CPE301 – SPRING 2019

Design Assignment 2C

Student Name: Francisco Mata Carlos
Student #: 1012593607
Student Email: matacarl@unlv.nevada.edu
Primary Github address: https://github.com/chicosisco/da_sub.git
Directory: repository/cpe301/DesignAssignments/DA2C

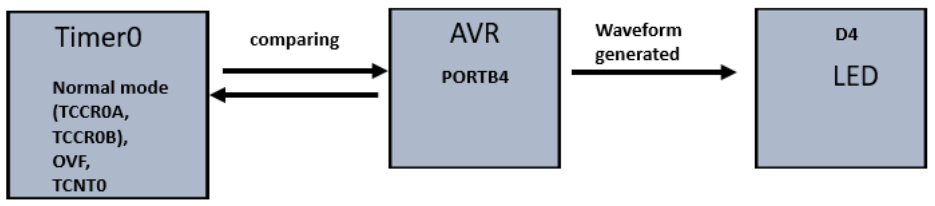
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The components used for this assignment are the next:

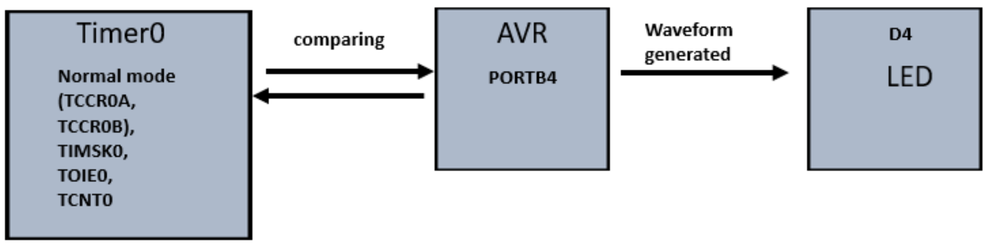
a. Atmega328p Xplained Mini
b. Multi-functional Shield
c. Oscilloscope and compensated probe
d. Atmel Studio 7

Block diagram with pins used in the Atmega328P
Block for generating a waveform with 60% duty cycle and a period of 0.725 seconds.

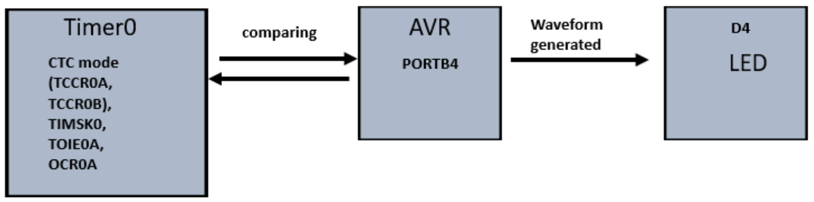
1_A



2_A

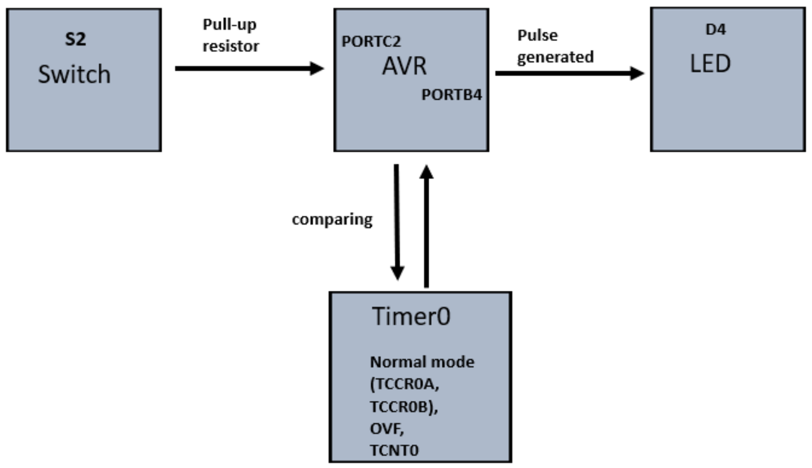


3_A

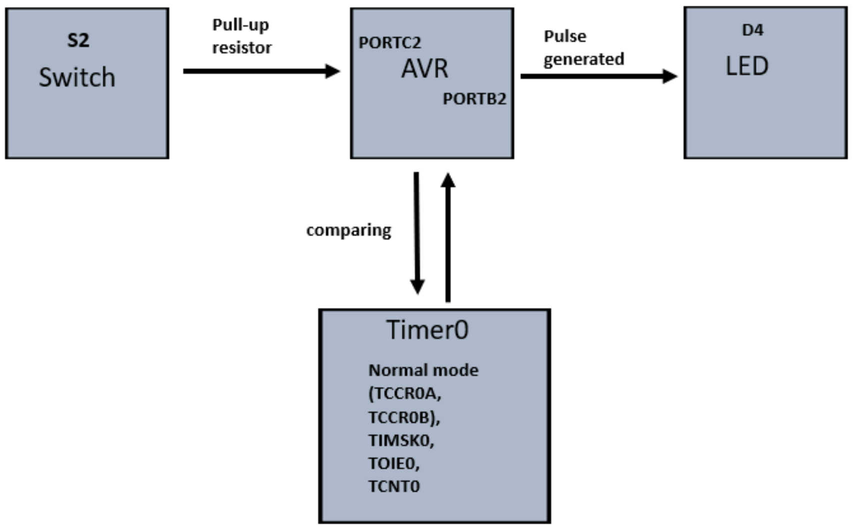


Block diagram for part 2 of assignment. Generating a pulse when a pushbutton is pressed and is demonstrated by an LED.

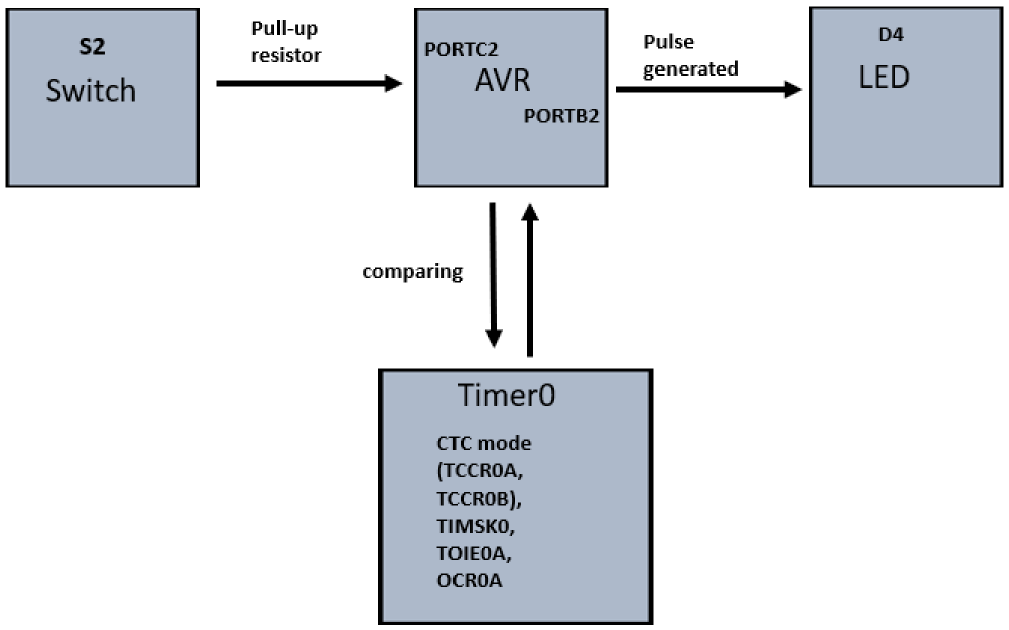
1_B



2_B



3_B



2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1
Implement Design Assignment 2A using Timer 0 – normal mode. Count OVF occurrence if needed. Do not use interrupts.

Task 1_A
1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period using Timer 0 – normal mode. Count OVF occurrence if needed. Do not use interrupts.

```
 * DA2C.c
 *
 * Created: 3/17/2019 3:25:11 AM
 * Author : Francisco Mata carlos
 */

#define F_CPU 16000000UL    /* clock runs at 16 MHz*/
#include <avr/io.h>
#include<util/delay.h>


int main()

{
       int over_flow=0; //over-flow counter
       DDRB |= (1<<DDB4); // PB4 as output
       TCCR0A = 0; // normal operation.
       TCNT0 = 0x00;   //start timer/counter

       TCCR0B |=(1<<CS02);  // setting prescaler to 256

    while (1)
    {

              //wait for the overflow event
              while ((TIFR0 & 0X01)==0);
              TCNT0=0X00;  //resetting counter to zero
              TIFR0=0X01;  // reset the overflow flag
              over_flow++;  //increasing overflow counter

              // if overflow is equal to 71 cycles turn on LED on PB4
              if (over_flow>=71){
              PORTB = (0<<DDB4);
              }
              else
              PORTB = (1<<DDB4);  // or turn off LED on PB4

              if (over_flow==177) {
                     over_flow=0; //resetting overflow counter
                     }
              }
    }
```

Task 1_B

1.b. Connect a switch to PORTC.2 (active high - turn on the pull up transistor) to poll for an event to turn on the led at PORTB.2 for 1.250 sec after the event, using Timer 0 – normal mode. Count OVF occurrence if needed. Do not use interrupts.

```
* DA2C_1b.c
 *
 * Created: 3/19/2019 5:00:50 PM
 * Author : Francisco Mata carlos
 */


#define F_CPU 16000000UL /* clock runs at 16 MHz*/

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main(void)
{

       DDRC &= (0<<2);
       PORTC |=(1<<2); //enable pull-up

    DDRB |= (1<<DDB4); // PB4 as output
```

```
        PORTB |=(1<<DDB4); // Set PB2 high to keep LED off
                           // as the starting position
        TCCR0A = 0; // normal operation.
        TCNT0 = 0;   //start timer/counter

        TCCR0B |=(1<<CS02)|(1<<CS00);  // setting prescaler to 1024

        while (1)
    {


               if (!(PINC & (1<<PINC1))) //checking if pinc is high and complement
               {
                       while (1)
                      {
                             //wait for the overflow event
                             while ((TIFR0 & 0X01)==0);
                             TCNT0=0X00;  //resetting counter to zero
                             TIFR0=0X01;  // reset the overflow flag

                             over_flow++;  //increasing overflow counter

                             // if overflow is less than or equal to 1 cycle the LED on PB4
                             // turns off and stays off once it breaks
                             if (over_flow<=1){
                             PORTB = (1<<DDB4);
                             break;
                             }

                             else{
                             PORTB = (0<<DDB4);  // or turn off LED on PB4
                             //break;
                             }
                             if (over_flow==78) {
                                    over_flow=0; //resetting overflow counter
                             }
                      }
               }

     //return 0;
     }
}
```

INITIAL/MODIFIED/DEVELOPED CODE OF TASK 2
Implement Design Assignment 2A using TIMER0_OVF_vect interrupt mechanism in normal mode.

Task 2_A
1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period using TIMER0_OVF_vect interrupt mechanism in normal mode.

```
/*
 * DA2C_2a
 *
 * Created: 3/17/2019 11:55:51 PM
 * Author : Francisco Mata Carlos
 */

#define F_CPU 16000000   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{

    DDRB |=(1<<DDB4);  // setting PB1 as output
       TIMSK0 |= (1<<TOIE0);
       TCNT0 = 0; // setting initial value for counter
       sei();     // enable interrupts
       TCCR0B |=(1<<CS02);  // setting prescaler to 256


    while (1)
    {

              //main loop
    }
}

       ISR(TIMER0_OVF_vect) // timer_0 overflow interrupt
       {
              while (!(TIFR0 & 0X01)==0);
              TCNT0=0X00;  //resetting counter to zero
              TIFR0=0X01;  // reset the overflow flag
              over_flow++;  //increasing overflow counter

              // if overflow is equal to 71 cycles turn on LED on PB4
              if (over_flow>=71){
              PORTB = (0<<DDB4);
              }
              else
              PORTB = (1<<DDB4);  // or turn off LED on PB4

              if (over_flow==178) {
                     over_flow=0; //resetting overflow counter
                     }
              }
```

Task 2_B

Connect a switch to PORTC.2 (active high - turn on the pull up transistor) to poll for an event to turn on the led at PORTB.2 for 1.250 sec after the event, using TIMER0_OVF_vect interrupt mechanism in normal mode.

```
/*
 * DA2C_2b.c
 *
 * Created: 3/20/2019 1:57:38 PM
 * Author : Francisco Mata Carlos
 */


#define F_CPU 16000000   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
       DDRC &= (0<<2);
       PORTC |=(1<<2); //enable pull-up

       DDRB |=(1<<DDB4);     // setting PB4 as output
       PORTB |=(1<<DDB4);    // set PB2 high to keep D2 LED off, as starting state
       TIMSK0 |= (1<<TOIE0);
       TCNT0 = 0;            // setting initial value for counter
       sei();                // enable interrupts
       TCCR0B |=(1<<CS02)|(1<<CS00);   // setting prescaler to 1024


       while (1)
       {

       }
}

ISR(TIMER0_OVF_vect)     // timer_0 overflow interrupt
{
       while (1)
     {

              if (!(PINC & (1<<PINC1))) //checking if pinc is high and complement by
                                                     //pressing on the switch
               {

                     while (1)
                    {
                           //wait for the overflow event
                           while ((TIFR0 & 0X01)==0);
                           TCNT0=0X00;     //resetting counter to zero
                           TIFR0=0X01;     // reset the overflow flag

                           over_flow++;    //increasing overflow counter
```

```
                        // if overflow is less than or equal to 1 cycle the LED on PB4
                        // turns off and stays off once it breaks
                        if (over_flow<=1)
                        {
                        PORTB = (1<<DDB4);
                        break;
                      }

                        else   {
                        PORTB = (0<<DDB4);       //  turn on LED on PB4 until
                                                // over_flow count
                                               //  resets again

                       }
                        if (over_flow==77) {
                               over_flow=0;         //resetting overflow counter

                                           }

                   }

            }

     //return 0;
     }

 }
```

INITIAL/MODIFIED/DEVELOPED CODE OF TASK 3
Implement Design Assignment 2A using TIMER0_COMPA_vect interrupt mechanism in CTC mode.

Task 3_A
1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period using TIMER0_COMPA_vect interrupt mechanism in CTC mode.

```
 * DA2C_3a
 *
 * Created: 3/17/2019 11:55:51 PM
 * Author : Francisco Mata Carlos
 */


#define F_CPU 16000000UL   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
       DDRB |=0x10;   //make PB4 an output
       OCR0A = 128;   // compare register value
```

```
        TCCR0B |=(1<<CS02);    // prescaler = 256
        TCCR0A |=(1<<WGM01);   // CTC mode
        TIMSK0 = (1<<OCIE0A);  // enable Timer 0 compare match interrupt
        sei();      // enable global interrupt

        while (1)
        {

        }
}

// every time there's a match with the comparator register
// it jumps into this comparator interrupt
ISR (TIMER0_COMPA_vect)
{

              //wait for the overflow event
              while ((TIFR0 & 0X02)==0);
              TCNT0=0X0;  //resetting counter to zero
              TIFR0=0X02;  // reset the overflow flag
              over_flow++;  //increasing overflow counter

              // if overflow is equal to 71 cycles turn on LED on PB4
              if (over_flow>=71){
              PORTB = (0<<DDB4);
              }
              else
              PORTB = (1<<DDB4);  // or turn off LED on PB4

              if (over_flow==176) {
                     over_flow=0; //resetting overflow counter
                     }
              }
```

Task 3_B

1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 sec period using TIMER0_COMPA_vect interrupt mechanism in CTC mode.

```
 * DA2C_3b.c
 *
 * Created: 3/20/2019 6:05:48 PM
 * Author : Baker
 */


#define F_CPU 16000000UL   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
       DDRC &= (0<<2);
       PORTC |=(1<<2);  //enable pull-up
```

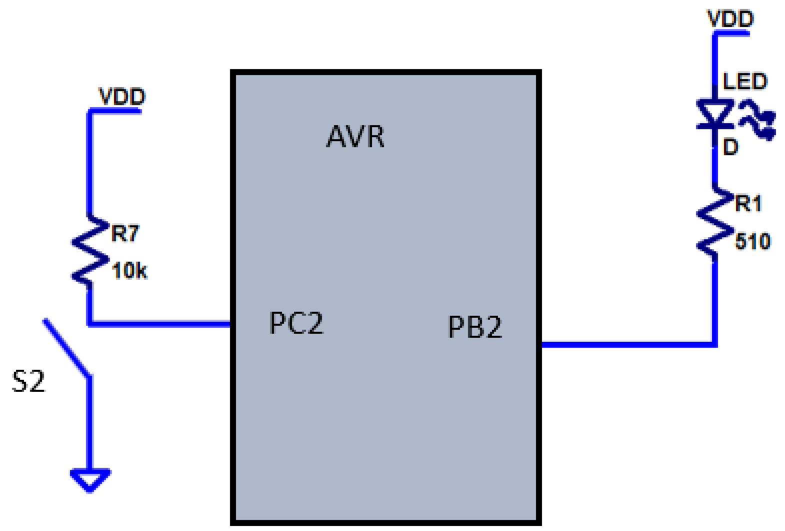
```
        DDRB |=0x10;   //make PB4 an output
        PORTB |= (1<<DDB4);
        OCR0A = 0;   // compare register value
        TCCR0B |=(1<<CS02)|(1<<CS00);    // prescaler = 256
        TCCR0A |=(1<<WGM01);   // CTC mode
        TIMSK0 = (1<<OCIE0A);  // enable Timer 0 compare match interrupt
        sei();      // enable global interrupt

        while (1)
        {

        }
}

// every time there's a match with the comparator register
// it jumps into this comparator interrupt
ISR (TIMER0_COMPA_vect)
{
        while (1)
      {

              if (!(PINC & (1<<PINC1))) //checking if pinc is high and complement by
                                                 //pressing on the switch
               {

                      while (1)
                     {
                            //wait for the overflow event
                            while ((TIFR0 & 0X02)==0);
                            TCNT0=0X00;     //resetting counter to zero
                            TIFR0=0X02;     // reset the overflow flag

                            over_flow++;    //increasing overflow counter

                            // if overflow is less than or equal to 1 cycle the LED on PB4
                            // turns off and stays off once it breaks
                            if (over_flow<=1)
                            {
                            PORTB = (1<<DDB4);
                            break;
                         }

                            else   {
                            PORTB = (0<<DDB4);       //  turn on LED on PB4 until
over_flow count
                                                                //  resets again

                          }
                            if (over_flow==77) {
                                   over_flow=0;        //resetting overflow counter

                                          }

                     }

              }
```

```
    //return 0;
    }

}
```

