Program jednotky pro příjem naměřených dat

```
#include <SPI.h>
#include "RF24.h"
#include <U8glib.h>

// OLED displej
U8GLIB_SSD1306_128X64 mujOled(U8G_I2C_OPT_NONE);

// nRF24L01
#define CE 7
#define CS 8
RF24 radio(CE, CS);

// Adresy tří vysílačů
byte adresaRychlosti[] = "RYCH1";
byte adresaTeplota[]   = "TEMP1";
byte adresaHmotnost[]  = "HMOT1";

// Tlačítko
#define BUTTON_PIN 2
volatile int mode = 0;
unsigned long lastPress = 0;
const unsigned long debounce = 300;

// Data
float v1 = 0, v2 = 0, a = 0, v_avg = 0;
float T = 0;
float m = 0;

void setup() {
```

```
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), zmenMod, FALLING);

  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(76);

  nastavPrijem();
  radio.startListening();

  Serial.println("READY");
}

void loop() {
  if (Serial.available()) {
    char prikaz = Serial.read();
    if (prikaz == 'M') {
      zmenMod();
    }
  }

  if (radio.available()) {
    if (mode == 0) {
      float prijataData[4];
      radio.read(&prijataData, sizeof(prijataData));
      v1 = prijataData[0];
      v2 = prijataData[1];
      a  = prijataData[2];
      v_avg = prijataData[3];
```

```
    Serial.print("V:");
    Serial.print(v1, 2); Serial.print(",");
    Serial.print(v2, 2); Serial.print(",");
    Serial.print(a, 2);  Serial.print(",");
    Serial.println(v_avg, 2);

  } else if (mode == 1) {
    float teplota;
    radio.read(&teplota, sizeof(teplota));
    T = teplota;

    Serial.print("T:");
    Serial.println(T, 2);

  } else if (mode == 2) {
    float hmotnost;
    radio.read(&hmotnost, sizeof(hmotnost));
    m = hmotnost;

    Serial.print("M:");
    Serial.println(m, 2);
  }
}

mujOled.firstPage();
do {
  vykresli();
} while (mujOled.nextPage());
```

```
  delay(500);
}

void zmenMod() {
  unsigned long aktualniCas = millis();
  if (aktualniCas - lastPress > debounce) {
    mode = (mode + 1) % 3;
    nastavPrijem();
    lastPress = aktualniCas;
  }
}

void nastavPrijem() {
  radio.stopListening();
  if (mode == 0) {
    radio.openReadingPipe(1, adresaRychlosti);
  } else if (mode == 1) {
    radio.openReadingPipe(1, adresaTeplota);
  } else if (mode == 2) {
    radio.openReadingPipe(1, adresaHmotnost);
  }
  radio.startListening();
}

void vykresli() {
  mujOled.setFont(u8g_font_unifont);

  if (mode == 0) {
    mujOled.setPrintPos(2, 15);
    mujOled.print("v1: "); mujOled.print(v1, 2); mujOled.print(" m/s");
```

```
    mujOled.setPrintPos(2, 30);
    mujOled.print("v2: "); mujOled.print(v2, 2); mujOled.print(" m/s");

    mujOled.setPrintPos(2, 45);
    mujOled.print("a : "); mujOled.print(a, 2); mujOled.print(" m/s2");

    mujOled.setPrintPos(2, 60);
    mujOled.print("vavg: "); mujOled.print(v_avg, 2); mujOled.print(" m/s");

  } else if (mode == 1) {
    mujOled.setPrintPos(2, 30);
    mujOled.print("Teplota: ");
    mujOled.setPrintPos(2, 45);
    mujOled.print(T, 2); mujOled.print(" C");
  } else if (mode == 2) {
    mujOled.setPrintPos(2, 30);
    mujOled.print("Hmotnost: ");

    mujOled.setPrintPos(2, 45);
    mujOled.print(m, 2); mujOled.print(" g");
  }
}
```

Program pro měření rychlosti a zrychlení optickou bránou

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include "RF24.h"

LiquidCrystal_I2C lcd(0x27, 20, 4);

// Tlačítka
#define BTN_UP     A0
#define BTN_DOWN   A1
#define BTN_LEFT   A2
#define BTN_RIGHT  A3
#define BTN_ENTER  9
#define BTN_BACK   10

// Senzory
#define SENSOR1_PIN 4
#define SENSOR2_PIN 5
#define SENSOR3_PIN 6

// nRF24L01
#define CE 7
#define CS 8
RF24 nRF(CE, CS);
byte adresaPrijimac[] = "prijimac00";
byte adresaVysilac[] = "RYCH1";

// Menu
const char* options[] = {
```

```
  "3 brany 10cm",
  "3 brany nastav",
  "2 brany nastav"
};
int menuIndex = 0;
const int menuSize = sizeof(options) / sizeof(options[0]);
int currentMode = 0;
bool inSelection = false;

// Stav tlačítek
bool upLast = HIGH, downLast = HIGH, leftLast = HIGH, rightLast = HIGH, enterLast = HIGH, backLast = HIGH;
bool wasPressed(bool current, bool &last) {
  bool pressed = (current == LOW && last == HIGH);
  last = current;
  return pressed;
}

// Globální parametry
float parameter = 0.1;
bool rezim1_initialized = false;
bool rezim2_initialized = false;

void setup() {
  lcd.init();
  lcd.backlight();

  pinMode(BTN_UP, INPUT_PULLUP);
  pinMode(BTN_DOWN, INPUT_PULLUP);
  pinMode(BTN_LEFT, INPUT_PULLUP);
  pinMode(BTN_RIGHT, INPUT_PULLUP);
```

```
  pinMode(BTN_ENTER, INPUT_PULLUP);
  pinMode(BTN_BACK, INPUT_PULLUP);

  pinMode(SENSOR1_PIN, INPUT_PULLUP);
  pinMode(SENSOR2_PIN, INPUT_PULLUP);
  pinMode(SENSOR3_PIN, INPUT_PULLUP);

  nRF.begin();
  nRF.setPALevel(RF24_PA_HIGH);
  nRF.openWritingPipe(adresaVysilac);
  nRF.openReadingPipe(1, adresaPrijimac);
  nRF.startListening();

  displayMenu();
}

void displayMenu() {
  lcd.clear();
  for (int i = 0; i < menuSize; i++) {
    lcd.setCursor(0, i);
    lcd.print(i == menuIndex ? "> " : "  ");
    lcd.print(options[i]);
  }
}

void loop() {
  bool up = digitalRead(BTN_UP);
  bool down = digitalRead(BTN_DOWN);
  bool left = digitalRead(BTN_LEFT);
  bool right = digitalRead(BTN_RIGHT);
```

```
bool enter = digitalRead(BTN_ENTER);
bool back = digitalRead(BTN_BACK);

if (!inSelection) {
  if (wasPressed(up, upLast) && menuIndex > 0) {
    menuIndex--; displayMenu();
  }
  if (wasPressed(down, downLast) && menuIndex < menuSize - 1) {
    menuIndex++; displayMenu();
  }
  if (wasPressed(enter, enterLast)) {
    inSelection = true;
    currentMode = menuIndex;
    lcd.clear(); lcd.setCursor(0, 0);
    lcd.print("Mod: "); lcd.print(options[currentMode]);
    delay(800); lcd.clear();

    // Reset příznaků režimů
    if (currentMode == 1) rezim1_initialized = false;
    if (currentMode == 2) rezim2_initialized = false;
  }
} else {
  switch (currentMode) {
    case 0: rezim_3_brany_10cm(); break;
    case 1: rezim_3_brany_nastav(); break;
    case 2: rezim_2_brany_nastav(); break;
  }

  if (wasPressed(back, backLast)) {
    inSelection = false;
```

```
      lcd.clear(); lcd.setCursor(0, 0); lcd.print("Zpet do menu...");
      delay(800); displayMenu();
    }
  }

  upLast = up; downLast = down;
  leftLast = left; rightLast = right;
  enterLast = enter; backLast = back;

