Department of Electrical Engineering,
Syed Babar Ali School of Science and Engineering,
Lahore University of Management Sciences, Lahore, Pakistan

EE 522 Embedded Systems

Snake Game using mini6410

Submitted to
Adeel Pasha
Date: May 18, 2016

by

Mirza Athar Baig
15060027
15060027@lums.edu.pk

Contents

Abstract

This report summarizes the work done as part of a semester project under the supervision of Dr. Adeel Pasha. The project was based on the embedded system design which required both hardware and software implementation. The project employed mini6410 with LCD screen and Qt application development tool. We developed the classic snake game, a single player single level game on LED matrix. This project is divided in two parts. One is driver development of LED Matrix and the second part deals with the whole game logic coded in Qt SDK.

Introduction:

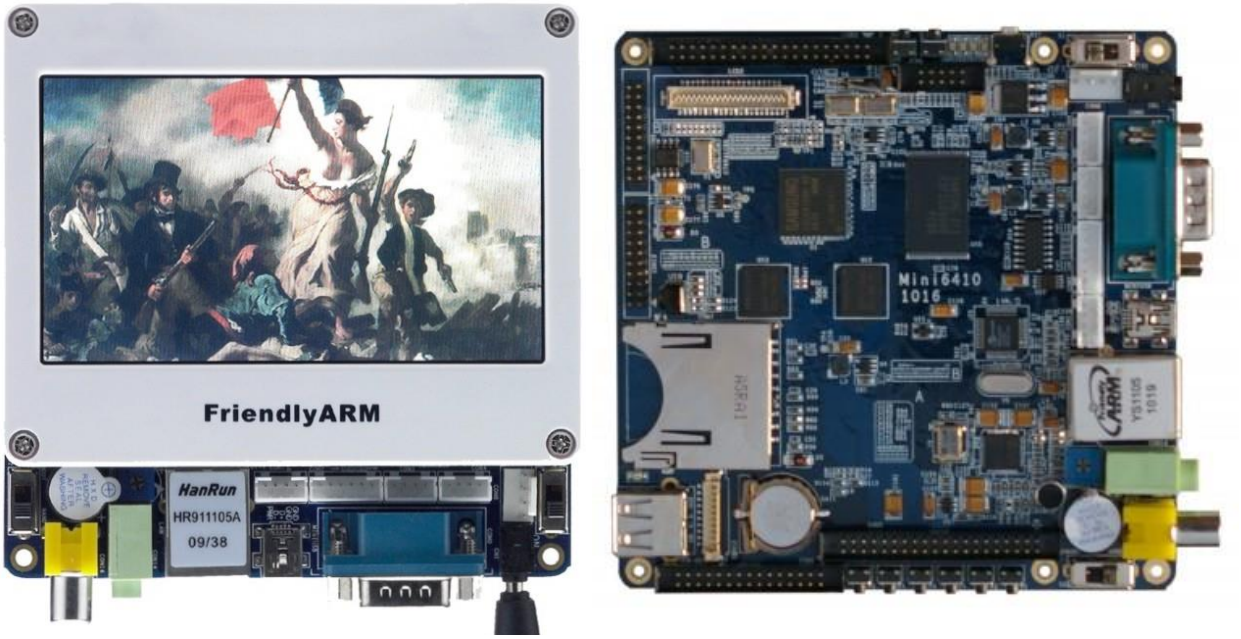
This project employed Friendly ARM mini6410 as embedded processing tool, Qt creator to implement the logic and Linux Ubuntu for driver development.

Hardware:

This chapter introduces the development board and other hardware which is utilized in this project

Mini6410:

The project was implemented on the mini6410 Board.



The Mini6410 Development Board is a high-performance controller board introduced and ideal for learning about ARM systems. It is designed based on the S3C6410 microcontroller, 256M Byte DDR SDRAM, 1G Byte NAND Flash, RTC, Audio and net on board. It has integrated RS232, USB, Ethernet, Audio In/Out, Keyboard, LCD, camera in, SD card and more other functions on board [1].

