

This is my self-study project for the Nova Scotia Community College for winter semester 2025. The main focus is on programming the Arduino for turning off and on LEDs, along with various inputs and outputs.

# Self-Study Project

Arduino Project

Peter Vaughan

---

## Table of Contents

Project Overview .....	2
Project Components .....	3
Project Components .....	3
The LEDS and Other Supplies .....	6
Main Communication .....	7
Communication .....	8
COM Ports .....	8
A-Series Input & Output .....	9
Other Info .....	10
My attempts .....	10
Summary .....	11
Submission .....	12
Rubric Outcomes Completed .....	12

## Project Overview

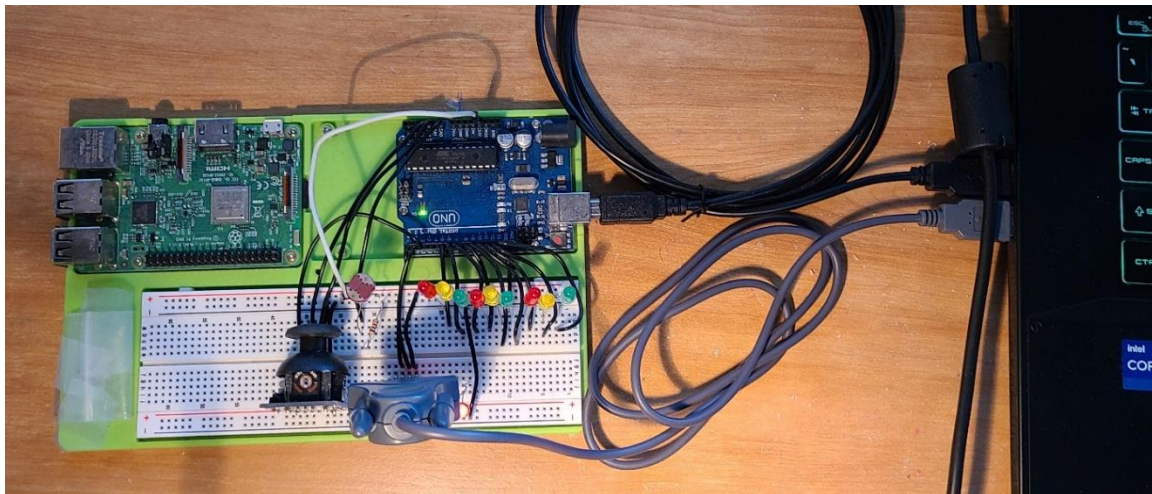
For this self-directed learning, I wanted to learn about developing software for an Arduino. I will be using the Arduino IDE, meaning that I would be using C programming. With the realization that this requires some specialized hardware, I will be mostly focusing on the software. The specialized hardware would just be an Arduino board, cables, and some attachments. Keep in mind that I do have a significant background in hardware design and manufacturing. This means that doing all the hardware work is going to be super-fast and super-easy for me.

I wanted to do this particular project because I wanted to do more projects that incorporate hardware and software. With that in mind, I realized I am in an IT programming program, so I would be more focus on the programming. For this project, I do not focus much on the hardware, because once the hardware is set up, I do not need to build or modify any more of the hardware.

The first two parts of the project would be to set up the Arduino and all necessary hardware. The hardware will include LEDs and an UART component. Then I would test the hardware to make sure that I set it up correctly. The testing would be through software. The other part of the project would be to program the LEDs to do various activities such as blinking lights, chasing lights and binary counter.

The last two parts of the project would be to develop an UART part of the program. The hardware design is super simple. The hardware part of this would be done in the first part of the project. I can use PuTTY to demonstrate the UART communication. PuTTY is a free open-source terminal emulator, serial console and network file transfer application, which I have used for similar projects. For this part of the project, I will type a command through PuTTY. This will change the LEDs and display some info on the PuTTY terminal, which my code on the Arduino will provide. The other part of the project would be to develop this project even further.

