Eugenio Principi - s236654
Arya Houshmand - s247275
Leonardo Perugini - s249807

Prima di tutto vengono settati i 4 set - points arbitrari che cambiano a seconda della presenza o meno di persone nell'ambiente in cui si trova il dispositivo; sulla base di questi valori infatti cambierà il comportamento del LED e della ventola.

```
//stabilisco i valori arbitrari di temperatura per Condizionatore e Riscaldamento, che variano alla presenza
//o meno di qualcuno nella stanza
//SE PRESENZA
float AC_P_MIN = 22, AC_P_MAX = 30;
float HT_P_MIN = 15, HT_P_MAX = 32;

//SE NON PRESENZA
float AC_NP_MIN = 20, AC_NP_MAX = 30;
float HT_NP_MIN = 10, HT_NP_MAX = 20;

//globali riferimento post verifica
float HT_min, HT_max, AC_min, AC_max;
```

I set - points possono essere modificati tramite seriale inserendo il comando nel formato "XXX val" dove XXX può essere ACM / ACm / HTM / HTm a seconda del valore che si vuole modificare.

```
void changeReferenceValuesFromSerial(){
  volatile float val;

  if(Serial.available() > 0) {
    String str = Serial.readString();
    //la stringa fornita deve essere nel formato 'XXX val'

    if(str.startsWith("ACm")){
      str = str.substring(4);
      val = str.toFloat();
      AC_min = val;
    }
    else if(str.startsWith("ACM")){
      str = str.substring(4);
      val = str.toFloat();
      AC_max = val;
    }
    else if(str.startsWith("HTm")){
      str = str.substring(4);
      val = str.toFloat();
      HT_min = val;
    }
    else if(str.startsWith("HTM")){
      str = str.substring(4);
      val = str.toFloat();
      HT_max = val;
    }
    else{
      Serial.println("Formato non abilitato");
    }
  }

}
```

All'interno del loop() viene confrontato per ogni istante il tempo trascorso dall'ultima rilevazione positiva di presenza: se la differenza tra i due istanti di tempo è maggiore del timeout del PIR, allora per questo sensore non è presente nessuno nella stanza.

```
//tempo corrente
time_now = millis();
//controllo continuo che negli ultimi 10 minuti non ci siano stati movimenti PIR
checkPIR(time_now);

//controllo sound
checkSound(time_now);

//se entrambi isPresentSound e isPresentPIR sono falsi anche la globale è falsa
if(!isPresentSound && !isPresentPIR){
  isPresent = false;
}

void checkPIR(long int time_now){
  if(time_now - start_time_PIR > timeout_pir){
    isPresentPIR = false;
  }
}

void checkSound(long int time_now){
  if(time_now - start_time_sound > timeout_sound){ //se superata la finestra temporale
    //controllo numero eventi
    if(countSounds < event_limit){

      isPresentSound = false;
    }
    //al termine della finestra azzera counter
    countSounds = 0;

  }else{ //se ancora nella finestra
    if(countSounds > event_limit){
      isPresent = true;
      isPresentSound = true;
    }
  }

}
```

Il cambiamento di stato dei sensori è intercettato dalla scheda Arduino grazie alla funzione di *attachInterrupt()*.

```
//interrupt per sensore PIR, attivato solo alla rilevazione
attachInterrupt(digitalPinToInterrupt(PIR_PIN), verifyPresence_PIR, RISING);

//interrupt per sensore suono, attivato solo al falling del segnale
attachInterrupt(digitalPinToInterrupt(SOUND_PIN), verifyPresence_sound, FALLING);
```

```
void verifyPresence_PIR(){
  //alla chiamata dell'interrupt rising cambia il valore del booleano
  isPresentPIR = true;
  isPresent = true;
  start_time_PIR = millis();
}

//funzione chiamata al FALLING del valore rilevato, aumenta il valore di countSounds
void verifyPresence_sound(){
  start_time_sound = millis();
  countSounds++;
}
```

Tuttavia per affermare che effettivamente non sia presente nessuno si controlla che non siano stati rilevati un numero *event_limit* di suoni nella finestra di tempo *timeout_sound.*

A seconda che siano presenti persone o meno all'interno della stanza, cambiano i valori di riferimento stabiliti precedentemente. Definiamo quindi i valori di coefficiente e termine noto delle funzioni affini del comportamento del LED e della ventola.

```
//a seconda che ci siano persone o meno modifica i valori di riferimento
changeReferenceValues(isPresent);

//dopo aver verificato presenza o meno cambio i coefficienti delle funzioni degli attuatori
updateCoeffTermNoto(isPresent);
```

```
//coeffVentola, termNotoVentola, coeffLED, termNotoLED;
float getCoeffVentola(boolean isPresent){
  //imposto come ordinata minima 40 poichè al di sotto di questo valore la ventola non gira
  float ymax = 255, ymin = 40, coeffVentola;

  coeffVentola = (ymax-ymin)/(AC_max -AC_min);
  return coeffVentola;
}
float getTermNotoVentola(boolean isPresent){
  float ymax = 255, ymin = 40, termNotoVentola;

  termNotoVentola = -(( (ymax-ymin)/(AC_max-AC_min) ) * AC_min) + ymin;
  return termNotoVentola;
}

float getCoeffLED(boolean isPresent){
  float ymax = 255, ymin = 0, coeffLED;

  coeffLED = (ymax-ymin)/(HT_max - HT_min);
  return coeffLED;
}

float getTermNotoLED(boolean isPresent){
  float ymax = 255, ymin = 0, termNotoLED;

  termNotoLED = -(( (ymax-ymin)/(HT_max-HT_min) ) *HT_min ) + ymin;
  return termNotoLED;
}
```

Tramite il sensore di temperatura rileviamo quanti gradi ci sono nella stanza e vengono aggiornati di conseguenza la velocità della ventola e la luminosità del LED.

```
temp = returnTemp();
velVentola = updateVentola(temp, velVentola);

pwmLED = updateLED(temp, pwmLED);


float updateVentola(float temp, float velVentola){
  velVentola = coeffVentola*temp + termNotoVentola;
  velVentola = checkVelVentola(velVentola);

  //modifica intensità ventola
  analogWrite(FAN_PIN, (int)velVentola);
  return velVentola;
}

float updateLED(float temp, int pwmLED){
  //modifica intensità LED
  pwmLED = returnLedPWM(temp);
  //il valore 255 corrisponde al LED spento e 0 alla massima luminosità
  //si tratta tuttavia di una differenza di luminosità apprezzabile solo nell'oscurità
  analogWrite(LED_PIN, pwmLED);
  return pwmLED;
}
```

Inoltre ricaviamo la percentuale di luminosità per il LED e velocità per la ventola a cui questi stanno lavorando, in modo tale da renderli visibili sul display lcd.
Sullo schermo LCD abbiamo un delay di 5 secondi che alterna le due pagine che vengono mostrate, in particolare la prima pagina riporta il valore di temperatura e le percentuali calcolate precedentemente, mentre la seconda mostra i set - points.

```
void LCDview(float temp, boolean isPresent, int percFan, int percLED){
  //PAGINA 1
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("T:");  lcd.print(temp); lcd.print("C");
  lcd.print("   P:"); lcd.print(isPresent);
  lcd.setCursor(0,1); lcd.print("AC:"); lcd.print(percFan); lcd.print("%  ");
  lcd.print("HT:"); lcd.print(percLED);
  lcd.setCursor(14,1);lcd.print("%");
  delay(5000);
  //PAGINA 2
  lcd.clear();
  lcd.setCursor(0,0);

  lcd.print("AC m:");lcd.print(AC_min,1); lcd.print(" M:"); lcd.print(AC_max,1);
  lcd.setCursor(0,1);
  lcd.print("HT m:");lcd.print(HT_min,1); lcd.print(" M:"); lcd.print(HT_max,1);
  delay(5000);
}
```

Nelle funzioni di setup() e di loop() abbiamo cercato di rendere quanto più leggibile il codice ricorrendo spesso all'uso di funzioni, in modo tale che il codice fosse piuttosto modulare.

Inoltre abbiamo inserito nel codice Arduino molti commenti che indicano lo scopo delle funzioni create e il ragionamento effettuato durante lo svolgimento del laboratorio, cercando anche di mantenere il codice ordinato per quanto possibile.