I SENSORI DI PROSSIMITA'

In questa lezione su Arduino, parleremo di un interessante trasduttore, che può essere utile per tantissimi progetti: il sensore di presenza o di prossimità

*I **sensori di prossimità** (chiamati anche **proximity**) sono dei sensori in grado di rilevare la presenza di oggetti nelle immediate vicinanze del "lato sensibile" del sensore stesso, senza che vi sia un effettivo contatto fisico. La distanza entro cui questi sensori rilevano oggetti è definita portata nominale (o campo sensibile). Alcuni modelli dispongono di un sistema di regolazione per poter calibrare la distanza di rilevazione. L'assenza di meccanismi d'attuazione meccanica, e di un contatto fisico tra sensore e oggetto, fa sì che questi sensori presentino un'affidabilità elevata.*

Dal punto di vista circuitale, un sensore di presenza, ha 3 PIN.

VCC: Per l'alimentazione

OUT: Ha un valore 1 o 0 logico, a seconda della presenza di un oggetto in movimento

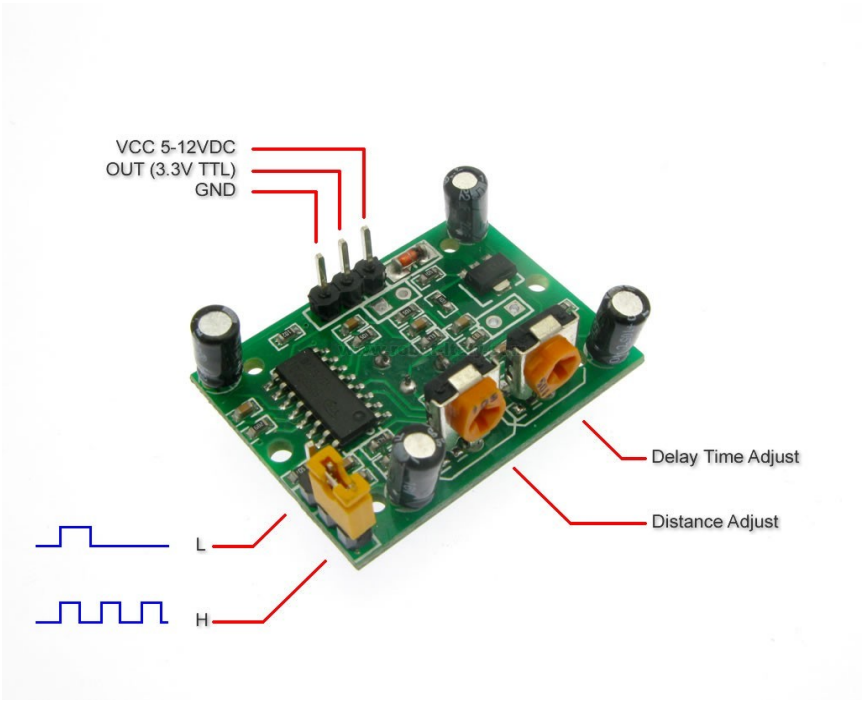
Ground

Ma vediamo come utilizzare il sensore PIR per monitorare la presenza di oggetti in movimento, creare un piccolo sistema d'allarme oppure accedere una luce.

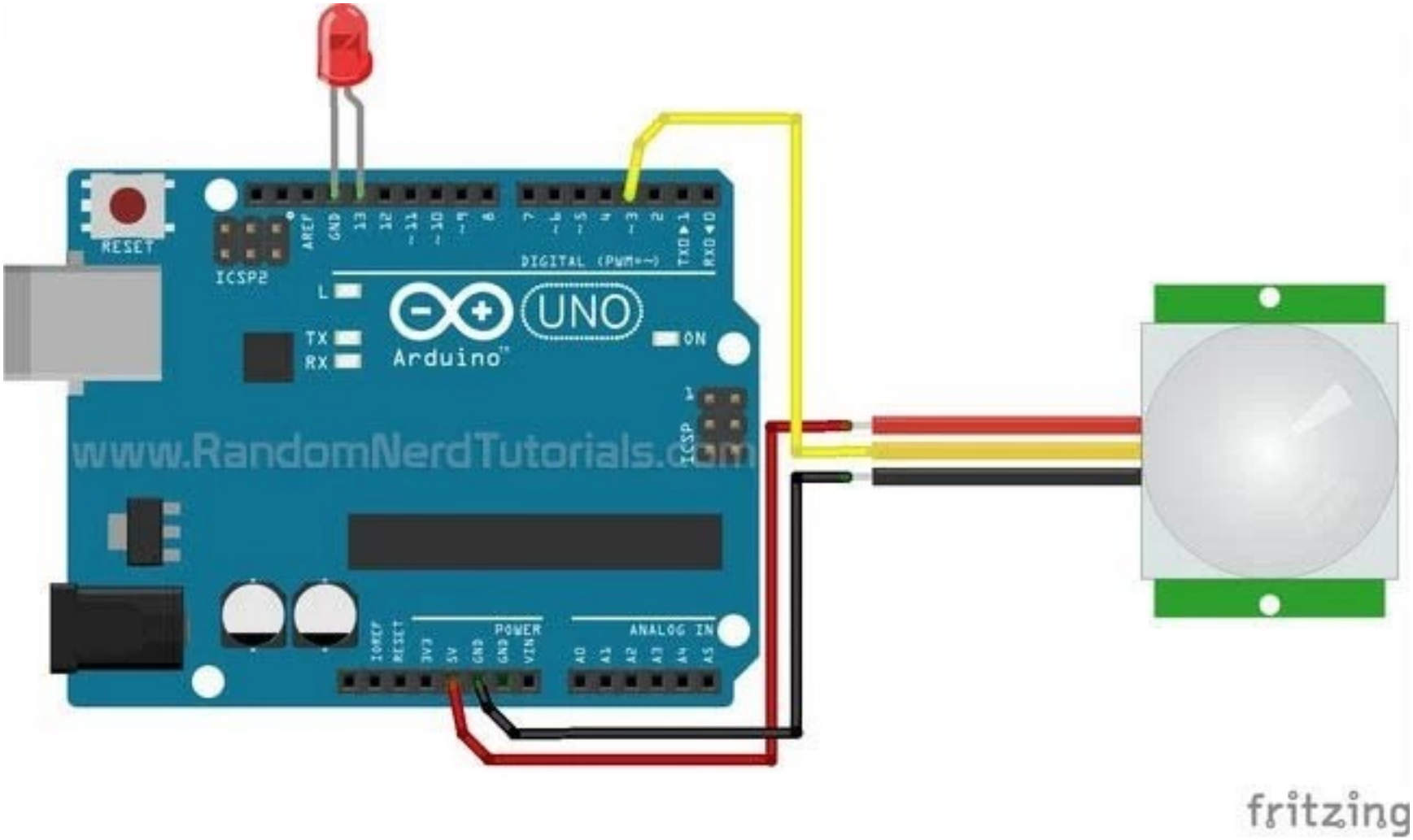
1° Programma:

Rilevare la presenza di un oggetto in movimento e accendere un LED

I collegamenti, per questo programma sono abbastanza semplici. Basta collegare i cavi del sensore PIR, come riportati in figura.



Da notare, una interessante caratteristica, è quella di poter aumentare il periodo in cui l'uscita sia ad un livello logico alto, regolando la manopola di destra. In questo caso, non si fa altro che variare una resistenza, che fa aumentare il tempo in cui l'uscita sia ad 1 logico.



Guarda il video : https://www.youtube.com/watch?v=vJgtckLzoKM

```
int led = 13;                // the pin that the LED is atteched to
int sensor = 3;              // the pin that the sensor is atteched to
int state = LOW;             // by default, no motion detected
int val = 0;                 // variable to store the sensor status (value)

void setup() {
  pinMode(led, OUTPUT);      // initalize LED as an output
  pinMode(sensor, INPUT);    // initialize sensor as an input
  Serial.begin(9600);        // initialize serial
}
```

```
void loop(){
  val = digitalRead(sensor);   // read sensor value
  if (val == HIGH) {           // check if the sensor is HIGH
    digitalWrite(led, HIGH);   // turn LED ON
    delay(100);                // delay 100 milliseconds

    if (state == LOW) {
      Serial.println("Motion detected!");
      state = HIGH;            // update variable state to HIGH
    }
  }
  else {
      digitalWrite(led, LOW); // turn LED OFF
      delay(200);             // delay 200 milliseconds

      if (state == HIGH){
        Serial.println("Motion stopped!");
        state = LOW;           // update variable state to LOW
    }
  }
}
```

2° Programma:

Rilevare la presenza di un oggetto in movimento e accendere un LED con calibrazione

```
/* Programam che utilizza il sensore di presenza PIR HC-SR501, per rilevare la presenza di un oggetto in movimento e accende un LED, collegato al PIN 13*/

// Tempo di calibrazione del sensore
int calibrationTime = 30;

//Il tempo in cui l'uscita sia bassa
long unsigned int lowIn;

// valore di millisecondi, per cui si ritiene che ci sia "quiete"
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = 3;    //il PIN di Arduino a cui è collegato il sensore
int ledPin = 13;   //il PIN a cui è connesso il LED
```

```
// Impostazione del sensore
void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(pirPin, LOW);

  //Fase di calibrazione
  Serial.print("calibrating sensor ");
    for(int i = 0; i < calibrationTime; i++){
      Serial.print(".");
      delay(1000);
      }
    Serial.println(" done");
    Serial.println("SENSOR ACTIVE");
    delay(50);
  }
```

```
void loop(){

  // Questo IF permette di stabilre se il sensore rileva un oggetto in movimento
   if(digitalRead(pirPin) == HIGH){
     digitalWrite(ledPin, HIGH);   /Accendiamo il LED
     if (lockLow){
        lockLow = false;
      Serial.println("---");
      Serial.print("motion detected at ");
      Serial.print(millis()/1000);
      Serial.println(" sec");
      delay(50);
      }
      takeLowTime = true;
    }
```

Millis() è una funzione del core di Arduino che restituisce il numero di millisecondi trascorsi dall'istante in cui si fornisce l'alimentazione alla scheda. Millis() si basa su di un contatore che viene aggiornato continuamente da un interrupt agganciato al timer 0, e per questo motivo il suo valore cresce costantemente, non essendo influenzato dal codice dell'utente.
Possiamo usare millis() per programmare l'esecuzione di alcune azioni, come se fosse un timer che dà la sveglia al nostro codice senza per questo bloccare l'esecuzione del medesimo in attesa che arrivi il momento giusto.

```
  // Questo IF permette di stabilire se non c'è più nessun movimento
  if(digitalRead(pirPin) == LOW){
   digitalWrite(ledPin, LOW);  //Si spegne il LED

   if(takeLowTime){
    lowIn = millis();
    takeLowTime = false;
    }

   if(!lockLow && millis() - lowIn > pause){

      lockLow = true;
      Serial.print("motion ended at ");      //output
      Serial.print((millis() - pause)/1000);
      Serial.println(" sec");
      delay(50);
      }
   }
}
```

3° Programma proposto:

Rilevare la presenza di un oggetto in movimento e eccitare un relè per azionare una ventola a 12 V DC

Riportare
- **Il diagramma a blocchi del sistema**
- **lo schema elettrico**
- **il flow-chart del programma**
- **il programma**