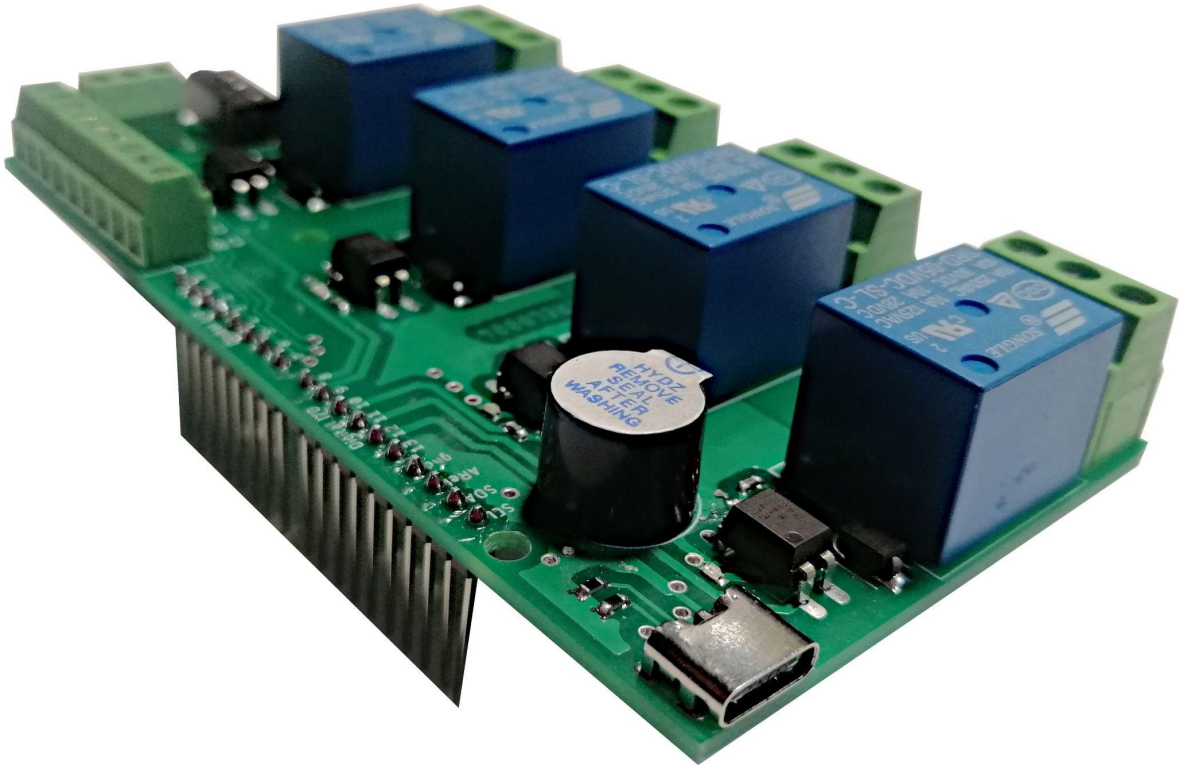


Shield de relevadores para Arduino 4 canales 10A/(125-250)VAC V1

El shield de relevadores de 4 canales para tarjetas tipo Arduino, incluye relevadores modelo SRD-05VDC-SLC ópticamente aislados y con fuente independiente, el cual proporciona una capacidad de carga de hasta 10A. La tarjeta de relevadores incorpora un conector USB-C que actúa como fuente de alimentación independiente del Arduino. Además cuenta con un buzzer activo conectado al pin D12. Adicionalmente cuenta con una entrada tipo jack de 3.5mm para un sensor de temperatura DS18B20 y terminales tornillos para el resto de pines no usados.

Especificaciones

- Canales: 4.
- Voltajes de carga:
  - 250 VAC.
  - 125 VAC.
  - 30 VDC.
  - 28VDC.
- Capacidad de corriente de carga: 10A.
- Pines de conexión con los relevadores: D11, D10, D9 y D8.
- Pin para sensor DS18B20: D3.
- Pin para buzzer. D12.

Contenido:

- Tarjeta shield relevador de 4CH.
- 2 separadores con tornillo m3.

- No incluye eliminador de 5V. Se vende por separado.

Compatibilidad:

- Arduino UNO R3.
- Arduino MEGA R3.

Esquemático

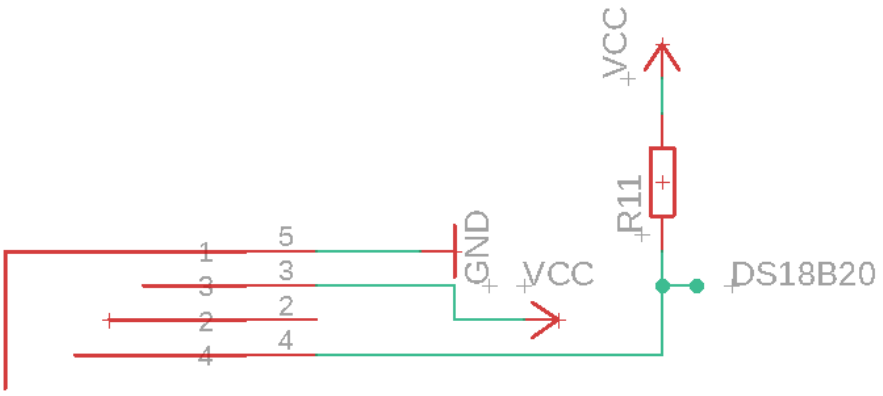


Figura 1. Esquemático de conexión con un sensor DS18B20 conectado mediante el jack de 3.5mm al pin D3.

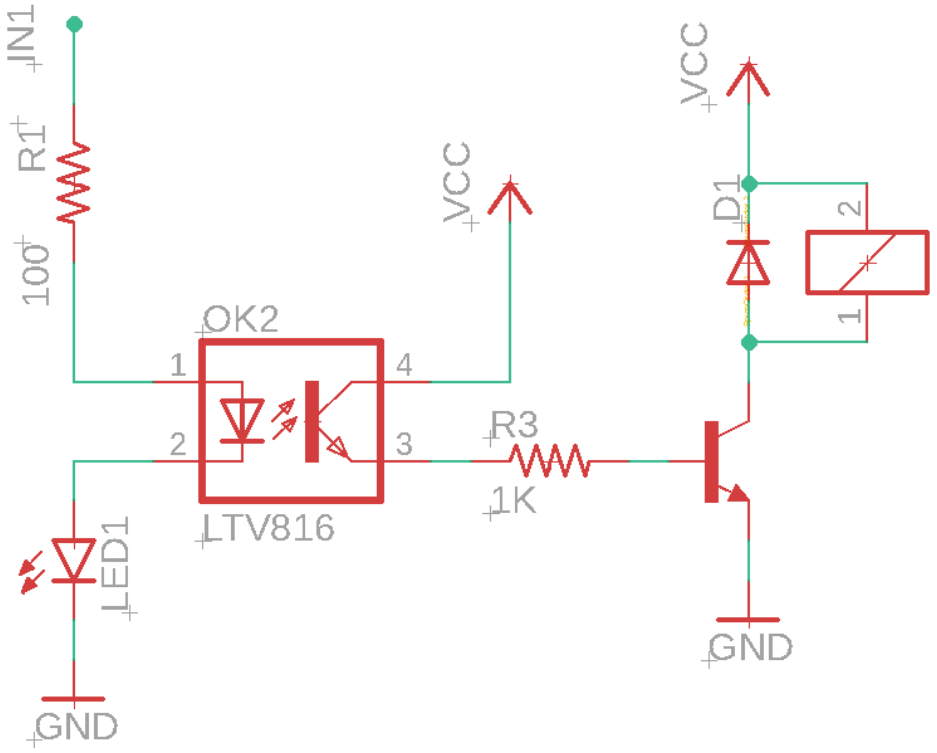
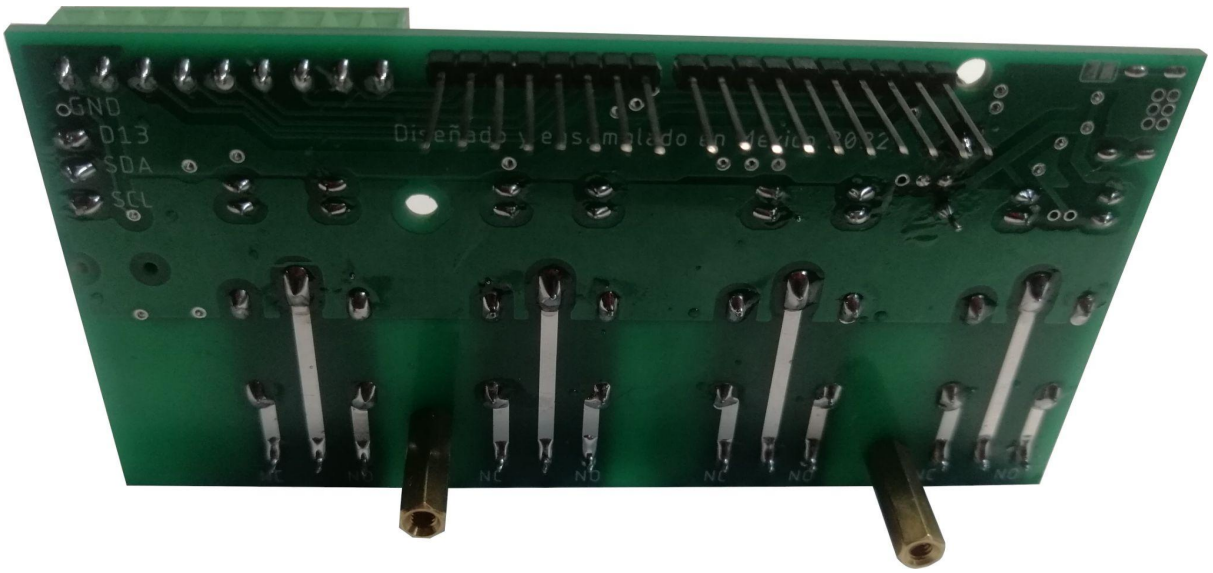
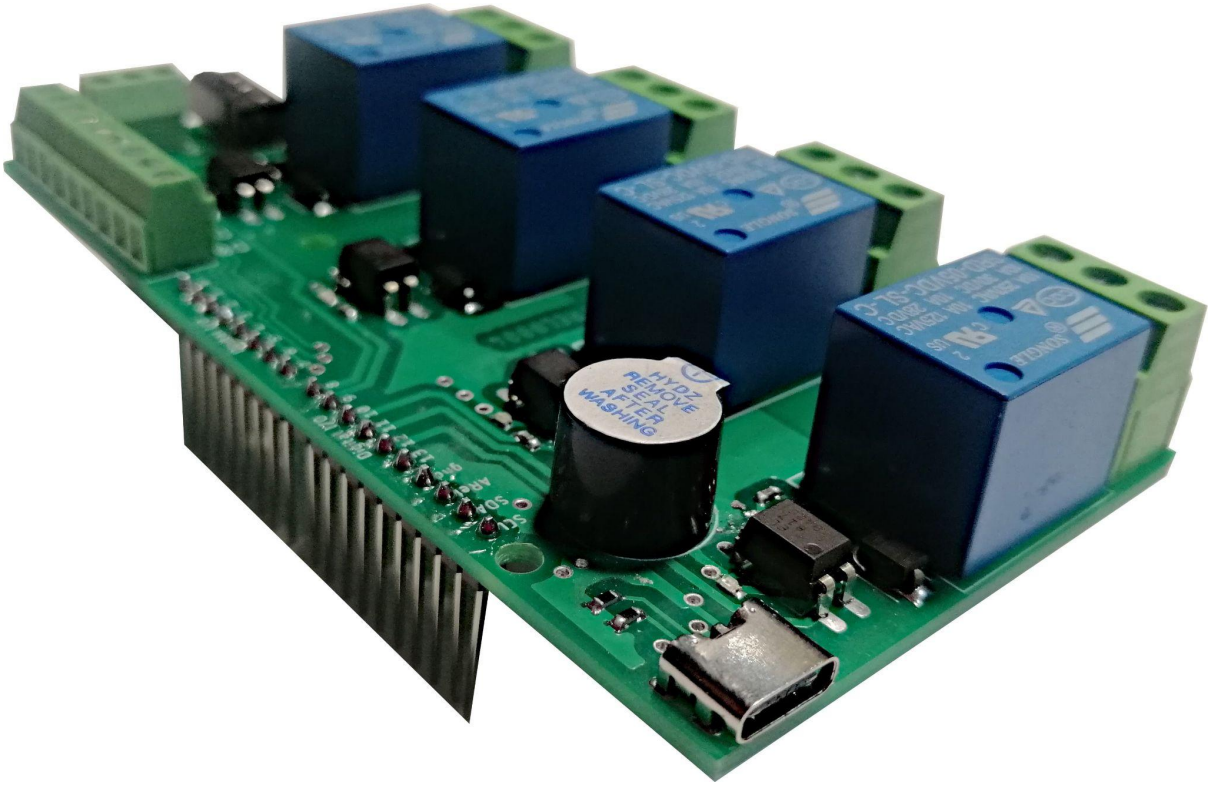


Figura 2. Circuito del relevador ópticamente aislado.


GND
D13
SDA
SCL
NC
NO
NC
NO
NC
NO
NC
NO


HYDZ
REMOVE
SEAL
AFTER
WASHING

Código de ejemplo:

https://gist.github.com/esmarr58/ec8d58d305b8ca31e0791257c3d94af6

Código de ejemplo

```
#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// http://www.pjrc.com/teensy/td_libs_OneWire.html
//
// The DallasTemperature library can do all this work for you!
// https://github.com/milesburton/Arduino-Temperature-Control-Library

OneWire  ds(3);  // on pin 10 (a 4.7K resistor is necessary)

void setup(void) {
  Serial.begin(9600);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(11, OUTPUT);
 pinMode(12, OUTPUT);
   digitalWrite(12, HIGH);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

 digitalWrite(8, HIGH);
 delay(500);
  digitalWrite(9, HIGH);
  delay(500);
   digitalWrite(10, HIGH);
   delay(500);
    digitalWrite(11, HIGH);
  delay(500);
 digitalWrite(12, LOW);
  delay(150);
  digitalWrite(12, HIGH);
  delay(150);


  Serial.println();
```

```
// the first ROM byte indicates which chip
switch (addr[0]) {
  case 0x10:
    Serial.println("  Chip = DS18S20");  // or old DS1820
    type_s = 1;
    break;
  case 0x28:
    Serial.println("  Chip = DS18B20");
    type_s = 0;
    break;
  case 0x22:
    Serial.println("  Chip = DS1822");
    type_s = 0;
    break;
  default:
    Serial.println("Device is not a DS18x20 family device.");
    return;
}

ds.reset();
ds.select(addr);
ds.write(0x44, 1);        // start conversion, with parasite power on at the end

delay(1000);     // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();
ds.select(addr);
ds.write(0xBE);         // Read Scratchpad


// Convert the data to actual temperature
// because the result is a 16 bit signed integer, it should
// be stored to an "int16_t" type, which is always 16 bits
// even when compiled on a 32 bit processor.
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3; // 9 bit resolution default
  if (data[7] == 0x10) {
    // "count remain" gives full 12 bit resolution
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
} else {
  byte cfg = (data[4] & 0x60);
  // at lower res, the low bits are undefined, so let's zero them
  if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
  else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
  else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
  //// default is 12 bit resolution, 750 ms conversion time
}
celsius = (float)raw / 16.0;
fahrenheit = celsius * 1.8 + 32.0;
Serial.print("  Temperature = ");
```

```
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");
   digitalWrite(8, LOW);
  digitalWrite(9, LOW);
   digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    delay(1000);
}
```