SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

SEMINARSKI RAD

# DETEKCIJA KVARA NA PROPULZORU RONILICE

Jerko Tolić

Zagreb, svibanj 2015.

# Sadržaj

# 1. UVOD

Tijekom duljeg korištenja povećava se mogućnost kvara na propulzorima ronilice, te je potrebno razviti binarni detektor kvara koji će detektirati kada propulzor više ne radi. Detektor kvara razvija se u *Matlab*-u i ROS-u na temelju već postojećih mjerenja napona i struja na propulzorima ronilice. Zatim se testira na površinskom vozilu gdje je moguće selektivno gasiti motore i time simulirati kvar. Kvar na propulzoru ronilice detektiramo onda kada pri promjeni upravljačkog napona struja armature ne mijenja stanje, odnosno ostaje približno jednaka nuli. Platforma na kojoj se testira binarni detektor kvara je PlaDyPos platforma napravljena u Laboratoriju za podvodne sustave i tehnologije (LABUST, Fakultet elektrotehnike i računarstva).

## 2. PlaDyPos platforma

Platforma PlaDyPOs napravljena je na Fakultetu elektrotehnike i računarstva u Zagrebu, u Laboratoriju za podvodne sustave i tehnologije. Platforma je prvenstveno razvijena kako bi slijedila ronioca, a kasnije su nadograđivane i druge funkcije, te danas ona služi za razna testiranja i isprobavanja upravljačkih algoritama.

Slika 1. PlaDyPos platforma

Platformu pogone četiri propulzora u „X“ konfiguraciji. Ovakva konfiguracija omogućuje gibanje u svim smjerovima, neovisno o orijentaciji plovila. Platforma ima tri stupnja slobode : napredovanje (engl. *surge*), zanošenje (engl. *sway*) i zaošijanje (engl. *yaw*). Budući da je broj propulzora veći od broja stupnjeva slobode, može se zaključiti da je platforma redundantna. Redundancija pridonosi dodatnoj sigurnosti ukoliko dođe do kvara jednog od propulzora.

# 3. Općenito o ROS-u

Robotski operacijski sustav (engl. *Robot operating system*) je programski paket koji sadrži širok raspon alata i biblioteka za izradu aplikacija za upravljanje robotima. Neke od mogućnosti koje nudi su apstrakcija sklopovlja, komunikacija između procesa, te upravljanje paketima. Jezično je neovisan, te trenutno podržava programiranje u C++, Python-u i Lisp-u. Također, podržava istorazinsku komunikaciju (engl. *Peer-to-peer*), odnosno komunikaciju većeg broja procesora različitih računala koji su povezani na istorazinskoj strukturi. Radi lakšeg snalaženja i upravljanja, podijeljen je na veći broj manjih alata za izgradnju i pokretanje različitih komponenti sustava. Između ostalog sadrži alate za pokretanje ROS-a, navigaciju sustavom datoteka, grafički prikaz slanja poruka, te promjenu vrijednosti parametara konfiguracije. Građen je modularno unutar stabla glavnog koda, te se funkcionalnost izvodi preko manjih programskih jedinica. Prednost navedene građe je mogućnost pojedinačnog testiranja manjih dijelova koda te njihovo ponovno korištenje bez obzira na prvotnu namjenu za koju su pisani.

ROS se konceptualno može podijeliti u tri glavne razine:

- Razina sustava datoteka (engl. *ROS FIlesystem Level*)
- Razina komunikacije (eng. *ROS Computation Graph Level*)
- Razina podrške korisnicima (engl. *ROS Community Level*)

## 3.1. Razina sustava datoteka

Paketi (engl. *Packages*) predstavljaju osnovnu, odnosno najnižu razinu sustava, te mogu sadržavati čvorove, postavke podataka, konfiguracijske te zavisne datoteke. Ovisnost između različitih paketa, te opis paketa definira se u manifestu (engl. *Package Manifest*). Skup paketa koji čini kompletnu funkcionalnost i tvori biblioteku više razine nazive se stog (engl. Stack). Također postoji i stog manifest (engl. Stack Manifest) koji ima istu funkcionalnost kao i paketni manifest, osim što je usmjeren prema stogovima, a ne paketima. U razinu sustava datoteka mogu se još dodati i tipovi poruka (engl. *Message types*) i tipovi usluga (engl. *Service types*). Tipovi poruka opisuju poruke, te definiraju strukturu datoteka koje se šalju, dok tipovi usluga opisuju usluge te definiraju zahtjeve i odgovore za podatkovne strukture usluge.

## 3.2. Razina komunikacije

ROS komunikacijski sustav sastoji se od čvorova (engl. *Nodes*), glavnog čvora (engl. *Master*), poruka(engl. *Messages*), usluga (engl. *Services*) i tema (engl. *Topics*).

Čvorovi su jedinstveni programi sa izvedbenim kodom. U nekom robotskom sustavu postoji mnoštvo čvorova, a da bi se mogla uspostaviti komunikacija između njih potrebno je pokrenuti glavni čvor koji regulira komunikaciju među čvorovima. Čvor koji želi objavljivati poruke na *topic*-u nazivamo *publisher*, a čvor koji se pretplaćuje na *topic* nazivamo *subscriber*. Poruke su strukture podataka koje definiraju oblik podataka koji se prenosi, dok je *topic* definiran vrstom ROS poruke. Kada *publisher* želi objaviti poruku na *topic*-u, prvo obavijesti glavni čvor. Nakon toga *publisher* počinje pisati poruku, a s*ubscriber* javlja glavnom čvoru da se želi pretplatiti na isti *topic*. Glavni čvor potom obavještava *publisher*-a i *subsrciber*-a o međusobnoj postojanosti, te oni počinju komunicirati.

Iako je *publish/subscribe* komunikacija vrlo fleksibilna komunikacijska metoda, prevelik broj jednosmjernih prijenosa nije primjeren za interakciju zahtjev/odgovor (engl. *request/replay*) koja je često zastupljena u sustavima distribucije. Umjesto toga moguće je koristiti interakciju pomoću usluga koje su definirane parom poruka, od kojih je jedna za zahtjev, a druga za odgovor. Čvor pruža uslugu pod određenim imenom, a korisnik slanjem zahtjeva za poruku poziva uslugu i čeka odgovor.

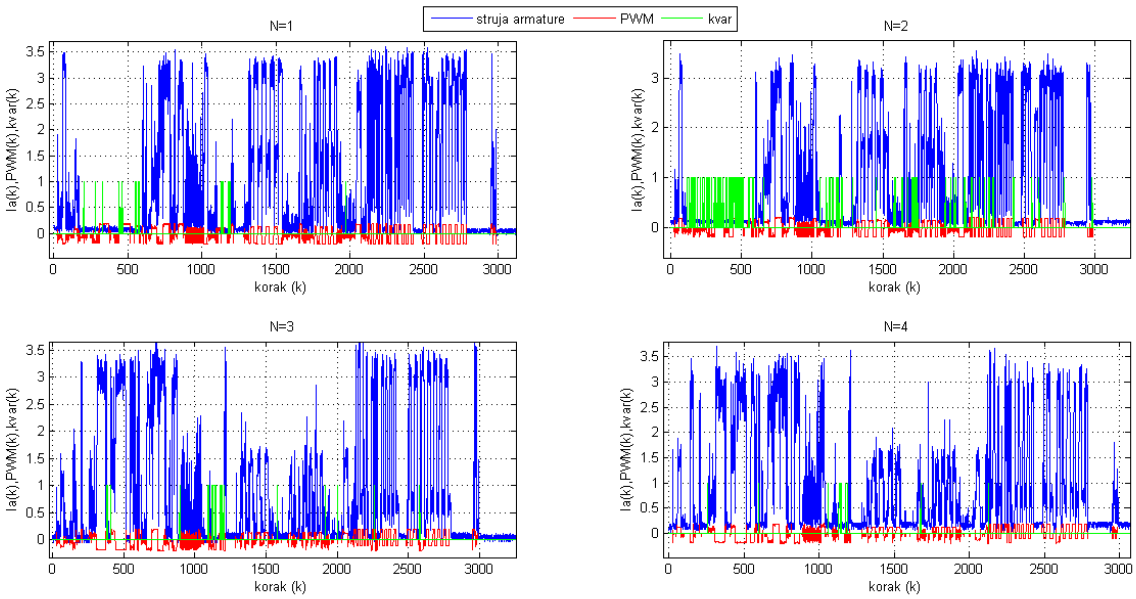
## 3.3. Razina podrške korisnicima

Razina podrške korisnicima služi za izmjenjivanje znanja, programa i iskustava među korisnicima ROS-a. Registrirani korisnici mogu izdavati vlastitu dokumentaciju, prepravljati i nadopunjavati već postojeće dokumente na središnjem forumu koji se naziva *ROS wiki*. Također, mogu postavljati pitanja i davati odgovore ukoliko naiđu na određene poteškoće koje ne znaju riješiti.

# 4. Binarni detektor kvara (*Matlab*)

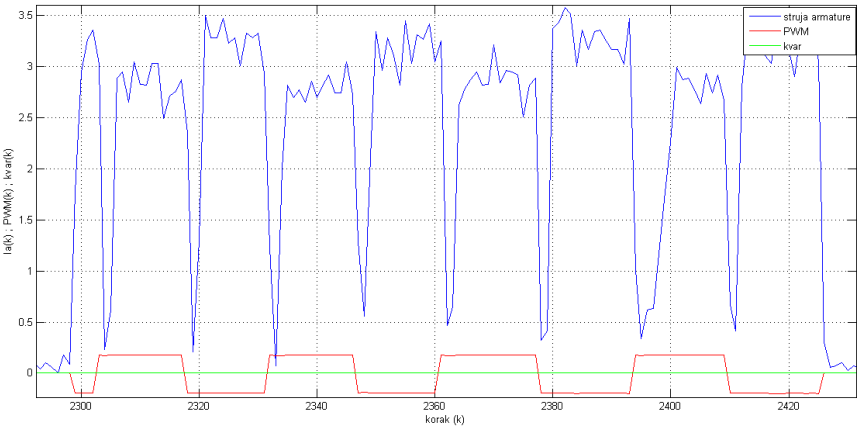
Binarni detektor kvara razvijan je u *Matlab*-u na temelju snimljenih logova. Logovi sadrže mjerenja upravljačkog signala, struje armature, brzine vrtnje te temperature motora. Za binarni detektor kvara bitni su samo podaci o upravljačkom PWM signalu i struji armature, a njih se izdvoji pokretanjem m-skripte iz DODATKA A. Nakon toga potrebno je izjednačiti vektore struja (DODATAK B). Vrijeme uzorkovanja upravljačkog signala je manje od vremena uzorkovanja struje armature, te je vektor podataka upravljačkog signala veći od vektora podataka struje armature. Matlab m-skriptom iz DODATKA C izjednačavaju se duljine navedenih vektora. Struja armature je dosta zašumljena, te kada je upravljački signal jednak nuli, njena vrijednost je ipak nešto veća od nule. Uzorci struje kada je PWM signal jednak nuli se izoliraju, te se računa njihova prosječna vrijednost. Onda kada se upravljački signal promijenio, a struja armature je jednaka nuli, odnosno manja ili jednaka gore izračunatoj prosječnoj vrijednosti, može se pretpostaviti kvar. Ako se i u sljedećih nekoliko uzoraka pretpostavi kvar, može se zaključiti da je propulzor ronilice u kvaru. M-skripta priložena u DODATKU D obavlja gore opisanu detekciju kvara.

Pokretanjem priloženih m-skripti dobiju se rezultati prikazani na slici ispod (Slika 2.).

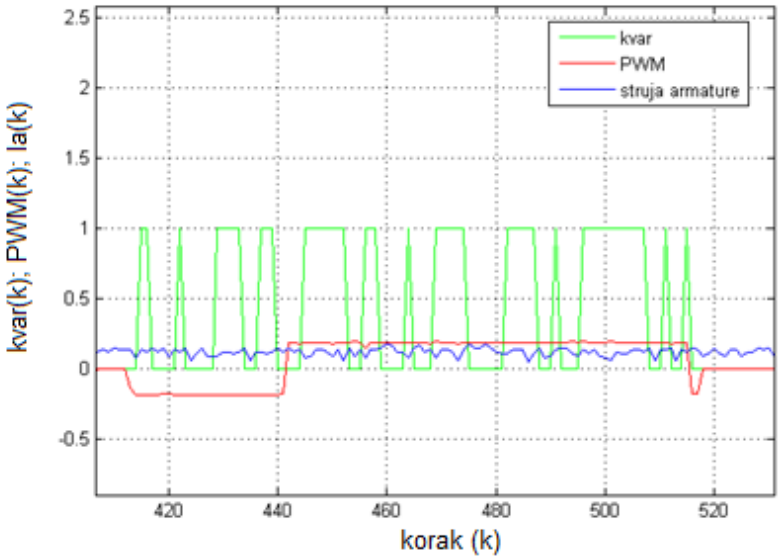



Slika 2. Dobiveni rezultati

Na slici iznad plavom bojom je prikazana struja armature, crvenom bojom PWM signal, a zelenom bojom trenutak u kojem se dogodio kvar. Vidljivo je da se na svakom od četiri propulzora u određenom trenutku dogodio kvar.




Slika 3. Uvećani prikaz normalnog režima rada na drugom propulzoru




Slika 4. Uvećani prikaza kvara na drugom propulzoru

# 5. Zaključak

Binarni detektor kvara se razvijao u *Matlab-u* na temelju već postojećih mjerenja upravljačkog PWM napona i struje armature. Onda kada bi se upravljački signal promijenio, a struja armature bila približno jednaka nuli, pretpostavio bi se kvar. Ako bi se i u sljedećih nekoliko uzoraka pretpostavio kvar, moglo se zaključiti da je propulzor ronilice u kvaru.

Sljedeće što je potrebno napraviti je implementirati binarni detektor kvara u ROS-u. Zahvaljujući mogućnostima koje pruža ROS, detektor će se moći testirati na površinskom vozilu gdje će se kvar simulirati selektivnim gašenjem motora.

# Literatura

[1] ROS Documentation ( 15.05.2015.) Dostupno na : *http://www.ros.org/wiki/*

[2] Isermann ,R. *Fault-Diagnosis Systems: An Introduction from Fault Diagnosis to Fault Tolerance*, Springer Science & Business Media, 2006.

[3] Barišić,M. , Mišković,N. *Fault Detection and Localization on Underwater Vehicle Propulsion Systems Using Principal Component Analysis* , Fakultet elektrotehnike i računarstva, Zagreb

# DODATAK A

```
clear all;
clc;
load('buddy_diagnostics_2015_05_15.mat')

k = length(data_meta.buddypwm_out.time.time);

%PWM
tpwm=zeros(1,k);
pwm1=zeros(1,k);
pwm2=zeros(1,k);
pwm3=zeros(1,k);
pwm4=zeros(1,k);
pwm5=zeros(1,k);
pwm6=zeros(1,k);


for i = 1:k
    tpwm(i)=data_meta.buddypwm_out.time.time(1,i);
end

for i = 1:k
    pwm1(i)=data.buddypwm_out.data(1,i);
end

for i = 1:k
    pwm2(i)=data.buddypwm_out.data(2,i);
end

for i = 1:k
    pwm3(i)=data.buddypwm_out.data(3,i);
end

for i = 1:k
    pwm4(i)=data.buddypwm_out.data(4,i);
end

% for i = 1:k
%     pwm5(i)=data.buddypwm_out.data(5,i);
% end
%
% for i = 1:k
%     pwm6(i)=data.buddypwm_out.data(6,i);
% end

%struja
idx1 = diag.BMotor1.values.current<10;
idx2 = diag.BMotor2.values.current<10;
idx3 = diag.BMotor3.values.current<10;
idx4 = diag.BMotor4.values.current<10;
idx5 = diag.BMotor5.values.current<10;
```

```
idx6 = diag.BMotor6.values.current<10;

tstruja = diag.BMotor1.header.stamp.time(idx1);
struja1 = diag.BMotor1.values.current(idx1);
struja2 = diag.BMotor2.values.current(idx2);
struja3 = diag.BMotor3.values.current(idx3);
struja4 = diag.BMotor4.values.current(idx4);
struja5 = diag.BMotor5.values.current(idx5);
struja6 = diag.BMotor6.values.current(idx6);

save('test.mat','tstruja','struja1','struja2','struja3','struja4','struja5','struja6'
,'tpwm','pwm1','pwm2','pwm3','pwm4','pwm5','pwm6');
```

## DODATAK B

```
% struja 1
if length(struja1)>length(tstruja);
    struja1=struja1(1:length(tstruja));

else if length(struja1)<length(tstruja);
        struja1(length(struja1)+1:length(tstruja)) = 0;
    end
end

% struja 2

if length(struja2)>length(tstruja);
    struja2=struja2(1:length(tstruja));

else if length(struja2)<length(tstruja);
        struja2(length(struja2)+1:length(tstruja)) = 0;
    end
end

% struja 3
if length(struja3)>length(tstruja);
    struja3=struja3(1:length(tstruja));

else if length(struja3)<length(tstruja);
        struja3(length(struja3)+1:length(tstruja)) = 0;
    end
end

% struja 4
if length(struja4)>length(tstruja);
    struja4=struja4(1:length(tstruja));

else if length(struja4)<length(tstruja);
```

```
        struja4(length(struja4)+1:length(tstruja)) = 0;
    end
end

% struja 5
if length(struja5)>length(tstruja);
    struja5=struja5(1:length(tstruja));

else if length(struja5)<length(tstruja);
        struja5(length(struja5)+1:length(tstruja)) = 0;
    end
end

% struja 6
if length(struja6)>length(tstruja);
    struja6=struja6(1:length(tstruja));

else if length(struja6)<length(tstruja);
        struja6(length(struja6)+1:length(tstruja)) = 0;
    end
end
```

## DODATAK C

```
% vrijeme uzorkovanja struje je oko 0.6 sekunde,
% a vrijeme uzorkovanja pwm-a je oko 0.1 sekunde.
% Zbog različitih vremena uzorkovanja imamo različite
% duljine vektora struja i pwm-a. Potrebno je skalirati navedene
% vektore po vremenu kako bi znali kako promjena pwm signala utječe
% na promjenu struje.

skaliranje_struja;

struja=[struja1;struja2;struja3;struja4;struja5;struja6];
pwm=[pwm1;pwm2;pwm3;pwm4;pwm5;pwm6];

n=length(tstruja);
m=length(tpwm);
vektor_ind = zeros(1,n);

for i = 1:n
    odstupanje = inf;
    for j = 1:m
        raz = abs(tstruja(i)- tpwm(j));
        if (raz < odstupanje)
            odstupanje = raz;
```

```
            index=j;
        end
    end
    vektor_ind(i)=index;
end

for i = 1:6
    for j = 1:n
    pwm(i,j)=pwm(i,vektor_ind(j));
    end
end
pwm = pwm (:,1:n);
```

## DODATAK D

```
% kvar detektiramo onda kad je pwm signal razlicit od nula,
% a iznos struje je približno jednak nuli. Budući da su mjerenja
% struje zašumljena potrebno je ispitati nekoliko uzastopnih
% uzoraka, te ukoliko se vrijednost struje i dalje ne mijenja, propulzor
% je u kvaru.
clc
podaci;
skaliranje_struja;
skaliranje_struja_napon;
kvar = zeros(6,n);
fault=zeros(6,n);
avg=zeros(1,i);

for i=1:6
    sum=0;
    k=0;
        for j= 2:n
        if(pwm(i,j)== 0 && pwm(i,j-1) == 0)
        x=sum;
        sum =x + struja(i,j);
        k = k+1;
        end
        avg(1,i) = sum/k ;
        end
end
```

```
for i = 1:6
    for j = 2:n
        if (((pwm(i,j) - pwm(i,j-1) ~= 0) || pwm(i,j) ~= 0)  && (struja(i,j) <=
avg(1,i)))
            kvar(i,j) = 1;

        end
    end
end

for i= 1:6
    ind = 0;
    for j = 3 : n
        if ( kvar(i,j) == 1 && kvar(i,j-1) == 1 && kvar(i,j-2) == 1  )
            ind = 1;
            fault(i,j)=1;
             fprintf('Kvar je detektiran u %d. uzorku na %d. motoru\n',j,i)
        end
    end
    if ( ind == 1)
        fprintf('Propulzor %d je u kvaru \n', i)
    else
        fprintf ('Nema kvara na %d. propulzoru \n',i)
    end
end
```