Olli Raudakoski, Bastien Gouila, Konstantin Lobkov

XY Plotter Control Project

Embedded Systems Project

Helsinki Metropolia University of Applied Sciences
Degree

Information Technology

Project Report

08/11/2016

Helsinki
Metropolia
University of Applied Sciences

Author(s) Title Number of Pages Date	Olli Raudakiski Bastien Gouila Konstantin Lobkov XY Plotter Control Project 16 pages + 4 appendices 8 November 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Smart Systems
Instructor(s)	Keijo Länsikunnas, Lecturer Joseph Hotchkiss, Instructor Sami Sainio, Instructor

Contents

1 Introduction

The purpose of this project is to design and implement an XY plotter control system for fast and precise pencil drawing or laser engraving by synchronizing drawing software with a physical drawing tool using an embedded microcontroller.

The system's components include:

- microcontroller (LPCXpresso 1549)
- drawing software (mDraw)
- physical drawing tool (XY Plotter)

The microcontroller is the primary component in this system. It controls the stepper motors of the XY plotter based on the commands received from mDraw (G-code) and monitors for possible errors. The user selects the vector image he wants to upload to mDraw and decides if the pencil or the laser should be used.

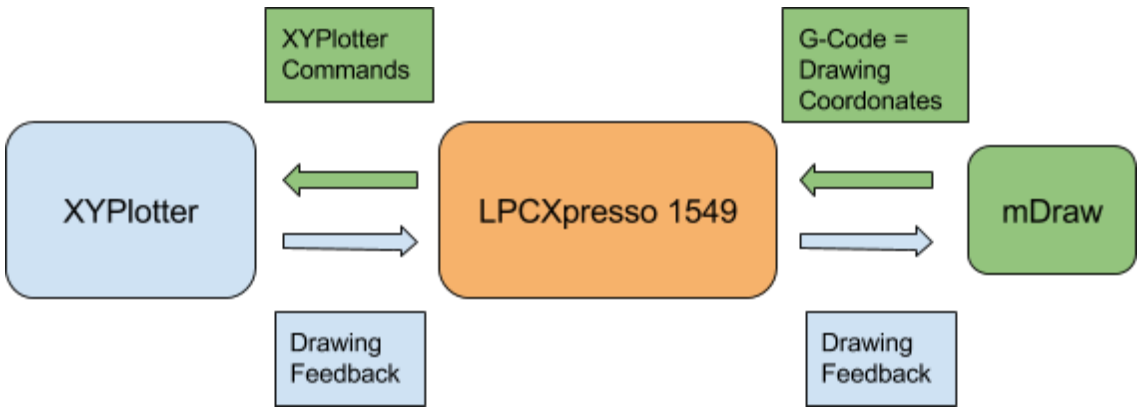

Figure 1. General schematic of the system.

2 Project Details

2.1 Section Overview

In this section we are going to give more background information about the project. This will include information regarding early stages of the project, preparation, and planning.

2.2 Team

We were originally a team of 4 people, as it has been personally assigned by the teacher Keijo in the beginning of the period, but since one of the members has never showed up and has not contributed anything worth of notice so far, we are, in fact, a team of three.

2.2.1 Tasks Distribution

The tasks distribution has been based on a person's specific skills: this allows for more efficient and rewarding work on the project and keeps up the morale. Team members have not been completely devoted to their specific parts, they have also worked on other members' areas of responsibility in their spare time.

After Joakim's disappearance, Konstantin has taken most of the group's responsibilities for the report and documentation, while Bastien and Olli has taken over Joakim's technical part.

- Olli Raudakoski: G-Code parser, serial communication, report and documentation
- Bastien Gouila: Motor, pencil and laser control, report and documentation
- Konstantin Lobkov: Error handling, report and documentation

2.2.2 Project Timetable

2.2.2.1 Schedule

We have managed to keep up with the schedule to some extent, but due to the extreme abundance of laboratory exercises in the corresponding course, ARM-Processors and Embedded Systems, and another course, Digital Signal processing, we, like most of the groups, have not been able to finish the project at the first specified deadline. However, we have been provided with some extra time, that allowed us to keep working on the project.

