|  |  |
| --- | --- |
| Robert Brewington  BrewSky Observatory  Casa Grande, AZ | **BrewFocuser** |
|  | BoxFront20.jpg |
| 4/30/2017 | Arduino based Telescope Focuser |
|  | Arduino based focuser implementing various protocols. |

BrewFocuser

Arduino Based Telescope Focuser

Table of Contents

[Introduction 3](#_Toc481316897)

[Features 4](#_Toc481316898)

[Moonlight Protocol 5](#_Toc481316899)

[Optec Protocol 6](#_Toc481316900)

[Robofocus Protocol 6](#_Toc481316901)

[Hardware 8](#_Toc481316902)

[Motor Selection 8](#_Toc481316903)

[Controls 11](#_Toc481316904)

[Push Buttons 11](#_Toc481316905)

[LEDs 11](#_Toc481316906)

[Protocol Selection 11](#_Toc481316907)

[Run/Load Switch 12](#_Toc481316908)

[Extended Temperature Jumper 12](#_Toc481316909)

[Power Connector 12](#_Toc481316910)

[Wiring 13](#_Toc481316911)

[Software 15](#_Toc481316912)

[Installation 16](#_Toc481316913)

[Install Arduino IDE 16](#_Toc481316914)

[Install Arduino Libraries 16](#_Toc481316915)

[Install BrewFocuser Arduino Code 17](#_Toc481316916)

[Select Desired Protocol 18](#_Toc481316917)

[Install ASCOM Driver 18](#_Toc481316918)

[Moonlight Protocol 18](#_Toc481316919)

[Robofocus Protocol 18](#_Toc481316920)

[Optec Protocol 18](#_Toc481316921)

[ASCOM Driver Configuration 19](#_Toc481316922)

[Moonlight Driver 19](#_Toc481316923)

[Robofocus Driver 19](#_Toc481316924)

[Optec Driver 21](#_Toc481316925)

[Temperature Compensation 21](#_Toc481316926)

[Option 1 - Use FocusMax. 21](#_Toc481316927)

[Option 2 - Use the Arduino. 23](#_Toc481316928)

# Introduction

I have been using a James Lacey Focuser controller for years, with its associated EasyFocus ASCOM driver. However, it will eventually fail and need to be replaced. It is obsolete now so I cannot buy another (or obtain replacement parts).

At the same time, a club member has been playing around with building a homemade focuser. I thought it would be interesting to build an Arduino based system.

My first instinct was to create a custom protocol and a custom ASCOM driver. After playing around, it occurred to me (duh!) that it would be more effective to emulate an existing protocol and simply use the manufacturer's ASCOM driver for that protocol. Possible protocols were Moonlight/EasyFocus, Optec, and Robofocus.

Looking around the web, a number of people have done similar things. The two most like mine are Orly Andico and George Carlson (his is based on Orly's work). Both of them use the Moonlight protocol and ASCOM driver.

Here are some links to other projects:

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Protocol | Link | Notes |
| Orly Andico | Moonlight | http://orlygoingthirty.blogspot.com/2014/04/arduino-based-motor-focuser-controller.html | Similar to mine |
| George Carlson | Moonlight | http://www.cloudynights.com/topic/466453-super-compact-electronic-focuser/ | Based on Orly's work |
| frostbyte | Robofocus, maybe Optec? | https://sites.google.com/site/arduinofocus/home | Has not been updated since 2011? Don't know if it ever got finished. Also had Optec issues. Couldn't find software. |
| msfastro | Moonlight | http://msfastro.net/articles/arduinofocuser/ | Based on Orly. Added Temp Sensor. No code posted yet. |
| Astrojolo/ SirJolo | Custom | http://astrojolo.blogspot.com/2013/02/ascom-telescope-focuser-with-arduino.html | Custom ASCOM driver / protocol |
| Robert Brown | Custom | https://sourceforge.net/projects/arduinofocuscontrollerpro/ | Many options, custom driver/protocol |
| ejholmes | Custom | https://github.com/ejholmes/Arduino-Focuser | Custom protocol/ASCOOM driver. Only supports ASCOM 5, has not been updated in awhile. |

# Features

The BrewFocuser controller has the following features:

* Uses manufacturer's controller program and/or ASCOM driver. This eliminates the need for custom drivers and programs. Hopefully the manufacturers provide well tested software.
* Software operation via FocusMax or other ASCOM software using manufacturer provided ASCOM driver.
* Manual operation via pushbuttons.
* Non-ASCOM operation using manufacturer provided software.
* Select either of 3 protocols: Moonlight, Robofocus, or Optec.

Note: Optec and Robofocus are not well tested.

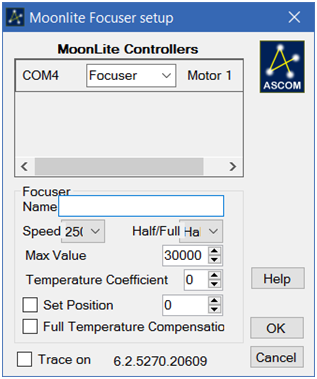
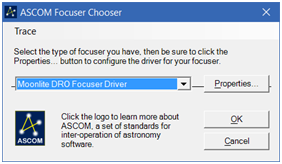
* Precise temperature readings (0.0625 degC).
* By default, temperature is reported in degrees C. However, an Expanded Temperature mode provides increased precision in “Units”. This may allow more precise temperature adjustment. Conversion formulas:
* Moonlight: degC = (Units\*2 /1024 \* 50 + 10)
* Robofocus: degC = Units/1024 \* 70 - 20
* Optec: no expansion available/needed.
* Remote Temperature sensor can be positioned on OTA. Steve Brady has shown that sensors located away from the OTA (typically on the controller box) may not work very well when doing Temperature Compensation.
* Temperature Compensation may be performed in the controller.

Note: this function appears to work, but is not well tested. I think it may be more prudent to perform Temperature Compensation in FocusMax (or other higher level software) instead of directly in the controller.

## Moonlight Protocol

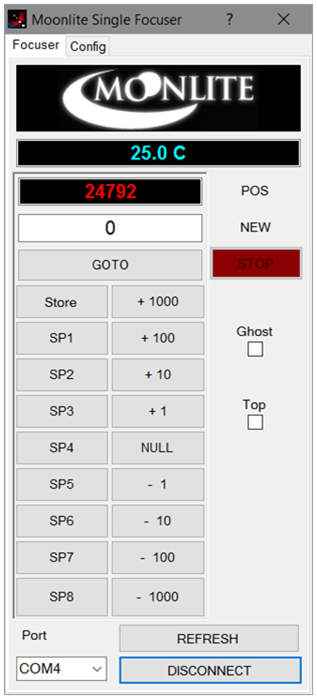
I used this in my initial implementation, and it worked very well using the EasyFocus ASCOM driver.

Of course, the EasyFocus driver is now obsolete. I downloaded the current Moonlight driver from the ASCOM site, and it now works with BrewFocuser.



ASCOM selection of the Moonlight driver, and Properties dialog for the driver.

Moonlight non-ASCOM controller program for manual focuser control.



NOTE: when you select Properties for the Moonlight driver (from the Chooser) it takes several seconds to find Moonlight devices and fill in the Moonlight Focuser setup screen above. At first, this process did not work for me. However, at some point it suddenly started working. I don’t know what changed to make it work.

This is the protocol I have tested most thoroughly, using FocusMax 4.

The non-ASCOM program MoonlightSingleFocuser may also be used to control the focuser manually.

## Optec Protocol

I also implemented the Optec protocol but had some issues with it. You are limited to 7000 steps, and the FHOME command does not match the format of the other commands.

Optec also provides manual control of the focuser with their TCF-S Commander program.

This protocol has not been well tested.

## 

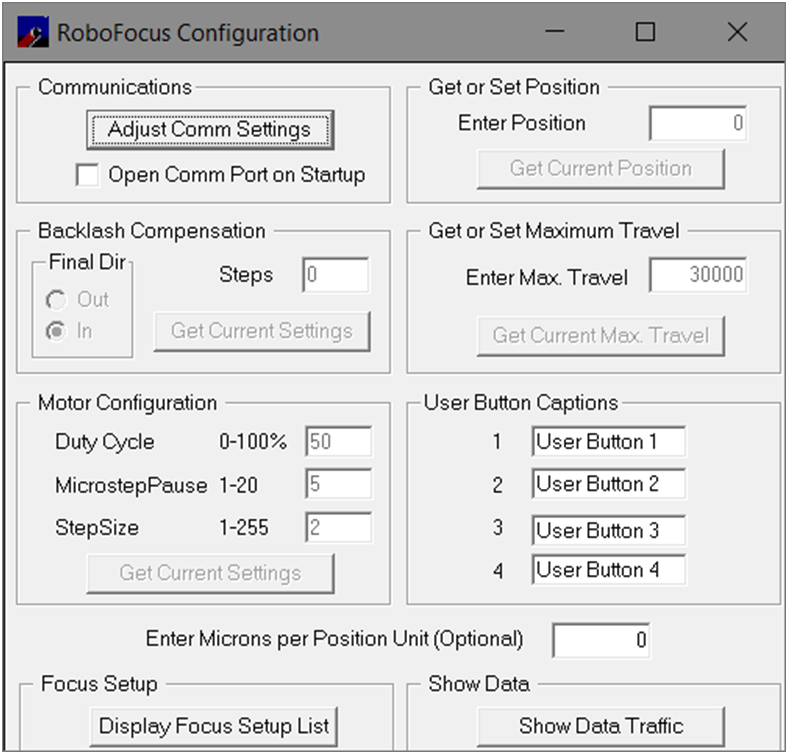
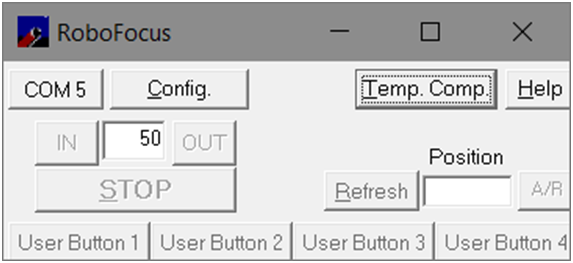
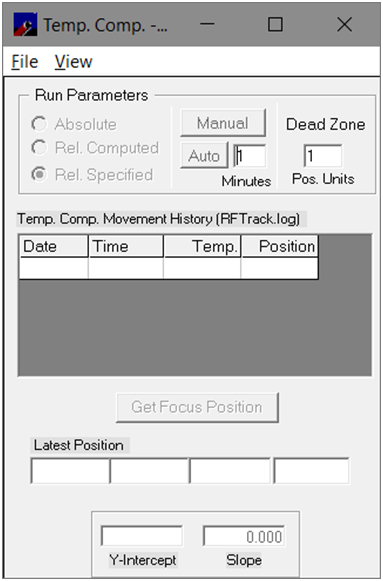
## Robofocus Protocol

Runs using either of two drivers: Basic driver 3.0.9 and Server 5.2.07a.

This has worked pretty well, although it is a bit kludgy having the Gui window come up. Using the interface is also a bit awkward.

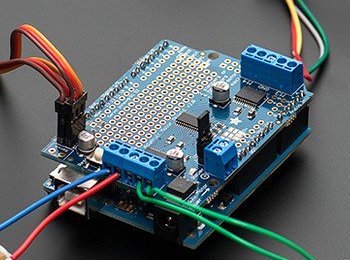
The Robofocus program can also be used as a standalone controller.

