Expertise
and insight
for the future

Arsi Arola, Duc Vo, Mikko Larke

XY-plotter documentation

Metropolia University of Applied Sciences

Smart systems

Project documentation

18 October 2020

Contents

1 Introduction

XY-plotter is device that can take any Scalable Vector graphic image (.svg) and either draw the picture in paper with actual pen or engrave picture on suitable medium (for example wood) with laser

1.1 Key features

- Logic for handling G-codes
- Use of real time operating system for handling thread timing
- Setting and using limit switches for work area handling
- Handling PWM for adjusting pen height and laser intensity
- Automatic calibration of work area
- RIT timers for stepping

2 User guide

2.1 List of needed hardware and software

- Makeblock XY-plotter
- LPC1549 LPCXpresso Board
- mDraw
- FreeRTOS
- Plotter emulator v1.4 (optional)

2.2 Hardware preparations

XY-plotter uses two axis rig for drawing or engraving given picture on physical medium. Rig works with stepper motor moving each axis and servo motor controlling pen height and laser intensity. To make sure that stepper motors won't run outside of working area device has limit switches on each end of axis guides.

2.3 Software preparations

Software used for reading picture data and sending G-codes is mDraw and it uses USB serial connection between computer and XY-plotters controlling unit to send and receive data. Note that communication to mDraw uses controlling units actual USB port, not de-bugging port.

For testing purposes instead of actual hardware device can use plotter emulator to mimic work of actual plotter hardware. Code in its current form is designed to run with emulator, and some of the GPIO pins used may need adjusting before running with real hardware.

2.4 Instructions

1. Connect computer with mDraw and plotter to controlling unit before switching device on.

2. After booting plotter waits for all limit switches to be cleared and then calibrates working area dimensions.

3. After calibration mDraw USB serial connection can be connected. Set mDraw dimensions to match working area.

4. Open .svg format image, scale and position image on canvas

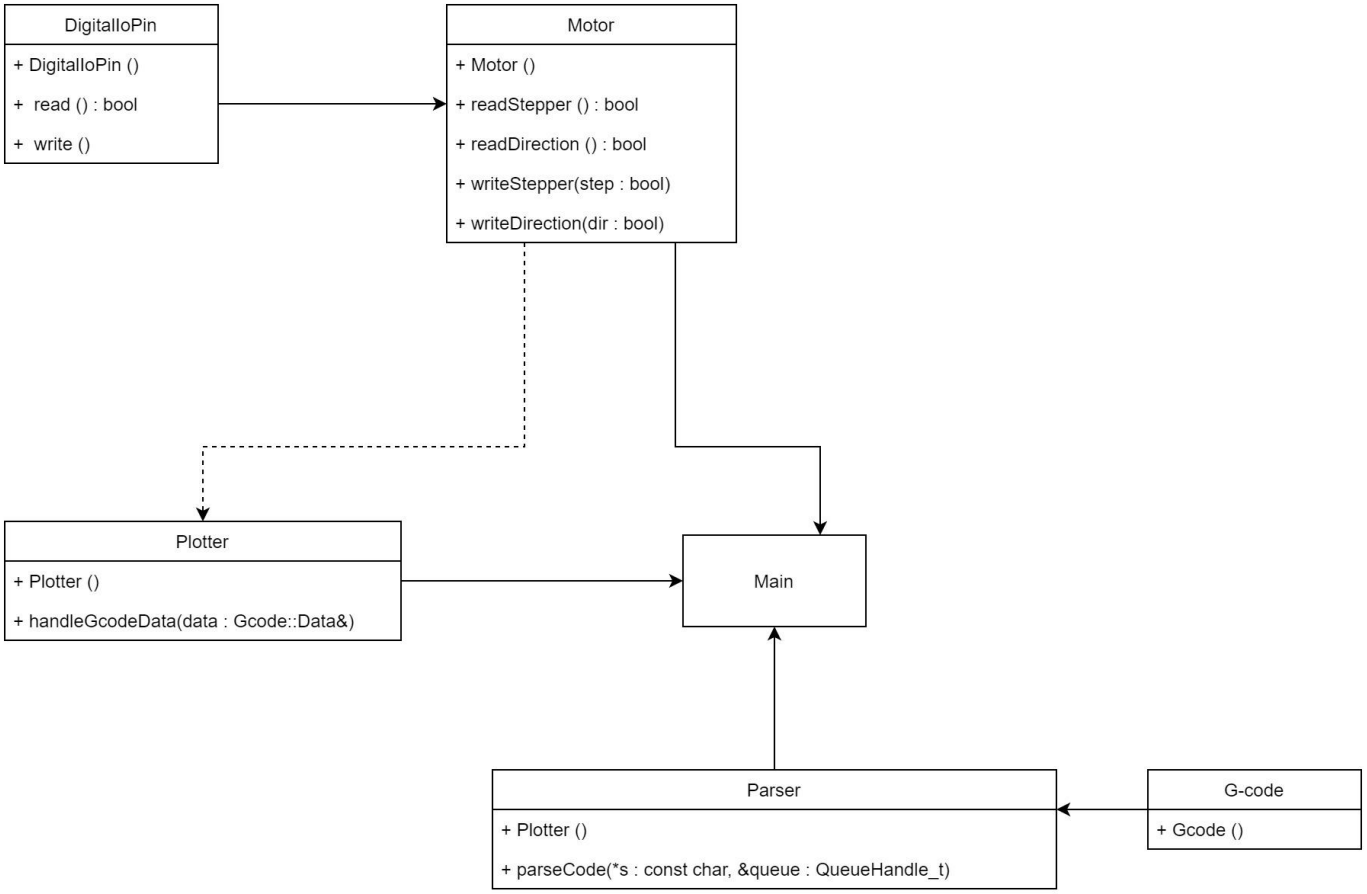
5. If needed plotting area and stepper directions can be changed in mDraw settings panel. After this plotter re-calibrates itself automatically.

6. Start plotting by pressing play button.

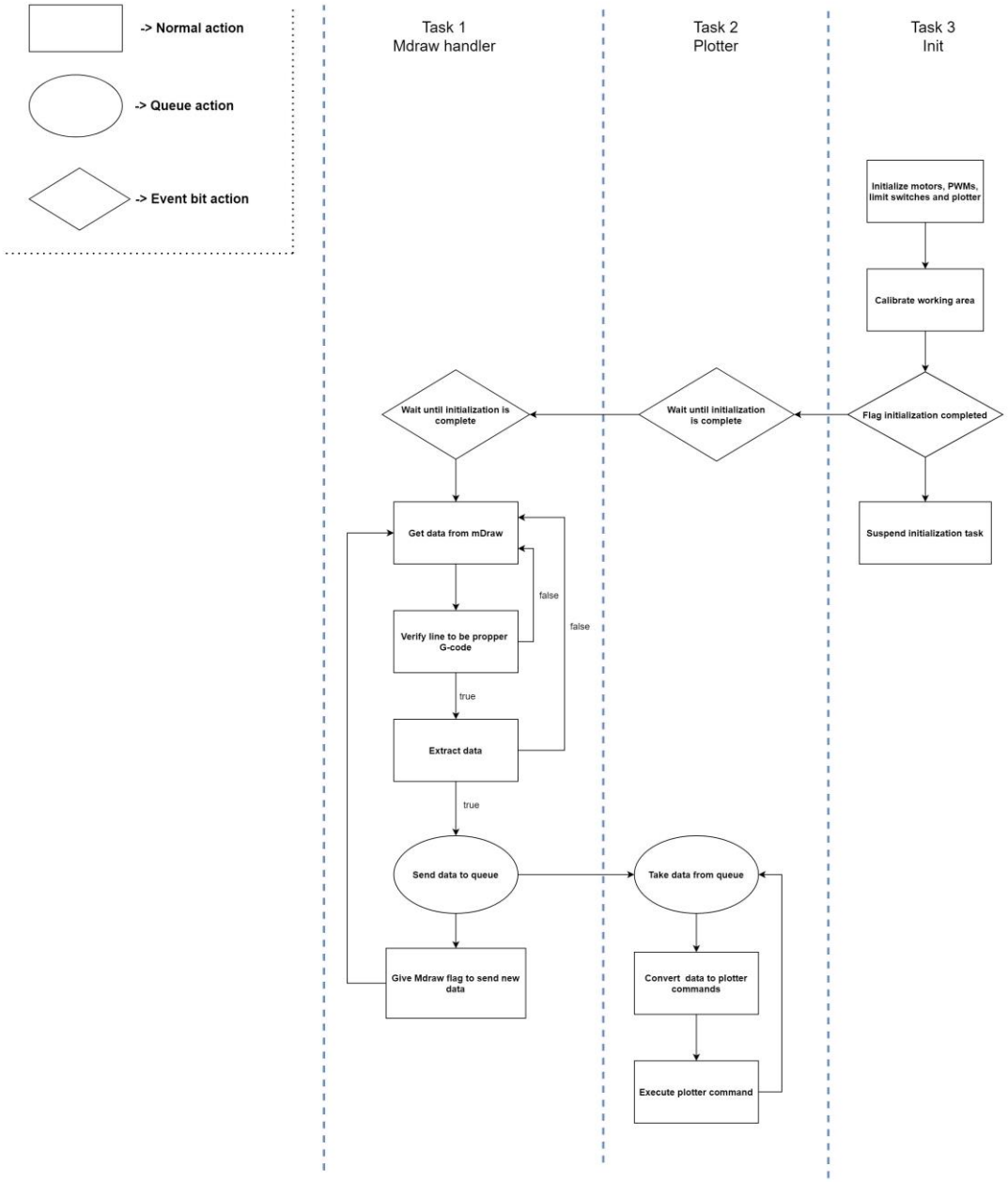
7. Wait for plotting to be finished.

