Organ shutters

Note:
To change the product logo for your own print manual or PDF, click "Tools > Manual Designer" and modify the print manual template.

Table of Contents

1 Introduction

This project is an interface system between the Syndyne system and the shutter drive mechanism mounted on the door. It accepts shutter control signals from Syndyne and drives the shutter motors as needed to open/close the shutters as requested.

It is designed to drive only one shutter assembly.

The intention is that both systems be entirely identical. HOWEVER, each Arduino board will have the limits stored in it which are unique to that pedal and shutter mechanism.

Hardware
The hardware is powered by 12VDC (for the Arduino) and 24VDC for the motors. The hardware uses a potentiometer to determine the current position of the shutters.

It has two circuit boards:

- The Arduino “Uno” (the larger board)
- The motor driver board (only using output 1 & 2)

The feedback potentiometer is a multiturn pot. It’s important to note that this pot does have an endstop at the maximum CW and CCW positions. It’s very important to understand that the shutter open/close range of pot positions must lie within the working range of the pot.

Firmware
The Arduino firmware is written in C++.

1.1 Theory of operation

Preparation of the "request"
The signal from Syndyne is a 7 bit value provided by any of the pipe driver boards configured to provide this information on its expression outputs (S1-7). We have taken the entire connector since note 73 is not used on that pipe driver board.

The 7 bit value from Syndyne is received on Arduino pins 1-7 (we ignore the value on pin 0 coming from note 73).

We need to know what values correspond to the pedal being in the fully open and fully closed positions. These values are stored in the EEPROM in the Arduino. These values are set as part of the calibration procedure. As such, these values are kept even after power has been removed from the system.

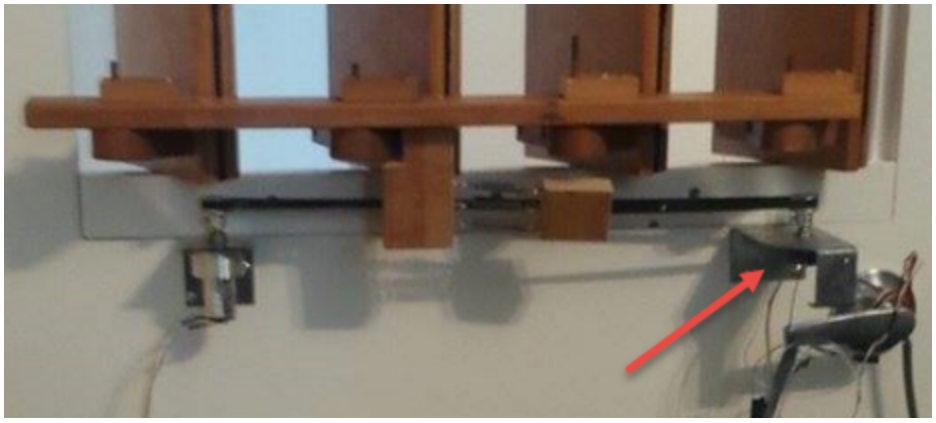
We now know that the pedal ranges from values X to Y as moved from fully open to fully closed. When we receive values from Syndyne, we can interpolate to calculate the percentage of the way between X and Y (ie: how "open" a position is the pedal at?)

Servo operation basics
A "servo" is a system that uses a motor with a feedback so that the system can determine what position the mechanism being driven is in. A servo has two inputs ("request" and "actual") and one output (motor drive control). Really, it's continuously comparing the two inputs and deciding the direction and speed of the motor so as to cause the actual mechanism position to match the "requested" position.

When the pedal position is changed, the servo will notice the discrepancy between the "requested" position and the actual position. It will respond by deciding what direction to drive the motor and also how hard to drive it. It'll also decide when to stop.

The intention is that the motor should reach the requested position fairly quickly. It should not be driving when the pedal is not being used. It may twitch slightly on rare occasions without being commanded to do so.

Servo feedback
The shutter mechanism also has a position sensing pot (shown below).



The Arduino must be aware what the pot position would be for fully open and fully closed. These values are set in the calibration procedure and are also stored in the EEPROM in the Arduino.

Having these limit values X and Y, we can use a measurement to interpolate how open the shutters really are (in terms of percent).

If a servo ever loses its feedback, it will drive hard and will never stop.

Consider this: A driver is trying to go forward. Supposing the only way he judges his speed is using the speedometer. If his speedometer breaks, he will wish to press the accelerator to the floor.

I have added some protection against this scenario. If it sees a value that's at either extreme of the sensor in the Arduino, it will consider the value to be faulty and indicate an error condition with the flashing yellow LED.

Servo comparison process
Having derived the % open value from the pedal and the % open value from the shutters, it knows what direction to drive the motor and how fast.

1.2 How to

Procedure: how to turn off the system
Safest is to disconnect the “brick” power supply from the wall outlet or unplug the Arduino power.

