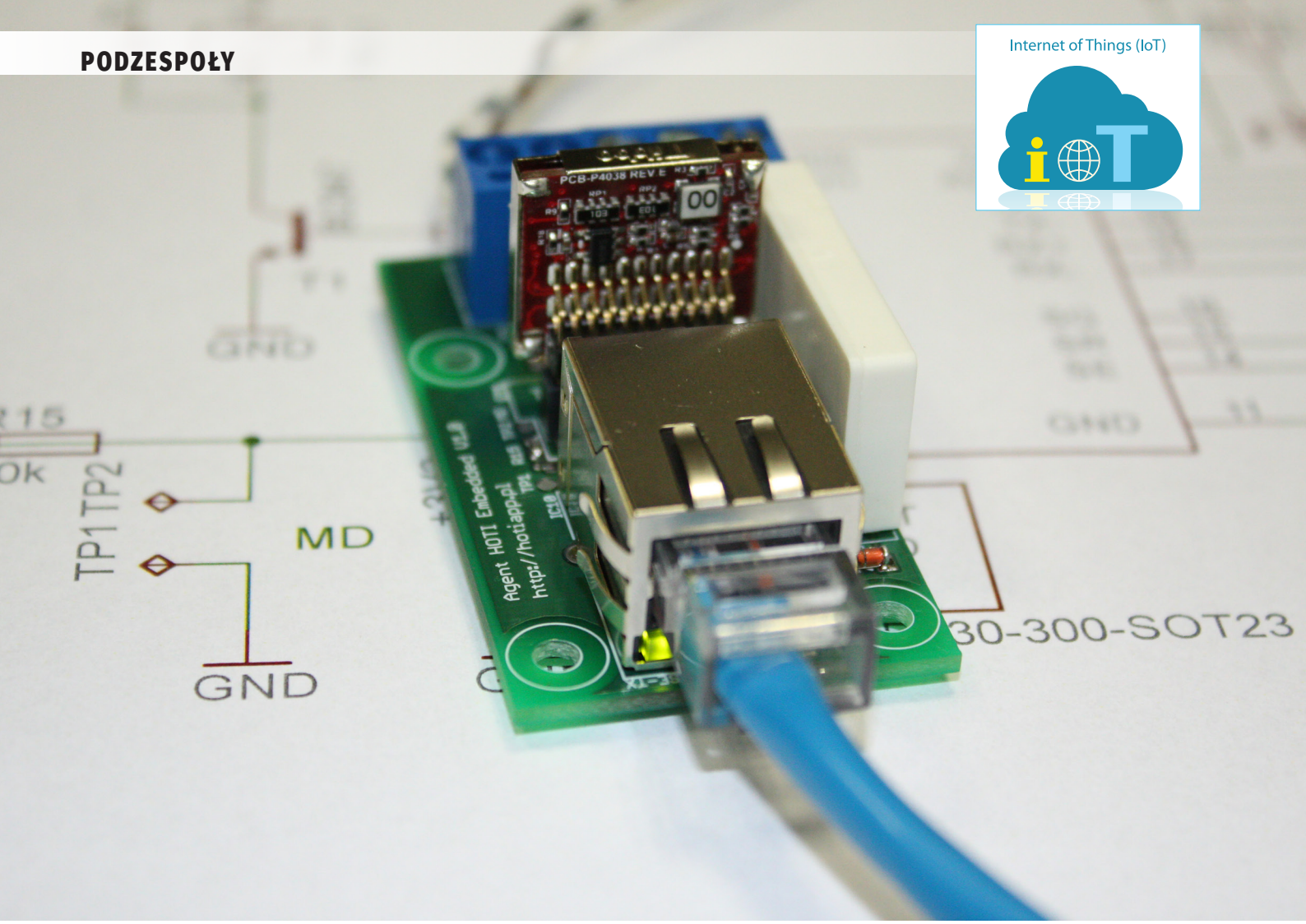


Internet of Things – krok po kroku

Zastosowanie modułu Tibbo EM500

W artykule przedstawiono internetowy system zdalnego sterowania, który dzięki swojej funkcjonalności wpisuje się w aktualny trend „Internetu Przedmiotów". System składa się z trzech komponentów: Agenta, serwera AggreGate oraz interfejsu WEB. Agent jest urządzeniem końcowym instalowanym po stronie użytkownika. Jest wyposażony w jeden przekaźnik elektromagnetyczny ze stykami NO/NC umożliwiającymi najprostszą akcję włączania i wyłączania zasilania odbiornika, którym chcemy sterować. Od strony sieci Agent wymaga połączenia przewodowego (np. z routerem) z Internetem. Agent jest elementem bezobsługowym wymagającym jedynie zasilania +5 V DC.

Architekturę systemu, którym posłużono się w artykule pokazano na **rysunku 1**. Serwer AggreGate jest gotowym komponentem firmy Tibbo, który w dużym uproszczeniu, zarządza utrzymywaniem połączeń z komunikującymi się z nim Agentami. Od strony użytkownika serwer udostępnia interfejs API w postaci WebServis'u. Interfejs ten umożliwia podgląd listy zalogowanych do serwera Agentów, sprawdzania statusu ich połączenia oraz wykonywanie udostępnianych przez Agentów funkcji (w naszym przypadku będą to funkcje pozwalające na sterownie przekaźnikiem oraz funkcje diagnostyczne).

Trzecim elementem systemu jest strona WWW z przyjaznym użytkownikowi interfejsem i z kontrolą dostępu (uwierzytelnianie hasłem).

Agent

Agent jest właściwym elementem wykonawczym systemu. Zbudowany został w oparciu o moduł programowalny (w języku C oraz w BASIC'u) EM500 firmy Tibbo. Moduł charakteryzuję się miniaturową budową. Znikome wymiary urządzenia (18,5 mm×16 mm×6,5 mm) wynikają z pozbawienia go zintegrowanego gniazda RJ45 (oraz transformatora).

Wygląd EM500 zaprezentowano na **rysunku 2**. Wyprowadzenia zgrupowano w jednym 22-pionowym konektorze (wykaz wyprowadzeń umieszczono w **tabeli 1**). Moduł dysponuje 512 kB pamięci Flash, która jest dzielona pomiędzy aplikację użytkownika (320 kB) a system operacyjny TiOS (192 kB). Ponadto, moduł ma 208 bajtów

Tabela 1. Wykaz wyprowadzeń modułu EM500 (źródło: dokumentacja Tibbo)

Numer wypr.	Funkcja	Opis
1 (1,2,3)	GPIO0/P0.0/INT0	I/O (P0.0); linia przerwania 0
2 (1,2,3)	GPIO1/P0.1/INT1	I/O (P0.1); linia przerwania 1.
3 (1,2)	GPIO2/P0.2	I/O (P0.2)
4 (1,2)	GPIO3/P0.3	I/O (P0.3)
5 (1,2)	GPIO4/P0.4	I/O (P0.4)
6 (1,2)	GPIO5/P0.5	I/O (P0.5)
7 (1,2)	GPIO6/P0.6	I/O (P0.6)
8 (1,2)	GPIO7/P0.7	I/O (P0.7)
9 (1)	RX	Linia wejścia portu szeregowego
10 (1)	TX	Linia wyjścia portu szeregowego
11	GND	GND
12	MD	Zewnętrzny przycisk MD (w praktyce jest to kolejna linia przerwania)
13	RST	Wejście zerujące
14	SE	Linia diody statusowej linku Ethernetowego
15	SR	Czerwona dioda statusowa LED
16	SG	Zielona dioda statusowa LED
17	RX-	Port Ethernet: linia RX-
18	RX+	Port Ethernet: linia RX+
19	TX-	Port Ethernet: linia TX-
20	TX+	Port Ethernet: linia TX+
21	AVCC	Wyjście napięcia zasilającego obwód zewnętrznego transformatora Ethernet.
22	VCC	Zasilanie układu: 3.3V (min 260mA).

Uwagi:
1. Akceptuje poziomy napięć CMOS (+5 V)
2. Może pełnić funkcję linii RTS/Wout/cout portu szeregowego
3. Może pełnić funkcję linii CTS/W0&1in/cin portu szeregowego

pamięci EEPROM (głównie do przechowywania danych konfiguracyjnych). W EM500 pamięć ta nabiera szczególnego znaczenia, gdyż moduł nie ma funkcjonalności wewnętrznego dysku Flash. System operacyjny modułu EM500 udostępnia wprawdzie obiekt *fd* (będący sterownikiem dysku Flash), jednak może on współpracować jedynie z zewnętrznymi pamięciami z interfejsem SPI. Innymi słowy – pamięć EEPROM jest jedyną pamięcią nieulotną modułu, dostępną z poziomu aplikacji.

Pamięć RAM modułu nieznacznie przekracza 16 kB i jest współdzielona pomiędzy zmiennymi a buforami interfejsów komunikacyjnych (ethernetowego oraz szeregowego). W praktyce oznacza to, że im więcej pamięci przeznaczymy na obsługę wielu połączeń TCP, tym mniej pozostanie do dyspozycji dla wykonywanej aplikacji.

Pewną komplikacją w stosowaniu EM500 jest to, że wymaga zewnętrznego obwodu zerowania. Ponadto, producent rekomenduje użycie specjalizowanych układów resetu np. MCP130-300, co nie wpływa korzystnie na budżet projektu. Stosowanie zewnętrznego transformatora oraz gniazda RJ45 również wymaga nieco uwagi.

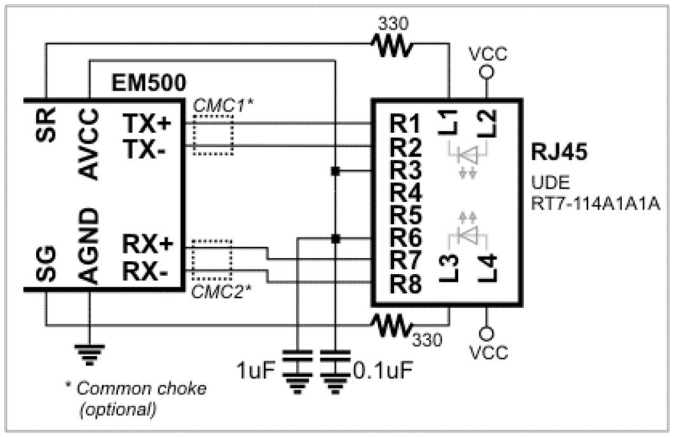
Rysunek 1. Architektura systemu.

Zgodnie zaleceniami połączenia pomiędzy modułem a elementami zewnętrznymi powinny być jak najkrótsze, co wymusza umiejscowienie modułu bardzo blisko gniazda. Port Ethernet jest zgodny ze standardem 10/100BaseT oraz auto-MDIX. Przykładowy schemat połączeń pokazano na **rysunku 3**.

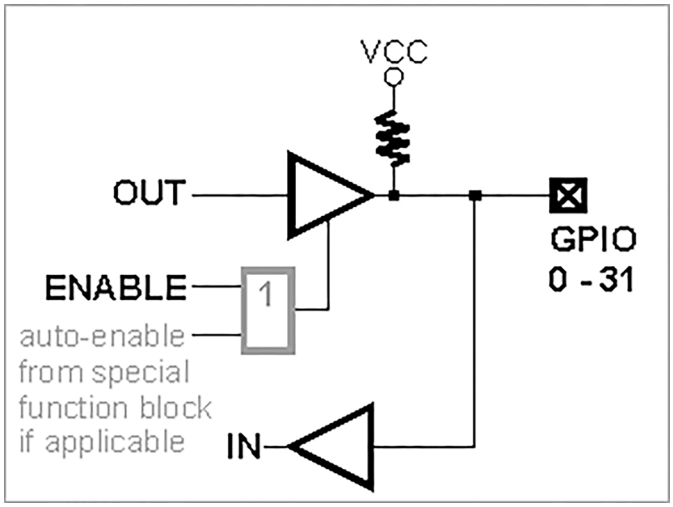
Poza interfejsem Ethernet moduł ma 8-bitowy port wejścia/wyjścia oraz wyprowadzeniami portu szeregowego (prędkość transmisji do 460,8 kb/s, kilka trybów pracy). Wewnętrzna budowa linii I/O wymaga krótkiego komentarza:

Rysunek 2. Moduł EM500

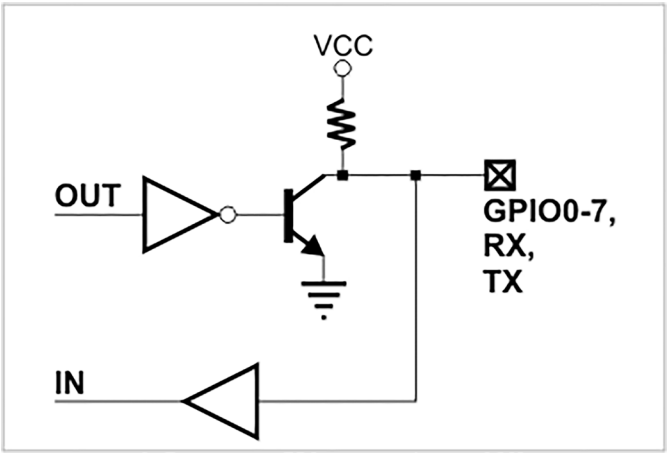
Standardowo wyprowadzenie portu może znajdować się w jednym z trzech stanów: z poziomem wysokim, z poziomem niskim lub w stanie wysokiej impedancji (w tym wypadku linia pracuje jako wejście). W praktyce oznacza to, że przy obsłudze jednej linii I/O musimy manipulować dwoma rejestrami: jednym odpowiedzialnym za kierunek linii, drugim za występujący na niej poziom. Tak zaprojektowane bufory linii wejściowych są w starszych (niż EM500) platformach (EM1202, EM1206, EM1000). Budowę takiego bufora pokazano na **rysunku 4**. Linia *ENABLE* jest odpowiedzialna za załączenie/wyłączenie drivera wyjścia (kontrolowanego przez linię OUT) przy przełączaniu trybu pracy. W module EM500 zastosowano inną budowę (**rysunek 5**). Bufor wyjściowy



Rysunek 3. Schemat podłączenia gniazda RJ45 ze zintegrowanym transformatorem (za dokumentacją Tibbo)



Rysunek 4. Budowa wewnętrznego bufora linii I/O w modułach serii EM1000



Rysunek 5. Budowa wewnętrznego bufora linii I/O w module EM500

Listing 1. Funkcja *agg_start()*
```
agg_start(  interface as pl_sock_interfaces,
            byref owner_name as string,
            byref device_name as string,
            byref password as string,
            byref agg_server_ip as string,
            agg_server_port as word,
            agg_server_tout as word,
            auto_reg as no_yes) as en_agg_status_codes
```

to tranzystor pracujący w układzie otwartego kolektora (niemalże otwartego – w obwodzie występuje rezystor podciągający o dużej rezystancji). Przy stosowaniu rozwiązań tego typu należy pamiętać o różnicach w wydajności prądowej. Dla trybu *sink* (poziom niski) port może przewodzić prąd o maksymalnym natężeniu 10 mA. Natomiast wydajność prądowa linii na poziomie wysokim jest znikoma, co wynika z dużej rezystancji opornika podciągającego. Ograniczenie to pozwala jednak na pracę linii w charakterze wejścia – właśnie z uwagi na dużą oporność rezystora, który determinuje impedancję wejściową linii. Należy jednak pamiętać o tym, iż aby odczytać wartość logiczną przyłożoną do linii wejściowej, należy najpierw ustawić ją (a tym samym – otworzyć tranzystor). Podejście takie nie upraszcza obsługi portu z poziomu programu (wciąż należy pamiętać o odpowiedniej konfiguracji linii dla odczytu stanu logicznego), zmniejsza jedynie liczbę rejestrów, którymi musimy manipulować.

Schemat ideowy Agenta i projekt jego płytki drukowanej można znaleźć w materiałach dodatkowych na serwerze FTP. Liczbę zewnętrznych elementów została ograniczona do minimum. Jedyną funkcją modułu jest sterowanie zewnętrznym przekaźnikiem elektromagnetycznym za pośrednictwem tranzystora bipolarnego. W obwodzie zasilania znajduje się liniowy regulator +5 V z kondensatorami filtrującymi. Obwód zerowania zrealizowano z użyciem wspomnianego wcześniej MCP130-300. Całość mieści się na niewielkiej dwustronnej płytce drukowanej.

Oprogramowanie Agenta zostało maksymalnie uproszone. Składa się ono z funkcji inicjalizujących interfejs sieciowy, uruchamiających wbudowany serwer *http* oraz inicjalizujących bibliotekę AggreGate, która jest odpowiedzialna za komunikację z częścią serwerową systemu.

Start biblioteki oraz połączenie z serwerem AggreGate realizowane są przy pomocy funkcji *agg_start()*.Funkcję pokazano na **listingu 1**, a opis argumentów umieszczono w **tabeli 2**.

Wywołanie funkcji w oprogramowaniu Agenta wygląda następująco:
```
agg_start(PL_SOCK_INTERFACE_NET,"iotagent",
                                config_mac_dec2hex(net.
mac),"agent2016",pub_agg_stg,6480,600,YES)
```

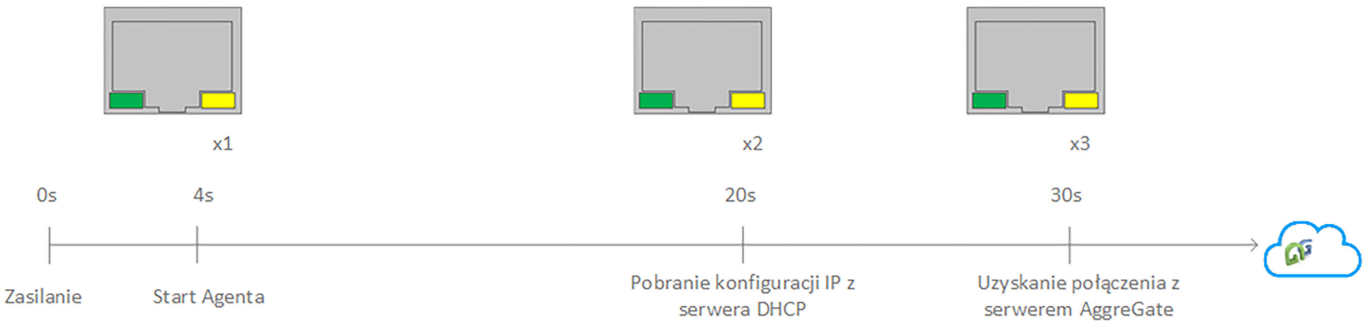
Nazwa Agenta to jego adres MAC. Adres odczytywany jest z własności *net.mac* a następnie jest konwertowany z postaci dziesiętnej na szesnastkową przy użyciu funkcji *config_mac_dec2hex()*. Adres IP serwera AggreGate jest przekazywany przez zmienną publiczną *pub_agg_stg*. Sygnalizacja aktualnego stanu Agenta prowadzona jest poprzez żółtą diodę LED w złączu Ethernet zgodnie ze schematem przedstawionym na **rysunku 6**.

Po podaniu zasilania moduł EM500 potrzebuje około 4 sekundy na uruchomienie aplikacji, co jest sygnalizowane pojedynczym błyskiem diody statusowej. Następnie, w sekwencji, realizowane są poniższe procedury:

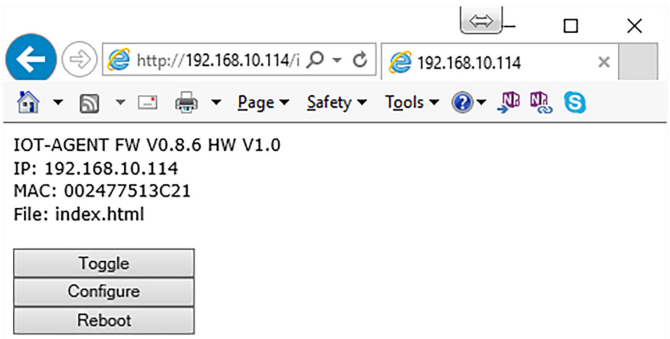
- Inicjalizowanie biblioteki „*STG – Settings*" będącej zbiorem funkcji odpowiedzialnych za zarządzanie pamięcią nieulotną modułu. W EM500 może to być wewnętrzna pamięć EEPROM lub zewnętrzna pamięć Flash (dołączona za pomocą SPI). Agent używa wyłącznie pamięci wewnętrznej, w której przechowywane są informacje o adresie MAC, rodzaju konfiguracji IP (statyczna bądź dynamiczna) oraz adresy IP: Agenta, bramy, i maski sieci dla konfiguracji statycznej.
- Jeśli moduł był skonfigurowany do pracy z dynamiczną konfiguracją IP, następuje inicjalizacja biblioteki DHCP i próba uzyskania adresów z lokalnego serwera DHCP. Przy powodzeniu, nowo uzyskane adresy IP, bramy i maski sieci zostają zapisane do pamięci nieulotnej oraz moduł sygnalizuje ten fakt podwójnym błyskiem diody statusowej. Jeśli natomiast nie udało się nawiązać komunikacji z serwerem DHCP, moduł zostanie skonfigurowany przy użyciu ostatnio zapisanych danych.
- Następnie moduł inicjalizuje lokalny serwer *http* (interfejs ten jest opisany w dalszej części) oraz linie I/O.

Tabela 2. Opis argumentów funkcji agg_start() (za dokumentacją Tibbo)

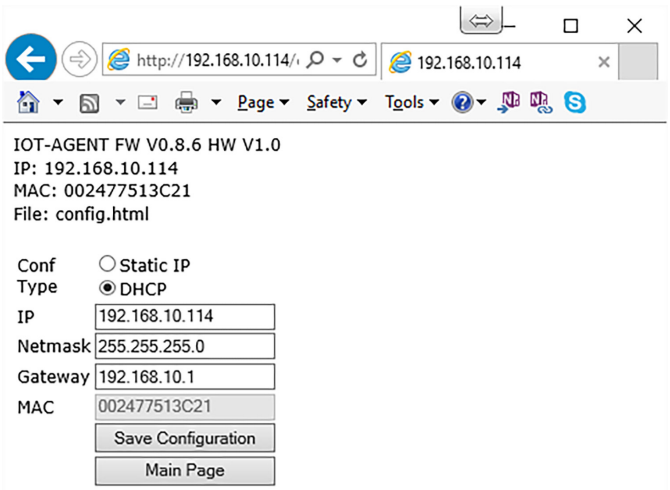
Lp.	Argument	Typ	Opis
1.	interface	pl_sock_interfaces	Interfejs sieciowy, którego biblioteka będzie używała do próby nawiązania połączenia z serwerem AggreGate. Ilość i rodzaj dostępnych interfejsów zależą od konkretnej platformy sprzętowej. W przypadku modułu EM500 będzie to interfejs Ethernet (PL_SOCK_INTERFACE_NET). W innych platformach dostępne są również: • PL_SOCK_INTERFACE_WLN – Wi-Fi • PL_SOCK_INTERFACE_PPP – interfejs wykorzystujący protokół PPP w przypadku stosowania modemów GPRS • PL_SOCK_INTERFACE_PPPeE – interfejs wykorzystujący PPP over Ethernet stosowany np. w modemach ADSL.
2.	owner_name	string	Nazwa użytkownika zdefiniowana na poziomie serwera AggreGate.
3.	device_name	string	Nazwa Agenta, pod którą moduł będzie występował na liści urządzeń użytkownika serwera AggreGate
4.	password	string	Hasło dostępu
5.	agg_server_ip	string	Adres IP serwera AggreGate
6.	agg_server_port	word	Port TCP serwera AggreGate
7.	agg_server_tout	word	Timeout komunikacji serwera z Agentem
8.	auto_reg	no_yes	Włączanie/wyłączanie autorejestracji agenta w serwerze AG



Rysunek 6. Schemat sygnalizacji stanu agenta od momentu włączenia zasilania



Rysunek 7. Główna strona interfejsu WWW Agenta



Rysunek 8. Podstrona konfiguracyjna Agenta

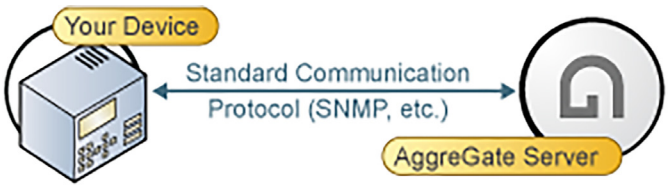
- Moduł inicjalizuje bibliotekę DNS i podejmuje próbę rozwiązania nazwy serwera AggreGate.
- W wypadku powodzenia następuje start biblioteki AGG odpowiedzialnej za komunikację modułu z serwerem. Po nawiązaniu połączenia dioda statusowa modułu mruga trzykrotnie.

Opisana sekwencja trawa około 30 sekund.

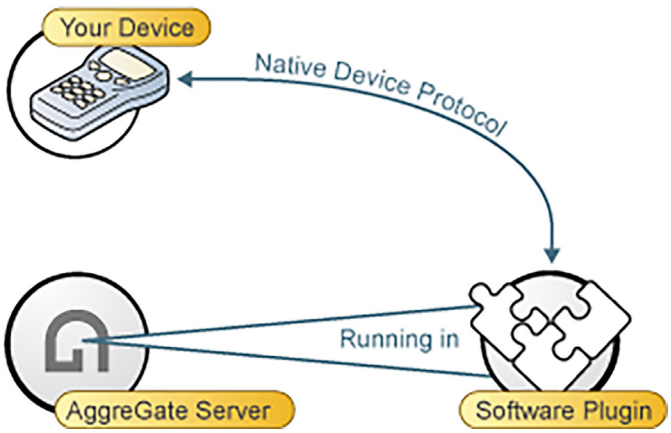
Jako dodatek Agent oferuje wbudowany interfejs WWW (dostępny z poziomu sieci lokalnej). Po wpisaniu w przeglądarce adresu IP agenta zostaniemy przeniesieni na stronę główną (**rysunek 7**). Dostępne są na niej informacje o adresie IP i MAC oraz trzy przyciski. Pierwszy z nich służy do przełączania stanów przekaźnika, drugi do przejścia do podstrony konfiguracyjnej Agenta (**rysunek 8**), trzeci do restartowania urządzenia. Na podstronie konfiguracyjnej możliwe jest przełączenie Agenta w tryb konfiguracji statycznej IP.

Serwer AggreGate (AG)

Jest to aplikacja rozwijana przez firmę Tibbo, która w zamyśle twórców ma służyć jako centralny punkt kontaktu dla zdalnych węzłów systemu. AggreGate Server jest napisany w Javie i możliwy do uruchomienia na systemach operacyjnych Windows, Linux i Mac OS.



Rysunek 9. Komunikacja z AG przy użyciu standardowego protokołu



Rysunek 10. Komunikacja z AG z użyciem dedykowanego sterownika

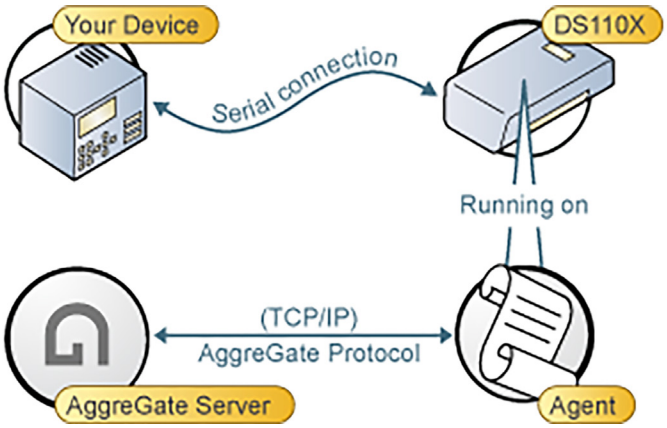
Urządzania (Agenci) w projektowanym systemie mogą być dołączone do serwera przy użyciu większości metod komunikacji, takich jak: Ethernet, WiFi, GPRS. System zapewnia rozbudowane mechanizmy uwierzytelniania oraz zarządzania użytkownikami i przyłączonymi Agentami. W opisywanym systemie będziemy używali jedynie części z dostępnych funkcji serwera AG.

AggreGate umożliwia komunikację z urządzaniami zdalnymi za pomocą kilku metod:

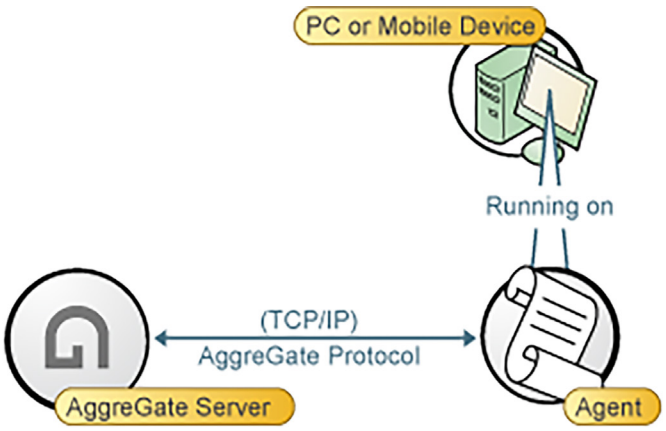
Komunikacja przy użyciu protokołów standardowych, takich jak SNMP, Modbus lub OPC. W tym wypadku nie musimy ingerować ani w część sprzętową, ani w oprogramowanie urządzenia, które chcemy dołączyć do systemu AG (**rysunek 9**).

Jeśli urządzenie nie używa do komunikacji standardowego protokołu wówczas możemy nawiązać połącznie za pośrednictwem sterownika (działającego po stronie serwera), który może być używany przez serwer jako plug-in. Implementacja takiego sterownika leży wówczas po naszej stronie, jednak wciąż nie ma konieczności ingerencji w część sprzętową urządzenia, które chcemy dołączyć do systemu (**rysunek 10**).

Agent sprzętowy. Jest to metoda wykorzystywana opisywanym systemie. Agent jest modułem, który za pośrednictwem przekaźnika może sterować fizycznymi funkcjami podłączanego urządzenia. Ciężar komunikacji sieciowej z serwerem AG leży wówczas po stronie Agenta, w którego oprogramowaniu zaimplementowano protokół komunikacyjny AG (**rysunek 11**). Metoda ta rozwiązuje ogólny problem komunikacji urządzenia ze światem zewnętrznym w przypadku, gdy taka komunikacja nigdy nie była przewidziana.



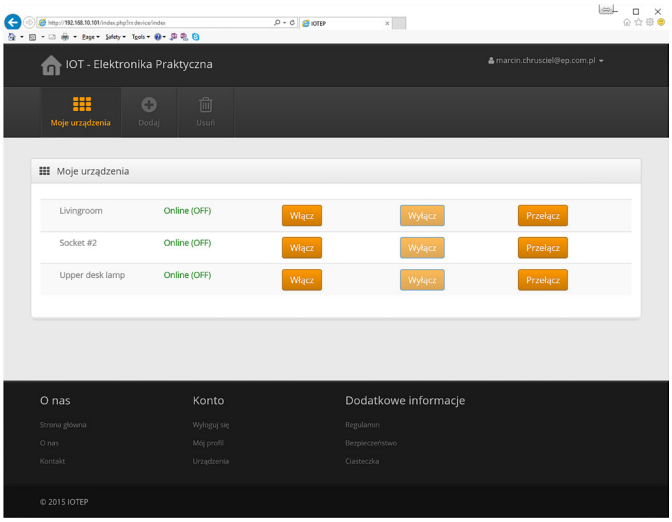
Rysunek 11. Komunikacja z AG przy użyciu Agenta sprzętowego



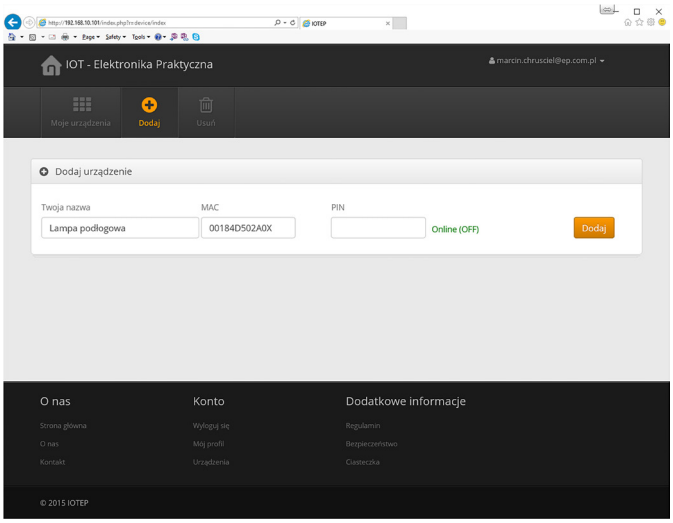
Rysunek 12. Komunikacja z AG przy użyciu Agenta programowego.

Agent programowy. Tibbo dostarcza (na licencji "*open source*") darmową bibliotekę dostępną w kilku językach programowania: Java, Android Java (Dalvik JVM), .NET, .NET Compact i C/C++). W bibliotece tej (podobnie jak w bibliotece AG napisanej dla Agenta) zaimplementowano protokół komunikacyjny (**rysunek 11**). Przy wyborze tej opcji komunikacji nie musimy modyfikować części sprzętowej urządzenia, ale oczywiście musimy mieć możliwość ingerencji w jego oprogramowanie.

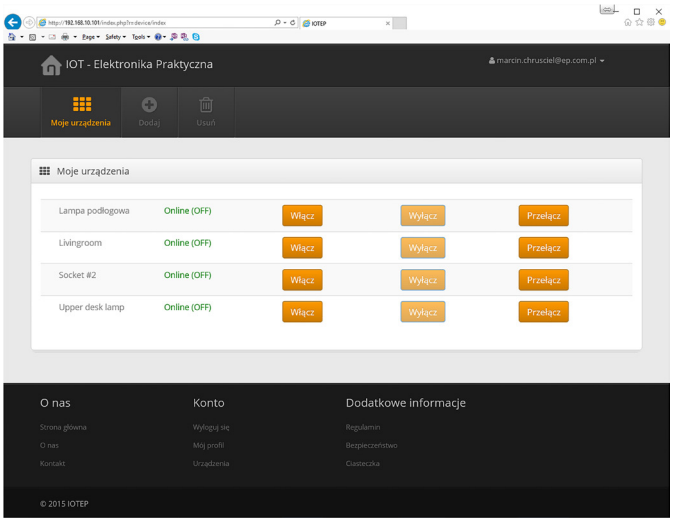
Do jednego serwera AG może być dołączonych wiele Agentów, z których każdy używa innej metody komunikacji. To czyni rozwiązanie niezwykle elastycznym oraz skalowalnym. Jako dodatek do Serwera AG Tibbo dostarcza niezależną aplikację klienta AG. Umożliwia ona dostęp uprawnionemu operatorowi do zasobów serwera oraz



Rysunek 13. Interfejs WWW systemu – widok urządzeń przypisanych do zalogowanego użytkownika.



Rysunek 14. Interfejs WWW systemu – ekran dodawania nowego urządzenia.



Rysunek 15. Interfejs WWW systemu – widok urządzeń po dodaniu nowego Agenta.

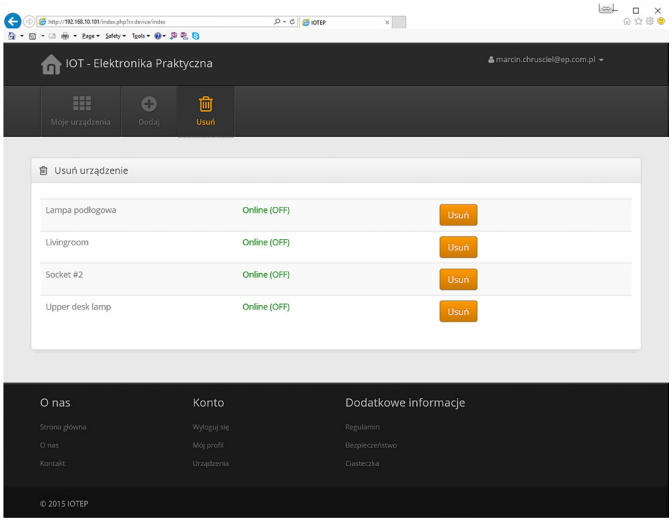
przeglądanie i zarzadzanie podłączonymi do niego Agentami. W tym kontekście jest to główne narzędzie administratora systemu.

Serwer AG udostępnia również kilka interfejsów komunikacyjnych przeznaczonych dla aplikacji tworzonych w ramach budowanego systemu. Najprostszym w użyciu jest WebService, nieco bardziej wymagającymi są SDK dla Javy oraz dla .NET.

Interfejs WWW

Stronę służącą do konfigurowania Agenta wykonano w PHP przy użyciu frameworka Yii. Natychmiast po zalogowaniu się, użytkownik otrzymuje dostęp do listy swoich urządzeń (Agentów) wraz z udostępnionymi funkcjami. U nas jest to wyłącznie kontrola przekaźnika (**rysunek 12**).

Poza kontrolą agentów użytkownik może dodawać i usuwać swoje urządzenia. Przy dodawaniu (**rysunek 13**). W tym celu należy znać adres MAC agenta oraz 4-cyfrowy kod PIN (adresy MAC i kody przechowywane są w bazie danych, do której dostęp ma tylko administrator systemu). Po wpisaniu adresu MAC system automatycznie sprawdza czy Agent został dołączany do sieci oraz czy zdołał się zalogować do serwera AG (zielony opis *Online(Off)*). Następnie, możemy dodać nazwę własną Agenta (np. *Lampa podłogowa*), po czym wpisujemy PIN oraz klikamy przycisk „*Dodaj*". Jeśli wszystko przebiegło prawidłowo wówczas system przełączy nas z powrotem do zakładki „*Moje urządzenia*", gdzie powinniśmy już mieć możliwość sterowania „Lampą podłogową" (**rysunek 13**). Ostatnią opcją jest usuwanie



Rysunek 16. Interfejs WWW systemu – widok ekranu usuwania urządzeń

urządzeń z listy. Funkcjonalność ta właściwie nie wymaga szerszego opisu (**rysunek 14**).

Podsumowanie

Systemy IOT nie różnią się właściwie od wcześniej popularnych systemów nazywanych M2M („*machnie to machnie*"), gdzie urządzeniami końcowymi podłączonymi do sieci Internet nie jest komputer PC, ale jednostka wbudowana spełniająca określoną funkcję. Sama komunikacja przy użyciu Internetu nie jest też wyzwaniem technologicznym. W systemach IOT ciężar przerzucono na łatwość obsługi i brak konieczności znajomości protokołów, konfiguracji sieci itp. W prezentowanym systemie udział użytkownika na etapie konfiguracji również został ograniczony do minimum. Musi on jedynie podłączyć Agenta do wolnego gniazdka Ethernet oraz zalogować się do interfejsu WWW, aby moc sterować urządzeniem wyposażonym w Agenta. Sam interfejs WWW jest skalowalny i dobrze wyświetla się na urządzeniach przenośnych. Ponieważ w systemie występuje trzecia strona (czyli serwer AG), to nie ma konieczności rekonfiguracji sieci lokalnej, za której pomocą Agent łączy się z Internetem. Połącznie do serwer AG zawsze jest inicjalizowane przez Agenta z wnętrza sieci lokalnej (podobnie zresztą jak każde połącznie z interfejsu WWW).

Sam serwer AG ma znacznie więcej możliwości, niż tylko te wykorzystywane w opisywanym systemie, gdzie poza odpytaniem Agentów o udostępniane przez nich metody i ich wykonywanie serwer jedynie podtrzymuje aktywne połącznie z Agentami. AG ma szereg modułów do integracji choćby z systemami typu SCADA. Pojemość systemu (liczba możliwych do obsłużenia Agentów) jest ograniczona wyłącznie zasobami sprzętowymi serwera (fizycznej maszyny), na którym jest uruchomiony AG. Serwer AG ma wbudowane mechanizmy typu *failover*, aby maksymalnie podnieść niezawodność systemu.

Opisany system ma bardzo ograniczoną funkcjonalność. Nic nie stoi jednak na przeszkodzenie, aby dołączyć do niego bardziej złożone moduły Agentów np. z większą liczbą linii sterujących lub z torami pomiarowymi. Każda taka rozbudowa będzie oczywiście wymagała zmiany części sprzętowej i oprogramowania samego Agenta oraz możliwości interfejsu WWW. Nie będziemy musieli natomiast zmieniać części serwerowej AG, co jest znacznym uproszczeniem przy rozwoju systemów IOT.

Marcin Chruściel