Implementation of 2D Graphics Engine Using LPC1769 CPU Modules

Neeraj Venugopal (011541907)
Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 95112
E-mail: neeraj.venugopal@sjsu.edu

Abstract

The Following project aims at displaying 2-Dimensional images on Liquid Crystal Display Module by interfacing it with an LPCXpresso 1769 Micro Controller. In this report we share our experience in achieving the above task.
First we implemented a line by joining 2 points using the source code provided by the graphic library of Adafruit LCD display. Then we built a square by joining 4 Lines and rotated each of the squares up to 9 levels by reducing its length by 20 percent per iteration. In the end we conclude the project by showing a tree image on the LCD display. This report also describes the hardware and software requirements in order to establish the interface.

1. Introduction

The LPC1769 is an ARM Cortex-M3 based microcontroller for embedded applications requiring a high level of integration and low power dissipation. This module has USB, UART, SPI, I2C ports that helps us to communicate with it effectively. We make use of SPI protocol to communicate with the LCD module. In our project we are going to use SPI port 0 in both the program and the hardware connections for communication purpose. Figure 1 portrays the same.
The LPC1769 module is connected to the CPU via USB cable which powers up the module. The LCD is connected to this port by connecting the appropriate pins with the SPI port that was used earlier (port 0) which we are using for communication. When an USB is connected to the LPC 1769 module the LCD lights up, portraying that the connections made are correct and the Module is ready for communications with the LCD.

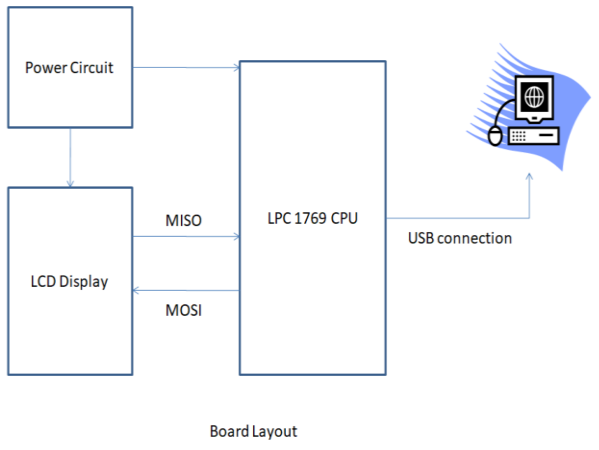

Figure 1: Block Diagram of the System Setup

2. Methodology

First we build the entire circuit on the wire wrapping board. This interface is used by the CPU to send a data buffer to the module. The data is sent in the form of an opcode and dummy bytes to the external LCD which is connected to the LPC 1769 module. The opcode and dummy bytes initialize the LCD. The module then sends the data and opcode which is received from the CPU to the LCD to display the required 2 Dimensional images such as a line, square, tree, background color etc. The Communication between the LCD and the LPC module happens serially. The LCD is connected properly to the CPU module to complete the communication. The SPI port of module and LCD must be defined and initialized properly.

2.1. Objectives and Technical Challenges

Following are the objectives and the challenges that we would be approaching in our project:

1. To build an interface circuit efficiently on a wire wrapping board using the external LCD.
2. Interface the GPIO pins.
3. To build a power circuit using a voltage regulator to output 5V, to drive the entire circuit.
4. To develop a program which initializes the LCD and perform all the tasks associated with it.
5. To develop a program to display a Line Segment on the LCD display.
6. To develop a program to create and display 10 squares of different sizes and rotate the squares by reducing the lengths of it by 20 percent at each iteration.
7. To develop a program to display a tree and convert it to display a thick forest.

2.2. Problem Formulation and Design

The Project can be broadly divided into 2 parts, one being the hardware and the other being the software requirements. Both of these topics are explained below:

2.2.1. Hardware Requirements

In the hardware requirements, the power circuit is built on the wire-wrapping board using the resistors, capacitors, LED (Light Emitting Diode), switch and LM 7805 voltage regulator IC. The circuit diagram for the same is shown below in figure 2. The output voltage is checked across the LED and the Ground using a digital multi meter and ensured that the required output voltage is achieved.

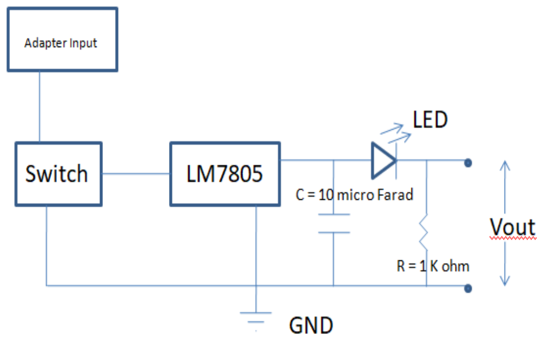

Figure 2: Power Circuit Block Diagram

After the power output is checked we then build the interface between the LCD and the CPU module. Figure 3 is a block diagram portraying these connections. Figure 3 also shows the port numbers and the pin numbers that are used for the sake of connections.

