Lab 9-prelab

Physical Assembly of Hardware for I/O Labs

Due: Before the start of your lab section on November 2, 3, or 7

In addition to the professor and the TAs, you may freely seek help on this assignment from other students.

In the I/O labs, we will use a microcontroller board with some peripherals. In this prelab, you will assemble the hardware for the I/O labs.

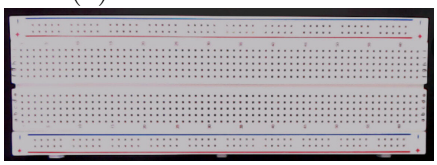
1 Obtaining the Hardware

The EE Shop has prepared "class kits" for CSCE 231; your class kit costs $30. The EE Shop is located at 122 Scott Engineering Center and is open M-F 7am-4pm. You do not need an appointment. You may pay at the window with cash, with a personal check, or with your NCard. The EE shop does *not* accept credit cards.

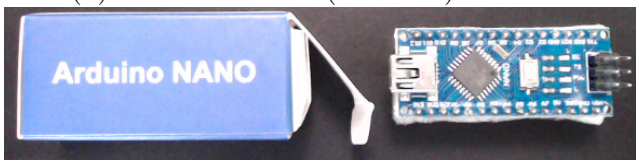
2 Inventorying the Hardware

Examine the contents of your class kit. It contains:

- One (1) full-sized solderless breadboard



- One (1) Arduino Nano (or clone) microcontroller board

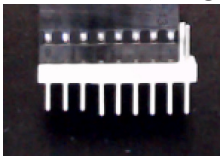


- One (1) USB cable (mini-USB shown; yours may be different)

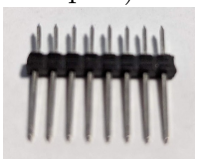
- One (1) 74LS20[1] dual 4-input NAND integrated circuit

- One (1) 4×4 matrix keypad

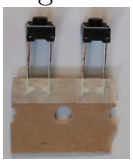
- One (1) 8-pin male-male header strip (might already be inserted into keypad's female connectors; might have more than 8 pins)



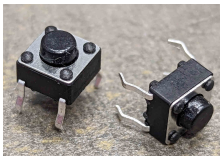
or



- Two (2) breadboard-mount momentary pushbuttons, aka tactile switches; these might have two leads (which might or might not be attached to cardboard strip), or they might have 4 prongs



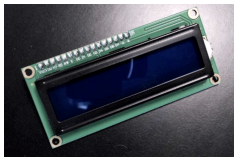
or



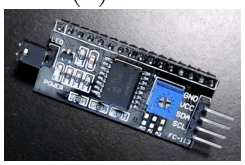
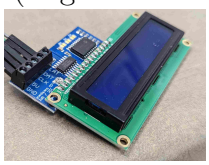
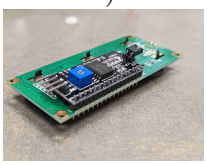
- Two (2) breadboard-mount slide switches

- One (1) 2×16 character LCD display module

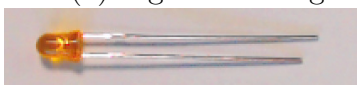
[1] Any 74x20 or 54x20 integrated circuit is acceptable.



- One (1) I^{2}C-LCD Serial Interface (might be attached to display module)

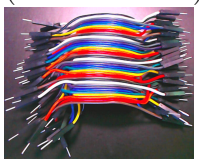
 or  or 

- One (1) Light Emitting Diode (LED) (color may be different from shown)

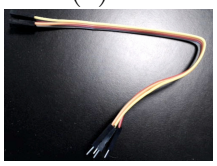


- One (1) 1kΩ resistor

- One (1) 40-conductor 10cm "rainbow" cable (male-to-male), *or* One (1) 20-conductor 10cm "rainbow" cable (male-to-male) and one (1) 20-conductor 20cm "rainbow" cable (male-to-male)



- One (1) 4-conductor 20cm "rainbow" cable (female-to-male)



There may be other items in the class kit. Set these aside; you will not need them for this prelab, though they may be used in a specific lab.

Assembling the Class Kit

You will assemble the hardware in the following steps. **At various checkpoints, you should pause to have a TA or classmate double-check your work.** When you do so, update the *checkpoints.txt* file to indicate who checked your work and when they did so.

You may want to store your partially- and fully-completed kit in a plastic food container or some other container to prevent jumper wires from being pulled out while in your backpack.

Note: *The following pages include diagrams and some photographs of the assembly. The wire colors in the diagrams do not match the wire colors in the assembly. The wire colors in the diagrams are coded by the purpose they serve, whereas the wire colors in the photographs are the colors of wires removed from the male-to-male rainbow cable.*

3 Microcontroller and IDE

A microcontroller, such as the Atmel ATmega328P[2] on the Arduino Nano, is a very simple processor when compared to a microprocessor designed for general-purpose computing. At the same time, a microcontroller has some features not present on a microprocessor, such as built-in analog-to-digital converters (ADCs).[3] A microcontroller board, such as the Arduino Nano, combines the microcontroller with other components[4] in a form factor convenient for experimentation.

The Arduino Nano has a USB port to connect to a computer and/or to provide power to the Arduino Nano. The six upward-pointing pins are used to program the Arduino Nano without using a host computer; we will not use these. It has thirty downward-pointing pins. `RX0` and `TX1` are used for asynchronous serial communication; as the USB interface also uses the same corresponding pins on the ATmega328P, we will not use these two pins (you will notice that when the Arduino Nano communicates with the host computer, the `RX` and `TX` LEDs will illuminate). Pins `D2-D13` are digital input/output pins. Pins `A0-A7` are analog input pins; however, `A0-A5` can also be used as digital input/output pins `D14-D19`. `AREF` (analog reference) is used to provide a reference voltage for the ADC (we will not use this pin). Pins `3V3` and `5V` provide regulated 3.3 volts and 5 volts for external circuitry; `5V` can also be used to power the Arduino Nano if connected to a regulated 5V power supply. `VIN` can be used to power the Arduino Nano if connected to an unregulated power supply, such as a 9V battery; the Arduino Nano's onboard voltage regulator will then provide regulated voltages needed. The `GND` pins are for the common ground; the ground portions of external circuitry and of external power supplies must be electrically connected to the Arduino Nano's ground. Finally, the `RESET` pins will reset the Arduino Nano if grounded (pressing the button in the middle of the Arduino Nano will also reset it).Note that, unlike a general-purpose computer, when a microcontroller is reset it will restart its program when the reset is released.

The ATmega328P microcontroller on the Arduino Nano is an 8-bit processor with 32KB of flash memory for the program and 2KB of RAM for data. While 8-bit logical operations, as well as 8-bit addition and subtraction, can be completed in one clock cycle, multiplication requires two clock cycles (16-bit operations require additional clock cycles). There is no

[2] https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[3] We will not use the ADCs in the I/O labs.

[4] Typically, a voltage regulator, a crystal oscillator, and a USB interface.

hardware divider, and there is no floating point hardware, so integer division (to include the modulo operation) and all floating point operations are performed in software, requiring hundreds of clock cycles.

If you have already read the first half of Chapter 8, the ATmega328P has separate instruction and data memory, similar to the simple processor design described in the first half of Chapter 8. If you have already read the second half of Chapter 8, the ATmega328P has a 2-stage pipeline (with *Fetch* and *Execute* stages). If you have already read Chapter 10, the ATmega328P does not have cache memory; however, the data memory is SRAM, the same memory technology used in microprocessors' memory caches. If you have already read Chapter 10, the ATmega328P does not have a memory management unit for virtual memory; instead, the ATmega328P uses only physical addressing.

3.1 Breadboard Terminology

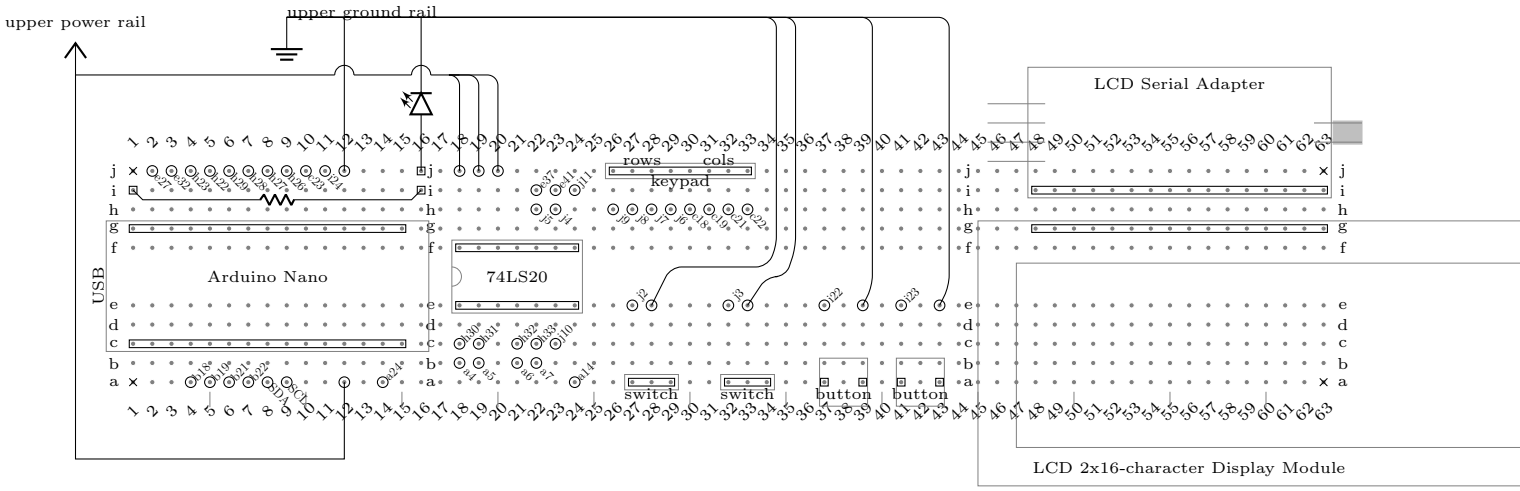
If you are not familiar with solderless breadboards, read the Breadboards for Beginners Guide at adafruit.com.

Even though breadboards are often viewed in "landscape" orientation (such as in the photo in Section 2 and as seen in the diagram figures) instead of "portrait" orientation, the numbered sections are called rows and the lettered sections are called columns. In the interest of preserving common usage, we will use this terminology. We will refer to specific contact points using the letter-number combination.

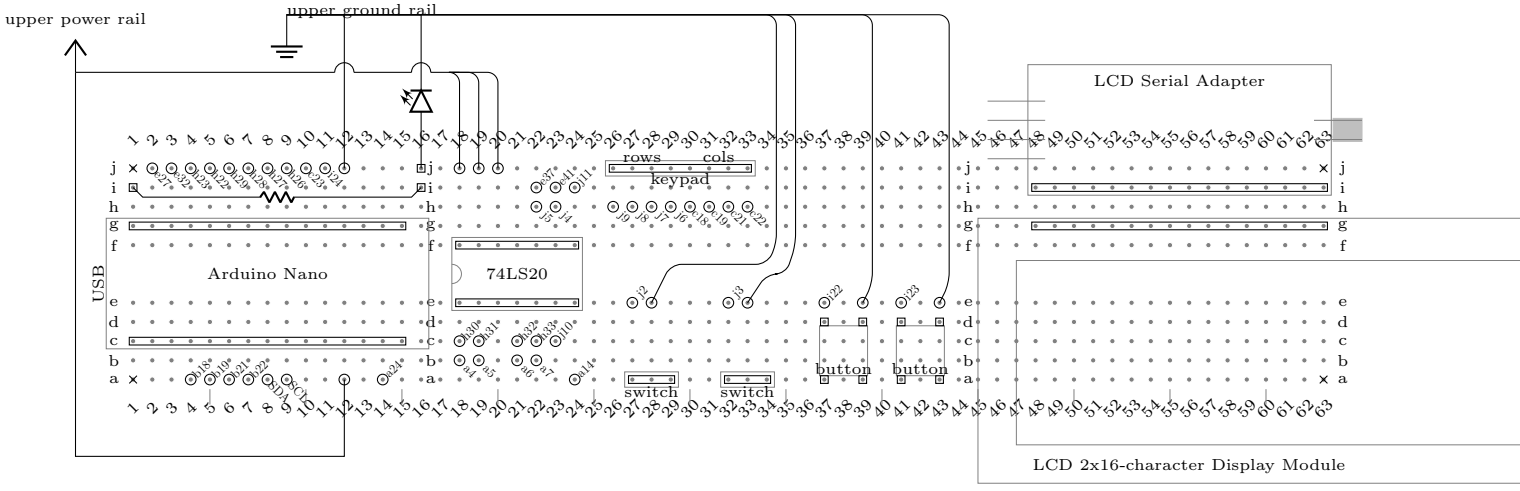
3.2 Optional: Breadboard Templates

Figure 1 is a set of two templates for the Cow Pi circuit (one with 2-lead pushbuttons and one with 4-prong pushbuttons) Each dot (·) represents a breadboard contact point. Each dot with a circle (⊙) is a contact point in which you will insert a jumper lead. Attached to most of these circles is the contact point for the other end of the jumper wire ( ). The footprints of several components are shown as light-gray outlines; resistors (—) and LEDs (—) are shown using their conventional symbols. Squares (⊡) are where you'll insert component individual pins, and rectangles ( ) are where you'll insert components' in-line pins. Finally, the four corners (×) are used to align the template on your solderless breadboard.

Optionally, print the page that has the template appropriate to your particular switches and pushbuttons. When (if) you do so, be sure to select "Actual size" (see Figure 2). If you mistakenly select a different option, the template will not line up properly with your breadboard: even a tiny scaling factor will add-up over the length of the breadboard. Using the lead from a jumper wire, punch holes into the four ×s at the corners (contact points a1, j1, a63, and j32); see Figure 3a. Place the lead from a jumper wire into each of the four holes, and insert the leads into the corresponding contact points on the breadboard, pinning the template to the breadboard. Confirm that the four jumpers are aligning the template to the breadboard by visually checking that the four leads are in the breadboard's contact points a1, j1, a63, and j32 (see Figure 3b).



(a) Template that uses 3-pin slide-switches and 2-lead pushbuttons.



(b) Template that uses 3-pin slide-switches 4-prong pushbuttons.

Figure 1: Templates to improve accuracy when constructing the Cow Pi circuit on a solderless breadboard.

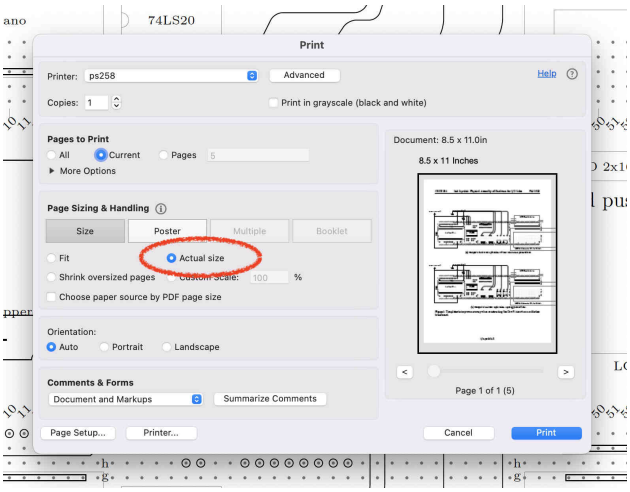
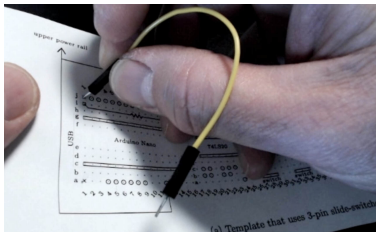


Figure 2: When printing a breadboard template, be sure to select "Actual size".



(a) Punching alignment holes in breadboard template.

(b) Confirming that the corners of the template are aligned with the corners of the breadboard.

Figure 3: Preparing a breadboard template.

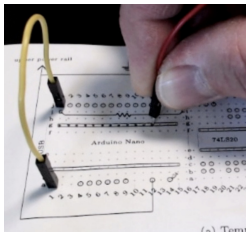


Figure 4: Pre-punching holes in the breadboard template before inserting a component.

3.3 Install the Arduino Nano onto the Breadboard

Orient the breadboard in front of you so that row 1 is on your left and row 63 is on your right; column a should be at the bottom, and column j should be at the top.

If you are using a breadboard template then use a jumper wire's lead to pre-punch holes into contact points g1–g15 and c1–c15. See Figure 4. After the paper provides some initial resistance, the jumper's lead will slide into the breadboard's contact point; remove the lead and move on to the next contact point. You want to pre-punch these holes because the paper provides less resistance when you're inserting a single lead than it does when you're inserting a component with multiple leads.

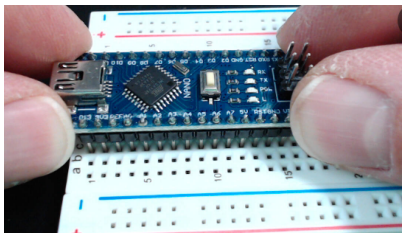
Remove the anti-static foam from the Arduino Nano's pins. You will place the Arduino Nano on the left side of the breadboard with the USB connector on the left (that is, facing away from the breadboard). Position the upper row of pins on contact points g1–g15 and the lower row of pins on contact points c1–c15. The left side of the Arduino Nano will obscure the labels for columns c–g. The right side of the Arduino Nano will cover contact points c16–g16 but won't use them. Double-check that:

- the pin labeled `D12` is in the upper-left, on contact point g1
- the pin labeled `D13` is in the lower-left, on contact point c1
- the pin labeled `VIN` is in the lower-right, on contact point c15
- the pin labeled `TX1` is in the upper-right, on contact point g15

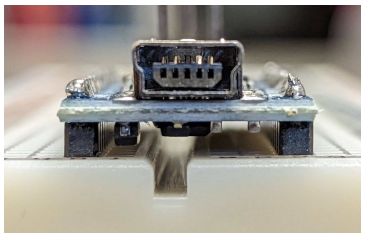
Gently press on both ends of the Arduino Nano to insert the pins into the contact points, using a slight rocking motion if necessary (Figure 5a). Press the Arduino Nano into the breadboard until it physically cannot be inserted any deeper (Figure 5b).

CheckPoint 1: Before proceeding further, have a TA or a classmate verify that you have correctly inserted the Arduino Nano into the breadboard. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

If you are using a breadboard template then you can now remove the jumper wires from



(a) Press gently on both ends of the Arduino Nano.



(b) The Arduino Nano fully inserted.

Figure 5: Inserting the Arduino Nano into the breadboard.

contact points a1 and j1; the Arduino Nano will keep the left side of the template pinned in place.

3.4 Install Arduino IDE

The Arduino IDE is installed on the lab computers. If you choose to install the Arduino IDE on your personal laptop, you can download it from https://www.arduino.cc/en/software. Alternatively, you can install a browser plugin to use the Arduino Web Editor. There are third-party plugins for many other IDEs; however, using these may limit our ability to help you if your have difficulties.

About Arduino Programs

An Arduino program is called a *sketch* for historical reasons.[5] For all intents and purposes, you can think of it as a C++ program[6] in which you write two functions, **`setup`** and **`loop`**, along with any helper code that you need. The file extension for sketches is ***.ino*** (as in, Ardu***ino***). The Arduino IDE will compile your sketch and link it to a **`main`** function that looks something like:

```
int main() {
    setup();
    while(1) {
        loop();
    }
}
```

(The actual **`main`** function[7] also calls a few other functions from the Arduino core library.)

[5]The Arduino language is based off of the Wiring language, which in turn is based off of the Processing language, which was designed to make computing accessible to artists.

[6]Your code in the I/O labs will be C code.

[7]https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/main.cpp

3.5 Connect to the Arduino Nano

Connect one end of the mini-USB cable to a lab computer or to your personal laptop.[8] Connect the mini-USB end of the cable to your Arduino Nano. The `PWR` LED will light up, and you may see the `L` LED repeatedly blink on-and-off. The `L` LED is connected to the Arduino Nano's pin D13, and Arduino microcontroller boards typically leave the factory with *Blink.ino* loaded, but it does not matter if yours does not have *Blink.ino* pre-loaded.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

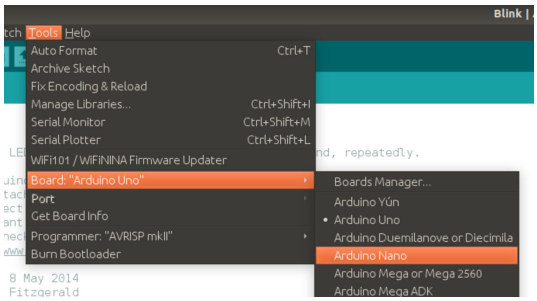
Open the Arduino IDE on the computer that your Arduino Nano is connected to. Connect the Arduino IDE to the Arduino Nano. If you are using Arduino IDE 1.8, see the Quickstart Guide on the Arduino Nano's documentation page for selecting the Arduino Nano board, processor, and COM port. If you are using Arduino IDE 2.0 or the Arduino Web Editor, COM port should be automatically detected. You will still need to select the board and processor; see Figure 6 and the discussion in Section 3.5.1.

3.5.1 Selecting the Correct "Processor"

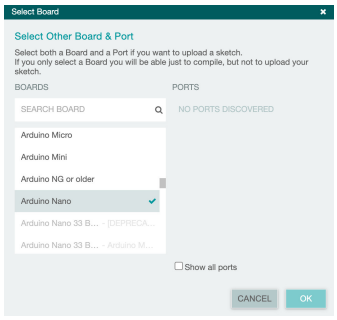
There are *three* choices for the Arduino Nano's processor, two of which specify the ATmega328P processor. Even though the difference is a USB interface issue, it is resolved through the Arduino IDE's "Processor" selection:

- Official Arduino Nanos use the FT232RL USB interface chip. Under the "Tools" menu, when choosing "Processor", select "ATmega328P".
- *Most* Arduino Nano clones use the CH340 USB interface chip. Under the "Tools" menu, when choosing "Processor", select "ATmega328P (Old Bootloader)". (If you are using the Arduino IDE 1.8.4 and earlier, which don't have the "(Old Bootloader)" option, simply select "ATmega328P").
- If you have an older Arduino Nano that the ATmega168 processor, replace it with one that has an ATmega328P processor.

