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 Pomocné funkcie časovačov

Výpis kódu 1: Obslužné funkcie interných časovačov Arduina.

```
// --------------------------------
// 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

// --------------------------------
// --------------------------------
```

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.

```
// --------------------------------
// 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;

// --------------------------------
// --------------------------------
```

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.

```
// --------------------------------
// 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;
}

float uSig = 0.0f, ySig = 0.0f, potSig = 0.0f, dtoffset = 0.0f,
    dtoffset_interval = 0.0f; // control, output, reference, time offset

unsigned long currentTime, dt, lastTime;

// --------------------------------
// --------------------------------
```

Definícia podstatných premenných, ktoré používame na uloženie terajších meraní, konfigurácia časovaní posielania dát, nakoniec ukladanie času trvania merania pre Arduino a diferencia 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.

```
// --------------------------------
// API calls
// --------------------------------

void updateOutput()
{
    ySig = AeroShield.sensorReadDegree();
}

void updatePotentiometer()
{
    potSig = AeroShield.referenceRead();
}

void updateControl(const float &u)
{
    uSig = u;
}

void writeControl()
{
    AeroShield.actuatorWrite(uSig);
}

// --------------------------------
// --------------------------------
```

Funkcie, ktoré možeme vidieť na výpise kódu 4 slúžia iba na prehladnosť a čistosť 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.

```
// ------------------------------
// Write data into serial comms
// ------------------------------

void printData()
{
    dt = currentTime - lastTime;
    dtoffset = dtoffset_interval - dt;
    if (dtoffset > 0)
    {
        if (dtoffset > 1000)
        {
            dtoffset = static_cast<unsigned long>(floor(dtoffset / 1000)
                ); // round to nearest 1ms
            delay(dtoffset);
        }
        else
        {
            dtoffset = static_cast<unsigned long>(floor(dtoffset / 100)
                * 100); // round to nearest 100us
            delayMicroseconds(dtoffset);
        }
        currentTime = micros();
        dt = currentTime - lastTime;
    }
    lastTime = currentTime;

    Serial.print(currentTime);
    Serial.print(" ");
    Serial.print(uSig);
    Serial.print(" ");
    Serial.print(ySig);
    Serial.print(" ");
    Serial.print(potSig);
    Serial.print(" ");
    Serial.print(dt);
    Serial.println();
}

// ------------------------------
// ------------------------------
```

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

```
// ------------------------------
// Read new values from serial com and inputs
// ------------------------------
```

```
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 posledna na riadenia komunikácie medzi Matlabom, a AeroShield-om. V prvej funkcii `readData()` čítame hodnoty zo samotného AeroShield-u, teda fyzikálne veličiny ako natočenie vrtuľky 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 posielaný 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
           baud rate
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
            0
44      OCR1A = cmr;                        // Set compare match register
            for desired timer count
45      TCCR1B |= (1 << WGM12);             // CTC mode
46      TCCR1B |= (1 << CS12) | (1 << CS10); // Set prescaler to 1024 and
            start the timer
47      TIMSK1 |= (1 << OCIE1A);            // Enable timer compare
            interrupt
48
49      sei();
50
51      Serial.println("--- MCU starting [" + String(Ts_user) + "] ---");
52      currentTime = micros();
53      lastTime = currentTime - (Ts_user * 1000); // better for statistics
54
55      // Initial reads and writes
56      readData();
57      updateControl(0.0f);
58      writeControl();
59
60      printData();
61  }
62  // -------------------------------
63  // -------------------------------
```

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