BAPATLA ENGINEERING COLLEGE

(AUTONOMOUS)

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION

INTERNET OF THINGS LAB

18ECL62

SUBMITTED BY

KORLA.RAVI TEJA

Y18AEC465

LIST OF EXPERIMENTS

1) Write an Arduino program to interface and to perform the LED blinking with 1 second delay using Arduino uno.
2) Write an Arduino program to interface and to perform the Analog Read Serial using Arduino uno.
3) Write an Arduino program to interface and to perform the Digital Read Serial using Arduino uno.
4) Write an Arduino program to interface and to perform the fade program for a led using Arduino uno.
5) Write an Arduino program to interface and to perform the Serial Read using Arduino uno.
6) Write an Arduino program to interface and to perform the keypad operation using Arduino uno.
7) Write an Arduino program to interface and to perform the IR SENSOR operation using Arduino uno.
8) Write an Arduino program to interface and to perform the LCD operation using Arduino uno.
9) Write a Arduino program to interface and to perform the ultrasonic sensor operation using Arduino uno.
10) Write a Raspberry pi program to interface and to perform the BLINK OPERATION using Raspberry pi board.

1) Write an Arduino program to interface and to perform the LED blinking with 1 second delay using Arduino uno

AIM: To Write an Arduino Program to Interface and To Perform the LED Blinking

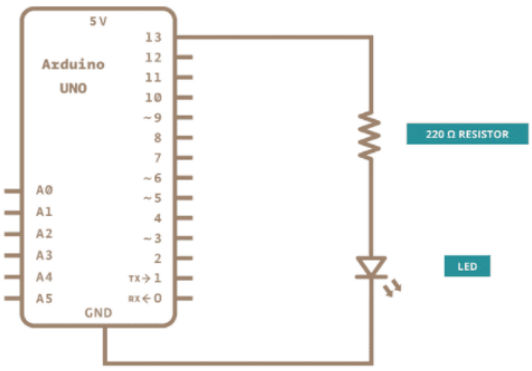
SOFTWARE USED: Arduino IDE

THEORY:

The LED_BUILTIN pin back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay () commands tell the board to do nothing for 1000 milliseconds, or one second.
Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to the correct LED pin independent of which board is used.

SOURCE CODE:

```
void setup () {
  // initialize digital pin LED_BUILTIN as an output.
  Pin Mode (LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop () {
digital Write (LED_BUILTIN, HIGH);          // turn the LED on
delay (1000);                                // wait for a second
digital Write (LED_BUILTIN, LOW);           // turn the LED off
delay (1000);                               // wait for a second
```



2) Write an Arduino program to interface and to perform the Analog Read Serial using Arduino uno

AIM: To Write an Arduino program to interface and to perform the Analog Read Serial

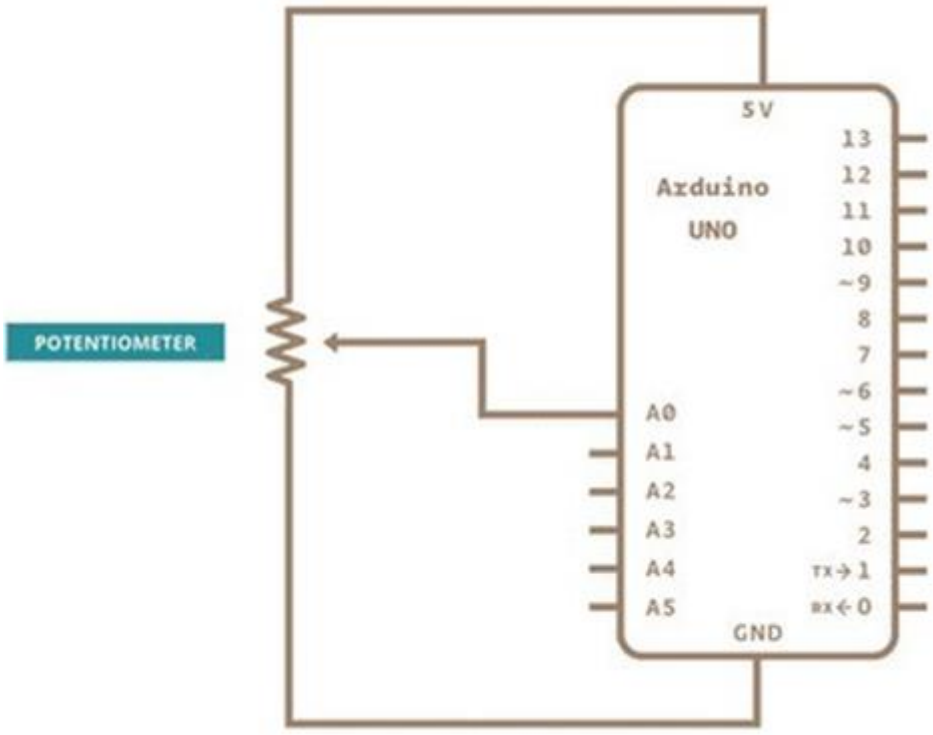
SOFTWARE USED: Arduino IDE

THEORY:

Reads an analog input on pin 0, prints the result to the Serial Monitor.

Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

The Arduino boards have a circuit inside called an analog-to-digital converter *or ADC* that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023.



SOURCE CODE:

```
void setup()
// initialize serial communication at 9600 bits per
Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
// read the input on analog pin 0:
int sensorValue = analogRead(AO);
// print out the value you read:
Serial.println(sensorValue);
delay(1);
// delay in between reads for stability
}
```

3) **Write an Arduino program to interface and to perform the Digital Read Serial using Arduino uno**

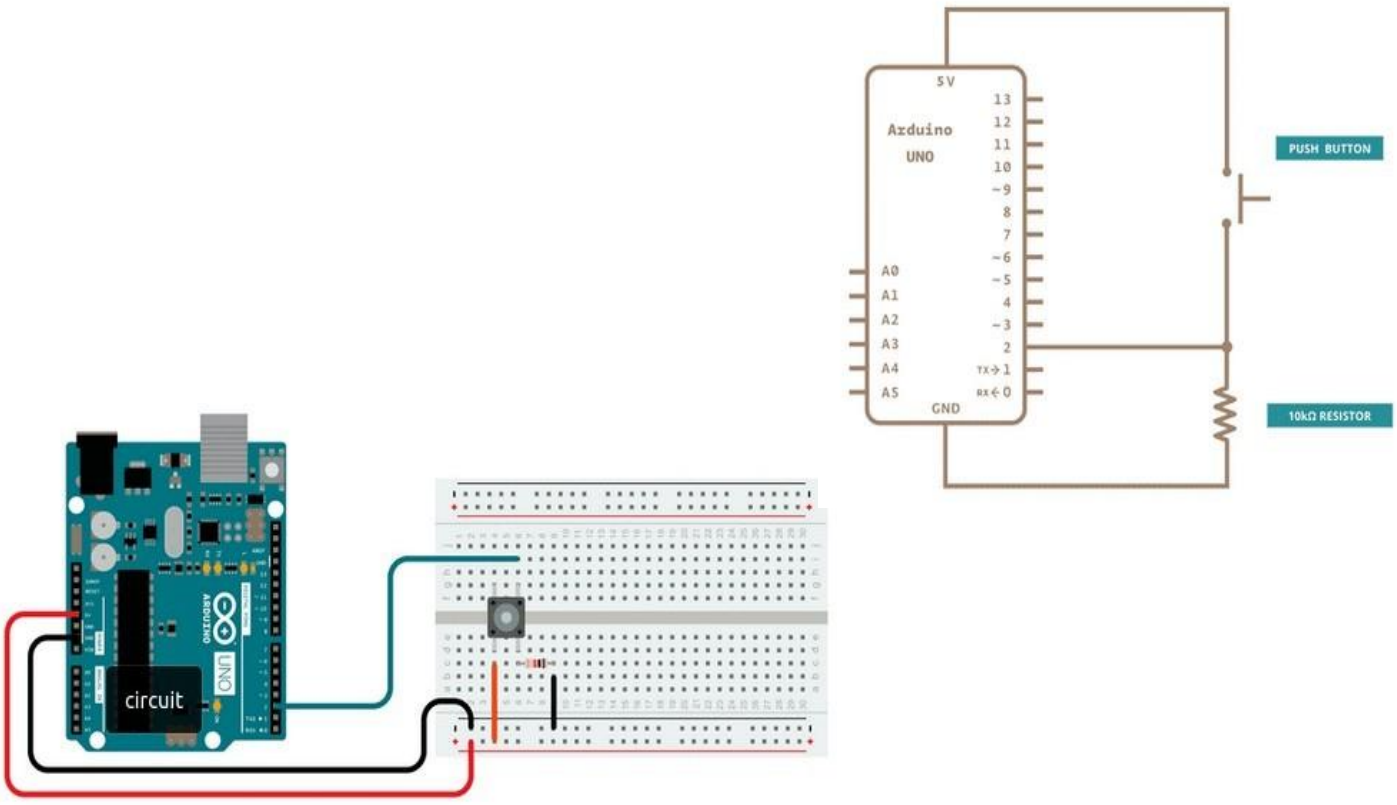
AIM: To Write an Arduino program to interface and to perform the Digital Read Serial

SOFTWARE USED: Arduino IDE

THEORY:

The very first thing that you do will in the setup function is to begin serial communications, at 9600 bits of data per second, between your board and your computer with the line. Next, initialize digital pin 2.

