SAN FRANCISCO
STATE UNIVERSITY

School of Engineering

Audio Spectrum Analyzer LED Display

Microcontroller Project

ENGR 478
Spring 2015

Group Name: Thirsty Thursdays		
Kimberly Loza	911059349	kloza@mail.sfsu.edu
Nestor Vasquez	911001863	nestor@mail.sfsu.edu
Rose Marie Dehesa	910694894	rmdehesa@mail.sfsu.edu

Date Submitted: 15 May 2015

Objective

The purpose of this project was to demonstrate the skills and knowledge acquired from a microcontroller's course curriculum by designing and implementing a microcontroller-based embedded system project. The goal of this project was to use the microcontroller to create an audio spectrum analyzer with a LED display.

Background

To design an audio spectrum analyzer, several different components were needed. Most audio analyzers have some type of identifier to signify whether there is a change in the audio's loudness. Audio's loudness can be represented as a range in numbers, a change in a light pattern, or a sound wave's amplitude. For this project, it was decided that using LEDs to display a change in a sound's loudness could be made possible if each LED represented a different range of audio loudness. In order to demonstrate that, the project required an analogRead pin connected to a microphone to take in audio and several digitalWrite pins to write to LEDs. Using 10 digitalWrite pins on the microcontroller made it possible to separate the 150 LEDs into ten segments.

Having 10 LED segments allowed each segment to specifically pick up a range of sound levels. If one of the segments picks up a sound level within its range it will light up; as well as, all the segments prior to the threshold reported by the analogRead. The lowest threshold value, the green LED, signifies the most sensitive LED range. Then as the threshold value increases in loudness, more LED segments will light up until it reaches its max threshold. The max threshold represents the 10^{th} LED segment with the red LEDs. The 10^{th} LED segment also is responsible for triggering the interrupt to make the on-board LED and all LEDs to blink. Reaching the max threshold lights-up all the LED segments, and this shows that a very loud sound was detected by the sound-sensor.

Motivation

The main motivation to create this project came from most of the microcontroller labs using LEDs. Making a project that could physically describe a change in the environment seemed like a very interesting project. This project's design goals were realistic to create from the material we have learned so far.

Summary of Materials Needed

- Over 150 generic LEDs
  There were green, dim yellow/red, and red LEDs

- 3 Bread Board

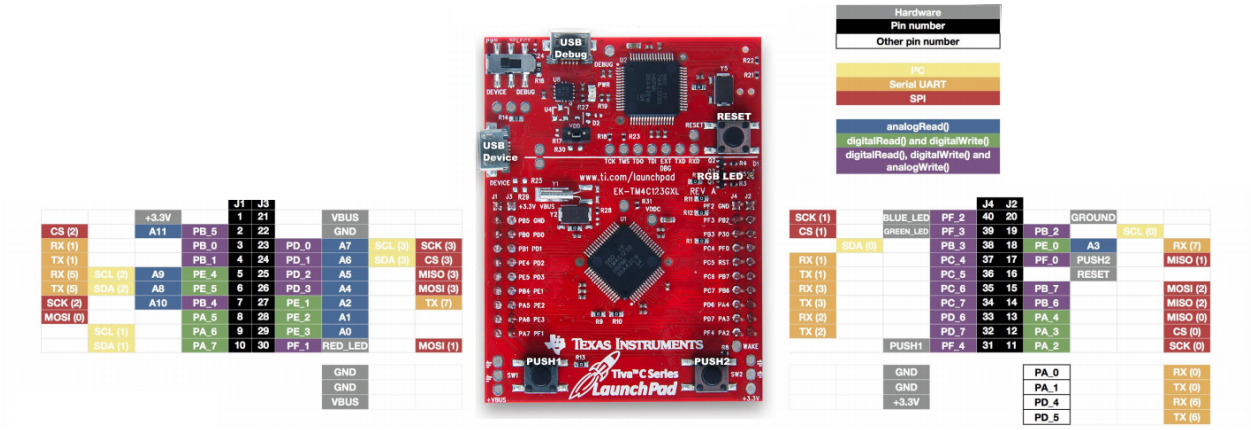
- Several Wires

- Grove Sound Sensor
  This device detects the sound strength of it's surroundings. It's design is based on the LM358 amplifier along with a electret microphone. It outputs an analog signal to the TI Launchpad as a threshold value.

Code Break Down

This project was created in Energia which required different pins than Kiel. The pin schematic is as follows:

Figure 1: Energia Pin Schematic



This project required the use of most of the purple pins depicted in the Energia Pin Schematic Figure above. Pins 31-40 represent the 10 LED segments, and pin 24 represents the sound-sensor.

Code:

```
/*****************************************************************************************************************

*       478 Project
*
*****************************************************************************************************************/

#define SOUND_SENSOR          24        // Sound Sensor
#define LED                   RED_LED   // LED pin
#define MAX_THRESHOLD_VALUE    2000     // Any number greater than max threshold will light up all 10 segments
#define MIN_THRESHOLD_VALUE    100      // The minimum threshold to light up segment 1
                               //The min should be 1/10th the MAX Threshold

//Thresholds for Segments
#define S2_THRESHOLD_VALUE    300
#define S3_THRESHOLD_VALUE    550
#define S4_THRESHOLD_VALUE    800
#define S5_THRESHOLD_VALUE    1000
#define S6_THRESHOLD_VALUE    1200
#define S7_THRESHOLD_VALUE    1400
#define S8_THRESHOLD_VALUE    1600
#define S9_THRESHOLD_VALUE    1700


#define ON                HIGH                // LED ON
#define OFF               LOW                 // LED OFF
#define _handle_led(x)     digitalWrite(LED, x)   // Handles on board LED

/* Holds the 10 segments for the LED sound equalizer display */
#define _handle_ledS1(x)   digitalWrite(31, x)    // Most left, Min value, Green      : LED PF_4
#define _handle_ledS2(x)   digitalWrite(32, x)    // Section 2 LEDS                   : LED PD_7
#define _handle_ledS3(x)   digitalWrite(33, x)    // Section 3 LEDS                   : LED PD_6
#define _handle_ledS4(x)   digitalWrite(34, x)    // Section 4 LEDS                   : LED PC_7
#define _handle_ledS5(x)   digitalWrite(35, x)    // Section 5 LEDS                   : LED PC_6
#define _handle_ledS6(x)   digitalWrite(36, x)    // Section 6 LEDS                   : LED PC_5
#define _handle_ledS7(x)   digitalWrite(37, x)    // Section 7 LEDS                   : LED PC_4
#define _handle_ledS8(x)   digitalWrite(38, x)    // Section 8 LEDS                   : LED PB_3
#define _handle_ledS9(x)   digitalWrite(39, x)    // Section 9 LEDS                   : LED PF_3
#define _handle_ledS10(x)  digitalWrite(40, x)    // Most right, Max value, Red LEDs  : LED PF_2


/* Global Variables */
int sound_value = 0;
int state = 0;

void setup() {
   /* Initialize led pin */
   pinMode(LED, OUTPUT);
   digitalWrite(LED, LOW);

   // Intialize the min value S1
   pinMode(31, OUTPUT);
   digitalWrite(31, LOW);

   // Intialize second section, S2
   pinMode(32, OUTPUT);
   digitalWrite(32, LOW);

   // Intialize S3
   pinMode(33, OUTPUT);
   digitalWrite(33, LOW);

   // Intialize S4
   pinMode(34, OUTPUT);
   digitalWrite(34, LOW);

   // Intialize S5
   pinMode(35, OUTPUT);
```

```
    digitalWrite(35, LOW);

    // Intialize S6
    pinMode(36, OUTPUT);
    digitalWrite(36, LOW);

    // Intialize S7
    pinMode(37, OUTPUT);
    digitalWrite(37, LOW);

    // Intialize S8
    pinMode(38, OUTPUT);
    digitalWrite(38, LOW);

    // Intialize S9
    pinMode(39, OUTPUT);
    digitalWrite(39, LOW);

    // Intialize S10
    pinMode(40, OUTPUT);
    digitalWrite(40, LOW);

    //Interrupt is fired whenever the sound sensor senses a value greater than MAX Threshold
   attachInterrupt(40, blink, RISING); // Rising triggers when pin goes from low to high
}

void loop() {
  // Critical time-sensitive code
  noInterrupts();

  // Reads the sound value as an integer
  sound_value = analogRead(SOUND_SENSOR);


  delay(100);

  if (sound_value <  MIN_THRESHOLD_VALUE) {./
    state = 0;
  } else if ((MIN_THRESHOLD_VALUE<=sound_value)&&(sound_value<S2_THRESHOLD_VALUE)) {
    state = 1;
  } else if ((S2_THRESHOLD_VALUE<=sound_value)&&(sound_value<S3_THRESHOLD_VALUE)) {
    state = 2;
  } else if ((S3_THRESHOLD_VALUE<=sound_value)&&(sound_value<S4_THRESHOLD_VALUE)) {
    state = 3;
  } else if ((S4_THRESHOLD_VALUE<=sound_value)&&(sound_value<S5_THRESHOLD_VALUE)) {
    state = 4;
  } else if ((S5_THRESHOLD_VALUE<=sound_value)&&(sound_value<S6_THRESHOLD_VALUE)) {
    state = 5;
  } else if ((S6_THRESHOLD_VALUE<=sound_value)&&(sound_value<S7_THRESHOLD_VALUE)){
    state = 6;
  } else if ((S7_THRESHOLD_VALUE<=sound_value)&&(sound_value<S8_THRESHOLD_VALUE)){
    state = 7;
  } else if ((S8_THRESHOLD_VALUE<=sound_value)&&(sound_value<S9_THRESHOLD_VALUE)){
    state = 8;
  } else if ((S9_THRESHOLD_VALUE<=sound_value)&&(sound_value<MAX_THRESHOLD_VALUE)){
    state = 9;
  } else if (sound_value >= MAX_THRESHOLD_VALUE){
    state = 10;
  }
```

```
 // All other code that can be interrupted
interrupts();

switch (state) {
  case 0:
       //Do nothing
  break;

  case 1:// S1
       _handle_ledS1(ON);
  break;

  case 2: // S2
       _handle_ledS1(ON);
       delay(10);
       _handle_ledS2(ON);
  break;

  case 3: // S3
       _handle_ledS1(ON);
       delay(10);
       _handle_ledS2(ON);
       delay(10);
       _handle_ledS3(ON);
  break;

  case 4:  // S4
       _handle_ledS1(ON);
       delay(10);
       _handle_ledS2(ON);
       delay(10);
       _handle_ledS3(ON);
       delay(10);
       _handle_ledS4(ON);
  break;

  case 5: // S5
       _handle_ledS1(ON);
       delay(10);
       _handle_ledS2(ON);
       delay(10);
       _handle_ledS3(ON);
       delay(10);
       _handle_ledS4(ON);
       delay(10);
       _handle_ledS5(ON);
  break;

  case 6: // S6
        _handle_ledS1(ON);
       delay(10);
       _handle_ledS2(ON);
       delay(10);
       _handle_ledS3(ON);
       delay(10);
       _handle_ledS4(ON);
       delay(10);
       _handle_ledS5(ON);
       delay(10);
       _handle_ledS6(ON);
  break;
```

```
case 7:  // S7
        _handle_ledS1(ON);
        delay(10);
        _handle_ledS2(ON);
        delay(10);
        _handle_ledS3(ON);
        delay(10);
        _handle_ledS4(ON);
        delay(10);
        _handle_ledS5(ON);
        delay(10);
        _handle_ledS6(ON);
        delay(10);
        _handle_ledS7(ON);
break;

case 8: // S8
        _handle_ledS1(ON);
        delay(10);
        _handle_ledS2(ON);
        delay(10);
        _handle_ledS3(ON);
        delay(10);
        _handle_ledS4(ON);
        delay(10);
        _handle_ledS5(ON);
        delay(10);
        _handle_ledS6(ON);
        delay(10);
        _handle_ledS7(ON);
        delay(10);
        _handle_ledS8(ON);
break;


case 9: // S9
        _handle_ledS1(ON);
        delay(10);
        _handle_ledS2(ON);
        delay(10);
        _handle_ledS3(ON);
        delay(10);
        _handle_ledS4(ON);
        delay(10);
        _handle_ledS5(ON);
        delay(10);
        _handle_ledS6(ON);
        delay(10);
        _handle_ledS7(ON);
        delay(10);
        _handle_ledS8(ON);
        delay(10);
        _handle_ledS9(ON);
break;
```

```
    case 10: // S10
            _handle_ledS1(ON);
            delay(10);
            _handle_ledS2(ON);
            _handle_ledS3(ON);
            delay(10);
            _handle_ledS4(ON);
            delay(10);
            _handle_ledS5(ON);
            delay(10);
            _handle_ledS6(ON);
            delay(10);
            _handle_ledS7(ON);
            delay(10);
            _handle_ledS8(ON);
            delay(10);
            _handle_ledS9(ON);
            delay(10);
            _handle_ledS10(ON); // Causes an interrupt
    break;

    default:
        _handle_led(OFF);
        _handle_ledS1(OFF);
        _handle_ledS2(OFF);
        _handle_ledS3(OFF);
        _handle_ledS4(OFF);
        _handle_ledS5(OFF);
        _handle_ledS6(OFF);
        _handle_ledS7(OFF);
        _handle_ledS8(OFF);
        _handle_ledS9(OFF);
        _handle_ledS10(OFF);  ;
     break;
  }

// Set LEDs back to default
  delay(100);
  // Sets all LEDs back to OFF default
  _handle_led(OFF);
  _handle_ledS1(OFF);
  _handle_ledS2(OFF);
  _handle_ledS3(OFF);
  _handle_ledS4(OFF);
  _handle_ledS5(OFF);
  _handle_ledS6(OFF);
  _handle_ledS7(OFF);
  _handle_ledS8(OFF);
  _handle_ledS9(OFF);
  _handle_ledS10(OFF);
}

void blink() {
  static boolean pulse = HIGH;
  pulse = (pulse==HIGH) ? LOW : HIGH; // Makes the LEDs blink

    _handle_led(pulse);
    delay (200);
    _handle_ledS1(pulse);
    _handle_ledS2(pulse);
    _handle_ledS3(pulse);
    _handle_ledS4(pulse);
    _handle_ledS5(pulse);
    _handle_ledS6(pulse);
    _handle_ledS7(pulse);
    _handle_ledS8(pulse);
    _handle_ledS9(pulse);
    _handle_ledS10(pulse);
    delay (300);
}
```

