EGR 326-908: Laboratory 3

**Lab 3: Capture and Compare with the MSP432**

Author: Professor:

Dixit Gurung Dr. Krug

Email:gurungd@mail.gvsu.edu

09/28/2020

**OBJECTIVE:**

The purpose of this laboratory was to create a program in CCS for the MSP432 to interface with the ultrasonic sensor and led using TimerA capture and compare.

**EQUIPMENT:**

**Table 1:** Detailed information on all apparatus used in the lab

|  |  |  |  |
| --- | --- | --- | --- |
| Part / Apprartus name/ Software | Description | Serial number | Notes / measured values |
| Lenovo Y- 700 | Personal laptop | PF0N7E08 | Windows 10 Home |
| CCS | IDE | - | Version 9.0.1 |
| MSP432 | 32 Bit Micro Controller | 02B1839001431 | - |
| LEDs | Red | - | - |
| Resistors | 10k Ohm  1k Ohm  2.2k Ohm | - | - |
| Ultra sonic sensor | HC - SR04 | - | - |
| Bread Board | - | - | - |

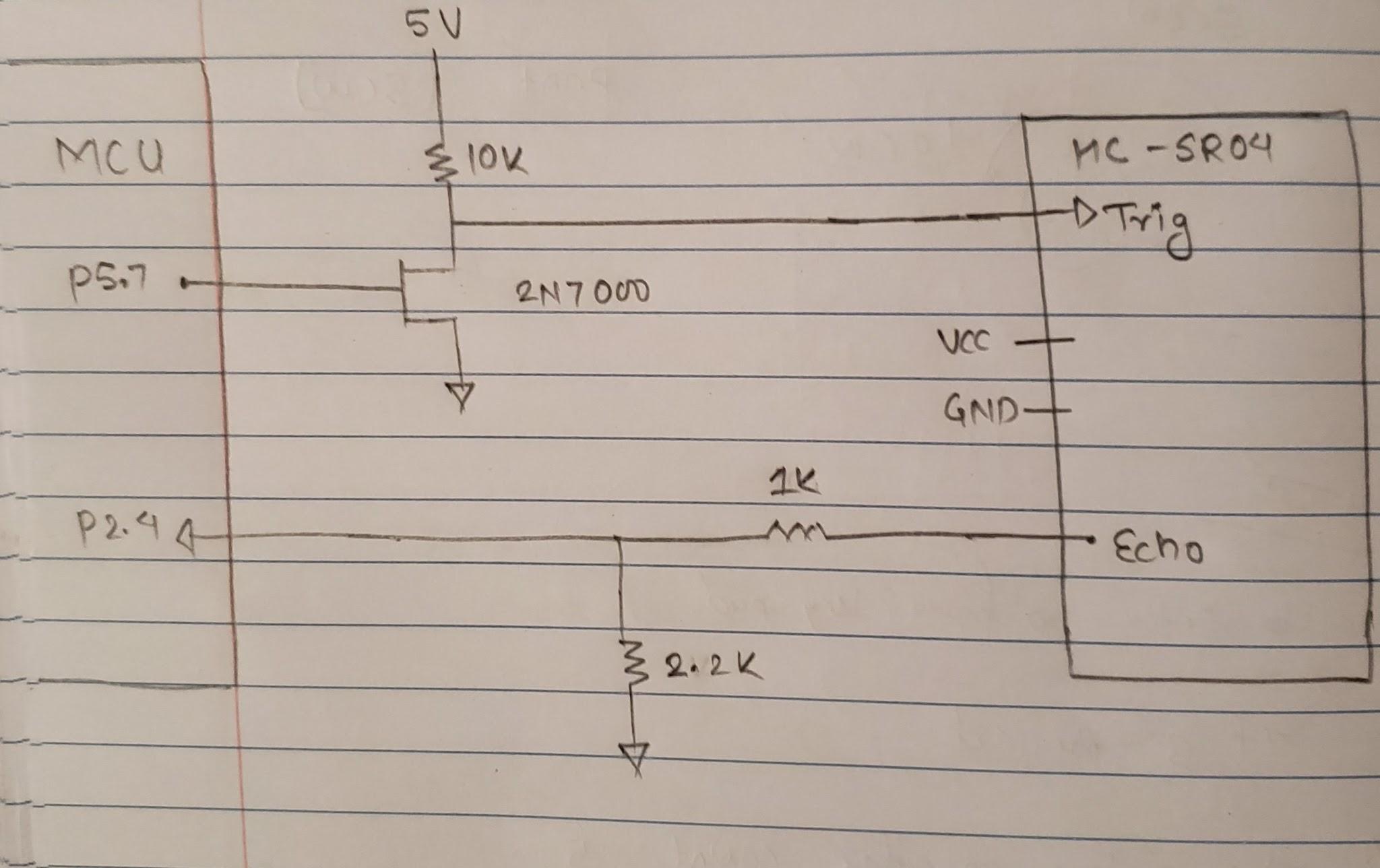
**Table 2:** Ports and pins used for buttons and LEDs

|  |  |
| --- | --- |
|  | Pins used |
| Trig pin | P5.7 |
| Echo pin | P2.4 |
| Red LED | P7.6 |

**PROCEDURE:**

**PART I**

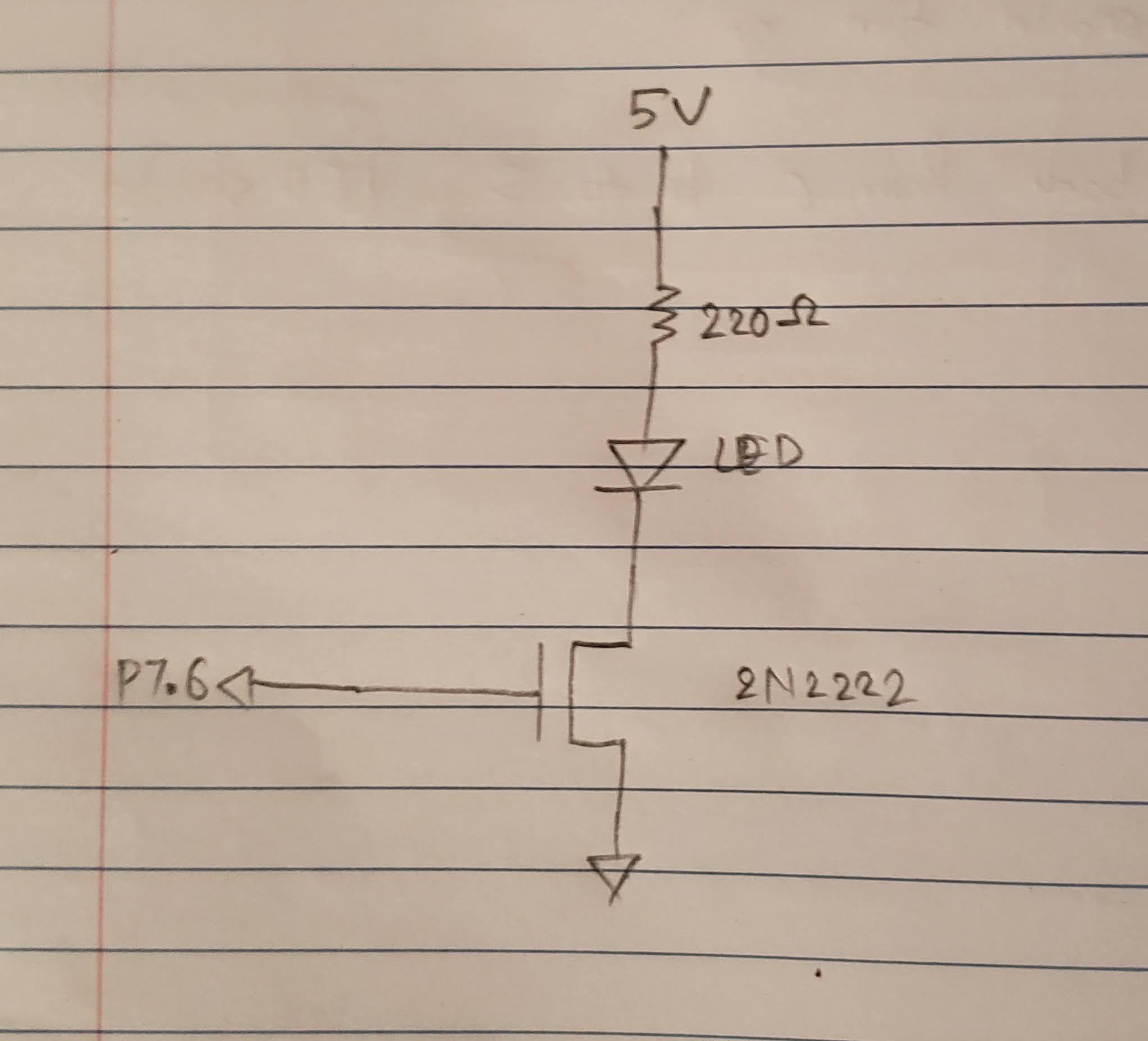
Part one of the laboratory required users to create a new project in Code Composer Studio to use MSP432 in capture mode to determine the length of a pulse generated by the proximity sensor. The first step was to construct a circuit for the HC-SR04 sensor. This circuit included a level shifter which can be seen in **figure 1** below. Next, this circuit was tested by using a function generator and oscilloscope. The input pulse for the trig pin was generated using TimerA PWM and was tested using an oscilloscope. Then, TimerA capture was used to capture the period of the input pulse from the echo pin and the number of cycles obtained was used to calculate the distance. Next, the calculated distance was printed in the console using printf statement. The code for this part can be found in **Appendix A.**