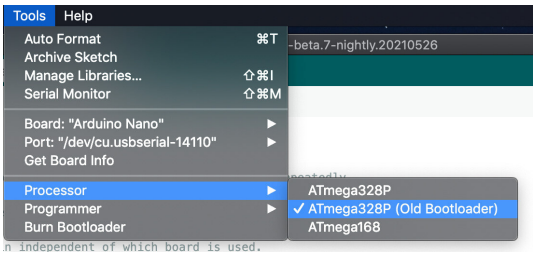
[8] You can connect it to a "wall wart" USB power supply to run the Arduino Nano, but you need to connect it to a computer to upload a new sketch to the Arduino Nano.



(a) Selecting the board with Arduino IDE 1.8.

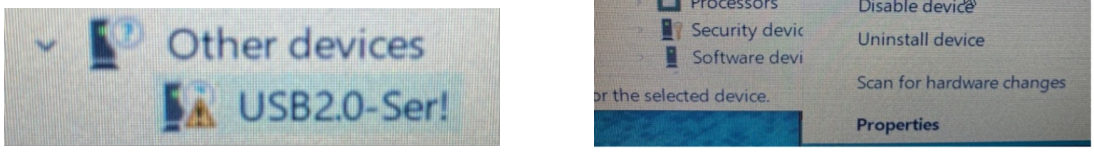


(b) Selecting the board with Arduino IDE 2.0.



(c) Selecting the processor after selecting the board.

Figure 6: Selecting board and processor in the Arduino IDE.



(a) (b)

Figure 7: Selecting board and processor in the Arduino IDE.

3.5.2 Updating USB Driver if Necessary

We have seen some Windows computers without the CH340 USB driver. If you encounter this problem and the Device Manager shows you the warning in Figure 7a, then the first thing to try is updating the driver. Right-click on USB2.0-Seri! (Figure 7b) and choose "Update driver". Then choose "Search automatically for updated driver software".

If Windows reports that "Windows has successfully updated your drivers" then you should now be able to connect to the Arduino Nano. On the other hand, if Windows reports that "Windows was unable to install your USB2.0-Ser!", then the How to Install CH340 Drivers page at sparkfun.com will guide you through manually downloading the driver and installing it.

Sparkfun's How to Install CH340 Drivers page also has instructions for installing the driver on MacOS and on Linux; however, we are not aware of any students needing to manually install the CH340 driver on MacOS.

3.5.3 No Driver Warning but Cannot Connect

Probably what happened is that your computer has the driver, but you're telling the IDE to connect to the wrong virtual COM port. The typical way to handle this is to disconnect the Arduino Nano from your computer, go to the part of the menu where you connect to the COM port, connect the Arduino Nano to your computer, and select whichever COM port appears after plugging in the Arduino Nano.

3.6 Upload a New Sketch

From the Arduino IDE's File menu, open the *Blink.ino* example:
File → *Examples* → *01.Basics* → *Blink*
Select *Save As...* and save the project as *MyBlink*.

Edit the values in the **`delay`** calls to change the delays between the LED turning on, off, and on again. Select values that will visibly have a difference, such as 250 or 2000. Compile the program using the "Verify" checkmark in the IDE's toolbar and make corrections if the program doesn't compile. Upload the program to your Arduino Nano using the "Upload"

arrow in the IDE's toolbar. (If you forget to compile first, the IDE will compile your program before uploading, but I find it useful to find compile-time mistakes before attempting to upload the program.)

If you successfully uploaded *MyBlink.ino* then you will see the following in the IDE's *Output* window:

. . . (elided configuration data). . .

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.01s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "/var/folders/p7/lx4gt70d0_34cpy8r0j3c95c0000gp/T/a
avrdude: writing flash (932 bytes):

Writing | ################################################## | 100% 0.33s

avrdude: 932 bytes of flash written

avrdude done.  Thank you.


--------------------------
upload complete.
```

and then the LED's on-off pattern will change, reflecting the **`delay`** values you assigned (Figure 8).

Handling Errors

If you get an error when attempting to upload a sketch, try these corrective measures:

1. Double-check that you have "ATmega328P (Old Bootloader)" selected (see Figure 6c).
2. Try uploading again (if you attempt to upload a sketch too soon after connecting your Arduino Nano to your computer, the USB interface won't have finished its handshake).
3. The Troubleshooting Guide recommends disconnecting your Arduino Nano and reconnecting it, then selecting whichever COM port appears.

If, instead of an error, your IDE "hangs" while collecting configuration data, try this corrective measure:

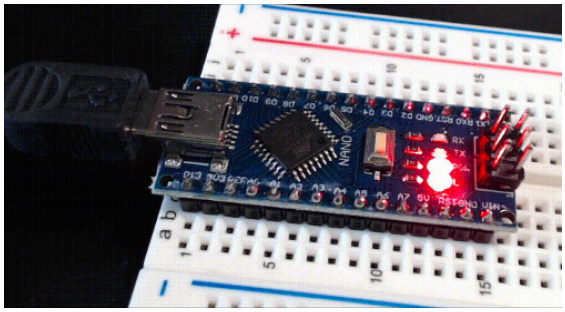


Figure 8: *MyBlink.ino* has a different on-off pattern than *Blink.ino*.

- Press the `RESET` button in the middle of the Arduino Nano; the IDE should begin uploading the sketch after you release the button.

CheckPoint 2: Before proceeding further, have a TA or a classmate verify that you have correctly uploaded new code to the Arduino Nano. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

4 Direct Input/Output Devices

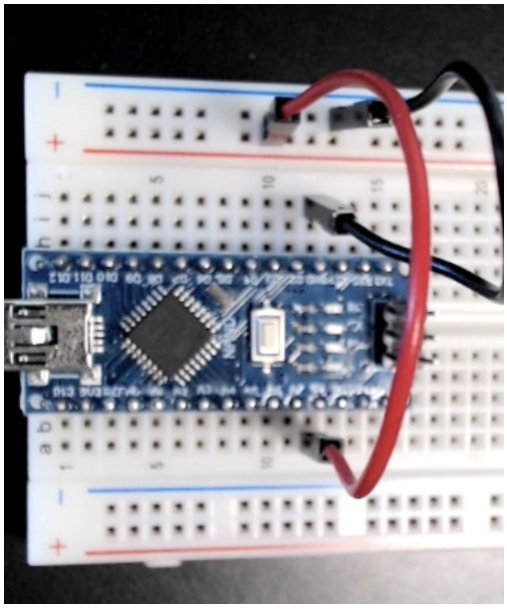
4.1 Connect Power and Ground to Power Bus Strips

The columns marked with red and blue stripes are the power bus strips, also known as the power rails. You will now provide power to the bus strips so that the other components can use power.

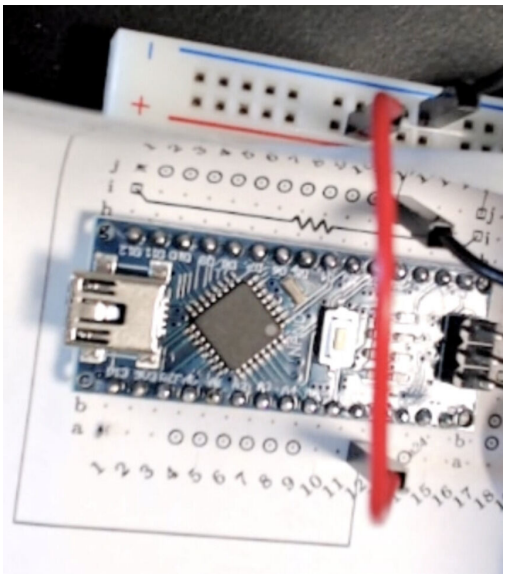
Before proceeding further, disconnect the USB cable from the Arduino Nano.

Take the male-to-male rainbow cable, and peel off two wires. Insert one end of a wire into contact point a12 (notice that contact point a12 is electrically connected to the Arduino Nano's `5V` pin, which is in contact point c12). Insert the other end of the `5V` wire into the upper power (+) rail marked with a red stripe. Now insert one end of the other wire into contact point j12 (notice that contact point j12 is electrically connected to one of the Arduino Nano's `GND` pins, which is in contact point j12). Insert the other end of the `GND` wire into the upper ground (−) rail marked with a blue stripe. See Figure 9a. If you are using a breadboard template, you will need to fold back the paper (or cut away some paper) to access the power rails; see Figure 9b.

CheckPoint 3: Before proceeding further, have a TA or a classmate verify that you have correctly connected the Arduino Nano to the upper power (+) rail and the upper



(a) Tapping power and ground from the Arduino Nano.



(b) Tapping power and ground from the Arduino Nano when using a breadboard template.

Figure 9: Providing power and ground to power busses.

ground (–) rail. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

4.2 Light Emitting Diode

You will now connect an external LED. An LED is a *light emitting diode*, and like all diodes it allows current to flow only in one direction. As shown in Figure 10, one lead on the LED is longer than the other, and this tells us which direction current will flow. When we insert the LED into the circuit, power will flow from one of the Arduino Nano's pins through the LED to ground. Most LEDs have so little internal resistance that, unless current is otherwise limited, enough current will flow through the LED to destroy the semiconductor material. The typical solution, which we will use, is to employ a *current-limiting resistor*. (If you look very closely at your Arduino Nano, you will see a tiny surface-mount resistor next to each built-in LED.)

Figure 11 shows a diagram of the components you will install for the LED output.

Take the 1kΩ resistor and place a right-angle bend in each lead about 0.4in (1cm) from the ends (we want the remaining length to be about 1.5in (3.8cm) – you do not need to be exact;[9] the leads are flexible enough that you only need to be approximate) – see Figure 12.

[9]If you want to try to be exact, you can use the breadboard's contact points to measure: they are 0.1in

Figure 10: The LED's longer lead connects to power; the shorter lead connects to ground.

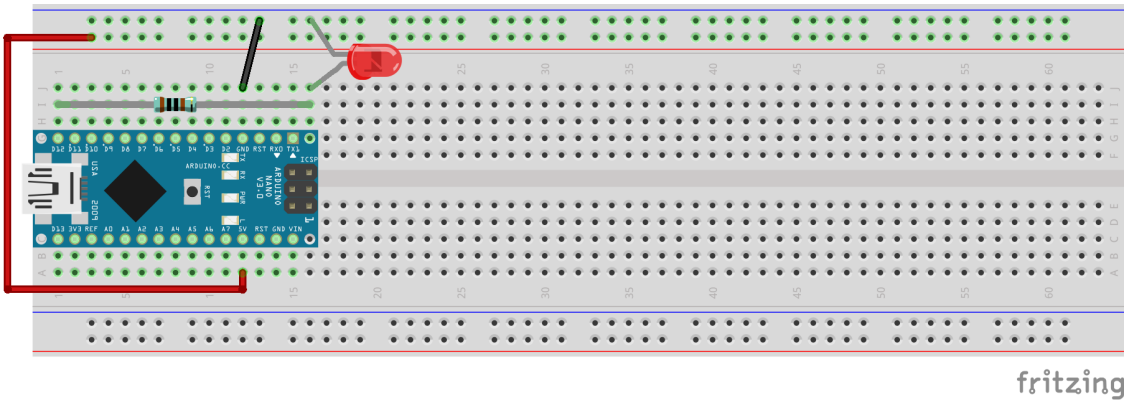


Figure 11: Diagram of component assembly for LED output.

Insert one of the resistor's leads into contact point i1 (electrically connected to the Arduino Nano's `D12` pin in g1) and the other into contact point i16. Gently press along the length of the resistor, causing the leads to deform slightly, until the resistor's height above the breadboard is about the same as the Arduino Nano's printed circuit board. See Figure 13a.

Take the LED and spread the leads apart slightly. Insert the longer lead (the anode) in contact point j16, and the shorter lead (the cathode) in the upper ground (−) rail. See Figure 13b.

When you have finished installing the external LED, there should be the electrical connections described in Table 1.

CheckPoint 4: Before proceeding further, have a TA or a classmate verify that you have correctly installed the LED and its current-limiting resistor. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

(2.54mm) apart.

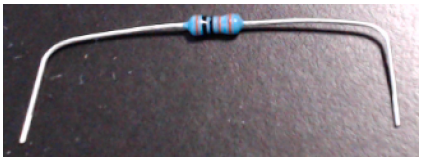
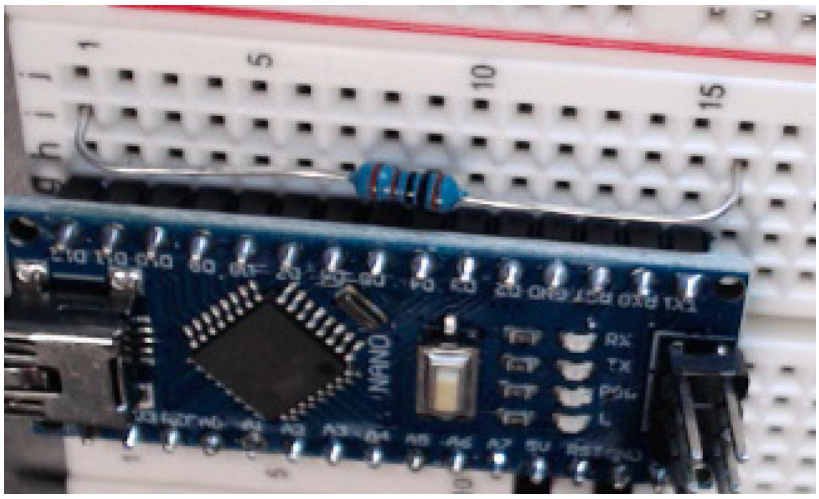
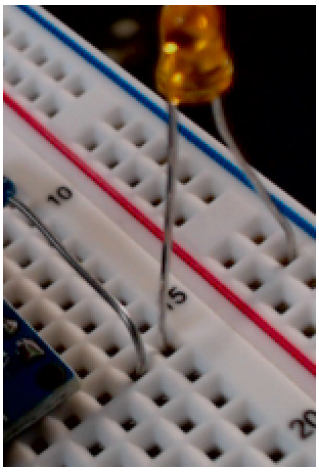


Figure 12: Bend the resistor's leads about 1cm from the ends.



(a) The resistor run between contact points i1 and i16.



(b) The LED's longer lead is in contact point j16, and its shorter lead is in the ground (–) rail.

Figure 13: Constructing the LED assembly.

LED lead	Resistor lead	Arduino Nano pin	Pulled High/Low
Anode	Right		
Cathode			Pulled Low
	Left	`D12`	

Table 1: Electrical Connections for External LED.

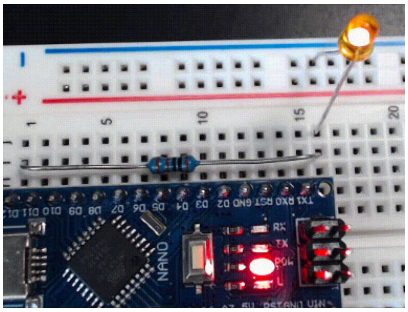


Figure 14: The revised *MyBlink.ino* causes the external LED to blink.

In the Arduino IDE, load *MyBlink.ino*. In the **pinMode** and the two **digitalWrite** calls, replace the `LED_BUILTIN` argument with `12`:

```
void setup() {
  pinMode(12, OUTPUT);
}

void loop() {
  digitalWrite(12, HIGH);
  delay(250);    // or whatever value you used
  digitalWrite(12, LOW);
  delay(1500);   // or whatever value you used
}
```

Re-connect the USB cable to your Arduino Nano. Compile the sketch and upload it to your Arduino Nano. Now, instead of the built-in LED, the external LED that you installed will blink (Figure 14).

4.3 NAND Gates

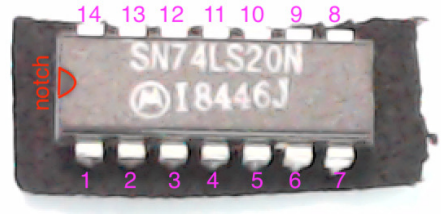
The 74LS20 "chip" is an integrated circuit (IC) that holds two 4-input NAND gates. It is in a *dual inline package* (DIP), and solderless breadboards are designed for the DIP form factor to straddle the breadboard's center divider. A notch on the left side of the DIP helps us orient the IC; the pins are numbered counter-clockwise from the lower-left to the upper-left (Figure 15a). The relationship between the 74LS20's pins and the NAND gates' inputs and outputs is shown in Figure 15b.

Figure 16 shows the wiring to install the 74LS20.

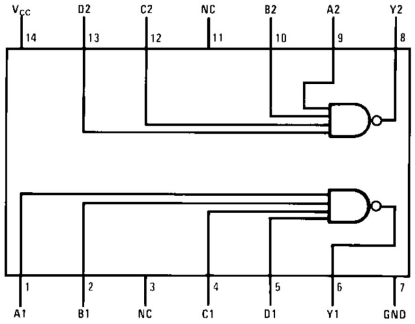
Before proceeding further, disconnect the USB cable from the Arduino Nano.

If you are using a breadboard template then use a jumper wire's lead to pre-punch holes into contact points f24–f18 and e18–e24.

Remove the anti-static foam from the 74LS20's pins. As described in the guide at adafruit.com, gently press the 74LS20's pins against a tabletop until they're approximately square to the IC's case. With its notch to the left, place the 74LS20 on the breadboard



(a) Pin numbers for the 74LS20.



(b) Connection diagram showing the 74LS20's pinout.

Figure 15: Pin information for the 74LS20.

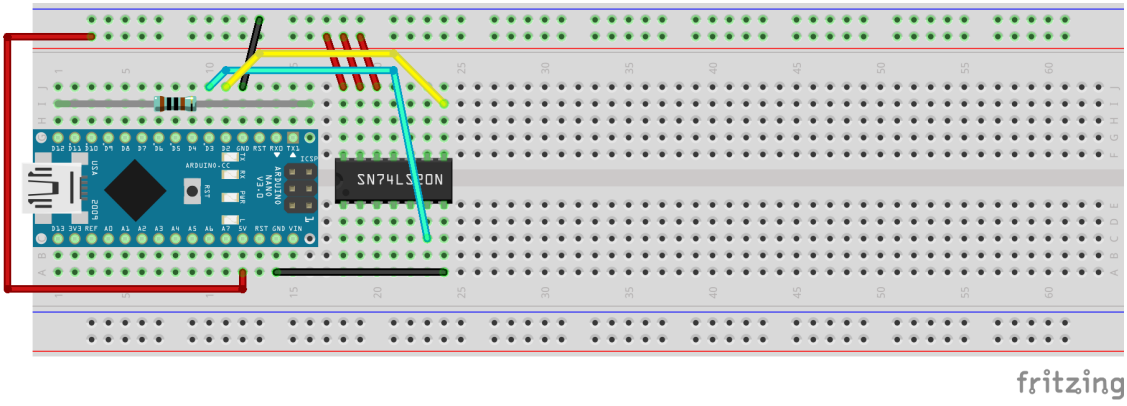
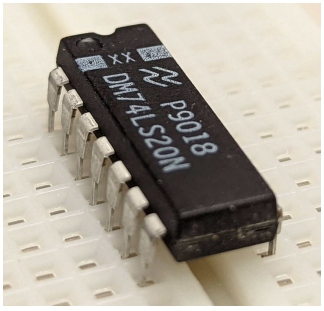
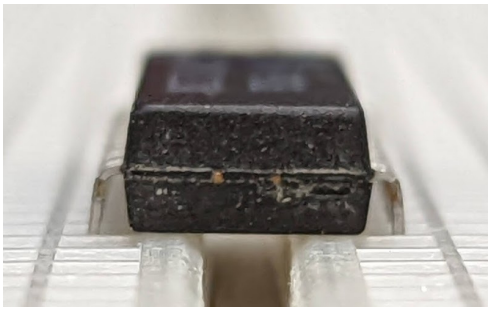


Figure 16: Diagram of the 74LS20's installation.



(a) Integrated circuit ready to be inserted.



(b) Integrated circuit fully inserted.

Figure 17: Inserting the 74LS20.

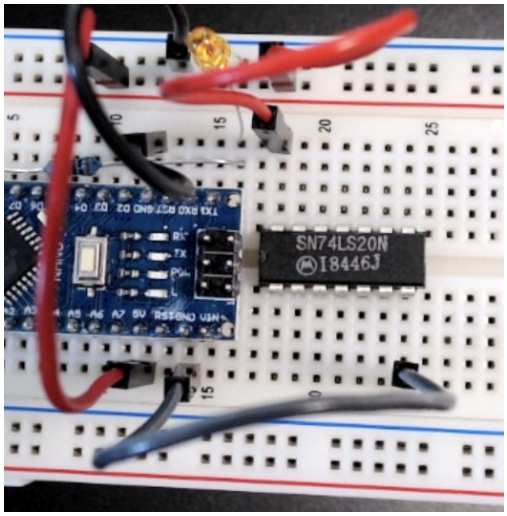
straddling the center divider on rows 18–24. Double-check that the 74LS20's pins 1–7 are on contact points e18–e24 and that pins 8–14 are on contact points f24–f18 (Figure 17a shows that the IC's pins are not splayed outside the contact points nor are folded under the IC's case). Gently press on the 74LS20 to insert the pins into the contact points, using a slight rocking motion if necessary. As shown in Figure 17b, the IC is fully inserted when its case is flush with the breadboard.

Peel one wire from the male-to-male rainbow cable; use this wire to connect contact point j18 (electrically connected to the 74LS20's V_{CC}, pin 14) to the upper power (+) rail. Peel off another wire from the male-to-male rainbow cable; use this wire to connect contact point a24 (electrically connected to the 74LS20's `GND`, pin 7) to the contact point a14(electrically connected to one of the Arduino Nano's `GND` pins). See Figure 18a.

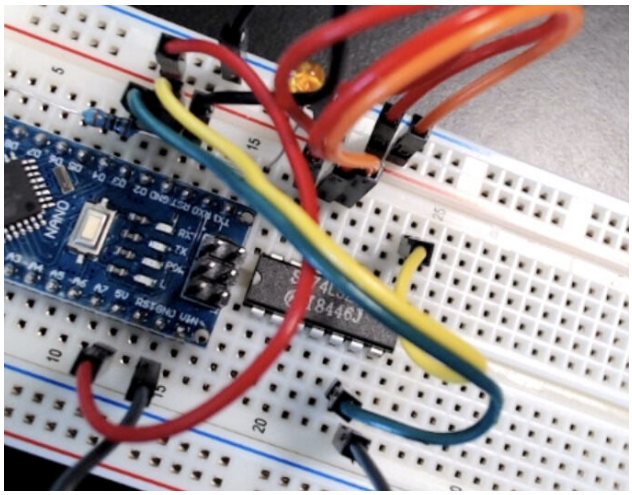
Peel four more wires from the male-to-male rainbow cable. Use two wires to connect contact points j20 and j19 to the upper power (+) rail. Use another wire to connect contact point i24(electrically connected to the 74LS20's `Y2`, pin 8) to contact point j11 (electrically connected to the Arduino Nano's `D2` pin). These three wires configured the 74LS20's upper 4-input NAND gate to act as a 2-input NAND gate; you will provide the inputs in Section 4.6. Use the fourth wire to connect contact point c23 (electrically connected to the 74LS20's `Y1`, pin 6) to contact point j10 (electrically connected to the Arduino Nano's `D3` pin); you will provide the inputs for the lower 4-input NAND gate in Section 5. See Figure 18b.

When you have finished wiring the 74LS20, there should be the electrical connections described in Table 2.

CheckPoint 5: Before proceeding further, have a TA or a classmate verify that you have correctly inserted and wired the 74LS20. Update *checkpoints.txt* file to indicate who checked your work and when they did so.



(a) The 74LS20 connected to power and ground.



(b) The 74LS20's outputs connected to the Arduino Nano.

Figure 18: Wiring the 74LS20.

74LS20 Pin	Arduino Nano pin	Pulled High/Low
6	D3	
7		Pulled Low
8	D2	
12		Pulled High
13		Pulled High
14		Pulled High

Table 2: Initial Electrical Connections for NAND Gates.

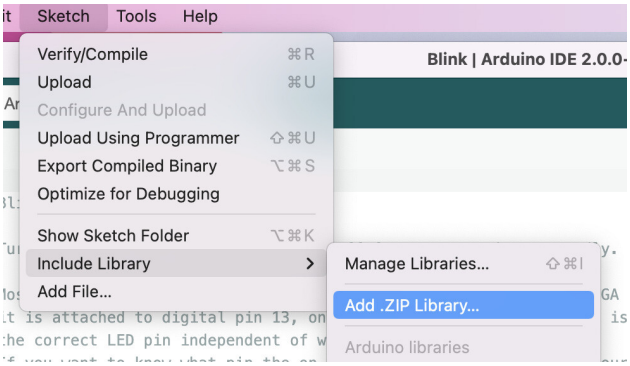


Figure 19: Adding a downloaded library to Arduino IDE.

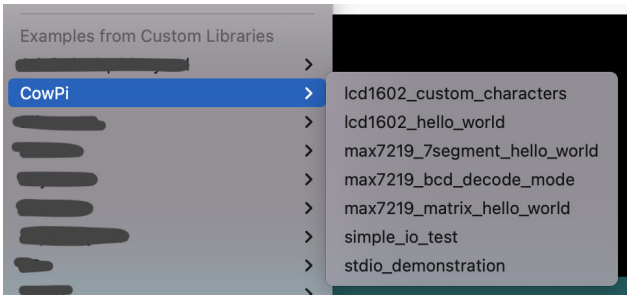


Figure 20: Code examples for the class hardware kit are now available.

4.4 Install the CowPi Library

If you have not already done so, download *CowPi.zip*.

In the Arduino IDE's Sketch menu, install the CowPi library:
Sketch → *Include Library* → *Add .ZIP Library...* (see Figure 19). In the resulting popup window, navigate to your "Downloads" directory (or wherever you saved *CowPi.zip* after downloading it), click on *CowPi.zip*, and click on "Open". You should immediately get the message, "Library Installed". Close the Arduino IDE and then re-open the Arduino IDE.

From the Arduino IDE's File menu, you now have additional examples under the "Examples from Custom Libraries" heading:
File → *Examples* → *CowPi* (see Figure 20). Open *simple_io_test*.

Find these lines in the **`setup()`** function:

