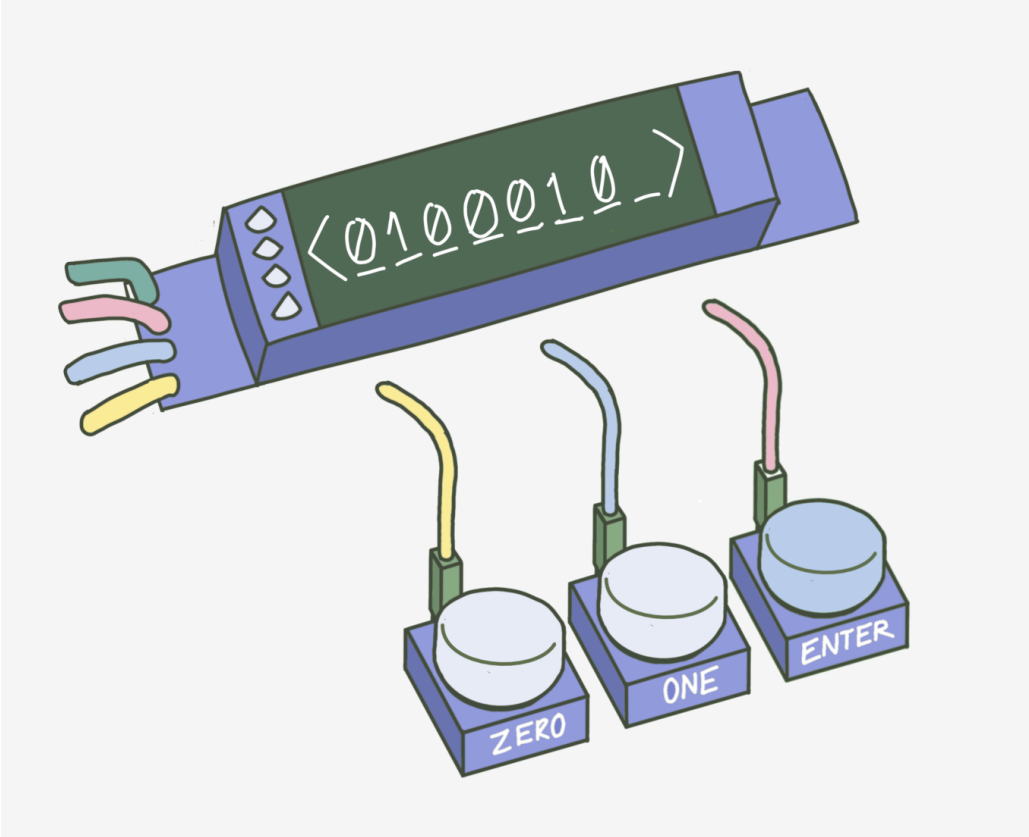
Project in a Box

Binary Keyboard



Developers Ferrari Guan, David Hong, Jin Noh, Ricky Wen, Kyle Estep, Jaimie Deng, Gowtham Duggirala, Kyle Trinh

Technical Writers Ferrari Guan, Ali Alabiad, Emma Chen

Last Updated 04-21-2025

Table of Contents

Introduction

In this workshop, you will learn how to create a programmable binary keyboard using Arduino and Python. You type letters using binary representation. For example, typing “01000001” will result in the letter “A”. You can also use the keyboard for playing the rhythm game OSU.

Before the workshop begins, look at the getting started page to make sure you have all the required software and hardware materials.

Learning Objectives

- Building circuits
- Programming in Arduino C
- Programming in Python
- Using libraries

Getting Started

Required Downloads and Installations

Arduino IDE is an Integrated Development Environment built specifically for using an Arduino. The IDE includes a built-in compiler and uploader to easily write, compile, and upload code to Arduino devices. Also, the IDE will allow you to send and receive data between your computer and Arduino.

For this workshop, we **highly recommend** downloading the most recent version of the **Legacy Arduino IDE** (1.8.XX) program, which can be found at the bottom of this link. Make sure you are downloading the Arduino IDE for the correct operating system.

Python is a beginner-friendly programming language known for its simple syntax and versatility. It is widely used in data science, automation, and more. Python has extensive libraries that allow users to build powerful software and applications.

For this workshop, any Python version newer than 3.11 is recommended. You can find the downloads for Python by following this link. Make sure you are downloading Python for the correct operating system.

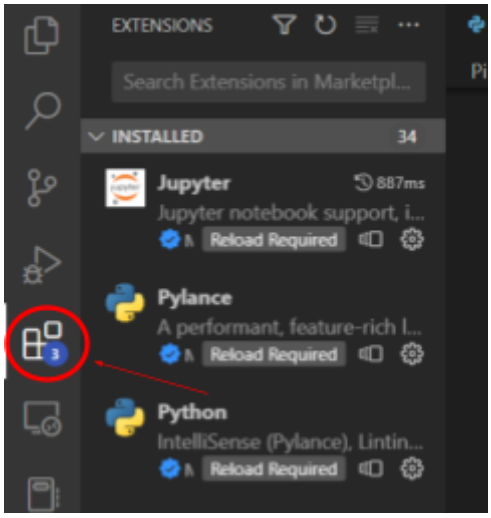
Once you download the Python setup file, open the setup file. Make sure to check the `Add python.exe to PATH` checkbox. Click `Install Now` to get started.

Next, go to the folder location where Python is downloaded, search for `pip.exe`, and click on it to install pip.

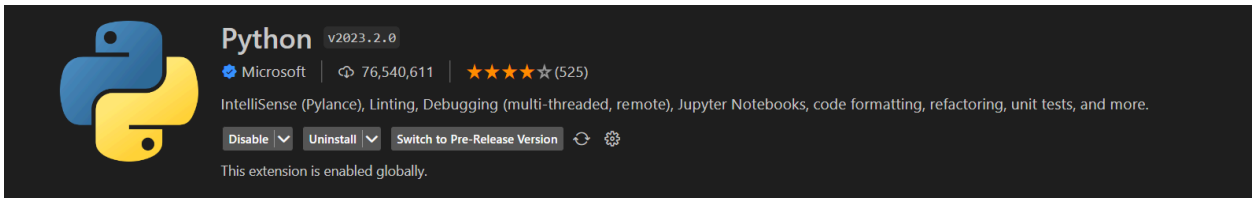
Visual Studio Code, also known as VS Code, is a code editor developed by Microsoft. VS code offers features to and supports a wide range of programming languages. VS code has cross-platform compatibility, good performance for coding and development, and Git integration.

For this workshop, the newest version of VS Code is recommended. You can find the downloads for VS Code by following this link. Make sure you are downloading VS Code for the correct operating system.

Open VS Code and go to `extensions`.



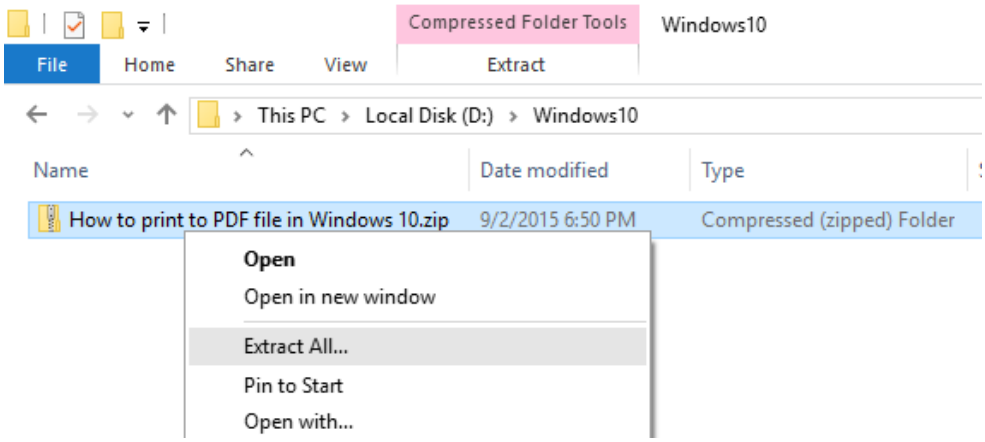
Type `Python` into the search bar and click `Download` on the Python extension.



Now that all the required software is installed, it is time to set up a virtual coding environment for all your code.

What's in the folder (along with the documentation)

1. **Binary_Keyboard(.zip File):**
    a. **Download the file:**
        i. By default, on **Windows**, the file will be in the **downloads** folder in your **file explorer**.
        ii. By default, on **Mac**, the file will be in the **downloads** folder in your **Finder**
    b. **Uncompress this zip file:**
        i. On **Windows**, right-click the file and select **Extract All**.

