Monitorowanie Raspberry Pi poprzez komunikator internetowy

Raspberry Pi to świetna platforma do stworzenia np. systemu automatyki domowej. Duża liczba gotowych modułów oraz liczne przykładowe aplikacje pozwalają niemal błyskawicznie zestawić dosyć złożony system nadzorujący mieszkanie lub dom. Problem pojawia się dopiero na etapie tworzenia interfejsu użytkownika. Umieszczenie w nim wszystkich funkcji oraz wbudowanie mechanizmu powiadomień jest po prostu mocno czasochłonne. Zamiast tego można swój bezinterfejsowy system wzbogacić o obsługę komunikatora internetowego i sterować domem w sposób tekstowy.

Na wstępie należy przyznać – sterowanie urządzeniami za pomocą komend tekstowych to obecnie archaizm, który w komercyjnych rozwiązaniach konsumenckich raczej nigdy nie będzie miał już zastosowania. Natomiast używanie komunikatora internetowego w aplikacji przemysłowej to pomyłka (z zastrzeżeniem, o którym piszemy później). Ale dla majsterkowicza, który zna się na elektronice i programowaniu, ale nie ma serca ani talentu (ani czasu) do opracowywania GUI, wykorzystanie komunikatora może być zbawieniem.

Który komunikator wybrać?

Na wstępie warto zdecydować się na któryś z dostępnych komunikatorów internetowych. Wybór nie jest mały, ale jeśli skoncentrować się na jego uniwersalności, niezawodności lub otwartości oraz dostępności narzędzi dla Raspberry Pi, można go szybko ograniczyć.

Ostateczna decyzja zależy od tego, kto i za pomocą jakiego urządzenia będzie łączył się z Raspberry Pi. Dobrym wyborem wydaje się użycie bardzo popularnej usługi WhatsApp, które nie tylko dostępna jest prawie na całym

Tabela 1. Biblioteki XMPP dostępne w różnych językach programowania

Nazwa	Język programowania
agsXMPP SDK	C# / .net / Mono
Aioxmpp	Python
AnyEvent::XMPP	Perl
as3xmpp	Flash / ActionScript
AXMPP	Ada
Babbler	Java
Blather	Ruby
cl-xmpp	Lisp
Coversant SoapBox SDK Studio	C# / .net / Mono / C++
dojox.xmpp	JavaScript
Dxmpp	C++
Echomine Feridian	Java
Eiffel	PHP
Emite	Java
Escalus	Erlang
Exmpp	Erlang
Frabjous	JavaScript
Gloox	C++
Headstock	Python
Hsxmpp	Haskell
Hxmpp	haXe
Iksemel	C
IP*Works Internet Toolkit	ActiveX, C++, C#,
Iris	C++

Tabela 1. cd.

Nazwa	Język programowania
jabber.net	C# / .net / Mono
jabber.py	Python
JabberLib	Tcl
Jabber Stream Objects (JSO)	Java
JAXL	PHP
jQuery-XMPP-plugin	JavaScript
Jreen	C++/Qt
JSJaC	JavaScript
Libstrophe	C
Libpurple	C/C++
Lightr	PHP
Loudmouth	C
MatriX	C# / .net / Mono
net::XMPP	Perl
node-xmpp	JavaScript
Oajabber	C++
Pontarius XMPP	Haskell
Pyxmpp	Python
pyxmpp2	Python
QXmpp	C++
seesmic-as3-xmpp	Flash / ActionScript
Sharp.Xmpp	C# / .net / Mono
Skates	Ruby
SleekXMPP	Python
Slixmpp	Python

Tabela 1. cd.

Nazwa	Język programowania
Smack	Java (Java SE 7 and Android)
stanza.io	JavaScript
strophe.js	JavaScript
StropheCappuccino	Objective-J
Swiften	C++
Tinder	Java
Txmpp	C++
Twisted Words	Python
Ubeity	C#
Verse	Lua
XIFF	Flash / ActionScript
xmpp-psn	Python
jaxmpp2	Java / Android / Google Web Toolkit
xmpp4js	JavaScript
XMPP4R	Ruby
xmpp4r-simple	Ruby
Xmppframework	Objective C
Xmpphp	PHP
Xmpppy	Python
XMPP-FTW	JavaScript
Z-XMPP	JavaScript

Tabela 2. Serwery XMPP dla różnych systemów operacyjnych

Nazwa	Platforma (System operacyjny)
Apache Vysper	Windows / Linux
Citadel	Linux
CommuniGate Pro	Linux / Mac OS X / Windows
Coversant SoapBox Server	Windows
Djabberd	Linux
Ejabberd	Linux / Mac OS X / Solaris / Windows
IceWarp	Linux / Windows
iChat Server	Mac OS X
in.jabberd	Linux
Isode M-Link	Linux / Solaris / Windows
Jabber XCP	Linux / Solaris / Windows
jabberd 1.x	Linux
jabberd 2.x	Linux / *BSD / Solaris / Windows
Jerry Messenger	Linux / Windows
Kwickserver	Windows
Metronome IM	Linux / Mac OS X
MongooseIM	Linux / Mac OS X
Openfire	Linux / Mac OS X / Solaris / Windows
Oracle Communications IM Server	Linux / Solaris / Windows
Prosody IM	Linux / Mac OS X / Windows
Psyced	Linux / Mac OS X / Windows
Siemens OpenScape	Linux
Tigase	Linux / Solaris / Mac OS X / Windows
Vines	Linux / Mac OS X
Wokkel	Linux / Solaris / Mac OS X

świecie, ale także oprogramowanie klienckie zostało przygotowane na różne systemy operacyjne, a nawet można z niego korzystać przez przeglądarkę internetową. Zaletą jest też szyfrowanie komunikacji w trybie „end-to-end", co oznacza – jeśli wierzyć usługodawcy, że przesyłane wiadomości są czytelne tylko dla nadawcy i adresata, a potencjalni włamywacze musieliby zastosować silne maszyny deszyfrujące lub niezłą socjotechnikę. Pewnym problemem jest natomiast to, że WhatsApp wymaga aktywnego numeru telefonu do rejestracji urządzenia klienckiego oraz że jeden numer może być wykorzystywany przez maksymalnie jedno urządzenie w danej chwili. To uproszczenie może sprawiać pewne trudności, ale w obecnych czasach całkiem łatwo jest choćby na chwilę zdobyć numer telefonu, za pomocą którego można zarejestrować się w WhatsApp.

Alternatywnym sposobem będzie użycie własnego (lub gotowego, bezpłatnego) serwera XMPP, a więc standardu bazującego na dawniej popularnym Jabberze. Aktualnie XMPP też wspiera szyfrowanie, a możliwość postawienia własnego serwera sprawia, że da się w pełni panować nad stanem komunikacji. Co więcej, XMPP jest otwartym standardem, opartym na XML-u i sformalizowanym przez organizację Internet Engineering Task Force. Lata doświadczeń dużych firm sprawiły, że XMPP uważa się za sprawdzony protokół, który można wykorzystać w przemyśle i świetnie nadaje się on do zastosowań w aplikacjach Internetu Rzeczy. Biblioteki XMPP są dostępne m.in. w Pythonie, JavaScripcie, C++, PHP i w Javie, przy czym w wielu przypadkach istnieje więcej niż jedno rozwiązanie dla danego języka (**tabela 1**). Wybór serwerów też jest bardzo duży (**tabela 2**), nie mówiąc już o liście gotowych aplikacji klienckich (**tabela 3**).

WhatsApp przez Yowsup

W niniejszym przykładzie pokażemy, jak najszybciej podłączyć Raspberry Pi do popularnego komunikatora, a więc do WhatsAppa. W dalszej części artykułu pokażemy, jak skorzystać z czystego XMPP.

By obsłużyć WhatsApp, użyjemy otwartej biblioteki Yowsup w wersji 2.5.0, której szczegóły można poznać na GitHubie, pod adresem ***https://github.com/tgalal/yowsup***. Biblioteka Yowsup została napisana w Pythonie – i jak chwalą się jej autorzy – posłużyła m.in. do utworzenia nieoficjalnej aplikacji klienckiej Wazapp na telefon Nokia N9 a także na system operacyjny Blackberry 10. Yowsup teoretycznie wymaga Pythona 2.6 lub 3.0 albo ich nowszych wersji, przy czym w trakcie naszych testów okazało się, że działa dopiero z Pythonem 3. Ponadto, konieczne jest zainstalowanie dodatkowych pakietów. Będzie to przede wszystkim python3-dateutil, a w przypadku chęci zapewnienia szyfrowania: protobuf, pycrypto i python3-axolotl-curve25519. Jeśli biblioteka ma być obsługiwana z linii komend, należy też doinstalować argparse, readline i ew. pillow, który pozwala przesyłać obrazy.

Teoretycznie, po pobraniu obrazu pełnego Raspbiana instalacja Yowsupa powinna być prosta. Powinno wystarczyć polecenie **pip install yowsup2**, ale już w wypadku obrazu Raspbiana w wersji lite, programu python-pip nie ma domyślnie zainstalowanego. Można go doinstalować poleceniem **sudo apt-get install python-pip** lub gdy chcemy korzystać z Pythona 3 (u nas dopiero tak działało) **sudo apt-get install python3-pip**. Wraz z programem python-pip doinstaluje się szereg narzędzi dla Pythona, które też mogą się przydać.

Ponadto instalator programu Yowsup2 kończy działanie poleceniem setup.py install, które wymaga uprawnień roota, by móc zmodyfikować wszystkie potrzebne pliki. Stąd instalację najlepiej wywołać za pomocą **sudo pip3 install yowsup2**. Gdyby instalacja nie powiodła się z jakiegoś powodu, można pobrać źródła programu yowsup i samodzielnie go skompilować. W tym celu należy:

Tabela 3. Aplikacje klienckie XMPP dla różnych systemów operacyjnych

Nazwa	Platforma (System operacyjny)
Adium	OSX
Apple Messages	OSX
AQQ	Windows
AstraChat	Mobile (Android, iOS) / Linux / OSX / Windows
Beem	Mobile (Android)
BitlBee	Linux
BlueJabb	Mobile (Android, Blackberry (BBOS), Nokia Symbian S40/S60 i Asha)
Boogie Chat	Mobile (iOS)
Buddycloud	Mobile / Web / Console
Candy	Przeglądarka
ChatSecure	Mobile (Android, iOS)
Coccinella	Linux / OSX / Windows
Conversations	Mobile (Android)
Converse.js	Przeglądarka
Coversant SoapBox Communicator	Windows
eM Client	Windows
Empathy	Linux
Finch	Console / Text-Mode
Gajim	Linux / Windows
GNU Freetalk	Console / Text-Mode
GreenJab	IBM i
IM+	Mobile
Instantbird	Linux / OSX / Windows
irssi-xmpp	Console / Text-Mode
jabber.el	Linux
Jabbim	Linux / OSX / Windows
JAJC	Windows
Jappix	Przeglądarka
Jitsi	Linux / OSX / Windows
JSXC	Przeglądarka
JWChat	Przeglądarka
Kadu	Linux / OSX / Windows
Kaiwa	Przeglądarka

Tabela 3. cd.

Nazwa	Platforma (System operacyjny)
Kopete	Linux
Mcabber	Console / Text-Mode
Miranda IM	Windows
Miranda NG	Windows
Monal IM	Mobile (iOS)
Movim	Przeglądarka
Mozilla Thunderbird	Linux / OSX / Windows
OneTeam for iPhone	Mobile (iOS)
OneTeam	Linux / OSX / Windows
Pidgin	Linux / OSX / Windows
Poezio	Console / Text-Mode
Profanity	Console / Text-Mode
Psi+	Linux / OSX / Windows
Psi	Linux / OSX / Windows
Quiet Internet Pager	Windows
qutIM	Linux / OSX / Windows
Salut à Toi	Linux / Console / Text-Mode / Przeglądarka
Sim-IM	Linux
Spark	Linux / OSX / Windows
SparkWeb	Przeglądarka
Swift	Linux / OSX / Windows
Talkonaut	Mobile
Tigase Messenger	Przeglądarka
Tigase Minichat	Przeglądarka
Tkabber	Linux / OSX / Windows
Trillian	Windows/ OSX / Mobile / Przeglądarka
V&V Messenger	Windows
Vayusphere	Mobile (BlackBerry)
VSTalk	Windows
WTW	Windows
Xabber	Mobile (Android)
xmpp-client	Linux / OSX
Xmppchat	Przeglądarka
XMPPWebChat	Przeglądarka
Yaxim	Mobile (Android)

- utworzyć podkatalog na program yowsup, wejść do niego a następnie pobrać spakowane źródła z githuba **wget https://github.com/tgalal/yowsup/archive/master.zip**,
- rozpakować je **unzip master.zip**,
- przejść do powstałego katalogu yowsup.

W naszym wypadku kompilacja była możliwa po pobraniu dodatkowych bibliotek oraz aktualizacji jednej z bibliotek programu pip. Konkretnie należało doinstalować pakiety poleceniami:

sudo apt-get install python3-dateutil
sudo apt-get install python3-setuptools
sudo apt-get install python3-dev
sudo apt-get install libevent-dev
sudo apt-get install ncurses-dev

a następnie zaktualizować bibliotekę six, która odpowiada za konwersję pomiędzy pythonem 3 a starszym **sudo pip3 install six –upgrade**. Wtedy można w końcu rozpocząć kompilację, która powinna przebiec już bez błędów, wywołując **sudo python3 setup.py install** w katalogu z programem. Następnie należy przeprowadzić autoryzację numeru telefonu podłączonego do sieci WhatsAppa. W tym celu należy wywołać polecenie **yowsup-cli registration –requestcode sms –phone 48123456789 –cc 48 –mcc 260 –mnc 98**, przy czym liczbę przy opcji phone należy zamienić na posiadany numer telefonu, parametr cc określa numer kierunkowy kraju, parametr mcc to tzw. Mobile Country Code – numer identyfikujący kraj pod kątem sieci komórkowych. Potrzebny jest też parametr mnc, który identyfikuje konkretną sieć komórkową w ramach danego kraju. Listę kodów MMC można znaleźć pod adresem ***https://goo.gl/XIqumG***, a polskich MNC (Mobile Network Code) na stronie ***https://goo.gl/Oc3Duy***. Kompletne tabele kodów państw i sieci oraz stan tych sieci można znaleźć pod adresem ***https://goo.gl/17qUqc***. Pomyślna rejestracja powinna zakończyć się potwierdzeniem („status: ok"), a ustawienia powinny zostać zapisane w pliku konfiguracyjnym.

Jeśli wszystko jest OK, wysłanie wiadomości whatsAppowej z Raspberry Pi będzie

wymagało już tylko wydania polecenia **yowsup-cli demos -s 48987654321 „To jest tresc wiadomosci”**, gdzie pierwszy parametr to numer telefonu adresata, a drugi to oczywiście treść nadawanej wiadomości.

Wysyłanie wiadomości można zautomatyzować, wbudowując w skrypty obsługujące elementy automatyki domowej odpowiednie procedury, które samodzielnie będą wywoływać komendę yowsup-cli z adekwatnymi parametrami. Można w ten sposób np. informować o przekroczeniu pożądanej temperatury na wybranym czujniku czy np. o wykryciu zdarzenia otwarcia drzwi do mieszkania.

XMPP na darmowym serwerze

WhatsApp jest usługą, którą kontroluje jedna konkretna firma i może zajść sytuacja, że zmieni coś w działaniu swojego serwisu, powodując, że istniejąca aplikacja przestanie działać. W przypadku skorzystania z XMPP unikamy tego problemu, szczególnie jeśli sami postawimy serwer XMPP.

Na potrzeby tego przykładu, by skrócić artykuł, skorzystamy jednak z gotowego, darmowego serwera XMPP. Wybór tego typu usług jest bardzo duży – można je znaleźć spisane pod adresem ***https://goo.gl/GYtZVa***, gdzie aktualnie zabrano 150 pozycji. Każdy z serwerów na liście ma podaną datę utworzenia, kraj, w którym się znajduje, organizację, która wydała mu certyfikat bezpieczeństwa, wykorzystywane oprogramowanie (dominuje Prosody i ejabberd) wraz z datą aktualizacji oraz ocenę klasy bezpieczeństwa. W Polsce dostępnych jest kilka serwerów, z czego nie wszystkie mają dobre oceny bezpieczeństwa. My skorzystaliśmy z serwisu ***jabster.pl***, który jak się okazało, wskazuje aktualnie na serwery ***jabbim.pl***.

Rejestracja w serwisie jabbim.pl jest bardzo łatwa. Wystarczy podać pożądany login i hasło oraz potwierdzić, że nie jest się robotem (**rysunek 1**). Chwilę później pojawia się ekran potwierdzenia. Jeśli wcześniej nie korzystało się z żadnego konta XMPP, warto stworzyć sobie od razu drugie, na którym będziemy odbierać wiadomości z Raspberry Pi. My do komunikacji z serwerem Jabstera będziemy używać gotowego, perlowego oprogramowania, dostępnego w repozytoriach Raspberry Pi – programu sendxmpp. Można go zainstalować wraz ze wszystkimi potrzebnymi zależnościami za pomocą polecenia **sudo apt-get install sendxmpp**.

Obsługa programu sendxmpp również jest łatwa. Najpierw należy stworzyć plik z danymi konfiguracyjnymi, a więc z nazwą konta i hasłem. Plik ten powinien się nazywać .sendxmpprc i być dostępny tylko dla właściciela. Tworzymy go poleceniem **nano ~/.sendxmpprc**, a w jego „wnętrzu” wpisujemy w jednej linijce, najpierw nazwę konta, z którego ma korzystać Raspberry Pi,

Rysunek 1. Rejestracja w serwisie jabbim.pl

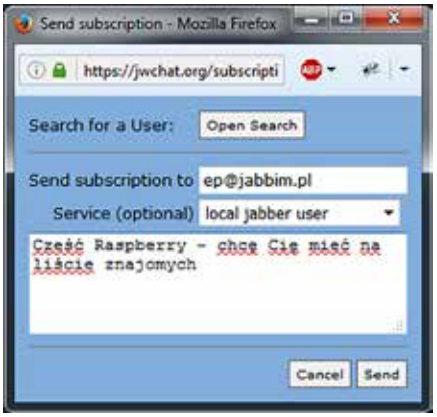
Rysunek 2. Webowy klient XMPP – jwchat.org

a następnie, po spacji, hasło. Po zapisaniu pliku należy zmienić dla niego uprawnienia. Inaczej program nie będzie działać **chmod 600 ~/.sendxmpprc**.

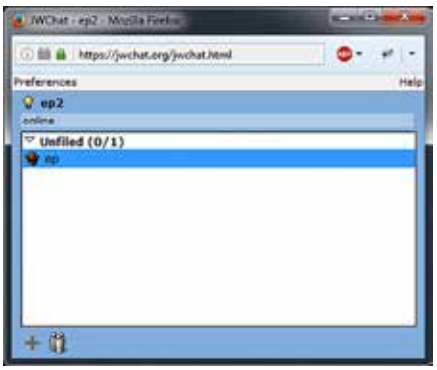
Teoretycznie już teraz można korzystać z usługi, ale warto chwilę poczekać – z naszych doświadczeń wynika, że serwer ***Jabbim.pl*** potrzebuje kilku minut, zanim „zorientuje się”, że ma nowego użytkownika. Samo wysłanie wiadomości wymaga przekazania na wejście programu sendxmpp przesyłanej treści oraz wskazania nazwy adresata. Przykładowo w ten sposób **echo „To jest wiadomość z Raspberry Pi” | sendxmpp -t ep2@jabbim.pl**. Jeśli na razie nie chcemy instalować żadnego specjalnego komunikatora, albo użyć dotychczasowych kont na potrzeby testów, możemy użyć darmowego, webowego klienta XMPP – np. jwchat.org (**rysunek 2**). Wystarczy wejść przeglądarką na stronę jwchat.org i zalogować się za pomocą danych podanych w jabbimie. My zalogowaliśmy się za pomocą drugiego z utworzonych kont.

Następnie warto dodać nowy kontakt do listy, tj. wysłać zaproszenie do konta użytego w Raspberry Pi (**rysunek 3**). Nie ma potrzeby potwierdzać przyjęcia zaproszenia, jeśli nie zależy nam na dzieleniu się statusami wiadomości. Mając już kontakt na liście (**rysunek 4**), możemy natomiast otworzyć czat do niego i rozmawiać z naszym minikomputerem.

Powłoka bash z Raspbiana pozwala na tak wiele, że możemy już za samą jej pomocą, bez



Rysunek 3. Dodanie nowego kontaktu do listy



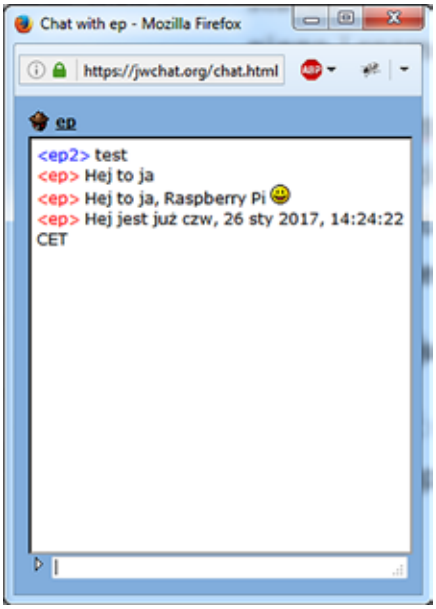
Rysunek 4. Lista kontaktów

konieczności pisania dodatkowych programów, przesyłać dosyć duo informacji przez komunikator. Przykładowo w komendzie echo można wyświetlić wynik dowolnego polecenia systemowego. Jeśli chcemy przesłać z Raspberry Pi aktualną datę, możemy użyć polecenia **echo „Jest już ” $(date) | sendxmpp -t ep2@jabbim.pl**. Efekt widać na **rysunku 5**. To otwiera wiele możliwości. Jeśli np. mamy do Raspberry Pi dołączony czujnik temperatury, z którego wskazanie pobieramy poleceniem **~/temp**, to wywołując **echo „Aktualna temperatura w domu to: ” $(~/temp) | sendxmpp -t ep2@jabbim.pl**, prześlemy ją do naszego klienta.

Raspberry Pi ma domyślnie wbudowany sensor temperatury, którego wskazanie można zdobyć poleceniem **/opt/vc/bin/vcgencmd measure_temp** i już teraz można je wykorzystać, bez żadnych dodatkowych czujników, ale trzeba mieć na uwadze, że będzie to wartość temperatury procesora, a nie otoczenia.

Regularnie lub w razie potrzeby

Raspberry Pi może wysyłać informacje regularnie, co określony czas, albo np. tylko w razie potrzeby, gdy wystąpi określone zdarzenie. W tym drugim przypadku wystarczy po prostu napisać stosowny program i umieścić w nim w odpowiednim miejscu wywołanie polecenia systemowego echo z określonymi parametrami. Natomiast, jeśli chcemy regularnie



Rysunek 5. Efekt działania polecenia „echo”

przekazywać jakieś dane, warto skorzystać z programu crond, który domyślnie dostępny jest w większości, jak nie wszystkich linuksach.

Program crond działa w oparciu o plik konfiguracyjny, tzw. crontab. W Raspberry Pi można go utworzyć lub edytować za pomocą polecenia **crontab -e**. Przy pierwszym uruchomieniu system zapyta o edytor tekstu, którego chcemy użyć do zmiany pliku konfiguracyjnego.

Listing 1. Kod skryptu przesyłającego ostrzeżenie o zbyt wysokiej temperaturze

```
#!/bin/bash
#pobieranie temperatury procesora i wycinanie z niej jedynie wartości liczbowej, po czym
przypisywanie jej do zmiennej temp
temp=$(/opt/vc/bin/vcgencmd measure_temp | cut -c6-7)
#sprawdzanie czy temperatura przekracza 45 stopni Celsjusza
if [ „$temp” -gt 45 ]; then
          #przesyłanie ostrzeżenia przez XMPP
          echo Procesor RPi ma zbyt wysoką temperaturę: $(/opt/vc/bin/vcgencmd measure_
temp) | sendxmpp -t ep2@jabbim.pl
fi
```

Struktura pliku crontab składa się z linii, w których kolejno podawane są minuty, godziny, dni miesiąca, miesiące i dni tygodnia oraz komendy do wywołania. Poszczególne wartości są od siebie oddzielane tabulatorami lub spacjami. Program crond po uruchomieniu wczytuje tablicę konfiguracyjną do pamięci i następnie co minutę sprawdza, które z wierszy należy w danej chwili wykonać. Wpisanie liczby na pozycji odpowiadającej np. godzinie będzie oznaczało, że komenda z tej linii ma się wykonać tylko, gdy aktualna godzina odpowiada wpisanej wartości. Należy jednak pamiętać, że by komenda była wykonana, muszą być spełnione wszystkie warunki w linii, czyli jeśli chcemy, by akcja była uruchamiana w poniedziałki o godzinie 15, musimy w drugiej kolumnie wpisać 15, a w piątej 1. W pozostałe kolumny powinniśmy wpisać * (gwiazdki), jeśli minuty, dni i miesiące nie mają dla nas znaczenia. Trzeba jednak zauważyć, że pozostawienie gwiazdki w polu minut sprawi, że komenda zostanie wykonana w każdy poniedziałek łącznie 60 razy – od 15:00 do 15:59, co minutę. Jeśli komenda ma być wykonywana tylko o 15:00, musimy w kolumnie minut wpisać 0. Możemy też zażądać, by wykonywała się co 5 minut – wtedy w kolumnie minut wpisujemy */5.

W ramach komend wskazywanych w tablicy crontab można podawać nazwy skryptów, które mogą zawierać szereg poleceń oraz zdarzenia warunkowe. W ten sposób możemy np. co 5 minut sprawdzać temperaturę, ale powiadamiać użytkownika tylko o jej przekroczeniu. Wtedy w crontabie wpisujemy linijkę ***/5 * * * * ~/temp.sh**, a w katalogu domowym użytkownika tworzymy skrypt temp.sh, taki jak na **listingu 1**.

Podsumowanie

Zaprezentowany sposób to jeden z najszybszych, by utrzymywać powiadomienia komunikatorem z Raspberry Pi. Dalej program można rozbudować o przyjmowanie zdalnych poleceń, wykorzystanie własnego serwera oraz o wykorzystanie eksperymentalnego protokołu XMPP-IoT, który jest opracowywany pod kątem Internetu Rzeczy. Ma on rozwiązywać m.in. problem akceptowania zaproszeń do listy znajomych przez komunikujące się ze sobą rzeczy.

Marcin Karbowniczek, EP

REKLAMA

młody m.technik
Ciekawi świata są zawsze młodzi

w prezencie
na
każdą okazję

https://goo.gl/TiDLmR