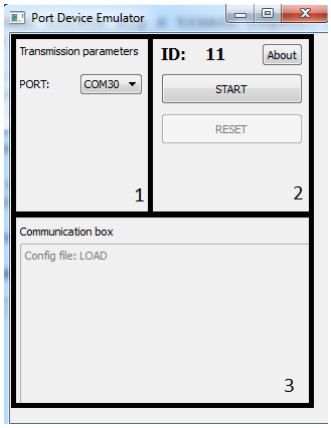
1 Program

Program *Serial Device Emulator*, ma na celu emulowanie urządzenia podłączonego do portu szeregowego. Program poprzez maszynę stanów odpowiada na komendy użytkownika symulując proces ustawiania właściwości urządzenia. Na rys. 1 pokazano interfejs użytkownika programu



Rysunek 1: Screen programu Serial Device Emulator

Serial Device Emulator, który korzysta z biblioteki QT dostępnej w języku Python, w którym stworzono symulator.

Interfejs programu składa się z trzech części. Część nr. 1 (*Transmission parameters*) służy do wyboru portu, z którym zostanie połączony program.

W części nr. 2 podano numer ID urządzenia, który jest wykorzystywany do inicjalizacji symulatora trybu konfiguracyjnego, co opisano w kolejnym punkcie. Dodatkowo w tej części znajdują się przyciski *Start* i *Reset*. Przycisk *Start* uruchamia program na słuchania portu szeregowego i interakcje z danymi wysyłanymi podczas konfiguracji. Przycisk *Reset* służy do resetowania stanu konfiguracji, tj. przejścia do stanu początkowego.

Ostatnia część okna programu to *Communication box* (sekcja 3), w którym program informuje o stanie symulowanej konfiguracji.

2 Rozpoczęcie pracy z programem

Do uruchomienia programu potrzebne jest środowisko Python wraz z kilkoma pakietami: m.in *pyserial,pyqt*. Kompilator Python wraz z domyślnie zainstalowanymi wieloma pakietami, można zainstalować wykorzystując paczkę *Winpython*, dostępną pod adresem `winpython.github.io`

W celu uruchomienia programu, w konsoli (np. Windows, jeśli ustawiono odpowiednie zmienne środowiskowe) lub w konsoli dostarczonej z WinPython. tj. *WinPython Command Prompt* należy wywołać polecenie (gdy ścieżka konsoli jest ustawiona na folder z plikami programu)

```
python main.py
```

które skompiluje i uruchomi program.

Z poprawnym uruchomieniem programu związane jest wczytanie pliku konfiguracyjnego. Plik konfiguracyjny musi znajdować się głównym folderze programu, tj. w folderze z plikiem *main.py*. Jego poprawne wczytanie jest komunikowane w *Communication Box* przez : *Config file: LOAD*. Pokazuje to rys. 2

Rysunek 2: Komunikat informujący o poprawnym wczytaniu pliku konfiguracyjnego

Następnie w sekcji *Transmission parameters* z listy rozwijanej należy wybrać port, na którym urządzenie będzie widoczne. Paramtery transmisji na wybranym porcie są mają następujące wartości:

- Prędkość (ang. *baudrate*): 9600
- Wielkość ramki (ang *byte size*): 8
- Bit parzystość (ang.*parity*): brak (ang. *None*)
- Bity stopu (ang. *stop bits*): 1
- Kontrola przepływu (ang. *flow control*): Brak (ang. *None*)

Podane wartości to domyśle ustawienia biblioteki *pyserial*, która odpowiada za komunikacje z portem szeregowym.

Połączenie z wybranym portem szeregowym nastąpi po wybraniu przycisku *START*. Poprawne połączenie programu z portem sygnalizuje odpowiedni wpis w sekcji komunikatów np. *Port COM7 is open*. Niepoprawne połączenie z wybranym portem sygnalizuje błąd. Ponadto, na rys. 3 pokazano, ż kolejne komunikaty są dopisywane od góry, co jest cechą sekcji *Communication Box*.

