EENG 3910: Project V – Digital Signal Processing System Design

Final Project

Due 09 May 2016

Chris Askings
Ian Hunt
Nathan Ruprecht
UNT – Electrical Engineering

Introduction

The purpose of the final project is to design and implement an embedded DSP system that uses what we have learned in this class. Our proposal was that we planned to do a geo fence. Using a predetermined "box," a GPS module will send our current coordinates to our Tiva board which is paired with the Tiva booster pack. The booster pack will give us a signal showing when we are inside or outside the boundaries of our fence.

Parts used:

Tiva Series Launchpad – TM4C123G
GPS Module – NEO-6M
Booster Pack – EDU-MKII

Results and Discussion

We were able to break up the final project into 3 portions (1 portion for each student): working with the GPS module, working with the booster pack, and data processing for system integration. Of course, we would help each other as they got stuck, and we all came together to check the final code, report, and presentation.

NEO-6M (GPS Module):

EDU-MKII (Booster Pack):

When it came to using a booster pack, the only difficulty was figuring out the pinout compared to the tiva board. The LEDs on the MKII were pins 37-39 which translated to F3, B3, and C4 on the Launchpad.

Initializing the LEDs was straight forward since we've seen and done it before. We enabled the correct port then specified each pin as an output since it's an LED.

```
void init_LEDs(void)
{
	// Red LED
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
	GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_3);

	// Green LED
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
	GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);

	// Blue LED
	SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
	GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_4);
}
```

If we were inside the preset "fence," then we were still okay and the green LED would be on. If not, the red LED would be lit.

```
if( condition ) {
      //We're good, green LED
      cur_LED=1;
} else {
      //We're bad, red LED
      cur_LED=0;
 }//end if
```

A simple case statement was used to switch which LED was on. It was a matter of raising 2 to the power of the pin to know the value in order to have that pin set to 1. Later on in the code is when all the LEDs are turned off (all pins set to 0).

```
switch( cur_LED )
{
      case 0://Red
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 8);
             break;
      case 1://Green
            GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 8);
             break;
      case 2://Blue
            GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4, 16);
             break;
}//End switch
```

Data Processing:

Data processing was just a lot of algebra. We picked the east parking lot of Discovery Park for our geo fence since it's so familiar and easy to test. We got the coordinates of the 4 corners and defined them as points 1 – 4 going clockwise from the most northern point. Since the box we made is at an angle, we couldn't just compare the x and y coordinate. Instead, we characterized each line based off of y = mx + b.

```
m12 = (y2-y1) / (x2-x1);
m41 = (y1-y4) / (x1-x4);
m43 = (y3-y4) / (x3-x4);
m32 = (y2-y3) / (x2-x3);

b12 = y1 - m12*x1;
b43 = y4 - m43*x4;
b41 = y4 - m41*x4;
b32 = y3 - m32*x3;
```

The coordinate from the GPS became our test value. We held the x coordinate the same and plugged it into the mx+b equation to find the range of y. If the test y was outside the range, then the condition failed and we were outside. If the test y was in range, we moved to testing the test x coordinate.

```
//Take lat_data and find the corresponding y values on the lines from point 1-2
//and point 4-3. These y values are a range.
```

```
//Test long_data against those corresponding y values for a vertical check
ycp1 = m12*lat_data + b12;
ycp2 = m43*lat_data + b43;
```

Same principle, holding the y value constant and plugging x into the (y-b)/m equation to get a range. Again testing the test value to the range.

```
//Take long_data and find the corresponding x values on the lines from point4-1
//and point 3-2.  These x values are a range.
//Test lat_data against those corresponding x values for a horizontal check
xcp1 = (long_data - b41) / m41;
xcp2 = (long_data - b32) / m32;
```

So in the end, we checked to see if the GPS coordinate was inside the vertical boundaries then horizontal boundaries. If both passed, "condition" was true and we were good. If one failed then we were bad.

```
if( (long_data <= ycp1) && (long_data >= ycp2) ) { //Check within vertical
boundaries
        if( (lat_data <= xcp2) && (lat_data >= xcp1) ) { //Check within
        horizontal boundaries
                return 1; //We're inside the fence
        }
} else {
        return 0; //DEAR GOD WE'RE BAD
}
```

Summary and Conclusion

In the end we win.

Changes from our initial proposal was the use of a LCD display.

References and Documentation: