Thomas Oczadly - 20260180

Software project documentation

Software Description

The aim of the software project is to create a program that can read user inputted data and output G-Code, that represents the information, to an Arduino. The Arduino then transmits the G-Code to a writing arm.

Firstly, the information on how to draw the shapes needs to be read from the text file "ShapeStrokeData.txt". This file contains information on several different shapes. This information includes the shape name and then the drawing instructions of each shape (x and y coordinated and whether to have the pen up or down). The coordinates need to be scaled by 20/16 to fit the given grid size (30 by 30). The information should be stored so it can be accessed later when it needs applying to the user instructions and then for converting them into G-Code.

Next the drawing instructions should be read from a text file. The text file should be input by the user so that multiple sets of instructions can be run without having to change the code every time another file needs to be accessed. The drawing instructions include whether to draw a grid, and information regarding where and what shapes to draw in the grid. This information should also be stored for future access. If a grid needs to be drawn, then a function should be called that reads pre-programmed G-Code instructions for the robot to draw the grid.

Then the program needs to find the shapes from the drawing instructions in the shape information stored. The data should be manipulated so that the shape is drawn centred at the given grid position.

Finally, the information needs converting into G-Code which can be sent line by line to the Arduino.

Project Files

main.c

File containing the main program.

rs232.h

Header file for rs232.c

rs232.c

Used to send G-code information to the robot

serial.h

Header file for serial.c

COM port number needs to be set here

serial.c

Used for setting up RS232 port

Used for setting program in serial mode or emulator mode

Uncomment (//#define Serial_Mode) for running program with the robot

ShapeStrokeData.txt

Text file containing the shape information. Contains the number of shapes in file, the shape, number of lines of instructions for that shape, and coordinate and pen status information. Custom shape (cube) included in this file.

DrawShapes.txt

Text file containing the instructions. Contains information whether to draw the grid, the shapes to be drawn and the grid position to draw them in.

Functions

Function to read and store shape information in ShapeStrokeData.txt. Called from main function.

*int ReadShapeInformation(struct ShapeData *Shape, FILE *fptrShapeInstructions, int *Numshapes);*

Parameters:

Shape – pointer to structure to return output variables into structure

fptrShapeInstructions – input pointer to access ShapeStrokeData file

Numshapes – pointer to return number of shapes in text file "ShapeStrokeData.txt"

Return value – returns 0 if successful, 1 if failed

Function to read and store instruction information from user inputted instruction file. Called from main function.

*int ReadInstructions(struct InstructionData *Instructions, int *NumInstructions);*

Parameters:

Instructions – pointer to structure to return output variables into structure

NumInstructions – pointer to return number of lines of instructions in text file

Return value – returns 0 if successful, 1 if failed

Function to convert instructions into G-code which is then sent to the robot. Called from main function after files have been read, COM port has been opened and initial commands have been sent.

*int ConverttoGCode(char *buffer, struct InstructionData *Instructions, struct ShapeData *Shape, int *Numshapes, int *NumInstructions);*

Parameters:

buffer – pointer to return output command

Instructions – pointer to structure to input variables into function from structure

Shape – pointer to structure to input variables into function from structure

Numshapes – pointer to input number of shapes in text file "ShapeStrokeData.txt"

NumInstructions – pointer to input number of lines of instructions in text file

Return value – returns 0 if successful, 1 if failed

Function to calculate pen positions for a shape and then send as G-code to the robot. Called from ConverttoGCode function after the shape from the instruction file has been found in the stored shapes from ShapeStrokeData.txt.

*int DrawShape(char *buffer, struct InstructionData *Instructions, struct ShapeData *Shape, int i, int j);*

Parameters:

buffer – pointer to return output command

Instructions – pointer to structure to input variables into function from structure

Shape – pointer to structure to input variables into function from structure

i – input corresponding to instruction number in for loop

j - input corresponding to shape number in for loop

Return value – returns 0 if successful, 1 if failed

Function to draw the grid. Sends as G-code to the robot. Called from ConverttoGCode if told to draw grid in instruction file.

*int CreateGrid(char *buffer);*

Parameters:

buffer – pointer to return output command

Return value – returns 0 if successful, 1 if failed

Key Data Items

Name	Data type	Rationale
ShapeData	struct	Define structure to store variables. This is the structure tag.
*Shape	struct ShapeData *	Define structure to store variables. Pointer acts as array of unknown size which will be allocated dynamically in the program.
ShapeName[30]	char ShapeData::	Series of characters representing string with length 30. Stored in Shape struct.
NumberLinesOfShape	int ShapeData::	Must be whole number as it represents the number of lines of coordinate information for shape. Stored in Shape struct.
PenStrokeData	struct	Define sub structure to store variables. Stored within *Shape structure. This is the structure tag.
*ShapePositionData	struct PenStrokeData * ShapeData::	Define structure to store variables. Pointer acts as array of unknown size which will be allocated dynamically in the program. Stored in Shape struct as a sub structure.
xPosition	float PenStrokeData::	Can be any number. Float as it needs to be scaled and may not stay a whole number. Stored in ShapePositionData struct.
yPosition	float PenStrokeData::	Can be any number. Float as it needs to be scaled and may not stay a whole number. Stored in ShapePositionData struct.
PenStatus	int PenStrokeData::	Int as it is either 1 or 0. Stored in ShapePositionData struct.
InstructionData	struct	Define structure to store variables. This is the structure tag.
Instructions	struct InstructionData	Define structure to store variables.
DrawGridValue	int InstructionData::	Whole number (1 or 0). Values represent whether to draw the grid. Stored in Instructions struct.
GridData	struct	Define sub structure to store variables. Stored within Instructions structure. This is the structure tag.
*ShapestoDraw	struct GridData * InstructionData::	Define structure to store variables. Pointer acts as array of unknown size which will be allocated dynamically in the program. Stored in Instructions struct as a sub structure.
ShapeGridPosition_x	Int GridData::	Represents whole number (1-3). Stored in ShapestoDraw struct.
ShapeGridPosition_y	int GridData::	Represents whole number (1-3). Stored in ShapestoDraw struct.
InstructionsShapeName[30]	char GridData::	Series of characters representing string with length 30. Stored in ShapestoDraw struct.
Numshapes	int	Must be whole number as it represents the number of shapes in text file.
NumInstructions	int	Must be whole number as it represents the number of lines of instructions in the text file.
fptrShapeData	FILE *	Pointer to shape data file (ShapeStrokeData.txt)

FileName[20]	char	Represents string with maximum of 20 characters. User inputted file name.
i	int	Used in for loops
j	int	Used in for loops
k	int	Used in for loops
buffer[100]	char	Represents string with maximum of 100 characters.
*fptr	FILE *	Pointer to shape instruction file inputted by user
NewLineIdentifier	char	Used for calculating number of lines in file. Represents character “\n”.
LineCount	int	Integer as it counts in whole numbers. Represents number of lines of instructions in the file.
StartPos_x	float	Can be any number (not just whole). Represents the starting x coordinate of the grid where the shape is to be drawn.
StartPos_y	float	Can be any number (not just whole). Represents the starting y coordinate of the grid where the shape is to be drawn.
TempPos_x	float	Can be any number (not just whole). Represents the x position the robot will move to when drawing the shape.
TempPos_y	float	Can be any number (not just whole). Represents the y position the robot will move to when drawing the shape.

Testing Information

Function	Test Case	Test Data	Expected Output
main	Opening file named “ShapeStrokeData.txt”	(No file named “ShapeStrokeData.txt” exists)	“Error opening file 'ShapeStrokeData.txt'”. Return 1.
main	Opening file named “ShapeStrokeData.txt”	(File named “ShapeStrokeData.txt” exists)	Program runs normally. “Success opening file 'ShapeStrokeData.txt'”
main	Allocation of Shape struct	(Allocation fails)	"Failed to allocate space for Shape". Return 1.
ReadShapeInformation	Allocation of ShapePositionData sub struct	(Allocation fails)	"Failed to allocate space for ShapePositionData". Return 1.
ReadInstructions	Get file name from user	“00000000000000000000” (21 zeros)	“File name is too large”. Return 1.
ReadInstructions	Get file name from user	“DrawShapes”	“Success! Opening file called: DrawShapes.txt”.

ReadInstructions	Get file name from user	“NonExistingFile.txt”	“Error opening file called: NonExistingFile.txt”.
ReadInstructions	Allocation of ShapestoDraw sub struct	(Allocation fails)	"Failed to allocate space for ShapestoDraw"
ConverttoGCode	Shape in instructions file is not in ShapeStrokeData.txt	“Circle”	“Shape 'Circle' not found in 'ShapeStrokeData.txt”

Flowchart(s)

Flowcharts included in separate PDFs. See “SystemFlowchart_main_TO_20261080” for the main function.