**Figure 1:** Circuit diagram for part I

**PART II**

This part of the lab required students to control the intensity of an LED with respect to the distance measured in part I. This part of the lab was pretty straight forward after the completion of part I since it used the distance calculated in Part I to control the intensity of an LED. Another TimerA PWM was initialized and the CCR[X] resistor was updated to control the intensity of the LED. The circuit for this section was similar to PART I and can be seen in **Figure 2** below. The code for this part can be found in **Appendix B** below.



**Figure 2:** Circuit diagram for part II

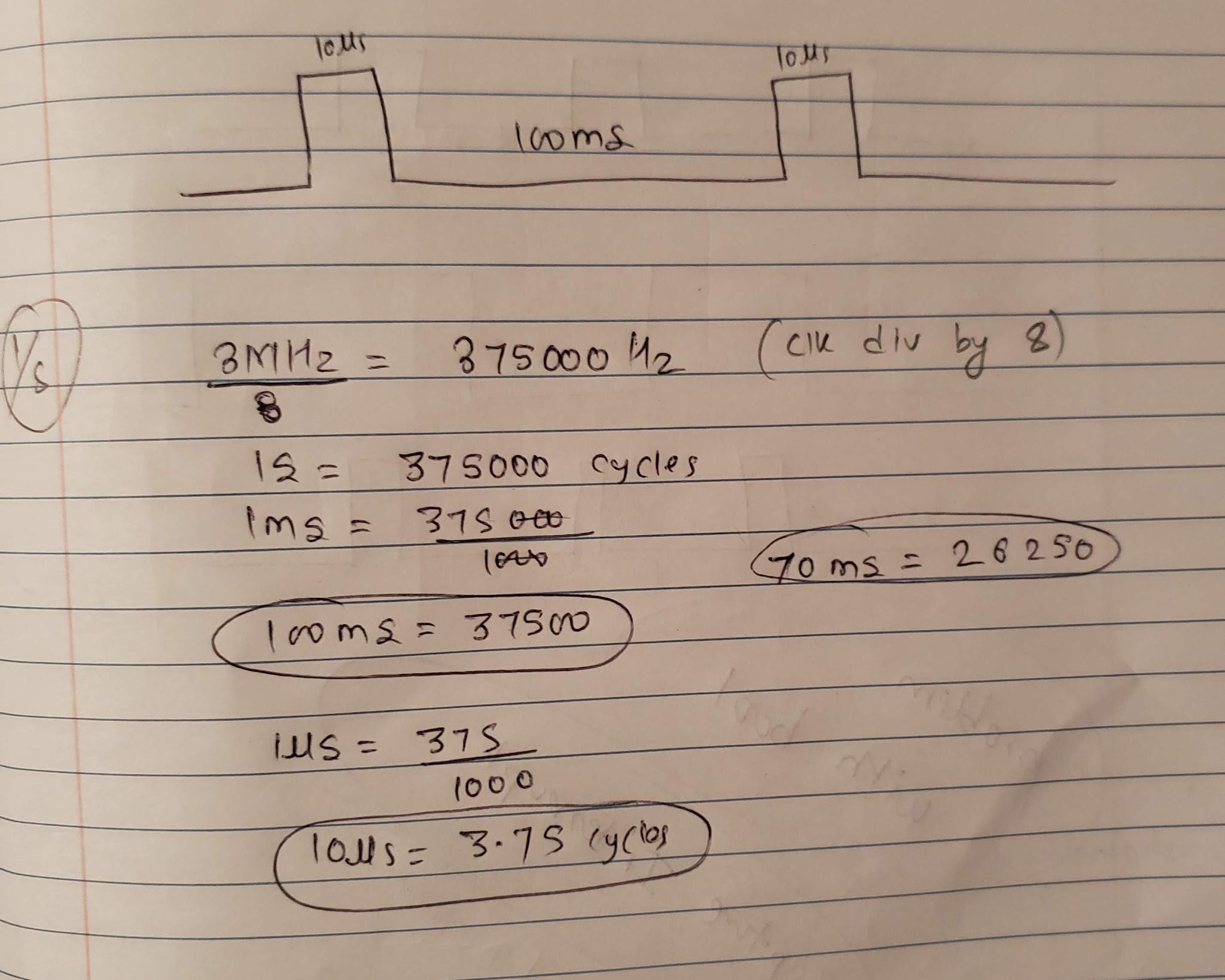
**CONCLUSION**

In conclusion, the lab overall went very smoothly. In part I, the distance measured by the HC-SR04 sensor was successfully printed in the console. Getting the timerA capture to function properly was a bit of a hassle but ultimately got it to work with the help of the instructor. Similarly, In part II the intensity of the LED was successfully controlled with respect to the distance measured by the sensor.

**LINK TO DEMO**

[**https://www.youtube.com/watch?v=DIYX9Ub9OTg**](https://www.youtube.com/watch?v=DIYX9Ub9OTg)

**Calculation for generating pulse for Trig pin**



**APPENDIX A //code for Part I & Part II**

/\*------------------------------------------------

\* Authors = Dixit Gurung

\* EGR 326 901

\* Date = 09/27/2020

\* Lab\_3

\* Description: The purpose of this laboratory was to create a program in CCS for

\* the MSP432 to interface with the ultrasonic sensor and led using TimerA capture and compare.

\*---------------------------------------------------\*/

//----------------------------------------------------- PART I AND PART II ------------------------------------------------------------

#include "msp.h"

#include <stdio.h>

volatile uint16\_t count = 0; //used in systick port handler

unsigned short lastedge = 0, currentedge = 0, period = 0; //variables to calculate cycles from the echo pins using capture

int edge\_flag = 1; //initially 1 becasue when the first interrupt on TA0 will be on rising edge

//After that it will change accordingly

float distance = 0; // distance variable to be determined

PWMperiod = 6000; //Period is 26250 because CCR[0] = 26250 for TimerA2 (70ms period CLK/8) PWM for led is also generated using TIMERA2

main(void)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;

TimerA\_trig\_Init();

TimerA\_Capture\_Init();

Led\_pwm();

NVIC->ISER[0] = 1 << ((TA0\_N\_IRQn) & 31); // Enable interrupt in NVIC vector

\_\_enable\_irq(); // Enable global interrupt

Led\_pwm();

while (1)

{

//since I used 2N4401 transistor 0.1\*period will be off time meaning 90% dutycycle

//In short it is opposite

if(distance <= 1){

TIMER\_A1->CCR[2] = 0.05\*PWMperiod; //Max intensity for < 1 in

}

else if(distance <= 2){

TIMER\_A1->CCR[2] = (0.05 \* PWMperiod); //2/10 max

}

else if(distance <= 3){

TIMER\_A1->CCR[2] = (0.95 \* PWMperiod);//3/10 max

}

else if(distance <= 4){

TIMER\_A1->CCR[2] = (0.8 \* PWMperiod);//4/10 max

}

else if(distance <= 5){

TIMER\_A1->CCR[2] = (0.6 \* PWMperiod); // 5/10 max

}

else if(distance <= 6){

TIMER\_A1->CCR[2] = (0.4 \* PWMperiod); // 6/10 max

}

else if(distance <= 7)