  delay(20);
}

// ------------------ Režim 0: 3 brány 10 cm --------------------
void rezim_3_brany_10cm() {
  static unsigned long t1, t2, t3;
  static bool s1, s2, s3;
  static bool resultShown = false;
  static bool displayed = false;

    if (!displayed) {
    lcd.clear();
    lcd.setCursor(0, 1); lcd.print("Cekam na mereni...");
    displayed = true;
  }

  int r1 = digitalRead(SENSOR1_PIN);
  int r2 = digitalRead(SENSOR2_PIN);
  int r3 = digitalRead(SENSOR3_PIN);

  if (!s1 && r1 == LOW) { t1 = micros(); s1 = true; resultShown = false; }
```

```
if (!s2 && r2 == LOW) { t2 = micros(); s2 = true; }
if (!s3 && r3 == LOW) { t3 = micros(); s3 = true; }

if (s1 && s2 && s3 && !resultShown) {
  // Urči pořadí průchodu
  float d = 0.1;
  float v1, v2, a;

  if (t1 < t2 && t2 < t3) {
    // Směr: S1 → S2 → S3 (vpřed)
    float dt1 = (t2 - t1) / 1e6;
    float dt2 = (t3 - t2) / 1e6;
    v1 = d / dt1;
    v2 = d / dt2;
    a = (v2 - v1) / ((t3 - t1) / 1e6);
  } else if (t3 < t2 && t2 < t1) {
    // Směr: S3 → S2 → S1 (zpět)
    float dt1 = (t2 - t3) / 1e6;
    float dt2 = (t1 - t2) / 1e6;
    v1 = -(d / dt1); // záporné rychlosti
    v2 = -(d / dt2);
    a = (v2 - v1) / ((t1 - t3) / 1e6);
  } else {
    // Neplatný nebo neúplný směr – ignoruj
    s1 = s2 = s3 = false;
    return;
  }

  zobrazVysledky(v1, v2, a);
  resultShown = true;
```

```
    s1 = s2 = s3 = false;
  }
}

// ------------------ Režim 1: 3 brány nastav --------------------
void rezim_3_brany_nastav() {
  if (!rezim1_initialized) {
    rezim1_initialized = true;
    editParameter("Roztec bran (cm):", &parameter);

    lcd.clear();
    lcd.setCursor(0, 0); lcd.print("Zadano: ");
    lcd.print(parameter * 100, 1); lcd.print(" cm");
    lcd.setCursor(0, 1); lcd.print("Cekam na mereni...");
    delay(1000);
    return;
  }

  static bool s1 = false, s2 = false, s3 = false;
  static unsigned long t1 = 0, t2 = 0, t3 = 0;
  static bool resultShown = false;

  int r1 = digitalRead(SENSOR1_PIN);
  int r2 = digitalRead(SENSOR2_PIN);
  int r3 = digitalRead(SENSOR3_PIN);

  if (!s1 && r1 == LOW) { t1 = micros(); s1 = true; resultShown = false; }
  if (!s2 && r2 == LOW) { t2 = micros(); s2 = true; }
  if (!s3 && r3 == LOW) { t3 = micros(); s3 = true; }
```

```
  if (s1 && s2 && s3 && !resultShown) {
    float d = parameter;
    float v1 = 0, v2 = 0, a = 0;

    if (t1 < t2 && t2 < t3) {
      // Směr: S1 → S2 → S3
      float dt1 = (t2 - t1) / 1e6;
      float dt2 = (t3 - t2) / 1e6;
      v1 =  d / dt1;
      v2 =  d / dt2;
      a  = (v2 - v1) / ((t3 - t1) / 1e6);
    } else if (t3 < t2 && t2 < t1) {
      // Směr: S3 → S2 → S1
      float dt1 = (t2 - t3) / 1e6;
      float dt2 = (t1 - t2) / 1e6;
      v1 = -d / dt1;
      v2 = -d / dt2;
      a  = (v2 - v1) / ((t1 - t3) / 1e6);
    } else {
      // Neplatné pořadí – ignoruj
      s1 = s2 = s3 = false;
      return;
    }

    zobrazVysledky(v1, v2, a);
    resultShown = true;
    s1 = s2 = s3 = false;
  }
}
```

