Introduction

The purpose of the lab was to familiarize the participants with micro-controllers, like the msp430, and the workings of analog inputs from the msp430 booster pack. To do such a thing we designed a 7-bit analog input from a joystick to two seven-segment display to facilitate in learning the basics of working with and converting analog input to two 4-bit binary outputs using the msp430 and its booster pack. The designing of the 4-bit counter was to facilitate in the learning and technical know-how of working with the MCU, and the seven segment displays, before actually programming and building the analog input part of the lab. Mostly all the designing and simulation was done on TinkerCad, where the Arduino was also programmed to act as the receiver and output of analog inputs. This was then built in a lab with a msp430 board.

Procedure

We first built the four-bit counter displayed on a seven-segment display, using inputs from the msp430. The design had not changed since the last lab procedure. The only changes were made to the pin outputs for the msp430 in the code. To recap, the msp430 outputs to 5 pins on the board where 4 pins connect to the BCD inputs, while one pin connects to the latch enable on the chip. The BCD then outputs to 7 pins which map to the 7-segments on the 7-segment display. The code itself, uses a few switch cases statements to determine the output of the pins on the MCU board, for the specific number. For the physical build, all that needs to be done is to connect the wires properly on the breadboard, with the VCC and Ground provided from the MCU. After mapping the wires properly, simply connect MCU to a laptop/computer with energeia on it, and then flash a the 4-bit counter program onto the controller. For my program I needed 4 consecutive pins for the code to work. The pins I chose were 33-36.

For the analog input to 7-segment display, we simply used the joystick on the msp430 booster pack, by connecting the pins on the msp430 to the booster pack inputs. This made it so that certain pins were not available to be used. Which had us adapt our code so that it did not require two pairs of consecutive pins to run. By doing so we changed our code from the last lab procedure. For the physical details, just make another copy of the four-bit counter described above. Have the BCD converters be mapped properly to the correct output pins on the msp430. After doing so, flash the msp430 with the program changes such that it recognizes the y-axial movement on the joystick and correctly maps it to a number between 0-99. The numbers are then broken into two different digits which are then shown on the seven-segment display.

Experimental Results

The following figures details completely built circuits using the msp430 on breadboards with the IC's and the display.

All code is in the References section, so, if need be, you can just look there for the code.

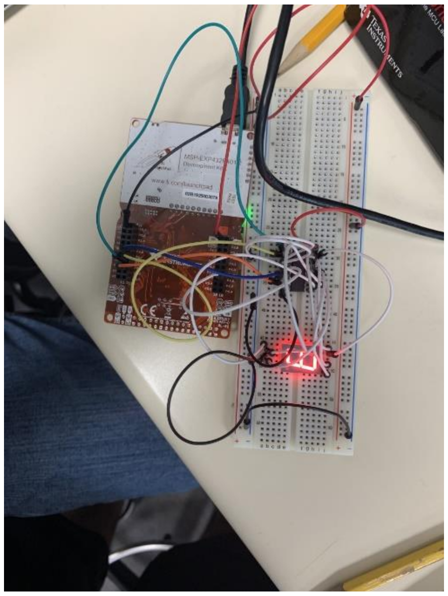

Figure 1.1: Shows the completed 4-bit counter.

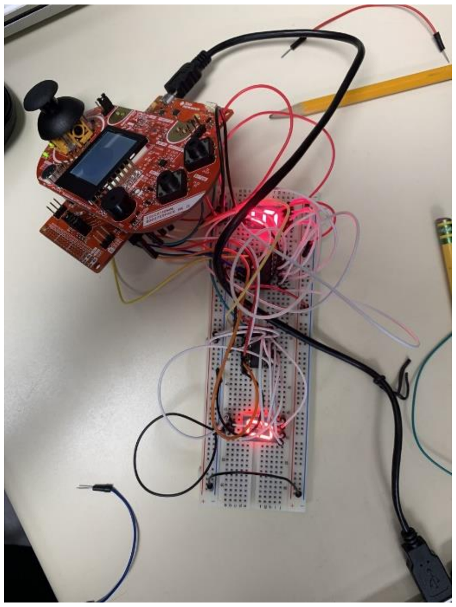

Figure 1.2: Showcases the completed 7-bit circuit, mapping 0-99 from the joystick input.

