**Listing 1. Główny kod programu javascriptowego**

// ładowanie bibliotek

**var** iot **=** require**(**'aws-iot-device-sdk'**);**

**var** \_ **=** require**(**'lodash'**);**

**var** SimCom **=** require**(**'simcom'**).**SimCom**;**

// konfiguracja połączenia z serwerami AWS IoT

**var** device **=** iot**.**device**({**

 keyPath**:** '/home/pi/AWSCerts/privateKey.pem'**,**

 certPath**:** '/home/pi/AWSCerts/cert.pem'**,**

 caPath**:** '/home/pi/AWSCerts/root.pem'**,**

 clientId**:** 'rpi01'**,**

 region**:** 'us-east-1'

**});**

// połączenie z modułem GSM poprzez kabel USB

**var** simcom **=** **new** SimCom**(**'/dev/ttyUSB0'**);**

// przypisanie akcji logowania informacji do zdarzenia otwarcia połączenia z modułem GSM

simcom**.**on**(**'open'**,** **function()** **{**

 console**.**log**(**'simcom open'**);**

**});**

// przypisanie akcji logowania informacji do zdarzenia błędu połączenia w module GSM

simcom**.**on**(**'error'**,** **function()** **{**

 console**.**log**(**'simcom error'**);**

**});**

// funkcja wywołująca komendę AT

**function** executeAtCmd**(**atCmd**)** **{**

 // logowanie informacji o uruchamianej komendzie

 console**.**log**(**'Execute AT Command: [' **+** atCmd **+** ']'**);**

 // wysłanie komendy AT do modemu GSM

 simcom**.**modem**.**execute**(**atCmd**).**then**(function(**lines**)** **{**

 // w przypadku powodzenia:

 // logowanie informacji zwrotnej z modemu

 console**.**log**(**'AT Response'**,** lines**);**

 // przesłanie odpowiedzi modemu do serwera w chmurze

 sendToServer**(**'AT Response: ' **+** JSON**.**stringify**(**lines**));**

 **},** **function(**error**)** **{**

 // w przypadku błędu:

 // logowanie informacji o błędzie z modemu

 console**.**error**(**'AT Command Error'**,** error**);**

 // przesłanie zwróconego przez modem błędu do serwera w chmurze

 sendToServer**(**'AT Command Error: ' **+** JSON**.**stringify**(**error**));**

 **});**

**}**

// funkcja wysyłająca wiadomość SMS przez modem

**function** sendSms**(**to**,** message**)** **{**

 console**.**log**(**'Sending SMS: ' **+** to **+** ' : ' **+** message**);**

 simcom**.**sendSMS**(**to**,** message**);**

 sendToServer**(**'Sending SMS: ' **+** to **+** ' : ' **+** message**);**

**}**

// przesłanie wiadomości do serwera w chmurze, do tematu subskrybowanego przez administratora

**function** sendToServer**(**message**)** **{**

 device**.**publish**(**'test/topic1'**,** message**);**

**}**

// przypisanie akcji do zdarzenia nawiązania połączenia z serwerem AWS IoT

device**.**on**(**'connect'**,** **function()** **{**

 console**.**log**(**'iot: connect'**);**

 // rozpoczęcie subskrypcji tematu używanego do przekazywania wiadomości do kota

 device**.**subscribe**(**'test/topic2'**);**

 // przesłanie wiadomości do chmury, informującej administratora o rozpoczęsciu nasłuchiwania komend przez kota

 sendToServer**(**'iot: connect'**);**

**});**

// przypisanie akcji do zdarzenia nadejścia wiadomości z chmury

device**.**on**(**'message'**,** **function(**topic**,** payload**)** **{**

 // logowanie treści otrzymanej wiadomości

 console**.**log**(**'iot: message'**,** topic**,** payload**.**toString**());**

 // przesłanie do chmury informacji powiadamiającej administratora o otrzymanej wiadomości

 sendToServer**(**'Received: ' **+** payload**.**toString**());**

 // zdekodowanie treści otrzymaje wiadomości (wykorzystywany jest format JSON)

 **var** req **=** JSON**.**parse**(**payload**.**toString**());**

 // pobranie treści atrybutu określającego typ nadchodzącej wiadomości

 **switch** **(**\_**.**get**(**req**,** 'type'**,** null**))** **{**

 // w przypadku gdy wiadomość dotyczy funkcji telefonu:

 **case** 'phone'**:** **{**

 // pobranie treści polecenia związanego z telefonem

 **switch** **(**\_**.**get**(**req**,** 'cmd'**,** null**))** **{**

 // operacje w przypadku gdy otrzymano polecenie wybrania numeru

 **case** 'dial'**:** **{**

 // poobranie numeru telefonu do wybrania

 **var** to **=** \_**.**get**(**req**,** 'to'**,** null**);**

 **if** **(!**\_**.**isNull**(**to**))** **{**

 // polecenie modemowi GSM (za pomocą komendy AT) nawiązania połączenia telefonicznego z numerem

 executeAtCmd**(**'ATD' **+** to **+** ';'**);**

 **}**

 **break;**

 **}**

 // operacje wykonywane, gdy otrzymano polecenie rozłączenia rozmowy

 **case** 'hangup'**:** **{**

 // polecenie modemowi GSM, aby rozłączył rozmowę

 executeAtCmd**(**'ATH'**);**

 **break;**

 **}**

 // operacje wykonywane, gdy otrzymano polecenie sprawdzenia jakości sygnału komórkowego

 **case** 'signal'**:** **{**

 // wysłanie modemowi GSM komendy AT, która zwraca informację o poziomie sygnału

 executeAtCmd**(**'AT+CSQ'**);**

 **break;**

 **}**

 // operacje wykonywane, gdy otrzymano polecenie sprawdzenia poziomu naładowania akumulatora

 **case** 'battery'**:** **{**

 // wysłąnie modemowi GSM komendy AT, która zwraca informację o poziomie nałądowani akumulatora

 executeAtCmd**(**'AT+CBC'**);**

 **break;**

 **}**

 // operacje wykonywane, gdy otrzymano polecenie wysłania wiadomości SMS

 **case** 'sms'**:** **{**

 // pobranie numeru telefonu, pod który ma być wysłana wiadomość SMS

 **var** to **=** \_**.**get**(**req**,** 'to'**,** null**);**

 // pobranie treści wiadomości SMS do wysłania

 **var** message **=** \_**.**get**(**req**,** 'message'**,** null**);**

 **if** **(!**\_**.**isNull**(**to**)** **&&** **!**\_**.**isNull**(**message**))** **{**

 // wywołanie funkcji wysyłającej SMS o zadanej treści, pod zadany numer

 sendSms**(**to**,** message**)**

 **}**

 **break;**

 **}**

 **}**

 **break;**

 **}**

 // w przypadku gdy wiadomość dotyczy funkcji związanej z GPSem

 **case** 'gps'**:** **{**

 // sprawdzenie treści otrzymanej komendy dla GPSu

 **switch(**\_**.**get**(**req**,** 'cmd'**,** null**))** **{**

 // operacje w przypadku otrzymania komendy włączającej odbiornik GPS

 **case** 'on'**:** **{**

 // wysłanie do modemu komendy AT włączającej GPS

 executeAtCmd**(**'AT+CGNSPWR=1'**);**

 **break;**

 **}**

 // operacje w przypadku otrzymania komendy wyłączenia odbiornika GPS

 **case** 'off'**:** **{**

 // wysłanie do modemu komendy AT wyłączającej GPS

 executeAtCmd**(**'AT+CGNSPWR=0'**);**

 **break;**

 **}**

 // operacje w przypadku otrzymania komendy sprawdzenia czy odbiornik GPS jest włączony

 **case** 'power'**:** **{**

 // wysłanie do modemu komendy AT pobierającej stan (włączenia/wyłączenia) odbiornika GPS

 executeAtCmd**(**'AT+CGNSPWR?'**);**

 **break;**

 **}**

 // operacje w przypadku otrzymania komendy sprawdzenia informacji stanu z GPSU

 **case** 'info'**:** **{**

 // wysłanie do modemu komendy AT pobierającej koordynaty i inne parametry z odbiornika GPS

 executeAtCmd**(**'AT+CGNSINF'**);**

 **break;**

 **}**

 **}**

 **break;**

 **}**

 **}**

**});**