

EEDG/CE 6302
Microprocessors and Embedded Systems
Electrical and Computer Engineering
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Dr. Tooraj Nikoubin

**Project 2: Part 1: Joystick to Seven – Segment Display using
MSP432 LaunchPad**

Submitted By:
Muripa Uppaluri (MXU220008)
Chandanam Sai Nived (SXC210186)

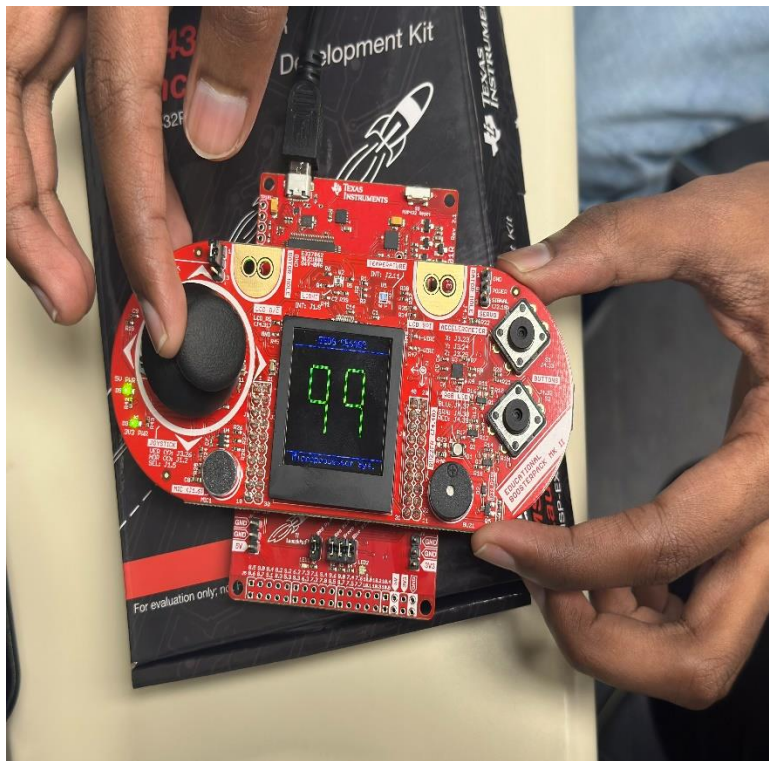
Table of Contents

Content	Page Number
Introduction	3
Lab Results	3
Working code	4

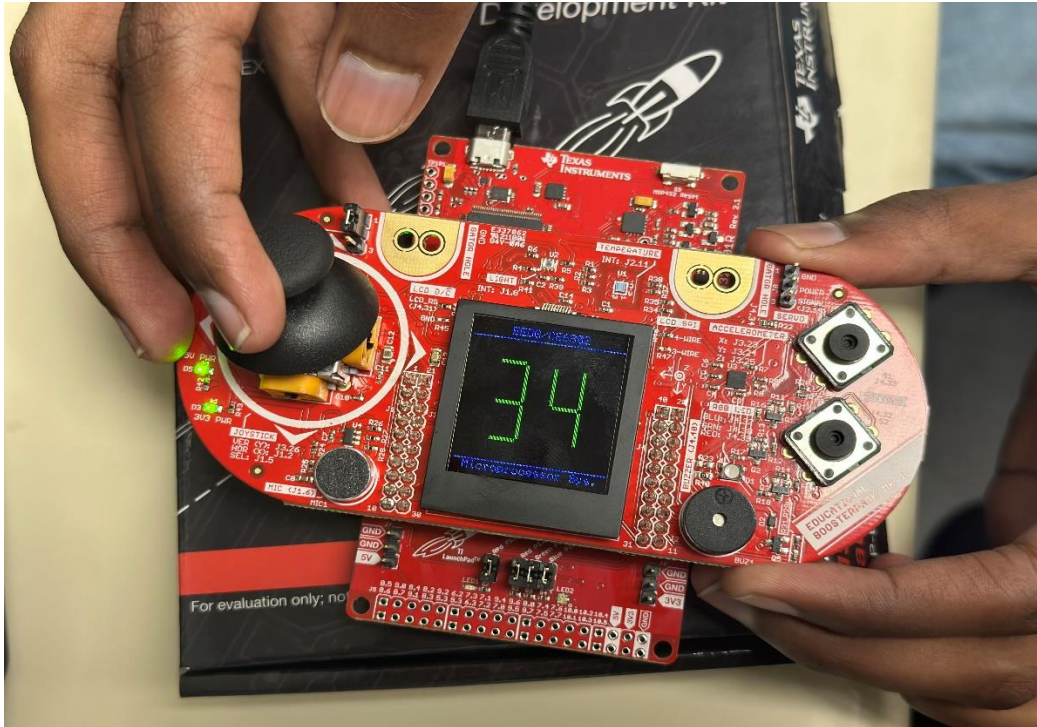
Introduction:

The SimpleLink MSP432P4111 Launchpad is the development kit with a 48MHz Arm Cortex-M4F as core. This launchpad enables us to develop high-precision sensor node applications. There are multiple sensors on the MKII. In this project, we will use joystick as the sensor to control the display on the LCD. We will use the LCD to represent seven segment display. The 7-Segment Display, also written as “Seven Segment Display”, consists of seven LEDs (according to their name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit to be displayed. We can select either the x or y axis of the joystick for this experiment. The goal of this project is to convert the analog input using joystick (range 0 to 99) and display it on the LCD screen.

