Automated Router Table

# Arduino automated lift and fence

<insert a picture of the Automated router table and HMI here>

**The Why:** I got bored, had a 12 yr old daughter who was learning C and wanted to investigate robots and a close friend told me of a colleague who had a router lift that could be as precise as 1/100” on his router table.

**Why do I need it?:** Why not? It started as an interest in learning the Arduino platform and a desire to have a very precise plunge router in my router table and turned into a fun project that gives me very accurate, precise control over the height adjustments, repeatability, and simply is just really cool. I do quite a bit of high end humidors, specialty boxes, and cabinets which all require recall on mortises, raised panel door cuts, and edging. Setting up the router for repeat cuts, or having to redo a piece of stock because of mistake or needing more was simply becoming defeating and causing me to lose interest in doing the work. Automating the settings, storing settings and having the ability to automatically recall heights and fence positions makes it very easy to “pattern” components for reuse.

**The Research:** Since I already had [Norm's deluxe router table](https://www.newyankee.com/product/deluxe-router-station/) with an [Incra Mast-R-Lift II](https://www.amazon.com/Incra-Master-Designed-Rockler-Tables/dp/B005O8B5VQ/ref=asc_df_B005O8B5VQ/?tag=hyprod-20&linkCode=df0&hvadid=309869401414&hvpos=&hvnetw=g&hvrand=1311987861506154877&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9028292&hvtargid=pla-568935714885&psc=1&tag=&ref=&adgrpid=63364097444&hvpone=&hvptwo=&hvadid=309869401414&hvpos=&hvnetw=g&hvrand=1311987861506154877&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9028292&hvtargid=pla-568935714885) in it, I didn’t want to pay for the new lift from MLCS as the Incra worked just fine – it was just difficult to set depth on mortise or groves when having the piece sitting on the bit. Having reviewed the MLCS video, it occurred to me to find a way to connect a stepper motor to the lead screw on the Incra. After a couple weeks trying to figure that out – I realized it was a lost cause. Incra does such a great job on machining and design – there isn’t really a way to get to the lead screw without doing some major machining. Dead end

After finding the Fuyu FSL40 linear guide, I had the brain flash. What if you could connect the guide block to the Incra lift? Because of the 2 bands that make the router motor collar, the idea is born. Rig a connector to the collars and connect it to the guide block and use the linear guide as the lift. Everything else was math and programming – figure out the pitch, linear travel per step and start the design.

But…. How to control it? Joystick and DRO? Buttons in a box, slider?

Back to the MLCS… what about a Human Machine Interface? Would that work? – of course it would, but how much work and where do I get a cool 1? Well, as with everything, go to the web and the first hits are from Amazon. There it was the [Nextion 7" HMI](https://www.amazon.com/Enhanced-Capacitive-Enclosure-NX8048K070-011C-WIshioT/dp/B07DXN9WMS/ref=sr_1_3?crid=2A8IDWAZJ6Y9K&dchild=1&keywords=nextion+7&qid=1613008548&sprefix=sexton%2Caps%2C206&sr=8-3), exactly what I was looking for. Now – to put it all together and do the build.

**The Build:**

Without going into all the learnings and path to get to the final product, I’ll list a few of the learnings that will help you with the next build

1. Get an Arduino starter kit – almost every gadget is in there for prototyping and it comes with a board punch for wiring up all the components
2. **Ground, Ground, Ground!!!!** Be sure to check all your grounds and link them securely back to the Arduino. You may get very strange behavior and responses if you have unsecured grounds
3. The Nextion --- you need a windows interface for the IDE. Being a Mac user – this was a surprise, but windows machines are really cheap these days
4. Prototype – every component will have specific nuances, and isolating them, getting them working independently will help you debug/figure out your config. The Arduino community forum is your best friend, but you will need to isolate the problems you have, find similar problem questions/solutions and you’ll find your answers pretty quickly
5. The Arduino has 5v power connectors, but get yourself an external 5V power supply for all the low voltage power you need. Similar to bad grounds, this unstable power introduces anomalies that are almost impossible to trace
6. Prebuilt libraries – they are everywhere, use as many as possible, don’t reinvent what has already been proven to work and has a library already available.
7. The UI, the Nextion is a blank slate, blank screen. Every button, every graphic on the buttons, every control you want to use will require you to be imaginative and find web icons that will work for you. This is very time consuming as most have to be resized to fit, and the Nextion does not show “pressed” automatically the way you would expect in some other UI tools.
8. Think right up front about the “production” system. There are links at the end where you can find the IoT type devices, mounting brackets and industrial grade components to take your prototypes to production.

**The system:** It’s really pretty simple system. The HMI to control the stepper motors, the Arduino, 2 stepper motor controllers, 2 ball screw assemblies and an SD card shield. Couple of power sources, distribution blocks, and a bunch of wiring – off you go.

The complexity was in the code and figuring out all the different UI components, how to keep the memory usage down, and when to stop coding capabilities. This build blog is more about how the code works and why the rationale behind each of the sections of the UI.

[Graphical user interface, text, application, chat or text message

Description automatically generated](https://www.amazon.com/gp/product/B081T2R76W/ref=ppx_yo_dt_b_asin_title_o05_s00?ie=UTF8&psc=1)The system is comprised of the Arduino at the heart of the system. It is housed inside this Internet of Things (IoT) shield so the system can be hardwired and not just pin connectors. The shield has all the same pin connectors as the Arduino and has a number of extra 5V and ground connectors. They come in handy for the limit switches, but for the SD card and anything else that requires a stable power source, use an external 5V power source. (click on the picture)

The UI has 4 main screens. The main working screen, a settings screen used for configuration and the saved settings screen where you can store a repetitive setting and the special bits screen.

The main working screen is just that – the primary screen where you do most of the work

**The design:** The software is designed read a series of config files which control how the system is set up. I didn’t want to have to rewrite code if I had to rewire, or change settings in the config parameters. I ended up using the same model, snippets of the same code when I wrote the dust collection automation and gate control.

Text, letter

Description automatically generated***LiftPinsConfig.cfg*** --- this is the primary file which sets the overall pin configuration. It holds all the linking of the pins to the sensors and the wiring to the stepper motors. In the header of the file, the definition of pin numbers starts just below the config to the SD card. Remember, there is a non-printable character delimiter between each pin designation. That shouldn’t be modified unless you are going to modify the *setup()* function and re-define the delimiter character.

Figure : snapshot of LIftPinsConfig.cfg file

***Heights.txt*** --- a configuration file for the number of steps on the stepper motor controller to the linear inches of movement as well as setting the display values for the fractional position of the lift and fence

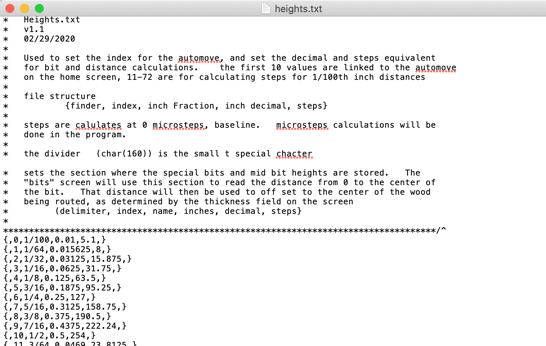


Figure : Heights.txt -- this file contains the configuration for the display and steps calculations

***Error***.***cfg*** --- a configuration file for dynamically displaying error messages in the error popup box. The popup has a label, yes/no buttons that dynamically display, and a detailed description of the error and remedy to the error

***RouterCory.log*** *–* filed used to store presets on the memory page. This filename can be whatever you like it to be and is set on the bottom of the settings screen. The format of the file is documented in the header.

Each of these files are specially delimited and described in the header. The design has a specific delimiter for parsing the file and knowing exactly what to do with each line.

Figure 2: LiftPins.cfg defines how the system is wired, how to control the speed of the ball screw and which file to use for storing presets

A picture containing text, electronics

Description automatically generated[A picture containing text

Description automatically generated](https://www.amazon.com/gp/product/B00DECZ7WC/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1)The stepper motor on the ball screw is controlled by the stepper motor controller. That controller is wired to the Arduino and controlled through specific pin configuration. As noted in the file above, the enablePin, directionPin, stepPin and the equivalent fence\* pins control whether or not the controller is on, which direction the motor should move and the pulses delivered from the Arduino. Because I have 2 ball screws on the table, I need 2 stepper motors and pins defined for each stepper motor. In the config above, pins 13, 6, and 11 control the lift while pins 30, 26 and 28 control the fence motor. There is a (-) pair for each of those pins that go to ground. The stepper motor is a 4 wire motor, which defines it as a 2 coil motor. The (+) (-) pair for each coil is wired into the controller and the controller gets a 12v power source to drive the motors. For production grade, I used the IoT versions from Mean Well. They are designed to run cool, mount on a Din rail and have plenty of wattage to run both stepper motors. (Click the picture for the source).

Graphical user interface, application, table, Excel

Description automatically generated**Text

Description automatically generatedIMPORTANT: The current code calculates travel and steps based on the dip switches set to 16 microsteps per step. If you modify to have finer control, you need to modify the steps parameter in the Heights.txt file to account for the change in microsteps. Or, if you want to modify the code, change the formula in the *calcSteps* function.**

Figure : Snapshot of the **Fractions for hundreths.csv**  Simple way to configure and create the Heights.txt content

A picture containing text

Description automatically generated

You will also notice in the config file, the documentation on the SD card. In the configuration line, there is a variable SD\_WRITE which is used when the code is to reload a card. Because the card has to be read in the setup section of the Arduino code, you can’t configure the SD card in the liftPins settings, if the configuration for your card is to change – you have to modify the setting in the Arduino code, recompile and reload to the system. I used the shields which I found on Amazon [Here](https://www.amazon.com/kwmobile-Reader-Adapter-Arduino-Microcontrollers/dp/B06XHJTGGC/ref=dp_prsubs_2?pd_rd_i=B06XHJTGGC&psc=1)

**The HMI interface:** As noted above, you will need a Windows machine to modify the interface. It is Nextion HMI, and the IDE for design can be found at [Nextion Website](https://nextion.tech/nextion-editor/). There is pretty good documentation for the functions, and the Arduino library can be found on the [Nextion - Arduino library on GitHub](https://github.com/itead/ITEADLIB_Arduino_Nextion). If you search on the Arduino cc forum, there are ways around using the library and simply reading the serial interface – but once I figured out some of the workarounds and nuances of the library – it was pretty solid.

The IDE is pretty straight forward, drop controls, configure parameters, and name uniquely name the controls you will want to build event handlers for in the Arduino interface. If you’re not familiar with C++ and how call back functions work, you’ll have a hard time understanding the linking between the HMI and the callbacks in the code. Simply put, the Arduino loop has a listener which monitors the configured serial port for command strings. The controls are configured in the NextListen variable declaration. When an event is detected, the call back function defined in your “attachPop” or “attachPush” section of the setup is called. Once you are in the callback functions – all programing logic is the same as any top flow programming language.

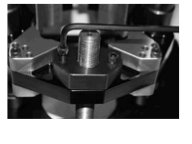
**Unless you are going to change the UI, you won’t have to worry about the linking of the controls to the behavior of the program.**

Once the UI is set, and all the controls/UI background/button images are set – you have to get the runtime to the HMI hardware. In the IDE, you create the TFT file on a micro SD. You can have only 1 TFT file on the SD or the HMI will not read or boot. Insert the SD into the slot on the HMI, cycle the power, and on boot-up, the HMI will read the configuration, validate it and give you a grey screen with the status. Take the SD out, cycle the power, and you should see the configuration come up on the Nextion HMI.

**Connecting the ball screws to the lift**

Figuring out how to get the ball screw assembly connected to the router lift took a while to figure out. But, in the end it was a pretty simple solution. I just had to mill a block of 2” aluminum to fit in between the 2 collars of the lift, drill some holes to mount the aluminum block to the ball screw rail nut, and take the ACME Lead screw collar off the router lift. Simple, and it works.

Drill 2 17/64” holes in the end of the guide rail assembly, 1 each side of the guide rail. These will be used to mount the assembly to the router lift plate. You don’t want to drill the holes in the plate until after you have the ball screw mounted to the router collar.

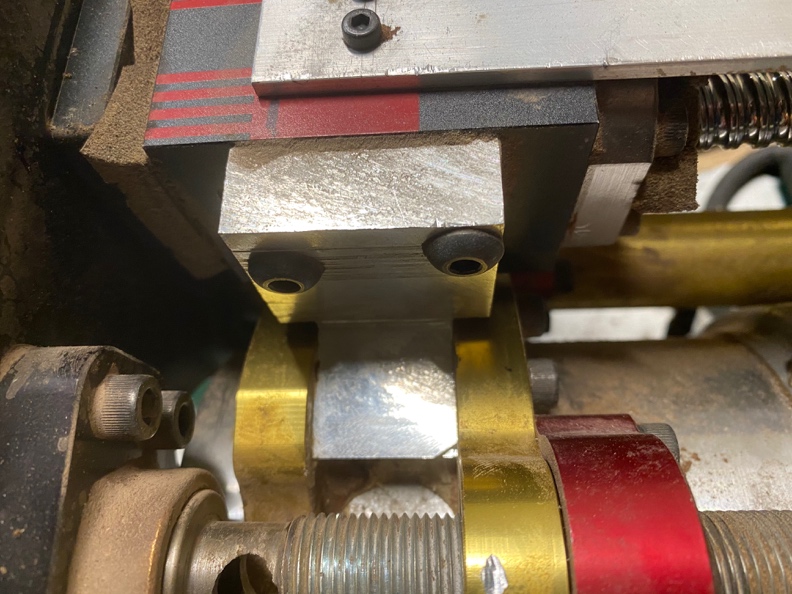
To create the free moving router lift, you have to remove the carriage nut assembly from the lift itself. The easiest way to explain this is to refer to the Incra Mast-R-Lift user manual, readjusting the thread tension section. In the [manual, figures 11-15](https://www.incra.com/manuals/MastRLiftII_Manual.pdf) are good pictures to understand the parts. Instead of reassembling the carriage nut assembly, simply remove the nut and anti-backlash nut from the LeadScrew. Once those nuts are removed, you will want to remove the dowel pin from the lock nut assembly. Now, the lift will move freely on the guide slides.

Remove nut and backlash nut from lead screw

Using a 7/32” allen wrench, remove the stop bolts from the bottom of guide rods. And slide the entire vice assembly off the guide rods. It’s easier to have the clamp assembly off the lift plate when you drill the holes in the plate, and adjust the aluminum block for the ball screw. My Mast-R-Lift was the original version, so the pictures are going to seem a little different than the manual. The components are the same, but it seems Incra improve on the Lead Screw and the locking mechanism.

Factory nut removed

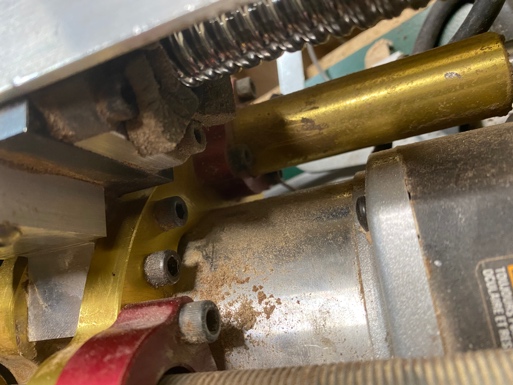
The following pictures are the working lift mechanism. I didn’t take any while building it – so I will describe the build after the pics.



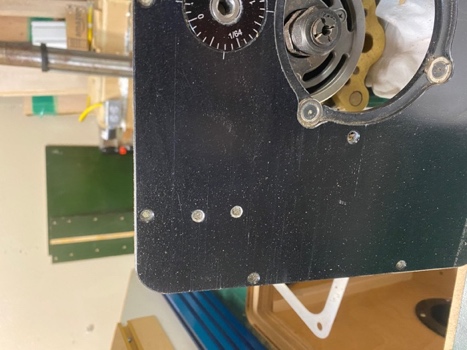
These are 4 mm cap screws. Using 2 of the mounting holes in the guide rail nut. The bracket is made from 2” x 2” solid aluminum stock

Using the existing ¼” shoulder bold from the lift, a 17/64” hole is drilled and the stock mounting holes are used.

When making this bracket, I used my table saw to chew out the bulk of the material. On the backside of the clamp mount, you will need to mill the corner at a 450 angle so the square block does not interfere with your router. Once you have the block the size you want, drill the 4mm holes in in the guide block side first. I simply clamped the piece next to the block, used a engineer square to mark the position of the holes, them center punched to guide the drill bit.

After you have the aluminum mounted to the guide block, slide the block between the clam, use a center punch to mark the position of the ¼” shoulder bolt, and drill a 17/64” hole through. Once you Replace the factor shoulder bolt, and you are now ready to mount the block to the router plate.

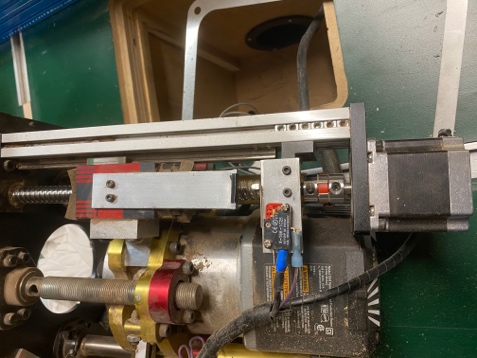
Factory shoulder bolt holding mounting block

Because I couldn’t find flat bottom taps, I had to drill the router plate through. With regular taps, I couldn’t tap the holes deep enough to have enough thread for the guile rail assembly to hold. These 2 bolts are the only thing holding the ball screw assembly to the router lift, and all the weight and pressure of the router, clamp and ball screw will be on these 2 bolts.

After mounting the aluminum block to the router clamp, slide the clamp back on to the guide shafts and slide the guide rail assembly tight to the router plate. The black end plate of the ball screw, should be tight against the router lift plate. Mark the 2 holes with a spring marker or a set punch. You will need to be very precise with the position of the holes, or, you will need to drill an oversized shoulder bolt hole in the aluminum block. If you are out of aligned even a little, your ball screw will bind and put excess lineal pressure and cause the ball screw net to “stick” in place and not move.

Drill your 2 holes with a #7 drill. Tap with ¼ -20 and chamfer the top side of the lift plate. You can either use a chamfer drill bit, or a small 45o router bit. I prefer to use the drill bit, as you can control the speed and depth. The chamfer will stop from marring and gouging your stock.

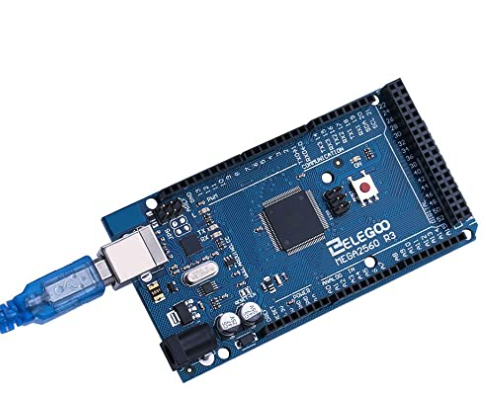
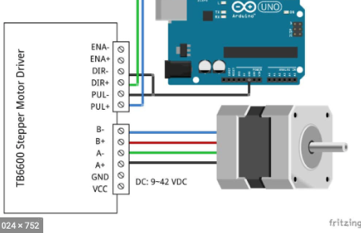
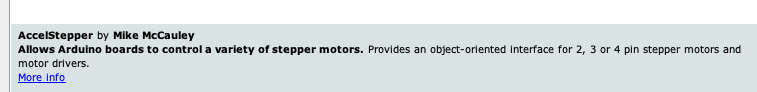
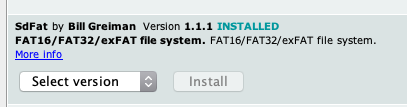
Because I used ½-20 x ½” socket head screws to mount the guide rail to the router lift plate, I counter sunk the holes 1/8” on the ball screw side.

Mounting the limit switches was a little bit of a challenge. I was using large single arm switches and simply drilled/tapped 3 mm holes to mount the switches. I don’t like this assembly, and will be modifying it to match what I did on the fence assembly. The smaller switches can be mounted directly to the guide rail with 4 mm screws and has a signal pole instead of a NO/NC relay type switch. These larger switches have a tendancy to fill up with sawdust and stop working. You know they stop working when you try to top out the router, and it just hammers on the end limit of the ball screw. My new assembly, will use the guide nut as the lever, simply press the micro switch and signal the Ardruino it has hit a limit.

Wiring the limit switches. Ground 🡪 common on the switch. 3V 🡪10K resister🡪 NC on the switch. The micro switch then has a green pole which goes back to the pin on the Arduino you have desinated for that switch. The larger switches here, are 3V 🡪 10K resister with Y to NC and the pin you designate and Ground 🡪 Common. In the micro switches, the “Y” is the switch itself and you don’t need the splitter prior to the switch.

**Connecting the ball screws to the fence**

**THE INSTALL:**

1. The system is designed to run on an [Arduino Mega2560](https://www.amazon.com/gp/product/B01H4ZLZLQ/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1). If you use a smaller Arduino, you will need to assess if you have enough pins for all the connections
2. *Before hardwiring and fixing everything to the router table, you may be best “dry fitting” all the components to make sure you have good ground connections, your pins are configured correctly, and your end stops are working correctly. I prototyped this in my office at the computer, did all the debugging and wiring and had a fully working system with the punchboards prior to doing any install on the router table.* Once I knew all the code was working, the wiring was solid, I just had to production mount on the router table. That way, I knew, any new bugs had to be with the install, not the code or components.
3. Mount the Arduino Mega2560 into the terminal block breakout. You’ll see all the pins are labeled and you have extra 5V, 3V and ground breakouts. This terminal block protects the board, and makes installation very easy
4. Mount the 12V power – this is only used to power the stepper motor drivers
5. Mount the 5V power – this is used to power the Arduino Mega, and all other 5V accessories. I only powered the end limit sensors through the Arduino onboard 5V power pins
6. I used a (+) and Ground [distribution block](https://www.amazon.com/gp/product/B07R44WC9R/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1) instead of wiring to power source directly
7. Make the 5V pigtail for the Arduino and wire it in to your distribution blocks. I used the simple connections from [Amazon - 12V power pigtail](https://www.amazon.com/gp/product/B07C7VSRBG/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1)  Don’t plug the Arduino to power yet, as you will want to have all other connections finished prior to powering the board
8. Connect all the components together according to your pin configuration in LiftPinsConfig.CFG. As noted, the SD card must be wired into the Arduino as defined in the header. If not – you will need to open the code and change the SD card parameter in the Setup function
9. Connect the (+) and ground from your stepper motors to the distribution blocks
10. Connect and configure your end stops. It’s hard to define exactly how you are going to do your custom setup, but I have defined my setup above
11. Copy LiftPinsConfig.cfg, Heights.txt, Error.cfg and RouterCory.log to a microSD.
12. Connect the SD Shield to the Arduino. The ping config is defined in LiftPinsConfig.cfg header file
13. Connect the Nextion HMI. It’s a simple serial interface. Black 🡪 Ground, Red 🡪 12V, RX 🡪 TX on Arduino (pin TX0 or Comms 14) , TX 🡪 RX on Arduino (RX0 or Comms 15)
14. Connect the stepper motors to your stepper motor controllers. Typically, your motors are wired in pairs with the Blue and Red as 1 pair and Black and Green being the other pair. Here is an example wiring he TB6600 controller.
15. Install the Arduino IDE on the laptop or computer you will use to upload the Arduino code to the Mega2560 Board
16. You will need the AcellStepper Arduino library. In the Arduino IDE, click Tools 🡪 Manage Libraries Once the dialog pops up, type “Accel” in the search box and scoll the results until you find  and install the library.
17. Copy the Nextion libraries to your Arduino working directory. You can get them from <https://github.com/itead/ITEADLIB_Arduino_Nextion> Add the library to your Arduino IDE by clicking Sketch 🡪 Include Library 🡪 Add Zip library. Select the Nextion library you downloaded
18. Add the SDFat Library by clicking Tools🡪 Manage Libraries Search for SDFat, scroll the results until you find  and click Install
19. If you have the system all wired up, it’s time to load the Arduino with the software.
20. Connect the Ardunio to you computer through the USB coard
21. Once plugged in, select Tools 🡪 Board and select the Ardunio Mega2560 (or whichever you are using Make sure your processor is the correct 1 by checking Tool 🡪 Processor
22. Select Tools 🡪 Port select the appropriate port. Typically, the selection already selected will be correct, as the IDE does a ping to the board once you select it
23. In the IDE, select Sketch 🡪 Verify/Compile Depending on the logging and verbose settings, you may see some red warnings appear in the status box. You can ignore them, unless they are fatal. If fatal, you have to debug your library setup. The code as represented on my GITHUB is my latest production code and compiles cleanly
24. Be sure you have power to the HMI. It must be started prior to the code being loaded and “setup” on the Arduino. If not, either the HMI will not drive your hardware, or you will see radical, unpredictable behavior. If you have started them out of order, just reset your Arduino and the hardware will re-link in the proper order.
25. Once you have the Verify/Compile working, you can load your code to the Arduino. Sketch🡪 Upload. If you run into errors, follow the standard debugging on the Arduino community forum. The typical errors I found was either the Arduino was not connected securely on the USB, or I had started an upload, not let it finish and tried to start another. If uploaded successfully, you should see a status message “Upload complete”
26. You have success! The Router lift and fence is ready to go.

Text

Description automatically generatedEngineering drawing

Description automatically generatedA close - up of a medical device

Description automatically generated with low confidenceA picture containing text, electronics

Description automatically generated