# Cameron McVeigh – 20465184

# System manual

# Software Description

The Programs main function is to draw out some text provided from a text file using a robot. To allow this to happen the program firstly opens the file named “SingleStrokeFont.txt”. This file contains data for each ascii character on what strokes are required by the robot to draw them out, in the format of X and Y coordinates for horizontal and vertical strokes and a Z value of either 1 for pen down or 0 for pen up. This data is read and stored into an Array of structs for easy and quick access to translate each character into their corresponding movement commands.

The program then prompts the user for a Font size input between 4 and 10mm, this value determines the vertical height of the character. A scale factor is then calculated as User input/18, this scaling factor is applied to character data later to ensure that each character drawn out conforms to the user specified font size, whilst maintaining width proportions (X and Y coordinates are scaled).

The code then prompts the user to provide the name of the text file containing the words wished to be drawn out. The program reads this text file one word character at a time until it reaches a space, indicating the end of a word and then stores this word in a WordArray. The number of characters in this word are found and therefore the size of the word in mm is found by summing the size of each character when scaled. The code calculates whether this word can fit on the current line, if it cannot fit then the word is wrapped onto the next line.

For each character in the current word being processed, the program translates the AsciiValue into the corresponding movement coordinates found in the StrokeFontData Array, this data is then scaled, and the code implements required offsets between each character.

Once each character in the word has been processed, the program translates the X ,Y and Z coordinates into G code format, storing this data as an array of chars. The code firstly needs to send initial setup commands in G code then the generated G code for the word, this is sent to the Arduino controlling the robot using rs232 serial port connection.

Only once each word is drawn out should the next word be read and processed. After all the text in the user inputted text file has been read and drawn out by the robot, the arm will move back to the origin point (0,0).

# Project Files

(Maximum 1 page)

Main.c - contains all of the code and functions needed to read all of the information in the text file, process it and output Gcode to be sent to the Arduino to control the robot.

Serial.c – Conditional compilation is used to change the mode of compilation. IE when *#define Serial\_Mode* is commented, the Gcode will be printed to the terminal. To use with Arduino - uncomment.

Serial.h – Header file used for managing serial communication. It contains preprocessor directives and declarations of functions that are used in the Serial.c file

Rs232.c – Contains a program for serial communication to the Arduino , this enables the PC to send and receive data via serial ports. The COM port need to be set here

Rs232.h – Header file for the rs232.c file. It provides function declarations for those used in rs232 necessary for serial communication, including setting of the COM port.

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| FontData | Struct | Stores font data for each ASCII character containing X,Y coordinates and pen state. Can be used to quickly translate characters to movement positions. |
| GcodeArray | Char Array | Used because Arduino accepts G code as a Character Array |
| FontDataArray | Array of FontData structs | Array used to hold Structs in format X,Y,Z for all of the ascii characters |
| CharMovementArray | Array of FontData structs | Array used to hold structs containing information on movements for the current character being processed |
| CharacterWidth | Const float | Constant float used as character width does not change, floating point division needed. |
| LineWidth | Const int | Constant integer used as line space is always 100mm, it does not change. |
| UserInput | Float | Float used to calculate scale factor from user input |
| TextFilename | Char | Char used to hold name of user inputed text file name |
| CurrentXPosition | Float | Float used to track X position in outputMovement array and allow for accurate offsets |
| CurrentYPosition | Float | Float used to track Y position in outputMovement array and allow for accurate offsets |
| WordArray | Char Array | Used to hold word currently being processed (char characters) |
| WordLength | Int | Wordlength counter |
| WordWidth | Float | Float used to ensure floating point accuracy |
| NumberofMovements | Int | Integer used to hold value of number of movements required for each word |
| asciiValue | Int | Integer representing asciivalue of each character |

# Functions

*Void PopulateFontDataArray ( Struct Multi\_FontData \*Fonts, const char \*filename)*

*Parameters:*

*Struct Multi\_FontData \*Fonts – Pointer to Multi\_FontData structure where fontdata read from the file will be store.*

*Const char \*filename – Pointer to Constant char filename which holds the name of the file to read the fontData from.*

*Return value – None (void)*

*Float GetUserInput()*

*Parameters:*

*Return value – A float , entered by user between 4 and 10 mm.*

*Void ScaleCoordinates(struct FontData CharMovementArray[], int count, float scalingFactor)*

*Parameters:*

*struct FontData CharMovementArray[] – Array of structs to hold movements for each character*

*int NumCharMovements – Integer holding the number of movements to be scaled for each character*

*float scalingFactor – the pre-calculated scaling factor to be applied to each X and Y coordinate*

*return value – None (void)*

*int RetrieveCharacterData(struct FontData FontDataArray[], int asciiValue, struct FontData CharMovementArray[], int \*NumCharMovements)*

*Parameters:*

*struct FontData FontDataArray[] – Array holding values for every ascii character*

*int asciiValue – Integer holding asciivalue of the current word*

*struct FontData CharMovementArray[] – Array of structs to hold the movements of each character*

*int \*NumCharMovements – pointer to integer tracking number of movements in each character*

*return value – integer holding the value of NumCharMovements*

*void ApplyOffset(struct FontData CharMovementArray[], int NumCharMovements, float PositionX, float PositionY)*

*Parameters:*

*struct FontData CharMovementArray[] – An array of structs containing font data for each character*

*int NumCharMovements – Number of movements per character*

*float PositionX – the current X offset to apply to each character*

*float PositionY – the current Y offset to apply to each character*

*return value – none(void)*

*float CalculateWordWidth (int WordLength, float CharacterWidth, float scalingFactor)*

*Parameters:*

*Int WordLength – Number of characters in the word*

*float CharacterWidth – Width of a single character (18.0)*

*float scalingFactor – Scaling factor applied to each character*

*return value – Total scaled width of the word*

*Void GenerateGcode( struct Fontdata CharMovementArray[], int NumCharMovements)*

*Parameters:*

*struct FontData CharMovementArray[] – Array of structs holding coordiantes for each Character*

*int NumCharMovements – The number of movements per character*

*return value- none(void)*

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| *Void PopulateFontDataArray* | Valid Input stroke file to populate array | SingleStrokeFont.txt | FontDataArray is populated with exact copy of text file |
| *Void*  *PopulateFontDataArray* | Invalid stroke file input | invalidFile.txt | “Error opening StrokeFontData file – Terminating” |
| *Float GetUserInput* | Invalid input | 3 | “Please ensure input value is between 4 and 10”  Asks user to input value again |
| *Float GetUserInput* | Valid Input | 4 | Returns UserInput |
| *Void ScaleCoordinates* | Character Input to be scaled by User input Value | Character H:  0 0 0  0 18 1  12 0 0  12 18 1  0 9 0  12 9 1  18 0 0  Using User input of 5mm | Scaled Value output:  0 0 0  0 5 1  3.33 0 0  3.33 5 1  0 2.5 0  3.33 2.5 1  5 0 0 |
| *int RetrieveCharacterData* | Character Input to retrieve movements | Character H | 0 0 0  0 18 1  12 0 0  12 18 1  0 9 0  12 9 1  18 0 0 |
| *void ApplyOffset* | Multiple character input to ensure each character is offset from the first | Input characters HH | Retrieved Values:  0 0 0  0 18 1  12 0 0  12 18 1  0 9 0  12 9 1  18 0 0  18 0 0  18 18 1  30 0 0  30 18 1  18 9 0  30 9 1  36 0 0 |
| *float CalculateWordWidth* | Input User scale factor and word to get word width output | Hello | Return word width of 90 |
| *GenerateGcode* | Input text file to get G code | File test.txt |  |