Electrical Systems

Donaldson Lab, University of Colorado Boulder

Gabriel Chapel, Ryan Cameron

Abstract

The following document describes the design of the electrical schematics and boards that were developed in house to run the Operant Cage system, developed by the Donaldson Lab at the University of Colorado at Boulder. Designed with ease of use in mind, the electronic components connect the mechanical systems with the software systems, making sure that everything can be easily put together by anyone, including those without any engineering expertise.

Introduction

In the Operant Cage system, there are many moving parts, servos, and sensors that all need to be controlled by the code base. In addition, that code has to be deployed on a hardware system that can handle the amount of inputs and outputs needed for each servo and sensor. Since the Operant Cage system is designed to work for multiple experiments and for a long time, the electrical hardware must be able to sustain that longevity. To this end, the systems were designed and manufactured with resilience and modularity in mind.
The electrical hardware system consists of the microcontroller, associated hats, custom printed circuit boards (PCBs), and wiring. This can all be powered through standard 120V AC wall outlets for simplicity, meaning that the system can be placed anywhere there is a wall outlet and sufficient space.

Components

The electrical hardware used for operating the Operant Cage are all either off-the-shelf or can be fabricated in-house. This includes a Raspberry Pi, microSD card, Adafruit Servo Hat, two custom PCBs, two power supplies, and the wiring needed to connect it all. The Raspberry Pi is in charge of controlling the electrical parts within the system using a Python code base that is loaded onto the Pi's SD card. The Servo Hat is placed on top of the Raspberry Pi in order to control the rotation of all of the servo motors. To power the servos, this board uses a power source separate from the one used for the PI, since they run on a higher voltage. To control the other components, we stack a custom board on top of the Servo Hat that routes power, ground, and signal lines to the appropriate sensors and integrates the signals from the PCB on the back of the cage. This system can also be used in conjunction with Bonsai, where it integrates an Arduino microcontroller that sends the commands to Bonsai for processing.

Raspberry Pi

The Raspberry Pi is the main controller of all the electrical signal routing and decision making that the system does. It serves as a stand-alone, lightweight computer with internet connection capabilities and a built in microcontroller, connected to GPIO pins. Since it is a computer, the lab is able to install all of its own software packages onto the Pi (using Python 3), and it allows for both remote usage through SSH terminal, and easy interfacing to the GPIO pins and microcontroller. The lab uses the Raspberry Pi 3 B+, with a 5V power supply and at least 32GB microSD card for memory storage. All of the control software is written in Python 3 and can be pulled onto the Pi through the GitHub repository, https://github.com/dprotter/RPI_operant.git. This requires some setup, which is detailed in an in-depth tutorial in a later section and also documented in the repository's README file. The Pi is powered by a 5V power supply or a 5V wall adapter.

Servo Hat

The Raspberry Pi has trouble sending consistent pulse width modulated (PWM) signals, which are necessary for controlling servo motors, so this system uses an Adafruit Servo Hat on top of the Pi to accomplish this. The hat provides power and control to all the servos within the system, up to a maximum of sixteen 3-pin servos. It requires its own 5-6V power source, separate from the one powering the Pi, since the voltage required by the servos is higher. This must be a maximum of 6V, but the current depends on the number of servos in the system. In the case of the Operant Cage, we use a 5V, 3A power adapter. This hat was chosen because of the ease of use through a premade Python 3 library, Adafruit CircuitPython ServoKit library CITE. The full documentation for the Servo Hat can be found here in the complete docs CITE.

Front PCB

In a system with moving parts, it is important that the electrical connections are secure, however, it is also helpful if components can be removed and replaced. The Servo Hat supports both jumper wires and directly soldering to the board, but to provide secure and modular connections, we, instead, designed a custom PCB to sit on top of it. This custom Front PCB sits directly on top of the Servo Hat and is meant to direct all of the sensor signals in the system to and from the Raspberry Pi's GPIO pins. It does not have any complex electrical components, but it does have JST-XH connectors, which are not necessarily easy to disconnect but can be if needed. The schematic and board diagram are shown below.