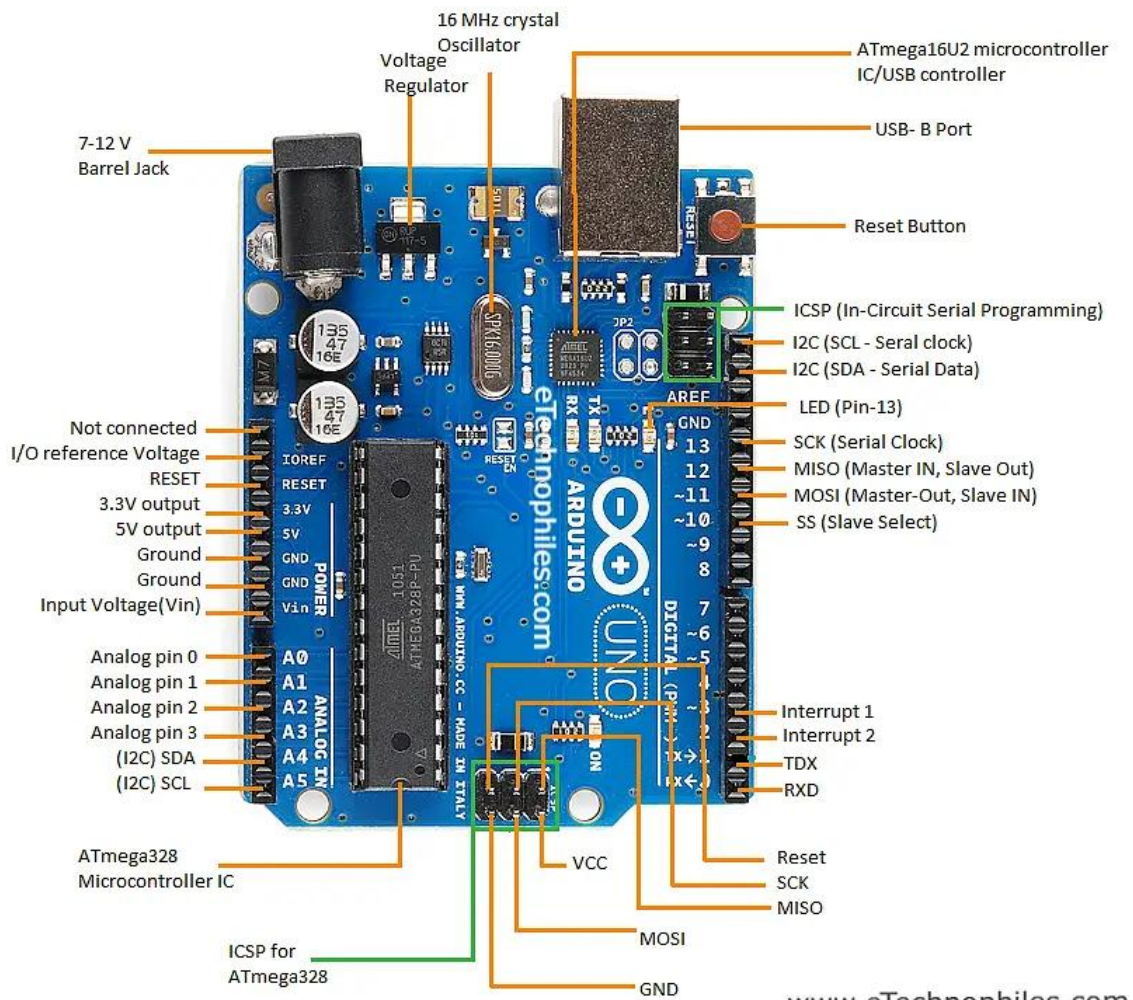
Overall, I extremely enjoyed this project. While I did go into concepts like the `LowPower` library, I did not lose focus on main project goals.



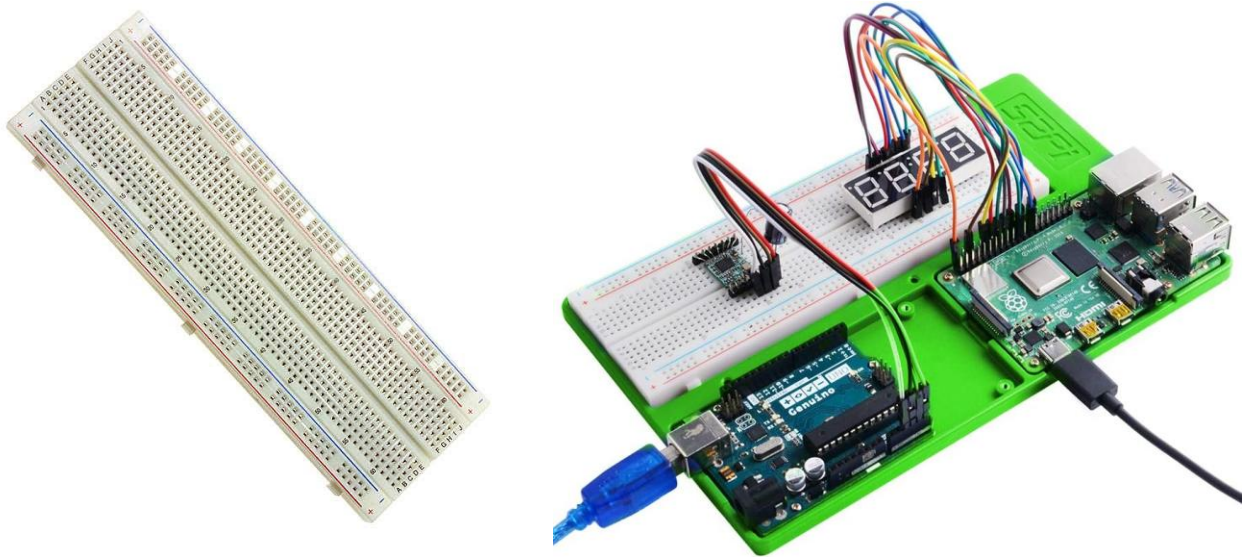
# Project Components

## Project Components

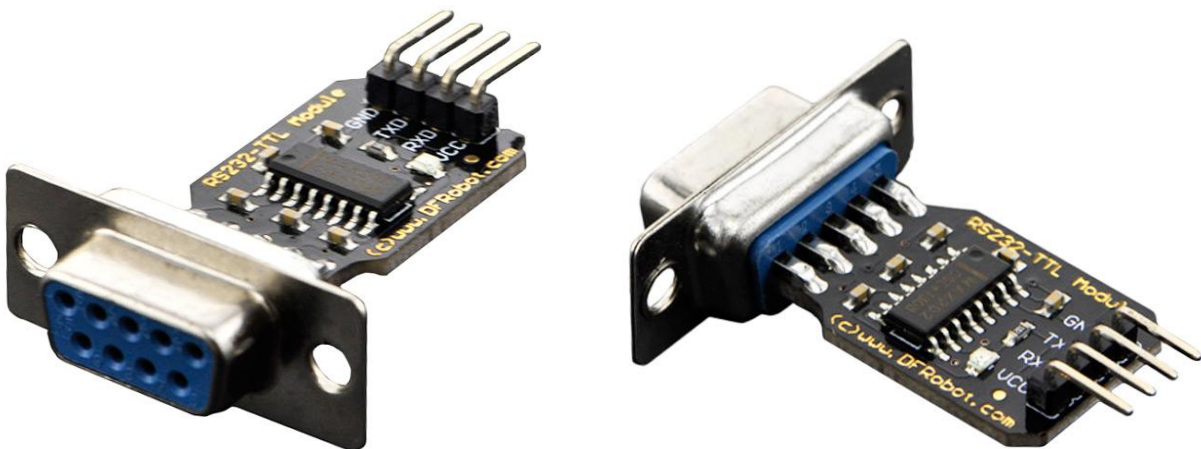
I used The Arduino Uno with a printer cable (Cable Cord USB A Male to USB B Male) to connect the board with the computer. Then I used various 22 AWG jumper wires to connect to the LEDs and other components to the Arduino to make a complete circuit. The Arduino was COM3 on my computer. The image is from an Arduino Forum.



I used the electronic breadboard and a tray to place and organize all the components. Since this was a programming class and not a hardware class, I did not focus on the colors of the wires. Images from Digikey (breadboard) and Amazon (Tray).



I used a RS-232 DFROBOT. It is a RS-232 connector on one side. Similar to VGA but there are differences. The other side is for the electronic breadboard for VCC (5V power), RX, TX and GND (ground). I used a RS232 Serial Cable DB9 9 Pin USB 2.0 Male A Converter Adapter.





The joystick component is added to use with the analog port. For my project, the joystick takes two pins in analog port. This was added to showcase more communication options. For my project, I wanted showcase as many communication and data concepts as possible.



## The LEDs and Other Supplies



I used Green LEDs, Yellow LEDs, and Red LEDs. There are more options for LEDs but this was a similar option. Image from Amazon (Bag of LEDs).

I do have range of resistors including a singular photo-resistor. Also, I do have several 22 AWG jumper cables. One of multiple concepts that I implemented is with a photo-resistor and a regular resistor as a voltage divider.

There were other components that I tried to use initially such as the UB232R and other Arduino attachments.

## Main Communication

My main form of communication is my COM3 (Arduino) and my COM5 (RS-232 module). To connect to these COM Ports is through Serial Monitor via Arduino IDE or “PuTTY”. Digital pins and analog pins are bi-directional. Digital Pin 0 and 1 are digital out or input. Digital Pin 0 can be RX or Digital Pin 1 can be TX. Any of these pins can only have one responsibility at a time. This is why most pins should never have more than one responsibility due to potential accidents.

For this project, I needed to get creative with communication with using the A-Series Input & Outputs (Analog Port). The Arduino Uno has only one RX and TX pins, but there are other ways to gather info from the other pins. The other Arduino boards have different capabilities but if programmer knows how to figure out solutions, anything is possible. The image is from an Arduino’s community blog website.

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

Board	Serial pins	Serial1 pins	Serial2 pins	Serial3 pins	Serial4 pins
UNO R3, UNO R3 SMD Mini	0(RX), 1(TX)				
Nano (classic)	0(RX), 1(TX)				
UNO R4 Minima, UNO R4 WiFi		0(RX0), 1(TX0)			
Leonardo, Micro, Yún Rev2		0(RX), 1(TX)			
UNO WiFi Rev2		0(RX), 1(TX)			
MKR boards		13(RX), 14(TX)			
Zero		0(RX), 1(TX)			
GIGA R1 WiFi		0(RX), 1(TX)	19(RX1), 18(TX1)	17(RX2), 16(TX2)	15(RX3), 14(TX3)
Due	0(RX), 1(TX)	19(RX1), 18(TX1)	17(RX2), 16(TX2)	15(RX3), 14(TX3)	
Mega 2560 Rev3	0(RX), 1(TX)	19(RX1), 18(TX1)	17(RX2), 16(TX2)	15(RX3), 14(TX3)	
Nano 33 IoT		0(RX0), 1(TX0)			
Nano RP2040 Connect		0(RX0), 1(TX0)			
Nano BLE / BLE Sense		0(RX0), 1(TX0)			

The Nano ESP32 board is an exception due to being based on the ESP32 core. Here, `Serial0` refers to `RX0` and `TX0`, while `Serial1` and `Serial2` are additional ports that can be assigned to any free GPIO.

