Computer Engineering and Mechatronics Project: Developing software for a drawing robot

Part 1: Project Planning Assignment

Introduction

You will be developing the software required to transmit commands from a file, via a virtual RS232 serial port, to a drawing robot such that it is able to 'draw out' shapes as read from a file as per Figure 1.

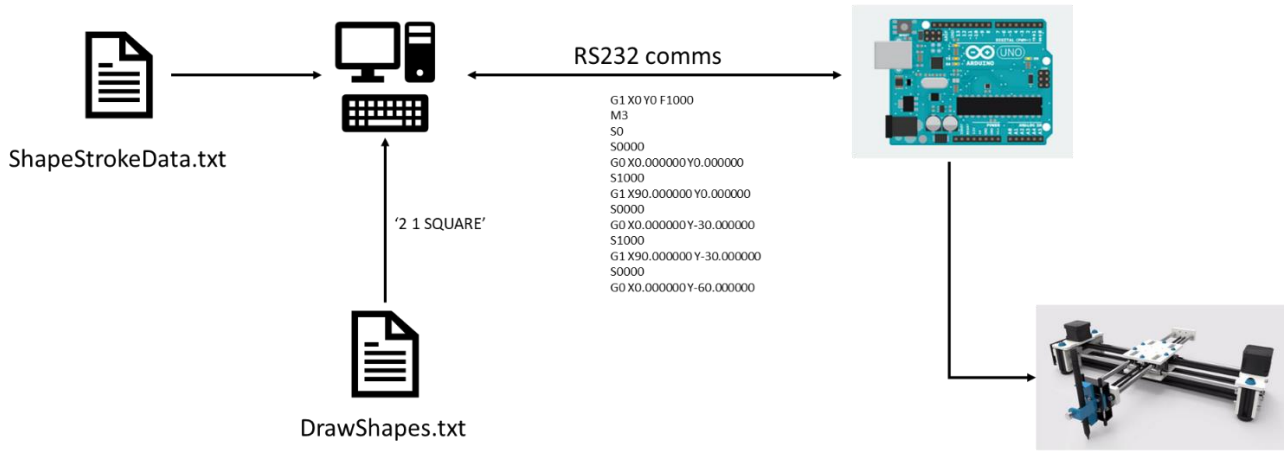

Figure 1: Overview of the application

The ultimate goal of the project is for the robot to draw a grid and various shapes within that grid. An example of possible output is shown in Figure 2.

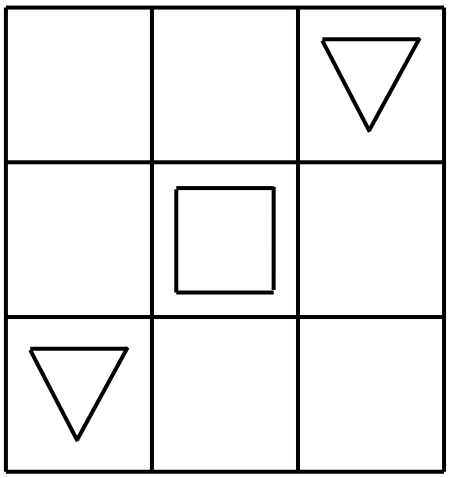

Figure 2: Sample output

The code written should meet the following specification:

- Read data from the 'ShapeStrokeData.txt' file (provided on Moodle). Details of the format are provided in Appendix 1.
    - The code should be written so that the entire shape data file is read and stored in memory in an appropriate data format.
    - Scale the x and y movements such that the height of a shape drawn is 20mm; it is 16 units in the shape file, so the movements derived from the shape data will need to be scaled within your program by the fraction 20/16 so that they are 20mm high when drawn.

- Read drawing instructions from an ASCII text file. An example file, 'DrawShapes.txt' is provided on Moodle. Details of the format are provided in Appendix 2.
    - The filename should be input by the user so that multiple drawing instructions files can be executed by the robot without recompiling the program.

- Generate and send G-Code commands (based on the drawing instruction file) to the Arduino to raise and lower the pen and to move the arm to specified X,Y locations. Details on the G-Code format is provided in Appendix 3. The Arduino has been pre-programmed to accept the G-codes and to transmit them to the writing arm.
    - The code should be written such that the drawing instructions are read from the file, processed, and sent to the Arduino one line at a time.
    - Generate commands to draw the grid if required. The dimensions for the grid are shown in Figure 3.

