**AKADEMIA GÓRNICZO-HUTNICZA**

**IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI AUTOMATYKI INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

**KIERUNEK AUTOMATYKA I ROBOTYKA II STOPIEŃ**
**SPECJALIZACJA INFORMATYKA W STEROWANIU I ZARZĄDZANIU**

**INFORMATYKA CZASU RZECZYWISTEGO**

# Symulator obecności w mieszkaniu

---

| L.p. | Członkowie grupy: | Nr. albumu | Adres email |
|---|---|---|---|
| 1 | Dominik Czyżyk | 401858 | czyzyk@student.agh.edu.pl |
| 2 | Bartosz Więcek | 400421 | bwiecek@student.agh.edu.pl |

*Opiekun: dr inż. Magdalena Szymczyk*

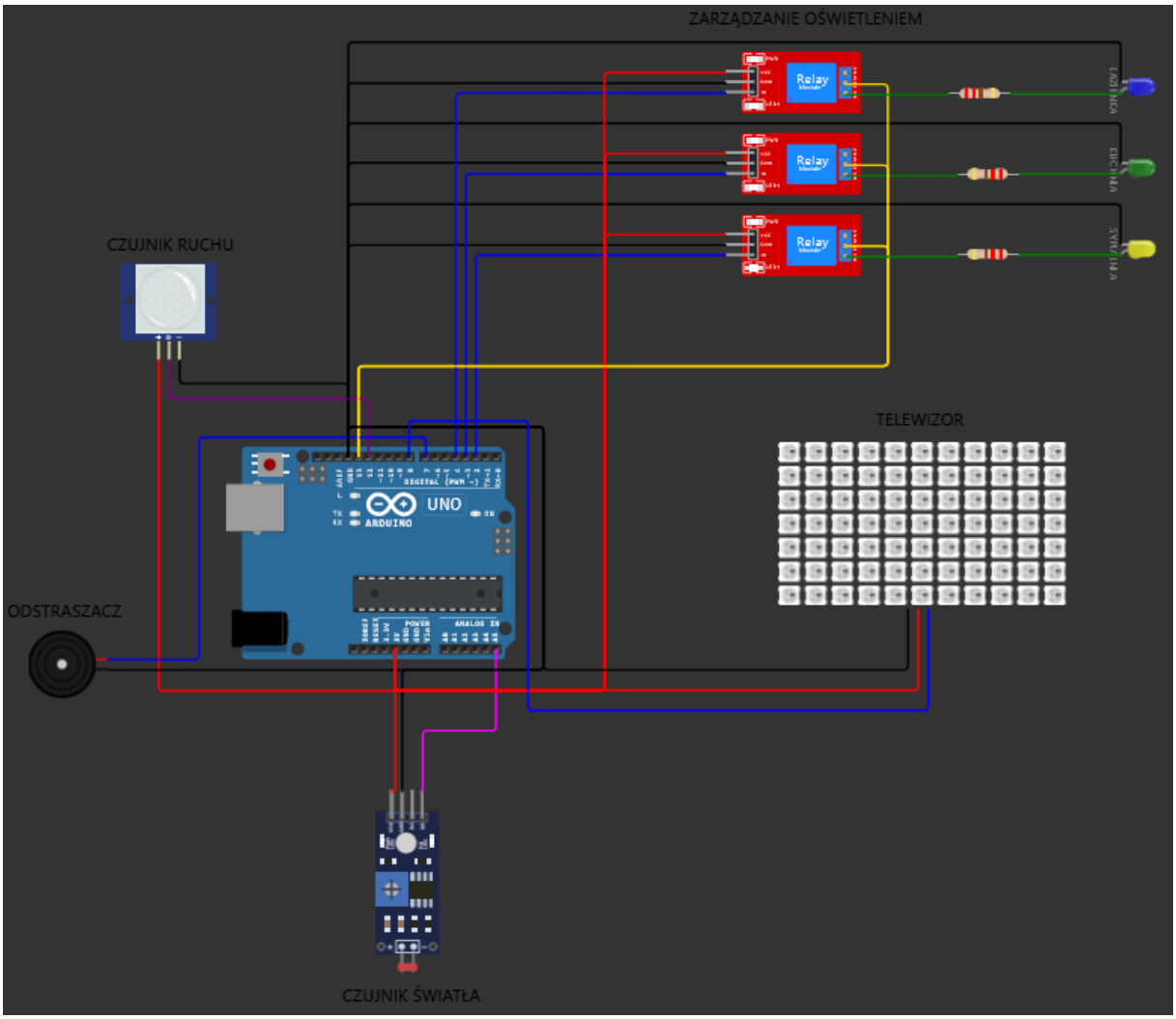
## Spis treści

# 1. Cel

Celem niniejszego projektu będzie zaprojektowanie symulatora obecności w mieszkaniu na płytce Arduino UNO. Jako środowisko programistyczne wykorzystana zostanie platforma Wokwi, która pozwala na tworzenie i symulowanie różnych układów wbudowanych w przeglądarce internetowej. Kod oprogramowania zaimplementowano przy pomocy języka C. Schemat połączeń reprezentowany jest z wykorzystaniem pliku JSON.

# 2. Specyfikacja problemu

W celu stworzenia symulatora obecności w mieszkaniu zaprojektowano układ przedstawiony na poniższym schemacie.




*Rys. 1. Schemat systemu symulującego obecność w mieszkaniu*

Na płytce Arduino UNO zaimplementowane zostały trzy funkcjonalności:

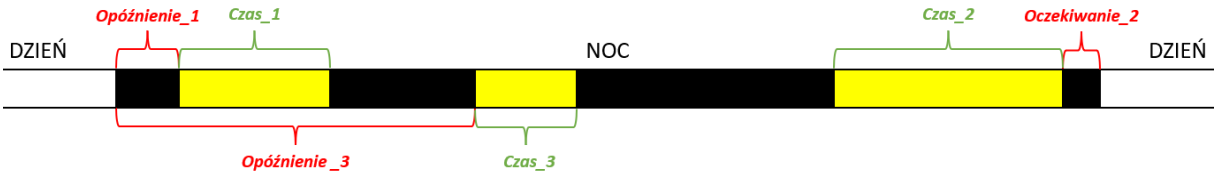
- sterowanie oświetleniem,
- uruchamianie elektronicznego odstraszacza,
- sterowanie telewizorem.

## Sterowanie oświetleniem

System symulujący obecność domownika w mieszkaniu ma za zadanie włączanie oraz wyłączanie światła podczas gdy na zewnątrz jest ciemno (trwa noc). Do detekcji dnia oraz nocy używany jest czujnik światła (przedstawiony na *Rys.1*) zamontowany w odpowiednim miejscu. Dla prawidłowego działania systemu konieczna jest znajomość długości trwania nocy. Przed pierwszym uruchomieniem systemu ustawiana jest ona ręcznie, natomiast wraz z upływem kolejnych dni oraz nocy wartość ta jest automatycznie aktualizowana. System steruje oświetleniem w trzech pomieszczeniach: sypialni, kuchni oraz łazience. W momencie wykrycia nocy (chwila, w której intensywność światła na zewnątrz spadnie poniżej zadanej wartości) dla każdego z trzech pomieszczeń generowane są w sposób losowy dwie liczby (zakresy wartości tych liczb są jednak dobrane w taki sposób, aby uzyskać „realistyczny efekt"). Liczby te oznaczają:

- dla sypialni:
  - ***Opóźnienie_1*** - czas, po którym uruchomione zostanie światło licząc od momentu wykrycia nocy (stosunkowo niewielki, symuluje wieczorną aktywność człowieka w sypialni),
  - ***Oczekiwanie_1*** - czas, po którym (zgodnie z zakładaną długością trwania nocy) rozpocznie się dzień liczony wstecz od przewidywanego momentu końca nocy (stosunkowo niewielki, symuluje poranną aktywność człowieka w sypialni),
  - ***Czas_1*** - czas, przez jaki światło w sypialni będzie zapalone (stosunkowo długi).
- dla kuchni:
  - ***Opóźnienie_2*** - czas, po którym uruchomione zostanie światło licząc od momentu wykrycia nocy (stosunkowo niewielki, symuluje wieczorną aktywność człowieka w kuchni),
  - ***Oczekiwanie_2*** - czas, po którym (zgodnie z zakładaną długością trwania nocy) rozpocznie się dzień liczony wstecz od przewidywanego momentu końca nocy (stosunkowo niewielki, symuluje poranną aktywność człowieka w kuchni),
  - ***Czas_2*** - czas, przez jaki światło w kuchni będzie zapalone (stosunkowo długi).
- dla łazienki:
  - ***Opóźnienie_3*** - czas, po którym uruchomione zostanie światło licząc od momentu wykrycia nocy (dowolnie długi, symuluje nocną wizytę w toalecie domownika),
  - ***Czas_3*** - czas, przez jaki światło w łazience będzie zapalone (stosunkowo krótki).

Momenty, w których poszczególne światła są włączone dobrze obrazuje poniższy schemat.




*Rys. 2. Schemat działania funkcji sterującej oświetleniem*

## Uruchamianie elektronicznego odstraszacza

Elektronicznym odstraszaczem jest w tym przypadku nagranie szczekającego psa. System symulujący obecność domownika w mieszkaniu ma za zadanie uruchomienie nagrania w momencie wykrycia ruchu w pobliżu drzwi wejściowych. Do detekcji ruchu wykorzystano czujnik (przedstawiony na *Rys.1*)

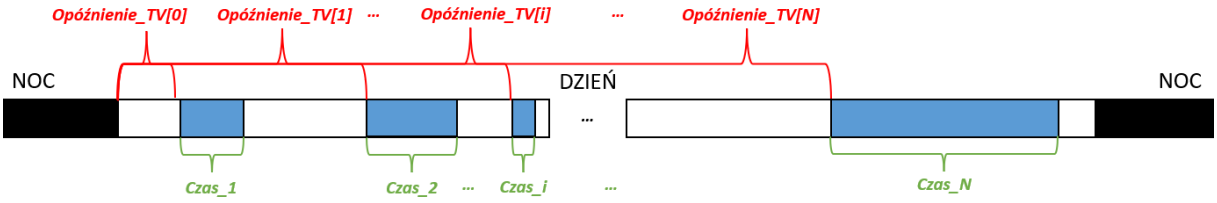
zamontowany przy drzwiach. Funkcja ta ma za zadanie odstraszyć potencjalnych włamywaczy, którzy pojawiliby się przy drzwiach wejściowych. Działa ona zarówno w dzień jak i w nocy.

## Sterowanie telewizorem

Zasada działania jest tu analogiczna do tej zastosowanej przy włączaniu i wyłączaniu światła, z tą różnicą, że telewizor uruchamiany jest kilkukrotnie (a nie jak światła - tylko raz) w ciągu dnia (a nie w nocy). Liczba uruchomień telewizora jest sparametryzowana. W momencie wykrycia dnia (chwila, w której intensywność światła na zewnątrz przekroczy zadaną wartość) losowymi liczbami wypełnione zostają dwie tablice, każda o długości równej ustalonej liczbie uruchomień telewizora. Tablicami tymi są:

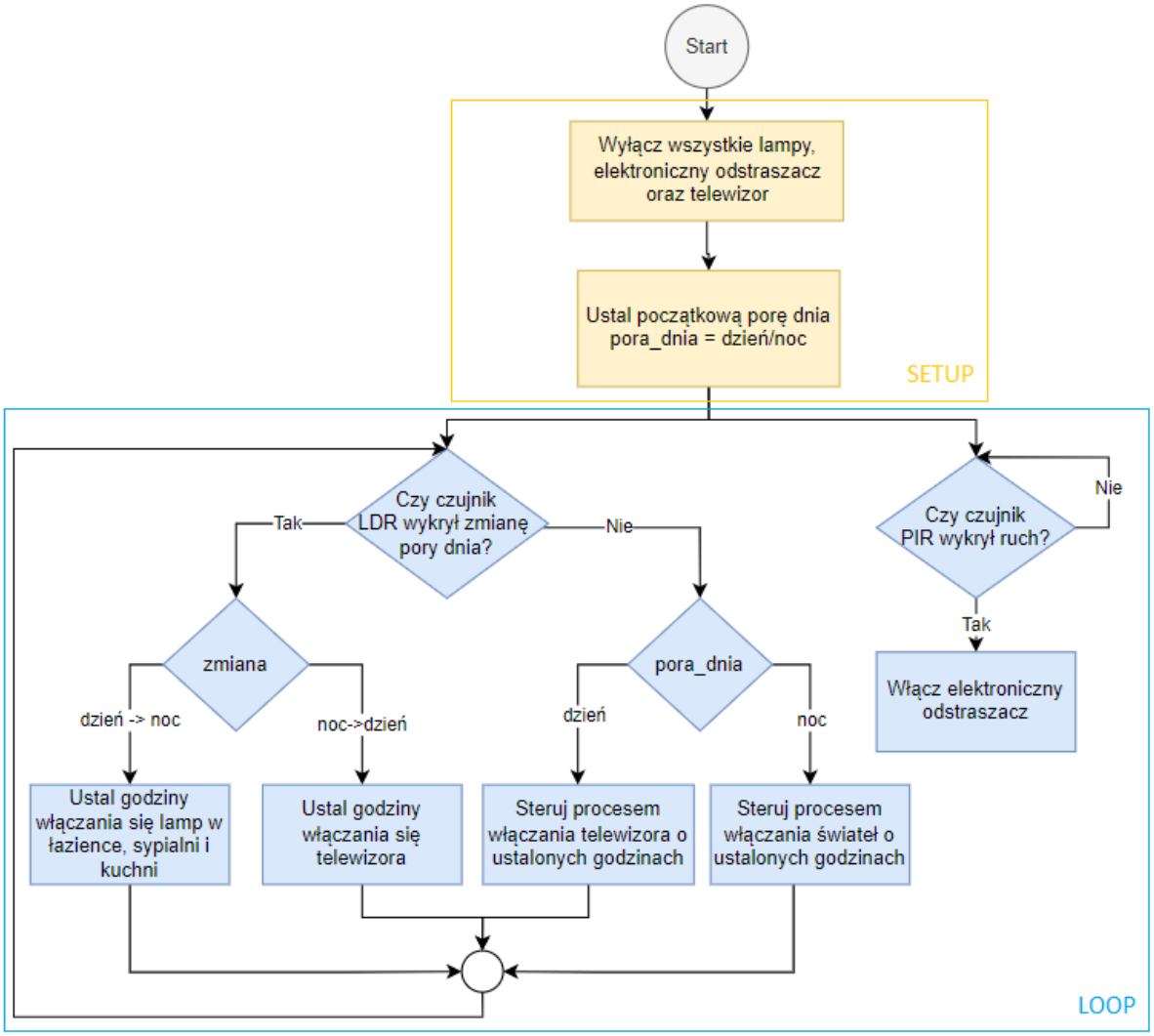
- ***Opóźnienia_TV*** - czasy, po których uruchomiony zostanie telewizor licząc od momentu wykrycia dnia,
- ***Czasy_TV*** – długości trwania kolejnych sesji oglądania telewizji.

Momenty, w których telewizor jest włączony dobrze obrazuje poniższy schemat.




*Rys. 3. Schemat działania funkcji sterującej telewizorem*

# 3. Schemat blokowy oprogramowania




*Rys. 4. Schemat blokowy oprogramowania*

Na powyższym zdjęciu przedstawiono schemat blokowy oprogramowania. Na początku wykonywane są operacje konfiguracyjne, znajdujące się w sekcji *setup()*. Do tych operacji zalicza się wyłączenie wszystkie urządzeń wchodzących w skład symulatora (telewizor, elektroniczny odstraszacz, lampy) oraz ustalenie początkowej pory dnia. Następnie zaczyna cyklicznie wykonywać się główny program, znajdujący się w sekcji *loop()*. Najpierw, czujnik LDR sprawdza czy nie doszło do zmiany pory dnia. W przypadku gdy do tej zmiany doszło, ustalane są godziny włączania się urządzeń: lamp - dla zmiany dzień->noc lub telewizora – dla zmiany noc -> dzień. W przeciwnym wypadku, tzn. gdy nie doszło do zmiany pory dnia, program steruje pracą tych urządzeń zgodnie z wcześniej zaplanowanym harmonogramem. Równolegle cały czas działa czujnik ruchu PIR, który w razie wykrycia ruchu w niepożądanym miejscu uruchomi elektroniczny odstraszacz w formie szczekania psa.

# 4. Kod programu

Kod programu:

```
#include <FastLED.h>
#define NUM_LEDS 77
#define LEDPin 8
CRGB leds[NUM_LEDS];

#define numberOfLedsInICR 24
int ledsToBeON[numberOfLedsInICR] = {12, 15, 16, 18, 19, 20, 23, 25, 29, 31,
34, 36, 40, 41, 42, 45, 47, 51, 52, 56, 59, 60, 62, 64};

#define numberOfLamps 2
#define LDRPin A5
#define PIRPin 12
int relayPin[numberOfLamps]={2, 3};
#define threshold 200
#define hysteresis 150
int restroomRelayPin = 4;


#define TVSessionsNumber 3
int TVPin = 8;
int TVLatency[TVSessionsNumber] = {10, 40, 80};
int TVDuration[TVSessionsNumber] = {10, 10, 10};

int alarmPin = 7;

int eveningLatency[numberOfLamps];
int eveningDuration[numberOfLamps];
int morningDuration[numberOfLamps];
int morningAnticipation[numberOfLamps];
int restroomLatency;
int restroomDuration;
int nightMinDuration = 90;
int nightMaxDuration = 180;
int nightDuration = 120;

int dayMinDuration = 90;
int dayMaxDuration = 180;
int dayDuration = 120;


enum {night, day} nightDay;


class chronometer
{
  unsigned long lastTime, actualTime;
```

```
  public :
  unsigned long elapsed;
  void init();
  void now();
  chronometer();
};

void chronometer::init()
{
  actualTime = millis();
  lastTime = actualTime;
}

void chronometer::now()
{
  actualTime = millis();
  if (actualTime < lastTime)
    elapsed = (0xFFFFFFFFul - lastTime + actualTime);
  else
    elapsed = (actualTime - lastTime);
}

chronometer::chronometer()
{
  actualTime = millis();
  lastTime = actualTime;
}

chronometer lampChrono;
chronometer restroomChrono;
chronometer TVChrono;


void nightDayDetection(void){
  // Wchodimy do IFa jeżeli na zewnątrz jest jasno.
  // Ustawiamy wsztstkie wejścia do przerzutników na LOW.
  // JEŻELI dotychczas była noc, to zmieniamy porę dnia na dzień i mierzymy
jak długo trwała poprzednia noc.
  // -------------------------------------
  if(analogRead(LDRPin) <= threshold)
  {
    for(int i = 0; i < numberOfLamps; i++)
      digitalWrite(relayPin[i], LOW);

    if(nightDay == night)
    {
      Serial.println("NIGHT -> DAY");
      nightDay = day;
      lampChrono.now();
```

```
      restroomChrono.now();
      TVChrono.init();
      if(lampChrono.elapsed> ((unsigned long) nightMinDuration * 1000ul) &&
lampChrono.elapsed < ((unsigned long) nightMaxDuration * 1000ul))
         nightDuration = (int) (lampChrono.elapsed / 1000ul);

      int TVLatencyMin[TVSessionsNumber] = {0, 50, 90};
      int TVLatencyMax[TVSessionsNumber] = {5, 55, 95};
      int TVDurationMin[TVSessionsNumber] = {5, 3, 8};
      int TVDurationMax[TVSessionsNumber] = {20, 15, 25};

      for(int i = 0; i < 3; i++){
        TVLatency[i] = random(TVLatencyMin[i], TVLatencyMax[i]);
        TVDuration[i] = random(TVDurationMin[i], TVDurationMax[i]);
      }
    }
  }
  // -------------------------------------

  // Wchodzimy do ELSE IFa jeżeli na zewnątrz jest ciemno.
  // JEŻELI dotychczas był dzień, to
  //    - zmieniamy go na noc
  //    - wywołujemy lampChrono.init() - zaapamiętuję momen rozpoczęcia się
nocy w polu 'latTime' klasy 'chronometr'
  //    - DLA KAŻDEJ DIODY
  //          wybieramy z użyciem funkcji 'random()'
  //                   jak długo ma świecić dioda wieczorem (eveningDuration)
  //                   jak długo ma świecić dioda rano (morningDuration)
  //          wybieramy z użyciem funkcji 'random()'
  //                   jak dużo czasu ma upłynąć od rozpoczęcia nocy do
zaświecenia diody (eveningLatency)
  //                   jak dużo czasu ma upłynąć od zgaśnięcia diody do
zakończenia nocy (morningAnticipation)
  // -------------------------------------
  else if(analogRead(LDRPin) > threshold + hysteresis)
  {
    if(nightDay == day)
    {
      Serial.println("DAY -> NIGHT");
      nightDay = night;
      lampChrono.init();
      restroomChrono.init();
      TVChrono.now();

      if(TVChrono.elapsed> ((unsigned long) dayMinDuration * 1000ul) &&
TVChrono.elapsed < ((unsigned long) dayMaxDuration * 1000ul))
         dayDuration = (int) (TVChrono.elapsed / 1000ul);

      for (int relayNumber=0; relayNumber<numberOfLamps; relayNumber++)
```

```
      {
        eveningDuration[relayNumber] = random(2, 20);
        eveningLatency[relayNumber] = random(15);
        morningDuration[relayNumber] = random(2, 20);
        morningAnticipation[relayNumber] = random(15);
      }

      restroomLatency = random(20, nightDuration - 20);
      restroomDuration = random(4, 7);
    }
  }
  // ------------------------------------
}

void lampManagement(void){
  int limit1, limit2, limit3, limit4, elapsed_seconds;
  lampChrono.now();
  for(int relayNumber = 0; relayNumber < numberOfLamps; relayNumber++)
  {
    limit1 = eveningLatency[relayNumber];
    limit2 = limit1 + eveningDuration[relayNumber];
    limit3 = nightDuration - morningAnticipation[relayNumber]-
morningDuration[relayNumber];
    limit4 = limit3 + morningDuration[relayNumber];
    elapsed_seconds = (int) (lampChrono.elapsed / 1000);
    if((elapsed_seconds >= limit1 && elapsed_seconds < limit2)||
(elapsed_seconds >= limit3 && elapsed_seconds < limit4))
        digitalWrite(relayPin[relayNumber], HIGH);
      else
        digitalWrite(relayPin[relayNumber], LOW);
  }
}

void alarmManagement(void){
  if(digitalRead(PIRPin))
  {
    digitalWrite(alarmPin, HIGH);
    tone(alarmPin, 60, 2000);
  }
  else
  {
    digitalWrite(alarmPin, LOW);
    noTone(alarmPin);
  }
}

void restroomLampManagement(void){
  int elapsed_seconds;
  restroomChrono.now();
```

```
  elapsed_seconds = (int) (restroomChrono.elapsed / 1000);
  if((elapsed_seconds >= restroomLatency && elapsed_seconds < restroomLatency
+ restroomDuration))
    digitalWrite(restroomRelayPin, HIGH);
  else
    digitalWrite(restroomRelayPin, LOW);
}

void playTV(void){
  for(int i = 0; i < numberOfLedsInICR; i++){
    leds[ledsToBeON[i]] = CRGB::Red;
  }
  FastLED.show();
  delay(500);
    for(int i = 0; i < numberOfLedsInICR; i++){
    leds[ledsToBeON[i]] = CRGB::Black;
  }
  FastLED.show();
  delay(500);
}

void TVManagement(void){
  TVChrono.now();
  int elapsed_seconds = (int) (TVChrono.elapsed / 1000);

  for(int i = 0; i < 3; i++){
    if((elapsed_seconds >= TVLatency[i] && elapsed_seconds < TVLatency[i] +
TVDuration[i]))
      playTV();
  }
}

void setInitialValueForNightDay(void)
{
  while(millis() < 4000)
  {
    if(analogRead(LDRPin) <= threshold)
      nightDay = day;
    else if(analogRead(LDRPin) > threshold + hysteresis)
      nightDay = night;
  }
  if (nightDay == night)
    Serial.println("NIGHT");
  else if (nightDay == day)
    Serial.println("DAY");
}

void setup()
{
```

```
  for (int i=0; i<numberOfLamps; i++)
  {
    pinMode(relayPin[i], OUTPUT);
    digitalWrite(relayPin[i], LOW);
  }
  pinMode(alarmPin, OUTPUT);
  digitalWrite(alarmPin, LOW);

  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  pinMode(12, INPUT);
  Serial.begin(9600);

  FastLED.addLeds<NEOPIXEL, LEDPin>(leds, NUM_LEDS);

  setInitialValueForNightDay();
}

void loop()
{
  nightDayDetection();

  if(nightDay == night){
    lampManagement();
    restroomLampManagement();
  }
  if(nightDay == day){
    TVManagement();
  }

  alarmManagement();
}
```

Plik definiujący połączenia w systemie:

```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 18.66, "left": -89.33,
"attrs": {} },
    { "type": "wokwi-relay-module", "id": "relay1", "top": -201.4, "left":
384, "attrs": {} },
    {
      "type": "wokwi-photoresistor-sensor",
      "id": "ldr1",
      "top": 396.86,
```

```
      "left": -4.73,
      "rotate": 270,
      "attrs": {}
    },
    {
      "type": "wokwi-led",
      "id": "Room_1",
      "top": -198.2,
      "left": 750.4,
      "rotate": 90,
      "attrs": { "color": "yellow", "label": "SYPIALNIA" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r2",
      "top": -168.55,
      "left": 603.4,
      "rotate": 180,
      "attrs": { "value": "220" }
    },
    { "type": "wokwi-relay-module", "id": "relay2", "top": -278.2, "left":
384, "attrs": {} },
    {
      "type": "wokwi-led",
      "id": "Room_2",
      "top": -275,
      "left": 750.4,
      "rotate": 90,
      "attrs": { "color": "green", "label": "KUCHNIA" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": -245.35,
      "left": 603.4,
      "rotate": 180,
      "attrs": { "value": "220" }
    },
    {
      "type": "wokwi-pir-motion-sensor",
      "id": "pir1",
      "top": -154.84,
      "left": -189.43,
      "attrs": {}
    },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": 177.9,
```

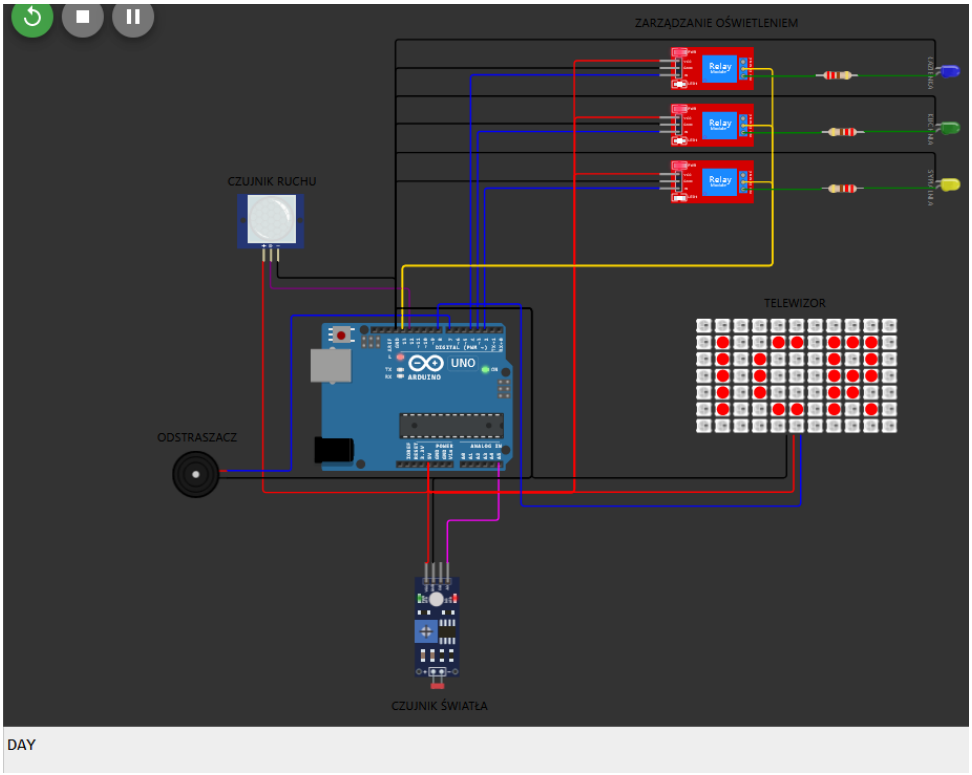
```
      "left": -281.1,
      "rotate": 270,
      "attrs": { "volume": "0.1", "label": "SYPIALNIA" }
    },
    { "type": "wokwi-relay-module", "id": "relay3", "top": -355, "left": 384,
"attrs": {} },
    {
      "type": "wokwi-led",
      "id": "Restrrom",
      "top": -351.8,
      "left": 750.4,
      "rotate": 90,
      "attrs": { "color": "blue", "label": "ŁAZIENKA" }
    },
    {
      "type": "wokwi-neopixel-matrix",
      "id": "TV",
      "top": 14.04,
      "left": 432.74,
      "attrs": { "rows": "7", "cols": "11" }
    },
    {
      "type": "wokwi-text",
      "id": "text1",
      "top": -20.42,
      "left": 526.34,
      "attrs": { "text": "TELEWIZOR", "color": "white" }
    },
    {
      "type": "wokwi-text",
      "id": "text2",
      "top": 162,
      "left": -297.21,
      "attrs": { "text": "ODSTRASZACZ" }
    },
    {
      "type": "wokwi-text",
      "id": "text3",
      "top": -186.12,
      "left": -202.09,
      "attrs": { "text": "CZUJNIK RUCHU" }
    },
    {
      "type": "wokwi-text",
      "id": "text4",
      "top": 527,
      "left": 19.62,
      "attrs": { "text": "CZUJNIK ŚWIATŁA" }
    },
```

```
    {
      "type": "wokwi-text",
      "id": "text5",
      "top": -399.9,
      "left": 350.25,
      "attrs": { "text": "ZARZĄDZANIE OŚWIETLENIEM" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r4",
      "top": -320.9,
      "left": 604.79,
      "rotate": 180,
      "attrs": { "value": "220" }
    }
  ],
  "connections": [
    [ "relay1:GND", "uno:GND.1", "black", [ "h0" ] ],
    [ "uno:2", "relay1:IN", "blue", [ "h-0.15", "v-115.96" ] ],
    [ "uno:5V", "relay1:VCC", "red", [ "v40.62", "h198.13", "v-509.98" ] ],
    [ "uno:A5", "ldr1:AO", "magenta", [ "v77.84", "h-69.47" ] ],
    [ "led1:C", "uno:GND.1", "black", [ "h-184", "v198.59" ] ],
    [ "led1:A", "r2:1", "green", [ "v0" ] ],
    [ "relay1:COM", "uno:13", "gold", [ "h39.6", "v113.59", "h-501.93" ] ],
    [ "ldr1:GND", "uno:GND.1", "black", [ "v-115.2", "h134", "v-230.4", "h-
185.03" ] ],
    [ "relay2:NO", "r1:2", "green", [ "h0" ] ],
    [ "r1:1", "led2:A", "green", [ "v0" ] ],
    [ "led2:C", "uno:GND.1", "black", [ "h-51.61", "v-0.33", "h-470.08" ] ],
    [ "uno:3", "relay2:IN", "blue", [ "v-267.66", "h6.33", "v0.2" ] ],
    [ "relay2:GND", "uno:GND.1", "black", [ "v-1.31", "h-393.12" ] ],
    [ "uno:5V", "relay2:VCC", "red", [ "v39.44", "h198.84", "v-422.4" ] ],
    [ "uno:13", "relay2:COM", "gold", [ "v-85.26", "h501.93", "v-104.2" ] ],
    [ "uno:GND.1", "Room_1:C", "black", [ "v-238.86", "h722.63" ] ],
    [ "r1:1", "Room_2:A", "green", [ "v0" ] ],
    [ "uno:GND.1", "Room_2:C", "black", [ "v-315.66", "h732.23" ] ],
    [ "uno:5V", "pir1:VCC", "red", [ "v40.11", "h-224.27", "v-313.11" ] ],
    [ "pir1:GND", "uno:GND.1", "black", [ "v21.63", "h255.37" ] ],
    [ "pir1:OUT", "uno:12", "purple", [ "v36.02", "h188.68" ] ],
    [ "uno:GND.1", "bz1:1", "black", [ "v-27.66", "h182.94", "v230.4" ] ],
    [ "bz1:2", "uno:7", "blue", [ "v0.4", "h86.4", "v-211.2", "h214.87" ] ],
    [ "uno:GND.1", "relay3:GND", "black", [ "v0" ] ],
    [ "relay3:IN", "uno:4", "blue", [ "v-0.2", "h-255.83" ] ],
    [ "uno:5V", "relay3:VCC", "red", [ "v39.44", "h198.13", "v-585.6" ] ],
    [ "uno:13", "relay3:COM", "gold", [ "v-85.26", "h501.93", "v-267.4" ] ],
    [ "uno:GND.1", "Restrrom:C", "black", [ "v-392.46", "h732.23" ] ],
    [ "uno:5V", "TV:VCC", "red", [ "v39.44", "h495.73" ] ],
    [ "uno:GND.1", "TV:GND", "black", [ "v-27.66", "h183.71", "v230.4",
"h346.92" ] ],
```

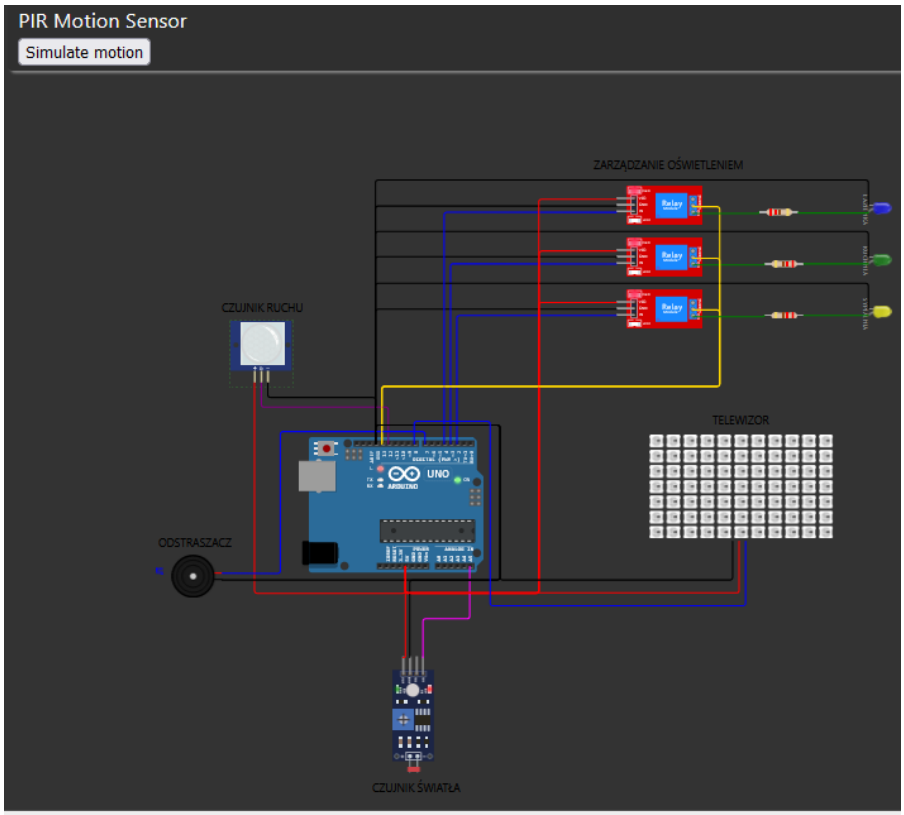
```
    [ "uno:8", "TV:DIN", "blue", [ "v-33.24", "h112.88", "v274.38", "h379.45"
] ],
    [ "relay1:NO", "r2:2", "green", [ "h0" ] ],
    [ "r2:1", "Room_1:A", "green", [ "v0" ] ],
    [ "uno:5V", "ldr1:VCC", "red", [ "v0" ] ],
    [ "relay3:NO", "r4:2", "green", [ "h0" ] ],
    [ "r4:1", "Restrrom:A", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

## 5. Testy oprogramowania

Podczas testów sprawdzono wszystkie zaimplementowane funkcje. W każdym z rozważanych przypadków zachowanie system było zgodne z oczekiwaniami. Poniżej znajdują się zrzuty ekranu reprezentujące wyniki poszczególnych testów.

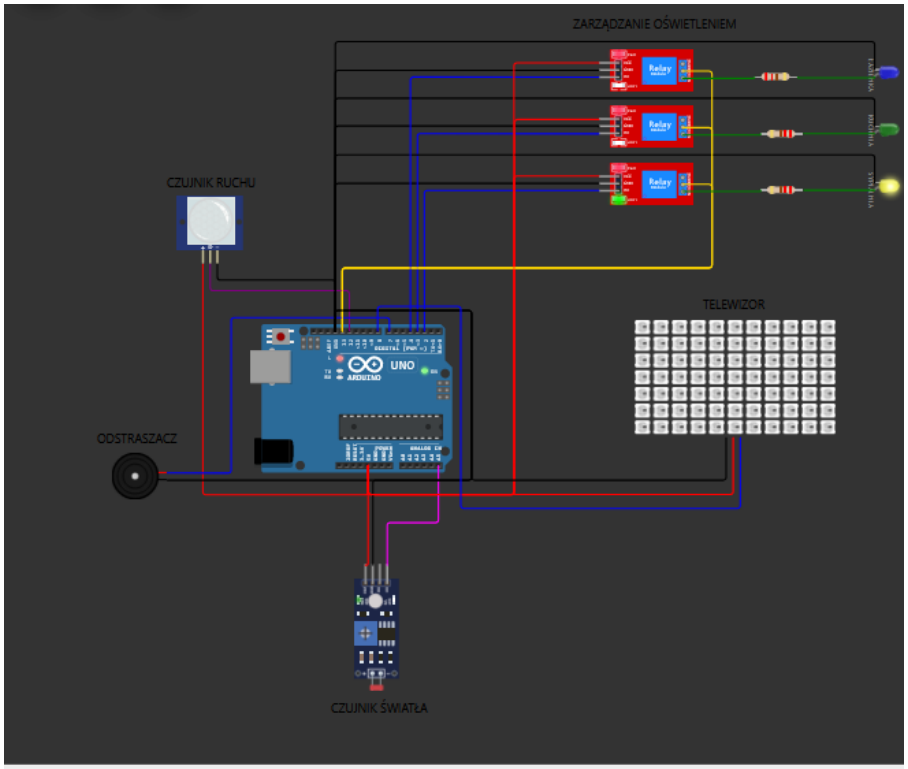



*Rys. 5. Test funkcji sterującej telewizorem*



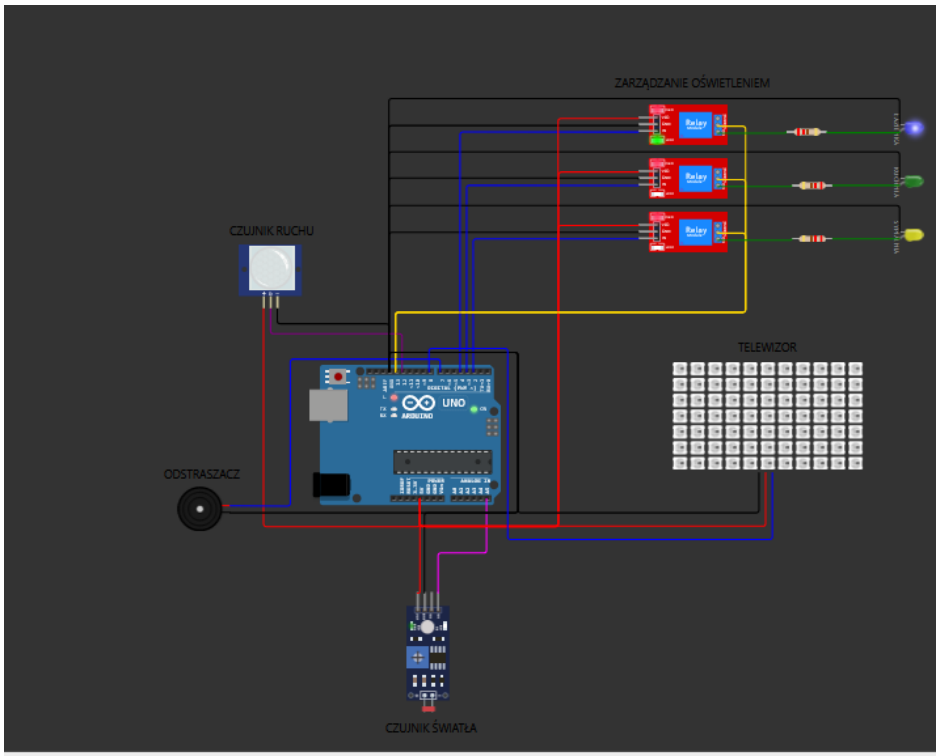

```
DAY
```

*Rys. 6. Test funkcji aktywującej odstraszacz*



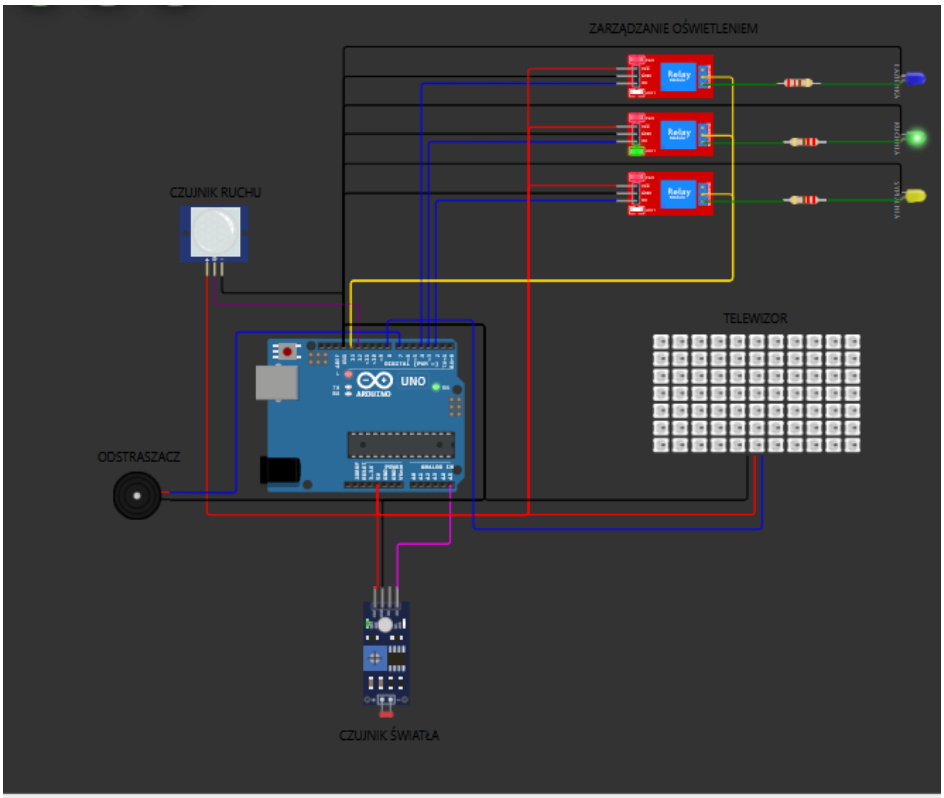

```
DAY
DAY -> NIGHT
```

*Rys. 7. Test funkcji sterującej oświetleniem - Sypialnia*




```
DAY
DAY -> NIGHT
```

*Rys. 8. Test funkcji sterującej oświetleniem – Łazienka*




```
DAY
DAY -> NIGHT
```

*Rys. 9. Test funkcji sterującej oświetleniem - Kuchnia*

# 6. Rozwój projektu

W ramach dalszego rozwoju projektu można byłoby się zastanowić nad zaimplementowaniem następujących funkcjonalności:

1. Powiązanie systemu z większą ilością urządzeń IoT np. zasuwające się rolety, uruchamiające się zraszacze w ogrodzie czy włączające się radio
2. Bardziej losowe włączanie się świateł np. kilkukrotne w ramach jednej nocy, a podczas innej lampy nie zaświeciłyby się w ogóle.
3. Zastosowanie referencyjnego czujnika światła. Obecnie, gdy główny czujnik ustawi się np. w mocno zacienionym miejscu może dojść do sytuacji, gdzie ciągle wykrywana jest noc. Zaburza to poprawne działanie programu.
4. Wysyłanie powiadomień do właściciela domu w przypadku, gdy czujnik ruchu wykrył pewne niepożądane ruchy. Wówczas nawet z odległego miejsca mógłby powiadomić on właściwe służby udaremniając próbę potencjalnej kradzieży.

# 7. Wnioski

Projekt polegał na stworzeniu symulatora obecności w mieszkaniu. W ramach niego nauczyliśmy się tworzyć proste układy z wykorzystaniem płytki Arduino. Temat projektowania układów wbudowanych był dla nas zupełnie nowy, co początkowo wzbudzało w nas pewne obawy. W kolejnych etapach budowy zapoznawaliśmy się z działaniem różnych czujników światła oraz ruchu. Interesującymi częściami układu były także: buzzer imitujący elektroniczny odstraszacz w formie szczekania psa oraz telewizor złożony z LEDów *NeoPixel*.

Zaimplementowaliśmy wiele różnych logik włączania się świateł w różnych częściach domu. W zależności od pory dnia (dzień/noc) włączają się różne urządzenia pracując w losowo określonych zakresach czasu. Ponadto, dołączając kolejne elementy układu uczyliśmy się poprawnej ich konfiguracji, zarówno po stronie podłączenia w układzie jak i implementacji w samym kodzie.

Na koniec przeprowadziliśmy szereg testów, które wykazały prawidłowe działanie stworzonego przez nas symulatora. W każdym z rozważanych przypadków testowych system zachowywał się zgodnie z oczekiwaniami.

Osiągnęliśmy założone cele i jednocześnie poszerzyliśmy naszą wiedzę oraz umiejętności w zakresie programowania systemów czasu rzeczywistego, zatem projekt możemy uznać za zakończony sukcesem.

# Załączniki

Link do repozytorium GitHub: https://github.com/DominikCzyzyk32/Presence-simulator