3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

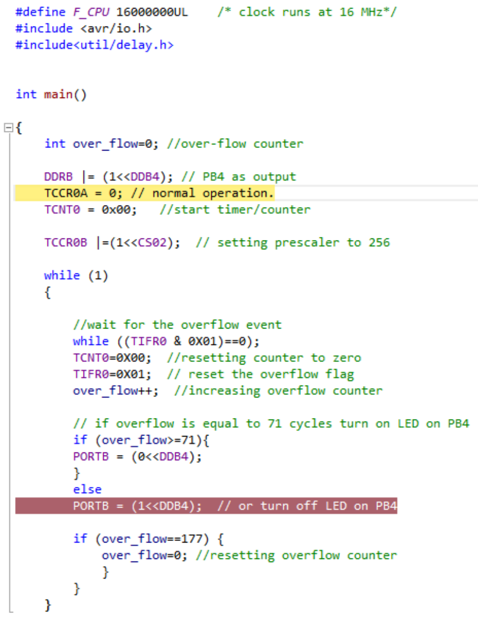
Same as above

4. SCHEMATICS

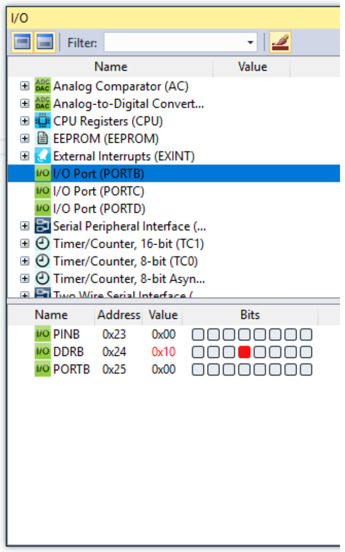


5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1_A



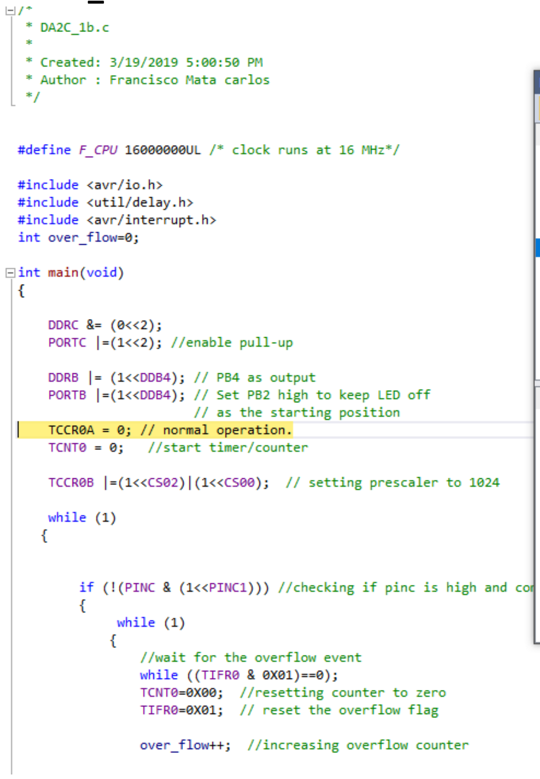
```
#define F_CPU 16000000UL    /* clock runs at 16 MHz*/
#include <avr/io.h>
#include<util/delay.h>


int main()

{
    int over_flow=0; //over-flow counter

    DDRB |= (1<<DDB4); // PB4 as output
    TCCR0A = 0; // normal operation.
    TCNT0 = 0x00;   //start timer/counter

    TCCR0B |=(1<<CS02);  // setting prescaler to 256

    while (1)
    {

        //wait for the overflow event
        while ((TIFR0 & 0X01)==0);
        TCNT0=0X00;  //resetting counter to zero
        TIFR0=0X01;  // reset the overflow flag
        over_flow++;  //increasing overflow counter

        // if overflow is equal to 71 cycles turn on LED on PB4
        if (over_flow>=71){
        PORTB = (0<<DDB4);
        }
        else
        PORTB = (1<<DDB4);  // or turn off LED on PB4

        if (over_flow==177) {
            over_flow=0; //resetting overflow counter
            }
        }
    }
```



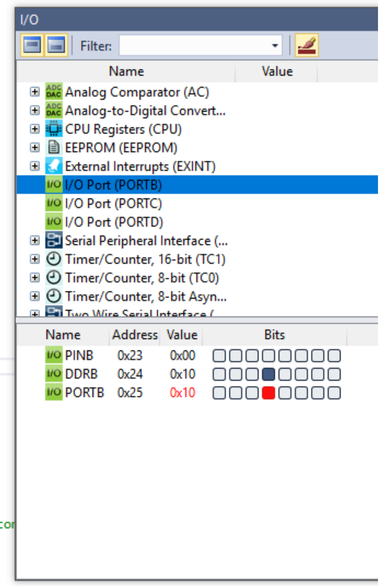
```
I/O
Filter:
Name                              Value
⊞ Analog Comparator (AC)
⊞ Analog-to-Digital Convert...
⊞ CPU Registers (CPU)
⊞ EEPROM (EEPROM)
⊞ External Interrupts (EXINT)
  I/O Port (PORTB)
  I/O Port (PORTC)
  I/O Port (PORTD)
⊞ Serial Peripheral Interface (...
⊞ Timer/Counter, 16-bit (TC1)
⊞ Timer/Counter, 8-bit (TC0)
⊞ Timer/Counter, 8-bit Asyn...
⊞ Two Wire Serial Interface (
```

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐☐☐☐☐☐
DDRB	0x24	0x10	☐☐☐🟥☐☐☐☐
PORTB	0x25	0x00	☐☐☐☐☐☐☐☐

Task 1_B



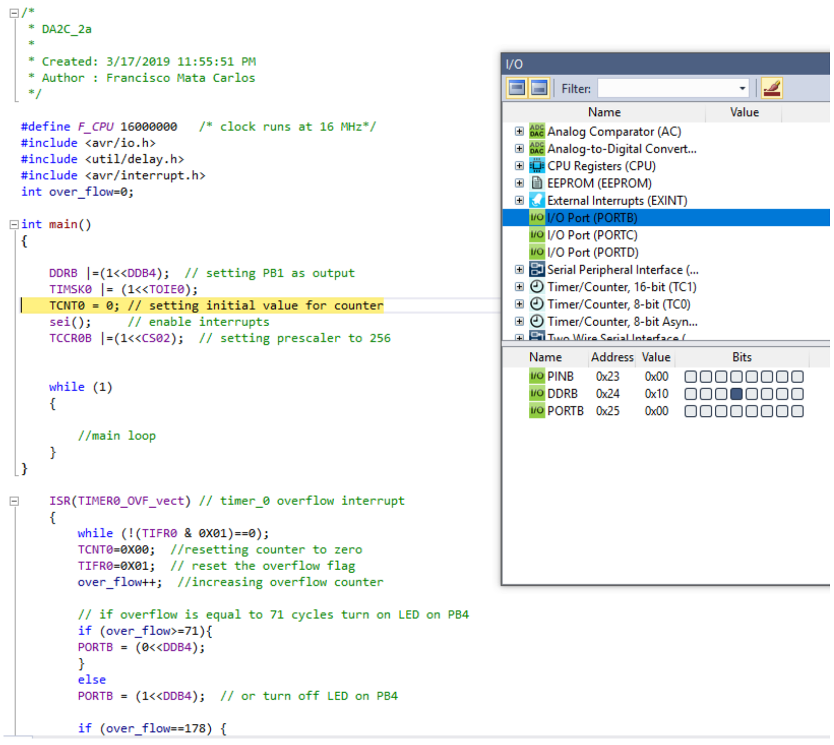
```
/*
 * DA2C_1b.c
 *
 * Created: 3/19/2019 5:00:50 PM
 * Author : Francisco Mata carlos
 */


#define F_CPU 16000000UL /* clock runs at 16 MHz*/

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main(void)
{

    DDRC &= (0<<2);
    PORTC |=(1<<2); //enable pull-up

    DDRB |= (1<<DDB4); // PB4 as output
    PORTB |=(1<<DDB4); // Set PB2 high to keep LED off
                       // as the starting position
    TCCR0A = 0; // normal operation.
    TCNT0 = 0;   //start timer/counter

    TCCR0B |=(1<<CS02)|(1<<CS00);  // setting prescaler to 1024

    while (1)
   {


        if (!(PINC & (1<<PINC1))) //checking if pinc is high and co
        {
            while (1)
           {
              //wait for the overflow event
              while ((TIFR0 & 0X01)==0);
              TCNT0=0X00;  //resetting counter to zero
              TIFR0=0X01;  // reset the overflow flag

              over_flow++;  //increasing overflow counter
```



```
I/O
Filter:
Name                              Value
⊞ Analog Comparator (AC)
⊞ Analog-to-Digital Convert...
⊞ CPU Registers (CPU)
⊞ EEPROM (EEPROM)
⊞ External Interrupts (EXINT)
  I/O Port (PORTB)
  I/O Port (PORTC)
  I/O Port (PORTD)
⊞ Serial Peripheral Interface (...
⊞ Timer/Counter, 16-bit (TC1)
⊞ Timer/Counter, 8-bit (TC0)
⊞ Timer/Counter, 8-bit Asyn...
⊞ Two Wire Serial Interface (
```