```
// ------------------ Režim 2: 2 brány nastav --------------------
void rezim_2_brany_nastav() {
  if (!rezim2_initialized) {
    rezim2_initialized = true;
    editParameter("Delka objektu (cm):", &parameter);

    lcd.clear();
    lcd.setCursor(0, 0); lcd.print("Zadano: ");
    lcd.print(parameter * 100, 1); lcd.print(" cm");
    lcd.setCursor(0, 1); lcd.print("Cekam na mereni...");
    delay(1000);
    return;
  }

  static unsigned long t_start1, t_end1, t_start2, t_end2;
  static int state = 0;

  int s1 = digitalRead(SENSOR1_PIN);
  int s2 = digitalRead(SENSOR2_PIN);

  // Směr S1 → S2
  if (state == 0) {
    if (s1 == LOW) { t_start1 = micros(); state = 1; }
    else if (s2 == LOW) { t_start2 = micros(); state = 4; }  // opačný směr
  }

  if (state == 1 && s1 == HIGH) { t_end1 = micros(); state = 2; }
  if (state == 2 && s2 == LOW) { t_start2 = micros(); state = 3; }
  if (state == 3 && s2 == HIGH) {
    t_end2 = micros(); state = 0;
```

```
    float v1 =  parameter / ((t_end1 - t_start1) / 1e6);
    float v2 =  parameter / ((t_end2 - t_start2) / 1e6);
    float deltaT = (t_start2 - t_start1) / 1e6;
    float a = (v2 - v1) / deltaT;

    zobrazVysledky(v1, v2, a);
  }

  // Směr S2 → S1
  if (state == 4 && s2 == HIGH) { t_end2 = micros(); state = 5; }
  if (state == 5 && s1 == LOW) { t_start1 = micros(); state = 6; }
  if (state == 6 && s1 == HIGH) {
    t_end1 = micros(); state = 0;

    float v1 = -parameter / ((t_end2 - t_start2) / 1e6);  // záporný směr
    float v2 = -parameter / ((t_end1 - t_start1) / 1e6);
    float deltaT = (t_start1 - t_start2) / 1e6;
    float a = (v2 - v1) / deltaT;

    zobrazVysledky(v1, v2, a);
  }
}

// ------------------ Editor vstupní hodnoty --------------------
void editParameter(const char* title, float* param) {
  char buffer[] = "010,0";  // výchozí hodnota v cm
  int cursor = 0;
  while (true) {
    lcd.clear();
```

```
lcd.setCursor(0, 0); lcd.print(title);
lcd.setCursor(0, 1);
for (int i = 0; i < 5; i++) {
  lcd.print(i == cursor ? '[' : ' ');
  lcd.print(buffer[i]);
  lcd.print(i == cursor ? ']' : ' ');
}
lcd.setCursor(0, 3);
lcd.print("ENTER=OK   ZPET=<-");

bool up = digitalRead(BTN_UP);
bool down = digitalRead(BTN_DOWN);
bool left = digitalRead(BTN_LEFT);
bool right = digitalRead(BTN_RIGHT);
bool enter = digitalRead(BTN_ENTER);
bool back = digitalRead(BTN_BACK);

if (wasPressed(left, leftLast)) {
  if (cursor > 0) {
    cursor--;
    if (cursor == 3) cursor--;  // přeskoč čárku
  }
}

if (wasPressed(right, rightLast)) {
  if (cursor < 4) {
    cursor++;
    if (cursor == 3) cursor++;  // přeskoč čárku
  }
}
```

```
if (wasPressed(up, upLast)) {
  if (buffer[cursor] >= '0' && buffer[cursor] <= '9')
    buffer[cursor] = (buffer[cursor] == '9') ? '0' : buffer[cursor] + 1;
}

if (wasPressed(down, downLast)) {
  if (buffer[cursor] >= '0' && buffer[cursor] <= '9')
    buffer[cursor] = (buffer[cursor] == '0') ? '9' : buffer[cursor] - 1;
}

if (wasPressed(enter, enterLast)) {
  for (int i = 0; i < 5; i++) if (buffer[i] == ',') buffer[i] = '.';
  *param = atof(buffer) / 100.0;
  lcd.clear(); lcd.setCursor(0, 0); lcd.print("Zadano: ");
  lcd.print(*param * 100, 1); lcd.print(" cm");
  delay(1000);
  break;
}

if (wasPressed(back, backLast)) {
  inSelection = false;
  rezim1_initialized = false;
  rezim2_initialized = false;
  displayMenu();
  break;
}

upLast = up; downLast = down;
leftLast = left; rightLast = right;
```

```
    enterLast = enter; backLast = back;
    delay(100);
  }
}


// ------------------ Výpis výsledků --------------------
void zobrazVysledky(float v1, float v2, float a) {
  float v_avg = (v1 + v2) / 2.0;

  lcd.clear();
  lcd.setCursor(0, 0); lcd.print("v1: "); lcd.print(v1, 2); lcd.print(" m/s");
  lcd.setCursor(0, 1); lcd.print("v2: "); lcd.print(v2, 2); lcd.print(" m/s");
  lcd.setCursor(0, 2); lcd.print("a : ");  lcd.print(a, 2);  lcd.print(" m/s2");
  lcd.setCursor(0, 3); lcd.print("vavg: "); lcd.print(v_avg, 2); lcd.print(" m/s");

// Nové 4-float pole
  float data[4] = {v1, v2, a, v_avg};
  nRF.stopListening();
  nRF.write(&data, sizeof(data));
  nRF.startListening();
}
```

Program pro měření hmotnosti

```
// --- nRF24L01 ---
#include <SPI.h>
#include "RF24.h"

// --- LCD (I2C) ---
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

// --- HX711 piny ---
const int pSCK = 6;
const int pDT  = 5;

// --- Tlačítko TARE ---
#define TLACITKO_NULOVANI 9

// --- HX711 módy ---
#define KANAL_A_ZESIL_128 1

// --- Kalibrace (podle obrázku) ---
float K = 0.00227487f;  // g per count  (~ 1 / 0.00227487 ≈ 439.45 count/g)

// --- nRF24 ---
#define CE 7
#define CS 8
RF24 radio(CE, CS);
byte adresa[] = "HMOT1";

long offset = 0;
```

```
// --- prototypy ---
long ctiHX(byte mod);
long avgRaw(byte n=4);
static inline float ema(float y,float x,float a){ return y + a*(x - y); }

void setup() {
  Serial.begin(9600);
  lcd.init(); lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Vaha - init");

  pinMode(pSCK, OUTPUT);
  pinMode(pDT , INPUT);
  digitalWrite(pSCK, LOW);
  (void)ctiHX(KANAL_A_ZESIL_128);  // nastavit kanal

  pinMode(TLACITKO_NULOVANI, INPUT_PULLUP);

  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(76);
  radio.openWritingPipe(adresa);
  radio.stopListening();

  delay(200);
  lcd.clear();

  // Rychlé auto-tare (≈0.6 s při 10SPS)
  offset = avgRaw(6);
  lcd.setCursor(0,0); lcd.print("Tare hotovo");
```

```
  delay(150);
  lcd.clear();
}

void loop() {
  // průměr 4 vzorky (~0.4 s při 10SPS; při 80SPS ~50 ms)
  long raw = avgRaw(4);

  // TARE – offset + reset filtru => okamžitá nula
  static float upr_f = 0;
  if (digitalRead(TLACITKO_NULOVANI) == LOW) {
    offset = raw;
    upr_f  = 0;
    lcd.setCursor(0,0); lcd.print("   Nulovano...   ");
    delay(120);
    lcd.clear();
  }

  long upr = raw - offset;

  // Dynamické EMA: hranice = skok cca 5 g (v countech podle K)
  float stepThreshold = 5.0f / K;  // ~ 5 g v countech
  float alpha = (fabs((float)upr - upr_f) > stepThreshold) ? 0.8f : 0.25f;
  upr_f = ema(upr_f, (float)upr, alpha);

  float h_g = upr_f * K;

  if (fabs(h_g) > 3000.0f) {
    lcd.setCursor(0,0); lcd.print("   PREKROCENO!   ");
    lcd.setCursor(0,1); lcd.print(" Limit: 3.00 kg ");
```

