STUDY OF ARDUINO IDE

The Arduino IDE is a software application where the Arduino programs are written in C or C++ and uploaded to the Arduino board

Installation

It is an open source software which can be downloaded for free from https://www.arduino.cc/en/software. Make sure to download the stable version and not the hourly builds or previous releases

Configuration

After successful installation, set the type to the model of Arduino board because this tells the IDE which microcontroller the Arduino board has so that it can compile the program accordingly

Head to the “Tools” menu in the navigator bar, choose the “Board” option and set the type of the Arduino board. Now the IDE knows about the board, now configure the IDE which USB port to use for sending data i.e. the compiled code to the Arduino board

Make sure to connect the Arduino board to the computer using a USB cable. After connecting,head to the “Tools” menu, choose the “Port” option and set the port to the USB port to the portwhich is connected the Arduino board

Writing programs

Every program requires the following structure as they are the essential part of the programvoid

```
setup {

}

void loop {

}
```

The “setup” function is executed only once and doesn’t change during the runtime. Code suchas setting the pins for input and output goes here. Always remember to place this function above the “loop” function

The “loop” function is executed post the execution of the “setup” function and is code inside this function is executed infinite no. of times. Most of the code that makes up the programs goes inside this function. The IDE provides many inbuilt functions that can be used here

Uploading programs

To send the programs to the Arduino board, click the right arrow button which is located just below the menu bar and to the left of the tick mark button. The tick mark button will compile the code and points out the errors in the code. By default, the upload button i.e. the right arrowbutton compiles the code before sending it to the Arduino board

 Reg no:

BASIC IN-BUILT FUNCTIONS

Arduino IDE provides us with many utility functions that helps us interpret and control the Arduino boards during the run times. Some of the basic function it provides are as follows:

pinMode(pin, mode):

This function is used to configure the pin specified to behave as Input or Output. pin–number representing the pin number in the board, mode – INPUT or OUTPUT

digitalRead(pin):

Function which is used to read the digital signal from the specified pin. Returns either HIGHor LOW. pin – number representing the pin number in the board

digitalWrite(pin, value):

This function is used to write a HIGH or a LOW value to a digital pin. pin – number representing the pin number in the board, value – HIGH or LOW

analogRead(pin):

This function is used to read the analog signal from the specified analog pin. Returns an integer value in the range 0 to 1023. pin – number representing the pin number in the board

analogWrite(pin, value):

This function is used to write aninteger value to the analog pin. The integer value should be within the range 0 to 255. pin – number representing the pin number in the board, value – 0 to255

tone(pin, value, duration):

This function is used to generate a square wave for the specified “pin” with a frequency of “value” which last for “duration” milliseconds (defaults to infinity)

noTone(pin):

This function is used to stop the square wave being produced for the specified “pin”

delay(value):

Function that pauses the programs for the specified milliseconds i.e. value

STUDY OF ARDUINO UNO BOARD

Reset Switch - When this switch is clicked, it sends a logical pulse to the reset pin of the Microcontroller, and now runs the program again from the start

USB Connector - This is a printer USB port used to load a program from the Arduino IDE onto the Arduino board. The board can also be powered through this port

USB Interface Chip - It converts signals in the USB level to a level that an Arduino UNO board understands

Crystal Oscillator - This is a quartz crystal oscillator which ticks 16 million times a second. On each tick, the microcontroller performs one operation, for example, addition, subtraction, etc.

Voltage Regulator – Protects the board from burning out when more than 5 volts are given tothe board and is located between the power port and USB connector

Power Port - The Arduino board can be powered through an AC-to-DC adapter or a battery. The power source can be connected by plugging in a 2.1mm center-positive plug into the powerjack of the board

Digital Pins - These pins can be used as either input or output pins. When used as output pinsthey act as power supply source and when used as input pins they read signals. Pins prependedwith ~ (3, 5, 6, 9, 10, 11) are digital pins and can also act as analog pins

TX RX LEDs - TX stands for transmit, and RX for receive. These are indicator LEDs which blink whenever the UNO board is transmitting or receiving data

Microcontroller - The microcontroller used on the UNO board is Atmega328P which is pre-programmed with bootloader and has 32KB flash memory and 2KB ram

Analog Input Pins – These pins reads analog values and converts them to system understandable digital value. Due to high resistance they just measure voltage

 Reg no:

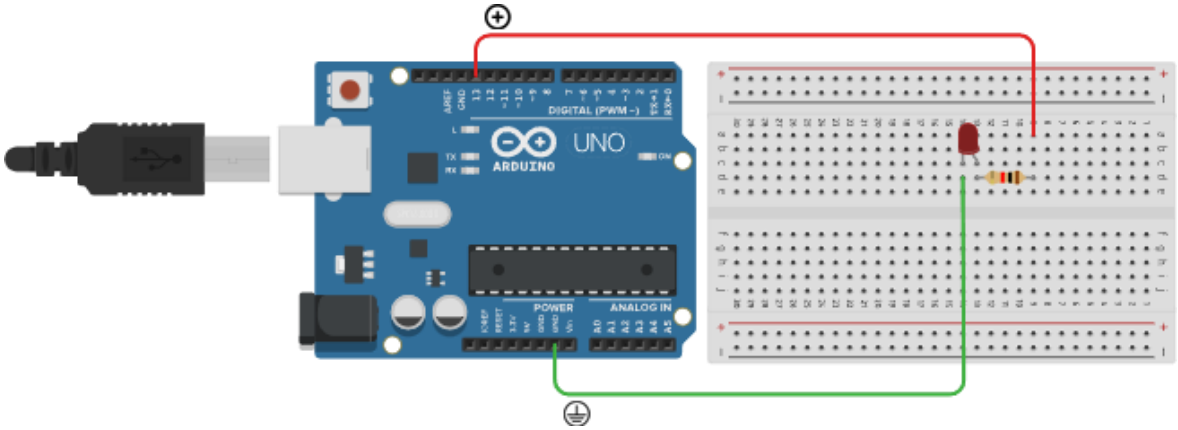
EX NO: 1A **WORKING WITH LED**

DATE:

AIM

To understand the working of Arduino board and the IDE by working with the LED.

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, 1 kΩ Resistor, LED (any colour), bread board and jumper wires

DESCRIPTION

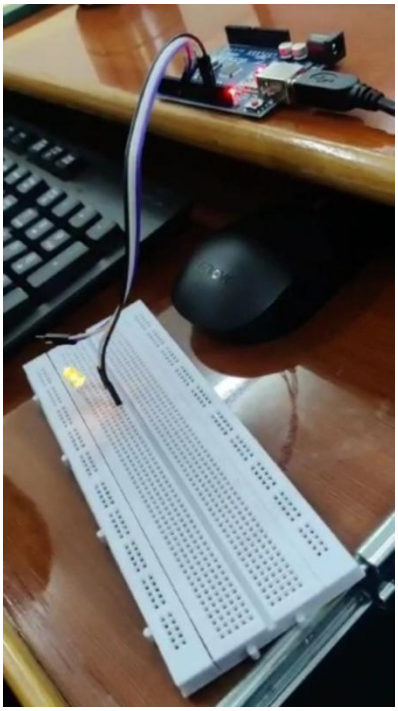
Insert one side of the jumper wire into GND on the board. Lead the other side to the breadboard. Take out LEDs and resistors. Place one end of the resistor to the right of the jumper wire in the same row. Attach the resistor to any column in the same row. Now place the LED. The short end of the LED is connected to the bread board on the same column of the connected jumper wire. The long end of the LED is connected on the same column as that of the resistor. Finally, connect another jumper wire from the Pin 13 in Arduino to the bread board. The other end should be connected the same column of the connected resistor.

PROGRAM

```
void setup()
{
        pinMode(13, OUTPUT);
}

void loop()
{
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);
}
```

 Reg no:

OUTPUT



RESULT

Reg no:

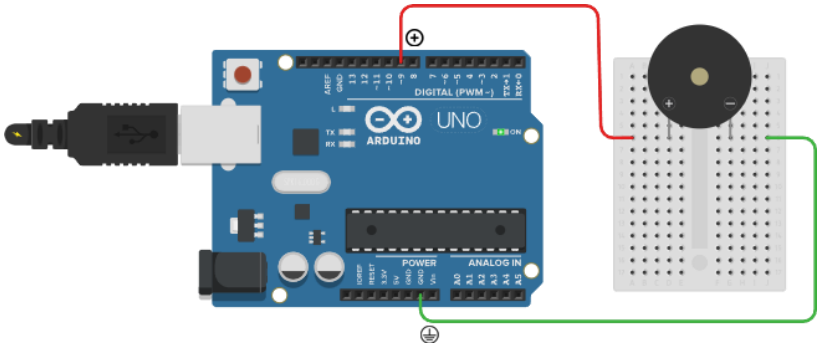
EX NO: 1A ii **WORKING WITH BUZZER**

DATE:

AIM

To integrate Piezo (sound sensor) with Arduino Uno R3

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED

Arduino Uno R3, Piezo (buzzer), bread board and jumper wires

DESCRIPTION

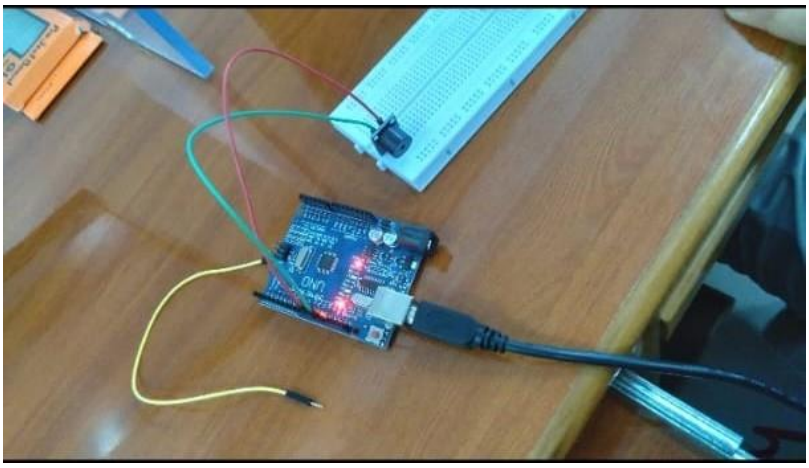
Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the cathode of the Piezo on the same row where the wire from GND is placed. Finally, connect the jumper wire from Pin 9 of the Arduino board to the anode of Piezo in the breadboard

PROGRAM

```
void setup()
{
	pinMode(9, OUTPUT);
}
void play (int cnt)
{
	for (int i = 1; i <= 3; i += 1)
	{
		tone(9, 140);
		delay(cnt);
		noTone(9);
		delay(cnt);
	}
}
void loop()
{
	play(500);
	delay(300);
	play(250);
	delay(300);
}
```

 Reg no:

OUTPUT



RESULT

Reg no:

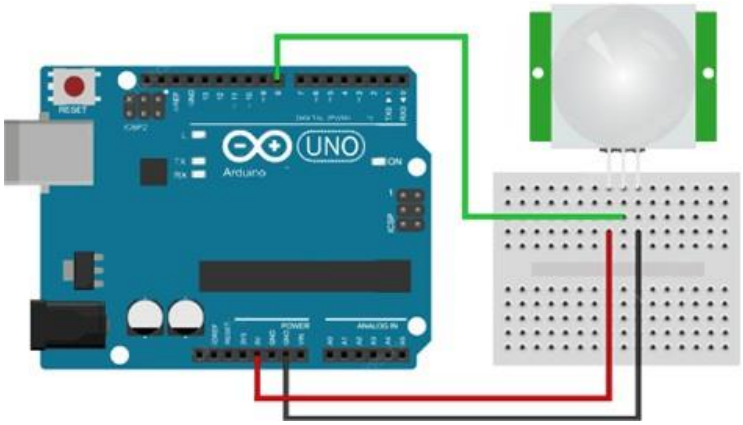
EX NO: 1B **MOTION DETECTION USING PIR SENSOR**

DATE:

AIM

To build an application for Motion Detection using PIR sensor.

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, PIR Motion Detector Sensor, bread board and jumper wires.

DESCRIPTION

Most PIR sensors have a 3-pin connection at the side or bottom. One pin will be ground, another will be signal and the last pin will be power. Power is usually up to 5V. Sometimes bigger modules don't have direct output and instead just operate a relay which case there is ground, power and the two switch associations. Interfacing PIR with microcontroller is very easy and simple. The PIR acts as a digital output so all that's needed to be done is listening for the pin to flip high or low. The motion can be detected by checking for a high signal on a single I/O pin.

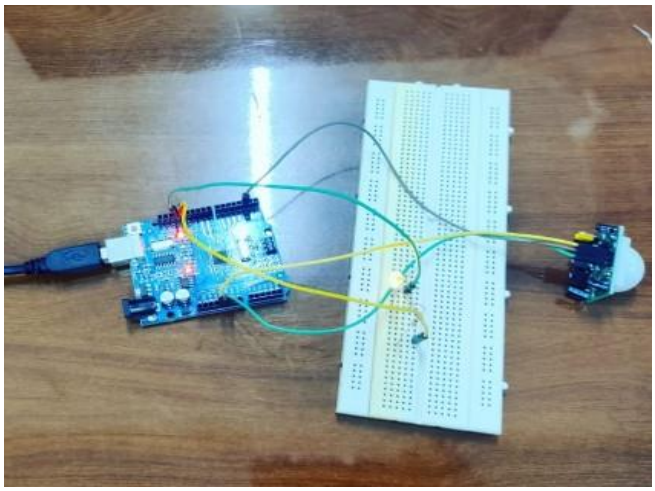
PROGRAM

```
int led = 13;                // the pin that the LED is atteched to
int sensor = 2;              // the pin that the sensor is atteched to
int state = LOW;
int val = 0;
void setup() {
  pinMode(led, OUTPUT);
  pinMode(sensor, INPUT);
  Serial.begin(9600);
}
void loop(){
  val = digitalRead(sensor);  // read sensor value
  if (val == HIGH) {           // check if the sensor is HIGH
    digitalWrite(led, HIGH);   // turn LED ON
    delay(500);                // delay 100 milliseconds


    if (state == LOW) {
```

 Reg no:

```
      Serial.println("Motion detected!");
      state = HIGH;       // update variable state to HIGH
    }
  }
  else
  {
      digitalWrite(led, LOW); // turn LED OFF
      delay(500);             // delay 200 milliseconds
      if (state == HIGH){
        Serial.println("Motion stopped!");
        state = LOW;       // update variable state to LOW
    }
  }

}
```

OUTPUT



```
16:00:06.550 -> Motion Detected!!
16:00:06.550 -> 0
16:00:07.048 -> Motion Detected!!
16:00:07.048 -> 0
16:00:07.546 -> Motion Detected!!
16:00:07.578 -> 0
16:00:08.043 -> Motion Detected!!
16:00:08.076 -> 0
```

RESULT

Reg no:

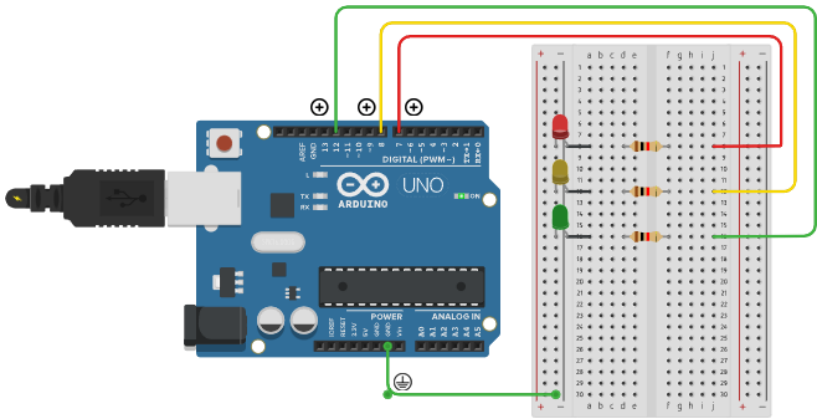
EX NO: 1C **STIMULATION OF TRAFFIC LIGHT**

DATE:

AIM

To stimulate traffic light behaviour using LEDs and Arduino Uno

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, 1 kΩ Resistor, LED (red, green and yellow), bread board and jumper wires

DESCRIPTION

Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the red, yellow and green LEDs cathodes (short end of led) parallel to each other on the same row where the wire from GND is placed. Similarly place the three resistor's one end to the anode (long end of led) of each of the LEDs. Finally, connect the jumper wires from Pin 12, Pin 8 and Pin 7 of the Arduino board to each of the resistors unconnected end in the breadboard

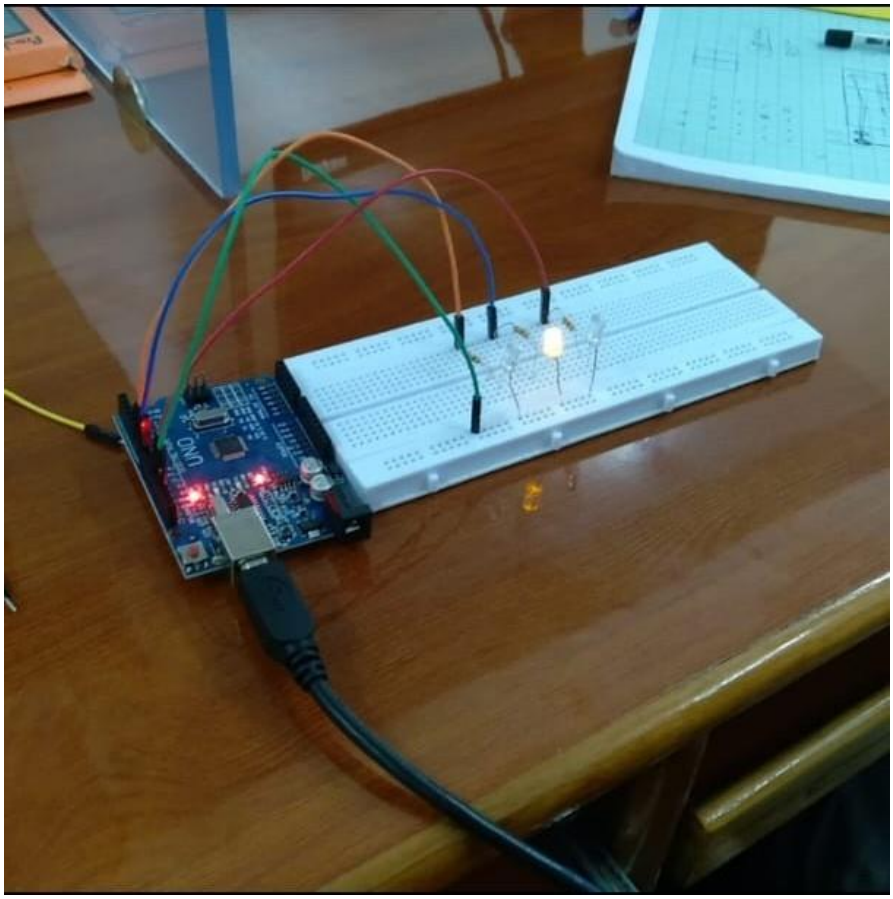
PROGRAM

```
void setup()
{
        pinMode(7,   OUTPUT);
        pinMode(8,   OUTPUT);
        pinMode(12, OUTPUT);

        digitalWrite(7,   LOW);
        digitalWrite(8,   LOW);
        digitalWrite(12, LOW);
}
```

 Reg no:

```
void loop()
{
        digitalWrite(12, HIGH);
        digitalWrite(8, LOW);
        digitalWrite(7, LOW);
        delay(1000);
        digitalWrite(12, LOW);
        digitalWrite(8, HIGH);
        digitalWrite(7, LOW);
        delay(1000);
        digitalWrite(12, LOW);
        digitalWrite(8, LOW);
        digitalWrite(7, HIGH);
        delay(1000);
}
```

OUTPUT



RESULT

Reg no:

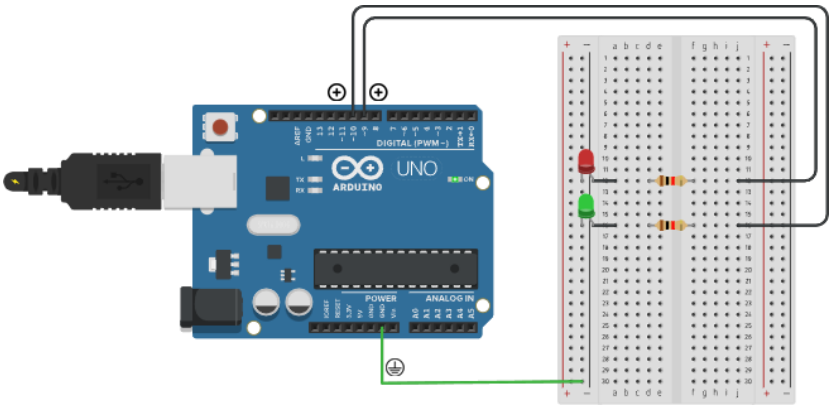
EX NO: 1D CONTROLLING LED INTENSITY USING PWM SIGNAL

DATE:

AIM

To control the intensity of the LED's using PWM signal with the help of Arduino Uno R3

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, 1 kΩ Resistor, LED (red and green), bread board and jumper wires

DESCRIPTION:

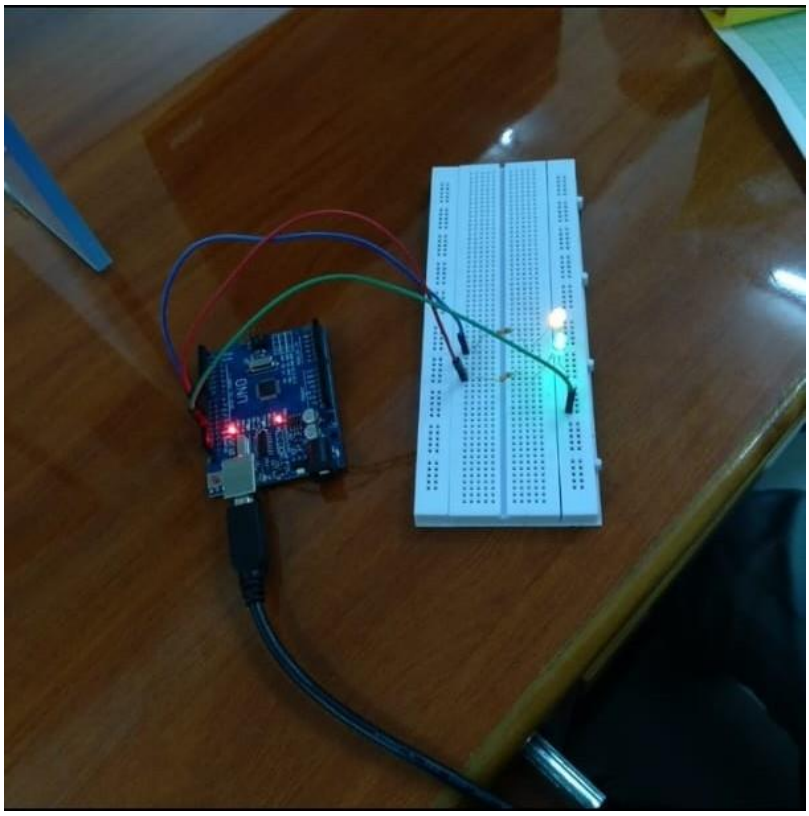
Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the red and green LEDs cathodes (short end of led) parallel to each other on the same row where the wire from GND is placed. Similarly place the two resistor's one end to the anode (long end of led) of each of the LEDs. Finally, connect the jumper wires from Pin 10 and Pin 9 of the Arduino board to each of the resistors unconnected end in the breadboard

PROGRAM

```
void setup()
{
        pinMode(9, OUTPUT);
        pinMode(10, OUTPUT);

        analogWrite(9, 0);
        analogWrite(10, 255);
}
void loop()
{
        int i = 0;
        for (i = 50; i <= 255; i += 10)
        {
```

```
                analogWrite(9, i);
                analogWrite(10, 255 - i);
                delay(500);
        }
        for (i = 50; i <= 255; i += 10)
        {
                analogWrite(9, 255 - i);
                analogWrite(10, i);
                delay(500);
        }
}
```

OUTPUT



RESULT

Reg no:

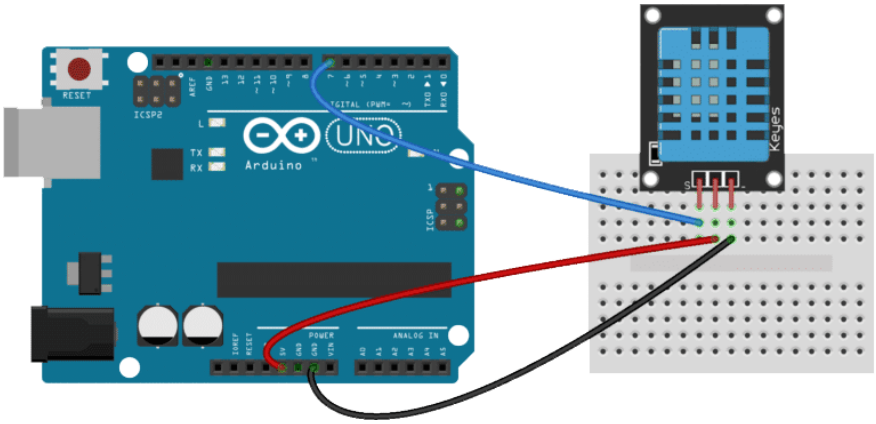
EX NO: 2 **WORKING WITH DHT SENSOR**

DATE:

AIM:

To integrate DHT (Humidity & Temperature) sensor with Arduino Uno R3

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED

Arduino Uno R3, DHT sensort, bread board and jumper wires

DESCRIPTION

Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the Ground of the DHT sensor on the same row where the wire from GND is placed. Connect the jumper wire from Pin 7 of the Arduino board to the signal of DHT sensor in the breadboard. Give 5V to DHT sensor's Vcc from Arduino board

PROGRAM

```
#include <dht.h>
dht DHT;
void setup()
{
        Serial.begin(9600);
}

void loop()
{
        DHT.read11(7);
        Serial.println("Temperature: ", DHT.temperature, " C");
        Serial.println("Humidity: ", DHT.humidity, " %");
        delay(1000);
}
```

 Reg no:

OUTPUT

Temperature: 26 C
Humidity: 88 %

RESULT

Reg no:

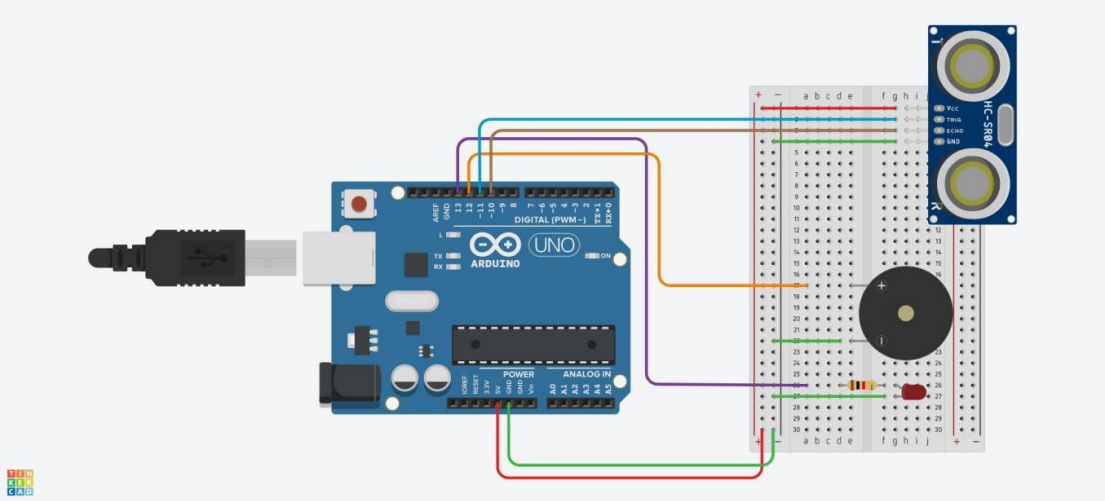
EX NO: 3A CROWD CALCULATION USING ULTRASONIC SENSOR

DATE:

AIM

To build an application for counting the total people tresspassed using the ultrasonic sensor

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, Ultrasonic Sensor (HC-SR04), bread board and jumper wires

DESCRIPTION

Take a GND and 5V wires from Arduino UNO to bread board. Connect PIN 13 to anode of LED with a resistor, PIN 12 to anode of sound senor, PIN 11 and PIN 10 to trig and echo of the ultrasonic sensor respectively. Take wire from common GND to cathode of LED, cathode of sound sensor and GND of ultrasonic sensor.

PROGRAM

```
int led = 13, sound = 12;
int trig = 11, echo = 10;
int count;
float prev;
void setup()
{
Serial.begin(9600);
pinMode(led, OUTPUT);
pinMode(sound, OUTPUT);
pinMode(echo, INPUT);
pinMode(trig, OUTPUT);
digitalWrite(led, LOW);
digitalWrite(sound, LOW);
digitalWrite(echo, LOW);
count = 0;
prev = 0;
}
```

```
  void on_detect()
  {
   count += 1;
   Serial.print("Count: ");
   Serial.println(count);
   digitalWrite(led, HIGH);
   tone(sound, 1024);
   delay(500);
   digitalWrite(led,  LOW);
   noTone(sound);
  }
void loop()
{
digitalWrite(trig, HIGH);
delay(1);
digitalWrite(trig, LOW);
float input = 0.17 * pulseIn(echo, HIGH);
if (input < 1000)
{
if (!(prev != 0 && prev - 10 <= input && input <= prev + 10)) {prev = input;
on_detect(); delay(1000);
}
}
}
```

OUTPUT

Count: 1

Count: 2

Count: 3

Count: 4

Count: 5

RESULT

 Reg no:

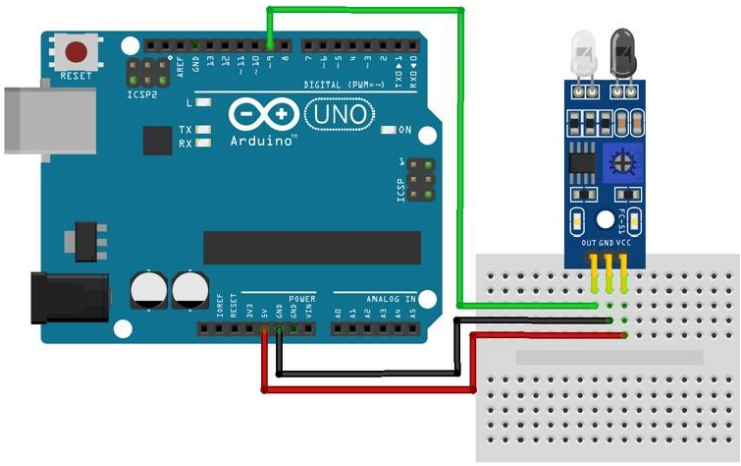
EX NO: 3B PROXIMITY DETECTION USING IR SENSOR

DATE:

AIM

To build an application for Proximity Detection using IR Sensor.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED

Arduino Uno R3, IR Sensor, LED, resistor, bread board and jumper wires.

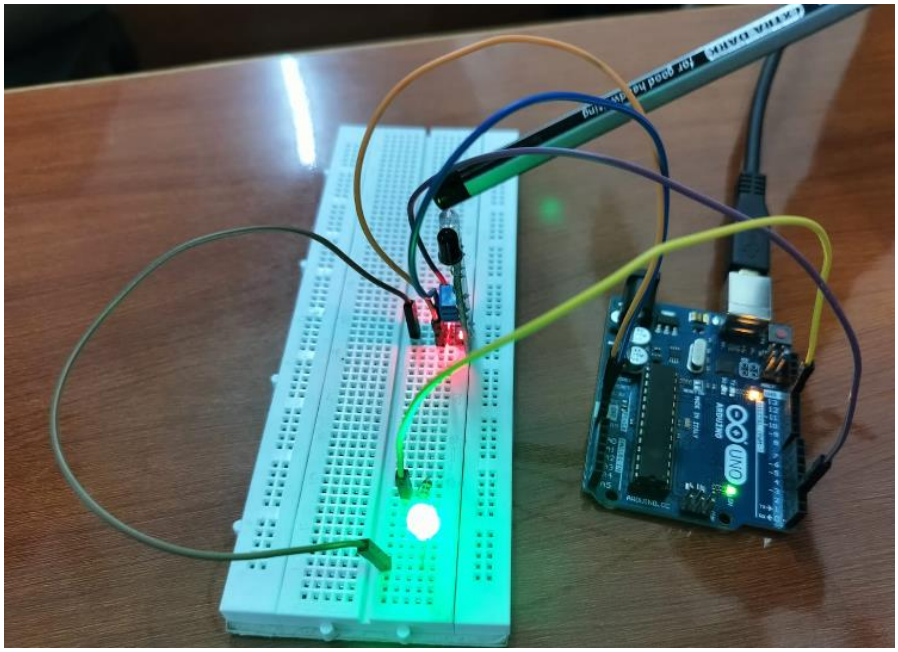
DESCRIPTION

The connections for the IR sensor with the Arduino are as follows: Connect the negative wire on the IR sensor to GND on the Arduino. Connect the middle of the IR sensor which is the VCC to 3.3V on the Arduino. Connect the signal pin on the IR sensor to any Digital Pin in Arduino. Use the LED to show the signal of the object detection.

PROGRAM

```
int IR=2;
int led=13;
void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(IR, INPUT);
 pinMode(led,OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
 int statussensor=digitalRead(IR);
 if(statussensor==1) digitalWrite(led,LOW);
 else digitalWrite(led,HIGH);
```

OUTPUT



RESULT

Reg no:

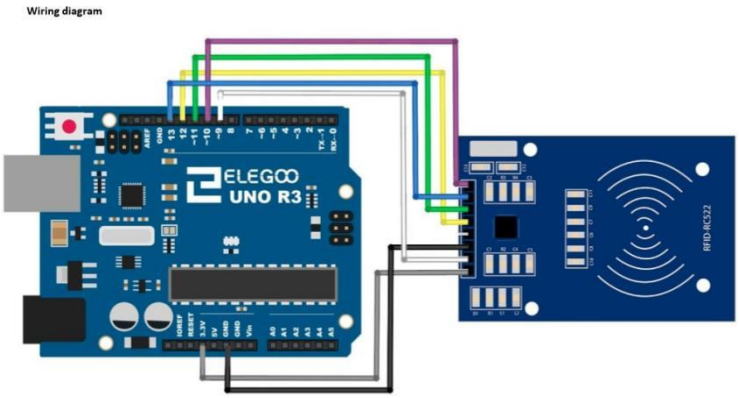
EX NO: 3C SECURITY ACCESS USING RFID SENSOR

DATE:

AIM

To build an application for Security Access using RFID sensor.

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, RFID Sensor, bread board and jumper wires.

DESCRIPTION

First, install the MFRC522 library for RFID module. There are 7 pins to connect to the RFID sensor. They are SDA, SCK, MOSI, MISO, RST which is connected to any digital pin in Arduino Uno. Then the rest are connected with 3.3 volts and GND.

Step 1

After downloading the zip files for MFRC522 , install it.
Go to Sketch → Include Library → Add .ZIP library
Then add the downloaded zip files.
Now the libraries are included in your Arduino IDE.

Step 2

Open two Arduino IDE window and Select Arduino UNO as Board and select the appropriate COM port.
Board: Tools > Board > Arduino/Geniuno UNO.

STEP 3

Select the serial device of the board from the Tools / Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

Step 4

Program

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9 // Configurable, see typical pin layout above
#define SS_PIN 10 // Configurable, see typical pin layout above
```

```
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
void setup() {
Serial.begin(9600); // Initialize serial communications with the PC
while (!Serial); // Do nothing if no serial port is opened (added for Arduinos based on
ATMEGA32U4)
SPI.begin(); // Init SPI bus
mfrc522.PCD_Init(); // Init MFRC522
delay(4); // Optional delay. Some board do need more time after init to be ready, see Readme
mfrc522.PCD_DumpVersionToSerial(); // Show details of PCD - MFRC522 Card Reader
details
Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}
void loop() {
// Reset the loop if no new card present on the sensor/reader. This saves the entire process when
idle.
if ( ! mfrc522.PICC_IsNewCardPresent()) {
return;
}
// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
return;
}
// Dump debug info about the card; PICC_HaltA() is automatically called
mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Step 5: Upload the sketch and open the Serial Monitor. As soon as you bring the tag closer to the
module, you'll probably get something like the figure given below. Do not move the tag until all the
information is displayed.

Output :

```
Firmware Version: 0x92 = v2.0
Scan PICC to see UID, SAK, type, and data blocks...
Firmware Version: 0x92 = v2.0
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: D3 44 CA 02
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 AccessBits
15 63 PCD_Authenticate() failed: Timeout in communication.
14 59 PCD_Authenticate() failed: Timeout in communication.
13 55 PCD_Authenticate() failed: Timeout in communication.
12 51 PCD_Authenticate() failed: Timeout in communication.
11 47 PCD_Authenticate() failed: Timeout in communication.
10 43 PCD_Authenticate() failed: Timeout in communication.
9 39 PCD_Authenticate() failed: Timeout in communication.
8 35 PCD_Authenticate() failed: Timeout in communication.
7 31 PCD_Authenticate() failed: Timeout in communication.
6 27 PCD_Authenticate() failed: Timeout in communication.
5 23 PCD_Authenticate() failed: Timeout in communication.
4 19 PCD_Authenticate() failed: Timeout in communication.
3 15 PCD_Authenticate() failed: Timeout in communication.
2 11 PCD_Authenticate() failed: Timeout in communication.
1 7 PCD_Authenticate() failed: Timeout in communication.
```

0 3 PCD_Authenticate() failed: Timeout in communication.

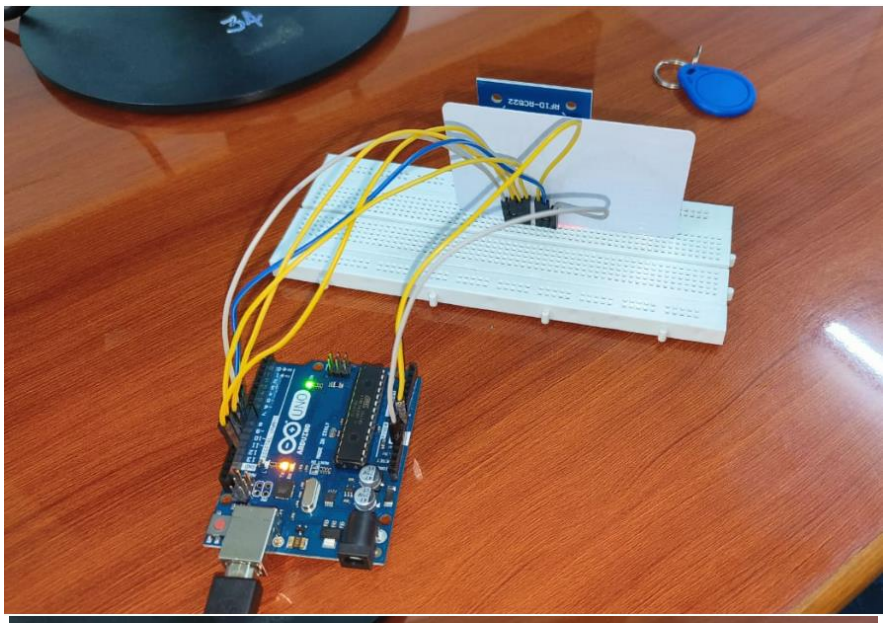
Card UID: D3 44 CA 02
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 AccessBits
15 63 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
14 59 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]

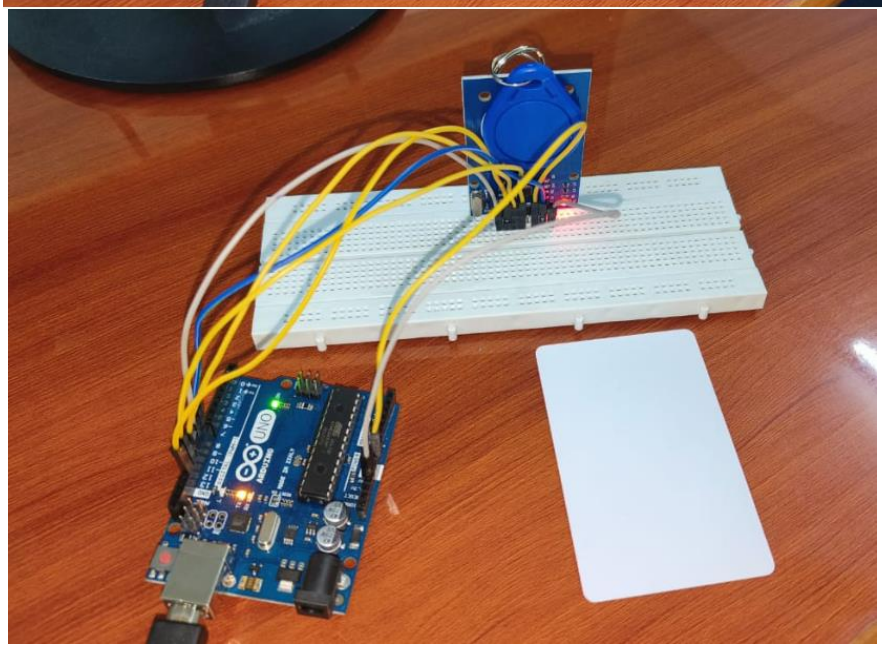
56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
13 55 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
12 51 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
11 47 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
46 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
10 43 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
9 39 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
37 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
8 35 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
7 31 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
6 27 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
26 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
5 23 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
4 19 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]

16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
3 15 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
2 11 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
1 7 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
0 3 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
0 D3 44 CA 02 5F 08 04 00 62 63 64 65 66 67 68 69 [0 0 0]

Card UID: 37 DF B4 C6
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 AccessBits
15 63 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
14 59 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
13 55 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF [0 0 1]
54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [0 0 0]
12 51 PCD_Authenticate() failed: Timeout in communication.
11 47 PCD_Authenticate() failed: Timeout in communication.
10 43 PCD_Authenticate() failed: Timeout in communication.
9 39 PCD_Authenticate() failed: Timeout in communication.
8 35 PCD_Authenticate() failed: Timeout in communication.
7 31 PCD_Authenticate() failed: Timeout in communication.
6 27 PCD_Authenticate() failed: Timeout in communication.
5 23 PCD_Authenticate() failed: Timeout in communication.
4 19 PCD_Authenticate() failed: Timeout in communication.
3 15 PCD_Authenticate() failed: Timeout in communication.
2 11 PCD_Authenticate() failed: Timeout in communication.
1 7 PCD_Authenticate() failed: Timeout in communication.
0 3 PCD_Authenticate() failed: Timeout in communication.

OUTPUT





RESULT

Reg no:

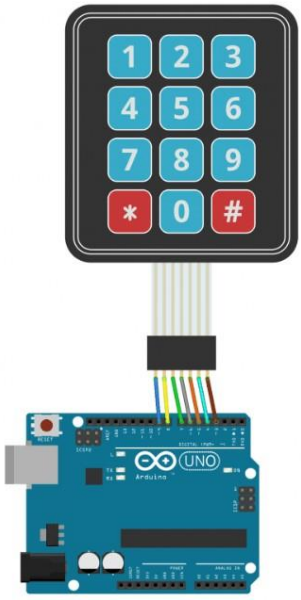
EX NO: 3D **PASSWORD ACCESS USING KEYPAD**

DATE:

AIM:

To build an application for providing Password Access using Keypad device.

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, Keypad Module, bread board and jumper wires.

DESCRIPTION

To use this module, it is necessary to install the Keypad library in the Arduino IDE. To install the Keypad library, go to Sketch > Include Library > Manage Libraries and search for "keypad". Click on the library, then click install. The keypad wires can be connected to any Digital Pin in the Arduino Uno.

The Arduino detects which button is pressed by detecting the row and column pin that's connected to the button.

This happens in four steps:

1. First, when no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW.

2. When a button is pressed, the column pin is pulled LOW since the current from the HIGH column flows to the LOW row pin.

3. The Arduino now knows which column the button is in, so now it just needs to find the row the button is in. It does this by switching each one of the row pins HIGH, and at the same time reading all of the column pins to detect which column pin returns to HIGH.

