**ITSC 305 – Reverse Engineering of IoT Systems**

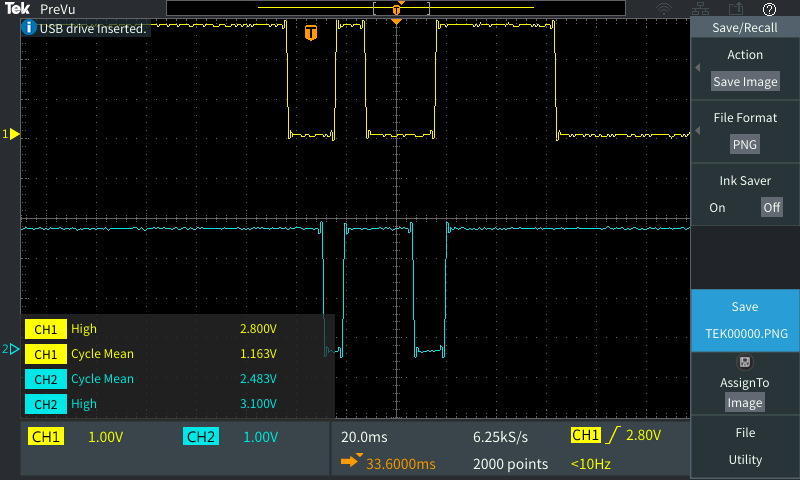
**Lab 4**

Submitted by Coleton Sanheim

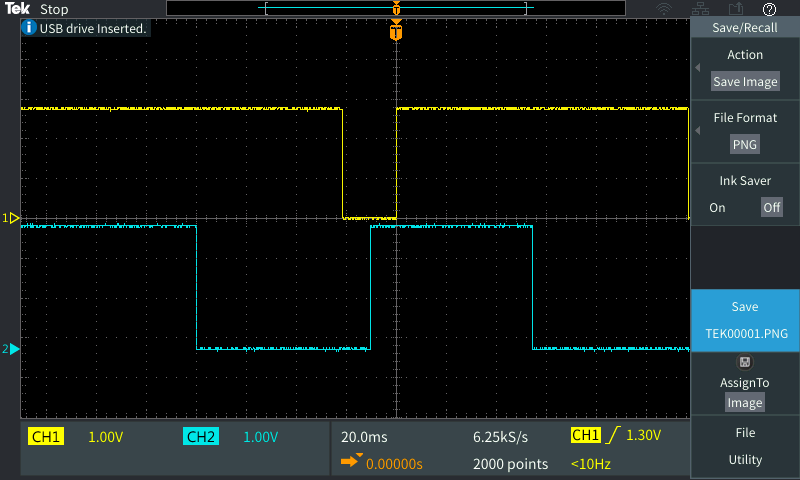
Part 1:

These are the oscilloscope captures of the rotary encoder. Yellow is A and Blue is B

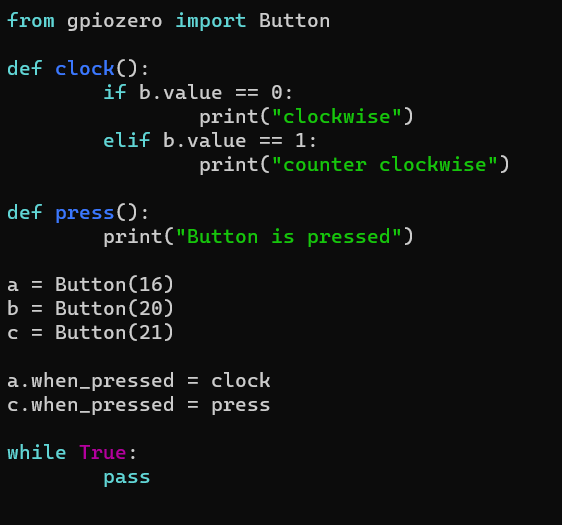
Clockwise rotation:



Counter-clockwise rotation:



Part 2:

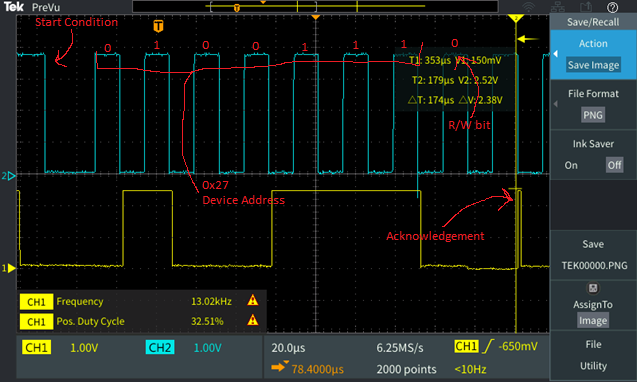


The code here is simple, by assigning a b and c to a button class then you can make a function that detects clockwise movement by checking when a is press if b is also pressed, and vice versa.

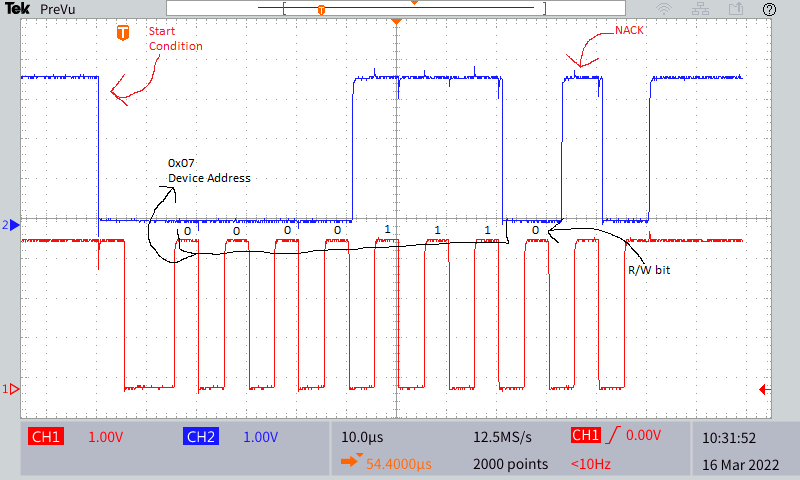
YouTube link: <https://www.youtube.com/watch?v=pzNUXW871gA&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=8>

Part 3:

ACK:

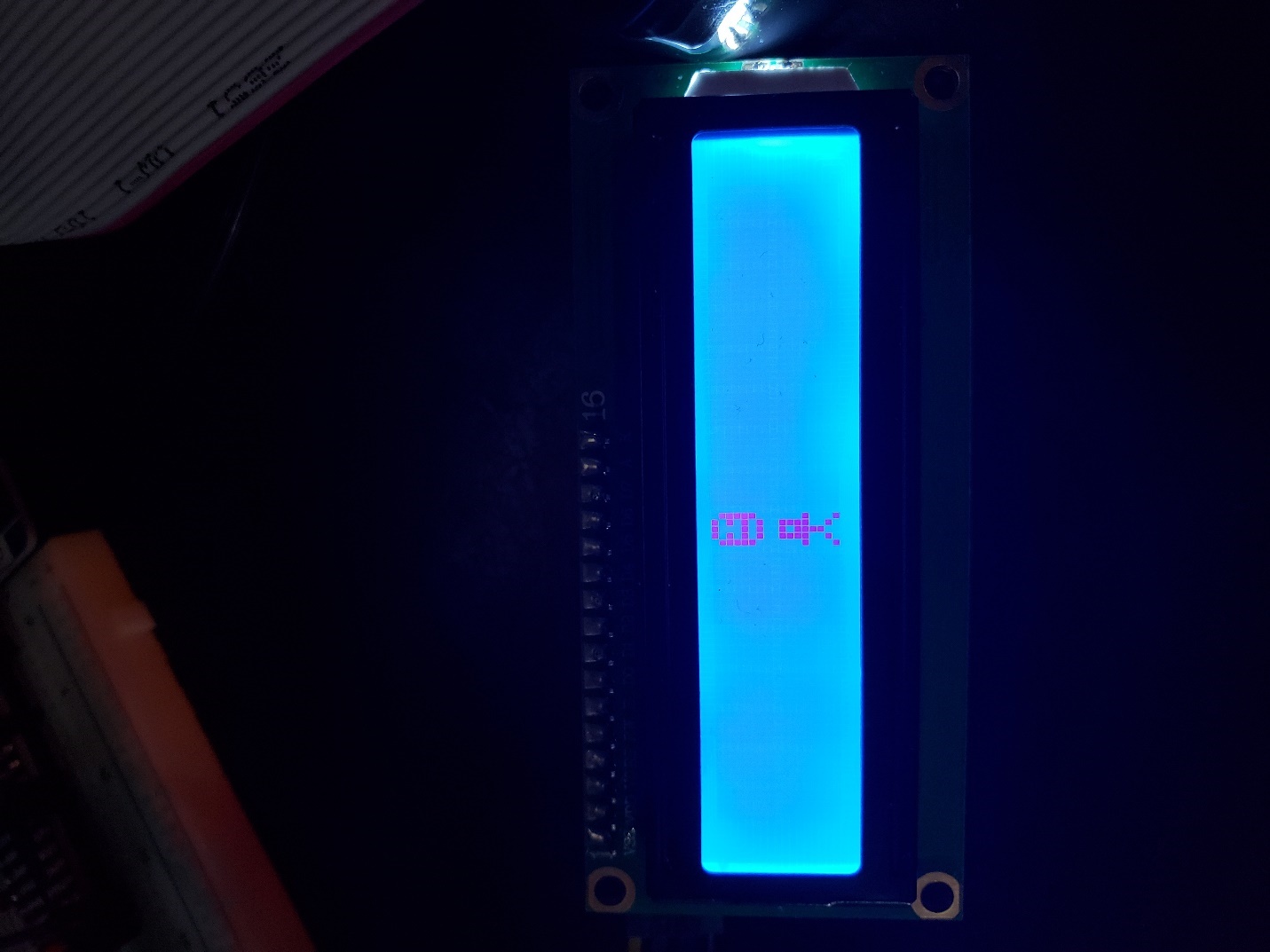


NACK: (NOTE, me and Rhean swapped photos, he got my ACK and I got his NACK as those respective photos were better than the ones we captured)



Part 4:



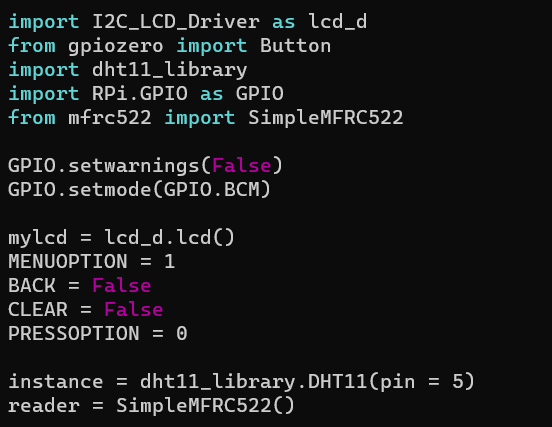


The custom character creation process was quite simple, first you have to load the custom characters into memory using the lcd\_load\_custom\_chars function, by passing a list of lists, where each list in the list is a character and each digit in the character corresponds to a row of pixels, by filling it with binary numbers where 1 is on and 0 is off I created a bitmap (like with the seven segment display) of the characters to print to the screen using the lcd\_write\_char function.

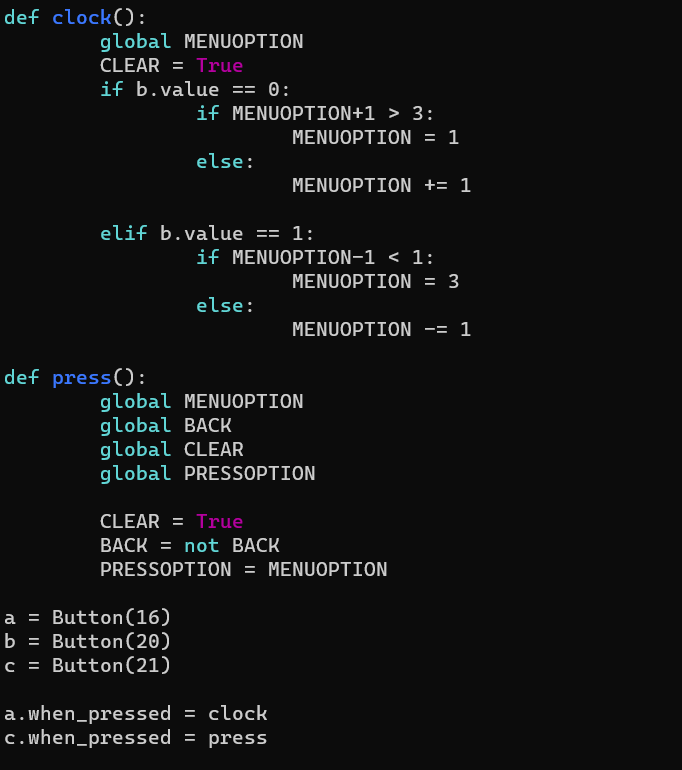
Part 5:

NOTE I split up the code into multiple sections for easier reading and description.

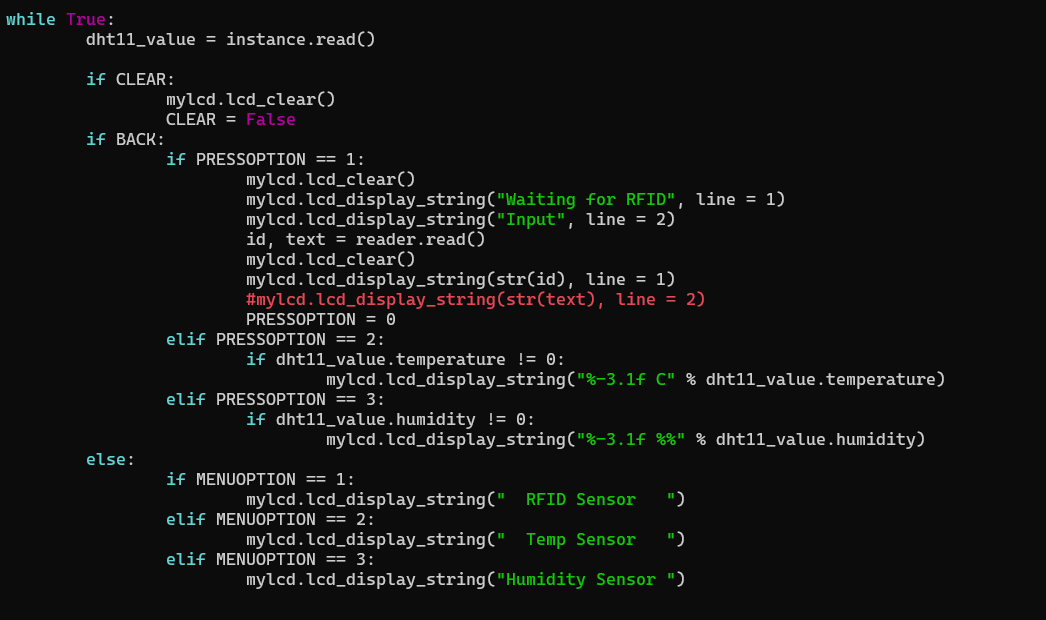
This is the header section, first is the libraries that are imported, I2C\_LCD\_Driver is used to control the LCD screen, gpiozero is used to define and recognise the rotary encoder. Dht11\_library is used for the dht11 sensor to detect temperature and humidity. RPi.GPIO is only required for mfrc522 to run properly and mfrc522 is for the RFID sensor. I also define some global variables here for use later in the program, MENUOPTIONS is used to keep track of what option on the menu the user is on, BACK is used as a flag to determine if a press of the button should bring you into the sensor or back out into the menu. CLEAR is used to keep track of any changes on the screen to know when to clear before new information is displayed. And finally PRESSOPTION is used to lock the user into the sensor display and not allow them to scroll through them until they go back to the menu.

’

This is the functions I defined for the rotary encoder. Clock is used to determine if clockwise or counter-clockwise and make a decision based on that. It will increment or decrement the menu option respectively and will wrap if it reaches the end values. Press is called when the rotary encoder is pressed in, it simply swaps the BACK variable and makes PRESSOPTION equal MENUOPTION.



This is essentially the “main” function of the program. First thing it does it is repeatedly updates the dht11 sensor whether or not you have that option selected. It will then clear the screen if any updates from the user has been detected. If BACK is true, then it will bring you into the respective sensor based on the value of PRESSOPTION, then if you press the button again and BACK is reset to false then it will show the menu option corresponding to the MENUOPTION variable.



YouTube link: <https://www.youtube.com/watch?v=WfcdswegT20&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=7>

Critical reflection:

The oscilloscope parts were very simple to perform once I figured out how to properly use the oscilloscope.

I had some difficulties in part 2 as I was trying to define two separate functions, one for clockwise movement and one for counter-clockwise movement. However, I discovered that it is a whole lot simpler to confine it to one function and it works better that way.

In part 4 I didn’t have any real issues, just did some research into the functions of the library and it was really straight forward.

Part 5 by far was the most complicated part of this lab, the first thing I did was use the code from part 2 and part 4 to be able to develop a simple menu system. The first major roadblock I faced was when I had the lcd screen functions placed inside the program functions, they didn’t work properly, sometimes it wouldn’t print, sometimes it would print gibberish and broken characters. I figured out this was because the lcd functions were place inside the program functions, and by taking them out they worked as intended, now why this is I have no idea but my best guess would be that because they are callback functions running in a separate thread they are not able to reference the screen properly. After that was solved I ran into an issue getting the dht11 sensor to work, the library I was trying to use (adafruit) wasn’t functioning properly so I ended up using the provided library instead. The Final problem I faced was getting the menu system working with the sensors, after finicking around with that for a while I decided to use the global variables I described and I got the functionally down.