# 2D VECTOR GRAPHICS PROCESSING ENGINE

Yashaswi Doddaveerappa (012468066)
*Computer Engineering Department, Charles W. Davidson College of Engineering*
*San Jose State University, San Jose, CA 94303*
*Email:* yashaswi.doddaveerappa@sjsu.edu

## Abstract

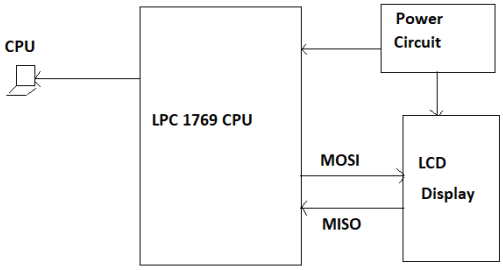
*The objective of this lab is to get a hands-on on interfacing LPC1769 with a LCD display. In this part of the project, we are implementing 2D Screen saver graphics and display it on LCD (Liquid Color Display) interfaces with LPC 1769 using the SPI (serial peripheral interface). The graphical LCD is an 18bit color LCD and LPC1769 is a Cortex-M3 micro-controller for embedded applications. A screen saver where we shrink the squares and is rotated for 10 times and display 2D tree with growing branches is displayed at random points.*

## 1. Introduction

The main objective of this lab is to design, build and design a system which is capable of displaying the screen saver and forest on LCD. LPC1769 is an ARM Cortex-M3 based microcontroller for embedded applications. This module has UART, USB, SPI, I2C that helps us to communicate with it effectively. We make use of graphical LCD which is able to display different shapes and pattern by communicating with SPI bus of LPC1769 module.

The LPC1769 module is powered up by connecting it to the CPU via USB cable. LCD is connected to the appropriate pins with the SPI port. The LCD lights up with indicates that the circuit is correct and the module is ready to communicate with the LCD.

*Block diagram of System*




## 2.METHODOLOGY

In this part of the section, we describe the implementation of the overall system i.e. the hardware and software methodology, objectives and the technical challenges.

We build the entire module on the wire wrapping board. The CPU makes use of interface to send the data to the module. The LCD which is been connected to the LPC is inputted with the data which takes the form of opcode and dummy bytes, this initializes the LCD. The 2D screen saver is displayed on the LCD when the LCD receive data input from CPU. The SPI port of module and LCD must be defined and initialized properly.

### 2.1 Objective and Technical Challenges

Objectives are:

- We have to build an interface circuit on wire wrapping board.
- In order to drive the circuit, we need the power supply which is built using voltage regulator.
- Interfacing the GPIO pins.
- To initiate the LCD, develop a program.
- Develop a program to create and display the squares on LCS screen.
- A program to display thick forest.

Challenges we faced:

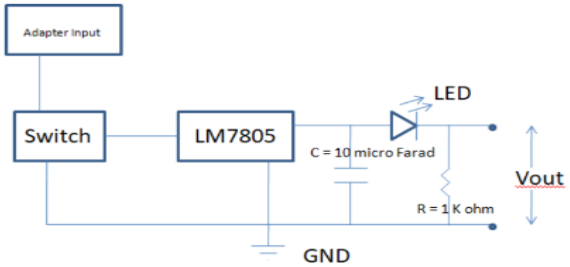
- Hardware: Few issues in LCD.
- Software: Generation of squares was not proper

### 2.2 Problem Formulation and Design

#### 2.2.1. Hardware Requirements

In order to drive the circuit, the power circuit is built using the IC LM7805 voltage regulator, switch, resistor, capacitors and LED (Light Emitting Diode) as shown below. Using multimeter we check the voltage between the LED and the ground.

Figure 1:Power Circuit



We now build the interface between the LCD and the CPU module. MISO, MOSI and SCK options of the micro-controller can be used as master to interface the SPI flash which is the slave. The CPU module and the LCD communicate with each other using SPI communication as shown.