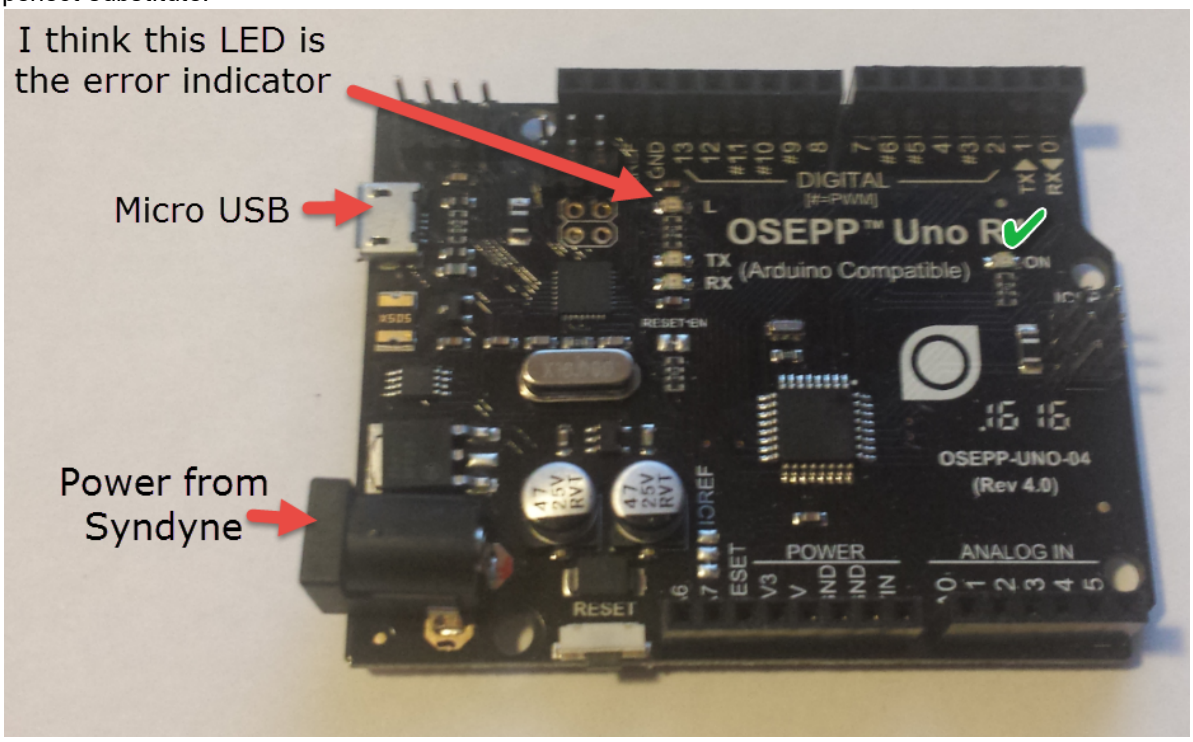
1.3 Hardware components

Healthy indications

The LEDs with the green checkmark should be illuminated (the motor controller one will be red).

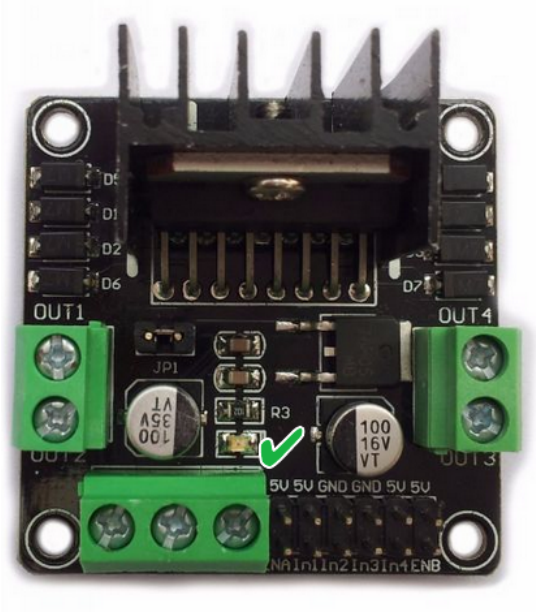
Arduino Uno

It's an "Arduino Uno" that happens to be made by OSEPP but any compatible Arduino Uno should be a perfect substitute.



