



Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Projekt z OSP

SuperEliteControlAndDataAcquisitionSmartRoom

Spis treści

Wprowadzenie.....	3
Definicja pojęć	3
Wymagane oprogramowanie.....	4
Konfiguracja systemu	4
Hardware	6
Wybór mikroprocesora	6
Programowanie mikroprocesora	7
Sterowanie oświetleniem.....	8
Sterowanie nagrzewnicą	9
Czujnik obecności	11
Kontaktron	12
Czujnik temperatury i wilgotności	13
Połączenie z wyświetlaczem	16
Komunikacja z komputerem	17
Postępowanie w przypadku znanych awarii.....	19
Software.....	22
Control and Indicators.....	22
Diagnostics.....	24
Alarms	25
Configuration	26
Opisy SubVI's	28
Wykaz elementów panelu czołowego	30
Opis kodu.....	35
Kolory RGB	39
Spis rysunków	39
Spis tabel.....	40

Wprowadzenie

Cel naszego projektu to stworzenie środowiska testowego inteligentnego pomieszczenia wraz z układem kontrolno-pomiarowym, stworzenie aplikacji umożliwiającej obsługę naszego urządzenia oraz implementacja komunikacji urządzenia z komputerem. Nasze środowisko testowe imituje prawdziwe pomieszczenie. Jest ono niewielkie w celu przyspieszenia procesów w nim zachodzących. Rolę układu kontrolno-pomiarowego pełni mikrokontroler AVR ATmega128 wraz z odpowiednimi czujnikami pozwalającymi na pomiar temperatury i wilgotności, kontrolę obecności i pobieranie informacji z kontaktronów. Urządzenie steruje nagrzewnicą w celu kontroli temperatury, a także oświetleniem. Komunikacja urządzenia z komputerem jest możliwa za pomocą interfejsu RS232. Monitorowanie oraz sterowanie urządzeniem jest możliwe z poziomu aplikacji stworzonej w środowisku LabVIEW.

Definicja pojęć

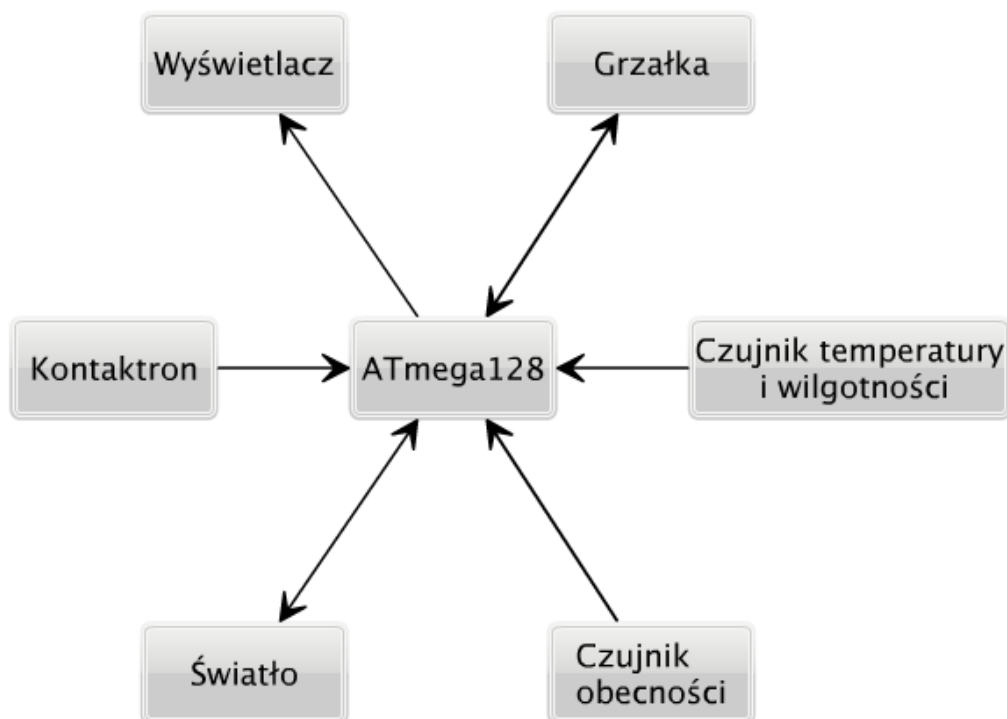
- EHL – Event Handling Loop – Pętla while w kodzie aplikacji SmartRoom zawierająca obsługę zdarzeń wygenerowanych przez użytkownika.
- PWM – Pulse width modulation – sterowanie z modulowaną szerokością impulsów. Wykorzystuje ona ciągły regulator PID oraz sygnał piłokształtny o określonej częstotliwości.
- PID – regulator proporcjonalno-całkująco-różniczkujący – twór matematyczny umożliwiający osiągnięcie zadanej przez użytkownika temperatury w możliwie krótkim czasie bez wystąpienia przeregulowań.
- Producent/Konsument – Schemat programowania aplikacji w LabVIEW polegający na wykorzystaniu dwóch pętli while pracujących równolegle, z których jedna zbiera dane, a druga dane te prezentuje. Komunikacja pomiędzy pętlami następuje z użyciem kolejek, bądź notyfikacji.
- VISA - The Virtual Instrument Software Architecture – biblioteka I/O zdolna do obsługi urządzeń pracujących w interfejsach VXI, GPIB oraz szeregowych (RS232).
- SECADASR – SuperEliteControlAndDataAcquisitionSmartRoom – stanowisko badawcze, czyli system mikroprocesorowy oraz peryferia.
- SmartRoom – aplikacja napisana w środowisku LabVIEW umożliwiająca sterowanie oraz kontrolę procesów zachodzących w SECADASR. Jest to również nazwa całego systemu w którego skład wchodzi SECADASR oraz system SCADA – SmartRoom.

- SubVI – VI będący częścią składową nadrzędnego VI-a
- VI – Virtual Instrument – Nieodzowna część każdej aplikacji napisanej w środowisku LabVIEW. Składa się z panelu czołowego (interfejs użytkownika), diagramu blokowego (zawiera kod źródłowy VI-a) oraz z ikony/panelu połączeń. W naszej aplikacji nadrzędny VI to MainVI.vi.

Wymagane oprogramowanie

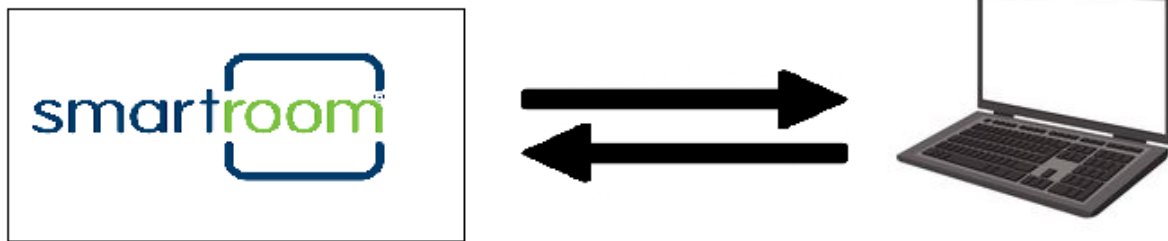
- Lab View Development System wraz z LabVIEW Control Design and Simulation Module, wersja 2016
- NI VISA, wersja 16.0 - Virtual Instrument Software Architecture – standard zapewniający komunikację komputera PC z wykonanym przez nas obiektem badawczym.
- MS Windows 7 i nowsze.

Konfiguracja systemu



RYSUNEK 1 SCHEMAT BLOKOWY UKŁADU ELEKTRONICZNEGO

Aby urządzenie poprawnie działało wystarczy podłączyć układ do komputera PC za pomocą konwertera RS232-USB oraz podłączyć do zasilania nagrzewnicę i oświetlenie. Należy także upewnić się, że układ jest podłączony zgodnie ze schematem ideowym. Szczegóły dotyczące połączenia peryferiów z mikrokontrolerem znajdują się w sekcji **Hardware** niniejszej dokumentacji.

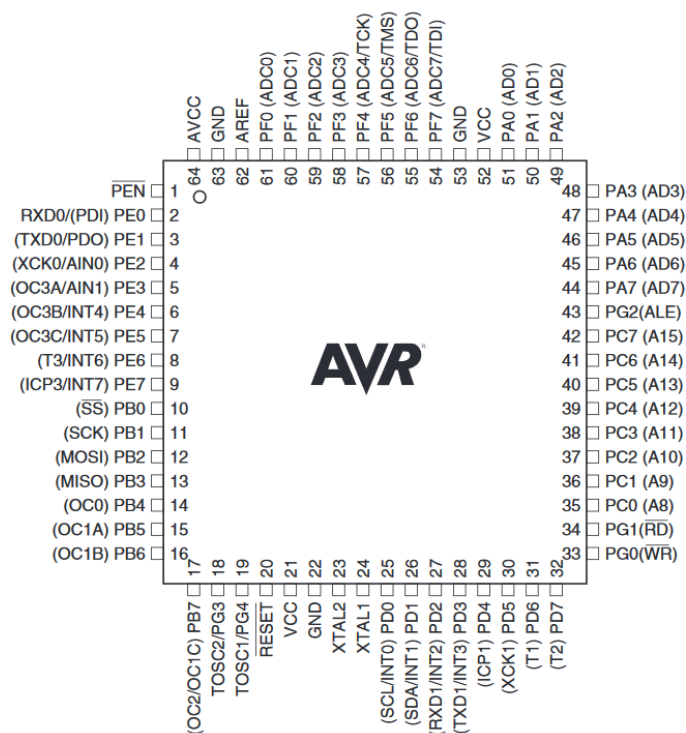


RYSUNEK 1 POŁĄCZENIE SMARTROOM <-> KOMPUTER

Hardware

Wybór mikroprocesora

Jako jednostkę obliczeniową naszego urządzenia wybraliśmy ATmega128. Postanowiliśmy użyć procesora z rodziny AVR aby polepszyć nasze umiejętności w dziedzinie programowania tych mikrokontrolerów.



