Mechatronics Lab
Experiment 1 report
LED control using Arduino Micro-controller

Purvag lapsiwala UIN:662689378
Date: 7/Feb/2018

Contents

1 Summary

In this experiment we have leaned to use IDE(integrated development environment) software of Arduino and also different I/O ports involved in arduino. We have also learned the usage of Interrupt in Arduino programming along with regular polling methods. We used this interrupt and polling method to program arduino to switch on and off the led light based on the dip switch signal and interrupt signal. Two types of Interrupt are used in the Arduino Programming for this experiment first one is timer interrupt and second one is External interrupt. The timer interrupt is used in the IDE using the library 'Timerone.h'.

2 List of Components

1.Arduino Uno

2.LED (4)

3.Switch

4.Resistors

3 Description of the experiment

3.1 Purpose and Objective

Purpose of the experiment is to learn the use integrated development environment (IDE) software of the Arduino microcontroller. We also learnt basic hardware interface between a microcontroller and digital input device like DIP switch and digital output devices like LED. We also learnt how to program the Arduino in three different paradigms i.e. Polling method, External Interrupt, Interrupts generated by Timers. Our program objective was to read the digital data from the switch and digital write it to the LED.

3.2 Theory

Polling Method: it is paradigm in which the microcontroller under program control constantly checks or reads the input data from the switch and writes it to the LED. Not efficient way to program because microcontroller is always busy with one task of checking for the input. The part which checks for the inputs from the switches and writes it as output in written in void loop() function, which runs indefinitely. External Interrupt: here the microcontroller does its own work until there is interrupt from the external source i.e. interrupt generated on the D2 and D3 pins. When the interrupt flag is raised it diverts it resource from current work and servers the user. D2 and D3 pins can be designed to give interrupt for different modes like Rising, Falling, Change, Low etc. Here we write a separate ISR for handling the interrupt.
Timer Interrupt: exactly works on above principle except the interrupt is generated at regular frequency when the software timer reaches zero, it then raises a flag to the microcontroller to serve the user. For this experiment we have used a Timer1.h library taken from internet, for using timer interrupts. When an interrupt comes the Arduino goes to the interrupt vector table (IVT) which contains the address of the ISR. Function handling the interrupts is known as interrupt service routine (ISR), it the handler basically which knows what to do when there is interrupt.

Pseudo code as trivial as reading the data from input register and writing it to the output register, here for coding purposes we have used in built C functions for ease of coding.
Regarding digital pins, the default state of digital inputs can be defined by either connecting the input port pin to a supply voltage (i.e., 5 VDC) or to ground via a resistor. If the connection is made to the supply voltage (i.e., default ON state), it is called pulled-up and the resistor is called the pull-up resistor. If the connection is made to the ground (i.e., default OFF state), it is called pulled-down and the resistor is called the pull-down resistor. The appropriate value for the pull-up and pull-down resistors are determined by the supply voltage and maximum current draw specifications of the microcontroller, here we have taken 100 kilo ohms as pullup resistor.
In our code for external interrupt, the LED are turned on for every FALLING cycle in an alternative fashion i.e. when one set is ON for first rising cycle then for the next rising cycle another set of LED are turned ON.

4 Circuits Diagrams and Images

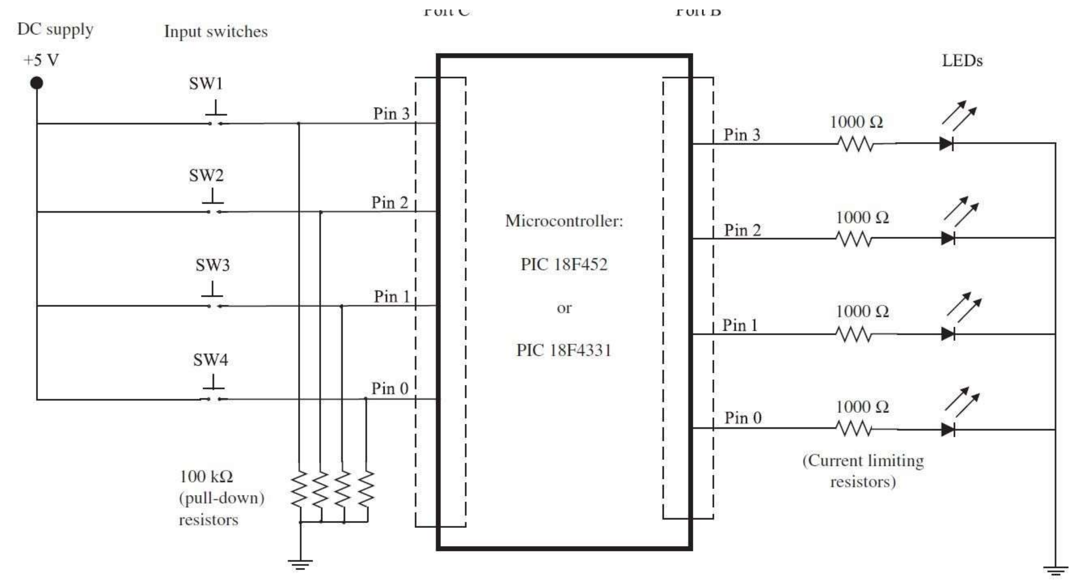


Figure 1: n the experiment the pull-down resistance configuration is used and Instead of PIC micro controller Arduino is used as shown in the next figure

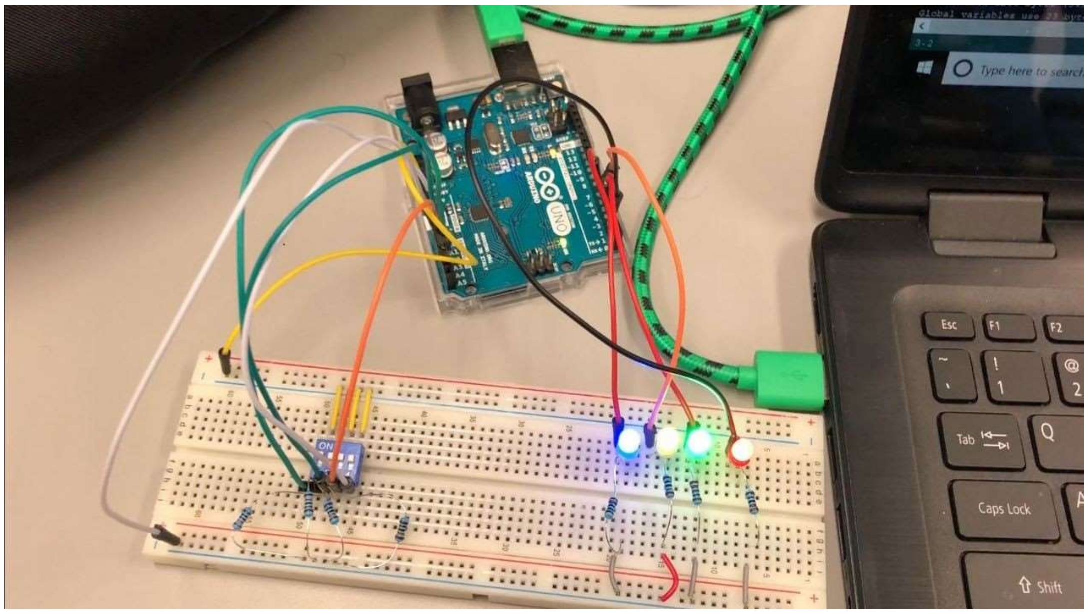


Figure 2: This is the actual circuit diagram

5 Procedure

1. The first step taken in the experiment is setup the arduino IDE and make the circuit connection as shown in the image, the pull down resistance combination is used in the this case.

2. Second step is to connect 8 DIP switch in the pull down configuration as shown in the circuit diagram.

3. Third step is to program a polling method program so to control the LED using the switch. The code is pasted below and the videos is uploaded along with this code.In the polling methods there is no interrupt but micro-controller is in a loop of given command.

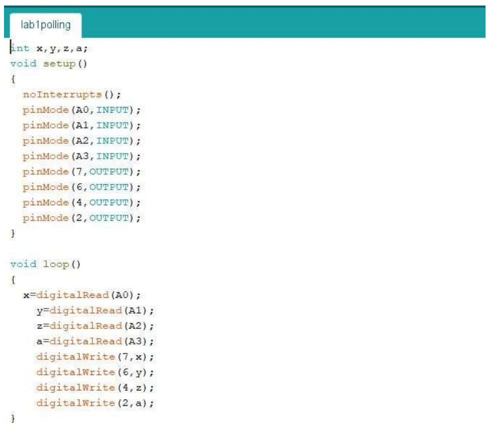
4. In this step, Timer interrupt is used to instead of polling method and to use this interrupt library 'TimerOne' is included in the Program in the header as TimerOne.h.
   '.initialize' method is used to produce interrupt in every 100 milli second. Then the method function "p" is attached as ISR using the method '.attachInterrupt'. So when timer1 produced Interrupt every 100 ms 'p' function is executed.

5. In the next step of experiment, External timers are used to instead of timer. The programmed is structured in such a way that if the pin 3 (i.e INT2) signal is 'Falling' then the alternative LED's are lit up. To achieve this 'IF' condition is used in the ISR of the Interrupt. Similar to the timer interrupt, External interrupt uses the attachInterrupt method in C library but the syntax is different when compared to the timer. Attach-Interrupt syntax in the External Interrupt is ' attachinterrupt(0 or 1, ISR, RISSING or FALLING OR CHANGING). The code and videos are uploaded along with this report.

6 Results and Code

6.1 Polling method

Polling method is used to control the LED fig 3 and fig 4 represents the code and images



```
int x,y,z,a;
void setup()
{
  noInterrupts();
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(7,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(2,OUTPUT);
}

void loop()
{
  x=digitalRead(A0);
    y=digitalRead(A1);
    z=digitalRead(A2);
    a=digitalRead(A3);
    digitalWrite(7,x);
    digitalWrite(6,y);
    digitalWrite(4,z);
    digitalWrite(2,a);
}
```

Figure 3: Flower one.

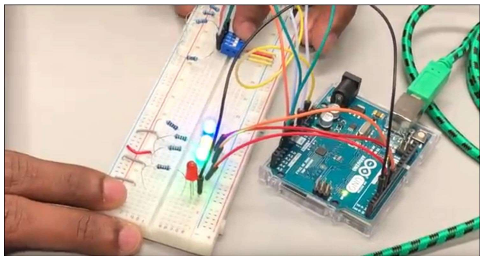
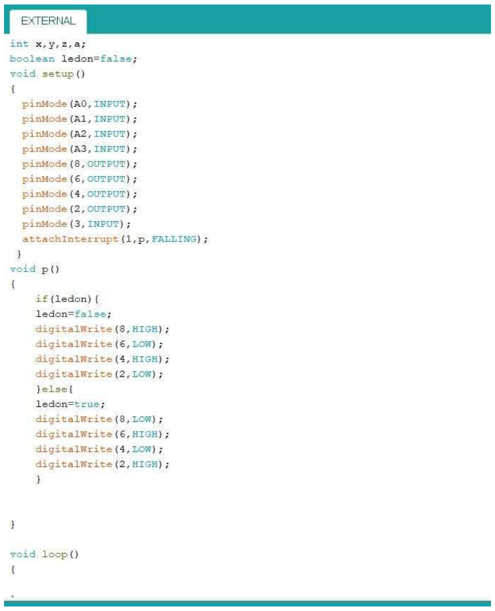


Figure 4: Flower two.

6.2 External Interrupt

This is code and Image of External Interrupt here the fig 6 and fig 7 illustrate the first and second LED combination achieved when the pin 3 value is falling. Every time signal falls in pin 3 LED’s combination changes



```
int x,y,z,a;
boolean ledon=false;
void setup()
{
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(8,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,INPUT);
  attachInterrupt(1,p,FALLING);
 }
void p()
{
    if(ledon){
    ledon=false;
    digitalWrite(8,HIGH);
    digitalWrite(6,LOW);
    digitalWrite(4,HIGH);
    digitalWrite(2,LOW);
    }else{
    ledon=true;
    digitalWrite(8,LOW);
    digitalWrite(6,HIGH);
    digitalWrite(4,LOW);
    digitalWrite(2,HIGH);
    }


}

void loop()
{
```

Figure 5: Code used for External Interrupt method

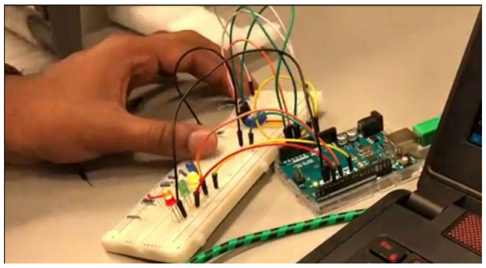


Figure 6: External Interrupt method first LED combination

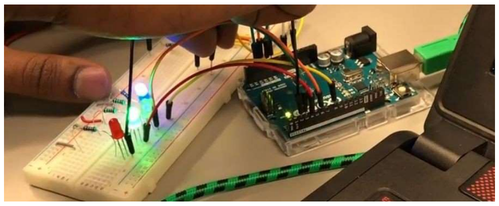
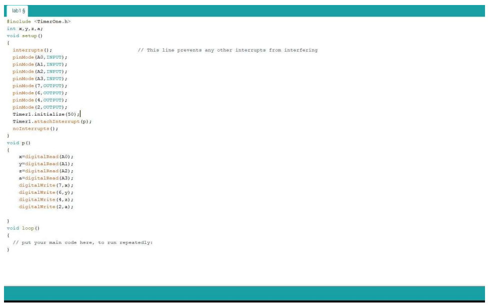


Figure 7: External Interrupt method Second LED combination

6.3 Timer Interrupt

Here we can see that all the four LED can be controlled using the Switch and timer Interrupt. Fig 8 and Fig 9 gives the



```
#include <TimerOne.h>
int x,y,z,a;
void setup()
{
  interrupts();                              // This line prevents any other interrupts from interfering
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(7,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(2,OUTPUT);
  Timer1.initialize(50);
  Timer1.attachInterrupt(p);
  noInterrupts();
}
void p()
{
    x=digitalRead(A0);
    y=digitalRead(A1);
    z=digitalRead(A2);
    a=digitalRead(A3);
    digitalWrite(7,x);
    digitalWrite(6,y);
    digitalWrite(4,z);
    digitalWrite(2,a);

}
void loop()
{
  // put your main code here, to run repeatedly:
}
```

Figure 8: Code used for timer method.

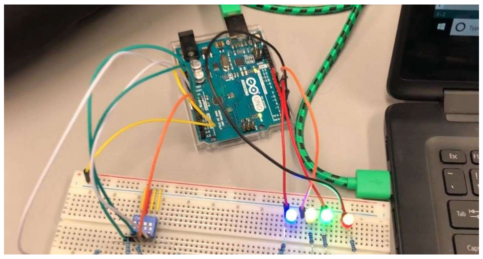


Figure 9: 4 LEd control using the timer Interrupt

7 Conclusion

In this experiment the objective i.e 4 LED’s are controlled using the polling, and Timer and External Interrupt method is completed using Arduino. Also in this experiment the basic concept of configuring I/O pins in arduino IDE both in C programming and registry level control is understood and practiced. This experiment illustrated the pull up and pull down resistance combination in the switching circuit.

8 References

1.Lab manual.