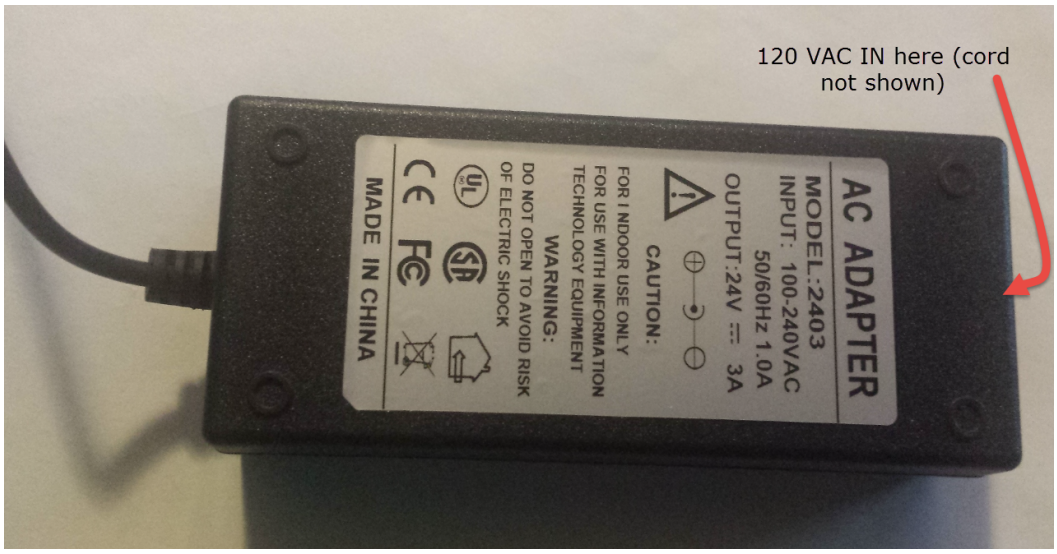
Motor driver board

This accepts instructions from the Arduino Uno. Red LED indicates whether the board is receiving power from the Arduino. It provides no indication whether the 24V supply is present.



24VDC "Brick" power supply

It'd be easy to forget this power supply is around so I'll show it here:



2 Design comments

We rely on the power coming from Syndyne so the Arduino and ULN2003A receives a ground from Syndyne. This means that if the Arduino were powered only from a computer via USB, the ULN2003A won't work properly.

3 Commissioning

Follow these procedures to address various problems that can occur.

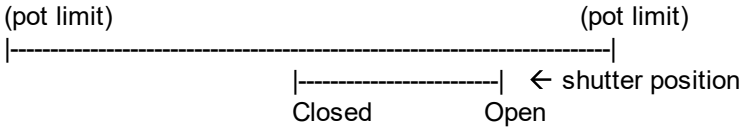
For all calibration procedures, the 24 power supply (the "brick") should be disconnected from AC power. This prevents the motor from driving against limits until the calibration is complete (this is noisy and can be quite a strain on the motor).

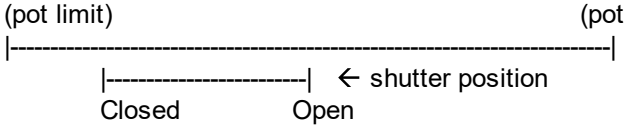
Depending on what you need to accomplish, you could start anywhere in this process but for a new system, you must start at the beginning.

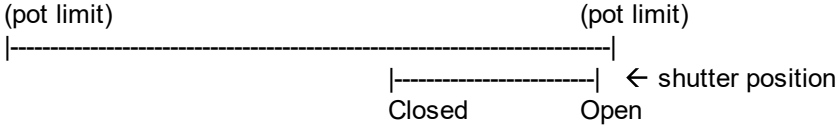
3.1 Feedback pot

The position detection potentiometer needs to be able to turn enough to allow the shutters to fully open and close. It is possible for the position of the pot to disallow one of these extremes. In such a case, a physical adjustment will be necessary. See the diagrams below.

Good scenarios:
Notice how the "limiting factor" is not the pot range but the range of the mechanism.

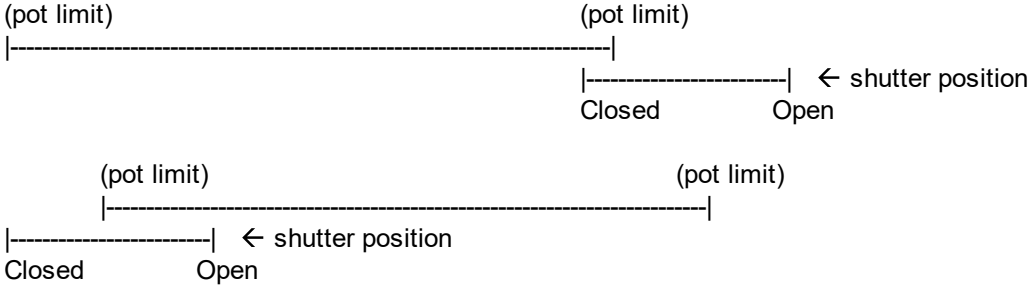






Bad scenarios:
Notice how the limit of the pot prevents the full range of motion that is required.