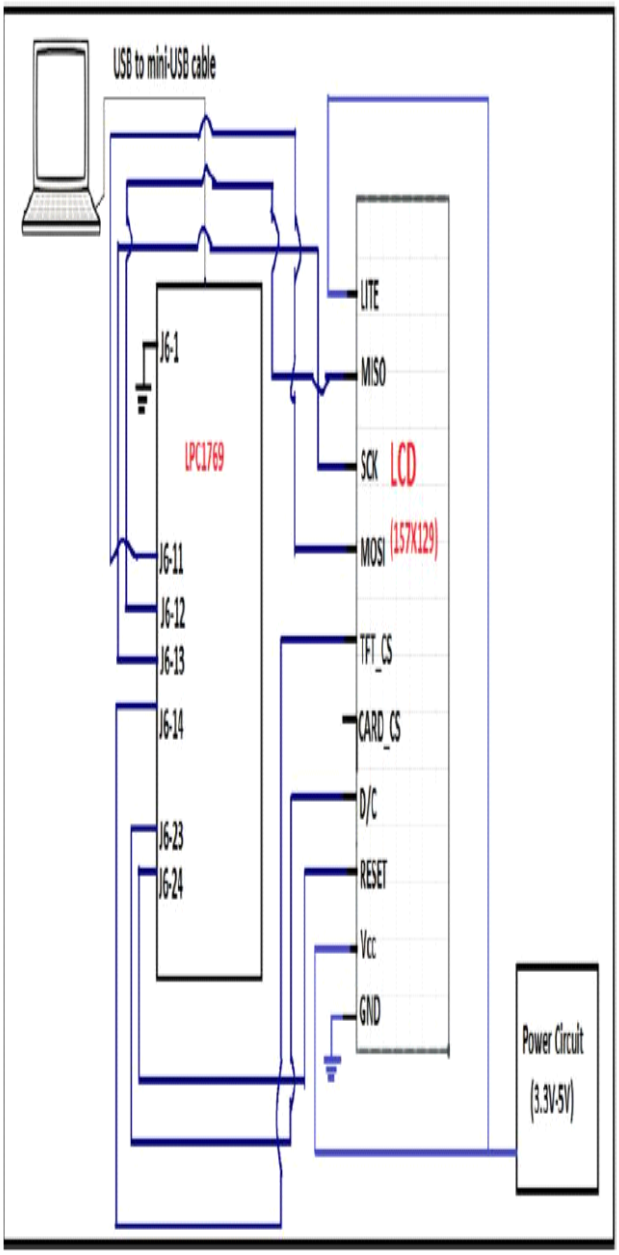



Figure 2 LPC and LCD interface

The power circuit and the interface forms the complete module as shown below.

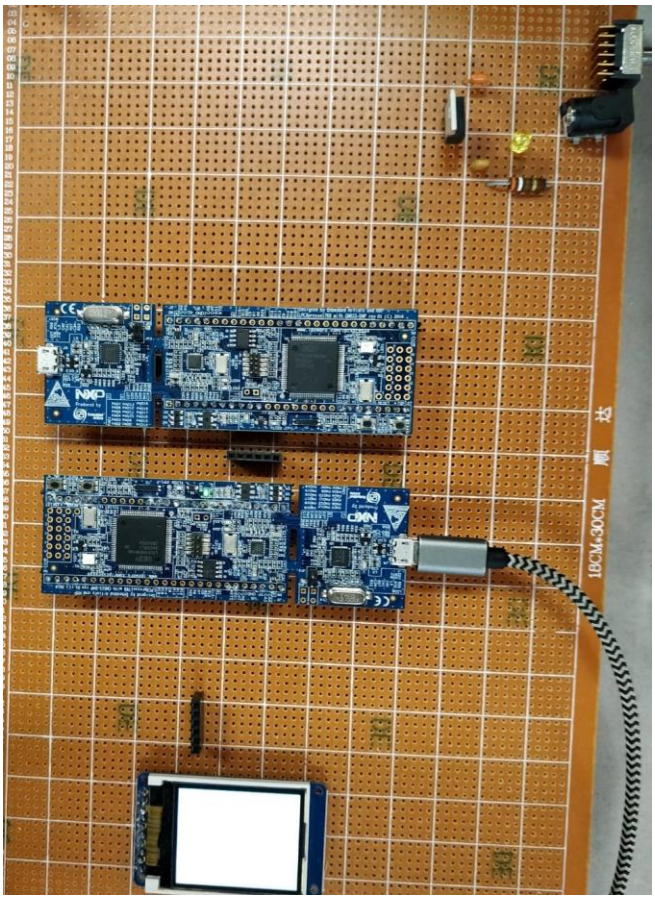



Figure3: Working Board

After the connection establishment, voltage verification is done using Multimeter. LCD light up is checked by connecting computer to mini USB.

**2.2.2 Software Requirement**

To implement this project, we have used MCUXpresso IDE version to debug the program. We have designed and dump the code on LPC 1769 using MCUXpresso. We import all the module on the IDE along with GPIO. In this IDE, we create a new project and save in the directory. The required source files are added. After code implementation we debug the code for error free, now the connection is established between the LPC module and the computer and it is detected. We run the program to display the 2D squares and the trees.

