STU FEI
ÚSTAV ROBOTIKY A KYBERNETIKY

Kolekcia učebných textov

september 2025
github.com/OkoliePracovnehoBodu/KUT
RJ

v1.0
KUTdevAS1

Laboratórne zariadenie AeroShield: softváre Matlab

Cieľom textu je zoznámenie sa s obslužným Matlab softvárom pre prácu so zariadením *AS*, ktorý slúži na meranie a ovládanie spomínaného zariadenia.

1 Hlavný program

Výpis kódu 1: Zavolanie funkcie merania.

```
% ----------------------------------
% Run the experiment using the following line of code
% ----------------------------------

[t, y, u, potentiometer, yhat, dyhat] = runArduinoPlot();

% ----------------------------------
% ----------------------------------
```

Zavolaním funckie `runArduinoPlot()` sa začne navrhnuté meranie, ktoré následne automaticky skončí, čí uz po vypršaní času merania alebo inou podmienkov, ktorú si vieme zadefinovať samostatne. Definícia ukončovacích podmienok bude spomenutá nižšie. Ako môžeme vidieť na výpise kódu 1, funkcia vracia dáta, ktoré následne možno použiť. Ktoré dáta vracia si vieme následne vo funkcii zadefinovať, poprípade si vieme dodať, ďalšie pre nás dôležité veličiny.

2 Dátový priečinok

Výpis kódu 2: Vytvorenie dátového priečinku.

```
% ----------------------------------
% Create the data repository
% ----------------------------------
DDIR = "dataRepo";
if ~exist(DDIR, "dir")
    fprintf("Creating the data directory...");
    mkdir(DDIR);
end
% ----------------------------------
% ----------------------------------
```

Kód, ktorý je ukázaný vo výpise 2, slúži na vytvorenie adresáru (priečinku) pre uloženie meraní. Do tohto priečinka sa bežne ukladajú merania av dvoch formátoch *CSV* a *MAT*. *CSV* súbor obsahuje merané veličiny dostatné po sériovej linke, *MAT* súbor obsahuje celý Matlab workspace - všetky definované premenné v Matlab-e. *Je odporúčané si manuálne vytvoriť nový adresár, do ktorého si budeme kopírovať (CTRL+C CTRL+V) merania, s ktorými chceme následne pracovať, aby sme vždy mali netknuté dáta z merania v zálohe.*

3 Definícia premenných

Výpis kódu 3: Definícia všetkých potrebných premenných.

```
% ----------------------------------
% Define all the parameters
% ----------------------------------

% Define time parameters

T_start = 0;

T_sample = 3;       % [ms] <1, 255>

% Define STOP TIME

T_stop = 60.0;      % [sec]

% Define control parameters
U_MAX = 100.0;
U_MIN = 0.0;
Y_SAFETY = 190.0;

% Define PID param
P = 1.0;
I = 0.30;
D = 0.19;

R_WANTED = 140;

% alpha - beta filter
alpha = 0.8;
beta = 0.2;

timer_t = [];
timer_y = [];
timer_yhat = [];
timer_dyhat = [];
timer_u = [];
timer_potentiometer = [];
% ----------------------------------
% ----------------------------------
```

Táto časť kódu (3) slúži na definíciu všetkých potrebných premenných pre dané meranie, v tomto prípade sme si zadefinovali premenné pre PID regulátor a $\alpha - \beta$ filter, aby sme ukázali možné riešenia implementácie riadenia a pozorovania. Následne v tab. 1 budú potrebné premenné pre priebeh merania popísané, ostatné premenné sú špecificky zvolené pre tento poskytnutý náhľadový kód - nebudú potrebne pre napr. meranie dynamických vlastností systému.

Tabuľka 1: Krátky popisok nevyhnutných premenných

Premenná	Jednotka	Popis
`T_start`	s	Čas začiatku merania.
`T_sample`	ms	Perióda vzorkovania v rozsahu 1 až 255.
`T_stop`	s	Čas trvania merania.
`U_MAX`	%	Maximálny akčný zásah.
`U_MIN`	%	Minimálny akčný zásah.
`Y_SAFETY`	°	Maximálna hodnota výstupnej veličiny.
`timer_t`	s	Pole časov v ktorých prebehla komunikácia.
`timer_y`	°	Pole výstupov zo zariadenia AeroShield.
`timer_u`	%	Pole vstupov do zariadenia AeroShield.
`timer_potentiometer`	%	Pole hodnôt z potenciometra.