Board	Serial0 pins	Serial1 pins	Serial2 pins	Serial3 pins	Serial4 pins
Nano ESP32	0(RX0), 1(TX0)	Any free GPIO	Any free GPIO		

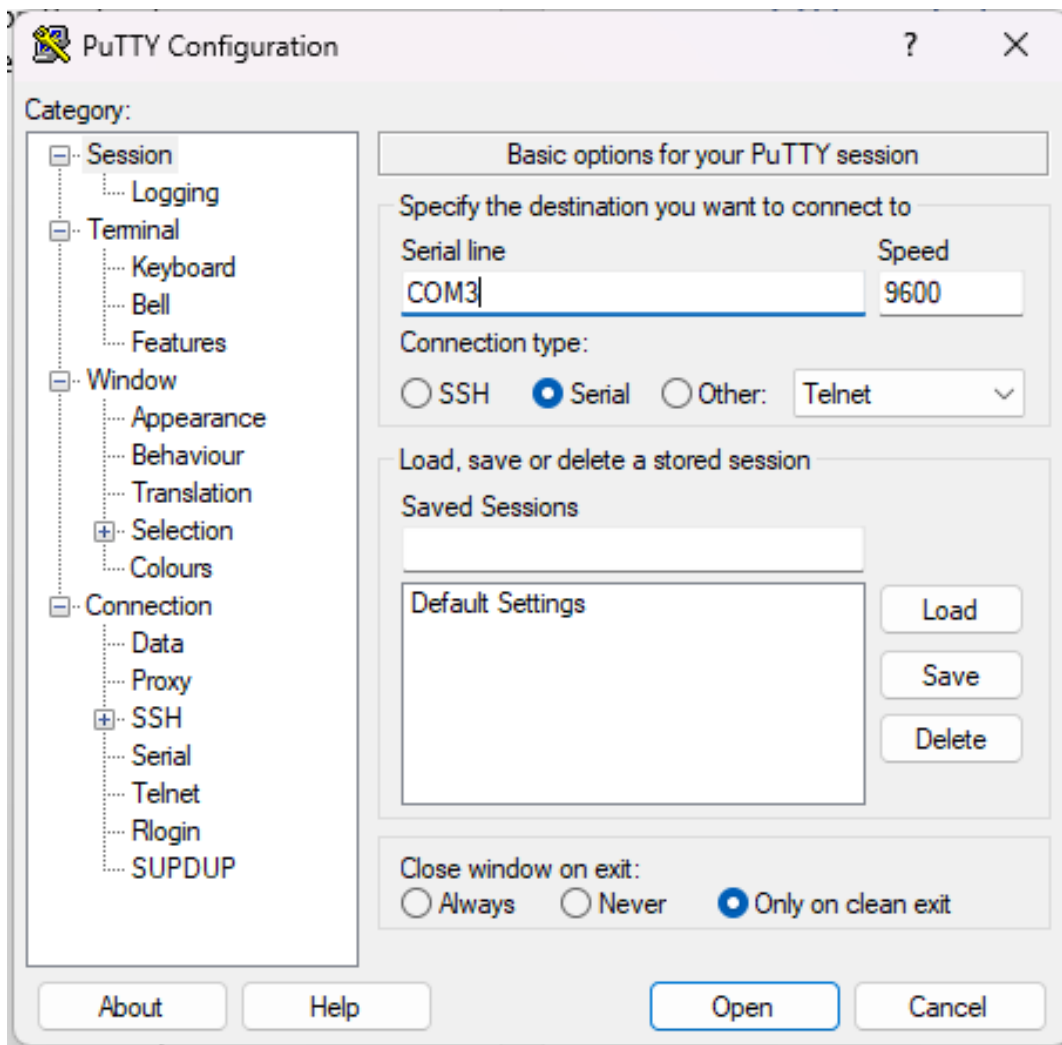


# Communication

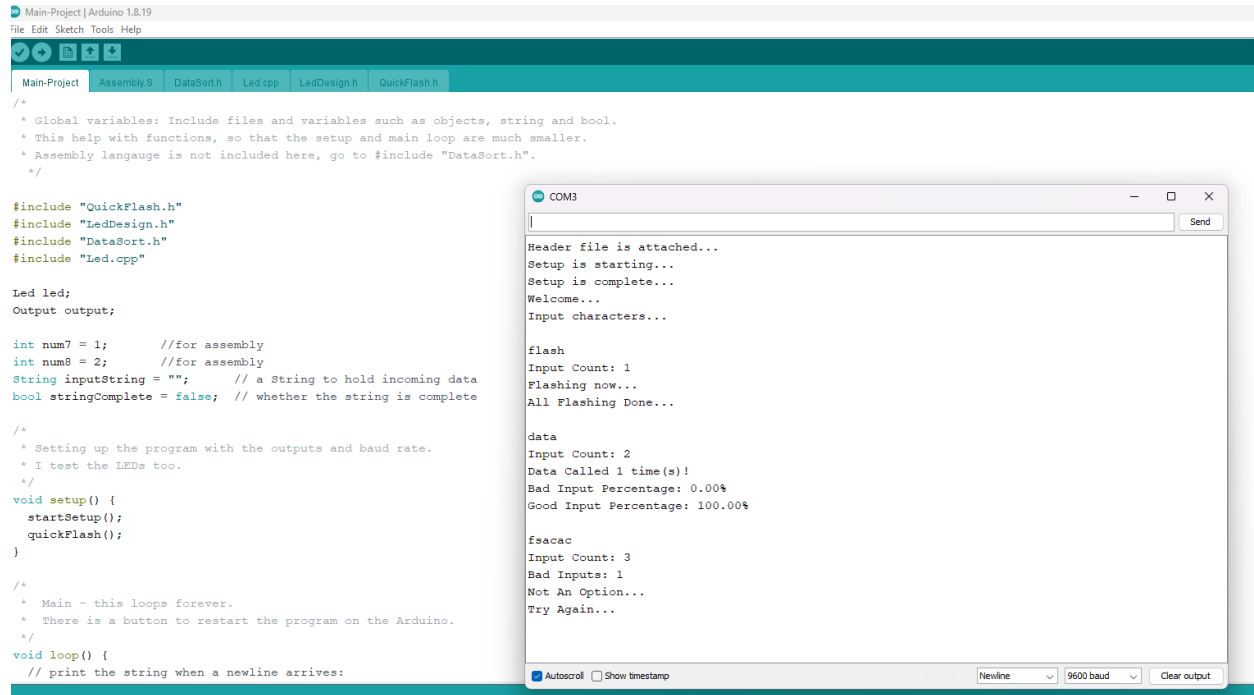
## COM Ports

The preference for Putty setup for Arduino communication:

- Set connection to "Serial" at the programmed baud speed rate. In my example, my Arduino program set the baud rate to 9600. Also, the PuTTY program will default to 9600. Adjust the COM1 to the correct COM Port. If you are unsure, you can go to the Arduino IDE to find out what COM Ports are active. For me, it is COM3 (Arduino) and COM5 (RS-232).
- If you adjust the program to take input, this step is important to do. Otherwise, you can skip this step. Click "Terminal" and check the box for "Implicit CR in every LF". Also, "Force on" the "Local echo". Adjust "The Function keys and keypad" under "Keyboard" to Linux. Otherwise, you may get a strange output.
- Click on "Open" to start the program. Then you can communicate from the computer to the Arduino.



On the Arduino IDE, verify what COM Port you want to use under “Tools” and then “Port”. If “Port” is grayed out, check the connections. Then go to “Serial Monitor” under “Tools”. Verify baud rate is matching the coded rate. You may change the COMs to check other components and concepts. I can only use one COM port at a time but I can switch COM ports if needed and appropriate. However, I may need to reset the Arduino if I want to check if it completely works regardless what program I use.



The screenshot shows the Arduino IDE interface. The main window displays a C++ sketch for an Arduino Uno. The sketch includes several header files and defines global variables. It sets up an LED and a serial port. The setup function initializes the LED and the serial port. The loop function prints the input string when a newline arrives. The Serial Monitor window is open, showing the output of the sketch. The output includes the setup sequence, the input string, and the flashing status.

```

/*
 * Global variables: Include files and variables such as objects, string and bool.
 * This help with functions, so that the setup and main loop are much smaller.
 * Assembly language is not included here, go to #include "DataSort.h".
 */

#include "QuickFlash.h"
#include "LedDesign.h"
#include "DataSort.h"
#include "Led.cpp"

Led led;
Output output;

int num7 = 1;    //for assembly
int num8 = 2;    //for assembly
String inputString = "";    // a String to hold incoming data
bool stringComplete = false; // whether the string is complete

/*
 * Setting up the program with the outputs and baud rate.
 * I test the LEDs too.
 */
void setup() {
  startSetup();
  quickFlash();
}

/*
 * Main - this loops forever.
 * There is a button to restart the program on the Arduino.
 */
void loop() {
  // print the string when a newline arrives:

```

The Serial Monitor window shows the following output:

```

Header file is attached...
Setup is starting...
Setup is complete...
Welcome...
Input characters...

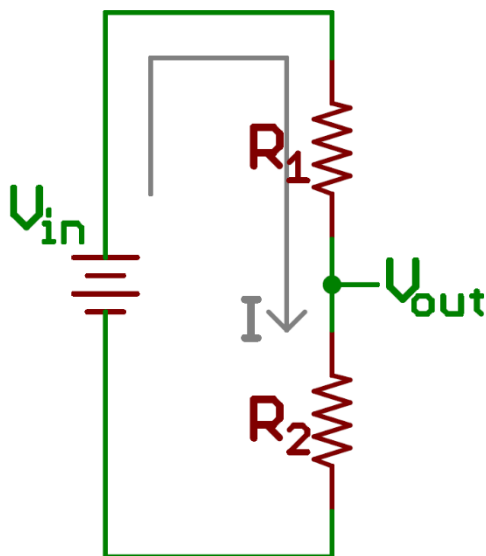
flash
Input Count: 1
Flashing now...
All Flashing Done...

data
Input Count: 2
Data Called 1 time(s)!
Bad Input Percentage: 0.00%
Good Input Percentage: 100.00%

fsacac
Input Count: 3
Bad Inputs: 1
Not An Option...
Try Again...

```

## A-Series Input & Output



This was to get creative for more inputs and outputs was to implement inputs from the Analog Port. One of multiple concepts that I implemented is with a photo-resistor and a regular resistor as a voltage divider. Another implementation was getting co-ordinates of the joystick's position. The joystick could have part of a larger project. However, I did not want to lose focus of my main goals in my overall project of communication and LEDs.

## Other Info

Pear Tutoring was helpful with building certain parts of my code. While most tutors are unfamiliar with the hardware aspect, they provided useful insight for going into depth with the code I was working with. While the Arduino and the Raspberry Pi are more of a device for hobbyists, this is one way to get people an intro into embedded systems.

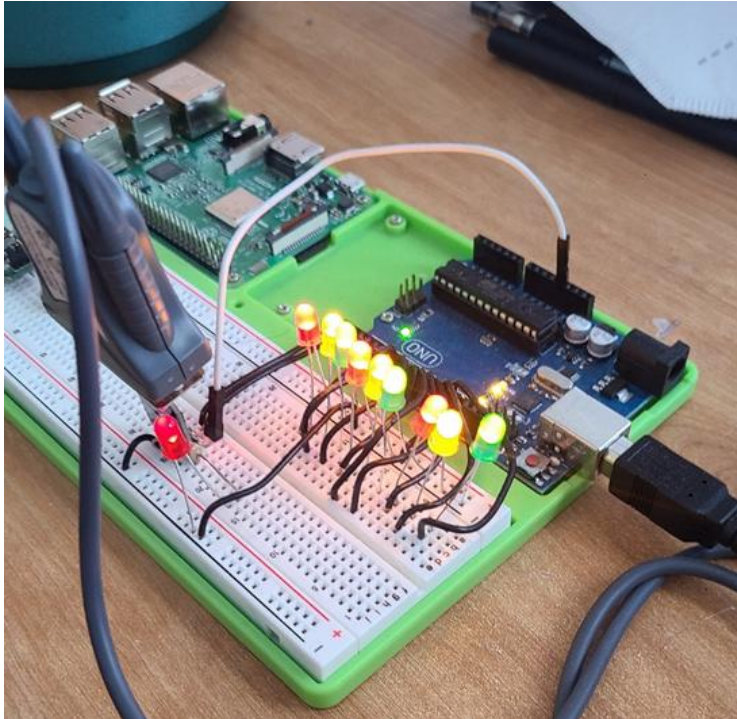
I started a YouTube channel with my Arduino product. This was great experience for me because it boosted my confidence for programming and showcasing my projects. I may continue doing these videos, even after finishing this project, semester, and program.