The latest version of the project schedule is as follows:

- Week 35 (29.8 - 2.9)
  - general knowledge on LPCXpresso, mDraw and XY-Plotter systems
  - install the required software: mDraw, Benbox, LPCXpresso software
- Week 36 (5.9 - 9.9)
  - general knowledge on LPCXpresso, mDraw and XY-Plotter systems
  - project scheduling and analysis
- Week 37 (12.9 - 16.9)
  - create respecting functions
  - begin implementation
- Week 38 and Week 8 (19.9 - 23.9)
  - implementation
- Week 39 (26.9 - 30.9)
  - finish implementation
  - begin verification
- Week 40 (3.10 - 7.10)
  - finish verification
  - project report
- Week 41 (10.10 - 14.10)
  - finish verification and report

2.2.2.2 Risk analysis

We have also taken into account various hazards and their timetable effect. Some of these, as it has been mentioned earlier, have actually happened.

Risk	Effect (Days)	Probability (%)	Schedule Effect (Days)	Countermeasure
Components not accessible	7	10	2-3	Ask from other group members and classmates if we can use theirs.
Teammate drops course	14	20	3	Assign this person's tasks evenly between the rest of the team.

knowledge				library or use Google.
Motivation drop	5	75	5	Watch Rocky montage videos from Youtube.
Feature creep	5	20	5	Focus on the main functionalities first.

2.3 Project Specifications

In this section we are going to give more specific information regarding the components of the project and how they work in conjunction with each other.

2.3.1 Requirements

The first objective of this project is to create a program in C++ using LPCXpresso development board. The program has to be able to let us draw basic images (like a square or a circle) using mDraw and controlling the pen of the XY Plotter.

When this objective is completed, following upgrades can be done:

- Accuracy of the drawing
- Possibility of drawing more complex pictures (with curves, for example)
- Using both the pen and the laser
- Acceleration and smooth moves for the pen
- Interruption management for errors when we hit a switch of the XY Plotter

2.3.2 Components Overview

2.3.2.1 mDraw

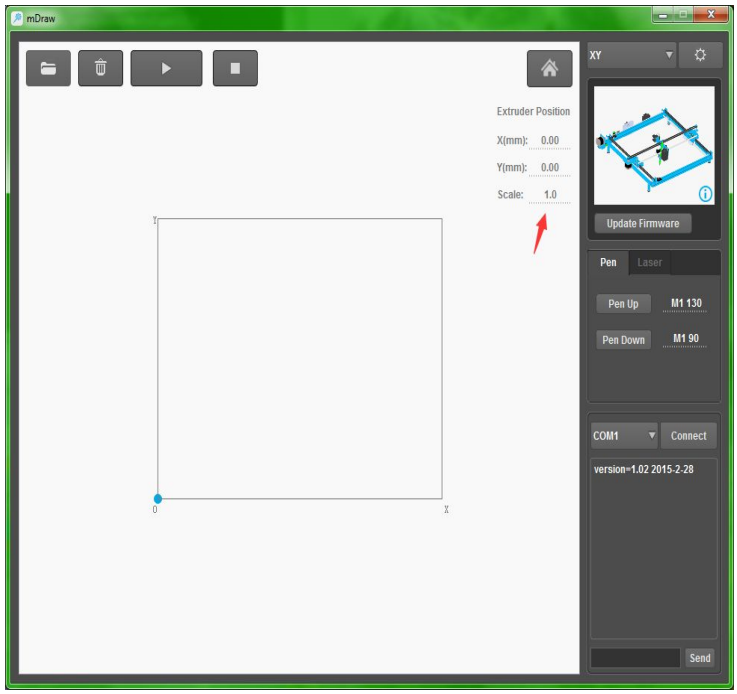

Figure 2. Picture of mDraw.

mDraw is a software which takes .svg images and converts the vectors to G-code. Our XY plotter reacts to only 5 G-codes.

2.3.2.2 XY Plotter

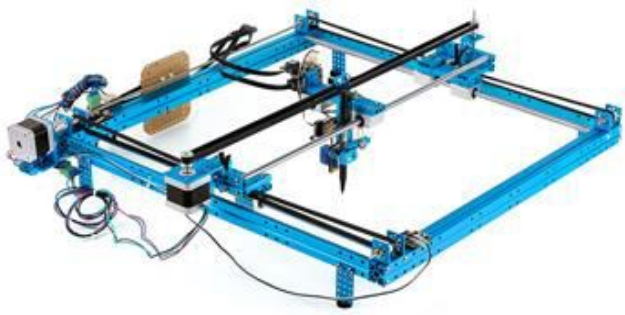

Figure 3. Picture of the XY plotter.

The XY Plotter is connected to the LPCXpresso.

The XY Plotter is made of 1 servo to move the drawing block and 2 switches per axis to know the limits.
When a limit switch is hit, the motor must stop.

2.3.2.3 LPCXpresso 1549

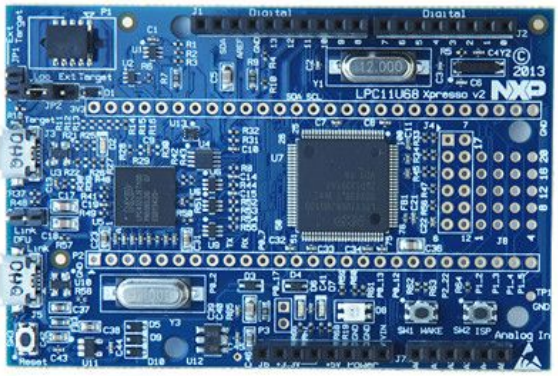

Figure 4. Picture of the LPCXpresso 1549.

The LPCXpresso 1549 is an embedded chip which creates the link between mDraw and the XYPlotter.
The chip parses the G-Code from mDraw and sends the corresponding commands to the XYPlotter.

3 Implementation

3.1 Section Overview

The following section will be described with the function prototypes and/or portions of the complete code. If the complete source code is needed, it can be accessed with this link
https://drive.google.com/open?id=0BxyAuAeQEyobTlNsNEk3bk9fSEE

The Main.cpp file can be found in the appendices 1, 2 and 3.

3.2 mDraw and G-code parser

3.2.1 Reading the commands

```
//This function is receiving the characters one by one from the serial port, putting them into a buffer and parsing it in order to recognize G-Code language. Finally it sends back the command type
bool GParser::takeCommand(int commandChar,int &num);
```

3.2.2 Parsing the commands

```
//Take the buffer and then separate the code in a two dimension array like [SentenceWord, WordContent]
void GParser::separate()
//Take the two dimension array and parse it in order to recognize G-Code and coordinates
void GParser::parse()
```

3.2.3 Executing the commands

The commands are sent from the Receiving Task to the Motor Task by using a queue and a semaphore is also protecting the use of the serial port to avoid a conflict between these two tasks.

Command example :

- M10 XY 310 350 0.00 0.00 A0 B0 H0 S80 U160 D90 = initialization
- M1 = use the pen
- M4 = use the laser

3.3 XY Plotter

3.3.1 Stepper motors and drawing tools

The XY Plotter is made of 2 stepper motors (one per axis) and 4 limit switches for the minimum and the maximum for both axis.
We have fixed the minimum and maximum on the axis according to the following figure.

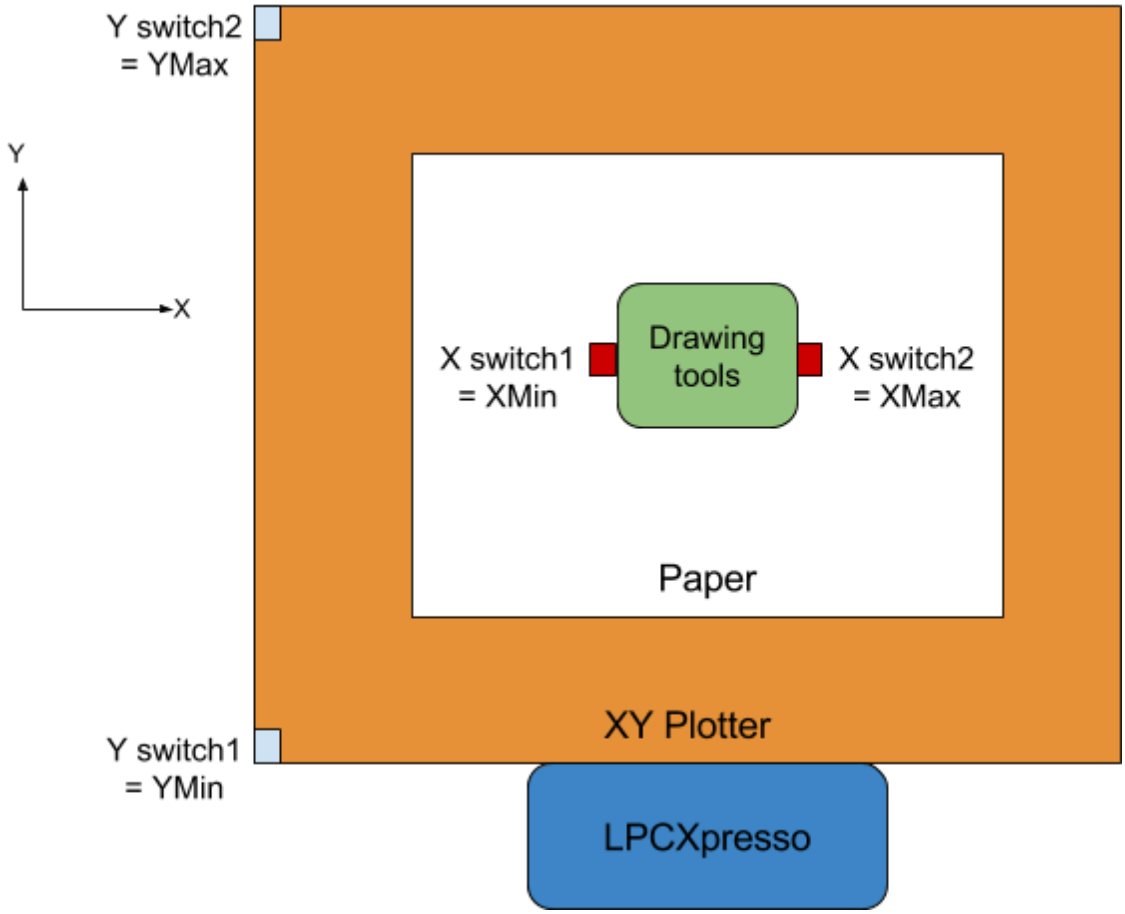

Figure 5. XY plotter overview

The XY Plotter offers two drawing tools:

- A pen that is controlled with two PWM signals = one means “Up” (not drawing) and the other means “Down” (drawing).
- A laser where the intensity of the laser is controlled by the percentage of a PWM signal = 0% laser off and 100% full power

.

For the initialization and the use of the PWM for the drawing tools refer to the appendix 4.

3.3.2 Drawing functions

The complete code of the following functions can be found in the file LPCXProject.cpp file located in the link provided in 3.1.

Calibration:

The calibration is done with the two following functions. They are identical, just the pin numbers are different according to the axis.

