Chaniporn Nerunchorn 6322771450
Ornnicha Phueaksri 6322773837

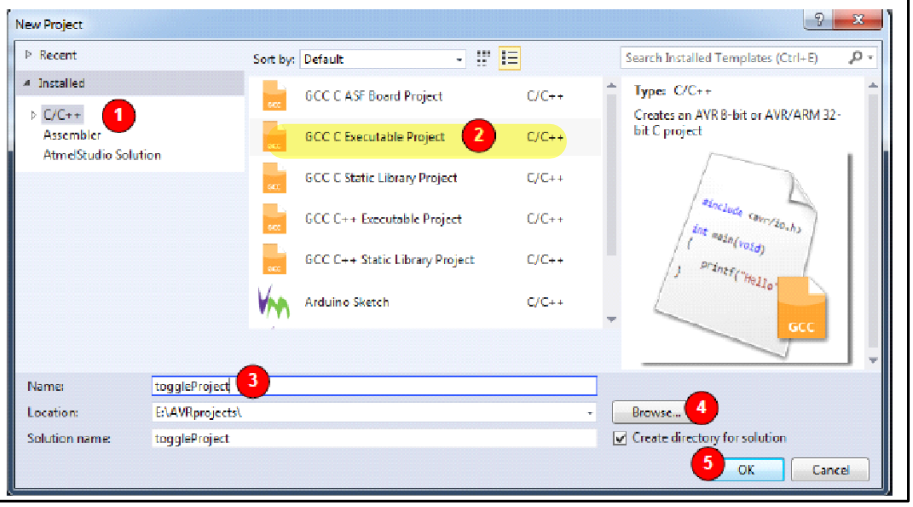
SIIT SINCE 1992	**School of ICT** **Sirindhorn International Institute of Technology**
CSS332 Microcontrollers and Applications	**Lab 8: AVR Programming in C**

Instructions: Answer the following exercises. During the lab class, please feel free to ask the instructor, the TAs, or other students if there is a question. When finishing all of them, the students can ask a TA to check the answers. The students submit this lab sheet with the answers to the Google Classroom (no submission, no score). (In the Google Classroom, do not forget to press the Confirm button to submit the work.)

In this lab, we will write C programs and upload them to the Arduino UNO board. To create a C program using the Atmel Studio 7, the only different step is that, when we create a new project, we choose C/C++ instead of Assembler as shown below. Please see the Step 2 on Page 4 of the sheet "C Programming in Atmel Studio 7", which and be downloaded from downloaded from (also, available in the Google Classroom):

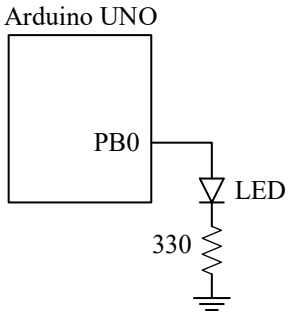
http://nicerland.com/eduFiles/AVR/Tutorials/CProgrammingInAtmelStudio7.pdf

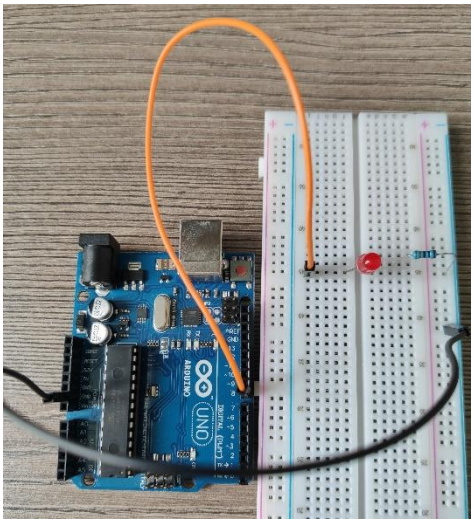
2. In the opened dialog,
   a. Choose ***C/C++***.
   b. Select ***GCC C Executable Project***.
   c. Name the project as ***toggleProject***.
   d. Choose the path where you like to save the project by clicking on the ***Browse*** button.
   e. Press ***OK***.