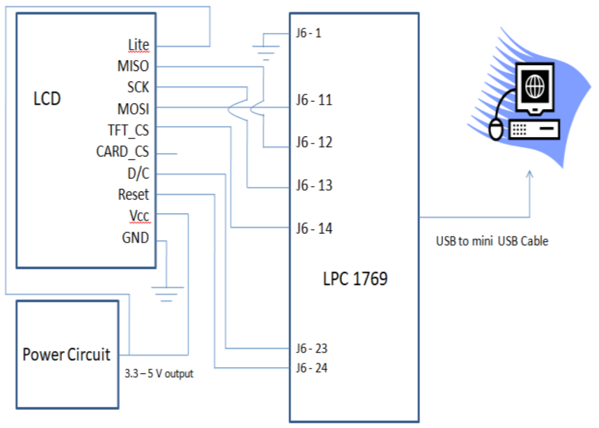

Figure 3: Block Diagram with all the Pin Number. (SPI Connections)

When we join the Figure 2 and Figure 3 we get the complete circuit and the module. Figure 4 shown below is the complete setup of the System on the wire wrapping board. Note the following is an actual picture of the circuit that has been built. The Second LPC module is not currently used. It has been affixed for the next project. As of now only one LPC module is being used.

A fair amount of space has been left for Flash Memory Debugging purpose. This part will be implemented during the second phase of the project.

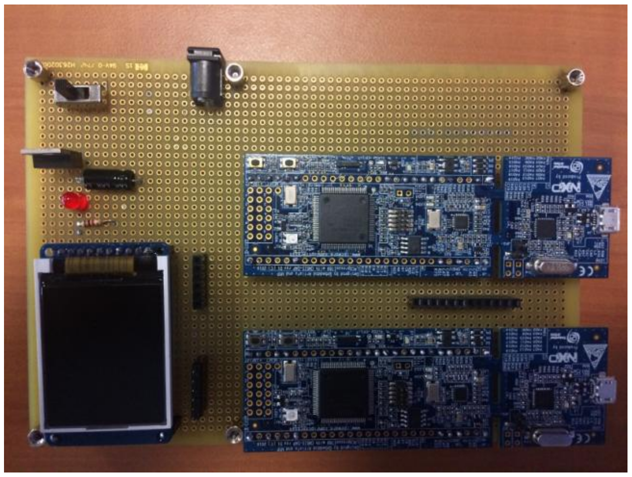

Figure 4: Complete Circuit on the wire wrapping board

Once all these connections are established a thorough check is done with the digital multi meter. Before implementing the software side the USB from the Computer is connected to Mini USB port using a USB 2.0 cable to check if the LCD lights up. Figure 5 shows that the circuit is ready for debugging and we can move on with the software implementation.

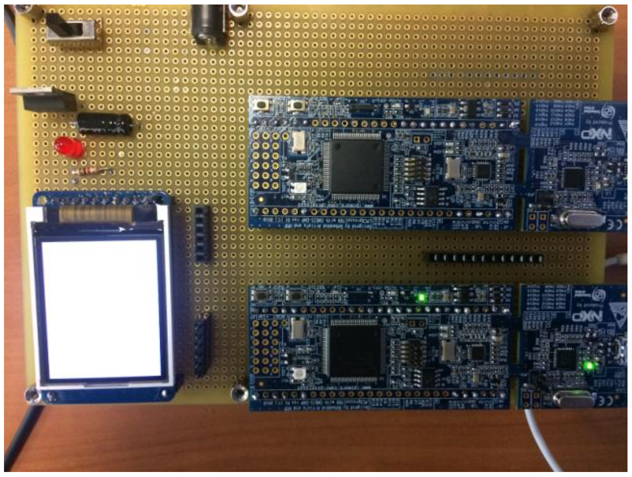

Figure 5: LCD glowing as soon as the USB is connected

2.2.2. Software Requirements

We need to build the code, debug the code and dump it onto the LPC 1769 module. For this purpose we use MCUXpresso IDE version - v10.1.1_606. The Software requirements involves initializing the LCD module, usage of Adafruit library for transmission of data buffer to LCD and polling the GPIO pins for external interrupt.
We make use of a drawline program which is already implemented by the graphic library of Adafruit LCD display. We import this module onto the IDE along with

the GPIO project and LPC1769 patch that has been provided. After importing all these projects the build all function is used in IDE. Once the all the project is built we then place a connection between the LPC Module and the Computer. The system then detects the LPC Module.
After the Module has been detected we do a debug of the code. Figure 6 shows the output attained when the drawline project has been dumped on to the Microcontroller. With this we are ready to dig in to the main section of building the code and get our desired output that is to display 2-Dimensional images on our LCD screen.

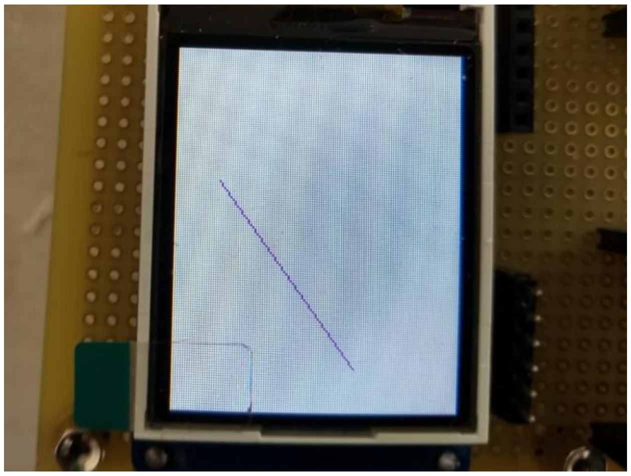

Figure 6: Displaying the draw line project by using the graphic library of Adafruit LCD display

2.2.3. Formulas used for calculation purpose

With the hardware and the software in place we are ready to move forward with our implementation part of the 2 Dimensional images on the LCD screen. We use certain 2 Dimensional vector concepts to attain our main objective. Below are the plans to design the patterns such as rotating squares and trees.