Output:



LCD display showing the maximum value that can displayed.



LCD display showing Y axis output.

Code:

```

/*
 * -----
 * DO NOT EDIT CODE ABOVE THIS LINE
 * -----
 */

// Include application, user, and local libraries
#include <SPI.h>
#include <LCD_screen.h>
#include <LCD_screen_font.h>
#include <LCD_utilities.h>
#include <Screen_HX8353E.h>
#include <Terminal12e.h>

```

```

#include <Terminal6e.h>

#include <Terminal8e.h>

// Declare header function with default values
void headerFooter(String header_text = "EEDG/CE6302",
                  String footer_text = "Microprocessor Sys.",
                  uint16_t color = blueColour);

```

```

// Define constants for the joystick pins
const int joystickXPin = 2;
const int joystickYPin = 26;

```

```

// Define screen
Screen_HX8353E myScreen;

```

```

/*
 * -----
 * DO NOT EDIT CODE ABOVE THIS LINE
 * -----
 */

```

```

// YOUR DECLARATIONS AND DEFINITIONS HERE

```

```

// Add setup code
void setup()
{
  /*
   * DO NOT EDIT BELOW THIS LINE
   */
}

```

```

Serial.begin(9600); // for LCD debug output

// By default MSP432 has analogRead() set to 10 bits.
// This Sketch assumes 12 bits. Uncomment to line below to set analogRead()
// to 12 bit resolution for MSP432.
analogReadResolution(12);

// Init screen
myScreen.begin();
myScreen.setPenSolid(true);

// Print info screen
infoScreen();
delay(2000);

// Clear screen and put header and footer on screen
myScreen.clear();
headerFooter();
/*
 * DO NOT EDIT ABOVE THIS LINE
 */

// YOUR SETUP CODE HERE (runs once)
}

// Add loop code
void loop()
{
    // YOUR LOOP CODE HERE (runs continuously after setup function)

```

```

int x = map(analogRead(joystickXPin), 0, 4096, -3, 100);

Serial.println(x);

setTens(x/10);

setOnes(x%10);
}

/*
* -----
* DO NOT EDIT CODE BELOW THIS LINE
* -----
*/

// Set specific colors
// Possible colors:
// white, red, green, blue, yellow, cyan, orange, magenta, violet, gray, darkGray
const uint16_t digit_color = greenColour;
const uint16_t header_color = blueColour;
const uint16_t error_color = redColour;

// Define error variables
bool error_flag = true;
bool one_error = false;
bool ten_error = false;
uint16_t last_x = 0;

// Segment on/off definitions
//////// 1 /////
/////  _____
///// | |

```

```
//// 2 |   | 6
```

```
///// |__3__|
```

```
///// |   |
```

```
/////4 |   | 7
```

```
///// |__5__|
```

```
// Segment mask
```

```
bool num_seg[11][7]={
```

```
    {1,1,0,1,1,1,1},
```

```
    {0,0,0,0,0,1,1},
```

```
    {1,0,1,1,1,1,0},
```

```
    {1,0,1,0,1,1,1},
```

```
    {0,1,1,0,0,1,1},
```

```
    {1,1,1,0,1,0,1},
```

```
    {0,1,1,1,1,0,1},
```

```
    {1,0,0,0,0,1,1},
```

```
    {1,1,1,1,1,1,1},
```

```
    {1,1,1,0,0,1,1},
```

```
    {1,1,1,1,1,0,0},
```

```
};
```

```
// Digit pixel map
```

```
uint16_t sev_seg0[7][4] = {
```

```
    { 3, 0,21,3},
```

```
    { 0, 3,3,32},
```

```
    { 3,35,21,3},
```

```
    { 0,38,3,29},
```

```
    { 3,67,21,3},
```

```
    {24, 3,3,32},
```



```

        {24,38,3,29}

    };

// Draw a number at a given (x, y) position with (0,0) at top left
void drawNumber(int value, uint16_t x_offset, uint16_t y_offset, uint16_t color){

    // Handle header before drawing digit

    // This should probably be its own function or the whole thing should be a class
    if (one_error | ten_error)
    {
        // Use error flag to prevent redrawing the header in event of no error
        error_flag = true;

        headerFooter(">>>>> ERROR <<<<<<", "Check serial monitor!", error_color);
    } else if (error_flag){
        error_flag = false;
        clearHeaderFooter();
        headerFooter();
    }

    // Draw rectangles for each segment
    for(int i=0;i<7;i++){

        if(num_seg[value][i])

            myScreen.dRectangle(sev_seg0[i][0] + x_offset,
                                sev_seg0[i][1] + y_offset,
                                sev_seg0[i][2],
                                sev_seg0[i][3],
                                color);

        else

            myScreen.dRectangle(sev_seg0[i][0] + x_offset,
                                sev_seg0[i][1] + y_offset,
                                sev_seg0[i][2],
                                sev_seg0[i][3],

