Centralina allarme

Testo progetto assegnato:
"*Realizzare una centralina per un appartamento con:*
1 Sensore porta e 2 sensori finestre (switch)
1 sensore PIR esterni alla porta ed 1 sensore PIR interno
2 Sensori PIR associati alle due stanze con finestre
Il Sistema è normalmente "dormiente" e deve essere risvegliato dall'interrupt proveniente da uno qualsiasi dei 7 sensori, collegati ad altrettanti pin di ingresso.
Il Sistema, una volta risvegliato deve essere in grado di rilevare la gravità della violazione, evitando falsi allarmi. secondo almeno le seguenti 3 scale di valori:
a) Possibile disturbo. Es.: segnale da una finestra, rientrato dopo pochi secondi e non seguito da altri.
b) Possibile Finestra/Porta lasciata aperta. Arriva segnale da una finestra aperta continuo ma non vi sono altri segnali di conferma intrusione dai PIR.
c) Intrusione confermata. La sequenza dei segnali è coerente con una effrazione con successiva intrusione."

1. Descrizione progetto

Il progetto ha come obiettivo quello di realizzare una centralina di allarme per un appartamento. Quest'ultimo è organizzato come rappresentato nella seguente figura.

Figura 1: *Piantina appartamento*

Al fine di rilevare intrusioni nell'appartamento, è stato deciso di monitorare la porta principale e le finestre al suo interno. Per quanto riguarda la porta d'ingresso, verrà utilizzato un sensore magnetico per rilevare l'apertura e la chiusura. Questo sensore permetterà di identificare lo stato della porta. Per garantire una maggiore sicurezza, saranno posizionati due sensori di movimento: uno all'interno e uno all'esterno della porta principale. Questi sensori rileveranno eventuali movimenti sospetti in prossimità dell'ingresso. Per rilevare

ulteriori movimenti all'interno dell'appartamento, verranno installati altri due sensori di movimento nelle due stanze che dispongono di finestre. Infine, per monitorare lo stato delle finestre (aperte o chiuse), verrà applicato un sensore magnetico per rilevare l'apertura di ciascuna finestra.
In questo modo, attraverso una combinazione di sensori di rilevamento, sarà possibile tenere sotto controllo l'accesso alla porta principale o dalle finestre e rilevare eventuali movimenti sospetti all'interno dell'appartamento, garantendo una maggiore sicurezza e protezione.

Il sistema progettato è in grado di rilevare la gravità della violazione, minimizzando così i falsi allarmi. Le violazioni sono divise in tre livelli:

a. *Possibile disturbo*.
b. *Possibile finestra/porta lasciata aperta*.
c. *Intrusione confermata*.

Il livello di violazione è determinata in base allo stato dei vari sensori secondo la seguente tabella:

Porta	**Finestra 1**	**Finestra 2**	**Porta esterno**	**Porta interno**	**Stanza 1**	**Stanza 2**	**Livello violazione**
-	**-**	**-**	**-**	**-**	**-**	**-**	-
apertura breve	-	-	-	-	-	-	a
-	apertura breve	-	-	-	-	-	a
-	-	apertura breve	-	-	-	-	a
-	-	-	movimento	-	-	-	a
-	-	-	-	movimento	-	-	a
-	-	-	-	-	movimento	-	a
-	-	-	-	-	-	movimento	a
aperta	-	-	-	-	-	-	b
-	aperta	-	-	-	-	-	b
-	-	aperta	-	-	-	-	b
aperta	-	-	movimento	movimento	-	-	c
-	aperta	-	-	movimento	-	-	c
-	-	aperta	-	-	movimento	-	c

Tali violazioni verranno segnalate tramite l’utilizzo di eventuali dispositivi di output.

