Lab 3: Analog to Digital and Digital to Analog Conversions

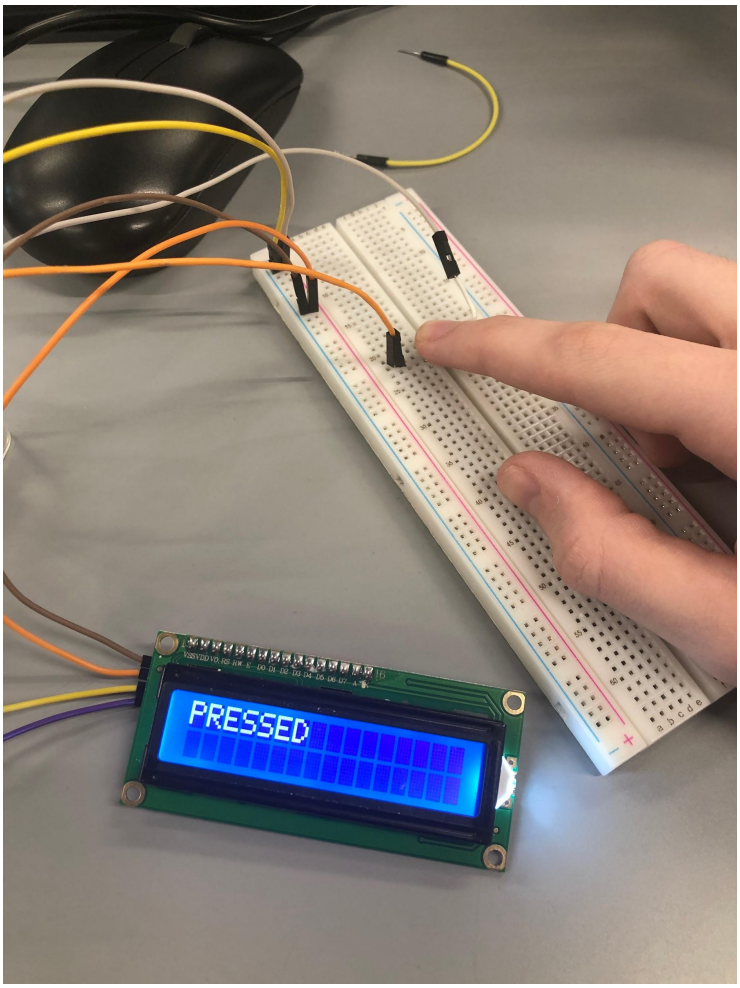
Luke McIntyre & Brendan Bovenschen

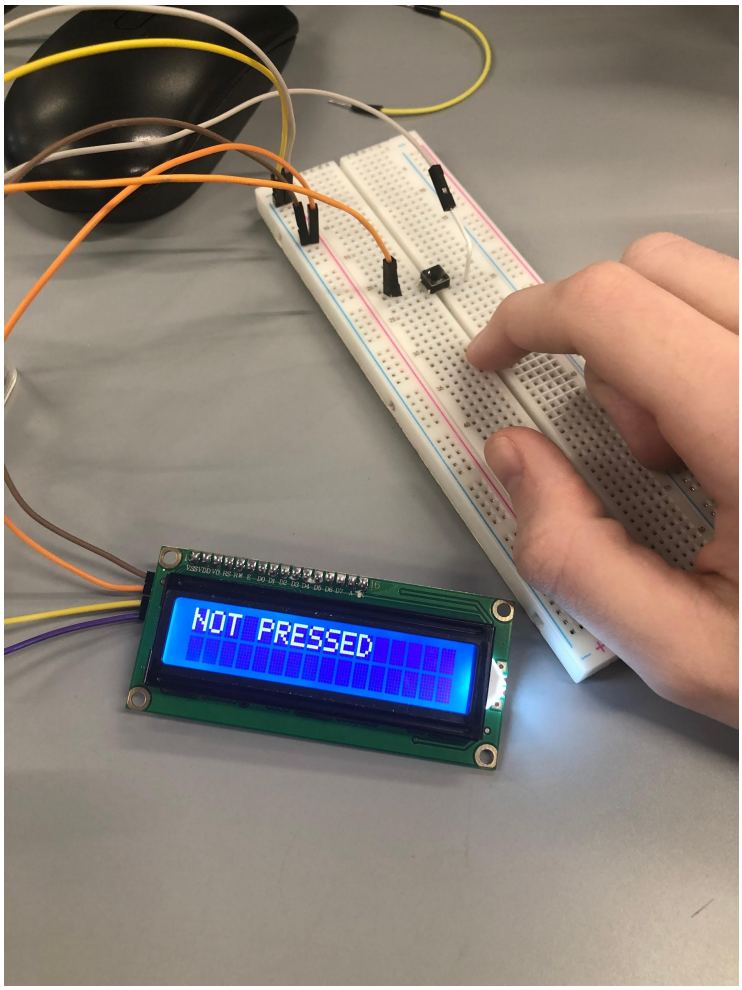
1 Exercise 1

Before implementing input to the LCD-Pi system, we must first understand sending a static output to the LCD. This is done using commands in the Raspberry Pi's I2C library for writing to an LCD. With a starting cursor X and Y coordinate input, we can print a C-string (static array of chars) at this cursor position and the rest of the characters will fill in sequentially. This is used to display