Lab 10-prelab

Physical Assembly of Hardware for I/O Labs

Due: Before the start of your lab section on April 5 or 6

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.

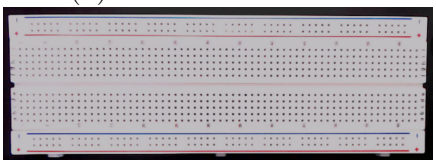
1 Obtaining the Hardware

The EE Shop has prepared "class kits" for CSCE 231; your class kit costs $20. 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. If you wish to pay by credit card, you must make the purchase from https://marketplace.unl.edu/ees/engineering-class-kits/csce231-kit.html the day before you visit the EE Shop.[1]

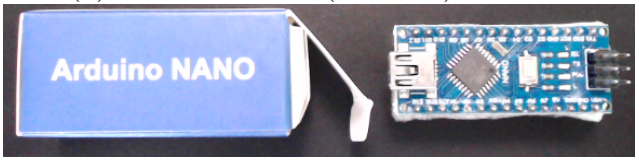
2 Inventorying the Hardware

Examine the contents of your class kit. It contains:

- One (1) full-sized solderless breadboard



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



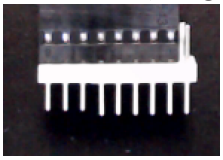
[1] The price listed on the website is $18.65; after sales tax is added, your total will be $20.

- One (1) Mini-USB cable

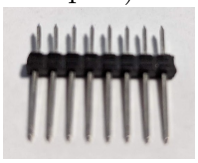
- One (1) 74LS20 dual 4-input NAND integrated circuit

- One (1) 4×4 matrix keypad

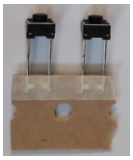
- One (1) 8-pin male-male header strip (might already be inserted into keypad's female connectors; might have more than 8 pins)



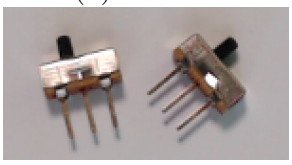
or



- Two (2) breadboard-mount momentary pushbuttons (might not be attached to cardboard strip)

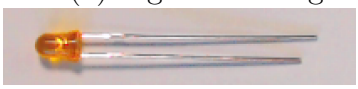


- Two (2) breadboard-mount switches



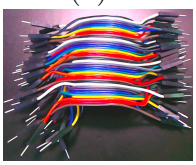
- One (1) 8-digit 7-segment display module

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

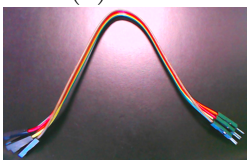


- One (1) 1kΩ resistor

- One (1) 40-conductor 10cm "rainbow" cable (male-to-male)



- One (1) 5-conductor 20cm "rainbow" cable (female-to-male)



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 both diagrams and 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 40-conductor male-to-male rainbow cable.*

3 Microcontroller

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

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

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.

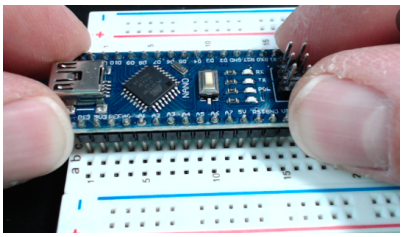
The Arduino Nano has a mini-USB port to connect to a computer and/or to provide power to the Arduino Nano. The six upward-pointing pins are used to program the Arduino Nano without using a host computer; we will not use these. It has thirty downward- pointing pins. `RX0` and `TX1` are used for asynchronous 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. `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-volt and 5-volt 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 the 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. There is no 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.

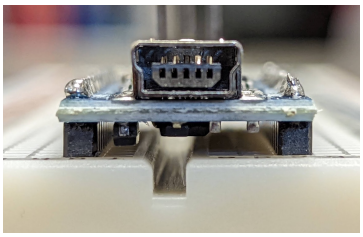
If you have already read the first half of Chapter 8, the ATmega328P has separate instruction and data memory, similar to the simple processor design described in the first half of Chapter 8. If you have already read the second half of Chapter 8, the ATmega328P has a 2-stage pipeline (with *Fetch* and *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] We will not use the ADCs in the I/O labs.

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



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



(b) The Arduino Nano fully inserted.

Figure 1: Inserting the Arduino Nano into the breadboard.

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 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.

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 1a). Press the Arduino Nano into the breadboard until it physically cannot be inserted any deeper (Figure 1b).

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.

3.3 Optional: Install Arduino IDE

The Arduino IDE is installed on the lab computers in 015 Avery, 021 Avery, and the Student Resource Center. 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.)

3.4 Connect to the Arduino Nano

Connect the USB-B 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

[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] Except for the Serial class that we'll use for debugging purposes, your code in the I/O labs will be C code.

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

[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.

with *Blink.ino* loaded, but these Arduino Nanos do not appear to have *Blink.ino* pre-loaded (this is inconsequential).

```
// 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 this Tutorial for selecting the Arduino Nano board, processor, and COM port (or this Tutorial for the Arduino Uno, which has more detail on selecting the COM port). If you are using Arduino IDE 2.0 or the Arduino Web Editor, COM port should be automatically detected; however, you will still need to select the board and processor; see Figure 2 and the discussion in Section 3.4.1.

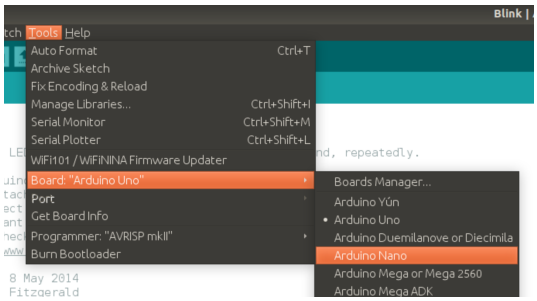
3.4.1 Selecting the Correct "Processor"

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:

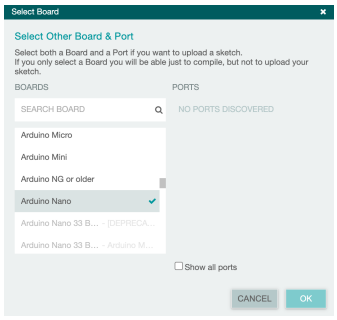
- Official Arduino Nanos manufactured in 2018 and later use the FT232RL USB interface chip. Under the "Tools" menu, when choosing "Processor", select "ATmega328P".
- Arduino Nanos manufactured in 2017 and earlier, and *many* Arduino Nano clones, use the CH340 USB interface chip. Under the "Tools" menu, 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 your Arduino Nano has the ATmega168 processor, replace it with one that has an ATmega328P processor. While the processor differences shouldn't be problem in the I/O labs, the older Arduino Nanos that use the ATmega168 have a slightly different pin-out that will affect constructing your class kit.

3.4.2 Updating USB Driver if Necessary

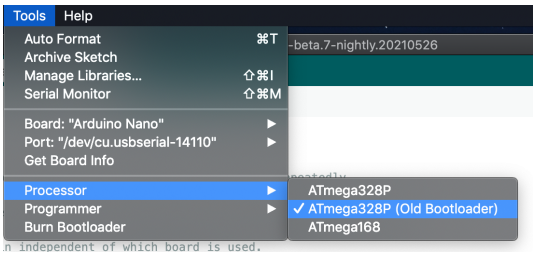
We have seen some Windows computers without the CH340 USB driver. If you encounter this problem and the Device Manager shows you the warning in Figure 3a, then the first



(a) Selecting the board with Arduino IDE 1.8.

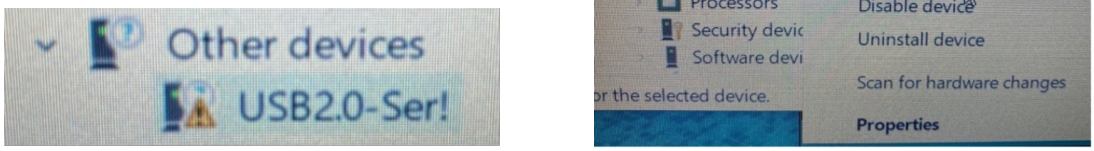


(b) Selecting the board with Arduino IDE 2.0.



(c) Selecting the processor after selecting the board.

Figure 2: Selecting board and processor in the Arduino IDE.



(a) (b)

Figure 3: Selecting board and processor in the Arduino IDE.

thing to try is updating the driver. Right-click on USB2.0-Seri! (Figure 3b) 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.

3.4.3 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. Tye 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.

3.5 Upload a New Sketch

From the Arduino IDE's 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/arduin
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 4).

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 2c).
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:

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

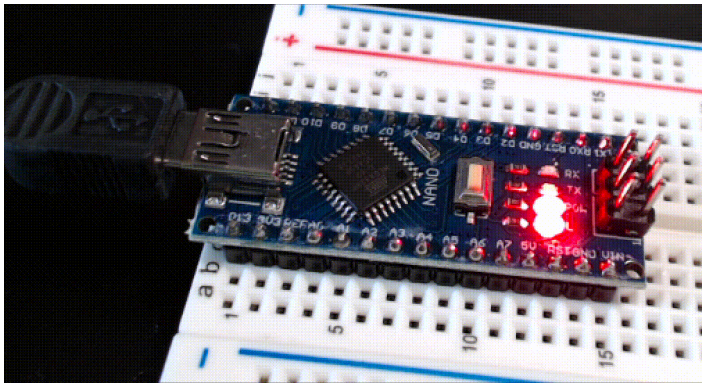


Figure 4: *MyBlink.ino* has a different on-off pattern than *Blink.ino*.

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 Output Devices

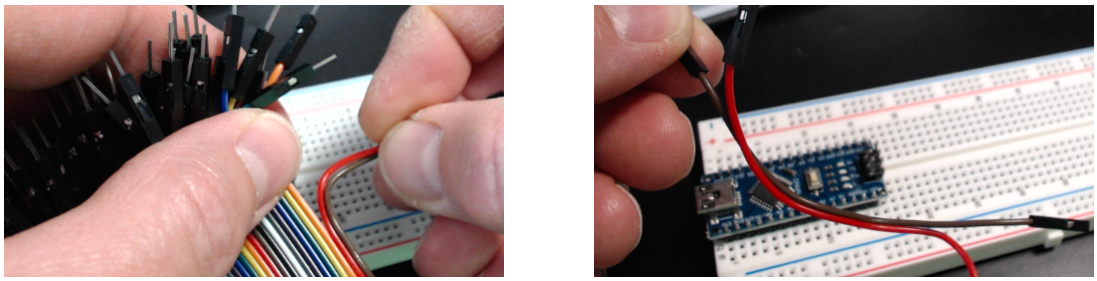
4.1 Connect Power and Ground to Power Bus Strips

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.

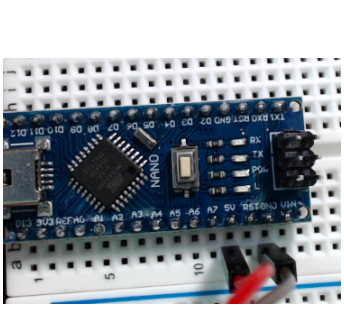