This protocol has also not been well tested.



Robofocus Control program. Main screen is at upper left; a couple of configuration screens are shown.

# Hardware

I use an Arduino Uno with the Adafruit Motor/Stepper/Servo shield version 2.3 ($20). This uses the AccelStepper library which works well.

I am using $10 Arduino Uno knock-offs (IEIK?) from Amazon. So far I have not had any problems with them.

Temperature sensing uses the Adafruit MCP9808 I2C breakout board. This is a very small PCB needing 4 wires from the Arduino: Vcc (5V), ground, and the two I2C wires. It has a precision of 0.0625 degrees C, which should make temperature compensation more reliable. Many sensors are only accurate to 0.5 degC.

A cable runs from the controller to the stepper motor. This cable runs 5 wires to the motor. A second cable runs 4 more wires to the temperature sensor. This allows the temperature sensor to be near the OTA.

The enclosure is a $1 plastic box from WalMart's Business supplies area. I also find small plastic boxes at the Dollar Store.

## Motor Selection

My motor is a standard Robofocus motor which I already had: Hurst motor LSG35012F76P (see http://www.beckwithelectronics.com/HURST/lds-lgs.htm). This motor runs on 12V, 3600 steps per rotation, 150 oz/in torque. These cost about $80? $140 from Robofocus, although I think this is with a bracket and coupler. Hurst also makes models with more steps per rotation.

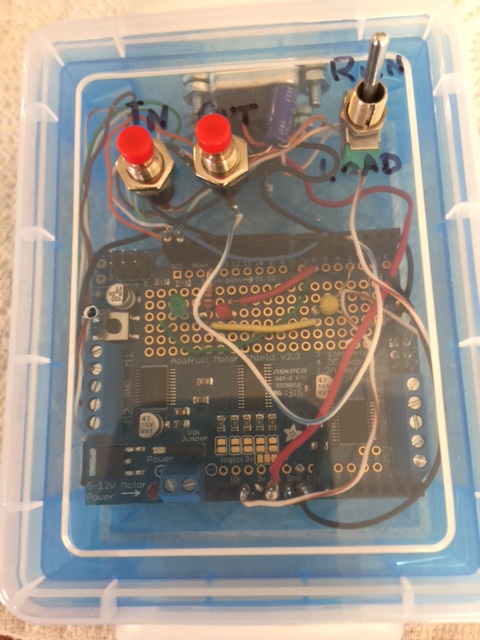
Be careful with selecting the motor. First, you need a lot of steps per rotation; the Robofocus motor provides 3600. Basic motors will not be close to this, they may get 200 steps or so. Hurst also make other models with even more steps per rotation. You need one that is *geared* to produce a large number of steps per rotation. In my case (Takahashi FSQ106) my critical focus zone is about 55 microns, which works out to about 15-20 steps at 3600 steps per rotation. You want a geared motor; some inexpensive motors have planetary gears, typically used in 3D printers. These may draw too much current, and could produce slippage.

The motor shield produces a limited amount of current (about 1.2 Amp). If the motor tries to draw too much current the small power LED on the motor shield will flash and the motor will not run correctly. This means either

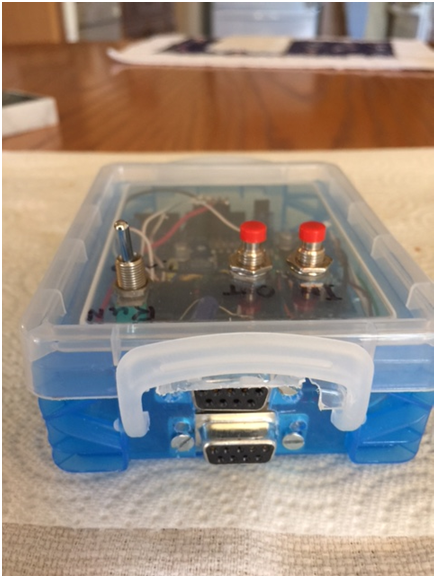
a) you need to provide reduced voltage. For example, many motors are designed to run off of 5V, or expect larger currents. 12V results in too much current for the shield to handle. I do not know if 5V motors will provide enough torque.

b) You need a motor with larger resistance in the coils. The Hurst motor has about 70 Ohms resistance per coil so it works well with the 12V available. One of my 5V test motors only has 10-15 Ohms, which draws too much current.

Top view of the focuser. In and Out buttons allow manual movements of the focuser. Note the 3 small LEDs (green, red, and yellow). See the LED descriptions under *Controls* below.

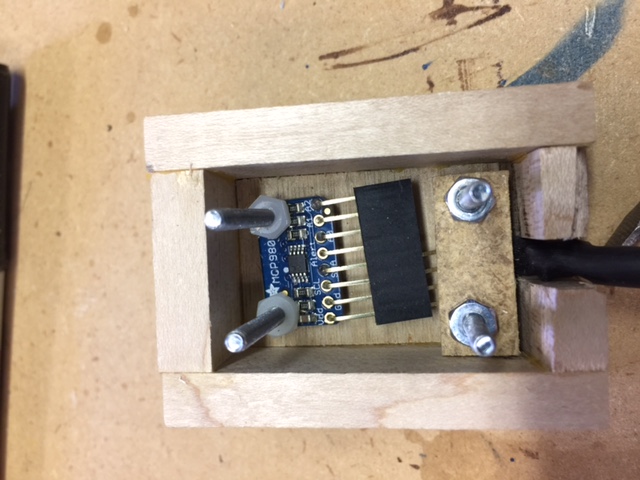
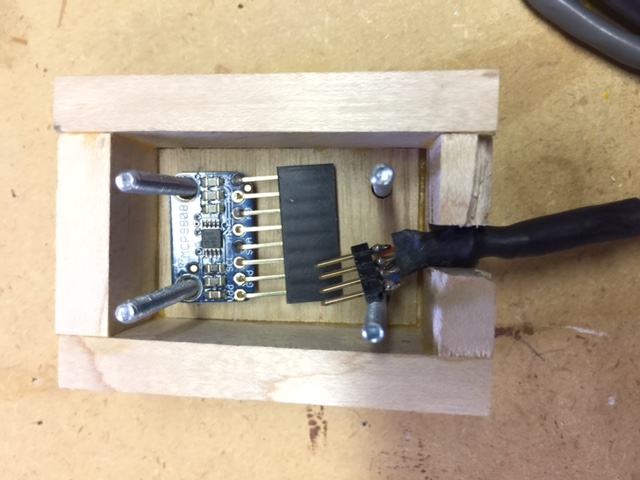


