Projekt kompetencyjny

Sem. VI Informatyka WEEIA

Półautonomiczny robot inspekcyjny

Analiza LLD

Spis treści

Wymagania 3

Dziennik zdarzeń (event log) 4

Budowa 4

Konstruktor klasy ( \_\_init\_\_() ) 4

Dodawanie zdarzenia ( traceAdd() ) 4

Przykładowy log 4

Prototyp kodu: 5

Obsługa ramek ruchu 6

Współdziałanie 6

Dziennik zdarzeń 6

Wysyłanie ramek 6

Budowa ramki ruchu 6

Schemat budowy 6

# Wymagania

* Poruszanie się za pomocą dedykowanej aplikacji
* Poruszanie się za pomocą wgranej trasy
* Omijanie przeszkód
* Przesyłanie danych z urządzeń pomiarowych
* System zmiany urządzeń pomiarowych
* Dedykowana aplikacja

# Dziennik zdarzeń (event log)

Klasa obsługująca dziennik zdarzeń powinna być tworzona w aplikacji głównej robota, oraz aplikacji głównej terminala sterującego. Dziennik zdarzeń powinien być tworzony jako pierwszy, już przy inicjalizacji klasy. Obiekt dziennika powinien być przekazywany w konstruktorze wszystkich klas używanych w tym projekcie.

## Budowa

### Konstruktor klasy ( \_\_init\_\_() )

Konstruktor powinien tworzyć plik o nazwie zadanej w parametrach konstruktora. Jeśli próba utworzenia pliku się nie powiedzie, powinien zostać wypisany wyjątek z informacji o błędzie z powodu którego nie udało się utworzyć pliku.

Konstruktor zapisuje też w polach klasy ścieżkę do pliku oraz nazwę pliku.

### Dodawanie zdarzenia ( traceAdd() )

Podstawową funkcjonalnością metody dodawania zdarzenia jest utworzenie łańcucha znaków w formacie:

XXXX: <wiadomość>, gdzie:

XXXX – miejsce z którego przyszła wiadomość, w formie łańcucha znaków, przekazywana przez klasę wysyłającą wiadomość. Obowiązkowo dużymi literami.

<wiadomość> - łańcuch znaków przekazywany przez klasę wysyłającą wiadomość do dziennika zdarzeń.

Dodanie zdarzenia jest realizowane przez otwarcie dokumentu w trybie nadpisywania (‘a’), oraz zamknięcie dokumentu po dodaniu wiadomości.

Otwieranie i zamykanie dokumentu w momencie dodawania wiadomości, ma umożliwić podgląd zdarzeń systemu w czasie rzeczywistym.

## Przykładowy log

CONN\_HNDL: Connection handler, welcome!

CONN\_HNDL: Own ip: 192.168.0.103 network: 192.168.0.

CONN\_HNDL: Adres: 192.168.0.105 port: 80 nazwa: 192.168.0.105

CONN\_HNDL: Adres: 192.168.0.105 port: 139 nazwa: 192.168.0.105

CONN\_HNDL: IP ADRESSES found: {('192.168.0.105', 80), ('192.168.0.105', 139)}

CONN\_HNDL: End of scanning!

FRAME\_HNDL: Frame handler welcome!

MAIN: Main client welcome

CONN\_HNDL: Conn not estabished for: ('192.168.0.105', 80)

CONN\_HNDL: Conn not estabished for: ('192.168.0.105', 139)

CLI: Command Line Client Welcome!

CLI: EXIT: ['q']

## Prototyp kodu:

class traceHndl:  
 #variables  
 traceLog = 0  
  
 #constructor  
 def \_\_init\_\_(self):  
 try:  
 self.traceLog = open("log.txt","w") #creating file log.txt  
 self.traceLog.close()  
 except:  
 print("ERROR: unable to open log file")  
  
 def traceAdd(self,traceName,traceBuff): #adding trace to log.txt  
 self.traceLog = open("log.txt","a")  
 self.traceLog.write(str(traceName)+": "+str(traceBuff)+"\n")  
 self.traceLog.close()

# Obsługa ramek ruchu

Obsługa ramek ruchu robota odbywa się za pomocą klasy dedykowanej do tworzenia, wysyłania, weryfikacji i dekodowania ramek ruchu.

## Współdziałanie

### Dziennik zdarzeń

Obiekt dziennika zdarzeń powinien być przekazywany przez parametr konstruktora. W konstruktorze powinien nastąpić pierwszy zapis do dziennika zdarzeń, potwierdzający poprawną inicjalizację obiektu obsługującego ramki ruchu.

#### Ograniczenia

Z powodu dużej liczby informacji przetwarzanych przez ten obiekt, zapis do dziennika zdarzeń powinien odbywać się zawsze w przypadku sytuacji nie normalnych z punktu widzenia działania całego systemu, jednak, możliwe jest włączenie zapisu do dziennika dodatkowych informacji, w celu ewentualnego złapania i poprawienia błędu w działaniu programu.

### Wysyłanie ramek

Brak oddzielnego interfejsu do przesyłania ramek. Przesyłanie będzie odbywać się za pomocą gettera zwracającego liczbę typu unsigned int.

## Budowa ramki ruchu

### Schemat budowy

| 1110 | XXXX | 0111 |

| nagłówek | treść | zakończenie |

### Pola ramki

#### Nagłówek

Nagłówek składa się z 4 bitów gdzie wszystkie oprócz najmłodszego bitu, ustawione na wartość 1. Najmłodszy bit ustawiony na 0. Takie ustawienie odpowiada liczbie 3584. Z ramki o prawidłowo ustawionym nagłówku po operacji ‘i’ bitowego (bitwise and) z liczbą 3840 powinien wyjść wynik 3584. Każdy inny wynik wskazuje na błąd w nagłówku ramki.

#### Treść

Treść składa się z czterech bitów, gdzie:

Bit nr 1 (najstarszy) – odpowiada za ruch do przodu

Bit nr 2 – odpowiada za ruch w prawo

Bit nr 3 – odpowiada za ruch w lewo

Bit nr 4 (najmłodszy) – odpowiada za ruch do tyłu

#### Zakończenie

Zakończenie składa się z 4 bitów gdzie wszystkie oprócz najmłodszego bitu, ustawione są na wartość 1. Najmłodszy bit ustawiony jest na 0. Takie ustawienie odpowiada liczbie 7. Z ramki o prawidłowo ustawionym zakończeniu, po operacji ‘i’ bitowego z liczbą 16 powinien wyjść wynik 7. Każdy inny wynik wskazuje na błąd w nagłówku ramki.