

**MICROPROCESSOR & MICROCONTROLLER**

**Laboratory Manual**

**Premier University Chittagong**

**Department of Electrical & Electronics Engineering**

**Premier University Chittagong**

**Department of Electrical & Electronic Engineering**

**Course Code: EEE-372**

**Course Title: Microprocessor & Microcontroller Lab**

|  |  |
| --- | --- |
| **Experiment no.** | **Name of Experiment** |
| **01.** | **Study on Different types of Microcontroller, Simulation software’s & Development Boards.** |
| **02.** | **Design a simple LED blinking circuit using PIC micro-controller.** |
| **03.** | **Design a 0-9 counter using 7 segment displays & using PIC micro-controller.** |
| **04.** | **Design a Traffic Control Light using LED & using PIC micro-controller.** |
| **05.** | **Design a stepper motor-based railway level Crossing & using PIC micro-controller.** |
| **06.** | **Familiarizing with Arduino, Arduino IDE & design a simple LED looping Circuit.** |
| **07.** | **PWM & it’s use in Arduino using Servo Motor.** |
| **08.** | **Design of an Arduino based Temperature Meter using LM 35 & LCD display.** |
| **09.** | **Design of a mobile control robotic car using Arduino, Bluetooth Module & Android Phone.** |
| **10** | **Sonar sensor-based distance calculator.** |
| **11** | **Bluetooth based Home Automation System.** |
| **12** | **To blink a led using Raspberry Pi and Python.** |

**Experiment No-01:** Study on Different types of Microcontroller, Simulation software’s & Development Boards.

**Objective:** To learn about Different types of Microcontroller, Simulation software’s & Development Boards.

**What is a Microcontroller?**

A microcontroller is a small, low-cost and self contained computer-on-a-chip that can be used as an embedded system. A few microcontrollers may utilize four-bit expressions and work at clock rate frequencies, which usually include:

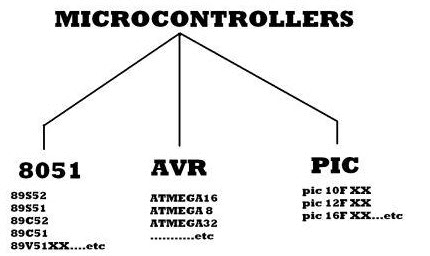
* An 8 or 16 bit microprocessor.
* A little measure of RAM.
* Programmable ROM and flash memory.
* Parallel and serial I/O.
* Timers and signal generators.
* Analog to Digital and Digital to Analog conversion

Microcontrollers usually must have low-power requirements since many devices they control are battery-operated. Microcontrollers are used in many consumer electronics, car engines, computer peripherals and test or measurement equipment. And these are well suited for long lasting battery applications. The dominant part of microcontrollers being used now a days are implanted in other apparatus.

How are Microcontrollers Classified?

The microcontrollers are characterized regarding bus-width, instruction set, and memory structure. For the same family, there may be different forms with different sources. This article is going to describe some of the basic types of the Microcontroller that newer users may not know about.

The types of microcontroller is shown in figure, they are characterized by their bits, memory architecture, memory/devices and instruction set. Let’s discuss briefly about it.



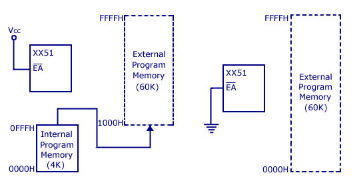
**4 Types of Microcontrollers:**

**Microcontroller 8051:**

It is a 40pin microcontroller with Vcc of 5V connected to pin 40 and Vss at pin 20 which is kept at 0V. And there are input and output ports from P1.0 – P1.7 and which having open drain feature. Port3 has got extra features. Pin36 has open drain condition and pin17 has internally pulled up transistor inside the microcontroller. When we apply logic 1 at port1 then we get logic 1 at port21 and vice versa. The programming of microcontroller is dead complicate. Basically we write a program in C-language which is next converted to machine language understand by the microcontroller. A RESET pin is connected to pin9, connected with a capacitor. When the switch is ON, the capacitor starts charging and RST is high. Applying a high to the reset pin resets the microcontroller. If we apply logic zero to this pin, the program starts execution from the beginning.

**Memory Architecture of 8051**

The memory of 8051 is divided into two parts. They are Program Memory and Data Memory. Program Memory stores the program being executed whereas Data Memory temporarily stores the data and the results. The 8051 has been in use in a wide number of devices, mainly because it is easy to integrate into a device. Microcontrollers are mainly used in energy management, touch screen, automobiles, and medical devices.



**Renesas Microcontroller:**

Renesas is latest automotive microcontroller family that offers high performance feature with exceptionally low power consumption over a wide and versatile extend of items. This microcontroller offers rich functional security and embedded safety characteristics required for new and advanced automotive applications. The core structure of microcontroller CPU support high reliability and high performance requirements.

The Renesas microcontroller offering low power, high performance, modest packages and the largest range of memory sizes combined together with characteristics rich peripherals.

[](https://www.elprocus.com/wp-content/uploads/2013/08/Renesas.jpg)

Renesas

Renesas offers the most versatile microcontroller families in the world for example our RX family offers a many types devices with memory variants from 32K flash/4K RAM to an incredible 8M flash/512K RAM.

The RX Family of 32-bit microcontrollers is a feature rich, general purpose MCU covering a wide range of embedded control applications with high speed connectivity, digital signal processing and inverter control.

The RX microcontroller family uses a 32-bit enhanced Harvard CISC architecture to achieve very high performance.

**AVR Microcontrollers:**

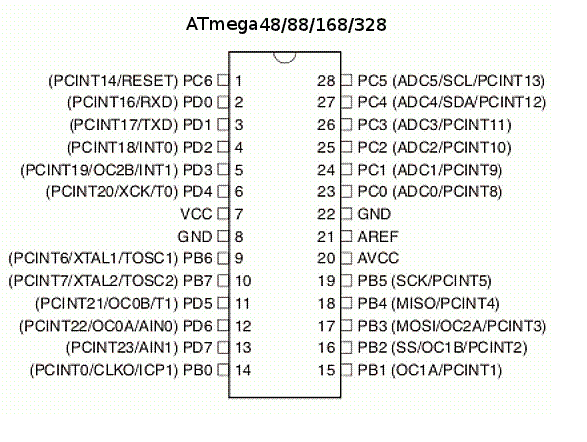
AVR microcontroller is developed by Alf-Egil Bogen and Vegard Wollan from Atmel Corporation. The AVR microcontrollers are modified harvard RISC architecture with separate memories for data and program and speed of AVR is high when compare to 8051 and PIC. The AVR is stands for Alf-Egil Bogen and Vegard Wollan’s RISC processor.

Difference between 8051 and AVR Controllers:

* 8051s are 8-bit controllers based on CISC architecture, AVRs are 8-bit controllers based on RISC architecture
* 8051 consumes more power than AVR microcontroller
* In 8051, we can program easily than the AVR microcontroller
* The speed of AVR is more than the 8051 microcontroller

Classification of AVR Controllers:

* AVR Microcontrollers are classified into three types:
* TinyAVR – Less memory, small size, suitable only for simpler applications
* MegaAVR – These are the most popular ones having good amount of memory (up to 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications
* XmegaAVR – Used commercially for complex applications, which require large program memory and high speed
* Features of AVR Microcontroller:
* 16KB of In-System Programmable Flash
* 512B of In-System Programmable EEPROM
* 16-bit Timer with extra features
* Multiple internal oscillators
* Internal, self-programmable instruction flash memory up to 256K
* In-system programmable using ISP, JTAG or high voltage methods
* Optional boot code section with independent lock bits for protection
* Synchronous/asynchronous serial peripherals (UART/USART)
* Serial peripheral interface bus (SPI)
* Universal serial interface (USI) for two/three-wire synchronous data transfer
* Watchdog timer (WDT)
* Multiple power-saving sleep modes
* 10-bit A/D Converters, with multiplex of up to 16 channels
* CAN and USB controller support
* Low-voltage devices operating down to 1.8v



**PIC Microcontroller:**

PIC is a peripheral interface controller, developed by general instrument’s microelectronics, in the year of 1993. It is controlled by the software. They could be programmed to complete many task and control a generation line and many more. PIC microcontrollers are finding their way into new applications like smart phones, audio accessories, video gaming peripherals and advanced medical devices.

There are many PICs, started with PIC16F84 and PIC16C84. But these were the only affordable flash PICs. Microchip has recently introduced flash chips with types that are much more attractive, such as 16F628, 16F877 and 18F452. The 16F877 is around twice the price of the old 16F84, but has eight times the code size, much more RAM, much more I/O pins, a UART, A/D converter and a lot more.

[](https://www.elprocus.com/wp-content/uploads/2013/08/PIC.png)

**Features of PIC16F877**

* Core Features:
* High-performance RISC CPU
* Up to 8K x 14 words of FLASH program memory
* 35 Instructions (fixed length encoding-14-bit)
* 368×8 static RAM based data memory
* Up to 256 x 8 bytes of EEPROM data memory
* Interrupt capability (up to 14 sources)
* Three addressing modes (direct, indirect, relative)
* Power-on reset (POR)
* Harvard architecture memory
* Power saving SLEEP mode
* Wide operating voltage range: 2.0V to 5.5V
* High sink / source current: 25mA
* Accumulator based machine

**Programming Software :**

* Proteus 7.0
* Micro C Pro.

**Experiment No-02:** Design a simple LED blinking circuit using PIC micro-controller.

**Objective:** To design a simple LED blinking circuit using PIC micro-controller.

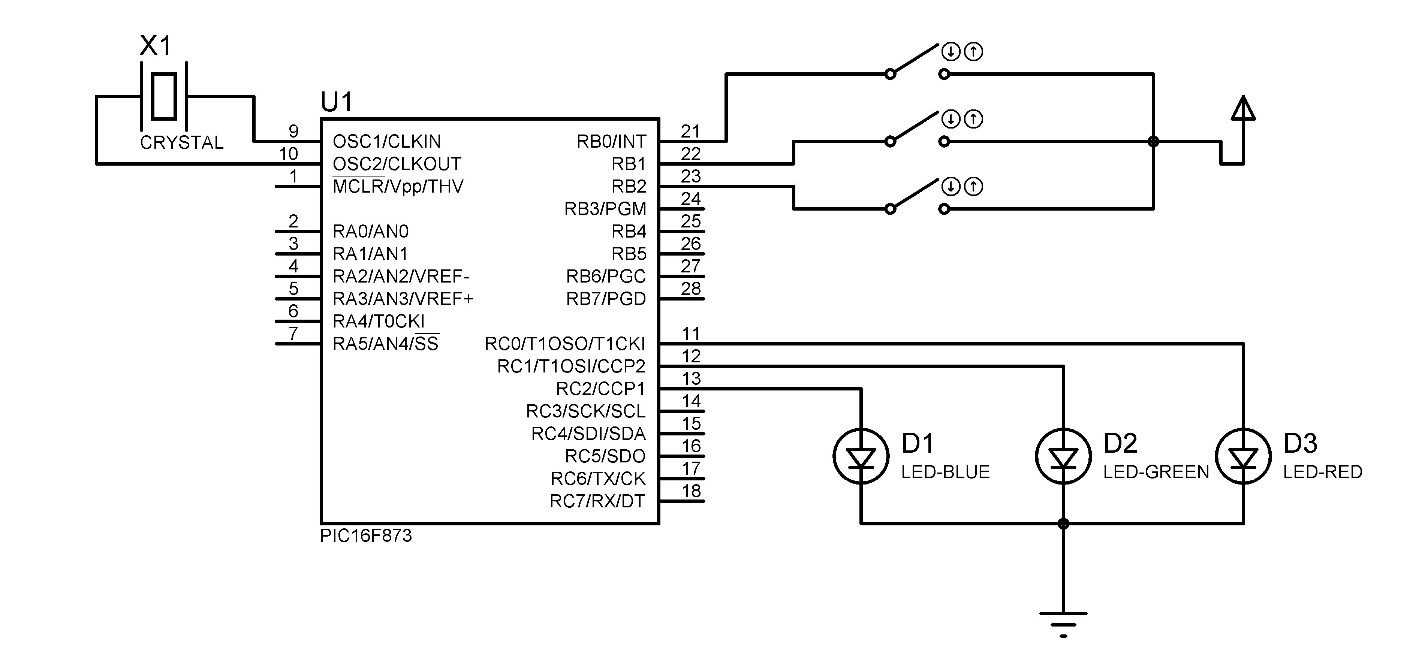
**Required Software:**

1. Proteus 7.0 Professional.
2. Micro C

**Instruments Required:**

1. PIC16f873 Microcontroller
2. LED
3. Switch
4. Crystal Oscillator
5. DC Power Supply

**Circuit Diagram:**



**Source code :**

void main()

{

trisb=0b00000111;

portb=0b00000000;

trisc=0b00000000;

portc=0b00000000;

while(1)

{

if(portb==0b00000001)

{

portc=0b00000001;

}

if(portb==0b00000010)

{

portc=0b00000011;

}

if(portb==0b00000100)

{

portc=0b00000111;

}

if(portb==0b00000000)

{

portc=0b00000000;

}

}

}

**Procedure:**

1. Open Proteus 7.0and design the given circuit.
2. Then in Micro c create a new Project and program it as given.
3. Then run the program it will create a .HAX file.
4. Put the Hex File in Micro controller And Run it.
5. Observe the LED sequence as per Programming.

**Discussion:**

**Experiment No-03:** Design a 0-9 counter using 7 segment displays & using PIC micro-controller.

**Objective:** To Design a 0-9 counter using 7 segment displays & using PIC micro-controller.

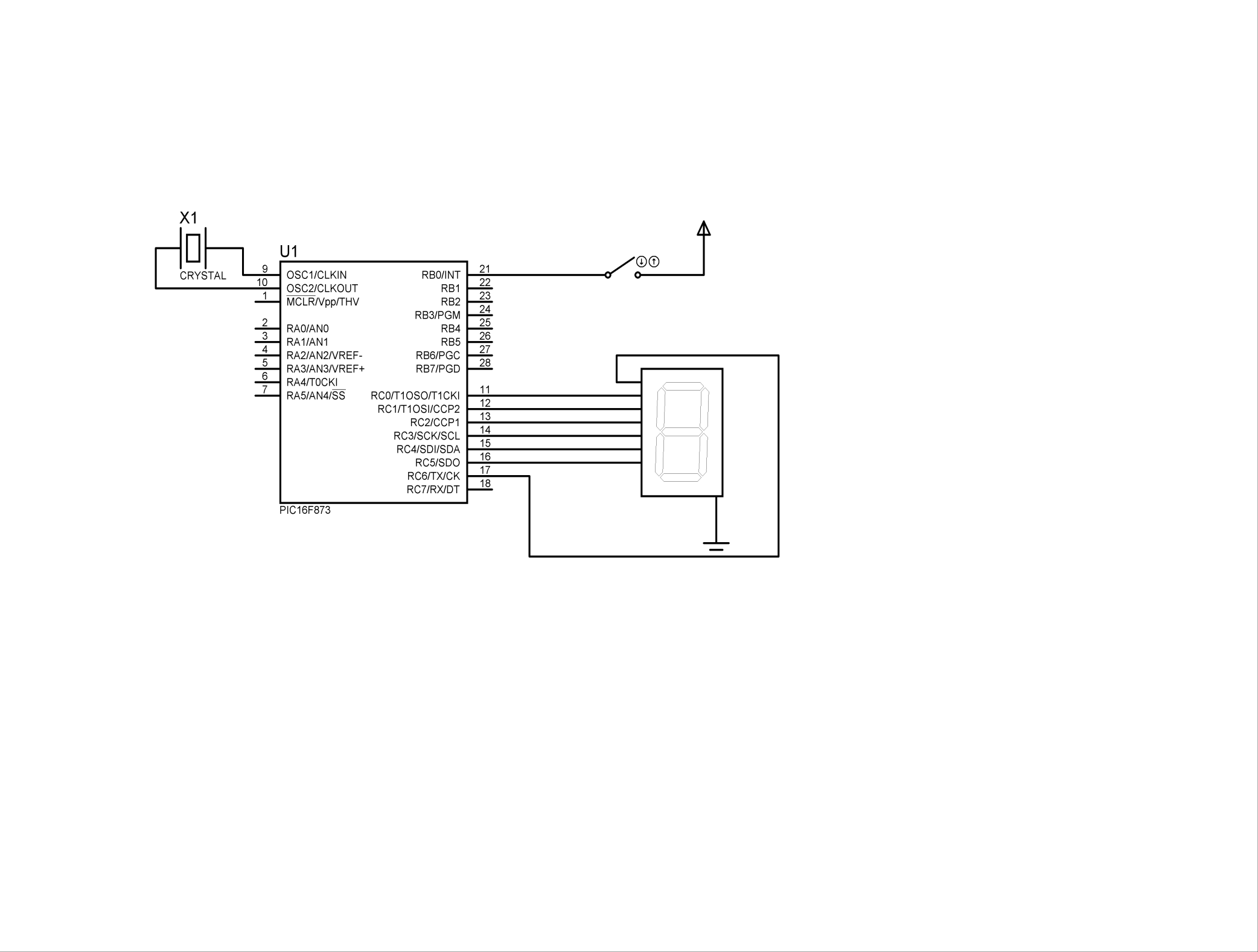
**Required Software:**

1. Proteus 7.0 Professional.
2. Micro C

**Instruments Required:**

1. PIC16f873 Microcontroller
2. Common Cathode Seven Segment Display.
3. Switch
4. Crystal oscillator
5. DC power Supply

**Circuit Diagram:**



**Source code:**

void main()

{

trisb=0b00000001;

portb=0b00000000;

trisc=0b00000000;

portc=0b00000000;

while(1)

{

if(portb=0b00000001)

{

portc=0b01011111;

delay\_ms(300);

portc=0b00000011;

delay\_ms(300);

portc=0b01101101;

delay\_ms(300);

portc=0b01100111;

delay\_ms(300);

portc=0b00110011;

delay\_ms(300);

portc=0b01110110;

delay\_ms(300);

portc=0b01111110;

delay\_ms(300);

portc=0b01100011;

delay\_ms(300);

portc=0b01111111;

delay\_ms(300);

portc=0b01110111;

delay\_ms(300);

}

}

}

**Procedure:**

1. Open Proteus 7.0and design the given circuit.
2. Then in Micro c create a new Project and program it as given.
3. Then run the program it will create a .HAX file.
4. Put the Hex File in Micro controller And Run it.
5. Observe the Segment as per Programming.

**Discussion:**

**Experiment No-04:** Design a Traffic Control Light using LED & using PIC micro-controller.

**Objective:** To Design traffic Control Light using LED & using PIC micro-controller.

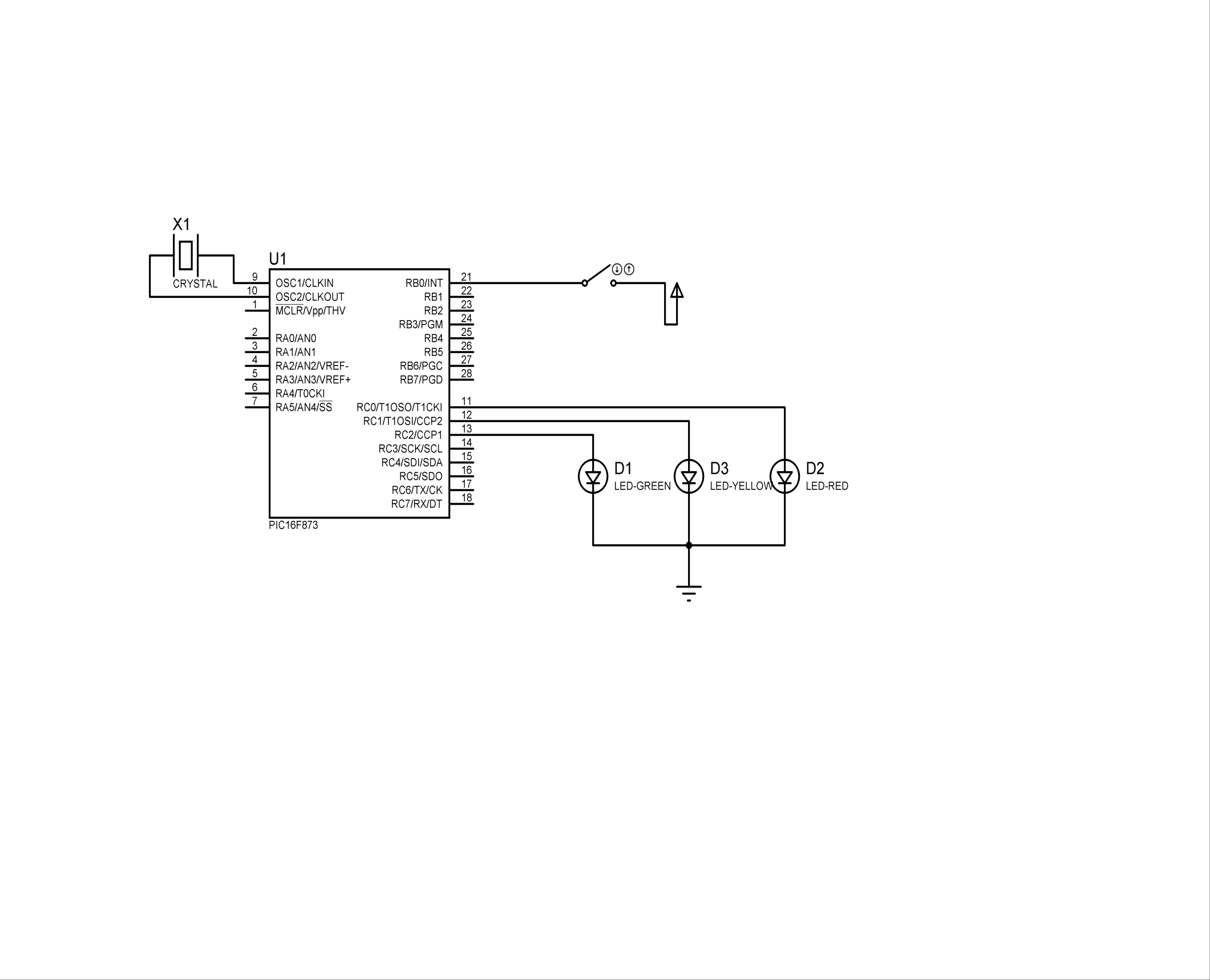
**Required Software:**

1. Proteus 7.0 Professional.
2. Micro C

**Instruments Required:**

1. PIC16f873 Microcontroller
2. LED ( RED, YELLOW, GREEN)
3. Switch
4. Crystal oscillator
5. DC power Supply

**Circuit Diagram:**



**Source code :**

void main()

{

trisb=0b00000001;

portb=0b00000000;

trisc=0b00000000;

portc=0b00000000;

while(1)

{

if(portb==0b00000001)

{

portc=0b00000001;

delay\_ms(1000);

portc=0b00000010;

delay\_ms(300);

portc=0b00000100;

delay\_ms(1000);

}

if(portb==0b00000000)

{

portc=0b00000000;

}

}

}

**Procedure:**

1. Open Proteus 7.0and design the given circuit.
2. Then in Micro c create a new Project and program it as given.
3. Then run the program it will create a .HAX file.
4. Put the Hex File in Micro controller And Run it.
5. Observe the LED as per Traffic LIGHT.

