

Andrzej Chmielowiec
14 sierpnia 2009

uCtools – Protokół komunikacyjny pomiędzy klientem, a serwerem

1 Wprowadzenie

Zadaniem aplikacji serwera uCtools jest umożliwienie komunikacji sieciowej pomiędzy mikrokontrolerami, a aplikacjami pakietu uCtools. Serwer może być niezależnym programem, do którego podłączane są urządzenia peryferyjne lub też może być wbudowany bezpośrednio w mikrokontroler.

W pierwszym przypadku może być traktowany jako serwer pośredniczący pomiędzy aplikacjami, a mikrokontrolerem. Jego zastosowanie pozwala na jednoczesne podłączenie do jednego urządzenia kilku programów lub zdalną pracę z fizycznie oddalonym układem.

Serwer wbudowany w urządzenie można natomiast traktować jako interfejs, który umożliwia komunikację z aplikacjami uCtools za pośrednictwem sieci TCP/IP.

2 Protokół komunikacyjny

Gdy klient łączy się z serwerem ten najpierw wysyła informację o swojej konfiguracji i parametrach. Jeżeli zostanie uzgodniona dalsza forma komunikacji, to klient może pobrać listę urządzeń, które są udostępniane przez serwer oraz nawiązać z nimi łączność.

W dalszej części będą używane następujące oznaczenia:

K – dane wysyłane przez klienta.

S – dane wysyłane przez serwer.

SEP – separator, którym jest znak 0x00.

2.1 Pobieranie parametrów serwera

Podczas pierwszego połączenia klienta z serwerem następuje pobranie parametrów serwera. Całość polega na wykonaniu następujących operacji:

[K] Klient nawiązuje połączenie z serwerem.

[S] Serwer wysyła informację o swojej konfiguracji i konfiguracji protokołu komunikacyjnego. Format informacji jest następujący:

```
Struct ServParams {  
    String type, SEP  
    String protocols, SEP
```

```
String authentication, SEP
}
```

[K] Klient odbiera parametry serwera i kończy sesji TCP.

Opis pól struktury ServParams:

1. Pole `type` zawiera napis `SERVPARAMS`.
2. Pole `protocols` zawiera nazwy lub numery protokołów, które są obsługiwane przez serwer. Jeżeli serwer obsługuje więcej niż jeden protokół, to pole są one połączone znakiem `+`. Na przykład: `BASIC+ENCRYPT`.
3. Pole `authentication` określa sposób weryfikacji klienta, który próbuje nawiązać komunikację z serwerem. Aktualnie obsługiwane są następujące typy weryfikacji:
 - (a) `NONE` – brak weryfikacji.
 - (b) `PASS` – wymagane jest hasło do serwera.

2.2 BASIC - prosty protokół komunikacyjny

Jest to najprostsza forma komunikacji pomiędzy klientem, a serwerem. Wszystkie informacje przesyłane są kanałem jawnym, a jedyną opcją ograniczenia dostępu do serwera jest ustawienie hasła. Protokół ten przeznaczony jest do komunikacji w sieci lokalnej lub zabezpieczonej sieci VPN (w przypadku zdalnych lokalizacji). Protokół ten może być wykorzystywany jedynie w przypadku, gdy serwer uwierzytelnia klientów metodami `NONE` lub `PASS`.

Pobieranie listy urządzeń podłączonych do serwera

Po uzgodnieniu protokołu komunikacyjnego klient przystępuje do pobrania listy urządzeń obsługiwanych przez serwer. Pobranie listy polega na wykonaniu następujących operacji:

[K] Klient nawiązuje połączenie z serwerem.

[K] Klient wysyła żądanie pobrania listy dostępnych urządzeń (format listy opisany jest w osobnej części dokumentacji):

```
String "DEVICELIST", SEP
String user, SEP [required if PASS mode]
String pass, SEP [required if PASS mode]
```

[S] Serwer wysyła listę dostępnych urządzeń:

```
String "DEVICELIST", SEP
String deviceListXml, SEP
```

[K] Klient odbiera listę urządzeń i kończy sesję TCP.

Podłączenie do urządzenia

Po pobraniu listy dostępnych urządzeń można nawiązać połączenie z wybranym układem i rozpocząć transmisję.

[K] Klient nawiązuje połączenie z serwerem.

[K] Klient wysyła żądanie połączenia z urządzeniem:

```
String "CONNECT", SEP
String id, SEP
String user, SEP [required if PASS mode]
String pass, SEP [required if PASS mode]
```

[S] Serwer wysyła informację o gotowości:

```
String "READY", SEP
```

[K] Klient odbiera informację o gotowości i utrzymuje sesję TCP.

Transmisja danych

Wysyłanie danych odbywa się na zasadzie poleceń i oczekiwania na odpowiedź. Polecenia mogą być inicjowane tylko przez klienta, który przesyła dane przeznaczone dla urządzenia. Następnie serwer czeka na odpowiedź i przesyła ją klientowi.

[K] Klient wysyła dane do urządzenia:

```
String "DATA", SEP
ByteString data
```

[S] Serwer wysyła dane otrzymane od urządzenia:

```
String "DATA", SEP
ByteString data
```

Pole data może zawierać dowolny ciąg bajtów i jest rozpoznawane jako bajty występujące po pierwszym separatorze.

Obsługa błędów

Sygnalizacja błędu po stronie serwera może nastąpić tylko na tym etapie protokołu, podczas którego serwer wysyła informację klientowi. Ramka błędu serwera ma następującą postać:

[S] Serwer wysyła informację o błędzie:

```
String "!ERROR", SEP
String errorDescription, SEP [OPTIONAL]
```

Zakończenie transmisji

Zakończenie transmisji polega na zakończeniu sesji TCP po stronie klienta lub serwera.

3 Format XML listy urządzeń podłączonych do serwera

Poniżej znajduje się przykład struktury XML, która reprezentuje listę urządzeń obsługiwanych przez serwer. Lista ta zawiera jedynie informację o logicznej strukturze serwera, która może być zupełnie niezależna od struktury fizycznej. Na przykład jedno fizyczne urządzenie można reprezentować jako dwa logiczne, itp.

```
<UCSERVER name="server name" id="0">
  <UCDEV name="microcontroller name" id="0" />
  <UCBOARD name="board name" id="0">
    <UCDEV name="microcontroller name" id="1" />
  </UCBOARD>
  <UCBOARD name="board name" id="1">
    <UCDEV name="microcontroller name" id="2" />
  </UCBOARD>
</UCSERVER>
```

3.1 Opis struktury XML

<UCSERVER>

name – nazwa serwera.

id – identyfikator serwera, który musi być równy 0.

<UCDEV>

name – nazwa urządzenia.

id – identyfikator urządzenia, który musi być liczbą unikalną (inną dla każdego urządzenia). Wykorzystywany jest przez serwer do utrzymywania komunikacji pomiędzy klientem, a urządzeniem. Kolejność nadawania identyfikatorów nie jest istotna.

<UCBOARD>

name – nazwa karty (grupy urządzeń).

id – identyfikator karty, który musi być liczbą unikalną (inną dla każdej karty).