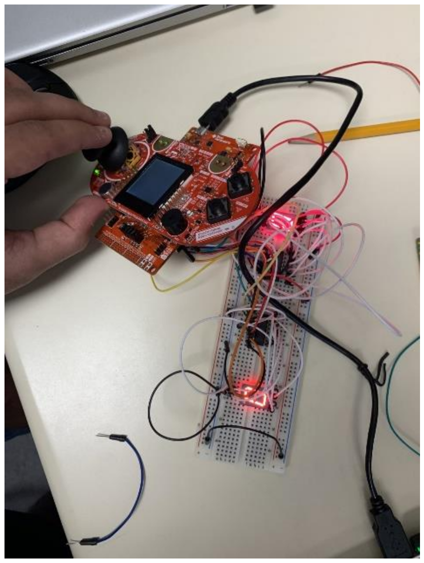

Figure 1.3: Showcases the analog input circuit working when the joystick is pulled down for the y axis.

Discussion and Conclusion

The lab was simple learning lesson on how to work with physical hardware and the msp430 mcu. Especially the booster pack with the joystick that was the analog input. I personally learned how to use the Analog pins on the msp430, and the related methods to Read and Write Analog inputs. Another substantial learning was done regarding documentation on the msp430 are different than a Arduino, since msp430 does not have as clear documentation as the Arduino.

References

Code for the 4-bit counter (Lab):

```
int W_PIN, X_PIN;     //USE THESE VARIABLES FOR W and X PINS
int Y_PIN, Z_PIN;     //USE THESE VARIABLES FOR Y and Z PINS
int w_value, x_value;   //USE THESE VARIABLES FOR
           //THE VALUES OF W and X
int y_value, z_value;   //USE THESE VARIABLES FOR
           //THE VALUES OF Y and Z
int counter = 0;    //USE THIS AS YOUR COUNTER VARIABLE

void printOutput();   //Function for printing output for
          //for varifivation

void setup()
{

  /*
  Put code here that you would like to happen once
  at the start of the program.
  */
  //YOUR CODE HERE

  W_PIN = 36;
  X_PIN = 35;
  Y_PIN = 34;
  Z_PIN = 33;

  pinMode(W_PIN, OUTPUT);
  pinMode(X_PIN, OUTPUT);
  pinMode(Y_PIN, OUTPUT);
  pinMode(Z_PIN, OUTPUT);

  pinMode(8, OUTPUT);
  digitalWrite(8, LOW);



  // DO NOT EDIT THE CODE BELOW THIS
  Serial.begin(9600); //Initilize the serial monitor
}

void loop()
{
  /*
  Put code here that you would like to be repeated.
  It is recomended that you increment your counter
  BEFORE you set the value of the pins
  */

  //YOUR CODE HERE

  counter+=1;

  delay(1000);

  digitalWrite(W_PIN, LOW);
  digitalWrite(X_PIN, LOW);
  digitalWrite(Y_PIN, LOW);
  digitalWrite(Z_PIN, LOW);

    switch(counter){
      case 1:
        w_value = 0;
        x_value = 0;
        y_value = 0;
        z_value = 0;
        break;
      case 2:
        w_value = 0;
        x_value = 0;
        y_value = 0;
        z_value = 1;
        digitalWrite(Z_PIN, HIGH);
        break;
      case 3:
        w_value = 0;
        x_value = 0;
        y_value = 1;
        z_value = 0;
        digitalWrite(Y_PIN, HIGH);
        break;
      case 4:
        w_value = 0;
        x_value = 0;
        y_value = 1;
        z_value = 1;
        digitalWrite(Y_PIN, HIGH);
        digitalWrite(Z_PIN, HIGH);
        break;
        break;
      case 5:
        w_value = 0;
        x_value = 1;
        y_value = 0;
        z_value = 0;
        digitalWrite(X_PIN, HIGH);
        break;
      case 6:
        w_value = 0;
        x_value = 1;
        y_value = 0;
        z_value = 1;
        digitalWrite(X_PIN, HIGH);
        digitalWrite(Z_PIN, HIGH);
        break;
      case 7:
        w_value = 0;
        x_value = 1;
        y_value = 1;
        z_value = 0;
        digitalWrite(Y_PIN, HIGH);
        digitalWrite(X_PIN, HIGH);
        break;
      case 8:
        w_value = 0;
        x_value = 1;
        y_value = 1;
        z_value = 1;
        digitalWrite(X_PIN, HIGH);
        digitalWrite(Y_PIN, HIGH);
        digitalWrite(Z_PIN, HIGH);
        break;
      case 9:
        w_value = 1;
        x_value = 0;
        y_value = 0;
        z_value = 0;
        digitalWrite(W_PIN, HIGH);
        break;
      case 10:
        w_value = 1;
        x_value = 0;
        y_value = 0;
        z_value = 1;
        digitalWrite(W_PIN, HIGH);
        digitalWrite(Z_PIN, HIGH);
        counter = 0;
        break;
     }



    //DO NOT EDIT THE CODE BELOW THIS
    printOutput(); //Print output to serial monitor
}

/*
This function prints the output to the serial console.
Use the serial console to verify that your values are correct
for your counter.

NOTE: If you increment your counter after you set the values,
then the value of the counter will be off by 1. As long as both
the binary values and the counter increment correctly, your design
will work.
*/
void printOutput()
{
  String outputString = "Counter = ";
  outputString += counter;
  outputString += ", WXYZ = ";
  outputString += w_value;
  outputString += x_value;
  outputString += y_value;
  outputString += z_value;
  outputString += ", W = ";
  outputString += w_value;
  outputString += ", X = ";
  outputString += x_value;
  outputString += ", Y = ";
  outputString += y_value;
  outputString += ", Z = ";
  outputString += z_value;
  Serial.println(outputString);
  return;
}

/*
Serial Monitor is below
*/
```

