# Name & Student ID

Lee Yue Shing 20517270

# Title

Developing a code which reads from a file and control the robot to draw out the text as read

# Software Description

This project involves developing software that enables a robot to "draw" text by interpreting character definitions from a specified font file. Communication between the software and the Arduino-controlled writing robot will occur via a virtual Rs232 serial port using G-Code. These commands will control the robot’s arm, allowing it to raise and lower the pen, as well as move to specified X, Y coordinates to produce the desired text.

The program must also ensure that the total width of the drawn text does not exceed 100mm. The code will read the entire font file and store it in memory using an appropriate data format. The characters can be scaled to a user-defined height between 4mm and 10mm. Words will be processed sequentially, with each word sent to the robot for output before the next word is read from the file. When writing, the software will calculate whether there is sufficient space remaining on the current line for the next word. If there is enough space, the word will be written; otherwise, the program will begin a new line. Additionally, lines of text will be spaced 5mm apart for consistent formatting.

# Project Files

**rs232.c**

This file provides an implementation for handling RS-232 serial communication on Linux/FreeBSD and Windows systems, including functions to configure, open, read from, write to, and manage serial ports. It supports various baud rates, data bits, parity, and stop bits configurations, ensuring compatibility across platforms.

**rs232.h**

This file provides a cross-platform RS-232 serial communication library with functions for opening, closing, reading, writing, and controlling serial ports, compatible with Linux/FreeBSD and Windows systems. It is also used to manage serial communication efficiently.

**serial.c**

Manages serial communication between the robot and the controlling computer. It includes functions for opening, closing, and interacting with a COM port, sending commands, and waiting for specific responses (like a $ or ok) from the robot to ensure proper synchronization.

*Parameters to be changed:*

-When for initial testing with VS code, make ‘#define Serial\_Mode’ as a comment by adding // in front.

-When connected to robot or arduino, make ‘#define Serial\_Mode’ as NOT a comment

**serial.h**

Defines basic functions and constants for serial communication, including port configuration, data handling, and checks for port availability. It supports tasks like waiting for specific replies or signals and managing the RS-232 port state in your project.

*Parameters to be changed:*

-Ensure number in ‘cport\_nr’ is equal to current port number -1

**SingleStrokeFont.txt**

Contains all the data of the strokes of each character for robot to write.

**text.txt**

Contain words to be read by the code and written out by the robot.

*Parameters to be changed:*

-Can be changed to different words for robot to write

# Key Data Items

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** | **Rationale** |
| filename | const char\* | Chosen because filenames are typically represented as strings and should not be modified within the function. |
| inputFile | FILE\* | A pointer type is necessary for reading data from files using the C standard library. |
| dataArray | struct DataRow\* | Used to manage an array of structures efficiently, allowing dynamic allocation and structured data storage. |
| numRows | int | An integer is appropriate for counting the number of rows as it ensures precision and simplicity. |
| col1, col2, col3 | int | Integer types are sufficient for representing coordinate data and discrete states (e.g., 1/0 for pen state). |
| asciiValue | int | Chosen because ASCII values are always integer representations of characters. |
| adjustedCol1 | double | Chosen to handle fractional values resulting from scaling and offset calculations for the x-coordinate. |
| adjustedCol2 | double | Chosen to handle fractional values resulting from scaling and offset calculations for the y-coordinate. |
| sValue | const char\* | String type is suitable for holding static, predefined values like speed settings. |
| gValue | const char\* | String type is suitable for holding static, predefined values like mode settings. |
| asciiFile | FILE\* | A pointer type is necessary to manage file operations and interact with the C file-handling library. |
| word | char[] | A character array is used because words are strings of text, and arrays are efficient for sequential storage. |
| wordWidth | int | Integer type is sufficient for calculating and storing discrete word widths. |
| height | int | Represents a discrete, whole-number multiplier, and integers are efficient for such use cases. |
| xOffset | double | A floating-point type allows for precise positioning along the x-axis. |
| yOffset | double | A floating-point type allows for precise positioning along the y-axis. |
| LINE\_WIDTH | int | An integer is suitable for a constant value representing a fixed limit. |
| ASCII\_MARKER | int | An integer is appropriate for representing a marker or flag used in comparisons. |
| i, j | int | Integers are ideal for loop counters due to their efficiency and precision. |

# Functions

1. **int getHeightFromUser()**

This function prompts the user to enter a height value between 4 and 10. It ensures the input is within the specified range.

**Parameters:**  
• None

**Return value:**  
• Returns an integer value between 4 and 10 (inclusive) representing the height.

1. **void readFile(const char \*filename, struct DataRow \*dataArray, int numRows)**

This function reads the font data from the specified file and loads it into the dataArray.

**Parameters:**  
• filename – The name of the file containing the font data.  
• dataArray – A pointer to an array of DataRow structures to store the read data.  
• numRows – The number of rows to be read from the file.

**Return value:**  
• This function does not return any value.

1. **void findAsciiData(struct DataRow \*dataArray, int numRows, int asciiValue, int height, double xOffset, double yOffset)**

This function searches for the ASCII value in the dataArray and sends the corresponding data to the robot with adjusted coordinates based on the provided height and offsets.

**Parameters:**  
• dataArray – A pointer to the array of DataRow structures containing font data.  
• numRows – The number of rows in the font data array.  
• asciiValue – The ASCII value of the character to be processed.  
• height – The height multiplier to adjust the character size.  
• xOffset – The current X-coordinate offset for the robot.  
• yOffset – The current Y-coordinate offset for the robot.

**Return value:**  
• This function does not return any value.

1. **FILE\* openTextFile(const char \*filename)**

This function opens a text file for reading and returns the file pointer.

**Parameters:**  
• filename – The name of the file to be opened.

**Return value:**  
• Returns a pointer to the opened file (FILE\*), or exits the program with an error message if the file cannot be opened.

1. **int readNextWordFromFile(FILE \*asciiFile, char \*word)**

This function reads the next word from the specified text file and stores it in the word parameter.

**Parameters:**  
• asciiFile – A pointer to the file from which the word will be read.  
• word – A character array to store the next word from the file.

**Return value:**  
• Returns 1 if a word is successfully read, and 0 if no more words are available.

1. **void processWord(struct DataRow \*dataArray, int numRows, char \*word, int height, double \*xOffset, double yOffset)**

This function processes each character of the provided word, finds its ASCII data, and sends the corresponding drawing instructions to the robot, adjusting the X-offset.

**Parameters:**  
• dataArray – A pointer to the array of DataRow structures containing font data.  
• numRows – The number of rows in the font data array.  
• word – The word to be processed.  
• height – The height multiplier to adjust the character size.  
• xOffset – A pointer to the current X-coordinate offset, which is updated during processing.  
• yOffset – The current Y-coordinate offset for the robot.

**Return value:**  
• This function does not return any value.

1. **int calculateWordWidth(const char \*word, int height)**

This function calculates the width of a word based on the number of characters and the provided height.

**Parameters:**  
• word – The word for which the width needs to be calculated.  
• height – The height multiplier to adjust the word's width.

**Return value:**  
• Returns the calculated width of the word in units adjusted by the height.

1. **void processWordsFromFile(FILE \*asciiFile, struct DataRow \*dataArray, int numRows, int height, double \*xOffset, double \*yOffset)**

This function reads words from the specified file and processes them one by one, adjusting the X and Y offsets as needed for line breaks.

**Parameters:**  
• asciiFile – A pointer to the file containing the text to be processed.  
• dataArray – A pointer to the array of DataRow structures containing font data.  
• numRows – The number of rows in the font data array.  
• height – The height multiplier to adjust character size.  
• xOffset – A pointer to the current X-coordinate offset, which is updated during processing.  
• yOffset – A pointer to the current Y-coordinate offset, which is updated during processing.

**Return value:**  
• This function does not return any value.

1. **struct DataRow\* allocateMemoryForDataArray(int numRows)**

This function allocates memory for an array of DataRow structures based on the specified number of rows.

**Parameters:**  
• numRows – The number of rows of DataRow structures to allocate memory for.

**Return value:**  
• Returns a pointer to the allocated memory (a DataRow array).  
• If memory allocation fails, the program exits with an error message.

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
| readFile("SingleStrokeFont.txt", dataArray, NUM\_ROWS) | Valid file with correct name | ‘SingleStrokeFont.txt’ | Data array is populated correctly with file data. |
| Wrong file name | ‘SingleeStrokeeFontt’ | Print ‘’Error opening file’’ and exit |
| openTextFile | Valid file and file name | Filename: "text.txt" | Returns valid FILE\* pointer to the file. |
| Invalid file /File does not exist | Filename: "MissingText.txt" | Print ‘’Error opening text file’’ |
| readNextWordFromFile | File contains valid words | File with words: "Hello World" | Returns each word sequentially: "Hello", "World". |
| Empty file | Empty file | Returns 0, indicating end of file. |
| calculateWordWidth | Normal valid word | Word: ‘Hello’ | Returns ‘5’, which is correct number of characters |
| processWord | Word contains supported characters | Text: ‘Hi12345’ | Sends correct commands to draw the text. |
| Word contains unsupported characters | Text: ‘@’ | No code is sent to regarding this character and skip to next character |
| processWordsFromFile | File contains multiple valid words | File: ‘This is a very long sentence’ | Move to next line when reach width limit |
| findAsciiData | ASCII value is in the dataset | ASCII value: 65 | Senc correct drawing commands for ASCII value 65. |
| ASCII value is not in the dataset | ASCII value: 130 | No commands sent to robot |
| main | |  | | --- | |  |  |  | | --- | | Test full program operation including initialization, file handling, and drawing words. | | text.txt containing words, SingleStrokeFont.txt containing font data, valid COM port, height = 5. | Successful program execution, correct robot drawing based on text file, and proper resource cleanup. |

# Flowchart(s)

A diagram of a flowchart

Description automatically generated