SOUTHERN ILLINOIS UNIVERSITY
CARBONDALE

ECE 296L-002 Intro to Software & Robotics Lab

Lab 4: LCD with Custom Characters

Performed by:

Chase Lotito
856093429

Instructor: Dr. James Phegley
TA: Muhammad Zia Hameed

February 28, 2024

Table of Contents

Introduction

The goal of this experiment to set up the graphical display for our eventual object-tracking robot, but first we must learn how to use the LCD screen to display any kind of information we want.

We will learn about proprietary functions that can interface with the hardware on the LCD screen---like how to add custom-made characters to a special register for arrows. This will lead us to defining arrow characters using hexadecimals to describe the bits that need to be displayed.

Assessment of Design

The design is inspired and guided by the design provided by Dr. James Phegley in the Lab 4 manual [1].

Overall, the LCD is controlled with user-defined commands in a separate python file named *myLCD.py* that was provided by the professor. This file contains that functions that streamline setup, configuration, clean up, displaying, and custom characters. We can interface with this file via python and then we can control the LCD using python running off a Raspberry Pi.

The custom character function works with the Raspberry Pi GPIO library to send the proper signal from the Pi to the LCD. Then, we can provide the custom character to the LCD with the *lcd_custom()* function---this function lives in the *myLCD.py* file. Finally, we provide the data associated with our custom character.

The characters on the LCD are 5 by 8 pixels, and the LCD has 2 rows of 16 characters. Imagining the 40-bit grid that constructs a character, each row of bright-pixels and low-pixels can be described using hexadecimal numbers, as our 2-bit hexadecimal values describe 5-bit binary values. In python, we can create a list of 8 hexadecimal values that contain the information for our characters.

In total, we can only write 8 characters, but we will only need 5 for the different directional arrows.

Overall, the design works to display all of the arrow characters on the first line of the LCD. Functionality to make the characters cycle with time could be done easily with a for loop. Also, I implemented my name via custom characters, even though the *lcd_string()* function can easily handle printing strings.

