the servo driver and turn this motor off five seconds later, we will use the following code:

```
setPWMFreq(819)
setPWM(3,0,4095)
```

This code works because, with a frequency of 819 pulses per second, the 4095th pulse will occur after five seconds. In this way, we can ensure that each motor will stop rotating after five seconds.

In order to incorporate the feedback from the Hall Effect sensors, we will need to concurrently:

1) Send the signal to the servo driver to start the PWM signal
2) Monitor the GPIO pin for the respective Hall Effect sensor

Once the Hall Effect sensor is tripped by the magnet on the sliding door and the feedback from the Hall Effect is read at the GPIO pin, we will then send the signal to the servo driver to stop the PWM signal. The motor will then stop moving the door.

To achieve this concurrency, we will be using concurrency libraries available in Python 3.4. For each desired action to a servo motor, we can execute a concurrent process to monitor the respective GPIO pin in order to know when to stop the door from opening or closing.

The Raspbian repository also contains a ‘pigpio’ library for monitoring GPIO pins on the RPi. This library has the ability to sample GPIO pins at a frequency between 100,000 and 1,000,000 Hz (6). This will help keep the response time of the RPi to the Hall Effect sensors to a minimum, so that we can stop the servo motors from opening/closing the doors as soon as the Hall Effect sensors are tripped.

4.4 Email Alert for Error Handling

If a feeder door is “jammed”, indicated by neither Hall Effect sensors being activated by a nearby magnet, the RPi will send out an email notification to the Google Calendar account.

The email notification will be sent via a command line tool to allow for scriptability and automation. The Google Calendar account will be set up to forward the alert to all parties in its Gmail forwarding list. Therefore, as different lab technicians come and go the forwarding list on this Gmail account can be updated to accommodate whoever is a part of the staff that will need to troubleshoot a jammed feeder door. In this way, any laboratory staff members can receive error messages to their personal email addresses.

5 SYSTEM TEST PLAN

The AutoFeed product will be tested on live rats and mice before it is available on the market. For disclosure, the premise behind AutoFeed is that it is designed to allow and disallow access to food for lab animals, namely rats and mice. Therefore testing procedures will be done on mice and rats. This does not prevent the use of AutoFeed to animals other than mice or rats, however, we cannot guarantee that the AutoFeed product will achieve user satisfaction if it is used on these other animals.

5.1 Test Items

I. Test that the dispensing apparatus does not have the potential to injure the animals in any way, ie look for sharp edges and/or exposed screws.
II. Test that when the slider door is open, the animals can reach the food, and not be blocked by the width of the bars
III. Test that the hopper is of the correct size to hold enough food for one fully grown rat (biggest consumer) for 1 week (roughly 300g). And that the food slides to within reach of the animals inside of the storage hopper.
IV. Test the reliability and functionality of both Hall Effect sensors, and how they interact with the magnet embedded in the sliding door
V. Test the reliability of the rack/pinion and motor set
VI. Test to make sure that the leftover food particles don't get stuck in the mechanical system and prevent the door from opening/closing
VII. Test that multiple cages can be controlled by the Google calendar to both open and close slider doors
VIII. Test that in the event of feedback Hall Effect sensors not being tripped at the expected time, an email will be sent to a group of recipients in the Gmail forwarding list of the account to with the Google Calendars

5.2 Testing Procedure

Testing on individual features will be done throughout the prototyping process. However, once integrated, all of the features working together will still need to be put through strenuous test situations to assure the functionality of AutoFeed. The quality assurance of the AutoFeed product is particularly important because it will be accessible by live animals, of which are expensive and fragile because they

are part of experimental lab groups. Therefore ensuring consistent and safe food deployment and removal is crucial to the systems overall function. After each of the individual features of AutoFeed are tested, the system will then be put through the following set up tests, that it is required to pass, before being put into production.

1. **Single Cage Without living Specimen:** The dispensing system will be filled with the food pellets that will be given to the rats during feeding time. The system will be controlled by the Google Calendar, where it will be programmed to close the slider door, and open the slider door, multiple times within the testing period. To test the email notification system, the slider door will be manually blocked to stop it from pressing the feedback switch. This test must satisfy the following test conditions: IV), V), VI), VIII).
2. **Single Cage With Living Specimen:** The cage will then be populated by a living Sprague Dawley rat (the same species that will be in the laboratory). The testing apparatus will be set up otherwise the same as in the first test. The rat will be left in the cage for at least 2 days. This test must satisfy the following conditions: I), II), III), VI).
3. **Multiple Cage Without Living Specimen:** Multiple cages will be set up with the AutoFeed dispensing apparatus and connected to the controller. Each cage will be programmed with a different feeding schedule and will be verified if they are accurate to the required times. This test must satisfy the following conditions: IV), V), VII).

With the chronological completion of these test procedures, Optimaus will ensure that the AutoFeed product is functioning as expected. With every change to the features or design characteristics included in AutoFeed, this system test plan will be adapted to accommodate said change.

6 CONCLUSION

As we move forward with our design and prototyping iterations, these specifications will hold us accountable to include the features necessary for a successful final product. Although we will be focusing on completing a single feeder controlled via Google Calendar, we will keep scalability as a priority in our design. During a test in animal research laboratories, many unique animal subjects are required simultaneously to receive enough data from experiments.

Therefore, we hope to provide the SFU animal research centre with a suite of feeder units such that they can fully implement AutoFeed into their experimentation practices. It is our ambition to have 16 feeders installed in the lab by next semester (Spring 2015). We are extremely motivated to have the opportunity to develop AutoFeed. The dedication of the engineers that comprise Optimaus makes the complete automation of laboratory feeders possible.

7 REFERENCES

1. M. Barr. (2007). *Introduction to Pulse Width Modulation (PWM)* [Online]. Available: http://www.barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation

2. Raspberry Pi Foundation. (2014). *Getting Started with Raspberry Pi* [Online]. Available: http://www.raspberrypi.org/learning/teachers-classroom-guide/getting-started-guide.md

3. Google Developers. (2014). *Get Started with the Google Calendar API* [Online]. Available: https://developers.google.com/google-apps/calendar/get_started

4. Google Developers. (2014). *Google Calendar API Usage Limits* [Online]. Available: https://developers.google.com/google-apps/calendar/pricing

5. Adafruit. (2014*). Adafruit 16 Channel Servo Driver withy Raspberry Pi, Library Reference* [Online]. Available: https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/library-reference

6. Pigpio. (2014). *The Pigpio Library* [Online]. Available: http://abyz.co.uk/rpi/pigpio/