### 2.2.3. Formulas used for calculation purpose

In this section we implement the 2D images on the LCD screen.

#### 2.2.3.1 Rotating squares

Initially we draw 4 lines which are at equal distance from each other. We use the below formula to rotate the square.

P(x,y)=P_1(x_1,y_2)+lambda*(P_2(x_2,y_2)- (P_1(x_1,y_1))

Now we reduce the length and iterate 10 times to create rotating squares as shown below.

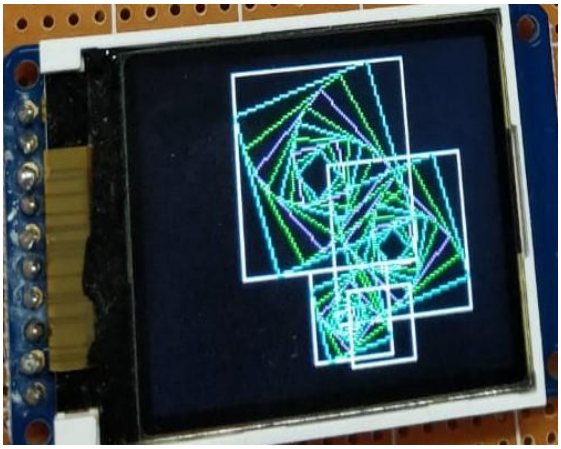


Figure 4 Generation of Random Square

#### 2.2.3.2 Tree

Random point is generated which identifies the coordinates in a tree. Now the angle at which it has to be moved is calculated and the we repeat the process up to desired height. The following shows the tree output

Figure 5 : Generation of Forest tree

## 3.IMPLEMENTATION

In this section we will describe the design of the hardware part for SPI interfacing and also the pseudo codes for the 2D screen savers.

### 3.1 Hardware Design

#### 3.1.1 Power Circuit Design.

The power circuit to power up the entire board is build using LM7805, resistors and capacitors. The input to the switch is the 9V which is generated from adaptor. The first pin of the LM7805 is connected to the switch, the third pin to the capacitor which is in parallel and LCD and 1ohm resistor in series. The output voltage is drawn from LED and given to the circuit to power up.

| | |
|---|---|
| Adaptor | 9V |
| LM7805 | 5V |
| LED | 1.8V |

#### 3.1.2 Bill of material

Following are the materials used in project

| Description | Quantity |
|---|---|
| Wire wrapping board | 1 |
| Wire Wrapping tool kit | 1 |
| Wire Spec of 1500mA | 20 |
| DC power supply | 1 |
| ½ Inch Standoff | 4 |
| RED LED | 1 |
| 1uf Capacitor | 1 |
| LM7805 | 1 |
| Slide switch | 1 |
| Adapter Jack | 1 |
| Resistor 390ohm | 1 |
| LPC module 1769 | 2 |
| LCD | 1 |

### 3.1.3 LCD and LPC interface

The interface is built on the wire wrapping board and is connected to the computer using USB cable.

| | |
|---|---|
| LITE | J6-28 |
| MISO | J6-12 |
| SCK | J6-13 |
| MOSI | J6-11 |
| TFT_CS | J6-14 |
| CARD_CS | X |
| D/C | J6-23 |
| RESET | J6-24 |
| VCC | J6-28 |
| GND | J6-1 |

### 3.2 Software design

This section consists of the algorithm, pseudo code and the flowchart to generate random squares and tress

### 3.2.1 Algorithm for 2D screensavers of rotating squares based on vector graphics

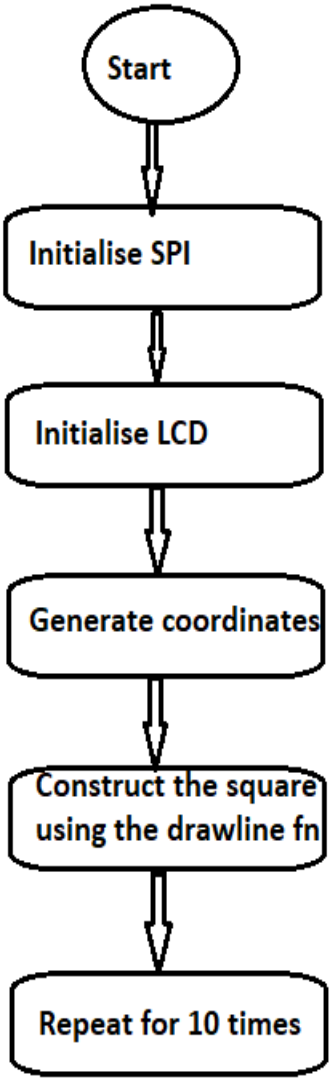
1. Initialise the SPI
- Enable SSP 0 module by ORing 21st bit of PCONP
- SSP_CLK selected as PCLK/4(by writing value 0 to PCLKSEL1)
- Set J6-11 - 14 pins functionality as SSP 0
- Set SSEL0 as GPIO out
- Make 8 bit mode by writing 111 to SSP 0 data width
- SCR register value populate 7.
- Pre scale CLK value is populated as 2

2. Initialise LCD
- Select LCD slave by making SSEL0 as 0
- D/C is connected to P0.21
- RESET is connected to PO.22
- Make both pins as output
- P0.22 pin value is made as logic 1
- Provide sufficient delay(500ms)
- P0.22 pin value is made as logic 0(to bring it outside reset)
- P0.22 pin value is made as logic 1
- Provide sufficient delay(500ms)
- P0.22 pin value is made as logic 0
- Initialize SSP buffer with 0 value
- Make LCD from out of sleep by sending 0x11 to SSP0
- Give sufficient delay for LCD
- To make display on give command 0x29 to SSP0
- Provide sufficient delay

3. Generate a random points X in the range 30 to 90, Y in range 30 to 150

4. Construct a square by using the Drawline function. Now reduce the points to 80% using P(X,Y)=P1(x1,y1)+Lamda P2(x2,y2) and repeat this for 10 more levels.

5. Repeat the whole process to get different screensavers for 10 different colours




Flowchart for generating squares

### 3.2.2 Algorithm for generating Trees

1. Initialise the SPI
- Enable SSP 0 module by ORing 21st bit of PCONP
- SSP_CLK selected as PCLK/4(by writing value 0 to PCLKSEL1)
- Set J6-11 - 14 pins functionality as SSP 0
- Set SSEL0 as GPIO out
- Make 8 bit mode by writing 111 to SSP 0 data width
- SCR register value populate 7.
- Pre scale CLK value is populated as 2

2. Initialise LCD
- Select LCD slave by making SSEL0 as 0
- D/C is connected to P0.21
- RESET is connected to PO.22
- Make both pins as output
- P0.22 pin value is made as logic 1
- Provide sufficient delay(500ms)
- P0.22 pin value is made as logic 0(to bring it outside reset)
- P0.22 pin value is made as logic 1
- Provide sufficient delay(500ms)
- P0.22 pin value is made as logic 0
- Initialize SSP buffer with 0 value
- Make LCD from out of sleep by sending 0x11 to SSP0
- Give sufficient delay for LCD
- To make display on give command 0x29 to SSP0
- Provide sufficient delay

2. Generate 1 random points X,Y
4. Draw a tree trunk using a draw line function .
5. Reduce the tree trunk to 80% using P(X,Y)=P1(x1,y1)+lamda P2(x2,y2) and use rotational algorithm to rotate the 80 % line 30' and -30' which form the branch of the tree.
6. Rotational algorithm includes the rotational matrix
7. Using this algorithm repeat the steps to generate leaves.
8. Once a tree is generated repeat these steps to generate a new tree.

### 3.2.3 Pseudo Code

#### 3.2.3.1 2D screen saver of rotating squares

```
void drawSquare(int16_t x0,int16_t y0,int16_t x1
,int16_t y1,int16_t x2,int16_t y2,int16_t x3,int16_t
                y3,uint32_t color){
               drawLine(x0,y0,x1,y1,color);
               drawLine(x1,y1,x2,y2,color);
               drawLine(x2,y2,x3,y3,color);
               drawLine(x3,y3,x0,y0,color);
                        }
          int main (void)


                        {
                uint32_t pnum = PORT_NUM;

                        pnum = 0 ;

                    if ( pnum == 0 )
                            SSP0Init();

                          else
                    puts("Port number is not correct");

                        lcd_init();

                    fillrect(0, 0, ST7735_TFTWIDTH,
                ST7735_TFTHEIGHT, BLACK);
                    int x0,x1,y0,y1,x2,y2,x3,y3;
                        int j=0;
                      while(1)
                        {
                      int range1=1+70-0;
                      int range2=1+80-0;
                     x0 = rand()%range1;
                    y0 = rand()%range2;
                        x1 = x0;
                       y1 = y0+40;
                       x2=  x1+40;
                        y2=  y1;
                        x3=  x2;
                       y3=  y2-40;
            uint32_t    col1[]={MAGENTA,RED,BLUE,PURPL
E,DARKBLUE,LIGHTBLUE,WHITE,RED,PURPLE,BROW
                        N};
                drawSquare(x0,y0,x1,y1,x2,y2,x3,y3,col1[j]);
                        lcddelay(1000);
                        int t=10;
                    int m0,m1,m2,m3,n0,n1,n2,n3;
                    int c[10]={1,2,3,4,5,6,7,8,9,10};
                        uint32_t
col[]={MAGENTA,RED,BLUE,PURPLE,DARKBLUE,LIG
HTBLUE,WHITE,RED,PURPLE,BROWN};
                        float l=0.2;
                    for(int i=0;i<=9;i++)
                                {
                        m0=x0+l*(x1-x0);
                            n0=y0+l*(y1-y0);
                            m1=x1+l*(x2-x1);
                            n1=y1+l*(y2-y1);
                            m2=x2+l*(x3-x2);
                            n2=y2+l*(y3-y2);
                            m3=x3+l*(x0-x3);
                            n3=y3+l*(y0-y3);

drawSquare(m0,n0,m1,n1,m2,n2,m3,n3,col[i]);
                                lcddelay(1000);
                        x0=m0;
                                y0=n0;
```

```
                x1=m1;
                y1=n1;
                x2=m2;
                y2=n2;
                x3=m3;
                y3=n3;
              }
          j++;
        if(j==9)
          {
            j=0;
          }

      }

    return 0;

      }
```