2.2.3.1. Rotating Squares

Firstly a square is connection of 4 lines that are equidistant from each other. Hence we first draw four lines on the LCD starting on any coordinates with any size. For the rotation part we use a 2D vector concept with the below formula.
P(x, y) = P_1(x_1, y_1) + lambda * (P_2(x_2,y_2) - P_1(x_1,y_1))
We try to decrease the length for each iteration by 20% hence making lambda to be as 0.8.
Once the points of the 4 coordinate moves the draw line function is called each time for these points. This would hence seem to be like rotating. This is carried up to 10 levels and then repeated for 10 different sets of squares.

Figure 7 shown is an example of one such square.

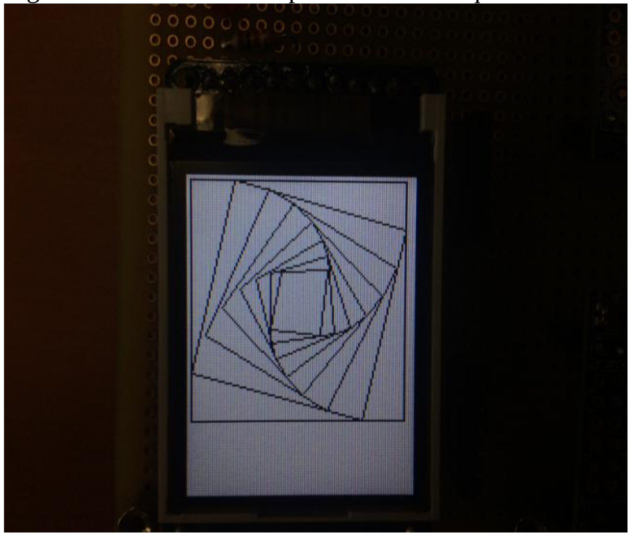

Figure 7: Displaying a single square that is made to rotate up to 10 levels.

2.2.3.2. Tree

Same formula is being used to generate the next coordinates in a tree. The difference here is to the number of levels that is being built and the angle at which each branch has been moved to. We then repeat this process to build up to the desired height.
The same function is called upon for spreading the tree throughout the entire display to make it look like a forest.
Figure 8 shows a sample tree that has been implemented with a sky on the back ground.

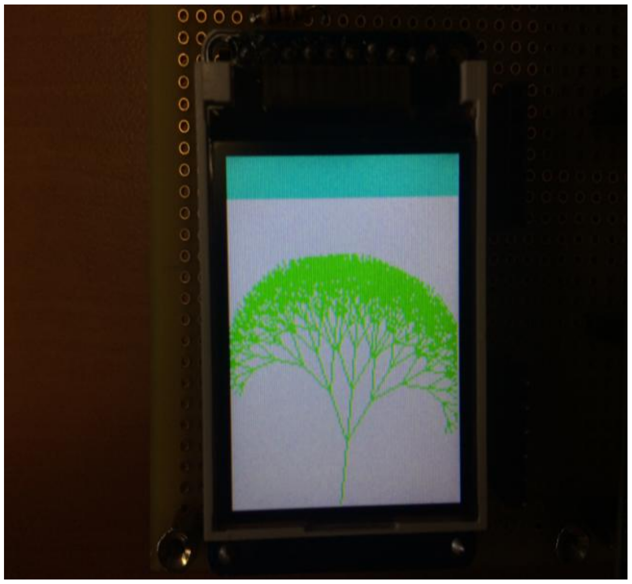

Figure 8: Display of a sample single tree.

3. Implementation

This section describes the design and implementation of hardware part for SPI Interfacing. The entire design may be classified into sections that are described in the Hardware Design. The section also provides the algorithms and pseudo codes used for the implementation of the Rotating Squares and the Forest.

3.1. Hardware Design

3.1.1. Power Circuit Design

The power circuit should be designed such that we receive an output voltage of 3.3 to 5 Volts from the circuit that is shown in Figure 2.
The input to the power circuit is a 110V AC that is taken via an adapter that outputs 9V adapter to the first pin of the switch. The Switch is then connected to the first pin of the LM7805 power regulator. The third pin of the LM7805 voltage regulator is connected to a capacitor in parallel and a Light Emitting diode which is in series with a 1 Kilo ohm resistor. Then the desired voltage output is drawn from the LED and fed to the entire module. Table 1 shows the power unit specifications.

Power Connector	Connected to external Power Source
Wall Mount Power	9V DC 5V DC 1500mA
LM 7805	Voltage Regulator 5V
LED (Red)	Power Indicator -8mA, 1.8V

Table 1: Power Unit Specifications

3.1.2. Interfacing LCD module with LPC

The serial interface circuit is built on the wire wrapping board. It consists of an LCD and LPC1769 module. The wire wrapping board is connected to the computer using the USB cable. The detailed design and connections are shown in Figure 3. An entire circuit for is shown above in Figure 4. A pin connectivity table for the LCD and the LPC module is show in Table 2.

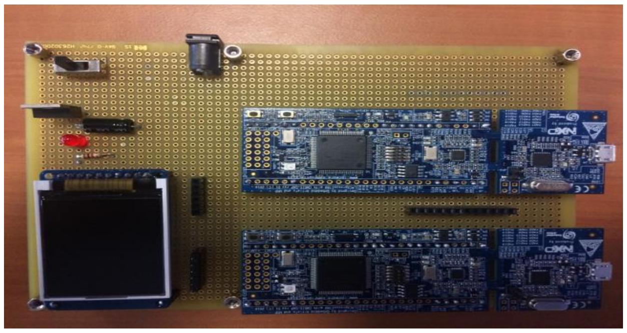

Figure 3: Shown again, the entire circuit Connections

LCD Module	LPC 1769	Description
LITE	J6-28 pin or Vout from power circuit	Accepted Voltage range from 3.3 to 5 V
MISO	J6 - 12	Master in Slave Out
SCK	J6 - 13	Serial Clock
MOSI	J6 - 11	Master Out Slave In
TFT_CS	J6 - 14	Slave Select
CARD_CS	X	Not Connected
D/C	J6 - 23	GPIO
RESET	J6 - 24	Reset pin
VCC	J6 – 28 pin or Vout from power circuit	Accepted voltage range 3.3 to 5V
GND	J6-1	Ground Pin

Table 2: Pin Connection of LPC and LCD Display

3.1.3. Bill of material

The materials used for the project are shown in the Table 3.

Description	Quantity
Wire Wrapping Board	1
Wire Wrapping Tool Kit	1
Wire Spec of 1500mA	20
DC Power Supply(Adapter)	1
½ Inch Stand Off's	6
Red LED	1
10 µF Capacitor	1
LM7805	1
Slide Switch	1
Adapter Jack	1
Resistor 1 Kilo Ohm	1
LPC Module 1769	2
LCD	1

Table 3: Bill of Materials

3.2. Software Design

As illustrated above in the software requirements part we first build the codes for the provided library. Then we implement our Squares and trees step by step. After implementing these squares and trees, it is then rotated, placed at random location and keeps generating till the keyboard interrupt has been pressed. The Algorithms, flowchart and pseudo code for building these modules are discussed in this section of the report.

3.2.1. Algorithms

There are few algorithms that have been implemented for this project. The first being the rotate square logic, that is to rotate a square by reducing its length by 20 percent in each iteration. The second being the deep forest emergence. The following algorithms are discussed in detail in this section of the report.

3.2.1.1. Rotating Square

1. Start
2. Initialize the SPI ports
3. Initialize the LCD with a white background
4. Generate Coordinates
5. Join these Coordinates using the drawline function.
6. The output received from step 5 would be a square. Reduce the length by 2 dimensional Matrix methods, as discussed in section **2.2.3.1.**
7. Join these new coordinates, hence making it look like a rotation.
8. Perform step 6 to step 7 up to 10 levels.
9. Perform the above 8 steps again for a different set of colors and the coordinates at least 10 times.

The Above logic can also be followed for generating squares at any desired location of users choice.

3.2.1.2. Generating Forest

1. Start
2. Initialize the SPI ports.
3. Initialize the LCD to a dark Back ground.
4. Generate the number of branch levels and the angle of the branches using random function.
5. Generate the coordinates of the tree's initial position.
6. Call the growTree Function with the attained parameters.
7. Use the drawline functionality to draw the first branch.
8. Calculate the next coordinates using the formulas discussed in section **2.2.3.2.**
9. Recursively call the treeGrow Function by decrementing the levels and drawing the corresponding lines using draw line function.
10. When the levels are between 0 to 3, change the color to Green to portray the leaves of the trees on the LCD.
11. Repeat these steps again and again by putting into an infinite loop.
12. This generates a series of trees at random location with random levels and angles.
13. When a keyboard interrupt is hit the loop must exit and the output in the LCD should not be erased.

3.2.2. Flowchart

Flowchart is the descriptive method of showing the logic implemented in a project. In this section we will show the rotating square algorithm in Figure 9 and the generating forest algorithm as in Figure 10 flowcharts.

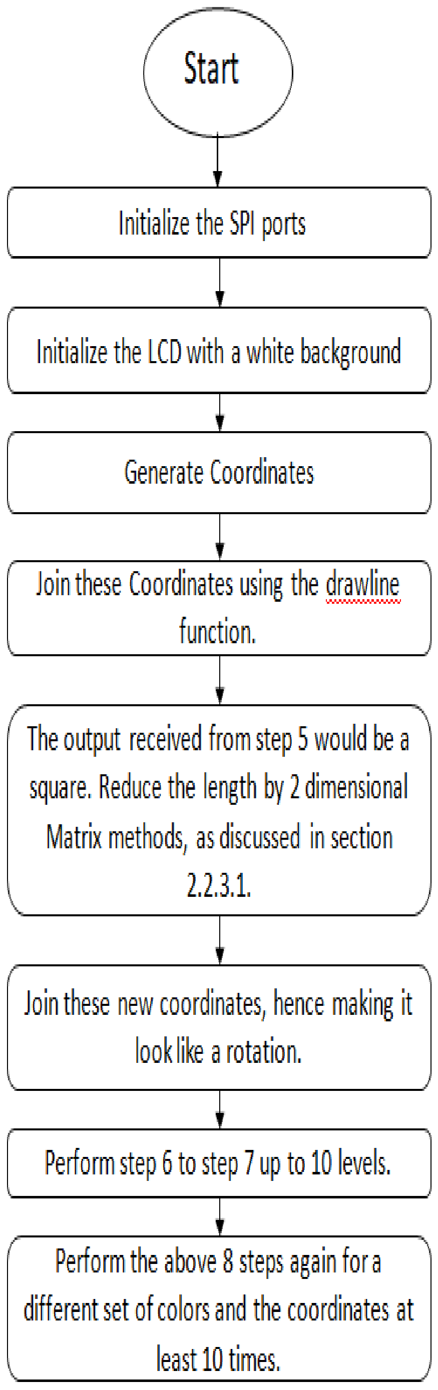
Start
Initialize the SPI ports
Initialize the LCD with a white background
Generate Coordinates
Join these Coordinates using the drawline function.
The output received from step 5 would be a square. Reduce the length by 2 dimensional Matrix methods, as discussed in section 2.2.3.1.
Join these new coordinates, hence making it look like a rotation.
Perform step 6 to step 7 up to 10 levels.
Perform the above 8 steps again for a different set of colors and the coordinates at least 10 times.