Rysunek 3: Komunikat informujący o poprawnym wczytaniu pliku konfiguracyjnego

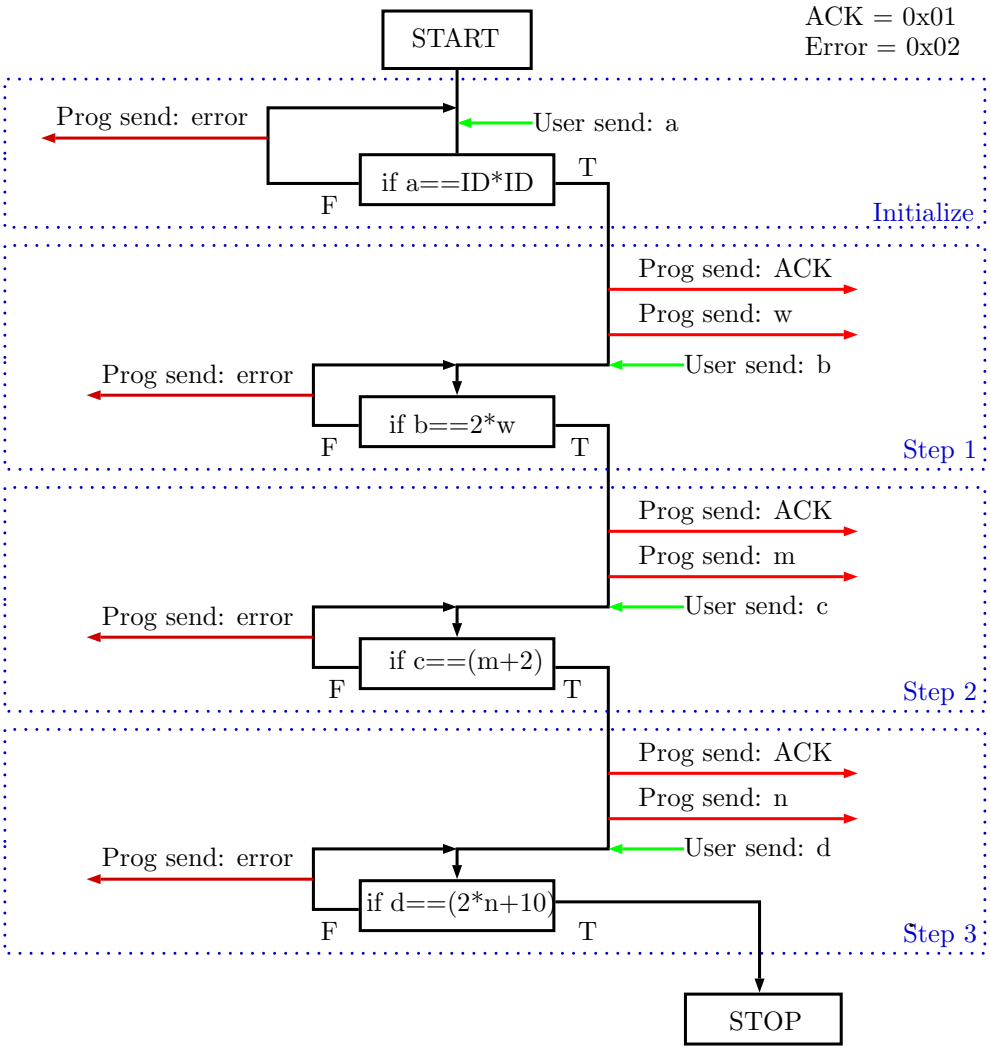
Po wybraniu przycisku *START* i poprawnym połączeniu z portem, przycisk start zostaje desaktywowany a program oczekuje na komunikacje, którą opisano w pkt. 3. Stąd, by skomunikować się z innym portem szeregowym należy ponownie uruchomić program. Przycisk *RESET*, który umieszczono pod przyciskiem *START* służy jedynie do przywrócenia procedury konfiguracji do jej początkowego stanu.

3 Symulowanie konfiguracji i jej procedura

Jak wspomniano wcześniej, program *Serial Device Emulator* ma za zadanie symulować procedurę konfigurowania urządzenia podłączonego przez port szeregowy. Sam jej przebieg określa schemat, który przedstawiono na rys. 4.

Jak zaznaczono na schemacie z rys 4, procesy symulowanej konfiguracji rozpoczyna się od fazy inicjalizacji. By jej przebieg zakończył się pomyślnie, użytkownik powinien wysłać na port do którego podłączono wirtualne urządzenie liczbę, której wartość jest równa ID^2. Jeśli tak jest, program wysyła na swój port potwierdzenie ACK (ten fakt należy już do kolejnego kroku), które jest określone jako cyfra 1 (hexadecymalnie 0x01), lub zgłasza błąd, poprzez wysłanie sygnału *error*, tj. 0x02, i ponownie oczekuje na wysłanie poprawnej wartości.

Jeśli faza inicjalizacji zakończyła się pomyślnie, program przechodzi do kroku pierwszego, którego początkiem jest wysłanie ACK. Następnie zaraz za nim zostaje wysłany kolejny bajd



Rysunek 4: Schemat przebiegu symulowanej konfiguracji

danych, który zawiera liczbę, w przypadku kroku I, będzie to liczba w. Zadaniem użytkownika w tym kroku jest jej odczytanie, i odesłanie takiej wartości, by równości podane w blokach *if* były spełnione. W przypadku kroku pierwszego należy odesłać wartość $2 * w$. Następnie, podobnie jak w inicjalizacji, spełnienie bądź niespełnienie warunku podanym w bloku *if* skutkuje odpowiednie wysłaniem kodu *ACK* bądź kodu *Error*.

Kolejne dwa kroki są analogiczne do pierwszego. Ich zmienną są tylko wartości nadesłanie przez program oraz postać zależności wg których należy obliczyć wartość wysyłaną przez użytkownika. Te ostatnie podano w schemacie z rys. 4 w odpowiednich blokach *if*.

Program kończy swoje działanie, po poprawnym wykonaniu kroku trzeciego, co oznaczono na schemacie jako blok *STOP*. Ponadto, zakończenie symulowanej konfiguracji jest sygnalizowane w *Communication Box* za pomocą komunikatu *Device running*.

4 Działanie przycisku reset

Wybranie przycisku *RESET* prowadzi do powrotu symulowanej konfiguracji do stanu początkowego, który na diagramie z rys. 4 jest utożsamiany z blokiem *START*.