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐☐☐☐☐☐
DDRB	0x24	0x10	☐☐☐■☐☐☐☐
PORTB	0x25	0x10	☐☐☐🟥☐☐☐☐

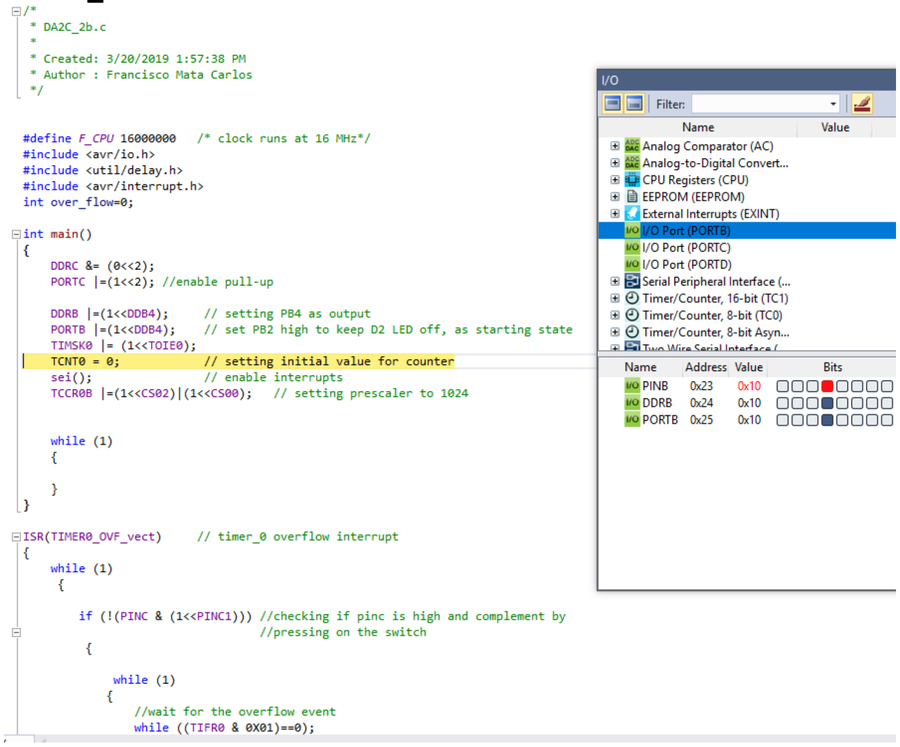
Task 2_A



```c
/*
 * DA2C_2a
 *
 * Created: 3/17/2019 11:55:51 PM
 * Author : Francisco Mata Carlos
 */

#define F_CPU 16000000   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{

    DDRB |=(1<<DDB4);  // setting PB1 as output
    TIMSK0 |= (1<<TOIE0);
    TCNT0 = 0; // setting initial value for counter
    sei();     // enable interrupts
    TCCR0B |=(1<<CS02);  // setting prescaler to 256


    while (1)
    {

        //main loop
    }
}

    ISR(TIMER0_OVF_vect) // timer_0 overflow interrupt
    {
        while (!(TIFR0 & 0X01)==0);
        TCNT0=0X00;  //resetting counter to zero
        TIFR0=0X01;  // reset the overflow flag
        over_flow++;  //increasing overflow counter

        // if overflow is equal to 71 cycles turn on LED on PB4
        if (over_flow>=71){
        PORTB = (0<<DDB4);
        }
        else
        PORTB = (1<<DDB4);  // or turn off LED on PB4

        if (over_flow==178) {
```

```
I/O
Filter:
Name                                Value
⊞ Analog Comparator (AC)
⊞ Analog-to-Digital Convert...
⊞ CPU Registers (CPU)
⊞ EEPROM (EEPROM)
⊞ External Interrupts (EXINT)
  I/O Port (PORTB)
  I/O Port (PORTC)
  I/O Port (PORTD)
⊞ Serial Peripheral Interface (...
⊞ Timer/Counter, 16-bit (TC1)
⊞ Timer/Counter, 8-bit (TC0)
⊞ Timer/Counter, 8-bit Asyn...
⊞ Two Wire Serial Interface (

Name    Address  Value   Bits
PINB    0x23     0x00    ☐☐☐☐☐☐☐☐
DDRB    0x24     0x10    ☐☐☐■☐☐☐☐
PORTB   0x25     0x00    ☐☐☐☐☐☐☐☐
```

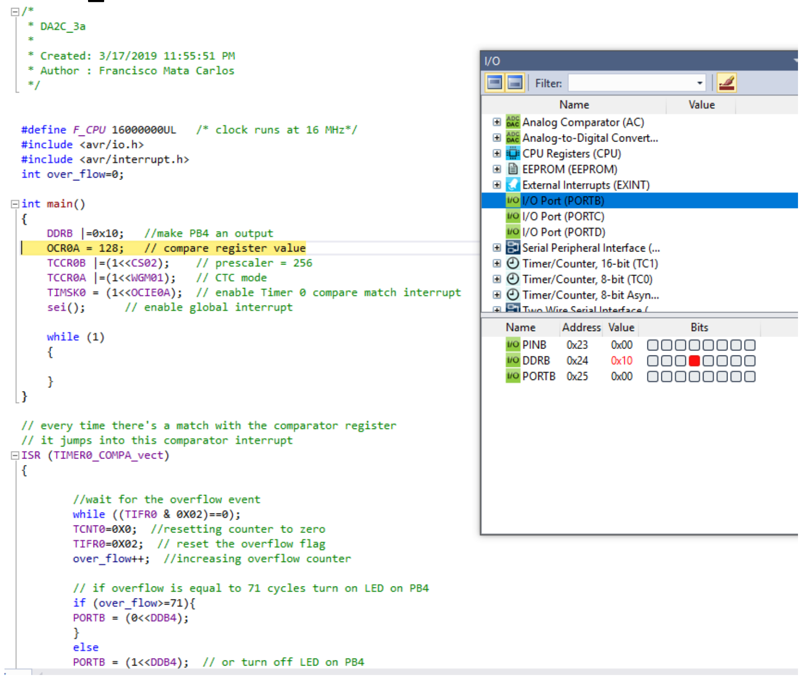
Task 2_B



```c
/*
 * DA2C_2b.c
 *
 * Created: 3/20/2019 1:57:38 PM
 * Author : Francisco Mata Carlos
 */


#define F_CPU 16000000   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
    DDRC &= (0<<2);
    PORTC |=(1<<2); //enable pull-up

    DDRB |=(1<<DDB4);      // setting PB4 as output
    PORTB |=(1<<DDB4);     // set PB2 high to keep D2 LED off, as starting state
    TIMSK0 |= (1<<TOIE0);
    TCNT0 = 0;             // setting initial value for counter
    sei();                 // enable interrupts
    TCCR0B |=(1<<CS02)|(1<<CS00);    // setting prescaler to 1024


    while (1)
    {

    }
}

ISR(TIMER0_OVF_vect)     // timer_0 overflow interrupt
{
    while (1)
     {

        if (!(PINC & (1<<PINC1))) //checking if pinc is high and complement by
                                  //pressing on the switch
         {

            while (1)
           {
              //wait for the overflow event
              while ((TIFR0 & 0X01)==0);
```

```
I/O
Filter:
Name                                Value
⊞ Analog Comparator (AC)
⊞ Analog-to-Digital Convert...
⊞ CPU Registers (CPU)
⊞ EEPROM (EEPROM)
⊞ External Interrupts (EXINT)
  I/O Port (PORTB)
  I/O Port (PORTC)
  I/O Port (PORTD)
⊞ Serial Peripheral Interface (...
⊞ Timer/Counter, 16-bit (TC1)
⊞ Timer/Counter, 8-bit (TC0)
⊞ Timer/Counter, 8-bit Asyn...
⊞ Two Wire Serial Interface (

Name    Address  Value   Bits
PINB    0x23     0x10    ☐☐☐🟥☐☐☐☐
DDRB    0x24     0x10    ☐☐☐■☐☐☐☐
PORTB   0x25     0x10    ☐☐☐■☐☐☐☐
```

Task 3_A

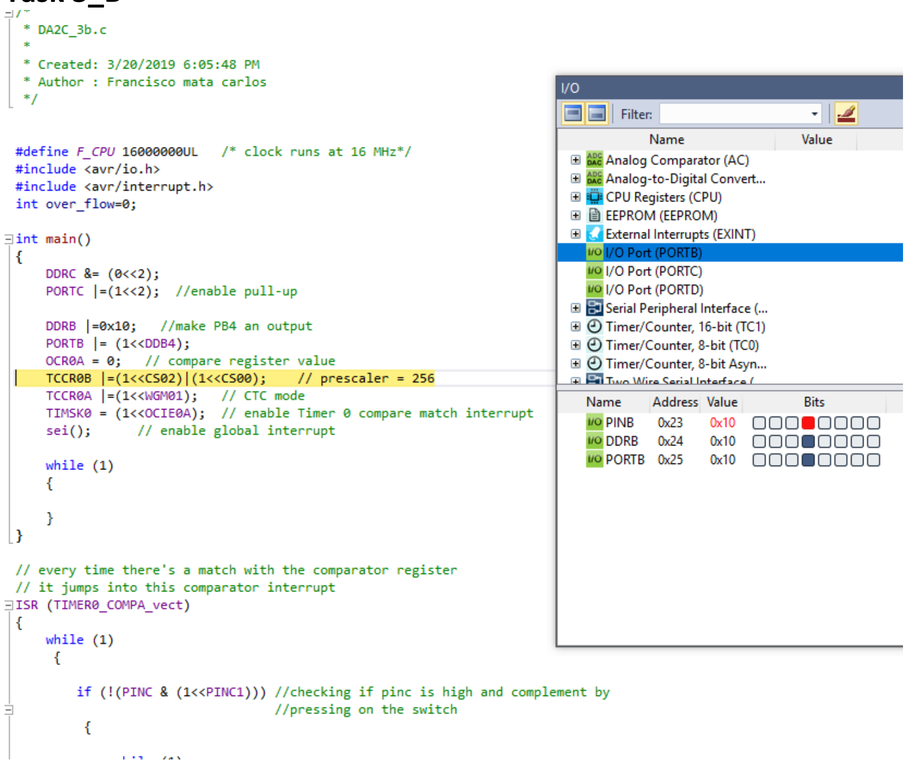


```
/*
 * DA2C_3a
 *
 * Created: 3/17/2019 11:55:51 PM
 * Author : Francisco Mata Carlos
 */


#define F_CPU 16000000UL   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
    DDRB |=0x10;   //make PB4 an output
    OCR0A = 128;   // compare register value
    TCCR0B |=(1<<CS02);    // prescaler = 256
    TCCR0A |=(1<<WGM01);   // CTC mode
    TIMSK0 = (1<<OCIE0A);  // enable Timer 0 compare match interrupt
    sei();      // enable global interrupt

    while (1)
    {

    }
}

// every time there's a match with the comparator register
// it jumps into this comparator interrupt
ISR (TIMER0_COMPA_vect)
{

        //wait for the overflow event
        while ((TIFR0 & 0X02)==0);
        TCNT0=0X0;  //resetting counter to zero
        TIFR0=0X02;  // reset the overflow flag
        over_flow++;  //increasing overflow counter

        // if overflow is equal to 71 cycles turn on LED on PB4
        if (over_flow>=71){
        PORTB = (0<<DDB4);
        }
        else
        PORTB = (1<<DDB4);  // or turn off LED on PB4
```

I/O
Filter:
Name
Value
Analog Comparator (AC)
Analog-to-Digital Convert...
CPU Registers (CPU)
EEPROM (EEPROM)
External Interrupts (EXINT)
I/O Port (PORTB)
I/O Port (PORTC)
I/O Port (PORTD)
Serial Peripheral Interface (...
Timer/Counter, 16-bit (TC1)
Timer/Counter, 8-bit (TC0)
Timer/Counter, 8-bit Asyn...

```
Name    Address  Value  Bits
PINB    0x23     0x00   ☐☐☐☐☐☐☐☐
DDRB    0x24     0x10   ☐☐☐🟥☐☐☐☐
PORTB   0x25     0x00   ☐☐☐☐☐☐☐☐
```

Task 3_B



```
/*
 * DA2C_3b.c
 *
 * Created: 3/20/2019 6:05:48 PM
 * Author : Francisco mata carlos
 */


#define F_CPU 16000000UL   /* clock runs at 16 MHz*/
#include <avr/io.h>
#include <avr/interrupt.h>
int over_flow=0;

int main()
{
    DDRC &= (0<<2);
    PORTC |=(1<<2);  //enable pull-up

    DDRB |=0x10;   //make PB4 an output
    PORTB |= (1<<DDB4);
    OCR0A = 0;   // compare register value
    TCCR0B |=(1<<CS02)|(1<<CS00);    // prescaler = 256
    TCCR0A |=(1<<WGM01);   // CTC mode
    TIMSK0 = (1<<OCIE0A);  // enable Timer 0 compare match interrupt
    sei();      // enable global interrupt

    while (1)
    {

    }
}

// every time there's a match with the comparator register
// it jumps into this comparator interrupt
ISR (TIMER0_COMPA_vect)
{
    while (1)
     {

        if (!(PINC & (1<<PINC1))) //checking if pinc is high and complement by
                                  //pressing on the switch
         {
```

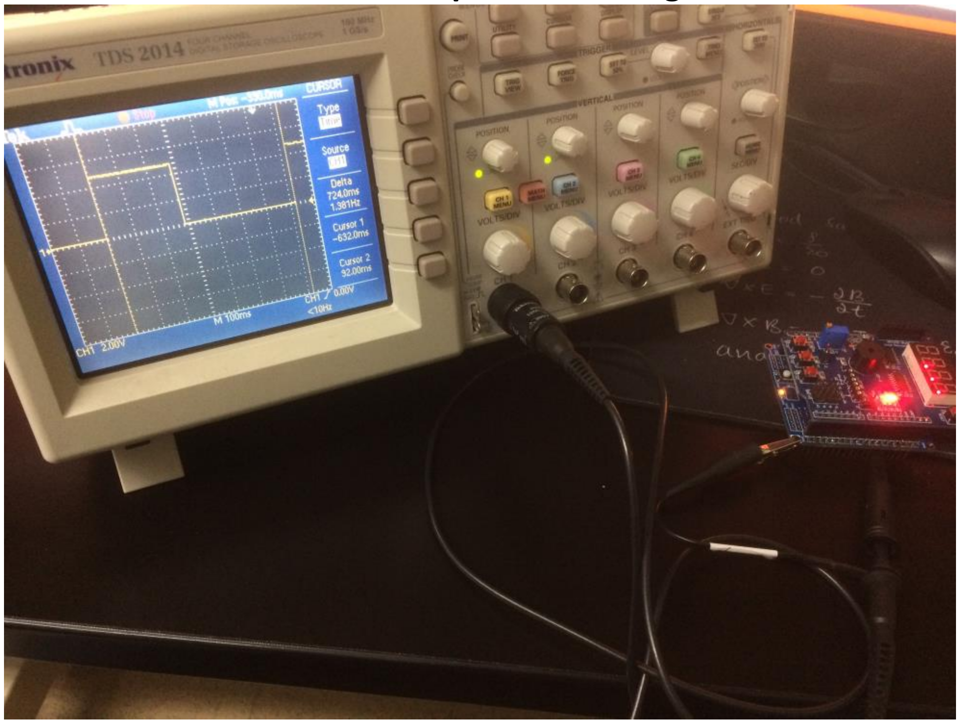
I/O
Filter:
Name
Value
Analog Comparator (AC)
Analog-to-Digital Convert...
CPU Registers (CPU)
EEPROM (EEPROM)
External Interrupts (EXINT)
I/O Port (PORTB)
I/O Port (PORTC)
I/O Port (PORTD)
Serial Peripheral Interface (...
Timer/Counter, 16-bit (TC1)
Timer/Counter, 8-bit (TC0)
Timer/Counter, 8-bit Asyn...

```
Name    Address  Value  Bits
PINB    0x23     0x10   ☐☐☐🟥☐☐☐☐
DDRB    0x24     0x10   ☐☐☐■☐☐☐☐
PORTB   0x25     0x10   ☐☐☐■☐☐☐☐
```

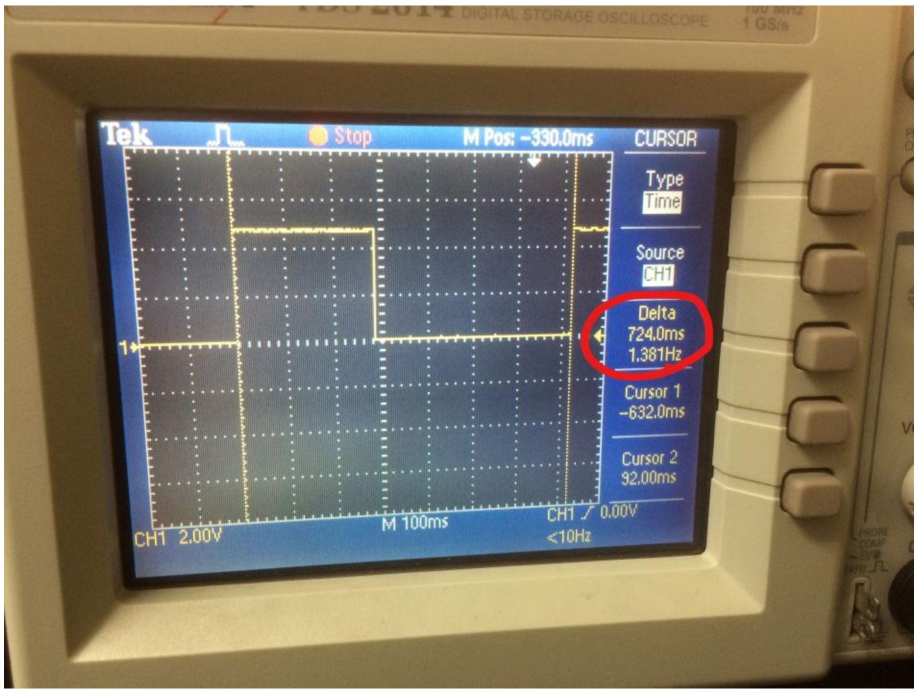
6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Task 1_A

