Concluding Thoughts.

In this conclusion, I will be discussing about my thoughts and about how well this assignment has went overall in detail. The conclusion will begin by explaining the processes and steps that I have taken in terms of designing the program as well as explaining the issues that I had faced along the way. I will also be discussing the solutions that I had used in order to fix them as well as if there were any solutions that could have been more streamlined and elegant in comparison to the method that I chose.

Part 1

To begin with, I felt that the first task was fairly straightforward, as it consisted of encoding and decoding MACE code. It had a lot of comparisons to the CS12020 practical sessions that I have been attending to, as the only main differences were that instead of translating a string of ASCII characters into Morse code, I had to simply translate ASCII into MACE code instead. I personally had no issues during this part of the assignment, thus there weren't any issues that needed to be fixed. This part of the assignment required me to write the most code for however, as I needed several different functions in order for the program to do a few key elements. This included ensuring that the program would indefinitely ask the user to keep inputting in a string of either ASCII/MACE as well as ensuring that the program would be able to effectively take the string of ASCII and encode it into MACE and vice versa.

Part 2

Part 2 of the assignment was also similar to the practical sessions, and thus I also found the tasks involved to be simple. Making the red LED transmit the MACE code when the first part of my solution received ASCII wasn't too complicated, and like, the same could be said when I had to make the blue LED transmit the unmodified MACE code if the solution to the first task received MACE instead. However, ensuring that the MACE time unit was between 20ms and 500ms depending on the state of the potentiometer was complex. I personally tackled this problem by dividing the integer value of the potentiometer by a ratio (2.046) in order to ensure the potentiometer's MACE time unit increases linearly between 20ms and 500ms (This is because analogRead reads an integer value between 0 and 1023). I then added an if statement stating that if the time unit is less than 20, then make the time unit equal to 20. Although this solution works and is practical, I feel that there is a more elegant solution available that I am unaware of. This is because my method of completing this task results in the first 40 integers of the potentiometer being irrelevant. I think there is a solution out there which makes sure that this problems doesn't occur.

Part 3

Part 3 of the assignment was much different to the rest of the assignment as a whole. In this part, there were 4 separate tasks that had to be complete. These tasks involved:

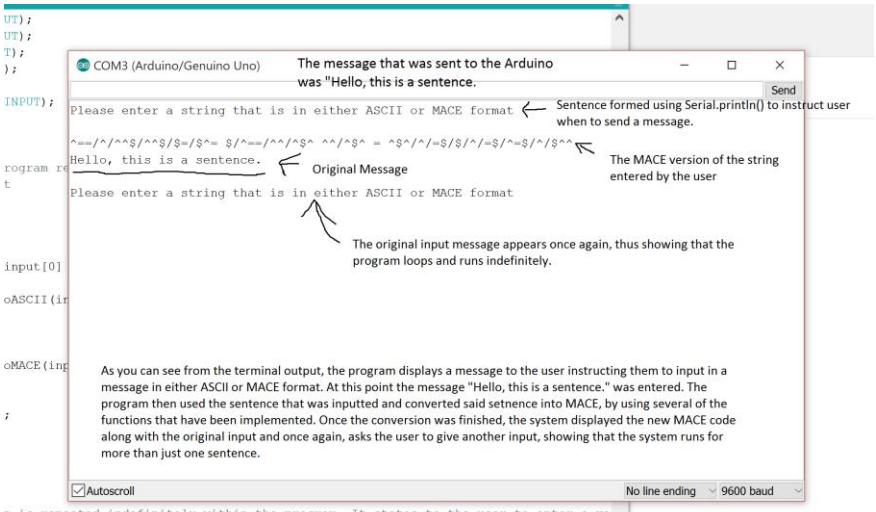
- Setting the brightness levels of each LED (With the exception of the blue LED).
- Turn the IR LED on for 1000ms and then turn it off again.
- Read the values of the IR sensor and transmit it on the serial port as "High" or "Low"
- Read the analog value from the potentiometer and encode it as a 4 digit MACE code.

