Jarkko Aho and Robin de Jong

XY-Plotter

Embedded System project

Helsinki Metropolia University of Applied Sciences

Smart Systems

Project document

17.10.2017

Helsinki Metropolia University of Applied Sciences

Contents

1 Introduction

This project was a part of a Smart Systems Embedded System project course. The project was a team work, and the team consist of three students. The team's goal was to create the firmware for Makeblock XY-Plotter drawing robot using LPCXpresso 1549 microcontroller. The team prepared for this project by doing a lab exercises, and these exercises were used as a part of the project. The plotter was required to draw at least a circle, and if there was time a more complex picture.

2 Components

2.1 LPCXpresso 1549

The LPC1549 LPCXpresso Cortex-M3 microcontroller is a low-cost development platform available from NXP supporting NXP's ARM-based microcontrollers. The LPC1549 was selected by the course teacher as a part of an other embedded system course.

2.2 Makeblock XY-Plotter

Makeblock XY-Plotter is a drawing robot that can move a pen or other instrument to draw digital artwork on flat surface. The plotter is equipped with two stepper motors, four switches, a servo motor and a laser engraver. Makeblock XY-Plotter is designed for Arduino microcontrollers but can be used with other microcontrollers as well.

3.1.1 Stepper Motors

42BYG Stepper motors move the pen and laser engraver in X and Y directions. The stepper motors are controlled with pulses from microcontroller which are sent to Me Stepper Motor Drivers.

3.1.2 Switches

The switches are located in X and Y end positions. The switches give current when pressed, and are used to make sure the stepper motors stop running if they are moving over their working area.

3.1.3 Servo Motor

The pencil used to draw the picture on paper is driven by a SG90 9g servo motor. The servo motor works with 20ms wavelength. When the pulse width is set to 1ms the pencil will be lifted off the paper, 1.5ms is the center position and 2ms is used to put the pencil on the paper.

3.1.4 Laser engraver

The laser engraver is Makeblock's own semiconductor diode laser. The laser's wavelength is blue-violet 405nm, the engraving accuracy is 0.1mm and it can output a 500mW of power. The laser is strong enough to burn pictures to wood but in this project it was used to draw in white paper. In this project the laser were implemented but never tested in practice.

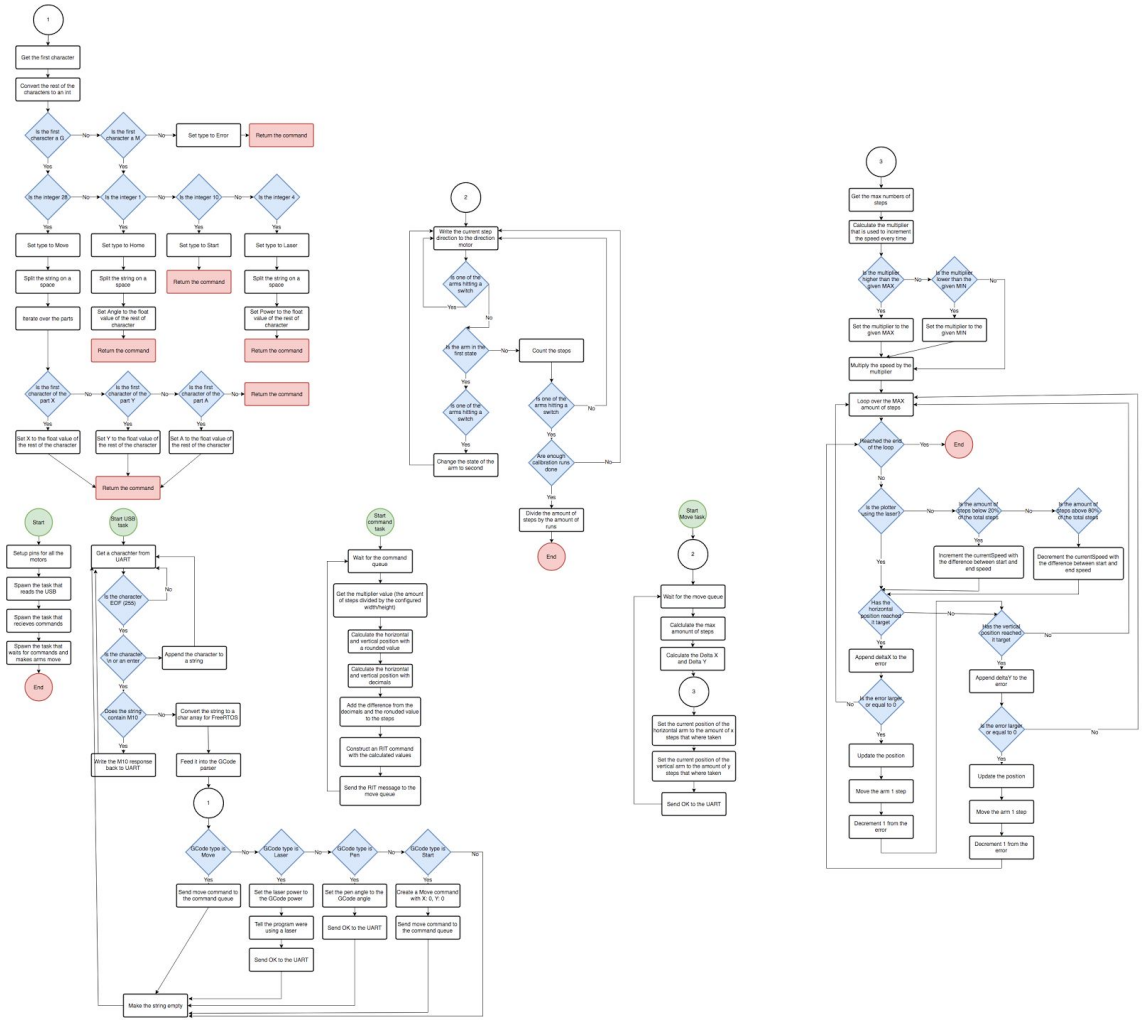
3 Programming

The program was written in C/C++. At the beginning of the project the project was completely in C++, but the team started to experience some strange errors which were solved by going with a C approach. Later on in the project the team couldn't get the plotter to draw curves nicely, due to this problem the team started consulting at other teams and their code for handling the coordinate parsing.

The C++ project was created with two of the SOLID principles in mind, namely: single responsibility principle and open closed principle. This created more tasks, queues, and semaphores than needed.

Due to the ability of not being able to draw curved lines, our team rewrote the code in C. This removed a lot of the tasks and queues we had in place for objects to communicate to each other.

The flow of the program is explained in the following image. A larger version can be found as an attachment.



3.1 Softwares

The C/C++ code was written with MCUXpresso IDE. To use the XY-Plotter, software is needed that converts SVG into G-Code; for this project mDraw is used. The team made most of the SVG test pictures themselves with Adobe Illustrator.

3.1.1 MCUXpresso IDE

MCUXpresso is the IDE used to develop the firmware for this project. The MCUXpresso IDE is an Eclipse-based development environment for NXP MCUs based on ARM Cortex-M cores, including LPC microcontrollers. The MCUXpresso IDE offers advanced editing, compiling and debugging features which are great for a student and professional use. The MCUXpresso IDE has a set of tools, plugins, and workspaces on top of Eclipse which makes the MCUXpresso a solid IDE to start from. If Eclipse is already installed all the tools, plugins, workspaces, and keybinds are copied into the IDE.

The IDE provides great debugging tools, but a big part of the tools are not usable on Mac OS. The mode can't be set to "ALL STOP", which allows the debugger to see all the threads. Mac OS can only see the first thread and debug on that thread.

3.1.2 mDraw

mDraw is a program developed by Makeblock, which converts SVG's into G-codes. The codes are given to the firmware and then used to drive the plotter.

Most problems arose when using mDraw, the software is made for their own firmware which gave us a lot of boundaries. One of the first problems we had was with sending the M10 command, which can be used to specify configurations of the plotter, for example the width and height of the drawing area. When we tried to handle the M10 at the place where the rest of the commands were handled, mDraw would crash. This was fixed by responding to the M10 command at the USB task, this ensured that the command was sent as soon as possible.

A big problem at the start was when mDraw crashed, which happens quite quick. When this happend on Mac OS, the program would bug out and the program couldn't select the XY-plotter or the USB port to use. Everytime mDraw crashed, it had to be removed from the laptop and reinstalled.