```

```

        blackColour);
    }

}

// Output the ones digit to the seven segment display
void setTens(int value) {
    // Offset definitions for tens digit
    uint16_t x_offset = 24;
    uint16_t y_offset = 29;

    // Error check and draw digit or 'E' for error
    if (value > 9) {
        ten_error = true;
        Serial.print("[ERROR]: Tens digit value of ");
        Serial.print(value);
        Serial.println(" is outside of expected range (0-9)!");
        drawNumber(10, x_offset, y_offset, error_color);
    } else {
        ten_error = false;
        drawNumber(value, x_offset, y_offset, digit_color);
    }
}

// Output the ones digit to the seven segment display
void setOnes(int value) {
    // Offset definitions for ones digit
    uint16_t x_offset = 72;
    uint16_t y_offset = 29;

```

```

// Error check and draw digit or 'E' for error
if (value > 9) {
    one_error = true;
    Serial.print("[ERROR]: Ones digit value of ");
    Serial.print(value);
    Serial.println(" is outside of expected range (0-9)!");
    drawNumber(10, x_offset, y_offset, error_color);
} else {
    one_error = false;
    drawNumber(value, x_offset, y_offset, digit_color);
}
}

void headerFooter(String header_text,
                  String footer_text,
                  uint16_t color) {
    // Determine header and footer x positions (in center)
    uint16_t header_x = (myScreen.screenSizeX() - myScreen.fontSizeX() * header_text.length())/2;
    uint16_t footer_x = (myScreen.screenSizeX() - myScreen.fontSizeX() * footer_text.length())/2;

    // Write out header and footer and lines to separate
    myScreen.gText(header_x, 0, header_text, color);
    myScreen.gText(footer_x, myScreen.screenSizeY()-myScreen.fontSizeY()-1, footer_text, color);
    myScreen.dLine(0, myScreen.fontSizeY() + 2, myScreen.screenSizeX(), 1, color);
    myScreen.dLine(0, myScreen.screenSizeY()-myScreen.fontSizeY()-3, myScreen.screenSizeX(), 1, color);
}

// Clears the header and footer area

```

```

// Only needs to be done when going from error to regular
void clearHeaderFooter() {
    myScreen.dRectangle(0, 0, myScreen.screenSizeX(), myScreen.fontSizeY(), blackColour);

    myScreen.dRectangle(0, myScreen.screenSizeY()-myScreen.fontSizeY()-1, myScreen.screenSizeX(),
myScreen.fontSizeY(), blackColour);
}

void infoScreen() {
    // Print background
    myScreen.clear(cyanColour);

    // Top lines
    String line1 = " EEDG/CE 6302 ";
    String line2 = " MSP432: LCD Lab ";
    uint16_t t1_width = myScreen.fontSizeX() * line1.length();
    uint16_t t2_width = myScreen.fontSizeX() * line2.length();

    myScreen.dLine((myScreen.screenSizeX() - t1_width)/2, 4, t1_width, 1, blueColour);
    myScreen.gText((myScreen.screenSizeX() - t1_width)/2, 5, line1, whiteColour, blueColour);
    myScreen.gText((myScreen.screenSizeX() - t2_width)/2, 5 + myScreen.fontSizeY() * 1, line2,
whiteColour, blueColour);

    // Bottom lines
    String line4_bot = " Created by: ";
    String line3_bot = " Nived ";
    String line2_bot = " & ";
    String line1_bot = " Muripa ";
    uint16_t b1_width = myScreen.fontSizeX() * line1_bot.length();
    uint16_t b2_width = myScreen.fontSizeX() * line2_bot.length();

```

```

uint16_t b3_width = myScreen.fontSizeX() * line3_bot.length();
uint16_t b4_width = myScreen.fontSizeX() * line4_bot.length();

myScreen.gText((myScreen.screenSizeX() - b1_width)/2, myScreen.screenSizeY() - (5 +
myScreen.fontSizeY() * 1), line1_bot, whiteColour, blueColour);

myScreen.gText((myScreen.screenSizeX() - b2_width)/2, myScreen.screenSizeY() - (5 +
myScreen.fontSizeY() * 2), line2_bot, whiteColour, blueColour);

myScreen.gText((myScreen.screenSizeX() - b3_width)/2, myScreen.screenSizeY() - (5 +
myScreen.fontSizeY() * 3), line3_bot, whiteColour, blueColour);

myScreen.gText((myScreen.screenSizeX() - b4_width)/2, myScreen.screenSizeY() - (5 +
myScreen.fontSizeY() * 4), line4_bot, whiteColour, blueColour);

myScreen.dLine((myScreen.screenSizeX() - b4_width)/2, myScreen.screenSizeY() - (5 +
myScreen.fontSizeY() * 4) - 1, b4_width, 1, blueColour);
}

/*
for (i=0,i<5,i++)
{
    for(j=0,j)
}
*/

```