Dokumentacja projektu PWiR

„Symulacja oświetlenia”

Przemysław Nocoń, grupa3b

Jakub Serweta, grupa3b

15.01.2020

1. Cel projektu

Stworzenie programu symulującego działanie oświetlenia. Symulator został napisany w języku Ada, który ułatwia współbieżne dostosowywanie działania, a także jasności działania poszczególnych elementów oświetlenia na podstawie danych przesłanych z sensorów oraz analogowych przełączników.

2. Opis działania symulatora

Program uruchamiamy poprzez procedurę ***Driver,*** która wywołuje procedurę ***Run*** generującą zadaną ilość czujników oraz świateł. Symulator oświetlenia obsługuje trzy rodzaje czujników, które mają wpływ na zapalenie oraz natężenie zapalonego światła. Są to:

- przełącznik włącz/wyłącz (***task type Switch_Sensor)*** – czujnik o najwyższym priorytecie, który posiada 3 tryby pracy. Pozwala na włączenie, wyłączenie lub automatyczne ustawianie wartości wszystkich dostępnych lamp.
- czujnik ruchu **(*task type Movement_Sensor)*** – czujnik działa jedynie wtedy, gdy jest wykryty za niski poziom nasłonecznienia. Ma on za zadanie włączyć lampę, za którą odpowiada i ustawić jej jasność na maksymalną na zdefiniowany odgórnie czas. Gdy w danej sekcji czujnik nie wykrywa ruchu, a światła powinny być zapalone będą one świeciły według parametrów przekazanych przez czujnik nasłonecznienia.
- czujnik nasłonecznienia (***task type Light_sensor***) – czujnik o najniższym priorytecie, również ma wpływ na lampy tylko gdy poziom światła jest dostatecznie niski. Na potrzeby symulacji generuje on losowo poziom nasłonecznienia i wysyła do serwera, ustawia odpowiednie natężenie światła na wszystkich lampach (domyślnie 10).

Domyślnie tworzone są 4 czujniki nasłonecznienia, 10 lamp zintegrowanych z czujnikami ruchu, a także jeden analogowy przełącznik.

Dane z poszczególnych czujników przekazywane są do serwera, który analizuje dane oraz uśrednia wartości z czujników nasłonecznienia, a następnie przekazuje je sterownikom poszczególnych lamp. Sterowniki lamp (***task body Lamp****)* natomiast na podstawie otrzymanych danych określają w jaki sposób mają świecić. W pierwszej kolejności biorą pod uwagę sygnały przełącznika, następnie, jeżeli jest wystarczająco ciemno kolejności i wystąpił jakiś ruch uruchamiana jest lampa powiązana z czujnikiem ruchu. Ostatnie w hierarchii ważności są dane z czujnika nasłonecznienia, które są ustawiane tylko wtedy, gdy jest dostatecznie ciemno i aktualnie nie został wykryty ruch. Schemat działania programu opiera się na architekturze klient-serwer i wymianie danych przez spotkania.

3. Opis struktury zadaniowej programu

- *driver.adb* – główny plik programu z procedurą ***Driver***, która odpowiada za rozpoczęcie symulacji wywołując poszczególne procedury. Ma nadany najwyższy priorytet przy kompilacji programu. W tym pliku można aktywować nowe czujniki/przełączniki jednak trzeba to zrobić ręcznie dopisując kod.
- *light_sensor.ads* – plik zawierający deklaracje poszczególnych procedur, zadań oraz typów danych związanych z czujnikami.
    - *subtype light_range* – nowy typ liczb zmiennoprzecinkowych z zakresu 0.0-100.0 służący do określenia poziomu natężenia światła
    - *subtype switch_range* – nowy typ liczb całkowitych z zakresu 0-2 definiujący pozycję przełącznika światła
    - *type Light_Sensor_Data* – typ rekordowy przechowujący dane wysłane przez czujnik nasłonecznienia (jego ID oraz poziom natężenia światła)
    - *type LSD_Ptr* – definuje dostęp do typu *Light_Sensor_Data*
    - *type Movement_Sensor_Data* – typ rekordowy przechowujący dane wysłane przez czujnik ruchu (jego ID oraz to czy wykrył ruch)
    - *type MSD_Ptr* – definuje dostęp do typu *Movement_Sensor_Data*
    - *type Switch_Sensor_Data* – typ rekordowy przechowujący dane wysłane przez przełącznik (jego ID oraz uruchomiony tryb)
    - *type SSD_Ptr* – definiuje dostęp do *Switch_Sensor_Data*
    - *task type Light_Sensor* – komunikacja pomiędzy zadaniami, zatrzymanie działania sensora nasłonecznienia
    - *task type Movement_Sensor* – komunikacja pomiędzy zadaniami, zatrzymanie działania sensora ruchu
    - *task type Switch_Sensor* – komunikacja pomiędzy zadaniami, zatrzymanie działania przełącznika i zmiana jego trybu
    - *task type Lamp* – zadanie pobierające dane z sensorów do lampy o podanym Id
    - *task type Lamp_Bufor* – definicja pomocniczego wątku dla czujnika ruchu.
    - *type Light_Sensor_Ptr, type Movement_Sensor_Ptr, type Switch_Sensor_Ptr*, *task type Lamp_Ptr oraz Lamp_Bufor_Ptr* – definiują dostęp do poszczególnych sensorów lamp oraz wątku lamp

- *package Light_Vector* **–** przechowuje instancję *Light_Sensor* (czujnika nasłonecznienia)
- *package Movement_Vector* **–** przechowuje instancję *Movement_Sensor* (czujnika ruchu)
- *package Switch_Vector* **–** przechowuje instancję *Switch_Sensor* (przełącznika)
- *package Solar_Power_Vector* – pomocniczy wektor przechowywujący ostatnich wartości otrzymanych z czujników nasłonecznienia
- *package Lamp_Vector* – przechowuje instancję *Lamp*
- *package LB_Vector* – przechowuje instancję *Lamp_Bufor*
- *package Id_Vector* – definicja wektora przechowującego Id, które jest już zajęte i nie może zostać przypisane nowemu czujnikowi. Tworzy osobne wektory dla poszczególnych typów czujników
- *task Serwer* – deklaracja zadania Serwer oraz pakietów, które uruchamia
- *procedure Add_Light_Sensor* – definicja procedury dodającej sensor
- *procedure Remove_Light_Sensor* – definicja procedury do usuwania sensora o zadanym Id
- *procedure Run* – definicja procedury generującej nową symulację
- *task Panel_Thread* – tworzy zadanie odpowiedzialne za panel wyświetlający dane w konsoli

➢ *light_sensor.adb* – plik zawierający implementację poszczególnych zadań oraz procedur, które zostały zdeklarowane w light_sensor.ads
- *task body Light_Sensor* – na potrzeby symulacji generuje w sposób losowy poziom nasłonecznienia i przekazuje dane do serwera
- *task body Movement_Sensor* – generuje sygnał oznaczający ruch w pomieszczeniu i przekazuje dane do serwera wraz ze swoim id
- *task body Switch_Sensor* – wysyła do serwera o trybie w jakim znajduje się włącznik
- *task body Lamp* – pobiera dane z przełącznika, sensorów ruchu oraz nasłonecznienia oraz odpowiednio ustawia parametry lampy (zgodnie z opisem w pkt. 2) aktualizując informacje wyświetlane w konsoli.
- *task body Lamp_Bufor* – pomocniczy bufor dla czujnika ruchu, który wysyła sygnał o końcu ruchu po 20 sekundach od ostatnio wysłanego sygnału ruchu
- *task body Serwer* – zbiera dane z poszczególnych czujników, wylicza średnią natężenia oświetlenia oraz wysyła odpowiednie wartości do wątków odpowiadających za poszczególne lampy
- *procedure Add_Light_Sensor* – tworzy nowy czujnik nasłonecznienia o najniższej wartości Id, która jest wolna na liście *Taken_L_Id*
- *procedure Remove_Light_Sensor* – usuwa czujnik nasłonecznienia oraz zwalnia jego Id

- *procedure Add_Lamp* – dodaje kolejny element oświetlenia z najmniejszym dostępnym Id
- *procedure Remove_Lamp* – usuwa lampę o zadanym Id
- *procedure Add_Switch_Sensor* – tworzy nowy przełącznik o najniższej wartości Id, która jest wolna na liście *Taken_S_Id*
- *procedure Remove_Switch_Sensor* – usuwa czujnik oraz zwalnia jego Id
- *procedure Panel_Thread* – tworzy osobny wątek, który uruchamia procedurę Run Panelu, który wyświetla wyniki symulacji w konsoli
- *procedure Run* – uruchamia, panel, serwer oraz dodaje zadaną ilość czujników nasłonecznienia, ruchu oraz jeden przełącznik

➢ *panel.ads* – plik zawierający deklaracje poszczególnych procedur, zadań oraz typów danych związanych z wyświetlaniem wyników w konsoli
- *protected Ekran* – definicja obiektu odpowiedzialnego za wyświetlanie na ekranie. Zawiera on procedury:
  - *procedure Pisz_XY*
  - *procedure Czysc*
  - *procedure Tlo*
- *procedure Update* – inicjalizacja procedury odświeżającej wyniki w konsoli.

➢ *panel.adb* – plik zawierający implementację poszczególnych zadań oraz procedur, które zostały zdeklarowane w panel.ads
- *protected body Ekran* – obiekt implementujący ekran, który aktualizuje się i wyświetla na bieżąco dane pochodzące z symulacji
  - *function Atryb_Fun*
  - *function Esc_XY* – procedura tworząca ekran o odpowiedniej szerokości.
  - *procedure Pisz_XY* – procedura odpowiedzialna za tworzenie kolejnych wierszy w konsoli
  - *prodedure Czysc* – procedura czyszcząca cały ekran
  - *procedure Tlo* – procedura wyświetlająca nagłówek
- *procedure Update* – procedura uaktualniająca wyświetlane dane o natężeniu światła poszczególnych lamp
- *procedure Run* – inicjuje cały panel wyświetlający wyniki symulacji

4. Informacje o stosowanych pakietach zewnętrznych

- *Ada.Text_IO* – pakiet do wczytywania oraz wypisywania informacji w konsoli
- *Ada.Containers.Vectors* – pakiet umożliwiający korzystanie z typów wektorowych
- *Ada.Numerics.Float_Random* – pakiet umożliwiający wygenerowanie losowych liczb zmiennoprzecinkowych
- *Ada.Numerics.Elementary_Functions*
- *Ada.Strings* – pakiet, dzięki któremu możemy korzystać z łańcuchów znaków
- *Ada.Strings.Fixed* – pakiet umożliwiający zastosowanie funkcji Trim

5. Instrukcja obsługi

Symulator wystarczy skompilować za pomocą komendy *gnatmake driver.adb***,** a następnie uruchomić go *./driver.*

Aby zmienić ilość aktywowanych czujników nasłonecznienia lub ilość lamp, należy ręcznie zmienić zakres działania pętli w procedurze *Run* w pliku *Light_Sensor.adb* (domyślnie tworzone jest 10 lamp połączonych z czujnikami ruchu, 4 czujniki nasłonecznienia oraz jeden przełącznik).

6. Możliwe rozszerzenia programu

Ciekawym pomysłem było by zrealizowanie stref oświetlenia. Dzięki nim możliwe było by dowolne łączenie czujników ruchu, a także przełączników z odpowiednimi lampami.

Można również dodać interfejs graficzny wraz z panelem administracyjnym, który ułatwiałby dodawanie, usuwanie, rozmieszczanie czujników, a także zaobserwowanie efektu końcowego ich działania.

7. Ograniczenia programu

Ograniczeniem symulatora oświetlenia jest jedynie moment, w którym pamięć zostanie zapełniona zbyt dużą ilością czujników oraz danych, które są przez nie przekazywane, jednak jest to mało prawdopodobna sytuacja. Za ograniczenie można również uznać konieczność zmieniania ilości czujników bezpośrednio w kodzie programu, zamiast poprzez panel administracyjny, jednak nie to było celem niniejszego projektu.