equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller.

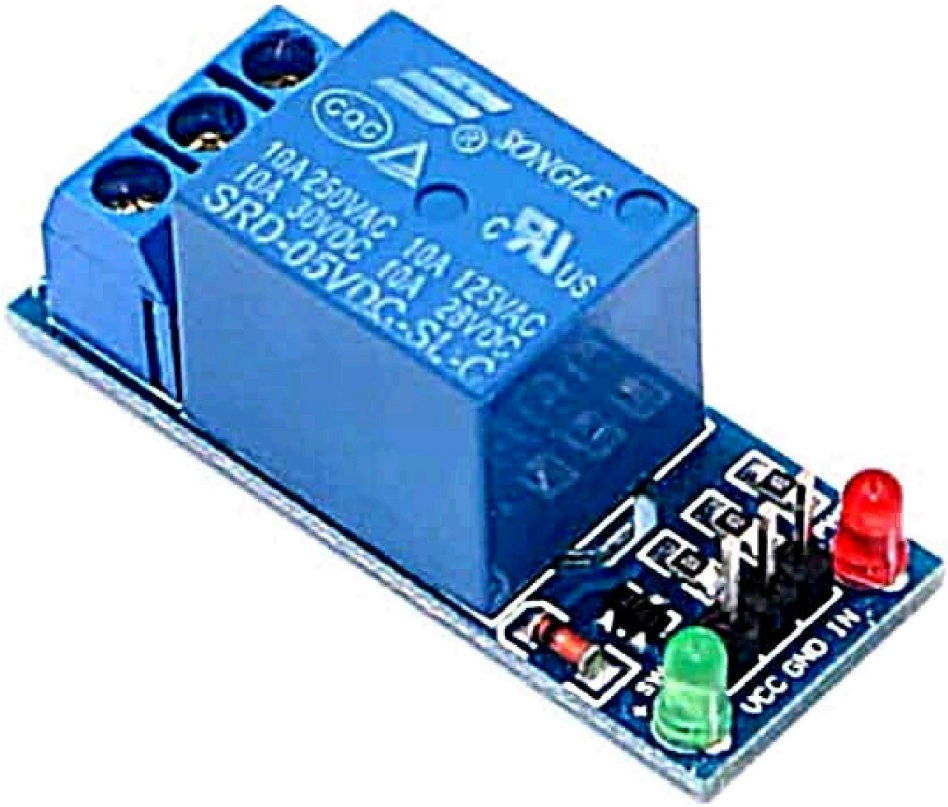
Principle

When the signal port is at low level, the signal light will light up and the opt coupler 817c (it transforms electrical signals by light and can isolate input and output electrical signals) will conduct, and then the transistor will conduct, the relay coil will be electrified, and the normally open contact of the relay will be closed. When the signal port is at high level, the normally closed contact of the relay will be closed. So you can connect and disconnect the load by controlling the level of the control signal port.

Features :

- Size: 75mm (Length) * 55mm (Width) * 19.3mm (Height)
- Weight: 61g
- PCB Colour: Blue
- There are four fixed screw holes at each corner of the board, easy for install and fix. The diameter of the hole is 3.1mm
- High quality Single relay is used with single pole double throw, a common terminal, a normally open terminal, and a normally closed terminal
- Optical coupling isolation, good anti-interference.
- Closed at low level with indicator on, released at high level with indicator off
- VCC is system power source, and JD_VCC is relay power source. Ship 5V relay by default. Plug jumper cap to use
- The maximum output of the relay: DC 30V/10A, AC 250V/10A

5Volts Relay



A relay is nothing more than a remote switch that uses an electromagnet to close a set of contact points. Relays are often used in circuits to reduce the current that flows through the primary control switch. A relatively low amperage switch, timer, or sensor can be used to turn a much higher capacity relay on and off. As relay diagrams show, when a relay contact is normally open (NO), there is an open contact when the relay is not energized. The other side has three low voltage pins (Ground, VCC, and Signal) which connect to the Arduino. Inside the relay is a 120-240V switch that's connected to an electromagnet. When the relay receives a HIGH signal at the signal pin, the electromagnet becomes charged and moves the contacts of the switch open or closed.

This is a 5V 4-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is

NodeMCU Usage:

This sensor module uses 2 digital pins (D1 and D2) on the NodeMCU for Trig and Echo, and does not necessarily have to be pins 1 and 2. The code for measuring distance with NodeMCU is given as:

```
// Generic Code for the working of UltraSonic Sensor  connected to the NodeMcu (used in the project)

// defines pins numbers

const int trigPin = 2;  //D4

const int echoPin = 0;  //D3

// defines variables

long duration;

int distance;

void setup () {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(9600); // Starts the serial communication

}

void loop() {

// Clears the trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2);
```

Code:

```
if(liters<=waterLevelLowerThreshold)
      waterLevelDownCount++;
    else waterLevelDownCount=0;

    if(liters>=waterLevelUpperThreshold)
      waterLevelUpCount++;
    else waterLevelUpCount=0;
    waterLevel.publish(liters);

    if(waterLevelDownCount==2)
    {
//TURN ON RELAY
      Serial.println("motor turned on");
      digitalWrite(MOTOR_CONTROL_PIN,LOW);//Relay is active LOW
    }
    if(waterLevelUpCount==5)
    {
//TURN OFF RELAY
      Serial.println("motor turned off");
      digitalWrite(MOTOR_CONTROL_PIN,HIGH);//Relay is active HIGH
    }
}
```

```
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);

// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
delay(2000);
}
```

```
void runPeriodicFunc()
{
  static const unsigned long REFRESH_INTERVAL1 = 1000; // 1sec
  static unsigned long lastRefreshTime1 = 0;
  if(millis() - lastRefreshTime1 >= REFRESH_INTERVAL1)
  {
   measure_Volume();
   lastRefreshTime1 = millis();
  }
}
```