STU FEI
ÚSTAV ROBOTIKY A KYBERNETIKY

Kolekcia učebných textov

september 2025
github.com/OkoliePracovnehoBodu/KUT
RJ

KUTdevAS2 v1.0

Laboratórne zariadenie AeroShield: firmváre PIO

Cieľom textu je opis firmwarovej stránky *AS* implementovaná na Arduino UNO R3, ktorá slúži ako komunikačný most medzi Matlab-om a fyzickým zariadením *AS*.

1 Popis firmváru

Výpis kódu 1: Sample Code Listing C++

```
#include <AeroShield.h>

// --------------------------------
// Define Timer related macros, functions
// --------------------------------

#define CPUF 16000000UL                                          // CPU frequency for Arduino UNO
#define DIF(Tms) (1000.0f / Tms)                                 // Frequency divider macro
#define CMR(PS, DIF) (static_cast<int>(CPUF / PS / DIF) - 1) // Compare match register value calculation macro
#define PS(CMR, DIR) ((CPUF / (CMR + 1)) / DIR)                  // Prescaler value calculation macro
#define CHECK_TIMER(CMR) (CMR < 256)                             // Check if CMR value fits in 8-bit register
#define CURRENT_PS 1024 // Current prescaler value

// --------------------------------
// --------------------------------

// --------------------------------
// Built-in LED configs
// --------------------------------

const static int BUILT_IN_LED_PIN = 13; // PIN pre zabudovanu LED
const static unsigned long T_sample = 1000;
const static unsigned long LED_onTime = 15;
bool LED_on = false;

// --------------------------------
// --------------------------------


// --------------------------------
// Required variables and constexpr functions
// --------------------------------

byte Ts_user = 1; // sampling time in ms

constexpr float TS_INTERVAL_MIN(byte Ts)
{
    return (static_cast<float>(Ts) * 1000.0f) * 0.95f;
```

```
39  }
40
41  float uSig = 0.0f, ySig = 0.0f, potSig = 0.0f, dtoffset = 0.0f,
        dtoffset_interval = 0.0f; // control, output, reference, time offset
42
43  unsigned long currentTime, dt, lastTime;
44
45  // --------------------------------
46  // --------------------------------
47
48
49  // --------------------------------
50  // API calls
51  // --------------------------------
52
53  void updateOutput()
54  {
55      ySig = AeroShield.sensorReadDegree();
56  }
57
58  void updatePotentiometer()
59  {
60      potSig = AeroShield.referenceRead();
61  }
62
63  void updateControl(const float &u)
64  {
65      uSig = u;
66  }
67
68  void writeControl()
69  {
70      AeroShield.actuatorWrite(uSig);
71  }
72
73  // --------------------------------
74  // --------------------------------
75
76
77  // --------------------------------
78  // Write data into serial comms
79  // --------------------------------
80
81  void printData()
82  {
83      dt = currentTime - lastTime;
84      dtoffset = dtoffset_interval - dt;
85      if (dtoffset > 0)
86      {
87          if (dtoffset > 1000)
88          {
89              dtoffset = static_cast<unsigned long>(floor(dtoffset / 1000)
                    ); // round to nearest 1ms
90              delay(dtoffset);
91          }
92          else
93          {
94              dtoffset = static_cast<unsigned long>(floor(dtoffset / 100)
                    * 100); // round to nearest 100us
95              delayMicroseconds(dtoffset);
96          }
97          currentTime = micros();
98          dt = currentTime - lastTime;
99      }
100     lastTime = currentTime;
101
102     Serial.print(currentTime);
103     Serial.print("␣");
104     Serial.print(uSig);
105     Serial.print("␣");
106     Serial.print(ySig);
107     Serial.print("␣");
108     Serial.print(potSig);
109     Serial.print("␣");
```

```
110     Serial.print(dt);
111     Serial.println();
112 }
113
114 // --------------------------------
115 // --------------------------------
116
117
118 // --------------------------------
119 // Read new values from serial com and inputs
120 // --------------------------------
121 void readData()
122 {
123     updateOutput();
124     updatePotentiometer();
125 }
126
127 int recvByte()
128 {
129     if (Serial.available() >= 4)
130     {
131         Serial.readBytes((char *)&uSig, 4); // Read 4 bytes from Serial
132         return 0;                           // Success
133     }
134     return 1; // No new data received
135 }
136
137 void processNewData()
138 {
139     if (recvByte())
140     {
141         currentTime = micros();
142
143         readData(); // Read the sensor and reference values
144
145         writeControl(); // Write the control signal to the actuator
146
147         printData();
148     }
149 }
150
151
152 // --------------------------------
153 // --------------------------------
154
155
156 // --------------------------------
157 // Setup the Arduino board
158 // --------------------------------
159
160 void setup()
161 {
162     Serial.begin(115200); // Initialize Serial communication at 115200
        baud rate
163     Serial.flush();
164     pinMode(BUILT_IN_LED_PIN, OUTPUT); // for LED
165
166     AeroShield.begin();
167     AeroShield.calibrate();
168
169     Serial.println("---␣MCU␣config␣---");
170     while (!Serial.available())
171     {
172         delay(1); // Wait for user to open terminal
173     }
174
175     Ts_user = Serial.read(); // sampling time in ms
176
177     if (Ts_user == 0)
178     {
179         Ts_user = 1; // minimum 1 ms
180     }
181
182     if (Ts_user > 10)
```

```
183     {
184         dtoffset_interval = static_cast<float>(Ts_user) * 1000.0f;
185     }
186     else
187     {
188         dtoffset_interval = TS_INTERVAL_MIN(Ts_user);
189     }
190 
191     // Calculate the timers
192     const int cmr = CMR(CURRENT_PS, DIF(T_sample));
193 
194     cli();
195 
196     // Timer1 for LED blink
197     TCCR1A = 0; // Normal mode
198     TCCR1B = 0;
199     TCNT1 = 0;                          // Initialize counter value to
            0
200     OCR1A = cmr;                        // Set compare match register
            for desired timer count
201     TCCR1B |= (1 << WGM12);             // CTC mode
202     TCCR1B |= (1 << CS12) | (1 << CS10); // Set prescaler to 1024 and
            start the timer
203     TIMSK1 |= (1 << OCIE1A);            // Enable timer compare
            interrupt
204 
205     sei();
206 
207     Serial.println("---␣MCU␣starting␣[" + String(Ts_user) + "]␣---");
208     currentTime = micros();
209     lastTime = currentTime - (Ts_user * 1000); // better for statistics
210 
211     // Initial reads and writes
212     readData();
213     updateControl(0.0f);
214     writeControl();
215 
216     printData();
217 }
218 
219 // -------------------------------
220 // -------------------------------
221 
222 
223 // -------------------------------
224 // Timer 1 callback function for when triggered
225 // -------------------------------
226 
227 ISR(TIMER1_COMPA_vect)
228 {
229     LED_on = !LED_on;
230     digitalWrite(BUILT_IN_LED_PIN, LED_on);
231 }
232 
233 // -------------------------------
234 // -------------------------------
235 
236 
237 void loop()
238 {
239     processNewData();
240 }
```

```
1 for (int i=0; i<iterations;i++)
2 {
3     do something
4 }
```

** TODO: Dopísať **

AS je laboratórne zariadenie predstavujúce reálny dynamický systém, ktorý sme predstavili v KUT dokumente Laboratórne zariadenie AeroShield:
orientačný prehľad. V prípade, že chceme odmerať statické vlastnosti systému, musíme vedieť rozsahy vstupov a výstupov, aby sme boli schopný korektne navrhnúť experiment

na zmeranie vlastností systému. V prípade AS vieme, že vstupný signál je v rozsahu $0 - 255$, musíme si zvoliť rozumnú veľkosť kroku a dĺžku trvania kroku, aby sme dostatočne zachytili vlastnosti systému. Zvoľme si teda podľa tab. 1.

Tabuľka 1: Nastavenia merania

Nastavenie	Hodnota	Jednotka
Veľkosť kroku	1	-
Trvanie	5	s

Nakoľko meriame statické vlastnosti, tak nás nezaujíma priebeh výstupného signálu v neustálenom stave. To znamená napr., ak vstupná hodnota je 10 a už sme pre túto hodnotu ukončili meranie, a chceme sa presunúť na hodnotu 50, tak akokoľvek sa na žiadanu hodnotu akčného zásahu dostaneme, je irelevantné. Nakoľko, musíme meranie uskutočniť až v čase, keď výstupný signál z AS je ustálený - ukážeme na príklade.

2 Meranie statickej charakteristiky

** TODO: Dopísať **

Z opisu predmetného dynamického systému vyplýva, že systém má jeden výstupný signál, jeden vstupný signál a manuálne nastaviteľnú polohu potenciometra.

- *Vstupný signál* nadobúda hodnoty v rozmedzí 0 až 255.
- *Výstupný signál* nadobúda hodnoty v rozsahu $-50°$ az $210°$.
- *Signal z potenciometra* nadobúda hodnoty v rozsahu 0 až 1023.

Tabuľka 2: Rozsahy a jednotky signálov

Signál	Rozsah hodnôt	Jednotka
Vstup	0 až 255	-
Výstup	$-50°$ až $210°$	$°$
Potenciometer	0 až 1023	-

3 Schematické znázornenie systému

AS_system.pdf

Obr. 1: Signály systému AS.

AS_system_pinout.pdf

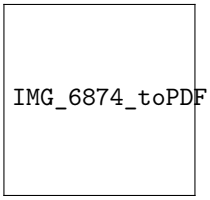
Obr. 2: Schéma pripojenia laboratórneho zariadenia AS ku GPIO pinom Arduino UNO R3.

Na obr. 2 môžeme vidieť, že A0 je analógový vstup potenciometra, ∼D5 je fyzický výstup akčného zásahu na logickej úrovni vo forme *PWM* (Pulse-Width Modulation), I2C slúži na odčítanie uhlovej polohy ramena.

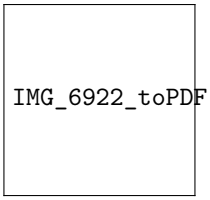
4 Fotografie

Zoznam fotografií:

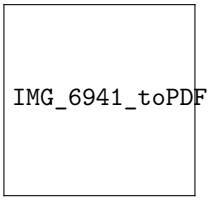
- Obr. 3: Celkový pohľad na laboratórne zariadenie LMOT.
- Obr. 4: Motor a tachodynamo.
- Obr. 5: Elektronické obvody zariadenia.
- Obr. 6: Predný panel so svorkami a potenciometrom.



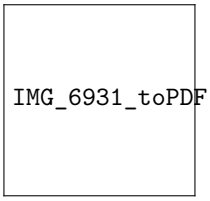
Obr. 3: Celkový pohľad na laboratórne zariadenie LMOT.



Obr. 4: Motor a tachodynamo.



Obr. 5: Elektronické obvody zariadenia.



Obr. 6: Predný panel so svorkami a potenciometrom.