# Name & Student ID: Angela Rangel, 20361247

# Title: System Manual Document

# Software Description

The project overall involves creating a software program that enables the writing robot to "draw out" text, based on a given file and text input. This program will read and interpret font data from SingleStrokeFont.txt, scale the text defined by the user input, and generate movement commands in G-code format to then be sent to the Arduino that controls the robotic arm’s movements.

All code must be developed using git for version control. This should include up to date commits as the project progresses for both documentations and codes. Also, an initial commit for the skeleton of the skeleton code.

As specified, the software must read the font data from the ‘SingleStrokeFont.txt’ file. After it is read, each character within the font file should be stored in a dynamic memory allocation, to free up the memory space once the robot has finished writing. This also ensures that the software has sufficient memory space and for the program to run at a faster time.

The program should ask the user to input the height of the text ranging from 4mm to 10mm. Create a function that adjusts the X and Y coordinates from the font file by a scale factor of to properly draw out the specified height in real life.

The code should read another text file that is obtained from the user keyboard’s input to draw out the text written within the file. The program should be able to process a file containing text of any length. This can be done by also storing it in a dynamic memory allocation to ensure there’s sufficient memory to run the software and automatically provides the right size array.

For each character within the text file, create a function that translates the font data into G-code commands. This function should include that each new letter that is instructed to be drawn, it must be offset in the X direction so that the letters won’t overlap each other. This is done by the starting position of the next character corresponds to the last position of the previous letter written. Another section to include within this function is to ensure that each text line are drawn with the space of 5mm apart from each other. This is prompted by the LF (ASCII 10) and CR (ASCII 13) codes respectively in the text file.

Finally a function that sends the G-code commands to the Arduino. These commands are then sent to the Arduino to communicate whether to raise or lower the pen and to control the arm’s movements to specific X-Y coordinates. The relevant commands of G-code that is related to the project is shown in Table 1. Once the pen finishes drawing, it should return to the point at origin (0,0) and rest once the pen is in a raised position.

|  |  |
| --- | --- |
| **Command** | **Description** |
| F1000 | feed rate, 1000 mm min−1 |
| G0 X Y | Move to the position X,Y |
| G1 X Y | Draw a straight line from the last position to X,Y |
| M3 | Turn on Spindle |
| S0 | Pen up |
| S1000 | Pen down |

**Table 1**: The relevant G-codes needed for the project.

# Project Files

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File Name** | **File Type** | **Purpose** | **Key Elements** | **Key Notes and Parameters** |
| Main.c | Source code (.c) | Implements the main functionality of the program. Coordinates loading font data, processing input text, generating G-code, and sending it to the robot. | **Functions include:**  main(),  LoadFontData(), GenerateGCode(), SendCommands().  This controls the program flow and user interactions. | Requires SingleStrokeFont.txt to be present in the working directory for loading font data. |
| SingleStrokeFont.txt | Text file | Text file containing font data for ASCII characters. Defines how each character is drawn, including coordinates and pen states | The Format for each character: 999 <ASCII> <NumMovements> followed by movement data (x, y, pen). | Must be in the working directory, for the main software to work |
| rs232.c | Source code (.c) | Implements functions for RS232 serial communication with the robot. | **Functions**: CanRS232PortBeOpened(), PrintBuffer(), CloseRS232Port() | Parameters to set: Input the COM port number and baud rate in the communication setup function. |
| serial.c | Source code(.c) | Manages low-level serial communication and G-code transfer to the robot or emulator. | - Uses #define Serial\_Mode for conditional compilation. | Ensure #define Serial\_Mode is correctly toggled for emulator or for the writing robot. |
| RobotTesting.txt | Text file | Sample text file for robot testing sessions and used to validate program functionality. | Contains sample text used for generating G-code and testing the drawing accuracy of the robot. | Designed for on-site validation during testing phases. |

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| FontData | structure | ‘Struct’ as it relates the relevant data together into a single entity. Stores the font data file’s information for all supported ASCII characters, including their movements and pen states. |
| Movement | structure | Represents individual drawing movements for each font, forming the basic elements of how characters are rendered. Reason it is structure is it encapsulates the details (X, Y coordinates, pen state) for each drawing movement of a character. |
| x, y | int | ‘x’ stores the x-coordinate and ‘y’ stores the y-coordinate of a movement. Coordinates are integers because data font file contain whole numbers only |
| pen | int | Indicates the pen state (1 = down, 0 = up). Since it’s only either ‘1’ or ‘0’ using integer as the data type is appropriate |
| ascii | int | ASCII values are integers and serve as indices to map characters to font data. |
| num\_movements | int | Number of movements required to draw a character. The number of movements can only be whole numbers hence why it is an ‘int’ |
| movements | Movement[] | Array of movements defining the strokes for a character. An array of Movement structs is the most efficient way to represent sequential drawing steps. |
| buffer | char[] | Temporary storage for G-code commands to be sent to the robot. G-code is a string hence why char[] is chosen as the data type |
| x\_offset | float | Tracks the current x-position for drawing. Floating-point precision is necessary for scaling operations and sub-millimetre adjustments. |
| y\_offset | float | Tracks the current y-position for drawing. Floating-point precision is necessary for scaling operations and sub-millimetre adjustments. |
| scale | float | Factor to scale font movements to the desired text height. Scaling involves division and multiplication which means the value for ‘scale’ would be a fractional/decimal value hence why it is stored as a float |
| height | float | User-input font height in millimetres. The user may input decimal values for text height hence why it is stored as a float |
| text | char[] | Stores the text to be converted to G-code. ‘Char’ is chosen since it’s an array of characters |
| filename | char\* | File name for input files (e.g., font or text files). |
| previous\_pen\_state | int | Tracks the last state of the pen to optimize G-code generation. ‘Int’ for pen up/down as it can only take integer values. |

# Functions

|  |
| --- |
| **void LoadFontData(char FontFile)**  Parameters: FontFile – A string representing the name of the font file to be read.  Return Value: *return 1* if it is successful,  *exit(1)* if failed  Operation Details: To load font data from a file and populate a ‘FontData’ array that defines the movements for drawing each character. |
| **void GenerateGcode(char \*text, float height, char \*buffer)**  Parameters: char \* text – Pointer to a string of text to convert into G-code.  Float height – Float value indicating the height of the font in mm  char \*buffer – Character array for storing generated G-code commands.  Return Value – 1 if it is successful, 0 if failed  Operation Details:   * The overall operation is to generate G-code commands for a text file specified by the user. * Include the initialisation lines of G-Codes to start the robot * Be able to adjust the font height according to the user input by multiplying the x and y coordinates by a factor of:      * Checks if the word fits in the remaining width. Limit per line is 100mm width. If this exceeds, move the word to the next line. * An if loop that only writes ‘S0’ or ‘S1000’ if the pen state changes * Finally, send those commands to the robot or emulator. |
| **void SendCommands(char \*buffer)**  Parameters: char \* buffer – Contains a string of G-code commands  Operation Details:   * This sends the G-code commands to the robot. * It uses PrintBuffer() to send the contents via RS232 * Also added ‘Sleep(100)’ so that the slight delay ensure a more stable communication |

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| LoadFontData() | Load the font data file: ‘SingleStrokeFont.txt’ | The ‘SingleStrokeFont.txt’ is present and within the same folder | Returns to 1. Display to user: "Font data loaded successfully. |
| LoadFontData() | Handle missing font file | No ‘SingleStrokeFont.txt’ is present within the folder | Display to the user ‘an error has occurred: file not found’ then exit the function. ***Exit(1)*** |
| main() | Handle missing font file | No text file found within the folder.  ‘nonexistent.txt’ | Display to the user ‘an error has occurred: file not found’ then exit the function. Returns to 0 |
| main() | The user has input outside the range of ‘4mm to 10mm’ height | User input: ‘2’ | Keep displaying to the user, ‘this is outside the allowable height of text range please try again and input between 4mm to 10mm’ using a while loop |
| GenerateGcode() | Invalid character found in the text file that is not defined by the font data file. | User input: ‘Ӕ’ | Display to user: ‘Error: Invalid or undefined character encountered.’  **return 0.** |
| GenerateGcode() | Attempt to generate the right G-code for a single character | Text file only containing: ‘H’ | G-code commands should be:  G1 0 0  S1000  G1 X0 Y18  S0  G0 X12 Y0  S1000  G1 X12 Y18  S0  G0 X0 Y9  S1000  G1 X12 Y9  S0  G0 X18 Y0 |
| SendCommand() | Send G-code commands to the Arduino and verify acknowledgment | G-code: N/A | Program displays ‘Command not found, please check your Generate GCode function’ |

# Flowchart(s)

May be included as separate pdf