Photo below shows the set up when testing for the waveform

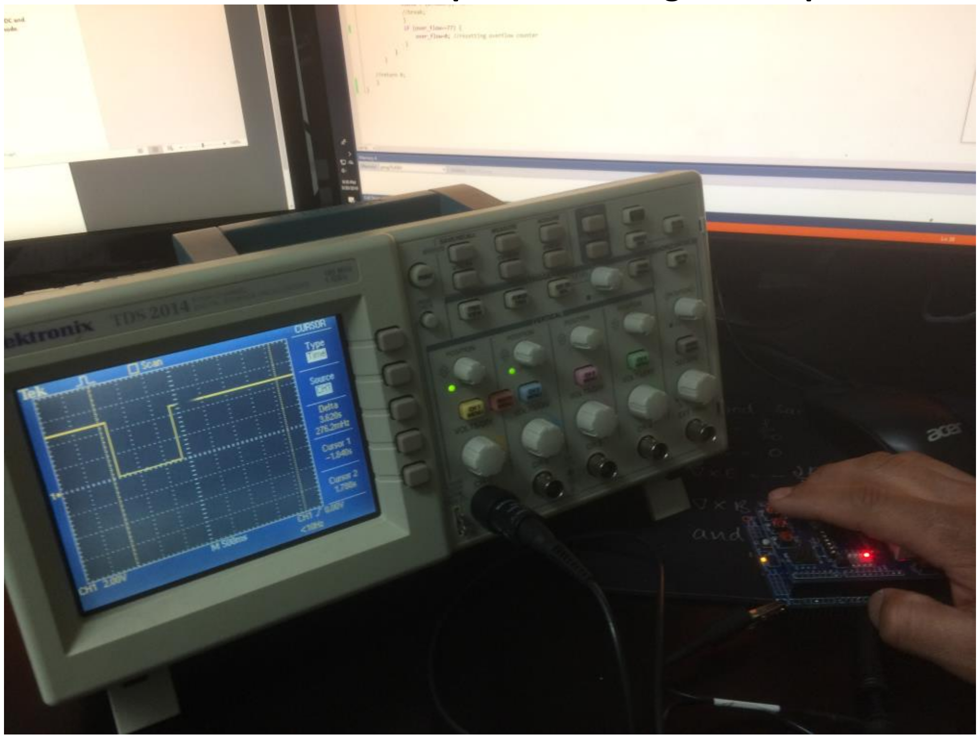


The photo below shows the period of the waveform

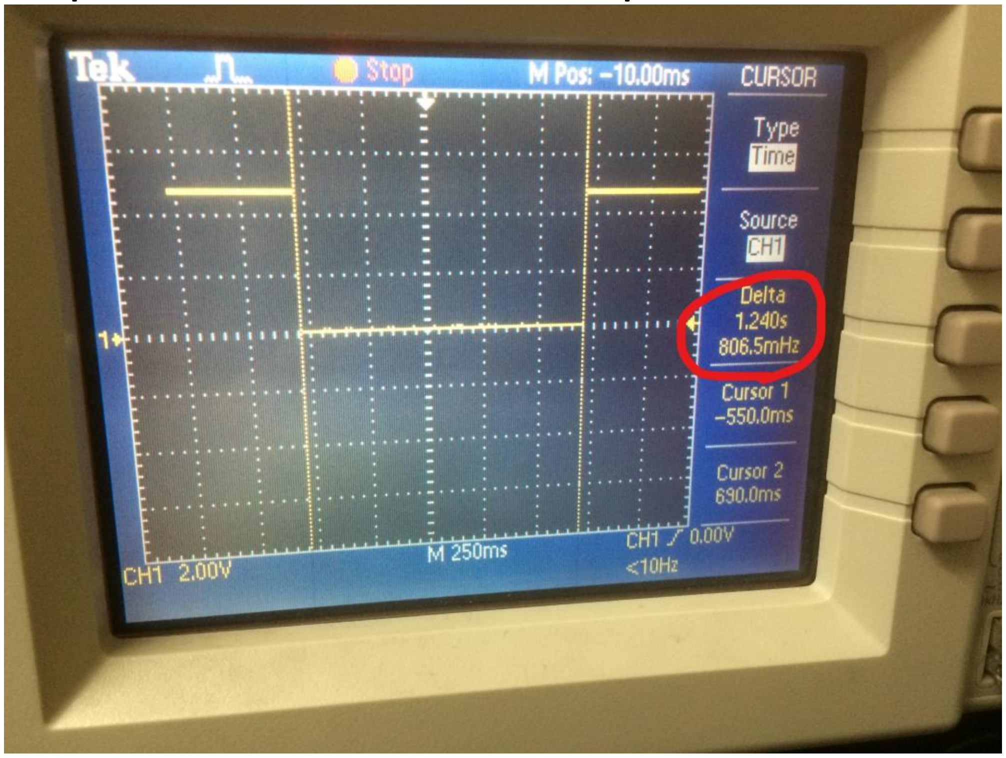


Task 1_B

Photo below shows the set up when testing for the pulse event

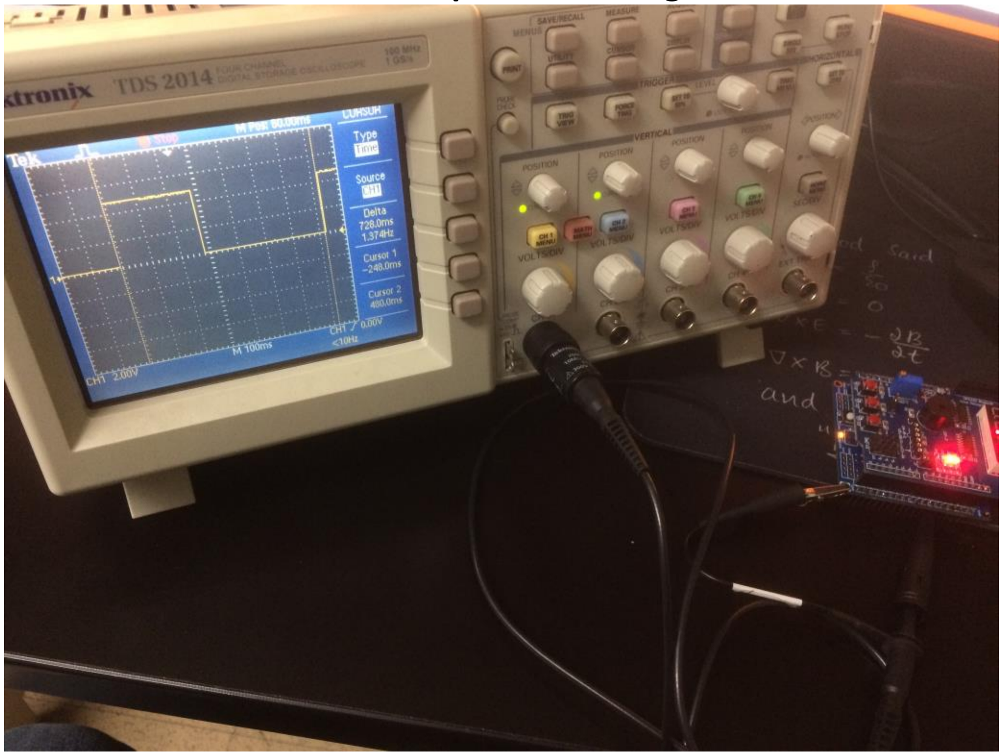


The photo below shows the time of the pulse event

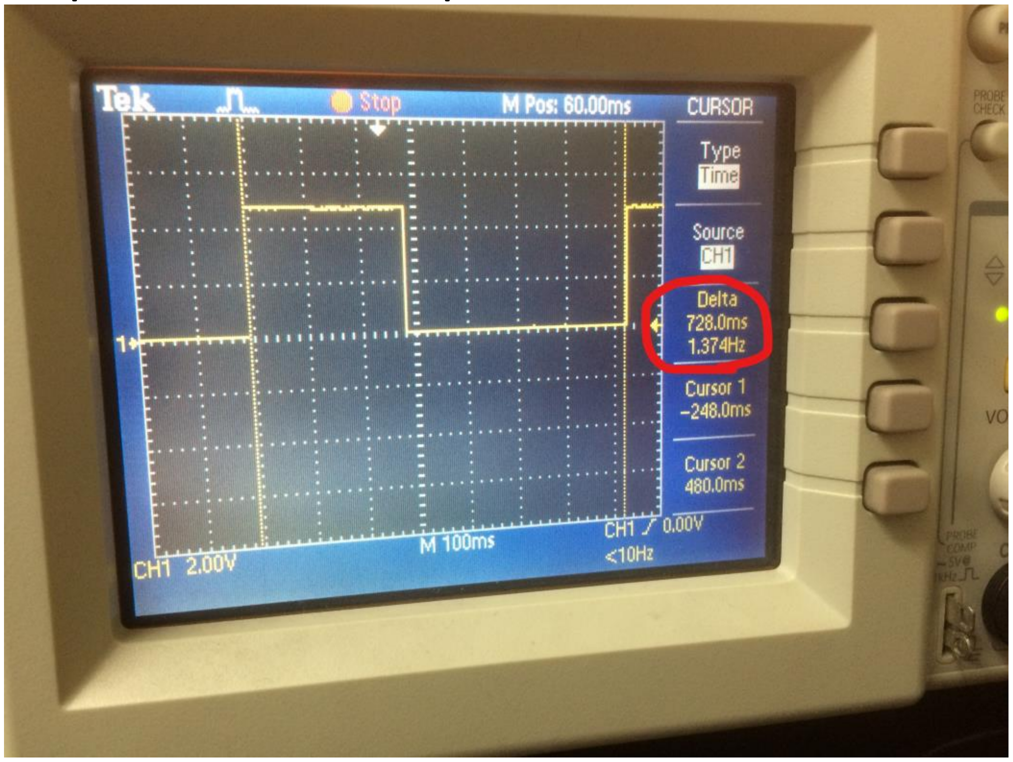


Task 2_A

Photo below shows the set up when testing for the waveform

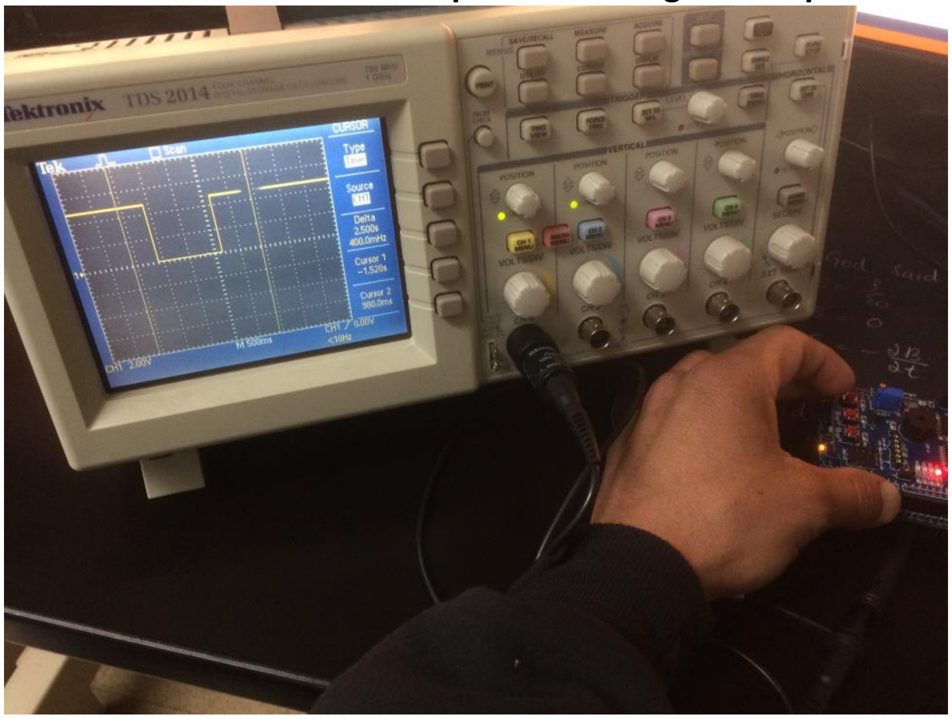


The photo below shows the period of the waveform

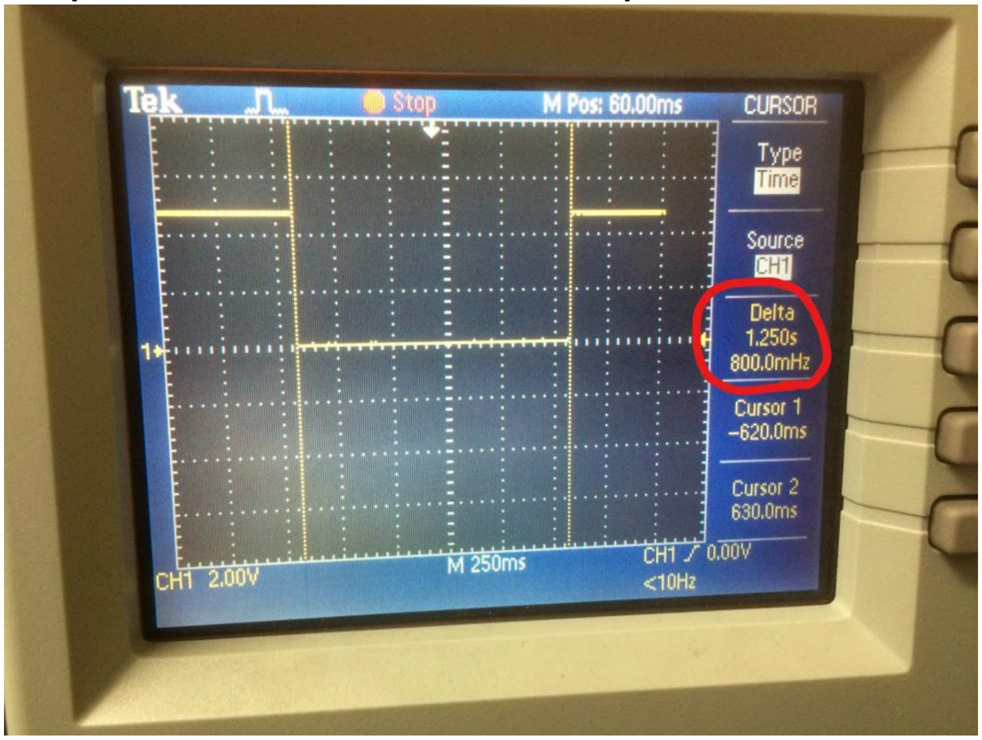


Task 2_B

Photo below shows the set up when testing for the pulse event

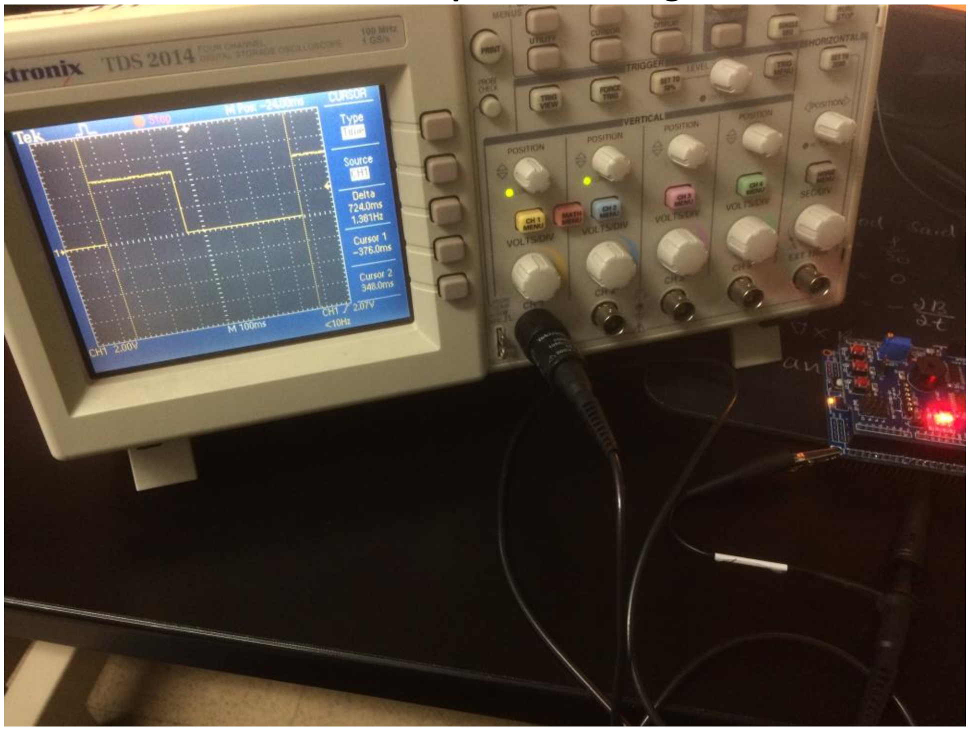


The photo below shows the time of the pulse event

