

	Akademia Górniczo-Hutnicza im. St. Staszica w Krakowie
	Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej
	Katedra Automatyki i Inżynierii Biomedycznej
	LABORATORIUM AUTOMATYKI POJAZDOWEJ
<p style="text-align: center;"><u>Laboratorium nr 3</u> Model układu sterującego dla świateł zewnętrznych w samochodzie</p>	

1. Cele ćwiczenia

Celem ćwiczenia jest zapoznanie się z podejściem modelowym (z ang. *model-based design*) do projektowania elektronicznych systemów sterowania w branży motoryzacyjnej.

2. Wymagane kwalifikacje osób realizujących ćwiczenie

- Znajomość środowiska MATLAB/Simulink/Stateflow
- Znajomość języka angielskiego
- Umiejętność czytania schematów elektrycznych
- Znajomość ogólnej zasady działania głównych elementów sterujących w samochodzie

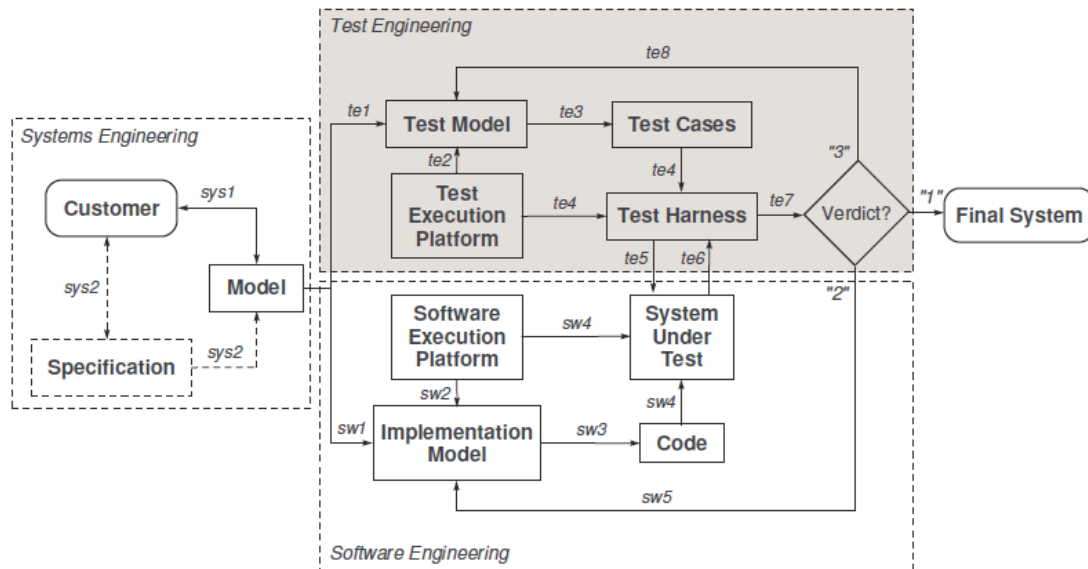
3. Opis stanowiska laboratoryjnego

Do wykonania ćwiczenia jest potrzebny komputer PC z zainstalowanym pakietem MATLAB oraz przybornikami: Simulink i Stateflow.

4. Wymagane informacje do realizacji ćwiczenia

Wymagania funkcjonalne dla układów automatyki samochodowej są najczęściej opisywane za pomocą równań matematycznych, maszyn stanów, charakterystyk statycznych i dynamicznych, itp. Jeżeli opis działania systemu jest przedstawiony w innej formie (np. tekstowej), to zadaniem inżyniera systemowego jest przekształcenie tego opisu do postaci abstrakcyjnej za pomocą odpowiedniego aparatu matematycznego. W ten sposób dostajemy model matematyczny funkcjonalności projektowanego urządzenia, który następnie można wykorzystać w symulacjach komputerowych do weryfikacji, kalibracji i optymalizacji działania systemu. Dalsze czynności inżynierskie polegają na

zaprojektowaniu i zbudowaniu fizycznego urządzenia, które będzie implementować funkcjonalność układu określoną przez model matematyczny.



Rys. 1. Podejście oparte na modelach do projektowania wbudowanych systemów sterowania

Tworzenie wbudowanego układu regulacji wymaga dokładnego przeanalizowania wymagań, czyli w tym przypadku modelu matematycznego, opracowania odpowiedniej architektury, stworzenia oprogramowania, testowania oprogramowania i całego systemu. Każdy błąd czy zlekceważenie pewnych wymagań we wstępnych fazach projektu może wydłużyć czas trwania projektu, może prowadzić do zwiększenia kosztów realizacji projektu czy też nawet może prowadzić do niepowodzenia całego przedsięwzięcia. Złe decyzje projektowe pociągają za sobą często konieczność modyfikacji nie tylko oprogramowania, lecz także sprzętu. W przypadku systemów do zastosowań krytycznych i związanych z bezpieczeństwem pojawia się też problem wiarygodności, czyli odporności systemu na awarie sprzętu, niespodziewane warunki pracy, zakłócenia elektromagnetyczne, błędy implementacyjne, itd. Badania pokazują, że w zależności od roli jaką ma spełniać urządzenie, 30 do 90% czasu poświęca się na czynności związane z testowaniem danego urządzenia.

Model reprezentujący funkcjonalne działanie systemu może być uzupełniony o wymagania związane z systemem operacyjnym, architekturą oprogramowania oraz platformą sprzętową, na której to oprogramowanie ma być wykonywane. W efekcie czego powstaje tak zwany model implementacyjny, który może być poddany transformacji aby automatycznie uzyskać kod źródłowy. W odróżnieniu od tradycyjnego podejścia, gdzie kod jest pisany ręcznie przez programistów, podejście oparte na modelach pozwala na szybszą i jakościowo lepszą implementację i pielęgnację systemu. Podobny schemat postępowania można zastosować do weryfikacji systemu.

Tab. 1. Opis aktywności w procesie projektowania wbudowanych systemów sterowania opartym na modelach

Obszar inżynierski	Aktywność	Opis
Inżyniera systemowa	<i>sys1</i>	analiza wymagań i przekształcenie ich do modelu
	<i>sys2</i>	budowa modelu na podstawie opisowej specyfikacji (opcjonalnie)
Inżyniera oprogramowania	<i>sys2</i>	analiza wymagań i utworzenie opisowej specyfikacji (opcjonalnie)
	<i>sw1</i>	utworzenie modelu implementacyjnego
	<i>sw2</i>	uzupełnienie modelu o ograniczenia związane ze sprzętowym środowiskiem uruchomieniowym
	<i>sw3</i>	automatyczna generacja kodu źródłowego
	<i>sw4</i>	integracja kodu ze sprzętowym środowiskiem uruchomieniowym
Inżyniera testowa	<i>sw5</i>	naniesienie poprawek na model
	<i>te1</i>	utworzenie modelu testowego
	<i>te2</i>	uzupełnienie modelu o ograniczenia związane z systemem testowym
	<i>te3</i>	automatyczna generacja przypadków testowych
	<i>te4</i>	utworzenie zestawu przypadków testowych i ich integracja z systemem testowym
	<i>te5</i>	wykonanie zestawu przypadków testowych
	<i>te6</i>	generowanie raportu z testów
	<i>te7</i>	ocena rezultatów z przeprowadzonych testów
	<i>te8</i>	naniesienie poprawek na model
	„1”	prawidłowe działanie systemu
	„2’	błąd w oprogramowaniu
	„3”	błąd w procedurze testowania

Weryfikacja to działanie polegające na sprawdzaniu czy powstały układ spełnia wyspecyfikowane wymagania. Innymi słowy, weryfikacja układów automatyki będzie polegać na odpowiednim dobieraniu funkcji sterujących oraz warunków początkowych, tak aby zbudowany układ fizyczny był jak najbardziej ”podobny” w zachowaniu do jego reprezentacji modelowej. Głównym elementem procesu weryfikacji jest tworzenie tzw. przypadków testowych. Przypadek testowy to zbiór wejść, warunków wykonania oraz oczekiwanych wyników utworzony, aby wykonać określoną ścieżkę w programie lub w modelu albo aby zweryfikować zgodność z określonym wymaganiem. Najważniejszymi elementami każdego przypadku testowego jest odpowiednia stymulacja systemu, czyli dobór funkcji sterujących oraz warunków początkowych oraz pomiar i ocena poprawności produkowanych przez system wyników. Do oceny produkowanych przez system wyników w praktyce wykorzystuje się takie urządzenia jak analizatory stanów logicznych, oscyloskopy, mierniki, emulatory sprzętowe, wbudowane programy śledzące, itp. W przypadku systemów dynamicznych sprawa się trochę komplikuje, gdyż analiza poprawności funkcji wyjścia systemu musi odbywać się w sposób ciągły w jakimś z góry określonym przedziale czasowym. Niewątpliwie istotnym elementem jest też budowa środowiska testowego, za pomocą którego będzie można wykonywać przygotowane przypadki testowe. Analiza wiarygodności oprogramowania czy też całego systemu jest przedmiotem wielu prac badawczych i nie opracowano do tej pory dobrej metodyki postępowania. O ile metody weryfikacji systemów statycznych są dosyć dobrze

opracowane, to weryfikacja systemów, w których dynamika odgrywa kluczową rolę stanowi duże wyzwanie zarówno dla naukowców jak i inżynierów.

Rys. 1 wraz z tab. 1 przedstawiają kluczowe elementy i aktywności składające się na podejście oparte na modelach do projektowania wbudowanych systemów sterowania.

3. Przebieg ćwiczenia

- A. Tematem laboratorium jest przygotowanie modelu w środowisku MATLAB/Simulink/Staflow na podstawie dostarczonych wymagań.
- B. Model ma symulować działanie układu elektronicznego odpowiedzialnego za oświetlenie zewnętrzne w samochodzie.
- C. Model powinien być tak skonstruowany, aby możliwa była zmiana sygnałów wejściowych podczas symulacji.
- D. Poglądowy schemat modułu sterowania dla świateł zewnętrznych w samochodzie jest przedstawiony na rys. 2.
- E. Sygnały wejściowe (w nawiasach podano możliwe wartości poszczególnych sygnałów):

Front Fog Switch (on, off)

Rear Fog Switch (on, off)

Low Beam (on, off)

Park Lights (on, off)

Turn Left (on, off)

Turn Right (on, off)

Battery Voltage (0.0-26.0V)

Key Position (off, accessory, run, crank)

Gear Position (reverse, 1st_gear, 2nd_gear, 3rd_gear, 4th_gear, 5th_gear, neutral)

Brake Pedal (on, off)

Hazard Switch (on, off)

- F. Sygnały wyjściowe (w nawiasach podano możliwe wartości poszczególnych sygnałów):

Hazard LED (on, off)

Front Fog Output (active, inactive)

High Beam Output (active, inactive)

Low Beam Output (active, inactive)

Park Left Output (active, inactive)

Park Right Output (active, inactive)

Licence Plate Output (active, inactive)

Stop Light Output (active, inactive)

Rear Fog Output (active, inactive)

Reverse Lights Output (active, inactive)

Turn Left Output (active, inactive)

Turn Right Output (active, inactive)

- G. General requirements

R03A.01:

The operational voltage range is 6V – 18V. Below and above this range all functions shall not work.

H. Turn indicators

R03B.01:

Left turn indicators shall blink (frequency 1.5Hz, duty cycle 50%) when (*Key Position* == “run” OR “crank”) AND (*Turn Left* == “on”).

R03B.02:

Right turn indicators shall blink (frequency 1.5Hz, duty cycle 50%) when (*Key Position* == “run” OR “crank”) AND (*Turn Right* == “on”).

I. Hazard warning function

R03C.01:

The hazard warning is requested by momentary switch. This function shall always be available, independently on *Key Position*.

R03C.02:

When the function is active, left and right turn indicators shall blink with the frequency 1.5Hz and duty cycle 50%.

R03C.02:

Hazard LED is used for indicating activation of hazard warning function. LED shall flash synchronously to the turn lights.

J. Stop lights

R03D.01:

Stop lights (*Stop Light Output*) shall be activated when *Brake Pedal* == “on”. Otherwise they shall be deactivated. The function shall always be available, independently on *Key Position*.

K. High beam function

R03E.01:

High beam function (*High Beam Output*) shall be activated when (*High Beam* == “on”) AND (*Key Position* == “run”) AND (*Low Beam* == “on”). Otherwise, it shall be deactivated.

L. Flash to pass function

R03F.01:

Flash to pass function is active in case of (*Flash To Pass* == “on”) AND (*Key Position* == “run”). The flash to pass function shall activate *High Beam Output*.

M. Reverse lighting

R03G.01:

Reverse lights (*Reverse Lights Output*) shall be switched on when (*Key Position* == “run”) AND (*Gear Position* == “reverse”). Otherwise, the lights shall be switched off.

N. Front fog lights

R03H.01:

The front fog function shall control front fog lights (*Front Fog Output*).

R03H.02:

Front Fog Output shall be “active” when ((*Key Position* == “run”) OR (*Key Position* == “crank”)) AND (*Front Fog Switch* changes from “off” to “on”) AND

((*Low Beam* == “on”) OR (*Park Lights* == “on”))
Otherwise, the output shall be “inactive”.

R03H.03:

The front fog function is requested by momentary switch.

O. Rear fog lights

R03I.01:

The rear fog function shall control rear fog light (*Rear Fog Output*).

R03I.02:

Rear Fog Output shall be “active” when

((*Key Position* == “run”) OR (*Key Position* == “crank”)) AND

(*Rear Fog Switch* changes from “off” to “on”) AND

(*Low Beam* == “on”)

Otherwise, the output shall be “inactive”.

R03I.03:

The rear fog function is requested by momentary switch.

P. Low beam function

R03J.01:

Low Beam Output shall be driven with PWM (frequency 100Hz, duty cycle 80%) when the low beam function is activated.

R03J.02:

The following outputs shall be activated as long as low beam function is activated:

Low Beam Output

Park Left Output

Park Right Output

R03J.03:

Low beam function shall be activated when *Low Beam* == “on”. The function shall always be available, independently on *Key Position*.

Q. Park lights

R03K.01:

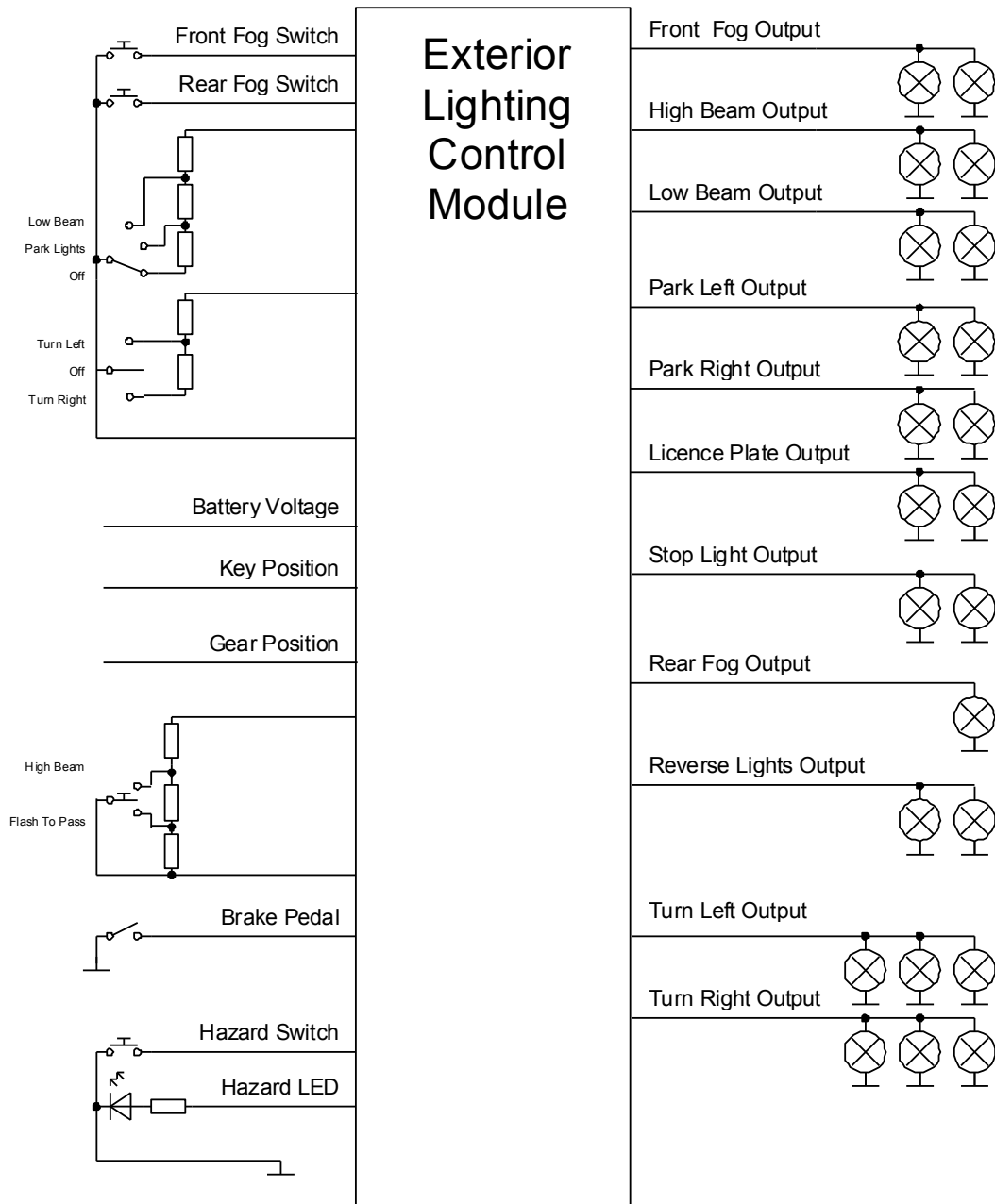
The parking lights shall be active as long as *Park Lights* == “on”. The function shall always be available, independent of *Key Position*.

R03K.02:

The following outputs shall be activated as long as park lights function is activated:

Park Left Output

Park Right Output



Rys. 2. Moduł sterowania oświetleniem zewnętrznym w samochodzie

4. Analiza i opracowanie wyników

Wyniki projektu należy przedstawić za pomocą prezentacji zawierającej następujące elementy:

- Informacje o zespole realizującym projekt;
- Sformułowanie problemu;

- Sposób rozwiązania problemu zawierający strukturę modelu;
- Wyniki przeprowadzonych symulacji, testów i eksperymentów.
- Wnioski.

5. Literatura

- [1] Broy, M., Krcmar, H., Zimmermann, J., Kirstan, S. *EETimes Europe: Model-based software development in the automotive industry*. <http://www.electronics-eetimes.com/en/model-based-development.html>, 2011.
- [2] Nicolescu, G., Mosterman, P.J. *Model-based design for embedded systems*. CRC Press, Boca Raton, London, New York, 2010.
- [3] Zander, J., Schieferdecker, I., Mosterman, P.J. *Model-based testing for embedded systems*. CRC Press, Boca Raton, London, New York, 2011.
- [4] Zander-Nowicka, J.. *Model-based testing of embedded systems in the automotive domain*. PhD thesis, Technical University Berlin, 2009.