Shutter Controller

Replacement for Foster System Shutter Controller

# Overview

Since the initial implementation of my Exploradome I have been frustrated by the Foster System shutter controller. The Rotation controller has worked well, although it seems to have some minor problems working with ACP. Also, the hardware seems to work well in both the rotation and shutter units. However, the shutter software is error prone and has been difficult to work with. For the last 3 revisions of the Foster software my shutter has been completely inoperable. So, I decided to build a replacement unit for the shutter controller. I continue to use the Foster unit for rotation; the new unit only handles opening and closing the shutter.

My system is an Exploradome I 8 foot dome with Foster Automated Rotation and Shutter controller system. I use ACP for imaging. The observatory has a dedicated HP desktop computer (Win 7 64 bit). The observatory is generally operated remotely.

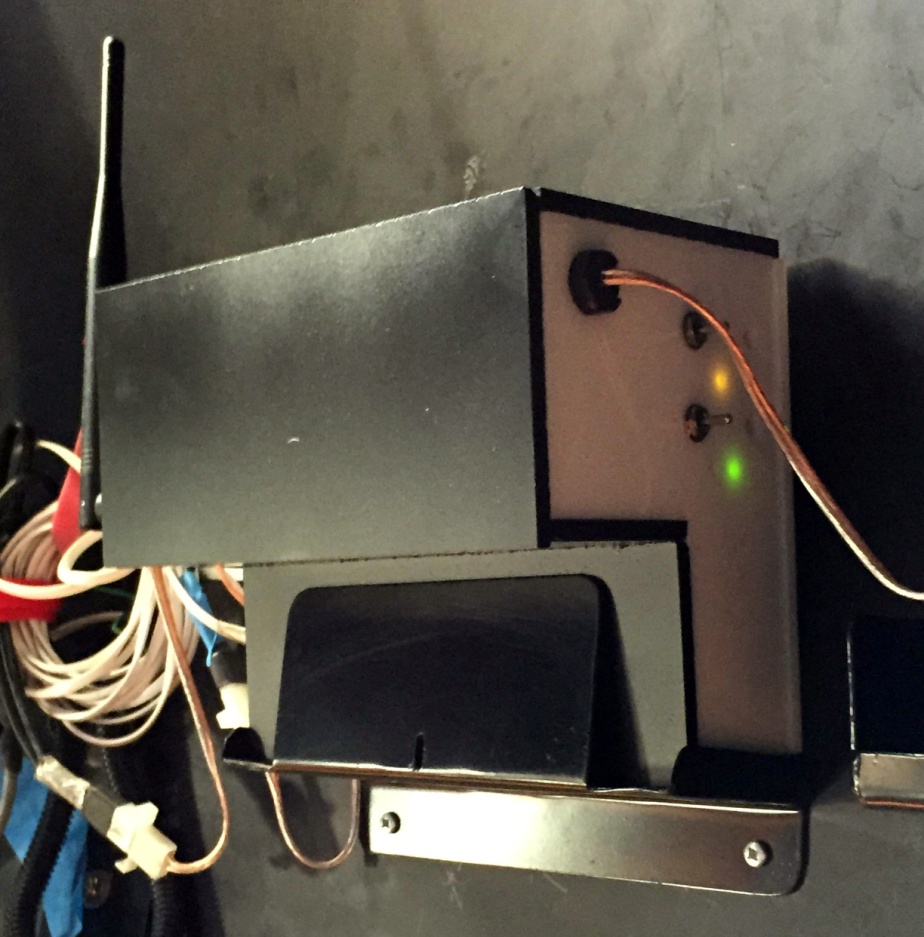


Figure . Controller mounted on dome wall. The two switches allow manual operation. The Yellow LED flashes indicating internet communications; the green LEDs indicate the shutters are closed.



Figure . Opposite side of controller, showing the WiFi antenna, power to shutter motors, and RJ11 connections for shutter limit switches.

