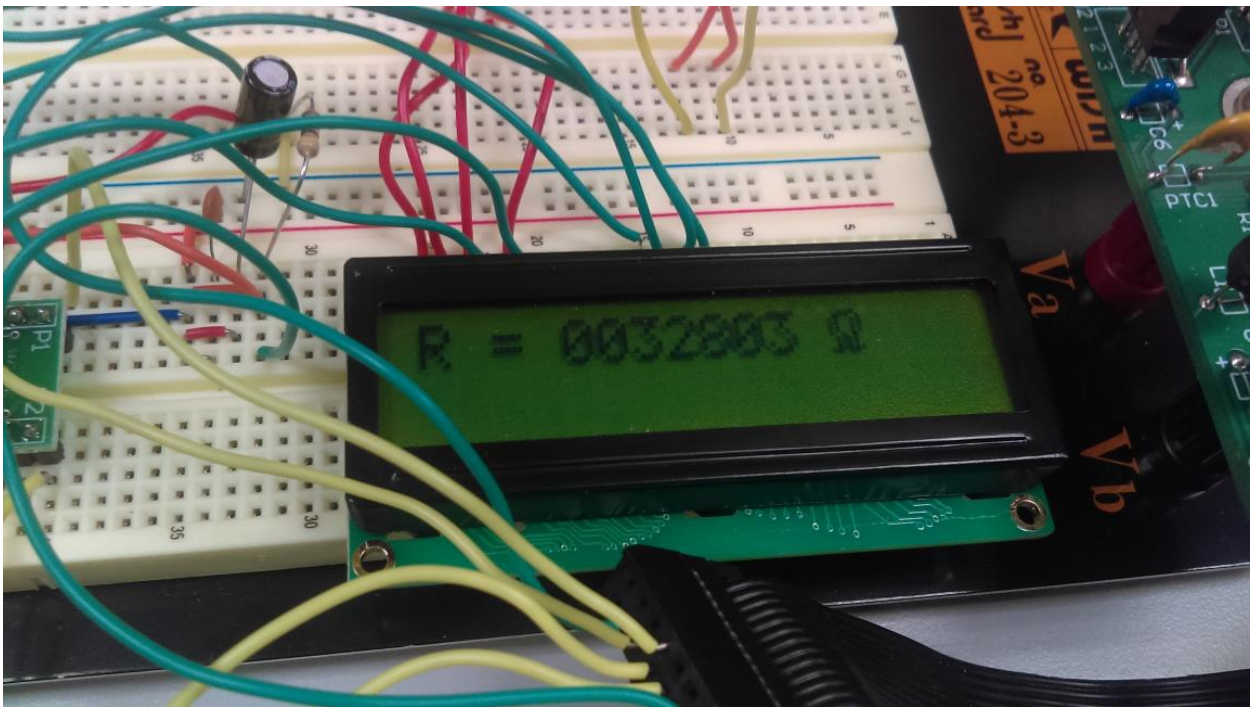
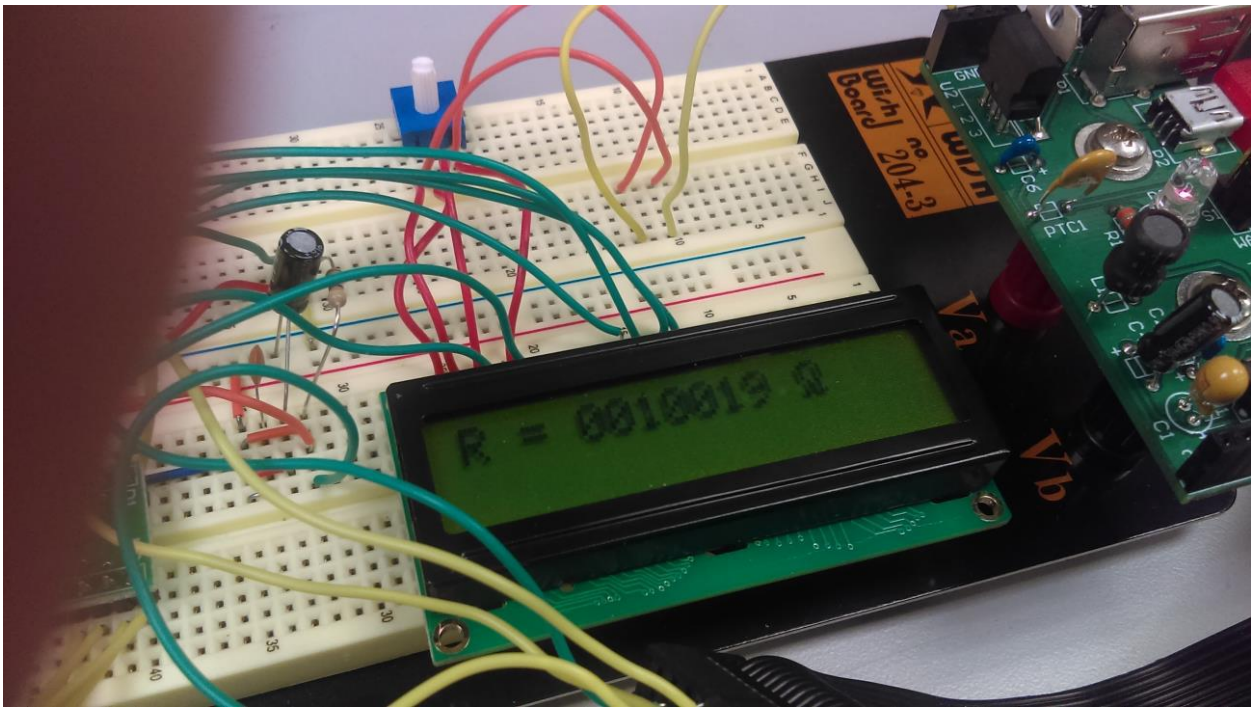
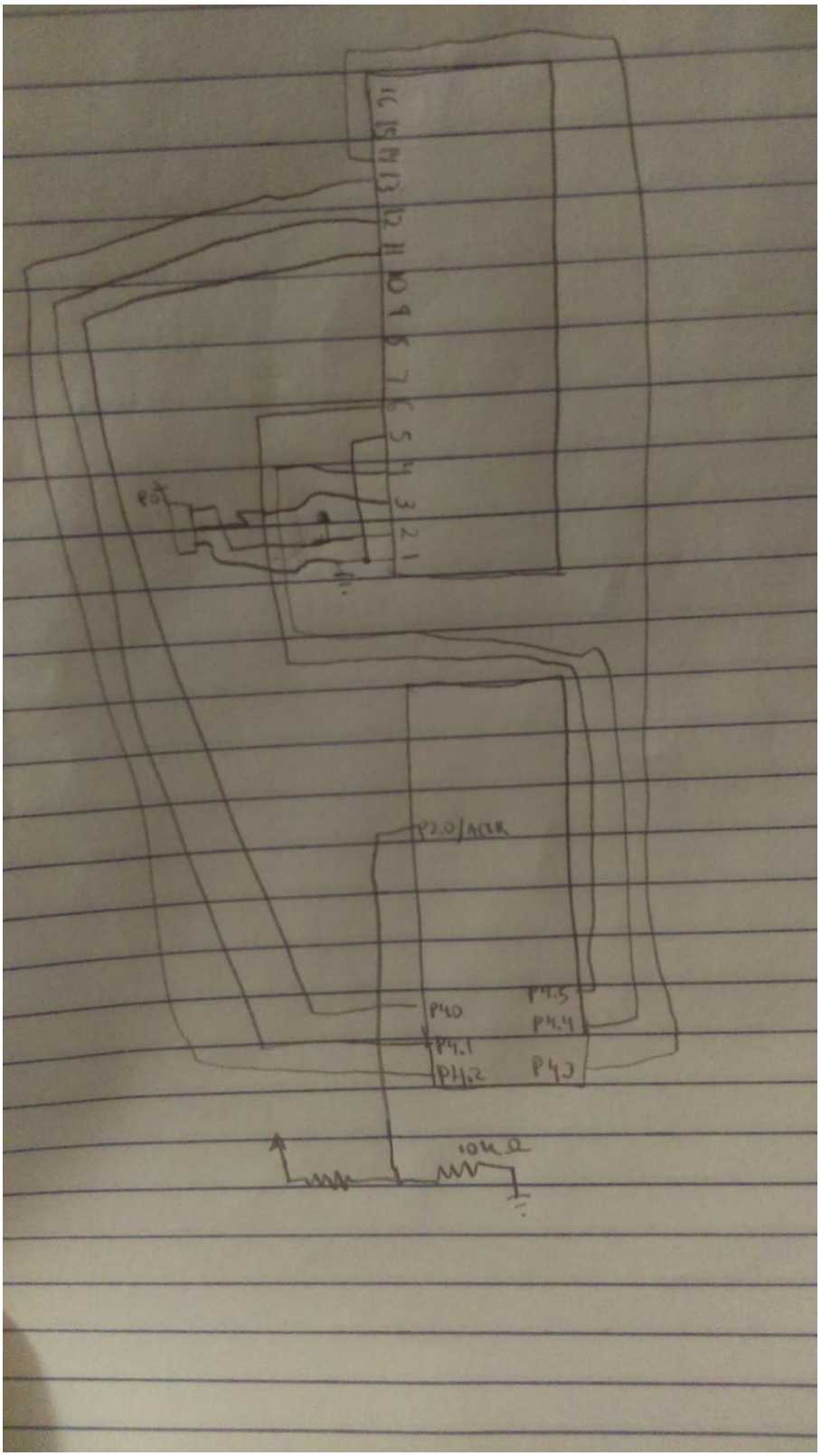
Ricardo Stefano Reyna 28/May/16
21010521
Basic Micro-Controller Applications (I/O, LCD Interface, A/D)
Introduction
This module consisted of two parts. In the first part was using a switch to toggle between a led and a buzzer. The second part consisted of using the adc to read the voltage and convert it to resistance using the voltage divider formula, and finally print the result on the lcd
Design
For the first part I connected each component to three different GPIO. One was an input (switch), and the other two were outputs. Depending on the position of the switch it would either blink the led at 2HZ or make the buzzer sound at 2Khz.
The second part consisted of an ohm meter. First I connected the lcd following the provided diagram except for the R/W being connected to ground, RS connected to P4.4 and enable connected to P4.5. For the ADC I use pin 8 which shares the value of P2.0 and the interrupt where I did 20 samplings and got the average. For the voltage divider I used a 10kohm resistor.
The following are two pictures of the lcd showing the values and a diagram of the lcd to microcontroller connections:



33k ohms



1kohm


16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
P2.0/ACLK
P4.0
P4.1
P4.2
P4.5
P4.4
P4.3
10kΩ

Conclusion
Part one got us to familiarize ourselves with the microcontroller and the GPIO. For part two I learned to set up the lcd, use the adc and most importantly interrupts.

Note 0: Part 1 code must be uploaded separately.
Note 1: Uploaded report will be automatically reviewed in E-learning
Note 2: Effective technical reports convey the key essential information in a concise and clear manner (Focus on quality, not length)

Appendix. (Code)

Part1

```
#include <msp430.h>


/*
 * main.c
 */
int i;


void delay_us(unsigned int us)
{
   unsigned int i;
   for (i = 0; i<= us/2; i++)
     __delay_cycles(1);
}


void beep(unsigned int note)
{
   long delay = (long)(10000/note);  //Note's semiperiod.
   P1OUT |= 0x01;     //Set P1
   delay_us(delay);   //play half
   P1OUT &= ~0x01;    //reset
   delay_us(delay);   //play half
```

```
}

int main(void) {
   WDTCTL = WDTPW | WDTHOLD;        // Stop watchdog timer
        P2DIR = 0x01; //Set output to P2
        P1DIR = 0x01; //Set output to P1
        P3DIR = 0x00; //Set input to P3
        P2OUT = 0x00;
        while(1) {
                if (P3IN & 0x01) {
                        P2OUT ^= 0x01;
                        for(i = 0; i < 20500; i++); //Trial and error with the o-scope
                }
                else {
                        beep(255); //Trial and error with the o-scope
                }
        }
        return 0;
}
```