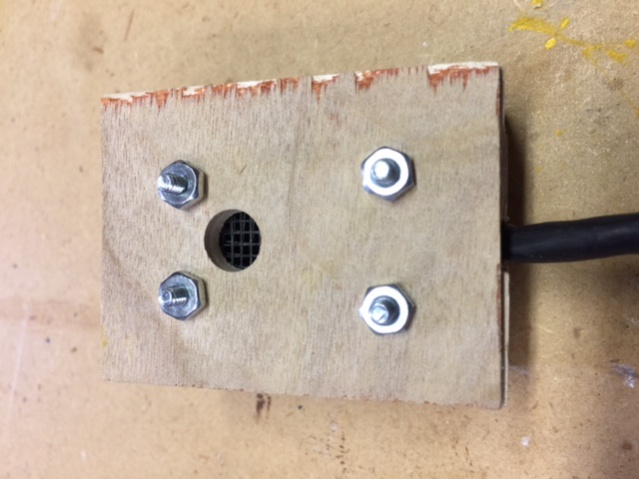
Small hole on the left allows access to the Rest button using a paperclip or small wire.



Front view. The top connector goes to the (optional) temperature sensor. The bottom connector goes to the motor.

Side view. The USB connector and the 12 volt power input.





**Temperature sensor**. Top Left shows the cable terminating in a header, which plugs into the header on the sensor. The small chip between the top screws is the sensor.

Top right shows the strain relief installed. Left image shows the cover in place. The screened hole allows air flow so the sensor accurately reflects the ambient temperature.

The sensor should be mounted near the OTA; Steve Brady indicates the best results are achieved in this configuration.

## Controls

### Push Buttons

Two pushbuttons allow manual movement of the motor inward and outward. These are rarely used; generally the controller is operated through software.

These buttons also allow selection of the desired protocol (Moonlight, Robofocus, Optec) when the controller is powered up. See *Protocol Selection* below.

### LEDs

An LED lights when the motor moves: Green is IN, Red is OUT.

A separate yellow LED lights when temperature compensation is enabled in the Arduino. See the discussion on Temperature Compensation.

When the focuser is powered up, the LEDs indicate two things:

* If the temperature sensor is not found, the yellow LED flashes a few seconds.
* The currently selected protocol is displayed.
  + Green – Moonlight protocol is active
  + Red – Robofocus protocol is active
  + Both Green & Red – Optec protocol is active.

The yellow LED also flashes whenever a command is received from a program. This is useful for debugging.

### Protocol Selection

When the controller is restarted the protocol used by the controller may be changed. Three protocols are available: Moonlight, Robofocus, and Optec.

To Restart the controller, either

1. Use a paperclip to press the Restart Button on the motor shield.
2. unplug both USB and power cable for a couple of seconds, then plug them back in.

When the controller is restarted, hold down the appropriate button(s) to select the desired protocol *before restarting the controller (before it goes through the initialization routine)*. Once the protocol has been registered a movement LED (IN /OUT) will indicate which protocol is active. Release the button at that point to continue controller operation.

* If the In button is pressed, the Moonlight protocol is selected. The green IN LED will light.
* If the Out button is pressed the Robofocus protocol is selected. The red OUT LED will light.
* If both buttons are pressed the Optec protocol is selected. Both LEDs will light.

This process only needs to be performed once. The controller remembers the protocol selected and will use it whenever repowered. In this case (repowering with no buttons pressed) it will flash the LEDs for a second to indicate which protocol is enabled.

### Run/Load Switch

The Arduino has a feature where the DTR signal on the Serial port is used to reset the Arduino. The bootloader is executed, which restarts the controller program. This can be a problem if a program (for example, an ASCOM driver) opens the serial port for communication with the controller with DTR Enabled. Many programs enable DTR; this causes The Arduino to restart. The program cannot then communicate with the Arduino.

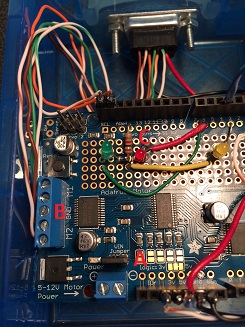
The fix for this is to place a capacitor between the Reset pin and ground. This causes the Arduino to ignore the reset signal. You can still reset the Arduino by a) removing power (both USB and the power cable) or b) pressing the Reset button on the motor shield.

A toggle switch inserts/removes the capacitor, allowing the Arduino to be used in either of two states:

**Load Mode**. No capacitor. In this position code can be downloaded into the Arduino. It also allows the IDE to use its Serial Monitor for debugging. Note that programs and ASCOM drivers will not be able to access the controller in this state.

**Run Mode**. In this position programs can connect through the Com Port. This position is used for normal operation. The Arduino IDE will not be able to communicate to the controller.

### Extended Temperature Jumper

An optional jumper can be installed between pin 12 and ground. This wire is located near label "C" in the image. Run the wire from pin 12 to the nearby Ground connection. See the Temperature Compensation discussion below for use of this jumper.