3.1.3 G-Code

G-Code (or RS-274) is the common name for the most widely used numerical control (NC) programming language. G-Code commands are used to drive the plotter but the commands used by mDraw are a bit different from the normal RS-274 standard. The commands mDraw uses, are shown in the following table.

GCode	Values	Description
M1	angle	Set the pencil servo to the given angle.
M4	power	Set the power of the laser to the given power.
M10	none	Send when connecting mDraw to the plotter. A response should be given back, which includes the following parameters: width, height, motor X direction, motor Y direction, motor switch, speed, pen up position, pen down position
G1	x, y, angle	Move to a specified position, the angle is not used in mDraw or in this project.
G28	none	Move to the home position (0, 0)

3.2 C/C++

The current code has a few different parts that make the system work. The most challenging part where moving the arms correctly. This means that 30 degree angle lines should be drawn correctly, this can be done with Bresenham's line algorithm.

3.2.1 Bresenham

Bresenham’s line algorithm allows lines with angles to be drawn on a computer screen. During the project three different implementation were used, but the team found out that only the last one, which is the one used in the final project, had the most precision. This implementation is taken from the official firmware (link in references).

This can also be used to know how much the horizontal and vertical arms should be moved.

3.2.2 Coordinates

To move the arms mDraw sends G1 commands to the plotter. The horizontal and vertical values in those commands are within the given width and height, the size is given at the start with a M10 command.

To transform one coordinate system to the other, it's important that both arms calibrate before drawing. Every time an arm moves a step, a step counter is incremented. When the arm is done calibrating it knows how much steps it needs to go from the start to the end. This steps value is than used to transform the mDraw coordinate to the XY-plotter coordinate, which is explained in the following steps:

1) Divide the amount of steps by the size given at the start.
2) Multiply the horizontal and vertical mDraw value with the previous value.

The new coordinates are then send to the queue to be processed by RIT.

3.2.3 G-Code Parser

Because the G-Code mDraw uses differs from RS-274, a custom G-Code parser was written. The first approach was STD packages, but after trying to implement more logic for the arm movement, the team kept getting hard faults. After many hours of debugging, and the help of teacher, the team started working on a different implementation.

Final implementation is the same as in the official firmware, which doesn't use string splitting and making new variables. It still returns the same objects, but with less memory needed. This fixed the hard fault problem and the team didn't have any hard fault problems any more.

3.3 Other problems

The LPC1549 has two micro USB ports allowing a code were one port is for sending M10 and “OK” to the mDraw and one for getting coordinates from the mDraw. This feature was not implemented in the final project because the other port was somehow connected to one of the button pins. Because of this the data was either corrupted or did not go through at all.

Same kind of problem arise when trying to send the M10 command from the USB handler. The data was lost or never send to the mDraw and the program never continued after calibration.

The mDraw was far from functioning program and is still, while writing this paper, full of errors and problems. The mDraw crashed frequently, and when used on Mac OS every time it crashed it required a reinstallation. When drawing a perfect circle where the start and the end point are in the same coordinates the mDraw did not send the correct end coordinate and a circle had a hole at the end.

4 Conclusion

The project was a great example of an embedded system device. It displayed how different motors are controlled with microcontroller, how algorithms are used as a part of a code and how the G-code works. The team performed nicely and the final project was a working prototype. The project could have been improved to support other SVG to G-Code programs, for now it only works with the mDraw.
The team learned the importance of interrupts and timing in the code, RTOS usage in ARM processors, how to control motors with pulses and different line drawing algorithms. With this information the team could build even a 3D printer.

The team’s source code can be seen in: https://glitter.jemoeders.website/Team2/Plotter

References

1 FreeRTOS documents

http://www.freertos.org/Documentation/RTOS_book.html

2 LPCXpresso documents

https://www.nxp.com/docs/en/user-guide/UM10736.pdf

3 mDraw

https://github.com/Makeblock-official/mDrawBot

4 Bresenham's line algorithm

https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

5 Makeblock's official XY-Plotter firmware
https://github.com/Makeblock-official/mDrawBot/blob/master/firmwares/xybot/xybot.ino

6 Makeblock XY-Plotter robot kit

http://store.makeblock.com/xy-plotter-robot-kit/

7 School documents

Other documents that were given in course's OMA workspace