The function counts the number of steps between the switches and returns this distance. When the block is calibrated, the origin and the end are fixed next to the switches without touching them.

```
//Return the number of steps between switches and put the block at the origin of the X axis
int CalibrationX (void );
//Return the number of steps between switches and put the block at the origin of the Y axis
int CalibrationY (void );
```

Coordinates Conversion:

```
//The function converts the millimeters coordinates into steps coordinates for the X axis
int CoordinatesXStep (float fXmillimeters, int iTotalNbStep);

//The function converts the millimeters coordinates into steps coordinates for the Y axis
int CoordinatesYStep (float fYmillimeters, int iTotalNbStep);

//The function converts the steps coordinates into millimeters coordinates for the X axis
float CoordinatesXMilli (int  iNbSteps, int iTotalNbStep)

//The function converts the steps coordinates into millimeters coordinates for the Y axis
float CoordinatesYMilli (int  iNbSteps, int iTotalNbStep)
```

Move the axis:

The MoveXAxisAcceleration() and MoveYAxisAcceleration() are moving their respective axis thanks to the RIT interrupts while the final position hasn't been reached.

```
//Move the block on the X axis and return the new position with the corresponding speed and return the current position
int MoveXAxisAcceleration(int  iStartPos, int iFinalPos, int MaxSpeed);
```

//Move the block on the Y axis and return the new position with the corresponding speed and return the current position

int MoveYAxisAcceleration(int iStartPos, int iFinalPos,int MaxSpeed);

// Calculate the new coordinates and move the X and Y axis at variable speed according to the dimensions of the XYPlotter. Return a boolean that indicates if the operation was successful

bool MoveXYPlotter(float fXCoordinates, float fYCoordinates, int NbTotalStepX, int NbTotalStepY);

The following piece of code shows how we move the drawing block if the movement on X is on the positive way. There is also the case where the movement is negative or null on x.

```
float dx = fXCoordinates - fSavedMilliXaxis;
float dy = fYCoordinates - fSavedMilliYaxis;
float Y = 0;
int iSpeed = NormalPps;
float distance = sqrt((dx*dx)+(dy*dy));
float distanceRemain = distance;
float YPrevious = fSavedMilliYaxis;
int iXCheck = 0;
int iYCheck = 0;

if(dx > 0){
    for (float X = fSavedMilliXaxis + 0.01 ; X <= fXCoordinates ;X = X + 0.01 ) {
        Y = fSavedMilliYaxis + dy * (X - fSavedMilliXaxis) / dx;//calculate the new Y coordinates every 0.01mm on X
        //distance remaining
        distanceRemain = sqrt( (((X-0.01)-fXCoordinates)*((X-0.01)-fXCoordinates)) + ((YPrevious-fYCoordinates)*(YPrevious-fYCoordinates)) );
        //decrease the speed while the drawing block is under 5mm from the end or from the beginning
        if(distanceRemain <= 5 || distanceRemain >= distance-5){
            if(iSpeed <=NormalPps){
                iSpeed = NormalPps;
            }
            else{
                iSpeed /=1.001;
            }
        }
        //increase the speed
        else{
            if(iSpeed >=NormalPps*5){
                iSpeed = NormalPps*5;//Maximum 5*12500
            }
            else{
                iSpeed *=1.001;
            }
        }
        //Move X and Y and remember the previous Y value for the newt cycle
        iXCheck = MoveXAxisAcceleration(CoordinatesXStep((X-0.01),NbTotalStepX),CoordinatesXStep(X,NbTotalStepX),iSpeed);
        iYCheck = MoveYAxisAcceleration(CoordinatesYStep(YPrevious,NbTotalStepY),CoordinatesYStep(Y,NbTotalStepY),iSpeed);
        YPrevious = Y;
```

//Calculate the new coordinates and move the X and Y axis at constant speed according to the dimensions of the XYPlotter. Return a boolean that indicates if the operation was successful

bool MoveXYPlotterLaser(float fXCoordinates, float fYCoordinates, int NbTotalStepX, int NbTotalStepY);

This function is the same than MoveXYPlotter() but the speed is constant because we want the lines identically burned with the laser.

//getter for returning the distance to that mDraw requires
float getDistance(float futureX, float futureY);
We calculate the distance with Pythagore:
distance = (futureX - previousX)² + (futureY - previousY)²

//Handler of the interrupt where the motors are controlled
void RIT_IRQHandler(void);

```
if(bAxisX == true){
    Chip_GPIO_SetPinState(LPC_GPIO, 0, 24, bXMotor);// X Motor
    bXMotor = !bXMotor;
}

if(bAxisY == true){
    Chip_GPIO_SetPinState(LPC_GPIO, 0, 27, bYMotor);// Y Motor
    bYMotor = !bYMotor;
}
```

The previous code is the part which is controlling the stepper motors

//Interrupt initialization and call
void RIT_start(int count, int pps);
pps is the value that we use to manage the speed because it fixes the intervals between the handlers according to this formula:
uint64_t cmp_value = (uint64_t) Chip_Clock_GetSystemClockRate()/ (uint64_t)pps;

3.3.3 Drawing possibilities

With the previous code we are able to draw basic figures like pictures with straight lines as well as more complex curves with a good accuracy.

The drawing speed is fast enough and we can increase the speed when we are drawing long lines with the pen.

For now, the pen works, but not the laser. We have faced some problems with the laser power controlled by the PWM:

- The laser PWM switches to some strange behaviour on our stepper motor by making them move when the PWM value is under 10%. This problem has also been reported by several groups.
- After 10% the laser works fine.

4 Conclusion

As the results of testing our project's design implementation proved to be more than satisfactory, we believe that our team has achieved the project's goal in creating a reliable XY Plotter control system and improved team members' professional competence in a corresponding field.

Here is the list of our set objectives:

- Drawing simple figures (with straight lines) = working
- Drawing complex figures (with curves) = working
- Drawing with accuracy = pretty good
- Drawing with acceleration for the pen = working
- Drawing with the laser = in theory it works, but in practice we got some strange behaviour with the PWM under 10% of laser power (similar problems observed in several groups)
- Interruptions handler with the limit switches = does not work, but a security measure has been implemented to stop the motors when the drawing block hits one limit switch.

However, our team has faced several difficulties during the development process, the hardest of them being an extremely heavy workload for quite some time of the project. The cause for this have been two other courses ("Digital Signal Processing" and "ARM-Processors and Embedded Systems") with a lot of time-consuming laboratory exercises, which resulted in an extreme shortage of time and punishing deadlines, even though the teachers have been extending those deadlines for us.

Another problem has been the lack of knowledge for some parts of the project (and the laboratory exercises), which have enhanced the impact of the previously described time shortage problem.

To sum up everything said before, our team believes that it has managed to achieve the most part of what we have planned for this project and believes that every team member has improved the related skills and increased his knowledge to some extent. However, in order to implement all the planned features in shorter time, we need more knowledge and experience in C++ programming language, because our lack of knowledge in the beginning of the development process required a lot of extra time and work to improve it. Additionally, our team would appreciate having some spare time by reducing the amount or/and difficulty of laboratory exercises.

References

This part of the report is unfortunately very small. Most of the required material has been covered by the laboratory exercises of the corresponding course "ARM-Processors and Embedded Systems" provided by our teacher Keijo. We have attached the appendices with some extracts from the source code.

1. http://forum.makeblock.cc/t/xy-plotter-v2-0-mdraw-version-xy-plotter-v2-0-laser-engrave-mdraw-version-release-post/922 - mDraw software and XYPlotter documentation
2. www.Google.com - SVG images