```
  } else {
    lcd.setCursor(0,0); lcd.print("Hmotnost:       ");
    lcd.setCursor(0,1); lcd.print(h_g, 2); lcd.print(" g        ");
    radio.write(&h_g, sizeof(h_g));
  }

  Serial.print("raw="); Serial.print(raw);
  Serial.print(" upr="); Serial.print((long)upr_f);
  Serial.print(" h[g]="); Serial.println(h_g, 2);
}

// --- čtení HX711 (24bit + kanál), bez dělení „zisku“ ---
long ctiHX(byte mod) {
  while (digitalRead(pDT) == HIGH) {}  // data-ready (DT LOW)
  long v = 0;
  for (byte i=0;i<24;i++){
    digitalWrite(pSCK,HIGH);
    v = (v<<1) | digitalRead(pDT);
    digitalWrite(pSCK,LOW);
  }
  for (byte i=0;i<mod;i++){ digitalWrite(pSCK,HIGH); digitalWrite(pSCK,LOW); } // 1x =
A@128
  if (v & 0x800000L) v |= 0xFF000000L;  // sign-extend
  return v;
}

// průměr z n vzorků A@128 (čekáme jen na DT, bez zbytečných delayů)
long avgRaw(byte n){
  long s=0; for(byte i=0;i<n;i++){ s += ctiHX(KANAL_A_ZESIL_128); }
  return s / n;
}
```

Program pro měření teploty

```
// Knihovny pro teplotní čidlo
#include <OneWire.h>
#include <DallasTemperature.h>

// Knihovny pro nRF24L01
#include <SPI.h>
#include "RF24.h"

// Knihovny pro I2C LCD displej
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Inicializace LCD (I2C adresa 0x27, 16 znaků, 2 řádky)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Pin pro DS18B20
#define DS18B20_PIN 4
OneWire oneWire(DS18B20_PIN);
DallasTemperature sensors(&oneWire);

// nRF24L01 nastavení
#define CE 7
#define CS 8
RF24 radio(CE, CS);
byte adresa[] = "TEMP1w";

void setup() {
  Serial.begin(9600);
```

```
  // Inicializace LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("   Spoustim...");

  // Inicializace teplotního čidla
  sensors.begin();

  // Inicializace nRF24L01
  radio.begin();
  radio.setPALevel(RF24_PA_LOW);
  radio.setChannel(76);  // stejný kanál jako přijímač
  radio.openWritingPipe(adresa);
  radio.stopListening(); // režim vysílače

  delay(1000);
  lcd.clear();
}

void loop() {
  // Načtení teploty
  sensors.requestTemperatures();
  float teplota = sensors.getTempCByIndex(0);

  // Výpis do sériového monitoru
  Serial.print("Naměřená teplota: ");
  Serial.print(teplota);
  Serial.println(" °C");
```

```
  // Výpis na LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Teplota:");
  lcd.setCursor(0, 1);
  lcd.print(teplota, 2);
  lcd.print((char)223); // Znak ° (stupně)
  lcd.print("C");

  // Odeslání přes nRF24L01
  bool ok = radio.write(&teplota, sizeof(teplota));
  if (ok) {
    Serial.println("Data odeslána.");
  } else {
    Serial.println("CHYBA: Nepodařilo se odeslat.");
  }

  delay(1000); // 1 sekunda pauza
}
```

Program pro formulářovou aplikaci

```
using System;
using System.IO.Ports;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace PC_s_Arduino2
{
   public partial class Form1 : Form
   {
      SerialPort serialPort;
      Thread readThread;
      volatile bool listening;
      volatile bool readyReceived;
      StringBuilder receivedBuffer = new StringBuilder();
      int currentMode = 0;

      public Form1()
      {
         InitializeComponent();
         RefreshPorts();
      }

      private void RefreshPorts()
      {
         comboBoxPorts.Items.Clear();
         comboBoxPorts.Items.AddRange(SerialPort.GetPortNames());
      }
```