Solution: loosen/remove belt and turn pot by hand until it is OK



3.2 Tools

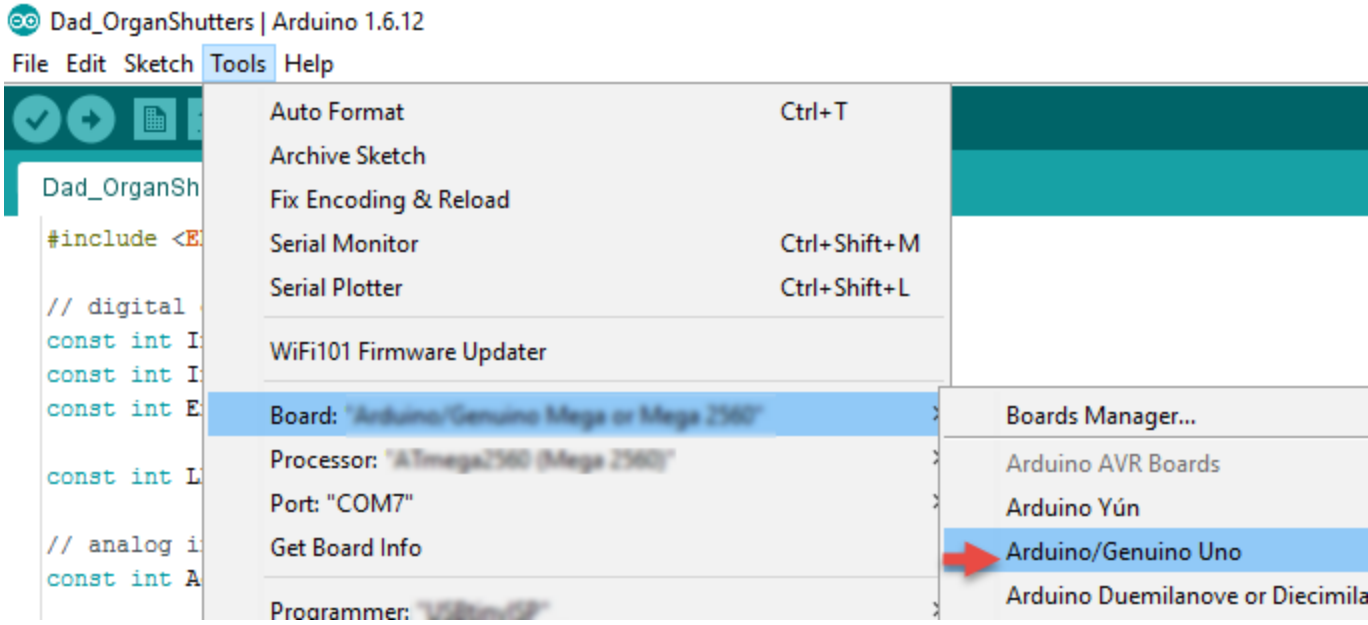
The Arduino can be programmed using the "Arduino IDE" software downloadable from https://www.arduino.cc/ . Choose the Windows installer.

You'll need a Micro USB cable to connect to your computer to the Arduino.

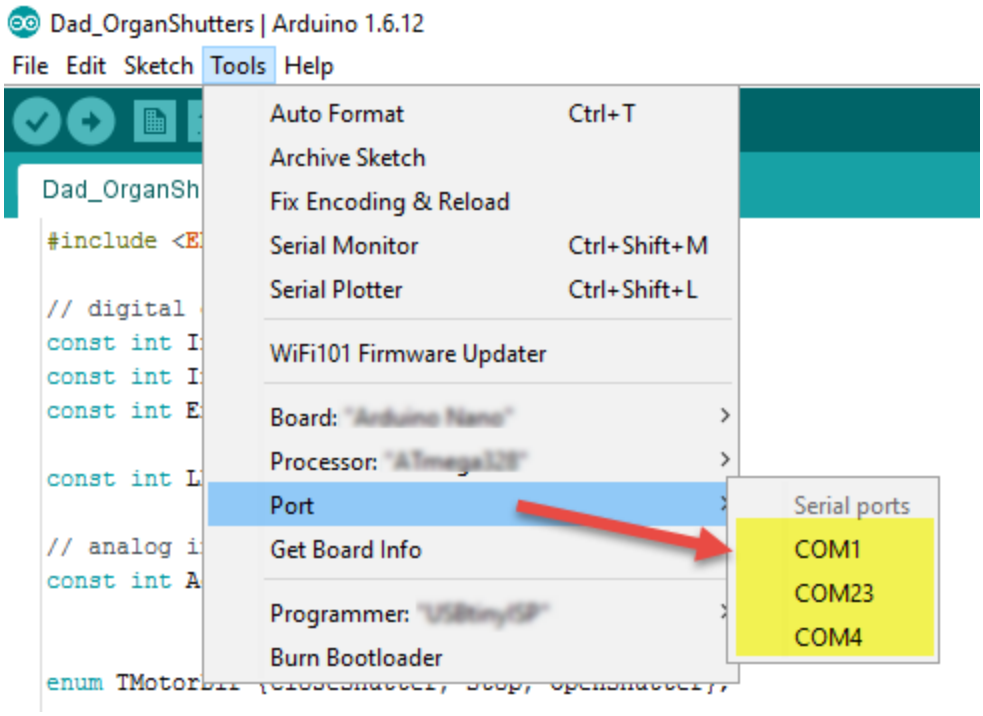
Terminal software
We use "terminal" software on your computer to communicate with the Arduino. It allows you to send/receive text from the Arduino.

There are many different possibilities but I shall offer the most common two. Before continuing, you'll need to get "terminal" software of some kind installed and ready to use.