Now that your setup has been completed, move into the main loop of your code. When your button is pressed, 5 volts will freely flow through your circuit, and when it is not pressed, the input pin will be connected to ground through the 10k ohm resistor. This is a digital input, meaning that the switch can only be in either an on state (seen by your Arduino as a "1", or HIGH) or an off state (seen by your Arduino as a "0", or LOW), with nothing in between.



SOURCE CODE:

```
// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushbutton = 2;
// the setup routine runs once when you press reset:
void setup() {
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
// make the pushbutton's pin an input:
pin Mode(pushButton, INPUT);
}
// the loop routine runs over and over again forever:
void loop() {
// read the input pin:
int buttonState = digitalRead(pushbutton);
Serial.println(buttonState);          // print out the state of the button:
delay(1);                             // delay in between reads for stability
```

4) **Write an Arduino program to interface and to perform the fade program for a led using Arduino uno**

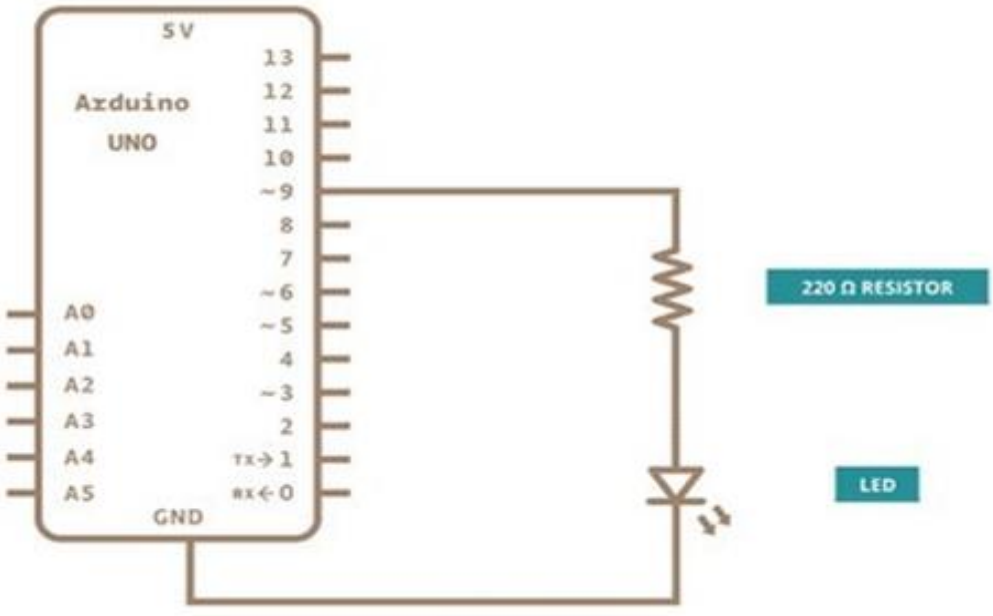
AIM: To Write an Arduino program to interface and to perform the Fade Program.

SOFTWARE USED: Arduino IDE

THEORY:

The analogWrite() function that you will be using in the main loop of your code requires two arguments: One telling the function which pin to write to, and one indicating what pwm value to write.

In order to fade your LED off and on, gradually increase your PWM value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle. In the sketch below, the PWM value is set using a variable called brightness. Each time through the loop, it increases by the value of the variable fadeAmount .If brightness is at either extreme of its value (either 0 or 255), then fadeAmount is changed to its negative. In other words, if fadeAmount is 5, then it is set to -5. If it's -5, then it's set to 5. The next time through the loop, this change causes brightness to change direction as well. analogWrite() can change the PWM value very fast, so the delay at the end of the sketch controls the speed of the fade. Try changing the value of the delay and see how it changes the fading effect.