3 Simplified class diagram

Full class diagram can be found here



4 Flow chart


-> Normal action
-> Queue action
-> Event bit action
Task 1
Mdraw handler
Task 2
Plotter
Task 3
Init
Initialize motors, PWMs, limit switches and plotter
Calibrate working area
Flag initialization completed
Wait until initialization is complete
Wait until initialization is complete
Suspend initialization task
Get data from mDraw
false
Verify line to be propper G-code
false
true
Extract data
true
Send data to queue
Take data from queue
Give Mdraw flag to send new data
Convert data to plotter commands
Execute plotter command

5 Functionalities

5.1 G-code parser

For parsing the C function sscanf is used to extract all data. Firstly the letter and number in the beginning of the line is extracted and checked that we got both. If we get both we can create a Gcode out of this and compare it to our existing Gcode variables if it matches any of them. All our Gcode variables hold a callback function for extracting data that is used to verify that all necessary data is received. Then this data is copied to back of a FreeRTOS queue that we can execute in the plotter. And since we use a queue the order of every command is always correct.

5.2 Line drawing algorithm

Integer version of Bresenham algorithm is used for plotting a line from x1,y1 to x2,y2 coordinates. This algorithm can only draw lines that increase in X and Y direction. To overcome this, direction of the motors is set accordingly so if the axis increases or decreases. The one axis that has the greater difference the motor is always moved, but the other axis only moves motor if the previous coordinate is different from the new calculated coordinate.

```
xMotor->writeDirection(x2_ > x1_ ? !xMotor->getOriginDirection() : xMotor->getOriginDirection());
yMotor->writeDirection(y2_ > y1_ ? !yMotor->getOriginDirection() : yMotor->getOriginDirection());
int x1      = x1_ < x2_ ? x1_ : x2_;
int x2      = x1_ > x2_ ? x1_ : x2_;
int y1      = y1_ < y2_ ? y1_ : y2_;
int y2      = y1_ > y2_ ? y1_ : y2_;
```

5.3 Scaling between mDraw and plotter

To draw accurate, the same ratio between the x-axis and y-axis of both simulator or Makeblock XY-plotter needs to be managed. First, number of steps between the x and y-axis of the frame is calculated. The ratio x/y steps is used to re-scale the mdraw frame so that the distance on the mdraw's map corresponding to the distance on the XY-plotter frame. The steps per millimeter are also needed to draw accurate.

```
// mdraw coordinate should be same as emulator or paper
if(totalStepX>totalStepY)
    savePlottingWidth = savePlottingHeight * totalStepX / totalStepY;
else
    savePlottingHeight = savePlottingWidth * totalStepY / totalStepX;

// Set step per mm to draw more accurate or same scale as svg picture
setXStepInMM(savePlottingWidth);
setYStepInMM(savePlottingHeight);

ITM_print("xTotal=%d, yTotal=%d\n", totalStepX, totalStepY);
ITM_print("xMM=%f, yMM=%f\n", xStepMM, yStepMM);
ITM_print("calibration done\n");
status |= CALIBRATED;
```