CAS
IEEE CIRCUITS AND SYSTEMS SOCIETY

IoT

"IOTSPARK: IGNITING INNOVATION WORKSHOP"

By IEEE CASS

"Enhance your understanding and proficiency in IoT through an engaging and interactive workshop."

WE START ON NOVEMBER 17TH

Friday from 9:30 to 3:30 pm

Venue at: MJCET

SAVE YOUR SPOT! AT JUST 50/-

Contact :Shaista Daniya 9392840085

Exp-1

Traffic Light using Arduino

Aim: To implement a sample Traffic Light signal using Arduino UNO.

Description:

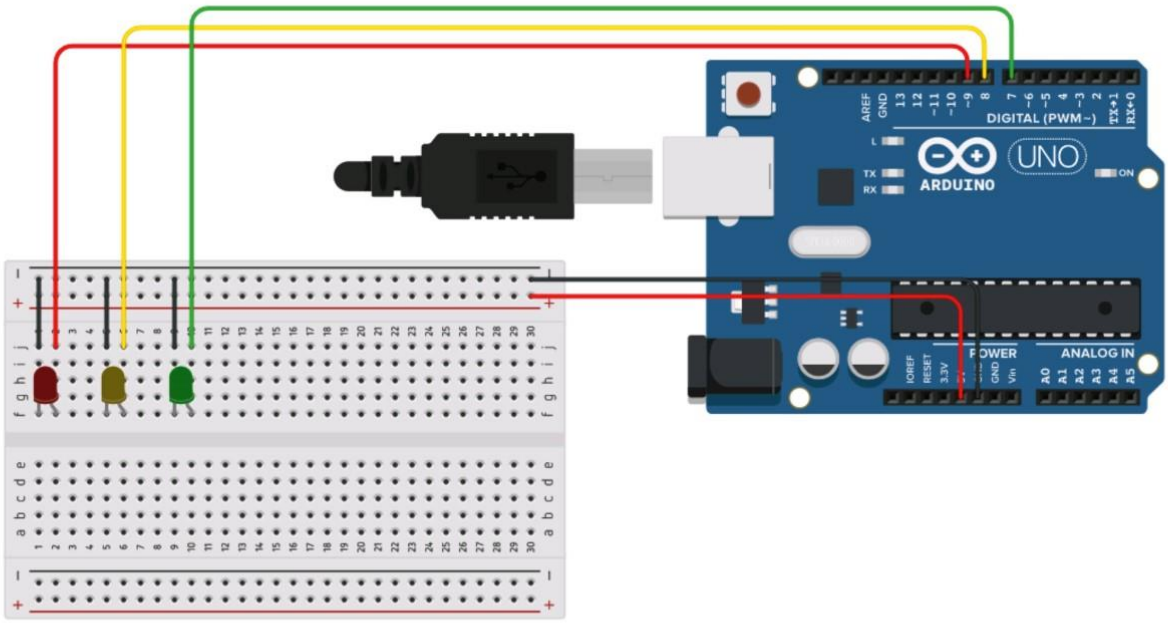
The LED have been powered by Arduino UNO (Board). It contains a code which uploaded to the board. And once it simulated LED Start's blinking like a traffic light.

15 Second for Red Light

6 Second for Yellow Light

20 Second for Green Light.

Circuit:



Procedure:

- Connect the **GND pin (Negative Pin)** of all led to **Pin GND** of Arduino.
- Connect Red LED **VCC Pin (Positive Pin)** to **Pin 9** of Arduino.
- Connect Yellow LED **VCC Pin (Positive Pin)** to **Pin 8** of Arduino.
- Connect Green LED **VCC Pin (Positive Pin)** to **Pin 7** of Arduino.

Code:

```
int red = 9;
int yellow = 8;
int green = 7;
void setup(){

  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);

}
void loop(){
digitalWrite(red, HIGH);
 delay(15000);
```

```
digitalWrite(red, LOW);


  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);


  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);


  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);


  digitalWrite(yellow, HIGH);
delay(1000);
```

```
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

digitalWrite(green, HIGH);
delay(20000);
digitalWrite(green, LOW);
//
digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
```

```
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);
    }
```

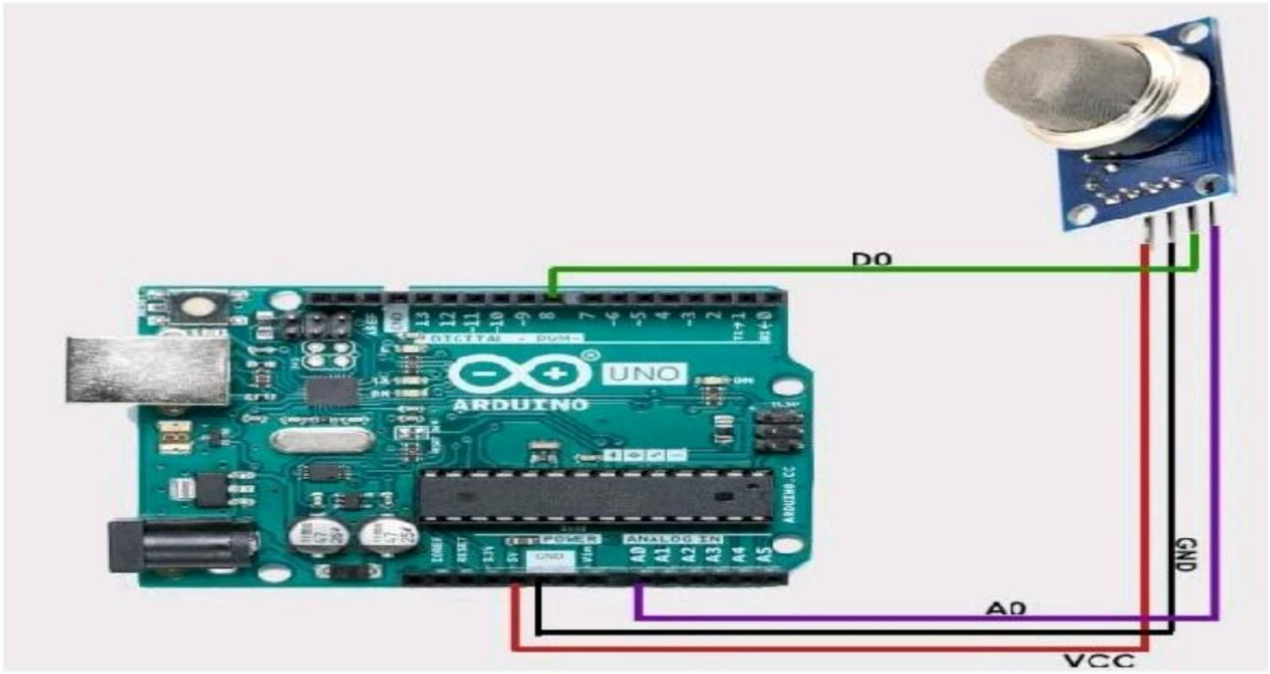
Exp-2

Arduino Interfacing with MQ-2 Gas Sensor Module

Aim:- To interface Arduino with MQ2 gas sensor module

Components:- Arduino uno,MQ2 gas sensor module,

Circuit:



Code:

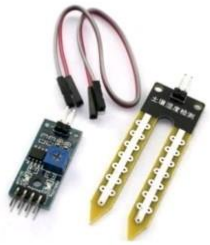
```
#define MQ2pin (0)
float sensorValue;  //variable to store sensor value
void setup()
{
 Serial.begin(9600); // sets the serial port to 9600
 Serial.println("Gas sensor warming up!");
 delay(20000); // allow the MQ-2 to warm up
}

void loop()
{
 sensorValue = analogRead(MQ2pin); // read analog input pin 0

 Serial.print("Sensor Value: ");
 Serial.print(sensorValue);
  if(sensorValue > 200)
 {
  Serial.print(" | Smoke detected!");
 }
  Serial.println("");
 delay(2000); // wait 2s for next reading
}
```

Exp-3

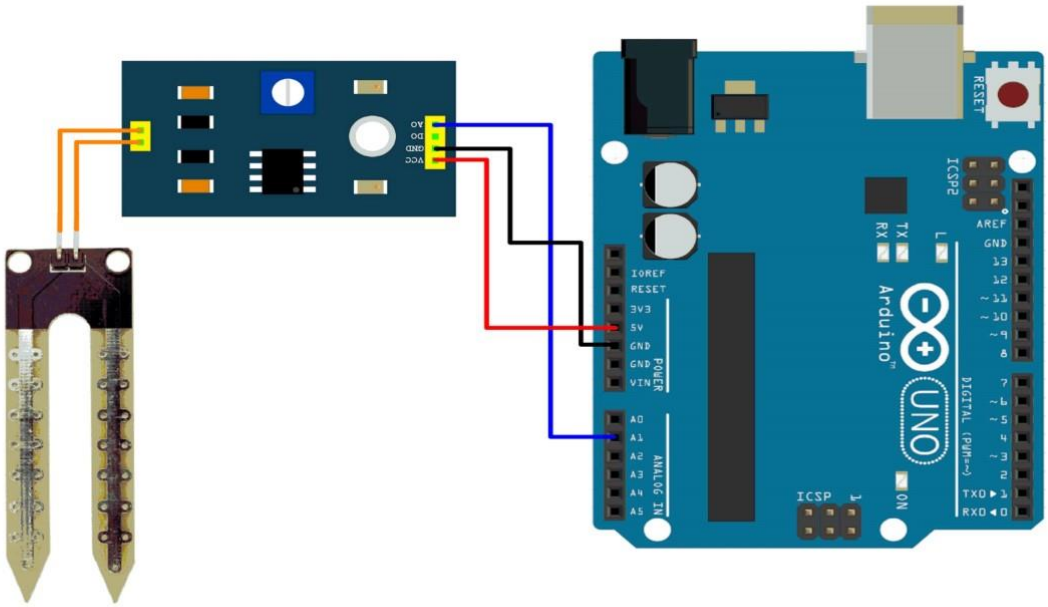
Interfacing Soil Moisture Sensor with Arduino UNO

Aim: To interface soil moisture sensor with Arduino UNO

Components:



Circuit:



Description :

Soil moisture is basically the content of water present in the soil. This can be measured using a soil moisture sensor which consists of two conducting probes that act as a probe. It can measure the moisture content in the soil based on the change in resistance between the two conducting plates.

The resistance between the two conducting plates varies in an inverse manner with the amount of moisture present in the soil.

Measure soil moisture using Arduino Uno:

Here, the analog output of soil moisture sensor is processed using ADC. The moisture content in terms of percentage is displayed on the serial monitor.

The output of the soil moisture sensor changes in the range of ADC value from 0 to 1023.

This can be represented as moisture value in terms of percentage using formula given below.

$$AnalogOutput = \frac{ADCValue}{1023}$$

Moisture in percentage = 100 – (Analog output * 100)

For zero moisture, we get maximum value of 10-bit ADC, i.e. 1023. This, in turn, gives 0% moisture.

Code:

```
const int sensor_pin = A1;  /* Soil moisture sensor O/P pin */


void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
}


void loop() {
  float moisture_percentage;
```

```
  int sensor_analog;
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
  Serial.print("Moisture Percentage = ");
  Serial.print(moisture_percentage);
  Serial.print("%\n\n");
  delay(1000);
}
```

Exp-4

Interfacing Servo motor with Arduino UNO

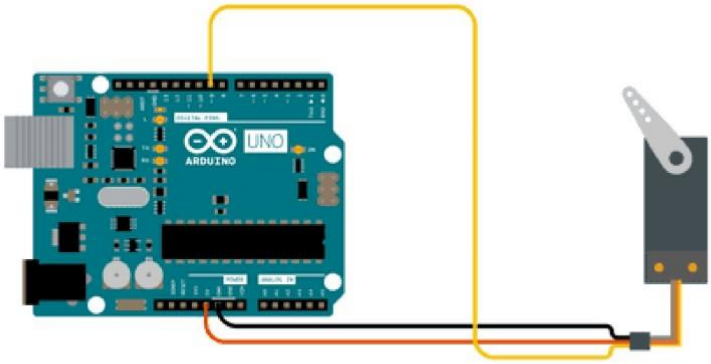
Aim: To Sweep the shaft of a RC Servo motor back and forth across 180 degrees.

Components: Arduino Board, Servo Motor ,hook-up wires

Circuit : Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to positive pole (+) of your power source. The ground wire is typically black or brown and should be connected to the negative pole (-) of your power source.

The signal pin is typically yellow or orange and should be connected to PWM pin on the board. Here, it is pin number 9.

Note: Always make sure to power your servo motor with a external power source. Connecting a servo directly to your board will cause your board to behave erratically and can damage your board



Code:

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(25);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(25);                       // waits 15ms for the servo to reach the position
  }
}
```

Exp-5

Interfacing DHT 11 Temperature and Humidity sensor with Arduino UNO

ranges and accuracy of the DHT11:

- Humidity Range: 20-90% RH
- Humidity Accuracy: ±5% RH
- Temperature Range: 0-50 °C
- Temperature Accuracy: ±2% °C
- Operating Voltage: 3V to 5.5V

HOW THE DHT11 MEASURES HUMIDITY AND TEMPERATURE

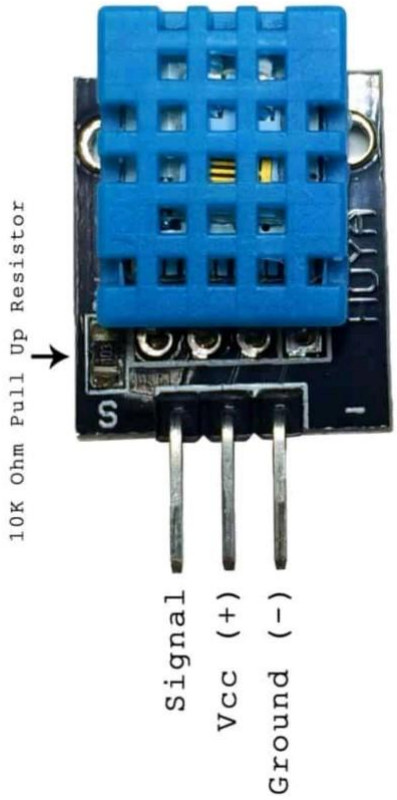
The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.

The DHT11 measures temperature with a surface mounted NTC temperature sensor (thermistor) built into the unit

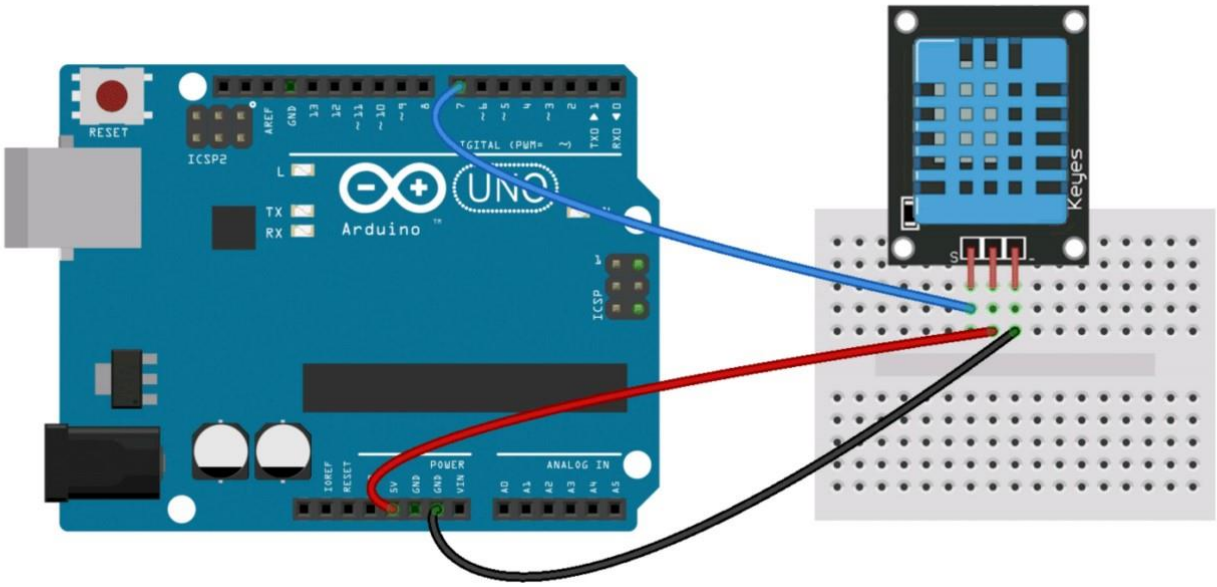
An IC mounted on the back of the unit converts the resistance measurement to relative humidity. It also stores the calibration coefficients, and controls the data signal transmission between the DHT11 and the Arduino:

The DHT11 uses just one signal wire to transmit data to the Arduino. Power comes from separate 5V and ground wires. A 10K Ohm pull-up resistor is needed between the signal line and 5V line to make sure the signal level stays high by default (see the datasheet for more info).

There are two different versions of the DHT11 you might come across. One type has four pins, and the other type has three pins and is mounted to a small PCB. The PCB mounted version is nice because it includes a surface mounted 10K Ohm pull up resistor for the signal line



Circuit:



DISPLAY HUMIDITY AND TEMPERATURE ON THE SERIAL MONITOR

Before you can use the DHT11 on the Arduino, you'll need to install the DHTLib library. It has all the functions needed to get the humidity and temperature readings from the sensor. It's easy to install, just download the DHTLib.zip file below and open up the Arduino IDE. Then go to Sketch>Include Library>Add .ZIP Library and select the DHTLib.zip file.

After it's installed, upload this example program to the Arduino and open the serial monitor:

```
#include <dht.h>

dht DHT;

#define DHT11_PIN 7

void setup(){
  Serial.begin(9600);
}

void loop(){
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(1000);
}
```

DISPLAY HUMIDITY AND TEMPERATURE ON AN LCD

A nice way to display the humidity and temperature readings is on a 16X2 LCD. To do this, first follow our tutorial on How to Set Up an LCD Display on an Arduino, then upload this code to the Arduino:

Code:

```
#include <dht.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
dht DHT;
#define DHT11_PIN 7

void setup(){
  lcd.begin(16, 2);
}

void loop(){
  int chk = DHT.read11(DHT11_PIN);
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(DHT.temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("Humidity: ");
  lcd.print(DHT.humidity);
  lcd.print("%");
  delay(1000);
}
```


THINGS


THINGS


THINGS


INTERNET OF
THINGS

Conclusion:

IoT Spark Workshop

The IEEE Circuit and System Society's workshop on IoT Spark, conducted on November 17, 2023, was a resounding success, leaving a lasting impact on participants. Our esteemed lecturers, serving as speakers, delivered engaging sessions that provided valuable insights into the intricate world of IoT.

The workshop focused on imparting essential knowledge of Arduino through a series of hands-on experiments. Utilizing the versatile Arduino IDE software, students were equipped with practical skills, fostering a deeper understanding of the core concepts.

This dynamic learning experience not only expanded participants' knowledge but also ignited a passion for innovation in circuit and system design. As we document the success of this event, we express our gratitude to all contributors and eagerly anticipate the continued exploration and application of IoT concepts by the inspired participants.