Control System for a 7 Carousel Projector Show

Comissioned by Robin Cameron
Elizabeth Adelaide
March 20, 2017

Table of Contents

Summary

The system controls 7 carousel projectors. It advances each projector in order. The default operation advances each projector consecutively with equal timing, then waits to start another cycle. The system consists of a python-coded GUI, an Arduino, and a relay system.

Operation

There are 8 wires coming from the box. 7 are five pin connectors which connect to the projectors. The other wire is a power cable for the Arduino. The 7 pin connectors are labeled with flags made from black or green tape. The number of flags corresponds to the projector number. Wires labeled with black type are odd. Wires labeled with green are even.

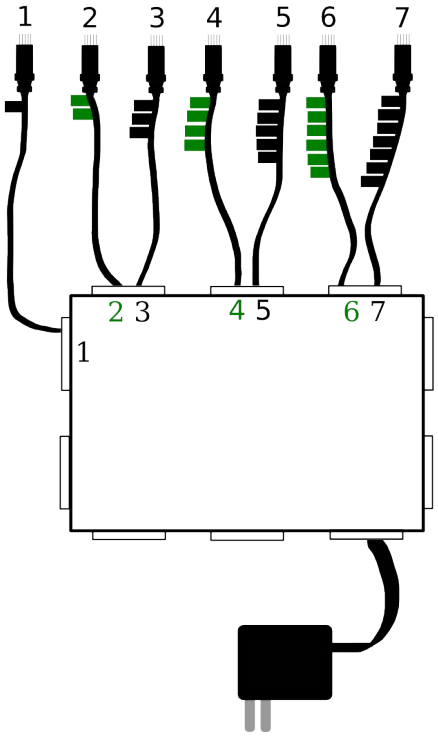

Figure 1: External Wiring Diagram

Figure 1 shows the external wiring diagram. The top-left corner of the controller box has one cable coming from the port. This is first projector. The next three ports have two wires coming from each. The projector increases for each port. Projector 1 will advance first, projector 7 will advance last in each sequence. There is a longer pause between projector 7 and projector 1 in default operation.

To setup, connect the five pin wires to the appropriate projectors. Turn on each projector and set them to the first slide. Then plug in the arduino. The projectors will then sequentially advance. *Table 1* shows the checklist for setting up operation each time.

Plug in five pin wire 1 (1 black flag) to projector 1	
Plug in five pin wire 2 (2 green flags) to projector 2	
Plug in five pin wire 3 (3 black flags) to projector 3	
Plug in five pin wire 4 (4 green flags) to projector 4	
Plug in five pin wire 5 (5 black flags) to projector 5	
Plug in five pin wire 6 (6 green flags) to projector 6	
Plug in five pin wire 7 (7 black flags) to projector 7	
Turn on projector 1, set to slide 1	
Turn on projector 2, set to slide 1	
Turn on projector 3, set to slide 1	
Turn on projector 4, set to slide 1	
Turn on projector 5, set to slide 1	
Turn on projector 6, set to slide 1	
Turn on projector 7, set to slide 1	
Plug in power adapter for controller	
Ensure that projectors are running in order	

Table 1: Checklist for operation

The controller should be plugged into a separate breaker than the projectors, or should be connected to a surge protector. A surge from the projectors could overload the arduino.

GUI Setup (Mac)

The GUI will allow you to change the timing of the arduino. The steps are to download the file, unzip it, run the shell scripts, and then open the file. The first step is to download the zip file. The zip file can be downloaded from github (https://elizabethadelaide.github.io/robinProjectorControl/). Go to the download direcrtory. This will either be the Desktop or the Downloads directory. Double click the file “projectorControl.zip”. The zip file will be extracted to a directory called “projectorControl”.

Next open up the terminal. To do so, press the search button (🔍) at the top right of the screen. Type “terminal” and press enter. A white window with a text prompt will appear. Go to the directory which the zip file was extracted to. This will either be the Desktop or the Download directory. In the terminal, type:

```
cd Desktop/projectorControl
```

For the desktop, then press enter, or type:

```
cd Downloads/projectorControl
```

For the Downloads directory, then press enter. Capitalization matters when typing in the terminal. You are now in the directory you extracted the file to. The next step is to change permissions for the scripts. To do so, type:

```
chmod 777 installone.sh installtwo.sh
```

Then press enter. Next, run the first install script by typing:

```
./installone.sh
```

Then press enter. A window will open up asking you to install xcode-development-tools. Click through the installation. When the installation finishes, return to the terminal. Run the second install script by typing:

```
./installtwo.sh
```

Then press enter. This will run a script which will prompt you to press return, and to enter your password. When the script finishes, close the terminal. In finder, go to the directory that you extracted the zip file to. In the directory named “projectorControl” there will be a file named “projectorControlApp”. Double click this file and a small window will open asking you to choose a port.

To update the app, run the same commands. The scripts will not prompt you as often.

Summary:

1. **Unzip File**
2. **Open terminal**
3. **`cd /path/to/directory/projectorControl`**
4. **`chmod 777 installone.sh`**
5. **`./installone.sh`**
6. **`./installtwo.sh`**
7. **Run projectorControlApp**

Using the GUI

The GUI allows you to alter the timing of the controller. It also you to stop, pause and resume the controller. While the controller is paused or stopped, you can advance an individual projector.

To use the GUI, first remove the top panel of the controller box. The top circuit board is the arduino. *Figure 2* shows a diagram of the arduino. Plug in a standard USB wire to the arduino, and connect to your computer.

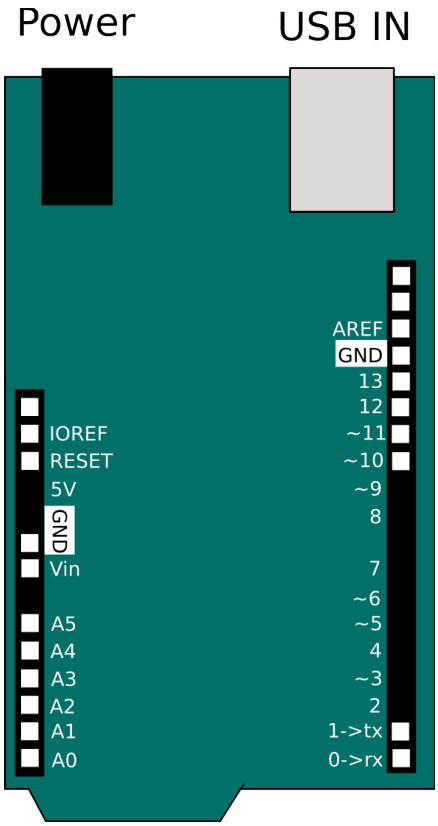

Figure 2: Arduino Diagram

Run projectorControl.app by double clicking it. The gui will open as a small window. Select the serial port that corresponds to your arduino. The serial port will resemble "/dev/tty.usbSerialMonitor". Often, there is another option that contains the word "bluetooth", this is likely a phone or bluetooth speaker and can be ignored. If the serial port is not valid, the button will turn red, this means the port is not connected correctly. If it is correct arduino port, try unplugging and plugging the USB wire in again. A valid serial port will turn green, and several more buttons will be shown. Note that the program can connect to something that is not an arduino. Try selecting a different serial port if the arduino is not responding.

Figure 3 shows the gui after a valid serial port is selected. Note that it shows sample ports, the names of the ports will differ from computer to computer.

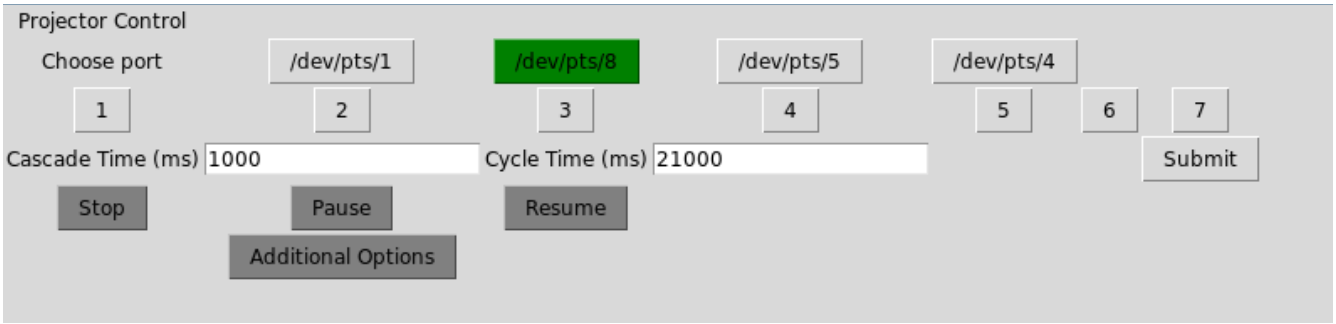

Figure 3: Sample GUI

To alter the timing of the arduino, change the values of the cascade time, and the cycle time. The cascade time is the time between each projector, and the cycle time it takes to start a new cycle. The GUI accepts inputs as milliseconds (ms), or thousandth of a second. 1 second is 1000 milliseconds.

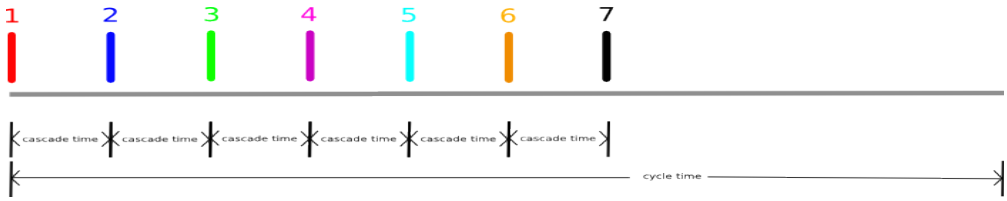

Figure 4: Timing diagram

Figure 4 shows the timing diagram. The cycle time can never be less than seven times the cascade time. The cascade time also must be more than the time each projector is pressed down, the default press time is 500ms (1/2 second).

The times are submitted to the arduino after pressing submit. The arduino will save these times to its memory, so that it will use the timing when it is turned on again. It does not need to be plugged in to a computer to run at the same timing. It does need to be plugged into a computer to update the timing.

Internal Operation

The internal system consists of an arduino uno microcontroller and a relay block. The relay block contains a 5V, 15-20mA interface, to match the arduino. The relays are rated for AC250V 10A, DC30V 10A. The projector control runs at 14V. A ribbon wire connects the relay interface to the arduino. Output pins on the arduino are 2-8, corresponding to projectors 1-7. Screw terminals connect the relay to the projector control. The red and yellow wire are connected to the normally open ports. The normally closed port is left disconnected. *Figure 5* shows the diagram for the projector control. Closing the circuit between the red and yellow wires will advance the projector. Reverse functionality could be added by using another bank of wires to close the circuit between the yellow and white wires. The brown wire controls the rack solenoid, and the black wire controls the focus.

