|  |  |
| --- | --- |
| etf_logo.jpg | **Електротехнички факултет, Универзитет у Београду**  **Катедра за рачунарску технику и информатику** |
|  |  |

**Пројекат из предмета**

**Програмирање у реалном времену**

професор: др. Драган Милићев

асистент: Бојан Фурлан

студент: Милан Бранковић

3129/11

bm113129m@student.etf.rs

Београд, 2012

Садржај

[1 Опис проблема 3](#_Toc327880284)

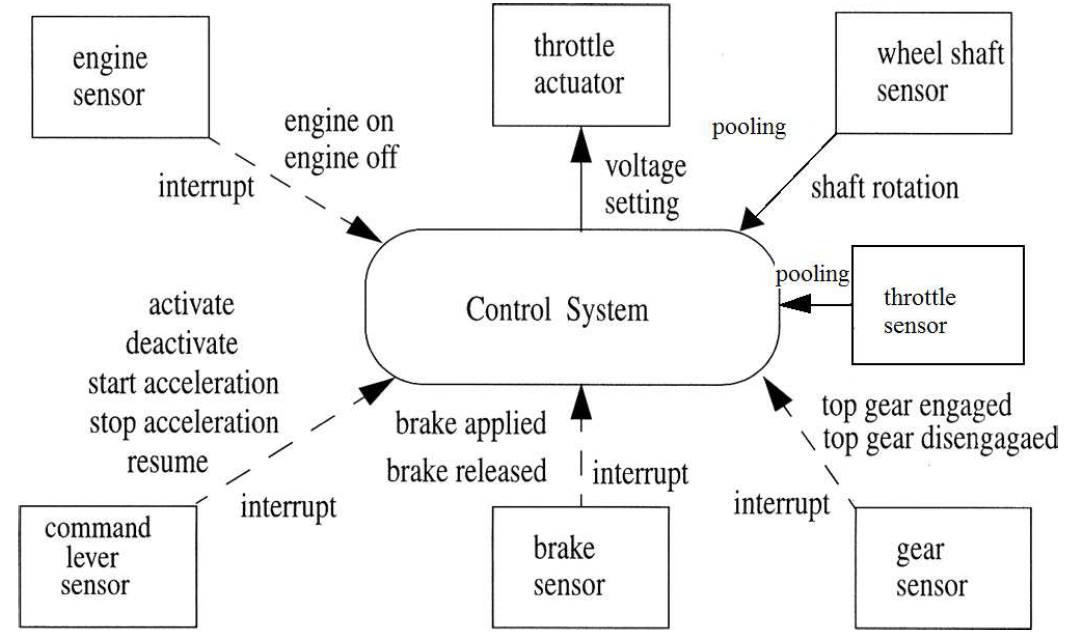
[Задатак 1 – RT UML модел 3](#_Toc327880285)

[Задатак 2 – Имплементација 5](#_Toc327880286)

[Задатак 3 – Прорачун распоредивости 6](#_Toc327880287)

1 Опис проблема

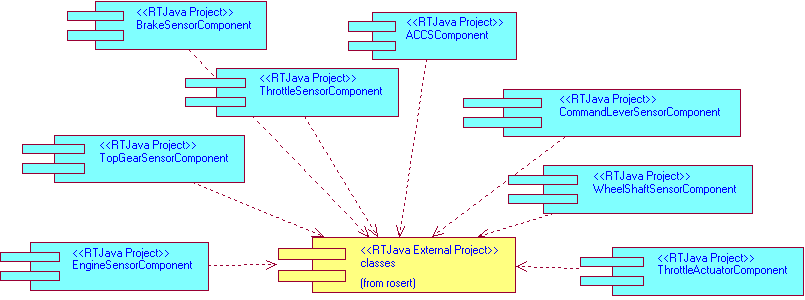
Систем за аутоматску регулацију брзине аутомобила (енг. *Automobile Cruise Control System* - ACCS) треба да аутоматски одржава брзину возила задату од стране возача. Релације из-међу управљачког система и спољних уређаја приказане су на следећој слици.

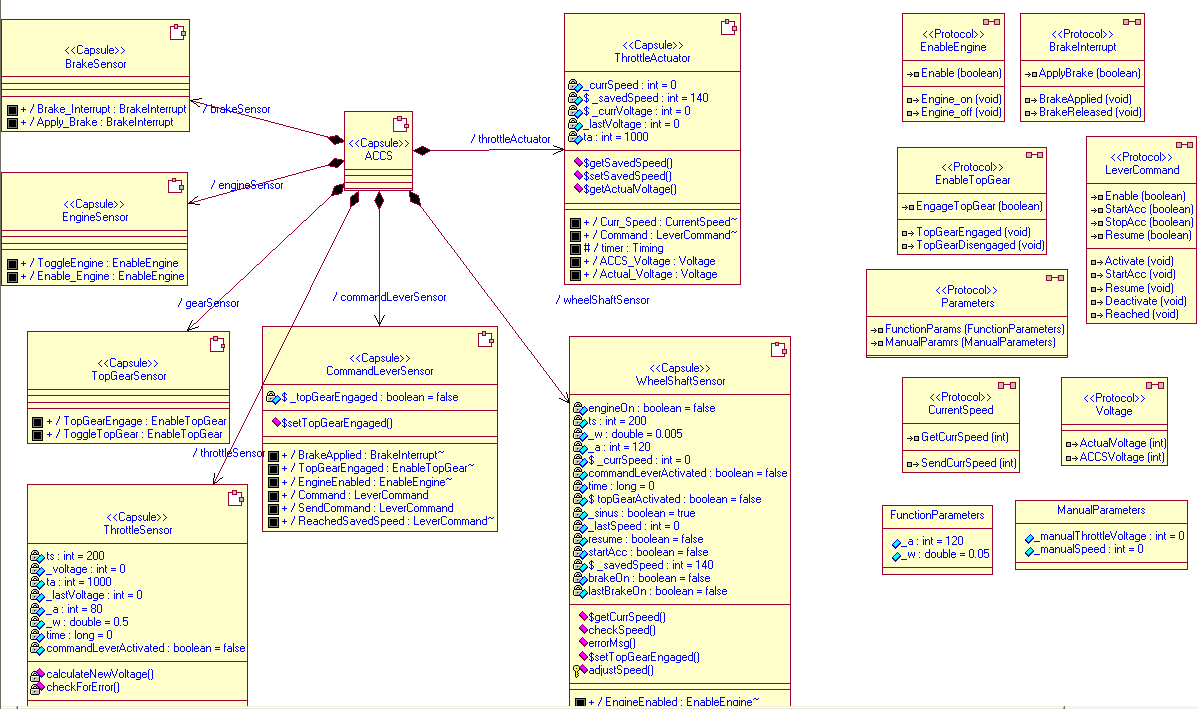


## Задатак 1 – RT UML модел

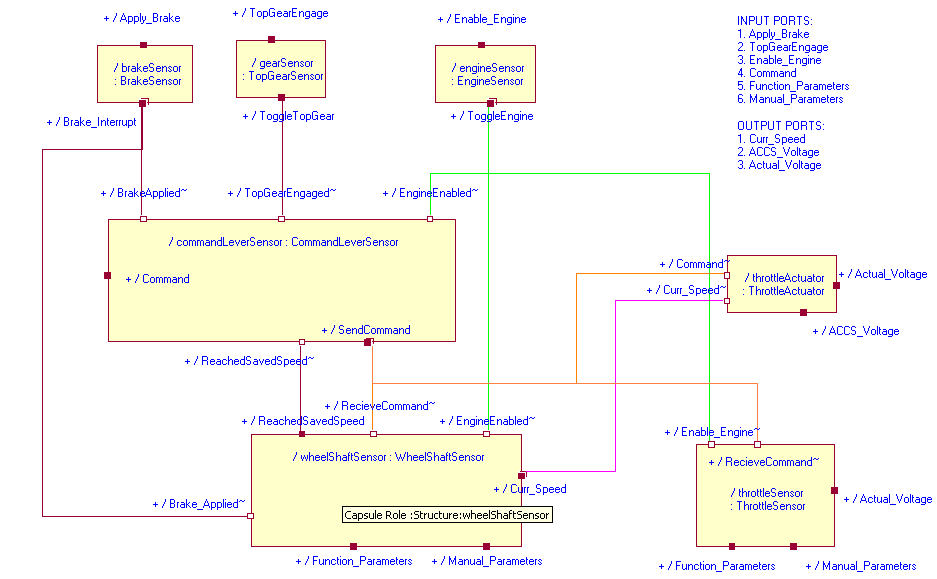
Помоћу алата Rational Rose Real Time моделовати дати систем.

*Решење:*

Слика 1. Component Diagram



Слика 2. Class Diagram

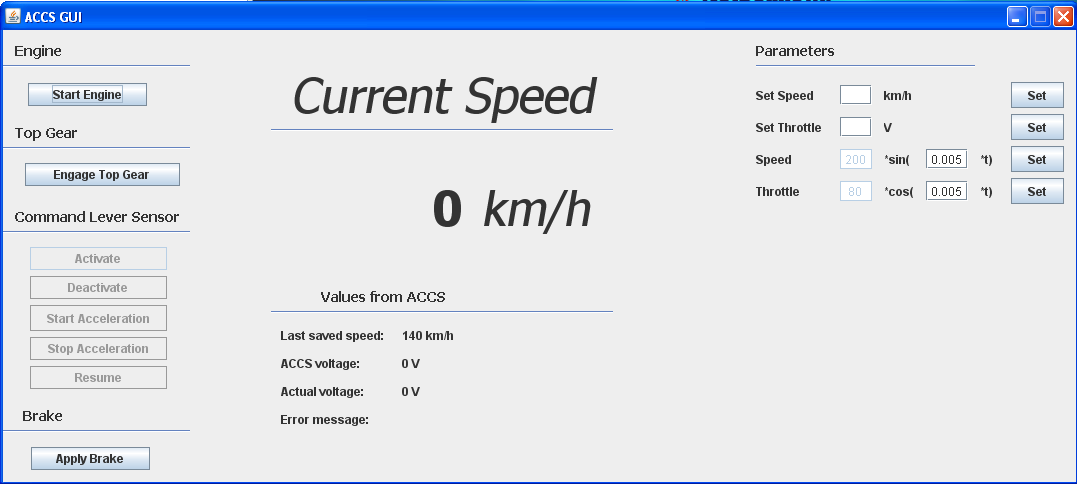
Слика 3. Structure Diagram

## Задатак 2 – Имплементација

Потребно је у потпуности имплементирати горе наведени софтверски систем.

Изглед корисничког интерфејса и ниво његове информативности и интуивности треба да одреди студент. У сваком случају, кориснички интерфејс мора да буде интерактиван.

*Решење:*



Слика 4. Изглед корисничког интерфејса

/\*

\* methods with which ACCS update GUI

\*/

private void setCurrentSpped(int speed) {

lblCurrSpped.setText("" + speed);

}

private void setSavedSpped(int speed) {

lblLastSavedSpeed.setText("" + speed + " km/h");

}

private void setACCSVoltage(int voltage) {

double v = (double) voltage / 10.0;

lblACCSVoltage.setText("" + v + " V");

}

private void setActualVoltage(int voltage) {

double v = (double) voltage / 10.0;

lblActualVoltage.setText("" + v + " V");

}

private void setErrorMsg(String msg) {

lblErrMsg.setText(msg);

}

private void clearErrorMsg() {

lblErrMsg.setText("");

}

/\*

\* methods with which GUI informs ACCS

\*/

Форма accs.port.signal.send();

## Задатак 3 – Прорачун распоредивости

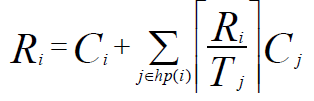
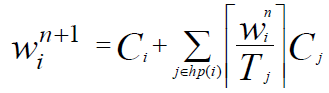
Потребно је извршити процену времена извршавања у најгорем случају за процесе у сис-тему реализованом у Задатку 2 на основу теста распоредивости за FPS заснованог на вре-мену одзива (енг. *response time*). Такође, на основу уведених претпоставки потребно је од-редити период T и временски рок D за сваки процес.

Време извршавања у најгорем случају сваког од идентификованих процеса треба што пре-цизније измерити на реализованој платформи.

*Решење:*

Распоређивање на основу фиксних приоритета (енг. *Fixed-Priority Scheduling*, FPS): сваки процес има свој фиксни, статички (пре извршавања) одређен приоритет. За извршавање се увек изабира онај спреман процес који има највиши приоритет.У RT системима приори-тети процеса су последице њихових временских карактеристика, односно временских ограничења, а не њиховог значаја за исправну функионалност система или његов интег-ритет.

Постоји један веома једноставан, али оптималан начин за доделу приоритета периодичним процесима по FPS шеми, тзв. додела приоритета по учестаностима (енг. *Rate- Monotonic Priority Ordering*, RMPO): периодичним процесима се додељују јединствени приоритети, монотоно уређени према учестаности процеса, тако да процес са краћом периодом има ви-ши приоритет, односно за свака два процеса i и j важи Ti < Tj => Pi > Pj, где је приоритет цео број и већа вредност означава виши приоритет.

Тест распоредивости за FPS заснован на времену одзива има две фазе: у првој фази се ана-литичким путем израчунава време одзива у најгорем случају за сваки процес. Време одзи-ва је време које протекне од тренутка активације (енг. *release time*) периодичног (тренутак почетка периоде) или спорадичног процеса (спољашњи догађај), до тренутка завршетка његовог извршења. У другој фази се проверава да ли је то време краће од задатог времен-ског рока (R ≤ D). Код FPS распоређивања, време одзива најприоритетнијег процеса јед-нако је његовом времену извршавања (R = C). Сви остали процеси ће трпети интерфе-ренцију. Интерференција представља време проведено у извршавању процеса вишег приоритета, док је посматрани процес спреман за извршавање. У општем случају, за про-цес *i* важи Ri = Ci + Ii. Коначна једначина изгледа овако:. Иако је формула тачна, вредност интерференције се не зна, јер се не зна ни Ri. Ова једначина има Ri на обе стране, али ју је тешко решити због функције горњег заокруживања. У општем случају може постојати много вредности Ri које задовољавају једначину. Најмања таква вредност представља време одзива у општем случају. Најједноставнији начин за решавање наведене једначине јесте формирање рекурентне релације

Спорадични процеси нису узети у разматрање код претходне анализе. Тако да се модел проширује да укључије и спорадичне процесе. Величина Т се за спорадичне процесе узима тако да представља минимално време између суседних појављивања. Иако спорадични процес у стварности може да се појави знатно ређе, анализа времена одзива ће дати резул-тат за најгори случај. Други елемент везан за спорадичне процесе јесте појам временског рока D. У једноставном моделу узима се да је D = T. Међутим, ово је често неприхватљи-во. Они се можда извршавају ретко, али су веома хитни. Зато модел мора да дозволи пос-тојање D < T. Тест распоредивости сасвим добро функционише и за ово ограничење, све док је услов заустављања *win+1* < *Di*испуњен. Он такође ради и за било који редослед приоритета. На жалост, веома често је време извршавања спорадичних процеса у најгорем случају много веће од просечног времена извршавања.

Као што је горе показано за систем процеса за D = T и FPS шемом распоређивања, RMPO је оптималан са друге стране у случају постојања спорадичних процеса, потребно је до-пустити и случај D < T. За систем процеса D < T постоји сличан приступ који је такође оптималан – монотоно уређење приоритета према временском року (енг. *Deadline Mono-tonic Priority Ordering*, DMPO). Код ове шеме, фиксни, статички приоритети се додељују процесима према њиховом временском року тако да важи Di < Dj => Pi > Pj.

*Теорема*: (о оптималности DMPO) Ако је дати скуп процеса распоредивпомоћу неке (било које) preemptive FPS шеме, онда је он сигурно распоредив и помоћу DMPO шеме.

Приоритете додељујемо према DMPO шеми, и радимо анализу за тако додељене приори-тете.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sensors | Periodic/Sporadic | T | D | P | C | R |
| Throttle | P | 200 | 30 | 6 | 5 | 5 |
| Wheel Shaft | P | 200 | 30 | 5 | 14 | 19 |
| Engine | S | 500 | 200 | 1 | 17 | 178 |
| Brake | S | 500 | 200 | 4 | 13 | 51 |
| Gear | S | 500 | 200 | 2 | 10 | 142 |
| Command Lever | S | 500 | 200 | 3 | 24 | 113 |

Из табеле се види да је за сваки процес испуњен услов R<D, стога је овај скуп процеса распоредив. Треба напоменути да је овај скуп процеса распоредив и помоћу FPS шеме засноване на искоришћењу.