# Name & Student ID: Akshay Nair, 20485507

# Title: Robot Planning Report

# Outline of the Problem to be Solved

(Maximum 1 page)

The project aims to develop a software application that enables a writing robot to draw text by sending commands through a virtual RS232 serial port to an Arduino microcontroller, which controls the robot arm. The software reads input text from a file, converts it into drawing instructions using a pre-defined single-stroke font, and transmits these commands as G-Code to the robot. The text height is user-adjustable between 4mm and 10mm, and the application ensures proper letter spacing (X-offset) and line spacing (Y-offset) while keeping the text within a 100mm width limit.

To achieve this, the software reads font data from the "SingleStrokeFont.txt" file, processes the input text, and calculates the appropriate scaling for X and Y movements based on the user-provided text height. It generates G-Code instructions for each character, adjusting positioning based on the previous character's endpoint, and handles line breaks through ASCII codes for line feed (LF) and carriage return (CR). Communication with the Arduino is facilitated using serial commands to control the pen’s movement, ensuring it raises and lowers as needed. The software concludes each task with the pen returning to the (0,0) position in a pen-up state.

Development is supported with resources such as code samples for serial communication, error handling, and G-Code transmission. The project will use git for version control, and the code will be documented with flowcharts, function definitions, and test cases. Validation will be conducted through comprehensive testing to ensure that the robot follows the input text's format, adheres to the drawing specifications, and produces readable text with correct spacing.

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| fontData | array or list | To store the font data read from SingleStrokeFont.txt, allowing the program to map each character to its drawing commands. |
| textInput | string | To hold the user input text that needs to be drawn. Strings allow easy processing and character-by-character access. |
| fontSize | int | Represents the height of the text (between 4 and 10mm). An integer is appropriate for size values. |
| maxWidth | int | The maximum width (100mm) of the writing area. This ensures the text fits within the defined space. |
| xPosition, Yposition | float | Used for the X and Y coordinates of the robot’s pen. Floats allow more precise positioning. |
| commandBuffer | list or queue | To store the generated G-Code commands for sending to the robot. A list/queue helps in sequential command execution. |
| lineBreakFlag | boolean | Indicates whether a line break is needed to maintain text within the maximum width. |
| currentChar | char | Holds the current character being processed for drawing. Useful for parsing and command generation. |
| penState | |  | | --- | | boolean |  |  | | --- | |  | | Represents the pen's state (up or down). True for pen down (drawing) and false for pen up (moving). |
| scaleFactor | float | Used for scaling font data to the user-defined size. Floats allow precise scaling calculations. |

Extend table as required

# Function Declarations

*Only include functions that you will develop.*

1. loadFontData from Text File (“SingleStrokeFont.txt”)

void int loadFontData(char\* filename, FontData\* fontStruct)

Parameters:

* filename – name of the file containing font data (input).
* fontStruct – pointer to a structure where the font data will be stored (output).
* Return value – void(no return value)

1. To get user input for Text Height and Name of File containing the text to be drawn out

float getTextHeight(float\*textHeight, char\*textFileName);

parameters:

* textHeight (output) – the pointer to store the height input which ranges between 4mm and 10mm.
* textFileName(output) – A pointer to hold the file that contains the text provided by the user.
* Return Value – void (no return value)

1. To scale font data which is determined by the user input for the Height

Void scaleFont(FontData\*fontData, floattextHeight);

Parameters:

* fontData (input) – The pointer is to be scaled by the font data
* textHeight (input) – Used to store the previous height input that ranges between 4mm and 10mm from the user.
* Return Value – void (no return value)

1. To calculate Scale Factor

float scaleHeight(float baseHeight, float scaleFactor);

parameters:

* baseHeight – the initial height of the letter (input).
* scaleFactor – the factor by which the height should be scaled (input).
* Return value – returns the new scaled height as a float.