## My attempts

- I was having issues saving a .txt file to the computer
  - I can save it to an external SD card reader but I needed additional Arduino hardware.
  - I needed more TX/RX communication or other work arounds, which I did not have with the Arduino Uno or other components.
  - While some hobbyists on blogs say I could save a .txt file directly to my computer. However, I could not figure out how to save a .txt file directly to my computer.
- There are many languages for programming. There are many inline programming styles but I wanted to have separate program files of that code style. However, I was limited to the programming-style regardless on how I set up program without much more advance knowledge.
  - Languages that I attempted to add was primarily Python. Users either write in C / C++ or python, not typically both.
  - I tried Bash and Java. These are uncommon languages for the Arduino.
- My memory limitation issues with struct and a few other program concepts. This caused the input to act very limited. This is one sign there is a memory management issue. While some issues were solved, the struct was problematic for the Arduino.
  - I did a struct as a separate file. It worked fine.

## Summary

I learned a lot about Arduinos. This adds to my passion for embedded systems. The future needs more people who have skills with hardware and software. There are big opportunities for me to explore my passion and gain wonderful employment.

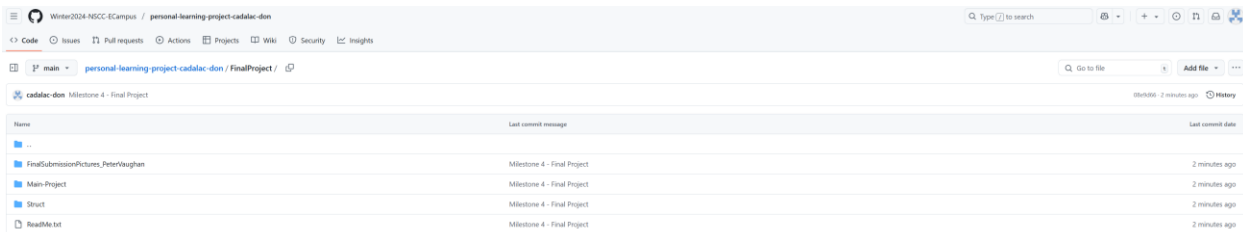


There are many concepts that I applied to the code such as COM Ports, Arduino libraries, random numbers, header files and inheritance. A few tutors and blogs were helpful when I was developing this project even further.

There are many concepts that I learned with the Arduino that could be applied to the Raspberry Pi. The project concepts are limitless. Overall, it was a great experience to explore this. This will help my future in embedded systems.

## Submission

- My Explanation Video and Presentation
- PDF Documentation
- GitHub: <https://github.com/Winter2024-NSCC-ECampus/personal-learning-project-cadalac-don>
- ReadMe File with Pictures
- FinalSubmissionPictures\_PeterVaughan Folder
- Arduino YouTube: [https://www.youtube.com/channel/UCOv\\_iZCx4CW5Y9S8YzXDdgg](https://www.youtube.com/channel/UCOv_iZCx4CW5Y9S8YzXDdgg)



The screenshot shows the GitHub interface for the repository 'personal-learning-project-cadalac-don'. The commit history table is as follows:

Name	Last commit message	Last commit date
..		
FinalSubmissionPictures_PeterVaughan	Milestone 4 - Final Project	2 minutes ago
Main-Project	Milestone 4 - Final Project	2 minutes ago
Struct	Milestone 4 - Final Project	2 minutes ago
ReadMe.txt	Milestone 4 - Final Project	2 minutes ago

## Rubric Outcomes Completed

- Coding Knowledge
- Requirement as learning contract
- Program Structure
- Problem Solving and Efficiency
- GitHub Repo and README File
- Classroom Presentation or Recording