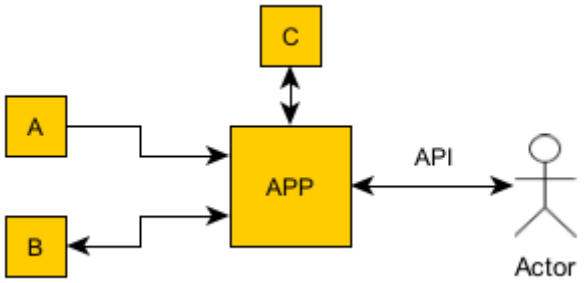
Zadanie do procesu rekrutacji:

Opis problemu:



Rysunek 1 Schemat systemu

W systemie występują dwa urządzenia fizyczne A i B podłączone do sieci WAN za pomocą bramek CAN-TCP. Bramki udostępniają po stronie WAN serwer TCP, który akceptuje/wysyła ramki o formacie przedstawionym w dalszej części dokumentu.

Aplikacja APP umieszczona jest w chmurze i utrzymuje z urządzeniami A i B łączność, za pomocą bramek, w formie krótkich wiadomości o ustalonej strukturze (nagłówek, rozmiar, dane) ze zmienną zawartością (CAN ID, wartości poszczególnych pól). Częstotliwość nadawania ramek jest stała dla poszczególnych urządzeń, zależy od ich konfiguracji i wynosi od kilku do kilkunastu na sekundę. Jedna bramka CAN-TCP może obsłużyć jednocześnie maksymalnie 4 połączenia TCP. Po nawiązaniu połączenia serwer bramki rozpocznie samoczynnie przekazywanie ramek pochodzących od urządzeń.

Urządzenie A ma możliwość raportowania stanu pracy, ale nie przyjmuje danych. Urządzenie B raportuje swój stan oraz akceptuje polecenia sterujące jego pracą (określone CAN ID wraz z parametrem). Stan urządzeń raportowany jest w formie przekazania kilku rejestrów (można przyjąć 16 bitowe).

Urządzenie B wyposażone jest w mechanizm watchdog, który musi otrzymywać komunikaty zerujące nie rzadziej niż co 500ms, niezależnie od pozostałych komunikatów. Zerowanie watchdog polega na wpisaniu ustalonej wartości do jednego z rejestrów. Niezależnie od tego mechanizmu, urządzenie B może przyjmować ramki ustalające rejestry kontrolujące jego pracę. Informacje te przekazywane są samoczynnie przez aplikację APP na podstawie wbudowanego algorytmu kontrolno-sterującego lub dodatkowo na żądanie użytkownika.

Blok C jest wymiennym elementem aplikacji, który reaguje na parametry przekazywane przez moduły A i B. Na potrzeby zadania można przyjąć, że moduły A i B przekazują pojedyncze parametry będące liczbami, a moduł C wykonuje na nich prostą operację arytmetyczną i udostępnia wynik z powrotem dla aplikacji. Moduł C jest blokiem pracującym w reżimie systemu czasu rzeczywistego. Implementowany przez niego algorytm wymaga by obliczenia wykonywać co 100ms. Wynik zależy od bieżącego czasu.

Aplikacja udostępnia na zewnątrz interfejs użytkownika w formie API, dostępny dla użytkownika za pomocą przeglądarki WWW. Użytkownik ma możliwość odczytania za pomocą API informacji przekazywanych przez urządzenia A i B oraz odczytania wyniku obliczeń wykonywanych przez moduł

C. Użytkownik ma możliwość ustalenia parametrów kontrolujących pracę algorytmu zawartego w aplikacji APP oraz parametrów pracy modułu B.

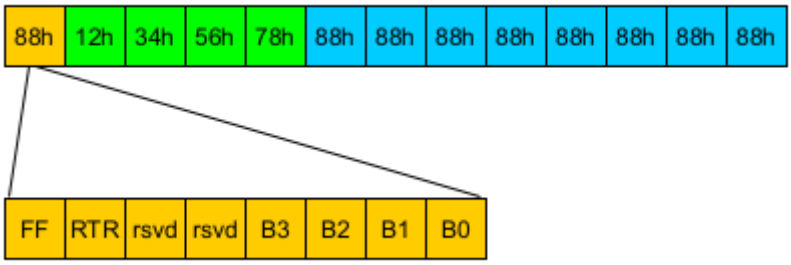
Opis zadania:

Zadanie polega na utworzeniu w języku Python:

- aplikacji APP, która udostępnia interfejs (np. FastAPI) umożliwiający:
    - monitorowanie/prezentację stanu urządzeń: A, B oraz modułu C
    - wysyłanie poleceń/konfigurowania urządzenia B i algorytmu APP
- emulatorów/zaślepek dla urządzeń A i B

Wyciąg z dokumentacji bramki CAN-TCP:

CAN data transfer to Ethernet data: Transfer CAN ID and data to 13 bytes



FF: Identifying bit to identify standard frame and extended frame, 1 is extended frame and 0 is standard frame.

RTR: Identifying bit to identify remote frame and data frame, 1 is remote frame and 0 is data frame.

Reserved: Reserved bit needs to be 0.

B3~B0: Data length bit to identify CAN frame

CAN to Ethernet: CAN side transmit data(Frame format: extended frame; frame type: data frame; ID: 12345678; data: 12 34 56 78 00) and Ethernet side will receive: 85 12 34 56 78 12 34 56 78 00 00 00 00. 0x85 means frame

format is extended frame, frame type is data frame and data length is 5; ‘12 34 56 78’ are CAN ID; ‘12 34 56 78 00 00 00 00’ are data with effective length 5 and the rest bits are supplied with 0.

Ethernet to CAN: Ethernet side transmit: 05 00 00 06 78 12 34 56 78 00 00 00 00. 0x05 means standard frame, data frame and data length is 5; 00 00 06 78 means CAN ID is 0678; 12 34 56 78 00 00 00 00 are data with effective length