```
void loop() {
  int sig = analogRead(TEMP_PIN);
  float R = ((1023.0/(float)sig) - 1.0)*R0;
  float log_sig = log(R/R0);
  float T = 1/((log_sig/B) + (1/298.15));
  float temp = T - 273.15;
  Serial.println("Temperature now: " + String(temp));

  if (temp>25.0 and temp<=30.0){
    int n = temp - 25.0;
    current_speed = n*tacca;
    analogWrite(FAN_PIN, (int) current_speed);
  }
  else if (temp>30.0){
    current_speed = 255;
    analogWrite(FAN_PIN, (int) current_speed);
  }
  else{
    current_speed = 0;
    analogWrite(FAN_PIN, (int) current_speed);
  }
  delay(3000);
}
```

ESERCIZIO 1:
Obiettivo: azionare la ventola sulla base della temperatura misurata dal sensore

Componenti hardware utilizzate: breadboard, sensore di temperatura, PWM fan.

Componenti software utilizzate: GPIO, segnali PWM

Sketch: in un primo momento, abbiamo misurato la temperatura, come fatto nell'esercizio del laboratorio precedente. In un secondo momento, abbiamo determinato il set-point della velocità della ventola, facendo la differenza tra la temperatura corrente e il valore di temperatura minimo e moltiplicando questo per la "*tacca*", cioè il valore massimo diviso l'ampiezza del range di temperatura.

ESERCIZIO 2:

```
void loop() {
  int sig = analogRead(TEMP_PIN);
  float R = ((1023.0/(float)sig) - 1.0)*R0;
  float log_sig = log(R/R0);
  float T = 1/((log_sig/B) + (1/298.15));
  float temp = T - 273.15;
  Serial.println("Temperature now: " + String(temp));

  if (temp>=15.0 and temp<=20.0){
    int n = 20.0 - temp;
    brightness = n*tacca;
    analogWrite(RLED_PIN, brightness);
  }
  else if (temp<15){
    brightness = 255;
    analogWrite(RLED_PIN, brightness);
  }
  else{
    brightness = 0;
    analogWrite(RLED_PIN, brightness);
  }
  Serial.println("Brightness: " + String(brightness));
  delay(3000);
```

Obiettivo: accendere il LED rosso sulla base della temperatura misurata dal sensore.

Componenti hardware utilizzate: breadboard, sensore di temperatura, led.

Componenti software utilizzate: GPIO, segnali PWM

Sketch: in un primo momento, abbiamo misurato la temperatura come fatto nell'esercizio del laboratorio precedente. In un secondo momento, abbiamo determinato il set-point del riscaldamento del LED, facendo la differenza tra la il valore di temperatura massimo e la temperatura corrente. Questa leggera variazione, a differenza dell'esercizio precedente, permette di raggiungere la massima intensità in corrispondenza del valore minore di temperatura e la minima intensità in corrispondenza del valore maggiore di temperatura.

Inoltre, per controllare la luminosità del LED usiamo i PIN che emettono un segnale PWM, ovvero un'onda quadra che permette di pilotare un LED o un motore con una tensione media dipendente dal duty cycle.

ESERCIZIO 3:

es3

```
const int PIR_PIN = 7;
const int timeout_pir = 30000*60;
unsigned long start_millis = 0;


void checkPresence(){
  int value = digitalRead(PIR_PIN);
  if (value == HIGH){
    Serial.println("Person Detected");
    start_millis = millis();

  }
}

void setup() {
  // put your setup code here, to run once:
  pinMode(PIR_PIN, INPUT);
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Lab 2.3 Starting");
  attachInterrupt(digitalPinToInterrupt(PIR_PIN), checkPresence, CHANGE);
}

void loop() {
    unsigned long now_millis = millis();
    if(now_millis - start_millis >= timeout_pir){
      Serial.println("Non ci sono più persone nella stanza");
    }
    else{
      Serial.println("Ci sono ancora persone nella stanza");
    }
    delay(1000);

}
```

Obiettivo: monitorare la presenza di persone nella stanza.

Componenti hardware utilizzate: sensore PIR.

Componenti software utilizzate: interrupt

Sketch: nella funzione *setup(),* oltre al settaggio dei PIN e all'avvio della comunicazione seriale, viene definita la funzione *attachInterrupt()*, nella quale, oltre a specificare il pin che vogliamo monitorare, dobbiamo specificare il nome della funzione da eseguire quando si scatena l'interruzione e il modo in cui l'interruzione deve essere interpretata. *Quando il sistema potrà assumere che non ci sono più persone nella stanza?* Per individuare questo istante, abbiamo creato una finestra temporale mobile che tiene conto dei movimenti verificatosi nella stanza negli ultimi 30 minuti. Questa viene realizzato facendo uso della funzione *millis()*: nel momento in cui viene rilevato un movimento, memorizzo l'istante in cui il movimento stesso è stato rilevato nella variabile *start_millis*. Di conseguenza, nella funzione *loop()* controllo in ogni istante se il tempo trascorso dall'ultimo rilevamento (*now_millis – start_millis*) è maggiore o uguale a *timeout_pir*. Solo in questo caso posso assumere che non ci sono più persone nella stanza.

ESERCIZIO 4:

es4

```
#include <TimerOne.h>
const int SND_PIN = 7;
int n_sound_events = 2;
volatile int tot_count = 0;
long sound_interval = 10000;
volatile unsigned long start_millis = 0;

void checkPresence() {
  int value = digitalRead(SND_PIN);
  if (value == LOW) {
    Serial.println("Person Detected by Noise Sensor"); //stampa di verifica
    if (start_millis - millis() >= 2000) { //NB: start_millis a questo punto
      tot_count += 1;                      //è ancora il valore precedente
    } //distinguo due rumori solo se distanziati di 2 sec.
    start_millis = millis();
  }
}

void setup() {
  pinMode(SND_PIN, INPUT);
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Lab 2.4 Starting");
  attachInterrupt(digitalPinToInterrupt(SND_PIN), checkPresence, CHANGE);
}

void loop() {
  unsigned long now_millis = millis();
  if (now_millis - start_millis >= sound_interval) {
    if (tot_count < n_sound_events) {
      Serial.println("Non ci sono ancora persone nella stanza");
    }
    tot_count = 0;
  }
  else{
    if (tot_count >= n_sound_events) {
      Serial.println("Ci sono ancora persone nella stanza");
    }
  }
  delay(1000);
}
```

Obiettivo: individuare la presenza di individui nella stanza.

Componenti hardware utilizzate: il sensore di rumore nella zona d'ombra del sensore PIR (sono combinati nell'esercizio 5).

Sketch: anche in questo caso abbiamo gestito il rilevamento di persone tramite interrupt, quindi, come nell'esercizio precedente, nella funzione *setup(),* oltre al settaggio dei PIN e all'avvio della comunicazione seriale, viene definita la funzione *attachInterrupt()*.

Quando il sistema potrà assumere che non ci sono più persone nella stanza? Come nell'esercizio precedente, abbiamo realizzato una finestra temporale mobile e quindi possiamo assumere che non ci sono persone nella stanza nel momento in cui il tempo trascorso dall'ultimo rilevamento è maggiore di *timeout_sound.* La particolarità di questo esercizio consiste nel fatto che l'assunzione della presenza di persone nella stanza può essere fatta solo se i rumori rilevati sono distanziati nel tempo, per evitare di considerare la presenza di persone a seguito di rumori occasionali.

Per realizzare ciò nella funzione *checkPresence()*, ogni qualvolta viene rilevato un rumore, facciamo la differenza tra l'istante in cui è stato rilevato l'ultimo rumore e l'istante corrente. Soltanto se questa differenza è maggiore di 2 secondi (valore casuale) si può assumere che i due rumori sono rumori distinti e, di conseguenza, considerare la presenza di persone nella stanza.

Inoltre, nella *loop()* allo scadere del *sound_interval* possiamo assumere che non ci sono persone nella stanza soltanto se il numero di rumori rilevati è minore di *n_sound_events.* Invece, se il numero di rumori supera *n_sound_events*, possiamo assumere che ci sono persone nella stanza.