Part2

```
#include <msp430.h>
/*
 * main.c
 */
#define HOME   0x02     // Home
#define CLEAR   0x01     // Clear screen and CR

void lcd_command(char);
void lcd_char(char);
```

```
void lcd_init(void);
void lcd_string(char*);
void adc_setup(void);
unsigned int adc_sam20(void);
void getResistance(unsigned int);
char uf_lcd_temp;
char uf_lcd_temp2;
char uf_lcd_x;
char ohm = 222;

int main(void) {
   WDTCTL = WDTPW | WDTHOLD;        // Stop watchdog timer
   lcd_init();
   adc_setup();

   while (1) {
         unsigned int avg_adc = 0;
         avg_adc = adc_sam20();          // sample 20 times and average out the result
         getResistance(avg_adc);         // print out the result to the screen
   }

         return 0;
}

void lcd_string(char *str) {
         //This writes words yo
         while (*str != 0) {
                  lcd_char(*str);
                  *str++;
```

```
        }
}

void lcd_command(char uf_lcd_x){
        P4DIR = 0xFF;
        uf_lcd_temp = uf_lcd_x;
        P4OUT = 0x00;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x >> 4;
        uf_lcd_x = uf_lcd_x & 0x0F;
        uf_lcd_x = uf_lcd_x | 0x20;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x & 0x0F;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        P4OUT = 0x00;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_temp;
        uf_lcd_x = uf_lcd_x & 0x0F;
        uf_lcd_x = uf_lcd_x | 0x20;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x & 0x0F;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
}

void lcd_char(char uf_lcd_x){
```

```
        P4DIR = 0xFF;
        uf_lcd_temp = uf_lcd_x;
        P4OUT = 0x10;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x >> 4;
        uf_lcd_x = uf_lcd_x & 0x0F;
        uf_lcd_x = uf_lcd_x | 0x30;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x & 0x1F;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        P4OUT = 0x10;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_temp;
        uf_lcd_x = uf_lcd_x & 0x0F;
        uf_lcd_x = uf_lcd_x | 0x30;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
        uf_lcd_x = uf_lcd_x & 0x1F;
        P4OUT = uf_lcd_x;
        __delay_cycles(22000);
}

void lcd_init(void){
        lcd_command(0x33);
        lcd_command(0x32);
        lcd_command(0x2C);
        lcd_command(0x0C);
```

```
	lcd_command(0x01);
}

// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void) {
	__bic_SR_register_on_exit(CPUOFF);	// Return to active mode
}

void adc_setup(void) {
	// ADC
	ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE; //0x1018, 16xADC10CLks, ADC10 on, and ADC10 interupt enable
	ADC10AE0 |= 0x01; //Enable reg 0
}

unsigned int adc_sam20(void) {
	unsigned int i;
	volatile unsigned int sum = 0;
	volatile unsigned int value;
	for(i = 0; i < 20; i++)
	{
		ADC10CTL0 |= ENC + ADC10SC;		// Start the conversion and enable conversion
		__bis_SR_register(CPUOFF + GIE);	// call ISR, low power mode0
		value = ADC10MEM;					// Save measured value
		sum += value;
	}
	return sum / 20;					// Average result
```

```
}


void getResistance(unsigned int adcIn) {
	static unsigned short flag = 0;
	volatile float voltage = ((float)adcIn/1023)*3.3; //average*(accuracy =
span(3.3V)/resolution(1024)), uP is from 0-1023
	float resistance = 1000*(33/voltage-10); //using voltage divider, 10k as a reference resistor


	if (adcIn > 930 || adcIn < 10) {
		if(flag == 0) {
			lcd_command(CLEAR);
			flag = 1;
		}
		lcd_string("Out of Range");
		lcd_command(HOME);
	}
	else {
		flag = 0;
		unsigned long intRes = resistance;
		int numArray[7];
		int i;
		char numToChar;
		lcd_string("R = ");

		for(i = 6 ;i >= 0; i--) {
			numArray[i] = intRes % 10;  //get last digit
			intRes /= 10;                                   //make number smaller
		}
```

```
                for(i = 0; i < 7; i++) {
                        numToChar = numArray[i] +'0'; //atoi pretty much
                        lcd_char(numToChar);
                }
                lcd_char(' ');
                lcd_char(ohm);
                lcd_command(HOME);
        }
}
```