



GNG 2101  
**Design Project User and Product Manual**

**TREMORE REDUCING MOUSE CONTROL**

Submitted by:

Mouse Maneuvers - GROUP 1.3

Maria Abril, 300227646

Jason Jin, 300184747

Matthew Emmanuel, 300240312

07/04/2023

University of Ottawa

# Table of Contents

- Table of Contents ..... ii
- List of Figures .....v
- List of Tables ..... vi
- List of Acronyms and Glossary ..... viii
- 1 Introduction..... 1
- 2 Overview..... 2
  - 2.1 Conventions .....5
  - 2.2 Cautions & Warnings.....5
- 3 Getting started.....8
  - 3.1 Configuration Considerations .....52
  - 3.2 User Access Consideration .....54
  - 3.3 Accessing/setting up the System.....54
  - 3.4 System Organization & Navigation .....54
  - 3.5 Exiting the System .....57
- 4 Using the System .....62
  - 4.1 <Mouse Movement>.....62
    - 4.1.1 <Joystick Click> .....62
- 5 Troubleshooting & Support .....64
  - 5.1 Error Messages or Behaviors .....64
  - 5.2 Special Considerations.....65
  - 5.3 Maintenance .....65

|       |   |    |
|-------|---|----|
| 5.4   | Support.....  | 66 |
| 6     | Product Documentation .....                           | 67 |
| 6.1   | Mechanical subsystem .....                            | 67 |
| 6.1.1 | BOM (Bill of Materials) .....                         | 67 |
| 6.1.2 | Equipment list .....                                  | 67 |
| 6.1.3 | Instructions.....                                     | 67 |
| 6.2   | Electrical subsystem.....                             | 68 |
| 6.2.1 | BOM (Bill of Materials) .....                         | 68 |
| 6.2.2 | Equipment list .....                                  | 69 |
| 6.2.3 | Instructions.....                                     | 69 |
| 6.3   | Software subsystem .....                              | 69 |
| 6.3.1 | BOM (Bill of Materials) .....                         | 69 |
| 6.3.2 | Equipment list .....                                  | 69 |
| 6.3.3 | Instructions.....                                     | 69 |
| 6.4   | Testing & Validation.....                             | 70 |
| 7     | Conclusions and Recommendations for Future Work ..... | 71 |
| 8     | Bibliography .....                                    | 72 |
|       | APPENDICES .....                                      | 73 |
| 9     | APPENDIX I: Design Files .....                        | 73 |
| 10    | APPENDIX II: Other Appendices .....                   | 74 |

iv [Click here to enter text.](#)

## List of Figures

- Figure 1: Final Prototype
- Figure 2: Final Prototype with all components together
- Figure 3: Electrical connections
- Figure 4: Flowchart
- Figure 5: Finding port information on Arduino IDE
- Figure 6: LINE 20 from the IntelliJ IDEA CE code
- Figure 7: **Arduino IDE and IntelliJ IDEA CE Applications**
- Figure 8: Electrical connections to the Arduino Uno Board
- Figure 9: Type A, Type B USB
- Figure 10: Type B USB connection to the Arduino Uno Board
- Figure 11: Type A connection to computer device
- Figure 12: Arduino Uno Light meaning it is ON
- Figure 13: New Sketch creation
- Figure 14: File icon to create a New Sketch if New Sketch does not appear from the start
- Figure 15: Board information
- Figure 16: Port information – connected properly
- Figure 17: Port information – not connected
- Figure 18: Manually selecting port information
- Figure 19: Windows operating system, COM port information
- Figure 20: Pasting Code in the blank space
- Figure 21: Uploaded the Joystick Program code to the Arduino IDE
- Figure 22: The 2 main files required to download
- Figure 23: Java SE Development Kit 8u251 File
- Figure 24: Creating new project on IntelliJ IDEA CE
- Figure 25: New project Tab
- Figure 26: Adding JDK file
- Figure 27: Finding the Java SE Development Kit 8u251 File
- Figure 28: Selecting the 1.8 Oracle OpenJDK version 1.8.0\_251 File
- Figure 29: Creating New Project
- Figure 30: Project Structure Tab
- Figure 31: Libraries Tab
- Figure 32: Getting setup to add the 2 libraries
- Figure 33: Finding the 2 library files in your download files folder
- Figure 34: Selecting the jna-3.2.7-sources 2 jar file
- Figure 35: Adding the jna-3.2.7-sources 2 jar file
- Figure 36: The jna-3.2.7-sources 2 jar file has been added
- Figure 37: Finding the mfz-rxtx-2.2-20081207-win-x86 file
- Figure 37: Finding the mfz-rxtx-2.2-20081207-win-x86 file
- Figure 39: Adding the RXTXcomm.jar file
- Figure 40: The Choose Modules Tab
- Figure 41: Updated Project Structure tab with the 2 main libraries added
- Figure 42: The RXTXcomm Tab
- Figure 43: Double click the mfz-rxtx-2.2-20081207-win-x86 file

vClick here to enter text.

- Figure 44: Adding the **rxtxSerial.dll** file*
- Figure 45: Fully updated Project Structure Tab*
- Figure 46: Adding the entire Project Structure*
- Figure 47: Pasting the code in the blank space*
- Figure 48: Uploaded the Java Code on IntelliJ IDEA CE*
- Figure 49: Verifying the Arduino Code*
- Figure 50: Done compiling message*
- Figure 51: Uploading the Code onto the Arduino Uno board*
- Figure 52: Done uploading message*
- Figure 53: Building the IntelliJ IDEA CE project*
- Figure 54: Checking for errors*
- Figure 55: Ensuring line 13 is correct*
- Figure 56: Running the code*
- Figure 57: Thumb Joystick Device*
- Figure 58: Arduino Uno Board and USB cable*
- Figure 59: Main components*
- Figure 60: Verifying and uploading the joystick program code*
- Figure 61: Stopping code run on IntelliJ IDEA CE*
- Figure 62: Exiting IntelliJ IDEA CE application*
- Figure 63: Exiting Arduino IDE application*
- Figure 64: TYPE A disconnected from computer device*
- Figure 65: The 5 wires disconnected from the Arduino Uno Board*
- Figure 66: File -> Repair IDE function*
- Figure 67: TIME\_OUT = <insert number 1-2000>*
- Figure 68: Assembly view of Mechanical subsystem*
- Figure 69: 3 different 3D Models*
- Figure 70: Conceptual Design*
- Figure 71: Analog Read of the joystick electrical connections*
- Figure 72: Analog Read of the joystick code*
- Figure 73: The coordinates of the X and Y axis of the joystick.*

## List of Tables

|                                     |      |
|-------------------------------------|------|
| Table 1. Acronyms .....             | viii |
| Table 2. Glossary .....             | viii |
| Table 3. Referenced Documents ..... | 73   |





# List of Acronyms and Glossary

**Table 1. Acronyms**

| <b>Acronym</b> | <b>Definition</b>               |
|----------------|---------------------------------|
| TRMC           | Tremor Reducing Mouse Control   |
| <b>GRD</b>     | Ground on the Arduino Uno Board |
|                |                                 |
|                |                                 |
|                |                                 |

**Table 2. Glossary**

| <b>Term</b>      | <b>Acronym</b>    | <b>Definition</b>  |
|------------------|-------------------|--|
| Arduino          | Arduino Uno Board | Arduino is an open-source platform used for building electronics projects. Consists of both a physical programmable circuit board and a piece of software. |
| Arduino IDE      | -                 | Arduino IDE is a software application, program languages C and C++. Allows to preform codes  |
| IntelliJ IDEA CE | -                 | IntelliJ IDEA CE is a software application that allows to perform codes.   |

ix [Click here to enter text.](#)

# 1 Introduction

This User and Product Manual (UPM) provides individuals who experience tremors with the necessary information to effectively use the tremor reducing mouse control and for prototype documentation. This document assumes that the users have basic computer literacy and familiarity with using a computer mouse.

This document is composed of information regarding the proper use of the tremor reducing thumb joystick mouse control. It also consists of all the necessary steps to be taken should when any of the components of the system does not behave as predicted as well as the steps taken to fully create the system. The purpose of this document is to provide individuals with an overview of the tremor reducing mouse control and its functionalities. It is intended for individuals who experience tremors and are seeking a solution to improve their computer use. It will explain the process of its software system, electrical system, and mechanical system. A step-by-step guide to how all the different concepts were applied to create the tremor reducing mouse control system. The document will also cover how the software system can be customized to meet individual user needs.

The document will also go over the key aspects that make the tremor reducing mouse control accessible, inexpensive, and sustainable. It will provide individuals with a detailed description regarding the purpose of each different subcomponent of the system. It will also go over the different tests that were done on the system to ensure high performance of each component of the system.

This document also safeguards any personal information about the individual and protects their user identity. The document also abides by any laws regarding the security and privacy of users. It is also important to empathize that all troubleshooting activities described in this document are completely safe for the user to carry out. However, in case of an extraordinary malfunction, it is highly recommended to contact the provider to ensure the safety of the user. Confidentiality is always maintained during any interaction between the user and the provider.

## 2 Overview

At the beginning of the semester, our client expressed the need to design an accessible, inexpensive, sustainable and efficient tremor reducing mouse control. The client expressed a variety of needs, the most prominent ones were for the TRMC to be:

- Easy mouse operation with high precision of smooth movement and clicking feature.
- Have a toggle click software function.
- The cost of the project must be inexpensive with a cap of approximately 100\$.
- Durable with resistance to wear and tear and comfortable for extended periods of use.

It is important to solve this problem because the utilization of a computer mouse can pose a significant challenge for individuals with disabilities, particularly those who experience tremors. This impediment hinders an individual's capacity to control and operate the mouse with precision, thereby compromising their quality of life by restricting their ability to work, communicate, and socialize with others via the internet. The current solution to this problem is our prototype. To create a thumb joystick technology computer mouse device that interprets a code that incorporates features such as automatic steadying, adjustable sensitivity, and adaptive filtering to improve accuracy and ease of use for individuals with tremors. Our prototype is preminent because it is developed with a software code that has the capability of filtering features that can be implemented on the thumb joystick mouse device. Offering users, the ability to fine-tune the thumb joystick mouse movement to meet their specific needs and preferences. This strategy proves to be effective due to the reality that individuals who experience tremor have different backgrounds. Furthermore, the prototype also consists of electrical components that are connected with the thumb joystick and Arduino Borad that allows the software aspect to be properly read. Finally, the prototype also consists of a 3D printed case, where the thumb joystick and electrical wires will be secured and ensure the safety of the user.

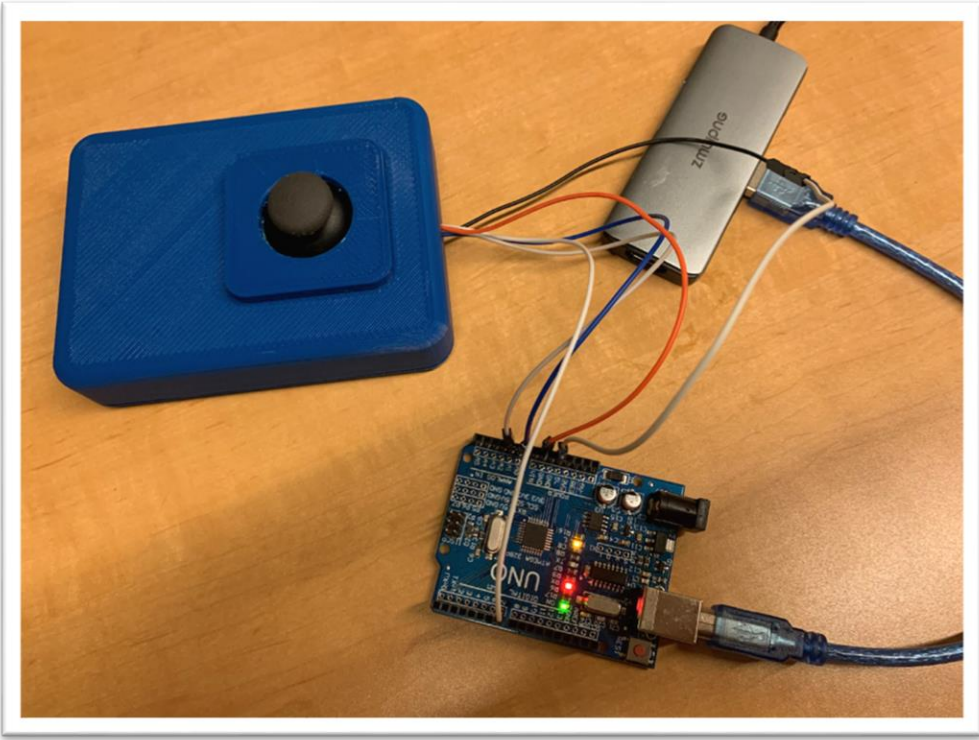
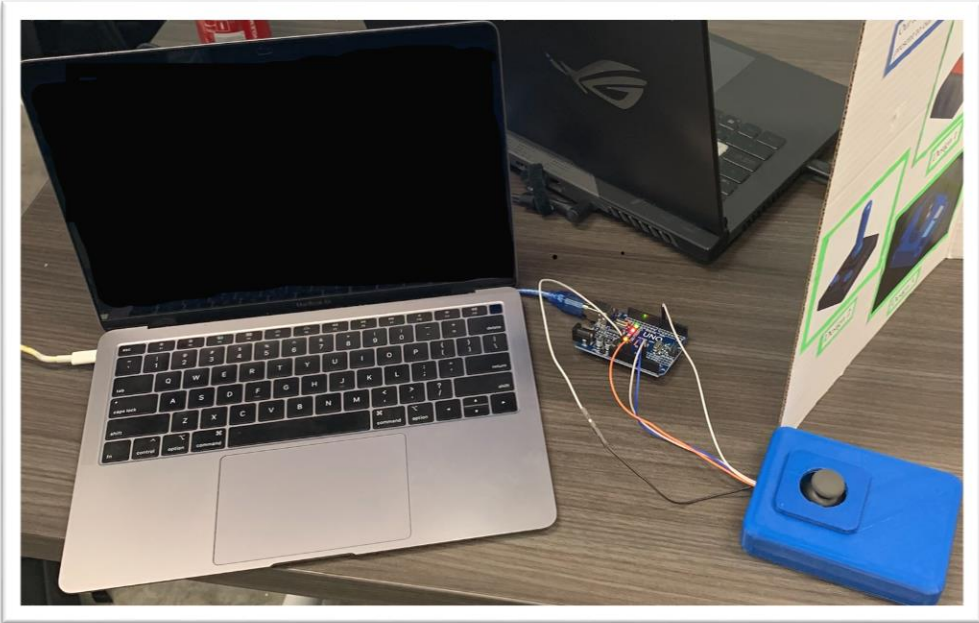
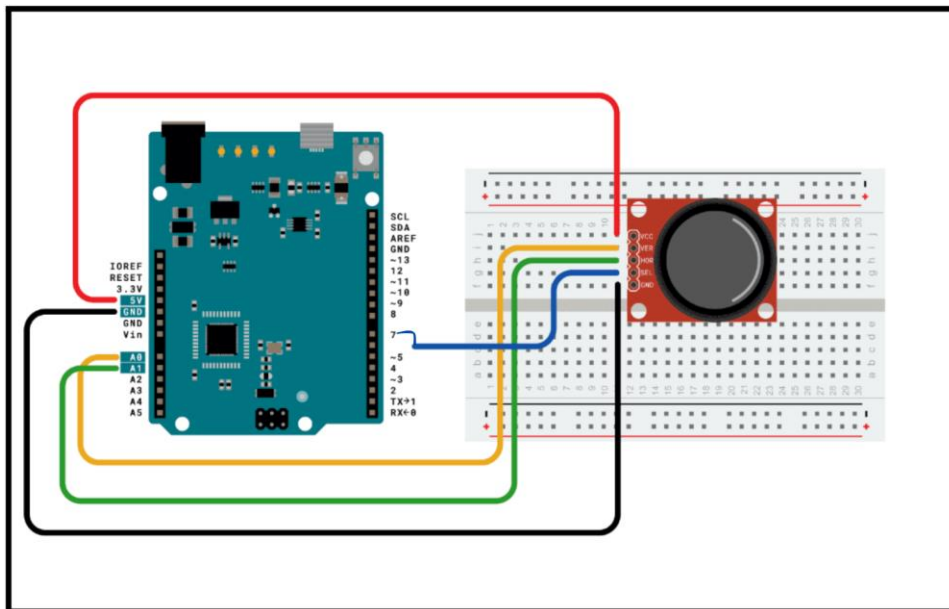


Figure 1: Final Prototype



*Figure 2: Final Prototype with all components together*

The final prototype consists of 1 thumb joystick, 1 Arduino Uno board and 1 USB Type A to USB Type B Cable – 3ft. As well as it consists of 5 electrical female-male wires that make the connection between the Arduino Uno board and the thumb joystick. Figure 3 demonstrates the connections that were implemented with the help of Thinkercad. Also, a computer device that the user wishes to use is necessary. To have the thumb joystick functioning, the connection of the USB cable to the computer device is required. Additionally, we have created a code program with the help of Arduino IDE and IntelliJ IDEA CE that allows the movement of the thumb joystick. 2 different codes are required that will be provided to the user. One code is for the Arduino IDE and the other is for IntelliJ IDEA CE. Furthermore, 3 different files are also required to download and upload on IntelliJ IDEA CE. 2 different libraries and 1 SDK file. To better understand, we have provided a flowchart Figure 4 that demonstrates how the software process of our code is intended to work.



*Figure 3: Electrical connections*

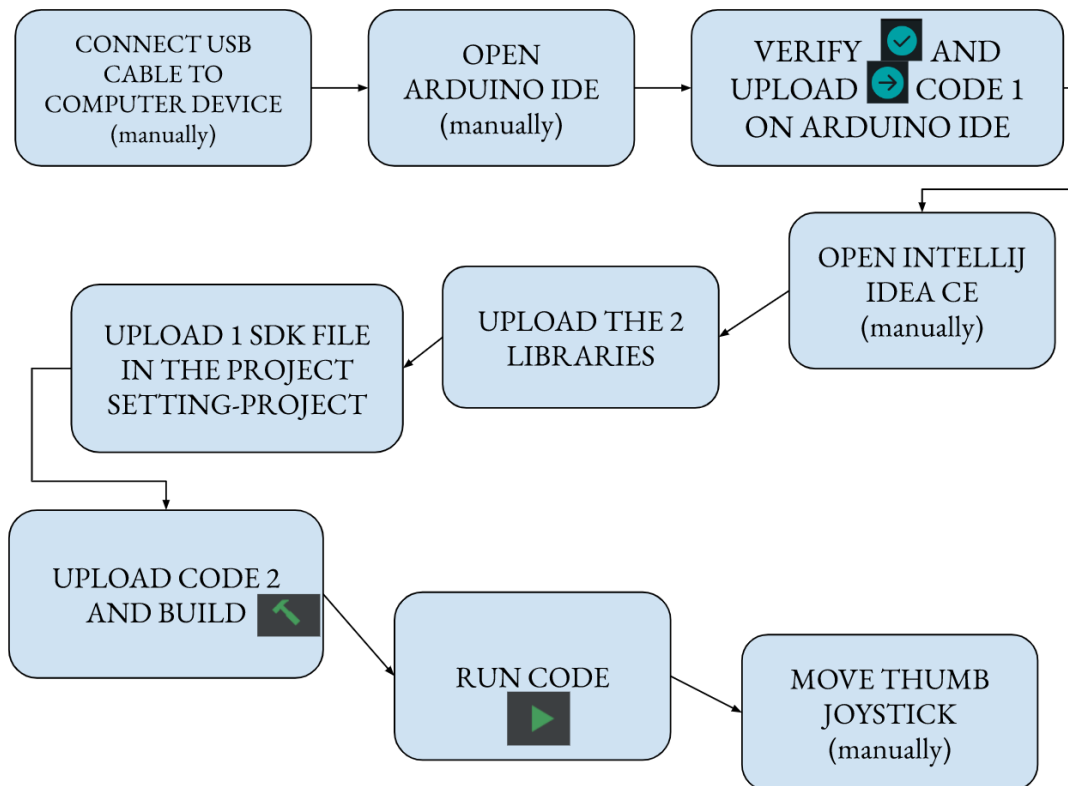


Figure 4: Flowchart

## 2.1 Conventions

In this document when the user is required to take action in order to complete a task it is indicated by a line beginning with the word ‘Action’.

