

Laboratórne zariadenie AeroShield: softwára Matlab

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

1 Hlavný program

Výpis kódu 1: Zavolanie funkcie merania.

```
1 % -----  
2 % Run the experiment using the following line of code  
3 % -----  
4  
5 [t, y, u, potentiometer, yhat, dyhat] = runArduinoPlot();  
6  
7 % -----  
8 % -----
```

Zavolaním funkcie `runArduinoPlot()` sa začne navrhnuté meranie, ktoré následne automaticky skončí, či už po vypršaní času merania alebo inou podmienkou, ktorú si vieme zadať 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 zadať, poprípade si vieme dodať, ďalšie pre nás dôležité veličiny.

2 Dátový priečinok

Výpis kódu 2: Vytvorenie dátového priečinku.

```
1 % -----  
2 % Create the data repository  
3 % -----  
4 DDIR = "dataRepo";  
5 if ~exist(DDIR, "dir")  
6     fprintf("Creating the data directory...");  
7     mkdir(DDIR);  
8 end  
9 % -----  
10 % -----
```

Kód, ktorý je ukázaný vo výpise 2, slúži na vytvorenie adresáru (priečinku) pre uloženie meraní. Do tohto priečinku sa bežne ukladajú merania av dvoch formátoch *CSV* a *MAT*. *CSV* súbor obsahuje merané veličiny dostátne 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.

```
1 % -----
2 % Define all the parameters
3 % -----
4
5 % Define time parameters
6
7 T_start = 0;
8
9 T_sample = 3;      % [ms] <1, 255>
10
11 % Define STOP TIME
12
13 T_stop = 60.0;     % [sec]
14
15 % Define control parameters
16 U_MAX = 100.0;
17 U_MIN = 0.0;
18 Y_SAFETY = 190.0;
19
20 % Define PID param
21 P = 1.0;
22 I = 0.30;
23 D = 0.19;
24
25 R_WANTED = 140;
26
27 % alpha - beta filter
28 alpha = 0.8;
29 beta = 0.2;
30
31 timer_t = [];
32 timer_y = [];
33 timer_yhat = [];
34 timer_dyhat = [];
35 timer_u = [];
36 timer_potentiometer = [];
37 % -----
38 % -----
```

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 potrebné pre napr. meranie dynamických vlastností systému.

Tabuľka 1: Krátky popisok nevyhnutných premenných

Premenná	Jednotka	Popis
T_start	s	Čas začiatku merania.
T_sample	ms	Periódka 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.

```
1 % -----
2 % Plot the measured data in real time
3 % -----
4 function plotData()
5     persistent hy hr hu;
6
7     % persistent example
8
9     try
10         if isempty(hy) || isempty(hr) || isempty(hu)
11             f = figure(9999); clf(f);
12             ax = axes(f);
13             hold on;
14             hy = plot(ax, nan, nan, '.b');
15             hr = plot(ax, nan, nan, '.r');
16             hu = plot(ax, nan, nan, '.k');
17
18             % example = plot(ax, nan, nan, '.g');
19
20             grid minor;
21             title("Real-Time System Response");
22             xlabel("t [s]");
23             ylabel("$\varphi [\circ]$", "Interpreter","latex");
24             legend(ax, "y","ref", "yhat", 'Location', 'southeast');
25
26         end
27
28         set(hy, 'YData', timer_y, 'XData', timer_t);
29         set(hr, 'YData', timer_potentiometer, 'XData', timer_t);
30         set(hu, 'YData', timer_yhat, 'XData', timer_t);
31
32         % set(example, 'YData', example_y, 'XData', example_x);
33
34         drawnow limitrate nocallbacks;
35     catch err
36         fprintf(2, "Plot thread: " + err.message + "\n");
37     end
38 end
39
40 tPlot = timer('ExecutionMode','fixedRate', 'Period', 0.5, 'TimerFcn', @
41     (~, ~) plotData());
42 start(tPlot);
43 % -----
44 % -----
```

Obslužná funkcia `plotData` na vykresľovanie 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 vykresľovanie, tak je pre funkčnosť a plynulosť dôležité zvýšiť periódu vykresľovania. Nakoľko počet dát na vykreslenie sa zvýši a vykresľovanie v Matlab-e, nie je obzvlášť rýchle.

5 Záznamové súbory merania

Výpis kódu 5: Inicializácia záznamových súborov.

```
1 % -----
2 % Initialize File Streams
3 % -----
4
5 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.

```
1 % -----
2 % Define serial port parameters, open and configure comms
3 % -----
4
5 if(exist("serPort", "var"))
6     serPort.flush("input");
7     clear serPort;
8 end
9
10 serPort = serialport('COM3', 115200, 'Timeout', 5);
11
12 serLine = readline(serPort);
13
14 while(~contains(serLine, "config"))
15     disp(serLine);
16     serLine = readline(serPort);
17 end
18
19 fprintf("Sending now\n");
20 write(serPort, cast(T_sample, "uint8"), "uint8");
21
22 % Read the first line from the serial port (MCU starting)
23 while(~contains(serLine, "start"))
24     disp(serLine);
25     serLine = readline(serPort);
26 end
27
28 disp(serLine);
29 write(serPort, 0.0, 'single'); % Necessary to send this command for
    stable sampling period
30
31 while(contains(serLine, "---"))
32     disp(serLine);
33     serLine = readline(serPort);
34 end
35
36 % Read and parse the calibration data
37 serLineList = str2num(serLine); %#ok<ST2NM>
38
39 % -----
40 % -----
```

V tejto sekcii kódu 7 popisujeme postup inicializácie sériovej komunikácie s Arduino a zároveň nastavenie periódy vzorkovania. Je v záujme čitateľa predom porozumieť, fungovanie sériovej komunikácie a poskytnutého kódu pred snahou modifikácie tejto časti kódu. V skratke, otvárame sériovú komunikáciu 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 zároveň 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 uhol 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 môžeme natvrdo resetovať Arduino.

8 Počiatočné hodnoty

Výpis kódu 8: Zaznamenanie počiatočných hodnôt.

```

1 % -----
2 % Extract the initial values from the received data
3 % -----
4 plant_time_init = serLineList(1);
5 plant_potentiometer_init = serLineList(2);
6 plant_output_init = serLineList(3);
7 plant_input_init = serLineList(4);
8
9 plant_time = serLineList(1) - plant_time_init;
10 plant_input = serLineList(2);
11 plant_output = serLineList(3);
12 plant_potentiometer = R_WANTED + serLineList(4)/100*20;
13 plant_dt = serLineList(5);
14
15 timer_yhat = [timer_yhat, plant_output];
16 timer_dyhat = [timer_dyhat, 0];
17
18 % Display the received data
19 tmp_printlist = [0, plant_time, plant_potentiometer, plant_output,
20                 plant_input, plant_dt, T_sample];
21 doUpdate(tmp_printlist);
22 % -----
23 % -----

```

Info.

9 Definícia premenných hlavne slučky

Výpis kódu 9: Nastavenie premenných v hlavnej slučke.

```

1 % -----
2 % Set the main loop parameters
3 % -----
4
5 % Set initial control input value
6 e_old = 0;
7 e_int_old = 0;
8 u = 0;
9 u_send = u;
10
11
12 % Get the initial time
13 time_start = datetime('now');
14 time_tick = time_start;
15
16 % -----
17 % -----

```

Info.

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 % -----
```

Info.

11 Spracovanie sériovej komunikácie

Výpis kódu 11: Spracovanie a zaznamenanie 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,
34                 plant_output, plant_input, plant_dt, time_delta];
35 doUpdate(tmp_printlist);
36
37 % -----
```

Info.

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;
```

Info.

13 Saturácia akčného zásahu

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

```
1 % -----
2 % Saturate the control output to the MAX and MIN values
3 % -----
4
5 u_send = u;
6
7 if u_send > U_MAX
8     u_send = U_MAX;
9 elseif u_send < U_MIN
10    u_send = U_MIN;
11 end
12
13 % -----
14 % -----
```

Info.

14 Posielanie sériovej komunikácie

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

```
1 % -----
2 % Send control input to the serial port
3 % -----
4
5 write(serPort, u_send, "single");
6
7 % -----
8 % -----
```

Info.

15 Konečná podmienka merania

Výpis kódu 15: Podmienka na bezpečné ukončenie merania.

```
1 % -----
2 % Check if the simulation should stop (safety precaution)
```



```

3 % -----
4
5 if time_elapsed >= T_stop || plant_output >= Y_SAFETY
6     configureCallback(serialPort, "off"); % Remove the callback from the
7     serial port, before exiting the loop
8     break;
9 end
10 % -----
11 % -----

```

Info.

16 Ukončenie časovačov

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

```

1 % -----
2 % Close and delete all the existing timers
3 % -----
4
5 for tim=timerfindall
6     stop(tim);
7     delete(tim);
8 end
9
10 % -----
11 % -----

```

Info.

17 Ukončenie komunikácie

Výpis kódu 17: Ukončenie sériovej a súborovej komunikácie

```

1 % -----
2 % Close the serial connection
3 % -----
4
5 % Send a final command and close the serial port
6 write(serialPort, 0.0, 'single');
7 serialPort.flush("input");
8 clear serialPort;
9 fclose(datafileID);
10 clear datafileID;
11
12 % -----
13 % -----

```

Info.

18 Uloženie merania

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

```

1 % -----
2 % Save the measurement into a .MAT file for easier access to data when
3 % using Matlab
4 % -----
5 logargout = readtable(FILEPATH, "VariableNamingRule","preserve","Delimiter",
6     ",",",");
7 save(FILEPATH_MAT);

```

```

8
9 % -----
10 % -----

```

Info.

19 Vykreslenie priebehu merania

Výpis kódu 19: Vykreslenie základných veličín procesu merania.

```

1 % -----
2 % Quickly plot the measurement - reference, output, and control signal
3 % -----
4
5 t = logsout.t;
6 y = logsout.y;
7 u = logsout.u;
8 r = logsout.r;
9 e = r - y;
10 dt = logsout.dtp;
11
12
13 figure(111);
14 hold on;
15 plot(t, y, '-k', 'LineWidth', 1.5);
16 plot(t, r, '-r', 'LineWidth', 1.5);
17 plot(t, u, '-b', 'LineWidth', 1.5);
18 title('Control Response');
19 subtitle("P = " + num2str(P) + ", I = " + num2str(I) + ", D = " +
    num2str(D));
20 legend('y(t)', 'ref(t)', 'u(t)', "Location", "best");
21 xlabel('t [s]');
22 ylabel('y [deg]');
23 grid on;
24 hold off;
25 % -----
26 % -----

```

Info.

20 Vykreslenie priebehu $\alpha - \beta$ filtra

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

```

1 % -----
2 %% Plot the data
3 % -----
4
5 figure(100);
6 subplot(3, 1, 1);
7 plot(t, y, t, yhat, t, potentiometer, 'LineWidth', 1.5);
8 grid minor;
9 legend('y', 'yhat', 'ref');
10 xlabel('t [s]');
11 ylabel('$\varphi [\circ]$', 'Interpreter', 'latex');
12 title('System response');
13 subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');
14
15 subplot(3, 1, 2);
16 plot(t, dyhat, 'LineWidth', 1.5);
17 grid minor;
18 xlabel('t [s]');
19 ylabel('$\omega [\circ/s]$', 'Interpreter', 'latex');
20 title('System velocity response');
21 subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');
22
23 subplot(3, 1, 3);

```

```

24 plot(t, (y-yhat), 'LineWidth', 1.5);
25 grid minor;
26 xlabel('t [s]');
27 ylabel('$\varphi [\circ]$', 'Interpreter', 'latex');
28 title('Observer error');
29 subtitle("$\alpha - \beta$ filter", 'Interpreter', 'latex');
30
31 % -----
32 % -----

```

Info.