

september 2025

github.com/OkoliePracovnehoBodu/KUT  
RJ

**KUT**devAS2<sup>V1.0</sup>

## Laboratórne zariadenie AeroShield: firmwára PIO

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

### 1 Pomocné funkcie časovačov

Výpis kódu 1: Obslužné funkcie interných časovačov Arduina.

```
1 // -----
2 // Define Timer related macros, functions
3 // -----
4
5 #define CPUF 16000000UL // CPU
6 // frequency for Arduino UNO
7 #define DIF(Tms) (1000.0f / Tms) //
8 // Frequency divider macro
9 #define CMR(PS, DIF) (static_cast<int>(CPUF / PS / DIF) - 1) // Compare
10 // match register value calculation macro
11 #define PS(CMR, DIR) ((CPUF / (CMR + 1)) / DIR) //
12 // Prescaler value calculation macro
13 #define CHECK_TIMER(CMR) (CMR < 256) // Check if
14 // CMR value fits in 8-bit register
15 #define CURRENT_PS 1024 // Current prescaler value
16
17 // -----
18 // -----
```

Definícia premenných a pomocných funkcií na prácu a nastavenie interných časovačov Arduina - viď blok 1.

### 2 Premenné zabudovanej LED

Výpis kódu 2: Definície premenných zabudovanej LED.

```
1 // -----
2 // Built-in LED configs
3 // -----
4
5 const static int BUILT_IN_LED_PIN = 13; // PIN pre zabudovanu LED
6 const static unsigned long T_sample = 1000;
7 const static unsigned long LED_onTime = 15;
8 bool LED_on = false;
9
10 // -----
11 // -----
```

Definície premenných na zabezpečenie blikania v kontexte zabudovanej LED na zadefinovanej frekvencii - viď výpis 2, priradenie konštánt pre zjednodušenie následného použitia premenných v kóde. LED\_on premenná zabezpečuje prepínanie stavu LED medzi emitovaním a zhasnutým stavom.

### 3 Premenné signálov a komunikácie

Výpis kódu 3: Definície premenných signálov a komunikácie.

```
1 // -----
2 // Required variables and constexpr functions
3 // -----
4
5 byte Ts_user = 1; // sampling time in ms
6
7 constexpr float TS_INTERVAL_MIN(byte Ts)
8 {
9     return (static_cast<float>(Ts) * 1000.0f) * 0.95f;
10 }
11
12 float uSig = 0.0f, ySig = 0.0f, potSig = 0.0f, dtoffset = 0.0f,
13     dtoffset_interval = 0.0f; // control, output, reference, time offset
14
15 unsigned long currentTime, dt, lastTime;
16
17 // -----
18 // -----
```

Definícia podstatných premenných, ktoré používame na uloženie terajších meraní, konfigurácia časovaní posielania dát, nakoniec ukladanie času trvania merania pre Arduino a diferenciaciu času medzi terajšou komunikáciou a predošlou, môžeme vidieť na výpise kódu 3. Hodnotu diferencie času sa používa na časovú synchronizáciu komunikácie, aby sa perióda vzorkovania hýbala v okolí žiadanej periódy vzorkovania, nakoľko zabezpečenie presnej periódy vzorkovania si vyžaduje hardwarové riešenie.

### 4 Metódy čítania a zapisovania

Výpis kódu 4: Obslužné funkcie čítania a zapisovania signálov.

```
1 // -----
2 // API calls
3 // -----
4
5 void updateOutput()
6 {
7     ySig = AeroShield.sensorReadDegree();
8 }
9
10 void updatePotentiometer()
11 {
12     potSig = AeroShield.referenceRead();
13 }
14
15 void updateControl(const float &u)
16 {
17     uSig = u;
18 }
19
20 void writeControl()
21 {
22     AeroShield.actuatorWrite(uSig);
23 }
24
25 // -----
26 // -----
```

Funkcie, ktoré môžeme vidieť na výpise kódu 4 slúžia iba na prehľadnosť a čistotu kódu. Zabezpečujú zapisovanie hodnôt do definovaných premenných signálov alebo fyzikálnu realizáciu žiadanej akčnej veličiny v tomto prípade funkcia `writeControl()`.

## 5 Sériová komunikácia

Výpis kódu 5: Funkcia na posielanie dát po sériovej linke.

```
1  // -----
2  // Write data into serial comms
3  // -----
4
5  void printData()
6  {
7      dt = currentTime - lastTime;
8      dtoffset = dtoffset_interval - dt;
9      if (dtoffset > 0)
10     {
11         if (dtoffset > 1000)
12         {
13             dtoffset = static_cast<unsigned long>(floor(dtoffset / 1000)
14             ); // round to nearest 1ms
15             delay(dtoffset);
16         }
17         else
18         {
19             dtoffset = static_cast<unsigned long>(floor(dtoffset / 100)
20             * 100); // round to nearest 100us
21             delayMicroseconds(dtoffset);
22         }
23         currentTime = micros();
24         dt = currentTime - lastTime;
25     }
26     lastTime = currentTime;
27
28     Serial.print(currentTime);
29     Serial.print("_");
30     Serial.print(uSig);
31     Serial.print("_");
32     Serial.print(ySig);
33     Serial.print("_");
34     Serial.print(potSig);
35     Serial.print("_");
36     Serial.print(dt);
37     Serial.println();
38 }
39 // -----
40 // -----
```

Kód 5 je komunikačná funkcia, ktorá zabezpečuje posielanie meraných veličín po sériovej linke von - do Matlab-u. Zároveň implementuje algoritmus, ktorý zabezpečuje komunikáciu s kvázi stabilnou periódou vzorkovania a odosiela dáta čo najskôr. Nakoľko dĺžka trvania jedného cyklu `loop()` je variabilná a závisí od tohto času následná komunikácia, tak sme časovanie implementovali spôsobom: ak program môže poslať dáta skôr ako je žiadané, tak čaká `dtoffset` čas a následne posiela dáta, v prípade, že je neskôr, tak okamžite posiela dáta, ktoré sú najnovšie v pamäti.

## 6 Čítanie signálov, sériovej linky a logika riadenia merania

Výpis kódu 6: Čítanie signálov

```
1  // -----
2  // Read new values from serial com and inputs
3  // -----
4  void readData()
```

```

5 {
6     updateOutput();
7     updatePotentiometer();
8 }
9
10 int recvByte()
11 {
12     if (Serial.available() >= 4)
13     {
14         Serial.readBytes((char *)&uSig, 4); // Read 4 bytes from Serial
15         return 0;                          // Success
16     }
17     return 1; // No new data received
18 }
19
20 void processNewData()
21 {
22     if (recvByte())
23     {
24         currentTime = micros();
25
26         readData(); // Read the sensor and reference values
27
28         writeControl(); // Write the control signal to the actuator
29
30         printData();
31     }
32 }
33 // -----
34 // -----

```

V bloku kódu 6 sú 3 rôzne funkcie, z čoho sú 2 na čítanie dát a posledná na riadenia komunikácie medzi Matlabom, a AeroShield-om. V prvej funkcii `readData()` čítame hodnoty zo samotného AeroShield-u, teda fyzikálne veličiny ako natočenie vrtulky a potenciometra. Funkcia `recvByte()` zabezpečuje čítanie žiadanej akčnej veličiny z Matlab-u po sériovej linke, čítanie je priamo implementované pre dátový typ `float32`. Nakoniec funkcia `processNewData()` zabezpečuje, meranie veličín zariadenia, spracovanie požiadaviek z Matlab-u a následnú fyzikálnu realizáciu - PWM signál posielať priamo na *Gate* pin MOSFET tranzistoru a odoslanie nameraných dát späť po sériovej linke.

## 7 Inicializácia Arduina

Výpis kódu 7: Inicializačná metóda Arduina.

```

1 // -----
2 // Setup the Arduino board
3 // -----
4 void setup()
5 {
6     Serial.begin(115200); // Initialize Serial communication at 115200
7     Serial.flush();
8     pinMode(BUILT_IN_LED_PIN, OUTPUT); // for LED
9
10    AeroShield.begin();
11    AeroShield.calibrate();
12
13    Serial.println("---MCU_config---");
14    while (!Serial.available())
15    {
16        delay(1); // Wait for user to open terminal
17    }
18
19    Ts_user = Serial.read(); // sampling time in ms
20
21    if (Ts_user == 0)
22    {
23        Ts_user = 1; // minimum 1 ms

```

```

24     }
25
26     if (Ts_user > 10)
27     {
28         dtoffset_interval = static_cast<float>(Ts_user) * 1000.0f;
29     }
30     else
31     {
32         dtoffset_interval = TS_INTERVAL_MIN(Ts_user);
33     }
34
35     // Calculate the timers
36     const int cmr = CMR(CURRENT_PS, DIF(T_sample));
37
38     cli();
39
40     // Timer1 for LED blink
41     TCCR1A = 0; // Normal mode
42     TCCR1B = 0;
43     TCNT1 = 0; // Initialize counter value to
44                0
45     OCR1A = cmr; // Set compare match register
46     // for desired timer count
47     TCCR1B |= (1 << WGM12); // CTC mode
48     TCCR1B |= (1 << CS12) | (1 << CS10); // Set prescaler to 1024 and
49                start the timer
50     TIMSK1 |= (1 << OCIE1A); // Enable timer compare
51                interrupt
52
53     sei();
54
55     Serial.println("---MCU_starting_" + String(Ts_user) + "_---");
56     currentTime = micros();
57     lastTime = currentTime - (Ts_user * 1000); // better for statistics
58
59     // Initial reads and writes
60     readData();
61     updateControl(0.0f);
62     writeControl();
63     printData();
64 }
65 // -----
66 // -----

```

V rámci inicializačnej funkcie 7 sa nastavuje komunikácia po sériovej linke (Universal Serial Bus - *USB*), módy GPIO pinov, inicializácia AeroShield firmwáru, perióda vzorkovania - posielá Matlab v našom prípade, výpočet dovoľeného oskorenia/oneskorenia posielania dát - slúži na zabezpečenie najrýchlejšej možnej komunikácie pre žiadanú periódu vzorkovania, časovač preblikávania zabudovanej LED, po celkovej inicializácii posielame správu o začiatku merania a nakoniec posielame počiatočný stav zariadenia. Následne po poslaní počiatočného stavu sa začne volať funkcia `loop()` samotným Arduino, ktorá následne riadi dej merania.

## 8 Obslužná funkcia časovaču (spínanie LED)

Výpis kódu 8: Spínanie LED v obslužnej funkcii *Časovača 1*.

```

1 // -----
2 // Timer 1 callback function for when triggered
3 // -----
4 ISR(TIMER1_COMPA_vect)
5 {
6     LED_on = !LED_on;
7     digitalWrite(BUILT_IN_LED_PIN, LED_on);
8 }
9 // -----
10 // -----

```

Metóda zobrazená vo výpise 8 je špeciálne definovaná samotným Arduino, na ktorú sme sa pripojili, aby sa následne volal kód, ktorý sme vo vnútri tejto funkcie definovali. Funkcia je volaná iba vtedy, keď sa Časovač 1 zopne, podľa toho ako si definujeme interval zopínania časovača v inicializačnej funkcii.

## 9 Hlavná slučka firmwáru

Výpis kódu 9: Hlavná slučka Arduino programu.

```
1 void loop()
2 {
3     processNewData();
4 }
```

Funkciu znázornenú vo výpise kódu 9 možno nazvať hlavnou funkciou celého programu, nakoľko pomocou tejto funkcie sa Arduino rozhoduje, čo má v každom čase po inicializácii robiť. Funkciu `loop()` si môžeme predstaviť ako nekonečnú slučku, ktorú Arduino volá vždy po inicializácii. To znamená, že v tejto funkcii definujeme, čo má Arduino robiť. V našom prípade sa volá funkcia `processNewData()`, ktorá sa stará o riadenie komunikácie medzi sériovou linkou a výstup/vstupom Arduina.