```
12      cowpi_setup(SPI);
13  //      cowpi_setup(I2C);
```

Comment-out the `cowpi_setup(SPI)` line, and uncomment the `cowpi_setup(I2C)` line. The hardware kit is configured slightly differently depending on which serial communication protocol is used with the display module, and this semester we will use the I^2C protocol.

Compile and upload *simple_io_test.ino* to your Arduino Nano in the same manner that you did for *Blink.ino*. Open the Arduino IDE's Serial Monitor: Tools → Serial Monitor. You will see the message:

```
This demonstration makes no assumptions about your CowPi's display module,
    so your display module may display garbage -- that's okay.
The `cowpi_setup` function will be called with either `SPI` or `I2C` so that
    the CowPi library knows where your slider switches are.
    Your instructor should have told you which to use.
The simple I/O test will print the status of the keypad and of each
    button, switch, and LED every half-second.
Press the Enter key on your host computer's keyboard to start.
```

While the Enter key would be appropriate on a "raw" serial terminal, for the Arduino IDE's Serial Monitor, you will start the rest of the program using one of three techniques:

If you are running Arduino IDE 1.8 then place the cursor in the Serial Monitor's text-entry field and click on the "Send" button.

If you are running Arduino IDE 2.0 on Windows or Linux then place the cursor in the Serial Monitor's text-entry field and press Control-Enter.

If you are running Arduino IDE 2.0 on MacOS then place the cursor in the Serial Monitor's text-entry field and press Command-Enter.

Every half-second, the Arduino Nano will send a message to the Serial Monitor:

```
Keypad:               Column pins:  1111    Keypad NAND: 0
Left switch: RIGHT    Right switch: RIGHT
Left button: UP       Right button: UP      Button NAND: 0
Left LED:    OFF      Right LED:    OFF
```

For now, no other output is possible.

4.5 Slider Switches

In this section, you will install the "slider" switches that toggle between their two positions, holding their position until toggled again. We will wire them such that when a switch is toggled to the left, it will produce a 0, and when it is toggled to the right, it will produce a 1. Figure 21 shows a diagram of the wiring for the slider switches.

Before proceeding further, disconnect the USB cable from the Arduino Nano.

If you are using a breadboard template then use a jumper wire's lead to pre-punch holes into contact points a27–a29 and a32–a34.

Insert one slider switch into contact points a27–a29. Place the other slider switch into contact points a32–a34.

For the two wires that will connect the switches to the Arduino Nano, you can use 10cm jumpers (especially if that is all that you have); however, if you use 20cm jumpers, then in Section 5 we will show how to keep some wires away from the controls. Peel off one wire from the male-to-male rainbow cable and use it to connect contact point e27 (electrically

Switch	Arduino Nano pin	Pulled High/Low
Left switch's left pin	`D11`	
Left switch's center pin		Pulled Low
Left switch's right pin	not connected / floating	
Right switch's left pin	`D10`	
Right switch's center pin		Pulled Low
Right switch's right pin	not connected / floating	

Table 3: Electrical Connections for 3-pin Slider Switches.

connected to the left switch's left pin) to contact point j2 (electrically connected to the Arduino Nano's `D11` pin). Peel off another wire from the male-to-male rainbow cable and use it to connect contact point e32 (electrically connected to the right switch's left pin) to contact point j3 (electrically connected to the Arduino Nano's `D10` pin).

Peel off two more wires from the male-to-male rainbow cable. You will use these to connect the switches center pins to the upper ground (–) rail. Specifically, place the end of one wire into contact point e28; place the other end of that wire into the upper ground (–) rail. Now place the end of the other wire into contact point e33; place the other end of that wire into the upper ground (–) rail. The switches' right pins will not be electrically connected to anything.

When you have finished setting up the switches' wiring, there should be the electrical connections described in Table 3.

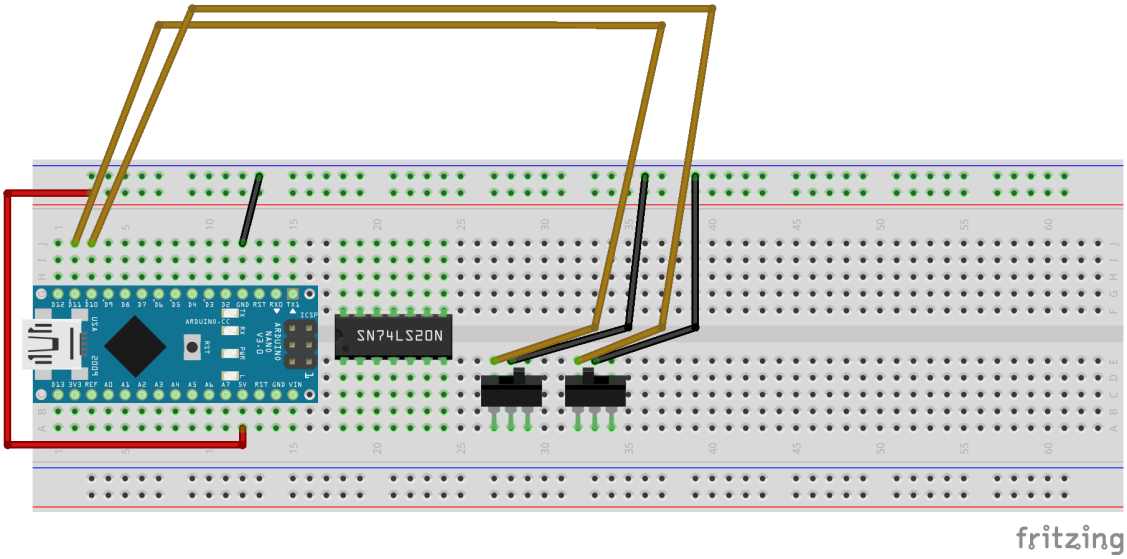
CheckPoint 6: Before proceeding further, have a TA or a classmate verify that you have correctly inserted and wired the slider switches. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

Connect your Arduino Nano to the computer. In the IDE's Serial Monitor, notice that Left switch is LEFT when the left switch is toggled to the left, and it is RIGHT when the left switch is toggled to the right. Similarly, Right switch is LEFT or RIGHT, depending on whether the right switch is toggled to the left or right.

4.6 Momentary Pushbuttons

This paragraph applies only to 2-lead pushbuttons:

> If your momentary pushbuttons are attached to a cardboard strip with tape, remove them from the cardboard strip. If your momentary pushbuttons' leads have metal tabs at the end (Figure 23), you will need to snip off the tabs before inserting the pushbutton leads into the breadboard; ordinary scissors will suffice



(a) 3-pin switches

Figure 21: Diagram of wiring associated with toggle switch input.

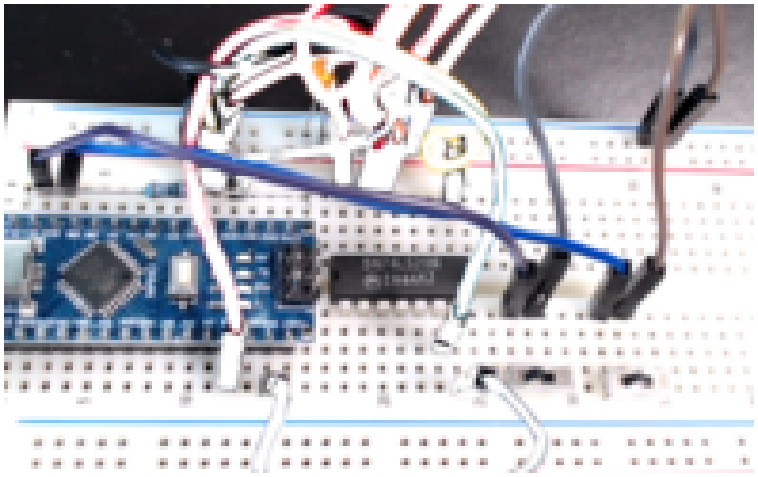


Figure 22: The slider switches, each with one pin grounded, one pin connected to the Arduino Nano, and one pin floating.

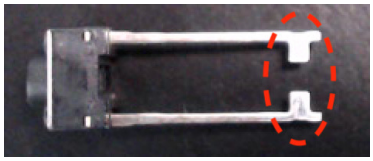


Figure 23: Some momentary pushbuttons have metal tabs on their leads.

for this task. Regardless of whether the leads have metal tabs at the end, you may optionally trim the leads to be about $\frac{1}{4}$in (6.4mm) long – you can use the exposed lead from a jumper wire as a reference – so that the pushbuttons sit flush on the breadboard. It is not necessary that they sit flush; this is simply to keep the buttons from wiggling under your fingers. *Do not cut the leads shorter than $\frac{1}{8}$in (3.2mm)!* **Be sure to use eye protection in case the leads' ends fly off when you snip them.**

This paragraph applies only to 4-prong pushbuttons:

The four prongs on a momentary pushbutton are electrically connected as two pairs. If you attach the pushbuttons in the wrong orientation, it will appear to the Arduino Nanoas though they are always pressed. Fortunately, there is only one orientation that will place the prongs in the specified contact points. As long as the prongs are in the specified contact points, your pushbuttons will work fine.

These are "normally open" momentary "switches" that close when pressed and re-open when released. We will wire the pushbuttons such that they normally produce a 1, and when pressed will produce a 0. Figure 24 shows a diagram of the wiring for the pushbuttons.

Before proceeding further, disconnect the USB cable from the Arduino Nano.

If you are using a breadboard template then use a jumper wire's lead to pre-punch holes into contact points a37, a39, a41, and a43. If you have 4-prong pushbuttons, then also pre-punch holes into contact points d37, d39, d41, and d43.

If you have 2-lead pushbuttons:

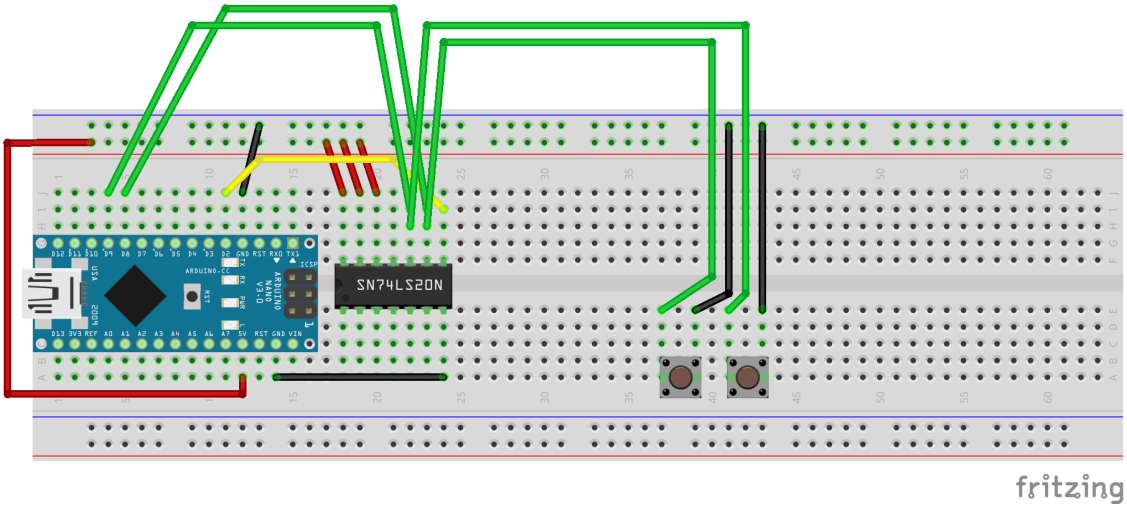
Insert the leads of one pushbutton into contact points a37 and a39. Insert the leads of the other pushbutton into contact points a41 and a43.

If you have 4-prong pushbuttons:

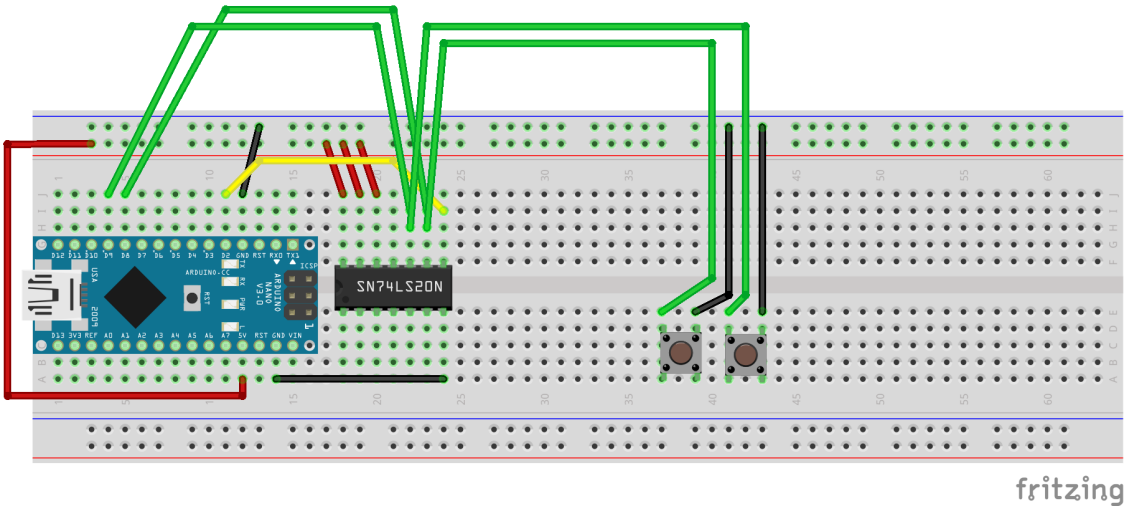
Insert the prongs of one pushbutton into contact points a37, d37, a39 and d39. Insert the prongs of the other pushbutton into contact points a41, d41, a43 and d43.

Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point e39 to the upper ground (–) rail. Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point e43 to the upper ground (–) rail.

Use two wires from the male-to-male rainbow cableto connect the ungrounded side of the left pushbutton to the 74LS20 and to the Arduino Nano: connect e37 to i22, and connect

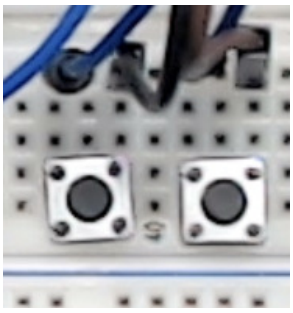


(a) 2-lead pushbuttons

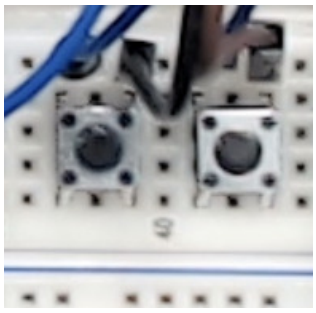


(b) 4-prong pushbuttons

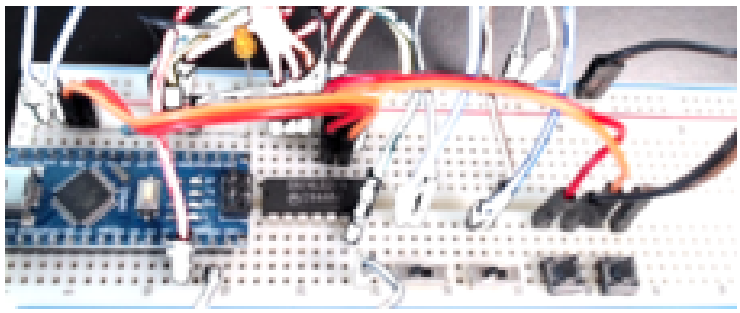
Figure 24: Diagram of wiring associated with momentary pushbutton input. *Note: connection between the 74LS20's pin 8 and the Arduino Nano's* `D2` *pin was previously installed in Section 4.3.*



(a) Pushbuttons with two leads.



(b) Pushbuttons with four prongs.



(c) The momentary pushbuttons, wired to the 74LS20 and the Arduino Nano.

Figure 25: Wiring the Momentary Pushbuttons.

h22 to j5. Now use another two wires from the male-to-male rainbow cableto connect the ungrounded side of the right pushbutton to the 74LS20 and to the Arduino Nano: connect e41 to i23, and connect h23 to j4. *Notice that there are no wires in g21–j21* because, as you can see in Figure 15b, the 74LS20's pin 11 is not connected ("NC") to anything.

See Figure 25.

When you have finished setting up the pushbuttons' wiring, there should be the electrical paths described in Table 4.

CheckPoint 7: Before proceeding further, have a TA or a classmate verify that you have correctly inserted and wired the momentary pushbuttons. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

Connect your Arduino Nano to the computer. In the IDE's Serial Monitor, notice that Left button is normally UP, but it becomes DOWN when you press the left button. Similarly, Right button is normally UP, but it becomes DOWN when you press the right button. Notice also that Button NAND is normally 0, but it becomes 1 whenever you press either button

Pushbutton	74LS20	Arduino Nano pin	Pulled High/Low
Left button's grounded lead			Pulled Low
Left button's ungrounded lead	Upper NAND Input	D8	
Right button's grounded lead			Pulled Low
Right button's ungrounded lead	Upper NAND Input	D9	
	Upper NAND Input		Pulled High
	Upper NAND Input		Pulled High
	Upper NAND Output	D2	
	pin 11 (g21–j21)	not connected / floating	

Table 4: Electrical Paths for Momentary Pushbuttons.

(or both).

Notice that both Left LED and Right LED are normally OFF. Position both switches to the right. Notice than when (and only when) the left button is to the right and you're also pressing the left button, Left LED becomes ON, and the LED labelled "L" on the Arduino Nanoilluminates. Similarly, when (and only when) the right button is to the right and you're also pressing the right button, Right LED becomes ON, and the LED that you installed illuminates. (There may be a delay of about a half-second between you pressing a button and the LED illuminating, and between you releasing a button and the LED deluminating. We will look at why this happens and how to prevent it in an upcoming lab.)

5 Matrix Keypad

Observe that the matrix keypad has sixteen buttons has eight pins in its female connector. As shown in Figure 26a, when the keypad is face-up and oriented for reading, the four pins on the left are the *row* pins, and the four pins on the right are the *column* pins. From left-to-right, we will name these pins `row1`, `row4`, `row7`, `row*`, `column1`, `column2`, `column3`, `columnA`. Figure 26b shows the membrane contacts and which Arduino Nano pin will be connected to each keypad pin.

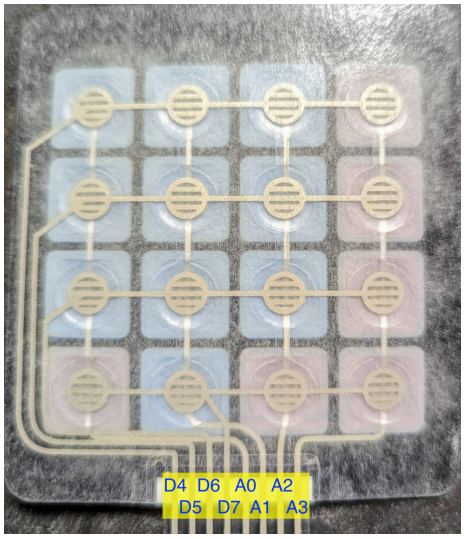
Figure 27 shows a diagram of the wiring for the matrix keypad.

Before proceeding further, disconnect the USB cable from the Arduino Nano.

If you are using a breadboard template then use a jumper wire's lead to pre-punch holes into contact points j26–j33. If your male-male header strip has more than eight pins, then pre-punch additional holes to the right (j34, j35, ...) as needed.

If your 8-pin male-male header strip is not already inserted into the keypad's female connectors, insert it into the female connectors now. If your male-male header strip has more than eight pins, position the excess pins to the right of the column pins. Connect your keypad to your breadboard such that `row1` is in contact point j26, and `columnA` is in contact point j33 (and any unused pins on the male-male header are in contact points j34, j35, etc.). **NOTE**: if you used 20cm wires to connect your slide-switches and/or pushbuttons to the

(a) Front of matrix keypad.



(b) Keypad's underlying contact matrix.

Figure 26: The numeric keypad's header has four row pins and four column pins.

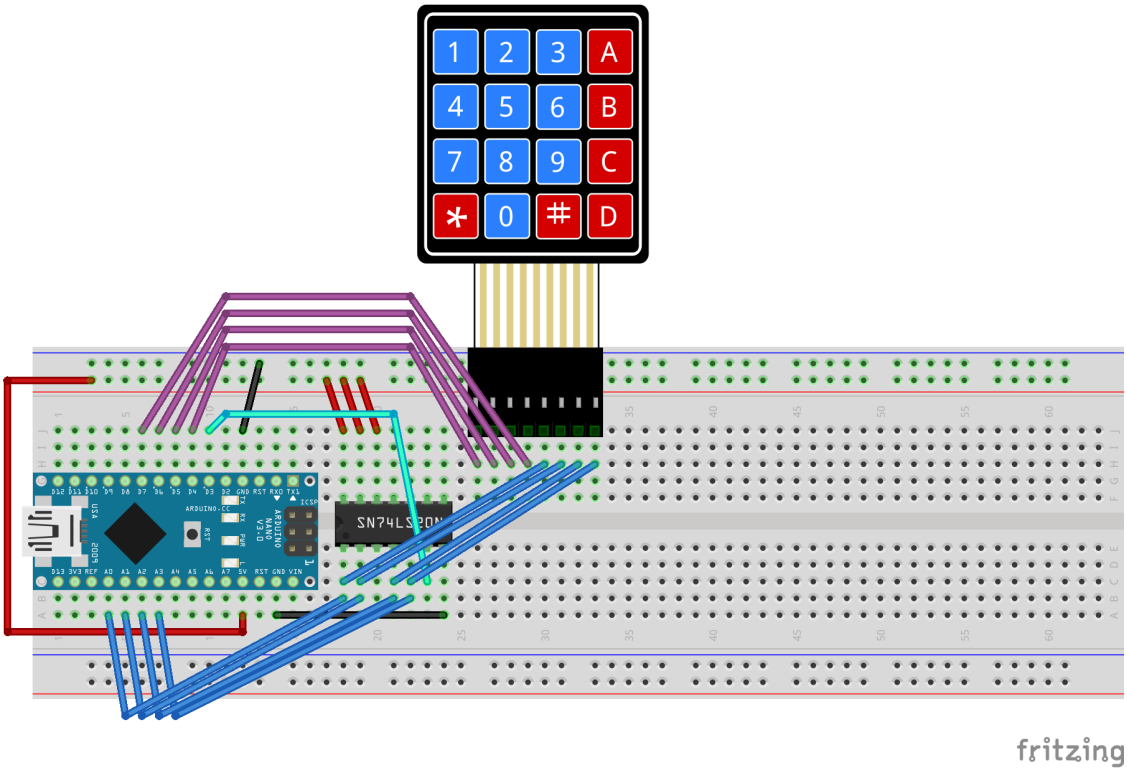


Figure 27: Diagram of wiring associated with matrix keyboard input. *Note: connection between the 74LS20's pin 6 and the Arduino Nano's* `D3` *pin was previously installed in Section 4.3.*

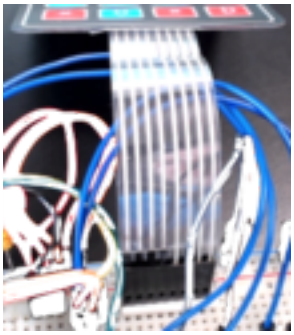


Figure 28: The keypad's ribbon cable can be used to pull long wires out of the way.

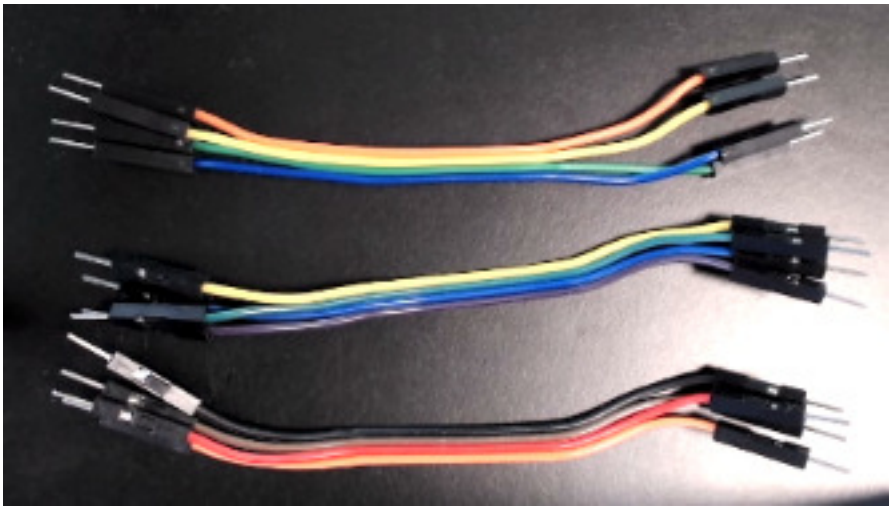


Figure 29: Three 4-conductor cables.

Arduino Nano, then you can use the matrix keypad's ribbon cable to pull these wires away from the circuit, reducing clutter near the controls (Figure 28).

Peel off three 4-conductor cables from the male-to-male rainbow cable(Figure 29). While you *can* use individual wires, having these 4-conductor cables will simplify keeping track of the wires.

Insert one end of one of the 4-conductor cables in contact points h26–h29, in the same breadboard rows as the keypad's row pins. Insert the other end of the cable in contact points j9–j6. You want the Arduino Nano's `D4` pin to connect to the keypad's `row1` pin, `D5` to `row4`, `D6` to `row7`, and `D7` to `row*`; you can use the wires' colors to make sure that you do so.

Insert one end of another 4-conductor cable in contact points h30–h33, in the same breadboard rows as the keypad's column pins. Insert the other end of the cable in contact points c18, c19, c21, and c22 (electrically connected to the 74LS20's `A1`, `B1`, `C1`, and `D1`: pins 1, 2, 4, and 5). *Notice that there are no wires in a20–d20* because, as you can see in Figure 15b, the 74LS20's pin 3 is not connected ("NC") to anything.

Insert one end of the remaining 4-conductor cable in contact points b18, b19, b21, and

Keypad pin	74LS20	Arduino Nano pin
`row1`		`D4`
`row4`		`D5`
`row7`		`D6`
`row*`		`D7`
`column1`	Lower NAND Input	`D14/A0`
`column2`	Lower NAND Input	`D15/A1`
`column3`	Lower NAND Input	`D16/A2`
`columnA`	Lower NAND Input	`D17/A3`
	Lower NAND Output	`D3`
	pin 3 (a20–d20)	not connected / floating

Table 5: Electrical Paths for Matrix Keypad.

b22. Insert the other end in contact points a4–a7 (electrically connected to the Arduino Nano's `D14/A0`–`D17/A3` pins). You want the 74LS20's pin 1 to connect the Arduino Nano's `D14/A0` pin and the keypad's `column1` pin, the 74LS20's pin 2 to connect `D15/A1` and `column2`, the 74LS20's pin 4 to connect `D16/A2` and `column3`, and the 74LS20's pin 5 to connect `D17/A3` and `columnA`; you can use the wires' colors to make sure that you do so.

When you have finished setting up the keypad wiring, there should be the electrical paths described in Table 5.

CheckPoint 8: Before proceeding further, have a TA or a classmate verify that you have correctly inserted and wired the matrix keypad. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

NOTE: Do not press more than one key on the matrix keypad at a time. There are certain combinations of keys that could result in a short-circuit from power to ground, possibly damaging your Arduino Nano. Your Arduino Nano has some safety measures to prevent damage in that situation, but it would be better for you not to test those safety measures.

Connect your Arduino Nano to the computer. In the IDE's Serial Monitor, notice that there is normally no character after `Keypad:`, that Column pins is normally 1111, and that Keypad NAND is normally 0. Press the 5 key on the matrix keypad. Notice that the first line of the message from the Arduino Nanois now

```
Keypad:      5        Column pins:  1011    Keypad NAND: 1
```

In general, when you press a key on the keypad, the corresponding character will be displayed after `Keypad:`, and Keypad NAND will become 1. When you press 1, 4, 7, or *, Column pins

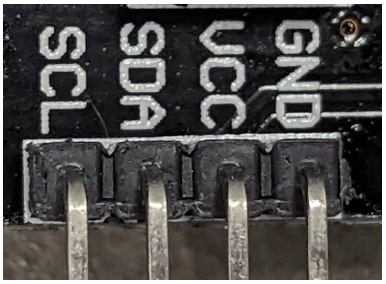


Figure 30: The display module's header has four pins.

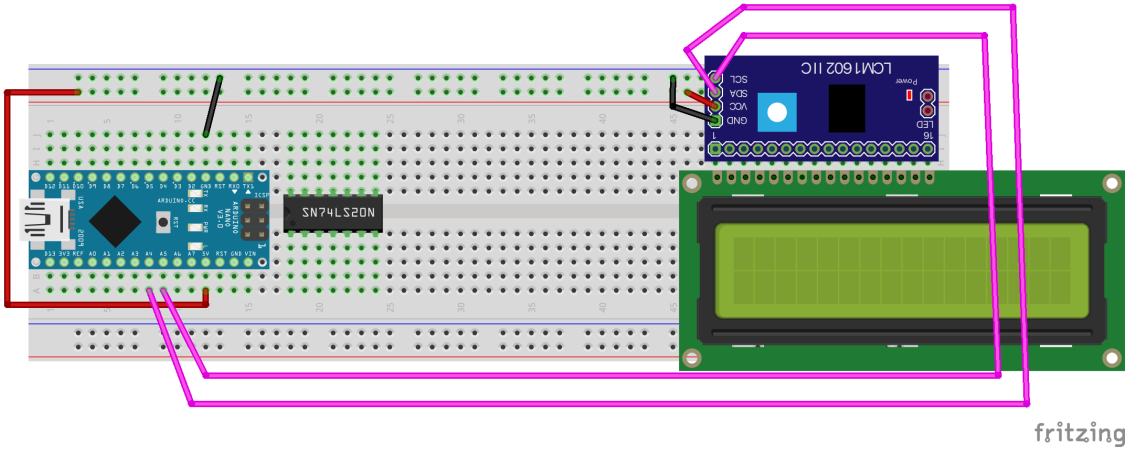


Figure 31: Diagram of display module's connections to the breadboard.

becomes 0111; similarly, pressing a key in the 2^{nd} column causes Column pins to become 1011; in the 3^{rd} column, 1101; and in the Ath column, 1110. Be sure to test all 16 keys.

6 Display Module

Examine the I^{2}C-LCD serial interface. Notice that the header has fourpins (Figure 30): `VCC` (common collector voltage), `GND` (ground), `SDA` (serial data), and `SCL` (serial clock). When the display module is oriented for viewing, these header pins will be on the left.

Figure 31 shows a diagram of the wiring to connect the display module to the breadboard.

Before proceeding further, disconnect the USB cable from the Arduino Nano.

If your I^{2}C-LCD serial interface is *not* attached to the LCD display module, then you will use the breadboard to provide the electrical connections between the serial interface and the display module. *If you are using a breadboard template* then use a jumper wire's lead to pre-punch holes into contact points i48–i63 and g48–g63. If you are using a breadboard template then you can now remove the jumper wires from contact points a63 and j63. Insert the LCD display module's sixteen pins into contact points g48–g63. With the four header pins pointing to the left, insert the I^{2}C-LCD serial interface's sixteen downward-pointing pins into contact points i48–i63.

Display Module Pin	Arduino Nano pin	Pulled High/Low
SCL	D19/A5	
SDA	D18/A4	
GND		Pulled Low
VCC		Pulled High

Table 6: Electrical Connections for External LED.

If your I²C-LCD serial interface *is* attached to the LCD display module, then the sixteen pins connecting the serial adapter to the display module do not need to be inserted into the breadboard. *If you are using a breadboard template* then you can now remove the jumper wires from contact points a63 and j63, but you do not need to do so (you might use a jumper wire looped from a63 to j63 to prevent the display module from sliding around).

Take the 4-conductor female-to-male rainbow cable and attach the four female connectors to the display module's four header pins.

Identify the wire that is connected to the display module's SCL pin; insert the male end of this wire in contact point a9 (electrically connected to the Arduino Nano's D19/A5pin). Insert the male end of the SDA wire into contact point a8 (electrically connected to the Arduino Nano's D18/A4pin). Insert the GND wire into the upper ground (−) rail, and the VCC wire into the upper power (+) rail.

When you have finished connecting the display module, there should be the electrical connections described in Table 6.

CheckPoint 9: Before proceeding further, have a TA or a classmate verify that you have correctly connected the display module to the breadboard. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

In the Arduino IDE, open the *File* → *Examples* → *CowPi* → *lcd1602_hello_world* example. Find these lines in the **setup()** function:

```
9  //    protocol = SPI;
10     protocol = I2C;
```

Make sure that the protocol = I2C line is uncommented and that the protocol = SPI line is commented-out.

Compile the program and upload it to your Arduino Nano.

You should see the display module's backlight blink on and off. If so, then you have correctly connected the display module and serial adapter even if you don't see a message on the display module.

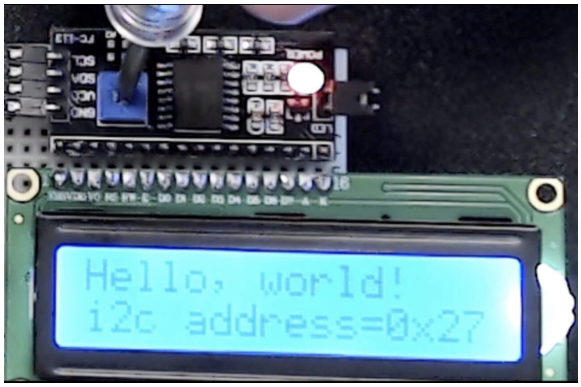


Figure 32: *lcd1602_hello_world.ino* blinks the backlight and displays a "Hello World" message.

Using a screwdriver, turn the trim potentiometer on the serial adapter until you can see the message:
`Hello, world!`
`i2c address=0x27`
(See figure 32.)

Kit Assembly is Complete

You have now finished assembling the class kit (Figure 33). In the upcoming I/O labs, you will use the kit to learn about memory-mapped I/O and about handling low-level interrupts.

7 Turn-in and Grading

When you have completed this assignment, upload *checkpoints.txt* to Canvas.

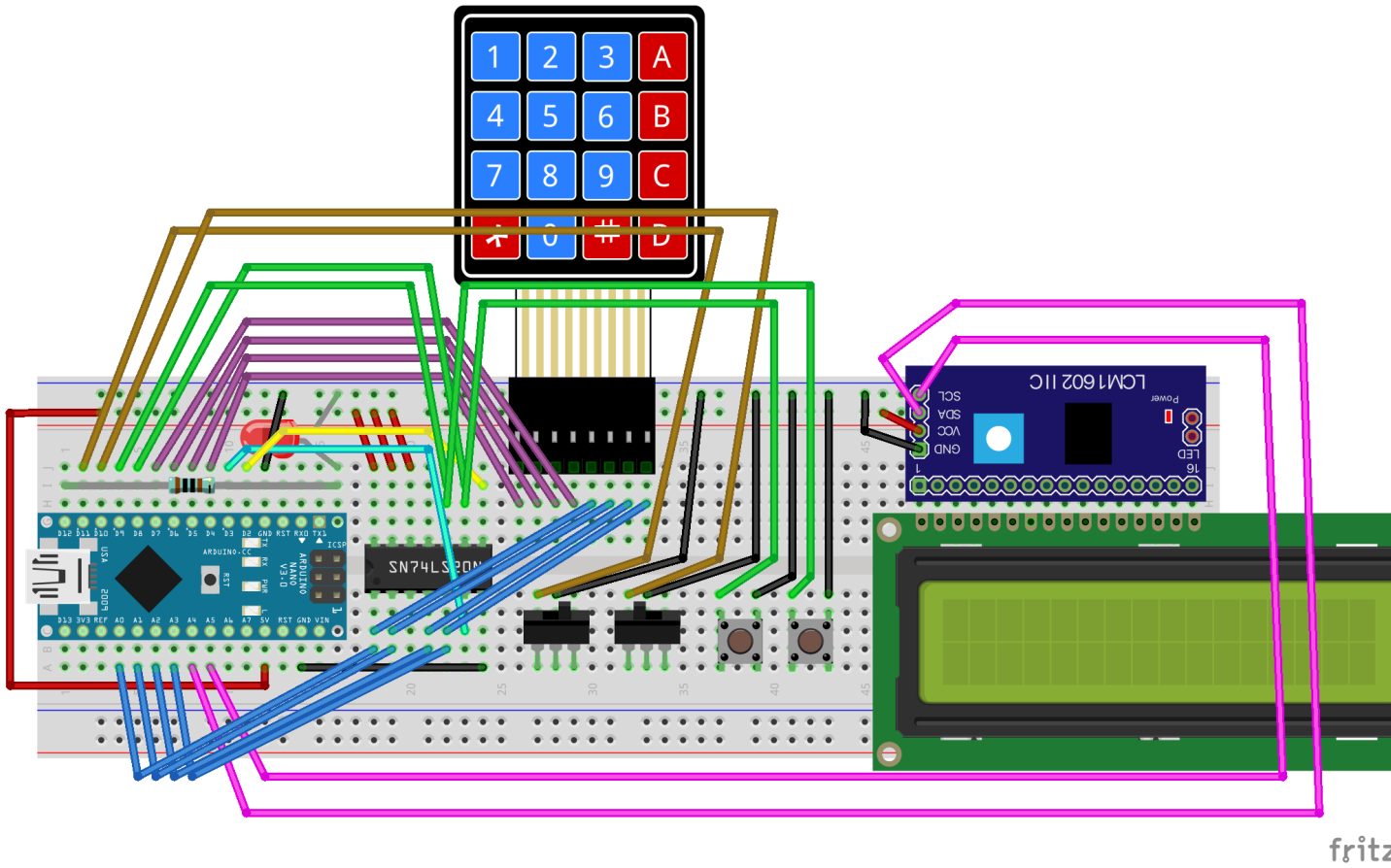
This assignment is worth 1 point, with up to 5 bonus points for helping other students.

Rubric:

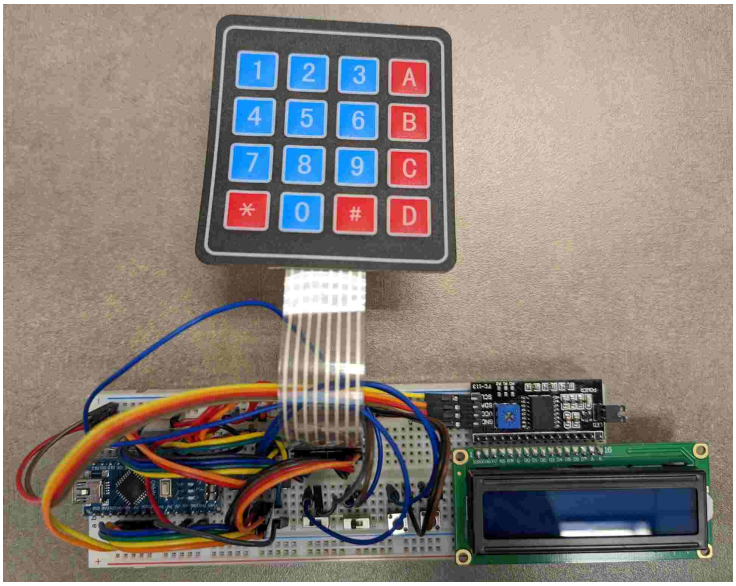
_____ **+0.5** You bring your fully-assembled class kit to your lab section the week it is due.

_____ **+0.5** You upload the completed *checkpoints.txt* file to Canvas.

_____ **Bonus +0.1** For each checkpoint you verify for fellow students, as reported in their *checkpoints.txt* files; maximum of 5.0 bonus points.



(a)



(b)

Figure 33: The fully-assembled class kit.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.