Wirtualny port szeregowy (015; 20.08.2009; *arduino; processing; blender*)

Wielokrotnie we wcześniejszych artykułach niezbędne do realizacji projektów było przesyłanie informacji pomiędzy programami np. między środowiskami Processing i Blender. Opiszę zatem przydatny sposób przesyłania danych pomiędzy aplikacjami, aby dalej nie trzeba było korzystać np. z pośrednictwa plików tekstowych.

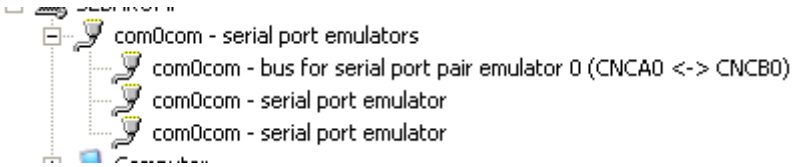
Com0com.

Com0com jest aplikacją typu open source dla środowiska Windows, która pozwala tworzyć pary wirtualnych portów szeregowych tak, że wyjście danych z jednego jest wejściem danych do drugiego i odwrotnie. Zanim dokładnie opisze sposób korzystania z tej aplikacji wspomnę, że projekt com0com udostępnia również aplikację *com2tcp*, która pozwala wysyłać dane z portu szeregowego poprzez internet, oraz aplikację *hub4com*, która umożliwia łączenie wielu portów szeregowych.

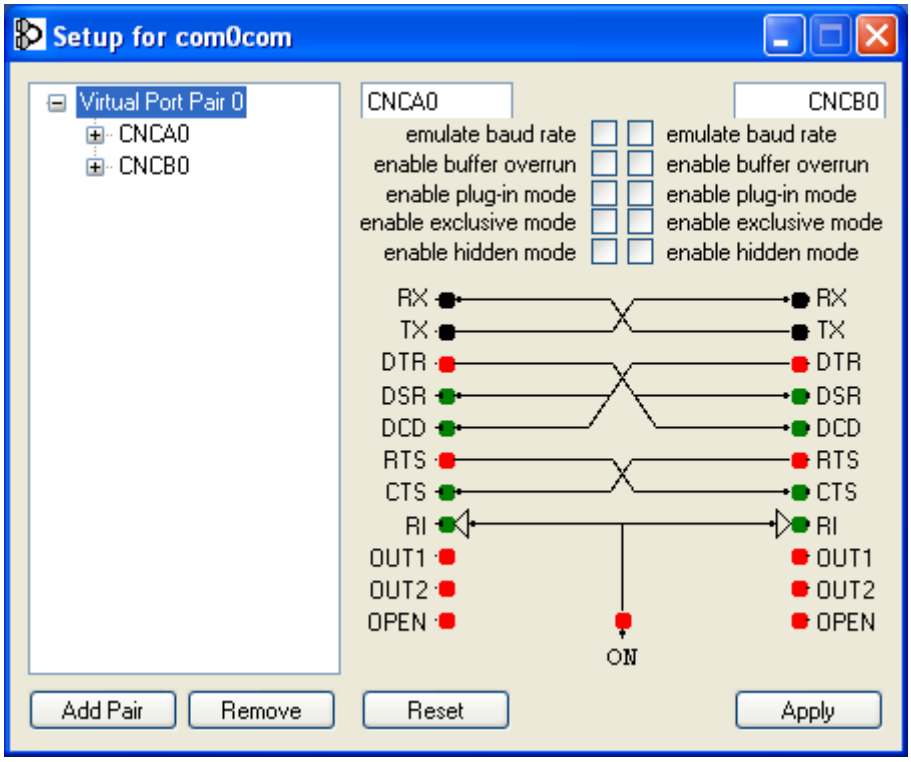
Tworzenie pary połączonych wirtualnych portów szeregowych.

Pod adresem http://sourceforge.net/projects/com0com/files/ odnajdujemy najnowszą wersję aplikacji *com0com* dla naszego systemu Windows (i386 lub x64, aktualnie najnowsza wersja nr 2.2.1.0) i pobieramy ją. Rozpakowujemy i uruchamiamy instalację. W czasie instalacji system wykryje dwa nowe urządzenie (CNCA0 i CNCB0), do których zainstalować trzeba sterowniki. W oknie instalacji sterowników zaznaczamy kolejno, że nie chcemy aby Windows połączył się z Windows Update w celu wyszukiwania sterowników, a następnie wybieramy opcję automatycznej instalacji urządzenia.

Przejdźmy do *manager'a urządzeń* systemu (właściwości mojego komputera / zakładka sprzęt). Na liście widnieć powinny następujące urządzenia:

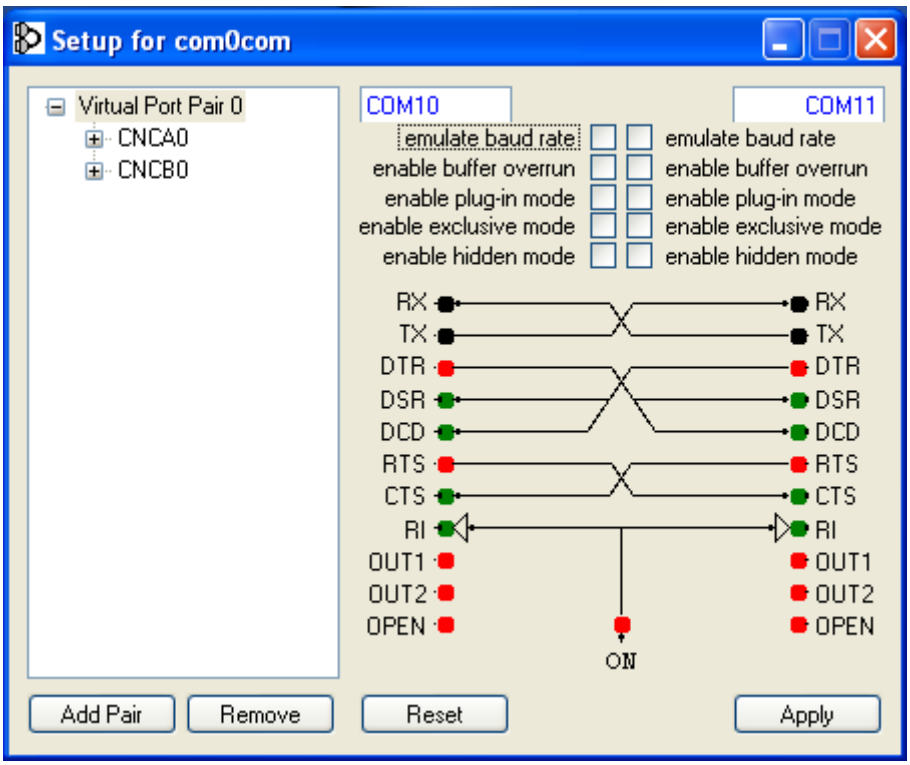


Poprzez *menu start/programy/com0com* uruchamiamy skrót Setup, otwierający następujące okno konfiguracji programu:



Okno to pozwala dodawać i usuwać nowe pary portów szeregowych, zmieniać ich parametry, a nawet przeciągnięciami myszki łączyć konkretne piny łączonych portów szeregowych. Warto zwrócić uwagę, na krzyżowe połączenie pinów RX i TX odpowiedzialnych w istocie za przesył danych.

Kolejnym krokiem jest zmiana domyślnych nazw portów, gdyż te mogą nie być widoczne przez aplikacji. Wpisujemy np. nazwy COM10 i COM11. Uwaga, gdy nazwa po wpisaniu będzie mieć czerwony kolor czcionki oznacza to, że ten port jest zarezerwowany i jego użycie może bardzo utrudnić pracę systemu (np. konflikty IRQ i zasobów). Wolna nazwa portu widnieć będzie niebieską czcionką.



Zatwierdzamy zmiany przyciskiem Apply.

Test połączenia pomiędzy portami wirtualnymi.

Otwórzmy dwa okna środowiska Processing i wpiszmy w nich dwa różne kody. Wysyłający dane na port COM10:

```
import processing.serial.*;
Serial portCOM10;
void setup() {
  portCOM10 = new Serial(this,"COM10", 9600);
}
void draw() {
  portCOM10.write(1);
  delay(200);
}
```

i odbierający dane z portu COM11:

```
import processing.serial.*;
Serial portCOM11;
void setup() {
  portCOM11 = new Serial(this,"COM11", 9600);
}
void draw() {
  if ( portCOM11.available() > 0) {
    print(portCOM11.read());
  }
}
```

Uruchommy pierwszy program, a następnie drugi. W konsoli drugiego okna powinny pojawiać się odbierane cyfry 1. Wyłączenie pierwszego programu przerywa transmisje, a ponowne uruchomienie wznawia. W przypadku nie działanie tego testu należy w obu programach w bloku *setup* dopisać polecenie *println(Serial.list())*, i sprawdzić czy po uruchomieniu programu na liście dostępnych portów widnieje utworzony przez nas port. Jeśli nie, należy sprawdzić poprawność utworzenia portów wirtualnych.

Uruchommy pierwszy program na kilka sekund i zamknijmy go. Po uruchomieniu drugiego okazuje się, że na porcie COM11 nie ma danych, które wysłał wcześniej pierwszy program. Jest tak dlatego, że wirtualne porty nie posiadają bufora danych.

Warto zwrócić uwagę, na pozostałe opcje portów, które pozwalają m. in. emulować prędkość transferu danych (*emulate baud rate*), blokować port (*plug-in mode*), gdy do portu sparowanego nie podłączony jest żaden program, jak również ukrywać porty.

Blender, Arduino i wirtualny port szeregowy.

Zwróciłem kiedyś uwagę, że podczas inicjacji połączenia szeregowego między Blenderem i Arduino następuje resetowanie mikrokontrolera, co bardzo spowalnia przesył danych. Problem ten rozwiązać można poprzez mechaniczną ingerencję w samą płytkę lub prostym programem w Processingu dobierać dane z Arduino (np. port COM4) i przesyłać je na np. na port wirtualny COM10. Utworzone wirtualne połączenie przesyłać będzie dane do portu COM11 z którego czerpać będzie Blender.

Skrypt w języku Python możliwy do zastosowania w Blenderze (również w silniku gry), odbierający pojedynczy bit danych z portu COM11 i wyświetlający go w konsoli:

```
import serial
myPort = serial.Serial('COM11', 9600)
y = ord(myPort.read(size=1))
print y
```

Jeśli chcielibyśmy, aby dane z Arduino bezpośrednio trafiały na port wirtualny, należało by port COM zarezerwowany dla Arduino (np. COM4) użyć jako jeden ze sparowanych portów wirtualnych. Jednak moje testy nie powiodły się, więc lepiej pozostać przy wyżej opisanym sposobie komunikacji.