# **Developing software for a writing robot**

# [William Alsbury, 20466404]

# **Outline of the Problem to be Solved**

The goal of this project is to develop software that will transmit commands from a file, via a virtual RS232 serial port, to a writing robot that is able to draw out text. The text in question must be within the height range of 4 and 10mm whilst also fitting within a width area of 100mm without any breaks in the words. If both parameters are adhered to then the project may be deemed as successful if the sentence generated is spelt correctly and makes grammatical sense.

To be able to make this goal possible, the code written must meet the following specifications for it to work. Each step will be explained to gain an understanding of what is required. Firstly, all code should be developed using git for version control. This means that code committed to the git repository can be edited and developed in stages providing a safety blanket for if something goes wrong later down the line. Version control also allows for external input when editing the code, however, this will not be utilised during this project.

Secondly, the program needs to open and read the contents of a file named ‘SingleStrokeFont.txt’ which has been constructed in the font file format provided. All the data in this file needs to be read and stored in an appropriate data format for it to be easily accessible by the program.

Thirdly, obtain the height at which the letters should be drawn using keyboard input (between 4 and 10mm). To do this, the program should prompt the user to enter the height for the letters with code that includes input validation to ensure the value is within the required range. Once the height is prompted this data will need to be scaled for it to be ‘drawn’ correctly. The font data specifies that the height of a letter is 18 units which acts as the base height within the font data coordinate system. To draw the letters to the specified height by the user, the program needs to scale the movement data from the font file. The scaling factor should be calculated as height/18 which adjust the movements to the specified height by the user.

The program should prompt the user to input the name of the file containing the text which will then be read and processed by the drawing robot. Each word sent to the robot must be processed one at a time before leaving the program. This is so that each letter can be scaled to the correct height before being output by the robot. Once one word has been output, the next word can be processed and scaled accordingly in a continuous workflow. Additionally, the code should be designed so that it can handle text files of any length by reading and processing the file in manageable segments and not by loading the file all at once.

Finally, before sending the font data to the robotic arm it needs to be converted into the G-Code format for it to be processed correctly by the Arduino. This will control the robotic arm by telling it when to raise and lower and move the pen to the specified x,y coordinates. If all of this is done correctly, the robotic arm should write lines of comprehensible text. To finish the process, the code should tell the robotic arm to return to its original position at (0,0), which at that point the code should terminate, and the program is finished.

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| Word | Char\* | Singles out a particular word in the text file so they can be examined one by one. |
| Scaled\_Data | Float | A float allows for precise scaling operations. |
| Font\_Data | Struct | Groups a character and its width together, making it easier to manage the mapping between them. |
| InputHeight | Int | Results in a whole number between 4 and 10mm |
| Width | Float | Stores the width of a single character as retrieved from Font\_Data. |
| Total\_Width | Float | Accumulates the total width of the word as each character's width is added. |
| X\_offset | Int\* | Pointer allows easy reference number for the x\_offset |
| Y\_offset | Int\* | Pointer allows easy reference number for the y\_offset |
| Read\_Font\_Data | Struct | Makes it easier for the program to manage |
| Scaled\_width | Float | Allows for a precise value |
| Font\_count | Int\* | Counts specific number of characters |
| gcode | Char\* | Pointer to process each word at a time |

# Function Declarations

1. void Open\_Font\_Data(struct filename)

Parameters:

* Filename – chosen text file containing font data
* Return Value – File handle returned

1. struct Read\_Font\_Data(struct filename, int\* font\_count)

Parameters:

* Filename - Chosen text file containing font data
* Font\_count – Number of character entries read from font data file
* Return Value – N/A

1. float Scaled\_Data(int InputHeight, float scaled\_height)

Parameters:

* InputHeight – Input height in millimetres
* Scaled\_height – converted Input height (x/18)
* Return Value - Returns scaled height if successful

1. array Open\_Text\_File(array Filename)

Parameters:

* Filename - chosen text file containing text
* Return value - File handle returned

1. void Process\_Word(char\* word, float scaled\_data, struct font\_data, int font\_count, int\* x\_offset)

Parameters:

* Word – pointer to return each individual word at a time
* Scaled\_data – float containing scaled height
* Font\_data – stored word data in font file format
* Font\_count - Number of character entries read from font data file
* X\_offset – Horizontal distance in coordinate system for spacing text properly
* Return Value – N/A

1. float Word\_Width(char\* word, float scaled\_width, char\* width, float total\_width, float scaled\_data)

Parameters:

* Word - Pointer to return each individual word at a time
* Scaled\_width – float to return scaled width of each character
* Width – Regular width of each character
* Total\_width – Float returns total scaled width of word
* Scaled\_data - Float containing scaled height
* Return Value – Total\_width

1. Void Send\_G-Code\_To\_Arduino(char\* gcode, Serial Connection)

Parameters:

* Gcode – Pointer to process gcode for each word one at a time
* Serial Connection – Connection between robot and the program
* Return value – N/A

1. void Line\_Breaks(int\* x\_offset, int\* y\_offset, int current\_line\_number)

Parameters:

* X\_offset - Horizontal distance in coordinate system for spacing text properly in mm
* Y\_offset – Controls spacing of the lines in mm
* Current\_line\_number – Which line of text the robot is operating on
* Return Value – N/A

1. void Return\_Pen(Serial Connection, int x\_origin, int y\_origin)

Parameters:

* Serial Connection - Connection between robot and the program
* X\_origin – Original coordinate of the pen of the x-axis
* Y\_origin – Original coordinate of the pen on the y-axis
* Return Value – N/A

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| Main() | Execution of code with valid data | Valid serial connection, input of 'SingleStrokeFont.txt', input height of 7mm, input of valid written text file containing ‘Apple Tree’ | ‘Apple Tree’ is drawn with correct spacing and pen returns to (0,0) |
| Open\_Font\_Data | Opening File | Inputting the desired file | File handle returned if successful |
| Read\_Font\_Data | Test reading a valid font file | A valid path to ‘SingleStrokeFont.txt’ with correct data formatting | Font data converted and stored as a struct |
| Scaled\_Data | Test scaled data | Height = 5, Height/18 | Converted height is 5/18 |
| Open\_Text\_File | Opening File | Inputting the desired text file | File handle returned if successful |
| Process\_Word | Test processing a single word | Word = ‘apple’, font\_data with data for each letter, scaled\_data for each letter | Correct G-Code is generated and sent for each character in ‘apple’. |
| Word\_Width | Test width of a single word | Word = ‘orange’, scaled\_width of each letter using scaled\_data and original width | Correct Total\_width for the word ‘orange’ |
| Send\_G-Code\_To\_Arduino | Test sending valid G-Code | Gcode for the word ‘grape’ is sent to the Arduino | G-Code sent successfully with acknowledgment |
| Line\_Breaks | Test handling a line break | X\_offset = 30, y\_offset = -15, current\_line\_number = 2 | X\_offset returns to 0, y\_offset = -20, current\_line\_number = 3 |
| Return\_Pen | Test moving pen to (0,0) | X\_origin = 0, y\_origin=0, valid serial connection | G-Code for lifting pen and moving to (0,0) sent successfully. Pen returns to (0,0) |

# Flowchart(s)

**(Included as a separate PDF)**