SOURCE CODE:

```
int led = 9;
int brightness = 0;
int fade Amount = 5;
// the PWM pin the LED is at:
// how bright the LED is:
// how many points to fade the led:
void setup() {
// declare pin 9 to be an output:
pinMode(led, OUTPUT);
}
void loop() {
// set the brightness of pin 9:
analogWrite(led, brightness);
// change the brightness for next time through the loop:
brightness = brightness + fadeAmount;
// reverse the direction of the fading at the ends of the fade:
if (brightness <= 0 || brightness >= 255) {
fade Amount = -fade Amount;
}
// wait for 30 milliseconds to see the dimming effect
delay(30);
}
```

5) **Write an Arduino program to interface and to perform the Serial Read using Arduino uno**

AIM: To Write an Arduino program to interface and to perform the Serial Read.

SOFTWARE USED: Arduino IDE

THEORY:

Reads incoming serial data.

`Serial.read()` inherits from the stream utility class.

`Serial`: serial port object.

The first byte of incoming serial data available (or -1 if no data is available). Data type: `int`.

SOURCE CODE:

```
int incomingByte = 0; // for incoming serial data
void setup() {
Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}
void loop() {
// reply only when you receive data:
if (Serial.available() > 0) {
// read the incoming byte:
incomingByte = Serial.read();
// say what you got:
Serial.print("I received: ");
Serial.println(incomingByte, HEX);
}
}
```

6) **Write an Arduino program to interface and to perform the keypad operation using Arduino uno**

AIM: To Write an Arduino program to interface and to perform the keypad operation.

SOFTWARE USED: Arduino IDE

THEORY:

Keypad is used as an input device to read the key pressed by the user and to process it.4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns.A key press establishes a connection between the corresponding row and column, between which the switch is placed.

Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way – one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad

byte rowPins[ROWS] = {R1, R2, R3, R4};

/* connect to the row pinouts of the keypad */

byte colPins[COLS] = {C1, C2, C3, C4};

/* connect to the column pinouts of the keypad */

If you do not connect the pins according to this function, the key press identification will not give results according to the keypad you have defined.

SOURCE CODE:

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the symbols on the buttons of the keypads
char hexaKeys [ROWS] [COLS] = {
{'1','2','3','A'},
{'4', '5', '6','B'},
{'7', '8','9','C'},
{'*','0' ,'#', 'D'}
};
byte rowPins[ROWS] = {37, 36, 35, 34}; //connect to the row pinouts of the keypad
byte colPins [COLS] = {33, 32, 31, 30}; //connect to the column pinouts of the
keypad
//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad ( makeKeymap (hexaKeys), rowPins, colPins,
ROWS, COLS);
void setup() {
Serial.begin(9600);
}
void loop() {
char customkey = customKeypad.getKey();
if (customkey){
Serial.println(customkey);
}
}
```

7) **Write an Arduino program to interface and to perform the IR SENSOR operation using Arduino uno**

AIM: To Write an Arduino program to interface and to perform the IR SENSOR operation.

SOFTWARE USED: Arduino IDE

THEORY:

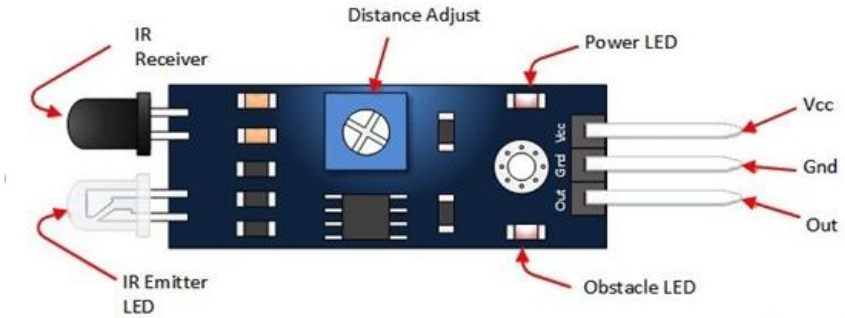
Infrared technology addresses a wide variety of wireless applications. The main areas are sensing and remote controls. In the electromagnetic spectrum, the infrared portion is divided into three regions: near infrared region, mid infrared region and far infrared region.

The wavelengths of these regions and their applications are shown below.

- Near infrared region — 700 nm to 1400 nm — IR sensors, fiber optic
- Mid infrared region — 1400 nm to 3000 nm — Heat sensing
- Far infrared region — 3000 nm to 1 mm — Thermal imaging

The frequency range of infrared is higher than microwave and lesser than visible light.

IR Sensor has three pins (GND, V, OUT). It works under 5v. Connect V pin of sensor with Arduino 5v.GND pin of IR connected to GND of Arduino. Interface OUT pin of sensor with 2nd pin of Arduino.



SOURCE CODE:

```
const int IR_Sensor=2;
void setup() {
pinMode(13, OUTPUT);
1/Pin 2 is connected to the output of IR_Sensor
pinMode (IR_Sensor, INPUT);
}
void loop() {
if(digitalRead(IR_Sensor)==HIGH) //Check the s
digitalWrite(13, HIGH); // set the LED on
else
digitalWrite(13, LOW); // set the LED off
// wait for a second
delay(1000);
}
```

8) **Write an Arduino program to interface and to perform the LCD operation using Arduino uno**

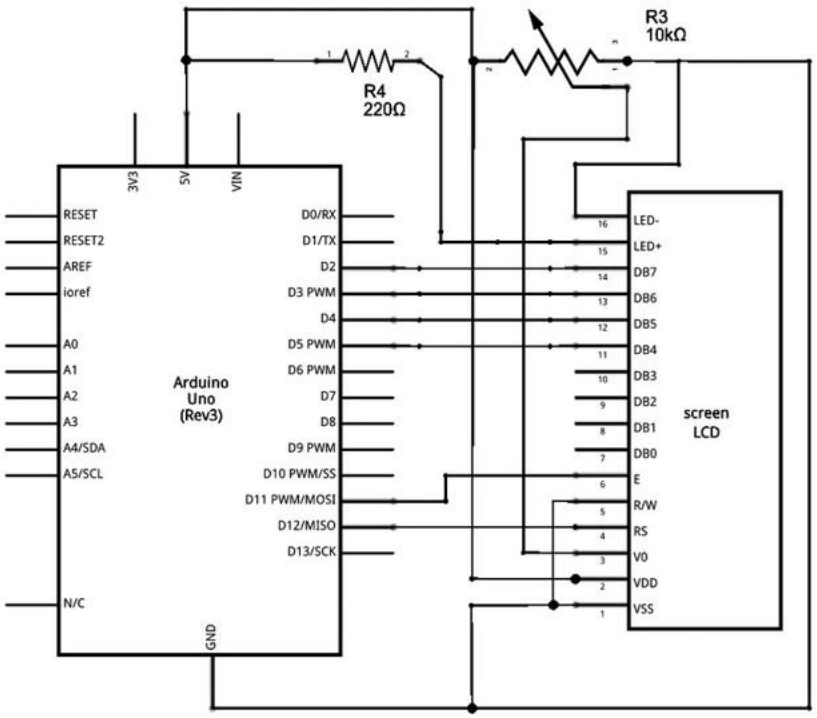
AIM: To Write an Arduino program to interface and to perform the LCD operation .

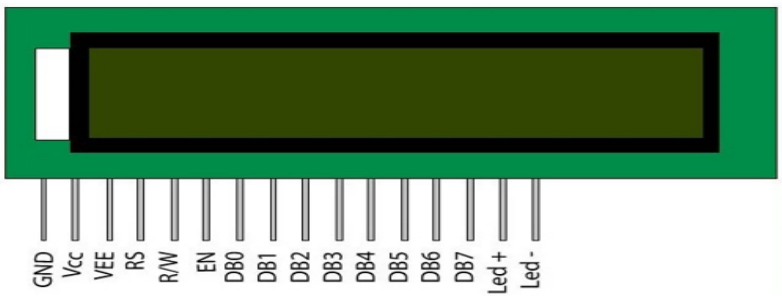
SOFTWARE USED: Arduino IDE

THEORY:

Before wiring the LCD screen to your Arduino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image above. To wire your LCD screen to your board, connect the following pins: LCD RS pin to digital pin 12. LCD Enable pin to digital pin 11.

the most commonly used one is **16×2 LCD Module** which can display 32 ASCII characters in 2 lines (16 characters in 1 line). Other commonly used LCD displays are 20×4 Character LCD, Nokia 5110 LCD module, 128×64 Graphical LCD Display and 2.4 inch TFT Touch screen LCD display.





SOURCE CODE:

```
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
}
void loop() {
lcd.setCursor(0,0);
//sets the cursor at row 0 column 0
lcd.print("16x2 LCD MODULE"); // prints 16x2 LCD MODULE
lcd.setCursor(2,1);
//sets the cursor at row 1 column 2
lcd.print("HELLO WORLD"); 1/ prints HELLO WORLD
}
```

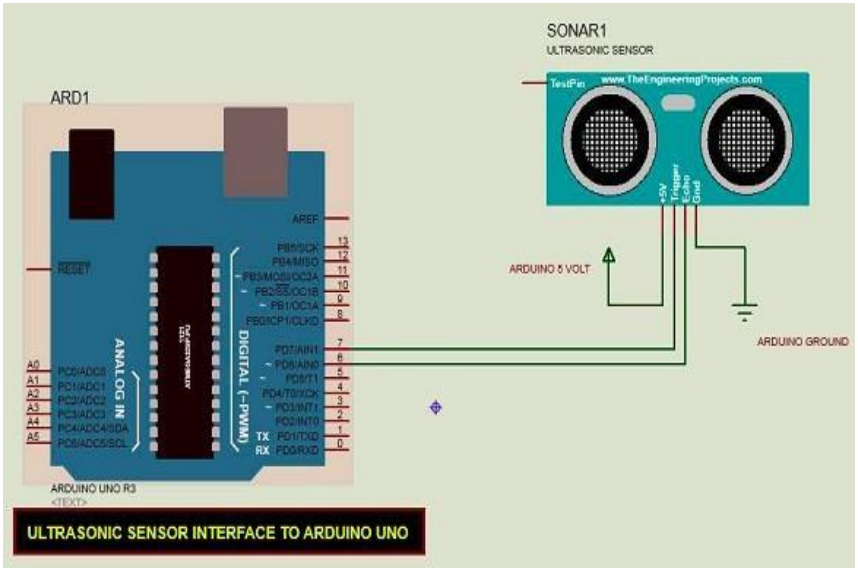
9) **Write a Arduino program to interface and to perform the ultrasonic sensor operation using Arduino uno**

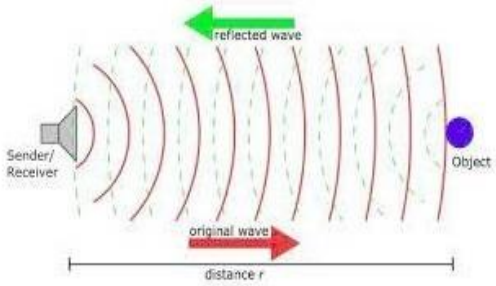
AIM: To Write an Arduino program to interface and to perform the ultrasonic sensor operation.

SOFTWARE USED: Arduino IDE

THEORY:

Echo Pin on digital pin 2 and trig Pin on digital pin 3. by using the keyword "define'. Next declare two variables, one is "duration". This is for store the duration of sound wave travelled. Other is "distance" for store the distance calculated. the serial communication with baurd rate as 9600. It is done by the keyword "Serial.begin(9600)". Then set the trigPin as "OUTPUT", by the keyword "pinMode(trigPin, OUTPUT)". Because the trigPin is the input pin of transmitter of sensor module. this purpose we use the keyword "digitalWrite(trigPin, LOW)". Then hold this state for 2 microseconds by the keyword "delayMicroseconds(2)".





SOURCE CODE:

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor
void setup() {
Serial.begin(9600); // Starting Serial Terminal
}
void loop() {
long duration, inches, cm;
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds (2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(10);
digitalWrite(pingPin, LOW);
pinMode(echoPin, INPUT);
duration = pulseIn(echoPin, HIGH);
inches = microsecondsToInches (duration);
cm = microsecondsToCentimeters (duration);
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
delay(100);
}
```

10) **Write a Raspberry pi program to interface and to perform the BLINK OPERATION**

AIM: To Write a Raspberry pi program to interface and to perform the Blink Operation using raspberry pi board

SOFTWARE USED: Arduino IDE

THEORY:

The circuit created we need to write the Python script to blink the LED. Before we start writing the software we first need to install the Raspberry Pi GPIO Python module. This is a library that allows us to access the GPIO port directly from Python.

With the library installed now open your favourite Python IDE (I recommend Thonny Python IDE more information about using it here).

Our script needs to do the following:

- Initialize the GPIO ports
- Turn the LED on and off in 1 second intervals

To initialize the GPIO ports on the Raspberry Pi we need to first import the Python library, the initialize the library and setup pin 8 as an output pin.

SOURCE CODE:

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings (False) # Ignore warning for now
GPIO.setmode (GPIO. BOARD) # Use physical pin numbering
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value
to low (off)
while True: # Run forever
GPIO.output(8, GPIO.HIGH) # Turn on
sleep(1) # Sleep for 1 second
GPIO.output(8, GPIO.LOW) # Turn off
sleep(1) # Sleep for 1 second
```