# Nigam Puskar: 20318398

# Robot Project Planning Report

# Outline of the Problem to be Solved

The purpose of the project is to create a piece of code that can read a text document, use a font from the ‘SinlgeStrokeFont.txt’ file and transmit the command via a virtual RS232 serial port to a writing robot, which will then draw the text out as read from the user inputted text from the file.

When reading the file, the text from the file will be stored in an appropriate data format (struct\_TestData) before being outputted, with the program being coded such that it can process a file with any length of text.

The user will also need to input the desired height of the letters: between 4mm and 10mm excluding ascenders and descenders. The program will be required to scale the letters based of the user input such that the robot will be able to scale the x and y movements derived from the font data and the user input to achieve the desired height.

When outputting the writing, it should fit within a 100mm width writing area and be written with no breaks in any of the words. This means that the program should assess if it can fit the next word into the end of a line, and if it can’t, the program should output the word onto the next line. Each letter will also need to be offset it the x-direction so that its origin relates to the final point in definition of the previous letter, with each line of text being 5mm apart from one another. Each word should also be processed and outputted before the next word.

Finally, the program will output the instructions via G-Code, sending the commands to the Arduino to raise or lower the pen (corresponding to not drawing or drawing, respectively), and moving the arm to specified x and y locations, with the pen finishing in a ‘pen-up’ position at the origin (0,0).

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| SingleStrokeFont Characters | Structural array | Will hold the data from the ‘SingleStrokeFont.txt’ file, with each different structure representing a new line in the text file (totalling 1027 lines). This makes it easier to use ‘for’ loops to iterate through the structure to locate exact characters |
| SingleStrokeFont | Structure | Hold 3 int variables (‘int a0, a1, a2’) to correspond with each element in the three columns of the text file. Makes it easy to identify and copy over data from each line into the ‘struct\_TestData’ |
| UserHeight | Const float | Float used to store the inputted height from the user, which will be used to scale the X and Y values for each character in the ‘struct\_SingleStrokeFont’ |
| Scale | Const float | Works out the scale needed for each x and y coordinate to match the user inputted height |
| \*X\_Local | float pointer | This contains the new local X coordinates used in the ‘scale’ function after they’ve been scaled down |
| \*Y\_Local | float pointer | This contains the new local y coordinates used in the ‘scale’ function after they’ve been scaled down |
| WordCount | Int | Holds the total amount of words in the text file, used to work out the number of structural arrays needed for the ‘struct\_TestData’ |
| TestData words | Structural Array | Will hold the data (word by word) from the ‘TestData.txt’ file, with each different structure representing a new word in the text file (totalling ‘WordCount’ lines). |
| TestData | Structure | Contains an empty array to hold the value of each character in a word from the text file, as well as holding 3 int variables (‘int b0, b1, b2’) to correspond with each element in the three columns of the ‘SingleStrokeFont.txt’ file that corresponds to the character in the ‘TestData.txt’ file |
| Malloc | Void Function | Allocates extra uninitialised space for each word in the ‘struct\_TestData’ structure empty array. |
| \*CharacterCount | Char pointer | Keeps track of the number of characters per line to determine if a new line is needed |
| Line | int | Keeps track of which line the writing is on |
| \*X\_offset | Float pointer | Keeps track of the total displacement in the x direction to determine if it within the 100mm width |
|  |  |  |

Extend table as required

# Function Declarations

*Only include functions that you will develop.*

*Example (remove before submission)*

*int TemperatureConversion( int InputTemp, float\* OutputTemp )*

*Parameters:*

*InputTemp – input temperature in degrees C*

*OutputTemp – pointer to return output temperature in degrees F*

*Return value – returns 1 if successful, 0 if failed*

*Int ScaleText(struct\_SingleStrokeFont Characters, \*X\_Local, \*Y\_Local)*

*Parameters:*

*INPUT VALUE*

*Characters[i].a0 - Input X coordinate from struct\_SingleStrokeFont*

*Characters[i].a1 - Input Y coordinate from struct\_SingleStrokeFont*

*OUTPUT VALUE*

*\*X\_Local – pointer to return Scaled x value*

*\*Y\_Local – pointer to return scaled Y value*

*RETURN VALUE*

*Return value – return 1 if successful, 0 if failed*

*Int NewWord(struct\_TestData words, int wordcount, int \*CharacterCount)*

*Parameters*

*INPUT VALUE*

*Words[i] – allocate characters to ‘words’ array*

*Wordcount – Input to be used to create the number of structural arrays for struct\_TestData*

*OUTPUT VALUE*

*\*CharacterCount – Pointer to return the number of characters in the word*

*RETURN VALUE*

*Return value – return 1 if successful, 0 if failed*

*Int X&YGlobalCoordinates(struct\_TestData words, float Scale, float \*X\_Local, float \*Y\_Local, float \*CharacterCount, int Line, float UserHeight, , float \*total\_offset,)*

*Parameters*

*INPUT/OUTPUT VALUE*

*Words[i].b0 – input to store global X coordinate*

*Words[i].b1 – input to store global y coordinate*

*Scale – Input Used for working out the global X and Y coordinate*

*\*X\_Local – Input Used for working out the global X coordinate*

*\*Y\_Local - Input Used for working out the global Y coordinate*

*\*CharacterCount – Pointer used for working out the global X coordinate*

*Line – Input used to work out the global Y coordinate*

*UserHeight – Input used to work out the total offset*

*OUTPUT VALUE*

*\*total\_offset – pointer to return the total displacement in the x direction*

*RETURN VALUE*

*Return Value - return 1 if successful, 0 if failed*

*NewLine(float \*X\_offset, \*CharacterCount int Line,)*

*Parameters*

*INPUT/OUTPUT VALUE*

*\*X\_offset – input resets to 0 when new line made*

*\*CharacterCount – input resets to 0 when new line made*

*OUTPUT VALUE*

*Line – output increments line by 1*

*RETURN VALUE*

*Return Value – Line value*

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| Scale () and Main() | See if the program runs correctly when the inputted ‘UserHeight’ is a value in and outside of the range (4 -10) | 1: 5mm  2: 2mm | 1: Program continues to run  2: Program will print “Invalid input” and prompt the user to input a valid input |
| Main() | See if the program can read a valid / invalid file (e.g. for SingleStrokeFont.txt) | 1: “SinglesStrokeFile.txt”  2. “rubbish.txt” | 1. Program continues to run  2. Program will return ‘NULL’ |
| NewWord() | See if the function creates a new word when the user inputs a typo: “hello,my name” instead of “hello, my name” | 1. “Hello, my name”  2. “Hello,my name” | 1. Program will continue to run without errors  2. Program will exit |
| X&YGlobalCoordinates | Ensure that, whatever the value is, it corresponds to the correct line | 1. Initialise line as -1  2. Initialise line as 1 | 1. Will not print accurately to the correct line (first line will be above 0,0  2. Will print words accurately to the correct line (below 0,0) |
| Main() and NewLine() | See if \*X\_offset correctly assigns prints the word on the same line or a new line | 1. \*X\_offset = 99mm  2. \*X\_offset = 101mm | 1. Will skip the ‘NewLine()’ function and move onto the next word to process serial port  2. Will enter the ‘NewLine()’ function and process the word onto the next line |
| Main() |  |  |  |

*Extend table as required. Note that ‘Function’ includes main()*

# Flowchart(s)

Included seperately