**3.2.3.2 Trees 2D**

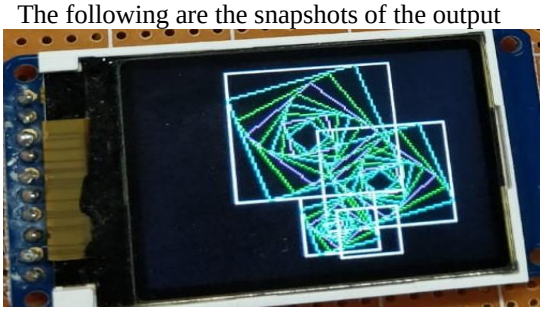
```
    void drawCircle(uint16_t r,uint16_t h,uint16_t k)
                {
              int n=(2*r)+1;
              int m=0,m1=0;
            for(int i=0;i<n;i++)
                {
              for(int j=0;j<n;j++)
                    {
                     m=h-r+i;
                    m1=k-r+j;
             if((((m-h)(m-h)) +((m1-k)(m1-k)))
           <= (r*r)+1)
                          {

        drawPixel(m,m1,WHITE);
                          }
                    }
                }
              }


void translateLine(int startx, int starty,int endx,int endy,int
          angle,uint32_t color){


          float deltax=endx-startx;
          float deltay=endy-starty;

      float alpha=angle * 3.141592653589/180;
          int resultx=0;
          int resulty=0;

        resultx = startx + ((deltax*cos(alpha))-
            (deltay*sin(alpha)));
resulty = starty + ((deltax*sin(alpha))+ (deltay*cos(alpha)));
drawLine(startx, starty, resultx, resulty, color);
          x_cor[tree_length]=resultx;
          y_cor[tree_length]=resulty;
              tree_length++;
resultx = startx + ((deltax*cos(-alpha))- (deltay*sin(-
              alpha)));
resulty = starty + ((deltax*sin(-alpha))+ (deltay*cos(-
              alpha)));
drawLine(startx, starty, resultx, resulty, color);
          x_cor[tree_length]=resultx;
          y_cor[tree_length]=resulty;
              tree_length++;
                }


          int main (void)
                {
        uint32_t pnum = PORT_NUM;


              pnum = 0 ;


          if ( pnum == 0 )
                SSP0Init();


              else
        puts("Port number is not correct");


              lcd_init();

        fillrect(0, 0, ST7735_TFTWIDTH-50,
      ST7735_TFTHEIGHT,BROWN );
        fillrect(ST7735_TFTWIDTH -
50,0,ST7735_TFTWIDTH,ST7735_TFTHEIGHT,BLUE);
        drawCircle(7,110,140);
              srand(time(0));
              int xxr[1024];
              int yyr[1024];
              int adder=20;
            int distance=50;
              float val=0.2;
            int range1=1+50-0;
            int range2=1+50-0;
      int       x0 = rand()%range1;
      for(int mk=0;mk<3;mk++){
            int  y0 = rand()%range2;
              if(x0<20){
                  x0=x0+30;
                  }
              if(y0<20){
                y0=y0+100;
                  adder=60;
                distance=30;
                  }
            else{
                  adder=20;
                distance=50;
                  }
```

```
int dewx=x0;
int dewy=y0-adder;
int x1=x0;
int y1=y0+distance;
drawLine(dewx, dewy, x0,y0, BROWN);
drawLine(x0, y0, x1,y1, BROWN);
lcddelay(500);
drawLine(dewx-1, dewy, x0-1,y0, BROWN);
drawLine(dewx-2, dewy, x0-2,y0, BROWN);
drawLine(dewx+1, dewy, x0+1,y0, BROWN);
drawLine(dewx+2, dewy, x0+2,y0, BROWN);
drawLine(x0-1, y0, x1-1,y1-5, BROWN);
drawLine(x0-2, y0, x1-2,y1-5, BROWN);
drawLine(x0+1, y0, x1+1,y1-5, BROWN);
drawLine(x0+2, y0, x1+2,y1-5, BROWN);
int xr=x0+(val*(x1-x0));
int yr=y0+(val*(y1-y0));
int angle=30;
translateLine(xr, yr, x1, y1, angle, BROWN);
int score=0;
for(int tt=0;tt<size;tt++){
xxr[tt]=xr+(val*(x_cor[tt]-xr));
yyr[tt]=yr+(val*(y_cor[tt]-yr));
translateLine(xxr[tt], yyr[tt], x_cor[tt],
y_cor[tt], angle, GREEN);
if(tt%2!=0){
xr=xxr[score];
yr=yyr[score];
score++;

}
}
x0=x0+30;
tree_length=0;
}
return 0;

}
```

