Introduction to Adafruit and the LCD Display – Hello, World!

Dr. Matthew Riehl

August 25, 2021

1 Introduction

For the next several weeks, we will be building circuits to investigate the properties of resistors, capacitors, and other electronic components. In order to do this, we first build a simple device that will be used to help us gather our data in later labs.

Note that this is NOT a programming class (no previous programming experience is needed); the code is provided and can be used without editing (although you are encouraged to play with the code to extend the functionality of the device...). Neither is this a circuits class; we will not be exploring the details of how the Arduino and other components work. Of course, we hope that this taste of programming and bread-boarding inspire you to take additional courses in computer science or electronics...

This week, we will set up the Adafruit Metro Express to send a message to the LCD display. You will wire the LCD to the Metro Express with the breadboard and jumper wires and use your computer to write the code and store it on the micro-controller. This code will be sent to the Metro Express which will execute the code until it is turned off or until the code is replaced with a different set of instructions. Once the code has been stored on the Metro Express, it can be disconnected from the computer and powered by any appropriate source.

In later labs, we will modify the circuit to gather data which will be displayed on the LCD screen. This has the benefit of allowing you to see what is going on inside the circuit to some extent. The downside is that any changes you make to your circuit or experiment will have to be coded into the program and loaded onto the micro-controller.

2 Materials

- Computer with the Mu IDE installed (codewith.mu/en/download[1])
- Adafruit Metro Express board and power cable
- Half-size breadboard
- LCD display module
- 220 Ω resistor (Ω is Ohm)
- 10 kΩ potentiometer
- Jumper wires

[1] for this exercise, Mu IDE 1.1.0-beta-5 was used

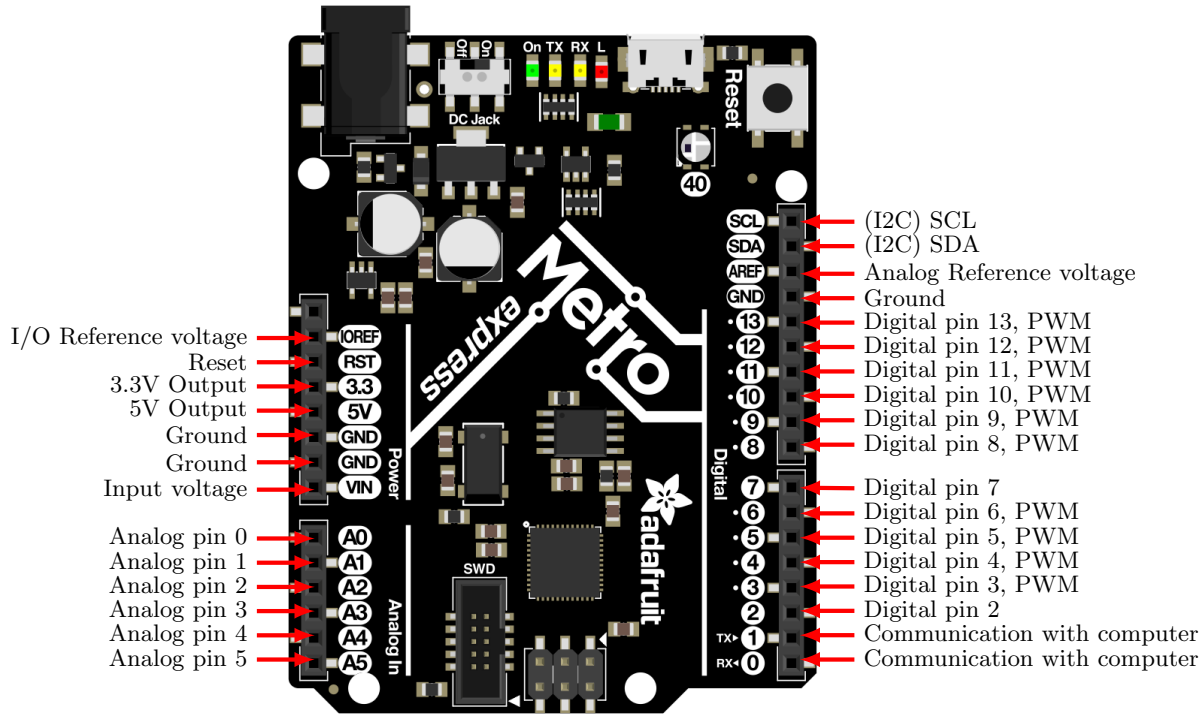

Figure 1: Pin-out Diagram for the Adafruit Metro Express. PWM is Pulse Width Modulation

3 Wiring the LCD to the Metro Express

We will build circuits by copying diagrams given in the handout. In some cases, the functions of pins will be described, but mostly we will leave the details to the engineers and programmers who designed the boards and accessories. Pin-out diagrams of the Metro Express (Figure 1) and the LCD board (Figure 2) are shown.