4. When the column pin goes HIGH again, the Arduino has found the row pin that is connected to the button.

PROGRAM:

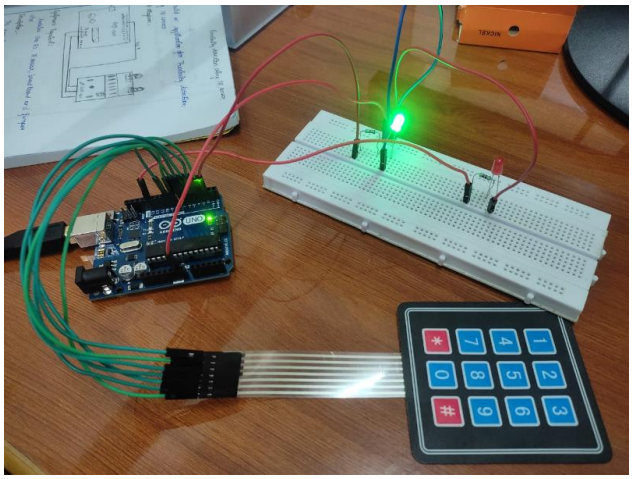
```
#include <Keypad.h>
const byte row_count = 4;
const byte column_count = 3;
int red_led=2;
int green_led=3;

byte row_pins[row_count] = {12, 11, 10, 9};
byte column_pins[column_count] = {8, 7, 6 };

char* password = "69420";
int currentposition=0;

char keys4x3[row_count][column_count] = {
 {'1', '2', '3'},
 {'4', '5', '6'},
 {'7', '8', '9'},
 {'*', '0', '#'}
};

Keypad myKeypad = Keypad(makeKeymap(keys4x3), row_pins, column_pins, row_count,
column_count);

void setup() {
  Serial.begin(9600);
  Serial.println("Enter Password: ");
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop() {

 char code = myKeypad.getKey();

 if (code!=NO_KEY)
 {
  Serial.print(code);
  Serial.println("");

    if (code == password[currentposition])
   {
     ++currentposition;
```

 Reg no:

```
    if(currentposition==5){
      Serial.println("ACCESS GRANTED! :D");
      digitalWrite(green_led, HIGH);
      delay(2000);
      digitalWrite(green_led, LOW);
     }
    }
     else{
     currentposition=0;
     Serial.println("ACCESS DENIED ;(");
     digitalWrite(red_led, HIGH);
     delay(2000);
     digitalWrite(red_led, LOW);
    }
  }
}
```

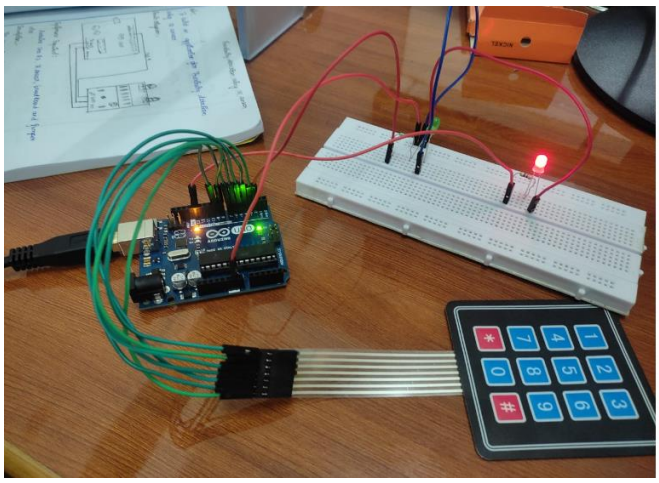
OUTPUT

Green LED lights up when password is correct



```
15:38:38.191 -> Enter Password:
15:38:39.950 -> 6
15:38:41.611 -> 9
15:38:42.541 -> 4
15:38:43.936 -> 2
15:38:44.933 -> 0
15:38:44.933 -> ACCESS GRANTED! :D
```

Red LED Lights up when password is incorrect



```
15:40:30.335 -> Enter Password:
15:40:32.195 -> 5
15:40:32.195 -> ACCESS DENIED ;(
15:40:34.753 -> 6
15:40:37.311 -> 4
15:40:37.311 -> ACCESS DENIED ;(
```

RESULT

Reg no:

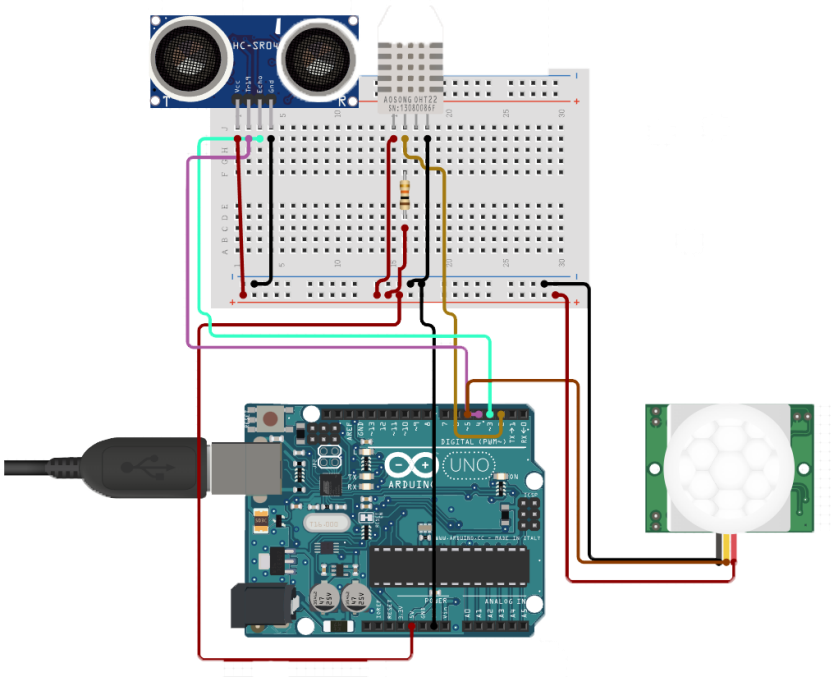
EX NO: 4 MOBILE APPLICATIONS USING DHT, PIR, US SENSORS

DATE:

AIM

To create a mobile application to view the data from DHT, PIR, and US sensors

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Bread board, Arduino UNO, PIR (motion) sensor, US sensor, DHT sensor, Resistor, and jumper wires

DESCRIPTION

Do the wirings according to the circuit diagram. Read data from the sensors using the different IO functions. Store them in a variable and call the send_data_to_mobile function passing all the variables as parameters. This function sends the data to the cloud, which sends it to the mobile application.

PROGRAM

SKETCH

```
#include <dht.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
dht DHT;
char* ssid = "WIFI NAME";
char* password = "WIFI PASSWORD";
char* server = "http://127.0.0.1:5000/send-data-mob";

WiFiClient client;
HTTPClient http;
void connect_to_wifi();
```

```
void setup() {
   pinMode(3, INPUT);
   pinMode(4, OUTPUT);
   pinMode(5, INPUT);
   Serial.begin(9600);
   connect_to_wifi();
}
void connect_to_wifi() {
   WiFi.begin(ssid, password);
   Serial.println("Connecting");
   while(WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.print(".");
   }
   Serial.println("");
   Serial.print("Connected to WIFI Successfully");
}
void send_data_to_mobile(float t, float h, float d, int o) {
   http.begin(client, server);
   http.addHeader("Content-Type", "application/json");
   string body = "{\"temperature\":" + String(t);
   body += ",\"humidity\":" + String(h);
   body += ",\"distance\":" + String(d);
   body += ",\"object\":" + String(o);
   int response = http.POST(body);
   Serial.println(response);
   http.end();
}
void loop() {
   DHT.read11(2);
   float temp = DHT.temperature;
   float hmdty = DHT.humidity;
   digitalWrite(4, LOW);
   delayMicroseconds(2);
   digitalWrite(4, HIGH);
   delayMicroseconds(10);
   digitalWrite(4, LOW);
   float obj_dist = 0.017 * pulseIn(3, HIGH);
   int obj_presence = digitalRead(5);
   send_data_to_mobile(temp, hmdty, obj_dist, obj_presence);
}
```

APP CODE

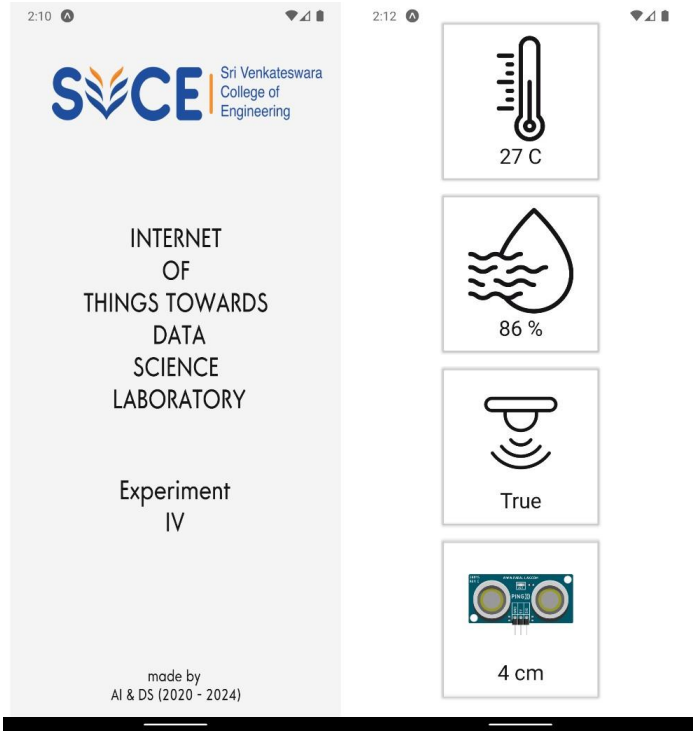
```
import React, {
  useEffect, useState
} from "react";
import {
  View, Text, Image, StyleSheet, ScrollView
} from "react-native";
import io from "socket.io";
```

