

EM27Cover Documentation  
Bill Simpson ([wrsimpson@alaska.edu](mailto:wrsimpson@alaska.edu))  
20 July 2017

This document describes the operation of the EM27/Sun cover and software on the hosting computer (the Windows machine that runs the EM27/Sun). The photo below shows the cover in open state.



### Hardware:

The EM27/Sun cover is operated by an Arduino Uno (R1), which runs a stepper motor that opens and closes the cover by a screw jack mechanism. The stepper motor is driven with 12V DC and a current-controlled (chopping) driver chip. A USB connection to the arduino provides logic power (+5V) and communications via a “virtual com port”. The code that is running on the arduino is available from:

<https://github.com/BillSimpson/EM27cover>

The hardware uses the Pololu DRV8825 stepper motor driver and the arduino “AccelStepper” library. There are two physical limit switches that indicate the open and closed states of the hatch. The driver takes single character ASCII text commands over the virtual com port, with the following command set:

- o = Open cover once; will not re-open after rain
- f = Open cover and try to keep it open as much as possible
- c = Close cover
- ? = Report cover state (open/closed & rain)

Generally, the arduino checks for text input every second, but if a move is requested, the driver tries to make that move (which takes ~23 seconds) and will not respond until that move is done. If the driver does not get to the requested limit in a timeout (about 28 seconds), it will report that the motor didn’t achieve the final state and there is an opportunity to enter a different command. If no other commands are entered, the driver will continue to seek the requested limit. When requested to be in a state (e.g. closed), the

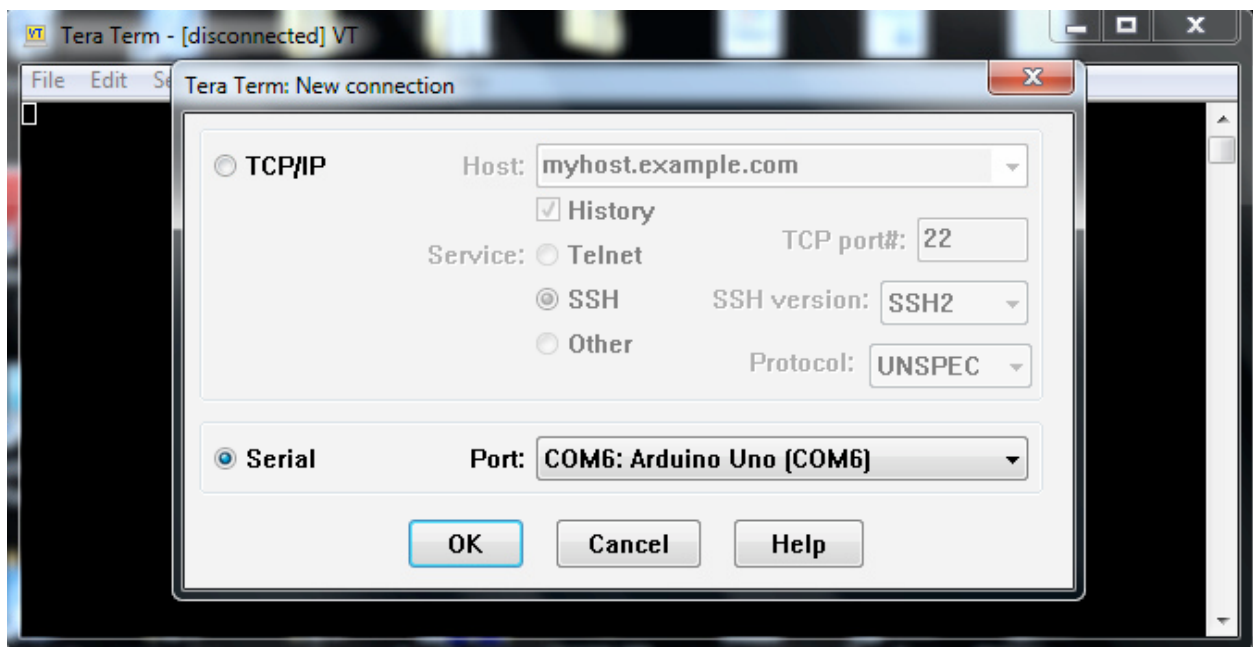
driver continues to be active, so if the wind blows the cover open a bit, the driver will force it closed again, reporting that it moved and how long that move took.

The purpose of the automation is really to protect the instrument from rain, so the rain sensor takes precedence, and if it is triggered, the cover will close. Typically, we will use the “o” command, which opens the cover but doesn’t re-open after the rain event ends. If one wishes to keep the cover open as much as possible, the “f” command can be used to force the cover open as long as the rain sensor is dry. Although the arduino reports rain closing events to the host computer over the virtual com port, the EM27/Sun software (e.g. OPUS and CamTracker) keep taking data even if the cover is closed.

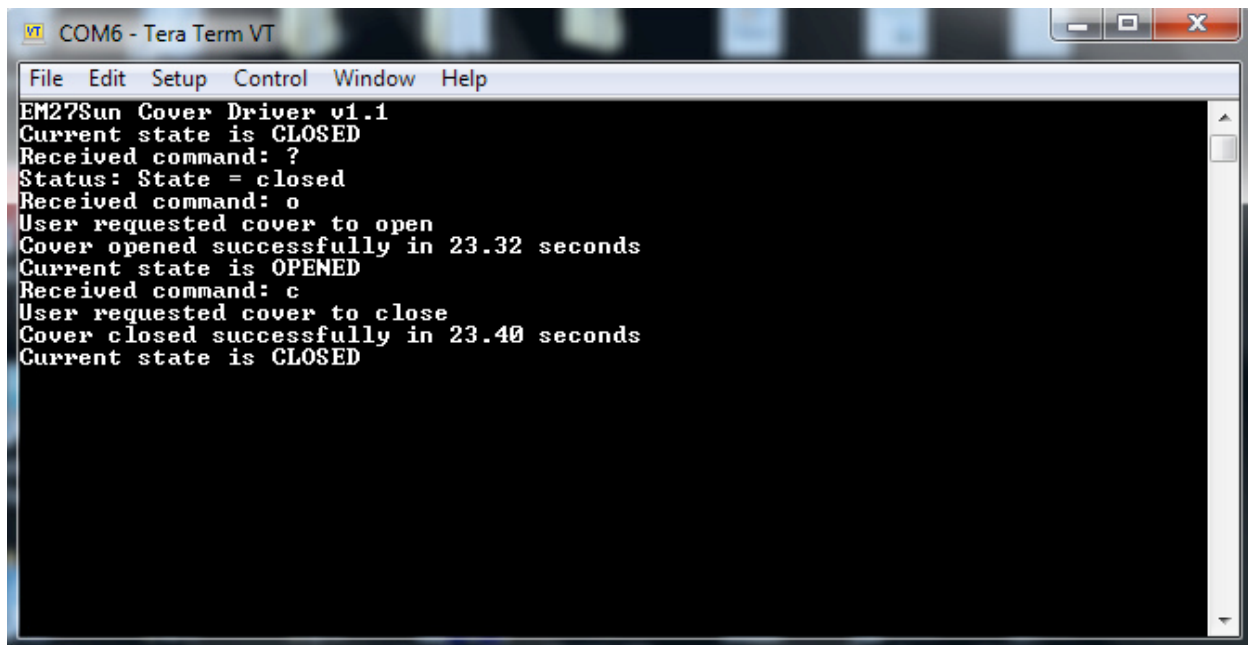
### Low-level operations:

You can communicate with the arduino by opening a terminal application (e.g. TerraTerm on PC or minicom on linux/mac). When you connect to the driver it executes a reset, which requests that the cover close. I am trying to prevent this reset, but have not succeeded yet.

A typical “TerraTerm” session starts with selecting the com port to use as in the figure below.



After selecting the port (COM6 in this case, but could vary if the device is plugged into a different USB port), select “OK” and the session will start. Once the session is running, you can type the single letter commands as described above and the arduino will respond. In the following image, I opened the driver, and sent the commands “?”, “o”, and “c”. Note that TerraTerm simply sends these letters when typed. Some terminal emulators may require that you hit return.

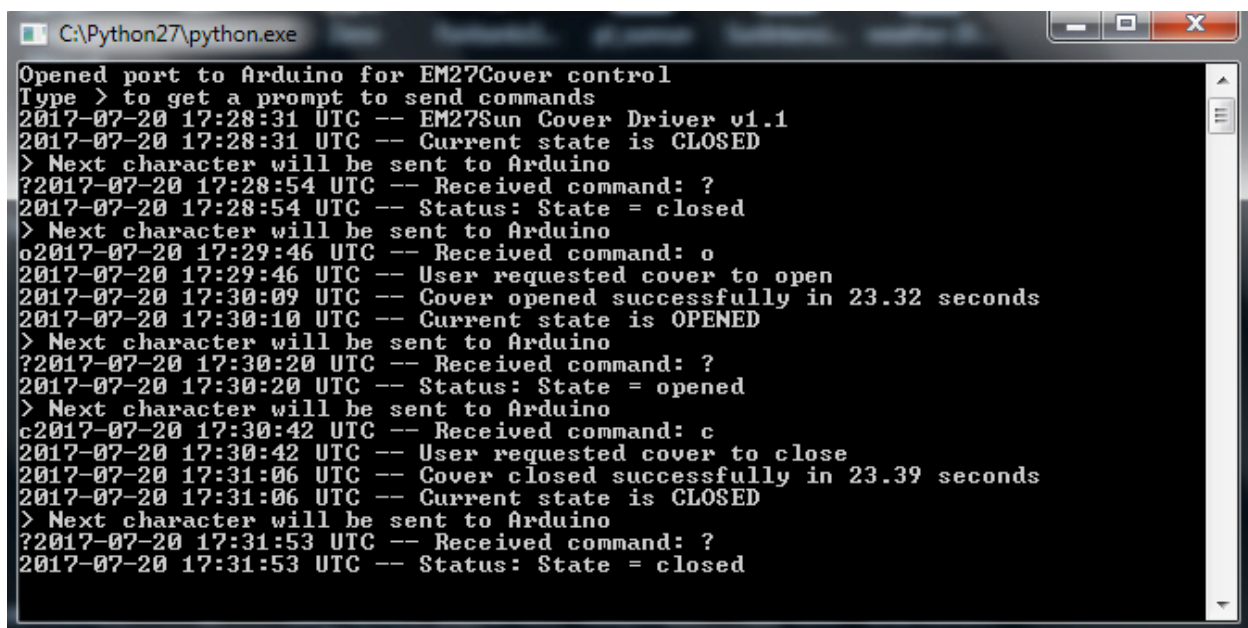


```
COM6 - Tera Term VT
File Edit Setup Control Window Help
EM27Sun Cover Driver v1.1
Current state is CLOSED
Received command: ?
Status: State = closed
Received command: o
User requested cover to open
Cover opened successfully in 23.32 seconds
Current state is OPENED
Received command: c
User requested cover to close
Cover closed successfully in 23.40 seconds
Current state is CLOSED
```

To quit, close the TerraTerm session with the red X. The arduino will maintain its last state.

### High-level logging operations:

For higher-level operations, we wanted to log the date/time of all actions, but the arduino has no knowledge of the true time, so the host computer's time is used for logging. A python script called "EM27CoverLog.py" carries out this purpose. This logging application only will work on a windows computer at this time. The image below shows operation of the cover using this python application:



```
C:\Python27\python.exe
Opened port to Arduino for EM27Cover control
Type > to get a prompt to send commands
2017-07-20 17:28:31 UTC -- EM27Sun Cover Driver v1.1
2017-07-20 17:28:31 UTC -- Current state is CLOSED
> Next character will be sent to Arduino
?2017-07-20 17:28:54 UTC -- Received command: ?
2017-07-20 17:28:54 UTC -- Status: State = closed
> Next character will be sent to Arduino
o2017-07-20 17:29:46 UTC -- Received command: o
2017-07-20 17:29:46 UTC -- User requested cover to open
2017-07-20 17:30:09 UTC -- Cover opened successfully in 23.32 seconds
2017-07-20 17:30:10 UTC -- Current state is OPENED
> Next character will be sent to Arduino
?2017-07-20 17:30:20 UTC -- Received command: ?
2017-07-20 17:30:20 UTC -- Status: State = opened
> Next character will be sent to Arduino
c2017-07-20 17:30:42 UTC -- Received command: c
2017-07-20 17:30:42 UTC -- User requested cover to close
2017-07-20 17:31:06 UTC -- Cover closed successfully in 23.39 seconds
2017-07-20 17:31:06 UTC -- Current state is CLOSED
> Next character will be sent to Arduino
?2017-07-20 17:31:53 UTC -- Received command: ?
2017-07-20 17:31:53 UTC -- Status: State = closed
```

To try to assure that an operator will not accidentally send commands, this application requires that you request a prompt (by typing the character '>'), and after entering the prompted state, the next character will be sent to the arduino.

The logging software produces a log file called "EM27CoverLog-YYYY-MM-DD.txt", where the date is the local date during which time the file was written. The log file is opened in append mode, so more observations on the same date will end up in the same file. The log file captures and time/date stamps the operations of the cover, and for the interaction above, the log file is:

---EM27CoverLog-2017-07-20.txt---

```
2017-07-20 17:28:31 UTC -- EM27Sun Cover Driver v1.1
2017-07-20 17:28:31 UTC -- Current state is CLOSED
2017-07-20 17:28:54 UTC -- Received command: ?
2017-07-20 17:28:54 UTC -- Status: State = closed
2017-07-20 17:29:46 UTC -- Received command: o
2017-07-20 17:29:46 UTC -- User requested cover to open
2017-07-20 17:30:09 UTC -- Cover opened successfully in 23.32 seconds
2017-07-20 17:30:10 UTC -- Current state is OPENED
2017-07-20 17:30:20 UTC -- Received command: ?
2017-07-20 17:30:20 UTC -- Status: State = opened
2017-07-20 17:30:42 UTC -- Received command: c
2017-07-20 17:30:42 UTC -- User requested cover to close
2017-07-20 17:31:06 UTC -- Cover closed successfully in 23.39 seconds
2017-07-20 17:31:06 UTC -- Current state is CLOSED
2017-07-20 17:31:53 UTC -- Received command: ?
2017-07-20 17:31:53 UTC -- Status: State = closed
```

To quit the logging, simply exit the python script by closing the window (red X). Again, the arduino will maintain its last state.

### **Work in progress / troubleshooting:**

When the com port is opened, the arduino executes a reset, which forces the cover to close (its safe state). Thus, if you quit out of the software and then re-open it, the cover will close, potentially losing some data. I have tried to prevent this auto-reset, but as of yet have failed.

If windows assigns the arduino a new com port number (e.g. other than COM6), the python logging code will be broken. You can use TerraTerm to find the new port number, and then edit the python code line that defines the com port:

```
com.port = 'COM6' # you can change this port name if needed
```

The python logging code is unlikely to work on linux or MacOSX due to use of the module "msvcrt". To port to a different OS, this module will need a substitute for the new OS.

**Installation:**

To modify the arduino code, you need to download the arduino IDE from [arduino.cc](http://arduino.cc) and add the "AccelStepper" library.

If you cannot find the "virtual com port" for the arduino, you may be missing its driver. Look for changes in the host's attached devices when you unplug / replug the arduino to find the arduino's serial port name. If there is no new port appearing, you could be missing the driver. Downloading the arduino IDE should install the virtual com port driver for the board, but one might also be able to download the driver by other means (e.g. by searching in google).

To run the python code, you need to install python 2.7 (or latest version < 3) and load the serial module (which I believe is called "pyserial").