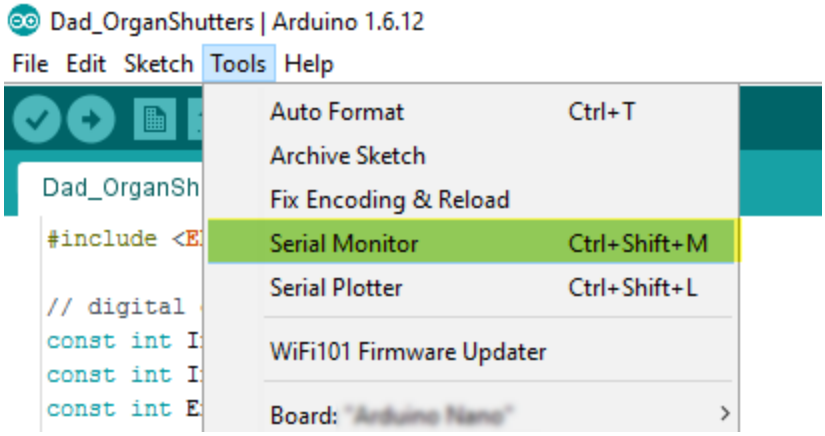
Getting "terminal" software on your computer: Arduino
1. Install the Arduino software (downloadable from https://www.arduino.cc/ , choose the Windows installer.)
2. connect your computer to the Arduino using a Micro USB cable
3. run the Arduino software
4. select the correct board (Arduino/Genuino Uno)



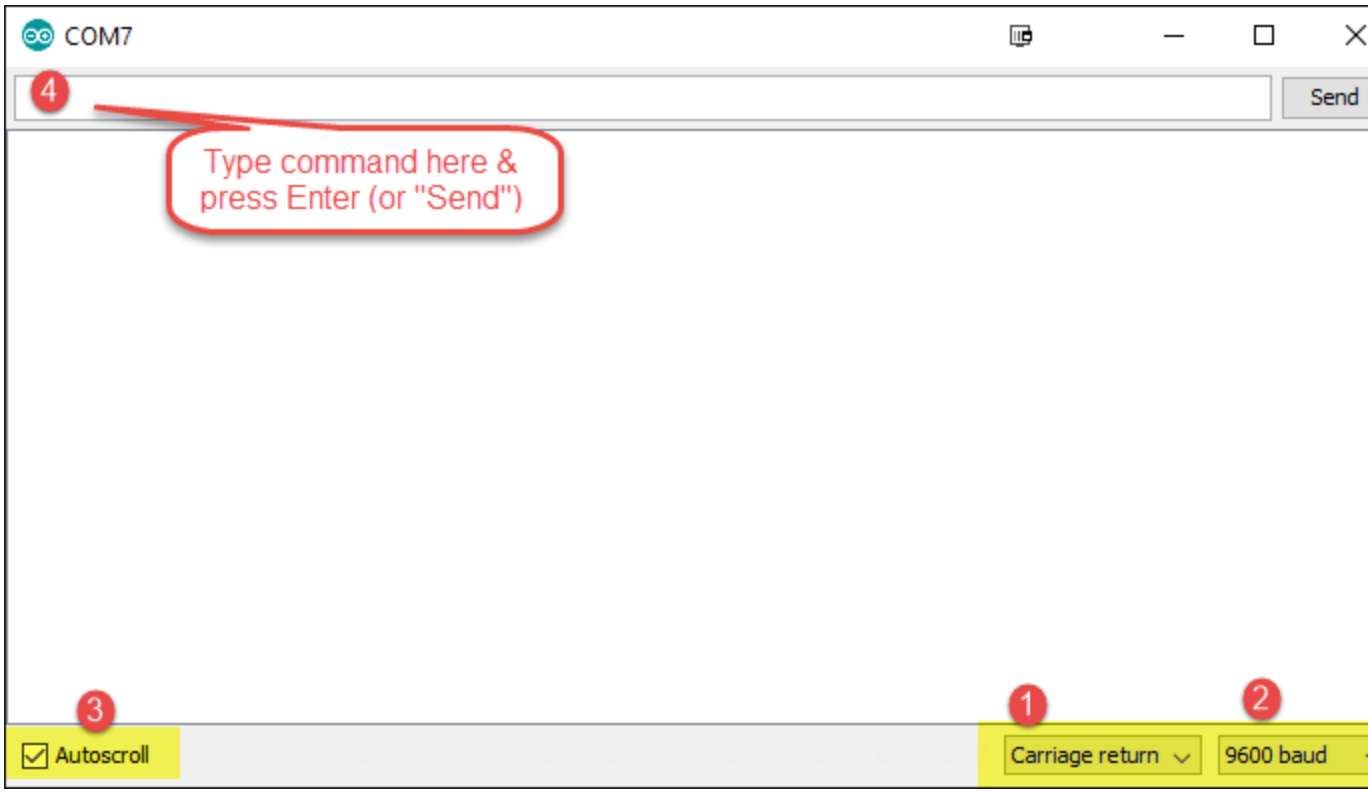
5. use the menu to select a port # (often, the Arduino Uno will be indicated beside the port #; not shown below)



6. open the serial monitor



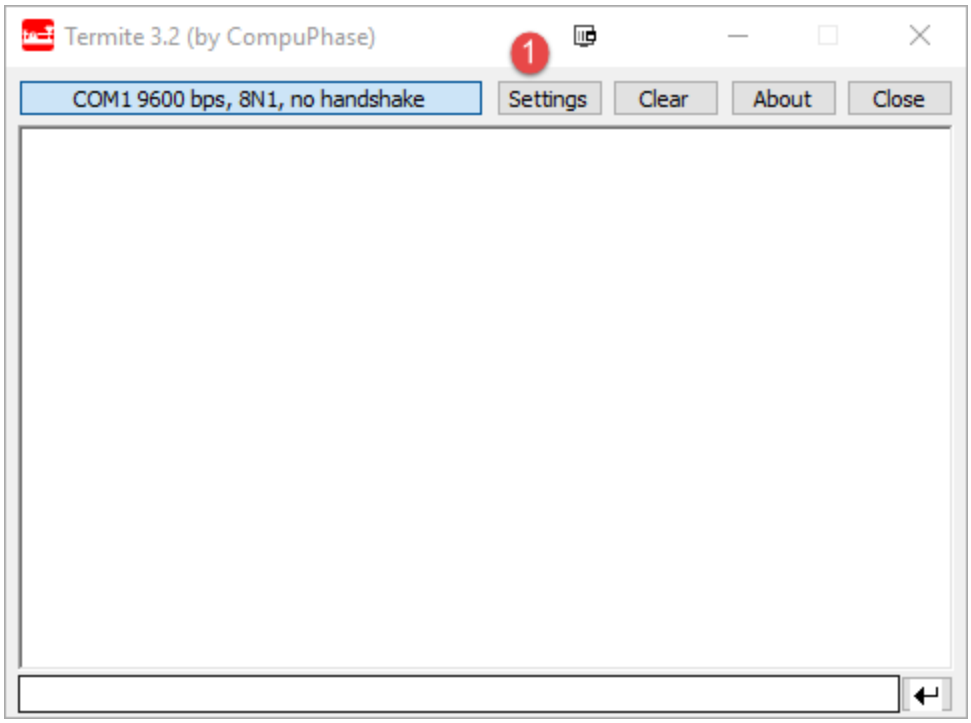
7. it will open the screen shown below
8. make the correct selections as shown below



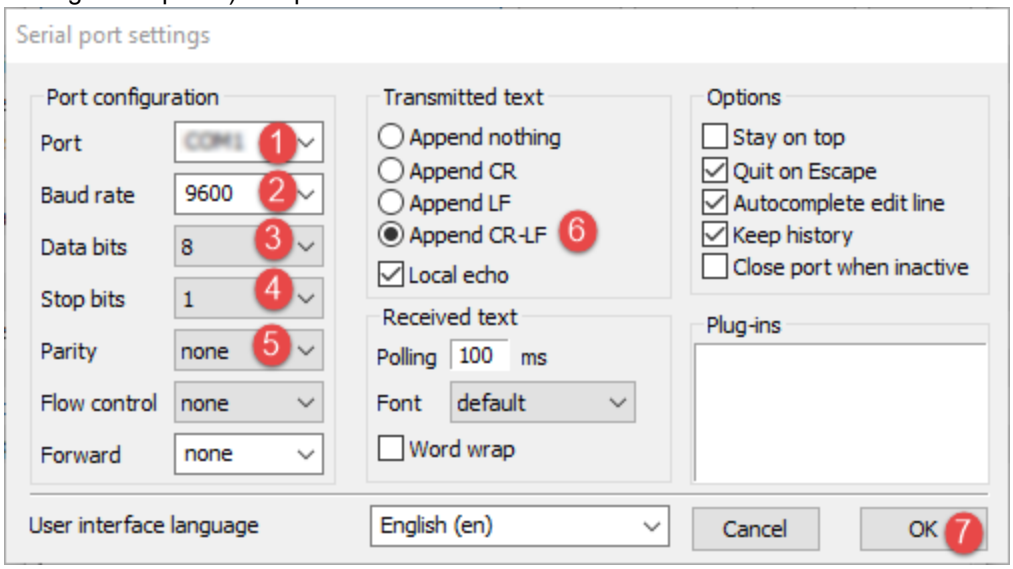
9. Type anything and press Enter (you should see a reply like “Unrecognized command!”); if not, try another port on step 4.

Getting "terminal" software on your computer: Termite

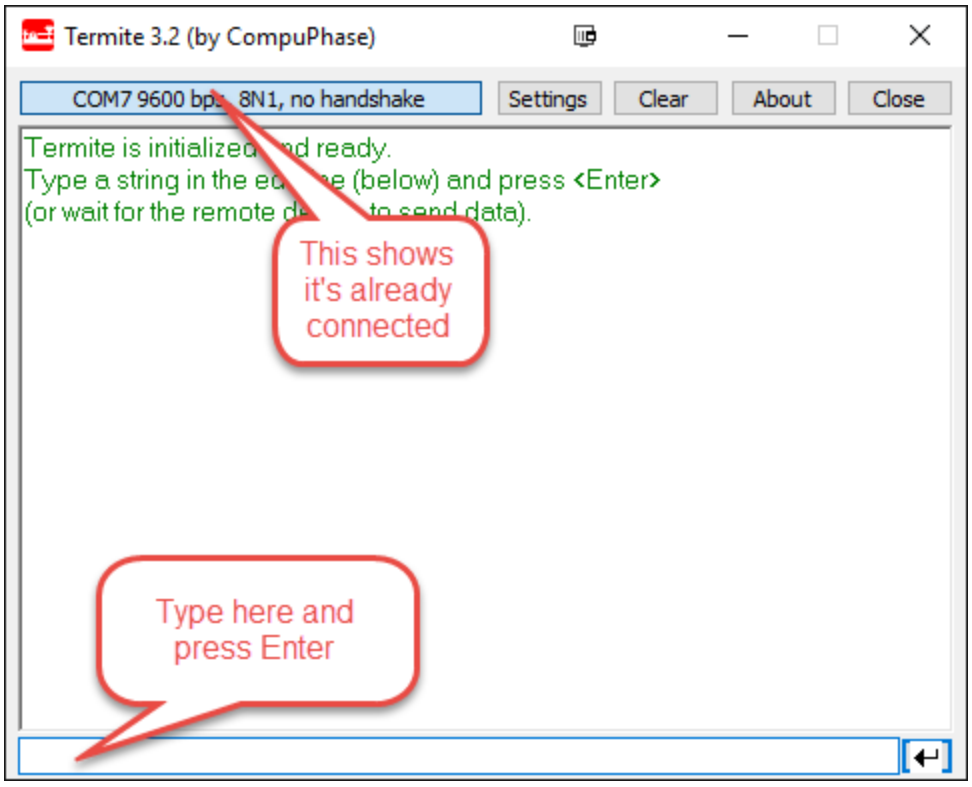
1. Install “terminal” software (such as https://www.compuphase.com/software_termite.htm) on your computer.
2. Run Termite
3. Press the Settings button



4. On your computer, connect a USB port to the Micro USB port on the Arduino (your computer might need to install drivers)
5. Make selections like below (finding the correct port # can require a process of elimination because Windows assigns the port #) and press OK.



6. It typically auto connects and looks like this:



7. Type anything and press Enter (you should see a reply like “Unrecognized command!”); if not, try another port on step 3.

3.3 Recalibrate pedal position

1. Disconnect the "brick" power supply
2. Connect your computer having the Arduino software installed to the Arduino's (Micro USB) serial port
3. Put pedal in fully closed position
4. Type SetPedalFullyClosedPosition and press Enter (Arduino will now remember this value)

```
SetPedalFullyClosedPosition
Executing command: SetPedalFullyClosedPosition
OK; new pedal 'close' limit is 124
```

5. Put pedal in fully opened position
6. Type SetPedalFullyOpenedPosition and press Enter (Arduino will now remember this value)

```
SetPedalFullyOpenedPosition
Executing command: SetPedalFullyOpenedPosition
OK; new pedal 'open' limit is 124
```

7. Optional: Reconnect the “brick” power supply
8. Optional: Type Test and press Enter

```
Test
Executing command: Test
Driving to fully closed position. Motor should finish quickly.
Press Enter when you're ready to continue.

Driving to fully opened position. Motor should finish quickly.
Press Enter when you're ready to continue.
```

Driving to fully closed position. Motor should finish quickly.
Press Enter when you're ready to continue.

Returning to normal operation.

3.4 Recalibrate shutter position

* extreme open/closed positions might not be perfectly accessible with the motor.

1. Disconnect the "brick" power supply
2. Connect your computer having the Arduino software installed to the Arduino's (Micro USB) serial port
3. Using your hands, gently move the shutters in a room to the “closed” position*
4. Type SetShutterFullyClosedPosition and press Enter (Arduino will now remember this value)
   SetShutterFullyOpenedPosition
   Executing command: SetShutterFullyOpenedPosition
   OK; new shutter 'open' limit is 322
5. Using your hands, gently move the shutters in a room to the “open” position*
6. Type SetShutterFullyOpenedPosition and press Enter (Arduino will now remember this value)
   SetShutterFullyClosedPosition
   Executing command: SetShutterFullyClosedPosition
   OK; new shutter 'close' limit is 321
7. Optional: Reconnect the “brick” power supply
8. Optional: Type Test and press Enter (see "Command list" for more information)
   Test
   Executing command: Test
   Driving to fully closed position. Motor should finish quickly.
   Press Enter when you're ready to continue.

   Driving to fully opened position. Motor should finish quickly.
   Press Enter when you're ready to continue.

   Driving to fully closed position. Motor should finish quickly.
   Press Enter when you're ready to continue.

   Returning to normal operation.

3.5 Command list

Command	Purpose
SetPedalFullyOpenedPosition	Record the value Syndyne provides when a given shutter pedal is set to the “fully opened” position SetPedalFullyOpenedPosition Executing command: SetPedalFullyOpenedPosition OK; new pedal 'open' limit is 124
SetPedalFullyClosedPosition	Record the value Syndyne provides when a given shutter pedal is set to the "fully closed" position SetPedalFullyClosedPosition Executing command: SetPedalFullyClosedPosition

	OK; new pedal 'close' limit is 124
SetShutterFullyOpenedPosition	Record the position where the shutters are when fully opened. SetShutterFullyOpenedPosition Executing command: SetShutterFullyOpenedPosition OK; new shutter 'open' limit is 322
SetShutterFullyClosedPosition	Record the position where the shutters are when fully closed. SetShutterFullyClosedPosition Executing command: SetShutterFullyClosedPosition OK; new shutter 'close' limit is 321
Test	Test Executing command: Test Driving to fully closed position. Motor should finish quickly. Press Enter when you're ready to continue. Driving to fully opened position. Motor should finish quickly. Press Enter when you're ready to continue. Driving to fully closed position. Motor should finish quickly. Press Enter when you're ready to continue. Returning to normal operation.
ResetError	Forget any error ResetError Executing command: ResetError OK
ShowInfo	Show current information showinfo Executing command: showinfo ROOM INFO ========= Status: ENABLED Raw: Pedal 'Open' limit is -1 Raw: Pedal 'Close' limit is -1 Raw: Pedal position is 124 Pct: Pedal position % is -1 Raw: Shutter 'Open' limit is -1 Raw: Shutter 'Close' limit is -1 Raw: Shutter position is 337 Pct: Shutter position % is -1
Disable	Disable the motor (after a restart, this will be forgotten) Disable Executing command: Disable DISABLED
Enable	Enable the motor

	Enable Executing command: Enable ENABLED
DebugOn	Show values it's working with (after a restart, this will be forgotten) DebugOn Executing command: DebugOn Debug is now on Pedal: 124 (%-1) Shutter: 323 (%100) Diff %: -1 Selected speed: 0 Direction: (0=cl, 1=st, 2=op) 1 Pedal: 124 (%-1) Shutter: 328 (%100) Diff %: -1 Selected speed: 0 Direction: (0=cl, 1=st, 2=op) 1
DebugOff	Normal operation Executing command: DebugOff Debug is now off

4 Troubleshooting

Troubleshooting: uncontrolled driving (gear noise)
If the motors are driving in a way that isn't healthy, unplug the "brick" power supply going to the motor driver board or unplug the Arduino power.

- Make sure belt slippage isn't a problem
- Make sure shutters move easily and full range is accessible
- Recalibrate shutter open, closed positions.

Troubleshooting: motor not driving
Is the yellow LED on the Arduino board flashing? (1 Hz). If so, see "Troubleshooting an error" section below.

Is the Arduino powered? If so, it will have at least one LED on.

Is the "brick" power supply plugged in and working? I don't think there are any visual indicators for this.

Assuming it's on but no error & not driving, you could connect your laptop, turn on debug mode ("DebugOn") and see what it's saying. You can also use "ShowInfo" to see the internal state.

Troubleshooting an error
If the Arduino detects something wrong, it will shut off the motors and flash its yellow LED on the Arduino.

Possible causes:

- Feedback loop broken (considered to be broken when the position feedback indicates 0 or maximum readout).
- The motor has been driving too long (suggests that it has stalled).

Solution:

1. Connect your computer (see "tools" instructions)
2. If there is an error, the Arduino will tell you about what's wrong.
3. You could use the "DebugOn" command to see more information about what the Arduino is seeing/ thinking

5 Repairs

Pedal mechanism
If the pedal mechanism were to be disassembled such that the position of the pot in relation to the pedal were to change, it will be necessary to recalibrate the pedal limits in the Arduino.

Shutter mechanism
If the shutter mechanism were to be disassembled such that the position of the pot in relation to the shutter were to change, it will be necessary to recalibrate the shutter limits in the Arduino.

Motor driver replacement
The motor driver circuit replacement requires no recalibration.

Arduino replacement
If the Arduino were to fail, the following can be completed:

1. Replace the Arduino board
2. Download the program into it using the Arduino IDE tool
3. Perform pedal & shutter recalibration procedure

Motor replacement
The motor is a 24 VDC motor. A suitable replacement could be a motor with a minimum torque of 40 oz-in, about 20 RPM, and 24VDC. If the motor were to be replaced, it's likely that the details of the motor speed algorithm in the routine "decideSpeedAndDirection" (in the Arduino's firmware) would need to be altered to optimize short and long-haul adjustments without "overshoots" for large changes and also stalling for small changes.

It could be adapted to use a 12 VDC motor but some extra considerations would be necessary:

- noise filtering to protect the electronics from motor interference in power supply
- noise filtering to protect the feedback pot signal from possible motor interference (capacitor?)
- it might be helpful to double up the wiring to the motor to handle the extra current draw required.

https://www.servocity.com/
https://www.sparkfun.com/
http://leeselectronic.com/en/

Motor belt replacement
You could use a coarser pitch of toothed belt but then you'll also need to change the sprocket to a matching pitch.

We have extra belt material attached to the door assembly.

Lee's electronics

6 Firmware source code

The firmware source code resides at:

https://github.com/MikeFromBC/PipeOrganShutters2017.git