## 2.2 Cautions & Warnings

When connecting the electrical wires to the Arduino Uno board ensure all connections are properly placed and secured in their corresponding label. This will ensure the wires do not fall out of place and allow the connections between the thumb joystick and Arduino Uno board to be properly read. Also, for users that are using a Windows operating computer device when plugging in the USB cable that is connected to the Arduino Uno board to the computer device be aware of the port information. The port information is found on Arduino IDE --> tools --> port, as Shown in

Figure 5. If the port information from the Arduino IDE does not match the port information on the IntelliJ IDEA CE code, necessary changes will be required.

→ ACTION: User change port information on the IntelliJ IDEA CE code. Line 20 as shown in Figure 6.

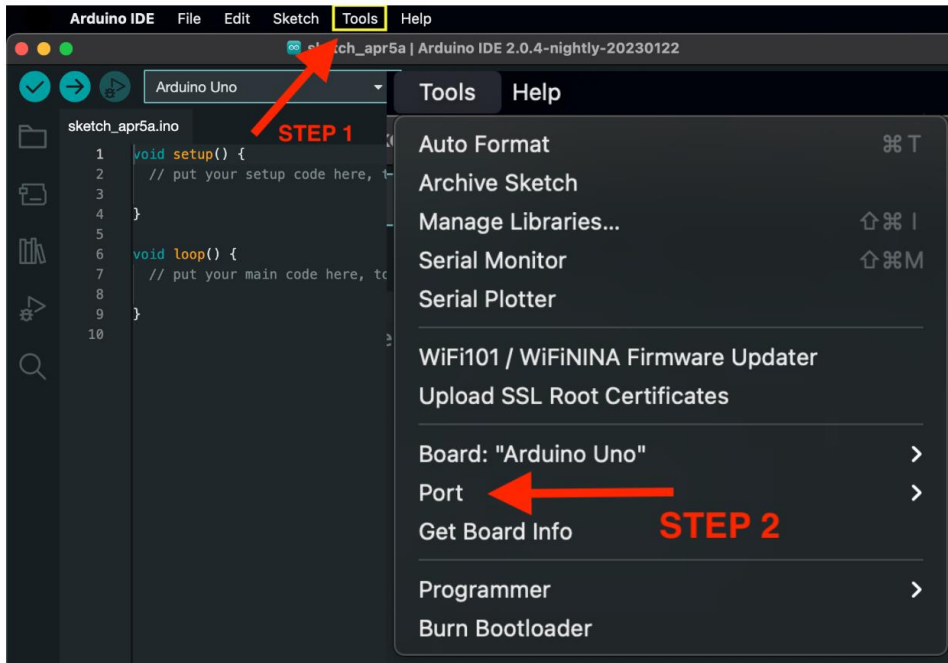


Figure 5: Finding port information on Arduino IDE



```
13 ▶ public class Mouse implements SerialPortEventListener {
    9 usages
14     SerialPort serialPort;
15     /** The port we're normally going to use. */
    1 usage
16     private static final String PORT_NAMES[] = {
17         "/dev/tty.usbserial-A9007UX1", // Mac OS X
18         "/dev/ttyACM0", // Raspberry Pi
19         "/dev/ttyUSB0", // Linux
20         "COM4", // Windows*****
21     };
22     /**
```

Figure 6: LINE 20 from the IntelliJ IDEA CE code

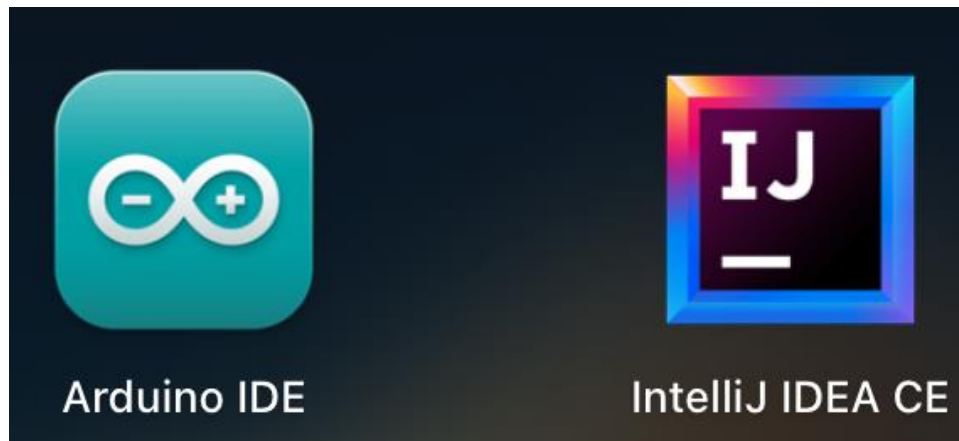
Another important aspect to note is that the code of IntelliJ IDEA CE presented in this document may not be consistent and may require troubleshooting the system at times. However, not to worry, as this document will provide a step-by-step guide to mitigate any potential issues that may arise.

### 3 Getting started

#### ACTION

##### STEP 1

→ Download **Arduino IDE** and **IntelliJ IDEA CE**



*Figure 7: Arduino IDE and IntelliJ IDEA CE Applications*

<https://www.arduino.cc/en/software>

<https://www.jetbrains.com/idea/download/#section=mac>

##### STEP 2

8

→ Connect the 5 electrical wires to that are attached to the thumb joystick to the Arduino Uno board as shown in Figure 8

- White wire = 7
- Blue wire = A0
- Gray wire = A1
- Orange wire = GRD
- Black & white wire = 5V

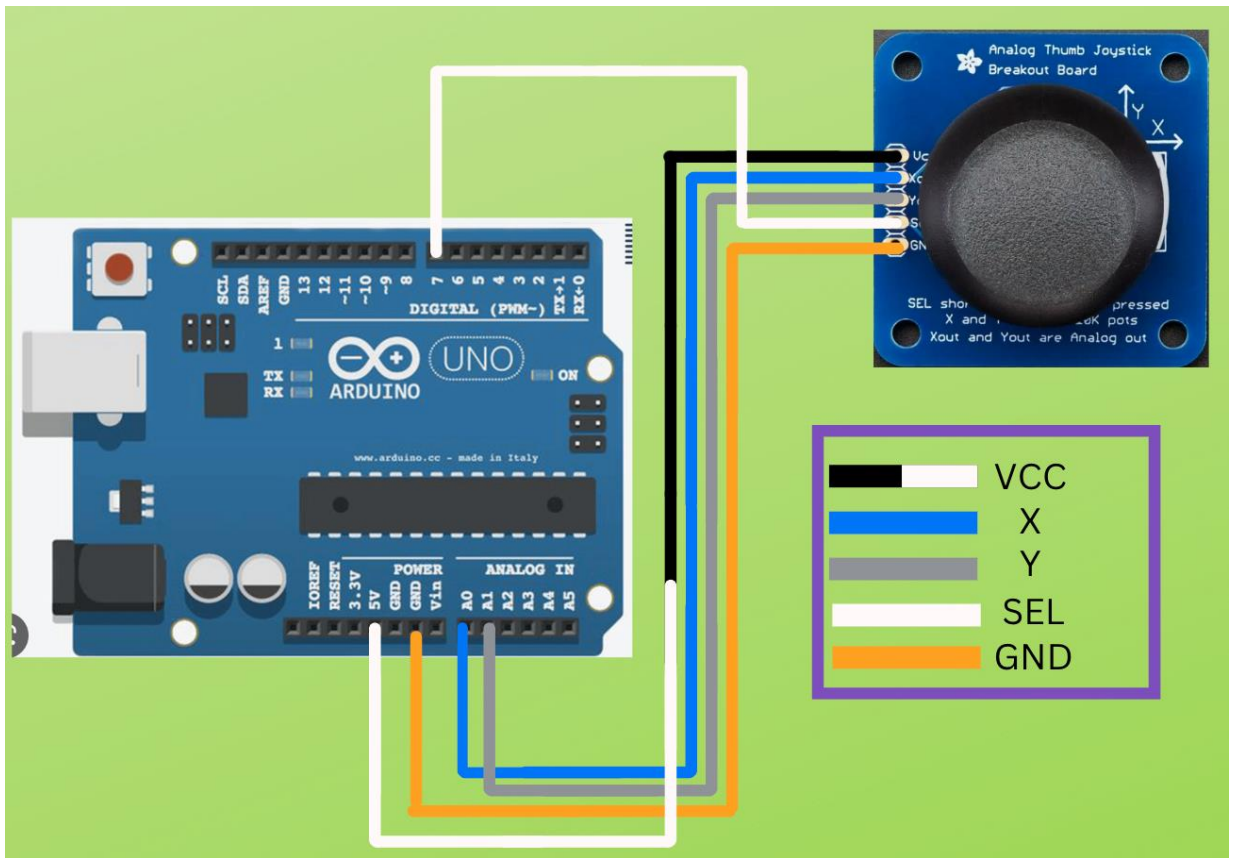
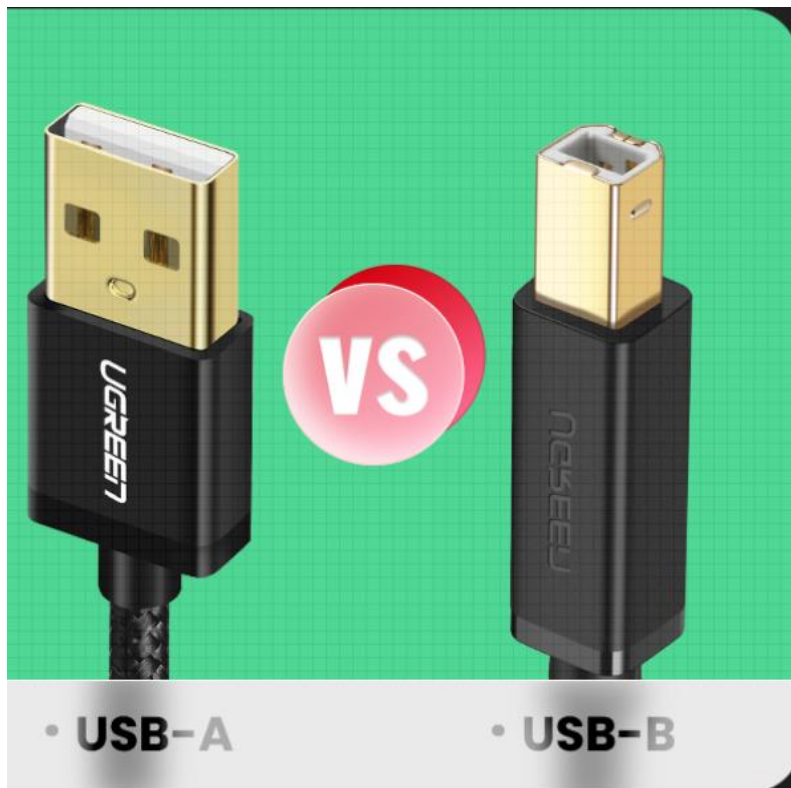


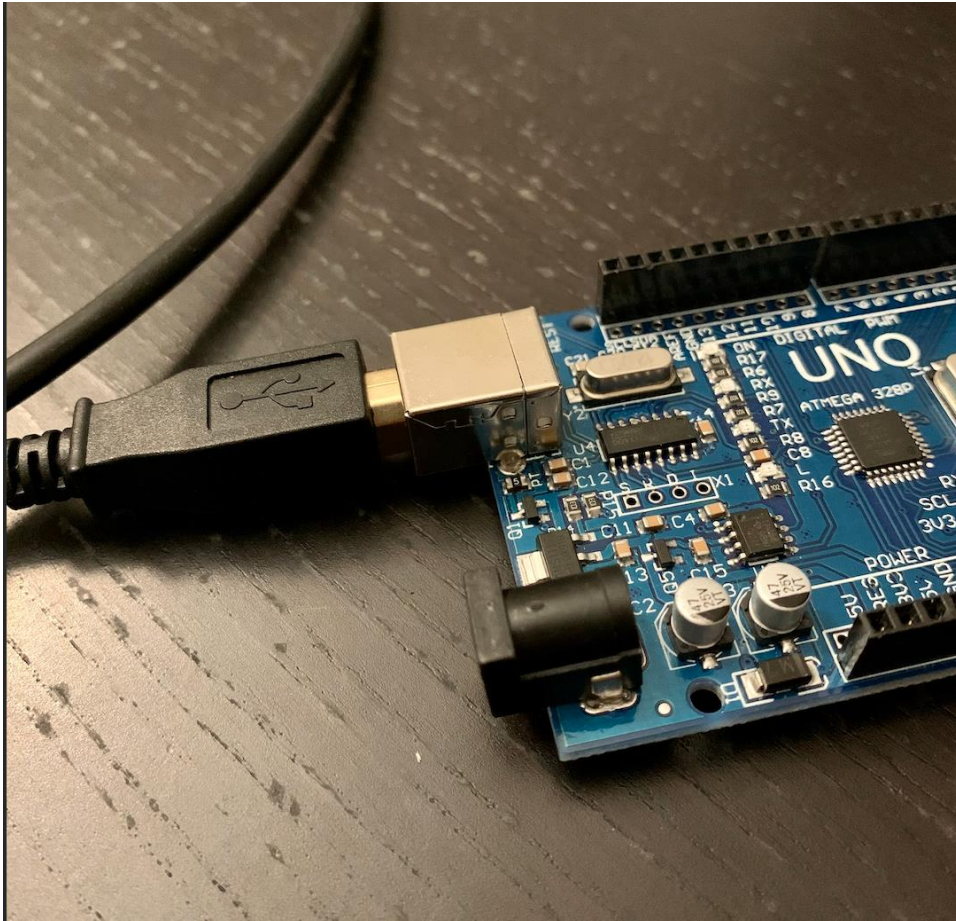
Figure 8: Electrical connections to the Arduino Uno Board

### STEP 3

- Properly identify Type A and Type B USB, as shown in Figure 9.
- Connect Type B to the Arduino Uno board as shown in Figure 10.
- Connect Type A to your computer device, Figure 11.
- A green, red and yellow light should appear on the Arduino Uno Board. Indicating that it is ON and that it has been properly connected. Figure 12.



*Figure 9: Type A, Type B USB*



*Figure 10: Type B USB connection to the Arduino Uno Board*



*Figure 11: Type A connection to computer device*



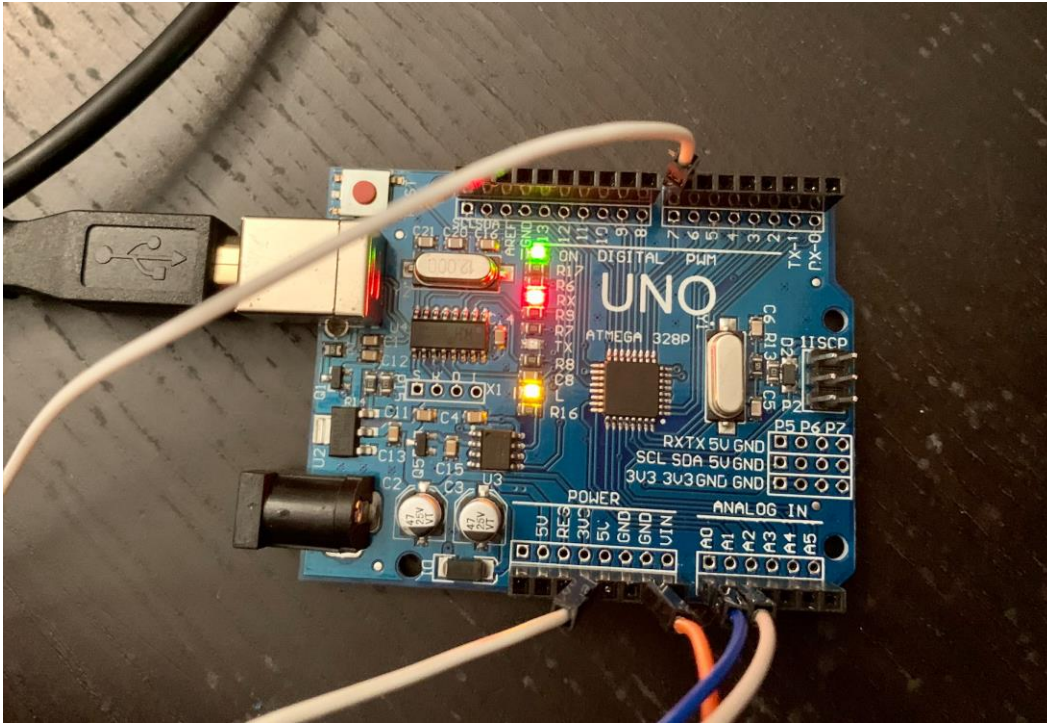


Figure 12: Arduino Uno Light meaning it is ON

#### STEP 4

- Open Arduino IDE on your computer device and click on New Sketch as shown in Figure 13.
- If New Sketch does not appear on your screen, click on the file icon on the left side as shown in Figure 14. Then click on New Sketch.

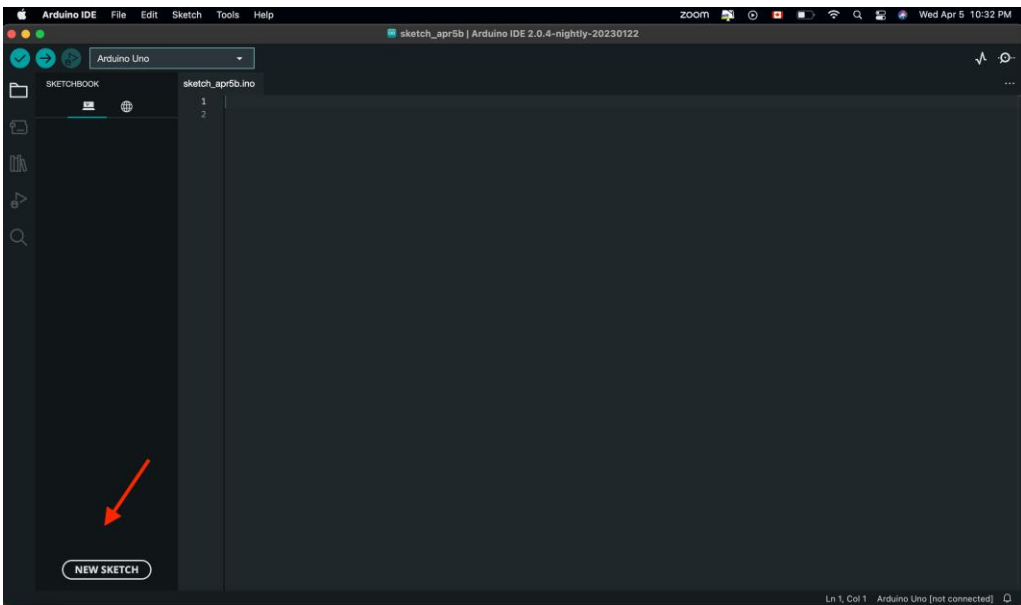


Figure 13: New Sketch creation

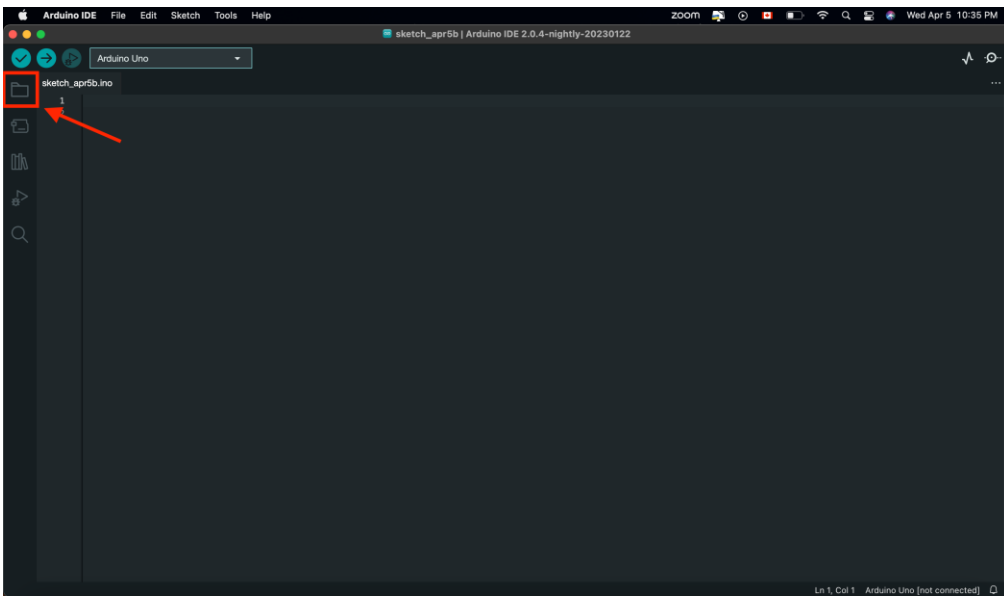


Figure 14: File icon to create a New Sketch if New Sketch does not appear from the start