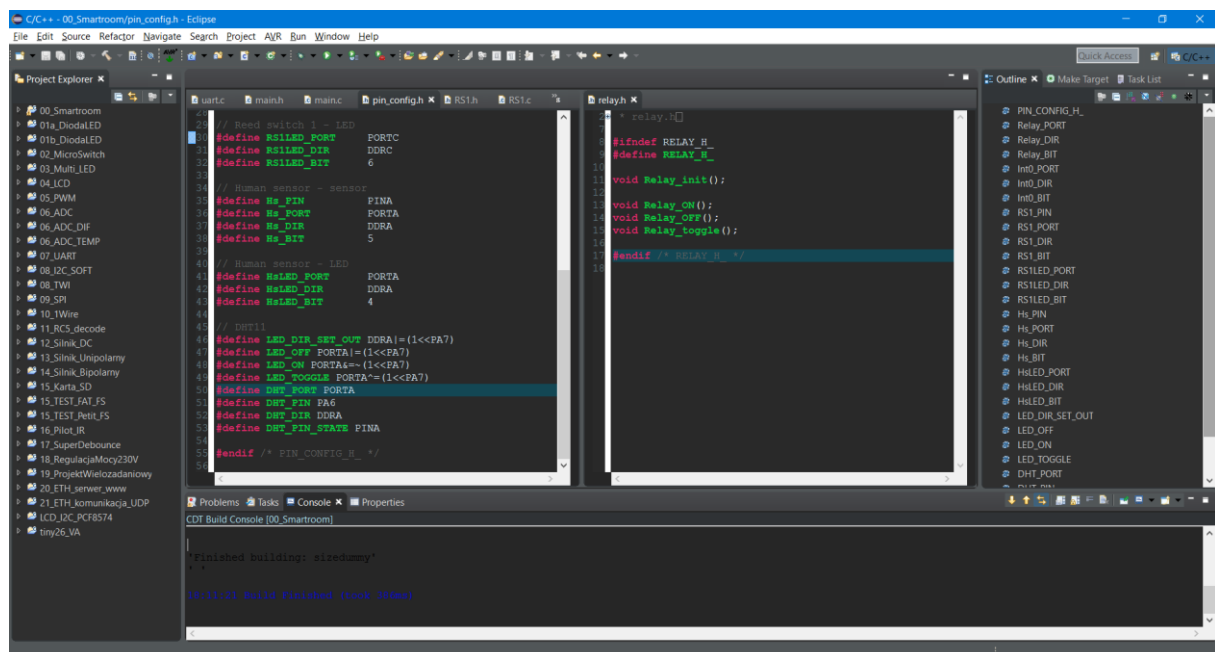
RYСУNEK 2 WYPROWADZENIA MIKROPROCESORA ATMEGA 128

Wybraliśmy wariant w obudowie SMD. Mikrokontroler jest stosunkowo tani, natomiast posiada wiele możliwości. Mikrokontroler posiada 64 piny. Wiele programowalnych wyjść/wejść może przydać się w rozwijaniu naszego projektu. Kolejną zaletą jest 8 pinów, na których możemy użyć przerwania zewnętrznego.

W celu dostania się do pinów mikroprocesora zakupiliśmy adapter QFP64 firmy ATMEL. W następnej kolejności przylutowaliśmy kilka listew goldpinów i w ten sposób nasza płytką z mikroprocesorem została ukończona.

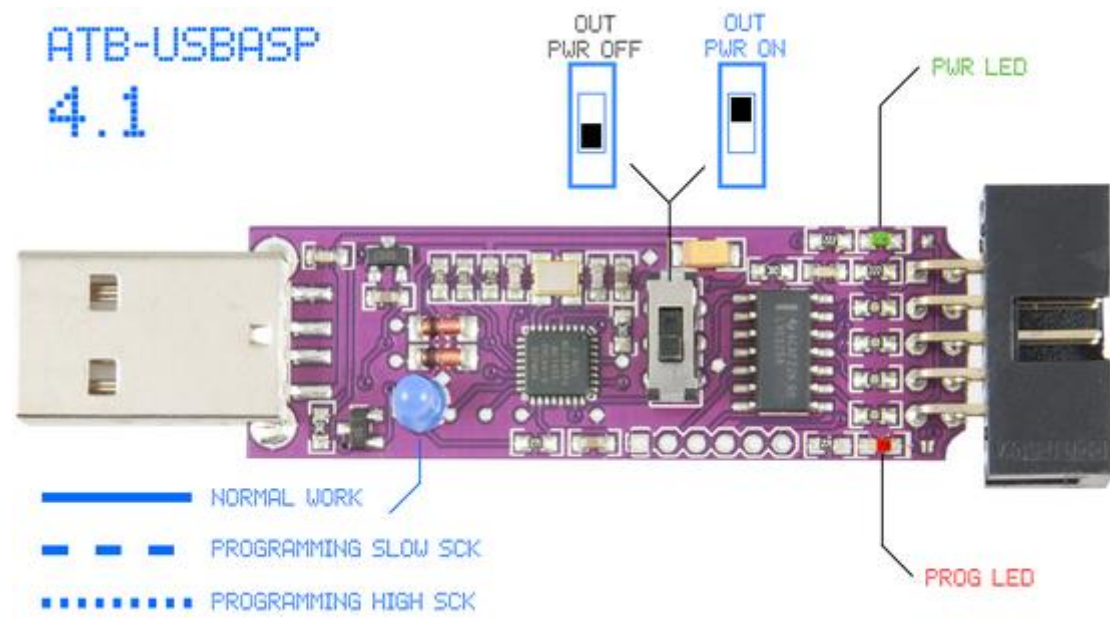
Programowanie mikroprocesora

Program dla naszego mikroprocesora został napisany w języku C w środowisku Eclipse z Atmel AVR Toolchainem.



RYSUNEK 3 ŚRODOWISKO ECLIPSE

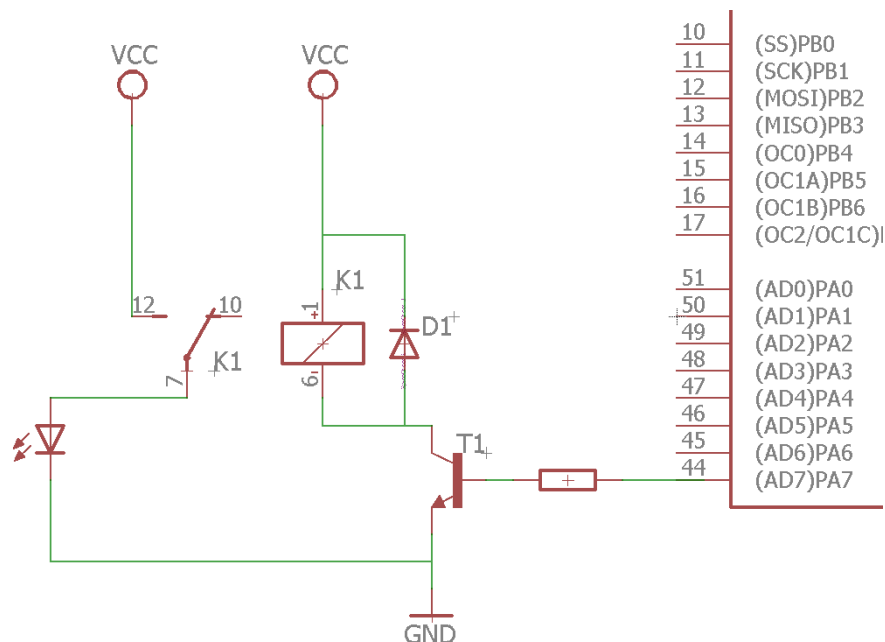
Mikroprocesor został zaprogramowany za pomocą programatora ATB-USBasp firmy Atmel.



RYSUNEK 4 ATB-USBASP ATMEL

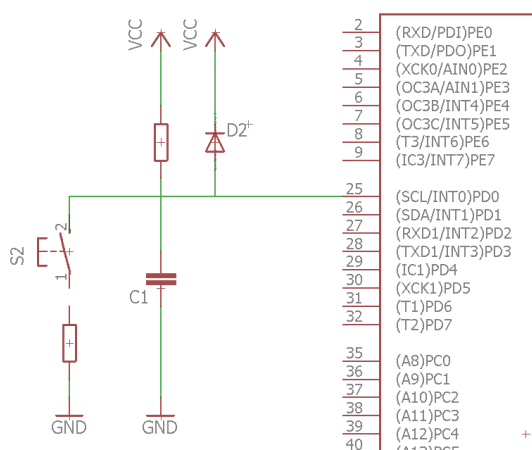
Sterowanie oświetleniem

Sterowanie oświetleniem odbywa się za pomocą klucza tranzystorowego NPN. Wystawiając stan wysoki (5V) na wyjście mikrokontrolera, nasycamy nasz tranzystor, dzięki czemu zasilamy cewkę przekaźnika. Równolegle z cewką zamontowaliśmy diodę półprzewodnikową dla bezpieczeństwa.

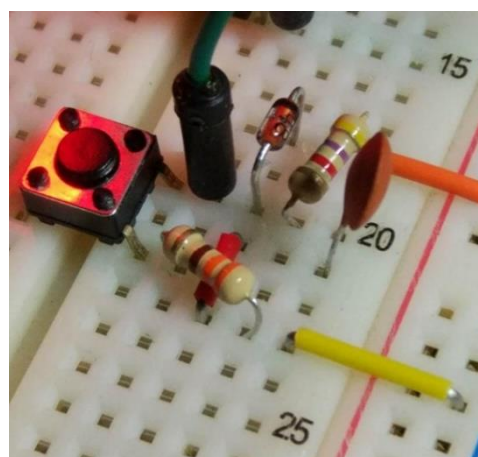


RYSUNEK 5 STEROWANIE OŚWIEPLENIEM

Zmiana stanu oświetlenia oprócz z poziomu aplikacji jest dostępna za pomocą przełącznika typu micro/switch. Charakterystycznym problemem używania tego typu przełączników jest powstawanie drgań na stykach przełącznika. Problem ten został rozwiązany dzięki sprzętowemu usuwaniu drgań styków.



RYSUNEK 6 SPRZĘTOWE USUWANIE DRGAŃ STYKÓW (SCHEMAT)



RYSUNEK 7 SPRZĘTOWE USUWANIE DRGAŃ STYKÓW (ZDJĘCIE)

Część programowa wykorzystuje obsługę przerwań zewnętrznych. Zgodnie z instrukcją Atmegi128 są wykorzystane odpowiednie rejestry wykorzystując zbocze narastające sygnału. Przerwanie jest ustawione na pinie PD0.

```
void Int0_init()
{
    Int0_DIR &= ~(1<<Int0_BIT); // input
    Int0_PORT |= (1<<Int0_BIT); // pull-high

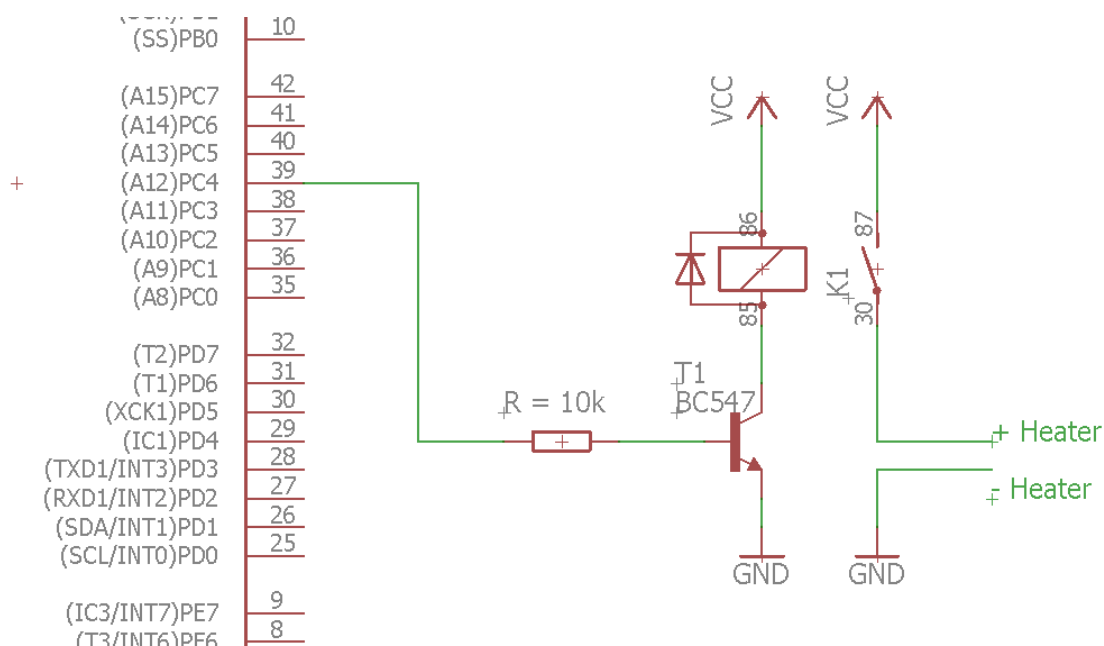
    EICRA |= (1<<ISC00) | (1<<ISC01); // rising edge of interrupt
    EIMSK |= (1<<INT0); // enable interrupt 0
}

void Relay_ON()
{
    Relay_PORT |= (1<<Relay_BIT);
    light = 1;
}

void Relay_OFF()
{
    Relay_PORT &= ~(1<<Relay_BIT);
    light = 0;
}

// Interrupt on PIN 2 PORT D (INT0)
ISR(INT0_vect)
{
    if(light == 0)
        Relay_ON();
    else if(light == 1)
        Relay_OFF();
}
```

