# Edward Moor 20415081

# Robot Writing Software Project System Manual

# Software Description

(Maximum 1 page)

The ultimate goal of the project is to develop a piece of software to enable a writing robot to “draw out” text from a file. The software will read in font data from a file named ‘SingleStrokeFont.txt’, read in the text to be drawn from another file named ‘TextData.txt, and define the hight of the output text via user input. The software will generate and send G-Code commands to the Arduino via a virtual RS232 Comms serial port. This provides instructions to the robotic arm how to raise, lower, and move the arm to specified X,Y locations in order to construct the text.

The software must be capable of processing text of any length, therefor utilisation of dynamic allocation of memory will be critical. It must write between a maximum writing width of a 100mm, without any breaks in the words “drawn”. The height of the text must be defined by user input but remain between 4-10mm. The software must scale the output text by a factor equal to the desired height/18 due to the 18 units in the font file.

The software should be developed with git for version control, there should be an initial commit consisting of the skeleton code. Further files generated throughout the development process, both code and documentation, must be committed to the repository.

The software will read in font data from a file named ‘SingleStrokeFront.txt’. Receive a user input for the text height and validate the input, ensuring it is within the specified range. The text to be drawn will be read from the text file. Each word in the text file must be read, processed and output to the writing robot prior to the reading of the next.

The program will run on a standard computer and will interface with an Arduino to communicate with an X,Y plotter drawing robot.

# Project Files

(Maximum 1 page)

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| textHeight | Double | Double can have fractional values e.g. 6.5mm text height |
| scaleFactor | Double | Double ensures accurate calculations of scale factor |
| numCharcaters | Int | The number of characters is a whole number, int is acceptable |
| fontFile | Const char\* | Allows for dynamic string handling and ensure the content cannot be modified |
| textfile | Const char\* | Allows for dynamic string handling and ensure the content cannot be modified |
| CharacterData | Struct | Groups the ASCII code, number of strokes, stroke data |
| strokeData | Int\*[3] | Pointer to X,Y,penU/D stored together for each ascii command |
| xOffeset | Double | Relies on precise positioning and may relate to the scale factor also a double |
| yOffset | Double | Relies on precise positioning and may relate to the scale factor also a double |
| Buffer | Char[100] | Stores G-code commands, fixed size to limit memory use |
| Line\_Spacing | Double | Involves adding potentially non integer values i.e. textHeight |

# Functions

***Prompt user for a text height between 4-10mm and validates input***

*Double getTextHeight()*

*Parameters:*

*None*

*Returns value:*

*Returns user inputted height as a double value if valid input. Requests user try again if invalid*

***Calculate the scale factor based on the given text height***

*Double calculateScaleFactor(double textHeight)*

*Parameters:*

*textHeight – valid text height between 4-10mm*

*Returns value:*

*Returns scale factor = textHeight/18 (e.g. SF=0.44 when textHeight =8mm)*

***Load font data into system***

*CharacterData\* loadFontData(const char filename, int numCharacters)*

*Parameters:*

*filename -name of font file*

*numCharacters – store number of characters loaded*

*Returns value:*

*Returns array of font data for each ASCII value, returns NULL if unsuccessful*

***Process the text file and generate G-code***

*void processText(const char \*filename, CharacterData \*fontArray, int numCharacters, double scaleFactor, double textHeight)*

*Parameters:*

*filename – name of text file*

*\*fontArray – pointer to array of loaded font data*

*numCharacters – number of characters in the array of font data*

*scaleFactor – scale factor of text defined as: desired text height / 18 units*

*textHeight – user input for desired text height*

*Returns Value:*

*None*

*NOTE:*

*Processes text file into G-code commands, output to the terminal window. When testing on the robot, some lines must be altered to allow in order to send correct G code commands to the robot.*

***Send commands to the robot***

*void OutputToTerminal(char \*buffer)*

*Parameters:*

*buffer – contains G code commands to be sent*

*Returns Value:*

*None*

*NOTE:*

*Left as a comment for emulator, must be uncommented during testing on robot*

***Free allocated memory***

*void freeMemory(CharacterData \*fontArray, int numCharacters)*

*Parameters:*

*fontArray – pointer to loaded font data*

*numCharacters – number of characters in the font array*

*Returns value:*

*None*

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| getTextHeight | Valid input within range | 8 | Returns 8 |
| -- | Invalid input outside of range | 12 | "Invalid text height. It must be between 4 and 10 mm.”  Requests input again |
| -- | Invalid input non numeric | “test” | "Invalid input. Please enter a numeric value.” |
| calculateScaleFactor | Calculate scale factor for valid text height | 8 | Returns 0.444 |
| -- | Lower edge case | 4 | Returns 0.222 |
| -- | Upper edge case | 10 | Returns 0.556 |
| loadFontData | Valid file | SingleStrokeFont.txt | Returns array of character data |
| -- | Invalid file | Incorrect or corrupted text file | “Error: Could not open font file”  Returns NULL |
| ProcessText | Single word, valid file | Test.txt – “The quick brown fox jumped over the lazy dog” | Outputs G code commands for text, with 5mm between each line and correct spacing between words |
| freeMemory | Valid array | fontArray | Frees all allocated memory |
| -- | NULL array | !fontArray | Array is empty so nothing is freed |

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

# Flowchart(s)

May be included as separate pdf