# Name & Student ID: Archie Clayton, 20454063

# Title: Developing software for a writing robot planning report

# Outline of the Problem to be Solved

The program is developed for controlling a writing robot that reads text from a file and then the robot reproduces the text from the file as drawn text on a piece of paper. This is achieved by generating and sending G-code commands that control the robot’s movements and allow it to write with the pen.

To begin, the program reads the data from the file, ‘SingleStrokeFont.txt,’ which contains X, Y & pen up/down data required to draw out all ASCII characters. This file will be read and stored in the memory of a structured array, where each entry in the array will represent a drawing command for an individual character, consisting of a command code, x-value, and pen state. The structure allows the program to efficiently access and display the drawing data for any character as specified by user input.

The user is then prompted to enter the height for the drawing of the characters, ranging from 4 to 10 mm. The font file uses units of 18 unit, so the movements need to be scaled by the fraction height/18 so that they are drawn at the required height.

The program then reads the text to be drawn from a text file specified by the user. This file may contain text of any length, but the program will process it in a way that ensures words fit within the 100 mm width limit. This will be achieved by making sure that if a word can’t be drawn on the same line within the guidelines, then the word will be placed on the next line down whilst ensuring that the lines will be spaced 5 mm apart vertically.

The G-code commands are sent to the Arduino which communicates with the robot arm. The G-code commands will then control the pen’s movement and position, as well as its state (up or down), for each character. This is achieved through the font data being converted into G-code commands such as the to set the pen speed (F1000), raise or lower the pen (S0 for pen up and S1000 for pen down). I will then use the commands to instruct the robot arm on what to do and making sure it finishes in the pen-up state at position 0,0.

In summary, this project will produce a program that at reads, scales, formats, and transmits drawing commands to a robot arm to draw text within a specified area. This will be achieved by reading the font data from a text file, allowing the user to choose a specified height and the text to be drawn being taken from a user-specified file. The program then converts the font data into G-Code commands, which are sent to the Arduino to control the robot’s arm to manage the pen’s position and state for each character and making sure it finishes in the pen-up position at the origin (0,0).

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| CharacterHeight | float | Stores the height inputted by the user (between 4-10mm) to scale the font data. |
| CharacterWidth | float | Stores width of each individual character used to calculate the word width and X\_offset |
| ScalingFactor | float | Uses characterHeight/18 so text is draw at correct height |
| MaxWidth | float | Sets the maximum allowable width (100mm) for each line and ensures it is consistent throughout. |
| LineSpacing | float | Sets the vertical spacing (5mm) between each line and ensures it is consistent throughout. |
| TextRead | Char[ ] | Stores text to be drawn which is read from the user’s file |
| FontDataArray | Struct FontChar [] | Array where each element corresponds to a character’s movement data (movements and pen status) for each ASCII character. |
| x\_offset | float | Tracks current x position on the lines and updates after each character or word |
| y\_offset | float | Tracks current Y position for each line and increases by 5mm for a new line |
| WordWidth | float | Stores calculated width of each line, allowing line width checks before placement on line. |
| GCodeCommands | list | A list to store the G-code commands for each letter to formulate the words. |
| MaxHeight | float | Constant ensuring robot doesn’t draw outside of area and checks y offset bounds. |
| FontFilePtr | FILE\* | Pointer to font file used to store font data from SingleStrokeFont.txt. |
| TextFilePtr | FILE\* | Pointer to the user’s text file used for reading the text to be drawn. |

# Function Declarations

Function to open font file and assign a file pointer

int OpenFontFile(FILE \*\*FontFilePtr);

Parameters:

FILE \*\*FontFilePtr: pointer to a file pointer that will open font file.

Return:

1 if file opens successfully, 0 if it fails to open

Function to read font data from a text file

int ReadFontData(FILE\* FontFilePtr, FontChar FontDataArray[])

Parameters:

FontFilePtr: A pointer to the opened font file (SingleStrokeFont.txt), passed into the function.

FontChar fontData[]: array to store the font data

Return:

1 if font data is successfully read, 0 if there is an error

Function to calculate the scaling factor for scaling the font based on user input height

CalculateScalingFactor(float UserHeight)

Parameters:

Float Userheight: stores the inputted height from the user to work out scale factor

Return:

A float of the scaling factor

Function to apply the scaling factor to the font data

void ScaleStrokeData(struct FontChar FontDataArray[], float ScalingFactor)

Parameters:

FontChar[] FontDataArray: Array containing font data for each character.

float ScalingFactor: Scaling factor to adjust the character size to the desired height

Return:

The data is now updated for the scaling factor

Function to open the text file containing the text to be drawn

FILE\* OpenTextFile(void);

Parameters:

None as user specifies file

Return:

1 if file opens successfully, 0 if it fails to open

Function to read the text from the file into the TextRead array

int ReadTextFile(FILE\* textFile, char\* textRead);

Parameters:

TextFile – FILE\*: A pointer to the opened text file.

TextRead – char[]: A buffer to store the contents of the text file.

Return:

1 indicates a successful reading, 0 indicates an error

Function to process the text

int ProcessTextFile(char\* TextRead, FontChar\* FontDataArray, float\* x\_offset, float\* y\_offset);

Parameters:

char\* TextRead: Text that has been read from file to be drawn.

FontChar\* FontDataArray: Array containing font data for each character.

float\* x\_offset: A pointer to current x-coordinate of text drawing position.

float\* y\_offset: A pointer to current y-coordinate of text drawing position.

Return:

An integer where 1 indicates text has been successfully processed and 0 indicates an error in the processing

Function to calculate total width of a word

float CalculateWordWidth(Char word[], FontChar FontData[], float ScalingFactor)

Parameters:

Char[] word: array of characters forming the words

FontChar[] FontDataArray: Array containing font data for each character.

Float ScalingFactor: Scaling factor to adjust the character size to the desired height

Return:

Total width of word in terms of x movement that has been scaled by scaling factor

Function to check if the word fits within the maximum width, if not, move to the next line

int CheckLineWidth(float x\_offset, float WordWidth);

Parameters:

float x\_offset: The current x-coordinate of the pen on the current line, which represents the position where the next word starts. This is updated after each word.

float WordWidth: The calculated width of the word to be drawn, which includes the combined width of all characters in the word

Return:

Integer returns 1 if word fits on current line, and 0 if word doesn’t fit and excceds maximum width.

Function to generate G-code commands for each character, drawing the word

void GenerateWordGCode(char\* word, FontChar\* FontDataArray, float\* x\_offset, float\* y\_offset, list\* GCodeCommands);

Parameters:

char\* word: Current word being processed

FontChar\* FontDataArray: Array containing font data for each character

float\* x\_offset: A pointer to current x-coordinate of text drawing position.

float\* y\_offset: : A pointer to current y-coordinate of text drawing position.

list\* GCodeCommands: A list to store the generated G-code commands for the word.

Return:

Function generates G-code commands for each letter in word and add it to GCodeCommands list.

Function to send G-code to Arduino

int sendGCode(list\* gCodeCommands);

Parameters:

list\* gCodeCommands: A list containing the G-code commands generated for the entire text to be drawn. Each command corresponds to an individual movement or pen state change.

Return:

An integer, where 1 indicates that the G-code commands were successfully sent to the Arduino and executed, and 0 indicates failure (e.g., if there is an issue with the communication or transmission to the Arduino).

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| OpenFontFile() | Valid File | FILE \*\*fontFilePtr = NULL; | Returns 1, and fontFilePtr is assigned to the opened file pointer. |
| OpenFontFile() | File not found | filename = "NAFontFile.txt" | Returns 0, displays error message: "Font file not found." |
| ReadFontData() | Valid Data | fontFilePtr points to SingleStrokeFont.txt with valid font data | Returns 1, the font data goes into FontDataArray. |
| ReadFontData() | Invalid Data | fontFilePtr points to a corrupted SingleStrokeFont.txt or a different text file | Returns 0, displays error message: "Invalid font data." |
| CalculateScalingFactor() | Valid user height | UserHeight = 8.0 | Returns ScalingFactor = 8.0 / 18.0 = 0.4444 |
| CalculateScalingFactor() | Invalid user height | UserHeight = 18.0 | Returns error message: “Value not in accepted range of heights.” |
| ScaleStrokeData() | Valid Scaling | Using font data array and scaling factor | Returns the font at the scaled height |
| OpenTextFile() | Valid File | Text file name = "input.txt" (input is the input from the user) | Returns 1 and the file pointer is assigned correctly. |
| OpenTextFile() | File not found | filename = "NATexttFile.txt" | Returns 0 and displays error message: "Text file not found." |
| ProcessTextFile() | Valid text | textRead contains no special symbols and not too many words. | Returns 1, processes text and updates x\_offset, y\_offset after drawing. |
| ProcessTextFile() | Invalid text | Textread contains special symbol or too long | Returns 0, displays error message "Invalid character in text." or “Please shorten text in file, too many words” |
| calculateWordWidth() | Valid word | Word = “Hello” | Returns the correct word width |
| checkLineWidth() | Word fits | The first word in the text file | Returns 1, x\_offset updates to the next word |
| checkLineWidth() | Word does not fit | Word that goes over the max\_width on the line | Returns 0, x\_offset returns to 0 and y\_offset is increased by 5mm. |
| generateWordGCode() | Valid word | Word = “Hello” | Returns 1, adds G-code commands for each character in word |
| sendGCode() | Successful send | The ‘GCodeCommands’ | Returns 1, successfully send G-code to arduino and completes drawing |
| sendGCode() | Failed send | The ‘GCodeCommands’ | Returns 0, displays error message: "Failed to send G-code to Arduino." |

# Flowchart(s)

Included on separate pdf