Take the 40-conductor male-to-male rainbow cable, and peel off two wires together. In the interest of keeping track of which wires are used for which purposes, it would be best to leave the two wires attached to each other as a 2-conductor cable (Figure 5). At one end of this 2-conductor cable, insert one lead 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) and the other into contact point A14 (notice that contact point A14 is electrically connected to one of the Arduino Nano's `GND` pins, which is in contact point c14); see Figure 6a. Insert the other end of the `5V` wire into the lower power (+) rail marked with a red stripe. Insert the other end of the `GND` wire into the lower ground (–) rail marked with a blue stripe; see Figure 6b.

Peel off another 2-wire cable from the male-to-male rainbow cable. Use this 2-conductor cable to connect the lower power (+) rail to the upper power (+) rail, and to connect the lower ground (–) rail to the upper ground (–) rail. If you place this cable on the right end of the breadboard, it will be out of the way during the remaining steps; see Figure 6c.

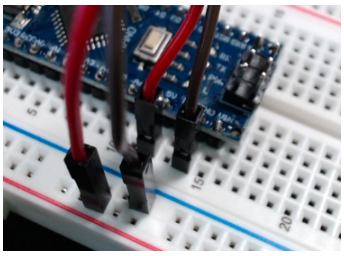


(a) (b)

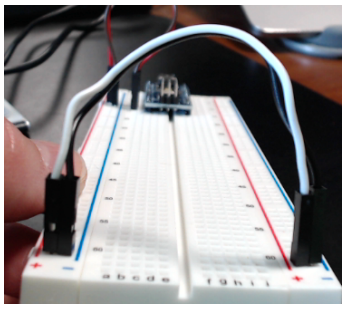
Figure 5: Removing a two-conductor cable from the 40-conductor cable.



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



(b) Connecting lower power bus to power and ground.



(c) Connecting upper and lower power busses.

Figure 6: Providing power and ground to power busses.

Figure 7: The LED's longer lead connects to power; the shorter lead connects to ground.

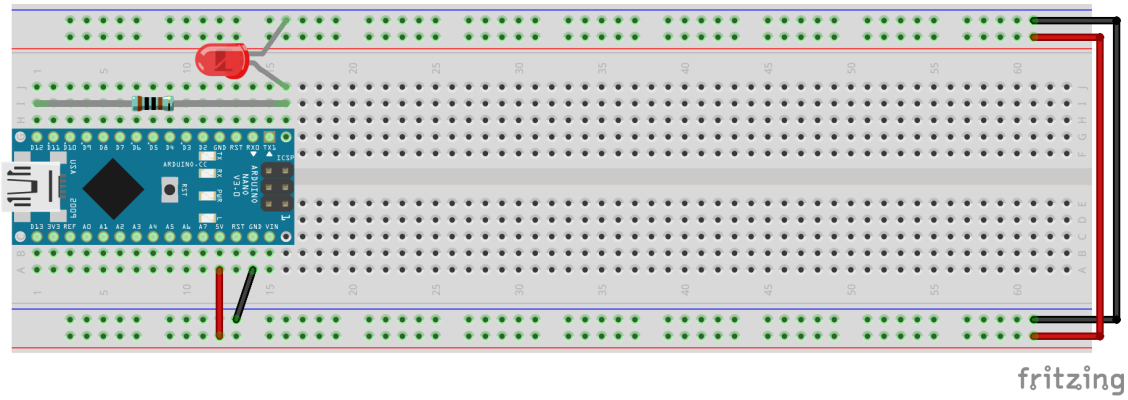


Figure 8: Diagram of component assembly for LED output.

CheckPoint 3: Before proceeding further, have a TA or a classmate verify that you have correctly connected the power (+) rails and the ground (–) rails. 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 7, 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 8 shows a diagram of 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 9.

[9]If you want to try to be exact, you can use the breadboard's contact points to measure: they are 0.1in (2.54mm) apart.

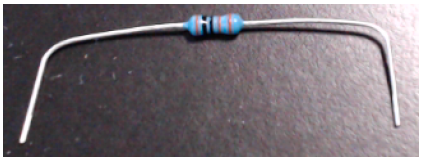
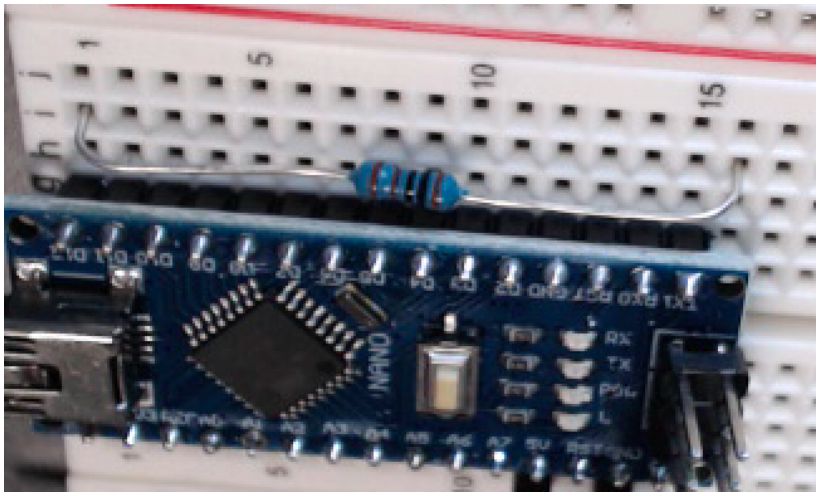
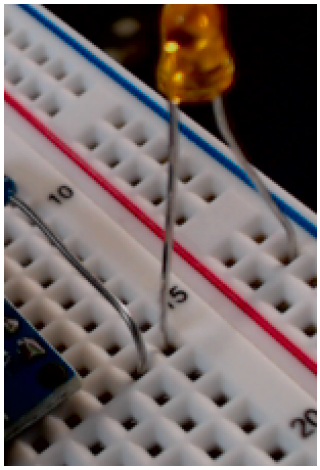


Figure 9: Bend the resistor's leads about 1cm from the ends.



(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 10: Constructing the LED assembly.

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 10a.

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 10b.

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.

In the Arduino IDE, load *MyBlink.ino*. In the **`pinMode`** and the two **`digitalWrite`**

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

Table 1: Electrical Connections for External LED.

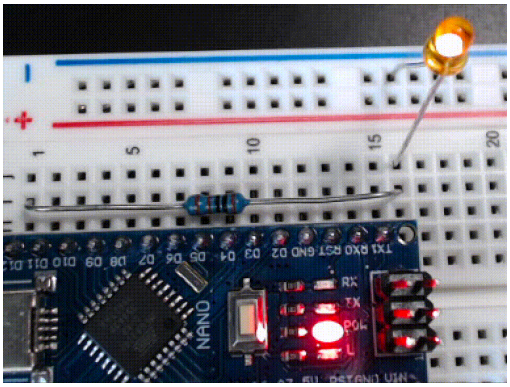


Figure 11: The revised *MyBlink.ino* causes the external LED to blink.

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 11).