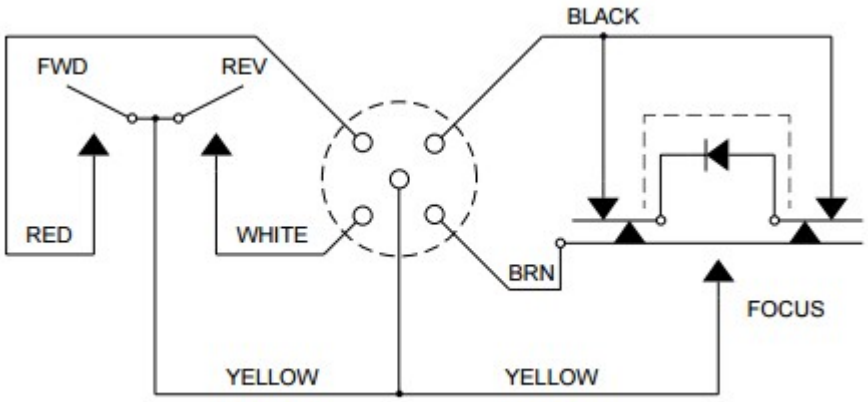

Figure 5: Projector Control Diagram[2]

Debugging

If there are issues with the system, go through a series of steps to identify and address issues.

Issue A: All Projectors Are Not Advancing

1. Make sure all projectors are plugged in and turned on.
2. Make sure controller is plugged in. There will be a green light on the power adapter. If there is no green light, check that the outlet is functional.
    a. If the outlet is functional and there is no green light, replace the power adapter:
        i. Unplug the adapter.
        ii. Remove the cover of the controller.
        iii. Carefully unplug the adapter from the arduino.
        iv. Remove the adapter from the rubber gasket.
        iv. Take the spare adapter, and plug into the outlet. Check that the light turns on.
        v. Unplug the adapter, and push the small end through the rubber gasket (port G), towards the interior of the controller.
        vi. Plug the small end into the arduino
        vii. Plug in the adapter into the outlet.
        viii. Check that the arduino turns on, and that the relay board has red lights.
        ix. If the adapter fails, or there is still no power to the arduino, see **Appendix C: Installing the spare controller.**
3. Unplug the controller, and remove the cover. Plug in the controller. Check if there is a green light from the arduino, and flashing red lights from the relay. If there are not see **Appendix C: Installing the spare controller.**
4. If there are green lights from the arduino, and not from the relay board, carefully push down the wires from the ribbon cable that enter the arduino. If any are not connected, see the **Appendix A: Arduino wiring guide.**

5. If there is no issue with the arduino, carefully lift the arduino with its plastic base out of the controller box. Make sure no wires get loose while doing so. Carefully push down the wires on the relay board. If any are not connected, see the **Appendix B: Relay wiring guide.**
6. If the issue has not been addressed, see **Appendix C: Installing the spare controller.**

Issue B: One or Some of the projectors are not advancing

1. Identifying the projector. The projectors are sequenced from 1-7, with a pause before repeating the cycle. Unplugging and replugging in the controller will start the sequence at 1. Additionally, the GUI can be used to pause the controller, and to individually specify a projector to advance, see **Using the GUI.**
2. Disconnect the projector from the controller. Connect the projector to a manual remote. Check if the projector advances. If it does not, replace it with the spare projector. If it does, reconnect the controller wire.
3. Unplug the controller. Remove the cover of the controller. Plug the controller back in. There are a series of red lights on the relayboard. These lights flash when the projector is advanced. Either using the GUI, or by waiting through a sequence, check if the light flashes for the non-operating projector(s).
    A. The light doesn't flash:
        i. Carefully check the wires on the arduino. If there is a loose or disconnected wire, see the **Appendix A: Arduino wiring guide.**
        ii. Carefully lift the arduino and its plastic base from the controller box. Make sure no wires are pulled as you do so. Carefully check the wires coming into the relay board. If there is a loose or disconnected wire, see the **Appendix B: Relay wiring guide.**
        iii. If there are no loose wires, see **Appendix C: Installing the spare controller**.
    B. The light does flash:
        i. Unplug the controller.
        ii. Carefully remove the arduino and its plastic base from the controller box. Check the wires coming into the screw terminals. If any wire is loose or disconnected, see the **Appendix B: Relay wiring guide**.
        iii. Double check that the projector is connected to the controller.
        iv. If there is no loose wire, see **Appendix C: Installing the spare controller**

Issue C: Cannot connect to the serial port in GUI

1. Close the GUI. Unplug the arduino from both the adapter and the USB. Plug in the USB again. Start the GUI. Check if the port appears now.
2. Unplug the arduino and restart your computer. Plug in the arduino then open open the GUI.
3. Install the arduino sofware (https://www.arduino.cc/en/guide/macOSX). Open the program, and go to “tools > Serial Port”. Check that the serial port appears. If it does not, remove the cover to the controller box. Check that there are green lights from the arduino and red lights from the relay. If there is not, follow the steps for **Issue A: None of the projectors are advancing**.

Appendix A: Arduino Wiring Guide

1. Carefully inspect the wires on the arduino. If any wire is loose or disconnected, examine the wire and make sure it is not broken.
2. If it is not broken replace it. See *Figure A1* for a wiring diagram. See *Table A1* for a chart of where wires can be connected.
3. If the wire is broken, make sure any loose metal pieces are removed from the controller. Use a spare male to female wire to connect the missing connection. Connect this wire to the corresponding connection on the relay. See **Appendix B: Relay Wiring Guide.** Note that while the color of the wire will not affect operation, it may assist future debugging to use the same color.

