EEE174 –CPE185 INTRODUCTION TO MICROPROCESSORS

LAB 9 – MATRIX KEYPAD

Lab Session: Wednesday 6:30PM - 9:10PM

Section 32385

Lab Instructor: Sean Kennedy

Student Name: Andrew Robertson

TABLE OF CONTENTS

PART 1

OVERVIEW

This lab is part 4 of 5 for the series of do it yourself labs geared towards helping with the final project. Since my group project involves many steps of opening and creating files all under differing timelines I thought it may be a good idea to be able to control states manually. A matrix keypad would be able to serve as the array of buttons it is so that we can set different labeled buttons to control different states.

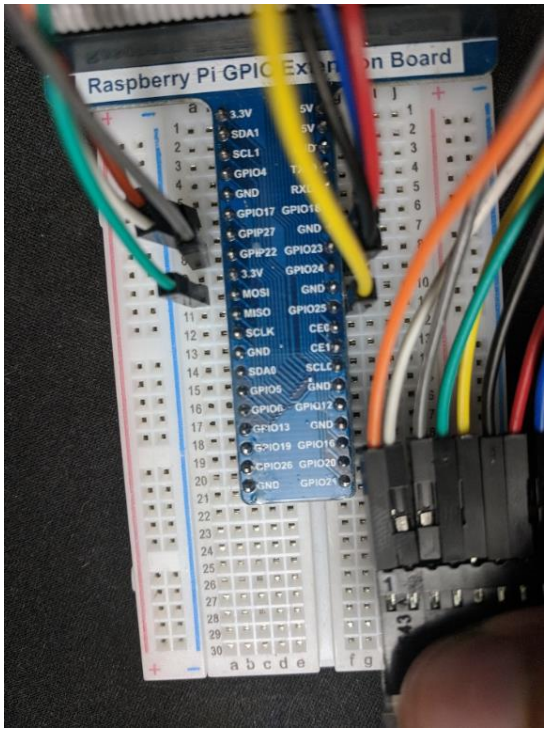
Lab Discussion

Work Performed / Solution:

Not much was required on my part to get this matrix keypad working so I'd like to talk about the idea behind it for a bit before the steps taken to make it work. A matrix keypad seems to be a grid of buttons that simply connect a column to a row. By doing this, you can devise what button was pressed by positional association and deal with it however you would like on the software end. Using the 4 rows and columns available via this keypad as an example, if 4 different signals total were sent through the 4 rows, we would only need to watch the columns for two things. First, which column received a signal and second, what the signal was.

The provided code is doing this with some added features. Knowing that buttons that have not been debounced can send many signals on one button press, there is a software debounce added to combat this issue.

To get the keypad working I needed to hook up the column pins and the row pins to GPIO and open nano to create the two provided programs for this keypad. Once this was done, all that was left was to launch the top program and test the keypad. The terminal output can be seen to right and my wiring on the left.



```
pi@raspberrypi:~ $ python Matrixkeypad.py
Program is starting ...
You Pressed Key : 1
You Pressed Key : 2
You Pressed Key : 3
You Pressed Key : 4
You Pressed Key : 5
You Pressed Key : 6
You Pressed Key : 7
You Pressed Key : 8
You Pressed Key : 9
You Pressed Key : 0
You Pressed Key : *
You Pressed Key : #
You Pressed Key : A
You Pressed Key : B
You Pressed Key : C
You Pressed Key : D
```

LISTING FILES(S):

Keypad code provided via:
https://github.com/Freenove/Freenove_RFID_Starter_Kit_for_Raspberry_Pi/blob/master/Code/Python_Code/22.1.1_MatrixKeypad/Keypad.py

Matrix Keypad code provided via:
https://github.com/Freenove/Freenove_RFID_Starter_Kit_for_Raspberry_Pi/blob/master/Code/Python_Code/22.1.1_MatrixKeypad/MatrixKeypad.py

Matrix Keypad Pinout provided via:
https://github.com/Freenove/Freenove_RFID_Starter_Kit_for_Raspberry_Pi/blob/master/Tutorial.pdf

CONCLUSION

While doing this lab I began to think this could be the way a basic keyboard works. When selecting a language or keyboard layout in an operating system all that would need to be changed is how the symbol grid is defined. The matrix of signals the keyboard would send must first go to an internal chip for interpretation before being sent over USB. The idea of using a battleship type concept for mapping a row and column to a location is very simple and scalable.