Interrupt

Using interrupts made it possible to separate the time sensitive code from all other code that could be interrupted. The interrupt was initialized as part of the 10th LED segment. This line of code initializes the interrupt:

```
//Interrupt is fired whenever the sound sensor senses a value greater than MAX Threshold
attachInterrupt(40, blink, RISING); // Rising triggers when pin goes from low to high
```

Then when the state 10 is triggered by the max threshold, this case statement is triggered:

```
case 10: // S10
        _handle_ledS1(ON);
        delay(10);
        _handle_ledS2(ON);
        _handle_ledS3(ON);
        delay(10);
        _handle_ledS4(ON);
        delay(10);
        _handle_ledS5(ON);
        delay(10);
        _handle_ledS6(ON);
        delay(10);
        _handle_ledS7(ON);
        delay(10);
        _handle_ledS8(ON);
        delay(10);
        _handle_ledS9(ON);
        delay(10);
        _handle_ledS10(ON); // Causes an interrupt
```

When the interrupt as the last line is triggered, the blink function is called:

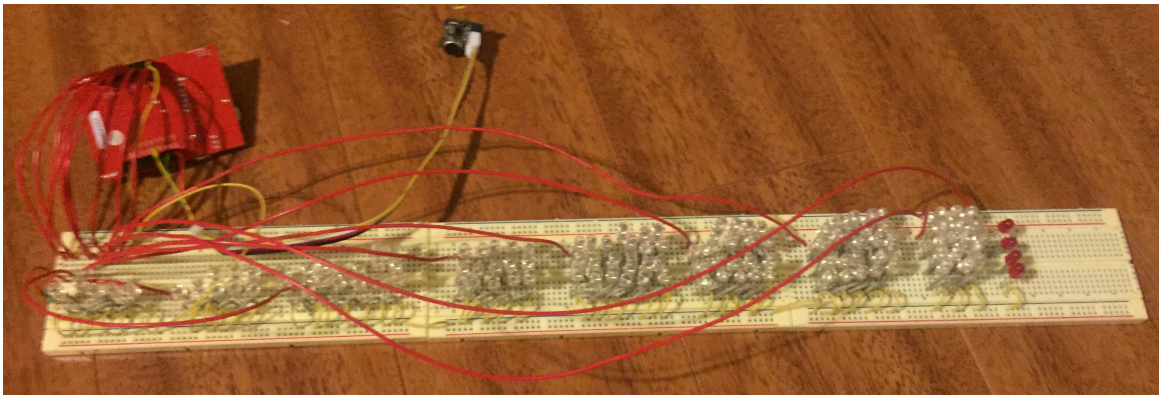
```
void blink() {
  static boolean pulse = HIGH;
  // Makes the LEDs blink
    pulse = (pulse==HIGH) ? LOW : HIGH;

    _handle_led(pulse);
    delay (200);
    _handle_ledS1(pulse);
    _handle_ledS2(pulse);
    _handle_ledS3(pulse);
    _handle_ledS4(pulse);
    _handle_ledS5(pulse);
    _handle_ledS6(pulse);
    _handle_ledS7(pulse);
    _handle_ledS8(pulse);
    _handle_ledS9(pulse);
    _handle_ledS10(pulse);
    delay (300);
}
```

The blink function makes the on-board LED and all LED segments blink, and then once completed it proceeds to set all LEDs back to their off default.

End Product

Here is the end product of the audio spectrum analyzer with the lights on and disconnected:



Here it is again with the audio off and only part of the spectrum lit up for a low sound:

Then finally here it is with the 10th segment activated and interrupt triggered:

The interrupt can be seen at the top left corner of the picture with the blue light.

Constraints

Economic constraint

For this design we feel that the performance/cost ratio was reasonable. We were able to implement the design to do what it was designed for without spending too much money. The only part that cost us money was the sound sensor. We were able to obtain the breadboards, LEDs, and wires at no cost. The designed turned out to be cost effective.

Sustainability

The projects looks like it is sustainable—we left it on/running for long period of times and it was working fine the whole time without causing no harm or malfunctioning.

Environmental Constraint

The project is environment friendly. We designed our display to be implement by LEDs, which are likely the most environmentally friendly lighting options available. LEDs provide many benefits— last as much as 20 times longer than other lighting sources, LEDs consume way less energy than other light sources, contain no mercury, produce less heat and have a longer life than other lighting sources.

Conclusion

Overall, the Audio Spectrum Analyzer LED Display project served as an excellent way to use the skills and knowledge acquired from a microcontroller's course curriculum to design and implementing a microcontroller-based embedded system. For future projects, we would like to improve this design by using the originally planned Adafruit LED strip, and also make the audio sampling closer to real-time. This project made the knowledge learned in the microcontroller applicable to a real-world product, which served as an interesting project. For the most part, we were very satisfied with the end product and hope to explore more possibilities with even more additional LEDs.