Appendix 1: Board initialization and Main()

```
#include "LPCXProject.h"
#define penTool     1
#define laserTool   0
QueueHandle_t xQueue = xQueueCreate( 20, sizeof(debugEvent) );
SemaphoreHandle_t xSemaphore = xSemaphoreCreateBinary();
GParser parser;
DrawingObject pen,laser;

int Xmillimeter=0;
int Ymillimeter=0;
int command=0;
int charCount=0;
int iNbTotalStepXaxis,iNbTotalStepYaxis;

static void prvSetupHardware(void)
{
    SystemCoreClockUpdate();
    Board_Init();
    //Chip_IOCON_PinMuxSet(LPC_IOCON,1,11,(IOCON_MODE_PULLUP |IOCON_DIGMODE_EN |IOCON_INV_EN ));
    // initialize RIT (= enable clocking etc.)
    Chip_RIT_Init(LPC_RITIMER);
    // set the priority level of the interrupt
    // The level must be equal or lower than the maximum priority specified in FreeRTOS config
    // Note that in a Cortex-M3 a higher number indicates lower interrupt priority
    NVIC_SetPriority( RITIMER_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY + 1 );
    Chip_GPIO_SetPinDIROutput(LPC_GPIO, 0, 24); // X Motor
    Chip_GPIO_SetPinDIROutput(LPC_GPIO, 0, 27); // Y Motor
    Chip_GPIO_SetPinDIROutput(LPC_GPIO, 0, 28); // Y Direction
    Chip_GPIO_SetPinDIROutput(LPC_GPIO, 1, 0); // X Direction
    /* Initial LED0 state is off */
    Board_LED_Set(0, false);
    Chip_IOCON_PinMuxSet(LPC_IOCON, 0, 29, IOCON_MODE_PULLUP);
    Chip_IOCON_PinMuxSet(LPC_IOCON, 0, 9, IOCON_MODE_PULLUP);
    Chip_IOCON_PinMuxSet(LPC_IOCON, 1, 3, IOCON_MODE_PULLUP);
    Chip_IOCON_PinMuxSet(LPC_IOCON, 1, 0, IOCON_MODE_PULLUP);
}
```

```
int main(void) {
    prvSetupHardware();
    xSemaphoreGive(xSemaphore);

    laser.init(LPC_SCT1,SWM_SCT1_OUT0_O,1000,laserTool,0,12);
    pen.init(LPC_SCT0,SWM_SCT0_OUT0_O,50,penTool,0,10);

    xTaskCreate(ReceiverTask, "take command", configMINIMAL_STACK_SIZE * 3, NULL,
            (tskIDLE_PRIORITY + 1UL), (TaskHandle_t *) NULL);

    xTaskCreate(MotorTask, "just for example", configMINIMAL_STACK_SIZE * 6, NULL,
            (tskIDLE_PRIORITY + 1UL), (TaskHandle_t *) NULL);

    //ITM_init();
    /* Start the scheduler */
    vTaskStartScheduler();
}
```

Appendix 2: G-Code Receiving and Parsing Task

```
static void ReceiverTask(void *pvParameters){
    int receivedChar;
    debugEvent debbie;
    while(1){
        if(xSemaphoreTake(xSemaphore,portMAX_DELAY)== pdTRUE){
            receivedChar=Board_UARTGetChar();
            if(receivedChar!=EOF){
                if(parser.takeCommand(receivedChar,charCount)){ //when enter is pressed true is returned
                    debbie=parser.getDebugEvent();
                    //ITM_write(debbie.echo);
                    if(debbie.Gcode==110){
                        xQueueSendToFront( xQueue, &debbie, portMAX_DELAY );
                    }
                    else{
                    xQueueSendToBack( xQueue, &debbie, portMAX_DELAY );
                    }
                }
            }

            xSemaphoreGive(xSemaphore);
        }
    }

}
```