Wire Color	**Arduino Connection**
Black	GND
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Purple	7
Brown	5V

Table A1: Wiring Connections

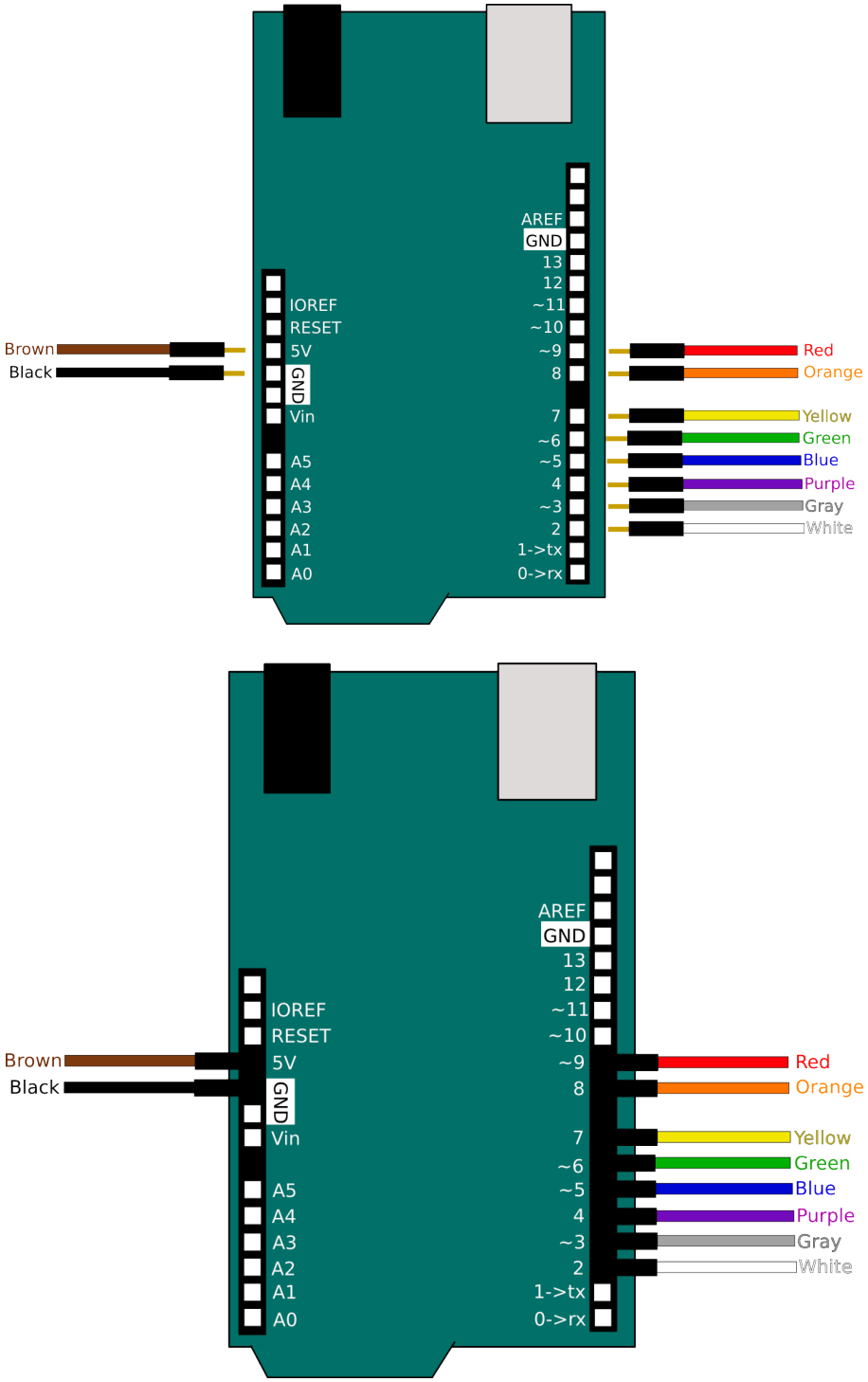

Figure A1: Arduino wiring diagram

Appendix B: Relay Wiring

1. Carefully inspect the wires on the relay. If any wire is loose or disconnected, examine the wire and make sure it is not broken.
2. If it is not broken replace it. See *Figure B1* for a wiring diagram. See *Table B1* for a chart of where wires can be connected.
3. If the wire is broken, make sure any loose metal pieces are removed from the controller. Use a spare male to female wire to connect the missing connection. Connect this wire to the corresponding connection on the arduino. See **Appendix A: Arduino Wiring Guide.** Note that while the color of the wire will not affect operation, it may assist future debugging to use the same color.

Wire Color	**Relay Connection**
Black	GND
Red	In1
Orange	In2
Yellow	In3
Green	In4
Blue	In5
Purple	In6
Brown	Vcc

Table B1: Wiring Connections

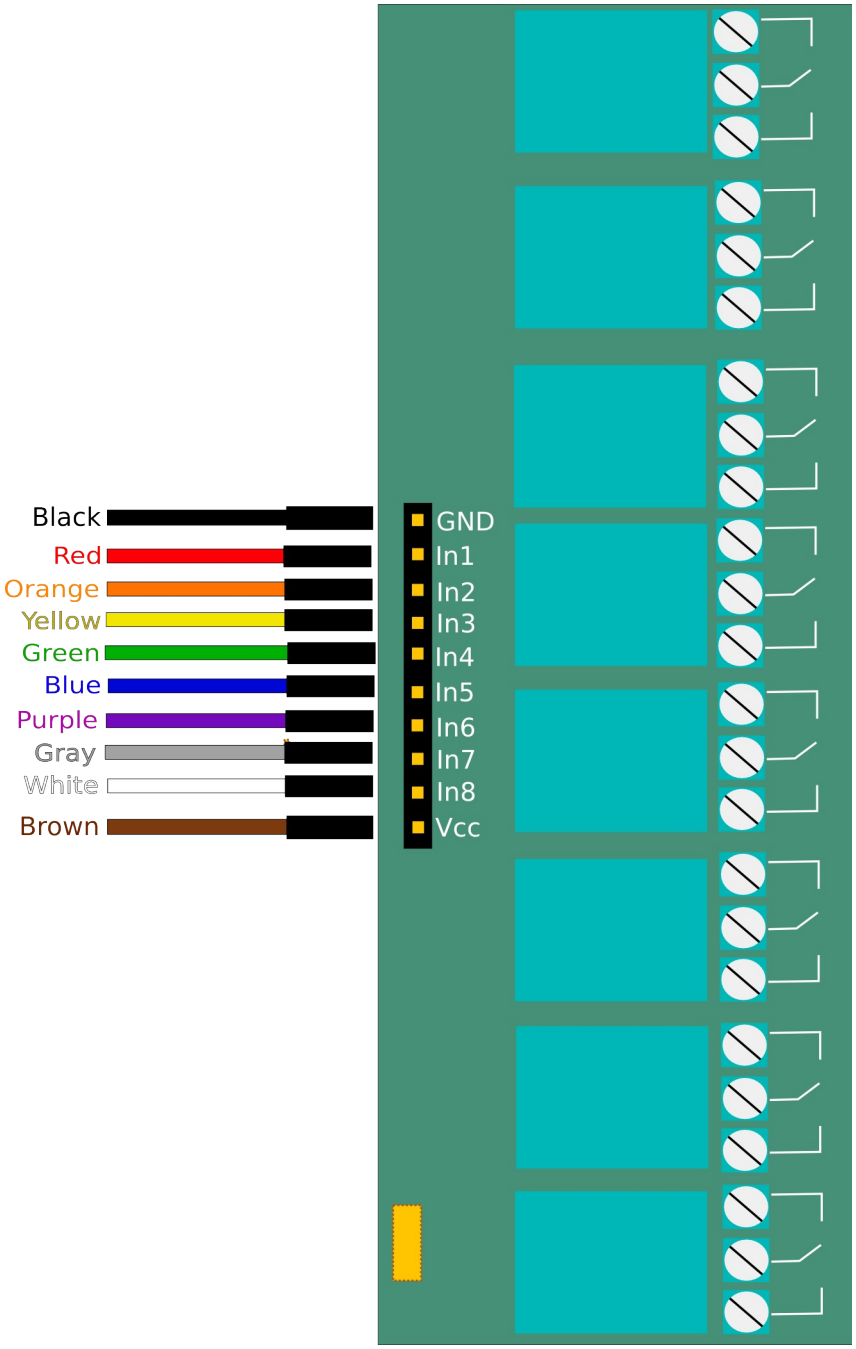

Figure B1: Relay wiring guide

Appendix C: Installing the Replacement Controller

The replacement controller contains an arduino and relay. The replacement requires the projectors to be connected and the AC adapter to be plugged in. Steps:

1. Open the replacement controller box. Push the AC adapter through port G of the box. Connect the adapter to the arduino. Plug the adapter into an outlet that is known to work. Confirm that green lights turn on on the arduino, and that red lights turn on on the relay.
2. Unplug the adapter from the outlet.
3. Carefully remove the arduino and its plastic base. Ensure that no wires are pulled in the process.
4. On the original box, use the small screwdriver to loosen the connections to the projectors. Carefully remove the wires from the screw terminals, and pull out of the box.
5. Examine the wires. If any are broken or too tangled to use, cut the entire wire. Strip off one inch of the black rubber shielding. Cut the brown, white and black smaller wires. Strip of an eight of inch of the red and yellow wires. Twist the copper wiring to make a stiff connection. See *Figure C1*.

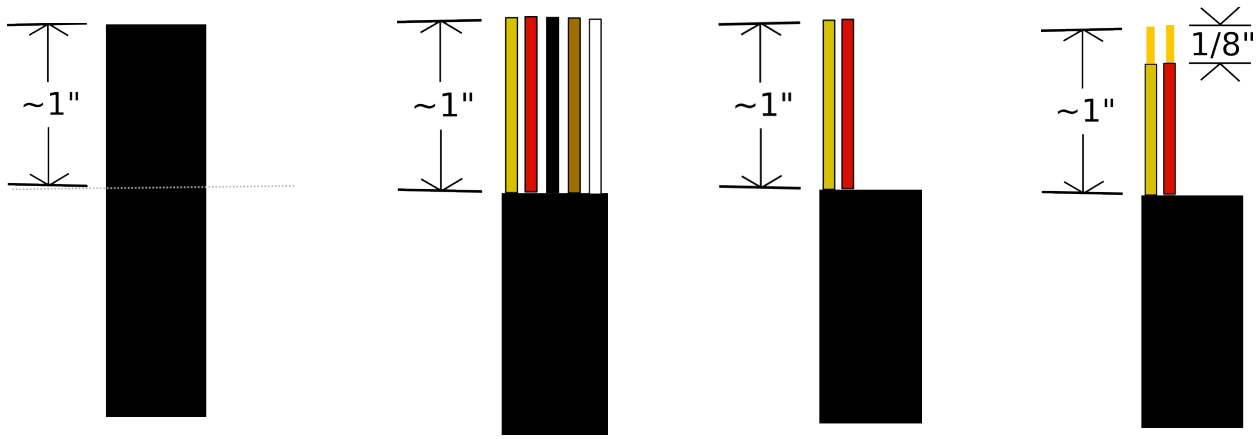

Figure C1: Stripping the projector wire

6. Push the wires through the ports as seen in *Figure 1*.
7. Loosen the normally open screw terminals in the replacement relay. These are the same terminals that were used in the original relay. They are the center terminal for each relay, and the terminal which has a line going away from it. See *Figure C2*.

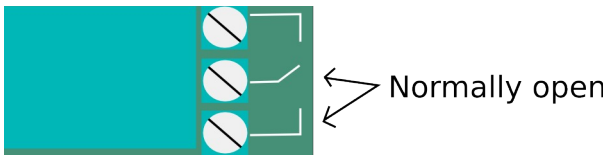

Figure C2: Normally open screw terminals

9. Carefully place the stripped wires into the terminals. The red wires for each projector go into the center terminal. The yellow wires go into the normally open terminal. See *Figure C3* for the full wiring diagram.

10. Tighten the terminals. Check that the wires are not loose.
11. Test the replacement box by turning on the projectors and plugging in the controller.

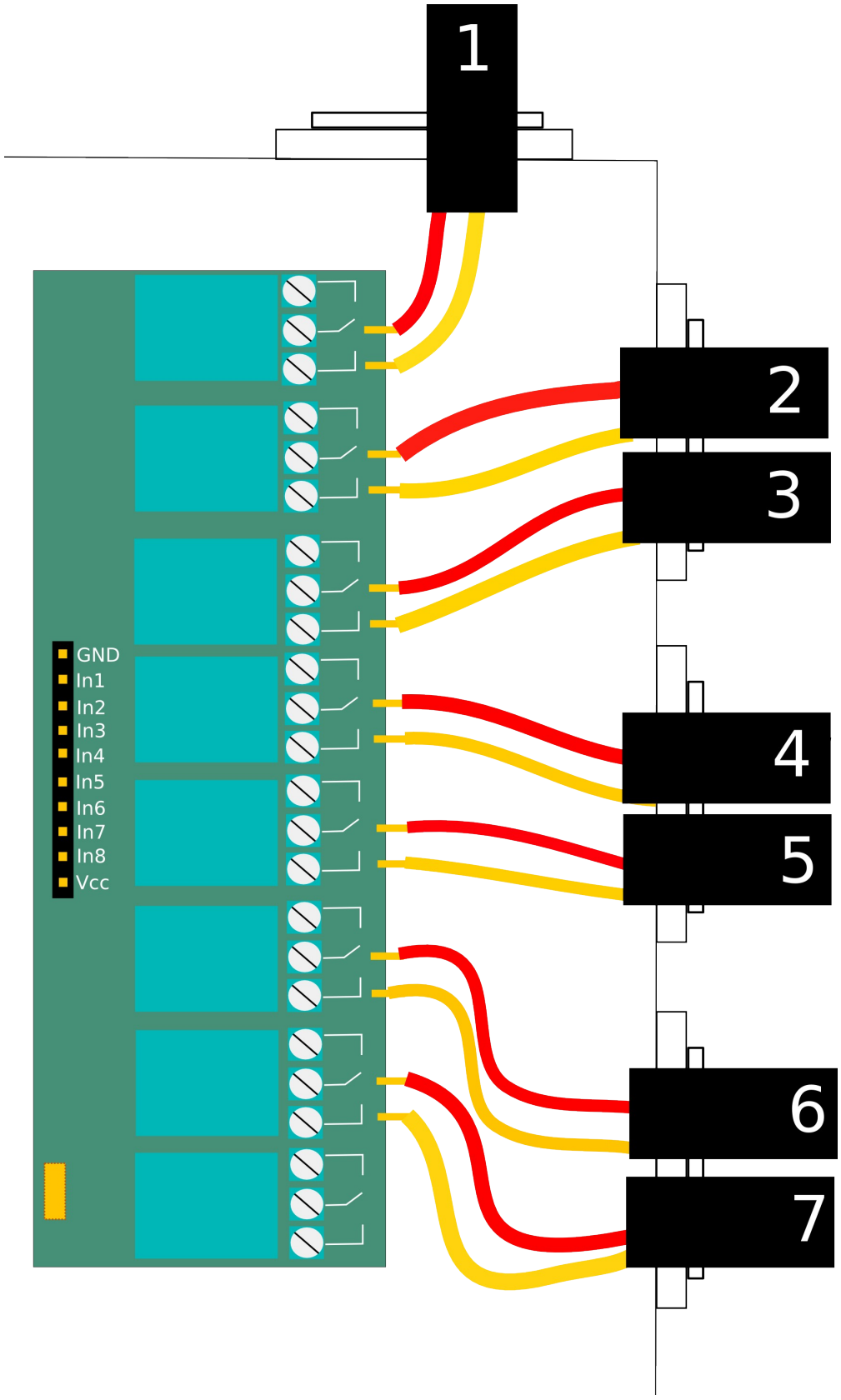

Figure C3: Wiring Diagram

Appendix D: Arduino Code

```
#include<EEPROM.h>
#define NUMPROJS 7

//Elizabeth Adeliade, March 2017
//Commissioned by Robin Cameron
//Controls a seven projector system
//Interfaces with a GUI using serial ports

int projectorPins[NUMPROJS] = {2,3,4,5,6,7, 8};
int timer[NUMPROJS] = {1000, 1000, 1000, 1000, 1000, 1000, 5000}; //time between each relay

int cascadeTime = 500; //time between each projector
int cycleTime = 10000; //time to start a new cycle

int clickTime = 250; //time to click down

const byte numChars = 32;
char receivedChars[numChars];

boolean newData = false;

//for EEPROM
int eeaddress = 0; //start from byte 0
int value[512];

void setup() {
  // put your setup code here, to run once:
  int i;
  int m;
  int temp;

  //initialize serial:
  Serial.begin(9600);

  //initialize projector pins
  for(i = 0; i < NUMPROJS; i++){
     pinMode(projectorPins[i], OUTPUT);
     digitalWrite(projectorPins[i], HIGH); //relays are open with high input
  }

  while (eeaddress < 512){
    value[eeaddress] = EEPROM.read(eeaddress);
    if (value[eeaddress] == '\n'){
      break;
    }
    eeaddress = eeaddress + 1;
  }

  if (eeaddress == 29){ //correctly read/valid data
    eeaddress = 0;
    for (i = 0; i < NUMPROJS; i++){
      temp = 0;
      for (m = 0; m < 4; m++){
        temp = temp + (value[eeaddress] * 255^(m));
        eeaddress++;
      }
      timer[i] = temp;
    }
  }

}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int ind;


  //go through each projector:
  for (i = 0; i < NUMPROJS; i++){
    receiveData();
    ind = processData(i);
    showNewData();

    //Serial.print("Ind: ");
    Serial.print(ind, DEC);
    Serial.print("\n");
    //Serial.println(i);

    switch(ind){
      case 10: //stop
        ind = stopProj();
```

```
        i = 0;
        break;
      case 13: //rewrite times
        Serial.println("Rewrite");
        rewriteTimes();
        break;
      case 11: //pause
        ind = stopProj();
        i = i;
        break;

      default:

     break;
    }
    digitalWrite(projectorPins[i], LOW);

    delay(clickTime);//give enough time to click
    digitalWrite(projectorPins[i], HIGH);
    delay(timer[i]);
  }
}

int rewriteTimes(){
  int i;
  int m;
  int newCascadeTime = 0;
  int newCycleTime = 0;
  int temp;
  uint8_t writevalue;

  for (i = 2; i <= 5; i++){
    temp = ((int)((uint8_t)receivedChars[i]));
    newCascadeTime += temp* pow(256, 5-i);

  }
  Serial.print("\n");
  for (i = 6; i <= 9; i++){
      temp = ((int)((uint8_t)receivedChars[i]));
    newCycleTime += temp* pow(256, 9-i);

  }
  cycleTime = newCycleTime;
  cascadeTime = newCascadeTime;

  if (cycleTime < NUMPROJS*cascadeTime){
    cycleTime = NUMPROJS * cascadeTime;
  }
  if (cycleTime < NUMPROJS*clickTime){
    cycleTime = NUMPROJS * clickTime;
  }

  for (i = 0; i < NUMPROJS- 1; i++){
       timer[i] = cascadeTime - clickTime;
    }
    //time between starting a new cycle
    timer[NUMPROJS - 1] = ((cycleTime) - ((NUMPROJS -1) * cascadeTime)) - clickTime;

  eeaddress = 0;
  //TOdo: write to EEPROM/
  for (i = 0; i < NUMPROJS - 1; i++){
    temp = timer[i];
    for (m = 0; m < 4; m++){
      writevalue = temp % 255;
      temp = floor(timer[i] / 255);
      EEPROM.write(eeaddress, writevalue);
      eeaddress++;
    }
  }
  EEPROM.write(eeaddress, '\n');
}


int stopProj(){
  int ind = 10;

  //Serial.println("Stopped");
  while (ind != 12){
    //Serial.println(ind);
    //delay(500);
    receiveData();
    ind = processData(ind);
    showNewData();

    if (ind >= 0 && ind < NUMPROJS){
      digitalWrite(projectorPins[ind], LOW);
      delay(clickTime);
      digitalWrite(projectorPins[ind], HIGH);
      ind = 10;
    }
```

```
  }
  return ind;
}

int processData(int index){
  int indicator;
   int i;

   indicator = index;


   if (newData){
     //Serial.println("Indicator");
     indicator = 10*((int)receivedChars[0] - 48)+ ((int)receivedChars[1] - 48);

     Serial.print("Indicator");
     Serial.print(indicator);
     Serial.print('\n');
     delay(500);

   }

   return indicator;
}

void receiveData(){
  static byte ndx = 0;
  char endMarker = '\n';
  char rc;

  //Serial.println(Serial.available());
  //Serial.println(newData);
  while (Serial.available() > 0 && newData == false){
    //Serial.println("info received");
    rc = Serial.read();
    if (rc!= endMarker){
       receivedChars[ndx] = rc;
       ndx++;
       if (ndx >= numChars){
          ndx = numChars - 1;
       }
    }
    else{
      receivedChars[ndx] = '\0';
      ndx = 0;
      newData = true;
    }
  }
}

void showNewData() {
 if (newData == true) {
   Serial.print("Received:");
   Serial.println(receivedChars);

   newData = false;
 }
}
```

Appendix E: GUI Code

Python Code:

```
import serial
import sys
import glob
import re
from tkinter import *

#Elizabeth Adelaide, March 2017
#Comissioned by Robin Cameron
#Python GUI for interfacing with arduino system using serial communication

def serial_ports():
    """ Lists serial port names

        :raises EnvironmentError:
            On unsupported or unknown platforms
        :returns:
            A list of the serial ports available on the system
    """
    if sys.platform.startswith('win'):
        ports = ['COM%s' % (i + 1) for i in range(256)]
    elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        # this excludes your current terminal "/dev/tty"
        ports = glob.glob('/dev/tty[A-Za-z]*')
        ports.extend(glob.glob('/dev/pts/*')) #for virtual ports
    elif sys.platform.startswith('darwin'):
        ports = glob.glob('/dev/tty.*')
    else:
        raise EnvironmentError('Unsupported platform')

    result = []
    for port in ports:
        try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
        except (OSError, serial.SerialException):
            pass
    return result


if __name__ == '__main__':
    print(serial_ports())

ports= serial_ports() #detect serial ports
for x in range(len(ports)):
          print(ports[x] + " " + str(x))



class PortChoose(Frame): #initial select port GUI
          def __init__(self, master=None):
                    Frame.__init__(self, master)
                    self.grid()

                    self.portLabel = Label(master, text="Choose port")
                    self.portLabel.grid(column=0, row=1)

                    self.portButton = [0 for x in range(len(ports))]
                    for x in range(len(ports)):
                              self.portButton[x] = Button(master, command = lambda x=x: self.selectPort(x), text=ports[x]);
                              self.portButton[x].grid(column=x+1, row=1)

          def selectPort(self, x):
                    try:
                              global ser
                              ser = serial.Serial(ports[x])
                              print("Connected to port " + ports[x])
                              self.portButton[x].configure(bg="green")
                              self.quit()
                    except serial.SerialException:
                              print("port unavaible")
                              self.portButton[x].configure(bg="red")


class GUI(Frame):

          def __init__(self, master=None):
                    Frame.__init__(self, master)
```

```
        self.grid()

        self.TitleLabel = Label(master, text="Projector Control")
        self.TitleLabel.grid(column=0, row=0)

        self.cascadeLabel = Label(master, text="Cascade Time (ms)")
        self.cascadeLabel.grid(column=0, row=3)

        self.cascadeEntry = Entry(master)
        self.cascadeEntry.grid(column=1, row=3)
        self.cascadeEntry.insert(END, "1000")

        self.cycleLabel = Label(master, text="Cycle Time (ms)")
        self.cycleLabel.grid(column=2, row=3)

        self.cycleEntry = Entry(master)
        self.cycleEntry.grid(column=3, row=3)
        self.cycleEntry.insert(END, "21000")

        self.cascadeButton = Button(master, command=self.cascadeClick, text="Submit")
        self.cascadeButton.grid(column=6, row=3)


        self.stopButton = Button(master, command = self.stopClick, text="Stop")
        self.stopButton.grid(column=0, row=4)

        self.resumeButton = Button(master, command = self.resumeClick, text="Resume")
        self.resumeButton.grid(column=2, row=4)

        self.pauseButton = Button(master, command = self.pauseClick, text="Pause")
        self.pauseButton.grid(column=1, row=4)

        self.advancedButton = Button(master, command = self.advancedClick, text="Additional Options")
        self.advancedButton.grid(column=1, row=5)
        self.advancedBool = 1
        self.advancedButton.configure(bg="gray")

        self.clickLabel = Label(master, text="Click Time(ms)")

        self.clickEntry = Entry(master)
        self.clickEntry.insert(END,"500")


        self.stopButton.configure(bg="gray")
        self.pauseButton.configure(bg="gray")
        self.resumeButton.configure(bg="gray")

        self.ProjNum= [0 for x in range(7)]
        for x in range(7):
                self.ProjNum[x] = Button(master, command= lambda x=x: self.selectProj(x), text=x+1)
                self.ProjNum[x].grid(column=x, row=2)


def cascadeClick(self):

        casc = self.cascadeEntry.get()
        cycle = self.cycleEntry.get()
        click = self.clickEntry.get()

        if (casc.isdigit() and cycle.isdigit() and click.isdigit()):
                cascnum = int(casc)
                cyclenum = int(cycle)
                clicktime = int(click)

                if (cyclenum < clicktime*7):
                        cyclenum = clicktime*7
                        self.cycleEntry.config(textvariastr(cyclenum))
                if (cyclenum < cascnum*7):
                        cyclenum = cascnum*7
                        self.cycleEntry.config(textvariable=str(cyclenum))

                ser.write(b'13') #code for changing time
                num = cascnum.to_bytes(4, byteorder='big')

                ser.write(num)

                num = cyclenum.to_bytes(4, byteorder='big')
                ser.write(num)
                ser.write(b'\n')

                print("Submitted")


def resumeClick(self):
        print("Resume")
        ser.write(b'12')
        ser.write(b'\n')
        self.stopButton.configure(bg="gray")
        self.pauseButton.configure(bg="gray")
        self.resumeButton.configure(bg="green")
```

```
        def stopClick(self):
                print("Stop")
                ser.write(b'10')
                ser.write(b'\n')
                self.stopButton.configure(bg="red")
                self.pauseButton.configure(bg="gray")
                self.resumeButton.configure(bg="gray")
        def pauseClick(self):
                print("Pause")
                ser.write(b'11')
                ser.write(b'\n')
                self.stopButton.configure(bg="gray")
                self.pauseButton.configure(bg="red")
                self.resumeButton.configure(bg="gray")
        def selectProj(self,x):
                print("Proj " + str(x))
                num = ("%02d" % x).encode()
                ser.write(num)
                ser.write(b'\n')
        def advancedClick(self):
                if (self.advancedBool):
                        self.advancedButton.configure(bg="green")
                        self.clickEntry.grid(column=2, row=6)
                        self.clickLabel.grid(column=1, row=6)
                        self.advancedBool=0
                else:
                        self.advancedButton.configure(bg="gray")
                        self.clickLabel.grid_forget()
                        self.clickEntry.grid_forget()
                        self.advancedBool=1


portFrame = PortChoose()

portFrame.mainloop()

guiFrame = GUI()

while True:
        line = ""
        if (ser.in_waiting):
                try:
                        line = ser.readline().decode("utf-8")
                        reg = re.compile('[0123456789]') #only read numbers from serial
                        number = ''.join(list(filter(reg.search, line))); #get number from serial

                        if (number.isdigit()):
                                number = int(number)
                                for i in range(0, 7):
                                        if (number == i):
                                                guiFrame.ProjNum[i].configure(bg="red")
                                        else:
                                                guiFrame.ProjNum[i].configure(bg="gray")


                except serial.SerialException:
                        print("Arduino temporarily not available")

        #if (number !=''):
        #       print(number)



        guiFrame.update_idletasks()
        guiFrame.update()
```

Mac Installation Code

installone.sh:

```
#!/bin/bash

#install xcode commandline tools
command -v ruby >/dev/null 2>&1 || xcode-select --install
```

installtwo.sh:

```
#!/bin/bash
#install homebrew as a package manager
command -v brew >/dev/null 2>&1 || ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

#add homebrew to path variable
command -v brew >/dev/null 2>&1 || export PATH=/usr/local/bin:/usr/local/sbin:$PATH

#install python3
command -v python3 >/dev/null 2>&1 || brew install python3

#install py2app
pip3 install -U git+https://github.com/metachris/py2app.git@master

#install tkinter
brew install python3-tk

#install serial libary
pip3 install pyserial

#install
py2applet --make-setup projectorControl.py
python3 setup.py py2app

ln -s dist/projectorControl.app projectorControlApp.app

chmod 777 projectorControl.app
```

Image Sources

1: Kodak. *Service Manual: Models 4200, 4200-J, 4200-kk, 4400, 4600, 4600-KK, 5600, 5600-J, and 5600-KK.* 1999. pg. 43 .Retrieved from: http://slideprojector.kodak.com/files/h5R2ls/carousel_sm.pdf