Przesyłanie analogowych i cyfrowych danych pomiarowych.

1. Nowe funkcje służące do przesyłania danych

Poznają Państwo dzisiaj nowe obiekty służące do przesyłania danych **w reżymie czasu rzeczywistego**. W zakładkach *Functions/Data Communication* oraz *Functions/Real-Time* można znaleźć wiele takich obiektów (są one również powtórzone w innych zakładkach). Z niektórych z nich już korzystaliśmy. Dziś zajmiemy się trzema obiektami:

1. RT FIFO

Funkcje **RT FIFO** znajdujące się w *Functions/Real-Time/RT FIFO* umożliwiają przesyłanie danych pomiędzy programami vi w reżymie czasu rzeczywistego na jednym komputerze. Komunikacja ta jest deterministyczna, gdyż nie wprowadza czasowego rozmycia (*jitter*) w przesyłaniu danych. Nie jest ona bezstratna, gdyż jeżeli pamięć FIFO jest pełna, to dane pisane są po sobie.

2. Queue Operations

Kolejki komunikatów są obsługiwane funkcjami znajdującymi się w zakładce *Functions/Data Communication/Queue Operations.* Służą one do przesyłania danych pomiędzy różnymi obiektami wewnątrz jednego programu vi lub pomiędzy programami na jednym komputerze. Są one stosowane w systemach zwykłych i RT.

3. Network Streams

Strumienie sieciowe służą do przesyłania danych przez sieć, np. pomiędzy komputerem docelowym i głównym lub między różnymi aplikacjami na jednym komputerze. Nie należy ich stosować do przesyłania danych wewnątrz jednej aplikacji. Umożliwiają one bezstratne, jednokierunkowe przesyłanie danych ciągłych z możliwością oczekiwania. Funkcje te znajdują się w zakładce *Functions/Data Communication/Network Streams. S*ą one dużo lepsze do przesyłania strumieni danych od zmiennych sieciowych, których do tej pory używaliśmy - zmienne sieciowe (*shared variables*) są zalecane do przesyłania danych nieciągłych, są optymalizowane do przesłania najnowszych zebranych danych i nie gwarantują bezstratności.

Proszę zapoznać się z funkcjami znajdującymi się w wymienionych wyżej zakładkach ***Functions/Data Communication*** oraz ***Functions/Real-Time***. Używanie tych funkcji wymaga pewnej wprawy, gdyż ich złe zdefiniowanie może zawiesić komputer. Dlatego proszę zrobić własną aplikację dokładnie według podanego niżej opisu, a dopiero później ją ewentualnie modyfikować.

2. Własna aplikacja

Tworzymy projekt Nazwisko_C w którym umieścimy: na komputerze głównym będzie program Nazwisko_Host_C.vi, a na komputerze docelowym dwa programy: Nazwisko_Target_C_1.vi oraz Nazwisko_Target_C_2.vi. Celem projektu będzie czytanie danych analogowych oraz cyfrowych[1] z urządzenia (karty PXI-6221) znajdującego się w kasecie PXI oraz przesyłanie danych analogowych na komputer główny. Przesyłanie sygnału analogowego ma się odbywać tylko wtedy, gdy przeczytane dane cyfrowe są równe TRUE lub gdy zażąda tego użytkownik na komputerze głównym (w programie Nazwisko_Host_C). Poniżej znajduje się opis zawartości poszczególnych programów:

a) Nazwisko_Target_C_1.vi

Program zawiera **dwie pętle czasowe** taktowane z okresem 10 µs.

Do komunikacji z kartą NI PXI-6221 użyjemy funkcji **Asystent DAQ** (*DAQ Assist*), która jest najprostszą funkcją umożliwiającą na stworzenie systemu akwizycji danych, a która znajduje się w *Functions> Measurement I/O> DAQmx – Data Acquisition* lub w *Functions> Express> Input.* Po ułożeniu bloczka na diagramie blokowym otworzy się okno umożliwiające jego konfigurację:

1. **W pierwszej pętli czasowej** będziemy odbierać sygnały analogowe przychodzące na pierwsze wejście analogowe karty PXI-6221, więc w tym przypadku wybieramy Acquire Signals> Analog Input> Voltage> kanał **ai0** urządzenia PXI-6221. Klikamy Finish i na następnym oknie jako Terminal Configuration wybieramy RSE (*ground-Referenced Single-Ended*, czyli sygnał mierzony względem wspólnej masy). Jako Acquisition Mode wybrać należy N Samples, Samples to read = 1000, rate (Hz) = 1000 (lub 1k). Zamykamy okno klikając OK.