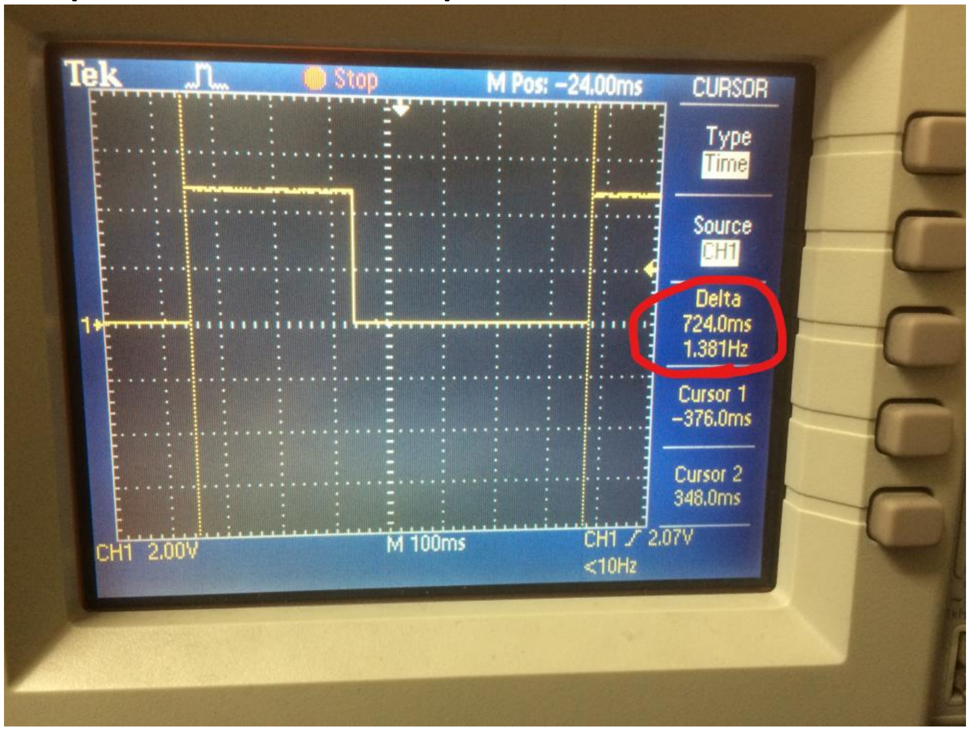


Task 3_A

Photo below shows the set up when testing for the waveform

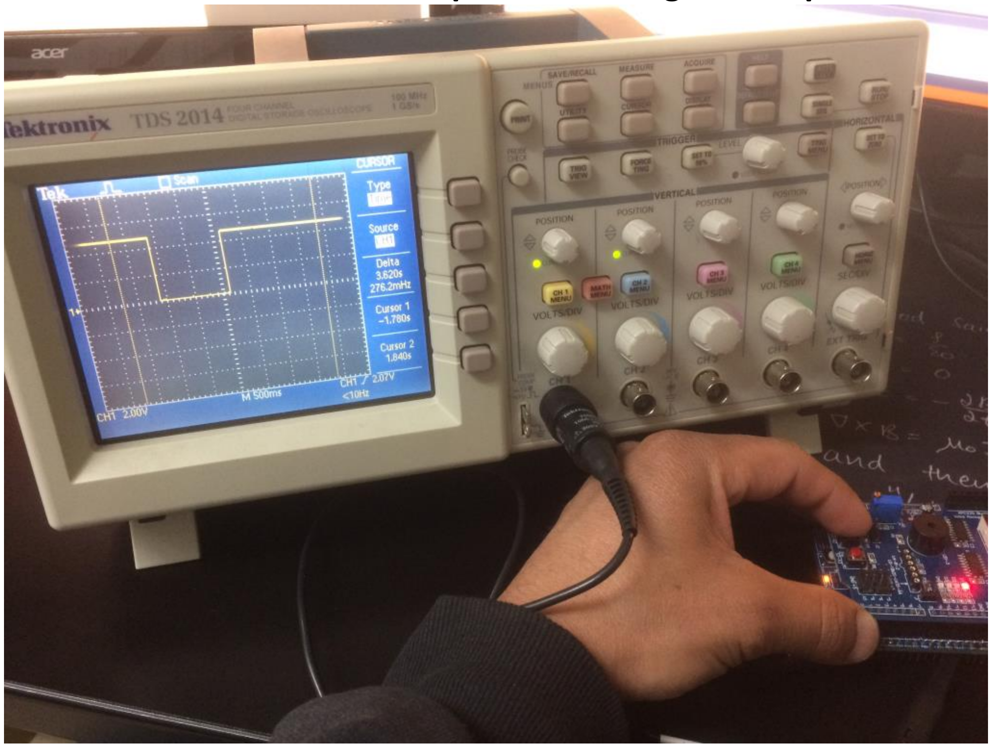


The photo below shows the period of the waveform

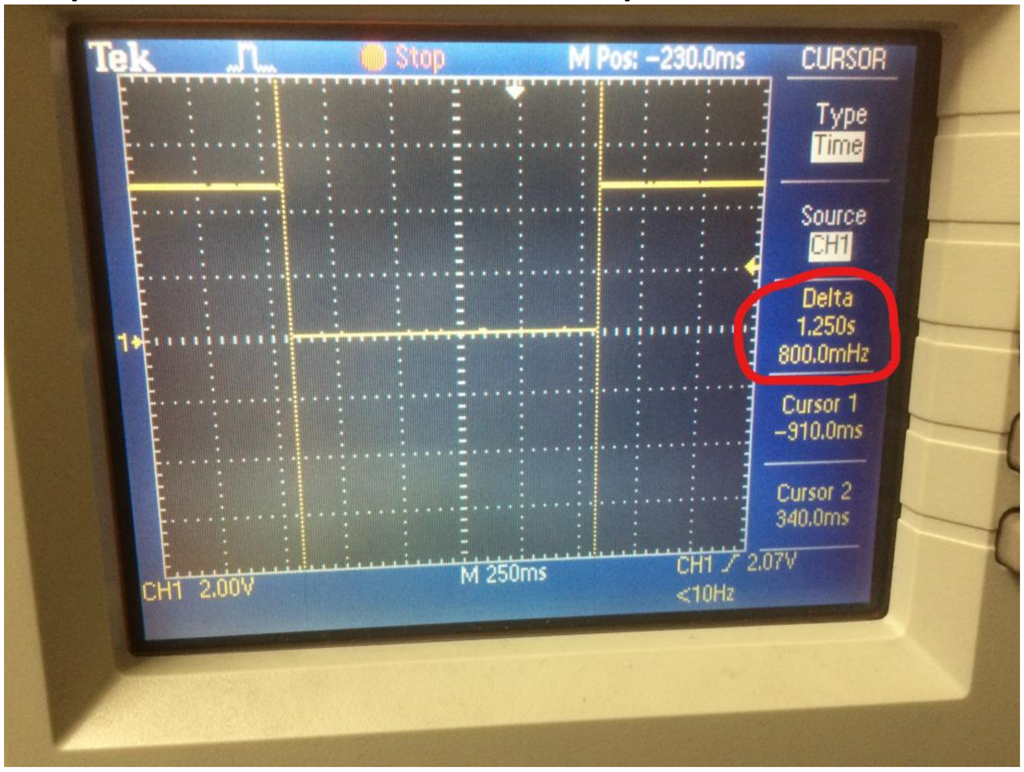


Task 3_B

Photo below shows the set up when testing for the pulse event



The photo below shows the time of the pulse event



7. VIDEO LINKS OF EACH DEMO

DA2C_1A
https://youtu.be/p1ocl3dLL1Y

DA2C_1B
https://youtu.be/bdinNLkg8vw

DA2C_2A
https://youtu.be/jHb87CrDjLo

DA2C_2B
https://youtu.be/mmECisDb9ek

DA2C_3A
https://youtu.be/m3VIlAYIQJo

DA2C_3B
https://youtu.be/6SgILz1tv3Y

8. GITHUB LINK OF THIS DA

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
Francisco Mata Carlos