Relay on an Arduino

Fig 1: SRD-05VDC-SL-C 5V relay

This 5V-relay will be exhibited in the following few test circuits. Think of as a switch that is controlled by a signal as to be open or closed.

The first circuit that will be tested will only have an Arduino, and a pushbutton switch theoretically connecting to a pull up resistor. However, since we are going to use a push button switch on a board that already has that resistor, we only need to make the required connections to the Arduino board to check the status of the switch based on the controlling signal.

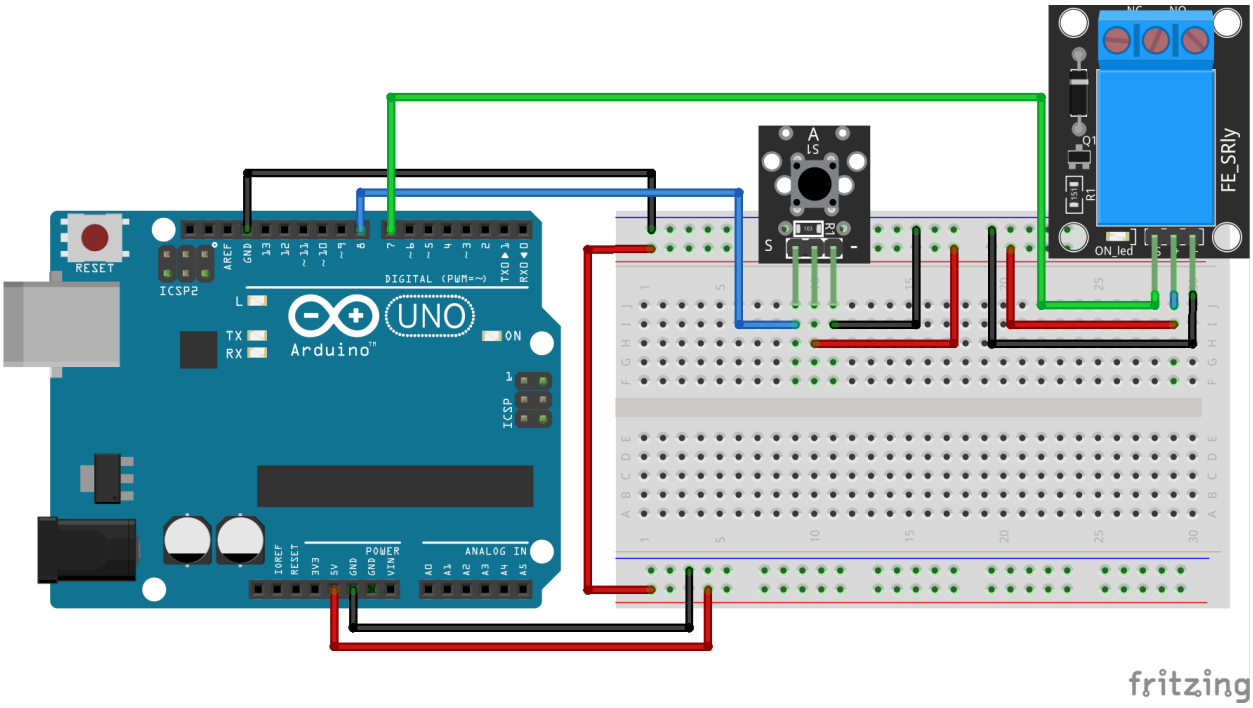
The **SRD-05VDC-SL-C** relay has three high voltage terminals (NC, C, and NO) which connect to the device you want to control. The other side has three low voltage pins (Ground, Vcc, and Signal) which connect to the Arduino.

- T1: NO, Normally Open 120-240V terminal, not connected to the C terminal (unless the relay is activated). Infinite resistance between T1 and C when the relay is not active

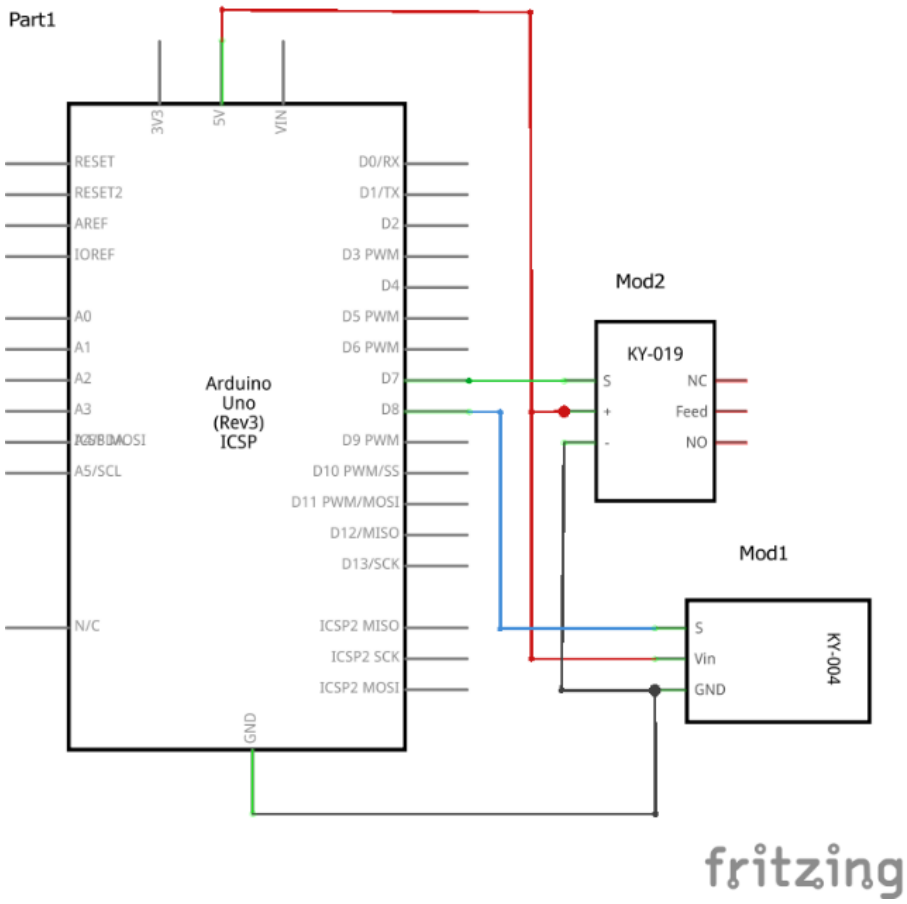
- T2: NC, Normally Closed 120-240V terminal, connected to the C terminal (unless the relay is activated). zero resistance between T1 and C when the relay is **not** active
- C: Common terminal
- Ground: Connects to the ground pin on the Arduino
- 5V Vcc: Connects the Arduino's 5V pin
- Signal: Carries the trigger signal from the Arduino that activates the relay

For the last three terminals, refer to the Fritzing circuit to locate those pins.

1. **Simple pushbutton switch circuit**



In this circuit, a pushbutton momentary switch already connected to a pull up resistor, has its signal pin connected to digital pin 8 so that the Arduino can acquire its status (key pressed or not).



Code

```
#define pinb 8
int pinOut = 7;
void setup(){
  Serial.begin(9600);
  pinMode(7, OUTPUT);
  pinMode(pinb, INPUT);
  Serial.begin(9600);
}

void loop()
{
```

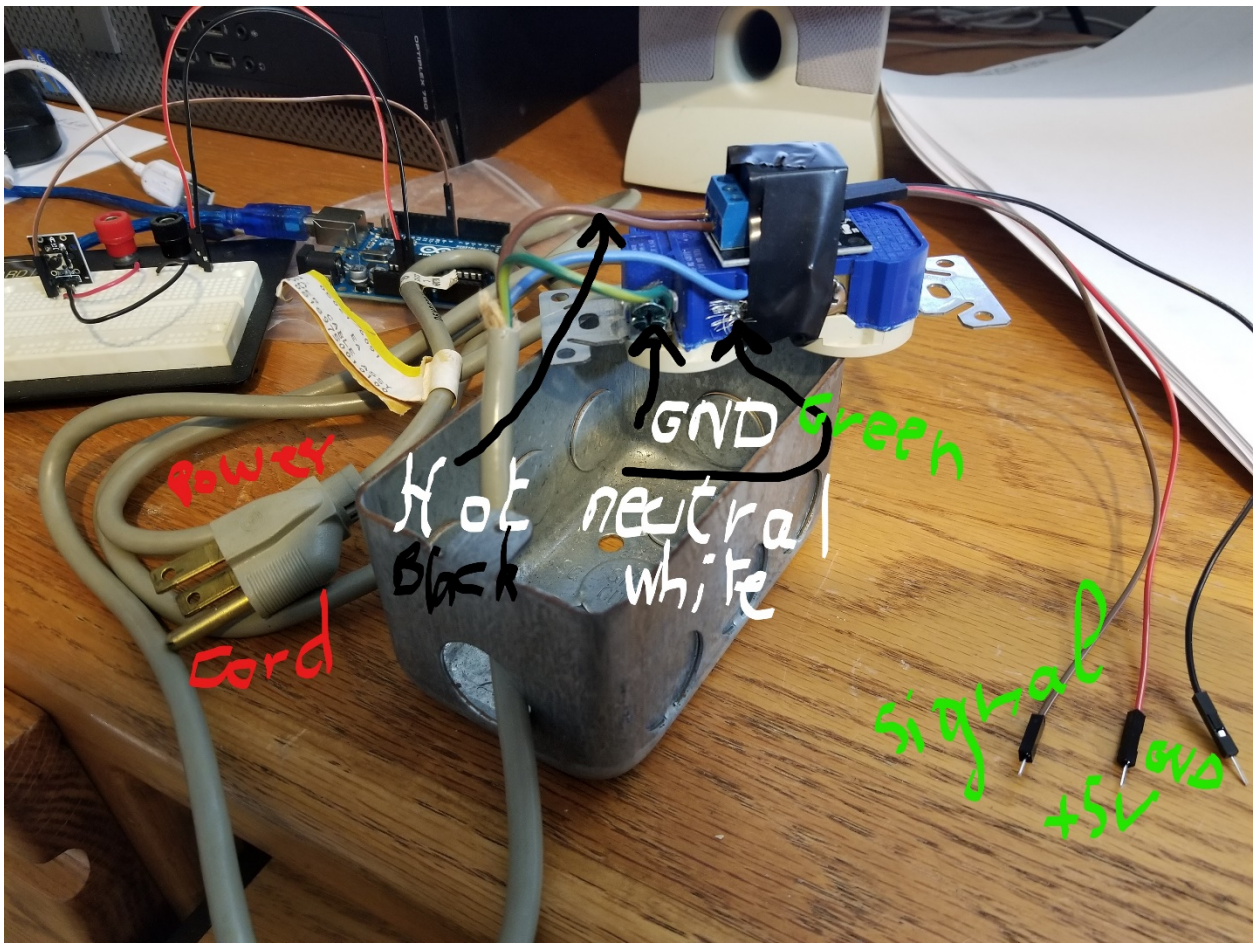
```
  int stateb=digitalRead(pinb);
  Serial.println(stateb);
  delay(100);
 if (stateb==0) {
    digitalWrite(pinOut, HIGH);
    }
  else {
    digitalWrite(pinOut, LOW);
    }
  delay(100);
}
```

Through the code, we find out that the status reads **one** when the key is **not pressed**, and **zero** when the key is **pressed**. This is shown through the serial monitor, except trust the claim that the key was pressed when the output was 0.

In this case, we are going to activate a relay when pressing the pushbutton switch. The if statement in the code tests the status of the button, and if zero (key pressed), the relay signal will be high (outputting a high on pin 7 connected to the relay signal pin), thereby activating the relay. In this case, T1 and C will be connected, and the current will flow between the lines connected to these two terminals.

2. Switch-Controlled Outlet

We are going to connect an outlet to this relay, and allow the outlet to be powered only when the button is pressed.



Safety is the mother of all priorities when it comes to high power. Always work on the devices with the power off, or better yet, no element connected to power, either the main power line or a powered Arduino. With standard outlets having a wide slot and a narrow slot (for safety reasons), the neutral line is connected to the screw on the side of the wide slot. That is also the reason switches, breakers are connected in series with the hot wire. An opening of the switch will shut down the power from reaching a device, and safer to touch.

It is clear from this picture that the color standards for power line are not universal. In the US, the power cord should have 3 lines:

- Black: Hot line (most harmful line to touch if powered). In this picture brown
- White: Neutral line. In this picture blue
- Green: Earth ground. In this picture striped green

Connections related to the power cord and the relay:

- Neutral line of power cord connected to the screw on the side of the wide slot
- Ground wire of power cord connected to the ground screw (see picture)
- Hot wire of power cord connected to the C (middle) terminal of the relay (see Fig 1)

One final additional connection: connect a black wire from the T1 terminal (NO, see Fig 1) to the screw of the outlet on the side of the narrow slot. When the relay is activated, the two black wires will be connected, providing power to whichever device will be plugged into this outlet.

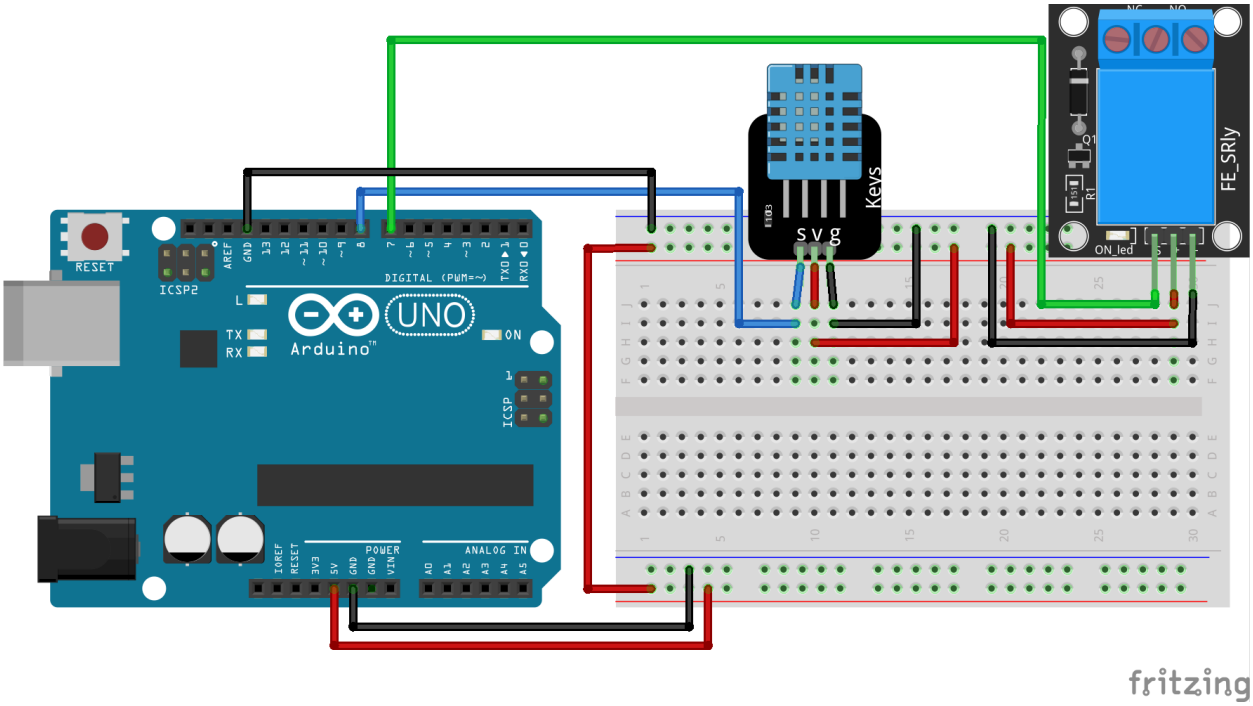
Connections related to the relay and Arduino:

- + pin of relay connected to +5V of Arduino
- - pin or GND connected to GND of Arduino
- Signal pin of relay connected in this case to digital output 7 of Arduino

As a test circuit, a light bulb was connected to the outlet. Every time, I press the momentary pushbutton, the light bulb would light up, and when I let go, it would turn off.

3. Temperature-controlled fan

In this circuit, we will be using a DHT11 sensor to read the ambient temperature, and the code will be such the outlet will be activated every time the temperature exceeds 25˚C for example. In that case, a fan that is plugged in the outlet will turn on trying to cool the room to temperatures below 25˚C, or at least make it more bearable to be there.



We can see that the DHT11 is pin-for-pin compatible with the momentary pushbutton switch. The only difference is in the code where we have to include the DHT11 library, read the ambient temperature in degrees Celsius, compare it to 25˚C, as an example, and then activate the relay if it exceeds that temperature so that the fan can turn on, and make the room more comfortable to stay in

Code

```
#include <dht.h>  // include DHT11 Library

dht DHT;

#define pintemp 8  //  Signal pin of DHT11 connected to digital output 8
int pinOut = 7; // Relay signal pin connected to digital output 7

void setup(){
  Serial.begin(9600);
  pinMode(pinOut, OUTPUT);
  pinMode(pintemp, INPUT);
 }

void loop()
{
  delay(5000);
  int chk = DHT.read11(pintemp);
  Serial.print("Temperature = ");
  Serial.print(DHT.temperature);
  Serial.print("  Humidity = ");
  Serial.print(DHT.humidity);
  if (DHT.temperature <= 25){
    digitalWrite(pinOut, LOW);  // No power to the fan
    Serial.print("  status of pinOut= ");
    Serial.println(LOW);
  }
  else {
```

```
    digitalWrite(pinOut, HIGH); // activate relay to power the fan

    Serial.print("  status of pinOut= ");

    Serial.println(HIGH);

  }


  delay(100);

}
```

```
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 21.00  Humidity = 33.00  status of pinOut= 0
Temperature = 20.00  Humidity = 32.00  status of pinOut= 0
Temperature = 20.00  Humidity = 32.00  status of pinOut= 0
Temperature = 20.00  Humidity = 32.00  status of pinOut= 0
```

One of the most useful things you can do with an Arduino is use it to control higher voltage electronic devices. Any device you normally plug into a wall outlet can be activated by a sensor or controlled in other ways by the Arduino. The possibilities are endless considering the variety of sensors and modules available to us today.