1. Generating G-Code for Each Character

void generateGcodeCharacter(char character, float xPosition, float yPosition, float scaleFactor, char “gCodeCommands);

parameter:

* Character - The character to be converted into G-Code (input).
* xPosition - The horizontal position offset to start drawing the character from the current position (input).
* yPosition - The vertical position offset to start drawing the character (input).
* scaleFactor - The scaling factor applied to adjust the size of the character (input).
* gCodeCommand – pointer to send generated G-Code commands
* Return Value – void (no return value)

1. To Generate G-Code for Text

void generateGCodeText(const char\* text, float xPosition, float yPosition, float lineSpacing, float scaleFactor, char “gCodeCommands);

parameter:

* Text - The string of text to be converted into G-Code (input).
* xPosition - The horizontal starting position for the text (input).
* yPosition - The vertical starting position for the text (input).
* linespacing - The vertical space between consecutive lines of text (input).
* scaleFactor - The factor by which each character is scaled (input).
* gCodeCommand – pointer to send generated G-Code commands
* Return value – void (no return value)

1. Sending the G-Code to the Arduino

void sendGCode(int numCommands, const char “gCodeCommands);

parameters:

* numCommands - The number of G-code commands to be sent.
* gCodeCommand – pointer to send generated G-Code commands
* Return value – void (no return value)

1. To control the Pen State

void penControl(bool penState, char “gCodeCommands);

parameter:

* penstate (input) – To store the values of a boolean parameter (true or false) indicating the desired state of the pen
* true - Lower the pen (pen down)
* false - Raise the pen (pen up)
* gCodeCommand – pointer to send generated G-Code commands
* Return value – void (no return value)

1. Reset to Origin

Void reset resetToOrigin(int “xPosition,int “yPosition, int “commandindex, char “gCodeCommands);

Parameters:

* xPosition (input/output)- The X-coordinate which the activation for the positions of each ASCII character
* yPosition(input/output) - The Y-coordinate which the activation for the positions of each ASCII character
* commandindex(input/output) – To keep track the current positions inside gCodeCommands
* gCodeCommands(output) - pointer to send generated G-Code commands

1. Recieive Acknowledgement from Arduino

Int recieveAck ();

Parameters:

* None
* Return Value – integer for which 0 for false and 1 for true

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| int loadFontData(char\* filename, FontData\* fontStruct) | Test loading a valid font file | SingleStrokeFont.txt | Font data is populated with each ASCII characters movement instructions returns successfully if file is valid |
| getTextHeight | Test if the user input for text height and file name is captured | textHeight = pointer; textFileName = pointer | The function should prompt the user for a height between 4mm and 10mm and a text file name, storing them in the output pointers. No return value. |
| scaleFont | Test if font data is scaled correctly based on height | fontData = pointer to FontData; textHeight = 8.0 | Font data should be scaled according to the height of 8.0mm. No return value. |
| scaleHeight | Test if the scale factor is correctly applied to the base height | baseHeight = 5.0; scaleFactor = 1.5 | Returns scaled height of 7.5 (5.0 \* 1.5). |
| generateGcodeCharacter | Test G-code generation for a single character | character = 'A'; xPosition = 0; yPosition = 0; scaleFactor = 1.0; gCodeCommands = pointer | Generates G-code commands for the character 'A' starting at position (0,0) with a scale factor of 1.0. |
| generateGCodeText | Test G-code generation for a string of text | text = "HELLO"; xPosition = 0; yPosition = 0; lineSpacing = 1.0; scaleFactor = 1.0; gCodeCommands = pointer | Generates G-code for the text "HELLO" with given positions, line spacing, and scale factor. |
| sendGCode | Test if G-code commands are sent to the Arduino | numCommands = 5; gCodeCommands = pointer to 5 commands | Sends the 5 G-code commands to Arduino. No return value. |
| penControl | Test controlling the pen state | penState = true; gCodeCommands = pointer | Lowers the pen (pen down) and stores the G-code command. No return value |
| resetToOrigin | Test resetting the drawing position to the origin | xPosition = 10; yPosition = 10; commandIndex = 2; gCodeCommands = pointer | Resets the position of the pen back to the origin (0,0) and updates the gCodeCommands. No return value |
| recieveAck | Test receiving acknowledgment from Arduino | None | Returns 1 for True received or 0 for no False. (Test case should simulate Arduino acknowledgment). |

*Extend table as required. Note that ‘Function’ includes main()*

# Flowchart(s)

May be included as separate pdf