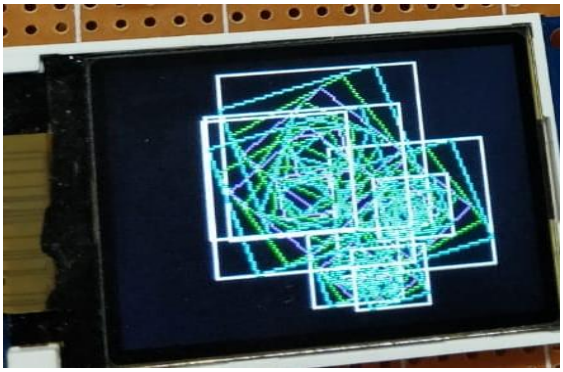
## 4.Testing and Verification

Hardware is tested for proper functioning and the LPC board is verified. Ensured all the Soldering are intact. The code is debugged and executed.

The following are the snapshots of the output



Generation of Random rotating squares



Generation of more Random squares

Generation of a Tree

## 5.Conclusion

The 2D vector graphics processing engine design was designed and tested successfully. The image is displayed on the LCD display on running the code using hardware connecting the LCD module with the LPCXpresso 1769.

## 6.Acknowledge

The work described in this paper was made possible by the contribution of Dr.Harry Li. Thorough discussion on LPC,IDE and LCD played a role in this project completion.

## 7.References

[1] H. Li. Author, “Guidelines for CMPE 240 Project Report” Lecture notes of CMPE 240, Computer Engineering Department, College of Engineering, San Jose State University, March6, 2006, pp.1

[2] http://www.adafruit.com/products/358

[3] LPCXpresso 1769 datasheet. http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf

[4] https://learn.adafruit.com/1-8-tft-display.