The board supports Linux 2.6.36, Android2.1 and Windows CE 6.0 operating system and is provided with complete basic drivers which enable a quick channel to evaluate the Samsung S3C6410 processor and customize application software.

There are total of 17 different ports available on Mini6410 board. Each port has CON and DAT registers. We used PORT E and PORT Q to access the rows and columns of the 8x16 LED Matrix [2].

LED Matrix

An 8X16 LED Matrix is used to display the overall functionality of the game. These LEDs were accessed by the customized character type driver that is built in LINUX environment.

Software:

A char type custom driver is developed to access the LED matrix. Linux Ubuntu 3.1.18 is used for the development of driver. The logical code for the snake game is developed on QT creator.

Implementation:

A Qt application is developed to control the overall logic of the game. The app uses four buttons for the movement of snake in up, down, left and right direction. 6 timers are used in the application to maintain the functionality of the game. This application sends the logic of game to the driver of LED matrix and it take the actions accordingly. The flow chart for the user space application and kernel space driver is shown in fig (2) [3]

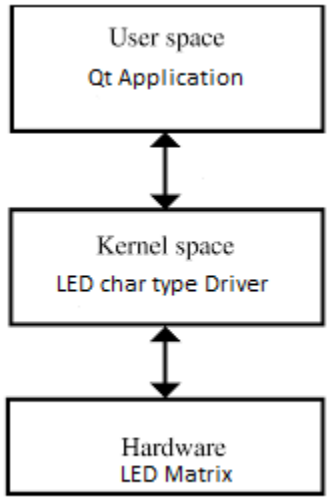


Figure 2

User interface of the Qt application is given in figure 3

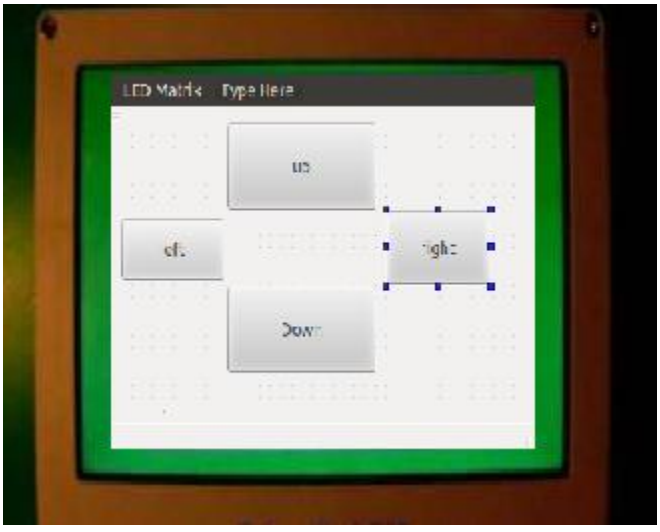


Figure 3

Complete code of the Qt application and driver is given in appendix A.

Conclusion

We have completed our knowledge of embedded system design after we finished the project and obtained more experience of hardware/software co-design.

In fully functional snake game:

i) Snake grows after eating fruit
ii) Fruit is placed randomly
iii) Snake speed is increased after eating fruits
iv) Whenever snake hits itself game is aborted

References:

[1]. http://www.friendlyarm.net/products/mini6410

[2]. S3C6410X User Manual

[3]. *Real-Time Systems*: Theory and Practice. By *Rajib Mall*

Appendix A:

QT Header File:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <stdio.h>
#include <stdlib.h>
#include <QTimer>
#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   explicit MainWindow(QWidget *parent = 0);
   ~MainWindow();

private slots:
   void on_pushButton_clicked();
```

```
    void on_pushButton_2_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_3_clicked();
    void snake_timer_expiry();
    void fruit_timer_expiry();
    void snake_print_expiry();
    void fruit_print_expiry();
    void fruit_p_expiry();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