**Discussion:**

**Experiment No-05:** Design a steeper motor based railway level Crossing & using PIC micro-controller.

**Objective:** To design a steeper motor based railway level Crossing & using PIC micro-controller.

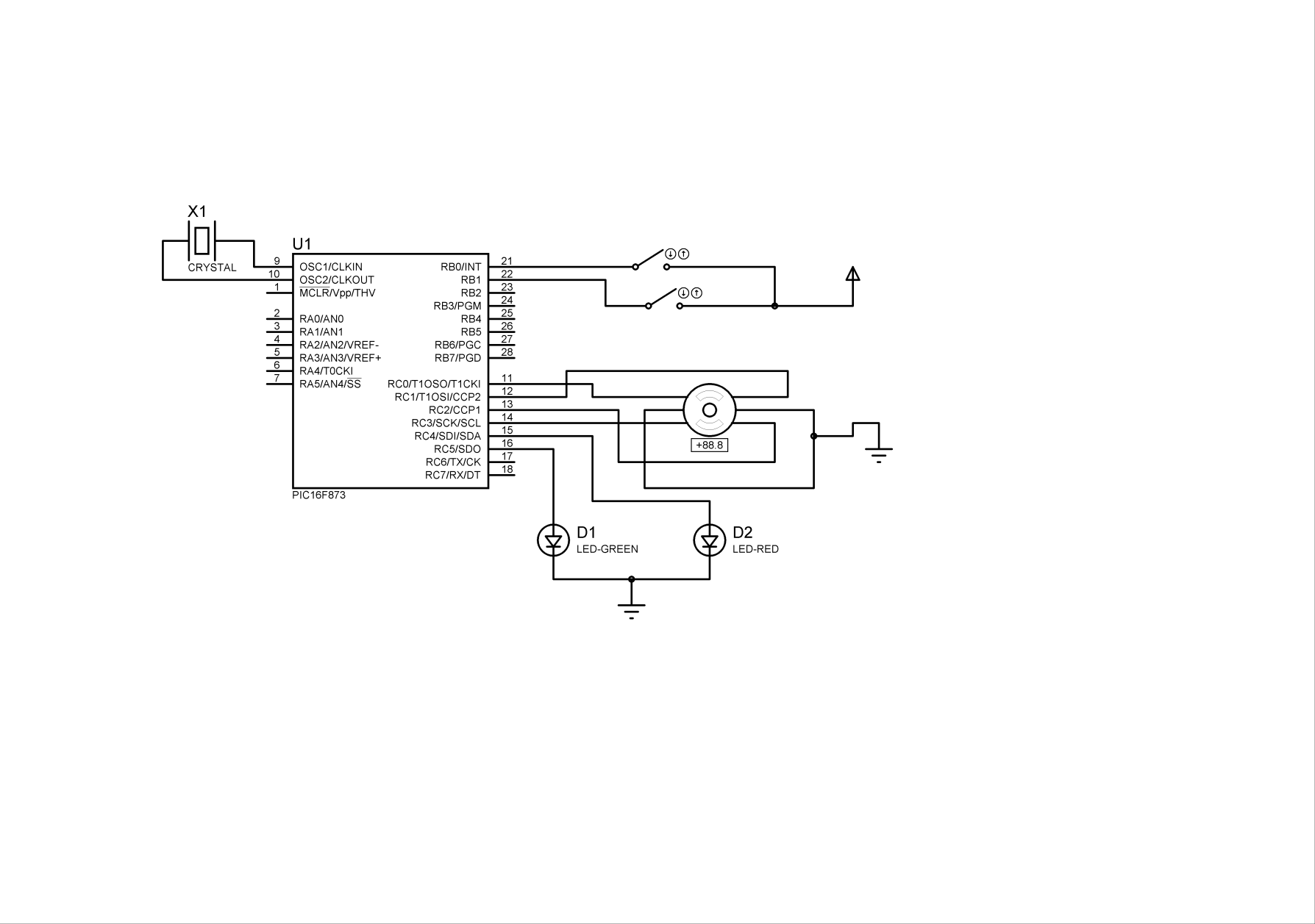
**Required Software:**

1. Proteus 7.0 Professional.
2. Micro C

**Instruments Required:**

1. PIC16f873 Microcontroller
2. LED ( RED, GREEN)
3. Steeper Motor.
4. Switch
5. Crystal oscillator
6. DC power Supply

**Circuit Diagram:**



**Source code:**

void main()

{

trisb=0b00000011;

trisc=0b00000000;

portb=0b00000000;

portb=0b00000000;

while(1)

{

if(portb==0b00000001)

{

portc=0b00010001;

delay\_ms(200);

portc=0b00010010;

delay\_ms(200);

portc=0b00010100;

delay\_ms(200);

portc=0b00011000;

delay\_ms(200);

}

if(portb==0b00000010)

{

portc=0b00101000;

delay\_ms(200);

portc=0b00100100;

delay\_ms(200);

portc=0b00100010;

delay\_ms(200);

portc=0b00100001;

delay\_ms(200);

}

if(portb=0b00000000)

{

portc=0b00000000;

}

}

}

**Procedure:**

1. Open Proteus 7.0and design the given circuit.
2. Then in Micro c create a new Project and program it as given.
3. Then run the program it will create a .HAX file.
4. Put the Hex File in Micro controller And Run it.
5. Observe the Steeper Motor Rotation.

**Discussion:**

**Experiment No-06:** Familiarizing with Arduino, Arduino IDE & design a simple LED looping Circuit.

**Objective:** To Familiarize with Arduino, Arduino IDE & design a simple LED looping Circuit.

1. **Arduino**
2. **Arduino IDE**
3. **Design a simple LED Looping Project**

**Description:**

**What is a Development Board:**

A printed circuit board designed to facilitate work with a particular microcontroller

**Typical components include:**

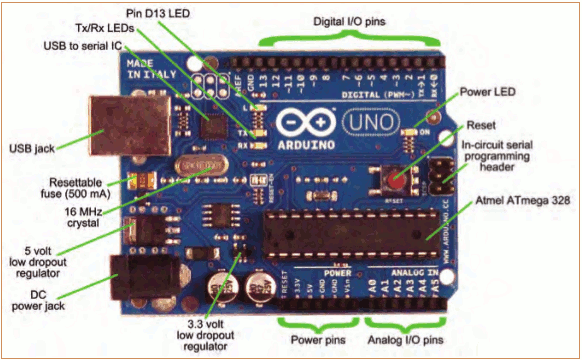
* + power circuit
  + programming interface
  + basic input; usually buttons and LEDs
  + I/O pins

**What is the Arduino:**

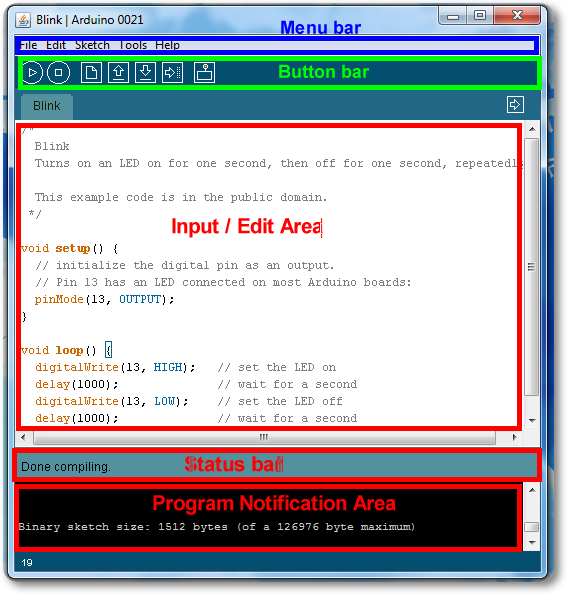
**Arduino** is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.

* Great for prototyping ideas
* Access to multiple I/O
* Drive motors, turn on lights, trigger controls.
* Low Power requirements
* Flexible / Open-source

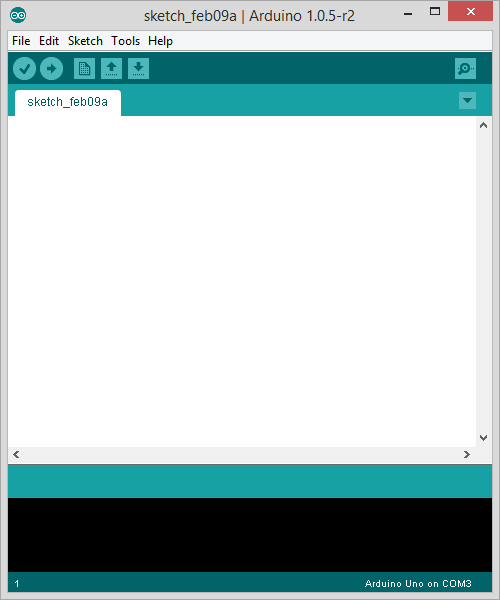
**Arduino Development Board:**



**Introduction to Arduino IDE:**

****

**Parts of the IDE main screen:**



Text area for

writing/editing

sketches.

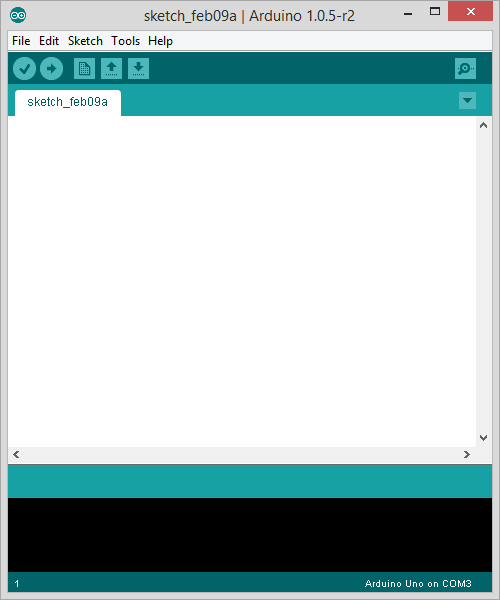
Error messages and other

feedback show up here.

Name of current sketch

Main menus

Action buttons/icons



Verify (AKA compile)

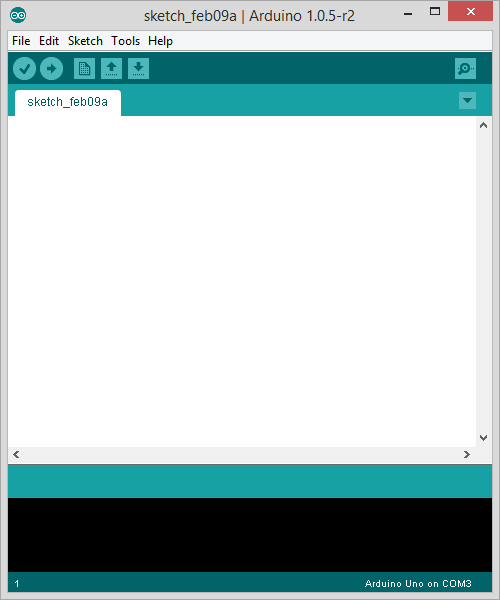
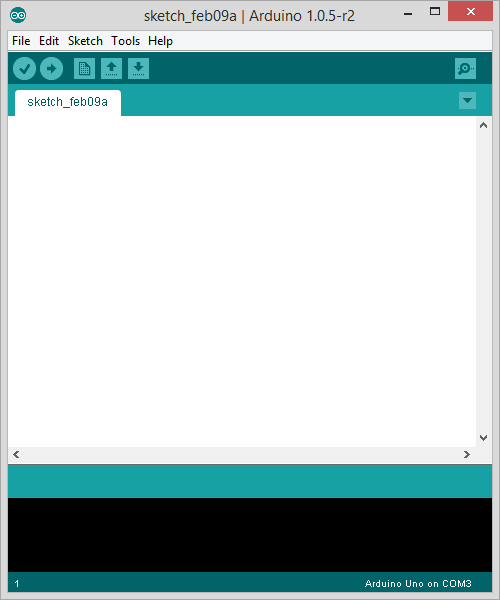
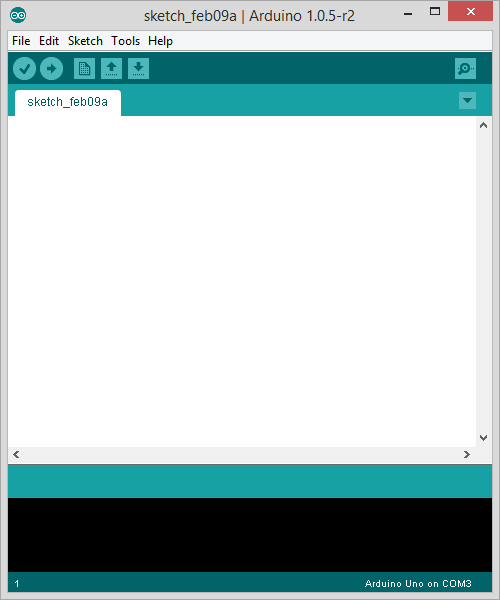
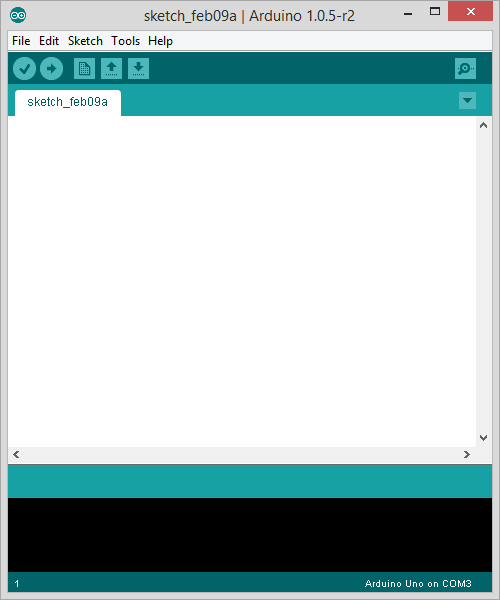
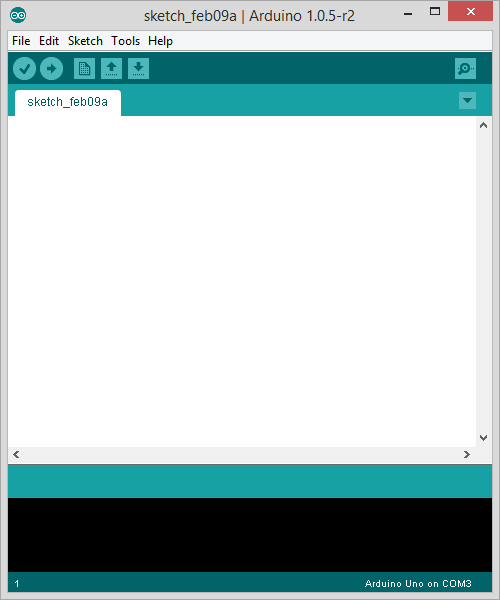
Upload (send to Arduino)