RYSUNEK 8 FRAGMENT KODU PRZEDSTAWIAJĄCY OBSŁUGĘ PRZERWANIA



RYSUNEK 9 SCHEMAT UKŁADU STEROWANIA NAGRZEWNICĄ

W zależności od odebranego znaku w kodzie ascii następuje włączenie bądź wyłączenie oświetlenia lub nagrzewnicy.

```
// odbieranie
tmp = uart_getc();

if(tmp == 33) // "!" Light on
    Relay_ON();

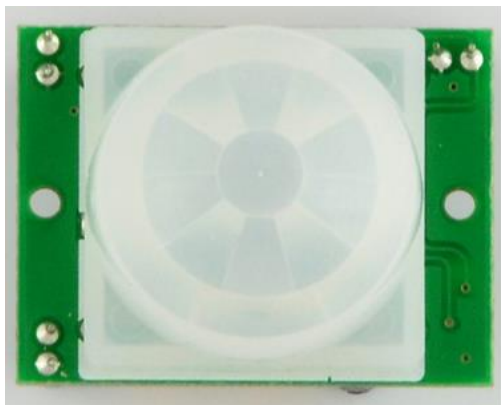
if(tmp == 64) // "@" Light off
    Relay_OFF();

if(tmp == 35) // "#" Heater on
    Heater_ON();

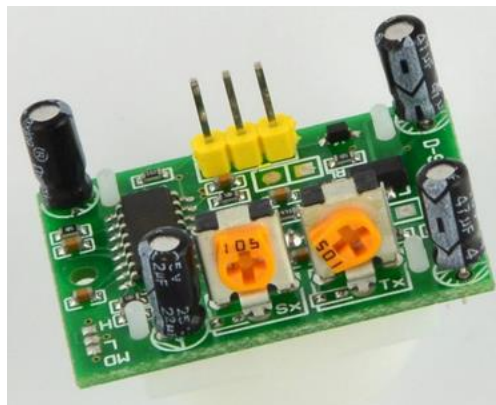
if(tmp == 36) // "$" Heater off
    Heater_OFF();
```

Czujnik obecności

Czujnik obecności wykorzystany w projekcie jest bardzo popularny i chyba najtańszy na rynku - PIR HC-SR501. Jest to pasywny czujnik podczerwieni wykrywający ruch. Posiada on detektor podczerwieni, a także plastikową soczewkę skupiającą światło. Zasada działania opiera się na pomiarze temperatury. Jej wzrost lub spadek uruchamiają alarm.



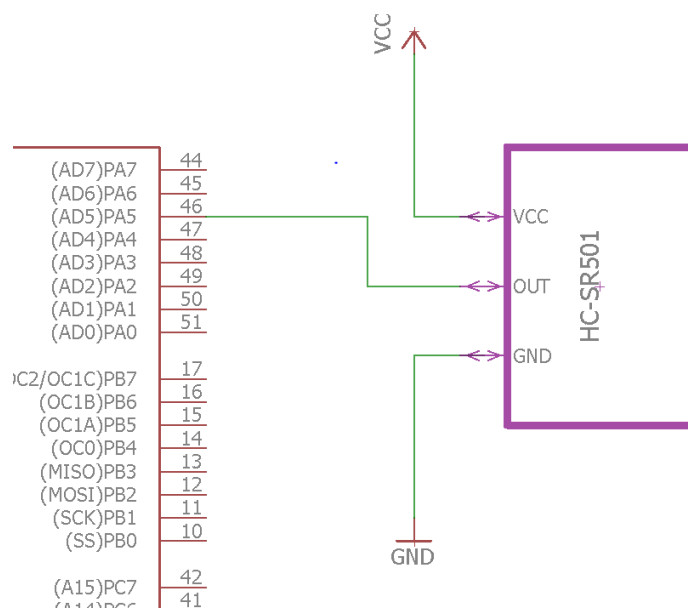
RYSUNEK 10 PIR - SOCZEWKA



RYSUNEK 11 PIR SPÓD

Czujnik posiada dwa potencjometry:

- T1, za pomocą którego regulujemy czas trwania stanu wysokiego alarmy po wykryciu obiektu
- T2, dzięki któremu regulujemy czułość, czyli dystans w którym wykrywany jest ruch



RYSUNEK 12 PIR SCHEMAT POŁĄCZENIA

```
#define Hs_detection() (Hs_PIN & (1<<Hs_BIT))
```

RYSUNEK 13 PIR MASKOWANIE BITOWE

Do detekcji stanu wysokiego na pinie czujnika oznaczającego alarm użyty został mechanizm maskowania bitowego, następnie zapalana zostaje dioda symbolizująca wykrycie ruchu, a także ustawienie zmiennej globalnej służącej do przechowywania informacji o stanie czujnika.

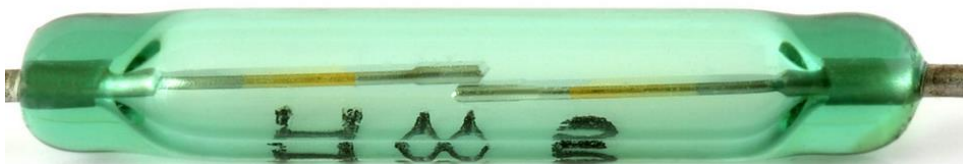
```
void HsLED_ON(void)
{
    HsLED_PORT |= (1<<HsLED_BIT);
    human_sensor = 1;
}

void HsLED_OFF(void)
{
    HsLED_PORT &= ~(1<<HsLED_BIT);
    human_sensor = 0;
}
```

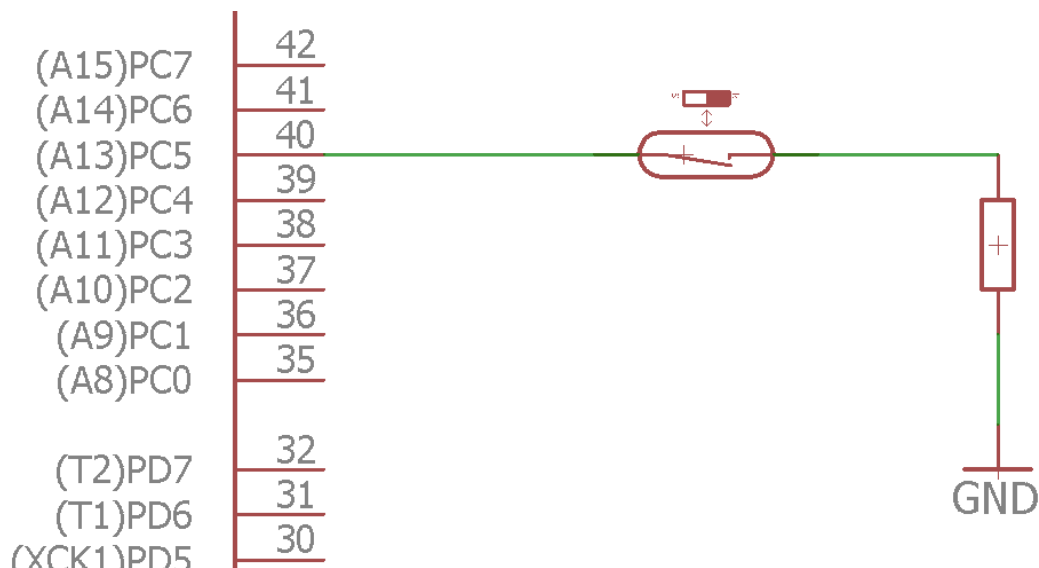
RYSUNEK 14 PIR STEROWANIE LED

Kontaktron

Kolejnym elementem systemu alarmowego jest kontaktron. Kontaktron to odpowiednio uformowane blaszki zamknięte w szklanej rurce, które pod wpływem pola magnetycznego stykają się tworząc zwarcie obwodu. W praktyce taki kontaktron wraz z magnesem montuje się w drzwiach lub oknach, dzięki czemu jesteśmy w stanie sprawdzić stan otwarcia lub zamknięcia.



RYSUNEK 15 KONTAKTRON - ZDJĘCIE



RYSUNEK 16 KONTAKTRON - SCHEMAT POŁĄCZENIA

Ustawiony zostaje pin, na którym znajduje się kontaktron jako wejście, a także podciągnięty zostaje do VCC za pomocą wewnętrznego rezystora. Drugi styk zwarty do masy.

```
void RS1_init()
{
    // Reed switch
    RS1_DIR  &= ~(1<<RS1_BIT); // input
    RS1_PORT |= (1<<RS1_BIT); // pull-high

    // LED
    RS1LED_DIR |= (1<<RS1LED_BIT);
}
```

RYSUNEK 17 KONTAKTRON - INICJALIZACJA JAKO WEJŚCIA W SYSTEMIE

Ustawienie maski jest analogiczne jak dla czujki PIR. Różnicą jest zanegowanie wyrażenia w celu otrzymania na wejściu stanu niskiego po aktywacji kontaktronu.

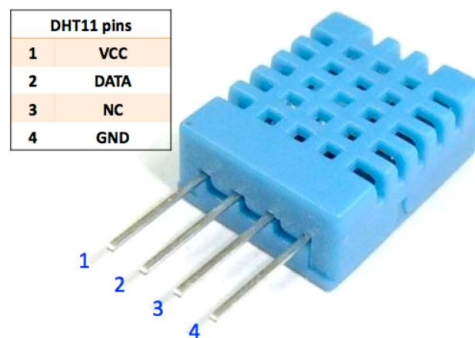
```
#define RS1_detection() !(RS1_PIN & (1<<RS1_BIT))
```

RYSUNEK 18 KONTAKTRON - ZANEGOWANE WYRAŻENIE MASKI

Czujnik temperatury i wilgotności

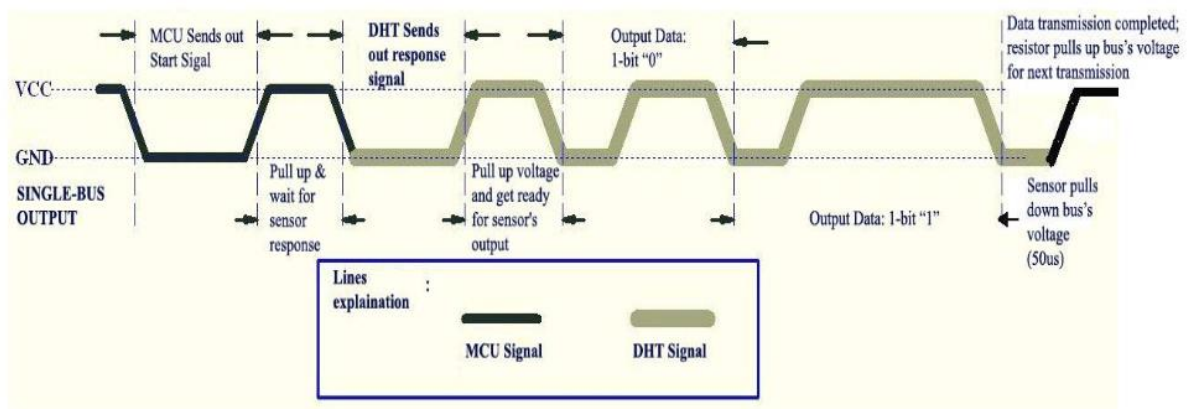
Kolejnym elementem naszego urządzenia jest czujnik wilgotności i temperatury DHT11. Wartości mierzone tego elementu konwertowane są do postaci cyfrowej, a komunikacja z mikroprocesorem odbywa się z wykorzystaniem interfejsu szeregowego. Czujnik jest bardzo

popularny i bardzo tani, a co za tym idzie jest wolny i niezbyt dokładny, jednak dla naszych zastosowań taki czujnik wystarczy.