QT Main Window File:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <QThread>

int fd;
int dx1;
int dy1;
int dx2;
int dy2;
int pc;
int snakex[10];
int snakey[10];
int snakelength = 4;
int direction = 1;
int fruitx;
int fruity;
int period;
QTimer *timer;
QTimer *snake_timer;
QTimer *fruit_timer;
QTimer *snake_print;
QTimer *fruit_print;
```

```
QTimer *fruit_p;
class Sleeper : public QThread
{
public:
   static void usleep(unsigned long usecs){QThread::usleep(usecs);}
   static void msleep(unsigned long msecs){QThread::msleep(msecs);}
   static void sleep(unsigned long secs){QThread::sleep(secs);}
};

MainWindow::MainWindow(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::MainWindow)
{
   ui->setupUi(this);
   fd = open("/dev/LED_Matrix", 0);

   //ui->setupUi(this);
   timer = new QTimer(this);
   snake_timer=new QTimer(this);
   fruit_timer=new QTimer(this);
   snake_print=new QTimer(this);
   fruit_print=new QTimer(this);
    fruit_p=new QTimer(this);
   connect(ui->pushButton, SIGNAL (clicked()), this, SLOT (on_pushButton_clicked()));
   connect(ui->pushButton_2, SIGNAL (clicked()), this, SLOT (on_pushButton_2_clicked()));
   connect(ui->pushButton_3, SIGNAL (clicked()), this, SLOT (on_pushButton_3_clicked()));
   connect(ui->pushButton_4, SIGNAL (clicked()), this, SLOT (on_pushButton_4_clicked()));
   snakex[0]=3;
   snakex[1]=2;
   snakey[0]=0;
   snakey[1]=0;
   snakex[2]=1;
   snakex[3]=0;
   snakey[2]=0;
   snakey[3]=0;
   fruitx = 1;
   fruity = 3;


   connect(snake_timer, SIGNAL(timeout()), this, SLOT(snake_timer_expiry()));
   period = 700;//rand()%2000;
   snake_timer->setInterval(period);
   snake_timer->start();

   connect(fruit_timer, SIGNAL(timeout()), this, SLOT(fruit_timer_expiry()));
   int period_fruit = 200;
   fruit_timer->setInterval(period_fruit);
   fruit_timer->start();
```

```
    connect(snake_print, SIGNAL(timeout()), this, SLOT(snake_print_expiry()));
    int period_snake_print = 1;
    snake_print->setInterval(period_snake_print);
    snake_print->start();

    connect(fruit_print, SIGNAL(timeout()), this, SLOT(fruit_print_expiry()));
    int period_fruit_print = 50;
    fruit_print->setInterval(period_fruit_print);
    fruit_print->start();

    // bund krne k lye
    connect(fruit_p, SIGNAL(timeout()), this, SLOT(fruit_p_expiry()));
    int p = 50;
    fruit_p->setInterval(p);
    fruit_p->start();


}

MainWindow::~MainWindow()
{


    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    if(direction == 3)
    {
        direction = 3;
        snake_timer_expiry();
    }
    else
    {
        direction = 1;
        snake_timer_expiry();
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    if(direction == 4)
    {
        direction = 4;
        snake_timer_expiry();
    }
```

```
    else
    {
        direction = 2;
        snake_timer_expiry();
    }
}

void MainWindow::on_pushButton_4_clicked()
{
    if(direction == 1)
    {
        direction = 1;
        snake_timer_expiry();
    }
    else
    {
        direction = 3;
        snake_timer_expiry();
    }
}

void MainWindow::on_pushButton_3_clicked()
{
    if(direction == 2)
    {
        direction = 2;
        snake_timer_expiry();
    }
    else
    {
        direction = 4;
        snake_timer_expiry();
    }
}
void MainWindow::fruit_p_expiry()
{


    for (int c=1;c<snakelength;c++)
    {
        if(snakex[c]==snakex[0] && snakey[c]==snakey[0])
        {
            timer->stop();
            snake_timer->stop();
            fruit_timer->stop();
            snake_print->stop();
            fruit_print->stop();
            fruit_p->stop();
```