ESERCIZIO 5:

es5

```
const int PIR_PIN = 7;
const int SND_PIN = 8;
volatile int tot_count = 0;
int n_sound_events = 50; //50 eventi
const int timeout_pir = 1800000; //30 minuti
const long sound_interval = 600000; //10 minuti
volatile unsigned long start_millis1 = 0;
unsigned long start_millis2 = 0;


void checkPresence() {
  int value = digitalRead(PIR_PIN);
  if (value == HIGH) {
    Serial.println("Person Detected by PIR Sensor");
    start_millis1 = millis();
    Serial.println(start_millis1);
  }
}

void setup() {
  pinMode(PIR_PIN, INPUT);
  pinMode(SND_PIN, INPUT);
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Lab 2.5 Starting");
  attachInterrupt(digitalPinToInterrupt(PIR_PIN), checkPresence, CHANGE);
}
void loop() {

  int value = digitalRead(SND_PIN);
  if (value == LOW) {
    Serial.println("Person Detected by Noise Sensor"); //stampa di verifica
    if (start_millis2 - millis() >= 5000) { //NB: start_millis2 a questo punto è ancora il valore precedente
      tot_count += 1;
    } //distinguo due rumori solo se distanziati di 5 sec.
    start_millis2 = millis();
  }

  unsigned long now_millis = millis();
  if (now_millis - start_millis1 >= timeout_pir) { //NESSUNO SI MUOVE
    if (now_millis - start_millis2 >= sound_interval) { //NESSUNO PARLA
      if (tot_count < n_sound_events) { //ANCHE SE C'E' STATO UN RUMORE SPORADICO
        Serial.println("Non ci sono persone nella stanza");
      }
      tot_count = 0;
    }
    else { //QUALCUNO PARLA
      if (tot_count >= n_sound_events) { // E PARLA ABBASTANZA
        Serial.println("Ci sono persone nella stanza");
      }
    }
  }

  else { //QUALCUNO SI MUOVE
    if (start_millis1 != 0) {
      Serial.println("Ci sono persone nella stanza");
    }
  }
  delay(1000);
}
```

Obiettivo: combinare le due misurazioni dell'esercizio 3 e 4 in modo da assumere la presenza di persone nella stanza se almeno uno dei due sensori rileva una presenza.
Componenti hardware utilizzate: sensore PIR, sensore di rumore, porta seriale
Componenti software utilizzate: interrupt, GPIO

Sketch:
nella *setup()* configuriamo in input i pin a cui abbiamo collegato i due sensori e colleghiamo la funzione *checkPresence()* a un qualsiasi evento (CHANGE) riscontrato sul PIR_PIN, cioè gestiamo con interrupt le rilevazioni del sensore di movimento. Da questo momento, quando lui rileverà un movimento (valore HIGH letto dal PIR_PIN), la variabile start_millis1 sarà aggiornata al valore corrispondente al tempo corrente di esecuzione.
Nella *loop()* gestiamo, invece, le rilevazioni da parte del sensore di rumore, introdotto per supplire ai limiti del sensore PIR, le sue zone d'ombra. A intervalli di un secondo leggiamo il valore digitale sul SND_PIN e, se c'è stata una rilevazione (LOW), la variabile start_millis2 sarà aggiornata al valore corrispondente al tempo corrente di esecuzione. Nel caso in cui la rilevazione precedente sia avvenuta più di cinque secondi prima, incrementiamo il contatore di movimenti.
Se il sensore PIR non ha più fatto rilevazioni per un tempo superiore a timeout_pir e quello di movimento, passata la sua finestra temporale sound_interval, non ha raggiunto il numero minimo di rilevazioni per confermare una effettiva presenza (ci possono essere stati rumori sporadici): posso dire che non c'è nessuno nella stanza. Se, invece, le rilevazioni del sensore di movimento sono in numero sufficiente o c'è stata almeno una rilevazione del sensore PIR, allora c'è qualcuno nella stanza.

```
void led(float temp) {
  if (temp >= temp_min_led and temp <= temp_max_led) {
    int n = temp_max_led - temp;
    brightness = n * tacca;
    analogWrite(RLED_PIN, brightness);
    Serial.println("Brightness:" + String(brightness)); //stampa di verifica
  }

  else if (temp < temp_min_led) {
    analogWrite(RLED_PIN, 255);
    brightness = 255;
    Serial.println("Brightness:" + String(brightness)); //stampa di verifica
  }
  else {
    analogWrite(RLED_PIN, 0);
    brightness = 0;
    Serial.println("Brightness:" + String(brightness)); //stampa di verifica
  }
}
```

ESERCIZIO 6:

Obiettivo: vogliamo cambiare i 4 set-point di temperatura (2 relativi al condizionamento e 2 relativi al riscaldamento) in base alla presenza di persone nella stanza.

Componenti hardware utilizzate: sensore di temperatura, motore dc, led rosso, sensore PIR, sensore di rumore, porta seriale.

Componenti software utilizzate: GPIO, PWM, interrupt.

```
void dcMotor(float temp) {
  if (temp > temp_min_dc and temp <= temp_max_dc) {
    int n = temp - temp_min_dc;
    current_speed = n * tacca;
    analogWrite(FAN_PIN, (int) current_speed);
  }

  else if (temp > temp_max_dc) {
    current_speed = 255;
    analogWrite(FAN_PIN, (int) current_speed);
  }
  else {
    current_speed = 0;
    analogWrite(FAN_PIN, (int) current_speed);
  }
}
```

Sketch:

Si tratta solo di combinare le funzionalità di tutti gli esercizi precedenti, aggiungendo il cambio dei set-up di temperatura. Configuriamo tutti i pin e gestiamo l'interrupt con la funzione *checkPresence()* dell'esercizio 5 nella *setup().*

Nella *loop(),* ottenuto il valore corrente di temperatura ogni secondo, lo passiamo alle due funzioni *dcMotor()* e *led(),* in cui abbiamo scelto, per pulizia di codice, di gestire la scrittura analogica dei valori di tensione sulla ventola e sul led (tramite PWM) esattamente nel modo descritto negli esercizi 1 e 2. Quelli che dovranno cambiare sono i set-point con cui definiamo il valore minimo e massimo del duty-cycle del segnale PWM, e questo sarà deciso dalla eventuale presenza di persone nella stanza; i set-point sono variabili globali, con valore iniziale uguale a quello usato negli esercizi 1 e 2. Perciò abbiamo esattamente riportato il codice dell'esercizio 5, aggiungendo la funzionalità di cambiare il valore dei 4 set-point massimi e minimi in funzione della rilevazione o meno da parte dei due sensori combinati.

```
unsigned long now_millis = millis();
if (now_millis - start_millis1 >= timeout_pir) { //NESSUNO SI MUOVE
  if (now_millis - start_millis2 >= sound_interval) { //NESSUNO PARLA
    if (tot_count < n_sound_events) { //ANCHE SE C'E' STATO UN RUMORE SPORADICO
      Serial.println("Non ci sono persone nella stanza");
      temp_max_dc = 35;
      temp_min_dc = 30;
      temp_max_led = 25;
      temp_min_led = 20; //SE NON CI SONO PERSONE IL MOTORE è FERMO E IL LED ACCESO
    }
    tot_count = 0;
  }
  else { //QUALCUNO PARLA
    Serial.println(tot_count);
    if (tot_count >= n_sound_events) { //PARLA ABBASTANZA
      Serial.println("Ci sono persone nella stanza");

      temp_max_dc = 20;
      temp_min_dc = 15;
      temp_max_led = 20;
      temp_min_led = 15;

    }
  }
}
```

```
else { //QUALCUNO SI MUOVE
  if (start_millis1 != 0) {
    Serial.println("Ci sono persone nella stanza");
    temp_max_dc = 20;
    temp_min_dc = 15;
    temp_max_led = 20;
    temp_min_led = 15;
    //SE CI SONO PERSONE IL LED è SPENTO E IL MOTORE GIRA
  }
}
```

```
void setup() {
  // put your setup code here, to run once:
  pinMode(PIR_PIN, INPUT);
  pinMode(SND_PIN, INPUT);
  pinMode(TEMP_PIN, INPUT);
  pinMode(FAN_PIN, OUTPUT);
  pinMode(RLED_PIN, OUTPUT);
  lcd.begin(16, 2);
  lcd.setBacklight(255);
  lcd.home();
  lcd.clear();
  lcd.print("Avvio...");
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Lab 2.7 Starting");
  analogWrite(FAN_PIN, (int)current_speed);
  analogWrite(RLED_PIN, (int)brightness);
  attachInterrupt(digitalPinToInterrupt(PIR_PIN), checkPresence, CHANGE);
}
```

```
lcd.setCursor(0, 0);
lcd.print("T:" + String(temp));
lcd.setCursor(7, 0);
if (pres == true) {
  lcd.print(" Pres:1");
}
else {
  lcd.print(" Pres:0");
}
lcd.setCursor(0, 1);
lcd.print("AC:" + String((int)current_speed * 100 / 255) + "% ");
lcd.setCursor(8, 1);
lcd.print("HT:" + String(brightness * 100 / 255) + "%");

delay(2000);

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("AC m:" + String(temp_min_dc));
lcd.setCursor(10, 0);
lcd.print("M:" + String(temp_max_dc));
lcd.setCursor(0, 1);
lcd.print("HT m:" + String(temp_min_led));
lcd.setCursor(10, 1);
lcd.print("M:" + String(temp_max_led));

delay(2000);
lcd.clear();
```

