# Edward Moor 20415081

# Robot Writing Software Project System Manual

# Software Description

(Maximum 1 page)

This C script is designed to control a writing robot, allowing it to "draw out" text from a file. The program reads font data from a file SingleStrokeFont.txt, processes text input from another text file , and generates G-Code commands to control the robotic arm via an Arduino using a virtual RS232 serial port.

Upon initialisation, key functions are declared as well as the structure CharacterData. This structure holds the ASCII codes, the number of strokes required to draw each character and the corresponding stroke data. Within the main function the names of text and font file are defined, and can be altered if the need arises. Font data is loaded into the system via the loadFontData function. It reads the font file, parsing it into the CharacterData structure, and dynamically allocates memory to store the stroke data for each character.

The getTextHeight function which prompts the user to enter a value for the text height between 4mm and 10mm and then validate their input. A scale factor is then calculated by dividing this user input by 18, accounting for the maximum height of the character in the font file. The robot is woken up, and made ready to draw.

The function processText is called. It first checks the text file opened correctly, and returns a NULL value if not. It iterates through the words within the text file, processing the letters within each one before outputting the G-code commands to the terminal and robot then moving onto the next word. As it progresses, it ensures each word is spaced apart from the prior, validating each line to be within the 100mm line limit, and handles line feeds with 5mm spacing and carriage returns where necessary.

All the files are closed, memory is freed, and the pen is return to the initial 0,0 position in pen up state.

It should be noted that for the purposes of this submission the code is set up for the emulator, in order to test on the robot the SendCommands must be uncommented from within the processText function and G0/G1, stating the pen up/down commands, must be changed to S0/S1000 respectively. Comments are included within the main.c script stating at specific points where such changes are to be made.

# GitHub Link

https://github.com/egyem6/Software-Project

# Project Files

(Maximum 1 page)

main.c: Main script where files are read into, processed and G code commands are output to the robot from. Parameters to be defined are the name of the font file and text file.

test.txt – Text file containing the test script “The quick brown fox jumped over the lazy dog”

SingleStrokeFont.txt – File containing font data for every 128 ASCII value

# 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 |
| main() | Successful 8mm text height operation. | 8mm Text Height  Test.txt | Appendix, figure 1 |
| 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)

A diagram of a flowchart

Description automatically generatedA diagram of a machine

Description automatically generatedA diagram of a flowchart

Description automatically generatedMay be included as separate pdf

A diagram of a flowchart

Description automatically generated

A diagram of a flowchart

Description automatically generated A diagram of a flowchart

Description automatically generated

A diagram of a process

Description automatically generated

# Appendix

A white paper with blue writing

Description automatically generated

Figure 1 – Expected output on emulator