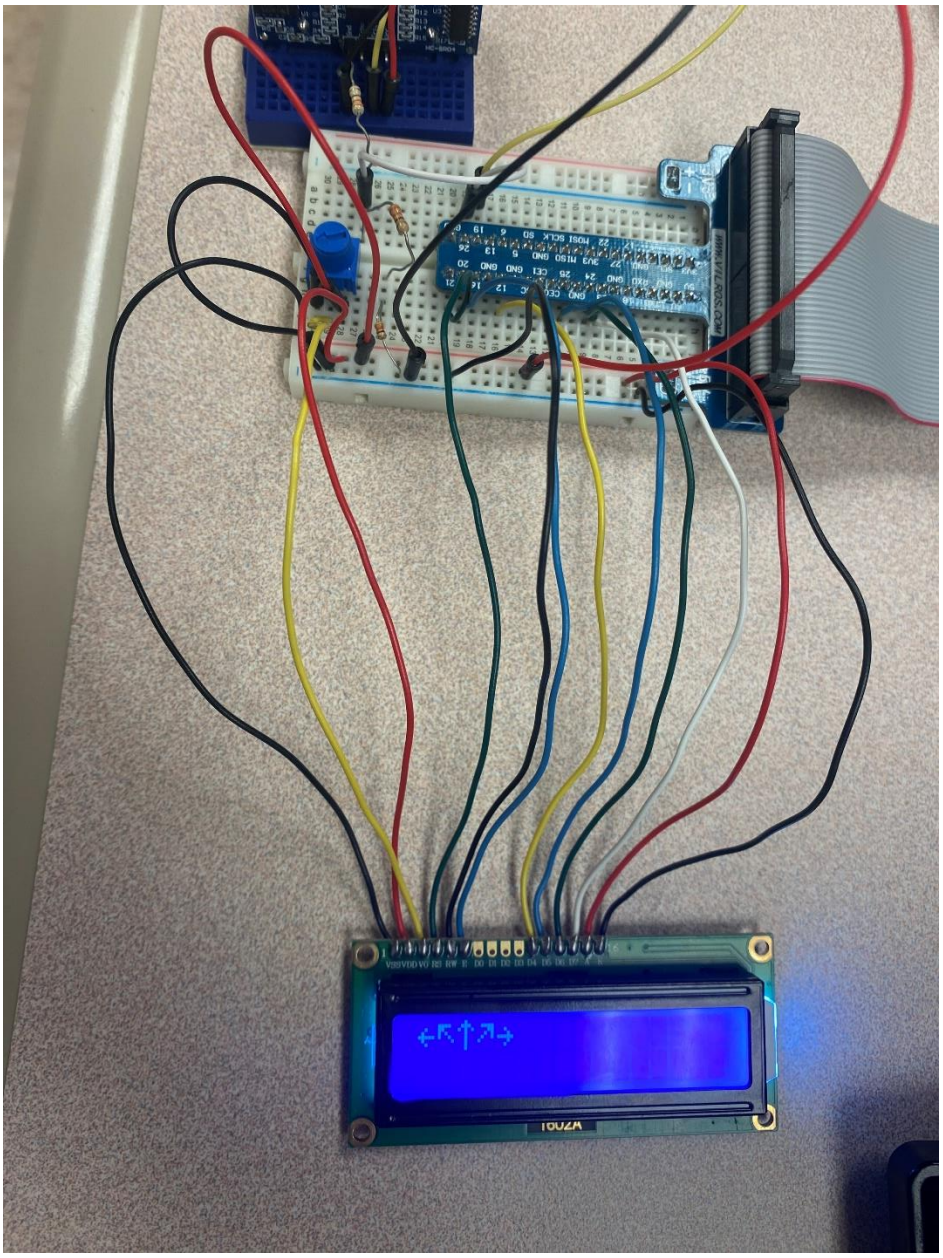


Figure 1: Circuit with custom arrow characters

Figure 2: Circuit with custom characters displaying my name

Conclusion

This lab was the foundation before integrating the LCD screen into our object-tracking robot.

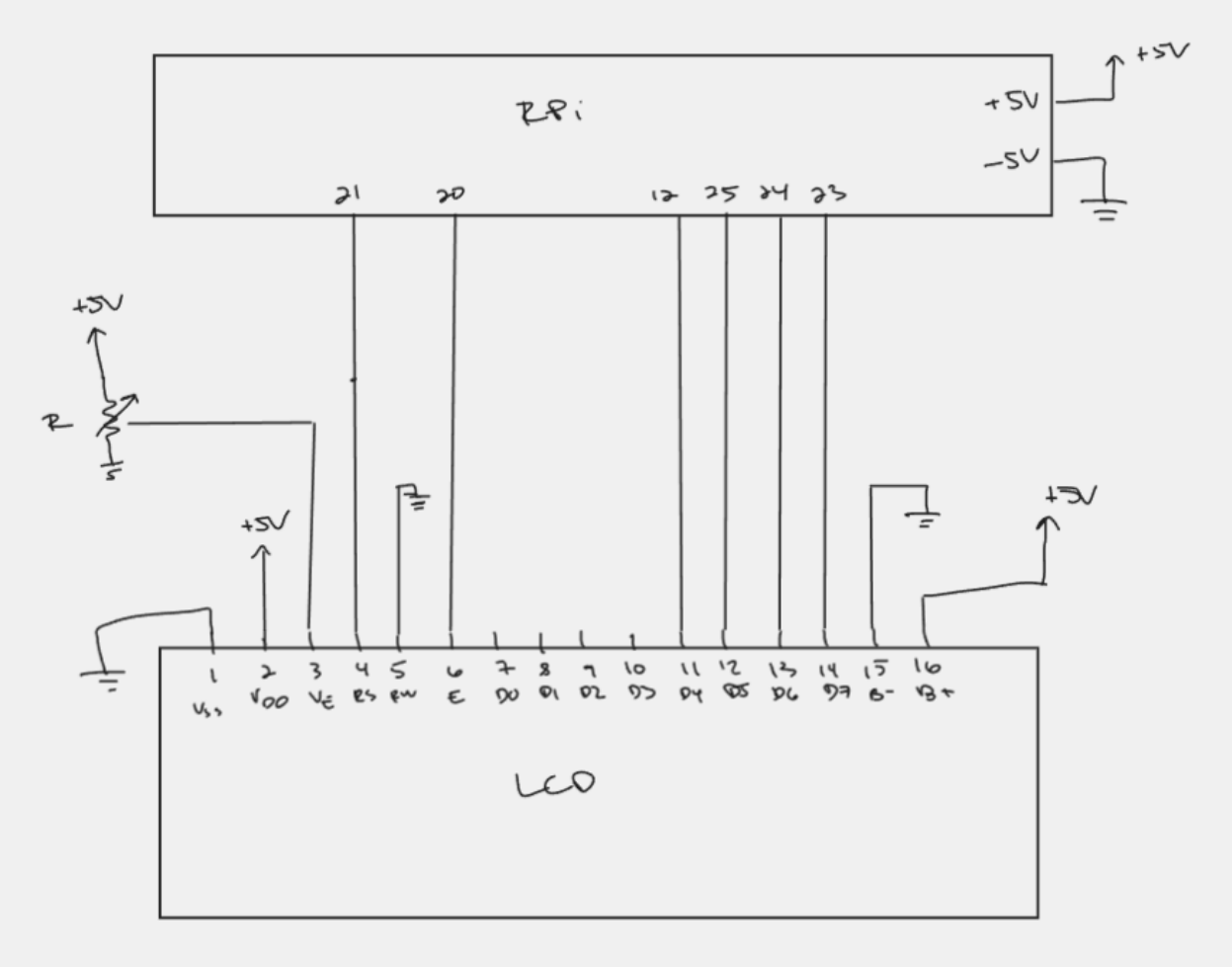
Having skills to implement a visual display on a product is paramount for making electronics that appeal to a mass consumer base. The screen drastically increases the usability and perceived functionality of a device.

Since high quality monitors cannot always be feasible for a product, it is important to understand simpler devices that require embedded programming to display information.

References

[1] J. Phegley, "Lab Number 4," SIUC ECE Dept, Carbondale, 2024.

Appendix A: Hardware Schematic



Appendix B: Code for the Software Developed

```
'''
Chase Lotito - SIUC - SP2024
ECE296L - Dr. James Phegley
RPi Lab 4 - Code from Phegley
LCD screen with CUSTOM CHARS
'''
import RPi.GPIO as GPIO
from myLCD import *
from time import sleep, time # time() returns time in sec since EPOCH

# RPi Pin Setups
RS = 21 # Register Select Pin
E = 20 # Enable Pin
D4 = 12 # Data Pin 4
D5 = 25 # Data Pin 5
D6 = 24 # Data Pin 6
D7 = 23 # Data Pin 7
BKL = 1 # Backlight

# Useful parameters
LN1 = 1
LN2 = 2

GPIO.setmode(GPIO.BCM) # BROADCOM PINS

# LCD Initialization Functions
start(GPIO, 2, 16) # define size of display
LCD(GPIO, RS, E, D4, D5, D6, D7, BKL) # define LCD pins
lcd_init(GPIO) # initialize LCD
backlight(GPIO, True) # turn on backlight always
```

```
# Custom Arrow Characters!
westArrow = [0x00, 0x00, 0x04, 0x08, 0x1F, 0x08, 0x04, 0x00]
northWestArrow = [0x00, 0x1E, 0x18, 0x14, 0x12, 0x01, 0x00, 0x00]
northArrow = [0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04, 0x04]
northEastArrow = [0x00, 0x0F, 0x03, 0x05, 0x09, 0x10, 0x00, 0x00]
eastArrow = [0x00, 0x00, 0x04, 0x02, 0x1F, 0x02, 0x04, 0x00]

# Characters for Chase
'''
C = [0x0E, 0x11, 0x10, 0x10, 0x10, 0x10, 0x11, 0x0E]
H = [0x11, 0x11, 0x11, 0x1F, 0x11, 0x11, 0x11, 0x11]
A = [0x0E, 0x11, 0x11, 0x1F, 0x11, 0x11, 0x11, 0x11]
S = [0x0E, 0x11, 0x10, 0x0E, 0x01, 0x01, 0x11, 0x0E]
E = [0x1F, 0x10, 0x10, 0x1E, 0x10, 0x10, 0x10, 0x1F]
'''

# Store the custom characters in myLCD
lcd_custom(GPIO, 0, westArrow)
lcd_custom(GPIO, 1, northWestArrow)
lcd_custom(GPIO, 2, northArrow)
lcd_custom(GPIO, 3, northEastArrow)
lcd_custom(GPIO, 4, eastArrow)
'''
lcd_custom(GPIO, 0, C)
lcd_custom(GPIO, 1, H)
lcd_custom(GPIO, 2, A)
lcd_custom(GPIO, 3, S)
lcd_custom(GPIO, 4, E)
'''
try:
    while True:
```

```
        # Display on LCD!
        # lcd_string(GPIO, "TIME: %.2f ms" % (trip_time * 1000), LN1)
        # lcd_string(GPIO, "DIST: %.1f cm" % (distance), LN2)
        lcd_string(GPIO, chr(0) + chr(1) + chr(2) + chr(3) + chr(4),
LN1)

        # Do a forever while loop to show the custom arrows changing
        i = 0
        while (i < 5):
            lcd_string(GPIO, chr(i), LN2) # Print the current arrow
            i += 1                        # set the next arrow
character
            sleep(1)
            if (i == 5):                  # once fully cycled -> reset
                i = 0


finally:
    lcd_shutdown(GPIO)
    GPIO.cleanup()
    print('PROGRAM TERMINATED')
```