```
private void comboBoxPorts_SelectedIndexChanged(object sender, EventArgs e)
{
   //  sem případně automatické připojení
}

private void buttonRefresh_Click(object sender, EventArgs e)
{
   RefreshPorts();
}

private void buttonConnect_Click(object sender, EventArgs e)
{
   if (comboBoxPorts.SelectedItem != null)
   {
      try
      {
         if (serialPort != null && serialPort.IsOpen)
         {
            listening = false;
            readThread?.Join();
            serialPort.Close();
            Thread.Sleep(200);
         }

         receivedBuffer.Clear();
         readyReceived = false;
         listening = true;

         serialPort = new SerialPort(comboBoxPorts.SelectedItem.ToString(), 9600);
         serialPort.ReadTimeout = 500;
```

```
            serialPort.WriteTimeout = 500;
            serialPort.Encoding = Encoding.ASCII;
            serialPort.DtrEnable = true;

            serialPort.Open();

            serialPort.DtrEnable = false;
            Thread.Sleep(100);
            serialPort.DtrEnable = true;

            serialPort.DiscardInBuffer();

            readThread = new Thread(ReadFromSerial);
            readThread.Start();

            Thread.Sleep(800);

            int waited = 0;
            while (!readyReceived && waited < 3000)
            {
                Application.DoEvents();
                Thread.Sleep(50);
                waited += 50;
            }

            if (readyReceived)
            {
                MessageBox.Show("✅ Připojeno k Arduinu.\n\nOdpověď:\n" +
receivedBuffer.ToString());
            }
            else
```

```
                {
                    listening = false;
                    readThread?.Join();
                    serialPort.Close();
                    MessageBox.Show("❌ Arduino neodpovědělo (timeout).\n\nZachyceno:\n" +
receivedBuffer.ToString());
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Chyba:\n" + ex.Message);
            }
        }
        else
        {
            MessageBox.Show("Vyber COM port.");
        }
    }

    private void ReadFromSerial()
    {
        try
        {
            while (listening)
            {
                try
                {
                    string chunk = serialPort.ReadExisting();
                    if (!string.IsNullOrEmpty(chunk))
                    {
                        receivedBuffer.Append(chunk);
```

```
                if (!readyReceived && receivedBuffer.ToString().Contains("READY"))
                {
                    readyReceived = true;
                }
                else
                {
                    Invoke(new Action(() =>
                    {
                        ParseAndDisplay(receivedBuffer.ToString());
                        receivedBuffer.Clear();
                    }));
                }
            }
        }
        catch { }

        Thread.Sleep(50);
    }
  }
  catch { }
}

private void ParseAndDisplay(string data)
{
  if (data.StartsWith("V:"))
  {
    labelMode.Text = "Rychlosti";
    var parts = data.Substring(2).Split(',');
    if (parts.Length == 4)
```

```
        {
            labelOutput1.Text = "v1 = " + parts[0] + " m/s";
            labelOutput2.Text = "v2 = " + parts[1] + " m/s";
            labelOutput3.Text = "a = " + parts[2] + " m/s²";
            labelOutput4.Text = "v_avg = " + parts[3] + " m/s";
        }
    }
    else if (data.StartsWith("T:"))
    {
        labelMode.Text = "Teplota";
        labelOutput1.Text = "T = " + data.Substring(2).Trim() + " °C";
        labelOutput2.Text = "";
        labelOutput3.Text = "";
        labelOutput4.Text = "";
    }
    else if (data.StartsWith("M:"))
    {
        labelMode.Text = "Hmotnost";
        labelOutput1.Text = "m = " + data.Substring(2).Trim() + " g";
        labelOutput2.Text = "";
        labelOutput3.Text = "";
        labelOutput4.Text = "";
    }
}

private void buttonSwitchMode_Click(object sender, EventArgs e)
{
    if (serialPort != null && serialPort.IsOpen)
    {
        try
```

