# Name & Student ID

Dylan Sanders - 20352508

# Title

Developing software for a writing robot

# Outline of the Problem to be Solved

This project is about creating a program that controls a writing robot to draw user-specified text on a surface. The program will take text entered by the user, turn it into movement commands, and then send these commands to the robot, which follows them to draw the text.

1. **Get Input from the User**: First, the program will ask the user to enter the **name of a text file** that contains the words to be drawn. This file can contain text of any length, and the program will process each word individually. Additionally, the program will prompt the user for the **height of the letters** (between 4mm and 10mm). This height setting will apply to all text in the file, ensuring that each letter is scaled the same.
2. **Load Font Data**: Each letter is composed of specific strokes, defined in a file called SingleStrokeFont.txt, which includes instructions for moving the pen up and down to form each character. All movements are offset relative to 0,0 and its origin corresponds to the final point in the definition of the previous letter written.
3. **Read and Process Each Word**: The program will open the specified text file and read it **word by word.** For each word, the program will generate drawing instructions (G-code) for each character, scaling the letters to match the chosen height. The program will also ensure that each line does not exceed a width of **100mm**; if a line becomes too long, it will automatically add a line break, with a **5mm space** between lines.
4. **Convert Text to Robot Commands (G-code):** For each word, the program translates each character into a set of G-code commands. These commands specify X/Y coordinates and pen movements (up/down) to tell the robot where and when to draw each stroke. Each word’s G-code is sent to the robot before moving on to the next word in the file.
5. **Send Commands to the Robot:** As it generates G-code, the program sends these commands to the robot over a digital RS232 serial connection.
6. Drawing of text: The robot moves to the specified X and Y coordinates, adjusting the pen's position (up or down) as needed to write the words chosen by the user from a text file. It uses the appropriate font and adjusts the letter height accordingly.

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| textfileName | String | Stores the name of the text file provided by the user that contains the words to be drawn. String is used to store the file name, as it allows easy handling of text and file paths, which are required for file input. |
| textHeight | float | Holds the user-defined height of each letter, ensuring letters are scaled consistently (4mm–10mm). float is used for textHeight because it allows precise scaling of letter heights in millimeters, including decimal values. |
| word | String | String is appropriate for word as it temporarily holds each text word from the text file as it is processed to generate drawing commands. |
| gCodeCommands | Array | An Array is used for gCodeCommands to store sequences of G-code instructions in order. |
| Position | Struct | Struct as will include the x and y coordinates obtained from the SingleStrokeFont.txt file. These coordinates will be integers. |
| penState | boolean | boolean is used for penState as it simply tracks two states: pen up (false) or pen down (true), making boolean ideal. |
| maxlineWidth | integer | Tracks if a line break is needed when the current line width exceeds 100mm. Since its 100mm and stays at that value integer is a good data type. |
| lineWidth | float | float is used for lineWidth as it needs to accumulate and compare precise widths, and float supports fractional values. |
| linespacing | integer | int is used here since the spacing is a fixed whole number (5mm), and no decimals are needed. |
| characterwidth | float | Width of the character can be more precise with a float and will make more accurate when calculating the line width. |
| WordSpacing | float | WordSpacing is needed to calculate the next position of the pen and float is appropriate as it can be more precise. |

# Function Declarations

Position scaleCharacterDimensions(struct position, float textHeight, struct fontData);

* **Parameters**:
  + Position: Contains x and y coordinates.
  + textHeight: The desired letter height defined by the user (between 4mm and 10mm).
  + fontData: Holds stroke instructions for each letter.
* **Return Value**: Returns a position struct with scaled X and Y coordinates.

Array<String> generateGCodeCommands(const String& word, struct position, bool penState);

* **Parameters**:
  + word: String to be converted into G-code commands.
  + Position: Contains x and y coordinates.
  + penState: Current state of the pen.
* **Return Value**: Array<String> – returns an array of G-code commands for the word.

bool checkLineWidthAndReset(float characterWidth, float\* lineWidth, int maxlineWidth, float Wordspacing);

* **Parameters**:
  + characterWidth: The width of the character to be added.
  + lineWidth: Pointer to the current line width (float).
  + maxlineWidth: The maximum allowable width for a line.
  + WordSpacing: Space between words.
* **Return Value**: Returns true if a new line is needed (i.e., lineWidth is reset), false otherwise.

float calculateWordWidth(const String& word, float characterWidth);

* **Parameters**:
  + word: String containing the word.
  + characterWidth: Width of each character.
* **Return Value**: float – returns the total width of the word.

bool readNextWord( FILE\* textfileName, const String& word)

* **Parameters**:
  + word: String containing the word
  + textfileName: Text containing the text the user wants to draw
* **Return Value**: True or False if the word is successfully read

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| scaleCharacterDimensions | Correct scaling | Will scale x and y movements by fraction height / 18 to fit required height | X and y movements will be scaled and fit required height |
| generateGCodeCommands | G code for a word | Will use the word, x,y and pen state to translate into g code | Returns correct g code command. |
| CalculateWordWidth | Test width of word | Uses the characters, width of characters. | Outputs width of the word |
| checkLineWidthAndReset | If the word fits on the line | Uses width data to see if room is left for whole word on the line | Will give true or false if the word can fit on the line. |
| readNextWord | Reads the next word of the file | The next word in the text file written by the user. | Will return true or false if the word has been correctly read |

# Flowchart(s)

Included as separate pdf