The instructor will demonstrate by using Exercise 1.

Exercise 1: In this exercise, we will connect a circuit and write a C program to turn on and off an LED every 1 second. Connect the circuit as shown below.



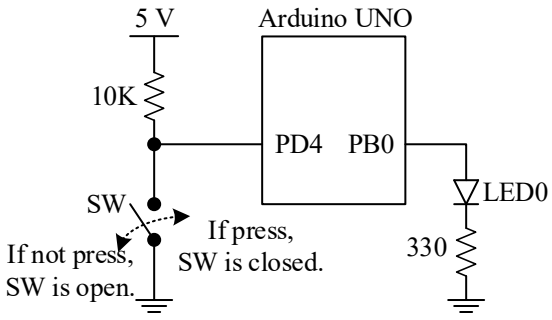


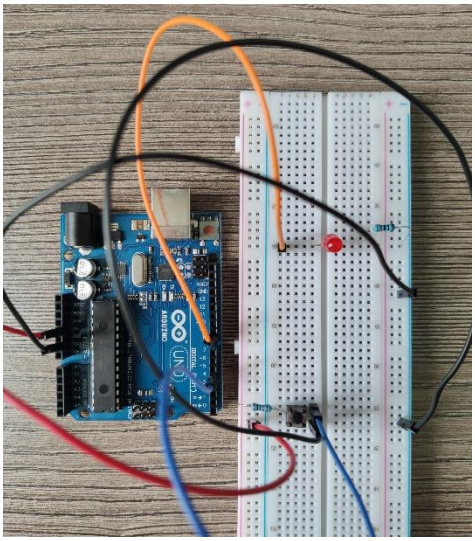
Write the C program as shown below which will turn on/off the LED every 1 second and upload to the Arduino UNO. Note that the instructor uses this exercise as demonstration, please see the video.

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

void PIN_SETUP() {          //A function to set up the pin modes
    DDRB |= (1<<0);         //Set PB0 as an output -> LED
    PORTB &= ~(1<<0);       //Set PB0 = 0
}

int main() {
    PIN_SETUP();            //Call the PIN_SETUP function
    while (1) {             //Repeat the while loop forever
        PORTB |= (1<<0);    //Turn the LED on
        _delay_ms(1000);    //for 1 second
        PORTB &= ~(1<<0);   //Turn the LED off
        _delay_ms(1000);    //for 1 second
    }
    return (0);
}
```

Take a video to demonstrate your result. Name it "Ex1" and submit to the Google Classroom.

Exercise 2: In this exercise, we will connect a circuit and write a C program to turn on and off an LED by pressing a switch. Connect the circuit as shown below.



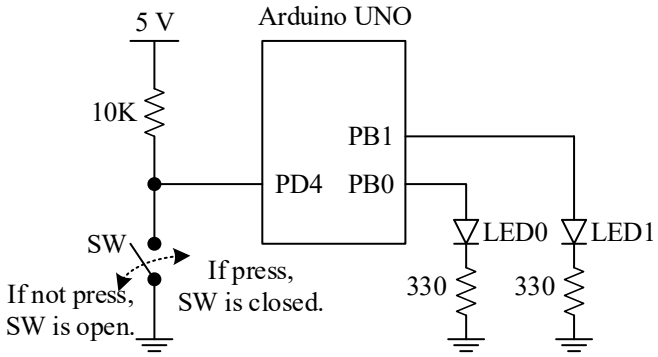


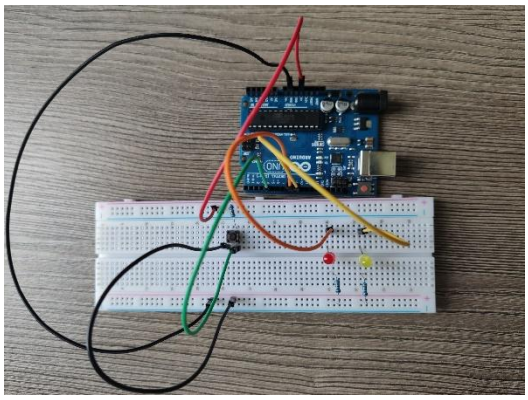
a) Please fill in the blanks to complete the following C program in order to fulfill the tasks required above.