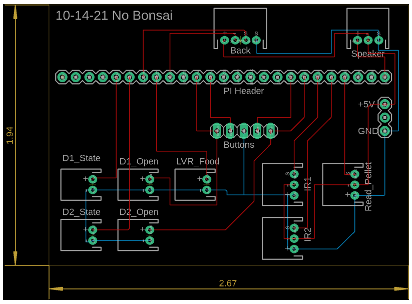

Fig. 1: Board Layout for the Front PCB

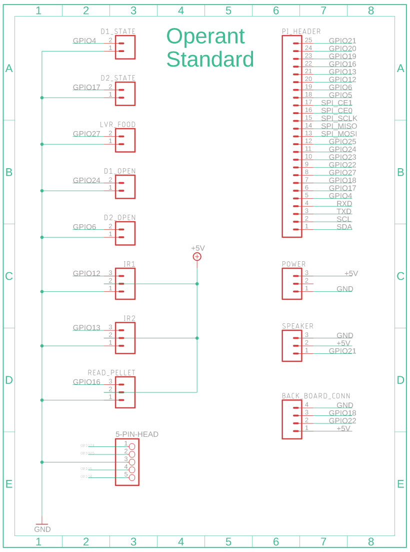

Fig. 2: Schematic for the Front PCB

We did develop two versions of this board, the second of which includes a JST connector for the Tx and Rx pins so that the Pi can communicate with other serial devices for extra functionality. In our case, it is integrated with an Arduino Uno microcontroller, in order for the software to talk to the Bonsai package [1].

Back PCB

Similarly to the Front PCB, we designed another custom board to route the wires to and from the sensors in the back of the cage. This Back PCB is relatively simple and small and provides JST-XH connections for the switches within the door levers, servos within the door levers, and the transmitters for the IR break beam sensors. The schematic and board diagram for this board are shown below.

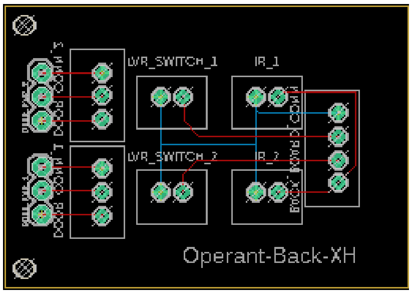

Fig. 3: Board Layout for the Back PCB

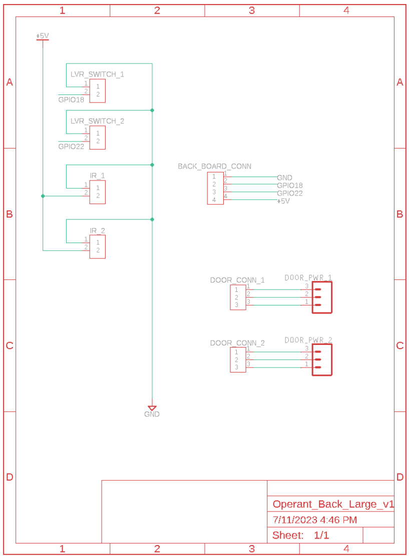

Fig. 4: Schematic for the Back PCB

Wiring

All of the wiring is routed under the base of the assembly to hide it and make sure that none of it gets in the way of the moving parts. The wires are all connected through male and female JST-XH connectors which are shown below. This allows for modularity in the design as well as ease of access for the user. To protect and hold the wires in place, we designed wire housings to be 3D printed and stuck to the underside of the cage. This makes sure that there are no loose wires that can get disconnected, kinked, or broken, and ultimately adds to the longevity of the design. The diagrams for the wire housings are shown below.

Raspberry Pi Software Setup

Prior to assembling all of the hardware and connecting the wires, it is important to set up the Raspberry Pi, since they do not typically have an OS installed out of the box. The first step is install the Raspberry Pi OS onto a microSD card, as provided by the official guides here https://www.raspberrypi.com/software/. Once this is set up, it is helpful to connect the Pi to a keyboard, mouse, and video monitor in order to directly interface with the Pi's desktop. After connecting the Pi to a monitor using an HDMI cable, it can be turned on by connecting to its 5V power supply. Note that if the Pi is powered before connecting to the HDMI cable, it may not be displayed on the monitor. Once the desktop is visible, navigate to the Terminal and enter the following commands, all in Python 3 and pip3: The first thing that should be done is to setup the Pi configuration

```
sudo raspi-config
```

And then in the Interface Options, turn on SSH, VNC, and Serial Port connections.

```
sudo apt update
sudo apt remove python3-numpy
sudo apt install libatlas3-base
sudo pip3 install numpy
```

Once these commands have been successfully completed, you may ensure that the packages are installed correctly by using a python3 command to check each one (including numpy).

```
python3
import <packageName>
```

To access the Operant Cage code, the GitHub repository must be cloned onto the Raspberry Pi, which is relatively straightforward for those familiar with git. Return to the terminal and run the following git command in the home directory to clone the repository.

```
git clone https://github.com/dprotter/RPI_operant.git
```

If there is a specific branch of the code that needs to be used, simply navigate into the newly cloned *RPI_operant* folder and enter

```
git checkout <branchName>
```

This will switch the working branch to the one specified, which can be verified using the following command:

```
git branch
```

Since this repository is still tied to GitHub, a file must be created in order for this individual Operant Cage's settings to be found and modified locally. To do this, once again open the Terminal and type in the following commands:

```
cd /etc
sudo mkdir RPI_operant
cd /home
cd RPI_operant
cp operant_cage_settings.py /etc/RPI_operant/operant_cage_set
```

This is where all of the local settings are stored for the electronic components, such as pin designations, servo speeds, and servo angles. To change the settings, go into the file using an in-line text editor on the Pi and change the numbers, and it will update all throughout the code base.
The last step is to create a directory for data records.

```
cd /home
mkdir test_outputs
```

This creates the directory where all of the logs will be stored from any testing and experiment files that are run. These steps work for the current version of the RPI_operant code base. As the code is modified and improved, the best place to find setup instructions will always be the4 README file on the GitHub repository.

It is also important to note that once the setup and install is complete, the monitor can be removed and the user can simply SSH into the command line of the pi using it's IP address. This IP address can be found by entering the following command into the Pi's Terminal:

```
hostname -I
```

Assembly

The following section will describe how to assemble the electrical components for the Operant Cage system. This should occur after the physical cage has been put together and fully assembled following the directions in the cage documentation CITE.

Raspberry Pi

Some minor assembly for the Raspberry Pi itself may involve attaching heat sinks (included with the Pi upon purchase) and inserting the SD card. Although the Servo Hat and custom front PCB bypass any need for wires to be directly connected to the Pi, all of the signals ultimately route to it, as shown by the pinout in the table below. Note that not all of the GPIO pins on the Pi are listed, and that is because these are the pins that come up through the Servo Hat and are available for the custom PCB to be connected to.

GPIO No.	Abbreviation	Description
4	D1_STATE	State switch on whether Door 1 is open or closed
6	D2_OPEN	Signal to tell Door 2 to open, interrupt switch
12	IR1	Signal line for IR sensor 1
13	IR2	Signal line for IR sensor 2
16	DISPENSE	Signal if a pellet has been dispensed/retrieved in the dispenser
17	D2_STATE	State switch on whether Door 2 is open or closed
18	LVR1	State switch for Lever 1, connecting to the back PCB
19	LED1	Signal wire for LED 1
20	LED2	Signal wire for LED 2
21	SPKR	Signal wire for the Speaker
22	LVR2	State switch for Lever 2, connecting to the back PCB
23	GPIO_SYNC	GPIO sync pin to send TTL pulses and ensure timing is correct
24	D1_OPEN	Signal to tell Door 1 to open, interrupt switch
27	LVR_FOOD	State switch on when the Food Lever is pressed

Table 1: Pinout description for the custom PCB and Raspberry Pi

All of the sensor pins are connected to the Pi's GPIO pins through the custom front PCB and all the servo wires are connected directly to the Servo Hat. These pins are defined in the software, as described in the previous section, and can be easily changed. The GPIO pins, however, are more difficult to modify than the servo pins since they rely on the design of the custom PCB. That said, be careful to ensure that any changes to the software settings are reflected by the hardware, especially within the custom front PCB. It is also important to note that almost all of the Pi's basic GPIO pins are used with this setup.

Servo Hat

This hat sits directly on top of the Raspberry Pi, with a double row of female pin headers connected to the bottom of the hat, as shown in the photo below.

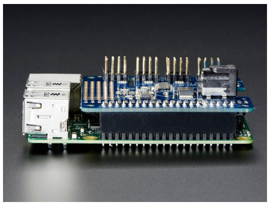

Fig. 5: Servo Hat connected to the Raspberry Pi

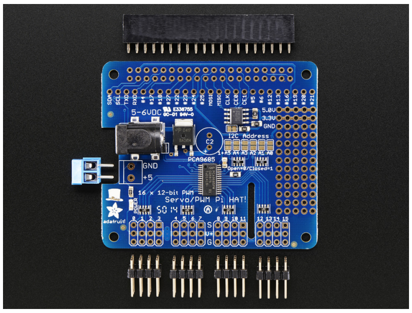

Fig. 6: Header pins to solder on

In the second photo, the male header pins to solder onto the top are shown in the four blocks at the bottom of the photo. These are soldered onto the top of the hat, where the servos will connect to. Then, so that the hat is level on top of the pi, M3 screws can be screwed into the holes on the front corners of the Pi for the hat to rest on. This just gives more support to the whole electrical assembly to reduce risk of header pins bending or breaking.