ESERCIZIO 7:
Obiettivo: vogliamo visualizzare nel display LCD il valore della temperatura misurata, la percentuale di attivazione della ventola e del riscaldatore (tra 0 e 100%), un'indicazione del fatto che il sistema stia rilevando la presenza di persone nella stanza o meno e il valore attuale dei 4 set-point.
Componenti hardware utilizzate: sensore di temperatura, motore dc, led rosso, sensore PIR, sensore di rumore, porta seriale, display LCD.
Componenti software utilizzate: GPIO, PWM, interrupt, libreria LiquidCrystal_PCF8574.

Sketch:
Ancora una volta, si tratta di integrare una funzionalità, in questo caso la visualizzazione di informazioni sul display, al codice definito in precedenza. Non riportiamo nell'immagine le funzioni *checkPresence(), dcMotor()* e *led(),* già ampiamente trattate nelle spiegazioni precedenti.
Nella *setup(),* oltre a configurare i pin, aprire la comunicazione con la porta seriale e gestire l'interrupt, usiamo le funzioni della libreria LiquidCrystal_PCF8574 per dare le impostazioni di configurazione iniziali al display, le stesse dell'esercizio 1.6 dell'HW_Lab_Part1.
Nella *loop()* riportiamo esattamente il codice dell'esercizio precedente, con tutte le funzionalità per scrivere il valore analogico nei pin del led e della ventola e definire i 4 set-point in base alla presenza di persone nella stanza. Ottenute tutte le informazioni utili, usiamo il formato presentato nell'immagine nel testo dell'esercizio per visualizzale sul display, attraverso le sue funzioni di libreria. La presenza di una persona nella stanza, definita con la variabile globale booleana pres, viene visualizzata come valore binario (0=assenza, 1=presenza); il valore corrente della velocità della ventola e della luminosità del led, in percentuale, è ottenuto attraverso una semplice proporzione. Come suggerito dal testo, alterniamo la visualizzazione di queste informazioni con le rimanenti richieste ogni 2 secondi; loro sono il valore corrente dei 4 set-point, salvato in alcune variabili globali, come nell'esercizio precedente.

ESERCIZIO 8:

```
Serial.println("Se vuoi aggiornare i 4 set-point");
Serial.println("Scrivi i 4 set-point nel formato: num1(temp_min_led),num2(temp_max_led),num3(temp_min_dc),num4(temp_max_dc)");
```

```
void inserisciValoriDaSeriale() {
  int i = 0;
  for (i = 0; i < 4; i++) {
    num[i] = Serial.parseInt();
    Serial.println(num[i]);
  }
  temp_min_led = num[0];
  temp_max_led = num[1];
  temp_min_dc = num[2];
  temp_max_dc = num[3];
}
```

Obiettivo: Permettere l'aggiornamento dei quattro set-point mediante semplici comandi inviati dal PC alla Arduino mediante porta seriale.
Componenti hardware utilizzate: sensore di temperatura, motore dc, led rosso, sensore PIR, sensore di rumore, porta seriale, display LCD.
Componenti software utilizzate: GPIO, PWM, interrupt, libreria LiquidCrystal_PCF8574.

```
else { //QUALCUNO PARLA
  if (tot_count >= n_sound_events) { //PARLA ABBASTANZA

    //Per evitare quanto descritto al punto 4. Qui controllo se il valore pres == false.
    //In caso affermativo la stanza era precedentemente vuota e quindi setto i valori di default della stanza piena.
    //Se poi i valori vengono settati da Seriale allora si considereranno questi.
    if (pres == false) {
      temp_max_dc = 25;
      temp_min_dc = 20;
      temp_max_led = 20;
      temp_min_led = 15;
    }
    Serial.println("Ci sono persone nella stanza");
    pres = true;

    if (Serial.available() != 0) { //SETTA I 4 SET-POINT
      inserisciValoriDaSeriale();
    }
  }
}
else { //QUALCUNO SI MUOVE
  if (start_millis1 != 0) {
    if (pres == false) { //Con questi valori di temperatura: SE CI SONO PERSONE IL LED è SPENTO E IL MOTORE GIRA
      temp_max_dc = 25;
      temp_min_dc = 20;
      temp_max_led = 20;
      temp_min_led = 15;
    }
    Serial.println("Ci sono persone nella stanza");
    pres = true;
    if (Serial.available() != 0) { //SETTA I 4 SET-POINT
      inserisciValoriDaSeriale();
    }
  }
}
```

