Stanisław WSZELAK

SYSTEM ZARZADZANIA I MONITOROWANIA MIKROKLIMATEM W NACZEPACH I KONTENERACH TRANSPORTOWYCH

Streszczenie

W publikacji zaprezentowano wykorzystanie mikrokomputerów SBC wielkości karty kredytowej w systemie zarządzania mikroklimatem w pomieszczeniach. Skupiono się na oprogramowaniu mikrokomputera do zapewnienia odpowiedniej temperatury i wilgotności w kontenerach transportowych, przewożących wrażliwe produkty, podatne na działanie nieodpowiednich warunków klimatycznych. Pokazano sposób zaprogramowania modułu centralnego (mikrokomputera) do współpracy z modułami roboczymi oraz współdziałania z bazą danych oraz serwerem Web, gromadzącym i prezentującym dane w sieci. Pominięto interfejs bezprzewodowy i komunikację z użytkownikiem, służącą do podglądu warunków klimatycznych, ponieważ problem ten został opisany w publikacji [2].

WSTĘP

Informatyzacja i automatyzacja to kolejna fala rewolucji przemysłowej. Rozwój komputerów i całego sektora IT prowadzi do całkowitej automatyzacji procesów produkcyjnych, sterowania i zarządzania obiektami, kontroli parametrów eksploatacyjnych, w celu zastąpienia człowieka i minimalizacji kosztów. Informatyzacja przynosi wzrost wydajności pracy, spadek kosztów produkcji i usług.

Zdalne monitorowanie i zarządzanie mikroklimatem (wilgotność i temperatura) w naczepach i kontenerach transportowych jest możliwe dzięki programowalnym systemom. System taki składa się z terminala (telefon komórkowy nowej generacji), programowalnej platformy pomiarowej i wykonawczej, zbierającej dane środowiskowe (temperatura, wilgotność), podejmując jednocześnie decyzje zgodne z algorytmem zadeklarowanym w programie systemu. Do monitorowania i raportowania danych pomiarowych niezbędna jest również aplikacja Web oraz baza danych przechowująca dane pomiarowe, konta użytkowników oraz dane dotyczące pracy systemu.

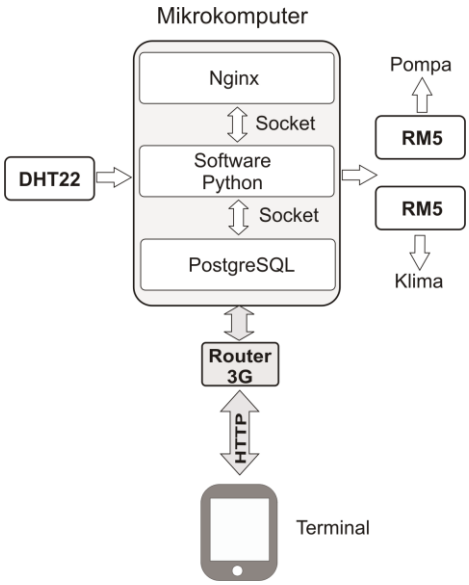
Ze względu na to, że w obiektach tego typu jak pojazdy samochodowe czy szynowe nie ma miejsca na typowe jednostki komputerowe, to nic nie stoi na przeszkodzie aby wykorzystać jednopłytkowe platformy komputerowe SBC (ang. *Single-Board Computer*). Pospolitym i ogólnie dostępną platformą SBC jest Raspberry Pi.

Dysponując mikrokomputerem Raspberry oraz wykorzystując infrastrukturę sieci bezprzewodowej (naziemnej i satelitarnej) możemy zaprojektować system wykonujący zadane procesy dla danego obiektu, kontrolę tych procesów, używając do kontroli typowych terminali mobilnych. Przy obecnie dobrze rozwiniętej architekturze telefonii komórkowej, jesteśmy w stanie kontrolować obiekty i urządzenia z dowolnego miejsca.

1. SYSTEM ZARZADZANIA MIKROKLIMATEM MIKOMPUTEREM SBC

System sterowania składa się z jednostki centralnej, modułów roboczych oraz modułów dodatkowych stanowiących rozszerzenia. Moduł to niezależne urządzenia elektroniczne, cechujące się wykonywaniem określonych zadań. Modułami roboczymi są urządzenia elektroniczne pełniące role pomiarowe (temperatury i wilgotności), interfejsy komunikacji bezprzewodowej oraz urządzenia wykonawcze (przekaźniki), które załączają nawiew powietrza, zraszacz wody lub inne podzespoły klimatyzacji.

Jednostkę centralną stanowi mikrokomputer Raspbbery, sprawuje on kontrolę i nadzór nad modułami roboczymi, steruje nimi, gromadzi i analizuje dane oraz komunikuje się z użytkownikiem, w tym przypadku za pomocą interfejsu bezprzewodowego (rysunek 1).



Rys. 1. *Elementy systemu zarządzania i monitorowania mikroklimatu w kontenerach transportowych*

1.1. Centralny moduł sterowania mikroklimatem

Centralnym modułem sterowania i zarządzania jest mikrokomputer Raspberry Pi, który jest wielkości karty kredytowej. To jednopłytkowa platforma komputerowa SBC, wykorzystująca układ typu SoC (*ang. System On A Chip*), z *Broadcom BCM2835/2836,* ze zintegrowaną kartą graficzną. W projekcie wykorzystano wersję z 4-rdzeniowym procesorem, taktowanym zegarem z częstotliwością 900 MHz oraz pamięcią operacyjną 1 GB RAM. Urządzenia Raspberry Pi występują w kilku wersjach, od A do V3. Wersje różnią się od siebie wyposażeniem płytki. Zasadnicza różnica istnieje pomiędzy modelami A i V, przede wszystkim w ilości dostępnej pamięci operacyjnej RAM, obsługą systemu operacyjnego (32 -bity, 64-bity), ilością portów USB, wyposażeniem w interfejs sieciowy. Najnowszą wersją jest V3, posiada wielordzeniowy procesor, wspiera 64-bitowe systemy operacyjne, w dodatku posiada na wyposażeniu moduły bezprzewodowe: Wi-Fi w standardzie 802.11n oraz Bluetooh 4.1 Low Energy.

System operacyjny centralnego modułu sterowania

Raspbbery Pi, w skrócie RPi działa pod Linuxem (Android, Debian), Unix (FreeBSD, NetBSD), Windows CE, OpenELEC, RaspBMC, Raspbian. W prezentowanym projekcie wykorzystano system operacyjny Raspbian. Jest to dystrybucja będącą rozwinięciem Debian Wheezy. Instalacja systemu może przebiegać według dwóch scenariuszy. Gotowym instalatorem o nazwie NOOBS oraz instalację obrazu img za pomocą bezpłatnego narzędzia Win32 Disk Imager. Wszystkie niezbędne pliki znajdują się pod adresem http://raspberrypi.org/downloads/. Raspbbery posiada czytnik kart microSD. Po włożeniu karty SD i sformatowaniu karty, obraz pobranego systemu operacyjnego należy wypakować z archiwum i zapisać na karcie pamięci SD, wykorzystując np. narzędzie Win32 Disk Imager.

Pierwsze uruchomienie Raspberry Pi wymaga podłączenia go do monitora i klawiatury, po to aby dokonać niezbędnych ustawień systemu. Po wystartowaniu systemu należy wydać komendę `sudo raspi-config`, która uruchomi graficzny konfigurator systemu Raspbian.

Serwer WWW i baz danych systemu

Serwer WWW i serwer baz danych niezbędny jest do gromadzenia i prezentacji danych pomiarowych. Serwery te zaimplementowano w centralnym module sterowania (w systemie Raspbian). Moduł centralny oprócz podstawowych funkcji takich jak zarządzanie procesami modułów roboczych, pełni również rolę komunikatora do komunikacji z użytkownikiem poprzez aplikację w środowisku Web. Najbardziej funkcjonalnym środowiskiem Web dla tego mikroprocesora jest Nginx. Cechuje go wysoka odstępczość, nawet przy silnie obciążonym serwisie. Nginx jest oprogramowaniem open source, zawiera serwer WWW oraz interpretator języka PHP. Instalacja składników ogranicza się do pobrania odpowiednich pakietów dla systemu Raspbian oraz ich instalacja (listing 1). Ostatnia linia uruchamia serwer nginx.

Listing 1. Instalacja nginx, PHP

```
sudo apt-get install nginx php5-fpm
sudo apt-get install mysql-server php5-mysql
sudo /etc/init.d/nginx start
```

Podstawową konfigurację serwera nginx zamieszczono w publikacji [1], polega ona na edytowaniu pliku /etc/nginx/sites-available/default.

Nadrzędnym celem aplikacji jest gromadzenie i przetwarzanie danych zebranych z czujników zewnętrznych, podłączonych do Raspberry przez szynę GPIO. Ta funkcjonalność wymusza określenia sposobu przechowywania danych.

Ze względu na udogodnienia w dostępie i modyfikacji oczywistym wyborem była relacyjna baza danych. Zdecydowano się na wykorzystanie do tego celu silnika bazy o nazwie PostgreSQL. Paczki instalacyjne są dostępne na wszystkie popularne systemy. Biblioteka implementująca silnik bazy w wersji na Linux zajmuje 33,8 MB. Baza ta posiada bardzo rozbudowane typy danych, najciekawsze z nich to JSON, XML oraz adresy sieciowe (INET, CIDR).

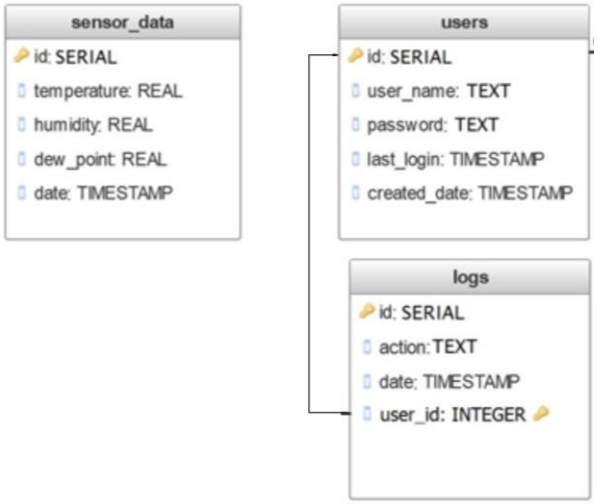
Instalacja bazy danych odbywa się przez wydanie komend menedżera pakietów (linia 1, listing 2).

Listing 2. Instalacja bazy danych PostgreSQL

```
(1) sudo apt-get install postgresql-9.4 fabric python-
    psycopg2
(2) sudo service postgresql start
(3) sudo su postgresql
```

Wśród instalowanych pakietów znajduje się „python-psycopg2”, który jest modułem dla języka Python, umożliwia on łączenie się z bazą z poziomu aplikacji lub skryptów tego języka programowania. Po instalacji następuje uruchomienie serwera bazy danych za pomocą komendy (linia 2). Przy konfiguracji bazy korzystano z poziomu konsoli `postgresql`, która zostaje wywołana komendą w linii 3.

Baza danych opiera się na trzech tabelach (rysunek 2). Główną tabelą jest `sensor_data`, to w niej zapisywane są wszystkie odczyty z czujników pomiarowych podłączonych do szyny GPIO.



Rys. 2. *Schemat bazy danych systemu*

Drugą tabelą jest tabela `users`, przechowuje ona logi użytkowników systemu. Ostatnia tabela to `logs`, w której zawarte są dane dotyczące pracy systemu i zmian w konfiguracji wprowadzonych przez panel administracyjny systemu, wraz z datą wystąpienia. Tabele tworzono typowymi narzędziami SQL.

Z uwagi na to, że tabela `sensor_data` jest kluczową w systemie, przedstawiono poniżej znaczenie pól tej bazy:

id (typu SERIAL), stanowi klucz główny do automatycznej inkrementacji, **temperature** (typu REAL), zmiennoprzecinkowa wartość temperatury w stopniach Celciusza, **humidity** (typu REAL), zmiennoprzecinkowa wartość wilgotności względnej wyrażona w procentach, **dew_point** (typu REAL), zmiennoprzecinkowa wartość temperatury punktu rosy, obliczonej na podstawie odczytanej temperatury powietrza i wilgotności względnej, **date** (typu TIMESTAMP), dokładna data odczytu parametrów z czujników.

Komunikacja jednostki centralnej z modułami

Do komunikacji jednostki z użytkownikiem służy interfejs sieci bezprzewodowej, w zależności od użytego modelu Raspbbery (802.11n w modelu V3 lub USB Hawaii dla pozostałych modeli). Organizację i konfigurację usługi bezprzewodowej na tym interfejsie, służącej do komunikacji systemu z użytkownikiem przedstawiono w publikacji [2].

Do komunikacji Raspberry Pi z modułami roboczymi (czujniki, sensory) służą piny GPIO oraz piny zasilające DC 3,3V i VDD 5V. Piny umożliwiają podłączanie wielu elementów elektronicznych, w większości sensorów, sygnalizatorów, przekaźników, siłowników.

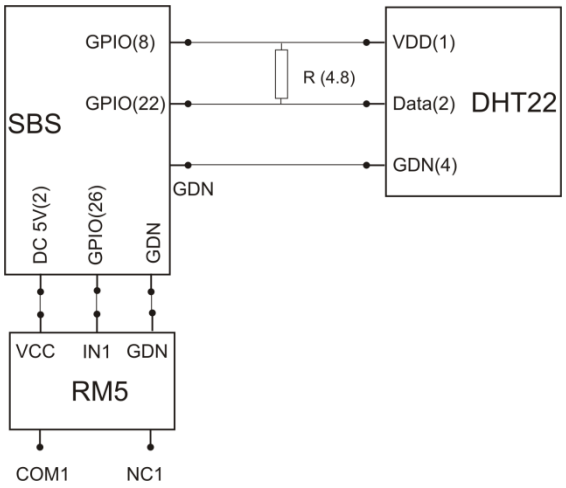
Programowanie pinów GPIO jest możliwe dzięki różnym bibliotekom napisanym w kilku językach, począwszy od C i C++ przez Pythona i Javę. Jednak najbardziej rozpowszechnionym językiem dla oprogramowania Raspberry jest Python i dla Raspbbery istnieje najwięcej gotowych bibliotek, obsługujących popularne elementy elektroniczne. Istnieją jednak pewne ograniczenie w wykorzystaniu pinów Raspbbery, pierwszym z nich jest brak zabezpieczenia prze-

ciw zwarciu pinów DC z masą. Nieumiejętne połączenie elementu z pinami może więc spowodować, że procesor ulegnie uszkodzeniu. Kolejnym ograniczeniem jest brak analogowego wejścia, tak więc wszystkie czujniki muszą być cyfrowe, co obecnie nie stanowi to żadnego problemu.

1.2. Moduły robocze systemu

Moduły robocze systemu to cały szereg elementów pasywnych i aktywnych podłączonych, do programowalnych pinów GPIO (rysunek 2). To przetworniki sygnałów nieelektrycznych na sygnały elektryczne i odwrotnie, fotorezystory, termopary, fotodetektory oraz przekaźniki, które załączają elementy wykonawcze (pompy, sprężarki, wentylatory itp.). W publikacji pod uwagę wzięte zostaną dwa przetworniki, wilgotności (0%RH do 100%RH) i temperatury (-40^{0}C do 80^{0}C). W modelu wykorzystano dostępny przetwornik DHT22, przetwarzający temperaturę i wilgotność (wbudowany higrometr i termometr). Przetwornik posiada cztery wyprowadzenia, końcówkę nr 1 podłączamy do zasilania (od 3,3V do 6V), końcówkę nr 4 do masy układu, natomiast końcówka nr 2 jest sygnałową (Data) i łączymy ją z pinem GPIO mikrokomputera.

Raspbbery jest bardzo czuły na poziom sygnału wejściowego, sygnał pochodzący z DHT22 może być na tyle wysoki, że może uszkodzić mikroprocesor lub zniekształcać pomiar, w związku z tym szynę Data należy z mostkować rezystorem o wartości 4,8Ω, drugi koniec łączymy z końcówką nr1 przetwornika.



Rys. 2. *Podłączenie modułów z mikrokomputerem*

Zasilanie urządzeń wykonawczych w modelu zrealizowano za pomocą przekaźników RM5 z optyczną izolacją, przekaźniki sterowane są napięciem 5V o dopuszczalnym natężeniu prądu 10A (na wyjściu przekaźnika).

1.3. Obsługa modułu DHT22

Do obsługo modułu DHT22 istnieje gotowa biblioteka open source napisana w języku Python, dostępna pod adresem: https://github.com/adafruit/Adafruit_Python_DHT. Instalację biblioteki przedstawiono w listingu 3.

Listing 3. Instalacja biblioteki do obsługi DHT22
```
cd /usr/src
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build

cd..
wget abyz.co.uk/rpi/pigpio/pigpio.zip
unzip pigpio.zip
cd PIGPIO
make
sudo make install
sudo pigpio
```

Pobraną bibliotekę modułu DHT22 umieszczono w folderze **libs** projektu. Import biblioteki przebiega następująco: `import pigpio` (nowa linia) `from libs.DHT22 import Sensor`. Jak łatwo zauważyć posiadamy klasę o nazwie `Sensor`, do której możemy odwoływać się w projektowaniu przydziału pinów do roli w systemie. Wywołanie konstruktora klasy `Sensor` zamieszczono w listingu 4.

Listing 4. Klasa Sensor z listą elementów
```
pi = pigpio.pi()
tempSensor = Sensor(pi, 22, LED=16, power=8)
```

`pigpio` jest konstruktorem klasy. Do pinu nr 16 podłączono diodę LED, aby ona sygnalizowała dokonanie odczytu z modułu DHT22. Odczyt pomiarów z przetwornika następuje poprzez wywołanie metody `trigger` (linia 1, listing 5).

Listing 5. Odczyt pomiarów metodą trigger
```
tempSensor.trigger()
time.sleep(0.2)
temperature = tempSensor.temperature()
humidity = tempSensor.humidity()
tempSensor.sensor_resets()
```

Rozpoczyna się cykl odczytu, który trwa pewien przedział czasowy a to wymaga wstrzymania pobierania danych o wartość np. 200 milisekund (`time.sleep(0.2)`). Po tym czasie następuje odczytywanie temperatury i wilgotności z obiektu czujnika (linia 3, linia 4). Na samym końcu należy wyzerować stan czujnika na kolejny odczyt (linia 5).
Dysponując wynikami z odczytów temperatury i wilgotności względnej można obliczyć wartość punktu rosy, czyli temperatury poniżej której zawarta w powietrzu para wodna ulega skropleniu. Wyliczenie takie przedstawiono w listingu 6, zaokrąglano do jednego miejsca po przecinku.

Listing 6. Programistyczne wyznaczenie punktu rosy
```
dew_point = math.pow(humidity/100, 1/8)*(112+0.9* tem-
perature)+0.1*temperature-112
dew_point = round(dew_point, 1)
```

1.4. Obsługa RM5

Do obsługi RM5 użyto biblioteki RPi.GPIO, którą należy zaimportować (listing 7, linia 1), następnie zadeklarować numer pinu (26) i skonfigurować ten pin GPIO jako wyjściowy (przedostatnia linia).

Listing 7. Obsługa RM5
```
import RPi.GPIO as GPIO
pin = 26
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
GPIO.setwarnings(False)
GPIO.setup(pin, GPIO.OUT)
GPIO.output(pin, True)
```

Sterowanie przekaźnikiem sprowadza się do ustawiania stanów na wyjściu pinu (linia ostatnia), wysokiego (True) albo niskiego (False). Stan wysoki włącza obwód a niski rozłącza go.

1.5. Zapisywanie danych pomiarowych do bazy

Komunikację skryptu Pythona z bazą danych PostgreSQL umożliwia biblioteka psycopg2, której instalację pokazano w punkcie 1.1.2. Import biblioteki do skryptu pokazano w listingu 8 (linia 1).

Listing 8.
```
(1) import psycopg2
(2) conn = psycopg2.connect(database="measurements",
    user="app", password="****", host="127.0.0.1",
    port="5432")
(3) cursor = conn.cursor()
(4) parameters = [temperature,humidity,dew_point]
(5) try:
    cursor.execute("INSERT INTO sensor_data (tempera-
    ture, humidity, dew_point, date) VALUES (%s, %s,
    %s, NOW())", parameters)
    conn.commit()
    except Exception, e:
    print "An error has occurred: %s" %e
    finally:
     conn.close()
```

Do połączenia z bazą wymagane jest wywołanie metody `connect` z następującymi parametrami: nazwa bazy (`measurements`), użytkownik (`app`), hasło, adres ip serwera (`localhost`), numer portu, pod którym dostępny jest serwer baz danych (5432).

Po nawiązaniu połączenia pobierany jest obiekt kursora (linia 3), za pomocą którego wykonuje się transakcje. Z powodu możliwości wystąpienia błędu w operacji nadpisywania w bazie należy zabezpieczyć, obudować blokiem `try-except`, w celu przechwycenia wyjątku, który spowodowałby zawieszenie się aplikacji.

W liniach od nr 5 następuje wstawianie nowego rekordu z danymi, pochodzące z odczytu do tabeli `sensor_data`. Zapisywane parametry to temperatura, wilgotność względna i temperatura punktu rosy. Dodatkowo umieszczany jest także czas wstawienia rekordu do bazy za pomocą komendy NOW(). Na koniec następuje zamknięcie połączenia z bazą.

1.6. Komunikacja pomiędzy procesami

System wymaga dwóch aplikacji, skryptu Pythona obsługującego urządzenia podłączone do szyny GPIO oraz skryptu obsługi aplikacji Web i bazy danych, niezbędnej do gromadzenia i wyświetlania danych pobieranych przez przetworniki (sensory). Aplikacje te muszą posiadać mechanizm integracyjny, muszą mieć możliwość porozumiewania się ze sobą, czyli posiadać zdolność do komunikacji międzyprocesowej. Możliwe jest to poprzez użycie socketów, które stanowią integralną część systemu UNIX, w działaniu trochę one przypominają socjety wykorzystywane w komunikacji sieciowej, jednak w tym przypadku komunikacja odbywać się będzie z udziałem kernela systemu operacyjnego (Listing 9, linia 1).

Listing 9. Skrypt integrujący aplikacje ze sobą
```
(1) server_socket = socket.socket(socket.AF_UNIX, sock-
    et.SOCK_STREAM)
(2) server_socket.bind("/tmp/www/projectsocket")
(3) os.system("chmod g+rwxs /tmp/www/projectsocket")
(4) server_socket.setblocking(1)
(5) server_socket.listen(5)
(6) server_socket.settimeout(5)
```

Socket wymaga pliku, do którego będą zapisywane dane strumieniem (linia nr 2). Skrypt Pythona traktujemy jak serwer a aplikację WWW jako klient serwera. Aplikacja musi mieć możliwość komunikowania się ze skryptem Pythona, należy nadać jej odpowiednie uprawnienia (linia nr 3). Łącze zostało zdefiniowane jako blokujące (linia nr 4), maksymalna liczba połączeń oczekujących została zdefiniowana w linii nr 5 (pięć połączeń), a maksymalny czas oczekiwania na pięć sekund (linia nr 6).

Serwer socketów musi być uruchamiany w osobnym wątku po to, aby jego praca nie wpływała na przebieg wątku głównego programu, toteż działanie opiera się na pętli, która wykonuje się przez cały czas trwania procesu. Odczyt danych ze strumienia odbywa się w paczkach po 1024 bajtów. Jeśli rozmiar danych jest większy od 0, to następuje odczytanie komunikatu i wykonanie stosownej akcji. Na każdy komunikat dawana jest odpowiedź OK, a treść odebranego żądania trafia do tabeli z logami w bazie danych. Takie rozwiązanie umożliwia kontrolę poprawności działania systemu i odtworzenie historii działań. Na potrzeby komunikacji stworzone zostały następujące klucze komunikatów, umieszczane na samym początku strumienia, na które reaguje serwer:

- a – test połączenia,
- w – ustawienie zmiennej `write_cycle`, druga część strumienia, odczyt danych z czujników i zapis do bazy, oczekiwaną liczbą jest liczba typu integer (Listing 10, linia 1),
- r – ustawienie zmiennej `relay_state`, druga część strumienia, oczekiwany jest tekst, określający stan obwodu sterowanego przez przekaźnik,
- t – ustawienie zmiennej `threshold`, druga część strumienia, oczekiwana liczba zmiennoprzecinkowa określająca próg temperatury, przy którym następuje zmiana stanu obwodu sterowanego przez przekaźnik,
- q – komunikat zakończenia skryptu Pythona obsługującego przetworniki.

Listing 10. Skrypt klucza komunikatu dla strumienia odczytującego i zapisującego dane do bazy
```
elif data[0] == 'w':
  config.write_cycle = int(data[1:])
  update_config()
  connection.send('ok')
  insert_log("socket - write_cycle request");
```

Wymiana danych pomiędzy aplikacją WWW a skryptem obsługującym czujniki odbywa się w dwóch etapach. W pierwszym etapie dane z formularza przekazywane są za pomocą javascript (skryptu) do pliku (aplikacji) `socket_communication.php` (listing 11).

Listing 11. Przekazanie danych z do skryptu PHP
```
var write_cycle = docu-
ment.getElementById('element_1').value;
     document.write('<div>Sending Write Cycle</div>');
        $.post('socket_communication.php', {
                msg_type : "write_cycle",
                msg : String(write_cycle)
                }, function () {});
```

Po tym następuje połączenie się z systemowym socketem i wysyłka parametrów interwału odczytu danych z czujnika (write_cycle). Drugi etap wymiany danych przedstawiono w listingu 12.

Listing 12. Drugi etap wymiany danych
```
case "write_cycle":
        if($write_cycle !== $data && $running === true){
                $data_string = "w".$data;
                socket_write($socket, $data_string, 1024);
                $answer = socket_read($socket, 1024);
                if($answer = "ok"){
                        echo 1;
                }else{
                        echo 0;
                }
        }
        break;
```

2. GRAFICZNA PREZENTACJA DANYCH POMIAROWYCH

Graficzna prezentacja danych najczęściej realizowana jest za pomocą wykresów. Kreślenie wykresów w architekturze klient-serwer można zrealizować na kilka sposobów. Jednym ze sposobów jest rysowanie wykresów po stronie serwera i zwracanie ich w formacie obrazu (biblioteka pChart). Innym sposobem to rysowanie wykresu po stronie klienta, czyli przez przeglądarkę klienta. W takim sposobie serwer zwraca tylko dane te, które są niezbędne do wykreślenia wykresu (biblioteka Google Charts). W publikacji zdecydowano się na drugi sposób, czyli za pomocą biblioteki Google Charts.

Aby biblioteka Google Charts mogła przetworzyć dane i narysować na ich podstawie wykres, dane muszą być dostarczone w odpowiednim formacie. Wymaganym formatem jest JSON, który zawiera kilka istotnych pól:

- **cols,** tablica obiektów definiujących kolumny (osie wykresu),
  - *id*; id kolumny,
  - *label*; nazwa kolumny,
  - *type*; typ danych, które zawiera kolumna,
- **rows,** tablica obiektów wierszy (danych),
  - *c*; tablica obiektów komórki, każda komórka zawiera pole v, które przechowuje wartość, komórki odpowiadają zdefiniowanym kolejno kolumnom.

Pobieranie danych z bazy (temperatura, wilgotność, punkt rosy) zrealizowano poprzez skrypt PHP (listing 13).

Listing 13.Funkcja pobierająca dane z bazy, formatująca do JSON i zwracająca je klientowi

```
(1)
$sql = "SELECT date, temperature FROM sensor_data WHERE
date>=".$start_date." AND date<=".$end_date;
$result = pg_query($sql) or die ('Failed to query data:
'.pg_last_error());

(2)
$data_table = "{\"cols\": [{\"id\": \"data\", \"label\":
\"Data\",   \"type\":   \"date\"},   {\"id\":   \"temp\",
\"label\":   \"Temperature\",   \"type\":   \"number\"}],
\"rows\": [";

while ($l = pg_fetch_row($result)) {
 $data_table       =       $data_table."{\"c\":[{\"v\":
\"Date(".$l[0].")\"}, {\"v\": ".$l[1]."}]},";
}
$data_table = substr($data_table, 0, -1);
$data_table = $data_table."]}";

(3)
echo $data_table;
```

Linie o w segmencie nr 1 to nic innego jak zapytanie do bazy o eksport danych, linie nr 2 formatują dane tak jak oczekuje biblioteka JSON, natomiast linia nr 3 przekazuje sformatowane dane klientowi.

2.1. Generowanie wykresu liniowego

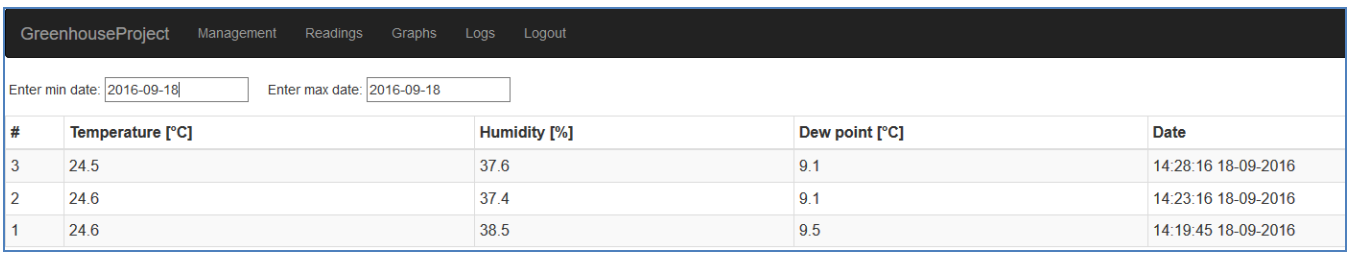
Wygenerowanie wykresu zrealizowane zostanie po stronie klienta w skrypcie JavaScript (listing 14).

Listing 14.

```
(1)     <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript"
    src="http://code.jquery.com/jquery-
3.1.0.js></script>
<script type="text/javascript">
(2)     google.charts.load('current', {'packag-
es':['corechart']});
google.charts.setOnLoadCallback(sendChartDataRequest);
(3)     function sendChartDataRequest() {
  $.post('readings_chart_data.php', {
      chart_type : 1,
start_date : 111,
  end_date       : 222
  }, function (data) {
var json_object = jQuery.parseJSON(data);
  drawChart(json_object);
  });
}
(4)     var data = new
google.visualization.DataTable(data_str)
(5)     var visualization = new google.visualization.
LineChart(document.getElementById('chart_div'));
visualization.draw(data, options);
```

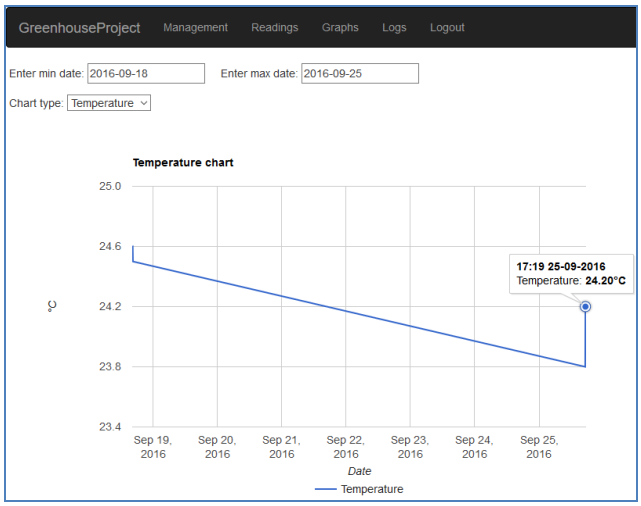
Skrypt najpierw łączy dwie biblioteki, Google Charts i jQuery (linie nr 1, 2), pierwsza rysuje a druga dostarcza komponent AJAX (służący między innymi do wykonywania asynchronicznych zapytań). Gdy komponenty zostaną załadowane następuje wysłanie żądania POST do serwera o dane wsadowe do wykresu wraz z parametrami określającymi oczekiwany typ i przedział czasowy (linie nr 3), gdy zostaną uzyskane przetworzenie zostają na użytek wykresu (linie nr 4). Wyświetlenie realizują linie o nr 5, w sekcji strony będącej kontenerem dla widoku.

Tablicę odczytów z przetwornika DHT22 przedstawiono na rysunku 3.



Rys. 3. *Odczyty z przetwornika DHT2*

Wykres odczytów zamieszczono na rysunku 4.



Rys. 4. *Wykres danych pochodzących z odczytu*

W zakładce czwartej (Graphs) istnieje możliwość zdefiniowania przedziału czasowego pobranych próbek oraz typ wykresu. Wybrano tylko graf temperatury, ale możliwy jest również wybór wilgotności względnej oraz temperatury punktu rosy.

PODSUMOWANIE

Systemy projektowane na mikrokomputerach Raspberry Pi nie należą do skomplikowanych. Problem sprowadza się do odpowiedniego zaprojektowania mikrokomputera pod sprawowane funkcje. Przedstawione rozwiązanie sterowania mikroklimatem podkreśla nieograniczony i bogaty potencjał innowacyjności w konstruowaniu systemów i układów do automatycznego sterowania i zarządzania obiektami czy procesami. Podstawowymi elementami systemu są moduły robocze, których zadaniem jest wykonywanie zaprogramowanych czynności i przesłanie danych do jednostki centralnej oraz mikrokomputer stanowiący jednostkę centralną. Kontrolę nad pracą modułów roboczych sprawuje moduł centralny będący jednocześnie serwerem bazodanowym oraz serwerem WWW podłączonym do sieci Internet.

Niniejsza publikacja może posłużyć studentom oraz inżynierom w projektowaniu automatyki przemysłowej, w robotyce, wszelkich innych rozwiązaniach związanych z cyfrowym przetwarzaniem sygnałów oraz z automatycznym zdalnym sterowaniem procesami.

BIBLIOGRAFIA

1. Wszelak S., *Inteligentne zarządzanie obiektami z wykorzystaniem mikrokontrolerów SBM i SBC (Arduino i Raspberry Pi).* Logistyka 2014, nr 6.
2. Wszelak S., *Bezprzewodowe usługi sieciowe z wykorzystaniem mikrokomputera Raspberry Pi.* Techniki Transportu Szynowego 2015, nr 12.
3. http://www.rpiblog.com [lipiec 2016].
4. https://www.thomas-krenn.com [lipiec 2016].
5. https://wireless.wiki.kernel.org [wrzesień 2016].
6. http://www.instructables.com [wrzesień 2015].

PROGRAMMABLE SYSTEM MANAGEMENT AND MONITORING MICROCLIMATE IN TRANSPORT CONTAINERS

Abstract

The publication presents the use of microcomputers SBC size of a credit card in managing microclimate indoors. Focuses on microcomputer software to ensure proper temperature and humidity in transport containers carrying sensitive products susceptible to unsuitable weather conditions. It shows how to program the central module (microcomputer) to work with the modules working and interacting with the database and Web server, presenting data on the network. Skipped wireless interface and communication with the user, used to preview the climatic conditions, because the problem is described in [2].

Autorzy:
dr inż. **Stanisław Wszelak** – Uniwersytet Kardynała Stefana Wyszyńskiego w Warszawie, Wydział Matematyczno-Przyrodniczy, Szkoła Nauk Ścisłych, Instytut informatyki; 01-938 Warszawa; ul. Wóycickiego 1/3. Tel. 600663426, e-mail:stanislaw@wszelak.com, www.wszelak.com