# Stefano Geymonat

# Student ID: 20465966

# Computer Engineering and Mechatronics Project: Developing software for a writing robot

# Software Description

The program controls a robot that draws text by interpreting font data and sending G-code commands to a writing robot. It begins by initializing communication with the robot through a serial port. If the port cannot be opened, it exits with an error. Once communication is established, the robot is "woken up" and positioned at the origin (X=0, Y=0). Initial commands are sent to set up parameters such as the drawing speed and the pen's state (up or down), preparing the robot to draw.

The program then loads font data from a file called SingleStrokeFont.txt. This file contains the coordinates and stroke information for each character in a simple single-stroke font. Each character's data is represented as a series of X, Y positions, and a Z value, which determines whether the pen should be up or down while moving to draw strokes. This data is stored in a structured array for efficient access during drawing.

Next, the user is prompted to provide a scaling factor for the drawing which must be a value between 4mm and 10mm, ensuring that the input falls within a valid range. The scaling factor is used to resize the font proportionally for the drawing. The user is also asked to provide the name of a text file containing the words or phrases to be drawn. This file is opened, and the program processes/reads it word by word. For each word, the program calculates its width (based on the font data and scaling factor) and checks if it fits on the current line of max width 100. If the word exceeds the remaining space, the robot moves to a new line by adjusting the Y-coordinate and resetting the X-coordinate.

For each character in the word, the program retrieves its corresponding stroke data, applies the scaling factor, and generates G-code commands to draw the character. These commands move the pen to the correct positions and toggle its state between up and down to draw the strokes. The robot is instructed to move incrementally along the X-axis to ensure proper spacing between characters and words.

Once all the words are processed, the robot is returned to its home position (X=0, Y=0), and the communication port is closed. The program handles user input validation, file errors, and ensures that the robot's movements are precise, resulting in accurately scaled and positioned text drawings.

# Project Files

Main.c –

RS232.c –

RS232.h –

Serial.c –

Serial.h –

SingleStrokeFont.txt –

Text.txt –

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| scale | float | Holds the user-specified scaling factor for resizing characters. |
| ch | int | |  | | --- | | Used to clear the input buffer when validating user input. | |
| remaining\_space | float | |  | | --- | | Tracks the available space left on the current line for text. |  |  | | --- | |  | |
| current\_Xpos | float | |  | | --- | | Keeps track of the robot's current X-coordinate during drawing. |  |  | | --- | |  | |
| current\_Ypos | float | |  | | --- | | Keeps track of the robot's current Y-coordinate during drawing. |  |  | | --- | |  | |
| word | Char[100] | |  | | --- | | Stores each word read from the input text file for processing. |  |  | | --- | |  | |
| filename | Char[200] | |  | | --- | | Holds the name of the input text file provided by the user. |  |  | | --- | |  | |
| SingleStrokeData | DataEntry[1027] | |  | | --- | | Array storing the font data for all characters in the single-stroke font file. |  |  | | --- | |  | |
| DataEntry | Struct | |  | | --- | | Structure that defines a single font stroke with X, Y, and Z values. |  |  | | --- | |  | |
| stroke\_count | Int | |  | | --- | | Stores the number of strokes required to draw a specific character. |  |  | | --- | |  | |
| buffer | Char[4000] | |  | | --- | | Temporary storage for constructing and sending G-code commands to the robot. |  |  | | --- | |  | |
| scaleFactor | float | |  | | --- | | Stores the normalized scaling factor for adjusting character dimensions. |  |  | | --- | |  | |
| fontFile | File\* | |  | | --- | | File pointer for reading font data. |  |  | | --- | |  | |
| inputFile | File\* | |  | | --- | | File pointer for reading the input text file with words to draw. |  |  | | --- | |  | |
| ascii\_val | Int | |  | | --- | | ASCII value of the character being processed for matching font data. |  |  | | --- | |  | |
| wordWidth | Float | |  | | --- | | Calculates the total width of a word based on the font data and scaling. |  |  | | --- | |  | |
| line\_spacing | Float(constant) | |  | | --- | | Adjusts the Y-axis when moving to a new line for text drawing. |  |  | | --- | |  | |
| CHAR\_WIDTH | Float(constant) | Represents the default width of each character in the font before scaling. |

# Functions

**Function:** float get\_scale\_factor()  
**Description:** Prompts the user to input a scaling factor between predefined limits and validates it.  
**Parameters:** None  
 **Return value:** Returns the scaling factor normalized by CHAR\_WIDTH.

**Function:** float calculate\_word\_width(const char \*word, float scaleFactor)  
**Description:** Calculates the width of a word based on the number of characters and scaling factor.  
**Parameters:**

word – Pointer to the word (string) whose width is being calculated.

scaleFactor – Scaling factor applied to each character's width.  
**Return value:** Returns the calculated word width as a float.

**Function:** int fits\_in\_line(float \*remaining\_space, float word\_width)  
**Description:** Checks if a word can fit within the remaining space on the current line.  
**Parameters:**

remaining\_space – Pointer to the available space left on the line.

word\_width – The width of the word to be drawn.  
**Return value:** Returns 1 if the word fits, otherwise returns 0.

**Function:** FILE\* open\_file(const char \*filename)  
**Description:** Opens a file with the specified filename for reading.  
**Parameters:**

filename – Pointer to the name of the file to open.  
**Return value:** Returns a file pointer (FILE\*) if the file is successfully opened, otherwise returns NULL.

**Function:** DataEntry\* find\_character\_data(char character, DataEntry \*fontData, int \*stroke\_count)  
**Short description:** Retrieves stroke data for a specific character from the font data array.  
**Parameters:**

character – The character whose stroke data is being searched.

fontData – Pointer to the array of font data (DataEntry structures).

stroke\_count – Pointer to store the number of strokes required for the character.  
**Return value:** Returns a pointer to the first stroke data entry for the character, or NULL if not found.

**Function:** void generate\_gcode\_for\_word(const char \*word, DataEntry \*fontData, float scaleFactor, float \*current\_Xpos, float current\_Ypos)  
**Short description:** Generates and sends G-code instructions to draw a word based on font data and scaling.  
**Parameters:**

word – Pointer to the word (string) to draw.

fontData – Pointer to the array of font data (DataEntry structures).

scaleFactor – Scaling factor for the font dimensions.

current\_Xpos – Pointer to the robot's current X-coordinate, updated after drawing each character.

current\_Ypos – Current Y-coordinate of the robot, used as the baseline for drawing.  
**Return value:** None

**Function:** void reset\_position(float \*current\_Xpos, float \*current\_Ypos, float scaleFactor, float \*remaining\_space)  
**Description:** Resets the drawing position to the start of the next line.  
**Parameters:**

current\_Xpos – Pointer to the X-coordinate, reset to 0.

current\_Ypos – Pointer to the Y-coordinate, incremented for the next line.

scaleFactor – Scaling factor used to calculate line spacing.

remaining\_space – Pointer to reset the remaining space on the line to LINE\_WIDTH.  
**Return value:** None

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| get\_scale\_factor() | |  | | --- | | User enters a valid scaling factor within the range. |  |  | | --- | |  | | |  | | --- | | Input: 6 | | |  | | --- | | Output: 0.333 (scale factor divided by CHAR\_WIDTH = 18.0). | |
| |  | | --- | | get\_scale\_factor() |  |  | | --- | |  | | |  | | --- | | User enters an invalid scaling factor (e.g., out of range or non-numeric). |  |  | | --- | |  | | |  | | --- | | Input: "abc" or -5 |  |  | | --- | |  | | |  | | --- | | Prompt reappears until valid input is given. |  |  | | --- | |  | |
| calculate\_word\_width() | |  | | --- | | Calculate the width of a word using the given scale factor. |  |  | | --- | |  | | |  | | --- | | Input: word = "Hello", scaleFactor = 0.5 |  |  | | --- | |  | | |  | | --- | | Output: 45.0 (5 characters × 18.0 × 0.5). |  |  | | --- | |  | |
| |  | | --- | | fits\_in\_line() |  |  | | --- | |  | | |  | | --- | | Check if a word fits within the remaining space on a line. |  |  | | --- | |  | | |  | | --- | | Input: remaining\_space = 50.0, word\_width = 40.0 |  |  | | --- | |  | | |  | | --- | | Output: 1 (True, word fits). Remaining space updates to 10.0. |  |  | | --- | |  | |
| |  | | --- | | fits\_in\_line() |  |  | | --- | |  | | |  | | --- | | Check if a word does not fit within the remaining space on a line. |  |  | | --- | |  | | |  | | --- | | Input: remaining\_space = 30.0, word\_width = 40.0 |  |  | | --- | |  | | |  | | --- | | Output: 0 (False, word doesn’t fit). Remaining space stays 30.0. |  |  | | --- | |  | |
| |  | | --- | | open\_file() |  |  | | --- | |  | | |  | | --- | | Open a valid file successfully. |  |  | | --- | |  | | |  | | --- | | Input: "SingleStrokeFont.txt" |  |  | | --- | |  | | |  | | --- | | Output: File pointer to the opened file. |  |  | | --- | |  | |
| |  | | --- | | open\_file() |  |  | | --- | |  | | |  | | --- | | Attempt to open a non-existent file. |  |  | | --- | |  | | |  | | --- | | Input: "NonExistentFile.txt" |  |  | | --- | |  | | |  | | --- | | Output: NULL. Prints "Error opening file: NonExistentFile.txt". |  |  | | --- | |  | |
| |  | | --- | | find\_character\_data() |  |  | | --- | |  | | |  | | --- | | Search for character stroke data that exists in the font array. |  |  | | --- | |  | | |  | | --- | | Input: character = 'H', valid fontData. |  |  | | --- | |  | | |  | | --- | | Output: Pointer to the first stroke entry for 'H' and the stroke count. |  |  | | --- | |  | |
| |  | | --- | | find\_character\_data() |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Search for a character not present in the font array. |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Input: character = '!', valid fontData. |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Output: NULL. |  |  | | --- | |  | |  |  | | --- | |  | |
| |  | | --- | | generate\_gcode\_for\_word() |  |  | | --- | |  | | |  | | --- | | Generate G-code instructions for a valid word. |  |  | | --- | |  | | |  | | --- | | Input: word = "Hi", valid fontData, scaleFactor = 0.5, current\_Xpos = 0, current\_Ypos = 0. |  |  | | --- | |  | | |  | | --- | | Output: G-code commands sent for each character in "Hi". current\_Xpos updated after each character. |  |  | | --- | |  | |
| |  | | --- | | generate\_gcode\_for\_word() |  |  | | --- | |  | | |  | | --- | | Attempt to generate G-code for a word with an invalid character (not in font array). |  |  | | --- | |  | | |  | | --- | | Input: word = "$", valid fontData. |  |  | | --- | |  | | |  | | --- | | Output: Prints "Character '$' - Stroke data not found." No G-code commands sent. |  |  | | --- | |  | |
| |  | | --- | | reset\_position() |  |  | | --- | |  | | |  | | --- | | Reset drawing position to the start of a new line. |  |  | | --- | |  | | |  | | --- | | Input: current\_Xpos = 10, current\_Ypos = -10, scaleFactor = 0.5, remaining\_space = 20. |  |  | | --- | |  | | |  | | --- | | Output: current\_Xpos = 0, current\_Ypos = -19. Remaining space resets to LINE\_WIDTH. |  |  | | --- | |  | |
| |  | | --- | | main() |  |  | | --- | |  | | |  | | --- | | Program initializes robot successfully, reads font data, processes an input text file, and generates G-code commands for each word, resetting position as needed for new lines. |  |  | | --- | |  | | |  | | --- | | Input: SingleStrokeFont.txt (valid font file), a text file with words (e.g., "Hello World"), scale factor = 5. |  |  | | --- | |  | | Output: Robot draws the words as G-code commands are sent line by line. Program exits after closing the COM port. |

# Flowcharts

All flowcharts added to folder as PDFs: