Interfacing LED with Arduino

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

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

- LED (13) → Arduino (13)
- GND → Arduino GND

Traffic light with LED interfacing using Arduino UNO

```
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(10, HIGH);
  delay(2000);
  digitalWrite(10, LOW);
  delay(1000);
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
  digitalWrite(8, HIGH);
  delay(2000);
  digitalWrite(8, LOW);
  delay(1000);
}
```

- Red LED → Arduino (10)
- Yellow LED → Arduino (9)
- Green LED → Arduino (8)
- GND → Arduino GND

LED with pushbutton using Arduino UNO

```
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
```

```
  }
}
```

- LED (13) → Arduino (13)
- Button → Arduino (2)
- GND → Arduino GND

Interfacing Ultrasonic Sensor With Arduino

```
const int pingPin = 7;
const int echoPin = 8;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  long duration, inches, cm;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  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);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

- VCC → Arduino 5V
- Trig → Arduino (7)
- Echo → Arduino (8)
- GND → Arduino GND

Interfacing Buzzer with Arduino

```
#define BUZZER 8

void setup() {
  pinMode(BUZZER, OUTPUT);
}

void loop() {
  tone(BUZZER, 85);
  delay(1000);
  noTone(BUZZER);
  delay(1000);
}
```

- Buzzer (+) → Arduino (18)
- GND → Arduino GND

//Melody note with buzzer

```
#define NOTE_C4 262
#define NOTE_G3 196
#define NOTE_A3 220
#define NOTE_B3 247

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    noTone(8);
  }
}

void loop() {
}
```

- Buzzer (+) → Arduino (18)
- GND → Arduino GND

Interfacing RGB Full Color LED with Arduino

```
int redpin = 11;
int bluepin = 10;
int greenpin = 9;
int val;

void setup() {
  pinMode(redpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  for(val = 255; val > 0; val--) {
    analogWrite(redpin, val);
    analogWrite(bluepin, 255 - val);
    analogWrite(greenpin, 128 - val);
    Serial.println(val);
    delay(1);
  }
  for(val = 0; val < 255; val++) {
    analogWrite(redpin, val);
    analogWrite(bluepin, 255 - val);
    analogWrite(greenpin, 128 - val);
    Serial.println(val);
    delay(1);
  }
}
```

- Red (R) → Arduino (11)
- Green (G) → Arduino (9)

- Blue (B) → Arduino (10)
- GND → Arduino GND

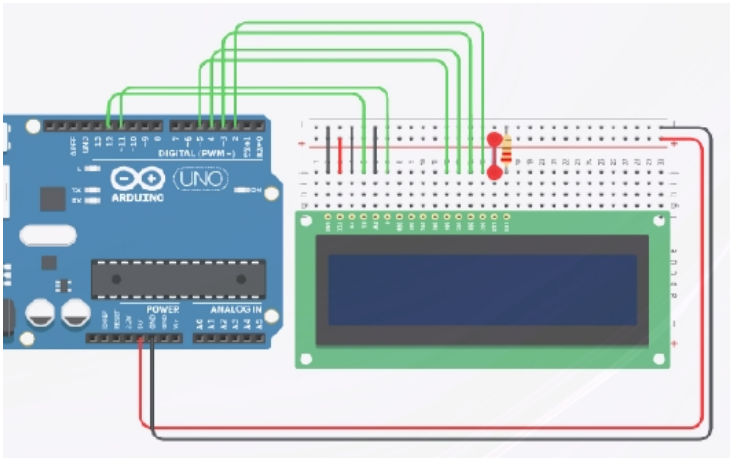
Temperature and Humidity Sensor

```
#include <dht.h>
#define DHT11_PIN 7
dht DHT;

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

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

- VCC → Arduino 5V
- Data → Arduino (7)
- GND → Arduino GND

LCD Interfacing with Arduino UNO

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

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

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Hello");
  delay(3000);
  lcd.setCursor(0, 1);
  lcd.print("Good Morning");
}
```

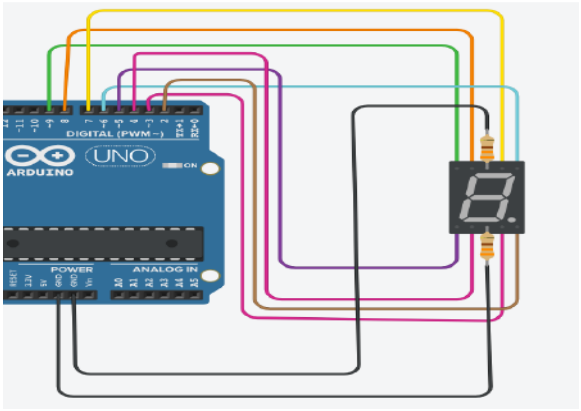



- RS → Arduino (12)
- EN → Arduino (11)
- D4 → Arduino (5)
- D5 → Arduino (4)
- D6 → Arduino (3)
- D7 → Arduino (2)
- VCC → Arduino 5V
- GND → Arduino GND

seven segment display

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
```

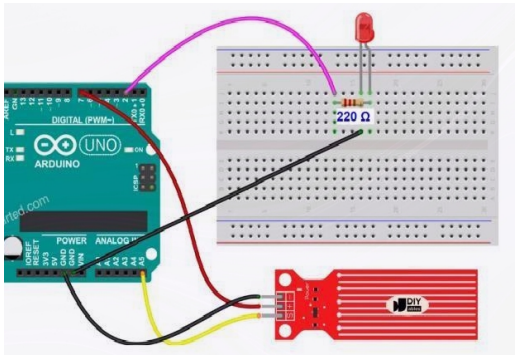
```
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  delay(1000);

}
```



- A → Arduino (7)
- B → Arduino (6)
- C → Arduino (3)
- D → Arduino (4)
- E → Arduino (5)
- F → Arduino (8)
- G → Arduino (9)
- DP → Arduino (2)
- com → Arduino GND

Interfacing Water Level Sensor with Arduino Uno

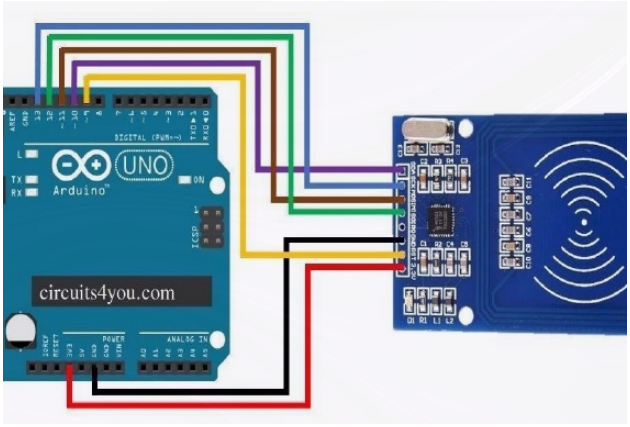
```
#define LED_PIN 2
#define water_sensor 7
#define SIGNAL_PIN A5
#define THRESHOLD 300

int value = 0;

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
  pinMode(water_sensor, OUTPUT);
  digitalWrite(water_sensor, LOW);
  digitalWrite(LED_PIN, LOW);
}

void loop() {
  digitalWrite(water_sensor, HIGH);
  delay(10);
  value = analogRead(SIGNAL_PIN);
  digitalWrite(water_sensor, LOW);

  Serial.print("The water level is : ");
  Serial.println(value);

  if (value > THRESHOLD) {
    Serial.println("The water is detected");
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
}
```



- VCC → Arduino 5V
- Signal → Arduino (A5)
- LED (Indicator) → Arduino (2)
- GND → Arduino GND

Interfacing RFID With Arduino Uno

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN 9
#define SS_PIN 10

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
  Serial.begin(9600);
  while (!Serial);
  SPI.begin();
  mfrc522.PCD_Init();
  delay(4);
  mfrc522.PCD_DumpVersionToSerial();
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks "));
}

void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```




- SDA → Arduino (10)
- SCK → Arduino (13)
- MOSI → Arduino (11)
- MISO → Arduino (12)
- GND → Arduino GND
- RST → Arduino (9)

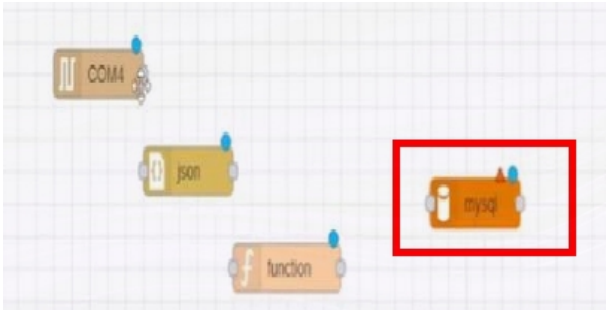
Ultrasonic sensor data collection using Arduino and MYSQL

```
const int pingPin = 7;
const int echoPin = 8;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  long duration, inches, cm;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  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);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

Code to be written in function pallet:

```
var value = JSON.parse(JSON.stringify(msg.payload)); //converting string to
// JSON object
value = msg;
var sensor1 = msg.payload.inches; //adding value to the payload
var sensor2 = msg.payload.cm; //adding value to the payload
msg.payload = [sensor1, sensor2]; //adding value to the payload
msg.topic = 'INSERT INTO distance(inches,cm) values (?,?);'; //query to insert
return msg;
```




Ultrasonic Sensor Data Collection with Arduino and MySQL

1. **MySQL Database Setup**:
    - Install XAMPP (Apache + MySQL) and phpMyAdmin.
    - Create a database `test` and a table `distance` with columns for sensor data (e.g., `inches`, `cm`).
2. **Node-RED Installation**:
    - Install Node.js, then Node-RED.
    - Use command: `npm install -g --unsafe-perm node-red`.
    - Run Node-RED via command line and access the web interface.
3. **Node-RED Configuration**:
    - Install the following Node-RED nodes: `node-red-node-mysql`, `node-red-node-serialport`.
    - Use pallets to transfer sensor data to MySQL.
4. **Ultrasonic Sensor Arduino Code**:
    - Measure distance using an ultrasonic sensor and convert data into inches and centimeters.
    - Send the data in JSON format to Node-RED for database insertion.

- VCC → Arduino 5V
- Trig → Arduino (7)
- Echo → Arduino (8)
- `GND → Arduino GND`

5. **Node-RED Flow Configuration**:
    - **Serial In Node**: Connect Arduino to Node-RED and set baud rate.
    - **JSON Node**: Convert data between JSON string and object.
    - **Function Node**: Write a function to parse the data and prepare an SQL query.
    - **MySQL Node**: Configure the connection to the MySQL database, including database name and credentials.

    Function Node Example:

    ```
    var value = JSON.parse(JSON.stringify(msg.payload));
    var sensor1 = msg.payload.inches;
    var sensor2 = msg.payload.cm;
    msg.payload = [sensor1, sensor2];
    msg.topic = 'INSERT INTO distance(inches, cm) values (?, ?);';
    return msg;
    ```

6. **Deploy and Verify**:
    - Connect all nodes, deploy the flow, and check the MySQL database for successful data insertion.
    - Ensure the connection is successful by confirming the "OK" status in Node-RED.

This process allows data from an ultrasonic sensor to be collected and stored in a MySQL database using Arduino, Node-RED, and MySQL.