```
        {
            serialPort.Write("M");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Chyba při přepínání režimu:\n" + ex.Message);
        }
    }
}

private void buttonReset_Click(object sender, EventArgs e)
{
    if (serialPort != null && serialPort.IsOpen)
    {
        try
        {
            serialPort.DtrEnable = false;
            Thread.Sleep(100);
            serialPort.DtrEnable = true;
            labelMode.Text = "Režim";
            labelOutput1.Text = "";
            labelOutput2.Text = "";
            labelOutput3.Text = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show("Chyba při resetu:\n" + ex.Message);
        }
    }
}
```

```
        private void labelMode_Click(object sender, EventArgs e) { }
        private void labelOutput1_Click(object sender, EventArgs e) { }
        private void labelOutput2_Click(object sender, EventArgs e) { }
        private void labelOutput3_Click(object sender, EventArgs e) { }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (serialPort != null && serialPort.IsOpen)
            {
                listening = false;
                readThread?.Join();
                serialPort.Close();
            }
        }

        private void labelOutput4_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Program pro kalibraci váhy

```
#include <Arduino.h>

// --- Piny HX711 ---
const int pSCK = 6;
const int pDT  = 5;

// --- Tlačítko (TARE/confirm) ---
#define BTN 9

// --- HX711 módy ---
#define A128 1
#define B32  2
#define A64  3

// >>> Nastav hmotnost kalibračního závaží v gramech <<<
const float KNOWN_WEIGHT_G = 1000.0;  // změň dle svého závaží

// ---- HX711 čtení ----
long readHX(byte mod) {
  while (digitalRead(pDT) == HIGH) {} // čekej na data ready
  long v = 0;
  for (byte i = 0; i < 24; i++) {
    digitalWrite(pSCK, HIGH);
    v = (v << 1) | digitalRead(pDT);
    digitalWrite(pSCK, LOW);
  }
  for (byte i = 0; i < mod; i++) { digitalWrite(pSCK, HIGH); digitalWrite(pSCK, LOW); }
  if (v & 0x800000L) v |= 0xFF000000L; // sign-extend
  return v;
```

```
}

long readAvgA128(int n=20) {
  long s = 0;
  for (int i=0;i<n;i++){ s += readHX(A128); delay(15); }
  return s / n;
}

void waitButtonRelease() { while(digitalRead(BTN)==LOW){} delay(50); }
void waitButtonPress(const char* msg){
  Serial.println(msg);
  while(digitalRead(BTN)==HIGH){}  // čekám na stisk
  waitButtonRelease();
}

void setup() {
  Serial.begin(9600);
  pinMode(pSCK, OUTPUT);
  pinMode(pDT,  INPUT);
  pinMode(BTN,  INPUT_PULLUP);
  digitalWrite(pSCK, LOW);

  // ustav kanál A@128
  (void)readHX(A128);
  delay(50);

  Serial.println("\n== KALIBRACE HX711 (kanal A@128) ==");
  Serial.println("1) Sundej vse z vahy.");
  waitButtonPress("-> Stiskni tlacitko pro zmereni NULY.");
  long raw0 = readAvgA128(20);
```

```
  Serial.print("RAW_0 = "); Serial.println(raw0);

  Serial.println("\n2) Poloz ZNAME ZAVAZI na vahu.");
  waitButtonPress("-> Stiskni tlacitko pro zmereni se zavazim.");
  long rawW = readAvgA128(20);
  Serial.print("RAW_W = "); Serial.println(rawW);

  long delta = rawW - raw0;
  Serial.print("\nDeltaRAW = "); Serial.println(delta);

  if (delta == 0) {
    Serial.println("Delta je 0 -> zkontroluj zapojeni a zavazi.");
  } else {
    float K = KNOWN_WEIGHT_G / (float)delta;   // g na count
    Serial.print("\n>>> Kalibracni faktor K = ");
    Serial.print(K, 8);
    Serial.println("  (g / count)");
    Serial.println("\nZkopiruj tuto hodnotu do sveho programu jako:");
    Serial.println("  float K = " + String(K, 8) + ";");
    Serial.println("\nPozn.: Kdyz hmotnost po pridani zavazi vychazi zaporna, dej K s minus znaminkem.");
  }
}

void loop() {
  // nic, vše proběhne v setup()
}
```