## 8.Appendix

8.1 Source code for the entire Project

```
/*
===============================================
================================
 Name        : DrawLine.c
 Author      : $Yashaswi
 Version     :
 Copyright   : $(copyright)
 Description : main definition
===============================================
================================
 */

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "LPC17xx.h"
/* LPC17xx definitions */
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>



/* Be careful with the port number and
location number, because

some of the location may not exist in that
port. */

#define PORT_NUM            0


uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];


#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29



#define swap(x, y) {x = x + y; y = x - y; x =
x - y ;}

// defining color values

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF
#define BROWN 0X8B4513
#define TREEBROWN 0XDEB887


const int tree_depth= 5;
int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;

const int size=1024;
int x_cor[2048];
int y_cor[2048];
int tree_length=0;


void spiwrite(uint8_t c)

{
      int pnum = 0;
      src_addr[0] = c;
      SSP_SSELToggle( pnum, 0 );
      SSPSend( pnum, (uint8_t *)src_addr, 1
);
      SSP_SSELToggle( pnum, 1 );
}



void writecommand(uint8_t c)

{
    LPC_GPIO0->FIOCLR |= (0x1<<21);
      spiwrite(c);
}



void writedata(uint8_t c)

{
      LPC_GPIO0->FIOSET |= (0x1<<21);
      spiwrite(c);
}


void writeword(uint16_t c)

{
      uint8_t d;
```

```
	d = c >> 8;
	writedata(d);
	d = c & 0xFF;
	writedata(d);
}



void write888(uint32_t color, uint32_t
repeat)

{

	uint8_t red, green, blue;

	int i;
	red = (color >> 16);
	green = (color >> 8) & 0xFF;
	blue = color & 0xFF;
	for (i = 0; i< repeat; i++) {
		writedata(red);
		writedata(green);
		writedata(blue);
	}

}
void drawPixel(int16_t x, int16_t y, uint32_t
color)

{

	if ((x < 0) || (x >= _width) || (y < 0)
|| (y >= _height))
		return;
	setAddrWindow(x, y, x + 1, y + 1);
	writecommand(ST7735_RAMWR);
	write888(color, 1);

}
void setAddrWindow(uint16_t x0, uint16_t y0,
uint16_t x1, uint16_t y1)

{

	writecommand(ST7735_CASET);
	writeword(x0);
	writeword(x1);
	writecommand(ST7735_RASET);
	writeword(y0);
	writeword(y1);
}


void fillrect(int16_t x0, int16_t y0, int16_t
x1, int16_t y1, uint32_t color)

{

	int16_t width, height;
	width = x1-x0+1;
	height = y1-y0+1;
	setAddrWindow(x0,y0,x1,y1);
	writecommand(ST7735_RAMWR);
	write888(color,width*height);
}



void lcddelay(int ms)

{
	int count = 24000;
	int i;
	for ( i = count*ms; i--; i > 0);

}



void lcd_init()

{

	int i;
	printf("LCD Demo Begins!!!\n");
	// Set pins P0.16, P0.21, P0.22 as
output
	LPC_GPIO0->FIODIR |= (0x1<<16);

	LPC_GPIO0->FIODIR |= (0x1<<21);

	LPC_GPIO0->FIODIR |= (0x1<<22);

	// Hardware Reset Sequence
	LPC_GPIO0->FIOSET |= (0x1<<22);
	lcddelay(500);

	LPC_GPIO0->FIOCLR |= (0x1<<22);
	lcddelay(500);

	LPC_GPIO0->FIOSET |= (0x1<<22);
	lcddelay(500);

	// initialize buffers
	for ( i = 0; i < SSP_BUFSIZE; i++ )
	{
	  src_addr[i] = 0;
		dest_addr[i] = 0;
	}

	// Take LCD display out of sleep mode
	writecommand(ST7735_SLPOUT);
```

```
	lcddelay(200);

	// Turn LCD display on
	writecommand(ST7735_DISPON);
	lcddelay(200);
}
/*****************************************
*******************************

 ** Descriptions:		Draw line function

 **

 ** parameters:		Starting point
(x0,y0), Ending point(x1,y1) and color

 ** Returned value:		None

 **

*****************************************
*******************************/

void drawLine(int16_t x0, int16_t y0, int16_t
x1, int16_t y1, uint32_t color)

{

	int16_t slope = abs(y1 - y0) > abs(x1 -
x0);

	if (slope) {

		swap(x0, y0);

		swap(x1, y1);

	}

	if (x0 > x1) {

		swap(x0, x1);

		swap(y0, y1);

	}

	int16_t dx, dy;

	dx = x1 - x0;

	dy = abs(y1 - y0);

	int16_t err = dx / 2;

	int16_t ystep;

	if (y0 < y1) {

		ystep = 1;

	}

	else {

		ystep = -1;

	}

	for (; x0 <= x1; x0++) {

		if (slope) {

			drawPixel(y0, x0, color);

		}

		else {

			drawPixel(x0, y0, color);

		}

		err -= dy;

		if (err < 0) {

			y0 += ystep;

			err += dx;

		}