Appendix 3: Motor control Task

```
static void MotorTask(void *pvParameters){
    char buffer[50];
    int iCurrentStepXaxis = 0;
    int iCurrentStepYaxis = 0;
    bool bEcho;
    debugEvent moveInfo;
    while(1){
        if(xQueueReceive( xQueue, &moveInfo, portMAX_DELAY)){
            if(xSemaphoreTake(xSemaphore,portMAX_DELAY)== pdTRUE){
                switch(moveInfo.Gcode){
                case 110:
                    vQueueDelete( xQueue );
                    Board_UARTPutSTR("M10 XY 310 350 0.00 0.00 A0 B0 H0 S80 U160 D90 \n");
                    pen.write(160);
                    laser.write(0);
                    iNbTotalStepXaxis = CalibrationX();
                    iCurrentStepXaxis = iNbTotalStepXaxis/2;
                    iNbTotalStepYaxis = CalibrationY();
                    iCurrentStepYaxis = iNbTotalStepYaxis/2;
                    xQueue = xQueueCreate( 20, sizeof(debugEvent) );
                    break;
                case 101:
                    pen.write(moveInfo.Xplane);
                    break;
                case 104:
                    laser.write(moveInfo.Xplane);
                    break;
                case 928:
                    //move home xy(0,0)
                    sprintf(buffer,"distance= %3.2f",getDistance(moveInfo.Xplane,moveInfo.Yplane));
                    Board_UARTPutSTR(buffer);
                    bEcho = MoveXYPlotter(0, 0, iNbTotalStepXaxis, iNbTotalStepYaxis);

                    break;
                case 901:
                    //move to moveinfo.Xplane and .Yplane
                    sprintf(buffer,"distance= %3.2f ",getDistance(moveInfo.Xplane,moveInfo.Yplane));
                    Board_UARTPutSTR(buffer);
                    bEcho = MoveXYPlotter(moveInfo.Xplane, moveInfo.Yplane, iNbTotalStepXaxis, iNbTotalStepYaxis);

                    break;
                }
                Board_UARTPutSTR("OK \n");
                xSemaphoreGive(xSemaphore);
                vTaskDelay(5);
            }
        }
    }

}
```

Appendix 4: Initialization and use of the PWM for the drawing tools

```
void DrawingObject::init(LPC_SCT_T *pSCT1, CHIP_SWM_PIN_MOVABLE_T  movable1,
        int cycleLenght1, bool tool1, int port, uint8_t pin){
        pSCT=pSCT1;
            movable =movable1;
            tool=tool1;
            cycleLenght=1000000/cycleLenght1;
            Chip_SCTPWM_Init(pSCT);
            pSCT->CONFIG |= (1 << 17); // two 16-bit timers, auto limit
            pSCT->CTRL_L |= (72 - 1) << 5; // set prescaler, SCTimer/PWM clock = 1 MHz

            Chip_SWM_MovablePortPinAssign(movable, (uint8_t)port, pin);
            pSCT->RES = (pSCT->RES & ~(3 << (0 << 1))) | (0x01 << (0 << 1));//clear config resolution register
                        /* Set and Clear do not depend on direction */
            pSCT->OUTPUTDIRCTRL = (pSCT->OUTPUTDIRCTRL & ~(3 << (0 << 1)));

                pSCT->MATCH[0].L = cycleLenght -1;
                pSCT->MATCHREL[0].L = cycleLenght -1;
                if(tool){//pen
                    //Chip_SWM_DisableFixedPin(SWM_FIXED_ADC1_2 );
                pSCT->MATCHREL[1].L = cycleLenght*0.05;// at the start 95% duty cycle
                }
                else{//laser
                    //Chip_SWM_DisableFixedPin(SWM_FIXED_DAC_OUT);
                    pSCT->MATCHREL[1].L = 0;//laser is shut down
                }
                pSCT->EVENT[0].STATE = 0xFFFFFFFF;          // event 0 happens in all states
                pSCT->EVENT[0].CTRL = (1 << 12);            // match 0 condition only

                pSCT->EVENT[1].STATE = 0xFFFFFFFF;          // event 1 happens in all states
                pSCT->EVENT[1].CTRL = (1 << 0) | (1 << 12); // match 1 condition only

                pSCT->OUT[0].SET = (1 << 0);                // event 0 will set SCTx_OUT0
                pSCT->OUT[0].CLR = (1 << 1);                // event 1 will clear SCTx_OUT0

                pSCT->CTRL_L &= ~(1 << 2);                  // unhalt it by clearing bit 2 of CTRL reg
}
```

```
void DrawingObject::write(uint8_t power=0){
                laserPower=(float)power/255;
                laserPower=cycleLenght*laserPower;
    if(!tool){//using laser

        pSCT->MATCHREL[1].L = laserPower;
    }
    else{// pencil
        if(power==160){//up
        pSCT->MATCHREL[1].L = cycleLenght*0.05;
        }
        else if(power==90){//down
            pSCT->MATCHREL[1].L = cycleLenght*0.09;
        }
    }
}
```