# Hanif Adelekan 2031424

# **Robot Text Writer Software Planning**

# Outline of the Problem to be Solved

(Maximum 1 page)

The goal of the program is to control a robot to physically draw out some specific text. This will be achieved in 4 main steps outlined below.

The program will first read the entire text file named ‘SingleStrokeFont.txt’ and store it in memory in a sensible data format and structure.

The program will then accept a user input from the keyboard between a range of 4 and 10 (mm). This number will dictate the height of each letter so the program will need to scale the x and y movements of the robot accordingly. The font data consists of 18mm per unit hence the program will divide the user inputted height by 18 to effect the size change in the text being drawn.

After this, the program will accept a user input which will be the name of a text file. The program will read text from this file word by word. Each word will be processed and sent with the necessary instructions to the robot to draw out before the word is read from the file. This coupled with the fact that the file will need to function regardless of the length of text in text file suggests that memory may need to be dynamically allocated and freed for each word. The program will generate the necessary G-code commands to send to the Arduino which will raise and lower the pen in the correct x,y coordinate locations. The program will send the appropriate instructions to write out these letters from the point 0,0 meaning it will need to set this point accordingly to the end point of each written letter, e.g. one letter is written from 0,0 to 18,0 so the next letter will start from 18,0 instead of 0,0. The program will also calculate whether an entire word of letters can fit in the remaining space of a line, if not, it must offset the y coordinate appropriately 5mm apart from the previous line. Once the robot has finished writing the words, it will return to 0,0 with the pen up to finish.

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| height | float | To store input height |
| Current word | char |  |
| penlowered | bool |  |
| scaledheight | int | To store the scaled height |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Extend table as required

# Function Declarations

*Only include functions that you will develop.*

*Double GetHeight(double InputUserHeight, double\* HeightScaled)*

*Parameters:*

*InputUserHeight – input desired height*

*OutputHeightScaled – pointer to return the height after it has been scaled*

*float UsermmToUnitHeightScaler( float InputmmHeight, float\* OutputScaledHeight )*

*Parameters:*

*InputmmHeight – input height in millimeters*

*OutputScaledHeight – pointer to return height scaled by /18 for the robot units*

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

*float ProcessText( int CurrentWord[3], float\* OutputScaledHeight )*

*Parameters:*

*InputmmHeight – input height in millimeters*

*OutputScaledHeight – pointer to return height scaled by /18 for the robot units*

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

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
|  |  |  |  |
|  |  |  |  |

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

# Flowchart(s)

May be included as separate pdf