W poprzednich artykułach naszego cyklu o Raspberry Pi (RPi) rozpatrywaliśmy konfiguracje, w których RPi był dostępny zdalnie w sieci domowej. Tym razem opiszę wariant łączenia się do niego bezpośrednio za pomocą interfejsu szeregowego UART.

Raspberry Pi: UART

Raspberry Pi (RPi) używany jest najczęściej w trybie zdalnego dostępu. RPi wpinamy do naszej sieci domowej bezpośrednio kablem Ethernet do rutera lub bezprzewodowo przez kartę WiFi umieszczoną w gnieździe USB. W takiej konfiguracji łączymy się z RPi za pomocą klienta SSH (np. Putty) z innego komputera znajdującego się w tej samej sieci.

Może się jednak zdarzyć, że RPi nie jest podłączone do sieci albo – w obliczu problemów ze startem – Linuks nie uruchomi usług zdalnego dostępu. Nie zawsze mamy pod ręką klawiaturę USB i dodatkowy monitor HDMI. W takiej sytuacji musimy znaleźć sposób, żeby bezpośrednio podłączyć się komputerem do RPi. Można to osiągnąć przez złącze UART. W takiej konfiguracji wykorzystujemy komunikację poprzez połączenie szeregowe. Niesie to ze sobą kilka szczególnych wyzwań:

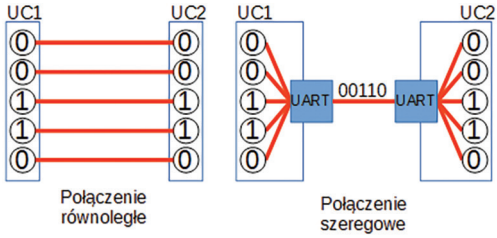
- szeregowy port UART, w który Raspberry Pi jest wyposażony, działa z poziomami napięć logiki charakterystycznymi dla niego, czyli 0-3,3V;
- obecnie niewiele komputerów jest wyposażonych w port szeregowy; a jeżeli już – to działają one zgodnie ze standardem RS232 operującym zakresem napięć zdecydowanie niebezpiecznych dla RPi;
- wszystkie nowoczesne komputery wyposażone są w złącze USB;
- złącze microUSB służy RPi tylko do zasilania (piny danych nie są podłączone); w trybie klienta gniazda USB można użyć najwyżej do skonfigurowania RPi jako urządzenie masowe.

Poniższy tekst opisuje rozwiązania tych problemów. Połączymy Raspberry Pi zarówno z komputerami wyposażonymi w port szeregowy, jak i tylko USB.

UART

UART (ang. *Universal Asynchronous Receiver/Transmitter)* oznacza protokół, który potrafi komunikować systemy mikroprocesorowe, posługując się asynchroniczną transmisją szeregową (ang. *serial*). Dane przesyła się po kolei, jeden bit po drugim. Dla odróżnienia, w przypadku metod równoległych (ang. *parallel*), dane transmitowane są jednocześnie poprzez wiele linii połączeniowych (**1**).

Asynchroniczność odnosi się tu do braku zegara, który kontroluje rytm wszystkich komunikujących się urządzeń. Dla przykładu, typowym połączeniem synchronicznym jest I²C. Każdy zapis i odczyt



1. Idea połączenia równoległego i szeregowego

danych odbywa się zgodnie z taktem zegara. W obliczu braku takiego źródła odniesienia, dane wysyłane asynchronicznie muszą być odpowiednio „opakowane" przez nadawcę. Dzięki temu odbiorca wiadomości zorientuje się, gdzie jest ich początek i koniec. Dodatkowo należy znaleźć sposób na sprawdzanie poprawności transmisji danych. W związku z tym możecie natknąć się na następujące pojęcia:

- bit startu (ang. *start bit*);
- bit stopu (ang. *stop bit*);
- bit parzystości (ang. *parity bit*);
- szybkość transmisji (ang. *baud rate*).

Bity startu/stopu to nic innego, jak sygnały oznaczające początek i koniec danych (może ich być kilka). Bit parzystości pozwala na pewną kontrolę poprawności odebranych danych. W zależności od trybu obliczania bit parzystości ustawia się tak, żeby dopełnił ilość jedynek w paczce. Dla trybu sprawdzania nieparzystości (ang. *odd parity*) bit parzystości zostanie ustawiony na „1" dla bitowej sumy danych parzystej lub „0" dla sumy nieparzystej. W ten sposób odbiornik dostanie nieparzystą liczbę jedynek. Dla trybu kontroli parzystości (ang. *even parity*) bit parzystości zostanie ustawiony na „0" dla sumy parzystej, a „1" dla sumy nieparzystej. Nadajnik zapisuje bity parzystości z danymi i wysyła paczkę. Odbiornik odczytuje paczkę i zlicza wszystkie wystąpienia „1" (ignorując bity startu i stopu). Jeżeli ich ilość nie zgadza się z wybranym trybem parzystości, odbiornik zgłasza błąd transmisji.

Kolejnym parametrem jest szybkość przesyłania danych podawana w bitach na sekundę (ang. *bauds*). Przyjmuje się najczęściej wartości: 9600, 56 700, 115 200. Ponieważ w transmisji asynchronicznej nie ma zegara, to właśnie szybkość transmisji służy do synchronizacji nadajnika i odbiornika. Szybkość ta przekłada się bowiem na czas trwania sygnału oznaczającego pojedynczy bit.

2. „Krzaczki” na Putty: problem z parametrami komunikacji

Z powyższego opisu wynika, że nadajnik i odbiornik „dogadają się” jedynie wtedy, gdy posłużą się takimi samymi wartościami parametrów transmisji. Jeżeli choć jeden z nich jest inny – transmisja się nie powiedzie. Rezultat różnie ustawionych parametrów to serie nic nieznaczących artefaktów wyświetlanych na konsoli (**2**). W pierwszej kolejności należy wtedy sprawdzić zgranie szybkości odbioru nadajnika i odbiornika.

W celu uniknięcia podobnych nieporozumień parametry połączenia opisuje się w odpowiednim formacie. Dla przykładu: „115200 8-N-1” oznacza:

- „115200”: szybkość przesyłania danych w bitach na sekundę (baudach);
- „8”: ilość bitów danych w paczce, najczęściej 5-9;
- „N”: brak kontroli parzystości; rzadziej może być np. E (ang. *even*) dla trybu kontroli parzystości lub O (ang. *odd*) dla trybu kontroli nieparzystości;
- „1”: pojedynczy bit stopu.

Powyższe ustawienia są charakterystyczne dla portu UART Raspberry Pi.

RS232

Przykładem protokołu szeregowego wykorzystywanego przez komputery klasy PC jest RS232. RS232 był kiedyś jednym z podstawowych standardów do wymiany danych. Stosowano go do podłączania myszy, modemów czy też samych komputerów. Wraz z upowszechnieniem się USB minęły jego czasy w zastosowaniach domowych (czy biurowych). Trudno w nowoczesnych komputerach czy laptopach znaleźć port RS232. Obecnie jest mało praktyczny i po prostu przestarzały. Stało się tak ze względu na rosnące wymagania co do szybkości transmisji, wzajemnego łączenia urządzeń (RS232 to połączenie 1:1), zasilania w jednym kablu z transmisją danych, a także potrzeby spójnego, znormalizowanego i przekrojowego podejścia (nie skoncentrowanego na warstwie elektrycznej).

3. Port RS232 w komputerze Dell OptiPlex 745

W zastosowaniach profesjonalnych prostota RS232 sprawia jednak, że nadal jest szeroko wykorzystywany. Można go też stosować do połączenia z Raspberry.

Na poziomie logicznym (organizacji paczek danych) RS232 jest zgodny z UART. Problem polega na tym, że RPi posługuje się logiką 0 – 3,3V (TTL). Interfejs RS definiuje całkiem inne poziomy napięć. Zależnie od implementacji:

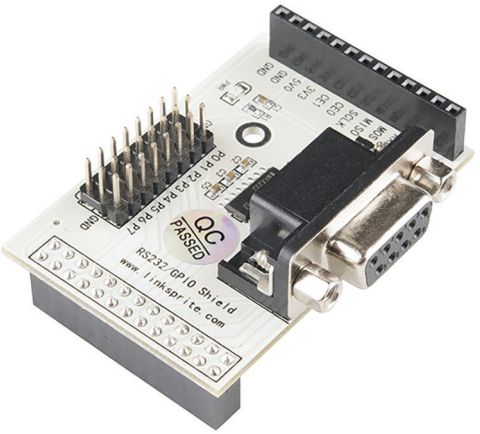
- wartość -15 V to logiczna „1” (ang. *mark*);
- wartość +15 V to logiczne „0” (ang. *space*);
- wartości z przedziału -3..+3 V uważane są za nieustalone.

Różne implementacje RS mogą używać poziomów ±12 V, ±10 V, maksymalnie do ±25 V. Ta różnica napięć między RPi i RS232 sprawia, że bezpośrednie podłączenie pinów TxD/RxD RPi (nadawania i odbioru) do portu RS drugiego urządzenia **doprowadzi do zniszczenia RPi**. Potrzebujemy więc takiego konwertera, który z jednej strony ograniczy napięcie interfejsu RS232 do poziomu RPi (RS232 do TTL; RPi w trybie odbiornika), a z drugiej dostosuje wyjścia z pinów RPi do poziomu transmisji RS (TTL do RS232, RPi w trybie nadajnika). Rolę tę spełniają konwertery UART-do-RS232.

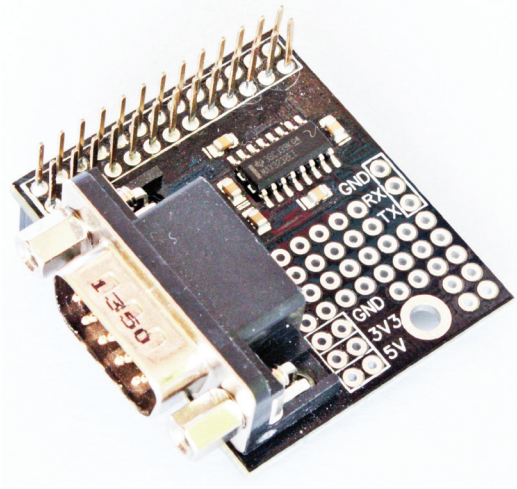
Możliwości są tu dwie:

- moduły rozszerzające, nakładane na złącze GPIO;
- moduły podłączane do GPIO za pomocą taśmy (lub kilku kabli).

Na rynku znajdziecie wiele modułów rozszerzających. nakładanych na złącze GPIO (**4, 5**). Koszt takiego rozwiązania to ok. 60 zł.



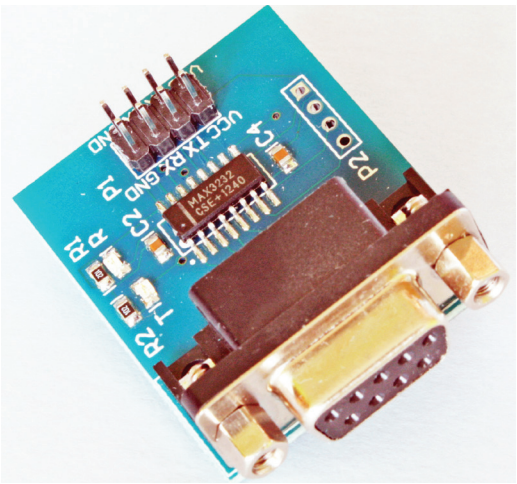
4. Moduł RS232 dla RPi linksprite.com (źródło: linksprite.com)



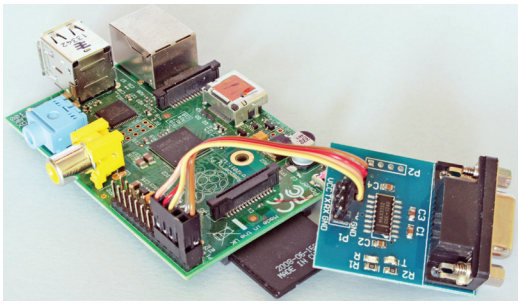
5. Moduł Serial Pi z miejscem na prototypowanie (abelectronics.co.uk)

Użycie podobnego rozszerzenia to sensowne rozwiązania dla przypadków, w których RS232 jest witalną częścią projektu. Dla potrzeb diagnostycznych wystarczy nam jednak możliwość czasowego podłączenia. Użyjemy do tego celu konwerterów, które podłącza się do GPIO za pomocą kilku żeńsko-żeńskich przewodów (**6**).

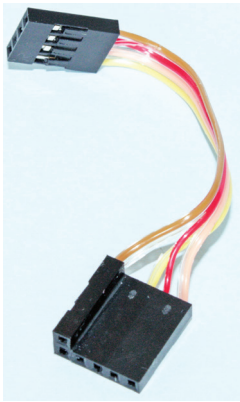
Proszę zwrócić uwagę na wykorzystanie układu MAX3232. W odróżnieniu od MAX232, operuje on na szerszym zakresie napięć zasilających. Działa zarówno na 3,3 V, jak i 5 V. Taki zakres pasuje zarówno do RPi, jak i Arduino. Szukajcie konwerterów opartych na układzie MAX3232 – są najbardziej uniwersalne. Model prezentowany na **zdjęciu 6** kosztował ok. 10 zł. Jak na razie sprawuje się bez problemów. Muszę jednak przyznać, że z dwóch zakupionych egzemplarzy, tylko jeden zadziałał poprawnie.



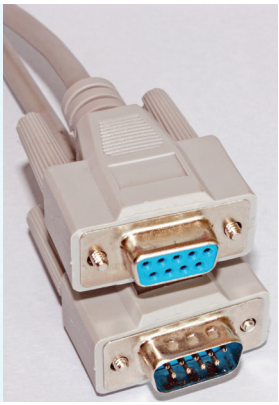
6. Przykładowy konwerter poziomów UART-do-RS232 (lctech-inc.com)



7. Podłączenie konwertera UART-do-RS232 do RPi



8. L-wtyczka do UART



9. Kabel RS232

Podłączenie konwertera UART-do-RS232

RPi oferuje zestaw pinów, które mogą być wykorzystane do komunikacji po UART. Są to:

- TxD (nadawanie): fizyczny pin 8 (GPIO14);
- RxD (odbiór): fizyczny pin 10 (GPIO15);
- Vcc (zasilanie) 3,3 V: fizyczny pin 1 (też 17);
- GND (masa): fizyczny pin 6 (też 9, 14, 20, 25).

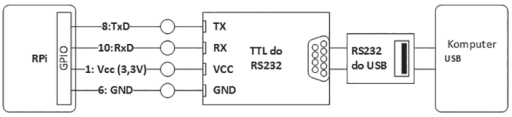
Piny RPi z konwerterem UART-do-RS232 należy połączyć w następujący sposób:

- Pin 8 RPi (TxD) do pinu TX konwertera;
- Pin 10 RPi (RxD) do pinu RX konwertera;
- Pin 1 RPi (3,3V) do pinu VCC konwertera;
- Pin 6 RPI (GND) do pinu GND konwertera.

Proszę zwrócić uwagę, że łączymy piny TxD z TX (nadawanie z nadawaniem) i RxD z RX (odbiór z odbiorem). Przykładowe podłączenie konwertera do RPi znajdziecie na **zdjęciu 7**.

Przeszkodą w podłączaniu może być „rozrzucenie" odpowiednich pinów RPi. Żeby ułatwić podłączanie, skleiłem wtyczkę z 4-pinowego i 2-pinowego gniazda „goldpin" (raster 2,54 mm jak dla GPIO) w kształt litery L (**8**). W ten sposób nie popełnię błędu. Proszę zwrócić uwagę, że konwerter jest zasilany z pinu 3,3 V. Takie też będą poziomy logiki TX/RX. Nie ma zagrożenia uszkodzenia RPi.

Powyższa konfiguracja, uzupełniona o dodatkowy kabel połączeniowy 9-pinowy męsko-żeński RS232 (**9**), wystarczy do połączenia RPi z komputerem



10. Idea łączenia UARTu RPi przez RS232 do portu USB

wyposażonym w port szeregowy. Jeżeli komputer go nie ma, można zaopatrzyć się w odpowiednią kartę rozszerzającą o porty RS232.

RS232 do USB

Zajmijmy się teraz przypadkiem połączenia RPi z komputerem, który nie jest (lub nie może być) wyposażony w złącze RS232 (np. laptop). Ideę postępowania prezentuje **rysunek 10**.

Mamy żeńską wtyczkę RS232 (ang. *DB9 female*) na wyjściu konwertera UART-do-RS232. Potrzebujemy przejściówki RS232-do-USB. Są one dość popularne na rynku. Kupując taką przejściówkę, zwróćcie uwagę na:

- układ konwertujący;
- wsparcie dla USB 1.1/2.0;
- kompatybilność z systemami operacyjnymi;
- rodzaj wtyczki.

RS232 i USB to dwa różne standardy i nie można po prostu „zdrutować" styków. Mimo że przejściówki RS232-do-USB wyglądają najczęściej jak zwykłe kable, kryją w sobie wyspecjalizowane układy konwertujące. Najbardziej popularne pochodzą od firmy Prolific (np. PL-2303 i podobne) oraz FTDI (np. FT232R). Z mojego doświadczenia wynika, że pod Windows najtrudniej jest uruchomić Prolifica. I nie chodzi o jego możliwości. Po prostu na rynku trudno trafić na produkt firmowy (oryginalny). Ceny takich kabli zaczynają się od 5 zł. Niestety, te najtańsze to najczęściej podróbki i próba ich instalacji pod Windows skończy się błędem sterownika numer 10 (lub podobnymi). Spotkałem oczywiście też tanie kable, które chodziły bezproblemowo – kwestia szczęścia.

Proponuję więc szukać kabli opartych na układach FTDI (**11**). Pracowałem z nimi na kilku systemach i na żadnym nie stwarzały problemów. Oczywiście są znacznie droższe (50 zł i więcej).

Przy zakupie przejściówek, warto sprawdzić z jakimi systemami operacyjnymi są kompatybilne.

11. Przejściówka RS232-do-USB firmy FTDI (z lewej) i inna

Mój przygodnie kupiony Prolific nie chciał działać z Win7 ani z Win8. Za to Ubuntu poradził sobie z nim bez problemów. Z reguły najłatwiej uruchomić takie przejściówki właśnie pod Linuksem albo Win7. Najwięcej problemów sprawia Win8, do którego nierzadko brakuje odpowiednich sterowników.

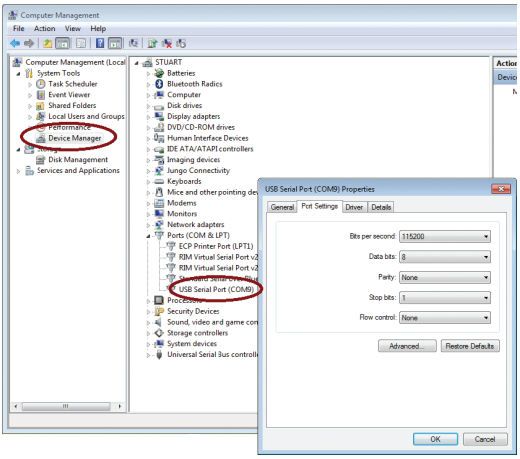
Pomniejszą przeszkodą mogą być same wtyczki. Mój kabel FTDI miał gniazdo ze śrubami mocującymi, podobnie jak konwerter UART-do-RS232. Nie dało się spiąć jednego z drugim bez rozkręcenia którejś z wtyczek. Żeby tego uniknąć, można użyć dodatkowego przedłużacza.

Składamy klocki

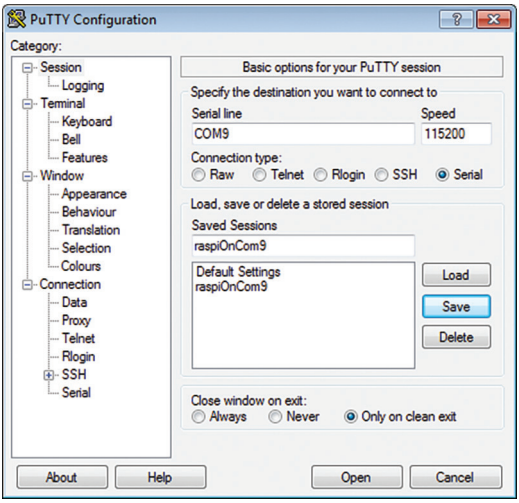
Do uruchomienia komunikacji szeregowej zostało dosłownie kilka kroków:

- podłączenie przejściówki RS232-do-USB do komputera, określenie numeru jej portu szeregowego;
- określenie szybkości transmisji konsoli na RPi;
- uruchomienie i skonfigurowanie Putty;
- wyłączenie RPi;
- podłączenie konwertera UART-do-RS232 do RPi;
- połączenie konwertera poziomów UART-do-RS232 do przejściówki RS232-do-USB;
- uruchomienie RPi;
- ustanowienie połączenia Putty z RPi.

Gdy już zaopatrzymy się w odpowiednią przejściówkę RS232-do-USB, podłączamy ją do naszego komputera. Musimy najpierw określić numer portu komunikacyjnego, który zostanie jej przypisany. Dla systemu Windows: otwórzcie Menedżera Urządzeń (np. dla Win7: „Start", prawy klik na „Komputer", z menu podręcznego „Zarządzaj", „Menedżera urządzeń" znajdziecie na liście po lewej) i rozwińcie „Porty (COM i LPT)", szukając „Port szeregowy USB". Zapamiętajcie numer (np. COM9). Jeżeli chcecie zobaczyć aktualne ustawienia portu, z menu podręcznego wybierzcie „Właściwości" i otwórzcie zakładkę „Ustawienia Portu" (**12**).



12. Menedżer urządzeń ustawił COM9 jak port przejściówki



13. Ustawienia Putty do komunikacji z portem szeregowym

W naszym przypadku nie musicie osobno konfigurować właściwości sterownika w Menedżerze Urządzeń. Putty poradzi sobie z ich odpowiednim nadpisaniem.

Teraz musimy poznać parametry transmisji konsoli UART RPi. „8N1" to wartość domyślna. Pozostaje określić szybkości transmisji. Jest to parametr startu kernela RPi. Można go odczytać z pliku „cmdline.txt" karty SD. Z poziomu Linuksa RPi (zakładam, że używacie dystrybucji Raspbian) znajdziecie ten plik w katalogu „/boot". Wyświetlcie jego zawartość za pomocą komendy „cat":

```
cat /boot/cmdline.txt
>dwc_otg.lpm_enable=0
console=ttyAMA0,115200 console=tty1 root=/dev/
mmcblk0p2 rootfstype=ext4 elevator=deadline
rootwait
```

Zwróćcie uwagę na wpis: „console=ttyAMA0,115200". Wynika z niego, że chodzi o szybkość 115 200. Alternatywnie kartę SD można włożyć do komputera z odpowiednim czytnikiem kart. Plik „cmdline.txt" znajduje się na partycji widocznej z poziomu Windows.

Na komputerze startujemy klienta Putty, ustawiając sesję na odpowiedni port szeregowy – tutaj COM9 – i szybkość: 115 200 (**13**).

W następnym kroku do wyłączonego z prądu Raspberry podłączamy konwerter UART-do-RS232. Ten z kolei łączymy z przejściówką RS232-do-USB. Teraz możemy otworzyć połączenie szeregowe z Putty (klawisz „Open") i włączyć RPi do zasilania. Jeżeli wszystko poszło dobrze – cieszymy się logami startowymi wyświetlanymi w oknie Putty.

Należy tu zwrócić uwagę, że wpięcie przejściówki do innego portu USB może spowodować przypisanie jej innego numeru portu szeregowego. Takie zachowanie jest zależne od samego układu i sterowników zainstalowanych w systemie operacyjnym.

Z Linuksem...

Nie ma żadnego problemu z instalacją Putty dla komputerów opartych na Linuksie. Dla przykładu, żeby zainstalować Putty na Ubuntu wystarczy wydać komendę: „sudo apt-get install putty" (komputer musi mieć dostęp do Internetu). Dostęp do przejściówki odbywa się poprzez port szeregowy, najczęściej oznaczony jako: „/dev/ttyUSB0". Pamiętajcie o nadaniu odpowiednich praw dostępu do tego portu: „sudo chmod 666 /dev/ttyUSB0". Komendę należy wykonać z terminala za każdym razem, gdy włączamy przejściówkę do portu USB.

Na skróty: UART do USB

Powyższe rozważania pozwalają na uzyskanie bardzo elastycznego rozwiązania. W sprzedaży znajdziemy również układy, które oferują bezpośrednie przejście z UART na USB (**14**).

Użycie takiego konwertera ma kilka zalet:

- nie potrzebujemy konwertera UART-do-RS232 (tanie czasami sprawiają problemy);
- nie potrzebujemy przejściówki RS232-do-USB (tanie zazwyczaj sprawiają problemy);
- nie potrzebujemy dodatkowych kabli RS232 (dla wygody przedłużacz USB).

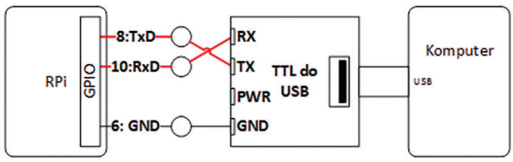
Najpierw należy się upewnić, że konwerter UART-do-USB **operuje logiką 3,3 V**. Niektóre układy tego typu oferują jedynie wyjście z logiką 5 V. Ten poziom jest jak najbardziej akceptowalny przez Arduino, ale nie przez RPi. Napięcie wyższe niż 3,3 V, przyłożone do pinów danych GPIO, może spowodować ich uszkodzenie.

Spójrzmy na przykładową realizację konwertera UART-do-USB, opartą na układzie FTDI FT232R (**15**).

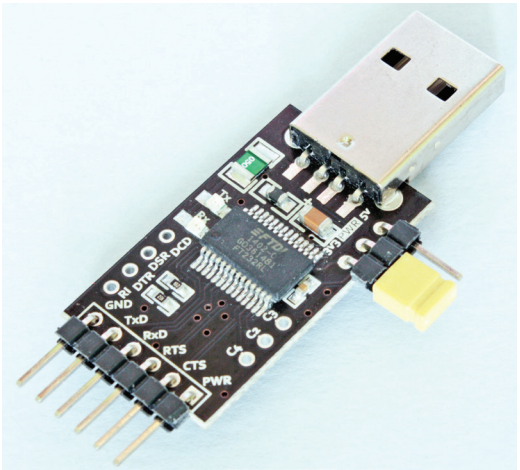
Konwerter ma dodatkową zworkę, która umożliwia przełączanie poziomów logiki między 3,3 i 5 V. Upewnijcie się, że dla RPi zworka jest przełączona na 3,3 V, a dla Arduino na 5 V. Następnie połączcie piny tak, żeby:

- Pin 8 GPIO (fizyczny, TxD, GPIO14) połączony był z RxD konwertera;
- Pin 10 GPIO (fizyczny, RxD, GPIO15) połączony był z TxD konwertera;
- Pin 6 Raspberry (fizyczny, GND) połączony był z GND konwertera.

Proszę zauważyć, że piny TxD i RxD spięte są na krzyż: TxD RPi z RxD konwertera i RxD RPi z TxD konwertera – odwrotnie niż w przypadku konwertera UART-do-RS232. Testowany przeze mnie konwerter UART-do-USB działał bezproblemowo



14. Idea łączenia UART RPi z USB komputera za pomocą konwertera UART-do-USB



15. Konwerter UART-do-USB z układem FTDI FT232R; zworka przełącza napięcie logiki między 3,3 i 5V (msx-elektronika.pl)

na Win8, Win7, Vista i Ubuntu. Koszt jego zakupu nie przekroczył 40 zł.

Proszę również zwrócić uwagę na szpilkę konwertera oznaczoną PWR. Konwerter UART-do-RS232, opisany wcześniej (**6**), zasilany jest poprzez fizyczny pin 1 RPi (3,3 V, dokładniej zasilany jest układ MAX3232). Szpilka oznaczona jako VCC oczekuje zasilania, jest **wejściem**. Konwertery UART-do-USB zbudowane są inaczej: szpilka opisana jako PWR to **wyjście**. Niestety, nie przyda się ona do zasilenia RPi. Ustawiając poziom logiki na 3,3 V, na PWR otrzymamy 3,3 V. To zbyt niska wartość, żeby zasilić RPi (zwłaszcza, że maksymalny prąd nie przekracza 100 mA). PWR może się przydać np. dla Arduino, które posługuje się logiką 5 V (500 mA do dyspozycji w opisywanym modelu). Ostatnią rzeczą, o którą musicie zadbać, jest połączenie pinu GND konwertera z pinem 6 (GND) RPi. Masa musi być wspólna.

Na marginesie proszę pamiętać, że fizyczny pin numer 2 RPi jest bezpośrednio połączony do szyny zasilania. **Nie ma żadnych dodatkowych zabezpieczeń** na wypadek zwarcia, skoków napięcia itp. (inaczej niż „oficjalne" zasilanie poprzez port microUSB). Musicie się upewnić, że podłączone do tego pinu zasilanie (ang. *back-power*) jest stabilne. Inaczej możecie bardzo skutecznie uszkodzić swoje RPi. Nigdy nie zasilajcie też RPi z dwóch źródeł jednocześnie: portu microUSB i przez pin 2.

Raspberry Pi model B+

Pierwsze 26 pinów Raspberry Pi B+ jest identyczne z GPIO Raspberry B. Nie ma żadnych różnic przy podłączaniu UART do tych modeli Raspberry.

Podsumowanie

Połączenie RPi przez UART pozwala na uzyskanie łatwego dostępu do konsoli. Taki sposób jest bardzo pomocny zarówno w diagnostyce RPi, jak i w codziennym użytkowaniu. Nie potrzebujemy wtedy sieci ani monitora. Możemy wykorzystać jakikolwiek komputer (również laptop) – nawet, gdy jest wyposażony jedynie w port USB. Konwertery UART-do-RS232 są tanie i zazwyczaj nie sprawiają żadnych problemów (pod warunkiem wykorzystania układu MAX 3232). Choć, jak wspomniałem, mnie udało się kupić jeden niedziałający. Więcej szczęścia (lub po prostu funduszy) potrzeba przy zakupie kabla-przejściówki RS232-do-USB. Te najtańsze bywają kapryśne. Chyba że zdecydujemy się na większy wydatek i zaopatrzymy w kabel oparty na oryginalnych układach (np. FTDI). Najlepiej więc kupować tam, gdzie istnieje możliwość wymiany. Układy UART-do-USB mogą być atrakcyjną alternatywą, pod warunkiem że ostrożnie dobierzemy je pod kątem wyjściowego poziomu napięć logiki. ■

Arkadiusz Merta

Źródła:
http://goo.gl/JpEQjj
http://goo.gl/tMviLq
http://goo.gl/PzB7fA
http://goo.gl/ktcDhn

Szanowni Czytelnicy. Ewentualne pytania do autora można kierować bezpośrednio, na adres:
arkadiusz.merta@mt.com.pl

Jak grzyby po deszczu pojawiają się nowe „owocowe" komputerki. Części konstruktorów nieobca jest Sakura, czyli „wiśnia", z 32-bitowym procesorem RX63N firmy Renesas, a niedawno do koszyka owoców dodano Banana PI, czyli „banana", o którym przeczytacie w „Elektronice Praktycznej" 11/2014 – www.ulubionykiosk.pl