Fundamentals of Applied Microcontrollers Laboratory Manual

Seth McNeill

Edition Fall 2022 (v0.5)
2022 September 13

Contents

Chapter 1

Introduction

This book is the accompanying lab manual to a class introducing microcontrollers to upper division, non-electrical engineering undergraduate students who have taken some C programming.

If you find this useful, please let me know. If you find any errors, areas that need improvement, or have any improvements to add please let me know.

The class textbook/manual is here.

1.1 License

Chapter 2

Arduino Startup

2.1 Installing the IDE

We want to try installing the IDE at least two different ways. First, on the lab computer, then on your personal computers if you have them. Both lab partners should try installing the software on the lab computer, each with their own login.

2.1.1 Lab Computer

This method may also work on your personal computers.

1. On the search bar, type in Microsoft Store.
2. Click on Microsoft Store
3. In the store search, type arduino
4. Arduino IDE App should appear, click on it
5. Click on Get
6. You do not need to sign into Microsoft to make this install work even if prompted
7. Once it has installed, run the Arduino IDE app.
8. It should load up with a window that looks like Figure 2.1.

9. Try rebooting the computer, logging in and seeing if the Arduino app is still present.

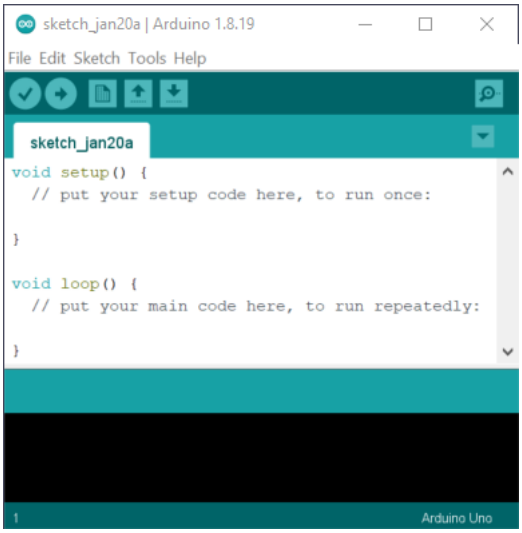

Figure 2.1: This is what the Arduino IDE should load up to.

2.1.2 Personal Computer

1. Go to software download page: https://www.arduino.cc/en/software
2. Download the Windows ZIP file (not the first link or the app)
3. Open the zip file and copy the folder inside (arduino-1.8.19 as of this writing) into your One Drive folder. This may take a while. If you are on your own computer, you can use any of the programs.
4. Once that transfer finishes, go into the folder and run arduino.exe. Windows will try to save you, but if you click More Info you can click Run Anyway.
5. Windows Defender Firewall will also complain. Uncheck the box that is checked and/or click Cancel.
6. It should load up with a window that looks like Figure 2.1.

2.2 Testing the Setup

2.2.1 Installing the Board Drivers

1. In order to get it to connect correctly to your board, you need to install the Arduino Nano Connect RP2040 board.

   (a) Navigate to Tools→Board: "Arduino Uno" (or similar)→Boards Manager

   (b) It should load as shown in Figure 2.2.

Figure 2.2: This what the Boards Manager loads up to.

   (c) In the search bar, type "arduino nano connect" (without the quotes)

   (d) The first item should be Arduino Mbed OS Nano Boards and should list the Arduino Nano RP2040 Connect.

   (e) Move your cursor over it and it should show an Install button. Click it to install the board library.

   (f) Wait for it to finish.

   (g) While you are waiting, plug your Nano Connect into your computer and let it install it.

(h) As it finished, I received a User Account Control warning asking if I wanted to let dpinst-amd64.exe make changes to my device. I said yes.

(i) Next it asked me if I wanted to install Arduino Universal Serial Bus devices. Again, click to Install.

(j) It popped up again and I clicked Install again. Now it should say that the Arduino Mbed OS Nano Boards has been installed.

(k) Close the Boards Manager.

2.2.1.1 Testing the Setup

This section can be done on either (or both) computers. The results from one run (between both lab partners) is all that needs to be turned in.

1. Now go to Files→Examples→01.Basics→Blink.
2. This will open another window with the Blink program.
3. Go to Tools→Board→Arduino Mbed OS Nano Boards and select the Arduino Nano RP2040 Connect
4. Go to Tools→Port and select the COM that isn't COM1 (mine showed up as COM5)
5. Click the right arrow under the word edit in the menu to Uplaod the sketch to the Arduino board.
6. It should say "Compiling sketch..." in the lower left and show a progress bar on the lower right.
7. Then it should switch to Uploading... and finally Done Uploading.
8. An orange light near the USB port on your board should be blinking.
9. Congratulations! You have programmed your board!
10. Now look in the program for the two delay statements. Try changing the values inside the parentheses and re-uploading it. Does the blinking change?