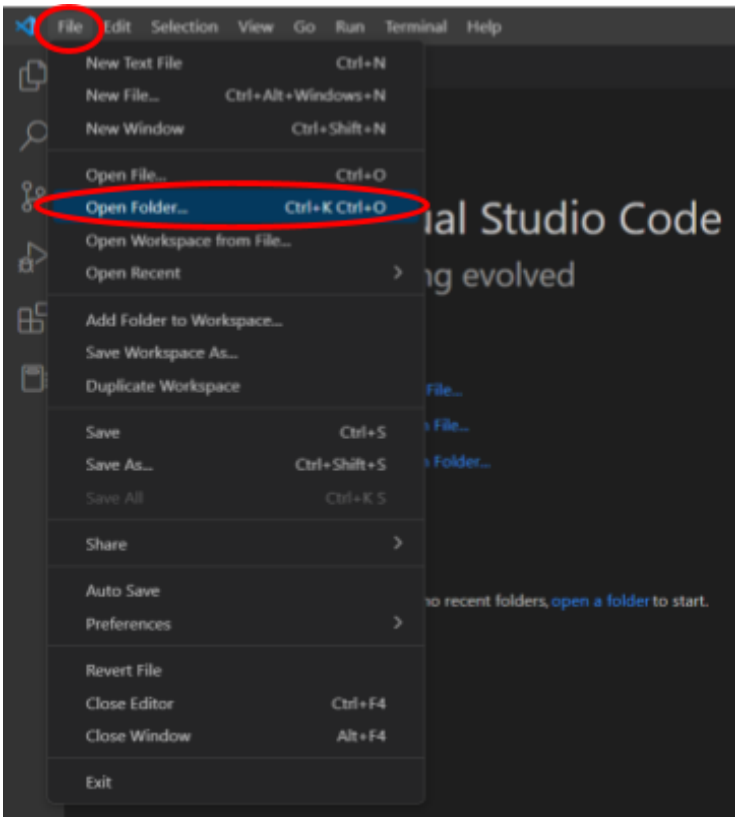


        ii. On **Mac**, double-click the file.

c. Now, you have a folder with the following contents:
    i. **Requirements.txt:** This text file consists of a list of libraries that we will download. (DO NOT EDIT THE FILE)
    ii. binary_keyboard_workshop.ino
    iii. typer_workshop.py

Open the uncompressed folder on VSCode:

1. Go to **File** (see picture below):



2. Choose **Open Folder** and select the folder you have decompressed

Downloading the libraries (Using the Requirements.txt file):

1. Make sure that all the files listed above are present.
2. Once you have opened the folder, left-click and then right-click (equivalent to two-finger click on some trackpads) the typer_workshop.py and click “Open in integrated terminal”. A terminal window should be opened that looks like this:

3. On the terminal, type the following **`python -m venv env`** (You may need to use python3 instead of python in that command)

   This will create a virtual environment for your code and prevent accidental damage to your system.

4. Now, activate the virtual environment:
   a. # on Windows: **`env\Scripts\activate`**
   b. # on macOS and Linux: **`source env/bin/activate`**
5. On the terminal, type **`pip install -r requirements.txt`**

```
PS C:\Users\aliz0\OneDrive\Desktop\Projects-2023\Geno\PiB-DiscordBot> pip install -r requirements.txt
>>
```

This will install all the needed libraries to allow you to get started coding your own binary keyboard.

Required Components

Component Name	Quantity
Arduino Nano	1
USB A to USB Micro B cable	1
Breadboard	1
Keyboard Switch	3
Male-to-Female Jumper Wires	4
OLED Screen	1
Enclosure pieces	7

Required Tools and Equipment

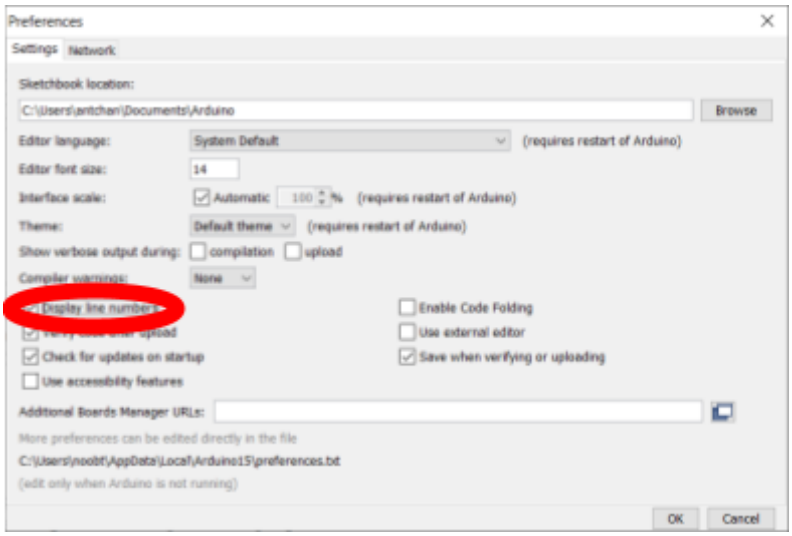
- Laptop with an internet connection
- Binary Keyboard Kit

Troubleshooting

Enabling Line Numbers

Enable line numbers in your Arduino IDE with the following instructions

- Navigate to **File → Preferences**
- Check the option for **Display line numbers**
- On Mac, it is **cmd + ,**
- On Windows, it is **ctrl + ,**

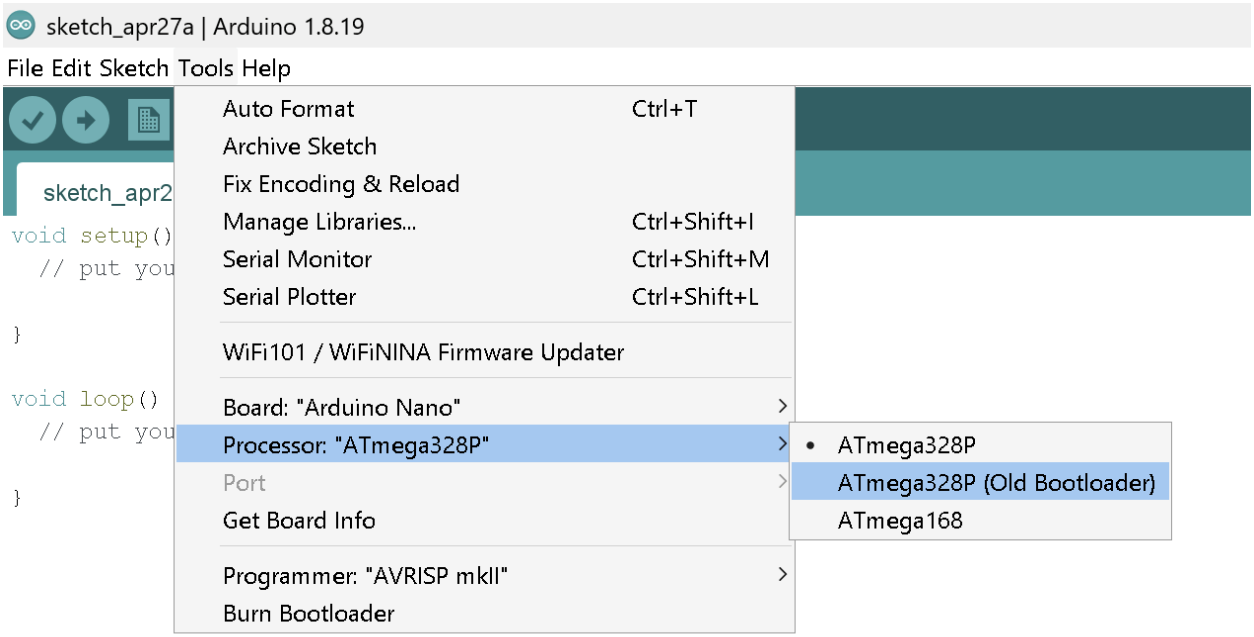


Driver Issues

If none of the ports work with your Arduino, you might not have the correct drivers installed.
Please download and install the CH340 driver here:
https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all#drivers-if-you-need-them

Code Upload Errors

- Try unplugging the Arduino and then plugging it back in.
- Double-check that you have the correct board and port selected for your Arduino.
- Double-check your breadboard connections.
- Double-check your code. Make sure pin number variable assignments match your wiring.
- Your Arduino Nano processor (ATmega328P) may be an older version. Try the old bootloader.



Challenge #1: Make the Keyboard

Introduction

In this challenge, you will build the keyboard! Once you are finished with this section, you will be able to click-clack away!

Objective

- Build the hardware and have a binary keyboard at your fingertips.

Background Information

As you work with more complex projects, you will be using a plethora of different components such as microcontrollers, sensors, displays, and etc. For each of these, you would need to connect them in a circuit and write code to configure those components. Fortunately, we are just starting off with putting components on the breadboard and wiring them up. For this challenge, we will provide detailed instructions on creating the circuit.