	}

}


/*

 Main Function main()

 */

void drawSquare(int16_t x0,int16_t y0,int16_t
x1,int16_t y1,int16_t x2,int16_t y2,int16_t
x3,int16_t y3,uint32_t color){
	drawLine(x0,y0,x1,y1,color);
	drawLine(x1,y1,x2,y2,color);
```

```
	drawLine(x2,y2,x3,y3,color);
	drawLine(x3,y3,x0,y0,color);
}


void drawTree(int startx, int starty,int
endx,int endy,int angle,uint32_t color)
{
	float deltax=endx-startx;
	float deltay=endy-starty;
	float alpha=angle * 3.141592653589/180;
	int resultx=0;
	int resulty=0;
	resulty = starty +
((deltax*sin(alpha))+ (deltay*cos(alpha)));
    resultx = startx + ((deltax*cos(alpha))-
(deltay*sin(alpha)));
	drawLine(startx, starty, resultx,
resulty, color);
	x_cor[tree_length]=resultx;
	y_cor[tree_length]=resulty;
	tree_length++;
	resultx = startx + ((deltax*cos(-
alpha))- (deltay*sin(-alpha)));
	resulty = starty + ((deltax*sin(-
alpha))+ (deltay*cos(-alpha)));
	drawLine(startx, starty, resultx,
resulty, color);
	x_cor[tree_length]=resultx;
	y_cor[tree_length]=resulty;
	tree_length++;
}


#if 1
int main (void)

{

	uint32_t pnum = PORT_NUM;

	pnum = 0 ;

	if ( pnum == 0 )
		SSP0Init();

	else
		puts("Port number is not
correct");

	lcd_init();

	fillrect(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT,TREEBROWN );

	srand(time(0));
	int xxr[1024];
	int yyr[1024];
	int adder=20;
	int distance=50;
	float val=0.2;
	int range1=1+50-0;
	int range2=1+50-0;
	int     x0 = rand()%range1;



	for(int mk=0;mk<40;mk++){
		int  y0 = rand()%range2;
	if(x0<20){
		x0=x0+30;
	}
	if(y0<20){
		y0=y0+100;
		adder=60;
		distance=30;
	}
	else{
		adder=20;
		distance=50;
	}
	   int dewx=x0;
	   int dewy=y0-adder;

	int x1=x0;
	int y1=y0+distance;
	drawLine(dewx, dewy, x0,y0, BROWN);
	drawLine(x0, y0, x1,y1, BROWN);
	lcddelay(100);
	drawLine(dewx-1, dewy, x0-1,y0, BROWN);
	drawLine(dewx-2, dewy, x0-2,y0, BROWN);
	drawLine(dewx+1, dewy, x0+1,y0, BROWN);
	drawLine(dewx+2, dewy, x0+2,y0, BROWN);
	drawLine(x0-1, y0, x1-1,y1-5, BROWN);
	drawLine(x0-2, y0, x1-2,y1-5, BROWN);
	drawLine(x0+1, y0, x1+1,y1-5, BROWN);
	drawLine(x0+2, y0, x1+2,y1-5, BROWN);
	int xr=x0+(val*(x1-x0));
	int yr=y0+(val*(y1-y0));
	int angle=30;
	drawTree(xr, yr, x1, y1, angle, BROWN);
	int score=0;

		int tt=0;
		while(tt<size){
		 xxr[tt]=xr+(val*(x_cor[tt]-
xr));
		 yyr[tt]=yr+(val*(y_cor[tt]-
yr));
		 drawTree(xxr[tt], yyr[tt],
x_cor[tt], y_cor[tt], angle, GREEN);
		if(tt%2!=0){
```

```
                    xr=xxr[score];
                    yr=yyr[score];
                    score++;

             }
             tt++;
       }
       x0=x0+10;
       tree_length=0;
       }
       return 0;

}

#endif

#if 0
int main (void)

{

       uint32_t pnum = PORT_NUM;

       pnum = 0 ;

       if ( pnum == 0 )
             SSP0Init();

       else
             puts("Port number is not
correct");

       lcd_init();
    fillrect(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, BLACK);
       uint32_t
my_colors[]={RED,GREEN,RED,BLUE};
    int chan=0;
       int x0,x1,y0,y1,x2,y2,x3,y3;
       int m0,n0,m1,n1,m2,n2,m3,n3;
       float l=0.2;
       for(i=0;i<40<i++)
       {

             int s0=rand()%30;
             int s1=rand()%70;
             int s2=10+rand()%50;
             x0= s0;
             y0=s1;

             x1=x0;
             y1=y0+s2;
             x2=x0+s2;
             y2=y0+s2;
             x3=x0+s2;
             y3=y0;

       drawSquare(x0,y0,x1,y1,x2,y2,x3,y3,WHIT
E);
             lcddelay(100);
             for(int i=0;i<9;i++)
             {
                    m0=x0+l*(x1-x0);
                    n0=y0+l*(y1-y0);
                    m1=x1+l*(x2-x1);
                    n1=y1+l*(y2-y1);
                    m2=x2+l*(x3-x2);
                    n2=y2+l*(y3-y2);
                    m3=x3+l*(x0-x3);
                    n3=y3+l*(y0-y3);

       drawSquare(m0,n0,m1,n1,m2,n2,m3,n3,my_c
olors[i%4]);
                    lcddelay(10);
                    x0=m0;
                    y0=n0;
                    x1=m1;
                    y1=n1;
                    x2=m2;
                    y2=n2;
                    x3=m3;
                    y3=n3;
             }
       }

return 0;
}

#endif
```