   Dane będą wysyłane poprzez pamięć **RT FIFO**. W tym celu na lewo od pętli musimy zainicjalizować komunikację FIFO poprzez umieszczenie tam funkcji ***RT FIFO Create***. Oto parametry tej funkcji, które trzeba zdefiniować:

   i. ***name*** – nadajemy nazwę stworzonemu RT FIFO np. FIFOData. Nazwa ta umożliwi nam czytanie wysłanych danych w drugim programie.

   ii. ***type*** – trzeba zdefiniować typ danych przesyłanych przez FIFO. Będziemy przesyłać tablicę danych zmiennoprzecinkowych. Dlatego do tego wejścia podłączamy stałą, będącą jednowymiarową tablicą typu *double*. Wartość stałej nie ma żadnego znaczenia, ważny jest tylko typ.

   iii. ***size*** – rozmiar bufora. Tu wystarczy wpisać 1, czy jedną tablicę.

   iv. ***elements in array*** – liczba elementów w tablicy. Tu wpisujemy 1000, gdyż funkcja *DAQ Assisstant* produkuje 1000 próbek za jednym razem (tak ją zdefiniowaliśmy).

   RT FIFO jest już zdefiniowane, więc wewnątrz pętli należy umieścić funkcję ***RT FIFO Write***, którą łączymy z ***RT FIFO Create***. Do wejścia *element* podajemy dane przeczytane przez funkcję *DAQ Assistant*. Na zewnątrz pętli czasowej, po prawej, umieszczamy funkcję ***RT FIFO Delete***, aby zlikwidować FIFO po ukończeniu działania programu.

   Aby zatrzymać pętlę wykorzystamy zmienną sieciową współdzieloną STOP, która będzie przekazywać wartość logiczną z komputera głównego. Podłączmy tę zmienną tak, by zatrzymać i pętlę, i funkcję *DAQ Assistant*.

2. **W drugiej pętli czasowej** również umieszczamy funkcję *DAQ Assistant*, ale ma ona czytać wejście cyfrowe **port0/line2** (pojedynczy bit) w trybie ***1 Sample (On demand)***.

[1] Dane cyfrowe (logiczne, binarne) to dane przyjmujące jedynie wartość 0 (FALSE), lub 1 (TRUE). Np. rejestracja sygnału TTL generuje dane cyfrowe.

Zmierzone informacje logiczne przesyłane będą za pomocą kolejki komunikatów. Analogicznie, musimy wpierw tę kolejkę zdefiniować. Dlatego też na lewo od pętli umieszczamy funkcję ***Obtain Queue*** o następujących wejściach:

i. ***name*** – nadajemy nazwę np. QueueBool, by można było z tej kolejki skorzystać w innym programie i czytać dane.

ii. ***element data type*** – tu również musimy podać rodzaj przesyłanych danych. Dlatego podpinamy stałą logiczną o dowolnej wartości.

iii. ***max queue size*** – rozmiar kolejki. Tu dajemy rozmiar równy 1.

Kolejka jest zdefiniowana, więc wewnątrz pętli umieszczamy funkcję ***Enqueue Element*** którą łączymy z funkcją ***Obtain Queue***, natomiast do jej wejścia *element* podajemy dane odczytane z *DAQ Assisant*. **Uwaga**: *DAQ Assistant* generuje jednoelementową tablicę logiczną i dlatego trzeba wyciągnąć z tej tablicy **jej pierwszy element** i wpisać go do kolejki komunikatów. Do wejścia *Timeout* podpinamy wartość 10 ms. Analogicznie, jak w przypadku FIFO, po prawej od pętli umieścimy funkcję usuwającą kolejkę po zakończeniu działania programu, czyli funkcję ***Release Queue***. Pętla zatrzymuje się analogicznie, jak pętla pierwsza, czyli poprzez podpięcie zmiennej sieciowej STOP tak, by zatrzymać pętlę oraz funkcję *DAQ Assistant.*

Dodatkowo, tak jak to już kiedyś robiliśmy, po zakończeniu obu pętli wpisujemy w zmienną współdzieloną STOP wartość FALSE.

b) Nazwisko_Target_C_2.vi

Tutaj znajduje się tylko jedna pętla czasowa o okresie równym również 10 μs. Wewnątrz tej pętli będziemy czytać dane wysłane w programie Nazwisko_Target_C_1.vi i dlatego tutaj również musimy zainicjalizować RT FIFO oraz kolejkę komunikatów. Dlatego na lewo od pętli umieszczamy odpowiednie funkcje inicjalizujące (można je skopiować z Nazwisko_Target_C_1.vi). W szczególności ich nazwy muszą być takie same.