11. In order to save files and have it portable, you need to change the directory where the Arduino IDE stores it's sketchbooks
    (a) Go to File→Preferences
    (b) Change the Sketchbook location to your OneDrive and a folder named arduino (lowercase is good)
    (c) My OneDrive was in C:\Users\mcneils2\OneDrive - Embry-Riddle Aeronautical University\arduino
12. Now try saving the blink sketch with your changed values.
13. Demonstrate your working blink and it's storage location to your instructor/TA
14. Here are some other Examples to test:
    (a) Basics → fade: change the variable led to be LED_BUILTIN, watch the red/orange LED pulse
    (b) Digital → DigitalInputPullup: Change the first pinMode call to use A0 instead of 2. The same for the digitalRead command (2→A0). Press the right button (SW1) and see the LED blink. Note that this program isn't written as well as the others since you have to change a number in two places. Could you rewrite it better?
15. Finally, create a sketch called `getIDs` using the code at https://github.com/semcneil/CEC325Examples/blob/main/getIDs/getIDs.ino
16. You will need to install two libraries to make this script run.
    (a) Go to Sketch → Include Library → Manage Libraries
    (b) Install the ArduinoECCX08 and OneWireNG libraries
17. Run `getIDs` and submit the results in the end of lab Canvas quiz.

2.3 Turn In

1. Make sure that the TA or instructor has signed off on your modified blink sketch.

2. Fill out the end of lab quiz prior to leaving. Note that it includes asking you for the output of the `getIDs` sketch.

Chapter 3

Multiplexer, LED Display, Binary, HEX

3.1 Purpose

The goal of this lab is to use binary and hexadecimal numbers in an applied setting. This is achieved by having you (the student) use the Arduino Nano Connect RP2040 to setup the PCA9535 I^2C Input/Output (IO) port chip to drive a 4-digit LED display.

3.2 Resources

1. PCA9535 datasheet
2. LED display datasheet
3. PCA9535 library (McNeill version)
4. PCB Schematic and Layout - see class manual in the Arduino Startup → Schematics and PCB section

3.3 Procedure

3.3.1 Add PCA95x5 library

1. Open the Arduino IDE

2. Open preferences
3. Note the Sketchbook location
4. Go to the website for the PCA9535 library (see 3.2 Resources section)
5. Click the green Code button and then Download ZIP
6. Save the zip file to your Downloads folder or somewhere else you can find it again
7. In the Arduino IDE select Sketch → Include Library → Add .ZIP Library
8. Select the zip file you downloaded earlier
9. Wait for the IDE to say "Library added to your libraries"
10. Check that the library is indeed installed one of two ways:
    (a) Go to Sketch → Include Libraries and look for PCA95x5
    (b) Go to File → Examples and look for the PCA95x5 examples
11. The PCA95x5 library is now added successfully.

3.3.2 Turn on some LEDs

An example script is shown in Listing 3.1. Use this as a starting point for your code. Be sure to change the header information to include all lab partner names and the correct date. This script displays an 8. in the first digit and zeros in the rest. Note the clearing lines to make it so that there is not ghosting of numbers to the right of where they are displayed.

1. Run the script in Listing 3.1 to make sure it runs
2. Change the script so that it displays four consecutive numbers such as 1234. You can use any 4 consecutive, single digit numbers.
3. Demonstrate your sketch working to a TA or instructor.

```
 /* 20220907LEDsClass1.ino
 *
 * Demo of LED display in class.
 *
 * Seth McNeill
 * 2022 September 07
 */

#include <PCA95x5.h>  // include library

PCA9535 LEDmux;  // create instance (object) of
library

void setup() {
  Serial.begin(115200);
  while(!Serial);
  Serial.println("Starting...");

  // initialize object
  Wire.begin();
  LEDmux.attach(Wire, 0x21);
  LEDmux.polarity(PCA95x5::Polarity::ORIGINAL_ALL);
  uint16_t mux_direction = 0;
  mux_direction = 0x0010;  // mostly outputs
  LEDmux.direction(mux_direction);

  Serial.println("Everything setup");
}


void loop() {
  int delayTime = 0;
  // use object
  LEDmux.write(0x000F);  // required to remove
ghosting on other digits
  LEDmux.write(0xFF0E);
  delay(delayTime);
  LEDmux.write(0x000F);  // required to remove
```

```
  ghosting on other digits
    LEDmux.write(0x3F0D);
    delay(delayTime);
    LEDmux.write(0x000F);  // required to remove
  ghosting on other digits
    LEDmux.write(0x3F0B);
    delay(delayTime);
    LEDmux.write(0x000F);  // required to remove
  ghosting on other digits
    LEDmux.write(0x3F07);
    delay(delayTime);
  }
```

Listing 3.1: This listing is a starting point for driving the LED display. This sketch may also be available on Canvas.

3.3.3 Count

Make a new sketch, based off your first sketch (meaning copy and paste it into the new sketch). This new sketch should count up from 0 to 9 (or 0000 to 0009) incrementing once per second.

3.3.4 Extra credit

Have your counting system count higher than 9.

3.4 Turn In

Turn in the following:

1. A video of your board counting that includes both of your faces and you saying your names.
2. A PDF of your first sketch that displays 4 consecutive numbers.
3. A PDF of your second sketch that counts.
4. .ino versions of both sketches.