Components

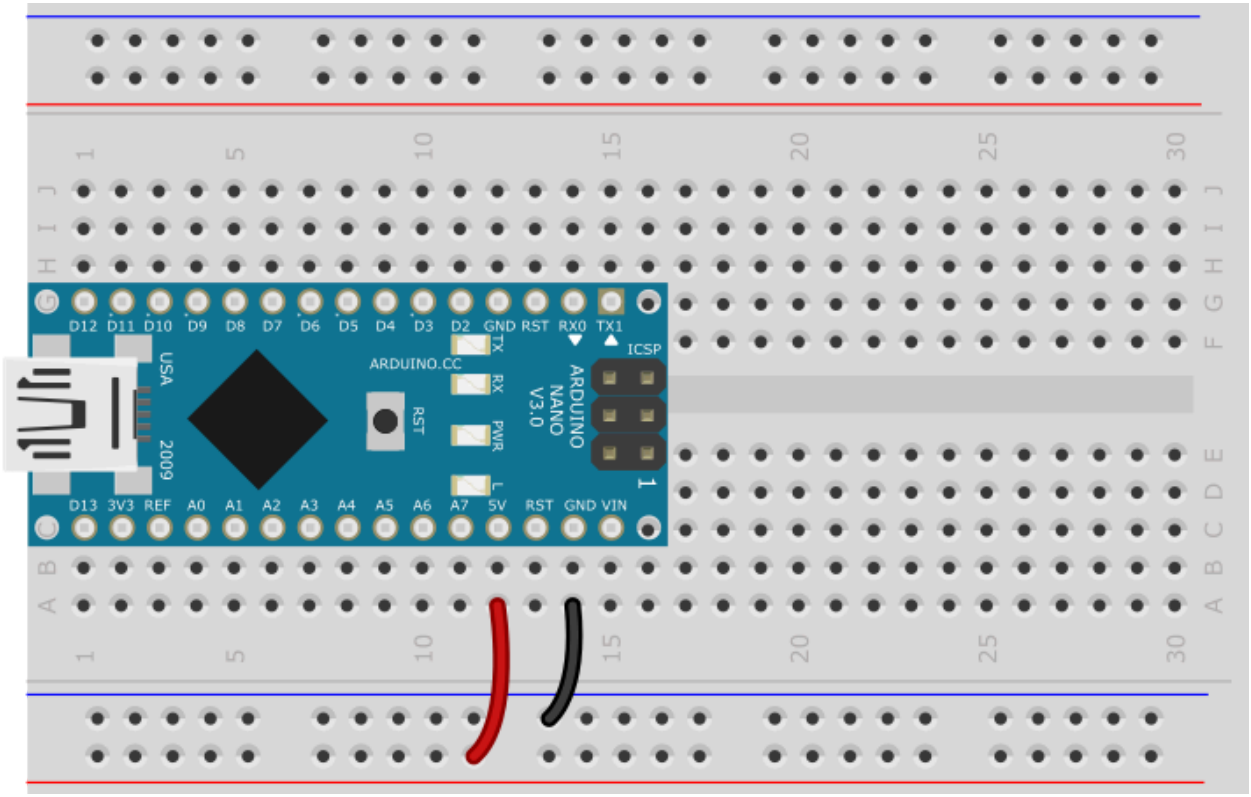
- Arduino Nano
- USB A to USB Micro B cable
- Breadboard
- Keyboard Switches
- Male-to-Male Jumper Wires
- OLED Screen

Instructions

Building the Circuit

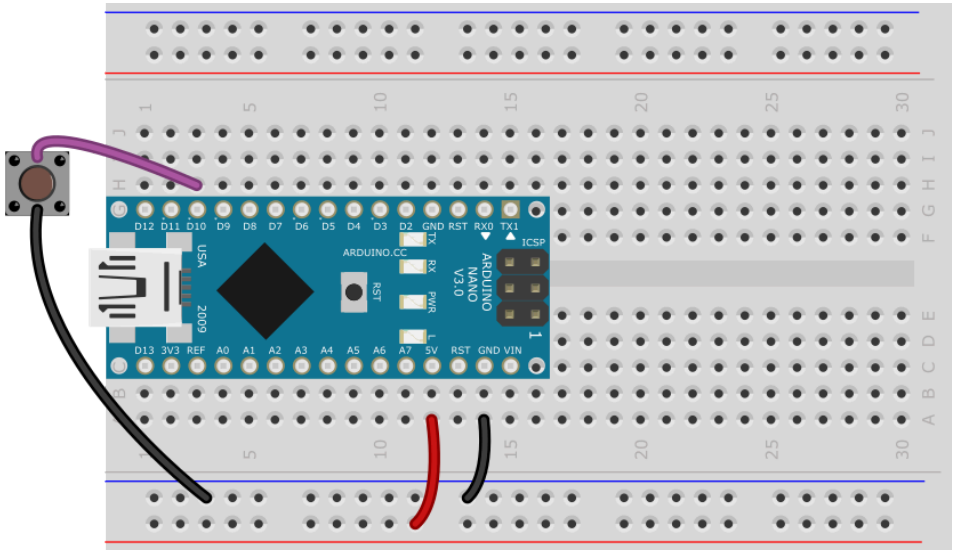
Step 1:

Place the Arduino Nano with the USB port facing out. On the Arduino, connect the 5V pin to the power (red) rail and connect the GND pin to the ground (blue) rail. Make sure you are using the small precut wires. Refer to the image below for a visual of this step.



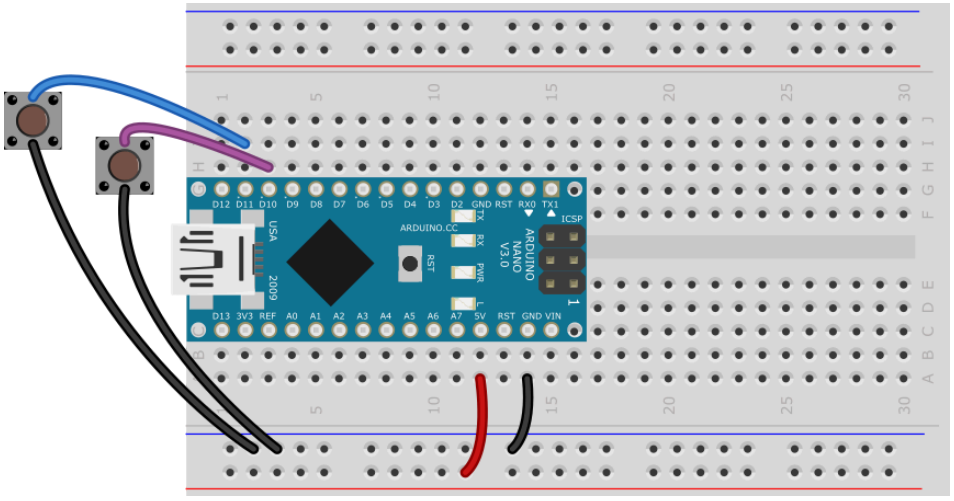
Step 2:

Connect one wire from a switch to D10 pin on the Arduino. Connect the other wire from the switch to the ground rail.



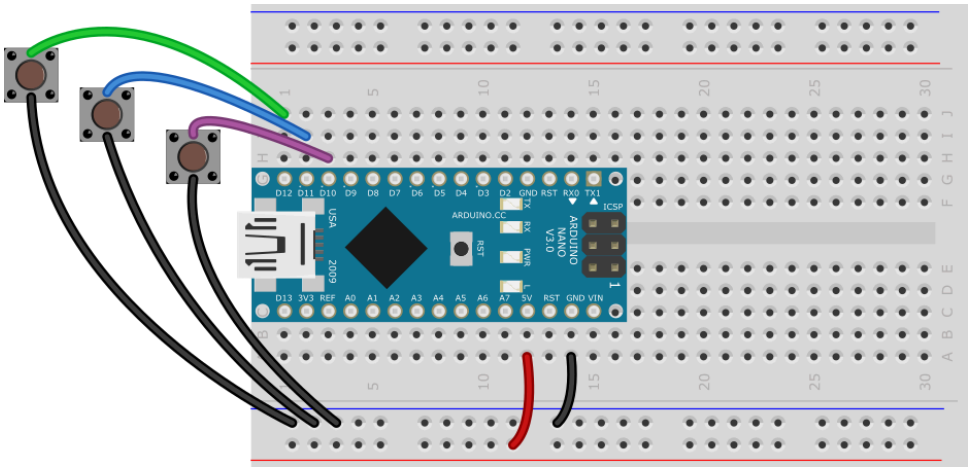
Step 3:

Repeat the step above with another switch, but connect one wire to the D11 pin on the Arduino.



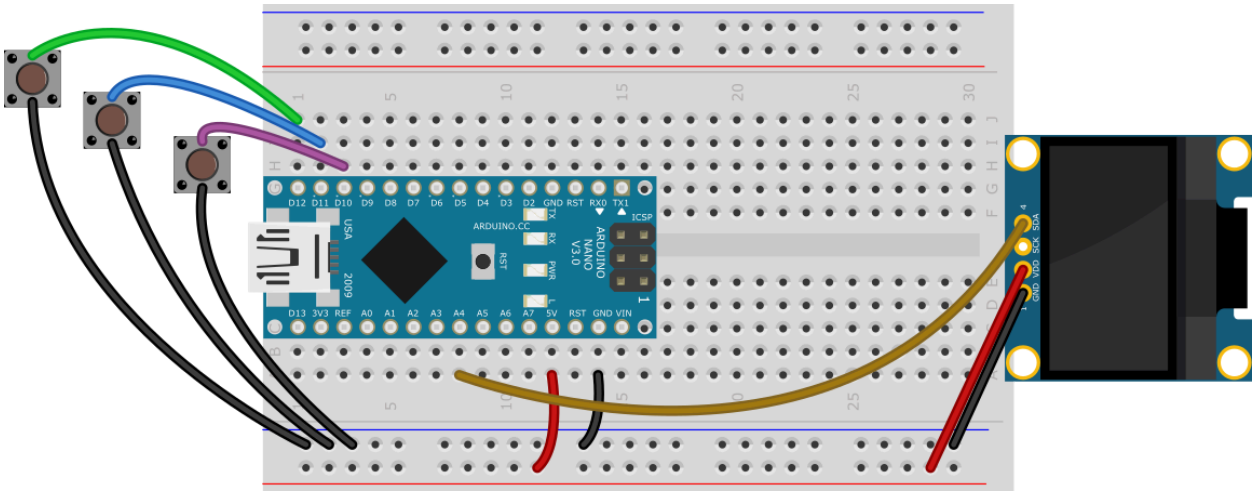
Step 4:

Repeat the step above with another switch, but connect one wire to the D12 pin on the Arduino.



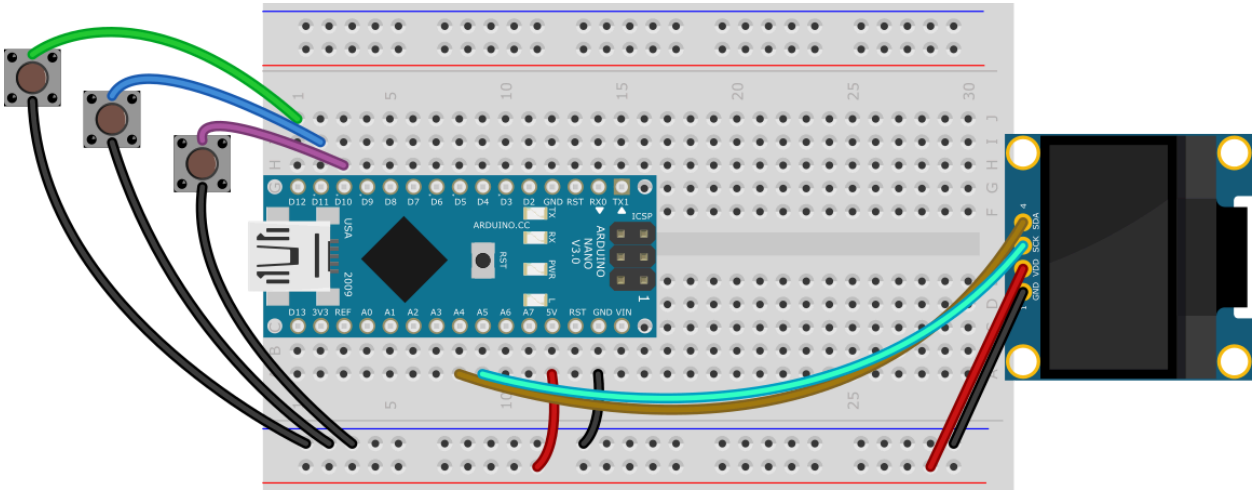
Step 5:

Use the male-to-female wires for this step. Connect the GND pin on the OLED to the ground rail. Connect the VDD pin on the OLED to the power rail. Connect the SDA pin on the OLED to the A4 pin on the Arduino.



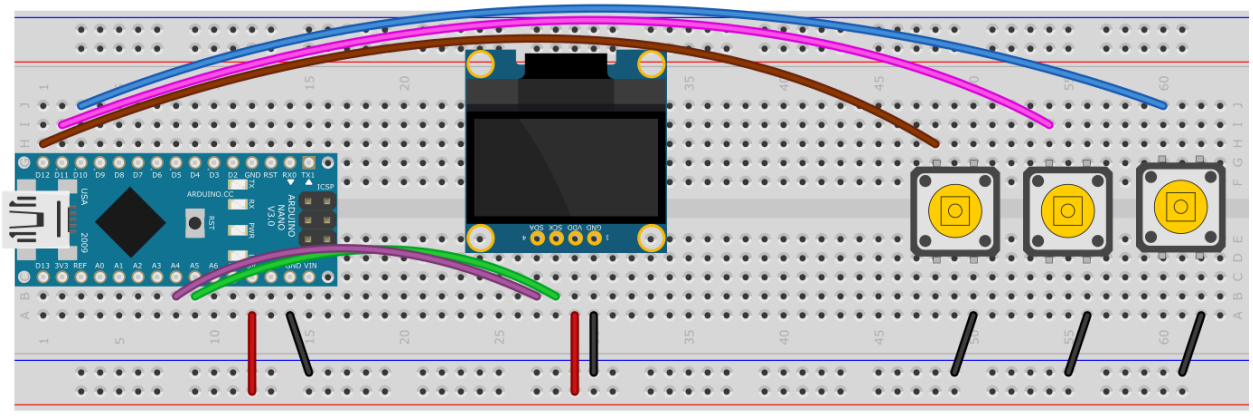
Step 6:

Use the male-to-female wires for this step. Connect the SCK pin on the OLED to the A5 pin on the Arduino.



The final circuit:

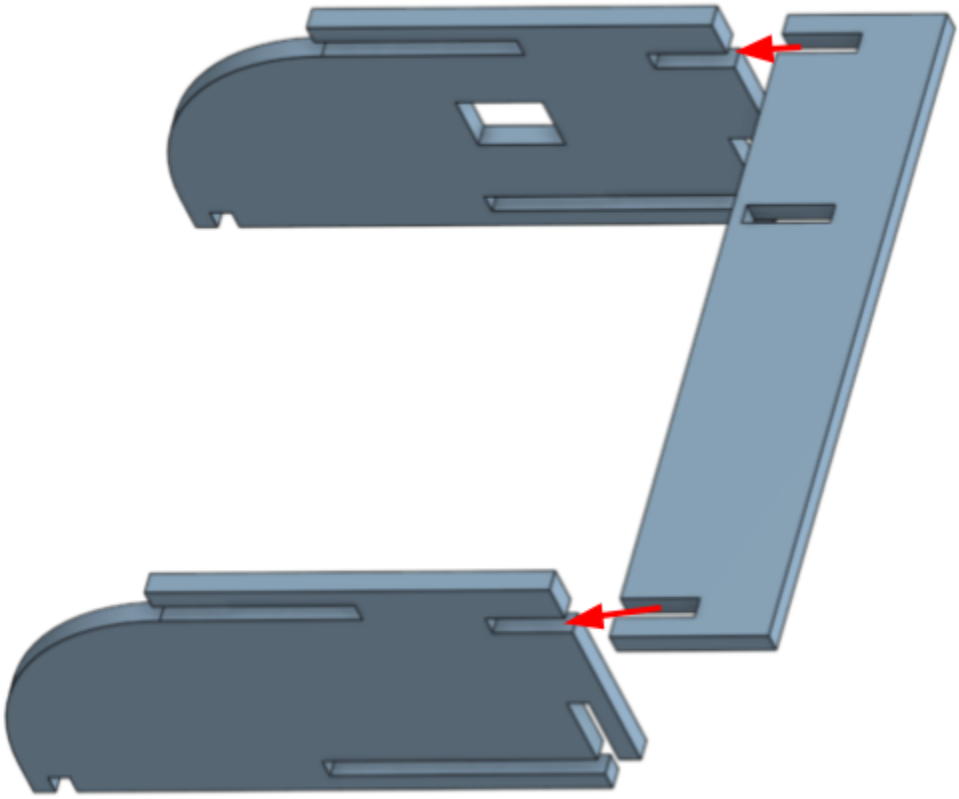
Congratulations on completing the circuit! The circuit above is equivalent to the one pictured below. The circuit below is for your reference.



Making the case:

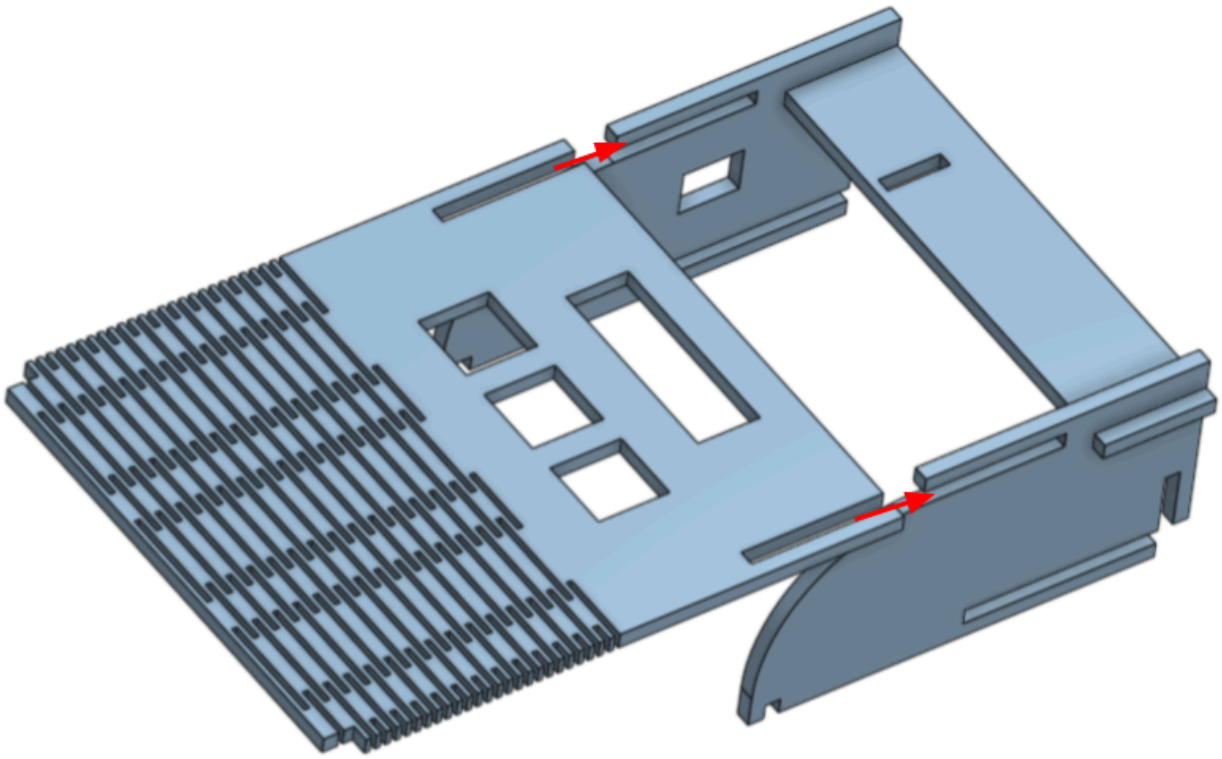
Step 1:

Take the OLED support panel and slot both side panels into it.



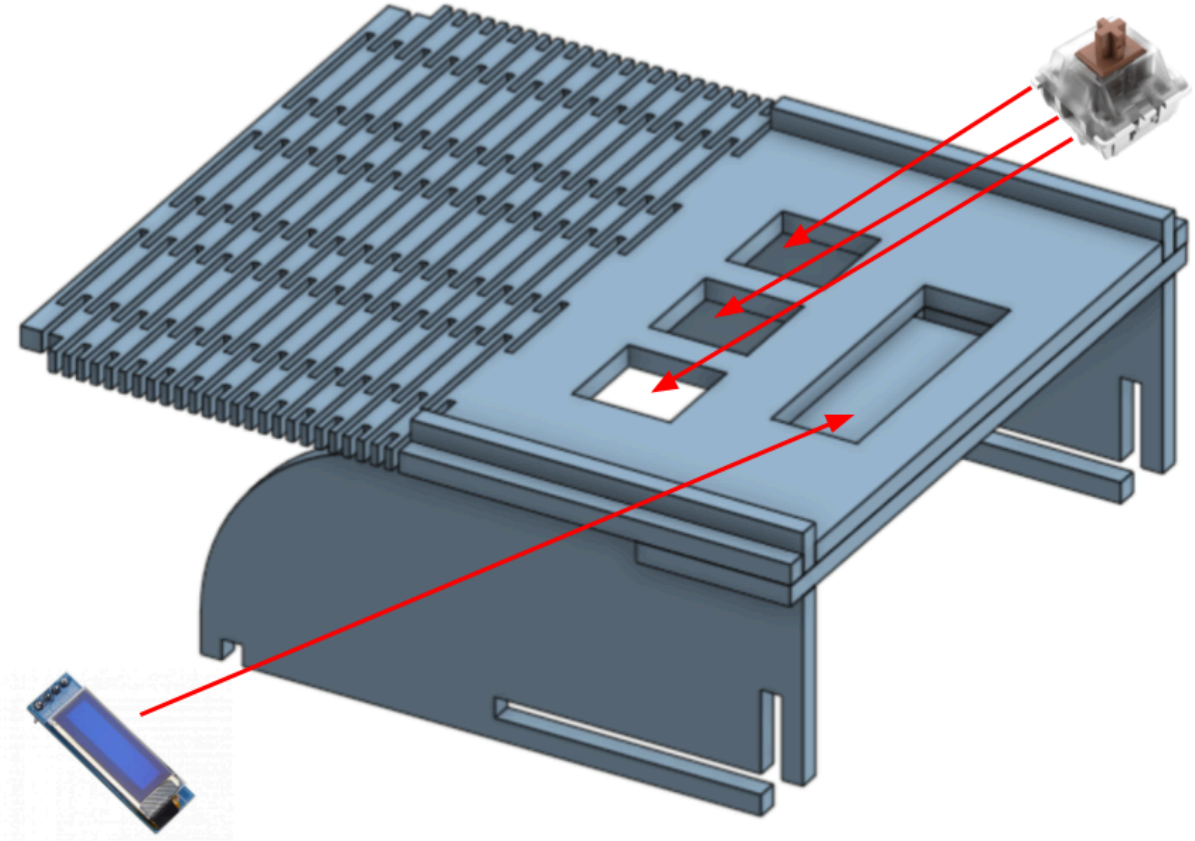
Step 2:

Slide the top panel into place between the two side panels and above the OLED support panel.



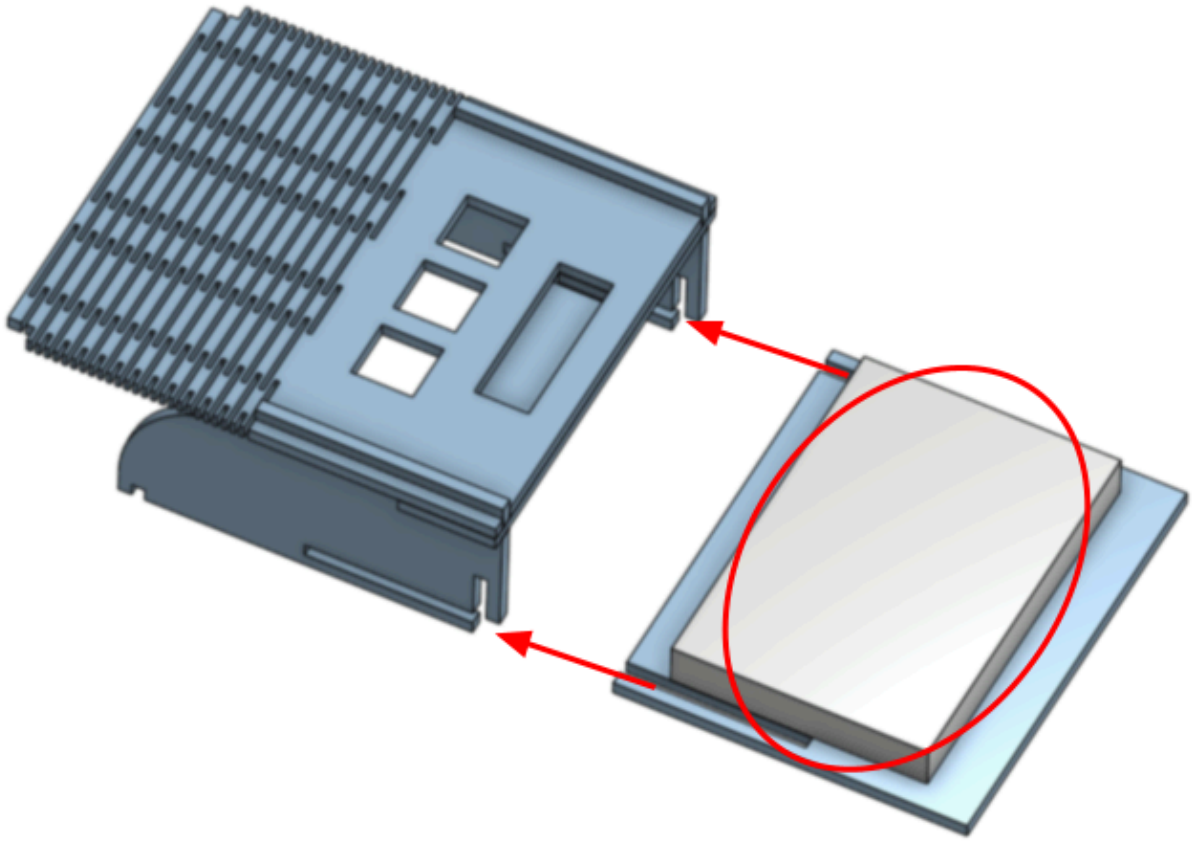
Step 3:

Place the OLED and key switches into the holes facing up on the top panel. Rewire all your components by repeating the circuitry steps.



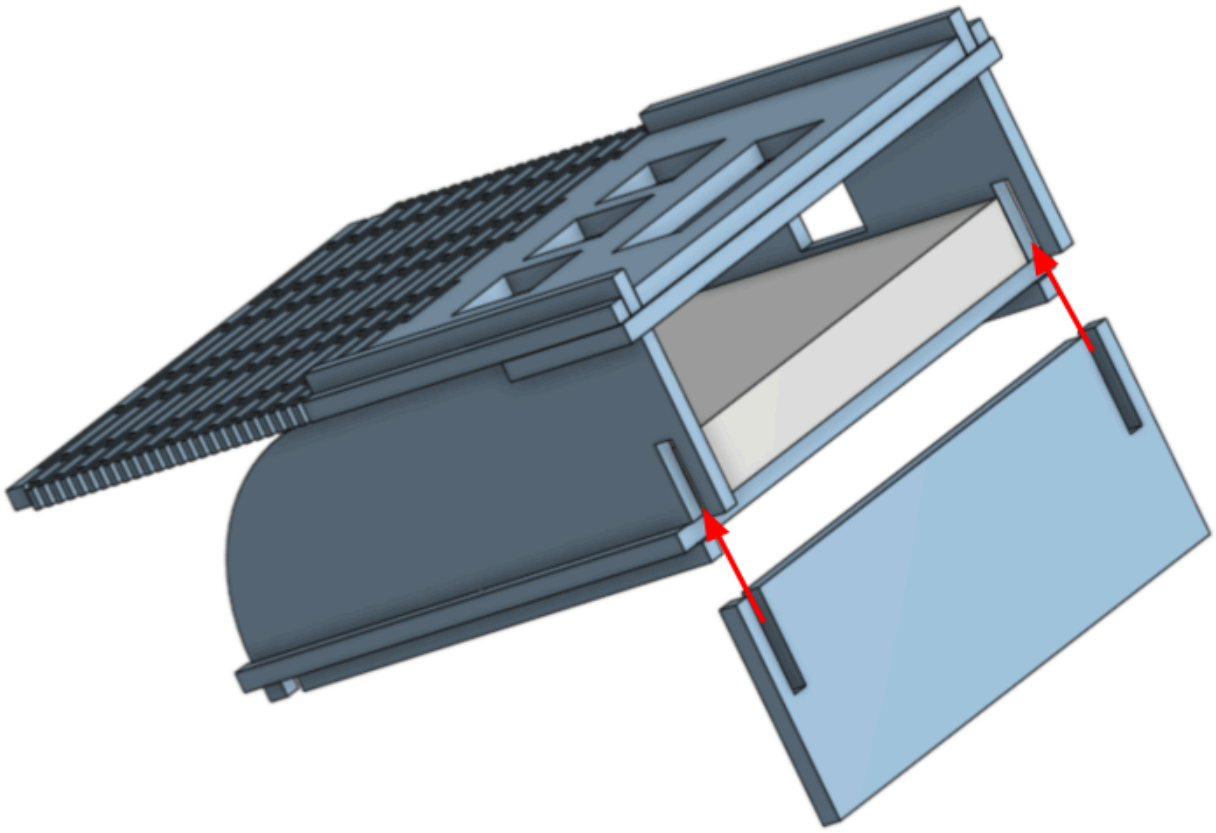
Step 4:

Place the breadboard on the bottom panel and slot the bottom panel into the two side panels.



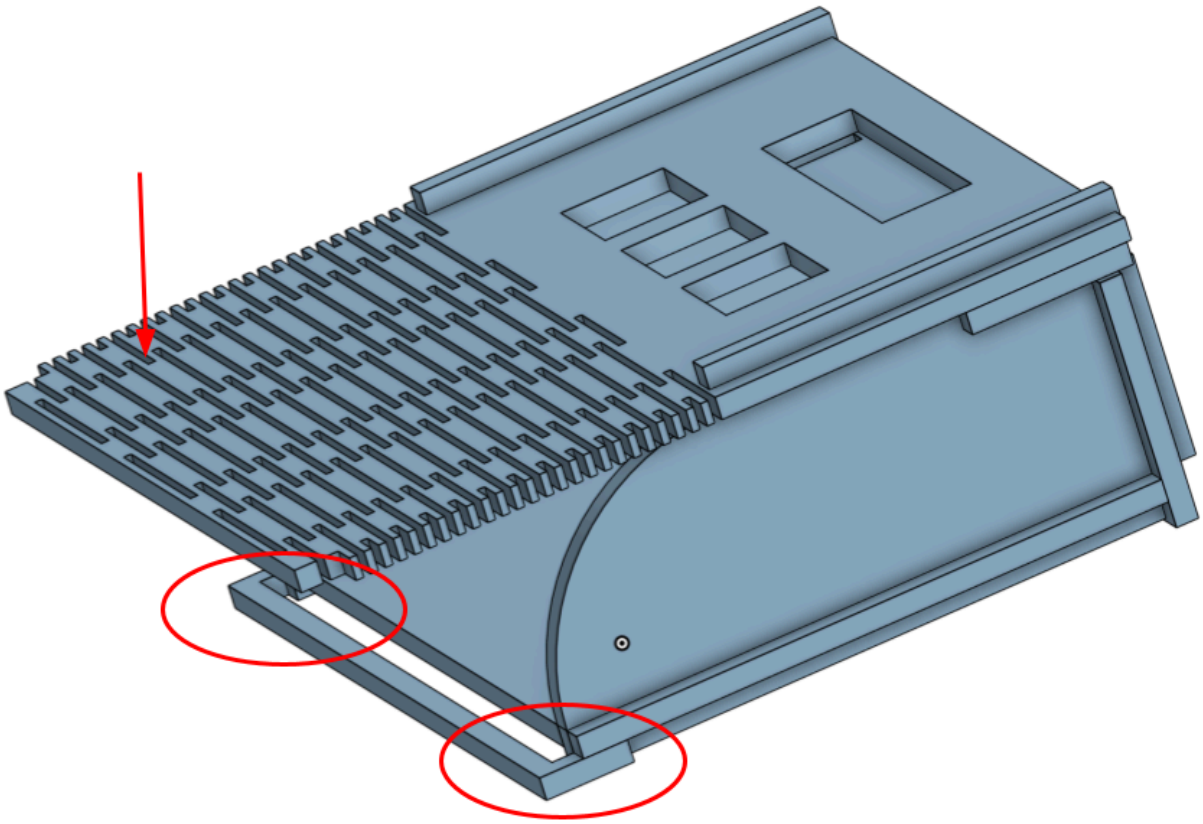
Step 5:

Slot the back panel into place between the two side panels.

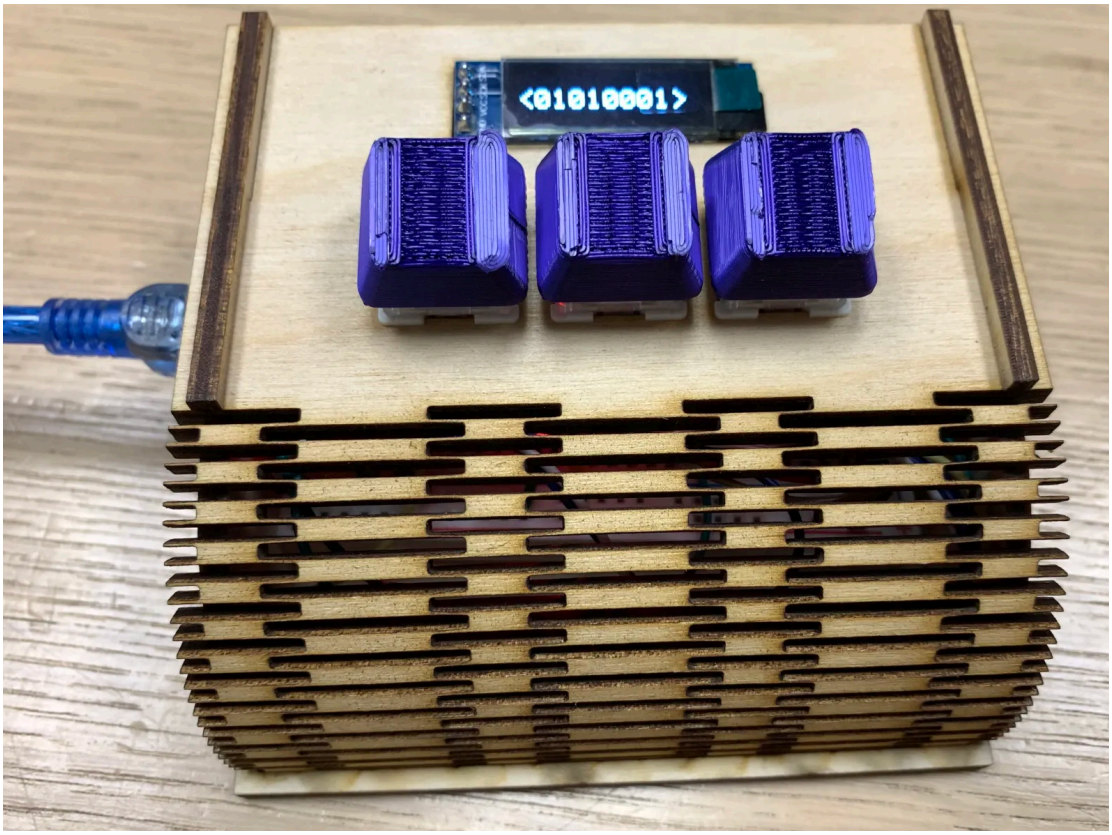


Step 6:

Place the bracket into the notches at the front of the side panels. Push the front tab of the top panel into the bracket.



The final product:



Challenge #2: Code the Arduino

Introduction

The keyboard does not do anything! Well, at least not yet. Now, you will program the Arduino and give life to your keyboard in this challenge.

Objective

- Installing libraries
- Code the Arduino to have a functioning keyboard.

Background Information

Firmware is a type of software that is embedded in hardware devices to help them function. When writing the code for the keyboard firmware, we must specify what each button does. We also need to make sure the OLED screen displays the correct characters when a button is pressed. Lastly, we need to ensure that the Arduino sends the correct serial signal when a button is pressed.

Components

- Finished Binary Keyboard hardware
- Laptop
- binary_keyboard_workshop.ino file

Instructions

First, you must open your Arduino IDE. Please read carefully because different versions of the Arduino IDE have slightly different library requirements for this project.

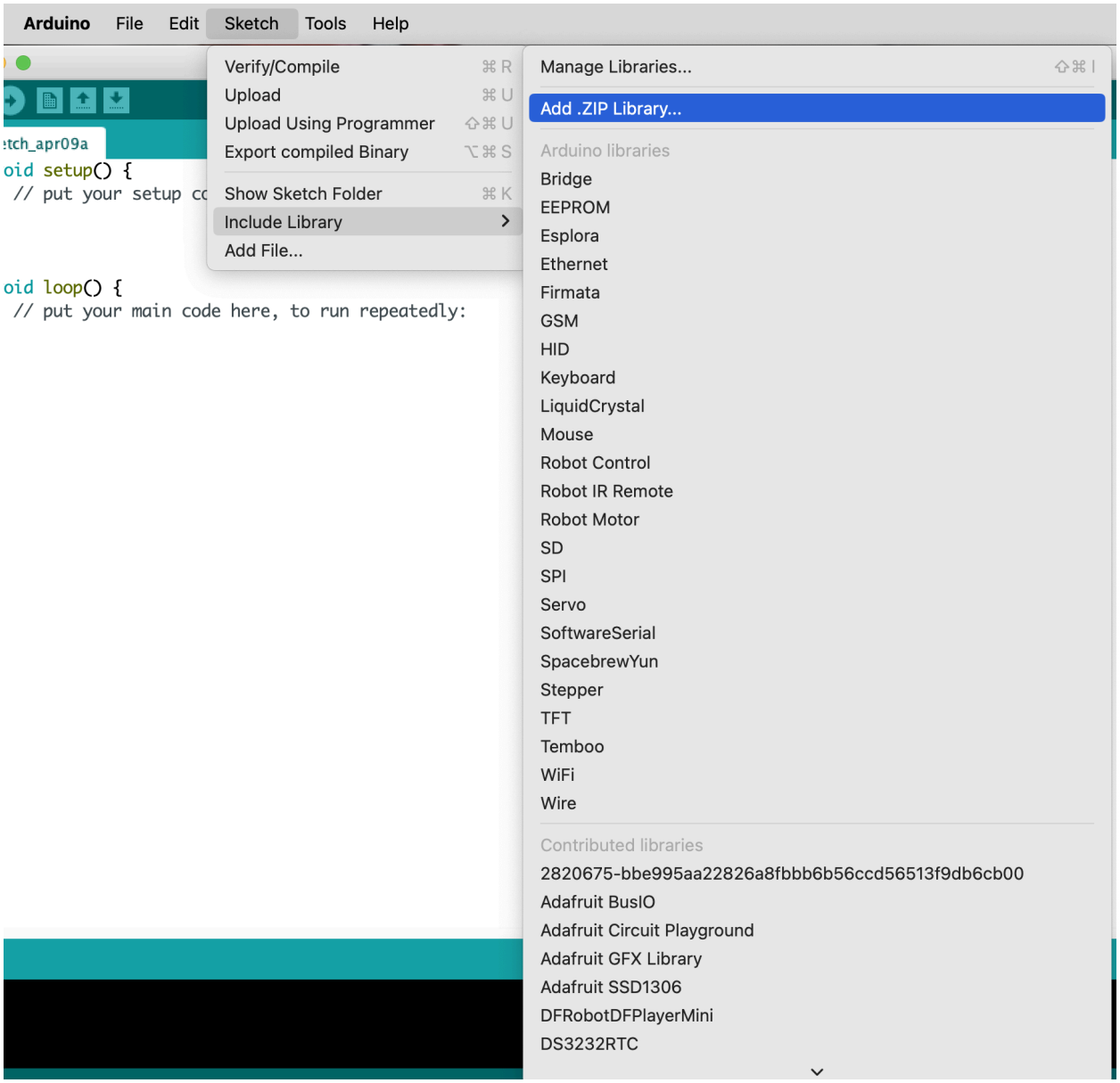
Library Installation (Arduino IDE Version 1.8.XX)

You will need the following:

- Adafruit_SSD1306
- Adafruit_GFX

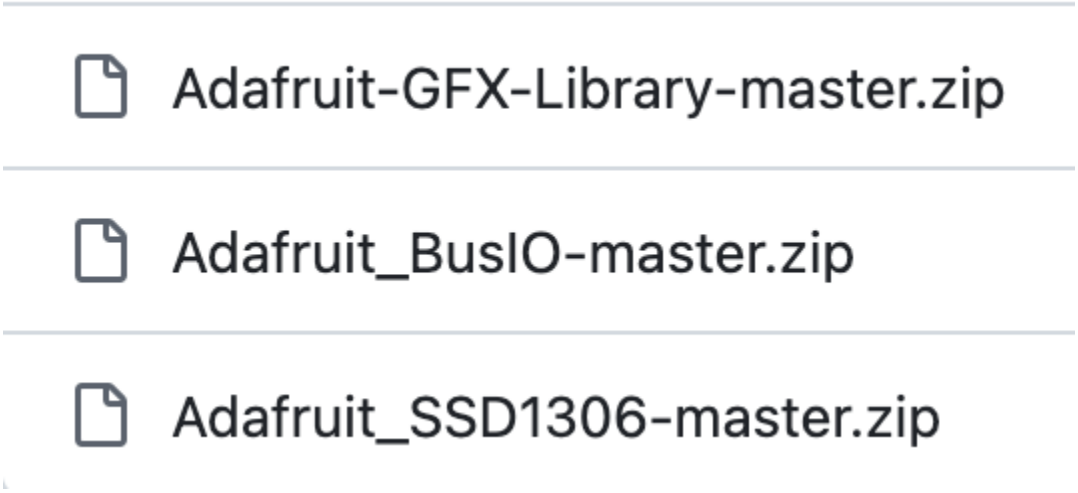
Step 1:

Hover over the **Sketch** tab on top of the page, then click on **Include Librar,y** followed by **Add .ZIP Library**



Step 2:

Click on **Adafruit-GFX-Library-master.zip** and **AdafruitSSD1306-master.zip**



This should install both libraries that you would need for the workshop; the other two (Wire.h and SPI.h) are already available

Start the Code

Open the binary_keyboard_workshop folder.ino file and quickly peruse it.

Step 1: Setting up the OLED display

The OLED display code takes the height of the screen and the width of the screen as function arguments. Function arguments are the width, height, the wire library, and the reset.

```
Adafruit_SSD1306 display(OLED_WIDTH, OLED_HEIGHT, &Wire, reset);
```

Step 2: Setting up the buttons

Variables are named storage locations that hold a value or data, similar to the variables in math. The button variables should be mapped to the pin numbers on the Arduino.

```
const int zeroButton = __;

const int oneButton = __;

const int enterButton = __;
```

Next, the variables need to be set as buttons.

```
pinMode(___, INPUT_PULLUP);

   pinMode(___, INPUT_PULLUP);

   pinMode(___, INPUT_PULLUP);
```

Step 3: Programming the logic

The logic for the firmware should be put into the void loop. When a key is not pressed, the signal is high. When a key is pressed, the signal becomes low. The one button being pressed corresponds to the one button being set to low. The buttons that you do not want to be pressed should be set to high. This logic extends to the other buttons.

```
if (digitalRead(___) == LOW && digitalRead(___) == HIGH && digitalRead(___) == HIGH){
            pressZero();
            delay(keyInputDelay);
        }
         else if (digitalRead(___) == LOW && digitalRead(___) == HIGH && digitalRead(___) == HIGH){
            pressOne();
            delay(keyInputDelay);
        }
         else if (digitalRead(___) == LOW && digitalRead(___) == HIGH && digitalRead(___) == HIGH){
            pressEnter();
            delay(keyInputDelay);
        }
         else if (digitalRead(___) == LOW && digitalRead(___) == LOW && digitalRead(___) == HIGH){
            pressBack();
```

```
            delay(keyInputDelay);

        }

         else if (digitalRead(___) == LOW && digitalRead(___) ==
LOW && digitalRead(___) == LOW){

            pressClear();

            delay(keyInputDelay);

        }
```