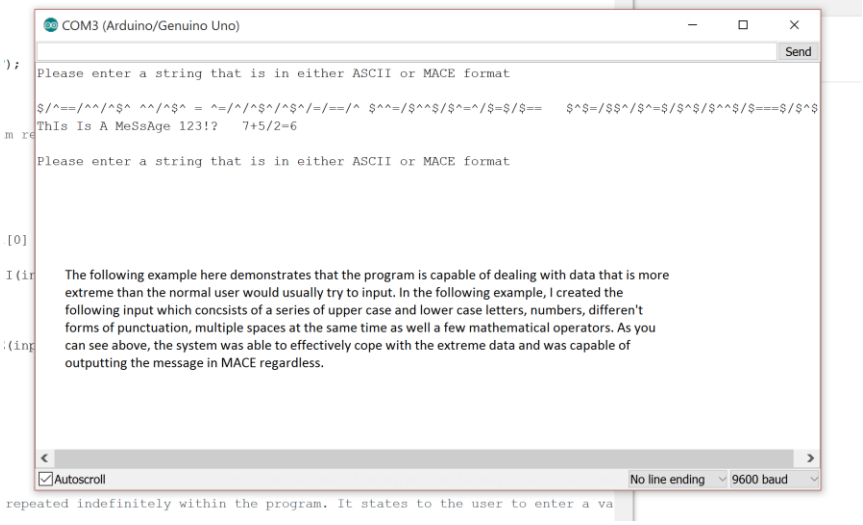
For setting the brightness levels of each individual LED, I personal tackled this problem by storing the 12 numbers into an array and then concatenating them into 4 block of 3 numbers each (each block for each individual LED). Once the concatenation has been completed, an analogWrite is used to write a PWM value on each LED. Finally, I made sure that if the PWM value of each LED is greater than 255 or less than 0, then they would be set to 255 or 0 correspondingly. The solution that I have put forth is practical, but I know that I the code used could have been more streamlined. Although it works really well, I feel that it could have been more efficient in certain places. One way I could have made it more efficient is if I were to use a second array to store all of the values of the concatenated number for each of the LEDs.

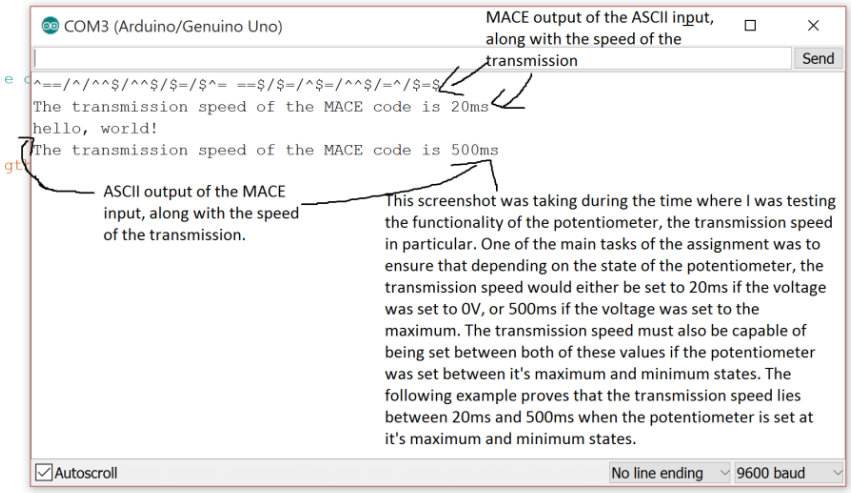
I personally didn't finish the IR LED turning on for 1000ms and turning it off again as well as the reading of the values of the IR sensors. This is due to a lack of time as well as a clear misunderstanding of how the infrared LEDs and sensors work. I, however, have left in the following code.

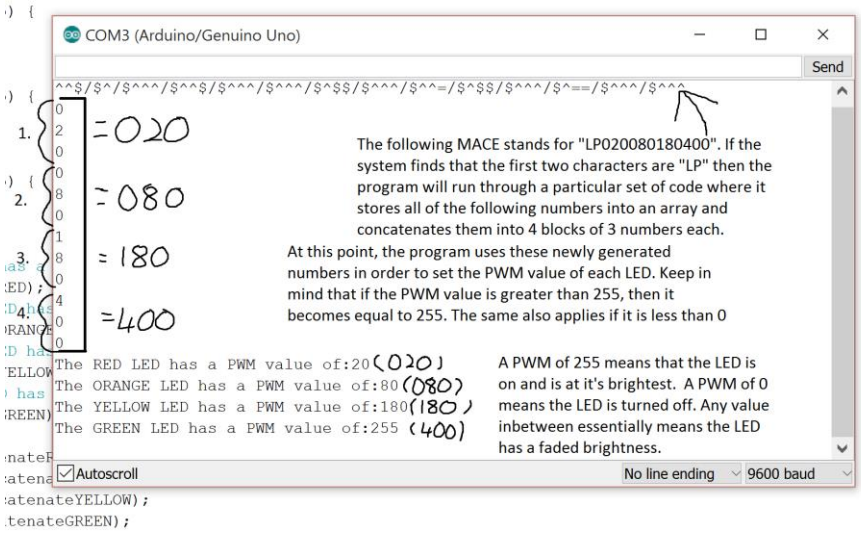
For the encoding of the analog value, I personally re-used the same code that I used in order to calculate for the MACE time unit (in order to give me the analog reading once again). Using this reading, I then performed a mathematical algorithm on said reading in order to give me a 4 digit code that is needed. I did this by multiplying the analog reading of the potentiometer by 18 until I had an integer value that was more than 1000. If for some reason the value was more than 10,000, then the answer would be divided by 2. This 4 digit code would then be encoded into MACE and be outputted to the user. I feel that this encoding is robust, and that the next step to take with this is to make the encoding algorithm more complex.

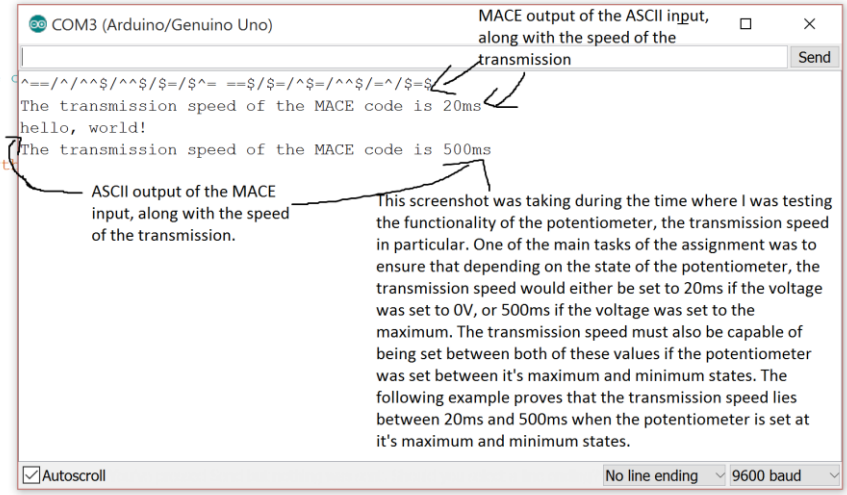
For the work that I have produced, I would award myself a mark of 65-70%. I feel that most of the work has been completed to a great standard, with only minor issues in terms of making the system more efficient. However, the lack of work done on the infrared portion of the assignment is going to cost me marks on this assignment.


COM3 (Arduino/Genuino Uno)
Send
Please enter a string that is in either ASCII or MACE format
^==/^/^^$/^^$/$=/$^= $/^==/^^/^$^ ^^/^$^ = ^$^/^/=$/$/^/=$/^=$/^/$^^
Hello, this is a sentence.
Please enter a string that is in either ASCII or MACE format
Autoscroll
No line ending
9600 baud
The message that was sent to the Arduino was "Hello, this is a sentence.
Sentence formed using Serial.println() to instruct user when to send a message.
The MACE version of the string entered by the user
Original Message
The original input message appears once again, thus showing that the program loops and runs indefinitely.
As you can see from the terminal output, the program displays a message to the user instructing them to input in a message in either ASCII or MACE format. At this point the message "Hello, this is a sentence." was entered. The program then used the sentence that was inputted and converted said setnence into MACE, by using several of the functions that have been implemented. Once the conversion was finished, the system displayed the new MACE code along with the original input and once again, asks the user to give another input, showing that the system runs for more than just one sentence.


COM3 (Arduino/Genuino Uno)
Send
Please enter a string that is in either ASCII or MACE format
$/^==/^^/^$^ ^^/^$^ = ^=/^/^$^/^$^/=/==/^ $^^=/$^^$/$^=^/$=$/$==
$^$=/$$^/$^=$/$^$/$^^$/$===$/$^$
ThIs Is A MeSsAge 123!? 7+5/2=6
Please enter a string that is in either ASCII or MACE format
Autoscroll
No line ending
9600 baud
The following example here demonstrates that the program is capable of dealing with data that is more extreme than the normal user would usually try to input. In the following example, I created the following input which concsists of a series of upper case and lower case letters, numbers, differen't forms of punctuation, multiple spaces at the same time as well a few mathematical operators. As you can see above, the system was able to effectively cope with the extreme data and was capable of outputting the message in MACE regardless.


COM3 (Arduino/Genuino Uno)
Send
^==/^/^^$/^^$/$=/$^= ==$/$=/^$=/^^$/=^/$=$/
The transmission speed of the MACE code is 20ms
hello, world!
The transmission speed of the MACE code is 500ms
Autoscroll
No line ending
9600 baud
MACE output of the ASCII input, along with the speed of the transmission
ASCII output of the MACE input, along with the speed of the transmission.
This screenshot was taking during the time where I was testing the functionality of the potentiometer, the transmission speed in particular. One of the main tasks of the assignment was to ensure that depending on the state of the potentiometer, the transmission speed would either be set to 20ms if the voltage was set to 0V, or 500ms if the voltage was set to the maximum. The transmission speed must also be capable of being set between both of these values if the potentiometer was set between it's maximum and minimum states. The following example proves that the transmission speed lies between 20ms and 500ms when the potentiometer is set at it's maximum and minimum states.


COM3 (Arduino/Genuino Uno)
Send
^^$/$^/$^^^/$^^$/$^^^/$^^^/$^$$/$^^^/$^^=/$^$$/$^^^/$^==/$^^^/$^^^
1. 020
2. 080
3. 180
4. 400
The following MACE stands for "LP020080180400". If the system finds that the first two characters are "LP" then the program will run through a particular set of code where it stores all of the following numbers into an array and concatenates them into 4 blocks of 3 numbers each.
At this point, the program uses these newly generated numbers in order to set the PWM value of each LED. Keep in mind that if the PWM value is greater than 255, then it becomes equal to 255. The same also applies if it is less than 0
The RED LED has a PWM value of:20 (020)
The ORANGE LED has a PWM value of:80 (080)
The YELLOW LED has a PWM value of:180 (180)
The GREEN LED has a PWM value of:255 (400)
A PWM of 255 means that the LED is on and is at it's brightest. A PWM of 0 means the LED is turned off. Any value inbetween essentially means the LED has a faded brightness.
Autoscroll
No line ending
9600 baud


COM3 (Arduino/Genuino Uno)
MACE output of the ASCII input, along with the speed of the transmission
Send
^==/^/^^$/^^$/$=/$^= ==$/$=/^$=/^^$/=^/$=$
The transmission speed of the MACE code is 20ms
hello, world!
The transmission speed of the MACE code is 500ms
ASCII output of the MACE input, along with the speed of the transmission.
This screenshot was taking during the time where I was testing the functionality of the potentiometer, the transmission speed in particular. One of the main tasks of the assignment was to ensure that depending on the state of the potentiometer, the transmission speed would either be set to 20ms if the voltage was set to 0V, or 500ms if the voltage was set to the maximum. The transmission speed must also be capable of being set between both of these values if the potentiometer was set between it's maximum and minimum states. The following example proves that the transmission speed lies between 20ms and 500ms when the potentiometer is set at it's maximum and minimum states.
Autoscroll
No line ending
9600 baud