Code for the 8-bit analog input (**Lab**):

```
/*
```

```
 Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the serial monitor.

 The circuit:
 * potentiometer connected to analog pin 0.
   Center pin of the potentiometer goes to the analog pin.
   side pins of the potentiometer go to +5V and ground
 * LED connected from digital pin 9 to ground

 created 29 Dec. 2008
 modified 9 Apr 2012
 by Tom Igoe

 This example code is in the public domain.

 */

// These constants won't change.  They're used to give names
// to the pins used:
const int analogInPin = 26;  // Analog input pin that the potentiometer is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

const int bitOne = 27;
const int bitTwo = 28;
const int bitThree = 29;
const int bitFour = 30;

const int bitFive =  12;
const int bitSix = 14;
const int bitSeven = 18;
const int bitEight = 35;

const int latchEn = 36;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);

  pinMode(bitOne, OUTPUT);
  pinMode(bitTwo, OUTPUT);
  pinMode(bitThree, OUTPUT);
  pinMode(bitFour, OUTPUT);

  pinMode(bitFive, OUTPUT);
  pinMode(bitSix, OUTPUT);
  pinMode(bitSeven, OUTPUT);
  pinMode(bitEight, OUTPUT);

  pinMode(latchEn, OUTPUT);
  pinMode(analogInPin, INPUT);
}

void loop() {

  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 99);

  int temp = outputValue;
  int digitOne = temp / 10;
  int digitTwo = temp % 10;

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  Serial.print("\t digitOne = ");
  Serial.println(digitOne);

  Serial.print("\t digitTwo = ");
  Serial.println(digitTwo);

  digitalWrite(bitOne, LOW);
  digitalWrite(bitTwo, LOW);
  digitalWrite(bitThree, LOW);
  digitalWrite(bitFour, LOW);

  digitalWrite(bitFive, LOW);
  digitalWrite(bitSix, LOW);
  digitalWrite(bitSeven, LOW);
  digitalWrite(bitEight, LOW);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
 // delay(2);

  for(int i = 0; i < 4; i++){
    int tempTwo = 0;

    switch(i){
      case 0:
        tempTwo = 12;
        break;
      case 1:
        tempTwo = 14;
        break;
      case 2:
        tempTwo = 18;
        break;
      case 3:
```