## **STEP 5**

### **ENSURING ARDUINO UNO IS PROPERLY CONNECTED TO COMPUTER DEVICE**

→ **Ensure the Board information is correct, Figure 15.**

**Click Tool --> Board:-->Arduino AVR Boards--> Arduino Uno**

→ **Ensure the port information has been properly connected. At the bottom right side of the Arduino IDE Tab, you will find the port information, as shown in Figure 16.**

→ **Figure 17 demonstrates if the connection has not been properly made. You can disconnect and connect again the Type A USB to your computer device.**

→ **Then go to Tool ---> Port --> and click the USB connection that corresponds to your computer device, as shown in Figure 18.**

- **Note: For users that are using a windows operating device, port device will say COM, Figure 19.**

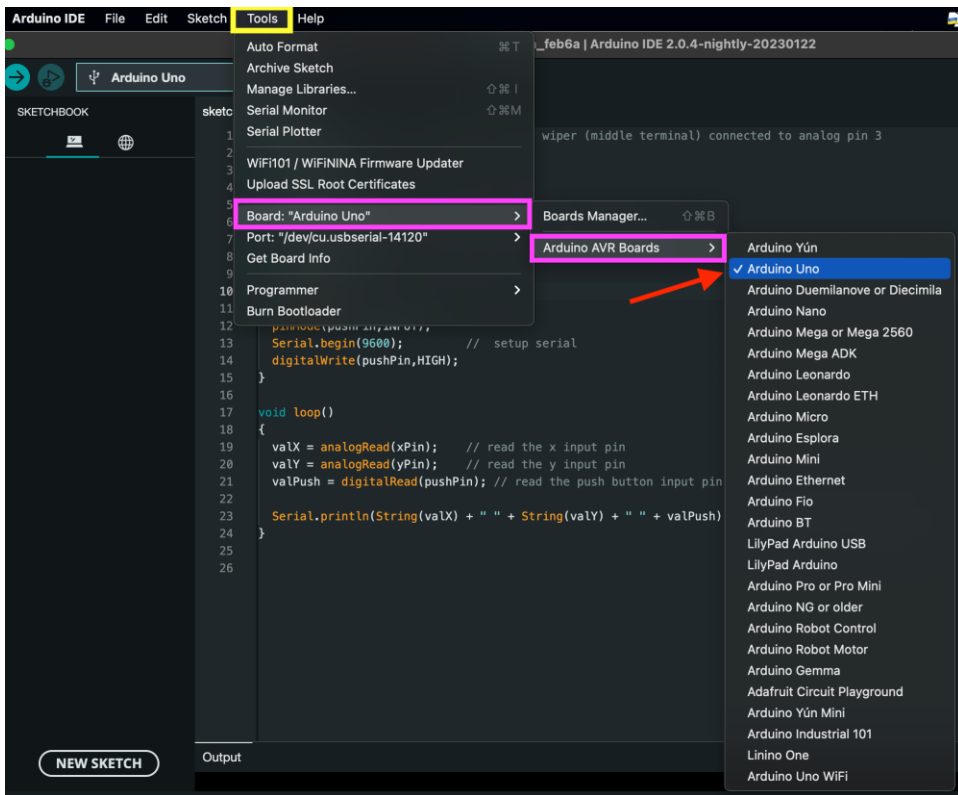


Figure 15: Board information



Figure 16: Port information – connected properly



Figure 17: Port information – not connected

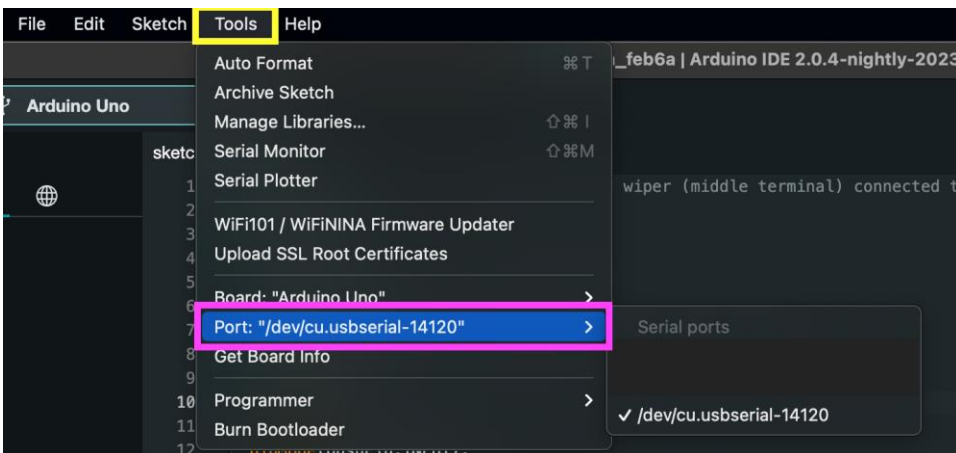


Figure 18: Manually selecting port information

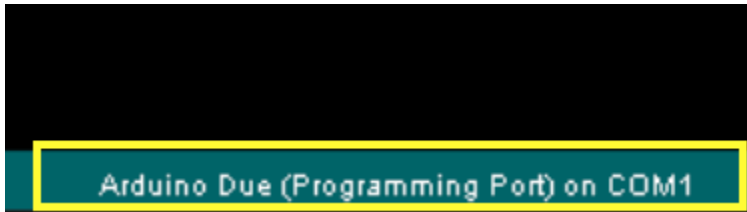


Figure 19: Windows operating system, COM port information

## SETTING UP THE SOFTWARE

### STEP 6

- Copy and upload the Joystick Program code to the Arduino IDE in the blank space provided as shown in Figure 20. If any text appeared when creating a New Sketch, simply delete.
- Figure 21 is what your screen should like it.

### Joystick Program code

```

int pushPin = 7;          // potentiometer wiper (middle terminal) connected to analog pin 3
int xPin                 = 0;
int yPin                 = 1;
int xMove                = 0;
int yMove                = 0;
// outside leads to ground and +5V
int valPush = HIGH;     // variable to store the value read
int valX          = 0;
int valY          = 0;
void setup()
{
  pinMode(pushPin,INPUT);
  Serial.begin(9600);    // setup serial
  digitalWrite(pushPin,HIGH);
}

```

18

```

void                                                    loop()
{
  valX    =    analogRead(xPin);                      // read the x input pin
  valY    =    analogRead(yPin);                      // read the y input pin
  valPush =    digitalRead(pushPin);                 // read the push button input pin

  Serial.println(String(valX) + " " + String(valY) + " " + valPush); //output to Java program
}

```

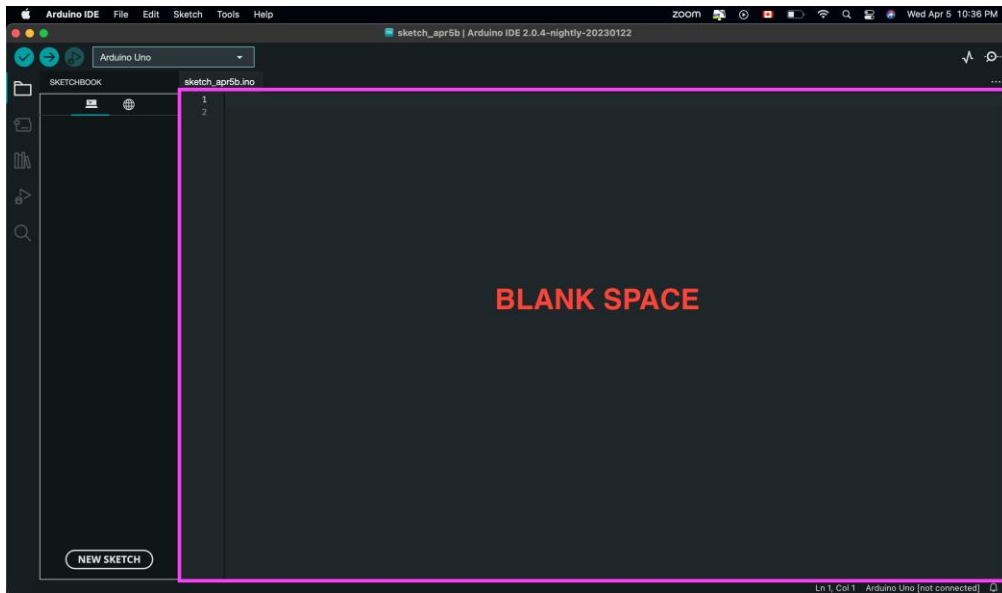


Figure 20: Pasting Code in the blank space

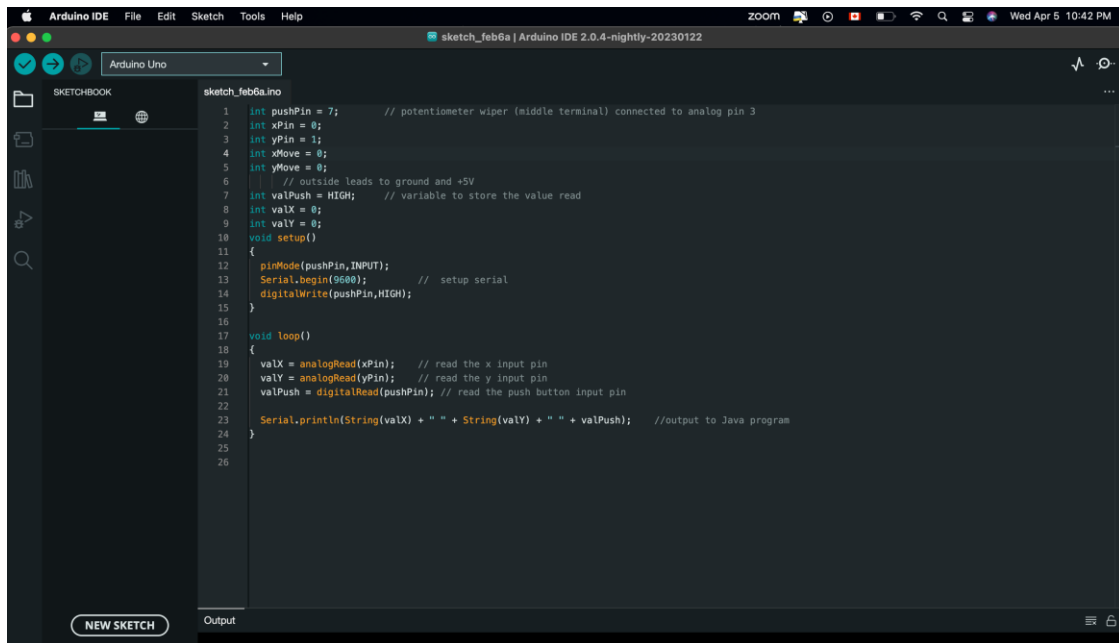


Figure 21: Uploaded the Joystick Program code to the Arduino IDE

## STEP 7

→ Download the 2 mainfiles required JNA and RXTX. Links are provided below:

- ◆ [jna-3.2.7-sources.zip](#)
- ◆ [mfz-rxtx-2.2-20081207-win-x86.zip](#)

→ Figure 22 is how the files should appear in your downloads file folder on your computer device.

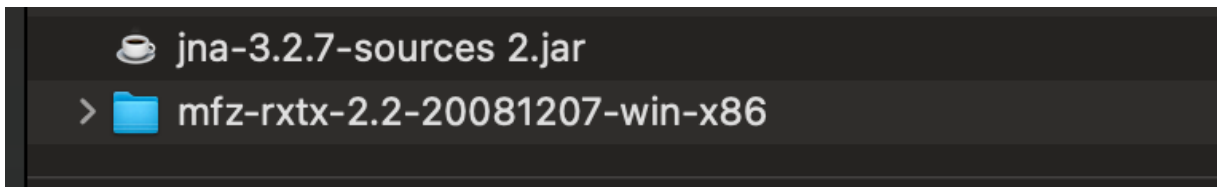


Figure 22: The 2 main files required to download

## STEP 8

→ Download the Java SE Development Kit 8u251 by using the link below. Click the download button that corresponds to the device operating system that the user is using (MacOS, Windows, etc.), as shown in Figure 23.

**Note:** It is required to create an oracle account to be able to download the file.

- <https://www.oracle.com/ca-en/java/technologies/javase/javase8u211-later-archive-downloads.html>

**Java SE Development Kit 8u251**

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

[JDK 8u251 checksum](#)

| Product / File Description          | File Size | Download  |
|-------------------------------------|-----------|---|
| Linux ARM 32 Hard Float ABI         | 72.87 MB  | <a href="#">jdk-8u251-linux-arm32-vfp-hflt.tar.gz</a> |
| Linux ARM 64 Hard Float ABI         | 69.77 MB  | <a href="#">jdk-8u251-linux-arm64-vfp-hflt.tar.gz</a> |
| Linux x86 RPM Package               | 171.71 MB | <a href="#">jdk-8u251-linux-i586.rpm</a>              |
| Linux x86 Compressed Archive        | 186.6 MB  | <a href="#">jdk-8u251-linux-i586.tar.gz</a>           |
| Linux x64 RPM Package               | 171.16 MB | <a href="#">jdk-8u251-linux-x64.rpm</a>               |
| Linux x64 Compressed Archive        | 186.09 MB | <a href="#">jdk-8u251-linux-x64.tar.gz</a>            |
| macOS x64                           | 254.78 MB | <a href="#">jdk-8u251-macosx-x64.dmg</a>              |
| Solaris SPARC 64-bit (SVR4 package) | 125.19 MB | <a href="#">jdk-8u251-solaris-sparcv9.tar.Z</a>       |
| Solaris SPARC 64-bit                | 88.16 MB  | <a href="#">jdk-8u251-solaris-sparcv9.tar.gz</a>      |
| Solaris x64 (SVR4 package)          | 133.64 MB | <a href="#">jdk-8u251-solaris-x64.tar.Z</a>           |
| Solaris x64                         | 91.9 MB   | <a href="#">jdk-8u251-solaris-x64.tar.gz</a>          |
| Windows x86                         | 201.17 MB | <a href="#">jdk-8u251-windows-i586.exe</a>            |
| Windows x64                         | 211.54 MB | <a href="#">jdk-8u251-windows-x64.exe</a>             |

*Figure 23: Java SE Development Kit 8u251 File*

## STEP 9

→ **Open** IntelliJ IDEA CE on your computer device and create a new project.

Go to file --> new --> project, as shown in Figure 24.

→ Figure 25 is the tab that will show up when creating a new project. Name your file.



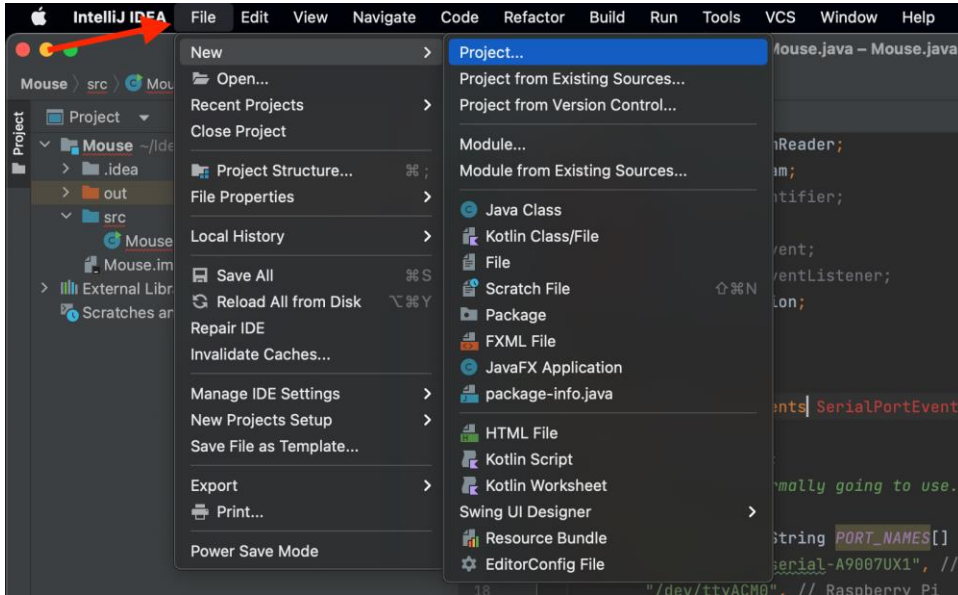


Figure 24: Creating new project on IntelliJ IDEA CE

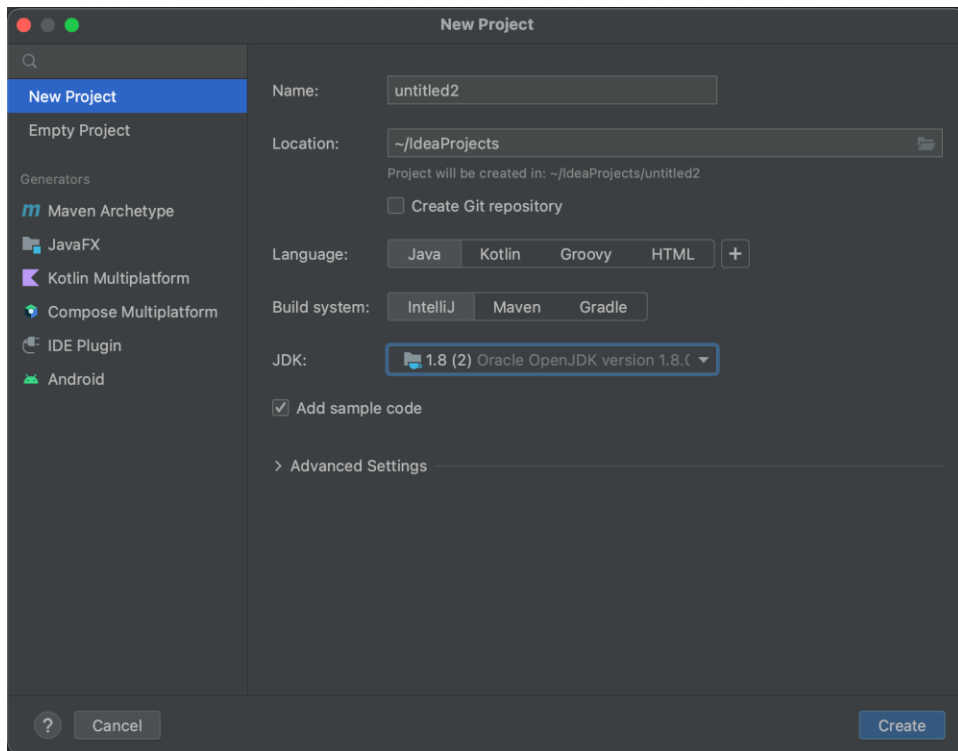


Figure 25: New project Tab

## STEP 10

- Now you will add the Java SE Development Kit 8u251 by clicking on the JDK section and (Add JDK), as shown in Figure 26.
- It will bring you to your own downloaded files folder on your computer, find the Java SE Development Kit 8u251. Once you have found the file, click on it once and then click the open button, as shown in Figure 27.
- You will click the JDK button again and select the file **(1.8 Oracle OpenJDK version 1.8.0\_251)**, as shown in Figure 28.
- Now you will create the project by clicking on Create, as shown in Figure 29.