```
1  #include <avr/io.h>
2
3  void PIN_SETUP() {              //A function to set up the pin modes
4      DDRB |= (1<<0);             //Set PB0 as an output -> LED0
5      PORTB &= ~(1<<0);           //Set PB0 = 0
6      DDRD &= ~(1<<4);            //Set PD4 as an input -> SW
7      PORTD |= (1<<4);            //Pull-up resistor for PD4
8  }
9
10 int main() {
11     PIN_SETUP();                //Call the PIN_SETUP function
12     while (1) {                 //Repeat forever
13         if ( ~PIND & (1<<4) ) {  //Check PD4=0 (press the switch)?
14             PORTB |= (1<<0);     //Yes, PB0=1 -> LED0 on
15             } else {
16             PORTB &= ~(1<<0);    //No, PB0=0 -> LED0 off
17         }
18     }
19     return (0);
20 }
```

b) Upload your C program to your Arduino UNO board. Take a video to demonstrate your result. Name it "Ex2" and submit to the Google Classroom.

Exercise 3: (**Using Timer0 overflow interrupt**). Similar to Exercise 1 in Lab 7, we will use the Time0 overflow interrupt to help create a time delay but in a C program. We have connected a circuit as shown below.





We would like to write an C program to do the following tasks.

- Task1 – The microcontroller monitors the status of the switch SW: if we press the switch SW, LED0 is on; if we do not press the switch SW, LED0 is off.
- Task2 – The microcontroller keeps turning on and off LED1, every 1 second.

As a result, the microcontroller will do Task1 in the main program and do Task2 by using the Timer0 overflow interrupt. Timer0 will be set up as follows:

- normal mode, pre-scaling number = 1024, the Timer0 overflow interrupt happens every 5000 μs.

As a result, we need that every 200 Timer0 overflow interrupts will turn the LED1 on or off (every 1 second). Recall the following related registers are set up as follows (from Exercise 1 in Lab 7):

- TCNT0 = 0xB2
- TCCR0A = 0x00
- TCCR0B = 0x05
- TIMSK0 = 0x01

Answer the following questions.

a) Similar to the structure of the Assembly program shown in Exercise 1 in Lab 7, we have the following C program to fulfill the tasks required above. Please fill in the blanks.

```
1   #include <avr/io.h>
2   #include <avr/interrupt.h>
3
4   unsigned char z = 200;
5
6   void PIN_SETUP() {       //A function to set up the pin modes
7       DDRB |= (1<<0);      //Set PB0 as an output -> LED0
8       PORTB &= ~(1<<0);    //Set PB0 = 0
9       DDRB |= (1<<1);      //Set PB1 as an output -> LED1
10      PORTB &= ~(1<<1);    //Set PB1 = 0
11      DDRD &= ~(1<<4);     //Set PD4 as an input -> SW
12      PORTD |= (1<<4);     //Pull-up resistor for PD4
13  }
14
```

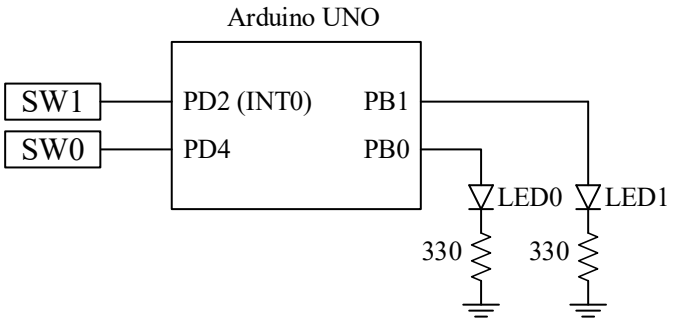
```
15  void TIMER0_SETUP() {   //A function to set TIMER0
16      TCNT0 = 0xB2;       //Set initial value of Timer0
17      TCCR0A = 0x00;      //Set the normal mode
18      TCCR0B = 0x05;      //Pre-scaling = 1024, start the timer
19  }
20
```

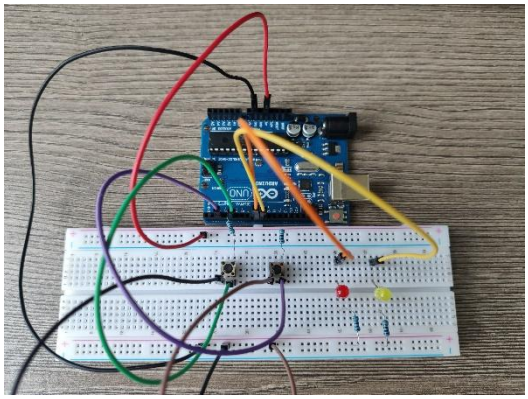
```
21  int main() {
22      PIN_SETUP();                //Call the PIN_SETUP function
23      TIMER0_SETUP();             //Call the TIMER0_SETUP function
24      TIMSK0 = (1<<TOIE0);        //Enable the Timer0 Overflow Interrupt
25      sei();                      //Enable the global interrupt
26      while (1) {
27          if ( ~PIND & (1<<4) ) {  //Check PD4=0?
28              PORTB |= (1<<0);     //Yes, PB0=1 -> LED0 on
29          } else {
30              PORTB &= ~(1<<0);    //No, PB0=0 -> LED0 off
31          }
32      }
33      return (0);
34  }
35
```

```
36  ISR (TIMER0_OVF_vect)       //ISR of Timer0 interrupt
37      z--;                    //Decrease z by 1
38      if (z==0) {             //Check z=0
39          z = 200;            //Yes, reset z = 200
40          PORTB ^= (1<<1);    //and toggle PB1
41      }
42      TCNT0 = 0xB2;           //Set TCNT0 = 0xB2
43  }
```

b) Upload your C program to your Arduino UNO board. Take a video to demonstrate your result. Name it “Ex3” and submit to the Google Classroom.

Exercise 4: (**Using the external interrupt INT0**). Similar to Exercise 2 in Lab 7, we will use the external interrupt INT0 but in a C program. We have connected a circuit as shown below.





We would like to write a C program to do the following tasks.

- Task1 – The microcontroller monitors the status of the switch SW0: if we press the switch SW0, LED0 is on; if we do not press the switch SW0, LED0 is off.
- Task2 – The microcontroller monitors the status of the switch SW1: if we press and release the switch SW1, the LED1 will be toggled (on off or off on).

As a result, the microcontroller will do Task1 in the main program and do Task2 by using the external interrupt INT0 (falling-edge trigger). Recall the following related registers are set up as follows (from Exercise 2 in Lab 7):

- EIMSK = 0x01
- EICRA = 0x02

Answer the following questions.

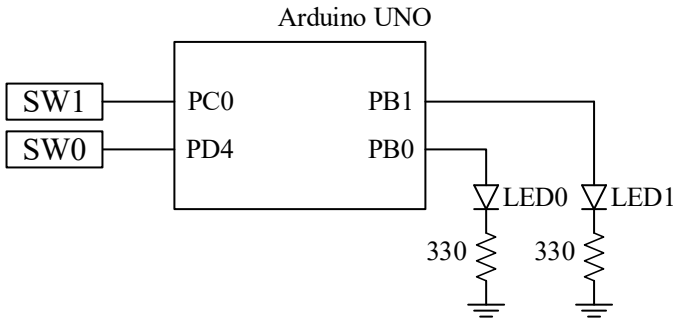
a) Similar to the structure of the Assembly program shown in Exercise 2 in Lab 7, we have the following C program to fulfill the tasks required above. Please fill in the blanks.

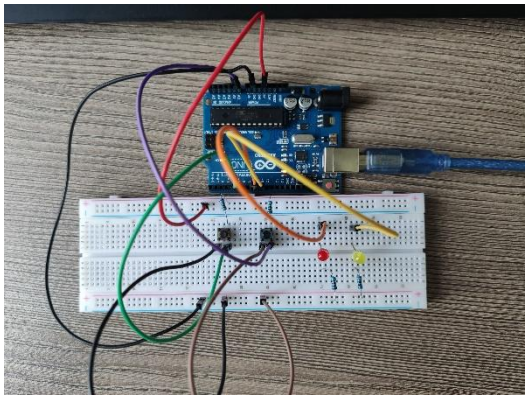
```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  void PIN_SETUP() {       //A function to set up the pin modes
5      DDRB |= (1<<0);      //Set PB0 as an output -> LED0
6      PORTB &= ~(1<<0);    //Set PB0 = 0
7      DDRB |= (1<<1);      //Set PB1 as an output -> LED1
8      PORTB &= ~(1<<1);    //Set PB1 = 0
9      DDRD &= ~(1<<4);     //Set PD4 as an input -> SW
10     PORTD |= (1<<4);     //Pull-up resistor for PD4
11     PORTD |= (1<<2);     //Pull-up resistor for PD2(INT0)
12 }
13
```

```
14  int main() {
15      PIN_SETUP();                    //Call the PIN_SETUP function
16      EIMSK = 0x01;                   //Enable the INT0 Interrupt
17      EICRA = 0x02;                   //Set the falling-edge trigger
18      sei();                          //Enable the global interrupt
19      while (1) {
20          if ( ~PIND & (1<<4) ) {     //Check PD4=0?
21              PORTB |= (1<<0);        //Yes, PB0=1 -> LED0 on
22              } else {
23              PORTB &= ~(1<<0);       //No, PB0=0 -> LED0 off
24          }
25      }
26      return (0);
27  }
28
29  ISR (INT0_vect)                     //ISR of INT0 interrupt
30      PORTB ^= (1<<1);                //and toggle PB1
31  }
```

b) Upload your C program to your Arduino UNO board. Take a video to demonstrate your result. Name it “Ex4” and submit to the Google Classroom.

Exercise 5: (**Using the pin change interrupt**). Similar to Exercise 3 in Lab 7, we will use the pin change interrupt but in a C program. We have connected a circuit as shown below.





We would like to write an Assembly program to do the following tasks.

- Task1 – The microcontroller monitors the status of the switch SW0: if we press the switch SW0, LED0 is on; if we do not press the switch SW0, LED0 is off.
- Task2 – The microcontroller monitors the status of the switch SW1: if we press and release the switch SW1, the LED1 will be toggled (on off or off on).

As a result, the microcontroller will do Task1 in the main program and do Task2 by using the pin change interrupt (via the switch SW1 connected to the pin PC0). Recall the following related registers are set up as follows (from Exercise 3 in Lab 7):

- PCICR = 0x02
- PCMSK0 = -
- PCMSK1 = 0x01
- PCMSK2 = -

Answer the following questions.

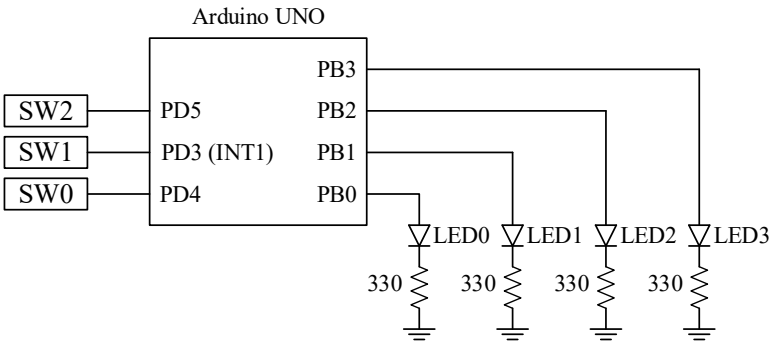
a) Similar to the structure of the Assembly program shown in Exercise 3 in Lab 7, we have the following C program to fulfill the tasks required above. Please fill in the blanks.

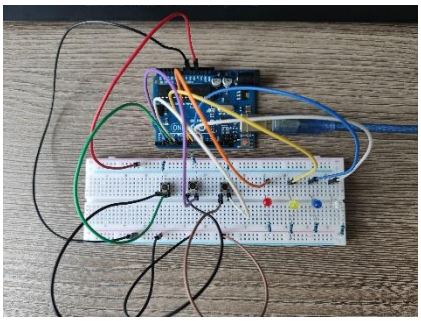
```
1  #include <avr/io.h>
2  #include <avr/interrupt.h>
3
4  unsigned char z = 2;
5
6  void PIN_SETUP() {       //A function to set up the pin modes
7      DDRB |= (1<<0);      //Set PB0 as an output -> LED0
8      PORTB &= ~(1<<0);    //Set PB0 = 0
9      DDRB |= (1<<1);      //Set PB1 as an output -> LED1
10     PORTB &= ~(1<<1);    //Set PB1 = 0
11     DDRD &= ~(1<<4);     //Set PD4 as an input -> SW
12     PORTD |= (1<<4);     //Pull-up resistor for PD4
13     PORTC |= (1<<0);     //Pull-up resistor for PC0 (Pin Change Interrupt)
14 }
15
```

