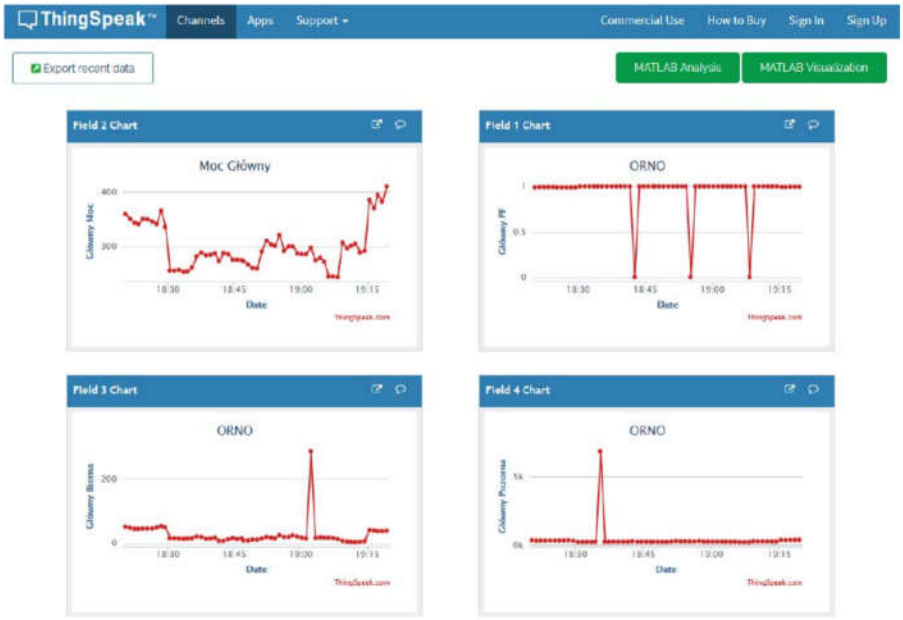
Interfejs do liczników ORNO

Rejestrator danych, link USB-RS485 poprzez Wi-Fi

System automatyki domowej nie kończy się na sterowaniu oświetleniem czy roletami w oknach. Równie ważne jest monitorowanie i rejestracja parametrów, takich jak temperatura, ciśnienie czy pobór energii. Dzięki temu można bardziej optymalnie zarządzać elementami systemu i np. określić, które urządzenia warto całkowicie odłączać od zasilania, w czasie gdy z nich nie korzystamy.



Ważne!
Z powodu znacznej objętości listingów, część z nich publikujemy na stronie z artykułem na http://bit.ly/3nrkwIp, a także w materiałach dodatkowych na media.avt.pl

W artykule zaprezentowano interfejs przeznaczony do liczników ORNO-WE504/514 (**fotografia 1**), ale po niewielkich modyfikacjach zadziała z praktycznie każdym licznikiem wyposażonym w port RS485. Funkcja linku USB (VCOM) – RS485 poprzez Wi-Fi pozwoli skorzystać z oprogramowania liczników dostarczonego przez producenta i łatwo kontrolować zużycie energii elektrycznej.

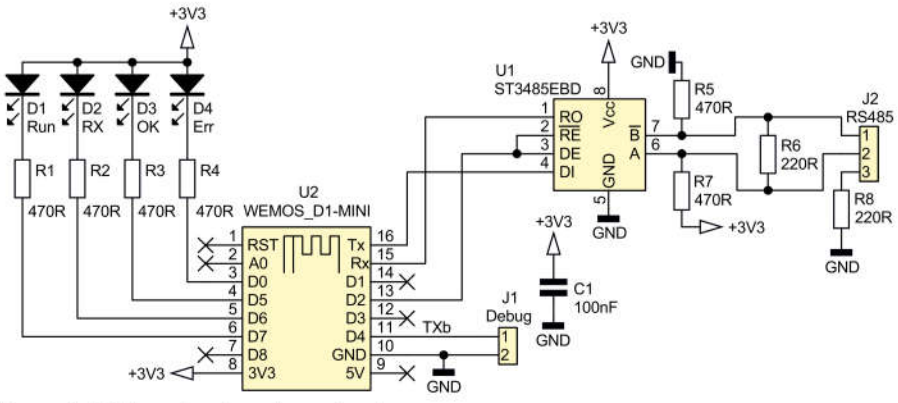
Fotografia 1. Licznik energii elektrycznej ORNO WE514

Budowa i zasada działania

Urządzenie składa się z modułu ESP8266 zamontowanego na płytce Wemos D1 Mini. Do płytki dołączono interfejs UART-RS485 na bazie układu MAX3485. Schemat połączeń został pokazany na **rysunku 1**. W interfejsie można zastosować układ MAX485, zasilając go z 5 V oraz dodając rezystor 470 Ω...1 kΩ na wyjściu RO (pin 1). Całość należy uzupełnić odpowiednim zasilaczem. Prototyp został zmontowany na pająka i pierwotnie była to płytka WemosD1, a dopiero później, tańsza, WemosD1-Mini (**fotografia 2**).

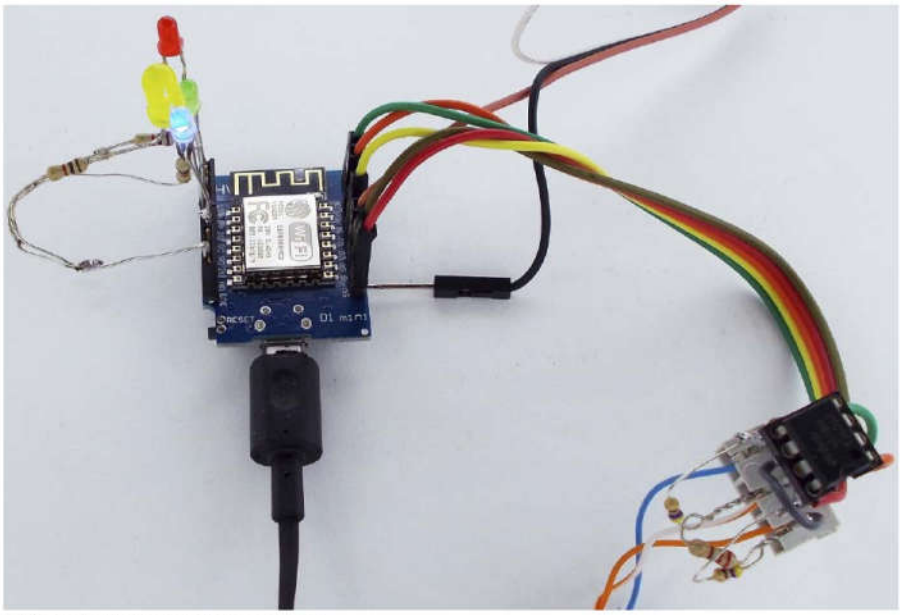
Pracując w trybie mostka USB-RS485, za pośrednictwem programu USR-VCOM urządzenie przesyła dane z wirtualnego portu COM na port 8888 wybranego adresu IP. Moduł ESP odbiera te dane i przesyła na RS-485. Kod programu, odpowiedzialny za to zadanie, został pokazany na **listingu 1** i znajduje się w pętli głównej programu.

Ze względu na to, że API *Arduinolibs* nie przewiduje możliwości stwierdzenia faktu wysłania znaku z bufora nadawczego, wprowadzono opóźnienie ustawiane definicją *#define TIM_SEND_BYTE_UART*, dzięki czemu nie było potrzeby modyfikowania kodu biblioteki, a kierunek transmisji zmieniany jest po wysłaniu znaku, jak pokazuje oscylogram z **rysunku 2** (żółty oscylogram pokazuje stan linii RE/DE).



Rysunek 1. Schemat połączeń urządzenia

Funkcja przesyłająca dane z UART przez Wi-Fi ma bardziej skomplikowane działanie choć jest krótsza – **listing 2**. Znaki otrzymywane przez UART są zapamiętywane w programowym FIFO, które dla ESP8266 ma rozmiar 128 znaków i taka jest maksymalna wielkość ramki, jaką można przesłać. Gdy przerwa pomiędzy znakami przekroczy 5 ms, zapamiętane znaki zostaną wysłane. W rzeczywistości funkcje są bardziej rozbudowane i zawierają wysyłanie informacji debugujących na UART1. Cały kod źródłowy znajduje w materiałach dodatkowych do projektu.



Fotografia 2. Prototyp urządzenia

Rysunek 2. Kierunek transmisji RS485 jest zmieniany po wysłaniu znaku, co pokazuje oscylogram

Listing 1. Kod programu odpowiedzialny za odbieranie danych i przesłanie przez RS485

```
if (localClient && localClient.connected() ) {
  if (localClient.available()) {
    size_t len = localClient.available();
    uint8_t sbuf[len];
    localClient.readBytes(sbuf, len);
    wyslij_na_RS485(sbuf, len);
  }
}

void wyslij_na_RS485( uint8_t *sbuf, size_t len) {
  digitalWrite(DIR485, HIGH);
  Serial.write(sbuf, len);
  // Get the number of bytes (characters) available for writing
  //in the serial buffer without blocking the write operation.
  while ( Serial.availableForWrite() != lenBufSerTx ) ;
  // Konieczne, bo używając "Serial.availableForWrite()"
  //stwierdzimy kiedy bufor jest pusty a nie kiedy znak wysłano z UART
  delayMicroseconds( TIM_SEND_BYTE_UART );
  digitalWrite(DIR485, LOW);
}
```

Przy pracy w trybie odpytywania licznika cyklicznie wywoływana funkcja wysyła zapytanie do licznika. Ma inną postać dla urządzeń WE504 i WE514 – **listing 3**.

Podczas prób z licznikami ujawniła się pewna wada modeli WE514/517. Bez względu na liczbę zapytanych rejestrów licznik odpowie tylko jednym. Odpytanie wszystkich rejestrów związanych z pomiarem mocy, napięcia itp. dla licznika 3-fazowego, przy maksymalnej dopuszczalnej prędkości komunikacji, zajmuje ok. 900 ms. Czas transmisji jest stosunkowo długi w porównaniu do liczby przesłanych danych – od wysłania zapytania do rozpoczęcia transmisji odpowiedzi mija 40...50 ms. W przypadku WE504 odpowiedzi są odsyłane w losowym czasie 28...100 ms. Przy prędkości transmisji 9600 b/s (czas bajtu ok. 1 ms) trwa to długo.

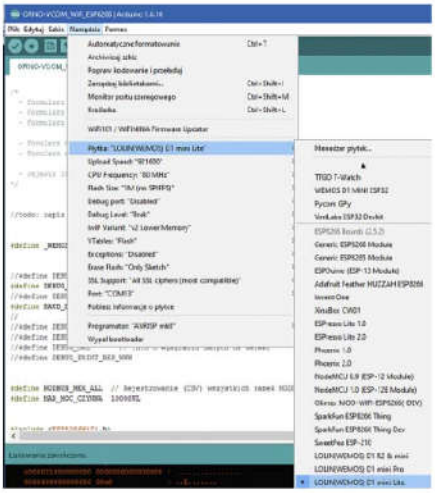
Po małej dygresji na temat funkcjonowania oprogramowania licznika, wróćmy do zasady działania interfejsu. Odpowiedzi napływające z licznika są analizowane w funkcji *Uart_do_Wifi()*, której kod jest zawarty na **listingu 4** (ze względu na objętość dostępny w materiałach dodatkowych). Po krótkiej analizie widać wiele funkcji debugujących. Były potrzebne, ponieważ czasem rejestrowana moc znacznie odbiegała od rzeczywiście pobieranej. Z istotnych funkcji warto napisać o *void obsluga_thingspeak()* pokazanej na **listingu 5**, której zadaniem jest wysyłanie danych na serwer ThingSpeak. Niestety, operacja ta blokuje działanie programu na ponad 5 sekund. W tym czasie nie będzie działać link VCOM. Ze względu na ten fakt, lepiej skorzystać z wysyłania danych na serwer WWW w sieci lokalnej czy w Internecie. Kod funkcji umożliwiającej to zadanie pokazuje **listing 6** (ze względu na objętość dostępny w materiałach dodatkowych). Wysłanie danych poprzez URL może być analizowane skryptem PHP pokazanym na **listingu 7**. Dane są zapisywane w formacie CSV, zajmuje to 50...80 ms

Listing 2. Funkcja przesyłająca dane z UART przez Wi-Fi

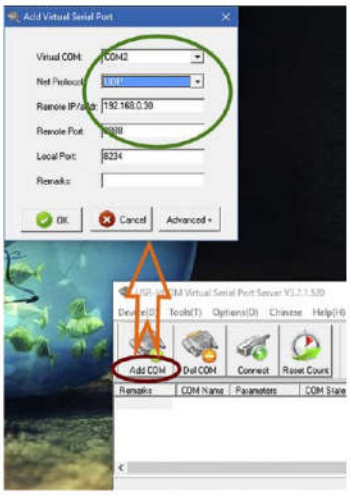
```
void Uart_do_Wifi() {
  uint32_t static timeoutWr, timCzasRd;
  size_t static prevLen;
  // jesli zmienil sie rozmiar bufora
  if ( prevLen != Serial.available()) {
    prevLen = Serial.available();
    if ( prevLen ) { // i jest rozny od 0
      // Przychodzacy znak ustawia timeout
      timeoutWr = millis() + 5;
      if ( ! timCzasRd ) timCzasRd = millis();
    }
  }

  // Gdy timeout minie
  if ( millis() >= timeoutWr ) {
    // Zatrzymujemy liczenie
    timeoutWr = 0xFF000000;

    size_t len = Serial.available();
    uint8_t sbuf[len];
    Serial.readBytes(sbuf, len);

    localClient.write(sbuf, len);
  }
}
```

Listing 3. Funkcja wysyłająca zapytanie do licznika

```
void Zapytania_Do_Orno() {
  // INFO: 514 BEZ WZGLĘDU NA TO O ILE REJESTRÓW PYTAMY ODPOWIE JEDNYM
  const byte zapytanie_514[][8] = {
    { 0x00, 0x03, 0x01, 0x31, 0x00, 0x01, 0xD5, 0xE8 }, // Pytanie o napiecie
    { 0x00, 0x03, 0x01, 0x39, 0x00, 0x02, 0x14, 0x2B }, // Pytanie o prad
    { 0x00, 0x03, 0x01, 0x40, 0x00, 0x02, 0xC5, 0xF2 }, // Pytanie o moc czynna
    { 0x00, 0x03, 0x01, 0x48, 0x00, 0x02, 0x44, 0x30 }, // Pytanie o moc bierna
    { 0x00, 0x03, 0x01, 0x58, 0x00, 0x01, 0x05, 0xF4 }, // Pytanie o PF
    { 0x00, 0x03, 0xA0, 0x00, 0x00, 0x0A, 0xE6, 0x1C }, // Pytanie o zuzyta energie czynna
    { 0x00, 0x03, 0xA0, 0x1E, 0x00, 0x0A, 0x86, 0x1A }, // Pytanie o zuzyta energie bierna
    { 0x00, 0x03, 0x01, 0x50, 0x00, 0x02, 0xC4, 0x37 }, // Pytanie o moc pozorna
    { 0x00, 0x03, 0x01, 0x30, 0x00, 0x01, 0x84, 0x28 }, // Pytanie o czestotliwosc
    { 0xFF }
  };

  // INFO: 504 MOŻNA PYTAĆ O DOWOLNĄ LICZBĘ REJESTRÓW
  const byte zapytanie_504[][8] = {
    {
      0x00, 0x03, //00 03   ID, Function READ
      0x00, 0x00, //00 00   adr rej H, L
      0x00, 0x0a, //00 10   liczba rej do odczytu H, L
      0x45, 0xd7  //45 d7   CRC L, H
    },
    { 0xFF }
  };


  if ( millis() >= timeoutOdpytanie ) {
    timeoutOdpytanie = millis() + CO_ILE_PYTAC_ORNO;
    timOdpowiedziOrno = millis();

    ++nrPytaniaOrno;
    if ( (zapytanie_514[nrPytaniaOrno][0] != 0xFF && typOrno == 514) ||
         (zapytanie_504[nrPytaniaOrno][0] != 0xFF && typOrno == 504) ) {

      uint8_t buf[8];
      if ( typOrno == 514 ) {
        memcpy( buf, (uint8_t*)zapytanie_514[nrPytaniaOrno], 6 );
      }
      else {
        memcpy( buf, (uint8_t*)zapytanie_504[nrPytaniaOrno], 6 );
      }

      uint16_t crc = CRC16( buf, 6 );
      buf[6] = crc;
      buf[7] = crc >> 8;
      wyslij_na_RS485( buf, 8 );
    }
    else {
      nrPytaniaOrno = -1;
      timeoutOdpytanie = millis() + PAZUZA_ZAPYTAN_ORNO;

        wyslij_na_RS485( dane, 17 );
      }
    }
  }
}
```



Rysunek 3. Wybór płytki, na którą kompilujemy program



Rysunek 4. Konfiguracja programu USR-VCOM

w przypadku sieci lokalnej, 150..180 ms w Internecie. Skrypt zapisuje dane z różnych liczników pod osobnymi nazwami, ponadto dzieli na okresy, dni i miesiące.

Jakkolwiek użycie własnych skryptów jest lepsze, bo nie wnosi ograniczeń co do liczby zapisanych danych (co ma miejsce na ThingSpeak nawet w wersji płatnej). Ma to też wadę ujawniającą się, gdy serwer jest nieosiągalny. W takiej sytuacji program jest zablokowany przez timeout trwający 5 sekund. Skutecznym rozwiązaniem jest skorzystanie z UDP. Testową funkcję można znaleźć pod nazwą obsluga_udp().

Uruchomienie

Zanim program zostanie wgrany do płytki, należy zmodyfikować kody źródłowe. W pierwszej kolejności należy skonfigurować sieć bezprzewodową. W tym celu odnajdujemy w kodzie definicje:

```
#define WIFI_SSID    "ssid"
#define WIFI_PASS    "pass"
```

i wpisujemy dane własnej sieci.

Jeżeli chcemy nadać statyczny adres IP komentujemy linie:

```
//#define DHCP
```

oraz wpisujemy dane własnej sieci:

```
IPAddress staticIP(192, 168, 0, 101);
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 255, 0);
```

Należy jeszcze wybrać płytkę, której używamy:

```
#define _WEMOS          1    // 0-D1, 1-MINI
```

Jeśli będziemy korzystać z Thingspeak, wpisujemy dane uzyskane w procesie rejestracji:

```
const char * thingspeak_apiKey = "apikey";
```

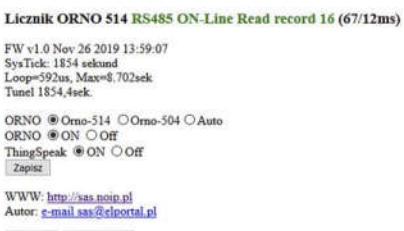
Odnajdując w kodzie frazę *if (ip[3] == 52)*, można uzależnić linki od adresu płytki. Fragmenty do modyfikacji wyróżniono kolorem:

```
  if ( ip[3] == 52 ) {
    strcat( buf, "<A HREF='https://thingspeak.com/channels/xxxxxxxx' TARGET='_blank' >Rejestrator</A>\r" );
    if ( ! ee.fl_ThingOn ) strcat( buf, "<font color='RED'>(Off)</font>" );
    strcat( buf, "</TD><TD><input type='button' value='Reload Page' onClick='document.location.reload(true)'>\r" );
    strcat( buf, "</TD></tr></TABLE>\r" );
    strcat( buf, "<P><iframe width='450' height='260' style='border: 1px solid #cccccc;' src='https://thingspeak.com/channels/xxxxxxxx/charts/2?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&title=Moc+G%C5%82%C3%B3wny&type=line'>< / iframe >" );
  }
```

Gdy dane będą zapisywane na własnym serwerze, należy wypełnić:

```
#define DATA_HOST_LOCAL_1 "192.168.0.120"      // IP lokalnego serwera
#define DATA_HOST_LOCAL_2 "192.168.2.121"      //IP lokalnego serwera
#define DATA_HOST_REMOTE_1 "domena.pl"     //IP serwera, może byc nazwa domenowa
#define DATA_HOST_REMOTE_2 "domena2.pl"     //IP serwera, może byc nazwa domenowa
```

Rysunek 5. Strona www, na której można zobaczyć aktualne parametry monitorowane przez licznik



Rysunek 6. Konfiguracja parametrów licznika przez stronę www

Jeśli serwerów jest mniej, zamiast adresu IP czy nazwy należy wpisać "".

Definicja:

```
#define LOGGER_URL "/sciezka/nazwa-skryptu.php?"
```

określa ścieżkę dostępu do skryptu PHP, który zapisuje dane. W przypadku prób z UDP należy uzupełnić:

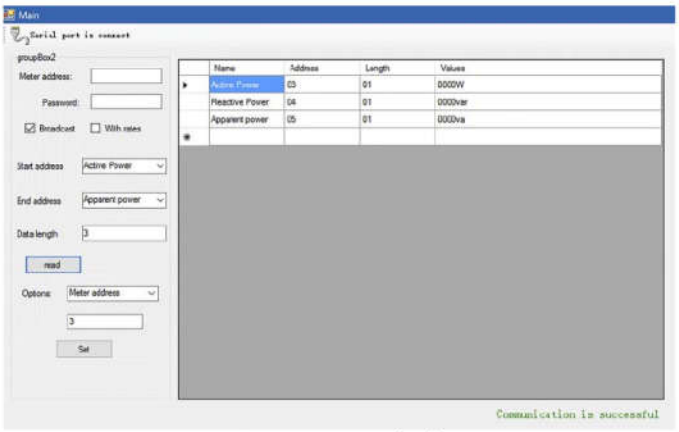
```
#define BROADCAST "192.168.0.255"
unsigned int localUdpPort = 2102;
```

Dla ułatwienia wszystkie miejsca w kodzie, które należy zmodyfikować, są odpowiednio oznaczone.

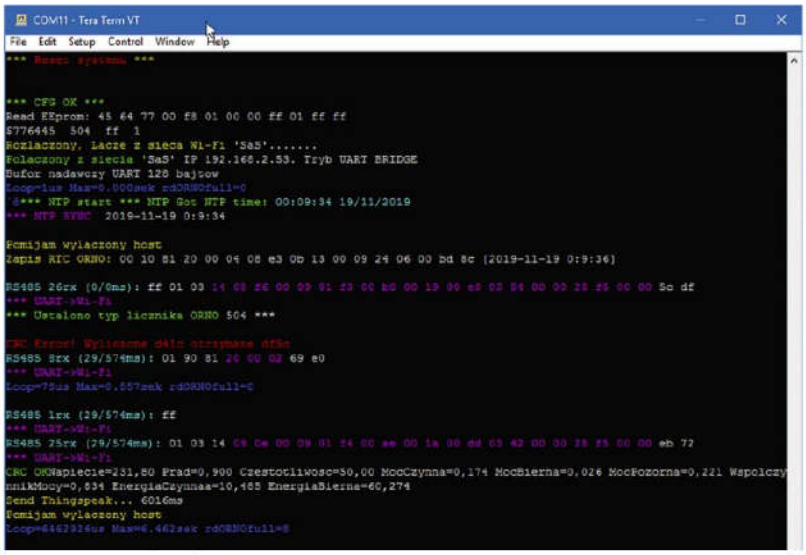
Po wprowadzeniu zmian należy wybrać płytkę, na którą kompilujemy program, jak pokazano na **rysunku 3**, oraz port COM, po czym należy skompilować i wgrać program na płytkę. W ustawieniach routera warto zadbać o to, aby płytka dostawała zawsze ten sam adres IP ewentualnie skonfigurować adres statyczny.

Jeśli zamierzamy korzystać z VCOM, należy zainstalować program USR-VCOM dostępny na stronie http://bit.ly/3on789v. Po zainstalowaniu i uruchomieniu należy nacisnąć przycisk ADD (**rysunek 4**). W oknie, które się pojawi, należy wybrać numer wirtualnego COM, pod jakim ma być dostępny link. Protokół komunikacyjny należy zmienić z domyślnego TCP na UDP. Pozostaje wpisać adres IP interfejsu oraz ustawić port na 8888.

Pod adresem IP interfejsu dostępna jest strona WWW. Można na niej zobaczyć aktualne parametry monitorowane przez licznik ORNO (**rysunek 5**). Pod linkiem FW dostępne



Rysunek 7. Program dostarczony przez producenta licznika



Rysunek 8. Informacje diagnostyczne wysyłane na port szeregowy UART1

są informacje o wersji programu, dacie konfiguracji oraz jest możliwość włączenia bądź wyłączenia rejestracji na serwerze Thingspeak oraz wyboru typu licznika (**rysunek 6**). Link „Rejestrator" kieruje na stronę Thingspeak, na której są reprezentowane dane wysyłane przez interfejs.

Uwagi końcowe

Płytkę WemosMini wraz z driverem RS485 można zmieścić w obudowie Z-105, natomiast razem z zasilaczem IRM01-5 w Z-106. Przy wyborze modelu licznika należy wziąć pod uwagę fakt, że model WE504 przy poborze mocy przez odbiornik poniżej 12 W zachowuje się niestabilnie. Raz pokazuje 0, innym razem rzeczywiście pobieraną moc. W tym samym czasie ORNO-514 poprawnie pokazuje pobór mocy nawet o wartości 2...3 W. Nie są to błędy w komunikacji, co potwierdzono analizatorem logicznym, jak i przy pomocy programu dostarczonego przez producenta (**rysunek 7**). Jeżeli więc licznik ma mierzyć małe prądy, trzeba zrezygnować z WE504 na rzecz WE514.

W ramkach danych MODBUS dla ORNO brak konsekwencji przy przesyłaniu danych 16 i 32 bit. Same dane 16-bitowe, wymagane przez MODBUS, wysyłane są poprawnie – najpierw młodszy, później starszy bajt, natomiast w polu danych najstarszy bajt wysyłany jest jako pierwszy. Trzeba o tym pamiętać, bo łatwo o pomyłkę podczas analizy ramki danych.

W materiałach dodatkowych można znaleźć poza kodami źródłowymi programu, logi zarejestrowane analizatorem LA2016. Logi można obejrzeć darmowym programem dostępnym na stronie producenta http://bit.ly/3hTwHNk.

Na port szeregowy UART1 wysyłane są informacje diagnostyczne (**rysunek 8**). O tym, czy, jakie i z jaką prędkością, decydują definicje:

```
#define DEBUG_UART1
#define BAUD_DEBUG 921600

#define DEBUG_MODBUS
#define DEBUG_RTC
#define DEBUG_URL
#define DEBUG_PRINT_REP_WWW
```

Dokumentacja zawiera kilka błędów, aby uwolnić Czytelników od błądzenia, w materiałach dodatkowych zamieściłem dodatkowy plik z moimi spostrzeżeniami i uwagami.

SaS, AVT
sas@elportal.pl