2. Flowchart generale

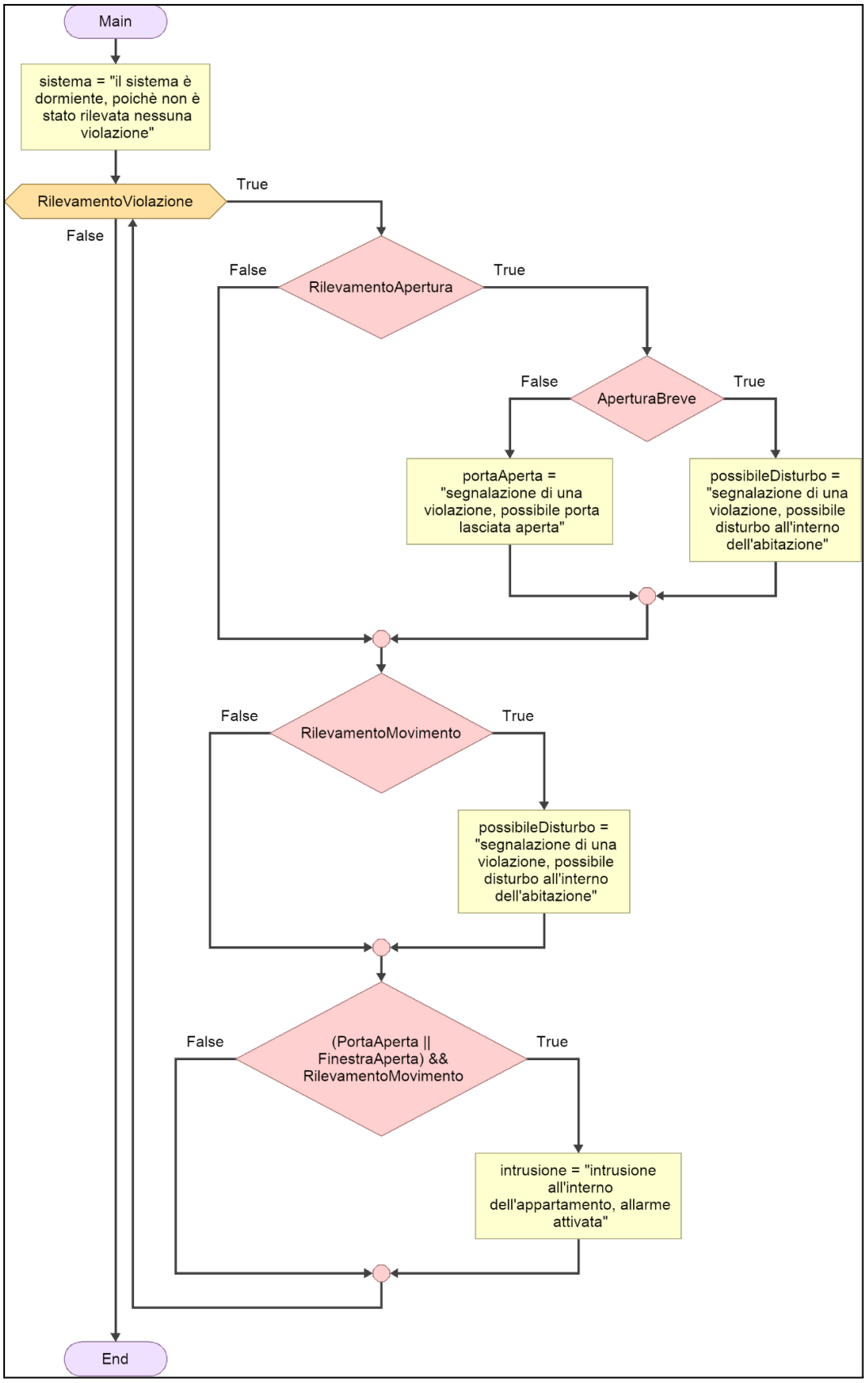

Figura 2: *Flowchart generale*

Il flowchart realizzato (*Figura 2*) schematizza la descrizione del progetto. In questo modo, ho una visione più chiara di ciò che andrò a realizzare.

3. Statechart

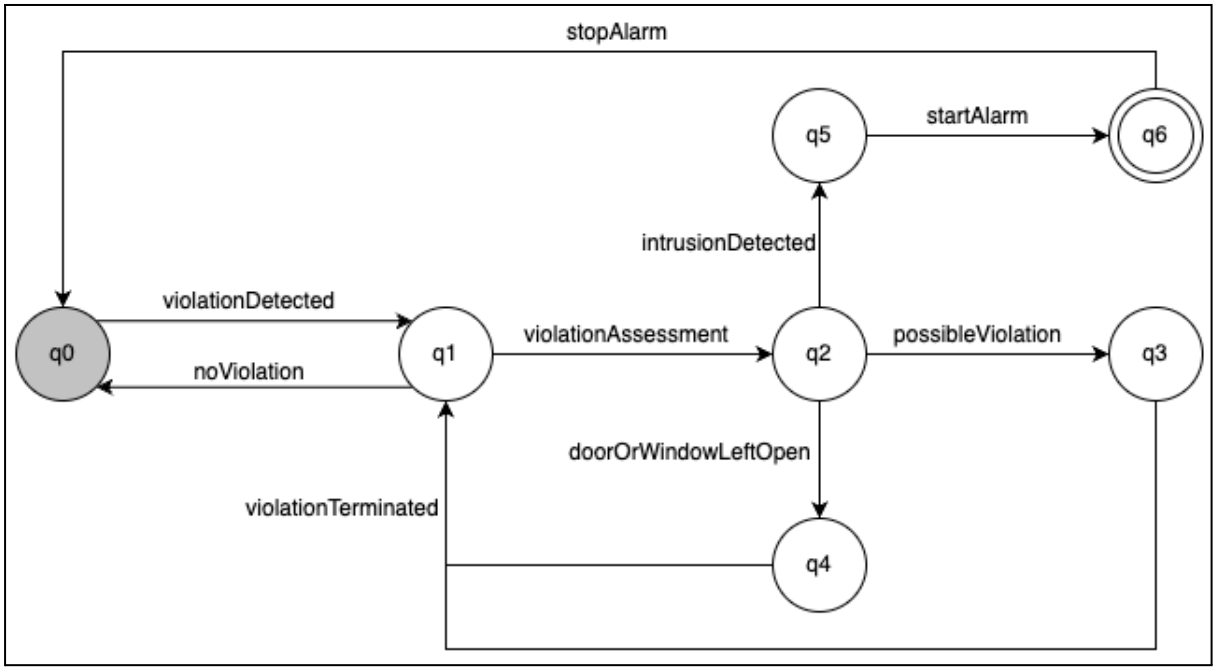

Figura 3: *Statechart*

Per avere una comprensione e una modellazione più chiara del comportamento dinamico del sistema ho realizzato uno statechart. Esso aiuta a strutturare e organizzare il flusso di stato, consentendo una progettazione e un'implementazione più chiara ed efficiente.
Il diagramma degli stati realizzato (*Figura 3*) rappresenta il comportamento del sistema di allarme tramite una serie di stati e transizioni tra di essi.

Il sistema della centralina di allarme inizia nello stato "dormiente" (***q0***), in cui è inattivo ed è in attesa di un'interazione o di un'interruzione. Quando viene rilevata un'interruzione (***violationDetected***), il sistema passa allo stato in cui è stata rilevata una violazione (***q1***), in cui inizia ad analizzare i segnali provenienti dai sensori (***violationAssessment***). In base ai segnali rilevati, il sistema passa agli stati successivi. Se si trova nella situazione in cui si ha semplicemente un possibile disturbo (***possibleViolation***) il sistema passa allo stato (***q3***) in cui segnala ciò. Stessa cosa nel caso in cui è stata rilevata solamente una porta o una finestra lasciata aperta (***doorOrWindowLeftOpen***), in questa situazione si passa allo stato (***q4***) in cui viene segnalato. Nel caso ci si trovi in uno di questi due ultimi stati, una volta che la violazione è terminata (***violationTerminated***) il sistema ritorna nello stato (***q1***) in cui viene gestita un'eventuale nuova interruzione. Nel caso non venga rilevata nessun altra violazione al momento (***noViolation***) il sistema può ritornare allo stato di partenza (***q0***), ove è "dormiente" ed è in attesa di un segnale da parte dei sensori. Infine, nel caso in cui la violazione viene valutata come intrusione (***intrusionDetected***), il sistema passa allo stato in cui viene segnalato ciò (***q5***). Successivamente il sistema passa allo stato di allarme (***q6***),

viene avviata l'allarme (***startAlarm***) e vengono intraprese le azioni appropriate. Una volta terminato lo stato di allarme (***stopAlarm***), il sistema torna al suo stato iniziale.