Ponieważ sygnał analogowy przesłany z Nazwisko_Target_C_1.vi będzie stąd wysyłany dalej na komputer główny, więc na lewo od pętli czasowej dokonamy trzeciej inicjalizacji. Umieszczamy funkcję ***Create Network Stream Writer Endpoint***, która inicjalizuje strumień danych do pisania przez sieć. Wejścia definiujemy następująco: *writer buffer size* oraz *data type* są takie same, jak przy definiowaniu RT FIFO, czyli odpowiednio 1 i tablica zmiennych typu *double*. *Writer name* niech będzie np. DataStreamW.

Przesyłanie danych mamy już zainicjalizowane. Wewnątrz pętli muszą się więc znaleźć funkcje ***Dequeue Element*** do czytania danych binarnych, ***RT FIFO Read*** do czytania danych analogowych oraz ***Write Single Element to Stream*** do pisania danych do strumienia sieciowego. Obie funkcje czytające mają zdefiniowany czas oczekiwania (*timeout*) równy 1 s. Funkcja pisząca do strumienia ma domyślną wartość -1, czyli czekanie do skutku.

Funkcje te należy tak podłączyć, by czytanie danych z FIFO i pisanie do strumienia sieciowego odbywało się tylko wtedy, gdy przesłana przez kolejkę komunikatów wartość logiczna jest równa TRUE lub wtedy, gdy zażądamy tego w programie Nazwisko_Host_C.vi na komputerze głównym. Do realizacji tej drugiej opcji wykorzystamy sieciową zmienną logiczną (taką jak STOP) o nazwie np. SignalSend.

Aby zatrzymać pętlę wykorzystamy zmienną sieciową współdzieloną STOP, którą już utworzyliśmy pisząc program Nazwisko_Target_C_1.vi. Analogicznie, po zakończeniu wykonywania pętli wpisujemy w zmienną współdzieloną STOP wartość FALSE.

c) Nazwisko_Host_C.vi

Komputer główny przyjmuje wysyłany z komputera docelowego sygnał analogowy i wysyła tam dwa sygnały logiczne: STOP i żądanie sygnału SignalSend. Potrzebować będziemy jednej, zwykłej pętli *while*. Oto opis programu:

1. Ponieważ będziemy czytać dane ze strumienia sieciowego, więc musimy go wpierw zainicjalizować do czytania, czyli użyjemy funkcji ***Create Network Stream Reader Endpoint***, którą umieszczamy na lewo od pętli *while*. Parametry inicjalizacji są takie, jak w Nazwisko_Target_C_2.vi, z tą różnicą, że:

    i. *reader name* – np. DataStreamR.

    ii. *writer url* – musimy podać adres funkcji piszącej do strumienia (ID kontrolera PXI i nazwa funkcji piszącej): **//169.254.148.181/DataStreamW**

    Programy Nazwisko_Host_C.vi oraz Nazwisko_Target_C_2.vi będą czekać dopóty, dopóki nie odnajdą się funkcje pisząca i czytająca. Podanie złego adresu spowoduje więc zawieszenie programów.

2. Zanim wykonana zostanie powyższa funkcja inicjalizujemy wartością FALSE zmienne sieciowe STOP i SignalSend.

3. Na panelu czołowym umieszczamy dwie kontrolki typu logicznego: STOP i SignalSend. Ich bloczki na diagramie blokowym będą znajdować się wewnątrz pętli *while*. Tę pierwszą podpinamy oczywiście do bloczka zatrzymującego pętlę oraz do zmiennej sieciowej STOP, a tę drugą do zmiennej sieciowej SignalSend.

4. Czytamy dane ze strumienia sieciowego za pomocą funkcji ***Read Single Element from Stream*** i otrzymane dane wyświetlamy na wyświetlaczu graficznym *Waveform Graph*. *Timeout* (limit czasu) funkcji czytającej wynosi 1 s.

Uruchamiamy wszystkie trzy programy w dowolnej kolejności i obserwujemy odbierany przebieg. Podczas uruchamiania programów dobrze jest skorzystać z możliwości podglądania odbieranych sygnałów logicznych i analogowych za pomocą okienka Probe.
Później można zmienić bufor FIFO i strumienia sieciowego na większy od 1. Można wtedy też zobaczyć (poprzez podglądnięcie wyjścia *empty?* funkcji ***RT FIFO Read***), kiedy funkcja ta sygnalizuje pusty bufor.

We wszystkich powyższych programach połączmy sensownie kabelki błędów.