4.3 Seven-Segment Display Module

Examine the 7-segment display module. Notice that the header has five pins (Figure 12): `VCC` (common collector voltage), `GND` (ground), `DIN` (data in), `CS` (chip select), and `CLK` (clock). When the display module is oriented for viewing, these header pins will be on the left.

Figure 13 shows a diagram of of the wiring to connect the display module to the bread-

Figure 12: The display module's header has five pins.

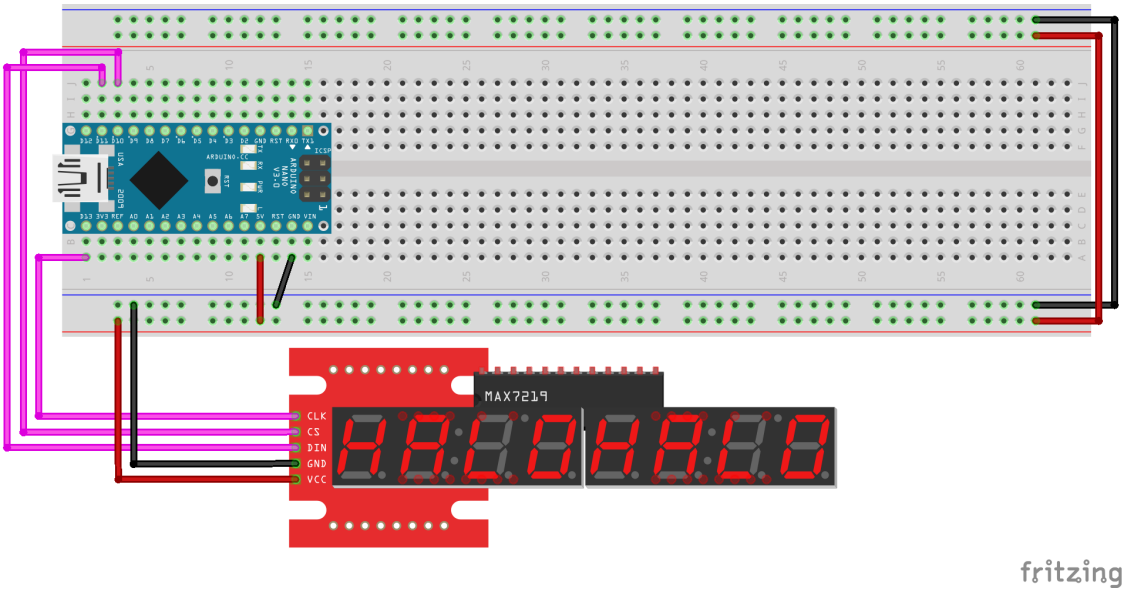


Figure 13: Diagram of display module's connections to the breadboard.

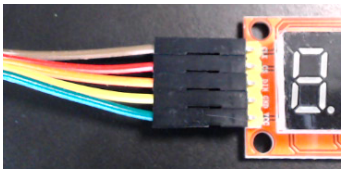
board.

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

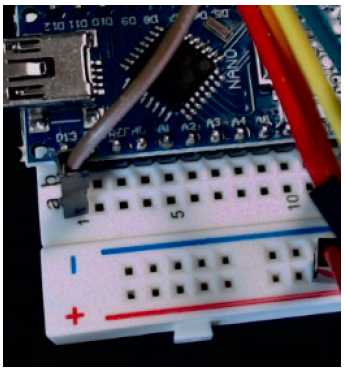
Take the 5-conductor female-to-male rainbow cable and attach the five female connectors to the display module's five header pins; see Figure 14a.

As you insert the male connectors into the breadboard, you may have to partially separate the wires at the male end. In the interest of keeping track of which wires are used for which purposes, do not fully separate the wires. Identify the wire that is connected to the display module's `CLK` pin; insert the male end of this wire in contact point a1 (electrically connected to the Arduino Nano's `D13` pin in c1); see Figure 14b. Insert the male end of the `CS` wire into contact point j3 (electrically connected to the Arduino Nano's `D10` pin in g3); see Figure 14c. Insert the male end of the `DIN` wire into contact point j2 (electrically connected to the Arduino Nano's `D11` pin in g2); see Figure 14d. Insert the `GND` wire into a ground (−) rail, and the `VCC` wire into a power (+) rail (Figure 14e).

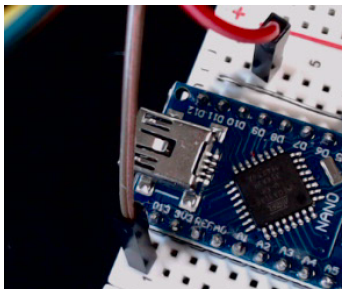
When you have finished connecting the display module, there should be the electrical connections described in Table 2.



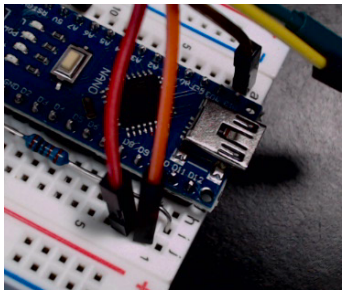
(a) Connect the female ends of the 5-conductor cable to the display module's header pins (your colors may be different).



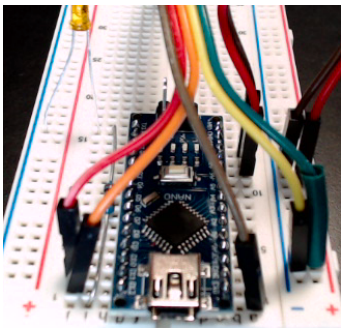
(b) The display module's clock will be driven by `D13`.



(c) The display module's chip-select will be driven by `D10`.



(d) The display module's data-in will be driven by `D11`



(e) The display module will be powered by the breadboard's power bus

Figure 14: Connecting the Display Module.

Display Module Pin	Arduino Nano pin	Pulled High/Low
`CLK`	`D13`	
`CS`	`D10`	
`DIN`	`D11`	
`GND`		Pulled Low
`VCC`		Pulled High

Table 2: Electrical Connections for External LED.

Figure 15: *DisplayTest.ino* illuminates all segments on a digit, one digit at a time.

CheckPoint 5: 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 *DisplayTest.ino*. Connect your Arduino Nano to the computer. Compile *DisplayTest.ino* and upload it to your Arduino Nano. You should see 8 move back and forth across the display (Figure 15). You may be able to see the Arduino Nano's built-in LED blink rapidly: recall that it's connected to pin 13, which is used as the clock signal when the Arduino Nano sends data to the display module.

5 Input Devices

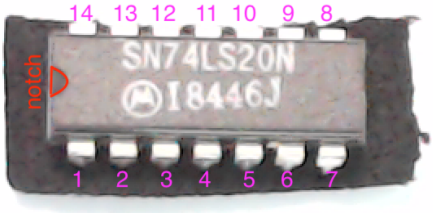
5.1 NAND Gates

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 16a). The relationship between the 74LS20's pins and the NAND gates' inputs and outputs is shown in Figure 16b.

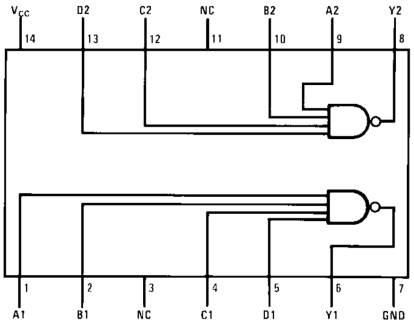
Figure 17 shows the wiring to install the 74LS20.

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

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 the its notch to the left, place the 74LS20 on the breadboard 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 18a shows that the IC's pins are not splayed outside the contact points nor are folded under the IC's



(a) Pin numbers for the 74LS20.



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

Figure 16: Pin information for the 74LS20.

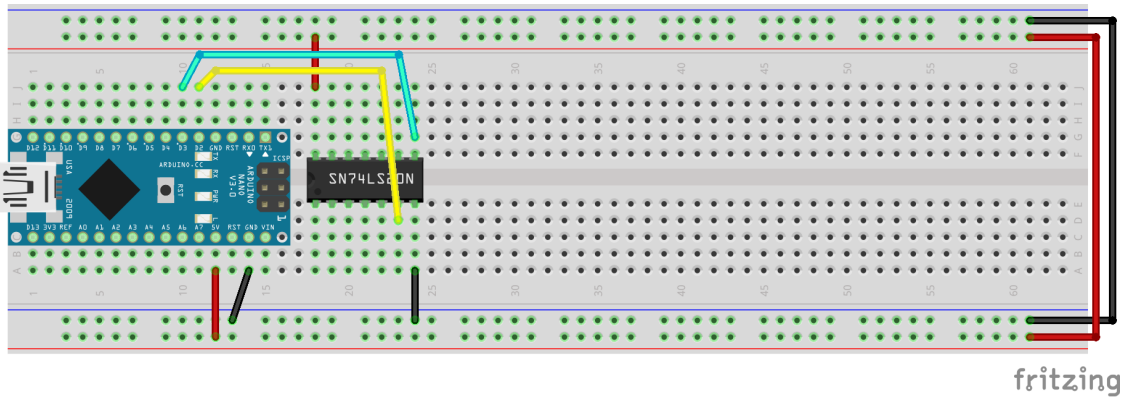
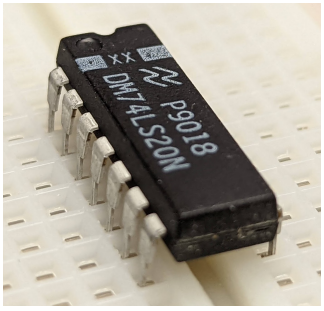
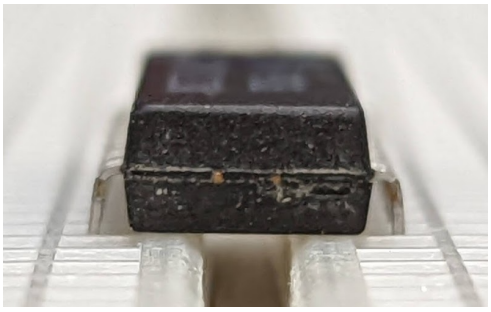


Figure 17: Diagram of the 74LS20's installation.

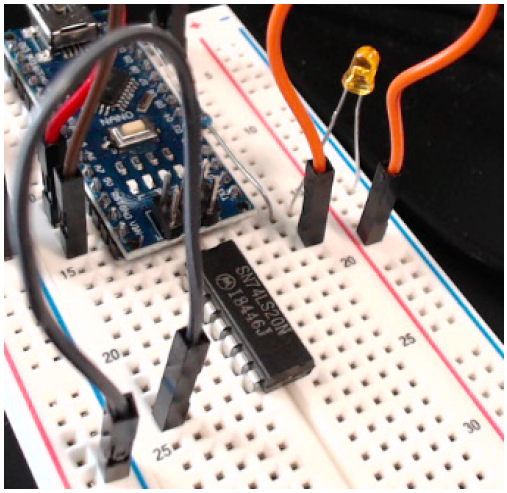


(a) Integrated circuit ready to be inserted.

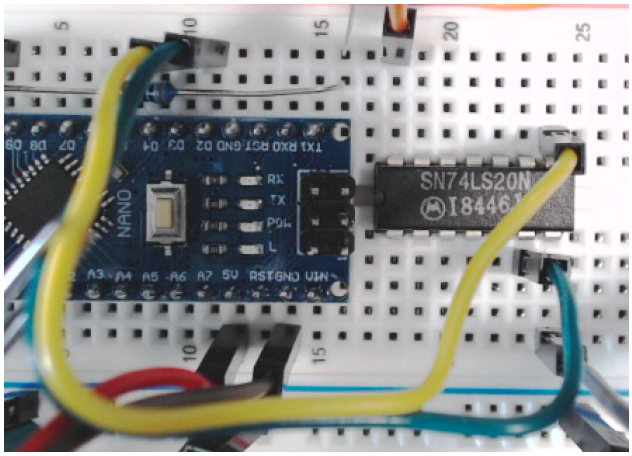


(b) Integrated circuit fully inserted.

Figure 18: Inserting the 74LS20.



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



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

Figure 19: Wiring the 74LS20.

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 18b, 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 a 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 a ground (–) rail. See Figure 19a.

Peel a two-wire cable from the male-to-male rainbow cable. Use this cable to connect contact point d23 (electrically connected to the 74LS20's `Y1`, pin 6) to contact point j11 (electrically connected to the Arduino Nano's `D2` pin), and to connect contact point g24 (electrically connected to the 74LS20's `Y2`, pin 8) to contact point j10 (electrically connected to the Arduino Nano's `D3` pin). See Figure 19b.

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

Table 3: Initial Electrical Connections for NAND Gates.

When you have finished wiring the 74LS20, 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 74LS20. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

5.2 Matrix Keypad

Observe that the matrix keypad has sixteen buttons has eight pins in its female connector. As shown in Figure 20, 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 20: The numeric keypad's header has four row pins and four column pins.

Figure 21 shows a diagram of of the wiring for the matrix keypad.

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.); see Figure 22a.

Peel off a 4-conductor cable from the male-to-male rainbow cable. Insert one end in contact points h26-h29, in the same breadboard rows as the keypad's row pins (Figure 22b). Insert the other end of the cable in contact points j9-j5 (Figure 22c). 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*` (Figure 22d).

Peel off another 4-conductor cable from the male-to-male rainbow cable. Insert one end 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 i19, i20, i22, and i23 (electrically connected to the 74LS20's `D2`, `C2`, `B2`, and `A2`: pins 13, 12, 10, and 9). See Figure 22e. Peel off another 4-conductor cable from the male-to-male rainbow cable. Insert one end in contact points h19, h20, h22, and h23. Insert the other end in contact points a4-a7 (electrically connected to the Arduino Nano's `A0-A3` pins). See Figure 22f.

When you have finished setting up the keypad 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 matrix keypad. Update *checkpoints.txt* file to indicate who checked your work and when they did so.

In the Arduino IDE, open *InputTest.ino*. Connect your Arduino Nano to the computer. Compile *InputTest.ino* and upload it to your Arduino Nano. Open the IDE's Serial Monitor

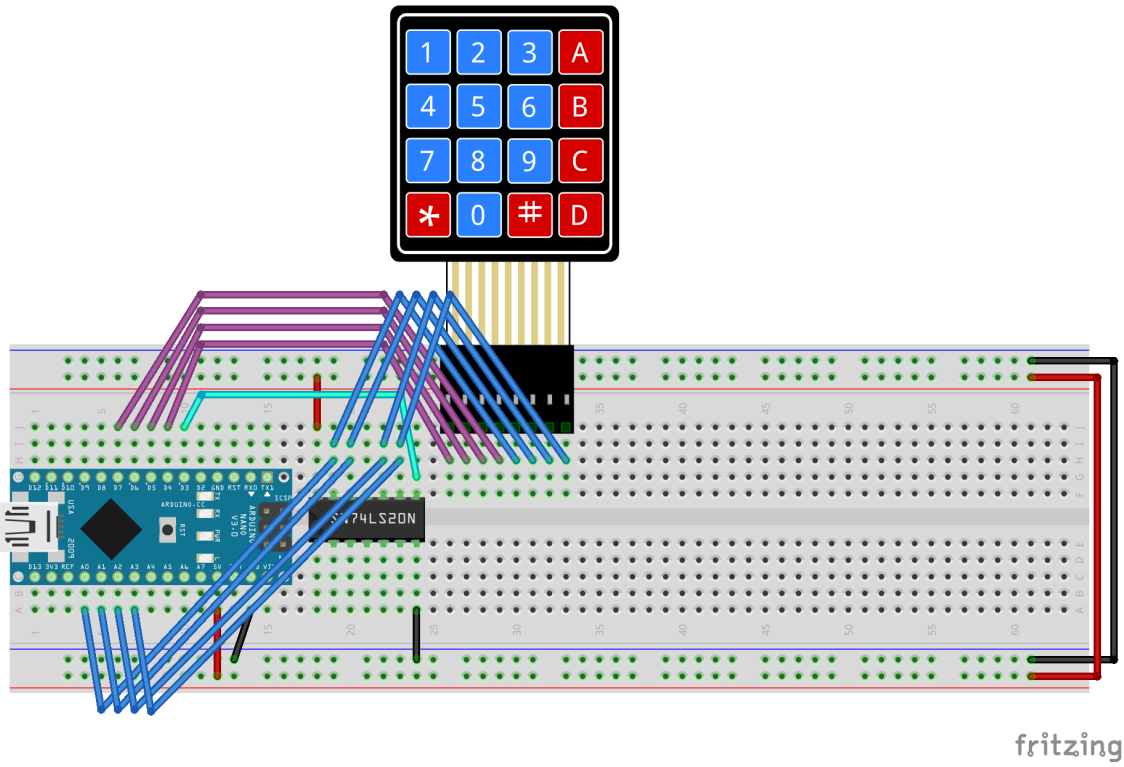
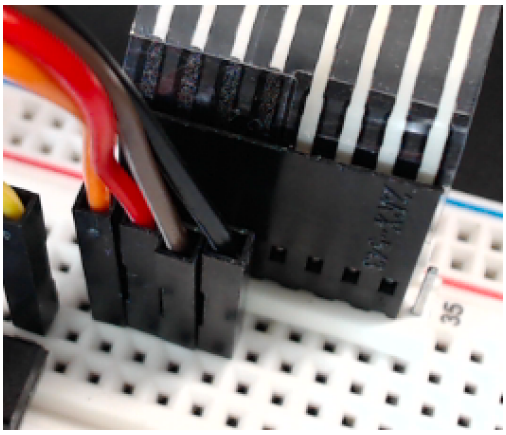


Figure 21: Diagram of wiring associated with matrix keyboard input. *Note: connection between the 74LS20's pin 8 and the Arduino Nano's* `D3` *pin was previously installed in Section 5.1.*

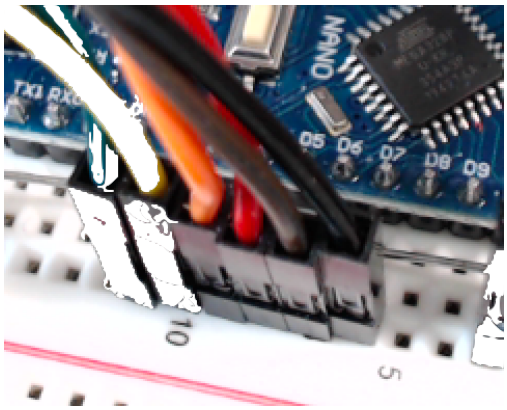
Keypad pin	74LS20	Arduino Nano pin
`row1`		`D4`
`row4`		`D5`
`row7`		`D6`
`row*`		`D7`
`column1`	Upper NAND Input	`A0`
`column2`	Upper NAND Input	`A1`
`column3`	Upper NAND Input	`A2`
`columnA`	Upper NAND Input	`A3`
	Upper NAND Output	`D3`

Table 4: Electrical Paths for Matrix Keypad.

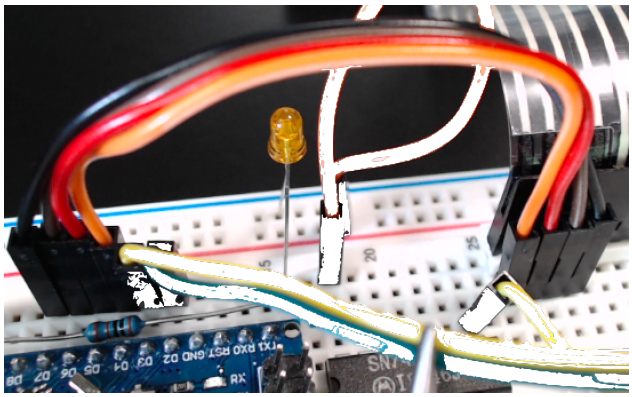
(a) The matrix keypad's connector, with the male-male header strip, just before being inserted into the breadboard. Note the excess header strip's excess pin to the right of the column pins, in contact point j34.



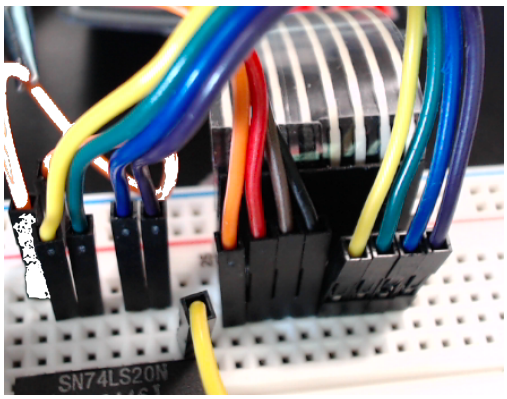
(b) One end of the "rows" cable electrically connected to the keypad's row pins.



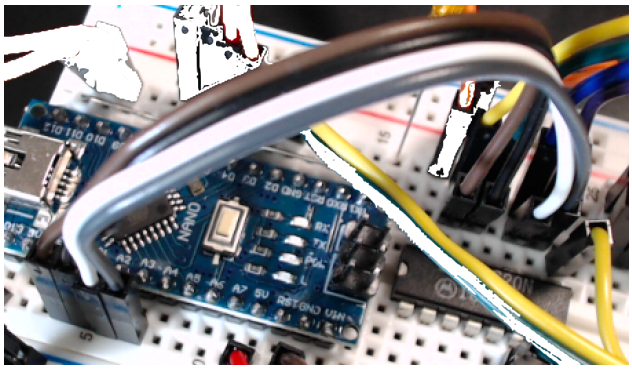
(c) The other end of the "rows" cable electrically connected to the Arduino Nano's `D4-D7` pins.



(d) Connection between the keypad's row pins and the Arduino Nano's `D4-D7` pins.



(e) Connection between the keypad's column pins and the 74LS20.



(f) Connection between the keypad's column pins and the 74LS20.

Figure 22: Wiring the Matrix Keypad.

Switch	Arduino Nano pin	Pulled High/Low
Left switch's left pin		Pulled Low
Left switch's center pin	`A4`	
Left switch's right pin	not connected / floating	
Right switch's left pin		Pulled Low
Right switch's center pin	`A5`	
Right switch's right pin	not connected / floating	

Table 5: Electrical Connections for Slider Switches.

(*Tools* → *Serial Monitor*). You will see several lines reporting the current value of various inputs. (You will also see the Arduino Nano's `TX` LED blinking as the Arduino Nano sends these lines to your computer.) Right now, KEYPAD COLUMNS is always 1111, and KEYPAD NAND is always 0. Notice that when you press 1, 4, 7, or *, KEYPAD COLUMNS becomes 0111; similarly, pressing a button in the 2^{nd} column causes KEYPAD COLUMNS to become 1011; in the 3^{rd} column, 1101; and in the Ath column, 1110. Whenever any keypad button is pressed, KEYPAD NAND becomes 1. Be sure to test all 16 keys.

5.3 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 23 shows a diagram of of the wiring for the slider switches.

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

Insert one slider switch into contact points a29-a31. Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point e29 to the upper ground (–) rail. Place the other slider switch into contact points a34-a36. Peel off another wire from the male-to-male rainbow cable, and use it to connect contact point e29 to the upper ground (–) rail.

Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point d30 (electrically connected to the left switch's center pin) to contact point a8 (electrically connected to the Arduino Nano's `A4` pin). Peel off another wire from the male-to-male rainbow cable, and use it to connect contact point d35 (electrically connected to the right switch's center pin) to contact point a9 (electrically connected to the Arduino Nano's `A5` pin). See Figure 24.

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 5.

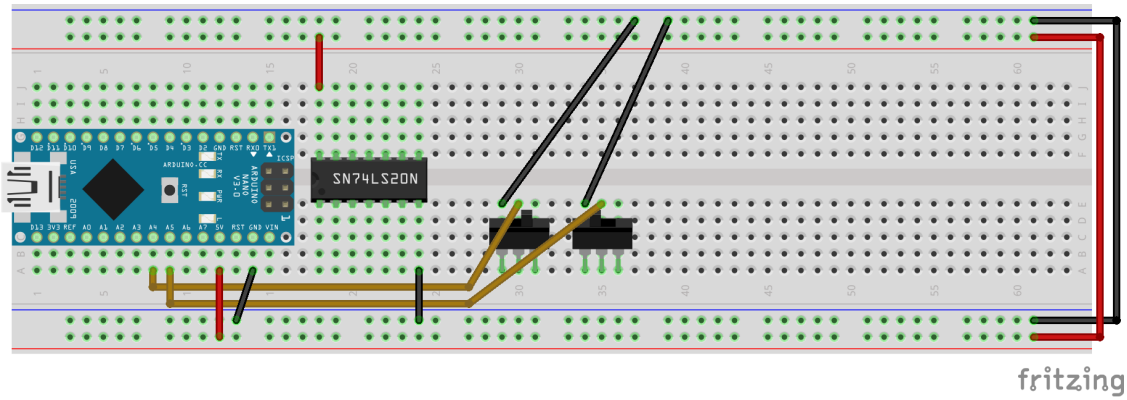


Figure 23: Diagram of wiring associated with toggle switch input.

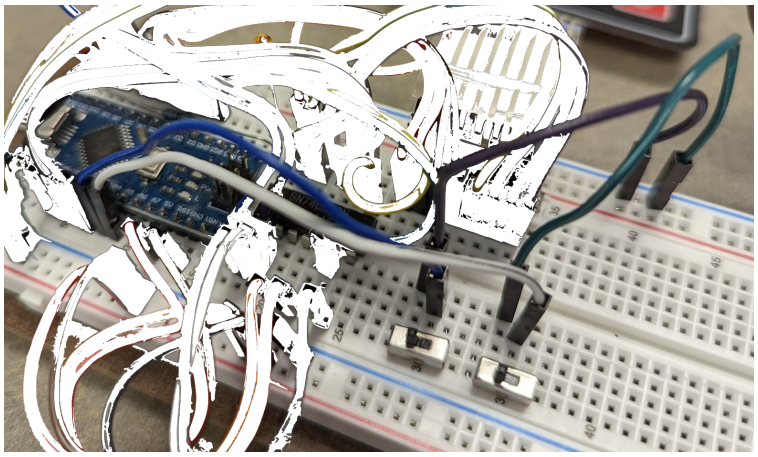


Figure 24: The slider switches, each with one pin grounded, one pin connected to the Arduino Nano, and one pin floating.

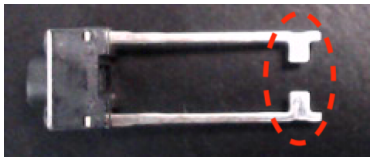


Figure 25: Some momentary pushbuttons have metal tabs on their leads.

CheckPoint 8: 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 the LEFT SWITCH is 0 when the left switch is toggled to the left, and it is 1 when the left switch is toggled to the right. Similarly, the RIGHT SWITCH's value is 0 or 1, depending on whether the right switch is toggled to the left or right.

5.4 Momentary 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 25), you will need to snip off the tabs before inserting the pushbutton leads into the breadboard; ordinary scissors will suffice 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.**

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 26 shows a diagram of of the wiring for the pushbuttons.

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

Insert the leads of one pushbutton into contact points a39 and a41. Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point e41 to the upper ground (–) rail. Insert the leads of the other pushbutton into contact points a43 and a45. Peel off one wire from the male-to-male rainbow cable, and use it to connect contact point e45 to the upper ground (–) rail. See Figure 27a.

Peel off a 2-conductor cable from the male-to-male rainbow cable, and use it to connect contact points a21 and a22 (electrically connected to the 74LS20's `C1` and `D1`, pins 4 and 5) to a power (+) rail. Peel off another 2-conductor cable and use it to connect contact points d39 and d43 (electrically connected to the ungrounded sides of the pushbuttons) to contact points b18 and b19 (electrically connected to the 74LS20's `A1` and `B1`, pins 1 and 2). See Figure 27b.

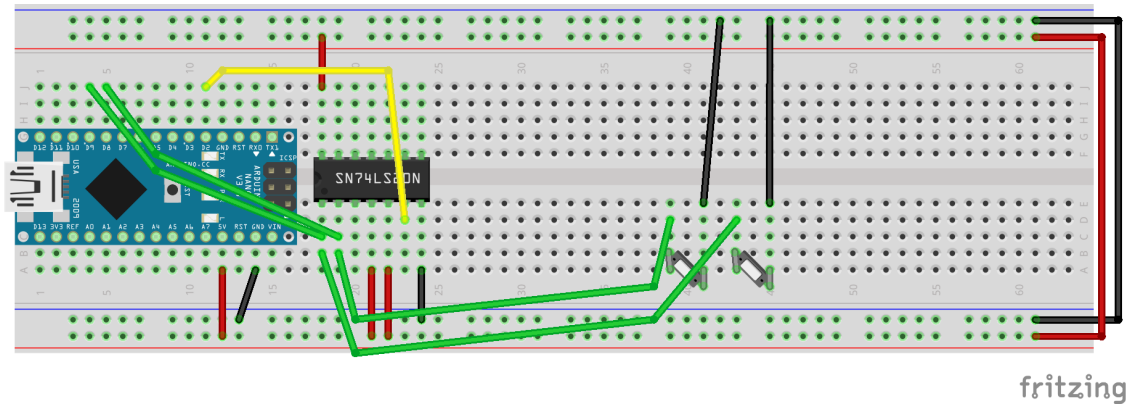


Figure 26: Diagram of wiring associated with momentary pushbutton input. *Note: connection between the 74LS20's pin 6 and the Arduino Nano's* `D2` *pin was previously installed in Section 5.1.*

Pushbutton	74LS20	Arduino Nano pin	Pulled High/Low
Left button's grounded lead			Pulled Low
Left button's ungrounded lead	Lower NAND Input	`D8`	
Right button's grounded lead			Pulled Low
Right button's ungrounded lead	Lower NAND Input	`D9`	
	Lower NAND Input		Pulled High
	Lower NAND Input		Pulled High
	Lower NAND Output	`D2`	

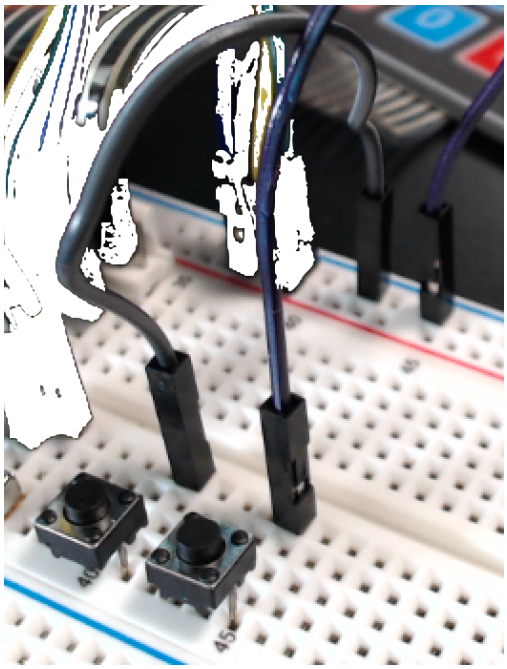
Table 6: Electrical Paths for Momentary Pushbuttons.

Peel off another 2-conductor cable from the male-to-male rainbow cable, and use it to connect contact points j4 and j5 (electrically connected to the Arduino Nano's `D8` and `D9` pins) to contact points c18 and c19 (electrically connected to the 74LS20's `A1` and `B1`, and to the pushbuttons through the cable you installed in the previous paragraph). See Figure 27c.

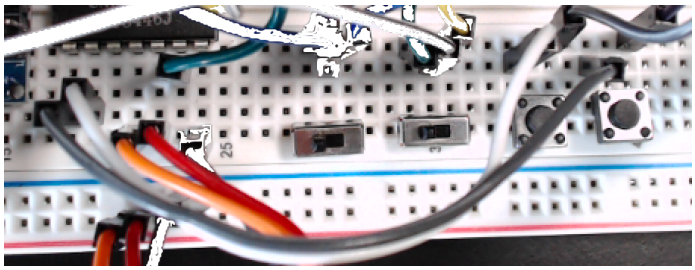
When you have finished setting up the pushbuttons' wiring, there should be the electrical paths described in Table 6.

CheckPoint 9: 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.

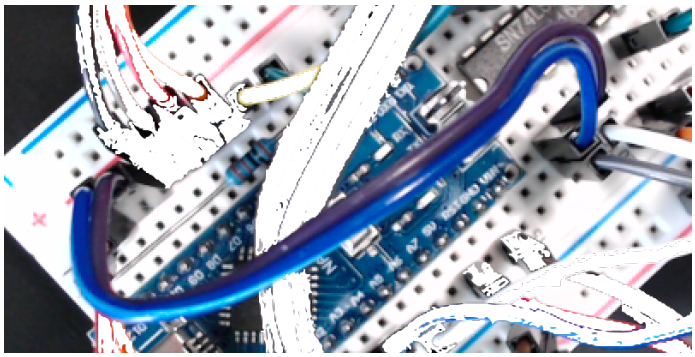
Connect your Arduino Nano to the computer. In the IDE's Serial Monitor, notice that the LEFT BUTTON is always 1, the RIGHT BUTTON is always 1, and the BUTTON NAND



(a) The momentary pushbuttons, each with one lead grounded.



(b) Connections between the momentary pushbuttons and the 74LS20.



(c) Connections between the Arduino Nano and wiring to the momentary pushbuttons.

Figure 27: Wiring the Momentary Pushbuttons.

is always 0. Notice that when you press a button, the Serial Monitor shows that its value becomes 0, and when you release a button, its value becomes 1 again. If either button is pressed, BUTTON NAND becomes 1, and it is 0 only when both buttons are not pressed.

Kit Assembly is Complete

You have now finished assembling the class kit (Figure 28). In the upcoming I/O labs, you will use the kit to learn about memory-mapped I/O and about handling low-level interrupts.

Turn-in and Grading

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

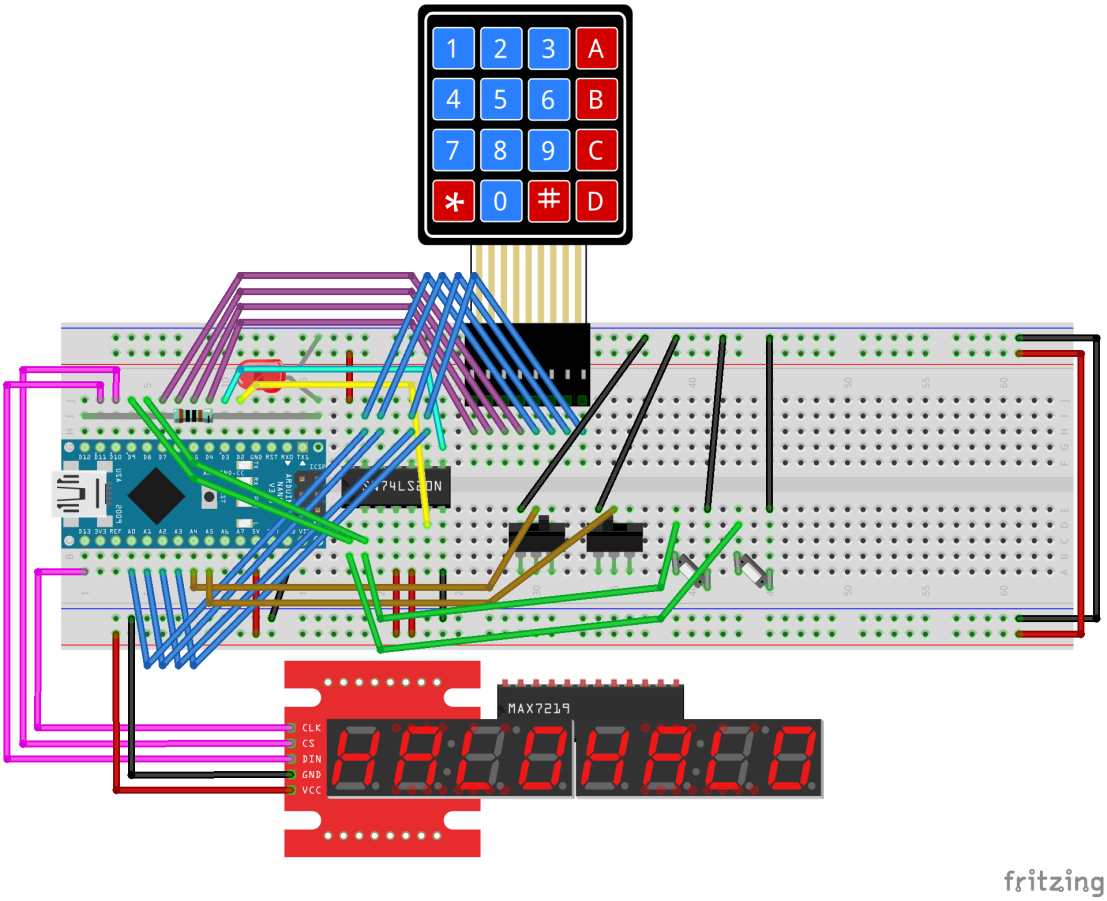
This assignment is worth 1 point, with up to 5 bonus points for helping other students.

Rubric:

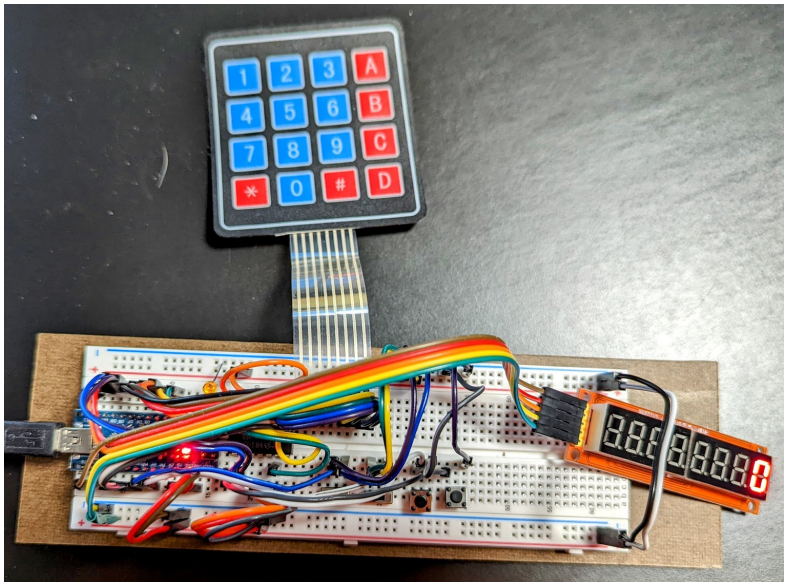
_____ **+0.5** You bring your fully-assembled class kit to your lab section the week it is due.

_____ **+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 28: The fully-assembled class kit.