4. Schema dei componenti

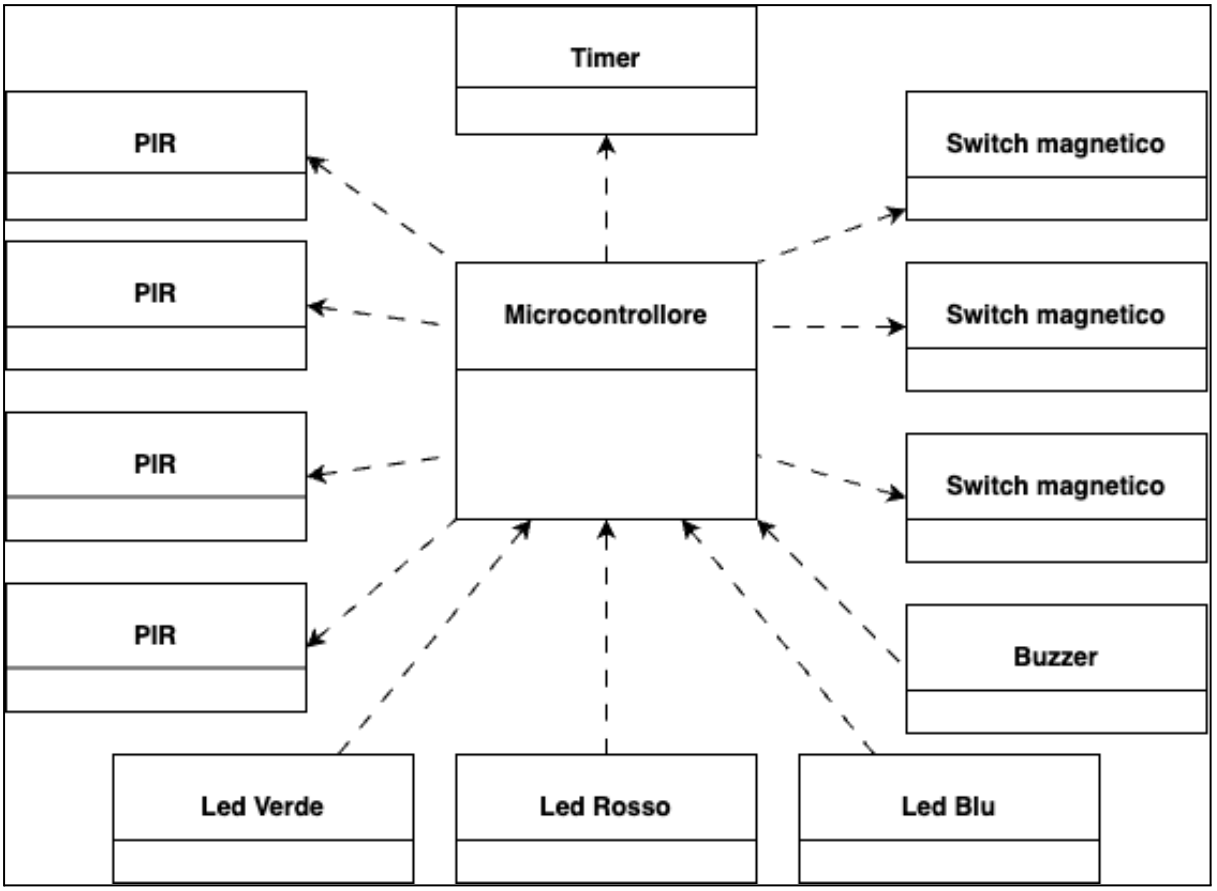

Figura 4: *Schema UML dei componenti*

Analizzando lo schema dei componenti realizzato (*Figura 4*), si può affermare che: i sensori PIR e gli switch magnetici inviano eventuali cambi di stato al microcontrollore, mentre il microcontrollore attiva eventualmente uno dei led e/o il buzzer in base alla gravità della violazione ricavata dalle segnalazioni ricevuta dai vari sensori.

Inoltre, si può dire che:

- I ***sensori PIR*** e gli ***switch magnetici*** sono interdipendenti dal microcontrollore, poiché sono loro ad inviare eventuali cambi di stato (interrupt) al microcontrollore.
- I ***led*** e il ***buzzer*** sono dipendenti dal microcontrollore, in quanto vengono azionati da quest'ultimo.
- Il ***microcontrollore*** è dipendente dai ***sensori PIR***, dagli ***switch magnetici*** e dal ***timer*** che stabilirà eventuali interrupt nel caso in cui una porta o una finestra rimanga aperta per un periodo prolungato.

5. Flowchart di dettaglio

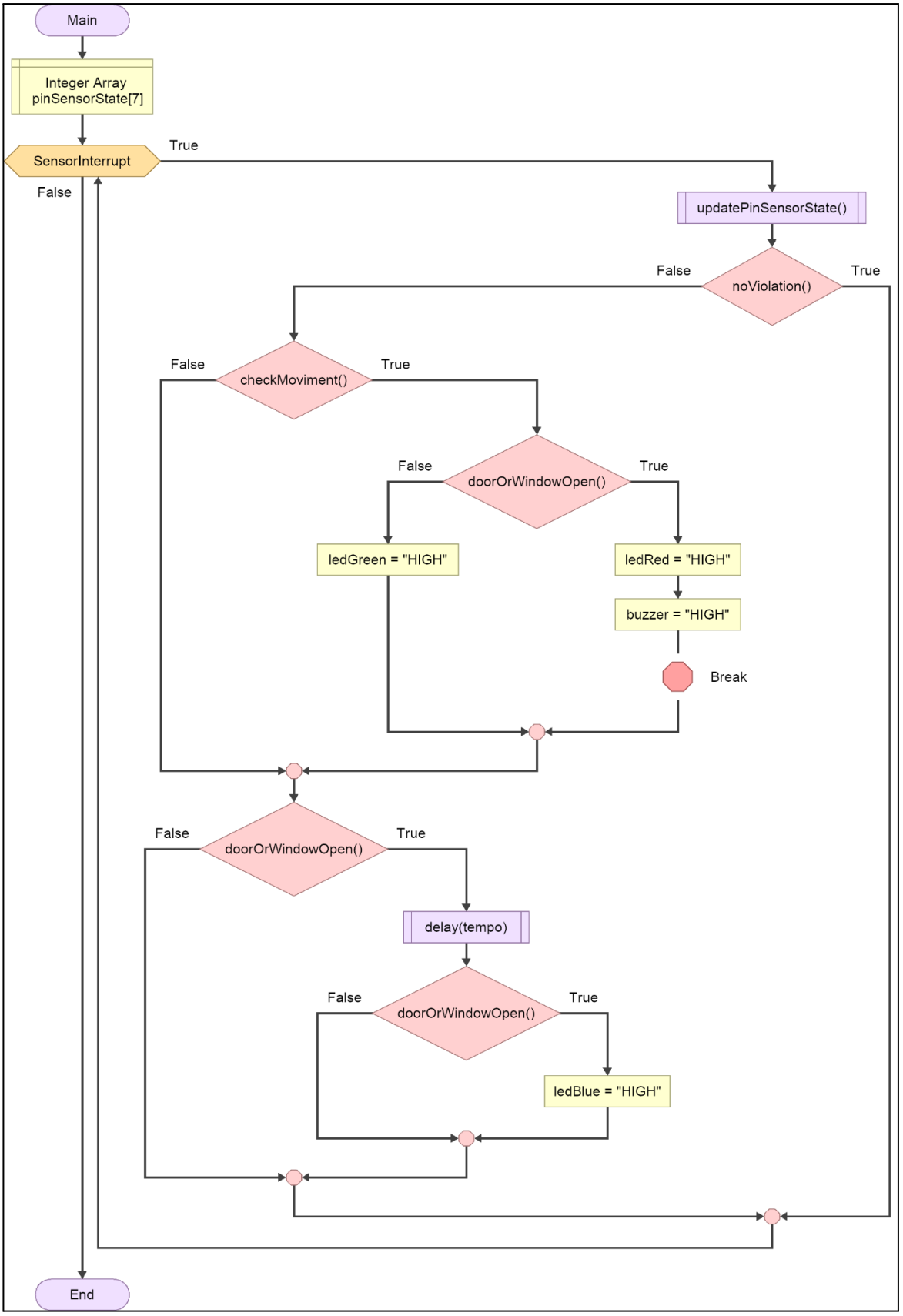

Figura 5: *Flowchart di dettaglio*

Per organizzare al meglio il codice che andrò ad implementare nel microcontrollore, ho realizzato un flowchart più specifico (*Figura 5*) rispetto a quello generale iniziale in modo da schematizzare in modo più chiaro possibile le azioni, dunque le funzioni che dovrò realizzare.

In tale flowchart le funzioni presenti sono:

- ***void updatePinSensorState():*** controlla lo stato di tutti i sensori e aggiorna i rispettivi valori nel vettore *pinSensorState[ ]*.
- ***bool noViolation():*** controlla la presenza di una violazione, dunque se lo stato di qualche sensore è impostato su “HIGH” (*Figura 6*).
- ***bool checkMoviment():*** controlla la presenza di un movimento, dunque se lo stato di qualche sensore di movimento è “HIGH” (*Figura 7*).
- ***bool doorOrWindowOpen():*** controlla se una porta o una finestra è aperta, dunque se lo stato di qualche switch è high (*Figura 8*).
- ***void delay(tempo):*** crea una pausa della durata della variabile tempo passata come parametro.

Nel vettore ***pinSensorState[ ]*** i primi quattro valori sono riferiti ai sensori di movimento, mentre gli altri tre agli switch delle finestre e della porta.

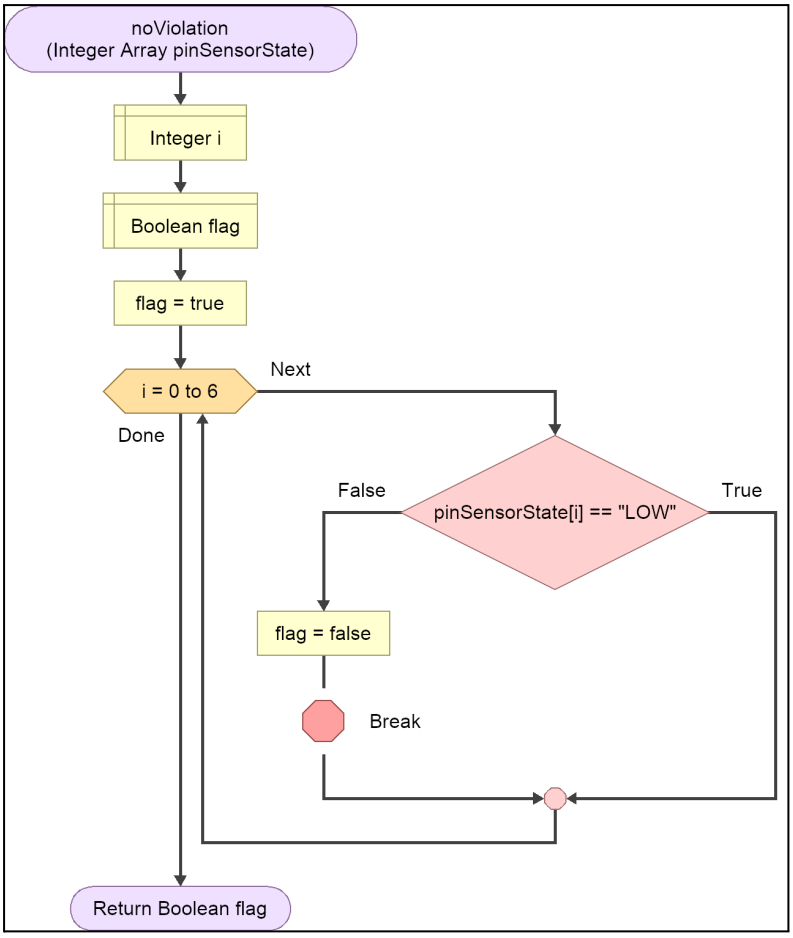

Figura 6: *Flowchart del metodo noViolation()*

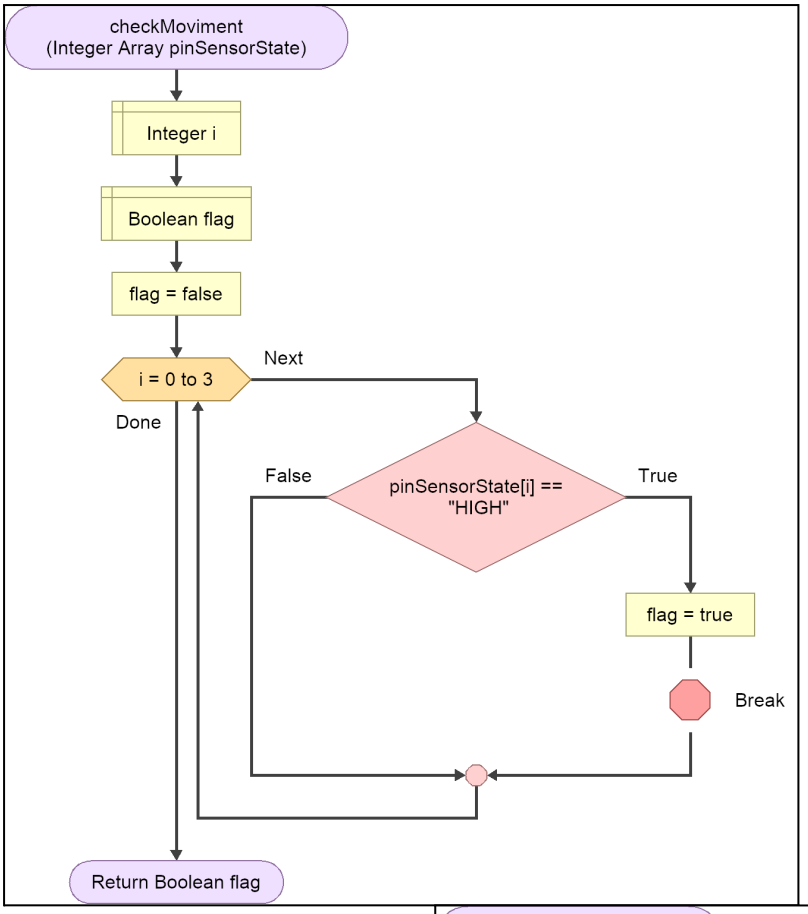

Figura 7: *Flowchart della funzione checkMoviment()*

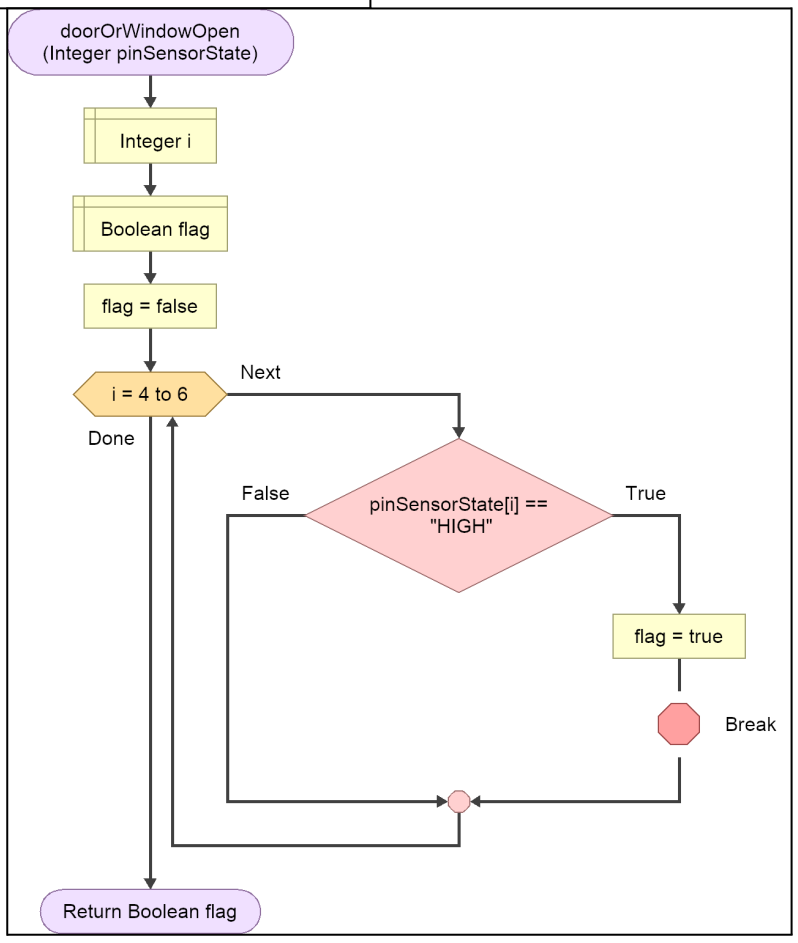

Figura 8: *Flowchart della funzione doorOrWindowOpen()*

6. Implementazione (su Wokwi)

Per l'implementazione della centralina di sicurezza su "Wokwi" ho utilizzato come microcontrollore Arduino Nano (**ATmega 328**). Il suo utilizzo è vantaggioso in questo caso, in quanto: ha delle dimensioni compatte, è una scheda di arduino molto popolare, ha un costo ridotto, dispone di un numero di pin di input/output digitali e analogici sufficienti per soddisfare le esigenze dei sensori e delle componenti necessarie. Dunque, in questa situazione, non avrebbe senso implementare tale progetto in un microcontrollore più performante.
Inoltre, come componenti di input collegati al microcontrollore ho utilizzato: 4 **sensori PIR** (*Pir Motion Sensor*) e 3 **slide-switch**. L'utilizzo di quest'ultimi sostituiscono gli switch magnetici previsti nella descrizione iniziale del progetto, poiché non disponibili in questo ambiente di simulazione. Invece, come dispositivi di output, utilizzati per poter segnalare la gravità delle violazioni e l'eventuale attivazione dell'allarme, ho utilizzato: tre **led** di colori diversi (rosso, blu, verde) ed un **buzzer**. Per rendere più fluida l'accensione dei led ho collegato, tra essi e i relativi pin, delle resistenze da 220 Ohm. Infine, ho inserito anche un **bottone** (*button*) che mi permette di spegnere l'allarme quando viene attivata (o eventualmente di resettare il sistema in caso di anomalie).

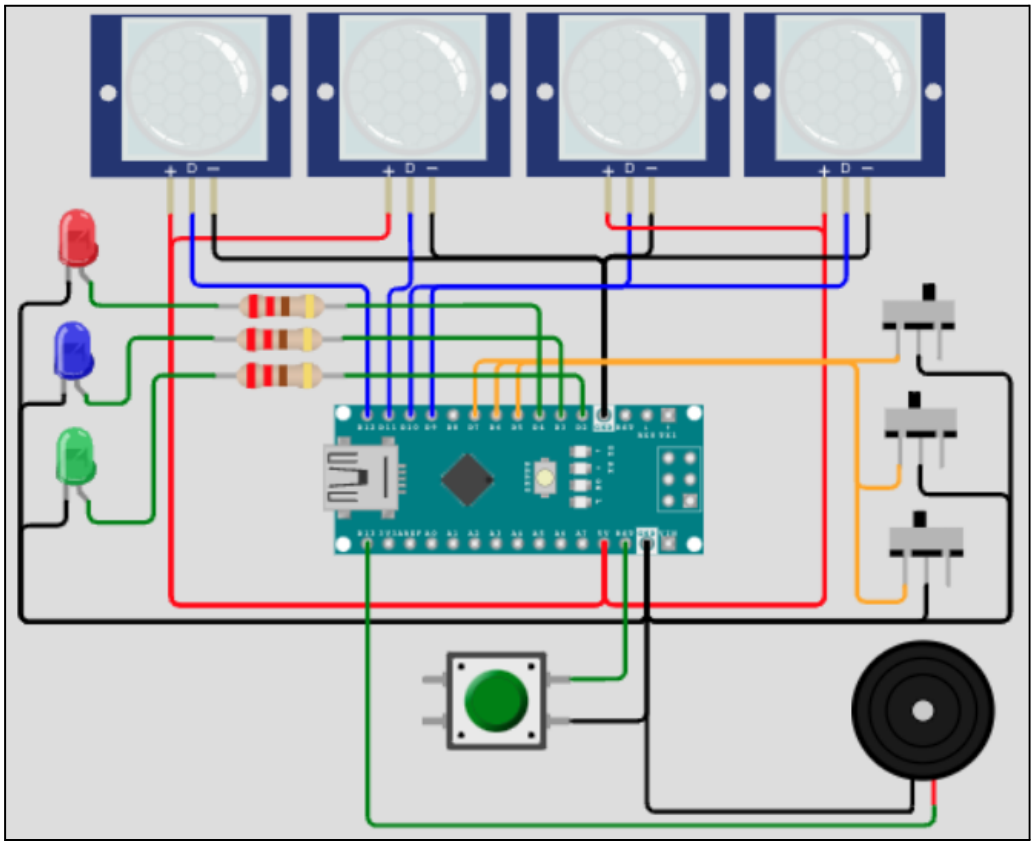

Figura 9: *Disposizione grafica dei componenti (su Wokwi)*

Successivamente, sono passato alla scrittura del codice. Per la realizzazione di quest'ultimo non ho usufruito di nessuna libreria, ma ho utilizzato direttamente i registri del microcontrollore. In questo modo, si ottiene una maggiore velocità di esecuzione, poiché operare direttamente con i registri permette di scrivere un codice più ottimizzato e di evitare

il sovraccarico delle chiamate di funzione. Inoltre, operando in questo modo sono riuscito ad adattare il più possibile il codice alle esigenze specifiche di questo progetto.

Per ottenere un prodotto più efficiente ho lavorato soprattutto tramite l'utilizzo di interrupt. Come prima cosa ho gestito gli interrupt causati dai vari pin a cui sono collegati i sensori. Nell'Arduino Nano da datasheet gli interrupt hardware (*External Interrupt*) sono solo due, INT0 e INT1 (mappati rispettivamente sui pin 2 e 3). Dunque, non essendo sufficienti per la gestione di tutti i sensori, ho dovuto utilizzare degli interrupt addizionali (*Pin Change Interrupt*), essi sono divisi in tre gruppi e sono distribuiti fra i vari pin della scheda. Nello specifico ho utilizzato il gruppo PCIE0 (in cui ho collegato i sensori PIR) e PCIE2 (in cui ho collegato gli slide-switch). Una volta abilitati gli interrupt nei pin specifici, ho valutato in base allo stato dei vari pin la gravità delle eventuali violazioni.
Inoltre, per la realizzazione della centralina di allarme ho impostato un timer, esso mi permette di stabilire quando una determinata porta o finestra rimane aperta per un periodo di tempo prolungato. Per poter impostare il timer in modo che vada in overflow dopo un minuto ho utilizzato la seguente formula:

$$time = dimTimer - (F\_CPU / Prescaler) * TempoDesiderato$$

Questi sono stati gli accorgimenti principali che ho dovuto tenere in considerazione per uno sviluppo efficiente ed efficace del sistema.

Ho implementato il codice cercando di essere il più coerente possibile alle specifiche riportate fino ad ora, aggiungendo commenti per renderlo esaustivo e più leggibile..

Link al progetto su Wokwi: https://wokwi.com/projects/368249522804096001

---

sketch.ino:

```
/*
 Titolo: Centralina allarme
 Ultimo aggiornamento: 2023-06-28
 Autore: Matteo Cardellini
*/

// variabili di stato dei sensori
volatile int pin5State = LOW;
volatile int pin6State = LOW;
volatile int pin7State = LOW;
volatile int pin9State = LOW;
volatile int pin10State = LOW;
volatile int pin11State = LOW;
volatile int pin12State = LOW;

void setup() {
 // definizione dei pin di output
 DDRD |= _BV(PORTD2);   // led verde          (pin 2)
 DDRD |= _BV(PORTD3);   // led blu            (pin 3)
 DDRD |= _BV(PORTD4);   // led rosso          (pin 4)
 DDRB |= _BV(PORTB5);   // buzzer             (pin 13)
 // definizione dei pin di input
 DDRB &= ~_BV(PORTB1);  // PIR stanza 2       (pin 9)
 DDRB &= ~_BV(PORTB2);  // PIR stanza 1       (pin 10)
 DDRB &= ~_BV(PORTB3);  // PIR interno porta  (pin 11)
 DDRB &= ~_BV(PORTB4);  // PIR esterno porta  (pin 12)
 DDRD &= ~_BV(PORTD5);  // finestra 2         (pin 5)
 DDRD &= ~_BV(PORTD6);  // finestra 1         (pin 6)
 DDRD &= ~_BV(PORTD7);  // porta              (pin 7)
 // abilito le resistenze di pull-up per i pin di input
 PORTB |= _BV(PORTB1) | _BV(PORTB2) | _BV(PORTB3) | _BV(PORTB4);
 PORTD |= _BV(PORTD5) | _BV(PORTD6) | _BV(PORTD7);

 // abilito gli interrupt nei pin dei sensori
 enabledPCIInterrupt();
 // imposto il timer
 setTimer1();
}
```

```
void loop() {
}

// Funzione che abilità gli interrupt PCI nei pin a cui sono collegati i
// sensori
void enabledPCIInterrupt() {
 PCICR |= _BV(PCIE0) | _BV(PCIE2);
 // Abilito i pin specifici
 // pin: 9, 10, 11, 12
 PCMSK0 |= _BV(PCINT1) | _BV(PCINT2) | _BV(PCINT3) | _BV(PCINT4);
 // pin: 5, 6, 7
 PCMSK2 |= _BV(PCINT21) | _BV(PCINT22) | _BV(PCINT23);
}

// Funzione che inizializza il Timer1
void setTimer1() {
 TCCR1A = 0;  // imposto a 0 i registri di controllo A (default tutti 1)
 TCCR1B = 0;  // imposto a 0 i registri di controllo B (default tutti 1)
 TCCR1B |= _BV(CS12) | _BV(CS10);  // imposto un prescaler di 1024
}

// Funzione di interrupt chiamata quando si verifica un interrupt nel gruppo
PCIE0
ISR(PCINT0_vect) {
 violationSeverity();
}

// Funzione di interrupt chiamata quando si verifica un interrupt nel gruppo
PCIE2
ISR(PCINT2_vect) {
 violationSeverity();
}

// Funzione che legge lo stato dei pin a cui sono stati collegati i sensori.
void updatePinState() {
 pin9State = PINB & _BV(PORTB1);
 pin10State = PINB & _BV(PORTB2);
 pin11State = PINB & _BV(PORTB3);
 pin12State = PINB & _BV(PORTB4);
 pin5State = PIND & _BV(PORTD5);
 pin6State = PIND & _BV(PORTD6);
```

```
 pin7State = PIND & _BV(PORTD7);
}

// Funzione che definisce la gravità di violazione. Viene invocata al cambio
di stato di uno dei 7 sensori.
void violationSeverity() {
 // aggiorno i valori delle variabili di stato
 updatePinState();

 // CASO 0: stato di allarme
 if (PIND & _BV(PORTD4)) { // il led rosso è attivo
   startAlarm();
   return;
 }

 // CASO 1: nessuna violazione
 if(noViolation()) return;

 // CASO 2: intrusione / possibile movimento
 if (checkIntrusionOrPossibleViolation()) return;

 // se arrivo qui, non c'è stato nessun movimento

 // CASO 3: possibile finestra/porta lasciata aperta
 checkDoorOrWindowLeftOpen();
}

// Funzione che attiva il buzzer in caso di intrusione e disattiva gli
// interrupt fino a quando l'allarme non viene spenta
void startAlarm() {
 // abbiamo un'intrusione, accendo il buzzer
 PORTB |= _BV(PORTB5);
 // disabilito gli interrupt PCI
 PCICR &= ~(_BV(PCIE0) | _BV(PCIE2));
}

// Funzione che controlla se è presente qualche violazione
bool noViolation() {
 if (!(pin9State || pin10State || pin11State || pin12State) &&
   !(pin5State || pin6State || pin7State)) {
   // non c'è nessun movimento e nessuna finestra/porta aperta
```

```
    PORTD &= ~_BV(PORTD2);  // spengo led verde
    PORTD &= ~_BV(PORTD3);  // spengo led blu
    PORTD &= ~_BV(PORTD4);  // spengo led rosso
    return true;
  }
  return false;
}

// Funzione che controlla il verificarsi di un intrusione o una possibile
// violazione
bool checkIntrusionOrPossibleViolation() {
  if ((pin9State || pin10State || pin11State || pin12State)) {  // movimento
    if (pin5State || pin6State || pin7State) {
      // movimento + porta/ finestra aperta
      PORTD &= ~_BV(PORTD2); // spengo led verde
      PORTD &= ~_BV(PORTD3); // spengo led blu
      PORTD |= _BV(PORTD4);  // accendo led rosso
    } else {
      // solo movimento
      PORTD |= _BV(PORTD2);  // accendo led verde
    }
    return true;
  }
  return false;
}

// Funzione che controlla se è stata lasciata una porta o una finestra
// aperta
void checkDoorOrWindowLeftOpen() {
  if (pin5State || pin6State || pin7State) {
    // porta / finestra aperta (breve tempo)
    PORTD &= ~_BV(PORTD4);  // spengo led rosso
    PORTD |= _BV(PORTD2);  // accendo led verde

    // resetto il timer
    // imposto il timer in modo che vada in overflow dopo 1 minuto
    TCNT1 = 65536 - (F_CPU / 1024) * 60;
    TIMSK1 |= _BV(TOIE1); // abilito l'interrupt di overflow del Timer1
  }
}
```

```
// Funzione di interrupt che viene chiamata quando si verifica un overflow
// nel Timer1
ISR(TIMER1_OVF_vect) {
 // è ancora aperta?
 if(pin5State || pin6State || pin7State) {
   // porta / finestra lascia aperta
   PORTD &= ~_BV(PORTD2); // spengo led verde
   PORTD |= _BV(PORTD3);  // accendo led blu
 }
 // disabilito l'interrupt di overflow del Timer1
 TIMSK1 &= ~_BV(TOIE1);
}
```

diagram.json:

```
{
 "version": 1,
 "author": "MATTEO CARDELLINI",
 "editor": "wokwi",
 "parts": [
     { "type": "wokwi-arduino-nano", "id": "nano", "top": 35.48, "left":
-37.89, "attrs": {} },
   {
     "type": "wokwi-pir-motion-sensor",
     "id": "pir1",
     "top": -142.4,
     "left": -142.46,
     "attrs": {}
   },
   {
     "type": "wokwi-pir-motion-sensor",
     "id": "pir2",
     "top": -142.53,
     "left": -45.12,
     "attrs": {}
   },
   {
     "type": "wokwi-pir-motion-sensor",
     "id": "pir3",
     "top": -142.53,
```

```
      "left": 52.88,
      "attrs": {}
    },
    {
      "type": "wokwi-pir-motion-sensor",
      "id": "pir4",
      "top": -142.53,
      "left": 149.75,
      "attrs": {}
    },
    { "type": "wokwi-slide-switch", "id": "sw1", "top": 83.31, "left": 215.41,
"attrs": {} },
     { "type": "wokwi-slide-switch", "id": "sw2", "top": -20.32, "left":
212.05, "attrs": {} },
    { "type": "wokwi-slide-switch", "id": "sw3", "top": 29.25, "left": 212.75,
"attrs": {} },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": 133.69,
      "left": 197.96,
      "attrs": { "volume": "1.0" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -61.99,
      "left": -166.88,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 37.39,
      "left": -167.85,
      "attrs": { "color": "limegreen" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": -10.89,
```

```
      "left": -168.93,
      "attrs": { "color": "blue" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": -15.1,
      "left": -85.71,
      "attrs": { "value": "220" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r2",
      "top": -0.48,
      "left": -87.46,
      "attrs": { "value": "220" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r3",
      "top": 16.49,
      "left": -86.88,
      "attrs": { "value": "220" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 146.67,
      "left": 6.03,
      "attrs": { "color": "green" }
    }
  ],
  "connections": [
    [ "pir1:VCC", "nano:5V", "red", [ "v176.82", "h207.24" ] ],
     [ "pir2:VCC", "nano:5V", "red", [ "v10.6", "h-96.88", "v166", "h193.75",
"v-28.19" ] ],
    [ "pir3:GND", "nano:GND.2", "black", [ "v18.6", "h-21.11" ] ],
    [ "pir2:GND", "nano:GND.2", "black", [ "v19.26", "h76.89" ] ],
    [ "pir1:GND", "nano:GND.2", "black", [ "v19.8", "h174.23" ] ],
    [ "pir1:OUT", "nano:12", "blue", [ "v31.9", "h78.35" ] ],
    [ "pir3:OUT", "nano:10", "blue", [ "v32.6", "h-97.79" ] ],
```

```
    [ "pir4:VCC", "nano:5V", "red", [ "v176.6", "h-84.43" ] ],
    [ "pir3:VCC", "nano:5V", "red", [ "v5.93", "h97.03", "v170.67", "h-98.18",
"v-28.19" ] ],
    [ "pir4:GND", "nano:GND.2", "black", [ "v19.26", "h-116.56" ] ],
    [ "pir2:OUT", "nano:11", "blue", [ "h0.38", "v33.44", "h-9.77" ] ],
    [ "pir4:OUT", "nano:9", "blue", [ "v32.31", "h-185.06" ] ],
      [ "sw2:2", "nano:GND.1", "black", [ "v7.77", "h40.99", "v112.21",
"h-161.76" ] ],
      [ "sw3:2", "nano:GND.1", "black", [ "v12.82", "h39.87", "v57.59",
"h-161.76", "v-35.78" ] ],
    [ "sw1:2", "nano:GND.1", "black", [ "v16.35", "h-112.69" ] ],
    [ "sw2:1", "nano:7", "orange", [ "v1.96", "h-188.74" ] ],
    [ "sw3:1", "nano:6", "orange", [ "v9.86", "h-18.81", "v-56.89", "h-161.03"
] ],
      [ "sw1:1", "nano:5", "orange", [ "v7.46", "h-21.47", "v-108.55",
"h-151.43" ] ],
      [ "led1:C", "nano:GND.1", "black", [ "v9.61", "h-21.46", "v144.29",
"h293.63" ] ],
      [ "led3:C", "nano:GND.1", "black", [ "v3.99", "h-19.41", "v98.81",
"h34.81" ] ],
      [ "led2:C", "nano:GND.1", "black", [ "v10.73", "h-20.49", "v43.79",
"h40.42" ] ],
    [ "bz1:1", "nano:GND.1", "black", [ "v2.12", "h-92.32" ] ],
    [ "bz1:2", "nano:13", "green", [ "v8.3", "h-222.47" ] ],
    [ "led1:A", "r1:1", "green", [ "v0" ] ],
    [ "r1:2", "nano:4", "green", [ "v-0.06", "h83.84" ] ],
    [ "led3:A", "r2:1", "green", [ "v3.83", "h18.98", "v-28.66" ] ],
    [ "r2:2", "nano:3", "green", [ "v-0.06", "h56.94" ] ],
    [ "led2:A", "r3:1", "green", [ "v9.35", "h29.02", "v-63.74" ] ],
    [ "r3:2", "nano:2", "green", [ "v-0.07", "h47" ] ],
    [ "btn1:1.r", "nano:RESET", "green", [ "v-0.11", "h10.4" ] ],
    [ "btn1:2.r", "nano:GND.1", "black", [ "h0" ] ]
  ],
  "dependencies": {}
}
```