# Design Points

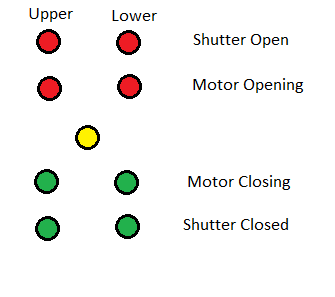
* The shutter controller is based on an Arduino Atmega 2560. I started with an R3 Uno, but discovered that the shields used all of the available pins. I need a number of pins for status LEDs, shutter switches, etc.

The Arduino uses 2 shields.

1. One is the Adafruit cc3000 shield which provides wireless internet access from a computer. It uses the Adafruit\_cc3000 library to implement a server in the Arduino.
2. The second shield is the Monster Motor shield from Sparkfun. This shield provides the high current required by the DC motors in the shutter. I was very nervous when I first connected power to the shield, but it worked perfectly. Sparkfun provides a sample program showing how to work with the shield. I basically created functions to run the motors based on their code.

* The controller is encased in a box made from 1/4 inch plexiglass. It turns out that plexiglass is somewhat translucent, and the Motor shield has a very bright LED indicating power. This caused the entire box to glow red, which is not good in the observatory! I fixed this by a) placing a small piece of tape over the LED, and b) painting the white plexiglass with black spray paint.

Figure . Controller cover painted black to block LED light.

* There are 9 small (3 mm) status LEDs on the side of the box. I was concerned that these would be too bright, but the 1000 ohm resistor keeps the current low enough that they are suitably dim. The LEDs mirror the lights in the program SimpleShutter.

One yellow LED is the "heartbeat ". When power is applied it is steady for 15-20 seconds while the Arduino connects to the network. Subsequently it flashes roughly once per second when it checks for incoming commands.

Each motor (upper and lower shutter) has 4 LEDs. The top red LED indicates the Open limit switch is active. The bottom green LED indicates the Closed switch is active. The other red LED is on while the motor is running to open the shutter. The second green LED indicates the motor is running to close the shutter.

* Two SPDT switches on the side of the box allow manual manipulation of the shutters. Opening and closing operations correctly stop based on the limit switches, but there are no other checks on the shutter positions. For example, you could potentially try to open the lower shutter while the upper shutter is still closed; this can cause shutter damage. The assumption is that manual operation means you are present and not doing something stupid.
* Power to the Arduino comes from the same 35A power supply used to perform dome rotation. The 12V supply comes into the terminal block on the base of the box. From there, the 12V runs directly to the Monster Motor shield for application to the motors.

The 12V supply is also routed to the small circuit board, where a SWADJ3 voltage regulator takes the voltage down to about 7.2V. This voltage is then supplied to the Arduino via the Vin pin. The Arduino can be supplied by 12V directly, but this might generate a lot of heat (the Arduino includes its own voltage regulator to get the voltage down to 5V). The Arduino needs about 1-2 amps to run the itself and the two shields. The cc3000 in particular can draw momentary power when running its radio; too little power can lead to strange internet communication problems.

* The small circuit board also includes a TMP36 temperature sensor. It is powered by the 5V pin on the Arduino, supplying a voltage to pin A10. The SimpleShutter program displays the temperature in degrees Fahrenheit. See the instructable on the Adafruit site for using the TMP36 chip.
* The small circuit board also uses a small voltage divider across the 12V supply to run a voltage to pin A8. This allows SimpleShutter to display the current voltage to the controller.
* Note that the shutter limit switches are normally closed; they become open when the switch is engaged. In the RJ11 connector pins 1&4 reflect the Shutter Open switch while pins 2&3 reflect the Shutter Closed switch. I imagine that different installations could be different:)

# Construction Notes

* The box is sized to fit the existing bracket already mounted on the dome. The top portion is thicker than the original Foster controller to accommodate the height of the Arduino stack (the shields add significant height, especially because I used double headers to connect the shields). This served two purposes:

1. this leaves more space between the shields. I am nervous about generating too much heat in the shields.
2. The shields can have conflicts, needing to use the same pins. In this case the two shields only have one pin in conflict- they both expect to use pin 10. My solution was to cut the header pin for pin 10, so pin 10 does not propagate to the motor shield. I then connect pin 45 to pin 10 on the motor shield. The Arduino sketch then uses pin 45 instead of pin 10, but the shield sees the signal on pin 10.

* I had to manually cut holes for the RS232 ports using a coping saw, then sizing them using a Dremel tool. Run the Dremel on low speed to avoid having the plexiglass melt.

I also had to use the Dremel to enlarge 1/2" holes drilled for the strain relief grommets. Strangely, 1/2" holes were too small for the grommets while 5/8" holes were too large.

* My original manual switches did not have a long enough threaded shaft to reach through the plexiglass. I had to find switches with a 3/8" shaft to allow the mounting nut to screw on.
* The status LEDs are connected to the Arduino using extension jumper cables. Unfortunately the LED wires do not fasten tightly to the connector, so I added a bit of glue from a hot glue gun to hold the wire to the connector.

# Construction Details

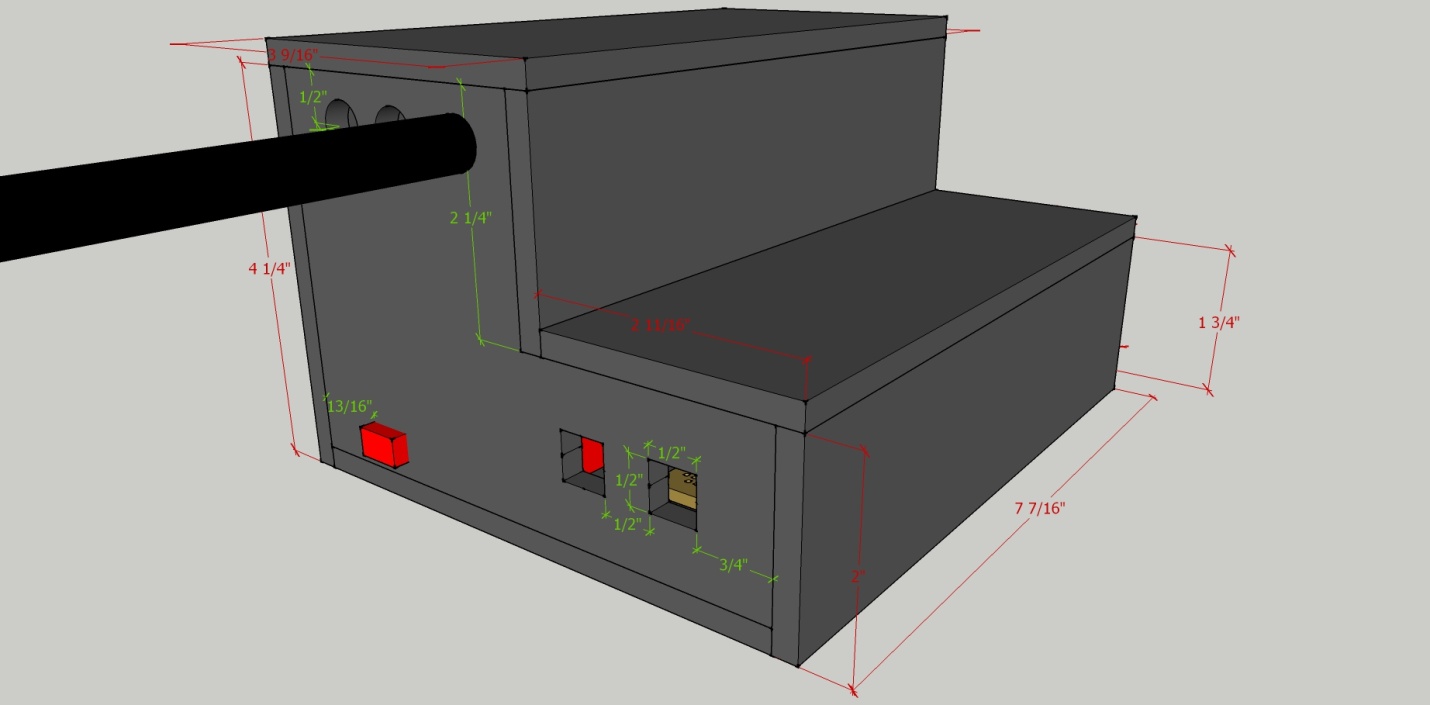


Figure . Controller dimensions.

Here are views of the interior of the box. I wired directly to the LEDs and manual switches rather than using an intermediate circuit board. I used nylon screws to attach the Arduino and circuit board to the box. The RJ11 connectors were glued.

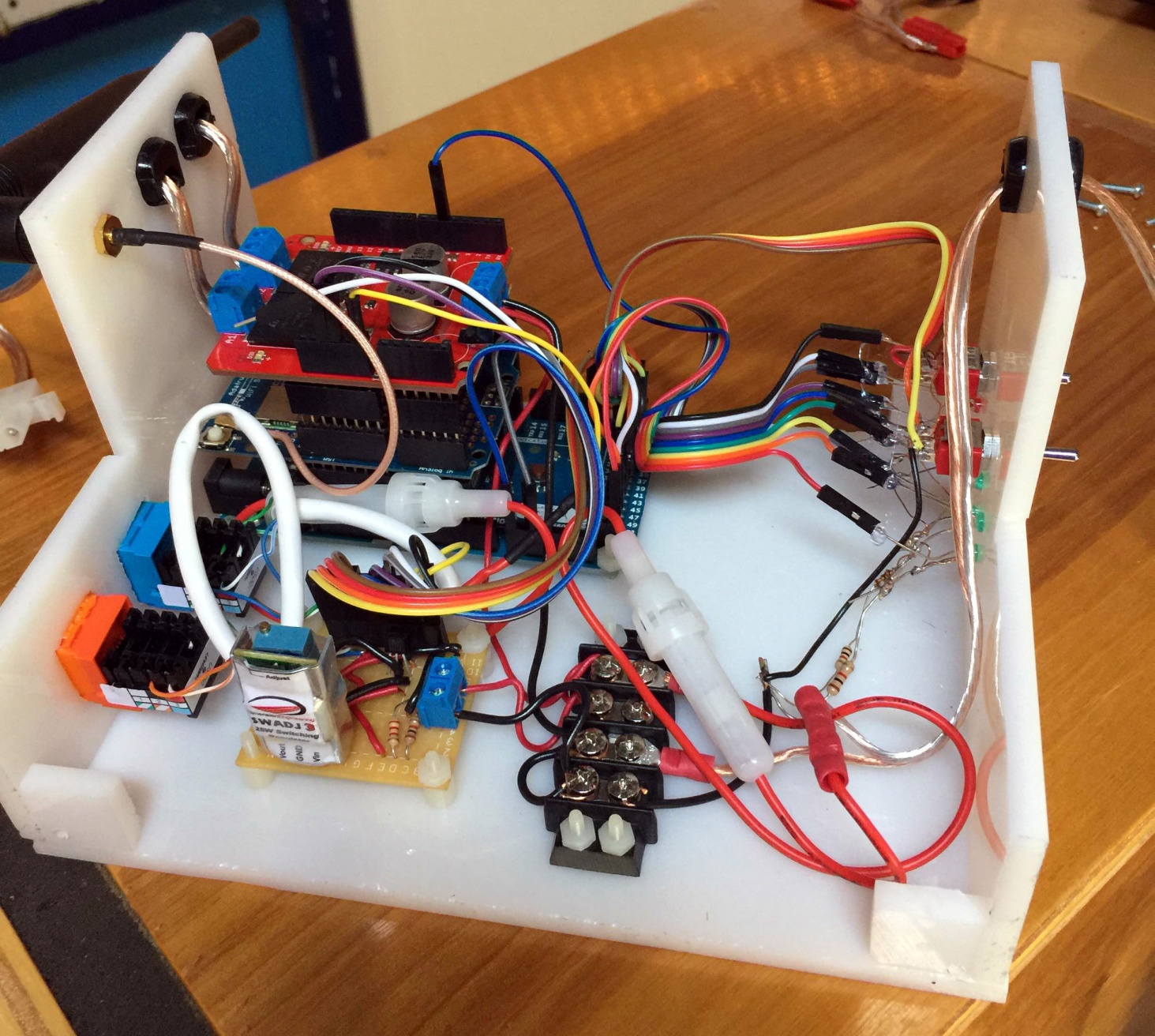


Figure . Interior of controller

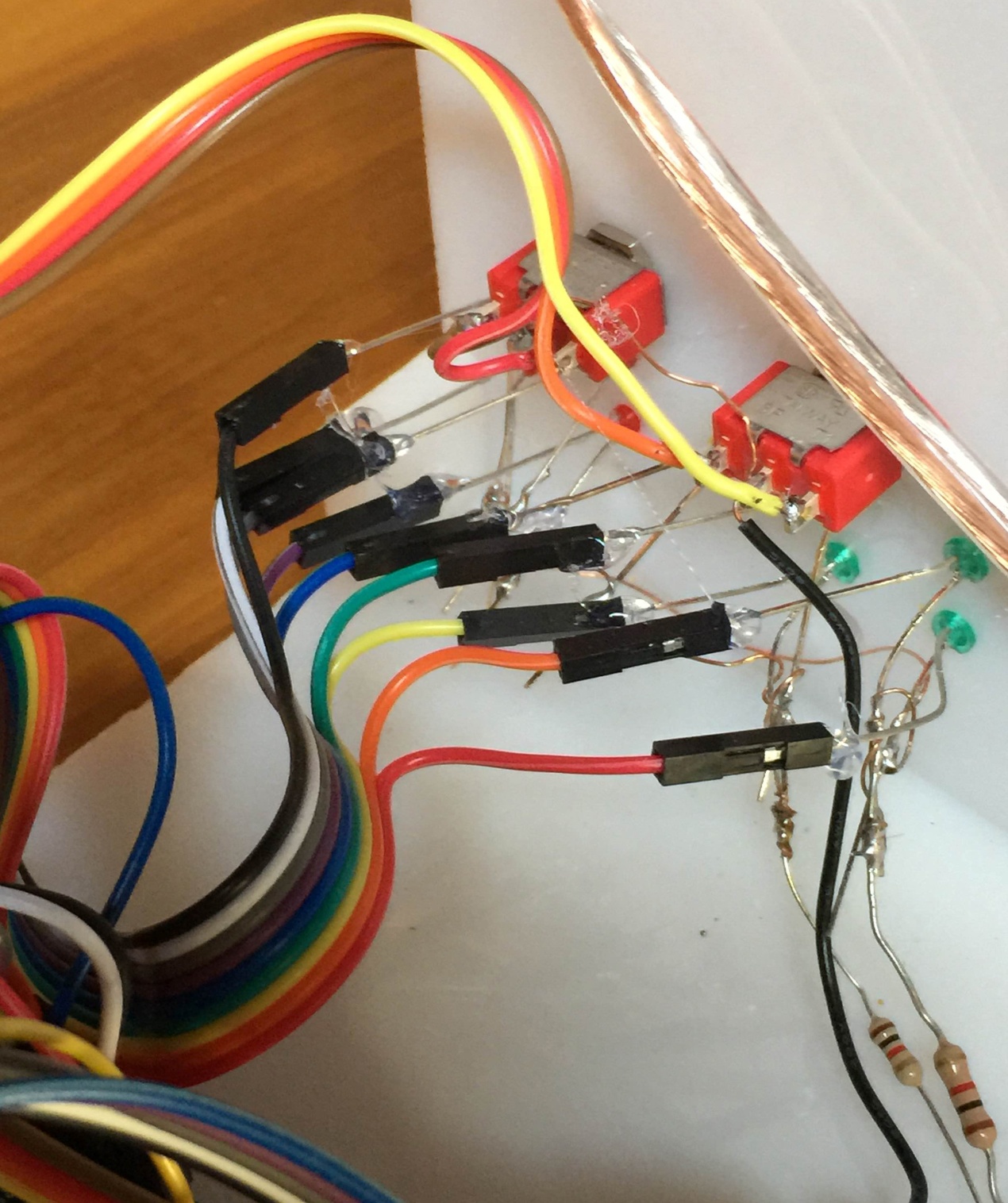


Figure . Extension cables connecting the LED wires to the Arduino. Note the hot glue used to ensure the connectors stay in place.

This image shows the headers between the shields. Note the cut header pin for pin 10.

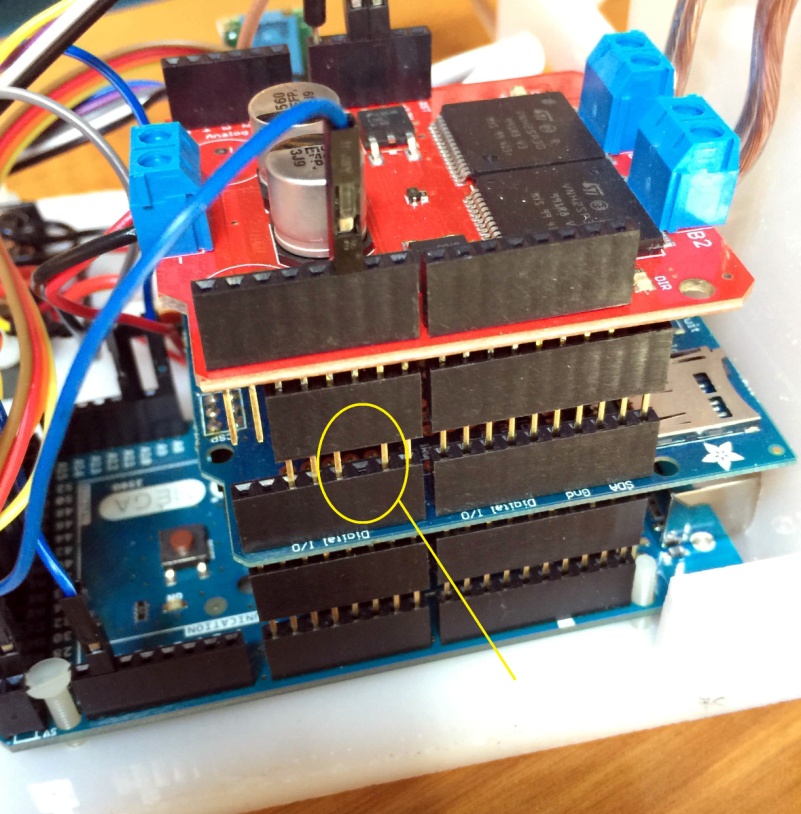


Figure . Double headers used for the shields. Note the clipped pin 10 so the red motor shield does not receive pin 10 from the Arduino. The blue connector routes pin 45 to pin 10 on the motor shield.

This image shows the wiring between the components. The circuit board on the left is how I wired it (my second version). I set the wire colors to match the actual jumper wires used, so I can reconnect the wires if they come apart in the future.