Some pins will not be used for our projects.

The wiring diagram is shown in Figure 3. It is useful to use the color of wire suggested in the diagram, because it will be easier to troubleshoot if the board does not work. It is suggested that you wire the breadboard first (black, red, and blue wires) and then connect the Metro Express with final wires.

You will need to find a 220Ω resistor. The resistance of each resistor is coded on the body by means of a series of colored bands. Most resistors have either 4 bands or 5 bands. See figure 5 for a description of the resistors and the colors used and figure 6 for an example of how to read the bands.

4 Code for the Metro Express

The code that we will use is written in the computer language "Python", which is often the first programming language students choose to learn. The code that will be used today is given below. Note that the text that occurs between """ and """ are comments, not part of the actual code. Also, any line that begins with # is a comment. If you delete all of the comments, the code is quite small.

This code is in the file Adafruit_Basic_Hello_World.py. To open the file for viewing or editing, first save it to your computer. Start the Mu IDE and load the program via the *load* button. To load the program onto the Metro Express, copy the file into the drive CIRCUITPY and rename it to "code.py", overwriting any existing

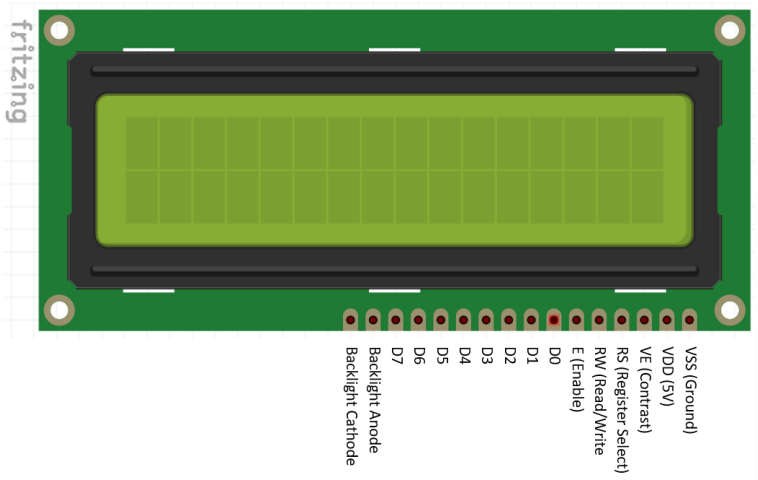

Figure 2: Pin-out Diagram for the LCD

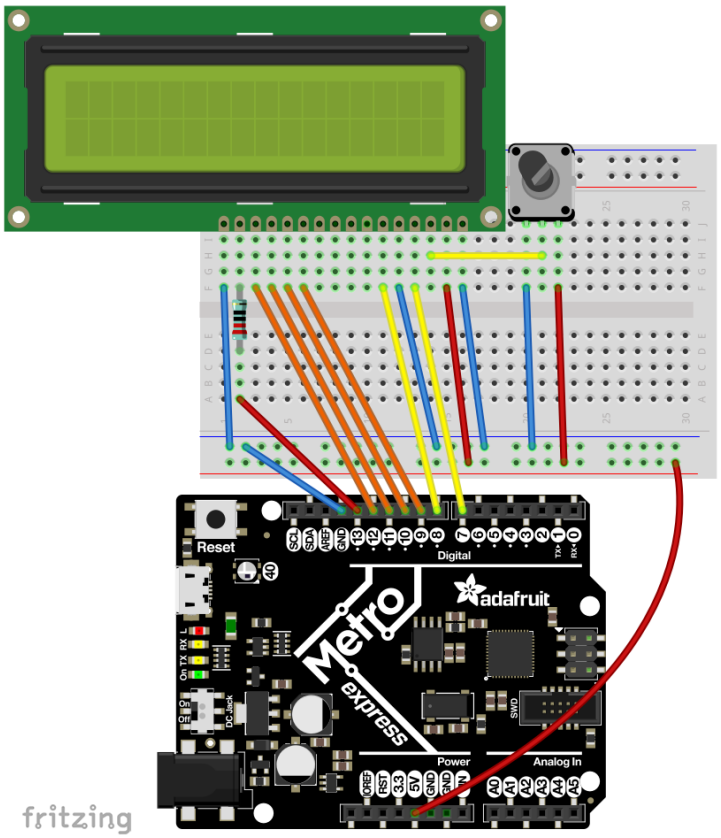

Figure 3: Hello World Wiring Diagram

code.py file. If you load “code.py” into the Mu IDE from the CIRCUITPY drive, it will automatically save to that location when you save the file.