our message on each row with the starting position of each line being (0,0) and (0,1).

2 Exercise 2

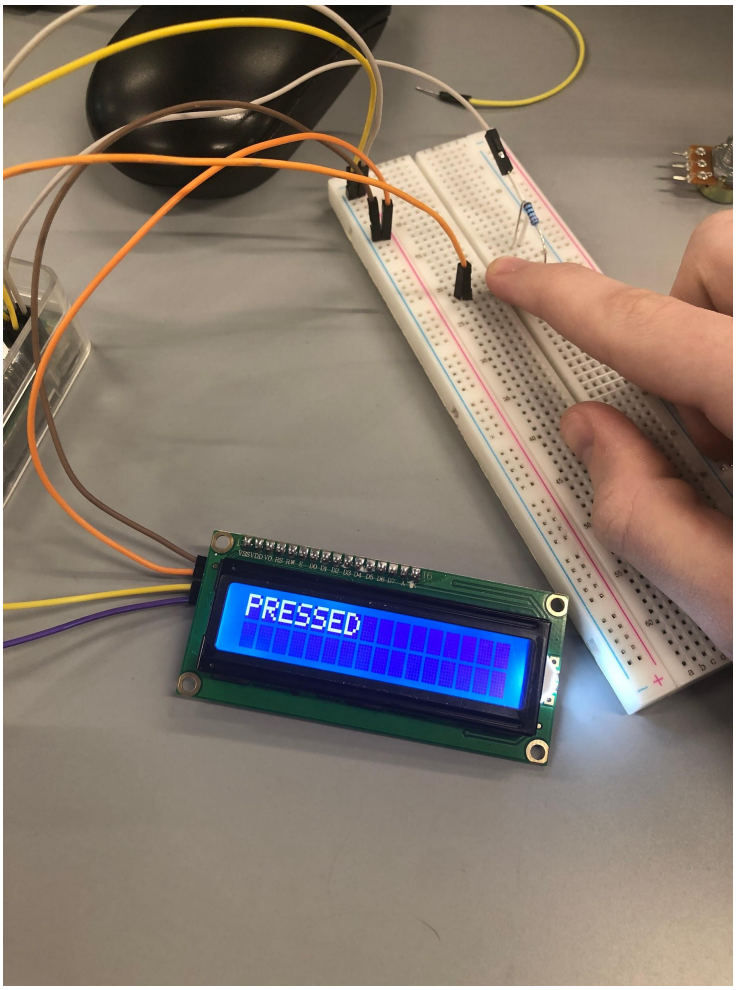
Polling involves detecting an input or event every clock cycle or as soon as the CPU allows it. This is inefficient as it can clutter programming when there are many inputs as well as slow down CPU processing in the priority of polling. In this example, we test for our button input using a while(true) loop in our main function. This also prints the message every run of the loop for both states which adds extra processing for the Pi's CPU.

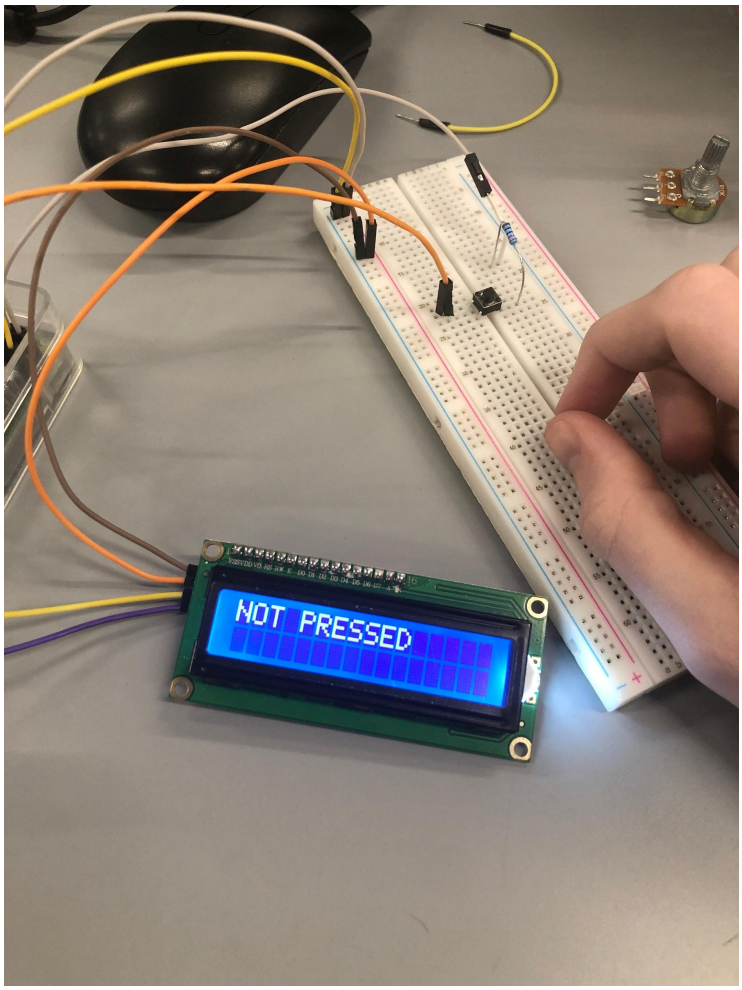




3 Exercise 3

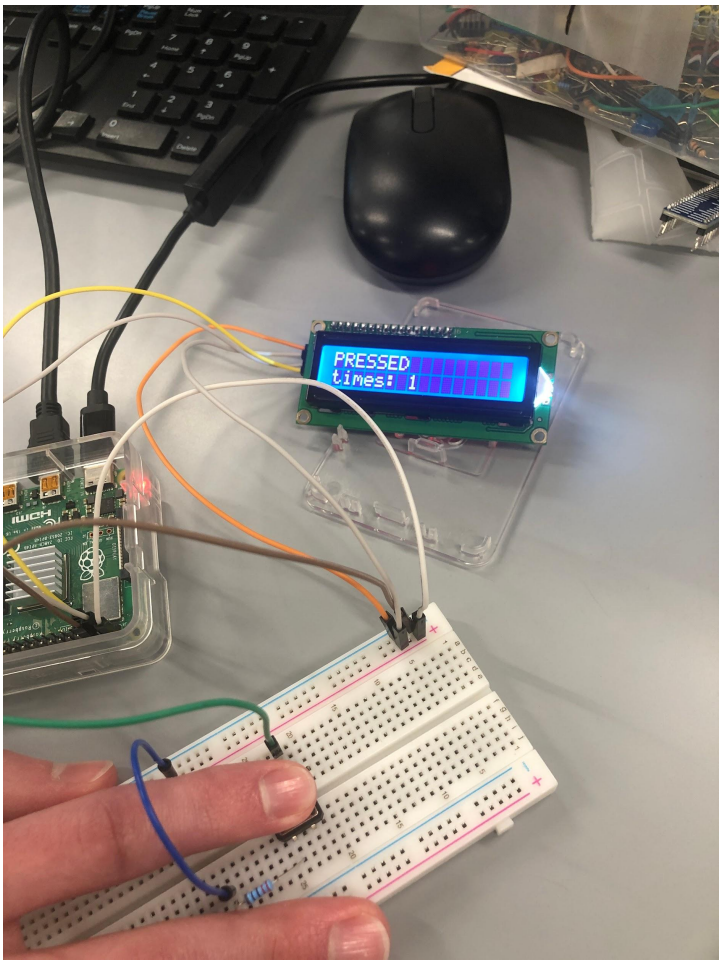
Interrupts are much more cost-effective in systems/microcontrollers which allow the processor to break apart from the current function only when the interrupt is activated. In our Raspberry Pi, this is accomplished by setting a function to be called when an input pin activates the interrupt. In our case, this interrupt was set to be called when the button changed from unpressed to pressed and vis versa. This called the function on both behaviors and would write to the LCD based on whether it was the rising or falling edge of the input.

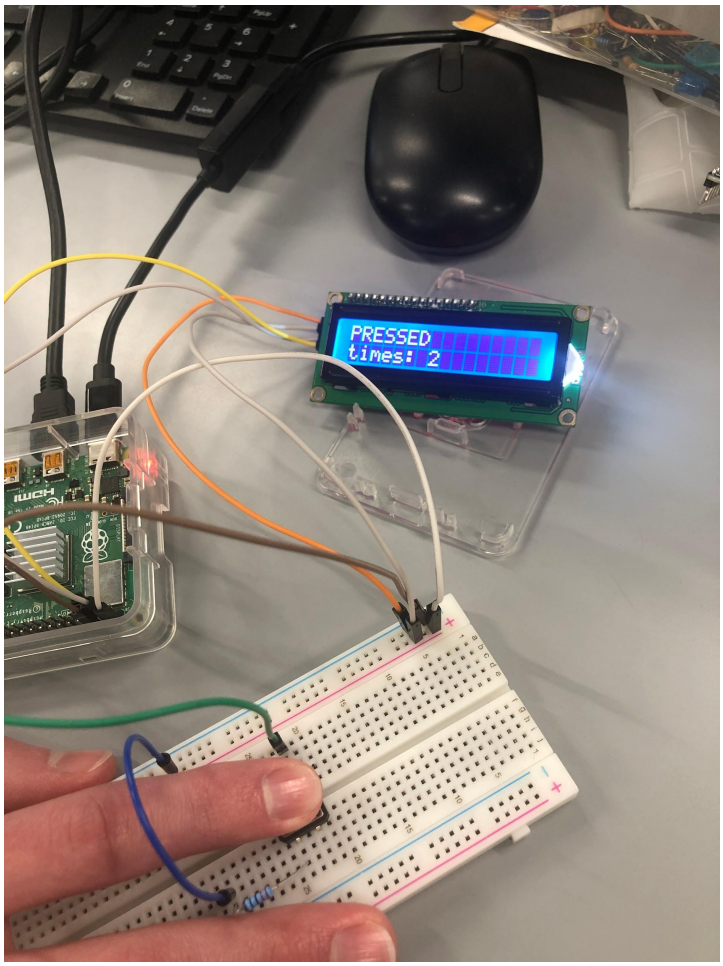




4 Exercise 4

A useful function for an interrupt system is a counter, which can be implemented to iterate when the interrupt is called. A global “count” variable was instantiated and iterated by ‘1’ when the button was pressed. However, due to no debouncing of the system, multiple presses could be measured from one button press. While the button is supposed to function as a digital component (with only inputs 0 and 1), the signal is analog and can produce many other values during the small period of time when the button is being pressed or released. The interrupt can be called multiple times during this state causing the exact number of presses to be incorrect.