4 Vykreslenie dát v reálnom čase

Výpis kódu 4: Definícia časovača na vykreslenie meraných dát v reálnom čase.

```
% ---------------------------------
% Plot the measured data in real time
% ---------------------------------
function plotData()
    persistent hy hr hu;

    % persistent example

    try
        if isempty(hy) || isempty(hr) || isempty(hu)
            f = figure(9999); clf(f);
            ax = axes(f);
            hold on;
            hy = plot(ax, nan, nan, '.b');
            hr = plot(ax, nan, nan, '.r');
            hu = plot(ax, nan, nan, '.k');

            % example = plot(ax, nan, nan, '.g');

            grid minor;
            title("Real-Time System Response");
            xlabel("t [s]");
            ylabel("$\varphi [^\circ]$", "Interpreter","latex");
            legend(ax, "y","ref", "yhat", 'Location', 'southeast');

        end

        set(hy, 'YData', timer_y, 'XData', timer_t);
        set(hr, 'YData', timer_potentiometer, 'XData', timer_t);
        set(hu, 'YData', timer_yhat, 'XData', timer_t);

        % set(example, 'YData', example_y, 'XData', example_x);

        drawnow limitrate nocallbacks;
    catch err
       fprintf(2, "Plot thread: " + err.message + "\n");
    end
end

tPlot = timer('ExecutionMode','fixedRate', 'Period', 0.5, 'TimerFcn', @
    (~, ~) plotData());
start(tPlot);
% ---------------------------------
% ---------------------------------
```

Obslužná funkcia `plotData` na vykreslovanie grafov v reálnom čase, ktorá je následne volaná časovačom `tPlot` v intervale 500 milisekúnd pokiaľ sa neukončí meranie. V prípade, že by sme chceli dodať ďalší signál do grafu, môžeme postupovať podľa už predrobeného príkladu, kedy sme v bloku kódu 4 zakomentovali 3 riadky. Avšak, keď pridáme nový signál na vykreslovanie, tak je pre funkčnosť a plynulosť dôležité zvýšiť periódu vykreslovania. Nakoľko počet dát na vykreslenie sa zvýši a vykreslovanie v Matlab-e, nie je obzvlásť rýchle.

5 Záznamové súbory merania

Výpis kódu 5: Inicializácia záznamových súborov.

```
% ---------------------------------
% Initialize File Streams
% ---------------------------------

DateString = convertCharsToStrings(datestr(datetime('now'), "
    yyyy_mm_dd_HH_MM_ss"));
```

```
6
7  FILENAME = "dataFile";
8
9  function fullpath = getfilename(dirpath, filename, datestr, ext)
10     if nargin < 3
11         error("At least the first 3 parameters need to be provided.");
12     elseif nargin == 3
13         ext = "csv";
14     end
15
16     fullpath = "./" + dirpath + "/" + filename + "_" + datestr + "." +
           ext;
17 end
18
19 FILEPATH = getfilename(DDIR, FILENAME, DateString);
20 FILEPATH_MAT = getfilename(DDIR, FILENAME, DateString, 'mat');
21
22 if(exist("datafileID", "var"))
23     fclose(datafileID);
24     clear datafileID;
25 end
26
27 datafileID = fopen(FILEPATH,'w');
28 fprintf(datafileID, 't, tp, r, y, u, dtp, dt\n');
29 % ---------------------------------
30 % ---------------------------------
```

Vytvorenie súboru na zapísanie merania a zapísanie názvov stĺpcov, ktoré sú do súboru zapisované. V prípade, že súbor už existuje alebo je aktívny v Matlab prostredí, tak sa zavrie, aby každé meranie bolo vložené do vlastného súboru. Ak chceme zmeniť cestu ukladania súborov merania, tak vo funkcii `getfilename` musíme zmeniť premennú `fullpath`. Keď pridáme nový signál do súboru, je nutné pridať aj meno stĺpca, aby nedošlo k nesprávnemu načítaniu, keď budeme dáta potrebovať načítať.

