

# ArduinoAstroControl

24/05/2010

Copyright Guy Webb 2010

ArduinoAstroControl is the second incarnation of my home-brew auto guider interface, following on from the [SimpleGpUsb](#) project. The new design is based on the very popular Arduino hardware, which gives me potential to add additional features in the future. The driver design has also been changed so that it now uses an ASCOM Local Server Executable, thus allowing multiple drivers to access the hardware at once.

This document is going to summarize my project, and hopefully provide you with all the information you need to make your own ArduinoAstroControl device should you want to. Rather than reproduce all of the reference material I used during this project I shall [hyperlink](#) in any relevant information to their original sources.

I place all of my ideas and source code into the public domain under the GPL v3 license, so feel free to tweak, modify and enhance to suit your own personal needs. Just make sure that you share your new creations with the world and give credit where credit is due. A copy of the GPL license is included with the source code.

## Hand Controller Modification



Figure 1: HEQ5 Hand Controller Modification

First things first... To be able to connect the ArduinoAstroControl device to my HEQ5 mount I needed to perform the same [Hand Controller Modification](#) as detailed for the Shoestring products. Their instructions are excellent, and although it was a bit scary taking my precious controller apart, it was all fairly straight forward.

Rather than buy the ShoeString kit I made up my own connector so that I had a proper RJ-12 socket coming out of the controller instead of a plug lead that required a coupler. I just got 5 lengths of suitably coloured wire, an RJ-12 socket, some heat shrink and a little plastic box to house it in. It's almost exactly the same as the ShoeString approach, I just preferred doing it this way.

The socket is wired as shown in Figure 2. It's obviously very important that you get your wiring correct or the whole thing will fail to work, and potentially damage your hardware!

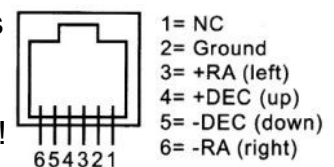


Figure 2: RJ-12 Socket Pins

I have done my best to follow the ST-4

“standard” but it seems that there is a fair amount of variance in the way it is actually implemented. Ultimately you don't need to adhere to the standard if you don't want to, so long as all the connections all tie up i.e. the ground wire on the hand controller goes to the ground wire on the ArduinoAstroControl device etc.

*Note: I actually left this step till last when I did it, as I wanted to make sure that my device actually worked first! You might want to do the same, I wanted to point out this necessary step up front in case it puts the fear of God into you!*

## PC Hardware Interface

This device makes use of an [Arduino](#) Duemilnove board. This is a widely available, open source electronics prototyping device, and it's provides plenty of scope for computer controlled systems such as this one. You can program the device to make it do what you want, in the case of ArduinoAstroControl it monitors the serial data sent to it and turns output pins on and off as requested, thus simulating pressing the hand controller buttons.

You will need to install the Arduino development environment so that you can compile and upload the device software to the board. The source file ArduinoAstroControl.pde has all of the Arduino code required, so simply load this into the development environment and upload it to your device.

## The Circuit

This circuit used is very simple, and is built around a quad Opto-Isolator. This is a convenient packaging of 4 phototransistors into one chip. I used these phototransistors as switches to make and break the connections that simulate button presses on the hand controller.

Conceptually it's important to note that the ArduinoAstroControl device does not output any current, it's just a switching system to make the "loop" within the hand controllers own circuit, which was tapped into when making the hand controller mod.

Figure 3 shows the circuit design, with the Opto-Isolator in the middle of the action. The output pins from the Arduino board act as the inputs, on the left side of the chip. Be sure to connect the cathodes to the ground pin of the Arduino. On the opposite side of the chip the emitter pins are connected to the appropriate pins of the RJ-12 socket as detailed in Figure 2 above. The ground wire from the hand controller circuit should be connected to the cathodes. With these connections in place, any Arduino output pins toggled to a "high state" will trigger the transistor switch and thus allow the current to flow through hand controller circuit, just as if a button had been physically pressed.

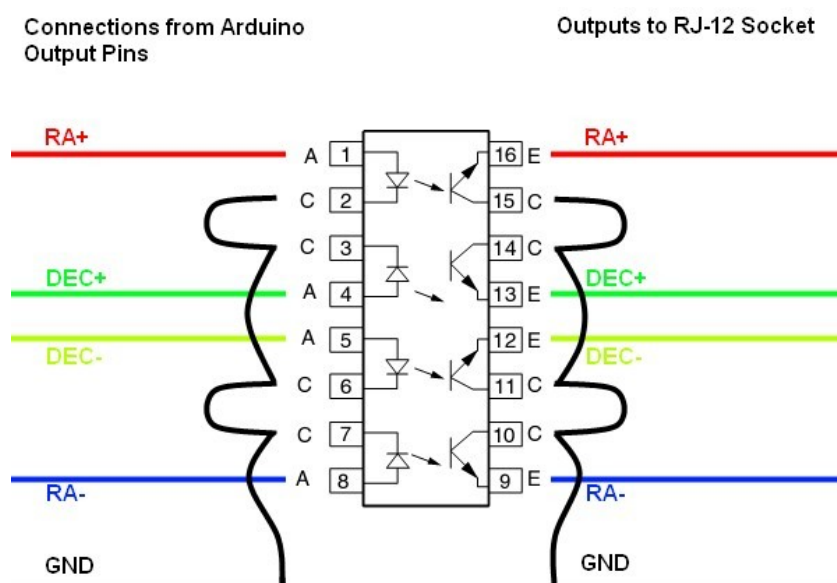


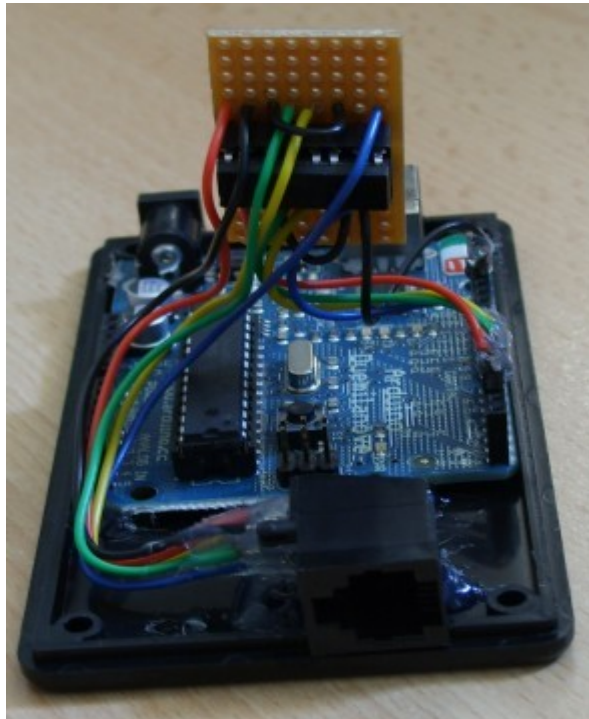
Figure 3: The Basic Circuit

The Opto-Isolator I used was a **ILQ74** that I picked up from my local Maplin.

## ***Putting It Together***

To make the final product I used a plastic enclosure from Maplin. It measures approximately 110mm x 65mm x 30mm, and is just the right size for the Arduino board to fit in to. I had to do a bit of cutting to make holes for the USB input socket, and the RJ-12 socket, but nothing too drastic. I affixed the Arduino board in place with a few splodges of hot glue around the edges.

As you can see it really is quite simple in the end. The 4 outputs of the Arduino board go to the 4 inputs on the Opto-Isolator (don't forget the ground connection too!), and then the wires from the RJ-12 socket go to the corresponding output pins, and of course to ground. And that's pretty much it!



*Figure 4: Internal Layout*

## ***Cables***

In trying to stick with ST-4 de facto standard for guiding products, I needed to make sure that I had the right sort of RJ-12 cable to connect the modified hand controller to the ArduinoAstroControl device.

Another quick Google brought up some nice documents, such as this one from ShoeString: [Guide Port Cabling](#). The cable and sockets used are of the RJ-12 variety, which can also be referred to as **6P6C** (6 positions, 6 connectors).

With the sockets wired as shown in Figure 2, I needed a cable wired like the one shown in Figure 5.

The crucially important thing to note here is the fact that the wires are in the same order from left to right on both plugs. This means that the ground pin on one socket will connect to the ground pin on the other socket.



Figure 5: Guiding Cable

## Software Support

For the ArduinoAstroControl device to be of any use I had to make sure that the freely available guiding software such as [PHD](#), and [GuideDog](#) could “talk” to it. Fortunately the [ASCOM Initiative](#) came to my rescue here, with their industry standard for all things astronomical.

By implementing the “Telescope” interface defined by them, software such as PHD and GuideDog can easily communicate with the device, without having to know anything about it. This should hold true for any other ASCOM compliant guiding software that you might use.

Before you can use my driver you must ensure that the <http://ascom-standards.org/> is installed on your machine. The current version is 5.5 at the time of writing. Then you can install the driver I developed for the ArduinoAstroControl device, which is included in the files download that goes with this document. The installer should configure everything you need to get your computer to communicate with your hardware once it's all built and ready to go.