Sketch:
Anche qui dobbiamo usare tutto il codice definito precedentemente integrando una nuova funzionalità, cioè la possibilità di ricevere in input da porta seriale i 4 set-point di temperatura. Nella *setup()* stampiamo nel monitor seriale il formato in cui si richiede che i valori siano passati, e la nuova funzionalità viene gestita nella funzione *inserisciValoriDaSeriale().* È stato definito un vettore num[] di 4 elementi come variabile globale, e lui raccoglierà, grazie alla funzione *Serial.parseInt(),* i valori passati da seriale, se passati con il giusto formato. Loro andranno ad aggiornare le variabili globali che rappresentano i miei quattro set-point.
La presenza di elementi in attesa di essere ricevuti sulla porta seriale, rilevata grazie alla funzione *Serial.available(),* viene controllata solo se sono rilevate persone nella stanza, che possano fisicamente inserire i valori in input, quindi all'interno delle due condizioni definite nella combinazione dell'azione dei due sensori, già ampiamente discussa.

ESERCIZIO 9:

```
int value = digitalRead(SND_PIN);
if (value == LOW) {
  Serial.println("Person Detected by Noise Sensor");
  if (start_millis2 - millis() >= 2500) {
    tot_count += 1;
  }

  Serial.println("Tot count: " + String(tot_count));
  if (tot_count%2 == 0 and tot_count!=0) {
    greenLedState = !greenLedState;
    digitalWrite(GLED_PIN, greenLedState);
  }
  start_millis2 = millis();
}
```

Obiettivo: vogliamo utilizzare il LED verde per emulare una lampadina "smart"; in particolare, il LED potrà essere acceso tramite un doppio battito di mani e spento allo stesso modo, oppure automaticamente quando il sistema non rivela la presenza di persone nella stanza.
Componenti hardware utilizzate: sensore di temperatura, motore dc, led rosso, led verde, sensore PIR, sensore di rumore, porta seriale.
Componenti software utilizzate: GPIO, PWM, interrupt.

```
unsigned long now_millis = millis();
if (now_millis - start_millis1 >= timeout_pir) { //NESSUNO SI MUOVE
  Serial.println("Non ci sono persone nella stanza");

  digitalWrite(GLED_PIN, LOW);
  pres = false;
  temp_max_dc = 30;
  temp_min_dc = 25;
  temp_max_led = 25;
  temp_min_led = 20; //SE NON CI SONO PERSONE IL MOTORE è FERMO E IL LED ACCESO

  tot_count = 0;
}
```

Sketch:
Dobbiamo rimuovere dal codice dell'esercizio 6, oltre al codice per la gestione del display LCD che non serve più, la funzionalità descritta all'esercizio 4 e usare il sensore di rumore non per rilevare persone nella stanza, ma per accendere e spegnere il led verde con il doppio battito di mani. Usiamo allora tutte le funzioni già presentate precedentemente, nella *setup()* aggiungiamo la configurazione del GLED_PIN come output e in lui scriviamo il valore digitale iniziale (spento).
Nella *loop()* implementiamo la funzionalità richiesta: leggiamo il valore del GLED_PIN ogni secondo (*delay(1000)*) e, se un rumore è stato rilevato (LOW), incrementiamo il conteggio dei rumori rilevati, come variabile globale, ogni qualvolta che viene rilevato un rumore a distanza di almeno 2,5 secondi dal precedente. Se ho rilevato un numero pari di rumori (battiti di mani), cambiamo stato al led verde. Per esserci battiti di mani devono esserci persone e, di conseguenza, se il sensore PIR non rileva presenza di persone nella stanza, spegne il led verde.