Optional: Changing the start up sequence

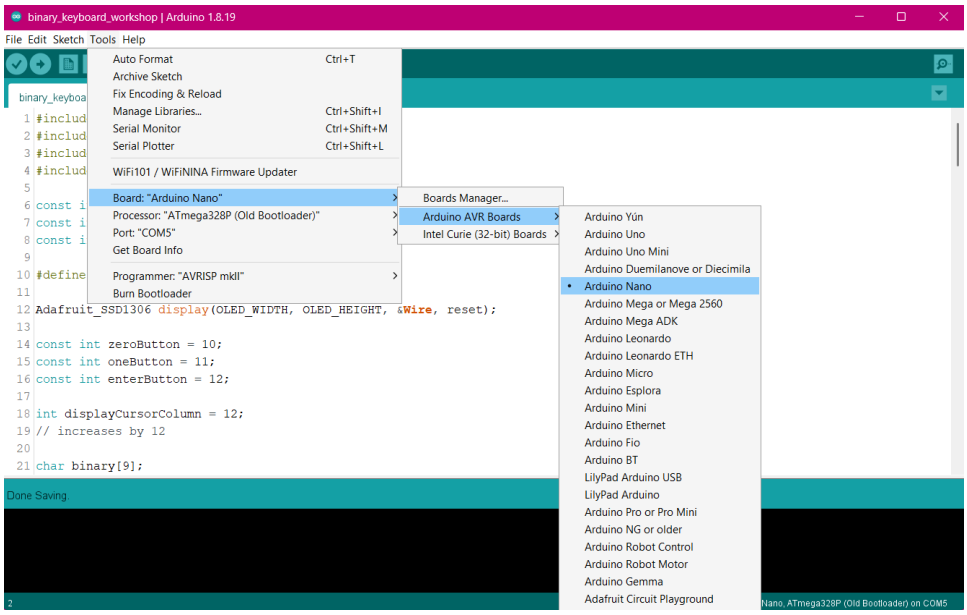
In the startText() function, the code that shows the start-up sequence words is shown. The startTextLines holds the data to be displayed for the start-up sequence. Make sure to chunk it into 10 characters so it can show up on the screen. Replace the text inside the quotation (“) marks to change the start up sequence text. You can add more text, but you would need to change the number 4 in the brackets to reflect your modified number of start up sequence text chunks.

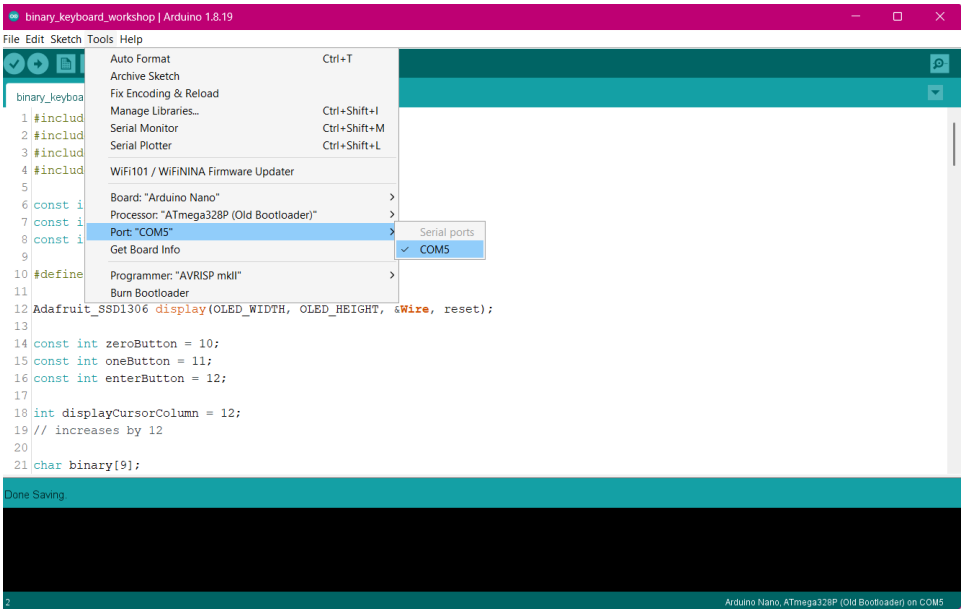
```
const char startTextLines[4][10] = {"Welcome to", "PIB Binary",
"Keyboard! ", "<________>"};
```

You can also implement a scrolling word sequence, but it will take up more memory. So, this will be outside the scope of the given hardware. If you have a more powerful Arduino model, you can implement this feature.

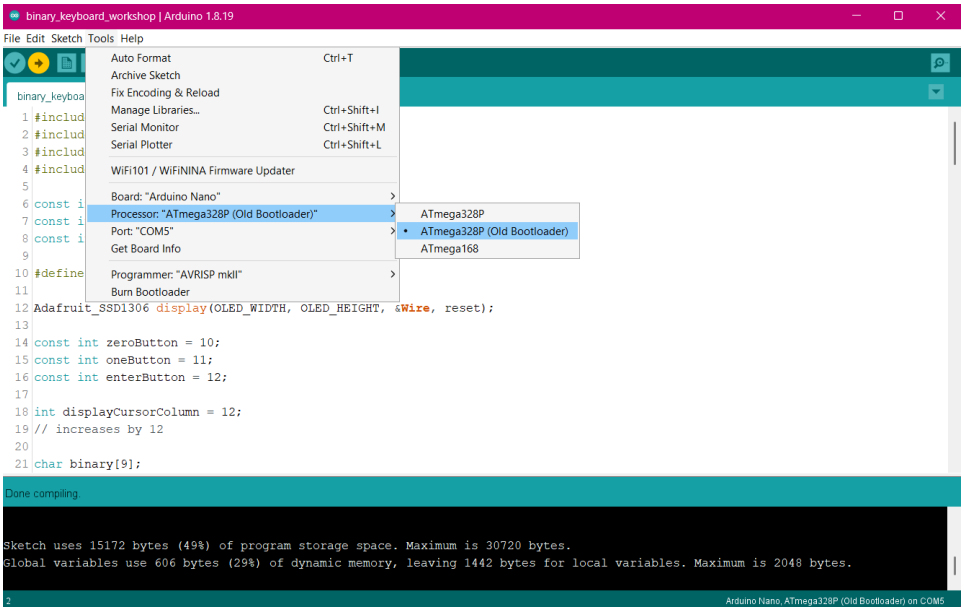
Uploading the Code:

Select the correct board and port on your Arduino IDE. See the following images for reference.





Select the correct processor on the Arduino IDE.



Upload the code to your Arduino board. Then, close the Arduino IDE.

Challenge #3: Code the Python

Introduction

Wait a minute, the keyboard turns on pretty, but it still does not type anything! In this challenge, you will make a Python program that converts your binary into real letters.

Objective

- Installing libraries
- Make the keyboard communicate with your computer.

Background Information

The Arduino Nano does not have built-in keyboard typing functionalities. So, we need to have something on the computer that interprets the data that the Arduino Nano outputs. So, this is where Python comes in handy. Pyserial is a library in Python that allows the computer to receive serial outputs from the Arduino Nano.

Components

- Finished Binary Keyboard hardware
- Laptop
- typer_workshop.py file

Instructions

If you have not completed the steps in the Getting Started section, please complete them.

Start the Code

First, we need to convert the received binary string data from the Arduino into a decimal number. Store this decimal number in the decimal_value variable.

```
decimal_value = int(___, 2)
```

Now that the data is a decimal number stored in decimal_value, we need to get the corresponding American Standard Code for Information Interchange (ASCII) character. These are characters like “J” or “{“. Store this character in the asciiKey variable.

```
asciiKey = chr(___)
```

Next, type the character using the pyautogui library. Arguments are inputs that you pass into a function, like the pyautogui typewrite function below. The first argument should be the letter that you obtained previously. The second argument should be the delay between each input in seconds. Fractions of a second are allowed in the second argument.

```
pyautogui.typewrite(___, interval=___)
```

Finally, you are done! Press the run button on the top right of VS Code. Start typing away, and enjoy your binary keyboard! The chart for all the binary to ASCII character conversions can be found here.

Appendix

Acknowledgments

Thank you to Nicholas ‘Nix’ J. Stein for providing design examples for the case.

This project was inspired by Convict Cantaloupe’s project.

Additional Resources

You may access the slides from the workshop lecture portion.

You may refer to the original GitHub repository for this project.

You may refer to the original Google Drive to see all the hardware details.

Additional Challenge: Adding the OSU! functionality

Check the GitHub repository shown above and follow the README.md file. Use the full functionality implementation of the Arduino and Python code.

Additional Challenge: Creating a service for the keyboard

Congratulations on completing all the previous challenges! This is the final challenge and is very experimental. This challenge will be very dependent on your own system, so use the hints below in conjunction with other guides.

Background Information

A start-up service on an operating system is a background process or application that automatically begins running during the system's boot process. You will be converting the python typer.py file into a start-up service so you can use the binary keyboard at any time.

Components

- Laptop or desktop computer
- The typer_workshop.py Python file

Instructions

Windows:

First, open Task Scheduler by pressing Win + R, typing taskschd.msc, and pressing enter. Click Create Basic Task. Name it BinaryKeyboardService. Set When the computer starts as the trigger. Set Start a Program as the action. Browse for python.exe, which is usually in the file path

C:\Users\<Your Username>\AppData\Local\Programs\Python\Python39\python.exe. In the "Add arguments" field, enter the path to the typer_workshop.py Python file. Lastly, click Finish.

MacOS:

Create a Launch Agent plist file in ~/Library/LaunchAgents/binary_keyboard.plist by using XML. Use launchctl to load and start the service. Please refer to Apple's documentation to perform this.

Linux:

There are many Linux configurations, so I cannot cover all of them. If you are using systemd, create /etc/systemd/system/binary_keyboard.service. Then, use systemctl to enable it. There are guides and documentation for this online. For other linux init systems, like runit and openrc, please refer to community documentation or guides on the internet.