The file Adafruit_Hello_World.py demonstrates a number of commands available for the LCD. See circuitpython.readthedocs.io/projects/charlcd/en/latest/api.html for a complete list of commands that can be used with the LCD.

```
"""
Simple test for monochromatic character LCD

This code is included with the files for Adafruit Labs Hello World by
Matthew Riehl and is in the public domain.  I have only modified the
code to better support the laboratory exercise.

Demonstrates the use a 16x2 LCD display.  The LiquidCrystal
library works with all LCD displays that are compatible with the
Hitachi HD44780 driver. There are many of them out there, and you
can usually tell them by the 16-pin interface.

This program prints "Hello World!" to the LCD
and shows the number of seconds since reset.


The circuit:
* LCD RS pin to digital pin 7
* LCD Enable pin to digital pin 8
* LCD D4 pin to digital pin 9
* LCD D5 pin to digital pin 10
* LCD D6 pin to digital pin 11
* LCD D7 pin to digital pin 12
* LCD Backlight (pin 15 on LCD) to pin 13 on Adafruit (add a 220 ohm series resistor)
* LCD R/W pin to ground
* LCD VSS pin to ground
* LCD VCC pin to 5V
* 10K potentiostat:
* ends to +5V and ground
* wiper to LCD VO pin (pin 3)

See https://learn.adafruit.com/character-lcds/python-circuitpython for more information.

"""
import time
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd


# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# Metro M0/M4 Pin Config:
```

```
lcd_rs = digitalio.DigitalInOut(board.D7)
lcd_en = digitalio.DigitalInOut(board.D8)
lcd_d7 = digitalio.DigitalInOut(board.D12)
lcd_d6 = digitalio.DigitalInOut(board.D11)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D9)
lcd_backlight = digitalio.DigitalInOut(board.D13)

# Initialise the LCD class
lcd = characterlcd.Character_LCD_Mono(
lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_backlight
)

start = time.time()
lcd.clear()
lcd.backlight = True
lcd.message = "Hello, World!"  # Change this to something cooler.

while True:  # This loop continues forever!
now = time.time()
lcd.message = ('\n{}  seconds'.format(int(now-start)))
```

5 Hello World Program in the Mu Editor

Connect the Metro Express to the computer with the cable. Open the Mu IDE, identified by the logo shown in Figure 4. In the "Mode" menu, select *CircuitPython* as the programming mode. From the "Load" dialog, find the CIRCUITPY drive and open it. There will be one or two .py files: main.py and (probably) code.py. Code.py is the file that will hold your code and be executed by the micro-controller. Main.py is a default program that will run if code.py is absent or corrupt. If code.py is not in the CIRCUITPY drive, you can drag a .py file to that location and rename it.

Select code.py to open the file in the Mu editor. Then, in the same menu, find "Adafruit_Basic_Hello_World.py" that you saved to your computer and open that file. Copy and paste this code into code.py and save this file. It will save to the Metro Express and immediately begin executing the commands. You may need to adjust the potentiometer to see the display on the LCD.

Figure 4: Mu Editor Logo

Now, the file Adafruit_Hello_World.py can be opened and saved to the code.py file on the CIRCUITPY drive. Take some time to read the code and try to modify it to see what it can do. More details on how the commands work can be found online (see the links in the comment section of the program) – this project is not intended to teach the details of programming. Rather, it is hoped that you will explore on your own.

Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	±1%
Red	2	2	2	100Ω	±2%
Orange	3	3	3	1kΩ	
Yellow	4	4	4	10kΩ	
Green	5	5	5	100kΩ	±0.5%
Blue	6	6	6	1MΩ	±0.25%
Violet	7	7	7	10MΩ	±0.10%
Grey	8	8	8	100MΩ	±0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	±5%
Silver				0.01Ω	±10%

Figure 5: Color banding codes for resistors

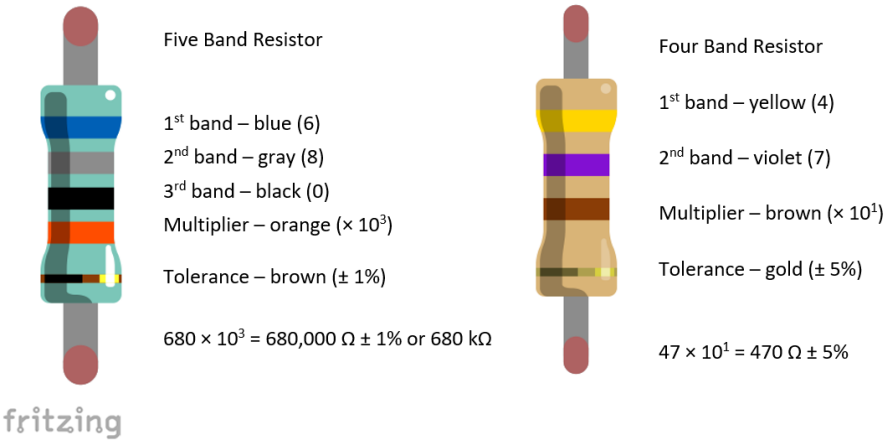

Figure 6: Examples of a 5-band and 4-band resistor