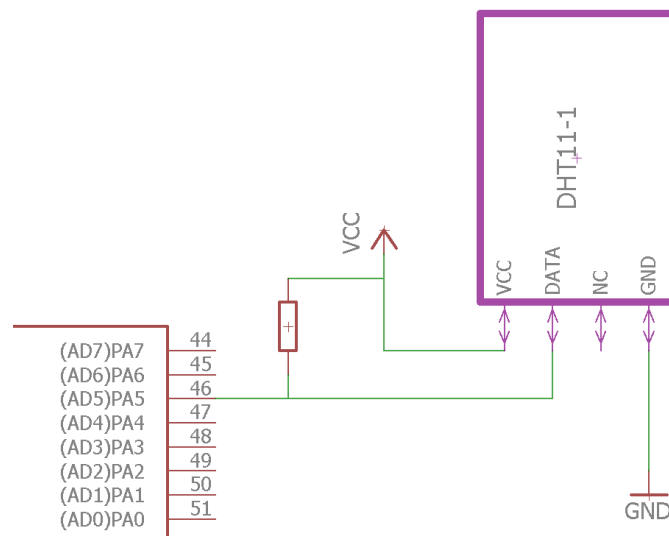


RYSUNEK 19 DHT11 - ZDJĘCIE

Sensor wysyła 40 bitów danych, po 5 bitów w jednym ciągu. Zestaw bitów ma następującą strukturę: Dane = Całkowity bajt wilgotności + Dziesiętny bajt wilgotności + Całkowity bajt temperatury + Dziesiętny bajt temperatury + Bajt sumy kontrolnej.



RYSUNEK 20 ZASADA PRACY CZUJNIKA DHT11



RYSUNEK 21 DHT11 - SCHEMAT POŁĄCZENIA

```

/***** Przesył danych *****/

for (i = 0; i < 5; ++i) {
    for (j = 7; j >= 0; --j) {
        TCNT0 = 0;

        /* Najpierw 50µs stanu niskiego */
        while (!(DHT_PIN_STATE & (1 << DHT_PIN))) {
            if (TCNT0 >= 70)
                return 0;          //70µs
        }

        TCNT0 = 0;

        /* Teraz dane. 26 to 28µs stanu wysokiego wskazuje
           zero, a około 70µs to wysłana jedynka */

        while (DHT_PIN_STATE & (1 << DHT_PIN)) {
            if (TCNT0 >= 90)
                return 0;          //90µs
        }

        /* żeby nie psuć zmiennej TCNT0, skopiujemy ją sobie */
        cnt = TCNT0;

        if (cnt >= 19 && cnt <= 34 ) {          //pomiędzy 20 i 35µs
            data[i] &= ~(1 << j); //ustaw zero
        }
        else if (cnt >= 59 && cnt <= 79) { //pomiędzy 60 and 80µs
            data[i] |= (1 << j); //jedynka
        }

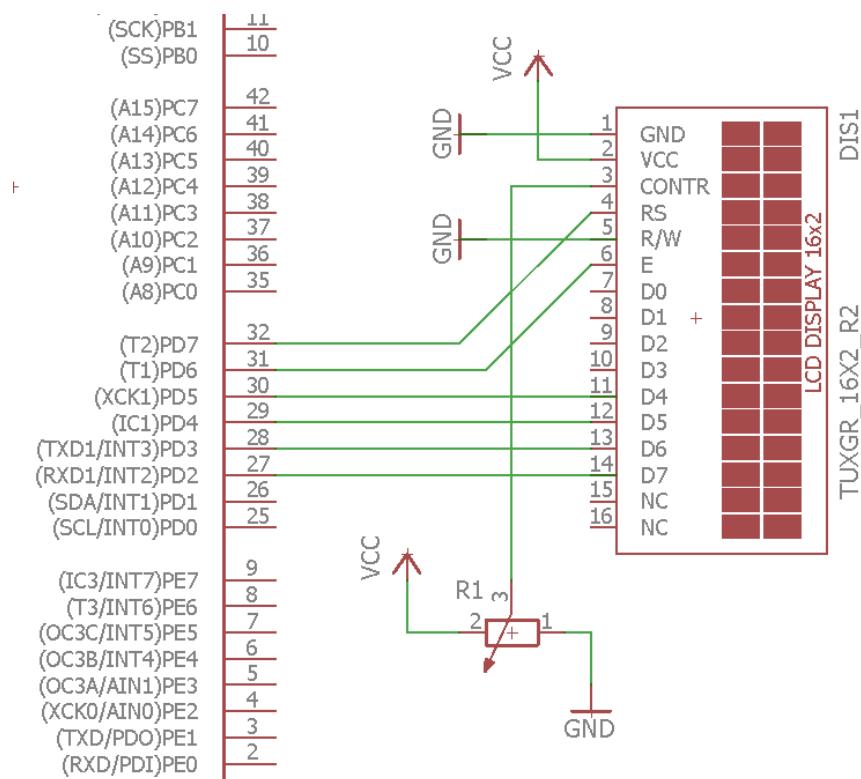
        else
            return 0;
    }
}

```

RYSUNEK 22 FRAGMENT KODU ODBIERANIA DANYCH Z CZUJNIKA DHT11

Połączenie z wyświetlaczem

W celu prezentacji danych z czujnika wilgotności i temperatury skorzystaliśmy z wyświetlacza LCD 2x16 ze sterownikiem hd44780.



RYСУNEK 23 WYŚWIETLACZ LCD - SCHEMAT POŁĄCZENIA

Wykorzystane zostały biblioteki napisane przez Mirosława Kardasia, dzięki czemu wyświetlanie na ekranie było proste.

```
lcd_locate(0,0);

lcd_str(tab1);
lcd_int(11);

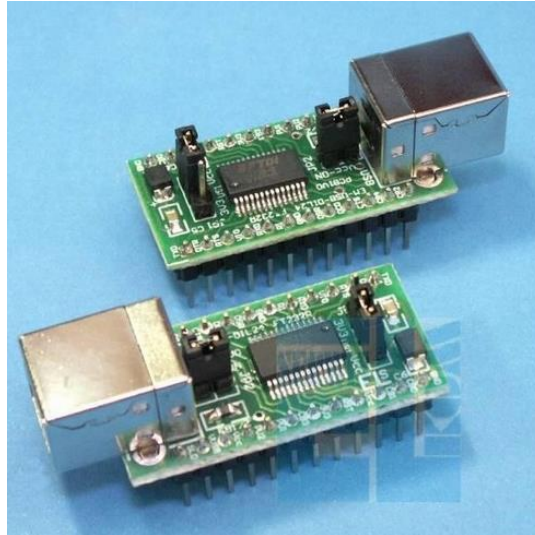
lcd_locate(1,0);

lcd_str(tab2);
lcd_int(12);
```

RYСУNEK 24 LCD - FRAGMENT
REALIZUJĄCY WYŚWIETLANIE TREŚCI

Komunikacja z komputerem

W celu komunikacji mikrokontrolera z PC skorzystaliśmy z interfejsu RS232. Jako, że coraz rzadziej w komputerach występują złącza D-SUB, wykorzystany został moduł EM-216 EM-USB-DIL-24 firmy ELKOM, czyli konwerter USB-RS232. Można wykorzystać ten moduł dzięki sterownikom, które emulują port szeregowy.



RYSUNEK 25 KONWERTER USB-RS232

Do komunikacji zostały użyte najbardziej popularny zestaw parametrów, czyli prędkość 9600, 8 bitów danych, brak bitów parzystości i 1 bit stopu. Cała komunikacja wychodząca i przychodząca oparta jest o procedury obsługi przerwań. Aby całkowicie zapomnieć o procesie oczekiwania na zakończenie wysyłania lub na zakończenie odbioru danych wykorzystaliśmy mechanizm buforowy, a dokładniej bufor cykliczny.

```
void uart_putc( char data ) {
    uint8_t tmp_head;

    tmp_head = (UART_TxHead + 1) & UART_TX_BUF_MASK;

    // pętla oczekuje jeżeli brak miejsca w buforze cyklicznym na kolejne znaki
    while ( tmp_head == UART_TxTail ){}

    UART_TxBuf[tmp_head] = data;
    UART_TxHead = tmp_head;

    // inicjalizujemy przerwanie występujące, gdy bufor jest pusty, dzięki
    // czemu w dalszej części wysyłaniem danych zajmie się już procedura
    // obsługi przerwania
    UCSR0B |= (1<<UDRIE0);
}
```

RYSUNEK 26 DODAWANIE BAJTU DO BUFORA CYKLICZNEGO

Aby wysłać nasze dane uformowaliśmy je w postać ramki wyglądającej następująco:

1. Bit startu – w naszym przypadku jest to '\$'.
2. Część dziesiętna temperatury.
3. Część jedności temperatury.
4. Część dziesiętna wilgotności.
5. Część jedności wilgotności.
6. Stan oświetlenia.
7. Informacja z czujnika obecności.
8. Informacja z kontaktronu.
9. Bit zapasowy.
10. Znak stopu – w naszym przypadku '&'.

```
temp_tens = 11/10;
temp_units = 11%10;

hum_tens = 12/10;
hum_units = 12%10;

frame[1] = temp_tens;           // temperature - tens value
frame[2] = temp_units;         // temperature - units value
frame[3] = hum_tens;           // humidity - tens value
frame[4] = hum_units;          // humidity - units value
frame[5] = light;              // light value
frame[6] = human_sensor;       // human sensor
frame[7] = reed_switch;        // reed switch
frame[8] = 0;

uart_putc(frame[0]);
for(i=1; i<=8; i++)
    uart_putint(frame[i],10);
uart_putc(frame[9]);
```

RYSUNEK 27 IMPLEMENTACJI RAMKI DANYCH

Ramka wysyłana jest w pętli głównej programu z opóźnieniem 1 sekundy. W pętli głównej znajduje się również odbiór danych, dzięki któremu aktualizowana jest informacja o stanie oświetlenia i nagrzewnicy.

```
// odbieranie
tmp = uart_getc();

if(tmp == 33)           // "!" Light on
    Relay_ON();

if(tmp == 64)           // "@" Light off
    Relay_OFF();

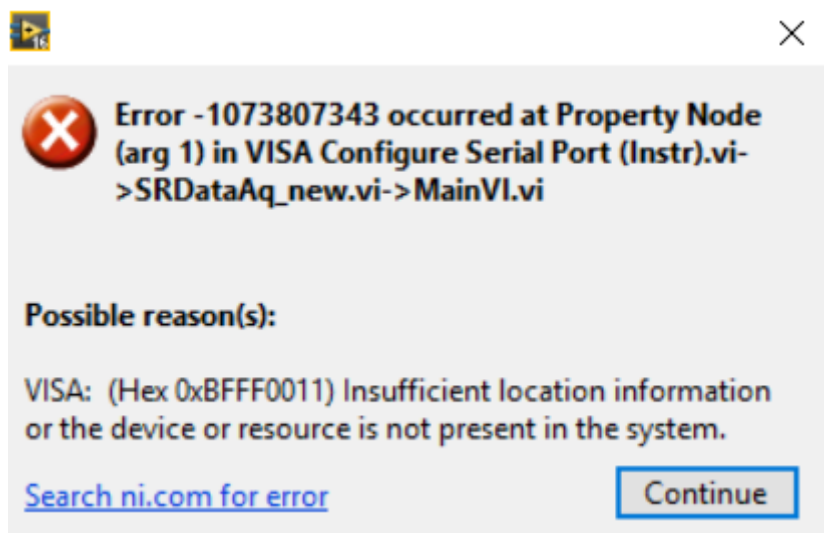
if(tmp == 35)           // "#" Heater on
    Heater_ON();

if(tmp == 36)           // "$" Heater off
    Heater_OFF();
```

RYSUNEK 28 ODBIÓR DANYCH PRZEZ ATMEGE

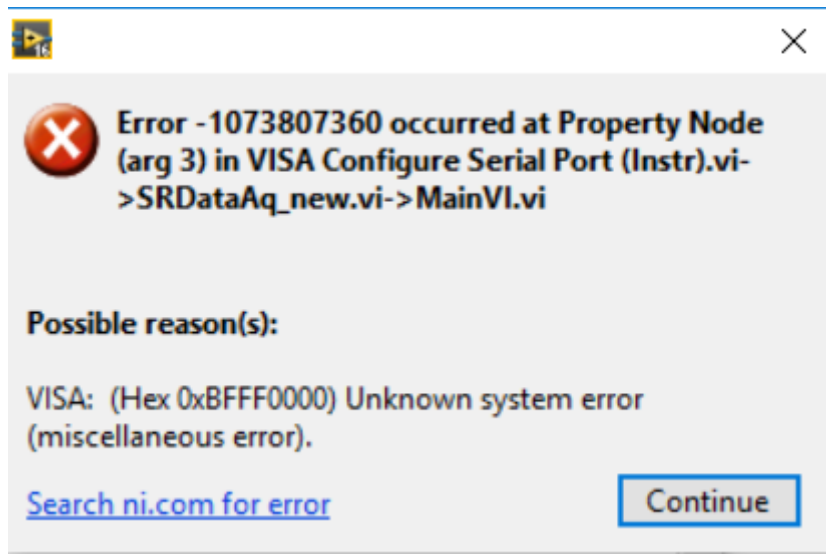
Postępowanie w przypadku znanych awarii

Error -1073807343



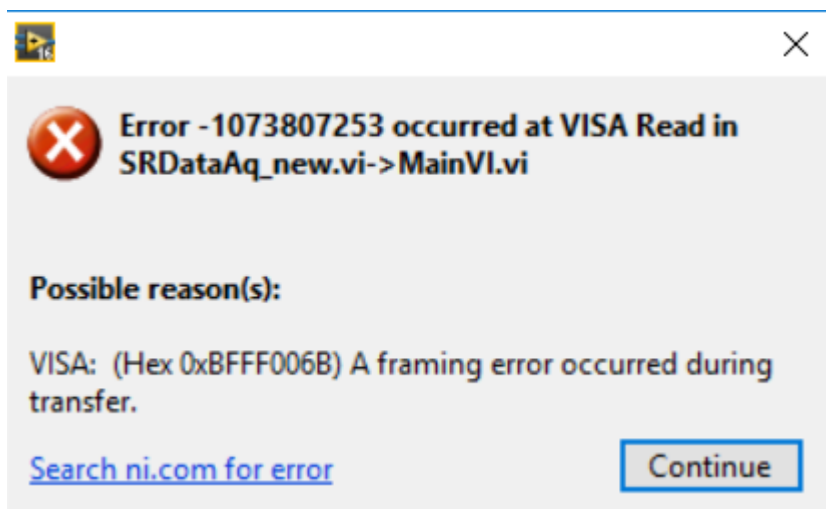
Powyższy błąd jest wynikiem niepoprawnej komunikacji między systemem mikroprocesorowym a aplikacją SmartRoom. By kontynuować pracę należy sprawdzić połączenie między układem mikroprocesorowym a komputerem (np. czy kabel USB jest wpięty do gniazda). Jeśli mimo tego, że układ jest poprawnie podłączony, należy ponownie uruchomić aplikację.

Error -1073807360



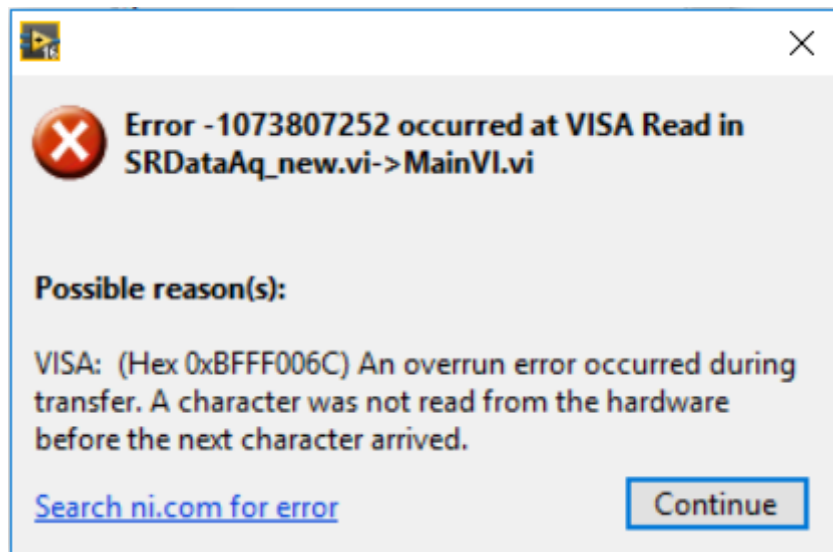
W przypadku wystąpienia błędu po wciśnięciu Continue należy sprawdzić pole data bits w zakładce Configuration. Jego wartość powinna wynosić 8. Jeśli mimo to błąd nadal występuje należy wyłączyć a następnie włączyć aplikację SmartRoom.

Error -1073807253



W przypadku wystąpienia błędu po wciśnięciu Continue należy sprawdzić pole data bits w zakładce Configuration. Jego wartość powinna wynosić 8. Jeśli mimo to błąd nadal występuje należy wyłączyć a następnie włączyć aplikację SmartRoom.

Error -1073807252



Błąd pojawiający się w przypadku gdy czas podjęcia decyzji przez użytkownika podczas wystąpienia innych okien dialogowych błędów jest zbyt długi. Po wciśnięciu Continue błąd nie powinien więcej wystąpić. Jeśli jednak błąd nadal występuje, należy wyłączyć a następnie włączyć aplikację LabVIEW.

Software

Opis interfejsu użytkownika

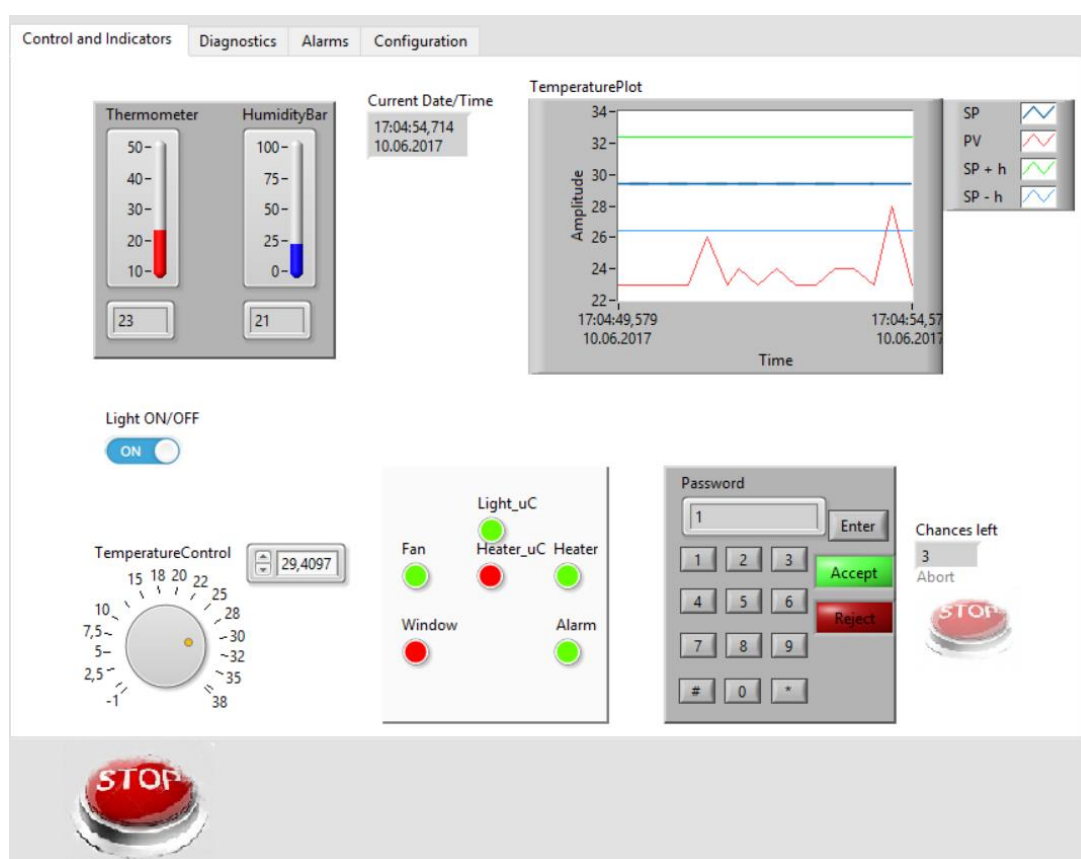
Interfejs użytkownika składa się z 4 zakładek:

- Control and Indicators
- Diagnostics
- Alarms
- Configuration

Po włączeniu aplikacji aktywna jest tylko klawiatura znajdująca się w zakładce Control and Indicators, dzięki której wprowadzamy hasło niezbędne do poprawnego korzystania z aplikacji ('1' a następnie Enter) oraz przycisk „Abort”, który to umożliwia wyłączenie aplikacji zanim poprawne hasło zostanie podane. Cała reszta kontrolerek oraz zakładek jest nieaktywna do czasu wprowadzenia poprawnego hasła.

Poza zakładkami znajdziemy przycisk „STOP”, dzięki któremu możemy wyłączyć aplikację gdy podamy już poprawne hasło.

Control and Indicators



RYSUNEK 29 CONTROLS AND INDICATORS

W tej zakładce jak już wspomnieliśmy, znajduje się klawiatura, poprzez którą wprowadzamy hasło. Kombinację wprowadzamy poprzez wciśnięcie odpowiednich przycisków, a następnie należy zatwierdzić przyciskiem „Enter”. Wprowadzane hasło jest wyświetlane na bieżąco w polu nad klawiaturą. Gdy wpisane hasło jest poprawne zapali się zielona kontrolka „Accept” oraz cały interfejs zostanie odblokowany i będziemy mogli korzystać z aplikacji. W przypadku wprowadzenia niepoprawnego hasła pojawi się powiadomienie „Type correct password!”, zapali się czerwona kontrolka „Reject” oraz liczba szans na wprowadzenie poprawnego hasła zmniejszy się o jeden. Liczba szans jest wyświetlana na prawo od klawiatury. Jeżeli 3 razy wprowadzimy niepoprawne hasło ukaże nam się powiadomienie „You’ve terminated SmartRoom before typing right password”, a następnie aplikacja zostanie automatycznie wyłączona. To samo powiadomienie zostanie wyświetlone, jeśli przed wpisaniem poprawnego hasła użytkownik zakończy działanie aplikacji poprzez wciśnięcie kontrolki „Abort”.

W zakładce Control and Indicators po lewej stronie od klawiatury znajduje się 6 kontroltek, które sygnalizują stany grzałki, czujnika ruchu, oświetlenia oraz alarmu. Kontrolki „Heater_uC” oraz „Light_uC” sygnalizują rzeczywisty stan urządzeń skorelowanych z systemem mikroprocesorowym. Natomiast „Heater” jest odzwierciedleniem pożądanego stanu grzałki. Stany powyższych kontroltek nie zawsze będą się pokrywać, ze względu na opóźnienia wynikające z transmisji danych między mikroprocesorem a komputerem i odwrotnie. Kolejnym elementem na panelu czołowym jest przełącznik do włączania i wyłączania światła. W tym celu użyliśmy znalezionej w internecie ikony ON/OFF. Znajduje się także pokrętło oraz pole numeryczne, za pomocą których zadajemy wartość temperatury jaką chcemy uzyskać w naszym pokoju. Temperaturę możemy zmieniać w zakresie od -1 do 38 stopni Celsjusza.

Zmierzona temperatura oraz wilgotność przedstawiane są w postaci słupka oraz pola numerycznego. Temperatura jest także przedstawiana wraz z wartością zadaną oraz strefą histerezy na wykresie czasu rzeczywistego, który znajduje się w prawym górnym rogu panelu czołowego. Ostatnim elementem w zakładce Control and Indicators jest aktualna data wraz z godziną.

Diagnostics

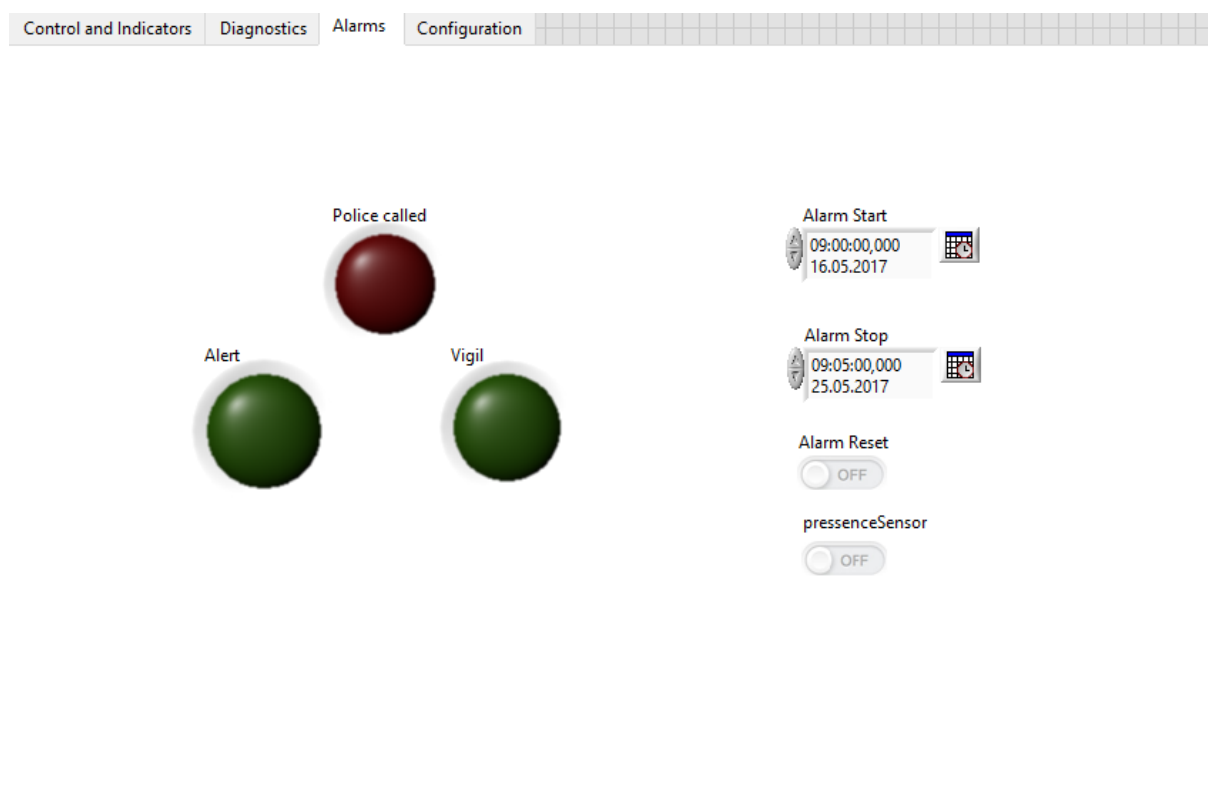


RYSUNEK 30 DIAGNOSTICS

W tej zakładce znajdują się wszystkie potrzebne informacje diagnostyczne. Zauważymy tutaj 5 pól tekstowych, w których wyświetlane są: ramka odczytana przez aplikację SmartRoom w pętli producenta(Prod), oraz konsumenta. Ramka wysłana przez aplikację do mikrokontrolera, pole informujące o błędach odbioru i przesyłu danych oraz pole zawierające liczbę wysłanych w ramce do systemu mikroprocesorowego bajtów. Widoczne jest również pole umożliwiające wpisanie ramki własnej treści.

Pod tymi polami znajdują się 4 kontrolki, które sygnalizują czy zostały odczytane dane z czujnika temperatury i wilgotności, z czujnika obecności oraz z kontaktronu. Po prawej stronie znajdują się 2 wykresy. Jeden z nich przedstawia przebieg aktualnej temperatury, a na drugim wyświetlane są przebiegi odpowiedzi regulatora PID oraz sygnał piłokształtny, niezbędne w celu określenia prawidłowości regulacji PWM.

Alarms



RYSUNEK 31 ALARMS

W tej zakładce znajdują się trzy duże kontrolki LED. Pierwsza czerwona sygnalizuje czy policja została wezwana (ta funkcjonalność nie jest sensu stricto, a jedynie poglądowa), druga zielona jest aktywna gdy alarm zostaje uruchomiony, a ostatnia zielona pokazuje czy włączony jest tryb czuwania. Na prawo od kontrolek są dwa pola, poprzez które ustawiamy czas startu i stopu trybu czuwania. Ostatnimi dwoma elementami są przełącznik ON/OFF oraz kontrolka ON/OFF. Za pomocą przełącznika resetujemy alarm, a kontrolka informuje nas o stanie czujnika obecności.

Configuration

Control and Indicators Diagnostics Alarms Configuration

VISA resource name
COM3

PopUp Time, s
10

Hysteresis width, C
0

PWM Control
ON

Serial Settings

baud rate
9600

data bits
8

parity
None

stop bits
1.0

flow control
XON/XOFF 1

Read
ON

Write
ON

Read Count
10

Command bypassed
ON

Proportional Gain (Kc)
1

Integral time [s] (Ti)
3

Derivative time [s] (Td)
1

Measurements filename
E:\testMEAS.xlsx

STOP

RYSUNEK 32 CONFIGURATION

Na ostatniej zakładce Configuration umieszczone są opcje, za pomocą których możemy konfigurować aplikację tak aby była kompatybilna z naszym SECADASR. Skład zakładki wchodzi:

Pole VISA Resource Name, gdzie wybieramy port, do którego podłączyliśmy nasze urządzenie.

Pole tekstowe PopUp Time, w którym wpisujemy lub wybieramy co ile sekund powinno wyskakiwać powiadomienie o błędnym podłączeniu urządzenia.

Tabela (ściślej klaster), dzięki której możemy wybrać opcje dotyczące transferu danych (prędkość transmisji, ilość bajtów danych itp.).

Przełącznik umożliwiający wymuszenie wysłanie ramki o treści zadanej przez użytkownika do systemu mikroprocesorowego. Treść ramki można modyfikować w polu Command w zakładce Diagnostics.

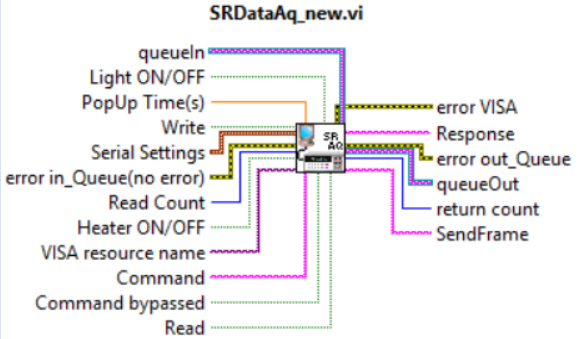
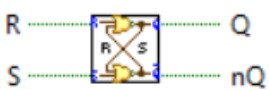
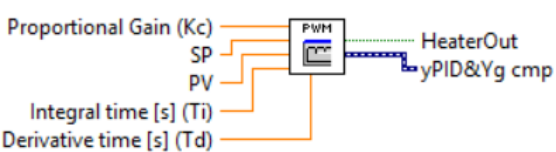

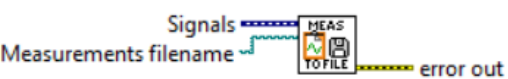
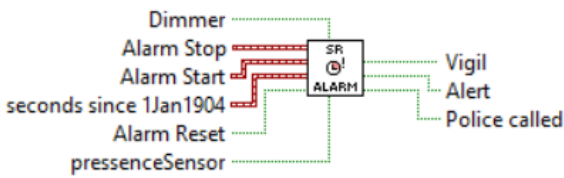
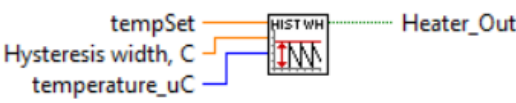

Pole ReadCount do określania liczby znaków w odbieranej ramce.

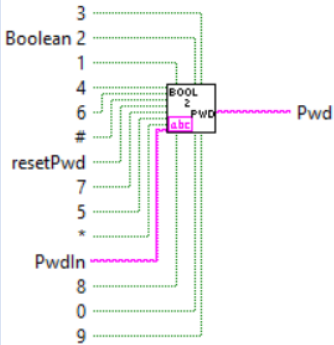
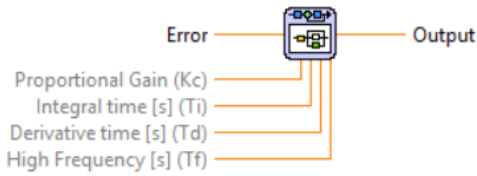
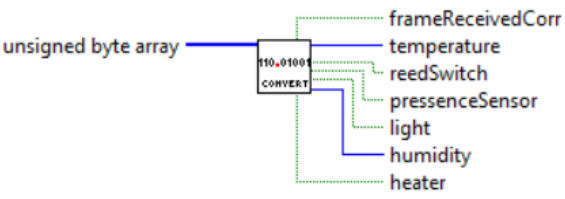
Kontrolka PWM Control jest odpowiedzialna za wybór rodzaju sterowania ogrzewaniem. Jeśli jest w trybie ON sterowanie z modulowaną szerokością impulsów jest aktywne, jeśli zaś jest w stanie OFF, wtedy to aktywne jest sterowanie dwupołożeniowe z regulowaną szerokością strefy histerezy.

Hysteresis width, umożliwia zadanie szerokości strefy histerezy (w przypadku gdy sterowanie PWM nie jest aktywne).

Wszystkie elementy panelu czołowego zostały szczegółowo opisane w części **Wykaz elementów panelu czołowego**.

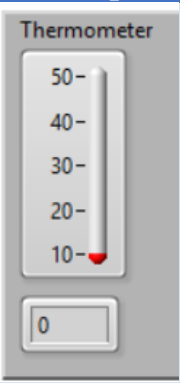
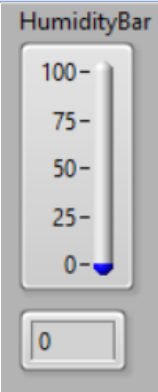
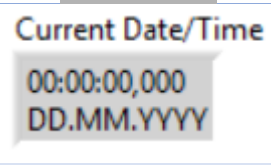
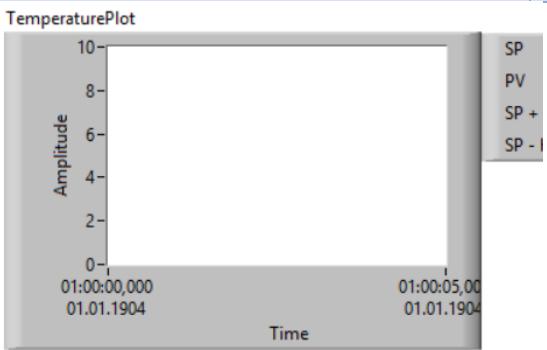
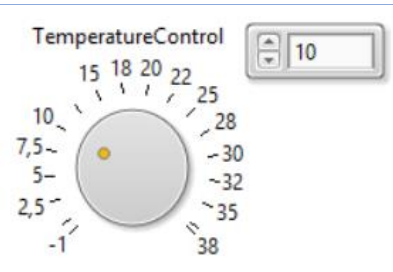
Opisy SubVI's


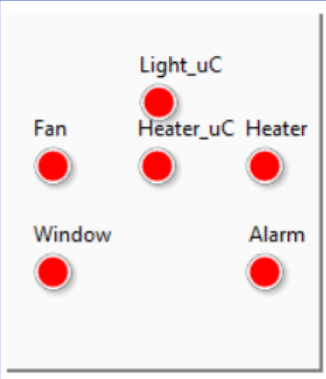

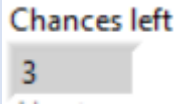

Ikona + wejścia/wyjścia	Nazwa	Opis funkcjonalny
 <p>SRDataAq_new.vi</p>	SRDataAq_new	Odczyt i zapis danych do mikrokontrolera
 <p>RS_flipflop.vi</p>	RS_flipflop	Przerzutnik RS
 <p>PWMControl.vi</p>	PWMControl	Regulacja temperatury za pomocą PWM
 <p>addRealTimeToWFC.vi</p>	addRealTimeToWFC	Oś X wykresu jest przedstawiona jako czas rzeczywisty
 <p>saveMeasToFile.vi</p>	saveMeasToFile	Realizuje zapis pomiarów do pliku o określonej nazwie
 <p>SRAlarm.vi</p>	SRAlarm	Wyzwała alarm w czasie pomiędzy „Alarm Start” i „Alarm Stop”
 <p>hysteresisControl.vi</p>	hysteresisControl	Regulacja dwupołożeniowa temperatury o ustalonej strefie histerezy
 <p>SRBoolToString.vi</p>	SRBoolToString	Zamienia typ bool na string w celu przesłania danych do mikrokontrolera

<p>appendToPwd.vi</p> 		<p>appendToPwd</p>	<p>Umożliwia dołączenie nowego znaku do pola Pwd jak również resetowanie pola Pwd w przypadku błędnego wpisania hasła.</p>
<p>SR_PID.vi</p> 		<p>SR_PID</p>	<p>Ciągły regulator PID o nastawach, które mogą być zmieniane w trakcie wykonywania programu.</p>
<p>uCoutToNum.vi</p> 		<p>uCoutToNum</p>	<p>Na podstawie ramki otrzymanej z mikrokontrolera dokonuje konwersji każdego z pól ramki na wartości logiczne i numeryczne</p>

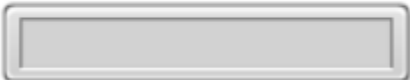
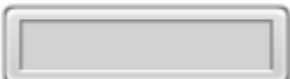


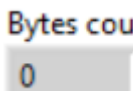
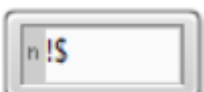
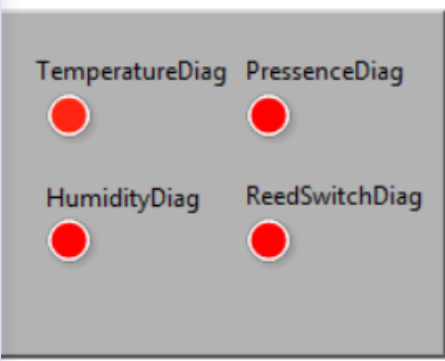
Wykaz elementów panelu czołowego

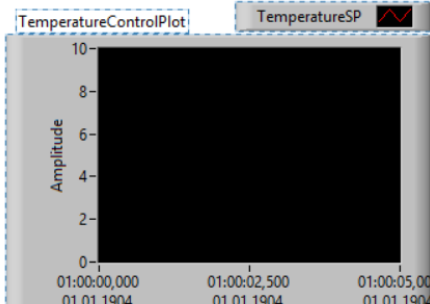
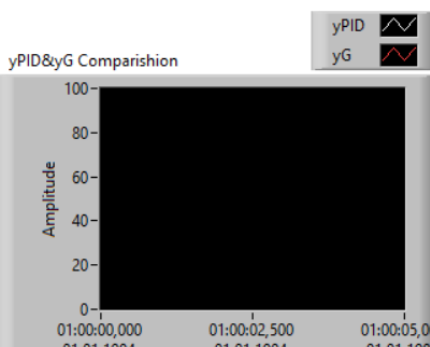
Controls and Indicators

Ikona na panelu	Nazwa	Opis
	Termometr	Pokazuje aktualną wartość temperatury odebraną z mikrokontrolera.
	Wilgotnościomierz	Pokazuje aktualną wartość wilgotności powietrza odebraną z mikrokontrolera.
	Data i czas	Pokazuje aktualną datę i czas.
	Wykres temperatury	Wykres pokazujący zmianę temperatury w czasie. Zaznaczony jest również wartość zadana (SP) oraz szerokość strefy histerezy.
	Gałka temperatury	Gałka pozwalająca kontrolować temperaturę zadaną.

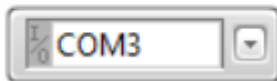
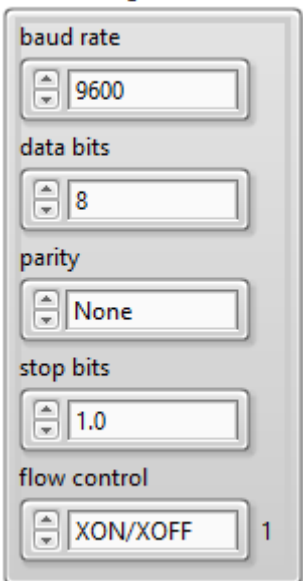
	Przełącznik światła	Umożliwia włączanie i wyłączanie światła.
	Kontrolki	<p>Fan – stan wiatraka(wł/wył)</p> <p>Window – stan kontaktronu(wł/wył)</p> <p>Heater – pokazuje czy grzałka powinna być włączona</p> <p>Heater_uC – stan grzałki(wł/wył)</p> <p>Light_uC – stan światła(wł/wył)</p>
	Hasło	<p>Pole do wpisywania hasła. Należy wprowadzić poprawne hasło aby móc korzystać z aplikacji.</p>
	Pozostałe próby	Informuje o ilości pozostałych prób, które użytkownik ma na wpisanie hasła.
	Przycisk STOP	Służy do zakończenia pracy aplikacji.

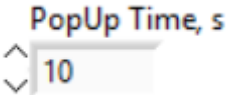


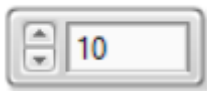

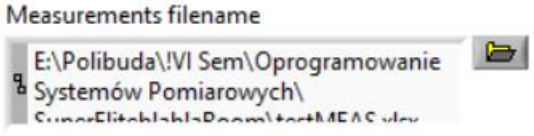
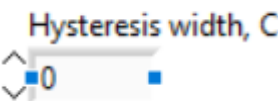

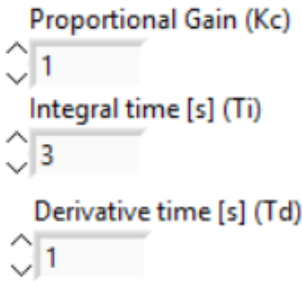
Diagnostic

Ikona	Nazwa	Opis
	Ramka odebrana w pętli konsumenta	Pokazuje dane odebrane z mikrokontrolera w pętli konsumenta
	Ramka odebrana w pętli producenta	Pokazuje dane odebrane z mikrokontrolera w pętli producenta
	Ramka wysyłana	Pokazuje dane wysyłane do mikrokontrolera. Gdy dane nie są wysyłane wyświetla się „~”
	Błędy	Wyświetla błędy i tłumaczy nieprawidłowe działanie SmartRoom
	Licznik bajtów	Pokazuje ile bajtów zostało wysłanych do mikrokontrolera
	Ramka do wysłania	Pozwala na nadpisanie ramki wysyłanej do mikrokontrolera
	Kontrolki	<p>TemperatureDiag – pokazuje czy dane z czujnika temperatury zostały odebrane</p> <p>PresenceDiag – pokazuje czy dane z czujnika ruchu zostały odebrane</p> <p>HumidityDiag – pokazuje czy dane z wilgotnościomierza zostały odebrane</p> <p>ReedSwitchDiag – pokazuje czy dane z kontaktronu zostały odebrane</p>

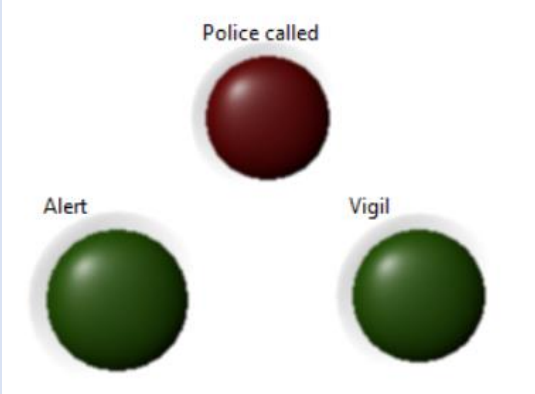



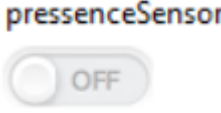
	Wykres wartości zadanej	Pokazuje zmiany wartości zadanej w czasie
	Wykres wyjścia regulatora oraz sygnału piłokształtnego	Pokazuje zmiany sygnału z regulatora temperatury oraz sygnału piłokształtnego. Istotne w kontekście doboru właściwych nastaw regulatora PWM.

Configuration

Ikona	Nazwa	Opis
	Źródło VISA	Pozwala na wybranie portu połączenia z mikrokontrolerem.
	Ustawienia portu szeregowego	Pozwala na zmianę ustawień portu szeregowego.

		Czas wyzwolenia wiadomości o błędzie	Pozwala na ustawienie interwału czasowego pomiędzy wiadomościami o błędzie połączenia.
		Czytanie danych	Pozwala na włączenie i wyłączenie czytania danych z mikrokontrolera.
		Wysyłanie danych	Pozwala na włączenie i wyłączenie wysyłania danych do mikrokontrolera.
		Długość ramki czytanej	Ustawia ilość bajtów w czytanej ramce.
		Przełącznik nadpisywania ramki	Gdy włączony, wysyłana ramka zostanie nadpisana ramką z zakładki „Diagnostics”.
		Ścieżka pliku z pomiarami	Umożliwia wybranie miejsca i pliku zapisu danych z pomiarów.
		Szerokość histerezy	Pozwala na zmianę szerokości histerezy do regulacji temperatury.
		Przełącznik regulatora PWM	Gdy włączony, regulacja temperatury będzie się odbywać za pomocą regulacji PWM. Gdy wyłączony regulacja temperatury będzie dwupołożeniowa.
		Parametry regulatora PID	Pozwalają na zmianę parametrów regulatora PID używanego przy regulacji PWM.

Alarms

Ikona	Nazwa	Opis
	Kontrolki	<p>Police called – pokazuje czy policja została wezwana</p> <p>Alert – pokazuje czy alarm został wyzwolony</p> <p>Vigil – pokazuje czy alarm jest w stanie czuwania</p>
	Początek stanu czuwania	Umożliwia wprowadzenie czasu początku stanu czuwania
	Koniec stanu czuwania	Umożliwia wprowadzenie czasu końca stanu czuwania
	Reset alarmu	Służy do wyłączenia alarmu
	Czujnik ruchu	Pokazuje obecny stan czujnika ruchu

Opis kodu

Aplikacja uC

Zadaniem aplikacji zaszytej w systemie mikroprocesorowym jest pobieranie danych z czujników, oraz ich wysyłanie poprzez interfejs komunikacyjny RS-232 do komputera. Aplikacja umożliwia również wystawianie odpowiednich wyjść mikrokontrolera co jest niezbędne do tego by sterować zarówno ogrzewaniem jak i oświetleniem. Aplikacja uC została szerzej opisana w części **Hardware** dokumentacji.

Aplikacja PC

Aplikacja stworzona na PC została opracowana z wykorzystaniem środowiska LabVIEW. Kod aplikacji jest bardzo złożony, ograniczymy się więc do kilku nadrzędnych mechanizmów, całość kodu wraz z komentarzami jest możliwa do obejrzenia poprzez otwarcie projektu w LabVIEW.

Nadrzędną strukturą wykorzystaną w aplikacji jest Flat Sequence, zapewnia ona sekwencyjne wykonywanie się zawartego w poszczególnych jej polach. Wykorzystanie tej struktury nie było niezbędne do właściwego wykonywania się kodu (wręcz w LabVIEW jest ona niezalecana), jednak zdecydowaliśmy się jej użyć aby nieco zwiększyć czytelność kodu. Równolegle pracuje również pętla EHL, która odpowiada za obsługę zdarzeń wygenerowanych przez użytkownika.

Flat Sequence składa się z trzech pól. W pierwszym następuje proces podawania hasła, w drugim polu wykonuje się główna część programu, a zatem szereg funkcjonalności opisanych wcześniej, w polu ostatnim struktury Flat Sequence następuje zamknięcie aplikacji.

W pierwszym polu struktury zastosowaliśmy pętlę while, aby wpisywanie hasła mogło odbywać się w sposób ciągły. Zależnie od tego czy wprowadzono poprawne hasło czy wciśnięto „Abort”, czy też niepoprawne hasło zostało podane więcej niż 2 razy, to w drugim polu struktury Flat wykona się kod odpowiadający głównym funkcjonalnościom, bądź też nastąpi przejście do trzeciego pola, a więc zamknięcie aplikacji.

Główna część aplikacji to dwie równolegle pracujące pętle while w strukturze Producent/Konsument. Pętla producenta odpowiada za zbieranie danych z mikrokontrolera oraz aplikacji i umieszcza je w kolejce. Pętla konsumentka odczytuje dane z kolejki i w odpowiedni sposób je przetwarza i prezentuje np. w postaci wykresów oraz kontrolek.

W trzecim polu struktury Flat następuje wyjście z aplikacji.

W celu głębszego zrozumienia kodu należy zapoznać się z kodem źródłowym programu dołączonego do dokumentacji.

Sposób komunikacji pomiędzy wątkami aplikacji

Jak zostało już wspomniane, wątki producenta i konsumenta komunikują się ze sobą z wykorzystaniem kolejek. Naturalnie, można wykorzystać notyfikacje, jednak uznaliśmy że istotnym dla nas jest by dane były buforowane i prezentowane nawet z opóźnieniem czego nie sposób uzyskać bez wykorzystania kolejek.

W kodzie źródłowym wyróżnić można trzy kolejki. Pierwsza odpowiedzialna jest za przechowywanie wartości temperatury zadanej, druga za przechowywanie danych odczytanych z mikrokontrolera, trzecia za przechowywanie wartości wilgotności. Trzecia kolejka wykorzystywana jest jedynie w celach testowych aplikacji, np. w przypadku konieczności weryfikacji poprawności przesyłania danych w innych kolejkach.

Każda z kolejek jest inicjalizowana w drugim polu struktury Flat Sequence, natomiast po zakończeniu wykonywania się pętli producenckiej są one niszczone.

Formaty danych, plików

Wszystkie pliki zawierające VI's mają rozszerzenie *.vi. Pliki utworzonych przez nas kontrolerek mają rozszerzenie *.ctl. Zaprojektowane przez nas menu charakteryzuje się rozszerzeniem *.rtm. Ikona aplikacji utworzona została w formacie *.ico. Całość spaja plik projektu z rozszerzeniem *.lvproj.

W celu zapisu pomiarów wykorzystaliśmy format *.xlsx. Zdajemy sobie sprawę, iż jest to format nieoptymalny w kontekście prędkości zapisu (lepiej byłoby wykorzystać pliki TDMS), jednak taki właśnie wyjściowy format plików z pomiarami założyliśmy na początku projektu.

Wykorzystane ikony

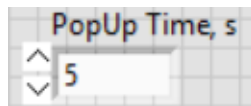
Prócz wykonanych przez nas ikon np. do opisów SubVI's wykorzystaliśmy kilka ikon z internetu, aby stworzyć własne unikalne kontrolki. Poniższe zestawienie obejmuje wygląd kontrolki jej wykorzystanie oraz wykorzystane grafiki niezbędne do jej utworzenia.

Num_Control

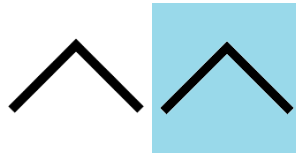


Kontrolka jest wykorzystywana na klawiaturze do wprowadzania hasła. Do tej kontrolki nie wykorzystywaliśmy żadnej dodatkowej grafiki.

NumArr_Control



Kontrolka jest wykorzystywana w niemal wszystkich miejscach tam, gdzie jest miejsce na kontrolki numeryczne. Wykorzystano obrazki ściągnięty z internetu:



ON_OFF_Control



Kontrolka pełni funkcje dwustanowego wskaźnika, bądź dwustanowego przycisku. Wykorzystano obrazki ściągnięte z internetu:



Stop_Control



Kontrolka pełni funkcje przycisku. Używana jest do tego, by wyłączyć aplikację SmartRoom. Do jej utworzenia wykorzystano obrazek ściągnięty z internetu o tej samym wyglądzie.

Kolory RGB

- Czerwony: jasny – R=255 G=0 B=0, ciemny - R=208 G=99 B=99
- Zielony: jasny – R=100 G=255 B=0, ciemny – R=107 G=235 B=107
- Niebieski: R=29 G=20 B=254

Spis rysunków

Rysunek 1 Schemat blokowy stanowiska badawczego	4
Rysunek 2 Schemat blokowy SMARTROOMU	5
Rysunek 3 Wyprowadzenia mikroprocesora ATmega 128.....	6
Rysunek 4 Środowisko Eclipse.....	7
Rysunek 5 ATB-USBasp Atmel8	7
Rysunek 6 Sterowanie oświetleniem	8
Rysunek 7 Sprzętowe usuwanie drgań styków (schemat)	8
Rysunek 8 Sprzętowe usuwanie drgań styków (zdjęcie)	9
Rysunek 9 Fragment kodu przedstawiający obsługę przerwania.....	10
Rysunek 10 Schemat układu sterowania nagrzewnicą.....	10
Rysunek 11 PIR - soczewka.....	11.
Rysunek 12 PIR spód	11.
Rysunek 13 PIR schemat połączenia	11
Rysunek 14 PIR maskowanie bitowe	12
Rysunek 15 PIR sterowanie LED	12
Rysunek 16 Kontaktron - zdjęcie	12
Rysunek 17 Kontaktron - schemat połączenia	13
Rysunek 18 Kontaktron - inicjalizacja jako wejścia w systemie.....	13
Rysunek 19 Kontaktron - zanegowane wyrażenie maski	13
Rysunek 20 DHT11 - zdjęcie	14
Rysunek 21 Zasada pracy czujnika DHT11.....	14
Rysunek 22 DHT11 - schemat połączenia.....	15.
Rysunek 23 Fragment kodu odbierania danych z czujnika DHT11	15
Rysunek 24 Wyświetlacz LCD - schemat połączenia	16
Rysunek 25 LCD - fragment realizujący wyświetlanie treści.....	16
Rysunek 26 Konwerter USB-RS232.....	17.
Rysunek 27 Dodawanie bajtu do bufora cyklicznego.....	17
Rysunek 28 Implementacji ramki danych.....	18
Rysunek 29 Odbiór danych przez ATMege.....	19
Rysunek 30 Controls and Indicators	22.
Rysunek 31 Diagnostics	24
Rysunek 32 Alarms	25
Rysunek 33 Configuration	26

Spis tabel

Tabela 1 Opisy SubVI's	28
Tabela 2 Controls and Indicators – panel czołowy	32
Tabela 3 Diagnostics – panel czołowy	33
Tabela 4 Configuration – panel czołowy	34
Tabela 5 Alarm	35