```
    }
  }

}
void MainWindow::fruit_timer_expiry()
{


  if(snakex[0]==fruitx && snakey[0]==fruity)
  {
    snakelength++;

    if (period>300)
    {
      period = period - 100;
    }
    fruitx=rand() % 8;
    fruity=rand() % 16;
    int cc=1;
    while(cc ==1)
    {
      for(int c=0;c<snakelength;c++)
      {
        if(snakex[c]==fruitx && snakey[c]==fruity)
        {
          fruitx=rand() % 8;
          fruity=rand() % 16;
        }
      }
      cc=2;
      for(int c=0;c<snakelength;c++)
      {
        if(snakex[c]==fruitx && snakey[c]==fruity)
        {
          cc=1;
        }
      }
    }
  }
}

void MainWindow::snake_print_expiry()
{

  if (pc<snakelength)
  {
    ioctl(fd,snakex[pc],snakey[pc]);Sleeper::usleep(1);
```

```
      pc++;
   }
   else
   {
      pc=0;
      ioctl(fd,snakex[pc],snakey[pc]);Sleeper::usleep(1);
      pc++;
   }

}
void MainWindow::fruit_print_expiry()
{
   ioctl(fd,fruitx,fruity);Sleeper::usleep(1);

}

void MainWindow::snake_timer_expiry()
{

      /*for(int x=0;x<8;x++)
      {
         for(int y=0;y<16;y++)
         {

            ioctl(fd,x,y);Sleeper::usleep(10);
            //ioctl(fd,x,y);Sleeper::usleep(1);
            //ioctl(fd,x,y);Sleeper::usleep(100);
         }
      }*/
      if(direction==1)
      {
         dx1=snakex[0];
         dy1=snakey[0];
         for(int dc=1;dc<snakelength;dc++)
         {
            dx2=snakex[dc];
            dy2=snakey[dc];
            snakex[dc]=dx1;
            snakey[dc]=dy1;
            dx1=dx2;
            dy1=dy2;

         }
         snakex[0]++;
         if(snakex[0]==8)
         {
            snakex[0]=0;
         }
```

```
}
if(direction==2)
{

   dx1=snakex[0];
   dy1=snakey[0];
   for(int dc=1;dc<snakelength;dc++)
   {
      dx2=snakex[dc];
      dy2=snakey[dc];
      snakex[dc]=dx1;
      snakey[dc]=dy1;
      dx1=dx2;
      dy1=dy2;

   }
   snakey[0]++;
   if(snakey[0]==16)
   {
      snakey[0]=0;
   }

}
if(direction==3)
   {
      dx1=snakex[0];
      dy1=snakey[0];
      for(int dc=1;dc<snakelength;dc++)
      {
         dx2=snakex[dc];
         dy2=snakey[dc];
         snakex[dc]=dx1;
         snakey[dc]=dy1;
         dx1=dx2;
         dy1=dy2;

      }
      snakex[0]--;
      if(snakex[0]==-1)
      {
         snakex[0]=7;
      }

   }


if(direction==4)
```

```
    {
      dx1=snakex[0];
      dy1=snakey[0];
      for(int dc=1;dc<snakelength;dc++)
      {
        dx2=snakex[dc];
        dy2=snakey[dc];
        snakex[dc]=dx1;
        snakey[dc]=dy1;
        dx1=dx2;
        dy1=dy2;

      }
      snakey[0]--;
      if(snakey[0]==-1)
      {
        snakey[0]=15;
      }

    }

}
```

Driver File:

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/slab.h>


#include <linux/input.h>
#include <linux/init.h>
#include <linux/errno.h>
#include <linux/serio.h>
#include <linux/delay.h>
#include <linux/clk.h>
#include <linux/wait.h>
#include <linux/sched.h>
#include <linux/cdev.h>

#include <linux/miscdevice.h>


#include <asm/io.h>
#include <asm/irq.h>
#include <asm/uaccess.h>

#include <mach/map.h>
```

```
#include <mach/regs-clock.h>
#include <mach/regs-gpio.h>

#include <plat/gpio-cfg.h>
#include <mach/gpio-bank-q.h>
#include <mach/gpio-bank-e.h>

#include <mach/map.h>
#include <plat/regs-timer.h>

#define DEVICE_NAME     "Dot_Mat"
#define RLV       0x1400
//reload value should be between 4k to 64k … see the datasheet for details
unsigned int rows=0;
unsigned int columns=0;
typedef struct {
        int delay;
} TIM_DEV;

static TIM_DEV TimDev;
void LED(unsigned int x, unsigned int y);
static irqreturn_t INTHandler(int irq,void *TimDev)
{

        unsigned TimerINTControl;
        unsigned TimerCNTB;
        unsigned TimerCMPB;
        int TMP;

        // Read Timer Registers
        TimerINTControl = readl(S3C_TINT_CSTAT);
        TimerCNTB = readl(S3C_TCNTB(0));
        TimerCMPB = readl(S3C_TCMPB(0));

        // Your logic to send the frame values to port Q (16 rows) and E (8 columns)
// You may use LED (x, y) function as we did in PIC-based assignment

        TimerINTControl |= S3C_TINT_CSTAT_T0INT;
        writel(TimerINTControl, S3C_TINT_CSTAT);
        writel(RLV, S3C_TCNTB(0));
        writel(RLV, S3C_TCMPB(0));
        LED(rows,columns);
        return IRQ_HANDLED;
}

void LED(unsigned int x, unsigned int y)
{
        unsigned Port_Q_Dat;
```

```
unsigned Port_E_Dat;
Port_E_Dat = readl(S3C64XX_GPEDAT);
Port_Q_Dat = readl(S3C64XX_GPQDAT);

Port_E_Dat |= (0x1 << (4));

writel(Port_E_Dat, S3C64XX_GPEDAT);
Port_E_Dat = readl(S3C64XX_GPEDAT);

if (test_bit(0,(const volatile long unsigned int *)&x)) {
        Port_E_Dat |= (0x1 << (1));
} else {
        Port_E_Dat &= ~(0x1 << (1));
}

if (test_bit(1,(const volatile long unsigned int *)&x)) {
        Port_E_Dat |= (0x1 << (2));
} else {
        Port_E_Dat &= ~(0x1 << (2));

}

if (test_bit(2,(const volatile long unsigned int *)&x)) {
        Port_E_Dat |= (0x1 << (3));
} else {
        Port_E_Dat &= ~(0x1 << (3));

}

if (test_bit(0,(const volatile long unsigned int *)&y)) {
        Port_Q_Dat |= (0x1 << (1));
} else {
        Port_Q_Dat &= ~(0x1 << (1));
}

if (test_bit(1,(const volatile long unsigned int *)&y)) {
        Port_Q_Dat |= (0x1 << (2));
} else {
        Port_Q_Dat &= ~(0x1 << (2));
}

if (test_bit(2,(const volatile long unsigned int *)&y)) {
        Port_Q_Dat |= (0x1 << (3));
} else {
        Port_Q_Dat &= ~(0x1 << (3));
}

if (test_bit(3,(const volatile long unsigned int *)&y)) {
```

```
                Port_Q_Dat |= (0x1 << (4));
        } else {
                Port_Q_Dat &= ~(0x1 << (4));
        }

        Port_E_Dat &= ~(0x1 << (4)); // This is to put EN-port to low

        writel(Port_E_Dat, S3C64XX_GPEDAT);
        writel(Port_Q_Dat, S3C64XX_GPQDAT);
}

static ssize_t MyWrite (struct file *filp, unsigned int var1,unsigned int var2) {

        rows = var1;
        columns = var2;



return 1;
}

static struct file_operations dev_fops = {
        owner:              THIS_MODULE,
        .unlocked_ioctl = MyWrite,
};

static struct miscdevice misc = {
        .minor  = MISC_DYNAMIC_MINOR,
        .name   = DEVICE_NAME,
        .fops   = &dev_fops,
};

static int __init dev_init(void)
{
        int ret;
        unsigned TimerControl;
        unsigned TimerINTControl;
        unsigned TimerCNTB;
        unsigned TimerCMPB;
        unsigned TimerCFG1;



        unsigned Port_Q_Dat;
        unsigned Port_Q_Con;
        unsigned Port_E_Dat;
        unsigned Port_E_Con;
        unsigned char data_init = 0;
```

```
        Port_E_Dat = readl(S3C64XX_GPEDAT);
        Port_E_Con = readl(S3C64XX_GPECON);
        Port_Q_Dat = readl(S3C64XX_GPQDAT);
        Port_Q_Con = readl(S3C64XX_GPQCON);



        Port_Q_Con = (Port_Q_Con & ~(255<<2))|(85<<2);

        Port_E_Con = (Port_E_Con & ~(65535<<4))|(0x1111U<<4);

        Port_Q_Dat = 0;
        Port_E_Dat= data_init | 1;

        writel(Port_E_Dat, S3C64XX_GPEDAT);
        writel(Port_E_Con, S3C64XX_GPECON);
        writel(Port_Q_Dat, S3C64XX_GPQDAT);
        writel(Port_Q_Con, S3C64XX_GPQCON);


TimerControl            = readl(S3C_TCON);
        TimerINTControl     = readl(S3C_TINT_CSTAT);
        TimerCNTB           = readl(S3C_TCNTB(0));
        TimerCMPB           = readl(S3C_TCMPB(0));

        TimerCFG1           = readl(S3C_TCFG1);

        TimerCFG1           &= ~(S3C_TCFG1_MUX0_MASK);

        TimerCNTB           = RLV;
        TimerCMPB           = RLV;

        writel(TimerCNTB, S3C_TCNTB(0));
        writel(TimerCMPB, S3C_TCMPB(0));
        writel(TimerCFG1, S3C_TCFG1);

        TimerControl        |= S3C_TCON_T0MANUALUPD;

        TimerINTControl     |= S3C_TINT_CSTAT_T0INTEN;

        writel(TimerControl, S3C_TCON);
        writel(TimerINTControl, S3C_TINT_CSTAT);

        TimerControl        = readl(S3C_TCON);

        TimerControl        |= S3C_TCON_T0RELOAD;
        TimerControl        &= ~S3C_TCON_T0MANUALUPD;
        TimerControl        |= S3C_TCON_T0START;
```

```
	writel(TimerControl, S3C_TCON);

	ret = request_irq(IRQ_TIMER0, INTHandler, IRQF_SHARED, DEVICE_NAME, &TimDev);
	if (ret) {
		return ret;
	}
	ret = misc_register(&misc);

	printk (DEVICE_NAME"\tinitialized\n");
	return ret;
}

static void __exit dev_exit(void)
{
	unsigned Port_Q_Dat;
	unsigned Port_Q_Con;
	unsigned Port_E_Dat;
	unsigned Port_E_Con;

	Port_E_Dat = readl(S3C64XX_GPEDAT);
	Port_E_Con = readl(S3C64XX_GPECON);
	Port_Q_Dat = readl(S3C64XX_GPQDAT);
	Port_Q_Con = readl(S3C64XX_GPQCON);

	// Write your logic to set Port Q and E as OUTPUT ports
	// Write your logic to set all pins of Port Q and E (except EN) low, so that LED matrix starts with
all LEDs turned off


	Port_Q_Dat = 0;
	Port_E_Dat= 1;

	writel(Port_E_Dat, S3C64XX_GPEDAT);

	writel(Port_Q_Dat, S3C64XX_GPQDAT);


	free_irq(IRQ_TIMER0, &TimDev);
	misc_deregister(&misc);
	printk (DEVICE_NAME"\tModule Unloaded\n");
}
module_init(dev_init);
module_exit(dev_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Asad");
```