```
int main() {
    PIN_SETUP();                    //Call the PIN_SETUP function
    PCICR = 0x02                    //Enable the PORTC Pin Change Interrupt
    PCMSK1 = 0x01                   //Enable the interrupt from PC0
    sei();                          //Enable the global interrupt
    while (1) {
        if ( ~PIND & (1<<4) ) {  //Check PD4=0?
            PORTB |= (1<<0);      //Yes, PB0=1 -> LED0 on
            } else {
            PORTB &= ~(1<<0);     //No, PB0=0 -> LED0 off
        }
    }
    return (0);
}

ISR (PCINT1_vect) {          //ISR of PCINT1 interrupt at C
    z--;
    if (z==0) {
        PORTB ^= (1<<1);         //and toggle PB1
        z=2;
    }
}
```

b) Upload your C program to your Arduino UNO board. Take a video to demonstrate your result. Name it "Ex5" and submit to the Google Classroom.

Exercise 6: (**Using the Timer0 overflow interrupt, external interrupt INT1, and pin change interrupt**) Similar to Exercise 4 in Lab 7, we will use theTimer0 overflow interrupt, external interrupt INT1, and pin change interrupt but in a C program. Connect the circuit as shown below.





Write a C program to do the following tasks.

- Task1 – The microcontroller monitors the status of the switch SW0: if we press the switch SW0, LED0 is on; if we do not press the switch SW0, LED0 is off.
- Task2 – The microcontroller monitors the status of the switch SW1: if we press and release the switch SW1, the LED1 will be toggled (on off or off on).
- Task3 – The microcontroller monitors the status of the switch SW2: if we press and release the switch SW2, the LED2 will be toggled (on off or off on).
- Task4 – The microcontroller keeps turning on and off LED3, every 1 second.

As a result, we design such that:

- the microcontroller will do Task1 in the main program,
- the microcontroller will do Task2 by using the external interrupt INT1 (fall-edge trigger),
- the microcontroller will do Taks3 by using the pin change interrupt (via the switch SW2 connected to the pin PD5),
- the microcontroller will do Task4 by using the Timer0 overflow interrupt (similar to Exercise 3 – used the same setup).

Recall the following related registers are set up as follows (from Exercise 4 in Lab 7):

- TCTN0 = 0xB2
- TCCR0A = 0x01
- TCCR0B = 0x05
- TIMSK0 = 0x01
- EIMSK = 0x02
- EICRA = 0x08
- PCICR = 0x04
- PCMSK2 = 0x20

Answer the following questions.

a) Similar to the structure of the Assembly program shown in Exercise 4 in Lab 7, we have the following C program to fulfill the tasks required above. Please fill in the blanks.

