# Name & Student ID

JOEL BRADLEE JOHNY

# Title

Software Project Final Submission

# Software Description

The software programmed is a controller for a robotic arm designed to draw characters from a text file using single stroke fonts obtained from a font file. The software processes text in text files inputted into the system by a user prompt. It then generates G-code instructions that are transmitted to the robot via the rs232 interface. The program is modular and robust by being able to handle input validation, scaling and alignment to produce an accurate rendering of the text within the specified constraints.

The operation begins by initializing the robot and establishing communication. Then, font data is loaded from the SingleStrokeFont.txt file which specifies the stroke information for each ASCII character to be drawn. The user then needs to provide a keyboard input for which text file to be read. Afterwards, the user needs to input a desired height for the text between 4-10 mm. The software then scales each character proportionally to the specified height and manages the line wrapping to fit within the maximum text width of 100 mm.

Key functionalities to the software include converting character strokes into G-code commands, managing the pen’s state, either it be down or up and ensuring proper positioning for continuous drawing. Spaces and new line for characters are handled adequately enabling the rendering of multi-line text. Configuration files such as serial.h allow customization of communication settings such as the COM port and baud rate.

Handling error is integrated to address issues like invalid input files or unsupported characters to ensure the system runs reliably. The programmed software concludes the operations by returning the pen to the origin in a pen up state before closing the communication port.

# Project Files

|  |  |
| --- | --- |
| **File Name** | **Description** |
| main.c | Contains the main program logic and initialization of the robot. |
| rs232.c | Manages rs232 communication with the robot. |
| rs232.h | Header file that provides declarations for rs232 communication with the robot. |
| serial.c | Manages serial communication modes for emulator and robot testing. |
| serial.h | Header file for additional communication configurations between emulator and robot modes. |
| SingleStrokeFont.txt | Font data file defining strokes for each character. |
| test.txt | Text file example for running the system |

# Key Data Items

|  |  |  |
| --- | --- | --- |
| Name | Data type | Rationale |
| fontData | Struct CharacterData[] | Stores the font definitions for all characters in the font data file. |
| textHeight | Int | Defines the height of text desired to scale the characters to. |
| current\_x  current\_y | Int | Tracks the pen’s position on the drawing surface. |
| prev\_pen\_state | Int | Tracks the pen’s previous state whether it be pen up or pen down |
| commandBuffer | Char[] | Holds G-code commands to send to the robot. |

# Functions

***Reads font definitions from a desired font file.***

*int loadFontData(FILE \*fontFile, struct CharacterData fontData[MAX\_CHARACTERS])*

*Parameters:*

*\*fontFile – Points to font data file*

*CharacterData fontData – Provides an array to store font data obtained from font file*

*Return value – returns 0 if successful, 1 if failed*

***Processes the input text and generates G-code for each character in the text file to be printed according to the specifications required.***

*void processTextandCalculateWidth(FILE \*textFile, struct CharacterData fontData[MAX\_CHARACTERS], int textHeight, int \*current\_x, int \*current\_y, int \*prev\_pen\_state)*

*Parameters:*

*\*textFile – Points to inputted text file*

*CharacterData fontData – Provides an array to store font data*

*textHeight – User iputted text height*

*\*current\_x – Points to the current x-coordinate*

*\*current\_y – Points to the current y-coordinate*

*\*prev\_pen\_state – Points to the pen’s previous state*

*Return value – returns 0 if successful, 1 if failed*

***Generates the G-code from the processTextandCalculateWidth function and sends the commands to the robot.***

*void generateAndSendGCode(int x, int y, int pen\_down, int \*previous\_pen\_state)*

*Parameters:*

*x – x-coordinate of the stroke to draw*

*y – y-coordinate of the stroke to draw*

*pen\_down – Pen state ( 1 : pen-down, 2 : pen-up )*

*\*previous\_pen\_state – Points to the previous pen state*

*Return value – None*

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| main() | All functions run successfully | Correct input data for all functions | Outputs G-code commands accordingly |
| Any function fails | Invalid text file name input | Prints ”Error opening text file.” and terminates program |
| Any function fails | Invalid text height input | Prints "Error: Text height must be between 4 and 10 mm.” and terminates program |
| loadFontData | Valid .txt file | SingleStrokeFont.txt | Font data is loaded correctly |
| Invalid .txt file | Empty file | Function shows error and terminates the program |
| processTextandCalculateWidth | Valid maximum height input | textHeight = 10 | Characters are scaled to textHeight and offset to meet the 100 mm boundary |
| Valid minimum height input | textHeight = 4 | Characters are scaled to textHeight and offset to meet the 100 mm boundary |
| Invalid maximum height input | textHeight = 11 | Prints "Error: Text height must be between 4 and 10 mm.” and terminates program |
| Invalid minimum height input | textHeight = 3 | Prints "Error: Text height must be between 4 and 10 mm.” and terminates program |
| generateAndSendCode | Valid G-code | G-code in correct format | G-code is generated and sent to robot |
| Invalid G-code | G-code in incorrect format | G-code is not generated and sent to robot |