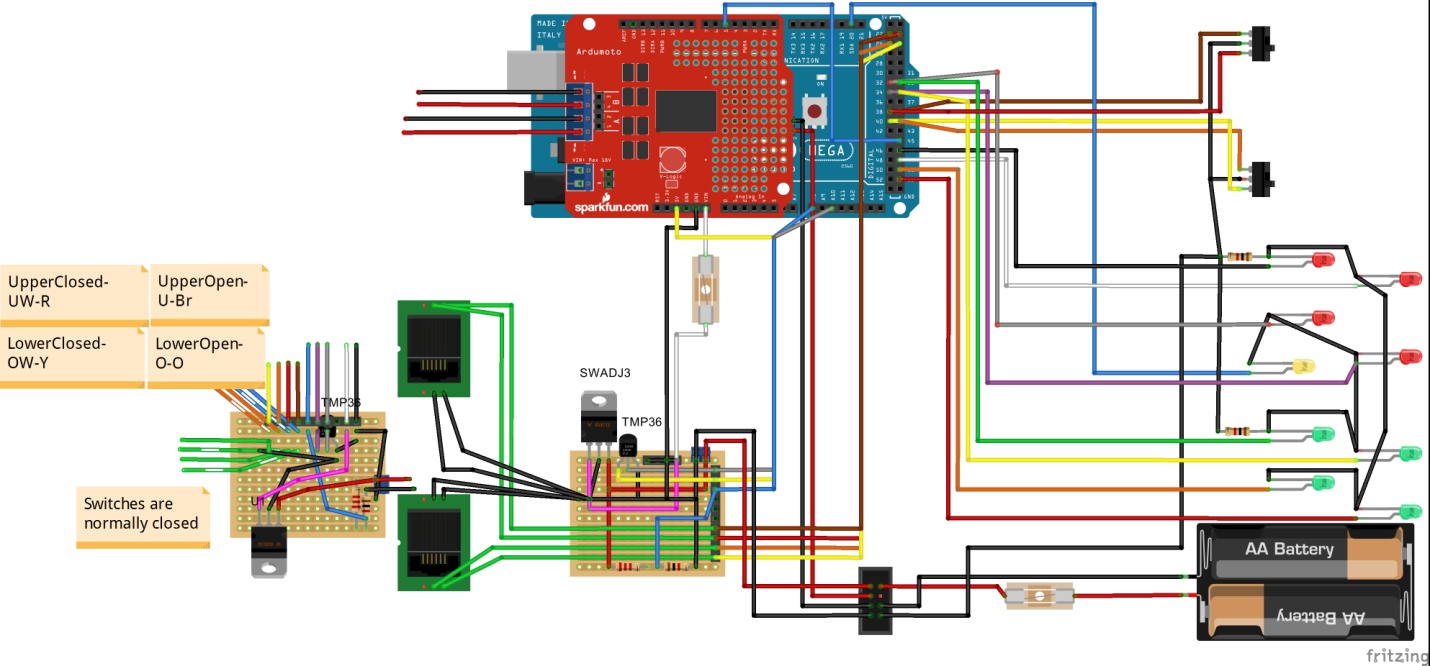


Figure . Wiring of the controller components

Software

**ShutterControl**

The Arduino is programmed with version 1.06 IDE. The main sketch is *ShutterControl.ino*, and has two supporting files:

*ShutterIO.ino* - handles the motor functions

*cc3000.ino* - handles communications with the wireless shield.

**SimpleShutter**

The manual switches are available for debugging purposes, but it is expected that normal operation will be through the program SimpleShutter. This program sends commands and receives status from with the Arduino through WiFi. It performs more checking for inappropriate operations to avoid damaging the shutters.

SimpleShutter can be run interactively to perform a variety of shutter operations. It can also be run from the command line with either "open" or "close" options. In this case, "open" causes both shutters to be opened. "Close" closes the two shutters. The command line option allows the shutter to be controlled from scripts. In particular I have observatory startup and shutdown ACP scripts.

Note that SimpleShutter is NOT an Ascom driver. ACP still uses the Foster Systems AstroMC Ascom driver to control the dome. AstroMC is configured with Automatic Shutter disabled. Note that in the current AstroMC version this generates an error to ACP:(