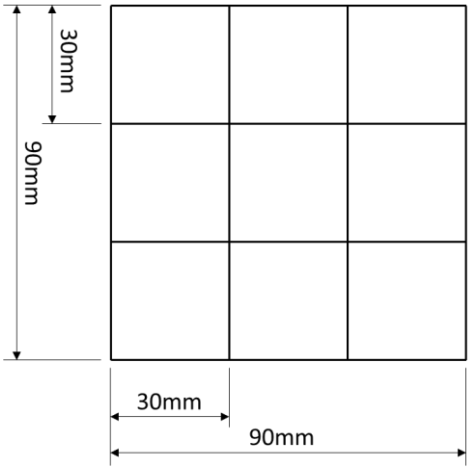

Figure 3: Grid configuration and dimensions

    - Generate commands to draw shapes at the specified grid positions. Each shape should be drawn centred in that grid position with a size of 20mm. (Note: there is no requirement for the code to be able to draw the shapes at other sizes). This is shown in Figure 4. Note that, from the pen starting position, y coordinates are negative.
- The pen should finish in the pen-up position.
- Multiple sets of shapes (from different drawing instruction files) should be able to be drawn in the same grid.

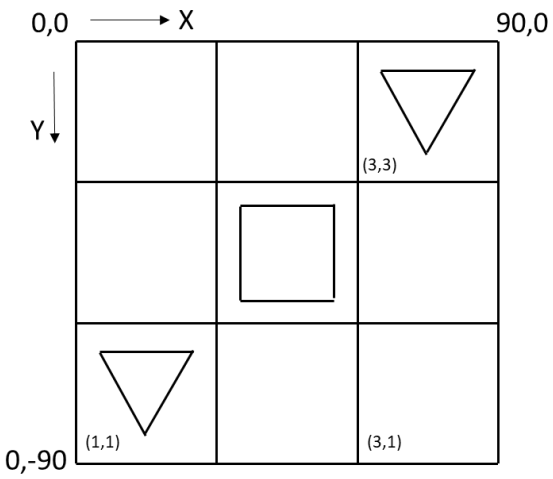

Figure 4: Drawing output showing grid coordinates and pen x,y coordinates

To assist you in developing your application you will be provided with code developed in VSCode to handle the serial communications and a sample project that shows how serial communication is initialised and then how data is sent and received (Appendix 3). A more complex worked example (Appendix 4), which will form the starting point for writing your program, illustrates the sending of G code to the Arduino and awaiting the "ok" acknowledgement that a new block of G code is expected.

Submission

The project planning assignment submission is a document which provides a specification for the software which will be developed in the final submission. Please note that you do **not** need to write any code for this submission.

Software Project Planning Submission (5%)

Learning outcomes:

- To be able to analyse a design brief to understand software requirements
- To produce a software design to fit a set of software requirements

Learning outcomes will be demonstrated in the software planning document submitted by:

- Giving an explanation of precisely what the program needs to do (explain it in plain English).

- Planning the program with the aid of flowcharts
  - Draw a flowchart with sufficient detail that it is possible to use it as a basis for writing the code for the program.
  - Include blocks to indicate function calls and flowcharts for the functions where appropriate

- Specifying the main data items required in the program.
  - Give data type and why this has been chosen.

- Defining the function declarations (prototypes) for all functions to be used in the program
  - Define all parameters and their types using meaningful names
  - Show whether parameters are input and/or output, whether they are changed and the return value, if any.

    - Includes functions which you will write yourself, not the ones which are supplied in the program template.

- Creating test data which will validate the program, confirming conformance of the program/function to its specification.
    - Ensure that all routes through the program are covered.
    - Include input data with expected outputs.

A template is provided for the submission on Moodle (ProjectPlanningTemplate22-23.docx). Save your planning document as **RobotPlanningXXX.pdf** where **XXX** is your initials. The flowchart(s) may be submitted in a separate pdf saved as **RobotFlowchartXXX.pdf** (do not submit a drawio file. Use the export function in draw.io to save to pdf).

The deadline for submission to Moodle of this document is **18:00 Tuesday 22nd November**.

References

Images: Robot arm, https://www.aliexpress.com/item/32792341105.html

Arduino board: https://www.arduino.cc/

RS232 library: https://www.teuniz.net/RS-232/index.html

The real robot: https://www.youtube.com/watch?v=OeswYL_EhH0

Appendix 1: Shape Strokes File Format

NOTE: This should be read and understood before considering the 'G-Code format'

The shape strokes file *ShapeStrokesData.txt* provided contains X,Y & pen up/down data required to draw out various shapes.

The general format for the file is shown in Figure 5, showing just one shape definition.

```
NumShapes N
SHAPE_NAME S
X Y P
X Y P
X Y P
```

Figure 5 – Template for shape information

Where:

- NumShapes: Static text (always 'NumShapes')
- N: The number of shapes defined in the file

- SHAPE_NAME: The string identifier for the shape name
- X: The X position to move to (relative to 0,0)
- Y: The X position to move to (relative to 0,0)
- P: Pen up/down (0=up so no line is draw, 1=down so causing a line to be drawn)

Some important things to note:

- ALL movements are **OFFSETS** relative to 0,0 (bottom left, the shape's local origin)
- The last 'X Y P' line moves the writing arm back to the local origin (0,0)

Example: Data for a single shape

If we consider the shape information for an inverted triangle as shown in Table 1.

Shape data	Details
INVERTED_TRIANGLE 5	Shape identifier is 'INVERTED_TRIANGLE' 5 lines of movement information
8 0 0	Move arm to position 8,0 with pen up
16 16 1	Move arm to position 16,16 with pen down (draws line)
0 16 1	Move arm to position 0,16 with pen down (draws line)
8 0 1	Move arm to position to 8,0 with pen down (draws line)
0 0 0	Move arm to position 0,0 with pen up (returns to the local origin)

Table 1: Line and pen drawing commands for inverted triangle

To draw a shape in the centre of the 30x30mm grid square it will be necessary to calculate an offset. This will be the combination of the offset to the bottom left point of the required grid square and the offset required in order to centre the 20x20mm (after scaling) shape.

Appendix 2: Drawing Instruction File Format

The drawing instructions file contains the instructions for what is to be drawn by the drawing robot.

The general format is shown in Figure 6.

```
DRAW_GRID N
X Y SHAPE_NAME
X Y SHAPE_NAME
```

Figure 6: Template for drawing instructions

Where:

- DRAW_GRID: Static text (always 'DRAW_GRID')
- N: Value 1 = draw the grid, 0 = do not draw grid
- X: The X grid coordinate
- Y: The Y grid coordinate
- SHAPE_NAME: String which identifies the shape to be drawn

Figure 7 shows the sample file 'DrawShapes.txt' (provided on Moodle) with corresponding output.

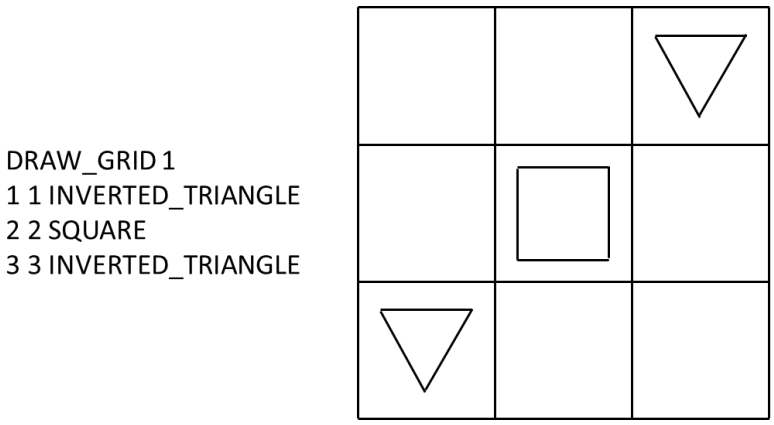

Figure 7: Sample drawing instruction file with corresponding output

Appendix 3: G-Code

G-Code is a well-established language originally developed for programming of numerical control (NC, now known as computer numerical control or CNC) machine tools – it is now widely used for control of 3D printers.

A file in G-code format is traditionally known as a "part program" (in that it is 'programming code' that is interpreted by software to control a system).

The subset of G codes and related to the project are shown in Table 3

Command	Description
F1000	feed rate (i.e. pen speed) 1000 mm min^{-1}
G0 X Y	Move to the position X,Y
G1 X Y	Draw a straight line from the last position to X,Y
M3	Turn on Spindle (needed for arm to work!)
S0	Pen up (original meaning is 'spindle speed 0')
S1000	Pen down (original meaning is 'spindle speed 1000 rev min^{-1}')

Table 3: Required G Codes

Generating G-Code from the shape information.

You will need to convert the shape data (as detailed in Appendix 1) to G-Code before it can be sent, via the communications channel, to the writing arm.

If we consider the case for sending a single inverted triangle shape starting at the origin (Table 1, Appendix 1) we would get the commands as shown in Table 4.

Points to note:

- Some commands are required to be sent before the 'drawing commands' to correctly initialise the arm
- The S1000/S0 commands need only be sent if the pen up/down state has changed

- In your application, the X & Y values shown in Table 4, would need to be scaled to obtain the correct shape size and offset to draw at the correct grid position.

Shape data	G-Codes to send	Details
Initialisation (send once)	F1000	Set pen speed
	M3	Turn on spindle
	S0	Pen up
8 0 0	G0 8 0	Go to 8,0
16 16 1	S1000 G1 X16 Y16	Pen down Move to 0,18 (line drawn as pen down)
0 16 1	G1 X12 Y0	Move to 12,0 (line drawn as pen down)
8 0 1	G1 X12 Y18	Move to 12,18 (line drawn as pen down)
0 0 0	S0 G0 X0 Y0	Pen up Move to 0,0

Table 4: Comparison of shape commands and G Code data

Appendix 4: Communication protocol

Each G-code command is sent via a virtual serial port at 115200 baud as a string of ASCII characters terminated with a "newline" character **\n**.

When the robot is ready to receive the next command it responds with the message "ok" followed by **\n**.

In practice the existing program on the robot stores a series of G-code commands in a "buffer" (a queue of memory) until the buffer is full.

When the buffer is full, no further "ok" message is sent until there is more space in the buffer. (This enables the robot to make a rapid series of movements without having to wait for each command to be sent individually).

Communication using a virtual serial port

True serial ports (using the RS-232 protocol including ±12 V signal levels) are rare on modern PCs, but communication with Arduinos and similar devices takes place via one or more virtual serial ports which follow similar protocols but at 5 V logic levels.

Use of serial communication in a C program is not straightforward so you will be using the RS-232 library written by Teunis van Beelen, available from https://www.teuniz.net/RS-232/index.html.

For convenience, his example files for transmitting and receiving text have been combined, modified and incorporated into a VSCode project, BlinkSerial which is available from the Moodle website. The folder also includes the RS-232 project web page and licence file, and a suitable test program for the Arduino. The combination of the two programs forms a rather unusual form of our old friend "Blink"!

You are encouraged to try out the program as follows:

1. Unzip the project to a suitable location.
2. Plug in your Arduino and make a note of the COM port number

3. Open BlinkSerialArduino.ino and compile and upload it.
4. Open the serial monitor, ensure that "newline" is selected and that the baud rate is set to 115200
5. You should be able to switch the LED on and off by typing "on" or "off" in the serial monitor.
6. Close the serial monitor or the next stage won't work!
7. Open BlinkSerialPC in VSCode
8. Set the COM port number in BlinkSerial.c as explained in the comment near the start of the main() function. Note that the number entered is the COM port number minus 1.
9. Compile and run BlinkSerialPC.c (linked with RS232.c). Check the COM port has been correctly set. If not, correct it and try again.
10. The Arduino should blink at 1 Hz, and the command window should display the commands sent to the Arduino and the acknowledgements received back.
11. To close the command window, press Ctrl-C.

Appendix 5: Example of sending G code to Arduino and awaiting acknowledgement

The zip file, RobotWriter4.0, includes the VSCode project which will form the starting point for your project. It also includes the RS-232 project web page and licence file, and a test program for the Arduino, SerialEchoBlink.ino. These are all available on Moodle.

The steps for getting this going are as follows:

1. Unzip the file RobotWriter4.0.zip file to a suitable location.
2. Plug in your Arduino and make a note of the COM port number
3. Open SerialEchoBlink.ino and compile and upload it.
4. Open the serial monitor, ensure that "newline" is selected and that the baud rate is set to 115200. The serial monitor should display a greeting terminated by a "$" sign e.g. "Test sketch to emulate writing robot $"
5. You should be able to switch the LED on and off by typing typing random text in the serial monitor and pressing "return" – each time the serial monitor should display "ok" and the LED should change state.
6. Close the serial monitor or the next stage won't work!
7. Open RobotWriter4.0_Skeleton in VSCode
8. Set the COM port number in serial.h as explained in the comment near the start of the program. Note that the number entered is the COM port number minus 1.
9. Build and run the project. Check the COM port has been correctly set. If not, correct it and try again.
10. The VSCode terminal window should show the "conversation" between the C program (running on the computer) and the sketch running on the Arduino, with a series of (hard-coded) G-code commands being sent to the Arduino and acknowledged with "ok".
11. To close the command window, simply press return at the end of the program. If for any reason the program gets stuck, press Ctrl-C.

Your job in the project is to replace the hard-coded G-code commands with your own programming in order to read the drawing instruction file and generate G-code to draw the grid and shapes accordingly on the robot. You should be able to use this program as the starting point for your own program, though you need to plan your work before you can do so!