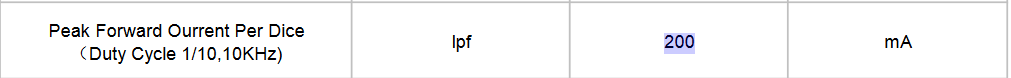
ITSC 305 – Reverse Engineering of IoT Devices

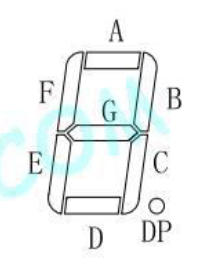
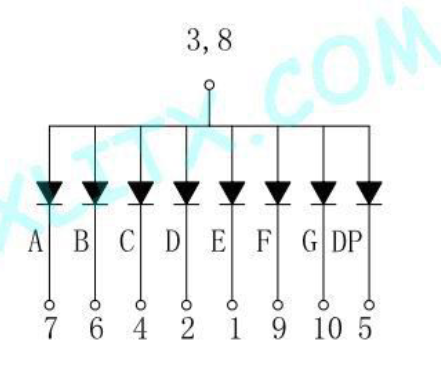
Lab 3

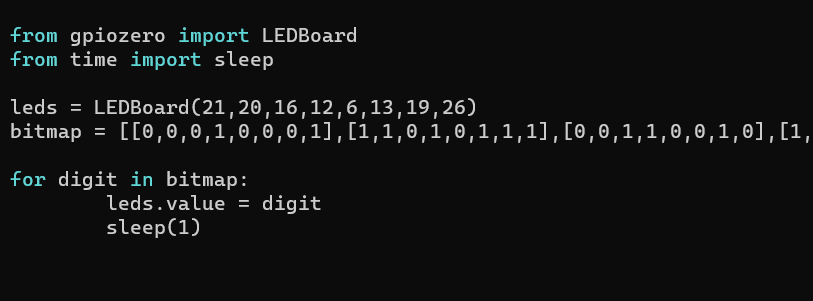
Submitted by Coleton Sanheim

Part 1:





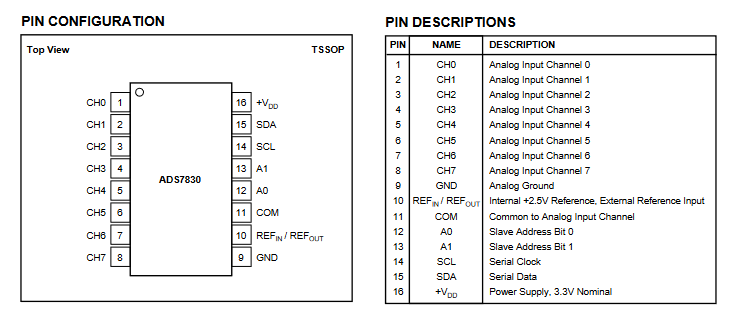


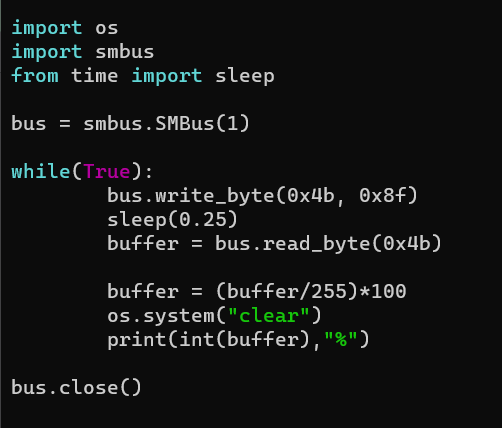
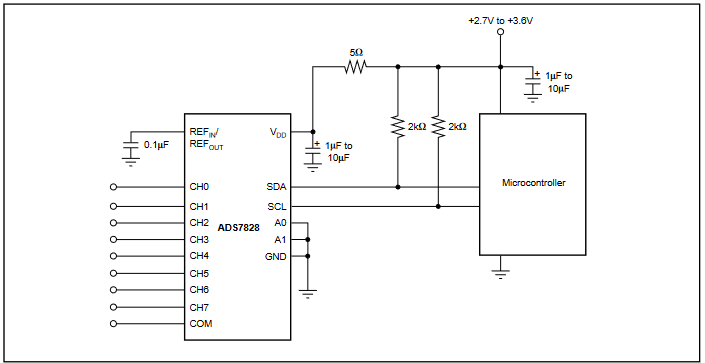