Start a new sketch

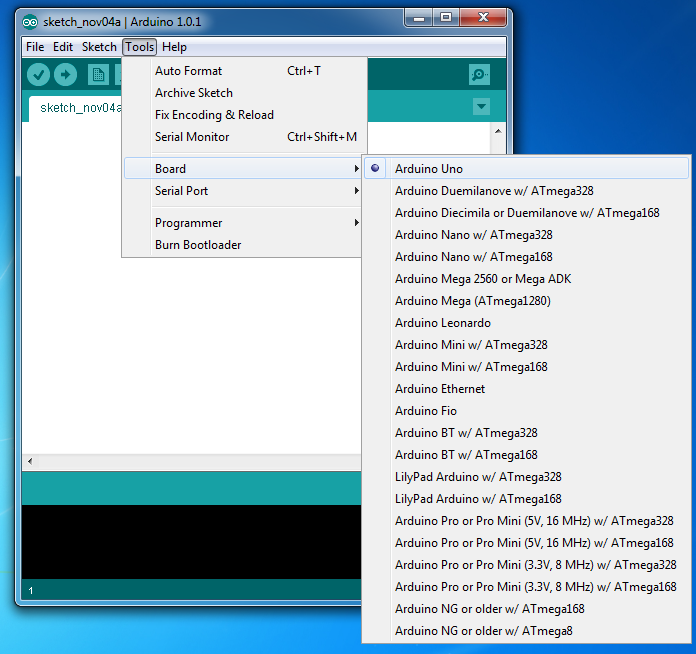
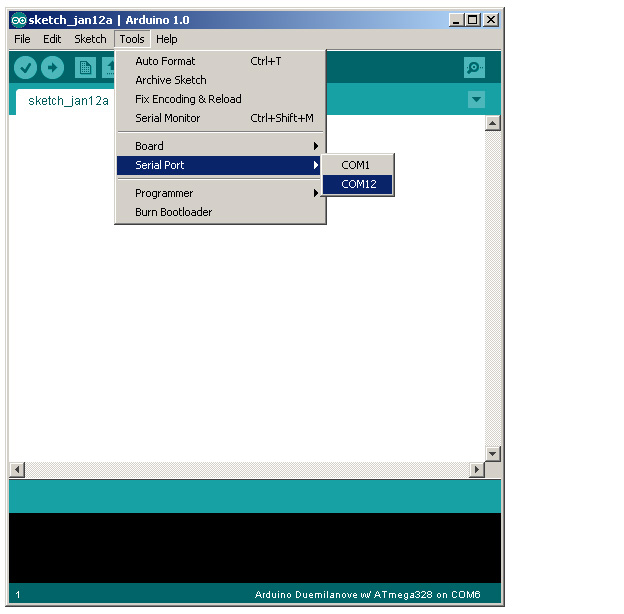
Open a sketch (from a file)

Save current sketch (to a file)

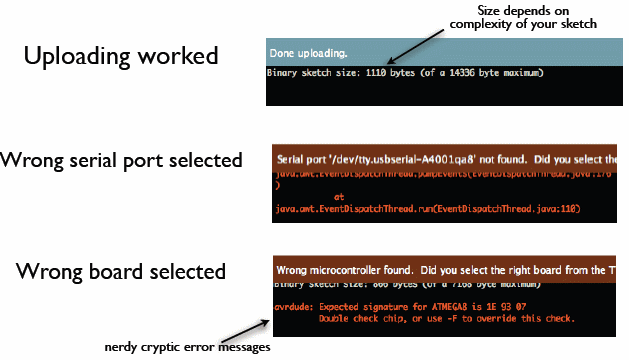
Open Serial Monitor window



**Select Serial Port and Board:**

****

**Status Messages:**

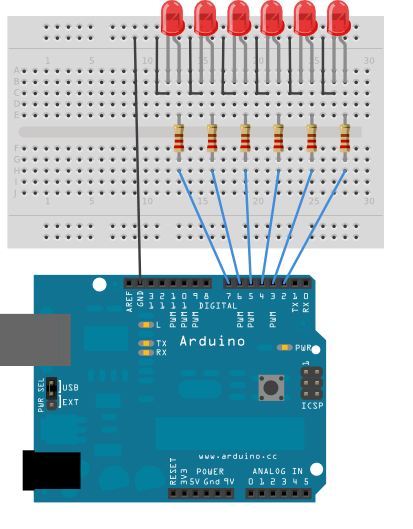


**LED Looping Project:**

**Instrument Required:**

1. Arduino Uno
2. LED
3. Bread Board
4. Connecting Wire
5. Power Supply

**Circuit Diagram :**

****

**Source Code:**

int timer = 100; // The higher the number, the slower the timing.

void setup() {

// use a for loop to initialize each pin as an output:

for (int thisPin = 2; thisPin < 8; thisPin++) {

pinMode(thisPin, OUTPUT);

}

}

void loop() {

// loop from the lowest pin to the highest:

for (int thisPin = 2; thisPin < 8; thisPin++) {

// turn the pin on:

digitalWrite(thisPin, HIGH);

delay(timer);

// turn the pin off:

digitalWrite(thisPin, LOW);

}

// loop from the highest pin to the lowest:

for (int thisPin = 7; thisPin >= 2; thisPin--) {

// turn the pin on:

digitalWrite(thisPin, HIGH);

delay(timer);

// turn the pin off:

digitalWrite(thisPin, LOW);

}

}

**Discussion:**

**Experiment No 07:** PWM & its use in Arduino using Servo Motor.

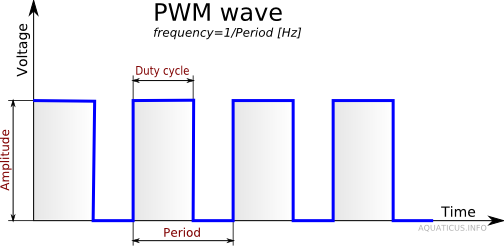
**Objective:** To learn about PWM and use it for driving a Servo Motor.

**Instrument Required:**

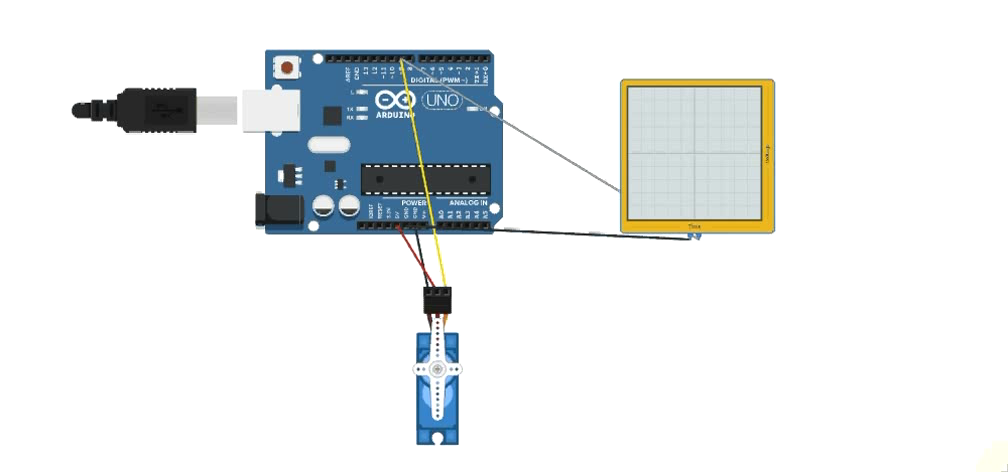
1. Arduino Uno
2. Servo Motor
3. Potentiometer
4. Resistor
5. Connecting wire
6. Power supply

**What is PWM & Its Uses:**

* + Pulse Width Modulation
* Why need PWM?
  + Digital devices can't produce any other voltage levels as outputs except 0s and 1s (let's say 0V and 5V). By varying the duty cycle of the pulses and their frequency we could easily mimic the intermediate voltage levels.
* Where we use PWM?
  + DC Motor speed control, servo motor , LED dimming, alternative of A potentiometer .

****

**Circuit Diagram:**

****

**Source Code:**

#include <Servo.h>

Servo myservo;

int apin = A2;

int mot\_pin = 9;

void setup() {

Serial.begin(9600);

pinMode(apin, INPUT);

myservo.attach(mot\_pin);

}

void loop() {

int v = analogRead(apin);

int d = map(v, 0, 1023, 0, 180);

myservo.write(d);

Serial.println(d);

delay(50);

}

**Discussion :**

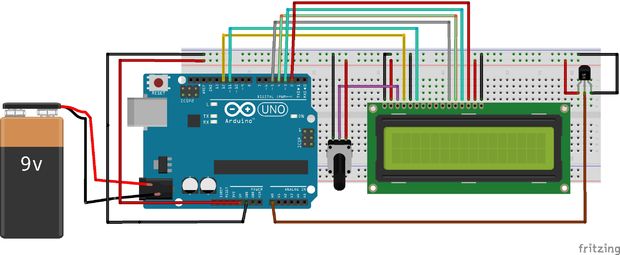
**Experiment No 08:** Design of an Arduino based temperature Meter using LM 35 & LCD display.

**Objective:** To design of an Arduino based temperature Meter using LM 35 & LCD display.

**Instrument Required:**

1. Arduino Uno
2. Potentiometer
3. LCD 16x2
4. LM 35 Temperature Sensor
5. Resistor
6. Connecting wire
7. Power supply

**Circuit Diagram:**

****

**Source Code:**

int outputpin= 0;

//this sets the ground pin to LOW and the input voltage pin to high

void setup()

{

Serial.begin(9600);

}

//main loop

void loop()

{

int rawvoltage= analogRead(outputpin);

float millivolts= (rawvoltage/1024.0) \* 5000;

float celsius= millivolts/10;

Serial.print(celsius);

Serial.print(" degrees Celsius, ");

Serial.print((celsius \* 9)/5 + 32);

Serial.println(" degrees Fahrenheit");

delay(1000);

}

**Discussion:**

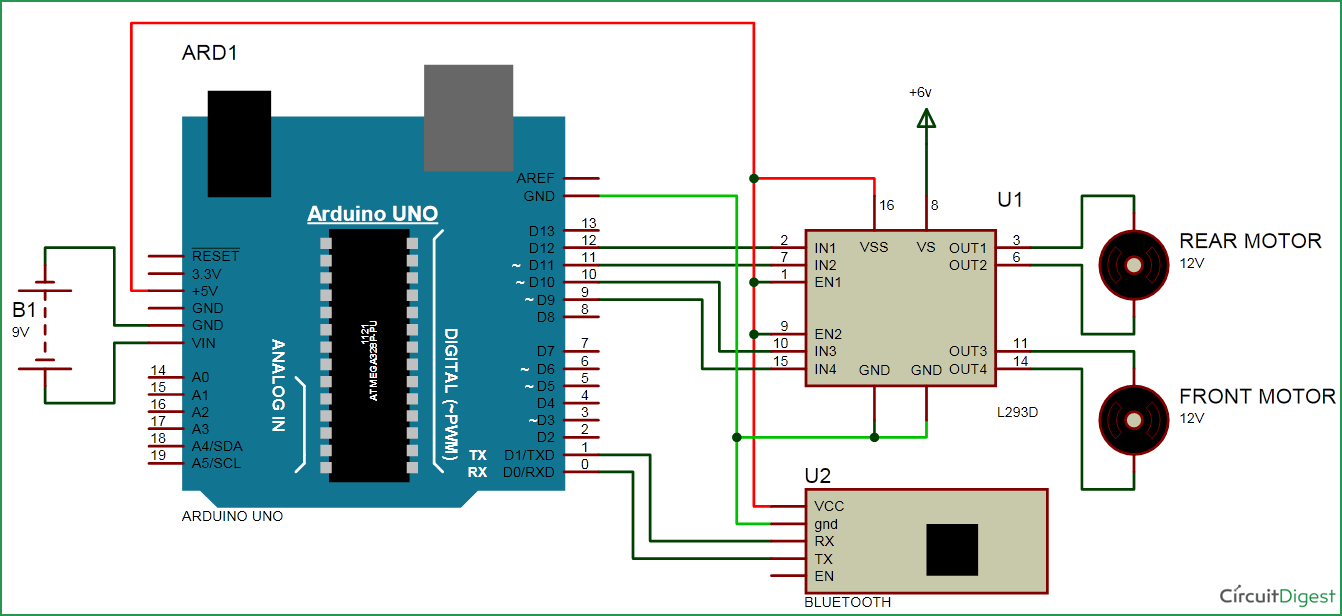
**Experiment No 09:** Design of a mobile control robotic car using Arduino, Bluetooth Module & Android Phone.

**Objective:** To Design of a mobile control robotic car using Arduino, Bluetooth Module & Android Phone.

**Instrument Required:**

1. Arduino Uno
2. L298 Motor Shield
3. Car Cassis With DC Motor
4. HC-05/06 Bluetooth Module
5. Android Phone
6. Resistor
7. Connecting wire
8. Power supply

**Circuit Diagram:**



**Source Code:**

#include <Servo.h>

int incomingByte = 0;

int A = 12;

int B = 11;

int C = 10;

int D = 9;

void setup()

{

pinMode(A, OUTPUT);

pinMode(B, OUTPUT);

pinMode(C, OUTPUT);

pinMode(D, OUTPUT);

Serial.begin(38400); // for HC-05 Serial rate 38400; and for HC-06 it is 9600.

}

void loop()

{

if (Serial.available() > 0)

{

// read the incoming byte:

incomingByte = Serial.read();

Serial.println(incomingByte);

// delay 10 milliseconds to allow serial update time

delay(10);

}

//if byte is equal to "70" or "F",every motor start at neutral

if(incomingByte==70)

{

digitalWrite(A, HIGH);

digitalWrite(B, LOW);

digitalWrite(C, HIGH);

digitalWrite(D, LOW);

}

else

{

if(incomingByte==83)

{

digitalWrite(A, LOW);

digitalWrite(B, LOW);

digitalWrite(C, LOW);

digitalWrite(D, LOW);

}

}

if(incomingByte==66)

{

digitalWrite(A, LOW);

digitalWrite(B, HIGH);

digitalWrite(C, LOW);

digitalWrite(D, HIGH);

}

else

{

if(incomingByte==83)

{

digitalWrite(A, LOW);

digitalWrite(B, LOW);

digitalWrite(C, LOW);

digitalWrite(D, LOW);

}

}

if(incomingByte==76)

{

digitalWrite(A, LOW);

digitalWrite(B, HIGH);

digitalWrite(C, HIGH);

digitalWrite(D, LOW);

}

else

{

if(incomingByte==83)

{

digitalWrite(A, LOW);

digitalWrite(B, LOW);

digitalWrite(C, LOW);

digitalWrite(D, LOW);

}

}

if(incomingByte==82)

{

digitalWrite(A, HIGH);

digitalWrite(B, LOW);

digitalWrite(C, LOW);

digitalWrite(D, HIGH);

}

else

{

if(incomingByte==83)

{

digitalWrite(A, LOW);

digitalWrite(B, LOW);

digitalWrite(C, LOW);

digitalWrite(D, LOW);

}

}

}

**Discussion :**

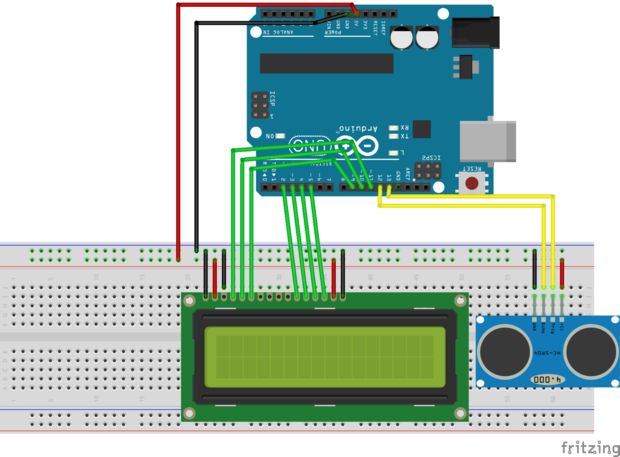
**Experiment No 10:** Sonar sensor based distance calculator.

**Objective:** To Sonar sensor based distance calculator.

**Instrument Required:**

1. Arduino Uno
2. LCD 16x2
3. Sonar HC-SR04
4. Potentiometer
5. Resistor
6. Connecting wire
7. Power supply

**Circuit Diagram:**



**Source Code:**

#include <LiquidCrystal.h> //Load Liquid Crystal Library

LiquidCrystal LCD(11,10,9,2,3,4,5); //Create Liquid Crystal Object called LCD

#define trigPin 13 //Sensor Echo pin connected to Arduino pin 13

#define echoPin 12 //Sensor Trip pin connected to Arduino pin 12

//Simple program just for testing the HC-SR04 Ultrasonic Sensor with LCD dispaly

//URL:

void setup()

{

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

LCD.begin(16,2); //Tell Arduino to start your 16 column 2 row LCD

LCD.setCursor(0,0); //Set LCD cursor to upper left corner, column 0, row 0

LCD.print("Target Distance:"); //Print Message on First Row

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

LCD.setCursor(0,1); //Set cursor to first column of second row

LCD.print(" "); //Print blanks to clear the row

LCD.setCursor(0,1); //Set Cursor again to first column of second row

LCD.print(distance); //Print measured distance

LCD.print(" cm"); //Print your units.

delay(250); //pause to let things settle

Serial.print("Distance: ");

Serial.println(distance);

}

**Discussion :**

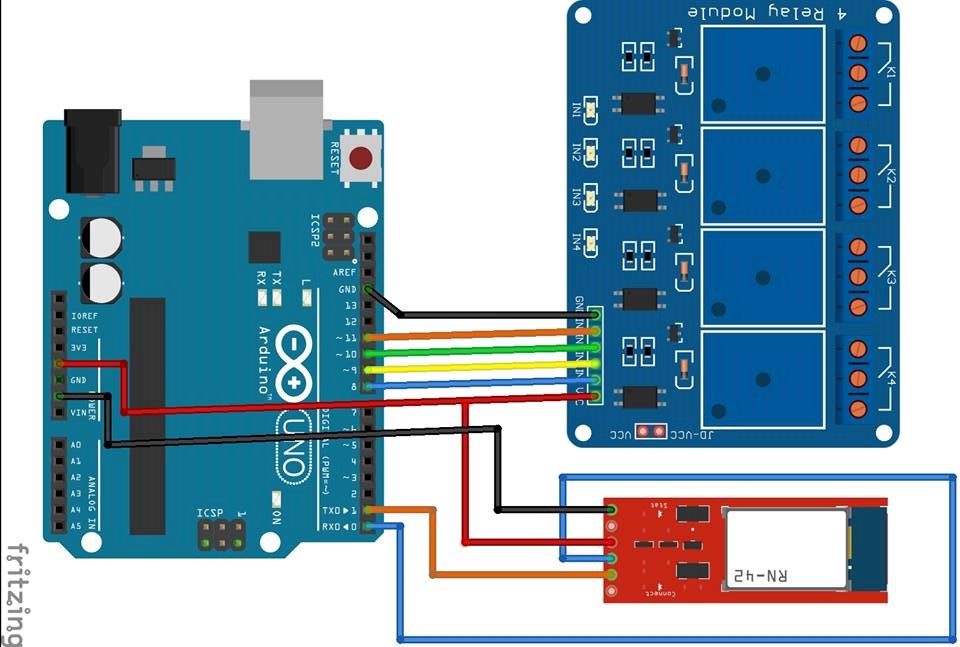
**Experiment No 11:** Bluetooth based Home Automation System.

**Objective:** To design of a Bluetooth based Home Automation System.

**Instrument Required:**

1. Arduino Uno
2. HC -05/HC-06 Bluetooth Module
3. Relay Module
4. Resistor
5. Connecting wire
6. Power supply

**Circuit Diagram:**

**­­**

**Source Code:**

char val;

int ledpin1 = 11;

int ledpin2 = 10;

int ledpin3 = 9;

int ledpin4 = 8;

void setup()

{

pinMode(ledpin1, OUTPUT);

pinMode(ledpin2, OUTPUT);

pinMode(ledpin3, OUTPUT);

pinMode(ledpin4, OUTPUT);

Serial.begin(9600);

}

void loop()

{

if( Serial.available() )

{

val = Serial.read();

Serial.println(val);

// delay 10 milliseconds to allow serial update time

delay(10);

}

if( val == '1' )

{

digitalWrite(ledpin1 , HIGH);

}

if( val == 'A' )

{

digitalWrite(ledpin1 , LOW);

}

if( val == '2' )

{

digitalWrite(ledpin2, HIGH);

}

if( val == 'B' )

{

digitalWrite(ledpin2, LOW);

}

if( val == '3' )

{

digitalWrite(ledpin3, HIGH);

}

if( val == 'C' )

{

digitalWrite(ledpin3, LOW);

}

if( val == '4' )

{

digitalWrite(ledpin4, HIGH);

}

if( val == 'D' )

{

digitalWrite(ledpin4, LOW);

}

if( val == '9' )

{

digitalWrite(ledpin4, HIGH);

digitalWrite(ledpin3, HIGH);

digitalWrite(ledpin2, HIGH);

digitalWrite(ledpin1, HIGH);

}

if( val == 'I' )

{

digitalWrite(ledpin4, LOW);

digitalWrite(ledpin3, LOW);

digitalWrite(ledpin2, LOW);

digitalWrite(ledpin1, LOW);

}

}

**Discussion :**

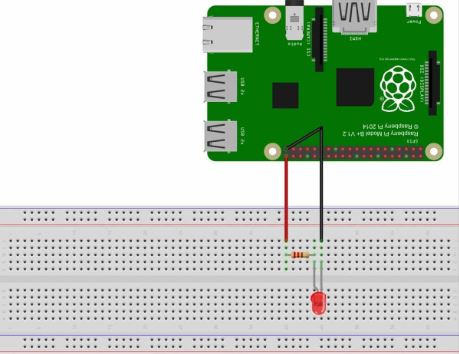
**Experiment No 12:** To blink a led using Raspberry Pi and Python.

**Objective:** To Design of a LED blinking circuit using Raspberry Pi 3 with the help of Python Language.

**Instrument Required:**

1. Raspberry Pi 3
2. LED
3. 220Ω/1KΩ
4. Connecting wire

**Circuit Diagram:**



**Source Code:**

import RPi.GPIO as IO

import time

IO.setmode (IO.BOARD)

PIN40  
IO.setup(40,IO.OUT)

IO.output(40,1)

time.sleep(1)

IO.cleanup()

time.sleep(1)

IO.setmode (IO.BOARD)  
IO.setup(40,IO.OUT)  
IO.output(40,1)  
time.sleep(1)  
IO.cleanup()  
time.sleep(1)

IO.setmode (IO.BOARD)  
IO.setup(40,IO.OUT)  
IO.output(40,1)  
time.sleep(1)  
IO.cleanup()  
time.sleep(1)

**Discussion:**