This assignment proved to be a rather interesting challenge, using a piece of hardware completely new to me, a language I hadn’t touched, and an environment I was unfamiliar with.

During this assignment I feel that I have completed all of the assigned tasks:

* My code translates from ASCII input to MACE, and MACE input to ASCII. This is stored as a string to later be manipulated.
* My code flashes a blue LED for a MACE input, and a red LED for an ASCII input.
* My code responds to particular commands:
  + LP – followed by 12 digits – will set the light intensity for the LEDs
  + +-IR will turn on the IR Transmitter for 1 second and turn it off
  + RXIR will detect if there is an IR signal, and output HIGH or LOW
  + VRV will give a current potentiometer reading (0-1023)

One of the biggest challenges was responding to the “LP” command, taking the following 12 digits and transforming this into a light intensity level. The first issue was the fact that the commands were only looking at a MACE string. This worked fine for the other command (which had to only check for one specific string) but not for LP, which needed to check the following 12 characters (which could be represented with 1-4 characters each). As a result, I decided to make this command look at an ASCII string, and then use the inbuilt “isdigit()” function which checks if a character is an int digit. This issue was compounded when I tried to extract this character – it would extract the ASCII value of the character, as opposed to the literal value. My workaround was to take this value, and minus the ‘0’ ASCII value, which takes it to what the value is – exactly what I need.

Another issue I had was at the beginning, mostly due to human error. I was not aware of the need for char\* data types, and as a result my code would not translate from ASCII to a MACE string.

I felt that error detection was an important part of this assignment, and have included a number of detection functions. In the LP command mentioned above, there is a check to see that there are exactly 12 digits, and then checks that each of these characters are integer values. It then checks that no set of 3 values is greater than 255 (maximum PWM value) and if it is it alerts the user and then sets the value to 255. There is also a “default” case on the switch statement, and an else on most if statements so that if conditions are not met, it will still print something to alert the user there has been an issue.

In terms of marks for this assignment, I would put myself in the upper 25% of the group. I would say I have completed all of the tasks to a reasonable level, and have got all of the commands in part 3 working.