Daanish Salmaan 20526141

Drawing Robot Software

Software Description

This software provides automated control of a two-axis writing robot through serial communication using standard G-code instructions. The program interprets text from an external file and translates it into coordinated pen movements using a predefined single-strike font.

Text processing is performed on a word-by-word basis to maintain readability and prevents words form being divided between lines. The available drawing width is monitored continuously and when insufficient space remains, the program moves the writing position to the next line. The user specifies the desired text height at runtime and this value is used to scale the font proportionally.

For each character, the software issues appropriate pen control commands, followed by motion commands that reproduce the required strokes on the drawing surface. Before writing begins, the robot is initialised into a ready state. Once all the text has been rendered, the pen is raised and the robot is returned to its home position.

Project Files

Main.c

- Contains the complete program logic required to control the writing robot
- Responsible for loading the font file, reading and processing the input text file, calculating scale of the characters based on the user-inputted height and generating G-code
- After execution begins, the only user input required is the height data
- Key page layout parameters, including the horizontal and vertical starting positions, maximum width, line spacing and maximum word length are defined at the top of the file

SingleStrokeFont.txt

- Defines the stroke-based font used by the program
- Each printable ASCII character is represented as a sequence of pen movements and a corresponding character width
- This file allows characters to be drawn using single continuous strokes where possible, improving drawing efficiency

Test.txt

- Contains the text message to be written by the robot or displayed in the emulator
- The contents of this file can be modified without changing the program

Serial.c / Serial.h

- Implements higher-level serial communication functionality between the PC and robot
- Handles the robots ‘wake up’ protovol, transmission od G-code commands and response handling
- Supports both emulator mode and robot mode
- Mode selection is controlled by commenting or uncommenting the #define Serial_Mode line

Rs232.c / Rs232.h

- Provides low level serial port communication routines for the windows operating system
- Manages port opening, configuration, data transmission and reception

- Used internally by the higher-level serial communication module

Key Data Items

Name	Data type	Rationale
Stroke	struct	Represents a single drawing movement from the font file. Each stroke stores an X offset, Y offset, and pen state, allowing clear separation of drawing and travel moves.
FontChar	Struct	Stores all stroke information for a single ASCII character, including the number of strokes, character width, and whether the character is defined in the font.
FontData[256]	Array	Lookup table for all extended ASCII characters. Using a fixed-size array provides fast, direct access when rendering characters.
TextLayoutState Layout	Struct	Maintains the current text layout state, including cursor position, scaling factor, line width usage, maximum allowed width, and line spacing. This replaces multiple BA global variables while keeping the same behaviour.
WordBuffer[MAX_WORD_LENGTH]	Char	Temporary buffer used to store each word read from the input text file. Prevents buffer overflow by enforcing a maximum word length.
textHeight	Int	Stores the user-selected text height (4–10 mm). An integer is sufficient because only discrete values are accepted.
scaleFactor	Float	Calculated from the selected text height divided by the font's base height (18 units). A floating-point type ensures accurate scaling.
cursorX, cursorY	Float	Track the current drawing position on the page. Floating-point values are required to support scaled coordinates.
currentLineWidth	Float	Tracks how much of the current line has already been used. Required for word-wrapping
maxLineWidth	Float	Defines the maximum drawable width before wrapping to a new line. Allows flexibility for different page sizes.
lineSpacing	Float	Controls the vertical distance between successive lines of text. Set dynamically based on the chosen text height.
buffer[100]	Char	Temporary buffer for formatting and transmitting individual G-Code commands.
FILE *fp	File	File pointer used for reading the font file and text file. Enables standard file I/O operations.

lastPen	int	Tracks the previous pen state (up/down) so pen commands are only sent when the state changes, reducing unnecessary G-Code.

Functions

*int LoadFontData(const char *filename, FontChar fontData[256])*

Parameters:

> *filename — input: path to font file (e.g., "SingleStrokeFont.txt")*
>
> *fontData — output: populated lookup table of all characters*

Return value – returns 1 if font loaded successfully, 0 if failed

float CalculateScaleFactor(int textHeightMm, int baseUnits)

Parameters:

> *textHeightMm — input: chosen height in mm (4–10)*
>
> *baseUnits — input: font unit height (18)*

Return value – scaling multiplier (mm per font unit)

int GetTextHeight(void)

Parameters:

> *none*

Return value – valid height (4–10) as an int
(Program exits if invalid input is given)

*int ReadTextFile(char *buffer, const char *filename, int maxLen)*

Parameters:

> *buffer — receives entire file contents*
>
> *filename — path to text file*
>
> *maxLen — max bytes to read*

Return value – number of characters read (0 if file open fails)

void InitialiseTextPosition(void)

Parameters:

> *none*

Return value – none

Operation:

> *sets Layout.cursorX = LEFT_MARGIN_MM*
>
> *sets Layout.cursorY = TOP_LINE_Y_MM*
>
> *sets Layout.currentLineWidth = 0*

void AdvanceToNextLine(void)

Parameters:

none

Return value – none

Operation:

Resets cursor to left margin

decreases Layout.cursorY by Layout.lineSpacing

resets Layout.currentLineWidth

*int GetNextWord(FILE *textFile, char *buffer, int maxLen)*

Parameters:

textFile — opened text file pointer

buffer — receives next word

maxLen — buffer size limit

Return value – 1 if a word is read, otherwise 0

*float CalculateWordWidth(const char *word)*

Parameters:

Word string

Return value – width of the word in mm (scaled)

*void RenderWord(const char *word)*

Parameters:

Word string

Return value – none

Operation:

calls RenderCharacter() for each defined character

advances Layout.cursorX and Layout.currentLineWidth by each character width + a trailing space

void RenderCharacter(char c)

Parameters:

c — character to draw

Return value – none

Operation:

looks up stroke list in FontData[(unsigned char)c]

scales + offsets each stroke using Layout.scaleFactor, Layout.cursorX, Layout.cursorY

outputs pen changes via S0 / S1000

outputs motion via G0 for pen up and G1 for pen down

*void GenerateGCode(const char *text)*

Parameters:

text — full text buffer

Return value – none

Operation:

parses buffer into words, checks wrap against Layout.maxLineWidth

calls AdvanceToNextLine() when a word does not fit

calls RenderWord() to emit commands

*int GenerateTextGCodeFromFile(const char *filename)*

Parameters:

filename — text file path

Return value – 1 if processed correctly, 0 if failed

Operation:

reads words using GetNextWord()

wraps and draws using the same logic as GenerateGCode()

*void SendGCodeToRobot(const char *command)*

Parameters:

command — one complete G-code command string

Return value – none

Operation:

sends the line to the robot (PrintBuffer)

waits for robot acknowledgement (WaitForReply)

delays briefly (Sleep) to maintain stability

void MoveToOrigin(void)

Parameters:

none

Return value – none

Operation:

sends S0 then moves robot to (0,0) with G0

void DrawEndShape(void)

Parameters:

none

Return value – none

Operation:

Draws your end shape

Testing Information

Function	Test Case	Test Data	Expected Output
main()	Normal execution	Valid font file and text file	Program runs successfully, draws text, lifts pen, and returns robot to (0,0).
GetTextHeight()	Valid input	User enters 7	Function returns 7 and program continues normally.
GetTextHeight()	Invalid input	User enters 3	Error message displayed and program terminates safely.
CalculateScaleFactor()	Valid scale calculation	textHeight = 9, baseUnits = 18	Function returns 0.5.
LoadFontData()	Valid font file	"SingleStrokeFont.txt"	Returns 1; all defined characters loaded into FontData.
LoadFontData()	Missing font file	"MissingFont.txt"	Returns 0; error message displayed; program stops.
GetNextWord()	Normal word extraction	Text file contains "HELLO WORLD"	First call returns "HELLO", second call returns "WORLD".
CalculateWordWidth()	Simple word	Word = "HI"	Returns correct width in mm including trailing space.
AdvanceToNextLine()	Line wrap triggered	Word width exceeds remaining line space	Cursor resets to left margin and Y position decreases by line spacing.
GenerateTextGCodeFromFile()	Single character	Text file contains "A"	Character is drawn correctly at starting position.
GenerateTextGCodeFromFile()	Multiple words	"HELLO THERE"	Words are drawn sequentially with

			correct spacing and wrapping.
GenerateTextGCodeFromFile()	Word wrapping	Long word with small MAX_LINE_WIDTH_MM	Word is moved entirely to the next line before drawing.
RenderWord()	Normal word rendering	"Hi"	Calls RenderCharacter() twice and updates cursor position correctly.
RenderCharacter()	Character with multiple strokes	Character = 'E'	Correct sequence of pen-up and pen-down G-Code commands generated.
RenderCharacter()	Undefined character	ASCII value not in font	Character is skipped without crashing.
SendGCodeToRobot()	Single command	"G1 X10 Y5 F1000\n"	Command sent to robot; program waits for acknowledgement.
SendGCodeToRobot()	Multiple commands	Loop of G-Code commands	Each command is sent only after receiving a response, ensuring synchronisation.

AI Statement

I used AI tools to review my work and check whether it met the assignment requirements.
AI was used to identify if I had missed any important functions, data items, or logical flows that should be included in the program design.
The content and structure were written and verified by me, and the AI support was limited to reviewing and suggesting improvements to ensure completeness and clarity.

Flowchart(s)

Included as separate pdf