As you can see above, the seven segment display is common-anode, has a max forward current of 200 mA, and the lead assignment is shown. The code is simple, a bitmap was constructed (can’t fit the entire list in the screen) and a loop iterates through it setting the 7-segment display.

YouTube link: <https://www.youtube.com/watch?v=aADB8Awmwi0&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=3>

Part 2:

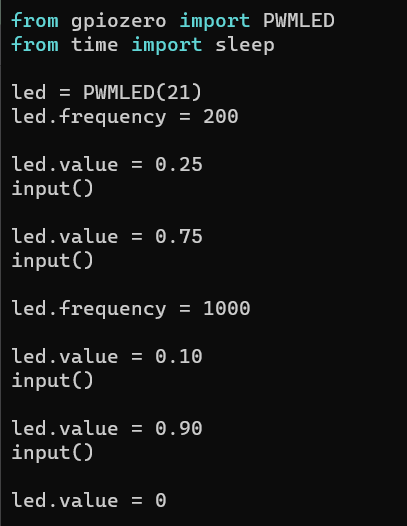




As shown above is the lead assignment and an example recommended connection diagram. The code shown writes a byte command 0x8f (10001111) which tells the A/D converter to grab the data from channel one which the potentiometer is connected to, then the program reads the data and displays it to the screen in the form of a percentage.

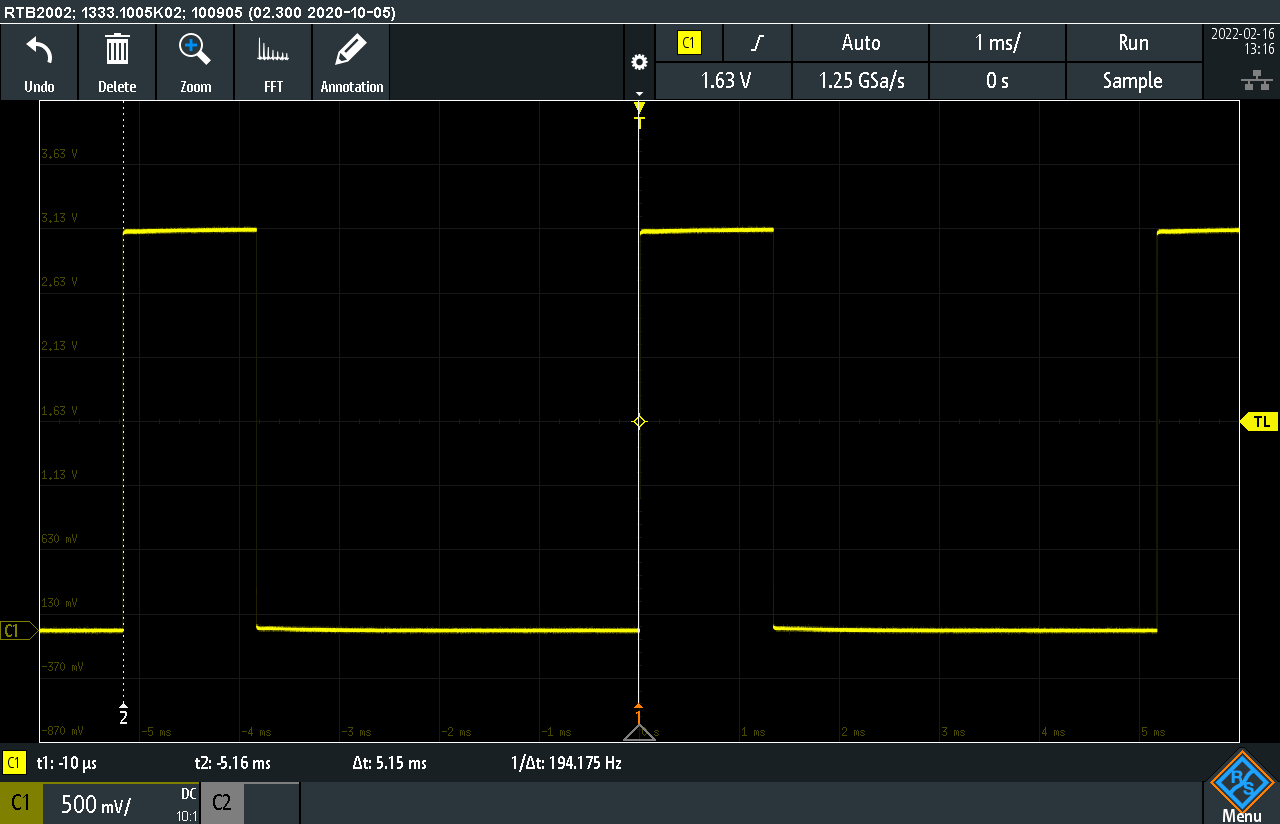
YouTube link: <https://www.youtube.com/watch?v=mR2D3msCZzw&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=4>