 Reg no:

```
const socket = io("http://127.0.0.1:000");const App = () => {
  const [data, setData] = useState({});
  useEffect(() => {
    socket.on(
      "data-recieved",
      sensorData => setData(JSON.parse(sensorData))
    );
  }, []);
  return (
    <ScrollView contentContainerStyle={{
      flexGrow: 1,
      alignItems: "center",
      justifyContent: "center"
    }}>
      <View style={styles.item}>
        <Image style={styles.image} source={require("./assets/icons/temperature.png")} />
        <Text style={styles.itemText}>{data.temperature} C</Text>
      </View>
      <View style={styles.item}>
        <Image style={styles.image} source={require("./assets/icons/humidity.png")} />
        <Text style={styles.itemText}>{data.humidity} %</Text>
      </View>
      <View style={styles.item}>
        <Image style={styles.image} source={require("./assets/icons/motion.png")} />
        <Text style={styles.itemText}>{data.object === 1 ? "True" : "False"}</Text>
      </View>
      <View style={styles.item}>
        <Image style={styles.image} source={require("./assets/icons/us.png")} />
        <Text style={styles.itemText}>{data.distance} cm</Text>
      </View>
    </ScrollView>
  );
};

const styles = StyleSheet.create({
  image: {
    width: 120,
    height: 120,
    resizeMode: "contain"
  },
  item: {
    alignItems: "center",
    margin: "2%",
    elevation: 2,
    padding: "4%",
    justifyContent: "space-between",
    width: 180,
    height: 180,
    borderRadius: 3
  },
  itemText: {
    fontSize: 24,
    color: "#171417",
```

```
  }
});

export default App;
```

OUTPUT:



RESULT

 Reg no:

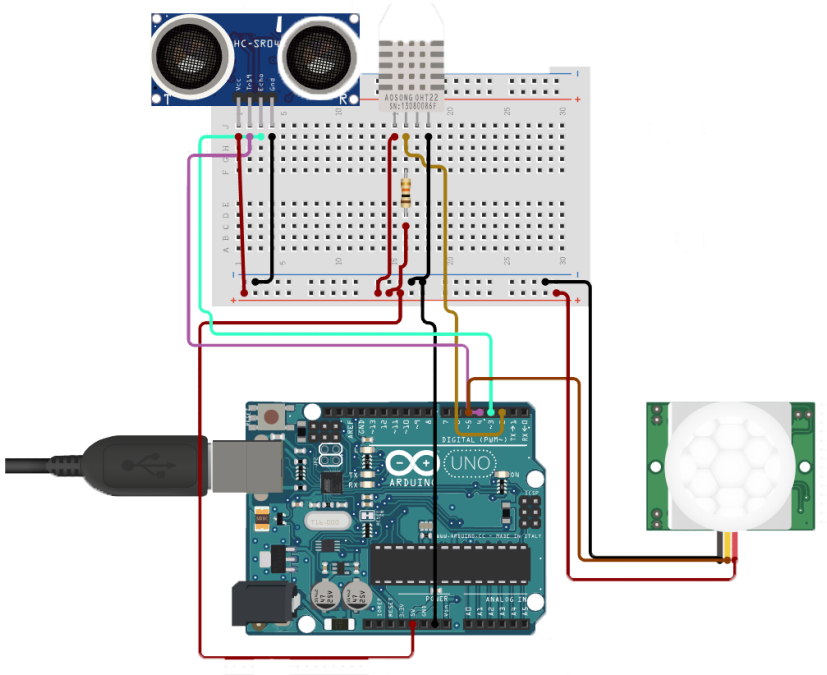
EX NO: 5 STORING DATA FROM DHT, PIR, US SENSORS IN CLOUD

DATE:

AIM

To store the data acquired from DHT, PIR, and US sensors in the cloud

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED:

Bread board, Arduino UNO, PIR (motion) sensor, US sensor, DHT sensor, Resistor, and jumper wires

DESCRIPTION:

Do the wirings according to the circuit diagram. Read data from the sensors using the different IO functions. Store them in a variable and call the send_data_to_cloud function passing all the variables as parameters. This function sends the data to the cloud, which sends it to the mobile application

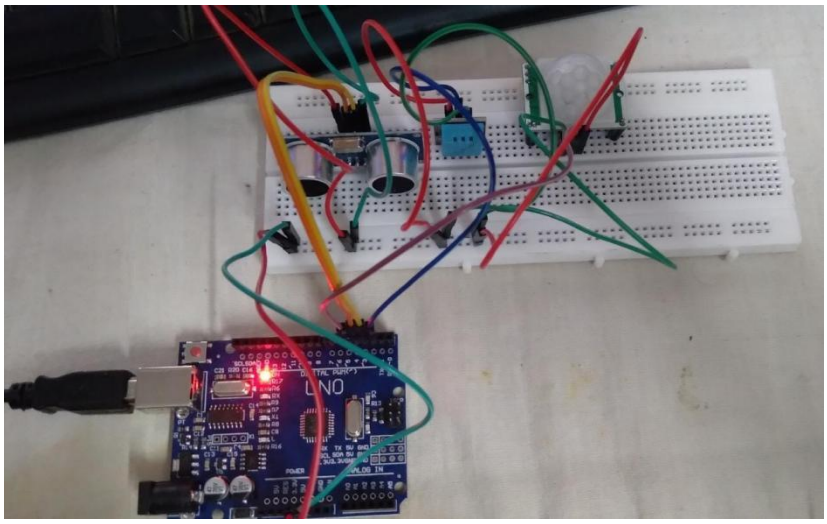
PROGRAM:

```
#include <dht.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
dht DHT;
char* ssid = "WIFI NAME";
char* password = "WIFI PASSWORD";
char* server = "http://127.0.0.1:5000/send-data";

WiFiClient client;
HTTPClient http;
void connect_to_wifi();
void setup()
{
```

```
    pinMode(3, INPUT);
    pinMode(4, OUTPUT);
    pinMode(5, INPUT);
    Serial.begin(9600);

    connect_to_wifi();
}

void connect_to_wifi() {
    WiFi.begin(ssid, password);
    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.print(".");
    }

    Serial.println("");
    Serial.print("Connected to WIFI Successfully");
}

void send_data_to_cloud(float t, float h, float d, int o) {
    http.begin(client, server);
    http.addHeader("Content-Type", "application/json");
    string body = "{\"temperature\":" + String(t);
    body += ",\"humidity\":" + String(h);
    body += ",\"distance\":" + String(d);
    body += ",\"object\":" + String(o);
    int response = http.POST(body);
    Serial.println(response);

    http.end();
}

void loop()
{
    DHT.read11(2);
    float temp = DHT.temperature;
    float hmdty = DHT.humidity;
    digitalWrite(4, LOW);
    delayMicroseconds(2);
    digitalWrite(4, HIGH);
    delayMicroseconds(10);
    digitalWrite(4, LOW);
    float obj_dist = 0.017 * pulseIn(3, HIGH);
    int obj_presence = digitalRead(5);
    send_data_to_cloud(temp, hmdty, obj_dist, obj_presence);
}
```


Reg no:

OUTPUT



A	B	C	D	E
timestamp	temperature	humidity	ultrasonic	pir
1667734672272	26.3	82	3	1
1667734672273	26.8	89	4	1
1667734672274	25.7	84	5	0
1667734672275	26.6	85	4	1
1667734672276	26.9	81	3	1

Data downloaded as CSV from cloud

RESULT

Reg no:

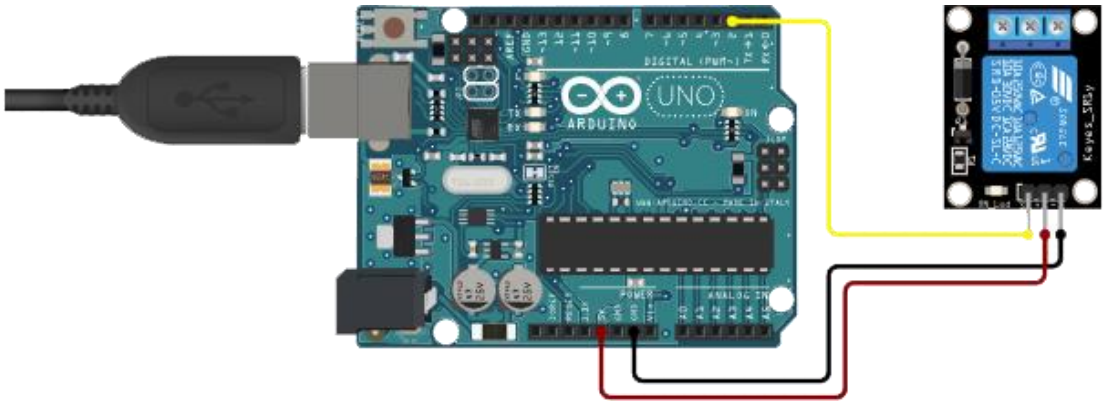
EX NO: 6 **WORKING WITH RELAY**

DATE:

AIM

To integrate Relay Control with Arduino Uno R3

CIRCUIT DIAGRAM



EQUIPMENT REQUIRED

Arduino Uno R3, Relay, bread board and jumper wires

DESCRIPTION

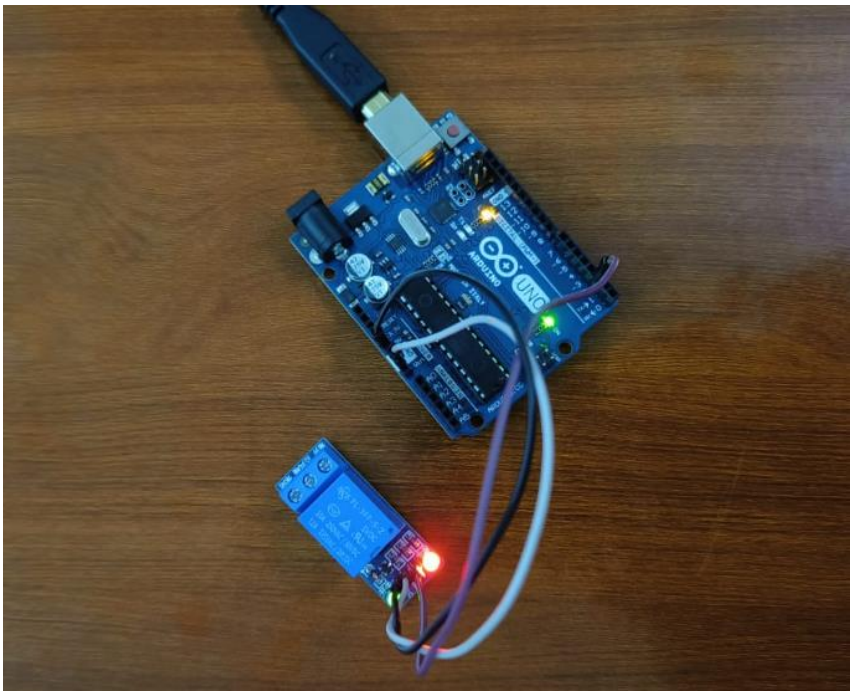
Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the cathode of the Relay on the same row where the wire from GND is placed. Connect the jumper wire from Pin 2 of the Arduino board to the anode of Relay in the breadboard. Finally connect 5V from Arduino board to the Relay's Vcc

PROGRAM

```
void setup()
{
	pinMode(2, OUTPUT);
}

void loop()
{
	digitalWrite(2, HIGH);
	delay(1000);
	digitalWrite(2, LOW);
	delay(1000);
}
```

 Reg no:

OUTPUT



RESULT

Reg no:

EX NO: 7 INSTALLATION OS IN RASPBERRY PI

DATE:

AIM

To install the desired OS into Raspberry Pi

EQUIPMENT REQUIRED

Raspberry Pi, Desktop / Laptop, Raspberry Pi Imager (Software), SD card

PROCEDURE

1. Download the Raspberry Pi Imager (software) and install it in your Laptop/Computer
2. Once installed, open the software and choose OS under Operating System section
3. Select the SD card to write the OS by clicking Choose Storage
4. Click write and pull out the SD card once the write is completed
5. Insert the SD card into Raspberry Pi
6. Use the username “pi” and password “raspberry” to login when booting up for the first time

RESULT

EX NO: 8 **PEAK MUSIC SENSING USING SOUND SENSOR**
DATE:

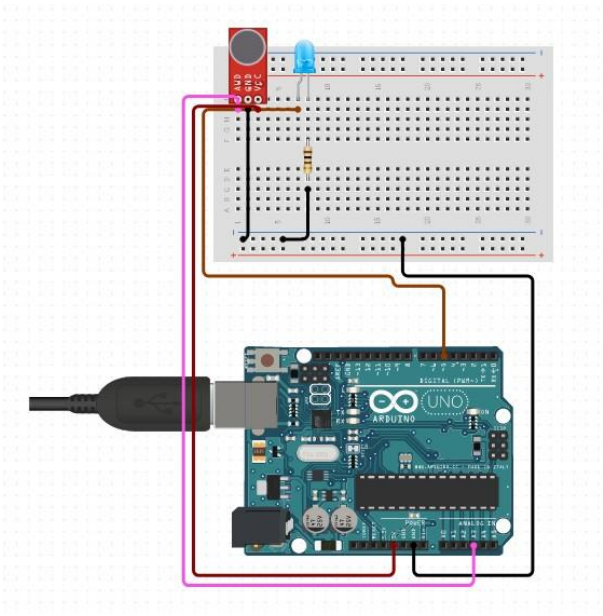
AIM

To write an application using arduino uno to sense the peak music from the sound sensor

EQUIPMENT REQUIRED

Arduino UNO, bread board, jumper wires, led resistor, sound sensor (microphone)

CIRCUIT DIAGRAM

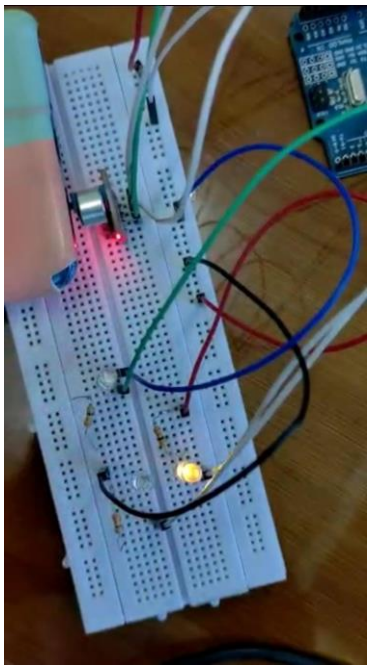


PROGRAM

```
#define LED 5
#define MIC    A3
void setup()
{
   pinMode(LED, OUTPUT);
   pinMode(MIC, INPUT);
}
void loop()
{
int input = analogRead(MIC);analogWrite(LED, input);
}
```


Reg no:

OUTPUT



RESULT

Reg no:

EX NO: 9 VIDEO SURVEILLANCE APPLICATION USING IOT

DATE:

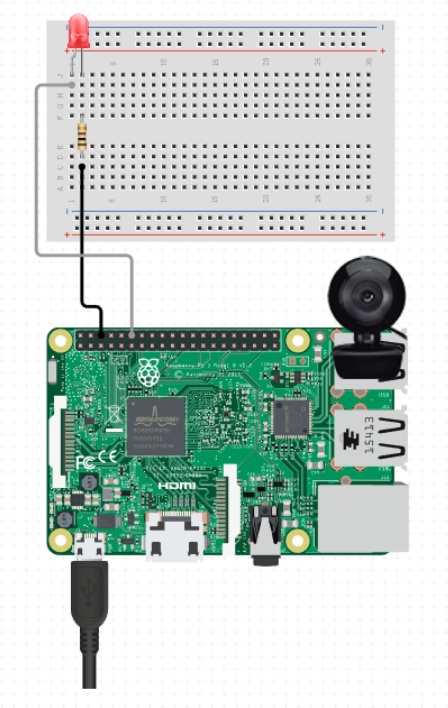
AIM:

To build an application using IoT to check whether the person in wearing a mask or not wearing a mask

EQUIPMENT REQUIRED

Raspberry Pi, USB Web Camera, LED, jumper wires, bread board

CIRCUIT DIAGRAM



DESCRIPTION

Make the connections like in the circuit diagram above. Import the necessary packages such as open cv to capture frames in the web camera video, tensorflow to import the trained model and predict the frame. Set PIN 8 to output mode and default state to low. Create an folder called “input” to store the captured image. In the loop, capture each frame using read method. Using haarcascade face detection model, detect the faces in the image. Use the coordinates of the first photo and obtain the face in the image. Store the cropped image in the “input” folder. Open the image and predict it with the trained model. If the predicted value is 0 i.e. person is wearing mask, then turn on the LED else turn off the LED

PROGRAM

```
import cv2
import os
import shutil
```

```
import RPi.GPIO as GPIO os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img , img_to_array
import numpy as np

# loading trained models
mask_model = load_model('model.h5')
face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# video feed from web cam
video = cv2.VideoCapture(0)

# creating folder to store images
if not os.path.exists("input"):
        os.makedirs("input")

# tracking total no. of frames
img_cnt = 0

# pin setup
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)

while True:
   ret, image = video.read()
   if not ret: break

   image = cv2.resize(image, (1280, 720), interpolation=cv2.INTER_AREA)
   gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   faces = face_model.detectMultiScale(gray_img, 1.1, 6)

  # detecting only the 1st face
  for (x, y, w, h) in faces[0]:
     org = (x-10,y-10)
     img_count += 1
     color_face = image[y:y+h, x:x+w]
     cv2.imwrite('input/%dface.jpg'%(img_count), color_face)
     img = load_img('input/%dface.jpg'%(img_count), target_size=(150, 150))
     img = img_to_array(img)
     img = np.expand_dims(img, axis=0)
     prediction = mask_model.predict(img)
     if predict == 0: # person is wearing mask
         GPIO.output(8, GPIO.HIGH)
     else: # person is not wearing mask
         GPIO.output(8, GPIO.LOW)
```

```
    k = cv2.waitKey(1)
    if k != -1: break

# cv2 clean up
video.release()
cv2.destroyAllWindows()

# deleting the input folder
shutil.rmtree("input")
```

OUTPUT

RESULT

Reg no:

EX NO: 10 **MINI PROJECT**

 Reg no: