Title:- Design & Implementation of Light control - Blinking a LED using arduino Micro controller.

Objectives: In this experiment we'll be trying to blink a LED with differente time difference. Just Introducing ourselves with how arduino works & how to code one of the most popular microcontroller.

Components:-
(i) Arduino Uno Microcontroller
(ii) Bread Board
(iii) 220 Ω resistors
(iv) Push button
(v) LED
(vi) connecting wires.

procedure:- First we connect the Led & the push button to the the arduino as shown in the diagram below- Then using the given usb cable we connect the arduino to a Laptop & upload the given code. The push button control the blink of the LED. Depending on the push button state the LED blinks at a different speed.

Diagram For experiment 1:-

code for the experiment 1:-

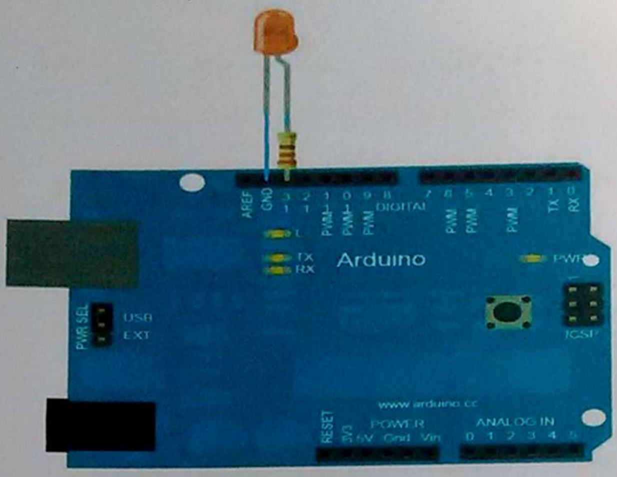


Figure 1: Blink The LED

```
// the setup function runs once when you press reset or power the board
void setup() {
      //initialize digital pin LED_BUILTIN as an output.
      pinMode(LED_BUILTIN, OUTPUT);
}

//the loop function runs over and over again forever
void loop() {
      digitalWrite(LED_BUILTIN, HIGH);
      // turn the LED on (HIGH is the voltage level)
       delay(1000); // wait for a second
      digitalWrite(LED_BUILTIN, LOW);
      // turn the LED off by making the voltage LOW
      delay(1000); // wait for a second
}
```

Result:- From this experiment we learnt how the push switches interact with the arduino microcontroller The push switches interrupt the digital signal coming from the digital I/O pin.

Digital I/O pins output is equivalent to digital signal 1 or 0, depending on the output voltage. The LED Flashes when it receives +5 v from the digital output pin.

Conclusion: We used the 220 ohm resistors to avoid over voltage surge, which may damage the LED. We must double check the connections before powering up the arduino.

Experiment No:-2

Experiment Name:- Design & Implementation of light control using a sensor device

Objective:- In this experiment we're going to test potentiometer as a sensor, controlling the brightness of a LED or a set of LED. from the sensor value.

Components:- 1): Arduino Uno Microcontroller

2) BreadBoard

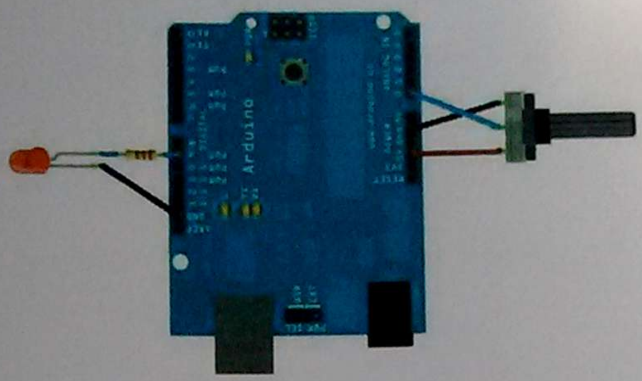
3) Potentiometer

4) LED

5) connecting wire.

Procedure:- We need to connect the components as shown in the diagram below. Then we connect the microcontroller board to a laptop to upload the code. Before uploading the code, we make sure there is no wrong wire connection. The potentiometer value is mapped in the range 0 to 255. Depending on the potentiometer input value the microcontroller mapped value is output to a digital pin connected to a LED, which fade the brightness of the LED.

Experiment 2 – diagram & code



```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  delay(2);
}
```

Result:- The potentiometer is nothing but a variable registor, if we turn the knob of the potenttiometer, it will change its resistance, thus its mapped value changes and it fade in or fade out the Led.

Conclusion :- In this experiment we worked with analog sensor device. And using this we tested the analog value mapping system.

Experiment - 3 :-

Experiment Name:- Programming a LCD Display with Arzduino Microcontroller

Objective :- In this experiment we're going to test interfacing a display module with arduino microcontroller. An LCD display of size 4×20 or 2×20 has 4 or 2 rows & 20 columns. Each segment of each row & each column can print an ASCII charaacter. It works as simple as follows-

Working principle :-

0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
1	1	1	1	1	1	1	1

bitwise representation for ASCII charaacter → 1

Each segment has a 8×8 matrix of pixels. For each of ASCII character it mapped some pixels to be turned off & some to be turned on.

Turned off pixels has no brightness on the other hand turned on pixels has full brightness & contrast. In Arduino total 4 pins ~~is~~ are used as data pins for the LCD module.

Components:-

i) Arduino Micro controller

ii) LCD display module

iii) Connecting wire

iv) Variable register.

v) Bread Board

Procedure: We need to connect the LCD module & the microcontrollers unit as shown in the figure.

The LCD has some extra pins to be connected for enabling backlight, brightness, chip select. etc.

The wires should be configured correctly.

After double checking the wire connection we connect the arduino module to a computer ~~del~~ device & upload the given code which is a simple 'hello world' program.

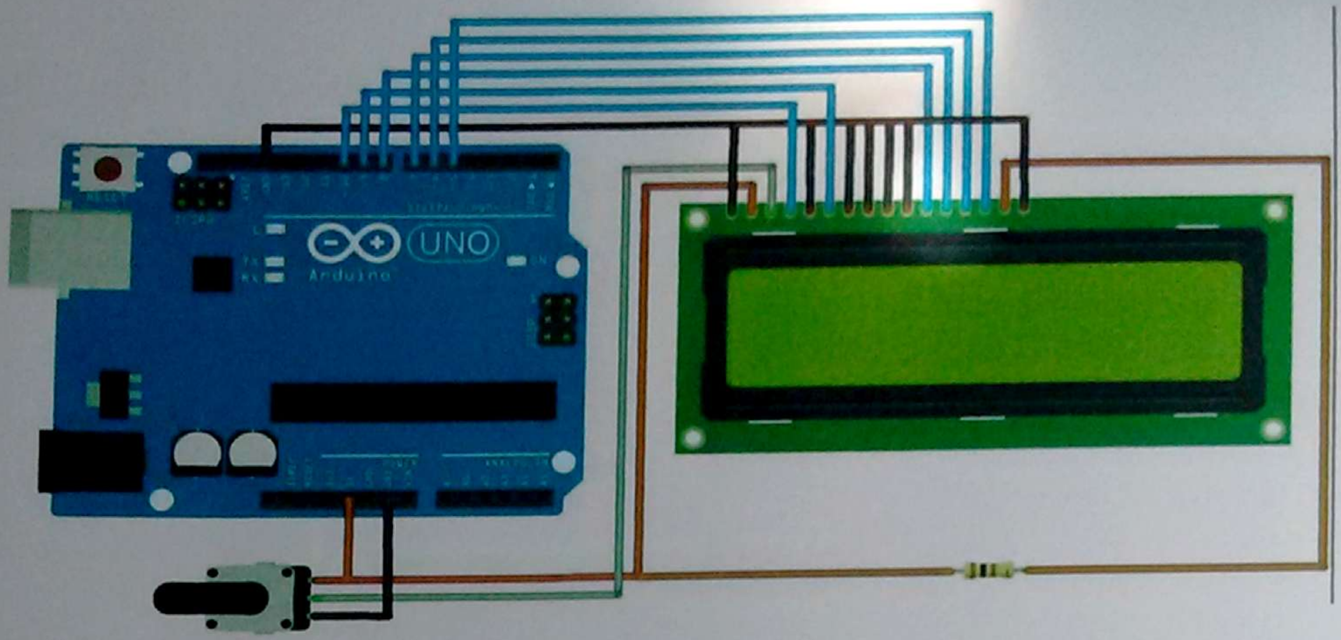


Figure 1: LCD interfacing

```
// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
     // set up the LCD's number of columns and rows:   lcd.begin(16,2);
     // Print a message to the LCD.
     lcd.print("hello, world!");
}
void loop() {
      // set the cursor to column 0, line 1
      // (note: line 1 is the second row, since counting begins with 0):
      lcd.setCursor(0, 1);
     // print the number of seconds since reset:
     lcd.print(millis() / 1000);
}
```

Result & Conclusion :-

There are several models available for LCD module. Wiring & interfacing techniques are different for each depending on chip model & version.

The LCD module print garbage character if it is not clear on first use using lcd.clear();

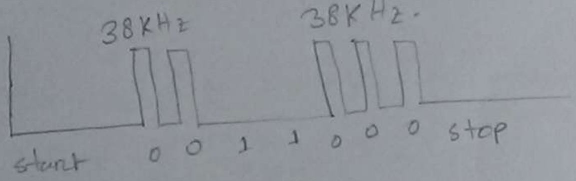
There are two extra pins need to be connected for those models which have backlight support.

Experiment - 4

Experiment Name:- Remote controlling a DC Motor using Arduino Microcontroller & IR sensor.

Objective & Hypothesis: This experiment will allow us to inteface an IR sensor and a DC motor with an arduino module And controll the motor using a IR Remote.

The IR remote oscilate & produce light blink at 38KHz frequency.



The sensor senses the light blinks and generates some voltage surge of voltage drop across the arduino digital pin which is then converted to hexadecimal codes for. each button on the Remote module.

Depending on the buttons pressed on the remote Module the arduino module reads data and controll the motor speed or on & off state.

Components :- ① Arduino Module (Uno)

② Remote control

③ IR sensor

④ Bread Board

⑤ Connecting wires

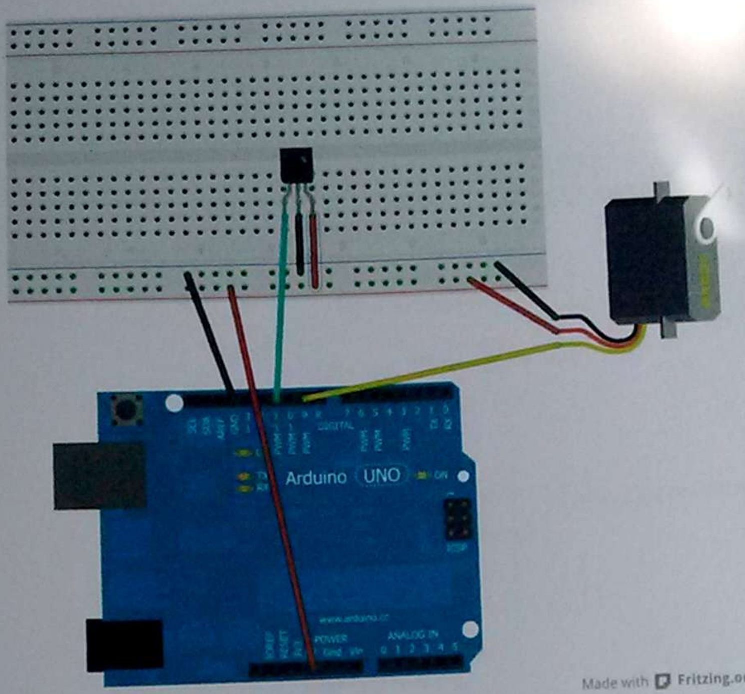
⑥ Motor (DC)

⑦ Motor shield.

Procedure : we connect the devices as shown in the diagram below. We double check the wired connection & connet the arduino module to the host computer to compile & upload the source code to its bootloader / NAND Flash memory.

We need to calibrate remote controll buttons first before using it as a controller. we need to get the hex values for which the motor motion & switching will work.

We'll make sure that there is no short circuit occured which may damage the microcontroller for ever



Made with Fritzing.org

Figure 1: Remote Controlling Motor

```
#include<IRremote.h>
#include <Servo.h>
Servo myservo;
IRrecv irrecv(6);
String button;
decode_results results;
int pos,j;
char c;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
  myservo.attach(9);
  pos=1;
  myservo.write(0);  // set servo to mid-point
}
```

Experiment CirCut & code

```
void loop(){
  if(irrecv.decode(&results)){
    button=String(results.value,HEX);
     if(button=="f0771735" || button=="40bd08f7"){
           Serial.println(button);
           for(j=pos;j<=pos+10;j++){
                     myservo.write(j);
                }
                if(pos<=170){
                     pos+=10;
                }
           }
     else if(button=="40bd8877" || button=="80eb69e1"){
           Serial.println(button);
           for(j=pos;j>=pos-10;j--){
                     myservo.write(j);
                }
                if(pos>=10){
                     pos-=10;
                }
           }
     irrecv.resume();
     delay(500);
     }
  }
```

Experiment -#5

Experiment Name:- DC Motor controll based on temparature

Hypothesis/Working principle:- In this experiment we tested interfacing techniques of ~~how~~ a temparature censor LM35. The LM35series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the centrigrade temperature. The LM 35 device has an advantage over linear lemperature sensors calibrated in kelvin scale.

The main features of LM 35 are - calibrated directly in celcius scale, linear +10mV/°C scale Factor. 0.5°C ensured Accuracy. Rated for full -55°C to +150°C Range. Operates from 4V to 30V. Less than 60μA current drain. Low self heating, 0.08°c in still air.

Components:- (I) Arduino Uno Module

(II) Motor driven L293D

(III) 7806 regulator IC

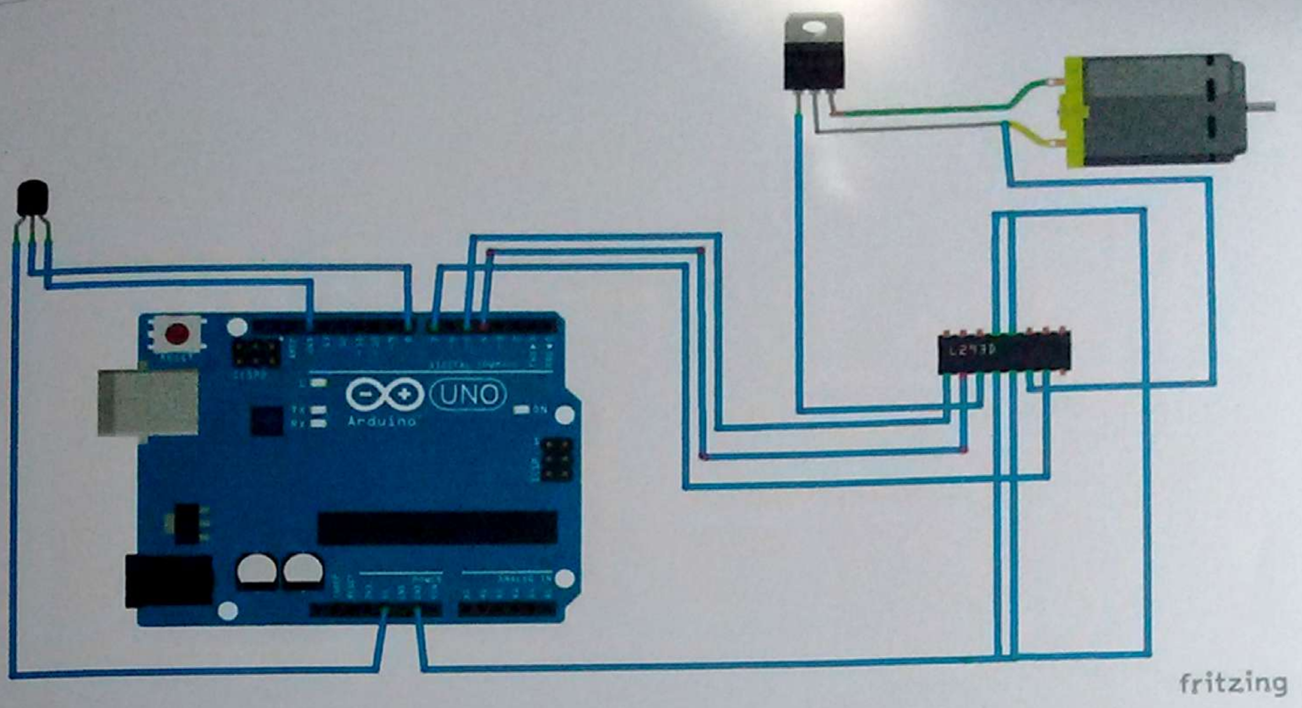
(IV) Bread Board

(V) Jumper wires

procedure:- we first connect the different parts as shown in the diagram below. The important module here is the L293D Motor driver. It has two motor driving capability. It can change its poles for the motors that causes the motors spin in the both clock wise & anti clock wise direction. If the motor requires motors requires a large amount of power to run, then it is better to use a independent power supply only for the motor driver. The motor driver reads the digital data from the arduino & changes the power supply +ve & -ve direction to change the spinning direction of the motors. The direction changing is done by the H bridge circuit intigrated in the L293D IC.

The LM-3M temperature sensor runs at 4V. & upto 30 v. It requires a few mili amps of current so that we can power it up yust using the arduino +5 V & GND supplies.

Figure & code



```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;

void setup()
{
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(switchPin, INPUT_PULLUP);
}

void loop()
{
  int speed = analogRead(potPin) / 4;
  boolean reverse = digitalRead(switchPin);
  setMotor(speed, reverse);
}

void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin,! reverse);
  digitalWrite(in2Pin, reverse);
}
```

Result & Conclusion :-

There are severals models availables for LCD module. Wiring & interfacing techniques are different for each, depending on chip model & version

The LCD module print garbage characters if it is not clear on first use using LCD:: clear();

There are two extra pins need to be connected for those models which have backlight support