The driver is very simple in nature, and only supports a small subset of the available features, just what is needed for guiding. There are a few configuration options that you can adjust via the properties dialog:

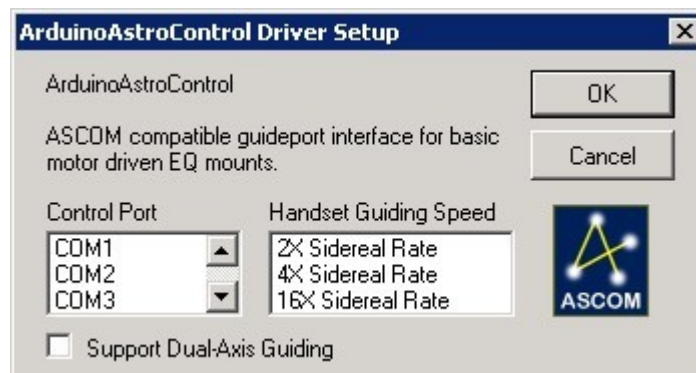


Figure 6: ASCOM Driver Properties

First and most important of all is the COM port that the ArduinoAstroControl device declares itself as being. You might need to go into the windows device manager and look under the ports item to find this out.

Next you can specify the guiding speed. On the hand controller for my HEQ5 there is a switch that offers 2x, 4x and 16x sidereal rate, and this option reflects those values. It is very important that the switch position on the hand controller and this software option be the same, otherwise all of the guiding corrections will be wrong and you'll find the scope

chasing around in circles! Generally you will want to use the 2x rate to make smaller and more refined adjustments. See the Appendices for more information about the guiding rates.

Finally you can specify if the declination axis is supported for guiding with the “Support Dual-Axis Guiding” option. If your mount only has a Right Ascension motor you should not tick this.

### ***The Finished Product***

Here's a rather unexciting picture of the finished device! You can see the RJ-12 socket on this end, the other has the USB socket.



*Figure 7: The Finished ArduinoAstroControl*

And of course the proof is in the pudding, so here's the first image I capture using the ArduinoAstroControl device to autoguide. It's about an hours worth of data, captured in 5 minute subs.



*Figure 8: First image captured*

## ***Disclaimer***

I hope that my instructions in this document have been clear, and I have done my best to not make any mistakes. However I am only human, so this cannot be ruled out. If you choose to copy what I have done I accept no responsibility for any damages or loss you may suffer. I offer this information freely to anyone who might like to use it, but I offer no warranty or promise of its fitness.

It all worked like a dream for me, and I hope that it does for you too, but I've got to cover my back! ; )



## **Appendix A**

Sidereal rate is  $\frac{1}{4}$  of a degree per minute, or 15 arcseconds. With this knowledge we can see that the guiding rates of the HEQ5 hand controller of 2x, 4x and 16x sidereal rate equate to the following:

- 2x =  $\frac{1}{2}$  a degree per minute
- 4x = 1 degree per minute
- 16x = 4 degrees per minute

I have discovered one slight issue with these rates with regard to my HEQ5 motor drives and controller. When moving the RA motor in *reverse* at “2x speed” the motor simply stops, meaning that the Earth’s rotation is responsible for any movement. This also means that the reverse speed at “2x” is in fact 1x sidereal rate. I have no way of knowing whether this deviation affects the 4x and 16x speeds as well, or whether it affects the DEC axis at all.

Unfortunately this slight discrepancy cannot be accounted for in the ASCOM driver as it is assumed that the forward and reverse speeds will be identical for a given rate.

In practice I have discerned no ill effects of this small variance, but it is there and should be considered.

## **Appendix B**

To support different types of mount the ArduinoAstroControl ASCOM driver makes use of a configuration file: *AxisRates.xml*. This file allows you to define the axis rates used by the driver, so that you can tailor it to suit your hardware. This file can be found in the *ArduinoAstroControlServedClasses* folder of the install directory (which defaults to `\Program Files\ArduinoAstroControl`).

The default file supplied is for my HEQ5 mount, which offers 2x, 4x and 16x speeds.

If you need to modify the rates it's a pretty straightforward matter. Each telescope axis has it's own set of rate definitions. Each definition has a name and two rate values, which are defined in degrees per second. In most cases you should be able to use one of the existing definitions, and multiply / divide rate values to get the value in degrees per second that you require. Note that the min and max rate values should be set to the same value, the driver does not support variable rates, and it's behaviour is undefined in this case.