**Jumper missing (default)**: temperature is reported in degC as normal. The Robofocus protocol will require performing the Temperature Calibration configuration (Temperature Compensation / View) before you can select Centigrade or Fahrenheit display of temperature.

**Jumper installed**: temperature is reported in "Units", for more precise compensation calculation. See *Features* for a discussion of the Extended Temperature Units.

### Power Connector

The power connector expects 12 Volts. This powers both the Arduino and the motor. If the power connector is unplugged the Arduino will continue to run on USB power, although USB does not provide enough current to run typical stepper motors. In this case, the jumper at label "A" should be installed.

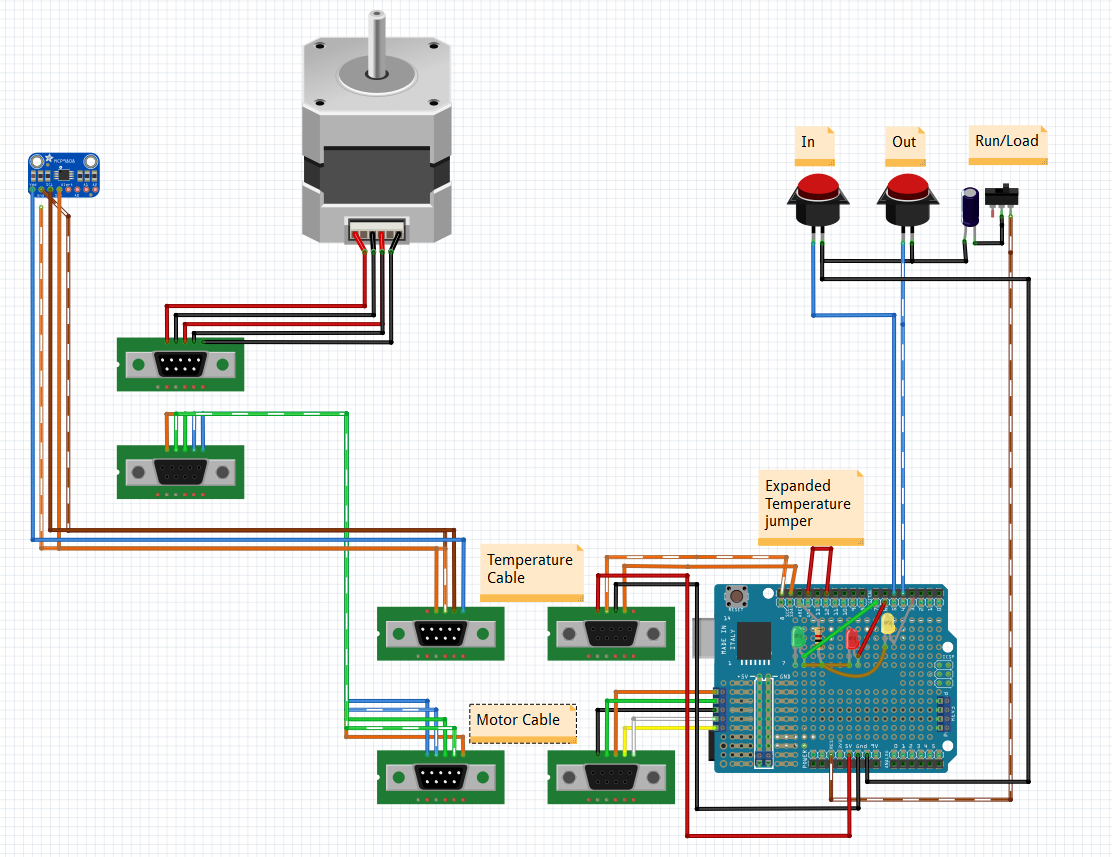
If you have a motor expecting a different voltage (for example, 5 Volts) you should

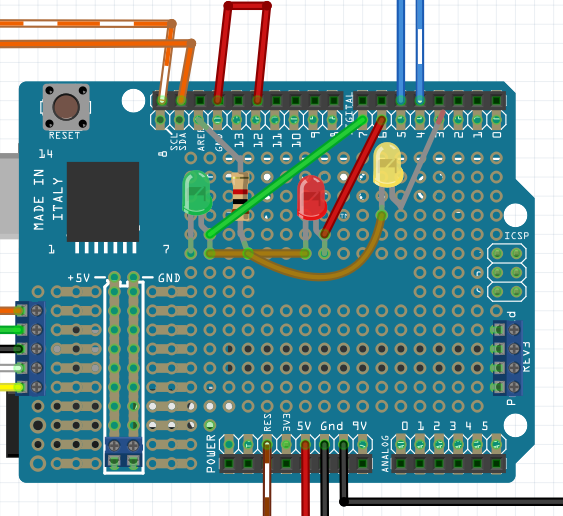
* Connect separate 5V power to the screw terminal on the shield, below the jumper at label "A". Remember the 1.2 amp current limit to the motor! The shield supports 4.7V - 13.5V.
* Remove the jumper on the shield. It is located behind the 2 pin terminal block for the power. The jumper says take power from the 12V power; no jumper means use the separate power.
* Do not plug anything into the power connector.

Now the Arduino is powered by USB while the motor runs on the separate supply.

# Wiring

The diagram shows the wiring of the system.



LED Wiring, pin assignments on the Motor Shield 

# Software

You will need

* I assume you are running ASCOM Platform 6 or better. You will also need the ASCOM driver for whichever protocol you wish to run. See the discussion below on obtaining ASCOM drivers.
* The Arduino IDE. I am using version 1.8.1. This is needed to load the controller code into the Arduino. It is also optionally used if you decide to make changes to the code. Download from https://www.arduino.cc/en/main/software.
* The AdafruitMotorShield library to run the Adafruit Motor Shield v. 2.3. See the page <https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/install-software> to download this library. Be sure to get the library for the version 2.3 Motor Shield; there is an older Version 1 shield which is incompatible.
* AccelStepper library from the same link in the previous library.
* Adafruit MCP9808 library. The page https://learn.adafruit.com/adafruit-mcp9808-precision-i2c-temperature-sensor-guide?view=all describes the use of the MCP9808 sensor, and has a link to the github source at https://github.com/adafruit/Adafruit\_MCP9808\_Library/archive/master.zip.
* My BrewFocuser sketch, composed of several files.
* BrewFocuser.ino (main file)
* EepromUtil.ino, EepromUtil.h Saves values in EEPRom
* MotorIO.ino, MotorIO.h handles functions for running the motor
* TempComp.ino, TempComp.h handles temperature readings, Compensation.
* Globals.h
* BaseProtocol.h / cpp Master class for protocol handling
* EasyFocus.h/cpp (implements the Moonlight protocol)
* Optec.h/cpp (implements the Optec protocol)
* Robofocus.h/cpp (implements the Robofocus protocol)

# Installation

I am currently running Windows 10, ASCOM 6.2. I use FocusMax 4 to perform all focusing, although any program using ASCOM (MaximDL, for example) should work.

## Install Arduino IDE

Download the Arduino IDE (https://www.arduino.cc/en/main/software). You really only need this to download the Arduino code (one time only). You use it in the future if you decide to modify the code and need to reload it.

Once the code is loaded the Arduino remembers it even if it is reset or repowered.

Follow their instructions to install the IDE; it is a pretty standard process.

## Install Arduino Libraries

Download the three libraries: AccelStepper, AdafruitMotorShield version 2, and Adafruit MCP9808.

Open the IDE. Go to the Sketch/ Include Library/ Manage Libraries dialog to add each library to the IDE. By default, these will be added in your Documents folder.

Assuming the additions have worked, clicking Sketch/ Include Library should show the libraries at the bottom of the menu.

## Install BrewFocuser Arduino Code

Copy the BrewFocuser files into a folder named BrewFocuser. In the Arduino IDE, open the file BrewFocuser.ino; the associated files will automatically be loaded into separate tabs.

Use the Tools menu selections to select the Arduino board (mine is Arduino Uno) and the COM port for the board. The current settings are shown at the bottom of the screen.

Make sure the Run/Load switch on the controller is set to the Load position.

Click the download button () to compile and download the code to the Arduino.

Flip the Run/Load switch to Run for access by programs.

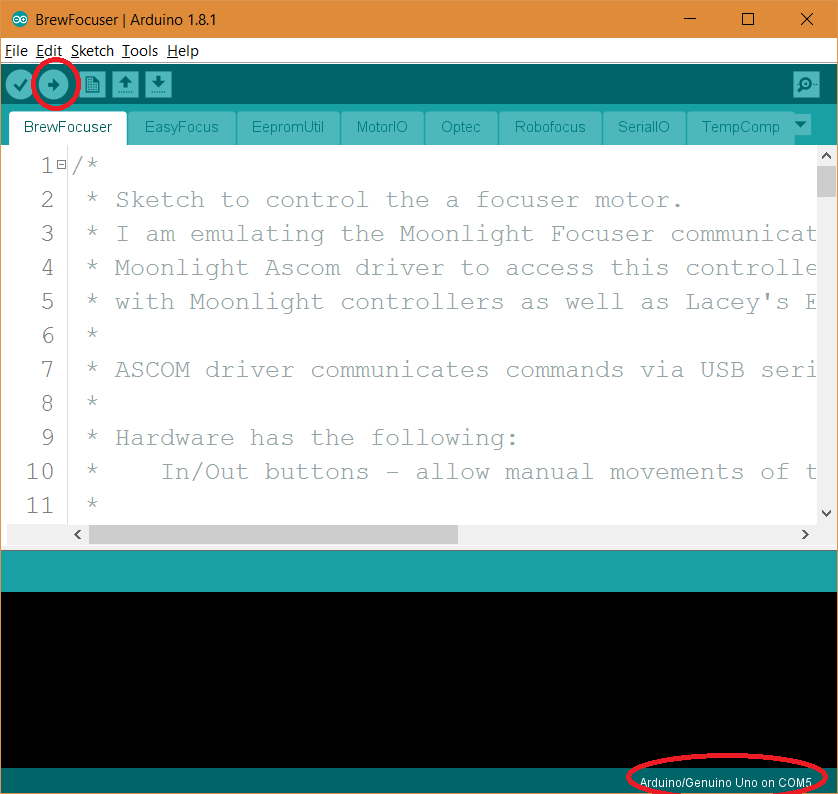


Figure . Arduino IDE to download code to the Arduino. The red circle at the top downloads to the Arduino. The current board and COM Port are shown at the bottom right of the screen.

## Select Desired Protocol

Unplug the USB Connector and remove the power connector. See the discussion above (Controls/Protocol Selection) for instructions on selecting the protocol desired.

## Install ASCOM Driver

Install the appropriate ASCOM driver and optional controller program for the protocol you plan to use. You can install multiple protocols, although there would seem to be no reason for this.

### Moonlight Protocol

Download from <http://focuser.com/downloads.php>. You will want The ASCOM driver, found in the link [MoonLite DRO Setup.zip](http://focuser.com/media/Downloads/MoonLite_Software/Ascom/MoonLite%20DRO%20Setup.zip).

Optionally, you may want the standalone program to run the focuser. Find the link for the NonAscom MoonliteSingleFocuser\_v1.1.zip at http://focuser.com/downloads.php. This program talks directly to the controller instead of using the ASCOM driver. This is useful for manually focusing; it can also be useful for testing the system.

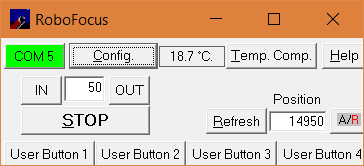
There are also a couple of documents describing the focusers and the protocol being used.

### Robofocus Protocol

There are several versions of Robofocus drivers available.

1. <https://www.homedome.com/downloads.htm>. From the manufacturers website, this provides a non-ASCOM program. Do not use this one; it is very old (marked as version 3.1) and has problems running in current Windows environments.
2. On the ASCOM site http://ascom-standards.org/Downloads/FocuserDrivers.htm, there is a “simple” ASCOM driver. This a typical ASCOM driver with no Gui other than the Properties screen. A feature of this driver is that it allows COM ports greater than 8.
3. Also on the ASCOM site is another program called the Robofocus Control Program (version 5.2.0a). This looks visually like the program on the manufacturer’s site. However, it seems to work. It also can serve as an ASCOM driver. However, it only allows COM ports up to 8.

After downloading from the ASCOM site run the installer.

Run the controller program standalone to verify operation of the controller. See the comments below for suggestions on configuring the driver.

Test that the ASCOM driver is available and works by running your focusing program (I use FocusMax). The Chooser should find the Robofocus driver.

### Optec Protocol

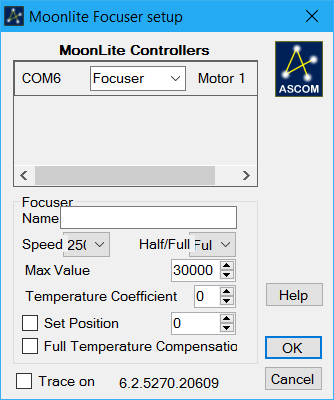
The Optec site <http://www.optecinc.com/astronomy/downloads/default.htm> has the download of the ASCOM driver. You want the TCF-S Focuser download page.

The install package OptecTCF-S\_Installer(6.0.4).zip (current version) can install both a controller program and the ASCOM driver.

## ASCOM Driver Configuration

When running a program such as FocusMax or a manufacturer control program, make sure the Run/Load switch is in the Run position. These programs generally use the DTR signal to start Serial communications. The Arduino has a feature where the DTR line causes the Arduino to reset, preventing Serial communications. The Run position bypasses this feature.

### Moonlight Driver

When the Moonlight DRO Focuser Driver is selected from the ASCOM Chooser, the Properties window should look something like this. You can change the default parameters. Speed is not used.

Make sure the Max Value is something reasonable (not zero). This is the maximum position allowed for the motor.

You can set the current position to a value. Remember to enable the Set position check box. This is a one-time operation; it does not reset every time the focuser is connected.

### Robofocus Driver

The Robofocus Control Program has several configuration screens as shown below. You may need to play with them to get things working as desired.

#### COM Port

Set the Com port to the same port used by the Arduino IDE to download the code. You may need to adjust the port settings appropriately. The Arduino uses 9600 baud; you can change this in the code.

#### Maximum Travel

It is useful to set the maximum position allowed. Beware of starting out at 0.

#### Current Position

Enter a number here and hit Enter to set the value of the current position of the motor. This is buggy - I often have trouble getting the screen into a state where I can enter the number.

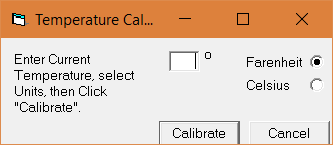
#### Backlash

Not implemented in the Arduino. These settings have no effect. I configure backlash handling in FocusMax.

#### Motor Configuration

These settings are not implemented. The Arduino Motor Shield handles these functions. You may wish to change the motor speed and acceleration in the Arduino code.

#### Temperature Readings



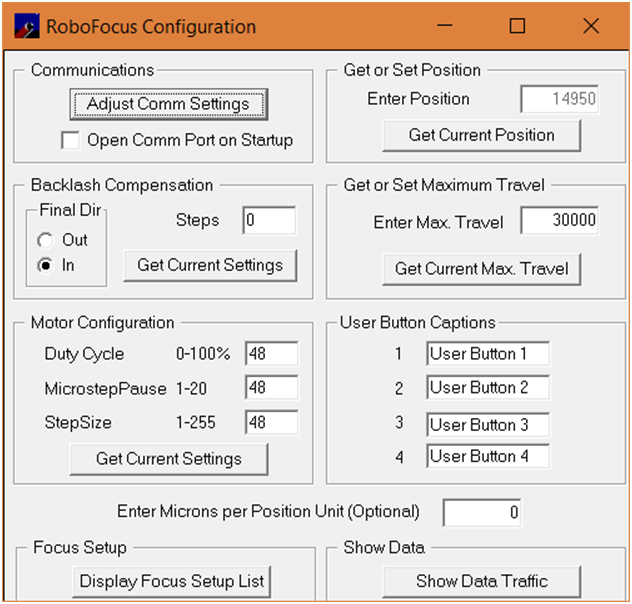


Figure . Main Configuration Screen

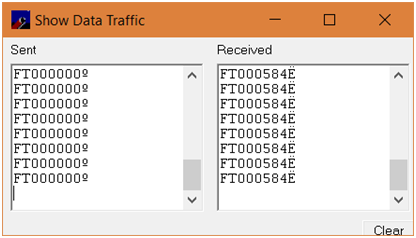
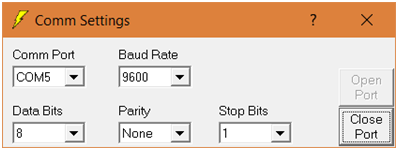
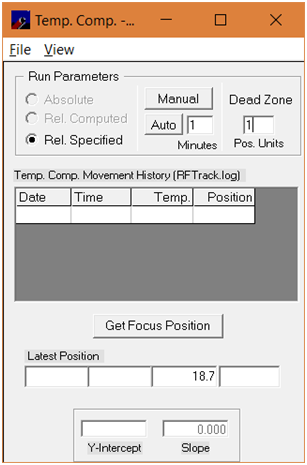


Figure . Data Traffic screen. Useful for debugging protocol communications

Figure . Com Port Settings



You may need to perform the Temperature Calibration before seeing temperature in degrees C.

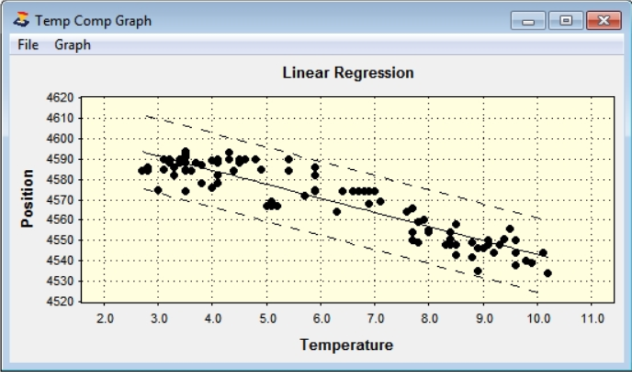
### Optec Driver

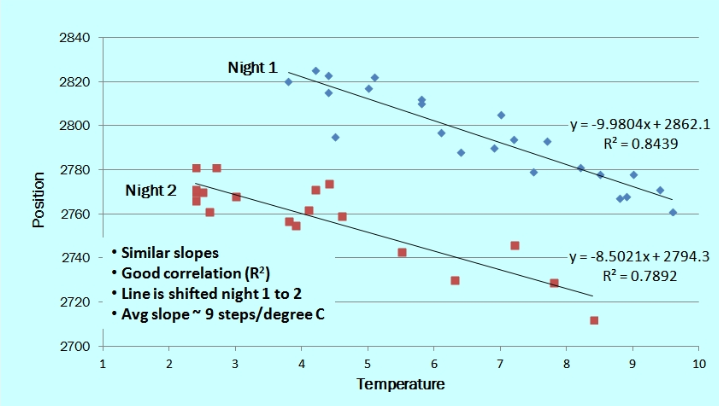
## Temperature Compensation

### Option 1 - Use FocusMax.

I expect to use FocusMax to do the temperature compensation for my unit. The Arduino simply returns temperatures when requested, and FM adjusts the focuser position as needed.

#### Determine Temperature Coefficient

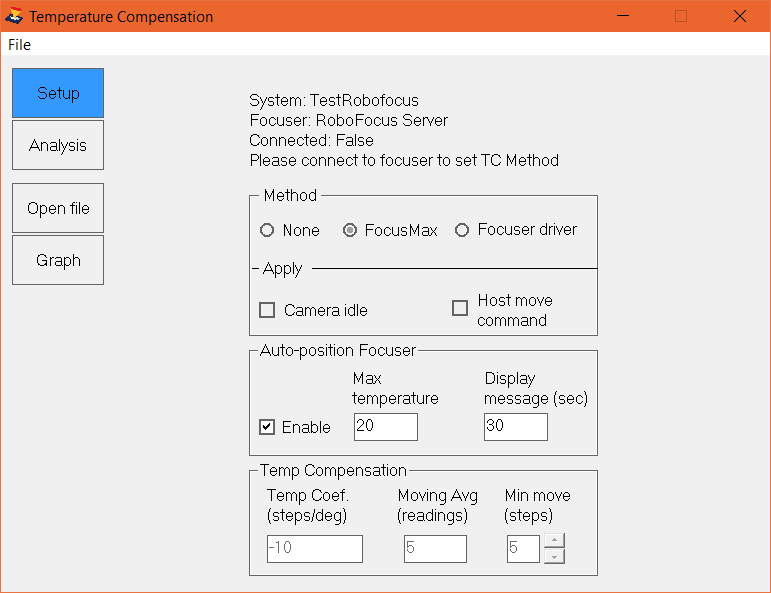
You need to have the Temperature Coefficient for your unique system. If you graph Focus Position as a function of Temperature, a) the result should be fairly linear, and b) the coefficient is the slope of the line.

Some systems also want the Offset (Y intercept) of the line. However, Steve Brady (author of FocusMax) has found that from night to night the offset can be different (See plot below by Steve Brady).

You can manually focus the system throughout the night, collecting the position and temperature data for each run. Alternatively, the Temperature Compensation Wizard in FocusMax will collect the data for you. See the FocusMax 4 Help document on using the Wizard.

NOTE – I use ACP to position the scope at a reasonable starting position of the sky, then use the FindBrightStar script to center a star of the appropriate magnitude. After this script finishes the green Tracking indicator will be flashing. *This means that in 10 minutes tracking will be turned off by ACP*. Click on the flashing indicator to make it steady green, so tracking will not be turned off.

#### Configure Temperature Compensation

Once you have your Temperature Coefficient, use Settings/Temperature Compensation to configure FocusMax to perform the compensation.

Check Camera Idle if compensation is to be performed even while no image is being collected.

The Temp Coefficient is the slope of the line from your measurements, in steps/deg.

The Moving Average smoothes out the temperature readings; the minimum is 5 readings.

Minimum move indicates that the position is only changed if it needs to change by more than the minimum. In the example screen shot, the position is not adjusted until it is off by 5 or more steps.

#### Enable Temperature Compensation

Go to the System Tab in FocusMax. Enable the Temp Comp checkbox to start Temperature Compensation running.

### Option 2 - Use the Arduino.

BrewFocuser has code to handle Temperature Compensation inside the controller.

This code has not been tested thoroughly☺, since it seems easier to compensate in FocusMax.

Some of the drivers have the capability to load the temperature coefficient into the controller and enable direct compensation within the controller.