Part 3:



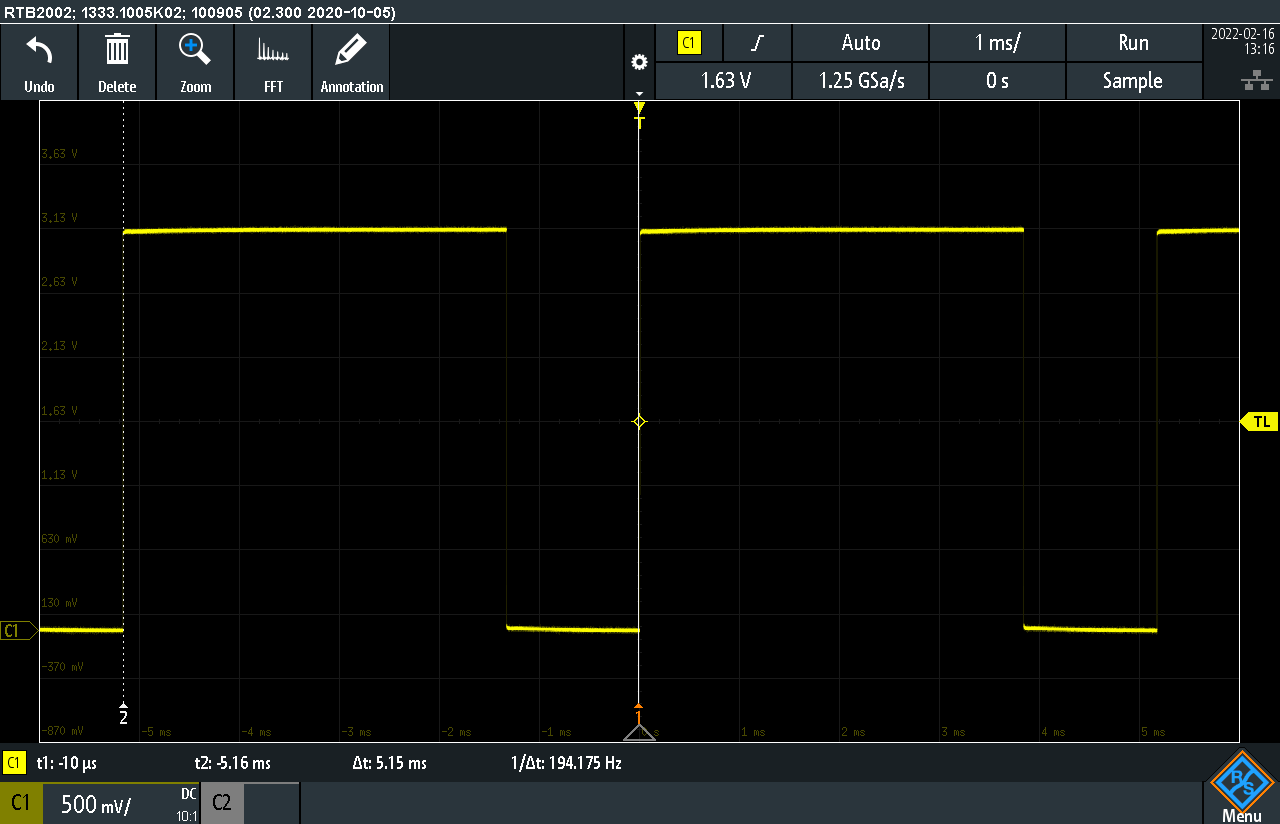
Here is the code for this part, it sets the frequency to 200, then cycles through the two requested duty cycles. Then it sets the frequency to 1000 and cycles through the duty cycles. Each step is separated by an input() to wait for an enter key press to continue.

