Analysis of the CAM translator/transmitter

My task was to create a series of functions to translate, store and print either CAM code or ASCII. This project allowed me to become far more comfortable with both C programming and the arduino.

I first had to decide on keeping my switch and if statements. The alternative would have been an array, which would be much more efficient. In the end, I decided on the array as it allows me to debug quicker, and would reduce the amount of clutter within the code, as well as the amount of memory used to store the data. The switch and if statements were convenient due to how easy they were to visualize, but were unnecessary. The array, while being simple, was prone to human error due to how many characters within it. I accidentally missed one, so output characters were always off by one in the array. For example, if asked to print “abc hello” the output would instead be “bcd ifmmp”.

I then had to create two functions that translated CAM code to ASCII, and vice versa. This was simple as I had already created something similar previously. I drew on the experience I had gained from the worksheets to create these functions. These then led onto independent functions, which I created in order to condense the characters entered in the serial port into strings.

These two string functions also performed additional tasks. These included adding pipes between characters as they were printed, and printing an error sign (#) to signify that an inputted character was not recognised. There were, however, issues with them. The “printCAM” function caused frustration due to how it did not seem to want to pass the translation to the print function. I had to use a debugger in order to discover which part of it wasn’t working.

After this I made a function that controlled the LED’s. Having one function would be much more efficient than having two for each LED (Blue and Red). I could simply use the function wherever I needed it. The duration of the timings here is determined by the potentiometer. The function takes a reading, then multiplies that by certain amounts in order to get different timings for different characters in the CAM code.

The penultimate function I created was the print statements. This first determined what language the input was, then printed the translation. These were simple to write too due to having already done the functions that controlled the LED’s and translation. The blue LED prints the unmodified code, while the red one prints the translated string version. Despite the ease, I found an error which let the program print the output onto the serial monitor, but wouldn’t flash the LED. This led me back to the “printCAM” function, where I was able to debug and find a solution.

Additionally, this is where I realised that my code wouldn’t work if a space was inputted first. I had to first remove most of my global variables as values weren’t clearing properly, storing values that would carry over into the next input. Luckily, all I had to do here was move all of these variables into the print functions. Then I had to modify my if statement to recognise a space, and to insert one into the string, then keep looking for the next input. A benefit of doing this is how much more security is given, due to the globals becoming local variables.

After this I created the infra red print function. This was just as easy as the previous function, as it just required me to write to the IR pin instead of the LED's. I just had to replicate the code I had already written for the LED function, for the IR light to work. A few modifications were required but these were also easy to perform. I couldn't go any further with this part of the assignment however, despite numerous attempts to do so. I couldn't get the receiver to work at all, and decided to focus more on fitting my program to the test harness.

I was able to finish all tasks up until partway through part 3. I could translate CAM or ASCII to the other, print them, flash their associated LED's and flash the IR light. If I were to change anything, it would be to implement a switch statement instead of an array. The array caused trouble due to my inexperience with them, leading to the error regarding the spaces before CAM input. This made the program unnecessarily complicated.

I would personally award myself 62% for the coding part of the assignment. I have completed part 2.1(encoding and decoding CAM code) and 2.2(transmitting CAM code by LED), and have been able to transmit using the IR light in part 2.3. Due to this, as part 2.1 equals 40% and part 2.2 equals 30%, I believe I should have at least 60%. The additional 2% comes from being able to transmit IR light. I have learned a large amount from this task, and used knowledge acquired over the term to implement my code.

For the assignment overall, I would give myself 58%, or a 2:2 due to not completing all of the program.