{

TIMER\_A1->CCR[2] = (0.3 \* PWMperiod); // 7/10 max

}

else if(distance <= 8){

TIMER\_A1->CCR[2] = (0.2 \* PWMperiod); // 8/10 max

}

else if(distance <= 9){

TIMER\_A1->CCR[2] = (0.1 \* PWMperiod); //9/10 max

}

else if(distance >= 9){

TIMER\_A1->CCR[2] = (PWMperiod); //greater than 9 inch will turn off the led

}

}

}

//This function generates pulse for trig pin of sensor and also generates pwm for led

void TimerA\_trig\_Init(void)

{

//P5.7 outputs pulse for trig pin

//setup P5.7 as an Timer A2.2 controlled OUTPUT, SEL = 01

P5->SEL0 |= BIT7;

P5->SEL1 &= ~ BIT7;

P5->DIR |= BIT7;

TIMER\_A2->CCR[0] = 26250; //70ms period CLK/8

TIMER\_A2->CCR[2] = 5; //10us ON time CLK/8

TIMER\_A2->CCTL[2] = TIMER\_A\_CCTLN\_OUTMOD\_3; // Reset/set

TIMER\_A2->CTL = TIMER\_A\_CTL\_TASSEL\_2 | // SMCLK

TIMER\_A\_CTL\_ID\_3| //divide CLK by 8

TIMER\_A\_CTL\_MC\_1 | // Up Mode ... Count up

TIMER\_A\_CTL\_CLR; // clear TA0R Register

}

void TimerA\_Capture\_Init(void)

{

P2->SEL0 |= BIT4; // TA0.1 input capture pin, second function

P2->SEL1 &= ~ BIT4; // TA0.1 input capture pin, second function

P2->DIR &= ~ BIT4;

TIMER\_A0->CTL |= TIMER\_A\_CTL\_TASSEL\_2 | // Use SMCLK as clock source,

TIMER\_A\_CTL\_MC\_2 | // Start timer in Continuous mode

TIMER\_A\_CTL\_CLR; // clear TA0R

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CM\_3 | // Capture rising and falling edge,

TIMER\_A\_CCTLN\_CCIS\_0 | // Use CCI2A

TIMER\_A\_CCTLN\_CCIE | // Enable capture interrupt

TIMER\_A\_CCTLN\_CAP | // Enable capture mode,

TIMER\_A\_CCTLN\_SCS; // Synchronous capture

}

void TA0\_N\_IRQHandler(void)

{

//reads the value at rising edge

if (edge\_flag)

{

lastedge = TIMER\_A0->CCR[1];

edge\_flag = 0;

}

//reads the value at falling edge and calculates no of cycles

else if (edge\_flag == 0)

{

currentedge = TIMER\_A0->CCR[1];

period = currentedge - lastedge;

edge\_flag = 1;

distance = period;

distance = distance / (148 \* 3);

printf("Distance = %f\n", distance); // prints distance into the monitor

//\_\_delay\_cycles(8000000);//2 sec delay for part II

}

TIMER\_A0->CCTL[1] &= ~(TIMER\_A\_CCTLN\_CCIFG);

}

//This function is setup for generating PWM for led

void Led\_pwm(void)

{

P7->SEL0 |= BIT6; //sets up TimerA2.1 on P5.6

P7->SEL1 &= ~ BIT6;

P7->DIR |= BIT6; //outputs PWM signal

TIMER\_A1->CCR[0] = 6000; //70ms period CLK/8

TIMER\_A1->CCR[2] = .99\*6000; //10us ON time CLK/8

TIMER\_A1->CCTL[2] = TIMER\_A\_CCTLN\_OUTMOD\_3; // Reset/set

TIMER\_A1->CTL = TIMER\_A\_CTL\_TASSEL\_2 | // SMCLK

TIMER\_A\_CTL\_ID\_3 | //divide CLK by 8

TIMER\_A\_CTL\_MC\_1 | // Up Mode ... Count up

TIMER\_A\_CTL\_CLR; // clear TA0R Register

}