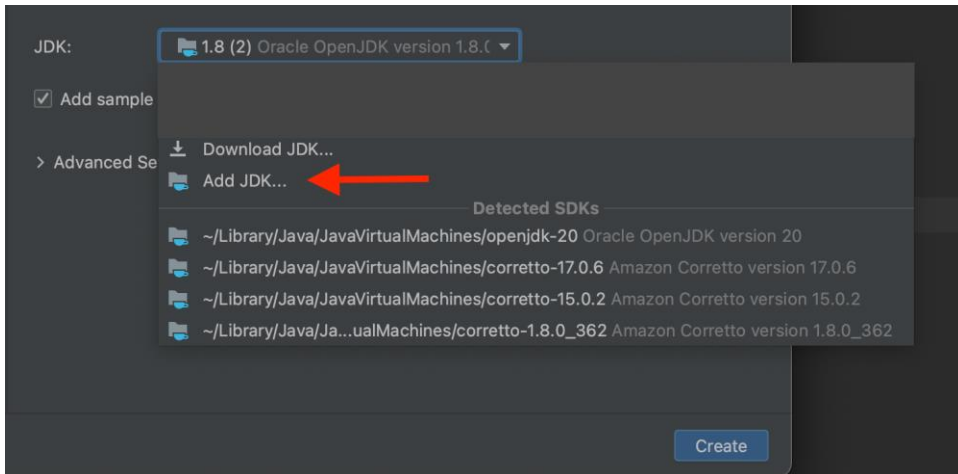


Figure 26: Adding JDK file

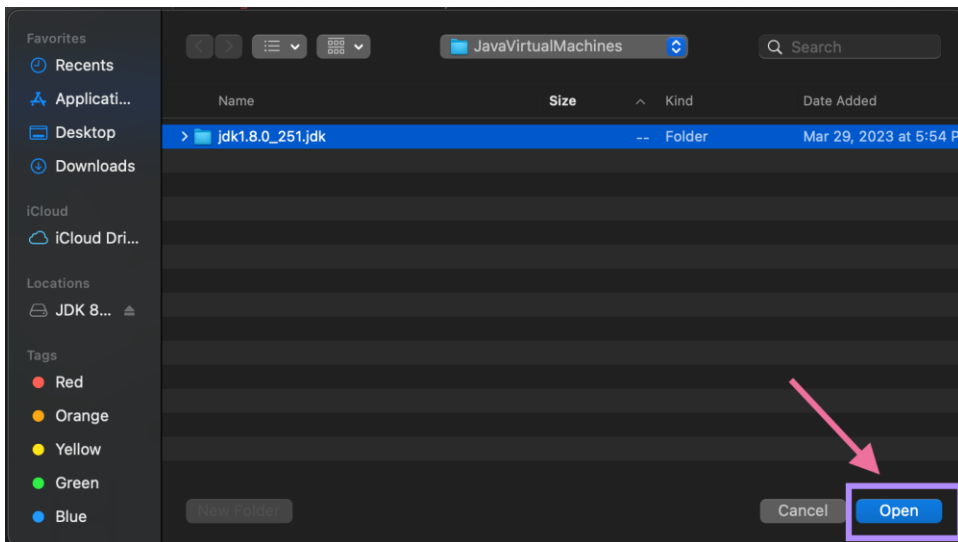


Figure 27: Finding the Java SE Development Kit 8u251 File



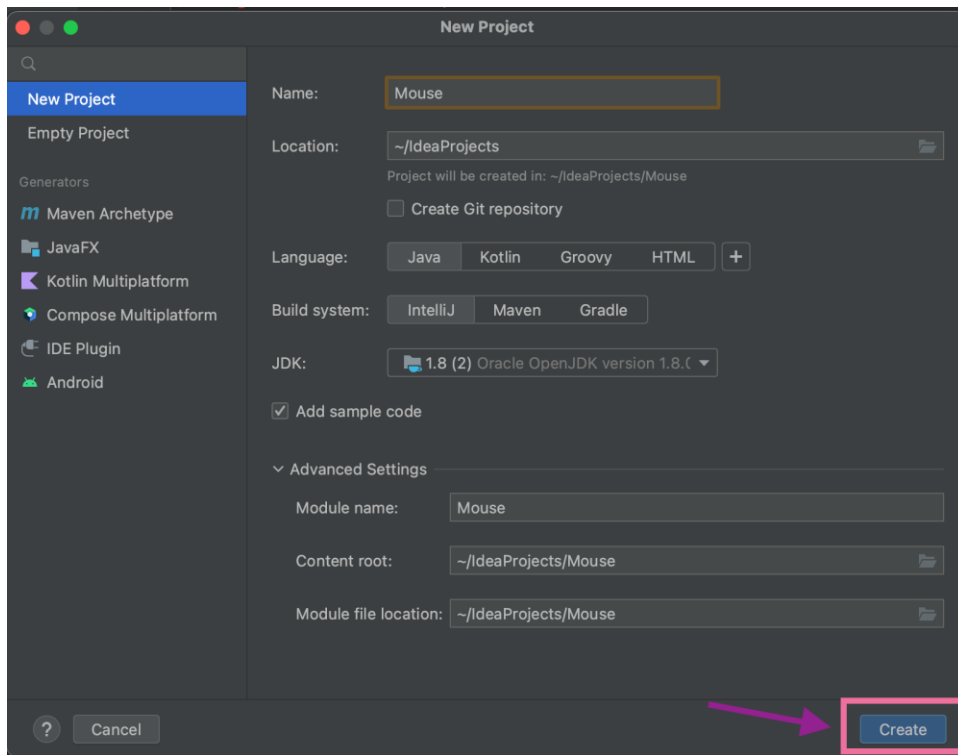


Figure 29: Creating New Project

## STEP 11

- Add the 2 libraries that were downloaded from step 7 to the new project you have creating from step 9&10.
- Go to Files --> Project Structure as shown in Figure 30.
- Go to the Libraries tab as shown in Figure 31.
- Click on the (+) icon and then click on Java to add your 2 library files, as shown in Figure 32.

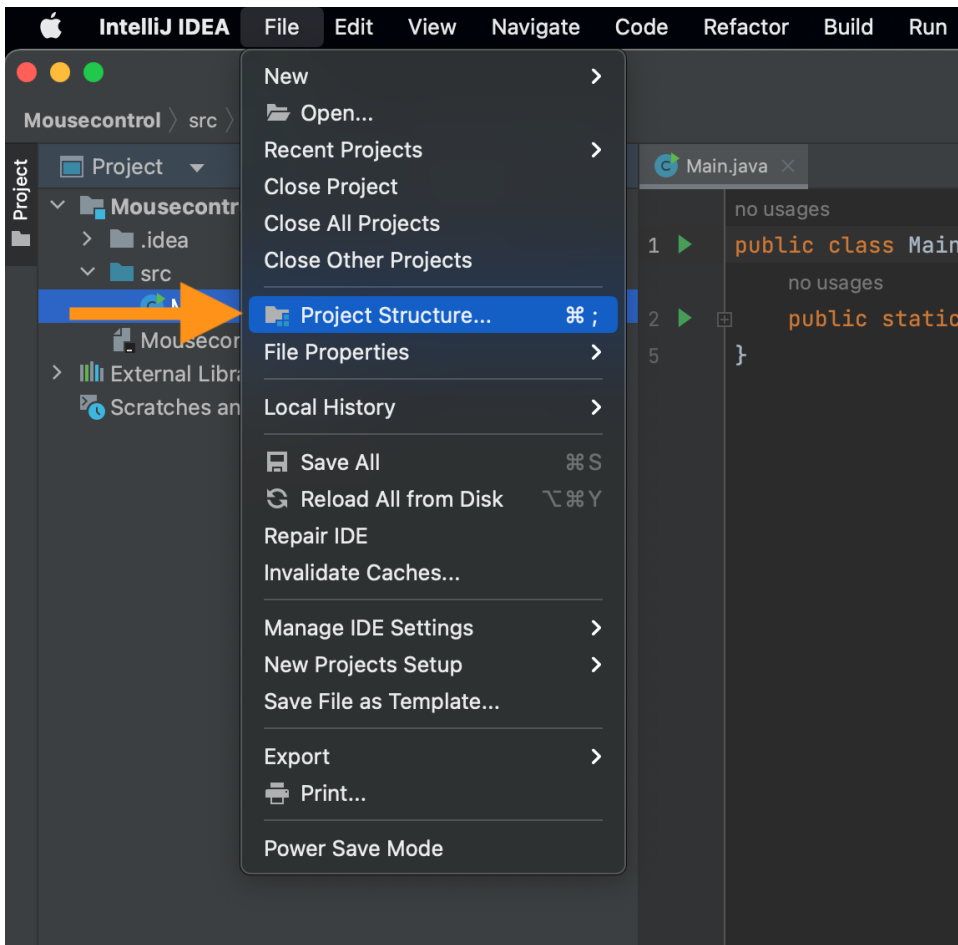
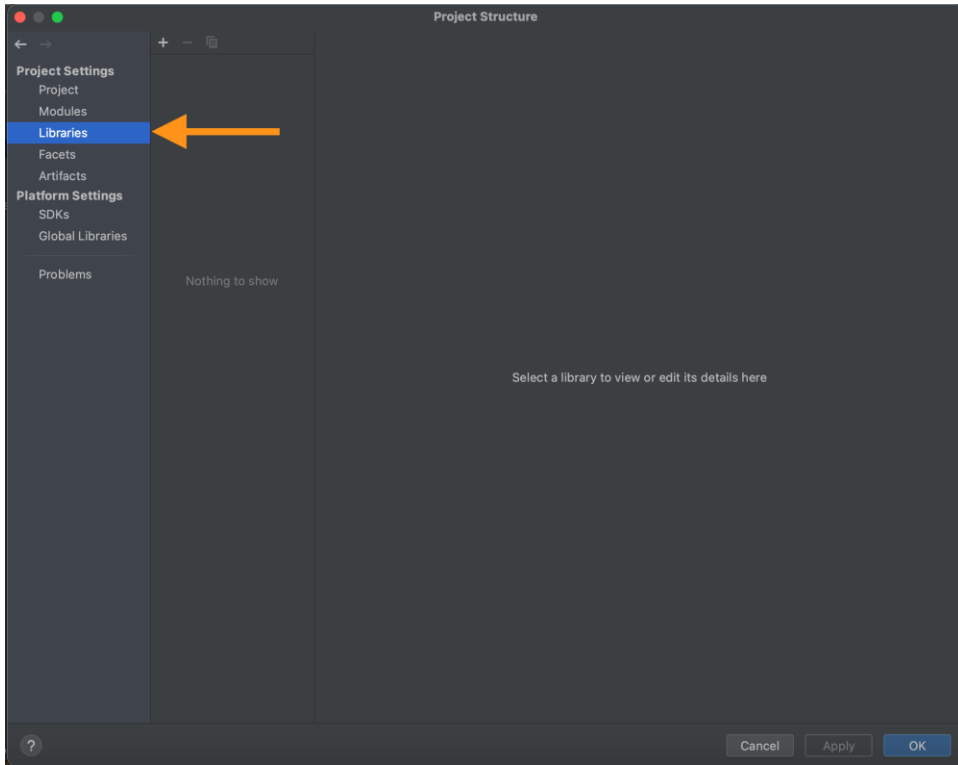
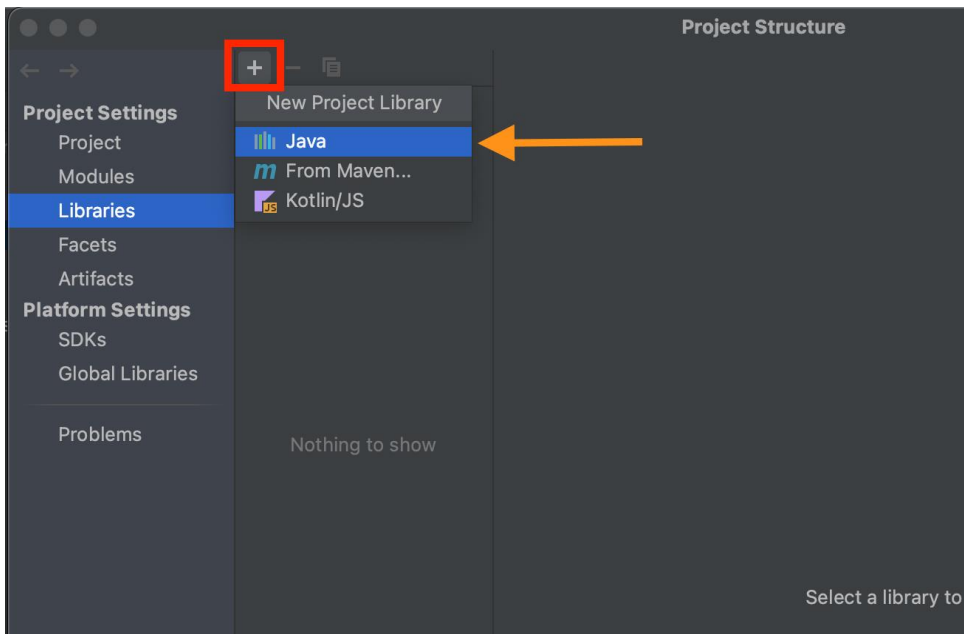


Figure 30: Project Structure Tab



*Figure 31: Libraries Tab*



*Figure 32: Getting setup to add the 2 libraries*

## **STEP 12**

- After clicking on the Java icon that was shown in step 11, Figure 32. Your download files document folder on your computer will pop up.
- Find the files that were downloaded from step 7, as shown in Figure 33.



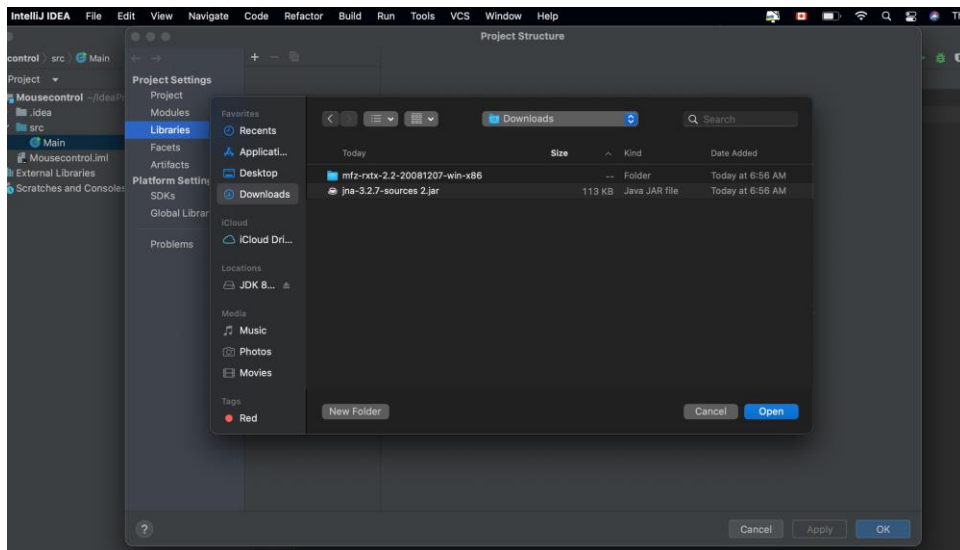
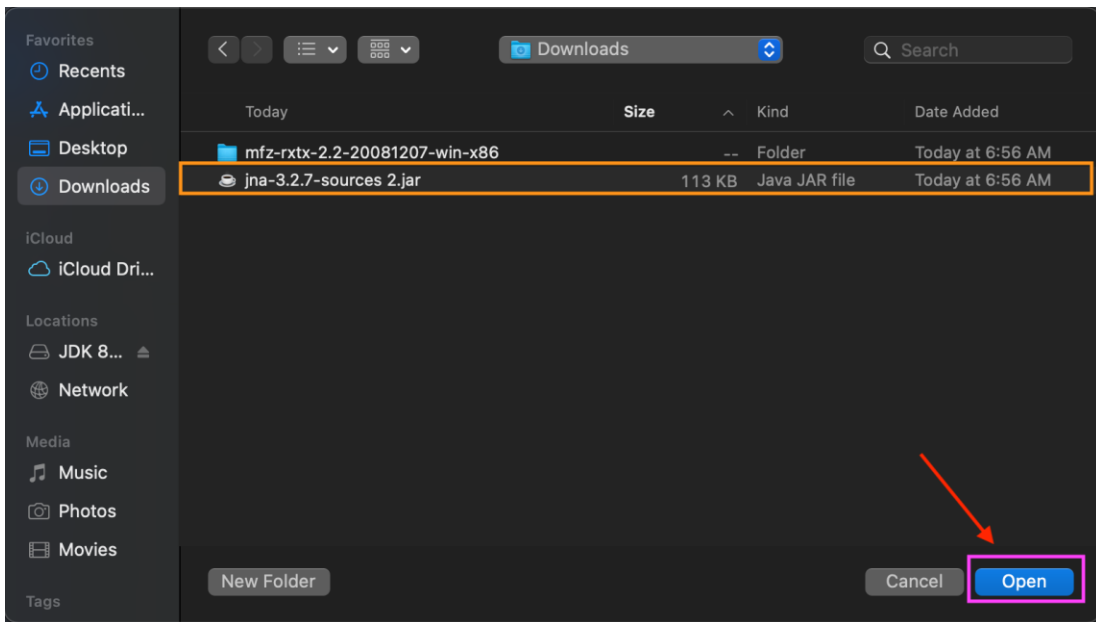


Figure 33: Finding the 2 library files in your download files folder

### STEP 13

- First select the **ina-3.2.7-sources 2.jar** and click on the open button, as shown in Figure 34.
- A Choose Modules Tab will appear simply click on **OK**. (Figure 35)
- Figure 36 is what the Project Structure Tab should look like.



*Figure 34: Selecting the jna-3.2.7-sources 2 jar file*

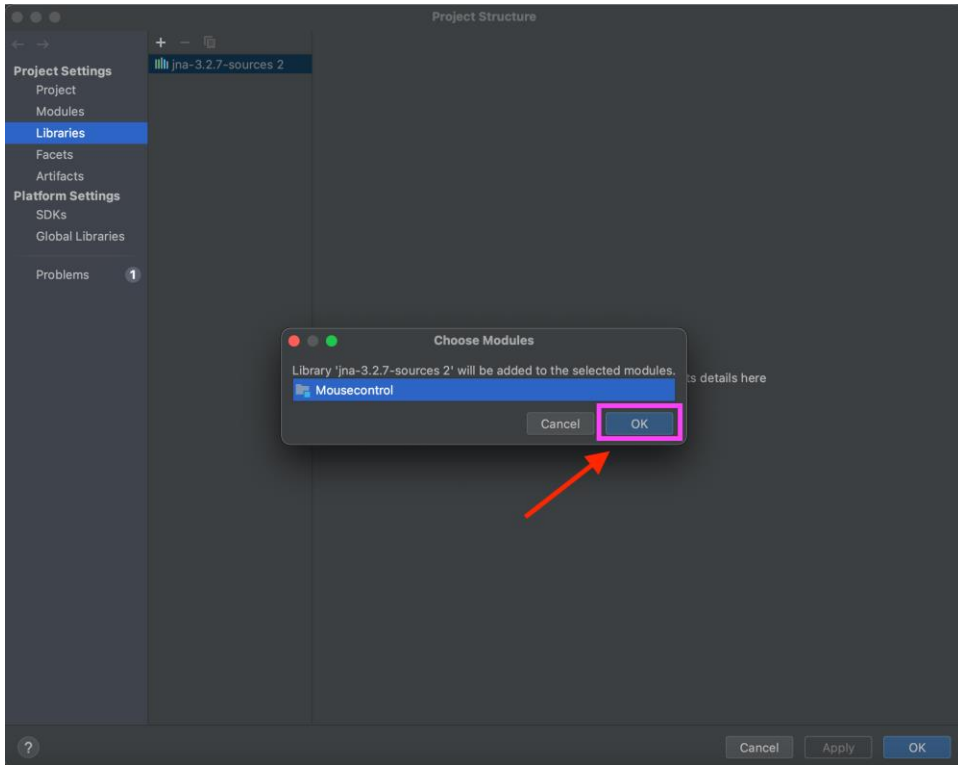


Figure 35: Adding the *jna-3.2.7-sources 2 jar file*

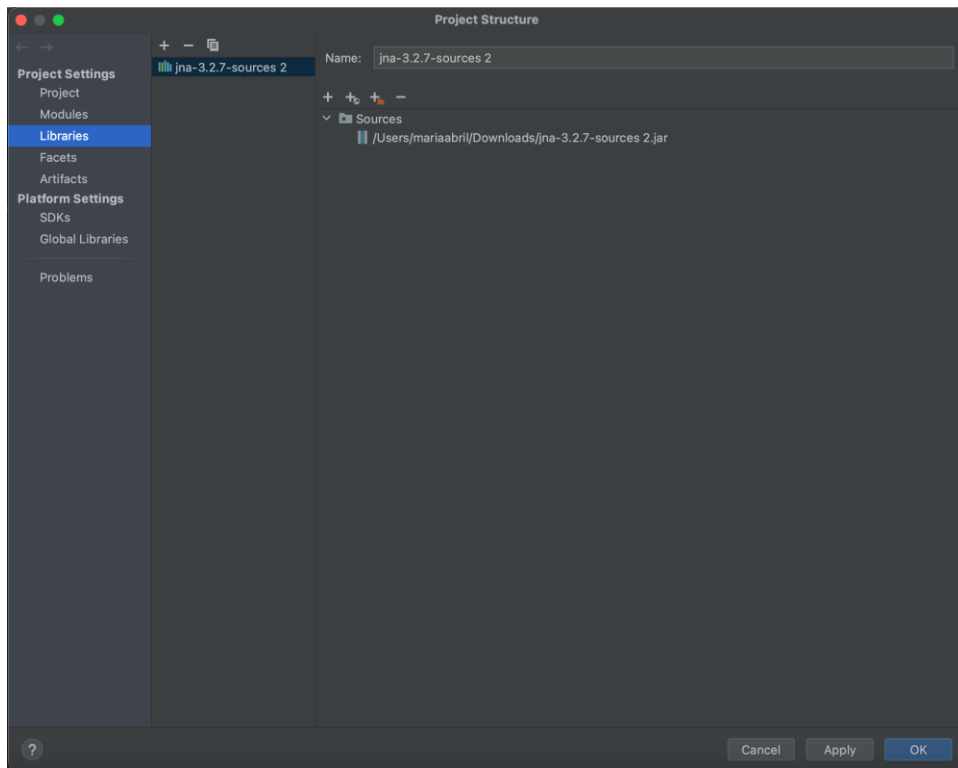


Figure 36: The jna-3.2.7-sources 2 jar file has been added

## STEP 14

- Now the user will add the **RXTXcomm.jar** file
- Repeat Step 11, Figure 32 and Step 12
  - Click (+) icon --> Java ---> Find the files the user downloaded from step 7, as shown in Figure 37.
- Double click the file **mfz-rxtx-2.2-20081207-win-x86**, Figure 38.
- Figure 39 is the new tab that will appear, click the **RXTXcomm.jar** file and click open.
- Figure 40, shows a Choose Modules Tab that will appear, simply click **OK**.

→ **Figure 41** is what your Project Structure Tab should look like with the 2 main libraries added.

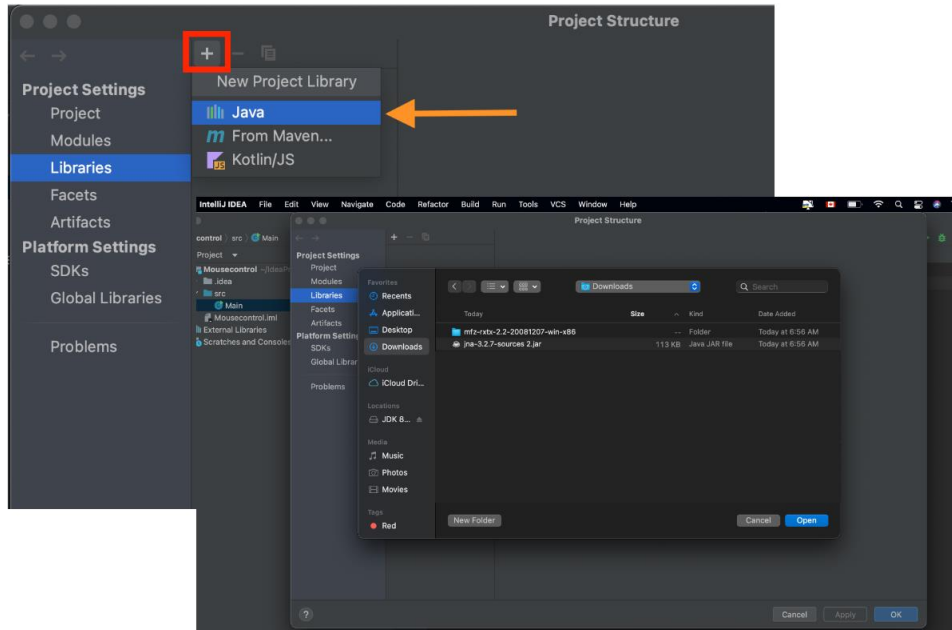


Figure 37: Finding the **mfz-rxtx-2.2-20081207-win-x86** file

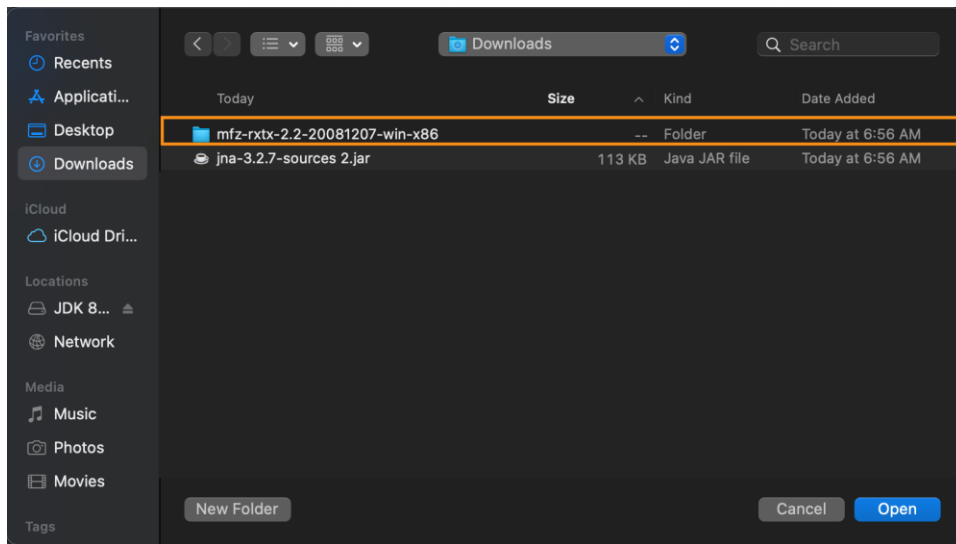


Figure 38: Double click the **mfz-rxtx-2.2-20081207-win-x86** file

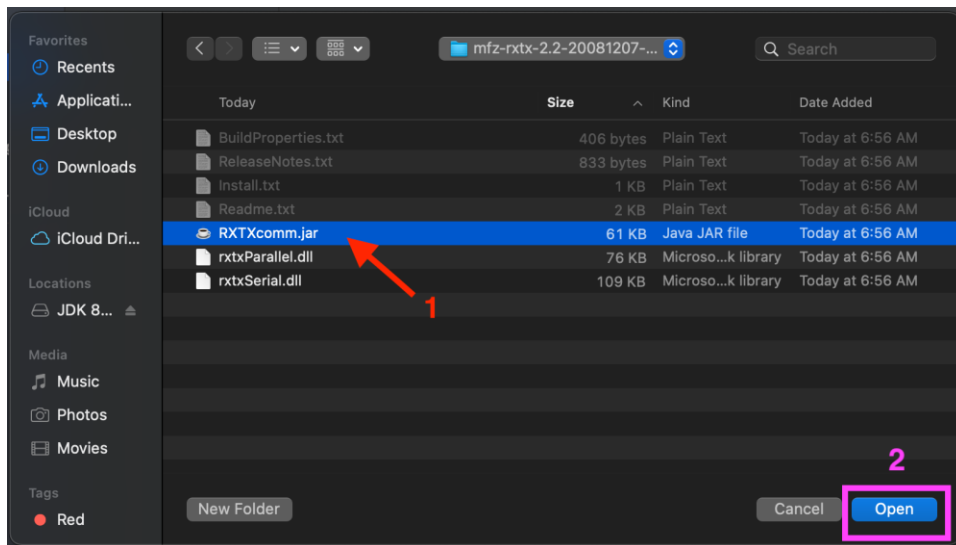
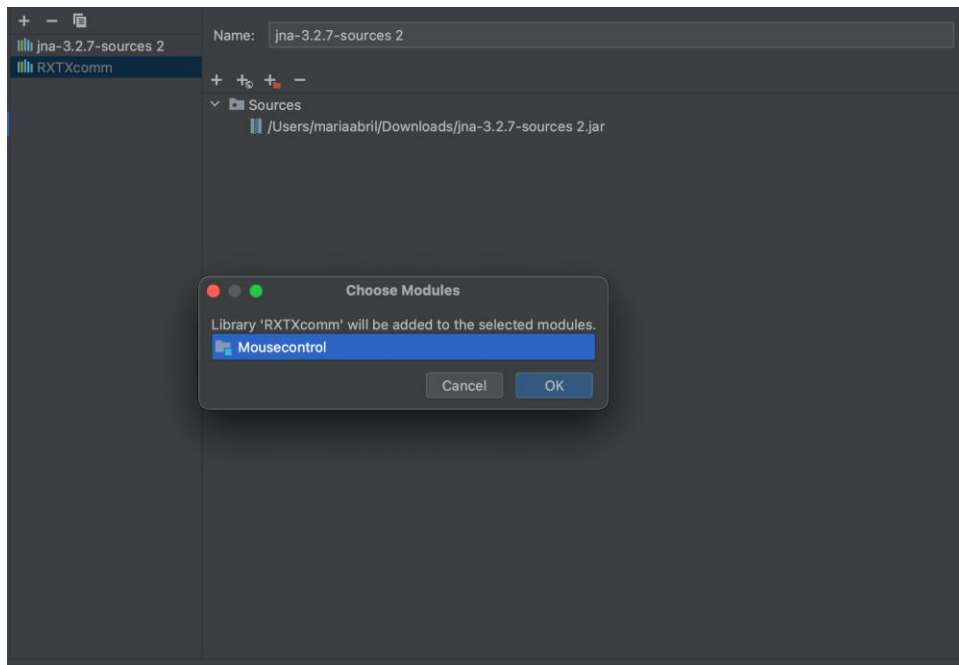


Figure 39: Adding the **RXTXcomm.jar** file



*Figure 40: The Choose Modules Tab*

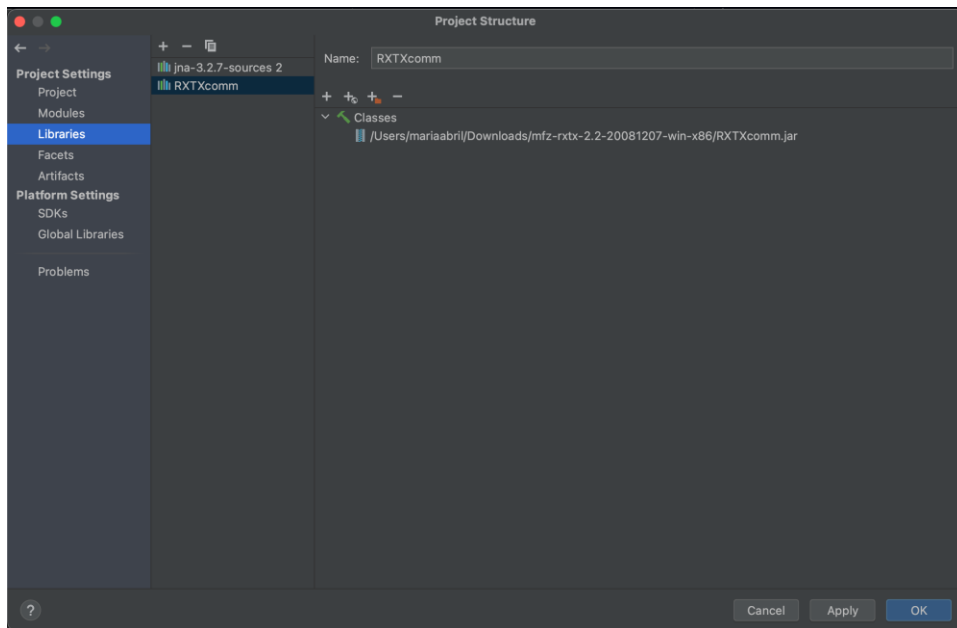


Figure 41: Updated Project Structure tab with the 2 main libraries added

## STEP 15

- Make sure the user is on the **RXTXcomm** Tab and click on the first (+) icon that is slightly to the right, as shown in Figure 42.
- Double click the file **mfz-rxtx-2.2-20081207-win-x86**, as shown in Figure 43.
- Figure 44 is the new tab that will appear, click the **rxtxSerial.dll** file and click open.
- Figure 45 is what your fully updated Project Structure Tab should look like with the 2 main libraries added and the 1 Native Library Location.



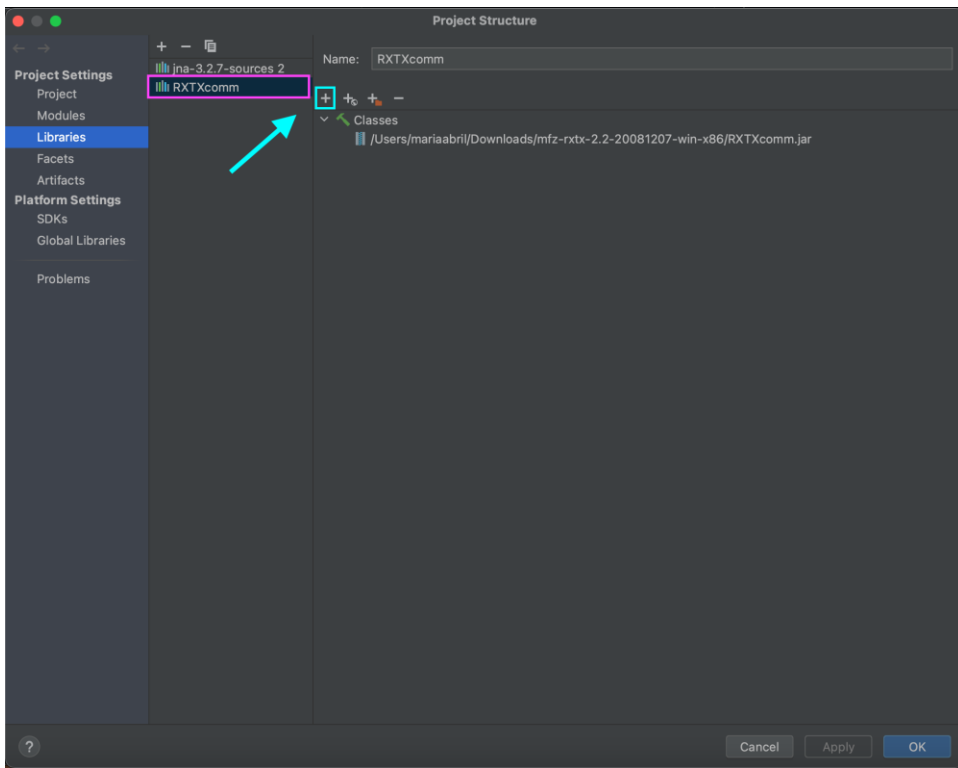


Figure 42: The RXTXcomm Tab

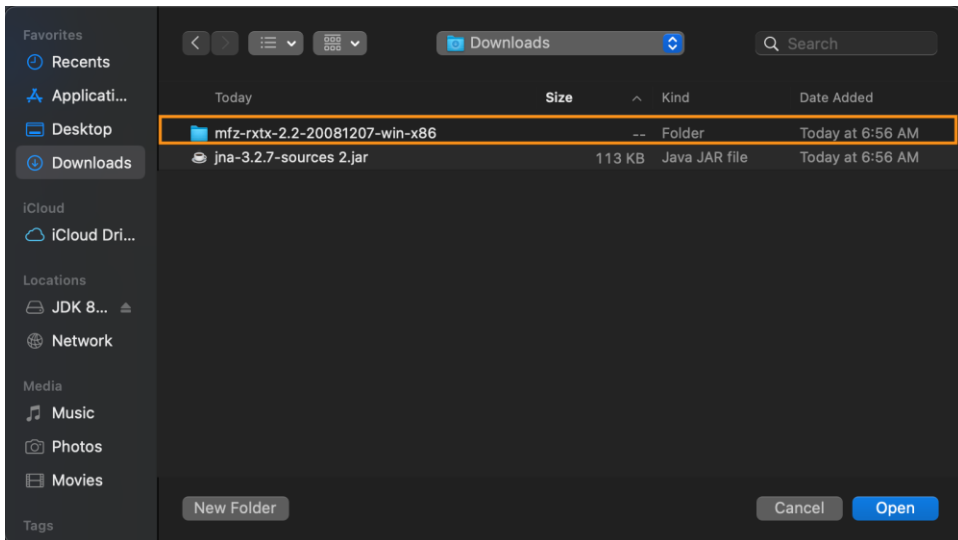


Figure 43: Double click the **mfz-rxtx-2.2-20081207-win-x86** file

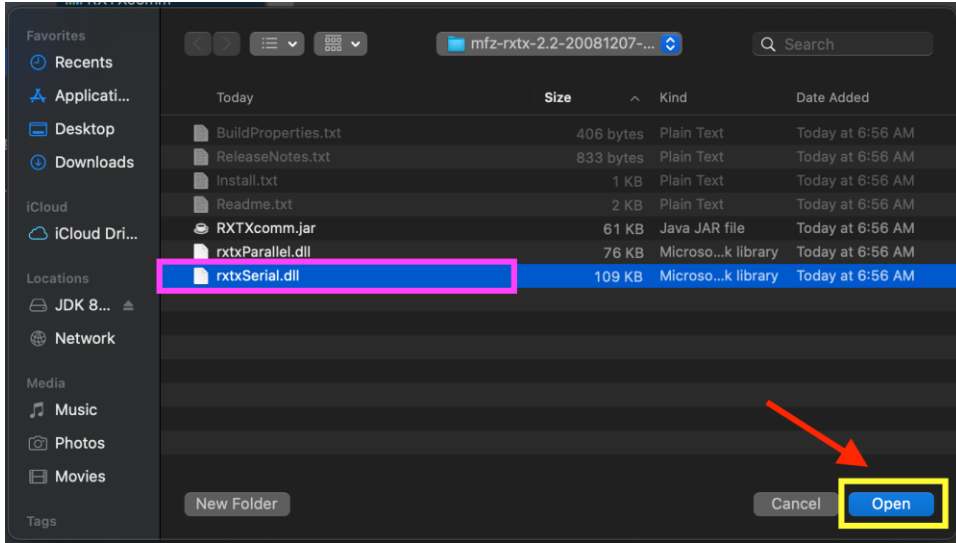
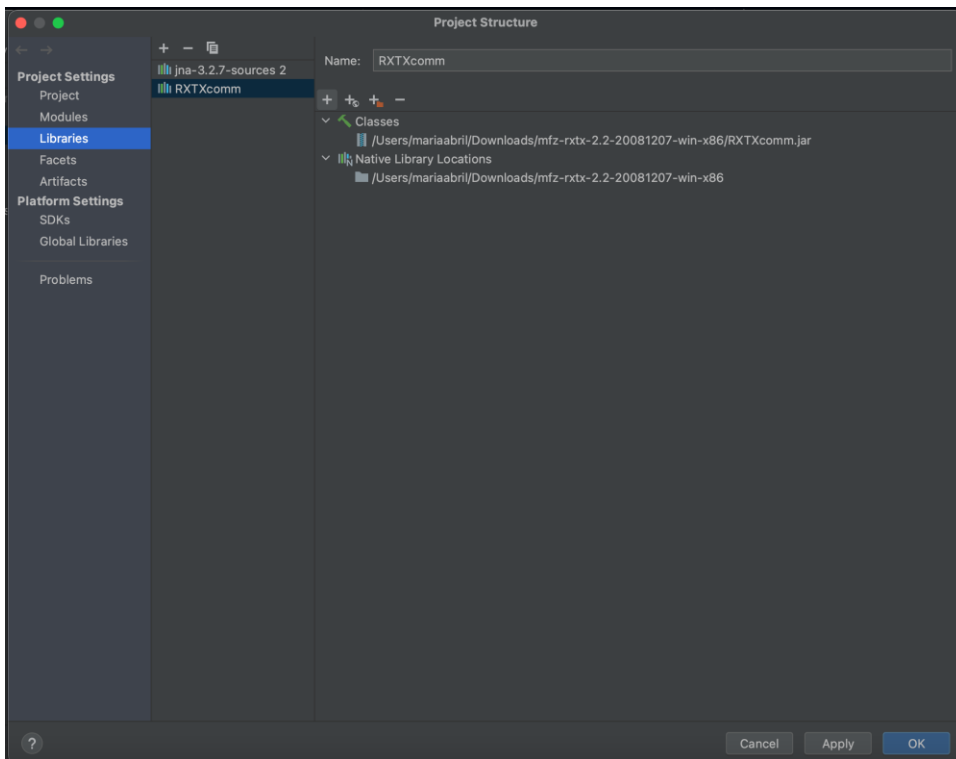


Figure 44: Adding the **rxtxSerial.dll** file



*Figure 45: Fully updated Project Structure Tab*

## **STEP 16**

→ **Now that everything is set up, press OK, as shown in figure 46**

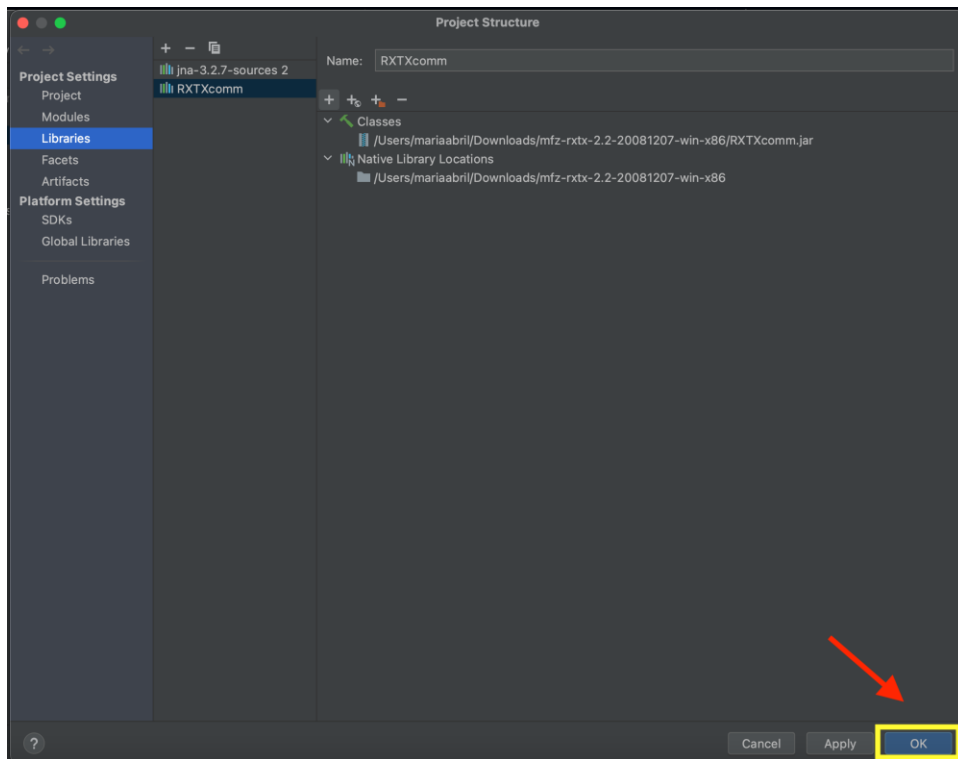


Figure 46: Adding the entire Project Structure

## STEP 17

- **Copy and upload the JAVA code program on IntelliJ IDEA CE** in the blank space, as shown in Figure 47:
- Figure 48 is what your screen should look like with the code added.

## JAVA CODE

```

import java.awt.*;
import java.awt.event.InputEvent;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.util.Enumeration;

public class Mouse implements SerialPortEventListener {
    SerialPort serialPort;
    /** The port we're normally going to use. */
    private static final String PORT_NAMES[] = {
        "/dev/tty.usbserial-A9007UX1", // Mac OS X
        "/dev/ttyACM0", // Raspberry Pi
        "/dev/ttyUSB0", // Linux
        "COM4", // Windows***** (changed)
    };
    /**
     * A BufferedReader which will be fed by a InputStreamReader
     * converting the bytes into characters
     * making the displayed results codepage independent
     */
    private BufferedReader input;
    /** The output stream to the port */
    private OutputStream output;
    /** Milliseconds to block while waiting for port open */
    private static final int TIME_OUT = 2000;
    /** Default bits per second for COM port. */
    private static final int DATA_RATE = 9600;

    int buttonOld = 1;

    public void initialize() {
        // the next line is for Raspberry Pi and
        // gets us into the while loop and was suggested here was suggested
        http://www.raspberrypi.org/phpBB3/viewtopic.php?f...
        //System.setProperty("gnu.io.rxtx.SerialPorts", "/dev/ttyACM0"); I got rid of this
        CommPortIdentifier portId = null;
        Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();
        //First, Find an instance of serial port as set in PORT_NAMES.
        while (portEnum.hasMoreElements()) {
            CommPortIdentifier currPortId = (CommPortIdentifier) portEnum.nextElement();
            for (String portName : PORT_NAMES) {
                if (currPortId.getName().equals(portName)) {
                    portId = currPortId;
                }
            }
        }
    }
}

```

43

```

        break;
    }
}
}
if (portId == null) {
    System.out.println("Could not find COM port.");
    return;
}
try {
    // open serial port, and use class name for the appName.
    serialPort = (SerialPort) portId.open(this.getClass().getName(),
        TIME_OUT);
    // set port parameters
    serialPort.setSerialPortParams(DATA_RATE,
        SerialPort.DATABITS_8,
        SerialPort.STOPBITS_1,
        SerialPort.PARITY_NONE);
    // open the streams
    input = new BufferedReader(new InputStreamReader(serialPort.getInputStream()));
    output = serialPort.getOutputStream();
    // add event listeners
    serialPort.addEventListener(this);
    serialPort.notifyOnDataAvailable(true);
} catch (Exception e) {
    System.err.println(e.toString());
}
}
}
/**
 * This should be called when you stop using the port.
 * This will prevent port locking on platforms like Linux.
 */
public synchronized void close() {
    if (serialPort != null) {
        serialPort.removeEventListener();
        serialPort.close();
    }
}
/**
 * Handle an event on the serial port. Read the data and print it. In this case, it calls the mouseMove method.
 */
public synchronized void serialEvent(SerialPortEvent oEvent) {
    if (oEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        try {
            String inputLine=input.readLine();
            mouseMove(inputLine);
            System.out.println("*****");
            //System.out.println(inputLine);
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
}

```

44

```

    }
    // Ignore all the other eventTypes, but you should consider the other ones.
}
public static void main(String[] args) throws Exception {
    Mouse main = new Mouse();
    main.initialize();
    Thread t=new Thread() {
        public void run() {
            //the following line will keep this app alive for 1000 seconds,
            //waiting for events to occur and responding to them (printing incoming messages to console).
            try {Thread.sleep(1000000);} catch (InterruptedException ie) {}
        }
    };
    t.start();
    System.out.println("Started");
}

```

```

// My method mouseMove, takes in a string containing the three data points and operates the mouse in turn
public void mouseMove(String data) throws AWTException
{
    int index1 = data.indexOf(" ", 0);
    int index2 = data.indexOf(" ", index1+1);
    int yCord = Integer.valueOf(data.substring(0, index1));
    int xCord = Integer.valueOf(data.substring(index1 + 1, index2));
    int button = Integer.valueOf(data.substring(index2 + 1));
    Robot robot = new Robot();

    int mouseY = MouseInfo.getPointerInfo().getLocation().y;
    int mouseX = MouseInfo.getPointerInfo().getLocation().x;

    if (button == 0)
    {
        if (buttonOld == 1)
        {
            robot.mousePress(InputEvent.BUTTON1_DOWN_MASK);
            robot.delay(10);
        }
    }
    else
    {
        if (buttonOld == 0)
            robot.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);
    }

    if (Math.abs(xCord - 500) > 5)
        mouseX = mouseX + (int)((500 - xCord) * 0.02);
    if (Math.abs(yCord - 500) > 5)
        mouseY = mouseY - (int)((500 - yCord) * 0.02);
}

```

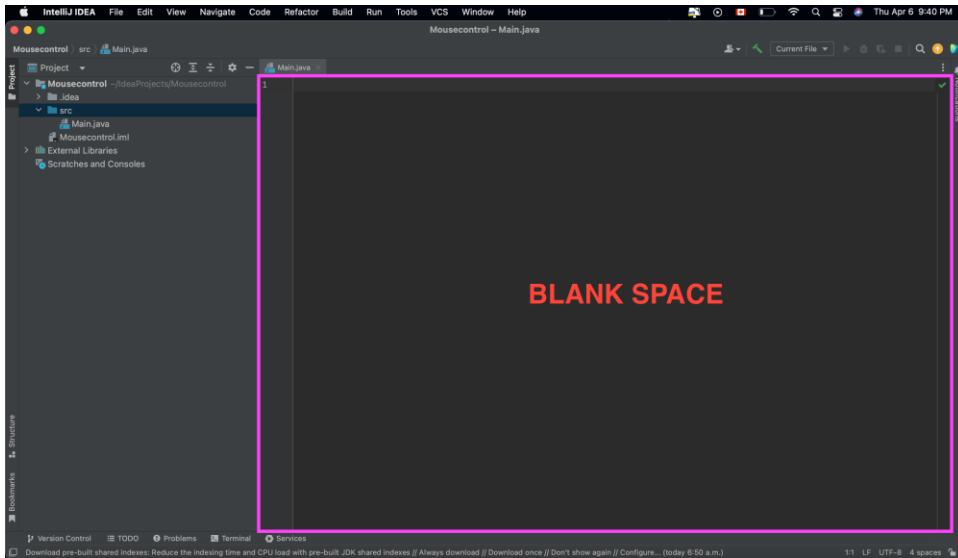
45

```

robot.mouseMove(mouseX, mouseY);

buttonOld
System.out.println(xCord + ":" + yCord + ":" + button + ":" + mouseX + ":" + mouseY);
return;
}
}

```



*Figure 47: Pasting the code in the blank space*



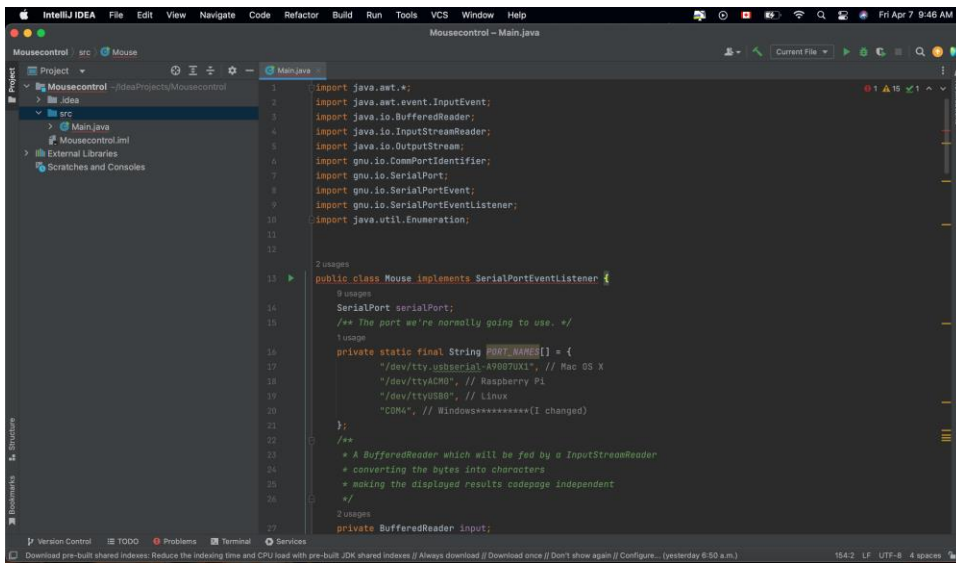


Figure 48: Uploaded the Java Code on IntelliJ IDEA CE

## RUNNING THE CODES

### STEP 18

- Go back to the Arduino IDE tab that was created following steps 4,5 and 6.
- Verify by clicking on the checkmark icon. It is found on the top left corner of the Arduino IDE tab, as shown in figure 49.
- A Done Compiling message should appear, figure 50.
- Then click on the → icon to upload the code onto the Arduino Uno board, as shown in Figure 51.
- A Done Uploading message should appear, Figure 52.

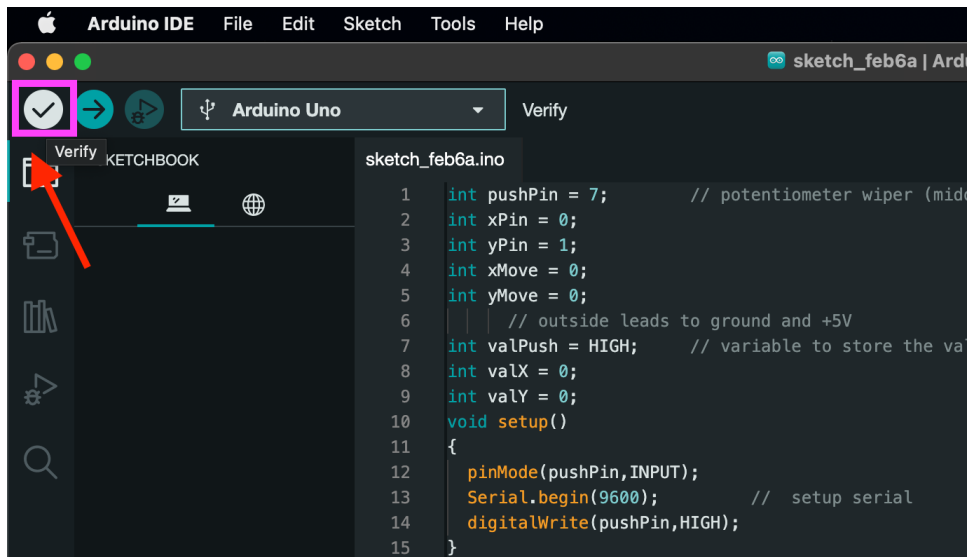


Figure 49: Verifying the Arduino Code

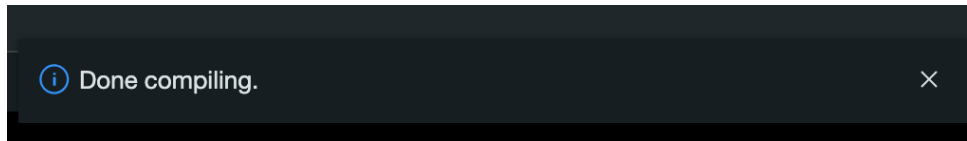


Figure 50: Done compiling message

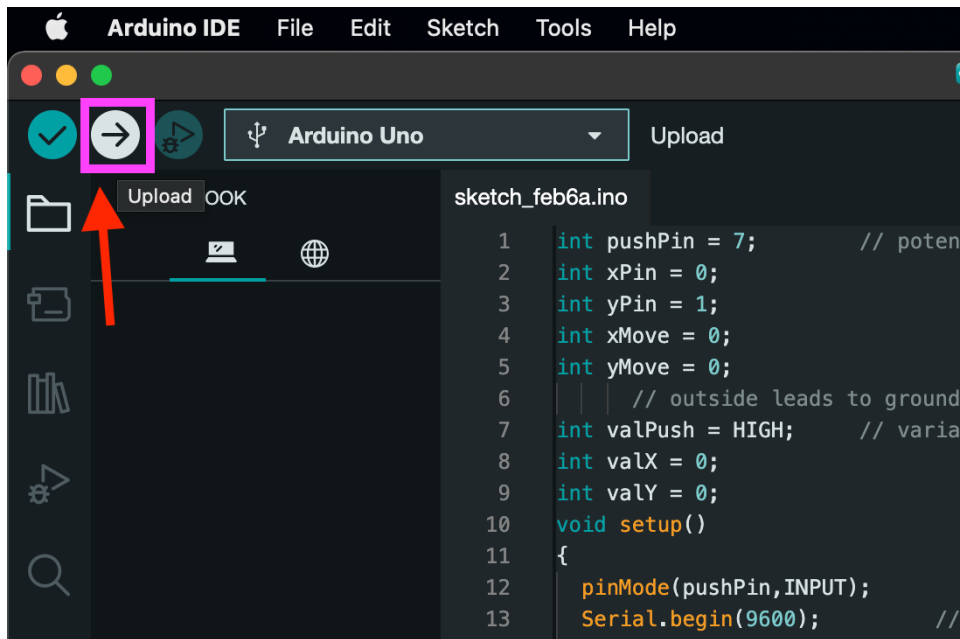


Figure 51: Uploading the Code onto the Arduino Uno board

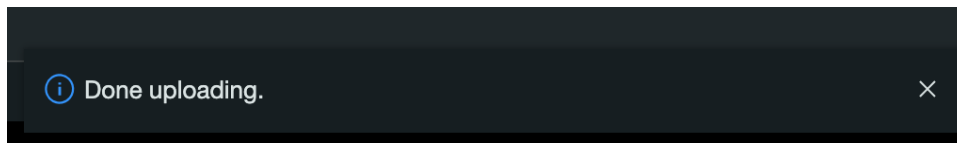


Figure 52: Done uploading message

## STEP 19

- Now go back to the IntelliJ IDEA CE project tab.
- Click on the build Project Icon. Found at the top right corner, as shown in Figure 53.
- To be aware if an error message appears click on the Build icon at the bottom left of the Tab, Figure 54.

- If there is a message that says something similar to **Class Mouse is public, should be declared in a file name Mouse.java**. You are required to go to line 13 and make sure it says **public class Main implements Serializable**, as shown in Figure 55.
- Then Run the program by clicking on the play button. Found at the top right corner, as shown in Figure 56.

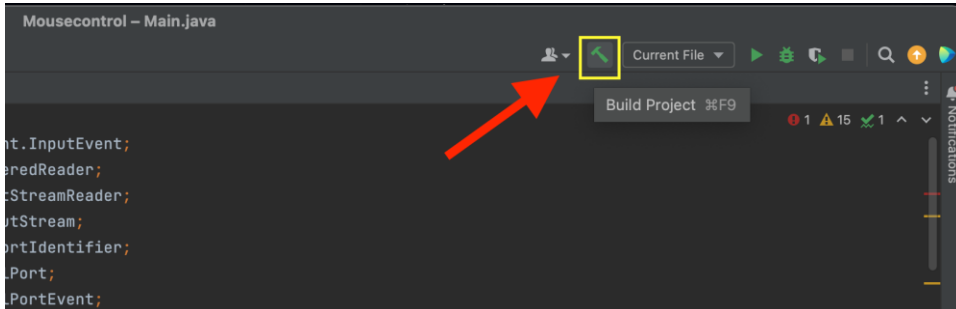


Figure 53: Building the IntelliJ IDEA CE project

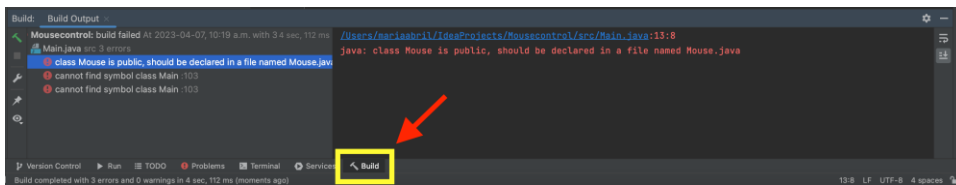


Figure 54: Checking for errors

```
1 import java.awt.*;
2 import java.awt.event.InputEvent;
3 import java.io.BufferedReader;
4 import java.io.InputStreamReader;
5 import java.io.OutputStream;
6 import gnu.io.CommPortIdentifier;
7 import gnu.io.SerialPort;
8 import gnu.io.SerialPortEvent;
9 import gnu.io.SerialPortEventListener;
10 import java.util.Enumeration;
11
12
13 public class Mouse implements SerialPortEventListener {
14     SerialPort serialPort;
15     /** The port we're normally going to use. */
```

Figure 55: Ensuring line 13 is correct

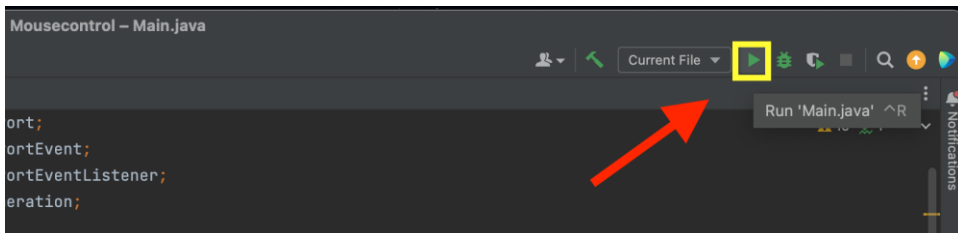


Figure 56: Running the code

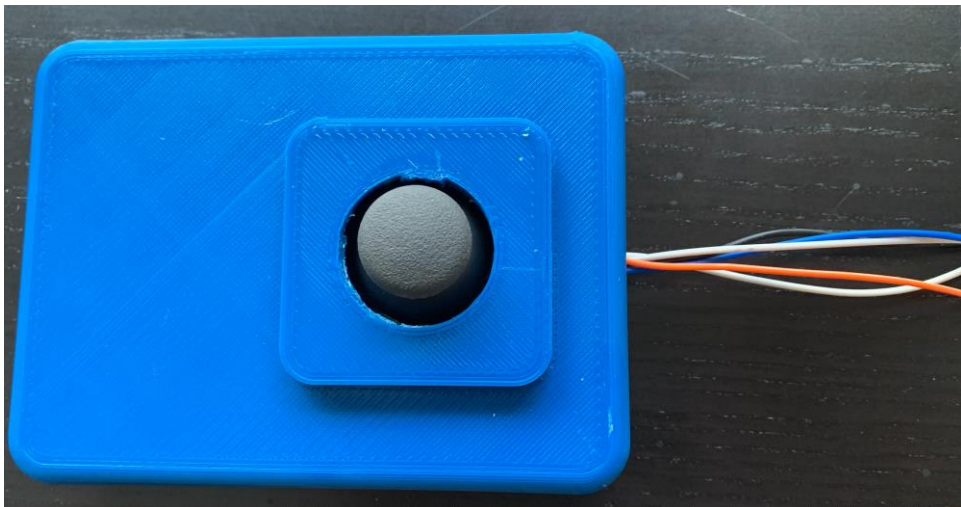
## STEP 20

→ You can now move the thumb joystick acting as a computer mouse.

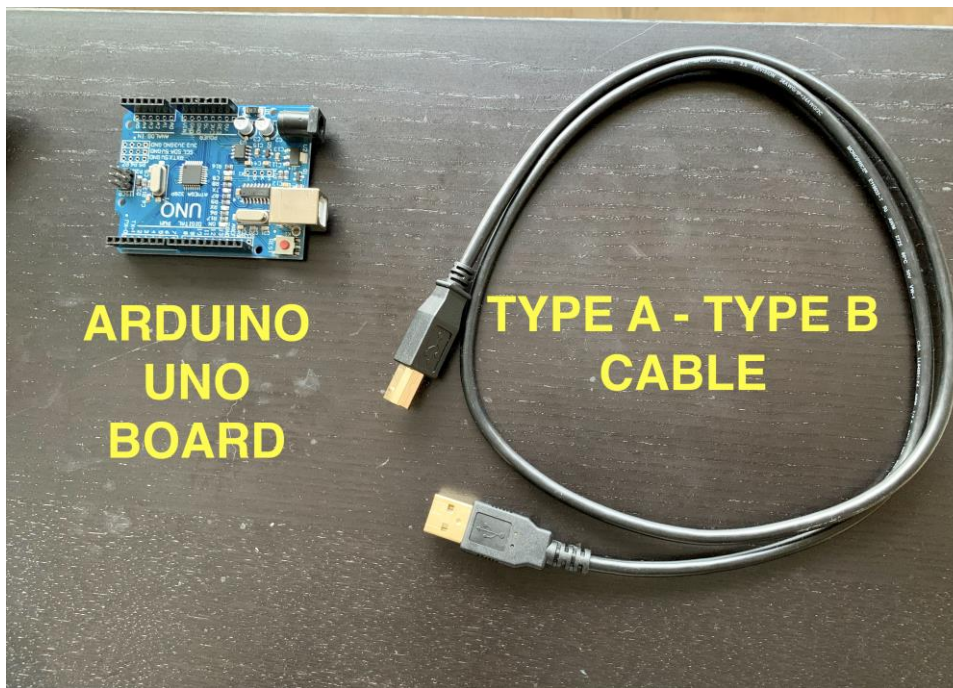
### 3.1 Configuration Considerations

#### Physical prototype:

**The only** equipment needed for this system is a computer device to connect the electrical system. The product will come with the Thumb joystick connected with the 5 electrical wires, all together in the mouse device case. As shown in Figure 57. It will also come with an Arduino Uno board and a Type A- Type B USB cable, as shown in figure 58.



*Figure 57: Thumb Joystick Device*



c

### **Software prototype:**

The prototype requires the Arduino IDE and IntelliJ IDEA CE applications to be installed on your laptop. All necessary codes and files that are also required to download/upload on your laptop are presented in this user manual. A detailed step by step instruction is explained. Once all of the software components are properly installed the user will be able to execute and start using the system.

Moreover, 2 main codes are needed to make the thumb joystick act as a computer mouse. One for the Arduino IDE and the other for the IntelliJ IDEA CE java program. 2 codes are required because the Arduino Uno language functions do not have the capability of a mouse library that allows devices such as a thumb joystick to act as a computer mouse. That being said, the Joystick Program Code needs to be uploaded onto the Arduino IDE. The code will be sent to the Arduino

53

Uno board which with the help of the IntelliJ IDEA CE Java code program, it has the ability to receive and process the serial output values from the Arduino Uno board. Together, this allows the thumb joystick to act as a computer mouse.

### **3.2 User Access Consideration**

This system is designed for individuals who experience tremors and are seeking a solution to improve their computer use. In terms of restrictions on accessibility or use, the main consideration would be the intensity of the user's tremors. It is very important to note that individuals who experience tremor have different backgrounds.

### **3.3 Accessing/setting up the System**

#### **Physical prototype:**

The physical prototype procedures that are required from the user is to have the 5 electrical wires properly connected to the Arduino Uno board. Ensuring each wire is in its proper placing as shown in **3.Getting Started, Step 2, Figure 3**. As well as properly identifying and connecting the TYPE A –TYPE B USB Cable.

#### **Software prototype:**

For the software components the procedures the user must complete are having the Arduino IDE and IntelliJ IDEA CE applications installed correctly on their device in order to run the program. Also, properly downloading and uploading all required files (RXTX and JNA) and codes (Joystick Program and Java). Once everything has been installed, the user will be able to move the thumb joystick in which it will act as a computer mouse. Allowing the user to move the cursor and open and close online tabs.

### **3.4 System Organization & Navigation**

The key features of this system are the software and electrical system.

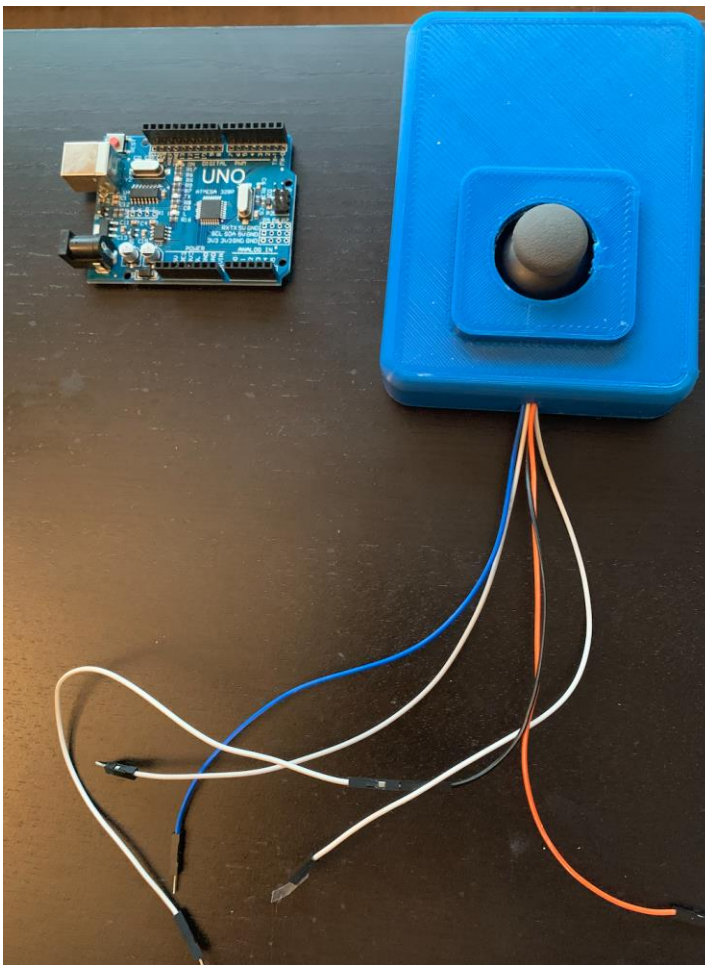
## **PHYSICAL PROTOTYPE**



## Electrical system

### 3.4.1- 5 wire connections to the Arduino Uno Board

- ◆ Main components:
  - Thumb joystick attached with the 5 wires all together in the case.
  - Arduino Uno board



*Figure 59: Main components*

- ◆ The electrical system is the main component for setting up this system because it will allow the IntelliJ IDEA CE java program to process the serial output values from the Arduino Uno. This aspect allows the thumb joystick to act as a computer mouse. That being said, it is very important to properly connect the 5 wires that are attached to the thumb joystick to the Arduino Uno Board. If not properly connected the java code will not be read.

## SOFTWARE PROTOTYPE

### 3.4.2- Arduino IDE

- ◆ Main components: The joystick program code
  - The joystick program code is very important because it activates the serial output values that the Arduino Uno board is projecting and will be read by the java code on IntelliJ IDEA CE.
  - The user needs to directly copy and paste the code on the Arduino IDE application. First verify and then upload it, Figure 60.



Figure 60: Verifying and uploading the joystick program code

### 3.4.3- IntelliJ IDEA CE

Main components:

1. 2 main libraries

i. RXTX

→ rxtxSerial.dll file

ii. JNA

2. JDK Java SE Development Kit 8u251

3. Java Code

The main components 1 and 2 work together to make the Java code run properly. It is important that the user installs the 2 main libraries properly (RXTX and JNA) by downloading the files that have been provided in **3. Getting Started** → **Step 7**. The JDK Java SE Development Kit 8u251 is also very important because without a JDK file the project will not be able to be created. With all the files downloaded the user can copy and paste the Java code and run the program properly.

The Arduino IDE and IntelliJ IDEA CE work together therefore it is important for all files to be installed properly. The Arduino IDE code allows the serial output values to be running, which then the Java program is able to receive and process the serial output values from the Uno and project to the thumb joystick.

### 3.5 Exiting the System

→ To properly exit the system, the software components need to be turned off first and then the physical prototype can be disconnected.

**Software prototype:**

→ **Stop running the code on IntelliJ IDEA CE by clicking the red square as shown in Figure 61.**

→ **Once the user has stopped running the code, you can exit the application by clicking the red (x) icon, top left corner, Figure 62.**

57

→ Now you can exit the Arduino IDE application by clicking the red (x) icon, top left corner. Shown in Figure 63.

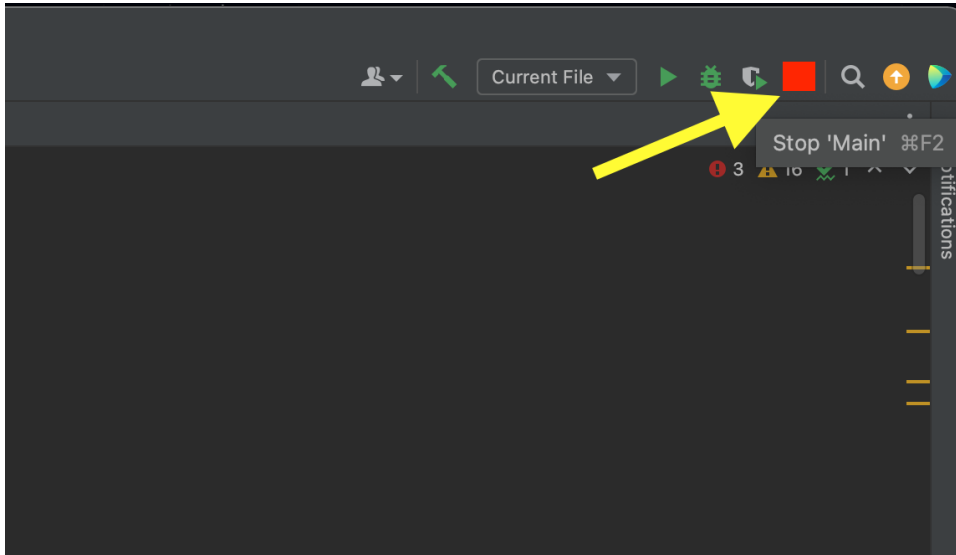


Figure 61: Stopping code run on IntelliJ IDEA CE

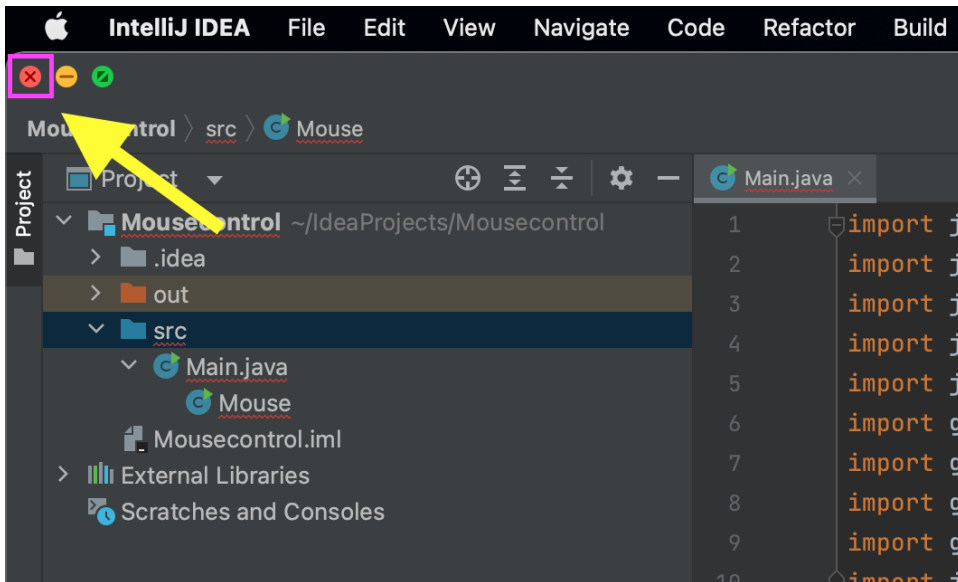


Figure 62: Exiting IntelliJ IDEA CE application

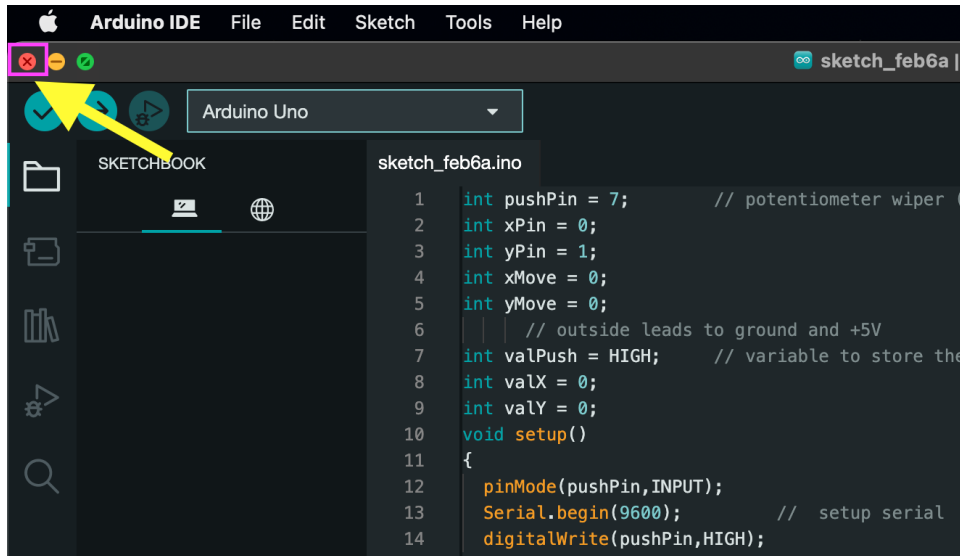


Figure 63: Exiting Arduino IDE application

### Physical prototype:

To disconnect the physical prototype, the user must disconnect the TYPE A USB that is connected to the computer. Shown in Figure 64. As well as the 5 wires that are connected to the Arduino Uno board. Shown in Figure 65.



*Figure 64: TYPE A disconnected from computer device*

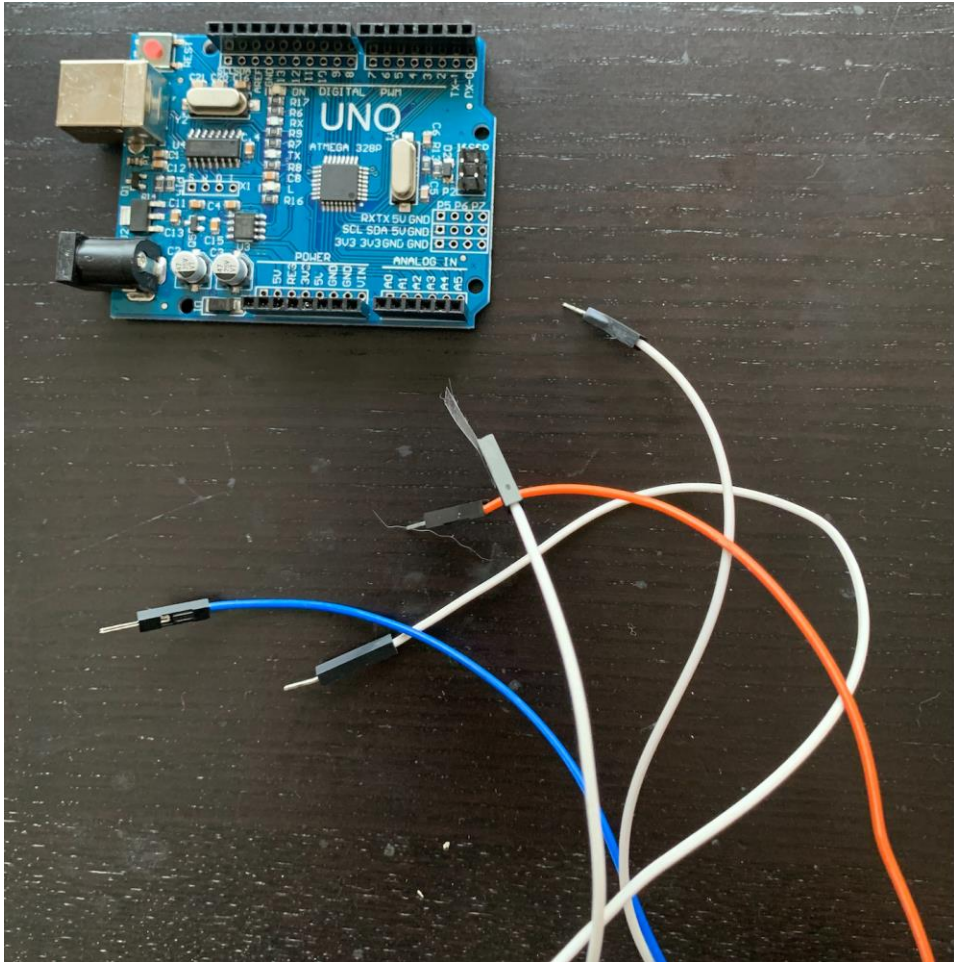


Figure 65: The 5 wires disconnected from the Arduino Uno Board

## **4 Using the System**

Using the system has been made extremely simple. The system is built of two main features: moving the mouse or pressing a button. When either of these functions are executed, the computer will run the specific action.

The following sub-sections provide detailed, step-by-step instructions on using the various functions or features of the <Mouse Movement> & <Click Function>.

### **4.1 <Mouse Movement>**

With the mouse movement system, all the user has to do is (ACTION) push the joystick in any desired direction to navigate pages, apps, and desktop. When the user pushes the joystick, the Arduino code sends the data received by the serial monitor to the computer of the user, which is then interpreted by the Java code. Once interpreted, the Java code moves the mouse pointer to the desired spot. It is possible that if the user puts too many inputs in that the mouse may delay in movements as it is reading all the movements input.

#### **4.1.1 <Joystick Click>**

Included with the mouse movement code, the code also includes a left click function. To engage this function the user needs to (ACTION) press the joystick in. When this is done the code will read the input and left click wherever the cursor is hovering over.

### **4.2 <Click Function>**

The click functions that are added onto the base of the mouse will be respectively set as the left and right click. The left click can be used alternatively with the click function on the joystick to

62



incorporate ease of use. All the users will have to do is (ACTION) click either the left click or right click button located on the side of the housing for the joystick. Once pressed, the arduino sends a signal to the java code to initiate the left click or right click function.

## 5 Troubleshooting & Support

When using any technology, it is normal to encounter problems. When this happens, you should consult this page. Some common errors a user will face are USB ports not picking up the mouse being plugged in, the mouse moving too fast, and a compile error when the mouse program is run.

### 5.1 Error Messages or Behaviors

A few errors messages users can receive are the following:

1. Port not found
2. GNU.IO not found
3. Compile Error

When a user faces these common errors codes, the fix is quite simple. If ever the user sees these codes when they run the mouse, they just need to check to see that the port they have connected to is identified correctly in the code, if it is, all the user has to do is restart their system and plug the mouse back into the correct USB port. As with the second error, the two things the users needs to check is if they have downloaded all the files correctly, if so then the user just needs to restart their system. Finally, if the user is encountering a compiler error, they will need to open IntelliJ and hover over the file tab in the top left corner and look for the option that says rebuild IDE.

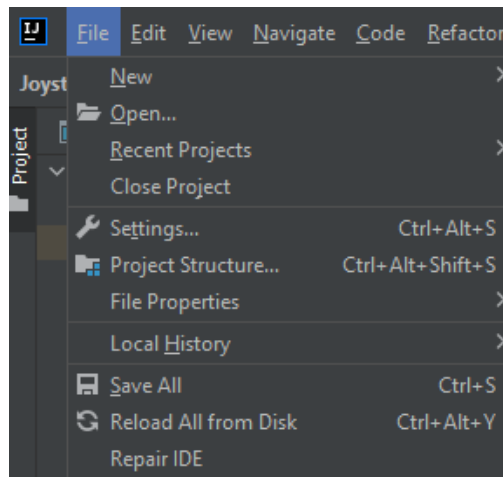


Figure 66: File -> Repair IDE function

## 5.2 Special Considerations

If the user finds that the mouse cursor moves too fast they can enter the Java code and change the number value on the delay in the ports which will allow for slower movements and more of a controlled feedback. For best feedback, the user should only input any number between 1-2000

```
private static final int TIME_OUT = 1000;  
/**  
 * Default bits per second for COM port.  
 */
```

Figure 67: TIME\_OUT = <insert number 1-2000>

## 5.3 Maintenance

The device should be regularly checked for any cracks or break to the case since those cracks will allow for dust and debris to mess with the function of the microcontroller placed inside the housing. The joystick must also be checked for stick drift. To do this all the user will need to do is

turn the mouse on and see if it moves on it's own. To prevent this from happeniuung the user should not let liquids get into the joystick module.

## 5.4 Support

If all problems persist or users have any problem that aren't listed in the user manual they should contact the team at any of these email addresses:

**Matthew Emmanuel:** [memma017@uottawa.ca](mailto:memma017@uottawa.ca)

**Maria Abril Vargas:** [mabri051@uottawa.ca](mailto:mabri051@uottawa.ca)

**Jason Jin:** [zjin052@uottawa.ca](mailto:zjin052@uottawa.ca)

These 3 email addresses belong to the creators of the product, so if users are to run into any problems, they can contact any of the three email addresses. The first point of contact should be Matthew Emmanuel, if the user is unable to reach Matthew, then they may go ahead and reach out to all Maria & Jason to ensure their email is seen. Expect replies within 1-3 business days. If any security incidents occur, the user must outline this information in the subject line of the email, when this is done they will be sent a form to fill out indicating what type of incidents occurred

## 6 Product Documentation

### 6.1 Mechanical subsystem

#### 6.1.1 BOM (Bill of Materials)

The bill of materials for the mechanical subsystem can be found at this link:

<https://docs.google.com/spreadsheets/d/1FBQZK72XGXYsVN2tQepPDlvyDCIL8CTrZBvWe8ArQk/edit#gid=0>

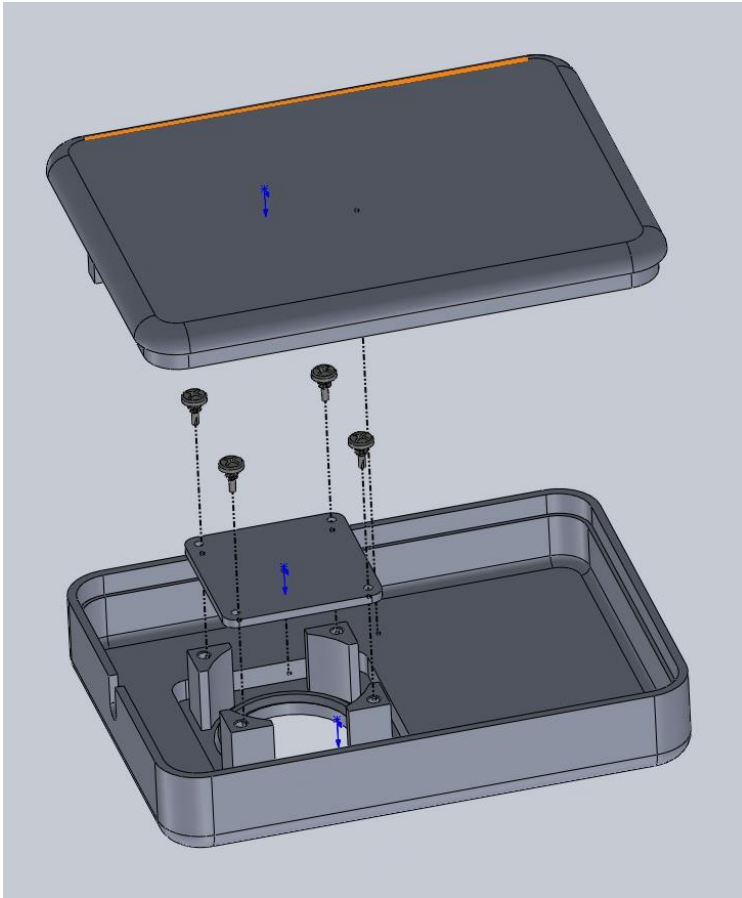
#### 6.1.2 Equipment list

- PLA 3D printing plastic
- A 3D printer (along with software to 3D print)
- Onshape, or another CAD software

#### 6.1.3 Instructions

To build the mechanical subsystem, 3D print the case as provided by the STL files here:  
[https://drive.google.com/drive/folders/1YQrdPED\\_7XFE70dDLusxKhZJQUMxy45t](https://drive.google.com/drive/folders/1YQrdPED_7XFE70dDLusxKhZJQUMxy45t)

Next screw the joystick to the top part of case. Then close the case.



*Figure 68: Assembly view of Mechanical subsystem*

## **6.2 Electrical subsystem**

### **6.2.1 BOM (Bill of Materials)**

The bill of materials for the Electrical subsystem can be found at this link:

<https://docs.google.com/spreadsheets/d/1FBQZK72XGXYsVN2tQepPDlvyDCIL8CTrZBvWe8ArQk/edit#gid=331665036>

## **6.2.2 Equipment list**

- Arduino UNO
- Breadboard
- Jump wire
- Soldering equipment
- Male pin header
- Joystick

## **6.2.3 Instructions**

7. Solder the male pin header to the Joystick and connect the jump wires as illustrated in Figure

## **6.3 Software subsystem**

### **6.3.1 BOM (Bill of Materials)**

The bill of materials for the Electrical subsystem can be found at this link:

<https://docs.google.com/spreadsheets/d/1FBQZK72XGXYsVN2tQepPDlvyDCIL8CTrZBvWe8ArQk/edit#gid=329486560>

### **6.3.2 Equipment list**

- Arduino IDE
- IntelliJ IDEA

### **6.3.3 Instructions**

Follow the steps in part 3 to set up the software part.

## 6.4 Testing & Validation

After conducting tests, we have confirmed that the mouse moving and simple clicking functions in our prototype are working. However, we have observed a delay time of 3-5 seconds. There is two video link attached below.

Simple click function:

<https://drive.google.com/file/d/12wNzSgj6qn4Bi7MTiUalGNFvj2r3Ktpo/view?usp=sharing>

Mouse moving function:

[https://drive.google.com/file/d/1j--4DWg8pld6VfWJJOmoymzddgshNI4J/view?usp=share\\_link](https://drive.google.com/file/d/1j--4DWg8pld6VfWJJOmoymzddgshNI4J/view?usp=share_link)



## 7 Conclusions and Recommendations for Future Work

The prototype was supposed to have the feature of toggle click function and left click right click function, but for the limited time we have, we only got the mouse moving function and simple clicking function.

Replacing the 3D printed parts with plastic components. This could potentially result in a stronger and more durable casing for our prototype, it will also simplify the manufacturing process.

Due to the lack of time and lack of prediction, the Arduino board is not fitting into the case we made when assembling the final prototype. Possible solutions could include either constructing a larger case or considering a smaller board, such as the Arduino Nano, as an alternative.

For our final prototype, we used an Arduino Uno board to transmit data over USB, which was subsequently received and processed by Java code on a computer. However, the response time of the program was measured to be approximately 3-5 seconds, which may adversely impact the user experience due to the prolonged delay. With the lack of knowledge of programming, we were unable to fix that. We have attempted to optimize the code for our prototype, but unfortunately, it did not yield any noticeable improvement in the response time. There are some other suggestions we haven't tried. Use Serial Buffering: Arduino Uno board uses serial communication for data transmission over USB. Try enabling hardware or software serial buffering on the Arduino Uno board to increase the data transmission speed. Hardware or software serial buffering allows for the accumulation of multiple bytes of data before transmitting them as a batch, which can reduce the overall transmission time. Use a Different Board: Arduino Uno is a popular microcontroller board, but it may not always be the best choice for high-speed data transmission. Choose a board with high-speed data transmission, such as Arduino Due, or switch to a different microcontroller platform that supports faster data transmission.

## 8 Bibliography

→ [jna-3.2.7-sources.zip](#)

→ [mfz-rxtx-2.2-20081207-win-x86.zip](#)

→ “Arduino software.” *Arduino*, <https://www.arduino.cc/en/software>. Accessed 7 April 2023.

→ “Download IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains.” *JetBrains*, <https://www.jetbrains.com/idea/download/#section=mac>. Accessed 7 April 2023.

→ “Java Archive Downloads - Java SE 8u211 and later.” *Oracle*, <https://www.oracle.com/ca-en/java/technologies/javase/javase8u211-later-archive-downloads.html>. Accessed 7 April 2023.

→ “Arduino - Joystick | Arduino Tutorial.” *Arduino Getting Started*, <https://arduinogetstarted.com/tutorials/arduino-joystick>. Accessed 16 February 2023.

→

# APPENDICES

## 9 APPENDIX I: Design Files

Table 3. Referenced Documents

| Document Name                 | Document Location and/or URL | Issuance Date |
|-------------------------------|------------------------------|---------------|
| Maker Repo                    | <a href="#">LINK</a>         | 04/7/2023     |
| 3D Design STL files           | <a href="#">LINK</a>         | 04/7/2023     |
| Joystick Program code         | <a href="#">LINK</a>         | 04/7/2023     |
| Java Code                     | <a href="#">LINK</a>         | 04/7/2023     |
| Libraries and JDK 8u251 files | <a href="#">LINK</a>         | 04/7/2023     |
| Finalized Design              | <a href="#">LINK</a>         | 04/7/2023     |

## 10 APPENDIX II: Other Appendices

Prototype 1 and prototype 2 are other documents that guided us with the creation of the final design.

### PROTOTYPE 1- 3D MODELS

→ For prototype 1 the team created 3 different 3D models which then allowed us to create a conceptual design. The conceptual design incorporates valuable feedback from our client, which includes a mouse base with two buttons and a detachable joystick feature. The goal of this innovative design is to be able to seamlessly switch between the handle and thumb joystick, providing users with optimal comfort and flexibility.

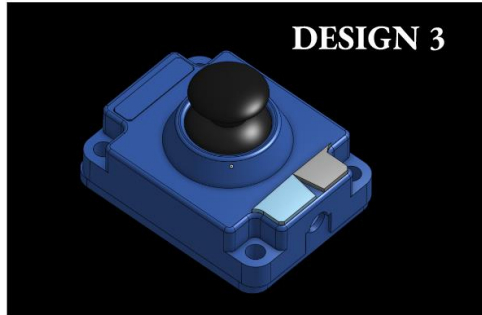
**DESIGN 1**



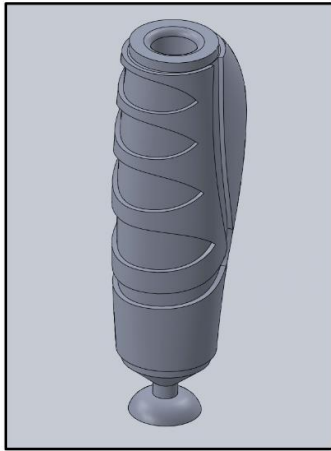
**DESIGN 2**



**DESIGN 3**



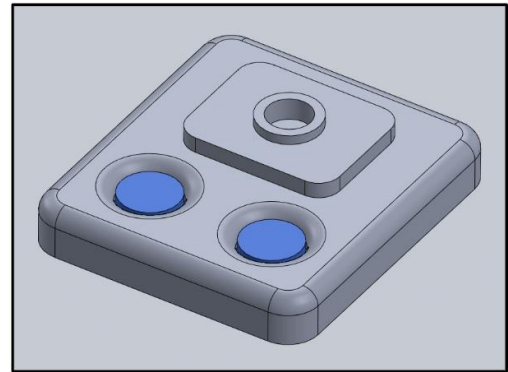
*Figure 69: 3 different 3D Models*



**Figure 1 -Handle Joystick**



**Figure 2 -Thumb Joystick**

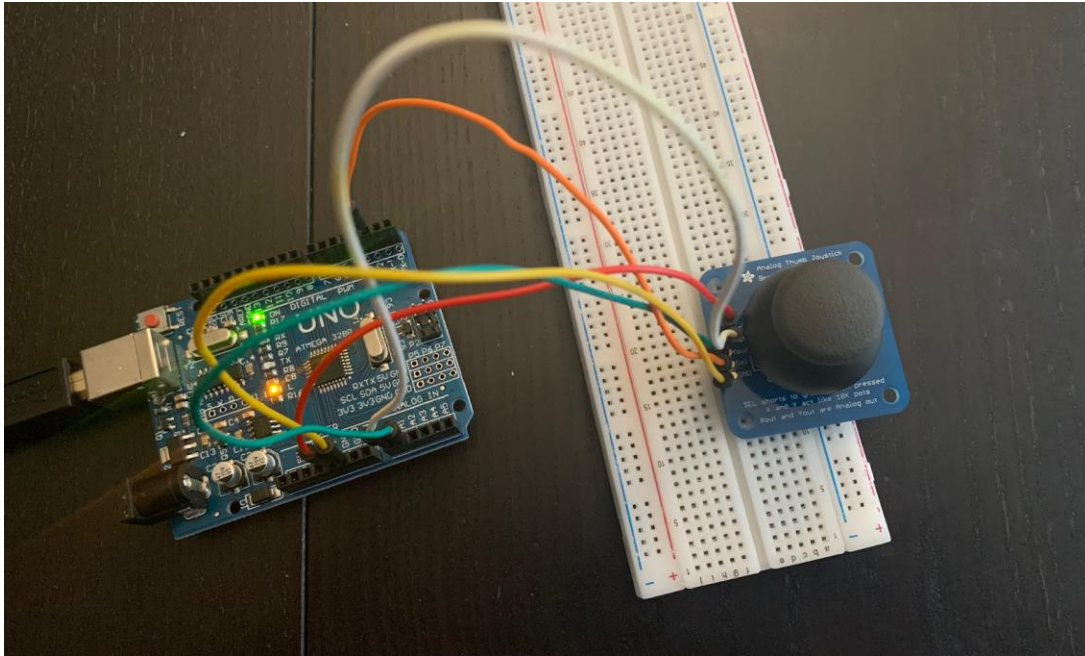


**Figure 3- Base**

*Figure 70: Conceptual Design*

#### PROTOTYPE 2 - Analog Read of the joystick

- The purpose and function of the analog read joystick prototype is to test and verify the design of a joystick. It will intake a code and report a range of values that depict a location on a digital screen.
- The electrical system was followed by the following link: “Arduino - Joystick | Arduino Tutorial.” *Arduino Getting Started*, <https://arduinogetstarted.com/tutorials/arduino-joystick>. Accessed 16 February 2023. Figure 71 shown the electrical connections that were made, and Figure 72 shows the code that was used.
- The code was read successfully and provided us with coordinates of the X and Y axis of the joystick. Together, the X and Y coordinates can be used to determine the direction and magnitude of the joystick's movement. Figure 73.



*Figure 71: Analog Read of the joystick electrical connections*

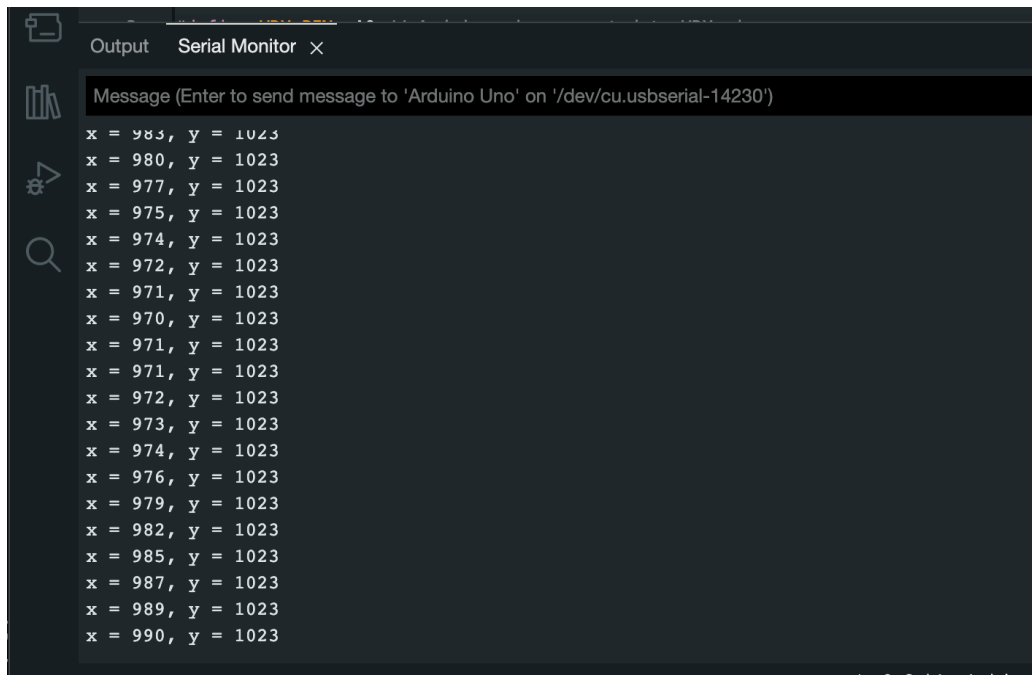
```

4  * This example code is in the public domain
5  *
6  * Tutorial page: https://arduinogetstarted.com/tutorials/arduino-joystick
7  */
8
9  #define VRX_PIN  A0 // Arduino pin connected to VRX pin
10 #define VRY_PIN  A1 // Arduino pin connected to VRY pin
11
12 int xValue = 0; // To store value of the X axis
13 int yValue = 0; // To store value of the Y axis
14
15 void setup() {
16     Serial.begin(9600) ;
17 }
18
19 void loop() {
20     // read analog X and Y analog values
21     xValue = analogRead(VRX_PIN);
22     yValue = analogRead(VRY_PIN);
23
24     // print data to Serial Monitor on Arduino IDE
25     Serial.print("x = ");
26     Serial.print(xValue);
27     Serial.print(", y = ");
28     Serial.println(yValue);
29     delay(200);
30 }

```

Figure 72: Analog Read of the joystick code



A screenshot of the Arduino IDE Serial Monitor window. The window title is "Output Serial Monitor x". Below the title bar is a text input field with the placeholder text "Message (Enter to send message to 'Arduino Uno' on '/dev/cu.usbserial-14230')". The main area of the window displays a list of 20 lines of text, each representing a joystick coordinate pair: "x = [value], y = 1023". The x-axis values range from 983 down to 990 in descending order. The y-axis value is constant at 1023 for all entries. On the left side of the window, there are several icons: a document icon, a list icon, a play icon, and a magnifying glass icon.

```
x = 983, y = 1023
x = 980, y = 1023
x = 977, y = 1023
x = 975, y = 1023
x = 974, y = 1023
x = 972, y = 1023
x = 971, y = 1023
x = 970, y = 1023
x = 971, y = 1023
x = 971, y = 1023
x = 972, y = 1023
x = 973, y = 1023
x = 974, y = 1023
x = 976, y = 1023
x = 979, y = 1023
x = 982, y = 1023
x = 985, y = 1023
x = 987, y = 1023
x = 989, y = 1023
x = 990, y = 1023
```

*Figure 73: The coordinates of the X and Y axis of the joystick.*