```c
#include <avr/io.h>
#include <avr/interrupt.h>

unsigned char y = 200;
unsigned char z = 2;

void PIN_SETUP() {      //A function to set up the pin modes
    DDRB |= (1<<0);     //Set PB0 as an output -> LED0
    PORTB &= ~(1<<0);   //Set PB0 = 0
    DDRB |= (1<<1);     //Set PB1 as an output -> LED1
    PORTB &= ~(1<<1);   //Set PB1 = 0
    DDRB |= (1<<2);     //Set PB2 as an output -> LED2
    PORTB &= ~(1<<2);   //Set PB2 = 0
    DDRB |= (1<<3);     //Set PB3 as an output -> LED3
    PORTB &= ~(1<<3);   //Set PB3 = 0
    DDRD &= ~(1<<4);    //Set PD4 as an input -> SW
    PORTD |= (1<<4);    //Pull-up resistor for PD4
    PORTD |= (1<<2);    //Pull-up resistor for PD3(INT1)
    PORTD |= (1<<5);    //Pull-up resistor for PD5(Pin Change)
}

void TIMER0_SETUP() {   //A function to set TIMER0
    TCNT0 = 0xB2;       //Set initial value of Timer0
    TCCR0A = 0x00;      //Set the normal mode
    TCCR0B = 0x06;      //Pre-scaling = 1024, start the timer
}

int main() {
    PIN_SETUP();                //Call the PIN_SETUP function
    TIMER0_SETUP();             //Call the TIMER0_SETUP function

    //INT0 Interrupt Setup
    EIMSK = 0x02;               //Enable the INT0 Interrupt
    EICRA = 0x06;               //Set the falling-edge trigger

    //PD5 Pin Change Interrupt
    PCICR = 0x04;               //Enable the PORTD Pin Change Interrupt
    PCMSK2 = 0x20;              //Enable the interrupt from PD5

    //Timer0 Overflow Interrupt
    TIMSK0 = 0x01;              //Enable the Timer0 Overflow Interrupt

    sei();                      //Enable the global interrupt
    while (1) {
        if ( ~PIND & (1<<4) ) {  //Check PD4=0?
            PORTB |= (1<<0);     //Yes, PB0=1 -> LED0 on
            } else {
            PORTB &= ~(1<<0);    //No, PB0=0 -> LED0 off
        }
    }
    return (0);
}
```

```
54 ISR (INT1_vect)                  //ISR
55     PORTB ^= (1<<1);             //and toggle PB1
56 }
57
58 ISR (PCINT2_vect)                //ISR
59     z--;                         //Decrease z by 1
60     if (z==0) {                  //Check if z=0?
61         PORTB ^= (1<<2);         //and toggle PB2
62         z=2;
63     }
64 }
65
66 ISR (TIMER0_OVF_vect)            //ISR
67     y--;                         //Decrease y by 1
68     if (y==0) {                  //Check y=0
69         y = 200;                 //Yes, reset y = 200
70         PORTB ^= (1<<3);         //and toggle PB3
71     }
72     TCNT0 = 0xB2;                //Set TCNT0 = 0xB2
73 }
```

b) Upload your C program to your Arduino UNO board. Take a video to demonstrate your result. Name it "Ex6" and submit to the Google Classroom.