6 Zapisovanie dát merania

Výpis kódu 6: Zapisovanie meraných dát do súboru a konzoly.

```
1  % ---------------------------------
2  % Write data into files
3  % ---------------------------------
4
5  function updateInfo(datafileID, dt, Ts, x)
6      if ((dt) > (Ts*1.05))
7          fprintf('%8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f  --\n', x); %
               write to console, measurement took too long
8      else
9          fprintf('%8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f\n', x); %
               write to console
10     end
11     fprintf(datafileID, '%8.3f,%8.3f,%8.3f,%8.3f,%8.3f,%8.3f,%8.3f\n', x
           ); % write to file
12
13     timer_t = [timer_t x(1)];
14     timer_y = [timer_y x(4)];
15     timer_u = [timer_u x(5)];
16     timer_potentiometer = [timer_potentiometer x(3)];
17 end
18
19 doUpdate = @(x) updateInfo(datafileID, x(end), T_sample, x);
20
21 % ---------------------------------
22 % ---------------------------------
```

Funkcia `updateInfo` vo výpise kódu 6 slúži na zapisovanie dát do súboru *CSV* a do konzoly.

Ak pridáme nový signál do zapisovania dát, musíme pridať `,%8.3f` do riadku s

komentárom *% write to file* pred text `\n`. Následne sa do súboru bude ukladať nový signál.

7 Sériová komunikácia

Výpis kódu 7: Inicializácia sériovej komunikácie a konfigurácia.

```
% ----------------------------------
% Define serial port parameters, open and configure comms
% ----------------------------------

if(exist("serPort", "var"))
    serPort.flush("input");
    clear serPort;
end

serPort = serialport('COM3', 115200, 'Timeout', 5);

serLine = readline(serPort);

while(~contains(serLine, "config"))
    disp(serLine);
    serLine = readline(serPort);
end

fprintf("Sending now\n");
write(serPort, cast(T_sample, "uint8"), "uint8");

% Read the first line from the serial port (MCU starting)
while(~contains(serLine, "start"))
    disp(serLine);
    serLine = readline(serPort);
end

disp(serLine);
write(serPort, 0.0, 'single'); % Necessary to send this command for
    stable sampling period

while(contains(serLine, "---"))
    disp(serLine);
    serLine = readline(serPort);
end

% Read and parse the calibration data
serLineList = str2num(serLine); %#ok<ST2NM>

% ----------------------------------
% ----------------------------------
```

V tejto sekcii kódu 7 popisujeme postup inicializácie sériovej komunikácie s Arduinom a zároveň nastavenie periódy vzorkovania. Je v záujme čitateľa predom porozumieť, fungovanie sériovej komunikácie a poskytnutého kódu pred snahou modifikácie tejto časti kódu. V skrate, otvárame sériovú komunikácie na porte `COM3`, v prípade, že dojde k situácii, kedy Matlab vypíše chybné hlásenie o nenajdení alebo nedostupnosti zariadenia na tomto porte, tak sú 3 možnosti, ktoré vieme aplikovať na vyriešenie tohto problému.

1. Vypojiť a následne zapojiť zariadenie z počítača pomocou USB káblu.
2. Pozrieť pripojené fyzické porty k počítaču v termináli pomocou príkazu `mode` na zariadení s operačným systémom Windows, `sudo dmesg | grep -i tty` na Linux/MacOS. Ak, nájdeme pripojené zariadenia, tak v prípade Windows-u zmeníme v kóde `COM3` na iný `COM`, pri Linux-e/MacOS je názov portu v tvare `/dev/ttyS`, následne aplikujeme rovnaký postup ako pri Windows zariadení.
3. Nakoniec, môže dojsť ku chybe komunikácie a zariadenie nebude reagovať, tak do Matlab konzoly napíšeme `save; clear;` a spustíme - uložíme si týmto spôsobom

dáta a zaroveň vymažeme celé prostredie Matlab-u. Následne spustíme meranie, ktoré predčasne ukončíme, keď sa inicializuje komunikácia a vstup do systému bude 0 - predčasné ukočenie vieme dosiahnuť tak, že pri spúšťaní podržíme vrtulku v nulovej pozícii (kyvadlo je vo zvyslej polohe, kde má uhoľ natočenia rovný 0). Keď meranie začne, tak vrtulku manuálne vyvrhneme nad bezpečnostnú hranicu výstupu - nami definovaná hodnota je 190°. Ak to nepomôže, tak vtedy je potrebné zariadenie odpojiť od napájania (zásuvky) a následne pripojiť, aby sa resetovalo Arduino, podľa možností možeme natvrdo resetovať Arduino.

8 Počiatočné hodnoty

Výpis kódu 8: Zaznamenanie počiatočných hodnôt.

```
% ----------------------------------
% Extract the initial values from the received data
% ----------------------------------
plant_time_init = serLineList(1);
plant_potentiometer_init = serLineList(2);
plant_output_init = serLineList(3);
plant_input_init = serLineList(4);

plant_time = serLineList(1) - plant_time_init;
plant_input = serLineList(2);
plant_output = serLineList(3);
plant_potentiometer = R_WANTED + serLineList(4)/100*20;
plant_dt = serLineList(5);

timer_yhat = [timer_yhat, plant_output];
timer_dyhat = [timer_dyhat, 0];

% Display the received data
tmp_printlist = [0, plant_time, plant_potentiometer, plant_output,
    plant_input, plant_dt, T_sample];
doUpdate(tmp_printlist);
% ----------------------------------
% ----------------------------------
```

Kód zobrazený na 8 sa týka počiatočnej komunikácie so zariadením, počiatočne snímané hodnoty sú cez sériovú linku poslané Arduinom, po čom započne samotné meranie. Počiatočné hodnoty sa používajú v prípade času, kedy odpočítavame od času trvania merania počiatočný čas, aby sme v rámci záznamu začínali časom v 0, nakoľko čas pred týmto odsekom kódu náleží inicializácii zariadenia a nie samotnému meraniu. Po prvýkrát voláme funkciu `doUpdate(x)`, ktorá zaznamená merané dáta do súboru a konzoly. V našom prípade nepoužívame hodnotu potenciometra navrátenú zariadením priamo, nakoľko tento signál používame na jemné menenie žiadanej hodnoty. Nakoľko sa pohybujeme v relatívne nestabilnom pracovnom okolí s našou požiadavkou referenčnej hodnoty, tak sme museli možnosť zmeny referenčnej hodnoty zjemniť natoľko, aby nedošlo k takej zmene, ktorá by navrhnuté riadenie destabilizovalo.

9 Definícia premenných hlavne slučky

Výpis kódu 9: Nastavenie premenných v hlavnej slučke.

```
% ----------------------------------
% Set the main loop parameters
% ----------------------------------

% Set initial control input value
e_old = 0;
e_int_old = 0;
u = 0;
u_send = u;

```

```
11
12 % Get the initial time
13 time_start = datetime('now');
14 time_tick = time_start;
15
16 % ----------------------------------
17 % ----------------------------------
```

Jednoduchá definícia premenných (viď 9), ktoré sa používaju najmä v rámci slučky merania, sú tu umiestnené čisto z dôvodu, aby sme sa nemuseli hýbať v kóde až na začiatok, keď chceme zmeniť alebo doplniť premenné. Možno sem umiestniť väčšinu premenných, až na tie od ktorých chceme, aby boli zaznamenané v súboroch.

10 Čítanie sériovej komunikácie

Výpis kódu 10: Definícia počúvateľa sériovej komunikácie.

```
1  % ----------------------------------
2  % Define serial link listener
3  % ----------------------------------
4
5  function readSerialData(src, ~)
6      data = readline(src);
7      src.UserData = data;
8  end
9
10 configureCallback(serPort, "terminator", @readSerialData);
11 % ----------------------------------
```

Výpis 10 slúži na zadefinovanie obslužnej funkcie pre čítanie dát zo seriového portu, Matlab ponúka možnosť definície funkcie, ktorá bude volaná, ked zaznamená nové dáta v sériovej zbernici. Funkcionalitu následne používame v slučke na riadenie komunikácie a výpočtov, ktoré sa môžu udiať iba ak máme k dispozícii nové merania od zariadenia.

11 Spracovanie sériovej komunikácie

Výpis kódu 11: Spracovanie a zaznamenie dát zo sériovej komunikácie.

```
1  % ----------------------------------
2  % Process the read data from the serial communication
3  % ----------------------------------
4  waitfor(serPort, "UserData");
5
6  % Get current time
7  time_curr = datetime('now');
8
9  % Calculate time elapsed since last iteration
10 time_delta = milliseconds(time_curr - time_tick);
11
12 % Read and parse the received data
13 serLineList = str2num(serPort.UserData); %#ok<ST2NM>
14
15 time_tick = time_curr;
16
17 % Calculate total time elapsed
18 time_elapsed = seconds(time_curr - time_start);
19
20 % Extract values from the received data
21 plant_time = serLineList(1) - plant_time_init;
22 plant_input = serLineList(2);
23 plant_output = serLineList(3);
24 plant_potentiometer = R_WANTED + serLineList(4)/100*20;
25 plant_dt = serLineList(5);
26
27 dx = plant_output - timer_yhat(end);
28 cyhat = timer_yhat(end) + alpha*(dx);
```

```
29 timer_yhat = [timer_yhat, cyhat];
30 timer_dyhat = [timer_dyhat, timer_dyhat(end) + beta*(dx/time_delta)];
31
32 % Record the received data
33 tmp_printlist = [time_elapsed, plant_time, plant_potentiometer,
       plant_output, plant_input, plant_dt, time_delta];
34 doUpdate(tmp_printlist);
35
36 % ---------------------------------
37 % ---------------------------------
```

Ako sme spomínali, že samotná slučka čaká na nové dáta z merania, tak to možno vidieť vo výpise 11, kde na začiatku po komentároch voláme funkciu `waitfor`, ktorá čaká pokiaľ object `serPort` nezaznamená zmenu v premennej `UserData`, ktorá sa spätne zapisuje v obslužnej funkcii čítania sériovej zbernice (viď 10). Následne, keď máme k dispozícii nové meranie, tak ako prvé dáta spravujeme a zapíšeme do premenných, ktoré držia hodnotu týchto signálov. Po spracovaní tieto dáta používame na vyhodnotenie $\alpha - \beta$ filtru. Nakoniec všetky signály, ktoré chceme zaznamenať vložíme do premennej `tmp_printlist`, čo následne vložíme do funkcie `doUpdate` ako parameter, aby toto meranie zaznamenala do súboru a vykreslila do grafu. V tejto časti sme definovali aj vlastný program - výpočet filtra, to však z dôvodu toho, že dáta potrebujeme zaznamenať do súboru a grafu, v prípade, že by sme tieto hodnoty nechceli zaznamenať, tak vieme výpočty posunúť do časti vlastného kódu (viď 12), a následne ich použiť pri riadení.

12 Vlastný program

Výpis kódu 12: Blok pre vlastný program

```
1  % ---------------------------------
2  % Insert your code here (for example, we have PID controller code
       implemented in this space)
3  % ---------------------------------
4
5  e = plant_potentiometer - plant_output;
6
7  e_der = (e - e_old) / (time_delta/1000);
8
9  e_int = e_int_old + (e * (time_delta/1000));
10
11 e_old = e;
12 e_int_old = e_int;
13
14
15 u = P * e  +  I * e_int + D * e_der;
```

Táto časť kódu 12 slúži na implementáciu užívateľom navrhnutý riadiaci systém a vlastný kód. Je však potrebné si dávať pozor na to, že nakoľko toto meranie beží v reálny čas, tak dĺžka trvania riadiaceho algoritmu ovplyvní samotné meranie, nakoľko Matlab nebude čakať na dáta zo zariadenia, nakoľko stále trvá výpočet akčného zásahu. Preto je potrebné prispôsobiť periódu vzorkovania dĺžke trvania riadiaceho algoritmu, inak môže dojsť k desynchronizácii dát, či ich stráte. A však výpočet je oveľa rýchelejší ako samotné vykreslovanie, preto v prípade, že potrebujeme čo najpresnejšie meranie pri nízkej perióde vzorkovania, tak pomôže vypnút vykreslovanie v reálnom čase. V konzoli aj v prípade, že si po meraní vykreslíme graf pre premennú `dt`, tak budeme vidieť stabilnejší priebeh, ako v prípade, že beží aj vykreslovanie.

13 Saturácia akčného zásahu

Výpis kódu 13: Obmedzenie akčného zásahu na maximálne a minimálne hodnoty.

```
1  % ---------------------------------
```

```
% Saturate the control output to the MAX and MIN values
% ----------------------------------

u_send = u;

if u_send > U_MAX
    u_send = U_MAX;
elseif u_send < U_MIN
    u_send = U_MIN;
end

% ----------------------------------
% ----------------------------------
```

Vnášame do našeho systému nelinearitu (viď 13), ktorú nazývame saturácia alebo obmedzenie akčnej veličiný a to len aby sme zabezpečili, že do zariadenia neposielame nevhodné hodnoty vstupných veličín. Týmto zabezpečujeme bezpečný beh zariadenia a najmä bezpečnosť okoliu, minimalizujeme možnosť poškodenia.

14 Posielanie sériovej komunikácie

Výpis kódu 14: Funkcia na posielanie žiadanej akčnej veličiny po sériovej linke.

```
% ----------------------------------
% Send control input to the serial port
% ----------------------------------

write(serPort, u_send, "single");

% ----------------------------------
% ----------------------------------
```

Jeden z najdôležitejších riadkov programu, ktorý zabezpečuje odosielanie akčnej veličiny do Arudina na spracovanie a následné prevedenie žiadaného akčného zásahu na zariadenie, môžeme vidieť na výpise 14.

15 Konečná podmienka merania

Výpis kódu 15: Podmienka na bezpečné ukončenie merania.

```
% ----------------------------------
% Check if the simulation should stop (safety precaution)
% ----------------------------------

if time_elapsed >= T_stop || plant_output >= Y_SAFETY
    configureCallback(serPort, "off"); % Remove the callback from the
        serial port, before exiting the loop
    break;
end

% ----------------------------------
% ----------------------------------
```

Kód v bloku 15 definuje ukočovaciu podmienku merania, v tomto prípade máme dve podmienky ukočenia.

1. Keď doba trvania merania prekročí nami nastavenú maximálnu dobu trvania teda `T_stop`.
2. Alebo v prípade, že výstupná veličina zo zariadenia presiahne hodnotu `Y_SAFETY`, nakoľko sýstém nevieme ovládať v momente, keď výstupná veličina prevrší hodnotu 180°.

16 Ukončenie časovačov

Výpis kódu 16: Ukončenie a odstránenie všetkých aktívnych Matlab časovačov.

```
% ---------------------------------
% Close and delete all the existing timers
% ---------------------------------

for tim=timerfindall
    stop(tim);
    delete(tim);
end

% ---------------------------------
% ---------------------------------
```

Na výpise 16 ukočujeme všetky existujúce časovače v prostredí Matlab, teda aj tie ktoré nespustilo naše meranie. Možno si kód upraviť, tak aby ukočilo iba časovače nami spustené, to však nechávame na čítateľa.

17 Ukončenie komunikácie

Výpis kódu 17: Ukončenie sériovej a súborovej komunikácie

```
% ---------------------------------
% Close the serial connection
% ---------------------------------

% Send a final command and close the serial port
write(serPort, 0.0, 'single');
serPort.flush("input");
clear serPort;
fclose(datafileID);
clear datafileID;

% ---------------------------------
% ---------------------------------
```

Zastavujeme zariadenie, ukončujeme sériovú komunikáciu so zariadením a zatvárame súbory, do ktorých sme zapisovali dáta merania. Vymazávame všetky objekty, ktoré sú spojené s týmito akciami, nakoľko sú už po ukončení nepoužitelné. (viď 17) Táto časť programu zabezpečuje bezpečné ukočenie všetkých komunikácií a dovedenie zariadenia do ustáleného stavu.

18 Uloženie merania

Výpis kódu 18: Ukladanie meracích dát do csv a mat súborov.

```
% ---------------------------------
% Save the measurement into a .MAT file for easier access to data when
    using Matlab
% ---------------------------------

logsout = readtable(FILEPATH, "VariableNamingRule","preserve","Delimiter
    ",",");

save(FILEPATH_MAT);

% ---------------------------------
% ---------------------------------
```

Na výpise 18 môžeme vidieť spôsob zapisovania dát do súboru *MAT*, kedy ukladáme všetky premenné, ktoré sa nachádzajú v prostredí Matlab. Zároveň ako prvé si možeme všimnúť, akým spôsobom sa majú načítavať merania z *CSV* súboru.

19 Vykreslenie priebehu merania

Výpis kódu 19: Vykreslenie základných veličín procesu merania.

```
% ---------------------------------
% Quickly plot the measurement - reference, output, and control signal
% ---------------------------------

t = logsout.t;
y = logsout.y;
u = logsout.u;
r = logsout.r;
e = r - y;
dt = logsout.dtp;


figure(111);
hold on;
plot(t, y, '-k', 'LineWidth', 1.5);
plot(t, r, '-r', 'LineWidth', 1.5);
plot(t, u, '-b', 'LineWidth', 1.5);
title('Control Response');
subtitle("P = " + num2str(P) + ", I = " + num2str(I) + ", D = " +
    num2str(D));
legend('y(t)', 'ref(t)', 'u(t)', "Location", "best");
xlabel('t [s]');
ylabel('y [deg]');
grid on;
hold off;
% ---------------------------------
% ---------------------------------
```

Nakoniec merania vo funkcii ešte zobrazíme namerané hodnoty, ktoré sú čítane priamo zo súborov, to slúži na spatnú kontrolu s tým, čo sa vykreslilo na graf reálneho času alebo konečný graf mimo funkcie (viď 20), ktorý však vykresluje dáta, ktoré sú uložene v pamäti Matlab-u. Preto graf na výpise 19 slúži na spätnú kontrolu.

20 Vykreslenie priebehu $\alpha - \beta$ filtru

Výpis kódu 20: Vykreslenie priebehu a porovnania odhadu stavu pomocou $\alpha - \beta$ filtra.

```
% ---------------------------------
%% Plot the data
% ---------------------------------

figure(100);
subplot(3, 1, 1);
plot(t, y, t, yhat, t, potentiometer, 'LineWidth', 1.5);
grid minor;
legend('y','yhat','ref');
xlabel('t [s]');
ylabel('$\varphi [^\circ]$', 'Interpreter', 'latex');
title('System response');
subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');

subplot(3, 1, 2);
plot(t, dyhat, 'LineWidth', 1.5);
grid minor;
xlabel('t [s]');
ylabel('$\omega [^\circ/s]$', 'Interpreter', 'latex');
title('System velocity response');
subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');

subplot(3, 1, 3);
plot(t, (y-yhat), 'LineWidth', 1.5);
grid minor;
xlabel('t [s]');
ylabel('$\varphi [^\circ]$', 'Interpreter', 'latex');
```

```
28 title('Observer error');
29 subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');
30
31 % ---------------------------------
32 % ---------------------------------
```

Po ukončení merania vykreslíme dáta 20, ktoré nám vrátila funkcia v ktorej dané meranie prebiehalo. Toto slúži na ukážku toho, že vieme s dátami pracovať priamo po meraní. V našom prípade vykreslujeme porovnanie nameraných hodnôt a pozorovaných výstupných hodnôt z $\alpha - \beta$ filtra. Tieto grafy sú vytvorené iba z dát, ktoré dostaneme z funkcie `runArduinoPlot()` a následne spracovanými hodnotami, ktoré sme vytvorili kombináciou meraných veličín.