Frequency 200, Duty Cycle 25%:

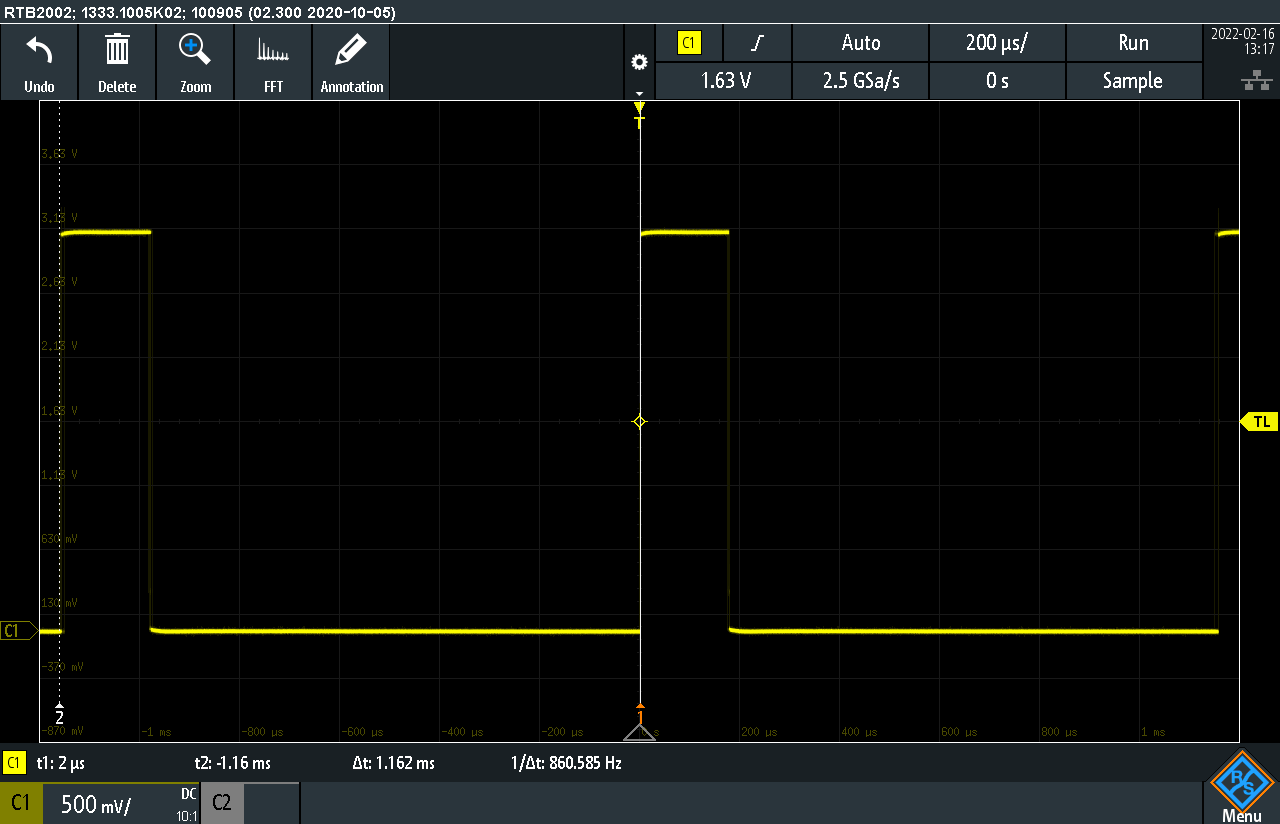


Part 3 (continued):

Frequency 200, Duty Cycle 75%:



Frequency 1000, Duty Cycle 10%:

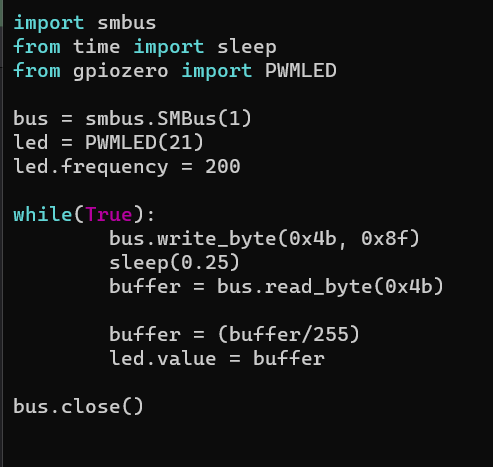


Part 3 (continued):

Frequency 1000, Duty Cycle 90%:



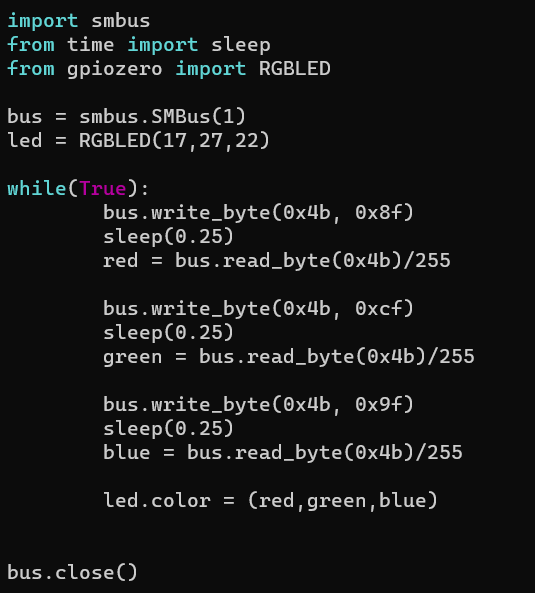
Part 4:



This code uses the same structure as part 2 but instead of printing the value to the screen it converts the value taken to be between 0 and 1 and sets the value of the led to be equal to the value grabbed from the potentiometer.

YouTube link: <https://www.youtube.com/watch?v=jmO-5KvNlIk&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=6>

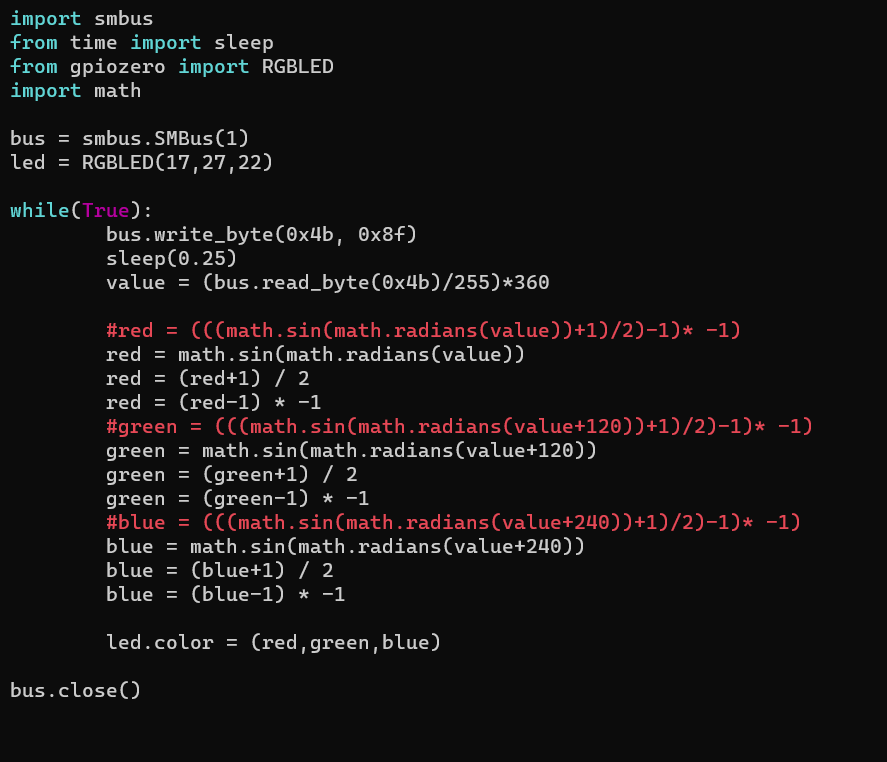
Part 4 (Alternate):



For this part the code is a little more complicated, it takes the same idea from part 4 and takes the value of each separate potentiometer and sets those values to each led of an RGBled.

YouTube link: <https://www.youtube.com/watch?v=2XZ7KiHbJuk&list=PLnRsOe1-tLI1NaS9RGdMRn1eGAGkkBFum&index=5>

Part 5:



For this part I grabbed the value from one potentiometer and converted it into a 360 degree value. I then split that value into three separate values, by taking the value and getting the sin values. The value is now between -1 and 1, so the lines afterwards are converting the value to be between 0 and 1.

YouTube link: <https://www.youtube.com/watch?v=koeZIn27U-k&ab_channel=Jack>

(unfortunately I forgot to record a video for this part so the provided video is from my groupmate Jack)

Critical Reflection:

In part 1 I didn’t run into any major problems. It was interesting trying to understand the information in the spec sheet and it was useful to learn how to interpret the information. It was tedious to create the bitmap but in the end, it worked a lot better than my first idea of setting each leds value manually.

The spec sheet for the A/D converter contained much more information than the previous one (as expected) and it took a while to fully understand how to create a command byte, but it was very useful information to learn as I understand that it is very common in many electronic components.

Part 3 was very simple, setting the frequency and the duty cycle of the led had no issues. The complicated part was reading these values with the oscilloscope and being honest I still have trouble setting it all up.

In part 4 I decided to do both versions of it for fun. The non-alternate version was easy as I basically copied the code from part 2 and adapted it to this part. The alternate version was more tricky trying to set it up with three potentiometers but I used the knowledge of creating command bytes from part 2 and with some research on the RGBled I got it working without any real problems.

Now part 5 was the trickiest one of all. Trying to do the same thing from part 4-alt but with only one potentiometer was very difficult to wrap my mind around. After the explanation from the instructor, I understood the basic concept of what we are accomplishing but I still don’t have a very good grasp of the theory behind it. I ran into a problem with the math.sin function that the values it was outputting was not what was expected, (we were expecting -1 to 1) and after some research I figured out that it was because we have to give the function the value in radians, thus the math.radians function. After that it was a simple matter of converting the values into 0 to 1 and it worked smoothly. This part required a lot of trial and error and I ended up with the two lines you see (value+1/2 and value-1\*-1) and it may seem strange but that is what worked after quite a bit of experimentation. In the end it works quite nicely.