Figure 9: Flowchart for Rotating Square.

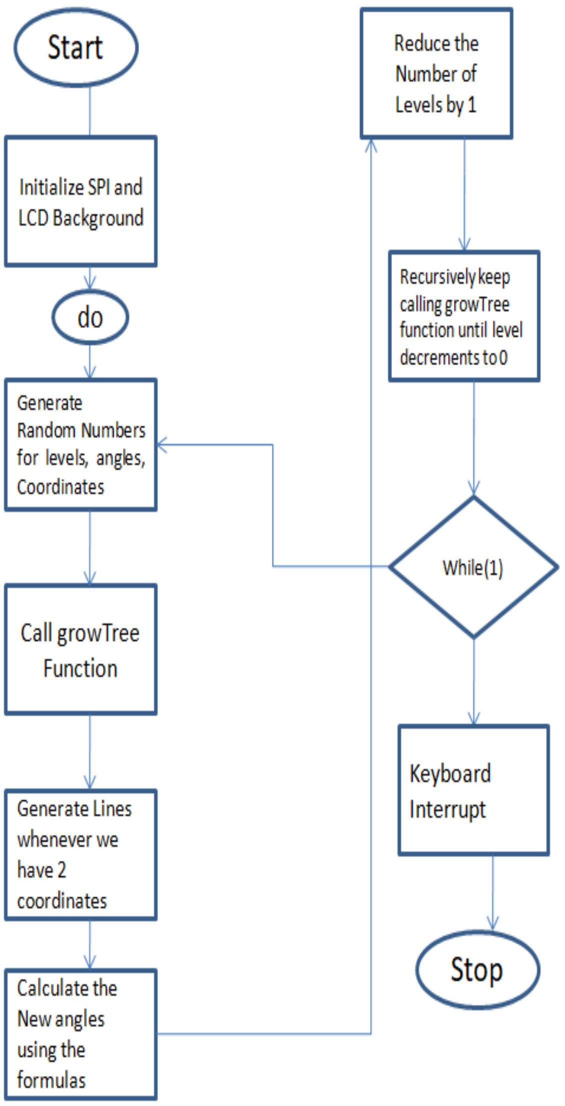
Start
Initialize SPI and LCD Background
do
Generate Random Numbers for levels, angles, Coordinates
Call growTree Function
Generate Lines whenever we have 2 coordinates
Calculate the New angles using the formulas
Reduce the Number of Levels by 1
Recursively keep calling growTree function until level decrements to 0
While(1)
Keyboard Interrupt
Stop

Figure 10: Flowchart for Forest Emergence.

3.2.3. Pseudo codes

The transformation of an algorithm to a general programming language is known as a pseudo code. Below are the actual function written in our Project.

3.2.3.1. Rotating Square

Below is the function used for rotating the squares

```
void rotating_mysquare1(int x1, int x2, int x3, int x4,
int y1, int y2, int y3, int y4,int color1, float lambda)
{
int q1x=0, q2x=0, q3x=0, q4x=0,  q1y=0,q2y=0, q3y=0,
q4y=0;
int i;
        for(i=1;i<=11;i++){
                lcddelay(200);
                        q1x = (x2+(lambda*(x1-x2)));
                        q1y = (y2+(lambda*(y1-y2)));

                        q2x = (x3+(lambda*(x2-x3)));

                        q2y = (y3+(lambda*(y2-y3)));
                        q3x = (x4+(lambda*(x3-x4)));


                        q3y = (y4+(lambda*(y3-y4)));
                        q4x = (x1+(lambda*(x4-x1)));
                        q4y = (y1+(lambda*(y4-y1)));

                drawLine(q1x,q1y,q2x,q2y,color1);
                drawLine(q2x,q2y,q3x,q3y,color1);
                drawLine(q4x,q4y,q3x,q3y,color1);
                drawLine(q1x,q1y,q4x,q4y,color1);
                        x1 = q1x;
                        x2 = q2x;
                        x3 = q3x;
                        x4 = q4x;

                        y1 = q1y;
                        y2 = q2y;
                        y3 = q3y;
                        y4 = q4y;
                }
}
```

3.2.3.2. Grow Tree

Below is the function to generate a forest. Note this function is in an infinite while loop. Hence generating a forest.

```
void growTree(int x0, int y0, float angle, int length, int
level, int color){

        int x1, y1, length1;
        float angle1;

                if (level <= 2)
                        color = GREEN;
                if(level>0){
                x1 = x0+length*cos(angle);
          y1 = y0+length*sin(angle);
          drawLine(x0,y0,x1,y1,color);
          angle1 = angle + 0.52;

          length1 = 0.8 * length;
          growTree(x1,y1,angle1,length1,level-1,color);

          angle1 = angle - 0.52;
          length1 = 0.8 * length;
          growTree(x1,y1,angle1,length1,level-1,color);
```

```
                angle1 = angle;
                length1 = 0.8 * length;
                growTree(x1,y1,angle1,length1,level-1,color);
            }
}
```

4. Testing and Verification

The connectivity of the hardware part is checked using a digital multi-meter keeping on connectivity mode. Similarly, the working on LPC board is checked using hello world program and then the rest of LCD work is done.

A Series of photos have been pasted to below that shows that the output we received is accurate and is as per project requirements.

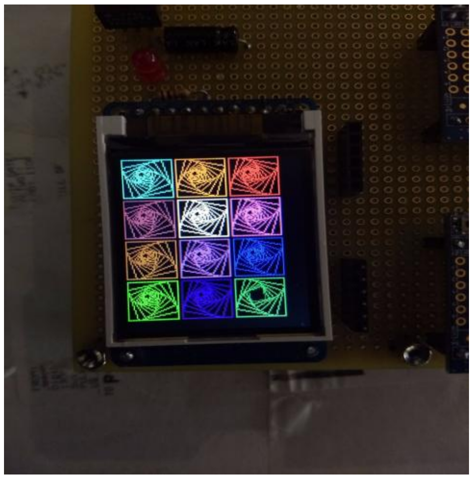

Figure 11: Rotation of Squares at a desired location

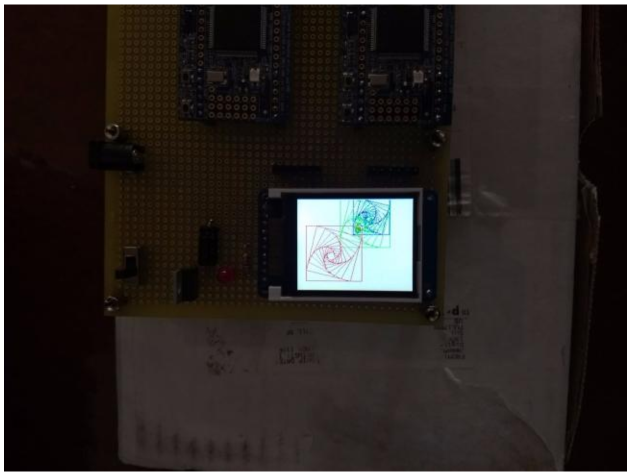

Figure 12: Generation of Random rotating Squares

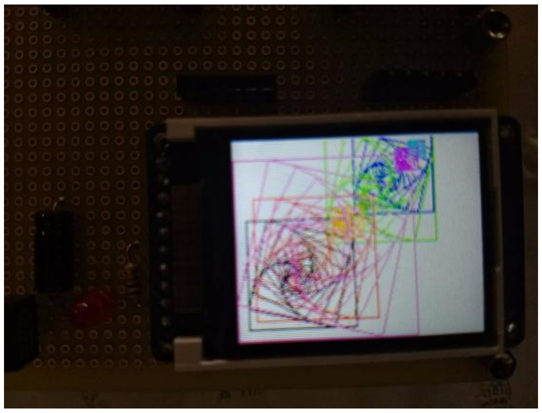

Figure 13: Generation of more Random Squares

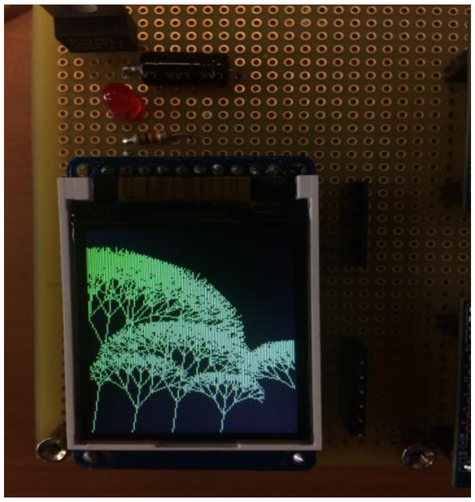

Figure 14: Generation of a Forest on a dark Background

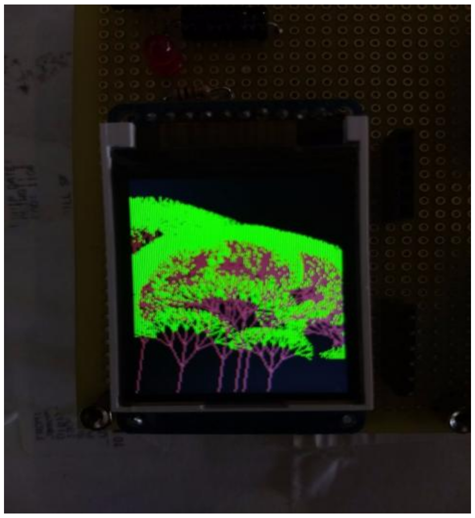

Figure 15: Generation of an entire Forest with branch and leaf distinction.

5. Conclusion

The graphics engine design was tested successfully on the hardware connecting the LCD module with the LPCXpresso 1769.. The image was displayed on LCD on running the program on NXP software. Hence, 2Dimensional graphics has been successfully shown.

6. Acknowledgement

The work described in this paper was made possible and achievable by the contribution of Dr. Harry Li. The electrical and electronic components were made available by Mouser and Fries Electronics.

7. References

[1] H. Li, "Author Guidelines for CMPE 146/242 Project Report", *Lecture Notes of CMPE 146/242*, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.
[2] http://www.adafruit.com/products/358
[3] 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        : Tree.c
 Author      : Harika, Neeraj
 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>
#include <time.h>
//#include <conio.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 BROWN 0xA52A2A
#define RED 0xFF0000
#define MAGENTA 0xFF00FF
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF
#define ORANGE 0xFFA500
#define INDIGO 0x4B0082
#define PINK 0xFF69B4
#define IR 0xCD5C5C
#define CHOC 0xD2691E
#define TGREEN 0x55AE3A

int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;

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 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 i;

 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);

}

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);

}

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


** 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;

  }

 }

}

void drawSquare()
{
				int x0, x1, x2, x3, y0, y1, y2, y3;
				int i;
				int j;

				x0 = 2;
				y0 = 2;
				x1 = 122;
				y1 = 2;
				x2 = 122;
				y2 = 122;
				x3 = 2;
				y3 = 122;

				drawLine(x0, y0, x1, y1, WHITE);
				drawLine(x1, y1, x2, y2, WHITE);
				drawLine(x2, y2, x3, y3, WHITE);
				drawLine(x3, y3, x0, y0, WHITE);

				rotating_mysquare(x0, x1, x2, x3, y0, 
y1, y2, y3, WHITE);

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

				x0 = 0;
				y0 = 0;
				x1 = 0;
				y1 = 35;
				x2 = 35;
				y2 = 35;
			x3 = 35;
				y3 = 0;

				int color1[] = {LIGHTBLUE, IR, 
CHOC, GREEN};
				for( j = 0; j < 4; j++)
				{
					drawLine(x0, y0, x1, y1, 
color1[j]);
					drawLine(x1, y1, x2, y2, 
color1[j]);
					drawLine(x2, y2, x3, y3, 
color1[j]);
					drawLine(x3, y3, x0, y0, 
color1[j]);
					rotating_mysquare(x0, x1, x2, 
x3, y0, y1, y2, y3, color1[j]);

					int k = 38;

					 x0 = x1;
					 x3 = x2;
					 y0 += k;
					 y1 += k;
					 y2 += k;
					 y3 += k;
				}

				x0 = 38;
				y0 = 0;
```

```
                    x1 = 73;
                    y1 = 0;
                    x2 = 73;
                    y2 = 35;

                    x3 = 38;
                    y3 = 35;

                    int color2[] = {ORANGE, RED,
WHITE, MAGENTA, PURPLE, BLUE, INDIGO,
TGREEN};

                         for( j = 0; j < 8; j++)
                         {

                              drawLine(x0, y0, x1,
y1, color2[j]);
                              drawLine(x1, y1, x2,
y2, color2[j]);
                              drawLine(x2, y2, x3,
y3, color2[j]);
                              drawLine(x3, y3, x0,
y0, color2[j]);

          rotating_mysquare(x0, x1, x2, x3, y0, y1, y2, y3,
color2[j]);

                              int k = 38;

                              if ( j == 1)
                              {
                                   x0 = 38;
                                   y0 = 38;
                                   x1 = 73;
                               y1 = 38;
                                   x2 = 73;
                                   y2 = 73;
                                   x3 = 38;
                                   y3 = 73;
                              }
                              else if ( j == 3 )
                              {
                                   x0 = 38;
                                   y0 = 76;
                              x1 = 73;
                                   y1 = 76;
                              x2 = 73;
                                   y2 = 111;
                         x3 = 38;
                                   y3 = 111;

                              }
                              else if ( j == 5)
                              {
                                   x0 = 38;
                                   y0 = 114;
                                   x1 = 73;
                                   y1 = 114;
                                   x2 = 73;
                                   y2 = 149;
                                   x3 = 38;
                                   y3 = 149;

                              }
                              else
                              {
                               y0 = y1;
                               y3 = y2;
                               x0 += k;
                               x1 += k;
                               x2 += k;
                               x3 += k;
                              }
                         }
}

void growTree(int x0, int y0, float angle, int length, int
level, int color){

     int x1, y1, length1;
     float angle1;

          if (level <= 2)
               color = GREEN;
          if(level>0){


          x1 = x0+length*cos(angle);
     y1 = y0+length*sin(angle);

     drawLine(x0,y0,x1,y1,color); //tree branch


          angle1 = angle + 0.52; //deviate right-
>0.52 rad/30 deg

     length1 = 0.8 * length; //reduction of length by
20% of previous length
     growTree(x1,y1,angle1,length1,level-1,color);

     angle1 = angle - 0.52; //deviate left->0.52
rad/30 deg
     length1 = 0.8 * length;
     growTree(x1,y1,angle1,length1,level-1,color);

     angle1 = angle; //center->0 deg
     length1 = 0.8 * length;
     growTree(x1,y1,angle1,length1,level-1,color);
     }
//        printf("exiting mytree\n");
```

```
}

void rotating_mysquare(int x1, int x2, int x3, int x4, int
y1, int y2, int y3, int y4,int color1)
{
	int q1x=0, q2x=0, q3x=0, q4x=0, q1y=0,q2y=0,
q3y=0, q4y=0; //new coordinates
	int i;
	float lambda = 0.8;

	for(i=1;i<=11;i++){
		lcddelay(200);
		q1x = (x2+(lambda*(x1-x2)));
		q1y = (y2+(lambda*(y1-y2)));

		q2x = (x3+(lambda*(x2-x3)));

		q2y = (y3+(lambda*(y2-y3)));
		q3x = (x4+(lambda*(x3-x4)));


		q3y = (y4+(lambda*(y3-y4)));
		q4x = (x1+(lambda*(x4-x1)));
		q4y = (y1+(lambda*(y4-y1)));

		drawLine(q1x,q1y,q2x,q2y,color1);
		drawLine(q2x,q2y,q3x,q3y,color1);
		drawLine(q4x,q4y,q3x,q3y,color1);
		drawLine(q1x,q1y,q4x,q4y,color1);

		x1 = q1x;
		x2 = q2x;
		x3 = q3x;
		x4 = q4x;

		y1 = q1y;
		y2 = q2y;
		y3 = q3y;
		y4 = q4y;
	}
}


void rotating_mysquare1(int x1, int x2, int x3, int x4,
int y1, int y2, int y3, int y4,int color1, float lambda)
{
	int q1x=0, q2x=0, q3x=0, q4x=0, q1y=0,q2y=0,
q3y=0, q4y=0; //new coordinates
	int i;

	for(i=1;i<=11;i++){
		lcddelay(200);
		q1x = (x2+(lambda*(x1-x2)));
		q1y = (y2+(lambda*(y1-y2)));

		q2x = (x3+(lambda*(x2-x3)));

		q2y = (y3+(lambda*(y2-y3)));
		q3x = (x4+(lambda*(x3-x4)));


		q3y = (y4+(lambda*(y3-y4)));
		q4x = (x1+(lambda*(x4-x1)));
		q4y = (y1+(lambda*(y4-y1)));

		drawLine(q1x,q1y,q2x,q2y,color1);
		drawLine(q2x,q2y,q3x,q3y,color1);
		drawLine(q4x,q4y,q3x,q3y,color1);
		drawLine(q1x,q1y,q4x,q4y,color1);

		x1 = q1x;
		x2 = q2x;
		x3 = q3x;
		x4 = q4x;

		y1 = q1y;
		y2 = q2y;
		y3 = q3y;
		y4 = q4y;
	}
}

/*
 Main Function main()
*/
int main (void)

{
	srand(time(0));
	 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);
	 drawSquare();
	 fillrect(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, WHITE);

	 float lambda;
```

```
            printf ("Enter Lambda Value for Random
Square rotation: \n");
            scanf ("%f",&lambda);
            float temp = 0.2;

            if(lambda == temp)
            {
                        lambda = 0.2;
            }
            else
            {
                        lambda = 0.8;
            }
            printf("Lambda Entered is: %f\n", lambda);

            // Random Square Logic
      int f;
             for (f = 0; f < 10 ; f++)
             {
                         int ai=(rand()% 130);
                         int aj=(rand()% 60) + 75;
                         int color[10]={BLUE, GREEN, RED,
TGREEN, BLACK, PURPLE, ORANGE, LIGHTBLUE,
MAGENTA, PINK};

                        drawLine(ai,ai,aj,ai,color[f]);
               drawLine(aj,ai,aj,aj,color[f]);
               drawLine(ai,aj,aj,aj,color[f]);
               drawLine(ai,aj,ai,ai,color[f]);

rotating_mysquare1(ai,aj,aj,ai,ai,ai,aj,aj,color[f], lambda);
             }
             // Random Square Logic End

     fillrect(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, WHITE);
            fillrect(0, 0, ST7735_TFTWIDTH, 18,
LIGHTBLUE);

            // One Big Tree //
            int x0 = 60;
            int y0 = 160;
            int h = 30;
            int ang = 30;
            int lev = 9;
            growTree(x0,  y0, h, ang, 7, BROWN);
            lcddelay(5000);

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

                        growTree(0,  y0, 30, 38, 7, BROWN);
                        growTree(25,  y0, 30, 20, 5, BROWN);
                        growTree(50,  y0, 30, 25, 6, BROWN);
                        growTree(80,  y0, 30, 15, 5, BROWN);
                        growTree(130, y0, 30, 20, 7,
BROWN);

                        char ch;

            while (1)
            {
                        char ch;

                        h = 30;
                        growTree(x0,  y0, h, ang, lev,
BROWN);
                        if ( h == 30 )
                                    growTree(65,  y0, h, ang, lev,
BROWN);
                        if ( h == 30 )
                                    growTree(45,  y0, h, ang, lev,
BROWN);
                        if ( h == 30 )
                                    growTree(x0,  140, h, ang,
lev, BROWN);

                        ang = (rand() % 10) + 20;
                        lev = (rand() % 6) + 3;

                        int temp_x = rand() % 100;
                        int temp_y = rand() % 160;
                        x0 = temp_x;
                        y0 = temp_y;
                        h = (rand() % 20 ) + 5;

                        ch = getchar();

                        if (ch == 'q' || ch == 'Q')
                                    break;
                        getchar();
            }
            printf("Exited While Loop");
            return 0;
}
```