5 Exercise 5

We can create a software debouncing system by simply delaying the function inside the interrupt which will stop the interrupt from being called again during this time. It cannot be called again during this time because the processes in our interrupt method cannot be run and cannot be accessed in parallel. Because of this, the interrupt cannot be called multiple times when the button is pressed down or released.

6 Supplemental Questions

1. Briefly summarize what you learned from this lab.

Throughout this lab, we were able to utilize many different techniques to display data on an external LCD screen connected to the Pi. In exercise 1, different commands found within the Pi's I2C library were used in order to determine correct cursor placement and display a character array as a string on the LCD. Within exercise 2, a button is implemented onto the breadboard, and the screen is designed to display whether or not the button is pressed. In order to achieve this, polling is used within a while loop, which is constantly checking whether or not the pin connected to the button is receiving an input. This information was then used to display the status of the button. Exercise 3 is almost exactly the same as exercise 2, but in this case an interrupt method is used instead of the previously used polling method. By pressing the button, the pin is able to detect when the button is pressed and then an interrupt is activated. This method of detection is much more efficient and allows for a while loop to not be constantly running, making the whole process less demanding. During exercise 4, interrupts are used again in order to create a counter that adds up all of the button presses and displays the total amount of button presses on the LCD along with the current status of the button press. Within this exercise it was very clear that debouncing was one of the main problems with the current design. When pressing the button many "false" button presses were detected causing the displayed count to be inaccurate most of the time. These extra button presses were the result of the interrupt function unintentionally getting called multiple times for just one button press. This problem was easily solved within exercise 5, where a simple delay is added so that it is not possible for the interrupt function to get called multiple times in very quick succession. After this implementation of a delay, the button press count on the LCD was then correct and only added 1 to the count every time the button was pressed. Through these exercises, multiple different techniques were learned such as how to use the LCD commands within the I2C library, how to implement interrupt techniques which are much more efficient that polling, and how to implement debouncing to always get the correct amount of input detection.

2. Explain the type of interfacing used to interface the LCD display.

In order to implement the LCD display, I2C first had to be enabled and then the I2C address of the newly connected LCD had to be retrieved through the use of the i2cdetect command. The display is controlled by sending different hexadecimal commands to the display

by using the send_command() command. The LCD also had to be initialized by setting it to be 4-bit with 2 lines and 5*8 dots. The cursor is used to determine the position of where the desired text is on the screen, and the target address of the cursor can be found and manipulated by using the line of code: address = 0x80 + 0x40 * y + x;

3. What is the advantage of using interrupts when compared with polling-based

Methods?

Polling requires more processing space from the CPU which can cause other processes in the program to slow down. This also requires more power from the system and is not economically efficient. While interrupts are only in systems that specifically have integrated interrupt functionality, they can be found in most microcontrollers and are much more efficient for executing a function when a desired pulse is measured.

4. In Exercise 4, did you get the exact number of pushes as you pushed? Please explain

the reason if you did not.

Since we hadn't implemented any debouncing system, the number of presses could become further incorrect with more presses. During each press, the analog signal of the button when rising or falling can fluctuate over and under the threshold for a digital input for the interrupt causing multiple presses during one physical button press. This also allows for the button to be half-pressed causing many more presses than intended. This was remedied by delaying the interrupt by half a second which could prevent the rapid-fire presses resulting from the analog signal.

7 Ideas & Suggestions (Optional)

Ideas: N/A

Suggestions: N/A

ACKNOWLEDGMENTS

I certify that this report is my/our own work, based on my/our personal study and/or research and that I/we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I/We also certify that this assignment/report has not previously been submitted for assessment anywhere, except where specific permission has been granted from the coordinators involved.

Author-1 Signature **Luke McIntyre**
Author-2 Signature **Brendan Bovenschen**

REFERENCES: N/A