# Hubert Czajczyk 20510716

# Developing Software for a Drawing Robot

# Software Description

The software, programmed in C language, was developed to control a writing robot that would write out using a robot, a text given by a user from via a text file, in a font specified by a user by console input.

The program is required to:

* Read and store font data from a text file “SingleStrokeFont.txt”
* Get text height (font size) and text file name from the user keyboard input
* Read and process text from the input file making sure the text fits within 100 mm width
* Process and send each word separately to the robot
* Convert font data to G-Code readable by the robot for each character before sending it to the robot

For the “test.txt” text file containing:

“The quick brown fox  
jumped over the lazy dog”

And for font size 6 the software generates GCode sent to the robot to draw:

A black background with white text

Description automatically generated

Figure G-Code emulator output

# Project Files

The project makes use of the following files:

* main.c – main function containing the logic of the program
* functions.c – file containing all developed functions used in the main.c
* functions.h – header file containing headers for all the functions and defining the structures used in main.c and functions.c
* rs232.c – provided file containing functions creating a virtual RS232 serial port
* rs232.h – header files for rs232.c
* serial.c – provided file containing functions enabling serial communication with the robot
* serial.h – header file for serial.c

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale and description |
| Character characters[] | Array of structs | An array of structures containing font data for all characters with data from the Font Data text file.  Each array member is a separate character.  Using an array of structures instead of a multidimensional array is clearer for the programmer and easier for diagnostic purposes. Using structs also allows to dynamically allocate memory of each character separately which is useful as each character contains a different number of lines of coordinates defining it. |
| typedef struct {  int ASCII\_Code;  int n\_lines;  Coordinates \*line;  } Character; | struct | Structure containing the ASCII code of a character defined, the number of lines of coordinates defining that character.  This structure contains another array of structures, each array member for each line of coordinates.  Coordinates line[] is a dynamically allocated array of structures based on a number of lines for each character. |
| typedef struct {      float fontX;      float fontY;      int P;  } Coordinates; | struct | Structure containing coordinates (X, Y, P) for a single line for a single character. |
| CharacterGCode charactersGCode[] | Array of structs | An array of structures containing the generated G-Code for a specific character. The same working principle and rationale as Character characters. |
| typedef struct {      int ASCII\_Code;      int n\_lines;      GCodeCoordinates \*line;  } CharacterGCode; | struct | Refer to Character structure. A separate struct has been used instead of reusing the Character struct because the same letter may be used multiple times with different offset values. |
| typedef struct {      char X[20];      char Y[20];      char G[20];      char S[20];  } GCodeCoordinates; | struct | Structure containing char variables making up GCode line parts to be out put to the robot |
| FontSize | int | Variable storing font size input from the user, used to calculate Scale. Standard font sizes are integers. |
| Scale | float | Variable used to rescaled the GCode coordinates in order to create a text of desired font size |
| TestFileName[100] | char | Variable storing text file name given by the user containing the text to be printed out. |
| OffsetX | float | Variable storing the offset for X coordinate to add to raw coordinates of the characters. Used float as scaled values are unlikely to be integers. |
| OffsetY | float | Variable storing the offset for Y coordinate to add to raw coordinates of the characters. Used float as scaled values are unlikely to be integers. |
| CharacterX | float | Variable storing the furthest X value for each character to ensure sufficient letter spacing. Used float as scaled values are unlikely to be integers. |
| Num\_of\_characters | size\_t | Variable storing the number of characters to be stored in struct Character characters[].  size\_t was used instead of int for the use in calloc |

# Functions

*// Function reading and storing data for writing characters*

*Int ReadAndStoreFontData( char \*FileName, Character \*\*characters, size\_t \*Num\_of\_characters )*

*Parameters:*

*FileName – (input) name of the file with the font data*

*Character character[] – (output) struct to store font data for each character*

*Num\_of\_characters – (output) number of characters to be stored*

*Return value – returns 0 if successful, 1 or 2 if failed*

*// Request FontSize from the user and calculate Scale*

*Int GetFontSizeAndScale( int \*FontSize, float \*Scale )*

*FontSize – (output) font size to be input by the user*

*Scale – (output) calculated scale from FontSize used to scale GCode coordinates*

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

*// Function generating GCode and storing it in CharacterGCode struct and outputting each character’s furthest X coordinate*

*Int GenerateGCode( int m, int letter, CharacterGCode \*\*charactersGCode, Character \*\*characters, float Scale, float OffsetX, float OffsetY, float \*CharacterX )*

*m – (input) element number of Character character[] storing the character for which GCode is to be generated*

*letter – (input) element number of the CharacterGCode charactersGCode[] storing the GCode for that character*

*Scale – (input) scale to multiply the coordinates by before generating GCode*

*OffsetX, OffsetY – (input) offset values to add to the coordinates before generating GCode*

*CharacterX – (output) X value across the drawing area obtained from the furthest X coordinate of the processed character*

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

*// free all allocated memory to arrays of structures*

*int freeCharacters( Character \*characters, CharacterGCode \*charactersGCode, size\_t \*Num\_of\_characters )*

*Character character[] – (input) array of structs to free allocated memory*

*CharacterGCode characterGCode [] – (input) array of structs to free allocated memory*

*Return value – returns 0*

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| *Int ReadAndStoreFontData()* | Invalid file | FileName | 1 |
| *Int ReadAndStoreFontData()* | “SingleStrokeFont.txt” | FileName | 0 |
| *Int ReadAndStoreFontData()* | failure to allocate memory | \*characters = calloc(),  (\*characters)[j].line = calloc() | 2 |
| int GenerateGCode() | failure to allocate memory | \*charactersGCode)[letter].line | 1 |
| int GetFontSizeAndScale() | 3, 6 | FontSize | 1, 0 |
| int main() | Invalid file name | TextFile | 1 |
| int main() | “test.txt” / “RobotTesting.txt” | TextFile | 0 |

# Flowchart(s)

Attached in separate PDF