Essay:

This essay outlines the programming task I have completed over the past 4 weeks with an Arduino and Arduino shield. The task at hand was essentially a translator between 2 languages – stak and Ascii. My program also has operations that allow it to perform tasks such as reading potentiometer values when certain inputs are entered. This essay also shows the complications that I had when writing the code and will end with a small conclusion.

After reading through the assignment, I realised the translation process was similar to Morse; looking at my char2morse code, I started the assignment, starting with the stak2ascci first mistakenly thinking the order I coded didn't matter however I found this to be a bad decision as it ultimately made the entire task longer to do. My problems at first began with the void loop and the stak2cahr function. This meant that my code was outputting the wrong letters and always outputting the error message. I realised I needed to change something and after looking at my code I saw that I was sticking too closely in line with my char2morse guidance code resulting in problems in the IF statements of my code e.g. inputting "@@" would get an output of "aa" instead of f. Seeing this I decided to sack the idea of using 2 functions and only use 1 for each translation way. After researching and discussing with friends, I discovered that using a constant pointer to create a lookup table was the solution to my issues.

After I began to use the constant pointer, I found I could use this as a lookup table for both ways stak to ascii and ascii to stak. This also meant that I could use 2 functions instead of my initial idea of using 4. I encountered an issue where my code displayed an error value after the correct answer, which I traced to an unrecognized null character in the characters array. This had an easy fix where in the void loop I just implemented the user input to be read to the next line and carriage input. This would eliminate any trailing characters and remove any error fault that was occurring. In fixing this problem I found a defensive programming method: a new function called Serial. readstringuntil which helped to make sure only the input was read. Another defensive programming method I used was Boolean. I implemented this in both of my translation functions which helped to give the error message when something unrecognized was inputted. After doing this I moved on to part 2; I recognized that my word separator was wrong, so I changed that to "=" as well.

I found part 2 to be easier than part 1, following the instructions from the first practical on Morse and I mirrored the layout used in the Morse assignment. I structured the sendDigital function in the same way as I did for Morse, using 100ms as the time unit instead of the potentiometer's time unit as I just wanted the logic down before doing any of the potentiometer code. After writing this function, I integrated it into the loop. Using the if statements from the ascii2stak and stak2ascii functions, I added a statement that enables the specific LED to be activated when the userInput requires that specific

function to be used. The implementation of the potentiometer also allowed me to set the correct time intervals between the LED flashes by mapping the potentiometer to values between 15ms and 300ms. This successfully completed part 2.

When I started with the PSV command, I misinterpreted the assignment and wrote the code assuming that when the ASCII PSV was input then the ASCII of what the potentiometer reading is should be outputted. This meant after rechecking the assignment I had to redo the code in a way that it would only run the PSV command when the stack code for PSV was inputted. Once I corrected this mistake, another issue arose: the lighting was malfunctioning, with the red and yellow LEDs failing to illuminate. I figured out it was because I didn't implement them within my loop and that fixed the problem.

For me, the TI function was the hardest part of the whole task. Looking at the task from face value after understanding how to use the tone () function I implemented the same format as I did for the PSV function. I realised that the TI function required a digit checker to see if the value was a digit and how many times, I had to iterate the tone () function. After researching I found a function called isdigit as my original plan of just seeing if the integer was to see if useInput was in between or equal to 0 to 9 was not working for some reason. Using this combined with a while loop I thought I had cracked it however even this wasn't working. Instead of using isdigit, I decided just to put the TI along with each decimal digit into the if statement and separate them using or which means that each value is checked for and if it matches the userInput then the rest of the code is run for TI.

After working on the project, I have attempted all parts and completed most. I believe that my work is sufficient and well done and I hope that my marker views it as the same. I believe with my quality of work I should be awarded well for it which is why I would consider this project to receive a 90% mark.