1. Analysis

1.1. Statement of Problem

A new transmission code (DisCode) needs be translated to and from ASCII. Once translated it should display the DisCode on the Arduino shield. It should also identify commands from the input and use them to interact with the Arduino shield.

1.2. Objectives

Part 1 – Encoding and decoding DisCode code

- Correctly translate ASCII characters to DisCode characters ✓
- Correctly format DisCode text ✓
- Correctly translate DisCode characters to ASCII characters ✓
- Only return the translated message ✓
- Receive more inputs after returning the translated message ✓

Part 2 – Transmitting DisCode by LED

- Transmission speed is controlled by potentiometer ✓
- If the translated message is DisCode, it is flashed on the blue LED ✓
- If the translated message is ASCII, the unmodified DisCode is flashed on the green LED ✓
- Transmission occurs immediately after serial transmission ✓

Part 3 – Respond to command messages

- Correctly receives the command message ✓

Command T

- The red, orange, yellow and blue LEDs are turned on in the correct order ✓
- Each LED is turned on every 250ms ✓
- All LEDs are turned off simultaneously ✓

Command P

- Correctly reads the 3 decimal digits ✓
- The blue LED is set to the brightness indicated by the number ✓
- The brightness is set to 0 after 500ms ✓
- If the number is higher than 255 the brightness does not change ✓

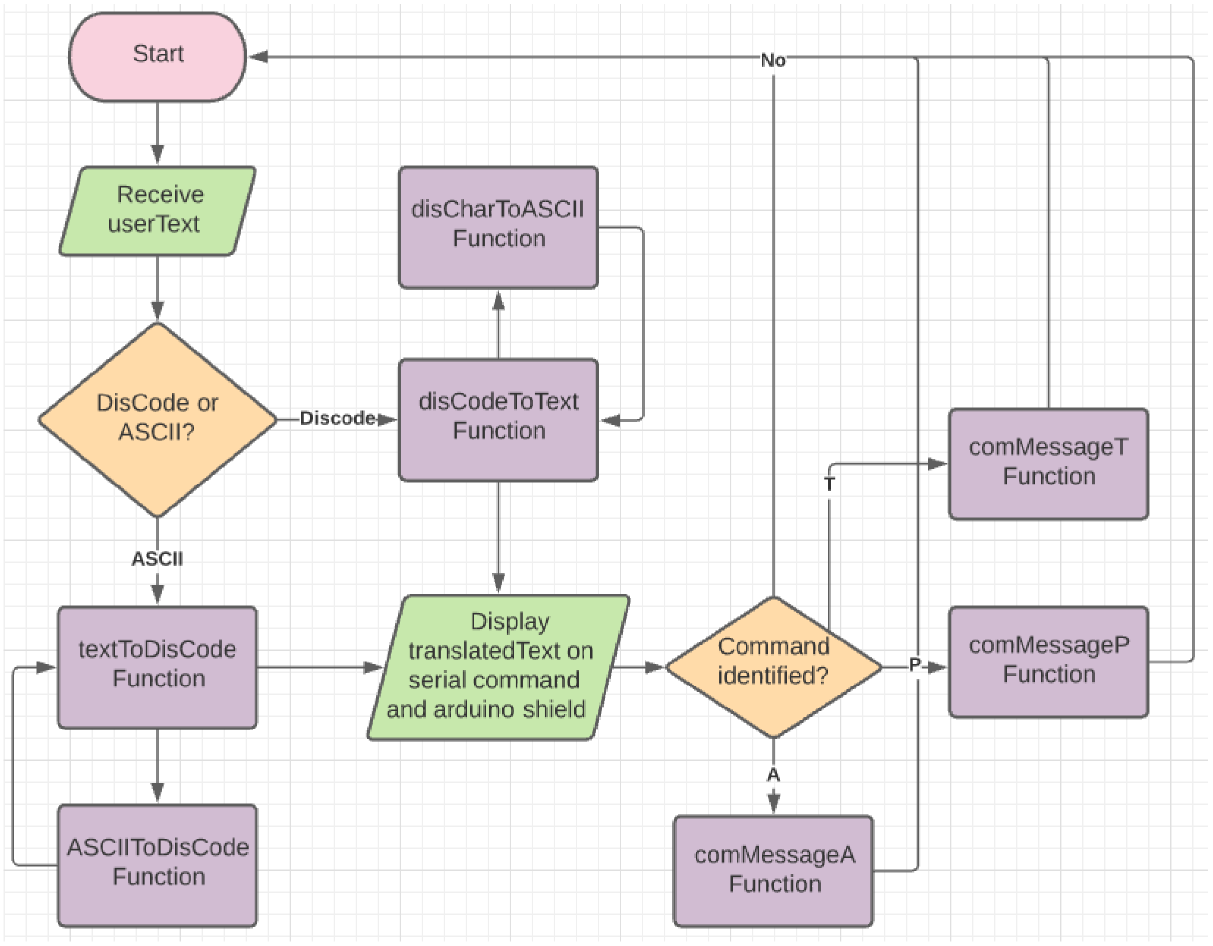
Command A

- Correctly reads the 6 decimal digits and splits them into two ✓
- The two numbers are added ✓
- Red displays the Hundreds, Orange displays the Tens, Yellow displays the Units ✓
- Red, Orange and Yellow LEDs synchronously flash the same number of times as the value of each digit (1 to 9) ✓
- Each flash lasts 125ms followed by an off period of 125ms ✓

- If the sum is higher than 999, the blue LED will be set on for the whole period of the display ✓

2. Design

2.1. System Flowchart



3. Testing

3.1. Test Plan

Test Number	Test Objective	Initial Test Outcome	Post Debug Test Outcome
1	Correctly translates ASCII characters into DisCode characters.	Correctly translated but put '|' at the end of the translated input.	Successful
2	Correctly translates DisCode into ASCII characters.	Successful	N/A
3	The program will receive an input after the program has already run once.	Successful	N/A

4	The potentiometer controls the transmission speed.	Nothing was displayed on the shield.	Successful
5	The DisCode is correctly displayed on the blue LED while it is transmitted to the serial port.	It couldn't work without 4.	Successful
6	The unmodified DisCode is flashed on the green LED, immediately after the ASCII is displayed on the serial port.	Fail ^	Successful
7	The transmission time for '$' is 1 time unit.	Fail ^	Successful
8	The transmission time for '+' is 2-time unit.	Fail ^	Successful
9	The transmission time for '^' is 4-time unit.	Fail ^	Successful
10	The delay between signals forming the same letter is 1 time unit.	Fail ^	Successful
11	The delay between characters forming the same word is 3-time units.	Fail ^	Successful
12	The delay between two words is 5-time units.	Fail ^	Successful
13	The command message 'T' is correctly retrieved.	Successful	N/A
14	The LEDs are set on in order if the command message is T and then set off.	Successful	N/A
15	The LED display lasts for 1 second in total each LED lit for 250ms.	Successful	N/A
16	The command message 'P' and the 3 decimal digits are correctly retrieved.	Successful	N/A
17	The blue LED is set to the brightness indicated by the 3-digit number.	Successful	N/A
18	The display lasts for 500ms.	Successful	N/A
19	If the 3-digit number is greater than 255 the brightness of the LED is not changed.	Successful	N/A

20	The command message 'A' and the 6 decimal digits are correctly retrieved.	Successful	N/A
21	The digits are interpreted as two, 3 digits numbers and are added together correctly.	Successful	N/A
22	The red, orange, and yellow LEDs display the units, tens and hundreds respectively.	Successful	N/A
23	The flashes of the LEDs are synchronous.	Successful	N/A
24	Each flash and off period lasts for 125ms.	Successful	N/A
25	Each LED flashes once for the value of 1, through to 9 times for 9.	Successful	N/A
26	If the sum of the numbers exceeds 999 the blue LED is set on for the whole period of the display.	Successful	N/A

3.2. Test Evaluation

Debugging my code was quite challenging because I didn't have many displayed syntax errors. This meant that I had to use pre-processing to display different stages of my code to check if I entered functions or correctly manipulated variables. The biggest problem with Part 1 of my code was my use of OR comparisons in my IF statements. Instead of writing a new statement (userText[i] == '^'), I would just try to compare different values (userText[i] == '^' || '$' || '+'. In the 3rd part of the program, I initially struggled to identify the command message because sometimes the command messages were identified as ASCII characters. However, I quickly solved the problem by changing the index of the for-loop till after the command message would have ended and I use if and else if instead of separate if statements.

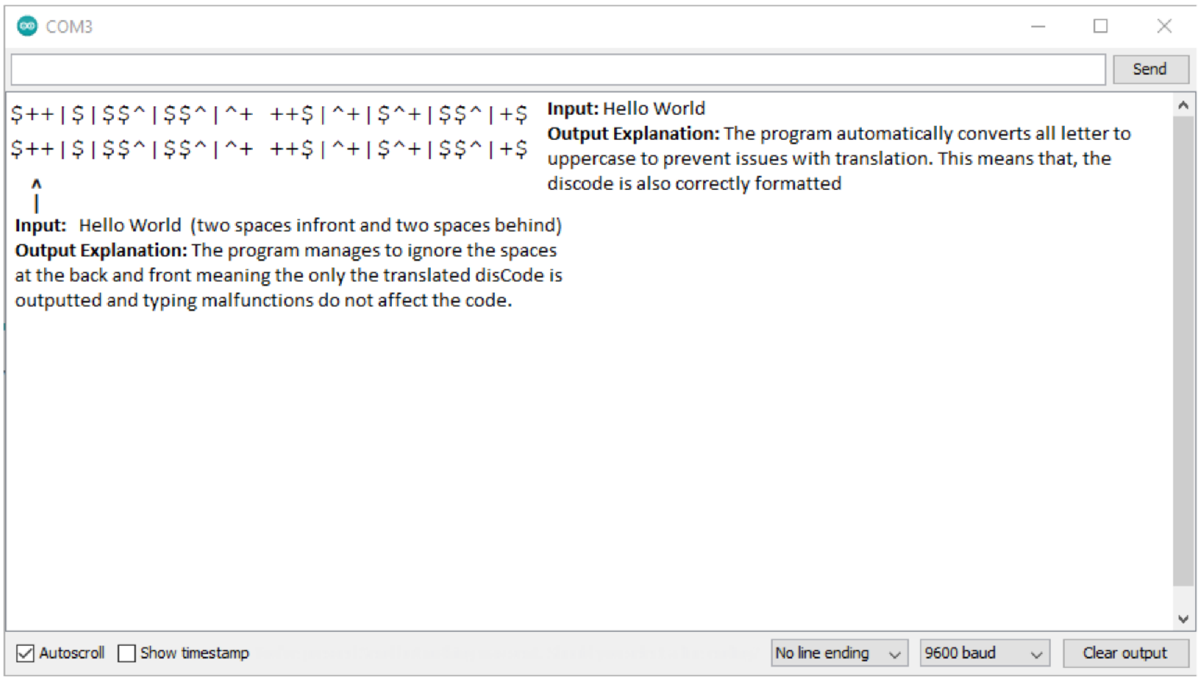
4. Evaluation

4.1. Final Evaluation

I believe that I deserve a 1st assessment mark because my programming meets most, if not all, of the assessment criteria. My program is well structured due to my deliberate naming of variables and functions, perfect indentation, and useful commenting of my code. I used many different functions to avoid repeatability of my code and to make my debugging process easier. I also used pre-processing to help with my debugging process since most of my errors were logical. The code for Part 1 and Part 2 works perfectly, accounting for multiple typing errors, and my code for Part 3 is well structured despite some aspects not being functional. This in my opinion demonstrates that the issues I experienced weren't my lack of programming skill but my limited knowledge and experience with C.

5. Screenshots

5.1. ASCII to DisCode translations

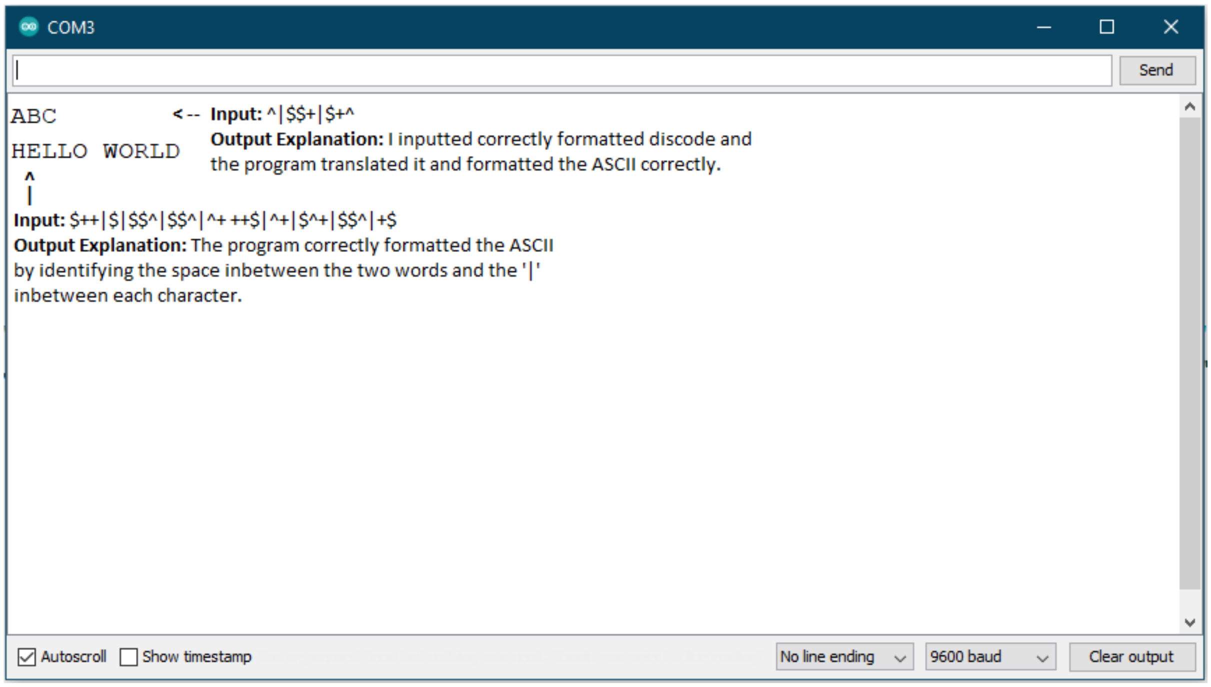


```
$++|$|$$^|$$^|^+  ++$|^+|$^+|$$^|+$
$++|$|$$^|$$^|^+  ++$|^+|$^+|$$^|+$
```

Input: Hello World
Output Explanation: The program automatically converts all letter to uppercase to prevent issues with translation. This means that, the discode is also correctly formatted

Input: Hello World (two spaces infront and two spaces behind)
Output Explanation: The program manages to ignore the spaces at the back and front meaning the only the translated disCode is outputted and typing malfunctions do not affect the code.

5.2. DisCode to ASCII translations



```
ABC
HELLO WORLD
```

Input: ^|$$+|$+^
Output Explanation: I inputted correctly formatted discode and the program translated it and formatted the ASCII correctly.

Input: $++|$|$$^|$$^|^+ ++$|^+|$^+|$$^|+$
Output Explanation: The program correctly formatted the ASCII by identifying the space inbetween the two words and the '|' inbetween each character.