Custom PCBs

The custom PCB designs outlined earlier have two options to be manufactured. They can either be soldered in-house using solderable breadboards, or they can be printed by a third-party manufacturer. Regardless of how the board is manufactured, they will require some in-house manufacturing to attach header pins and JST connectors. It is important to note that two sizes of JST connectors have been used to attach the wires to the board, both XH and PH. This was just due to availability at the time of manufacturing, but it is generally easier to change out the XH size.

Third Party Printing

If sending files to a third party to print, the user should use the attached schematic and board files made in EAGLE to give whichever company used the necessary information to print the board. The lab uses JLCPCB to manufacture and ship the boards. The design files used are output from EAGLE as a collection of Gerber files, which can be found on the GitHub page as well.

Interface with Operant Cage

Once the front PCB and Servo Hat are assembled, they can be stacked on top of the Raspberry Pi. The Pi sits on the bottom of the stack, topped by the Servo Hat, and followed by the custom front PC, where all of the sensors are connected. This stack can be mounted on the front right side of the Operant Cage base, as explained in the Operant Cage documentation, using 3mm screws. A picture of the final setup is shown below, illustrating the controller stack.

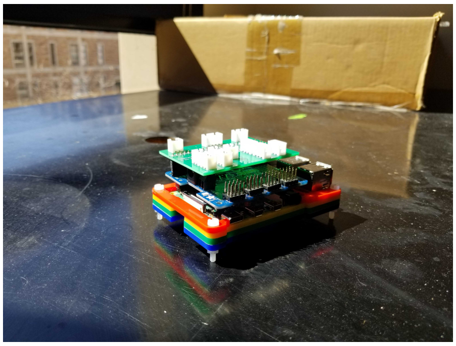

Fig. 7: Stacked headers from bottom to top: Raspberry Pi, Adafruit Servo Hat, Custom PCB

Connecting the Wires

Once the shield and PCB are set up, all of the servos and sensors need to be connected to the correct JST terminals on the board. There are a total of twelve sensors to connect between the PCB boards, which consist of three lever switches, five IR components, and four state switches for the doors.
In addition to the sensors, there are six servos on the system that need to be connected. As opposed to the sensors which connect to the custom PCBs, most of the servos connect to the Servo Hat on the pi, with two connecting to the back custom PCB. Then, those servos are routed through longer servo cables under the board to the servo hat on the front. All of the connections and the corresponding board are shown in the following table.

Sensor	**Board**
D1_STATE	Front
D2_STATE	Front
D1_OPEN	Front
D2_OPEN	Front
LVR1	Back
LVR2	Back
LVR_FOOD	Front
IR1 (Front)	Front
IR2 (Front)	Front
IR1 (Back)	Back
IR2 (Back)	Back
DISPENSE	Front

Table 2: Describes which sensor connects to which PCB

With the sensors and servos connected here, the back PCB and the front PCB need to be connected to each other. This is done through a single four-wire ribbon cable that is on the bottom of the back PCB and has a connection to the top of the front PCB.

Wire Holders

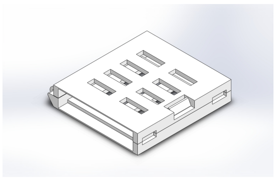

Fig. 8: Isometric view of the cable holder

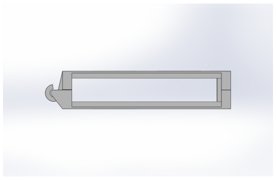

Fig. 9: Side view

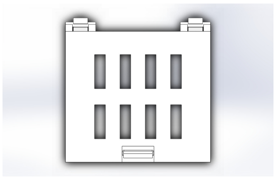

Fig. 10: Top view

Finally, the wire holders detailed above are glued onto the bottom of the cage using epoxy, ensuring lasting adhesion to the acrylic. In addition, these wire holders will provide support for the center of the cage, acting as 0.5" spacers to match the feet and preventing sag in the bottom piece of acrylic. Before the holder is attached to the cage though, the user should glue strips of foam to the opening edge of the holder, which will help keep the wires in place and prevent them from moving around when the cage is moved. Once the holder is glued in place and the foam is on, the wires can be clamped into the holder, and the holder latched shut.

Appendix A

Appendix B

References

[1] Bonsai Installation Guide. https://bonsai-rx.org/docs/packages/