

UNIVERZITET U KRAGUJEVCU
PRIRODNO-MATEMATIČKI FAKULTET
INSTITUT ZA MATEMATIKU I INFORMATIKU

Tehnička dokumentacija

Realizacija linearne i polinomne regresije

Tim 3:

Marija Jolović 46-2021

Stefan Stanišić 81-2021

Anđelina Maksimović 56-2021

Mentori:

Prof. dr Aleksandar Peulić

Kristina Vasić, asistent

Avgust, 2025

Sadržaj

Uvod – opis projekta	3
Hardverske komponente.....	3
STM32 Nucleo-C031C6 razvojna ploča.....	3
Akcelerometar (ACC modul)	4
USB kabl (micro-USB – USB-A)	4
Breadboard i žice (jumper wires)	4
Povezane komponente	6
UART komunikacija (USART2).....	6
ADC (Analog-to-Digital Converter).....	6
Tera Term softver.....	6
Opis rada sistema	7
ADC	8
Realizacija linearne regresije.....	10
Realizacija polinomne regresije.....	12
UART2.....	13
Analiza rezultata	14
Linearna regresija	15
Polinomna regresija	15
Opšta ocena	15
Zaključak.....	16

Uvod – opis projekta

Projekat pod nazivom „Linearna i polinomna regresija nad 4 analogna ulaza“ realizovan je na mikrokontrolerskoj platformi STM32 Nucleo C031C6T6 korišćenjem STM32CubeIDE okruženja. Sistem koristi četiri analogna ulaza: dva fiksna signala (GND i 3.3V) i dva varijabilna signala sa akcelerometra (X i Z ose) povezanim na pinove PA1 i PA2. Podaci sa ulaza obrađuju se na mikrokontroleru pomoću algoritama za linearnu i polinomnu regresiju (drugog stepena), čime se određuju koeficijenti koji aproksimiraju odnos između prikupljenih vrednosti. Rezultati proračuna prikazuju se putem UART komunikacije na računaru.

Cilj projekta je demonstracija primene metoda regresione analize u realnom vremenu na ugrađenom sistemu, uz prikaz performansi linearnog i polinomnog modela i njihovo poređenje. Implementacija obuhvata konfiguraciju ADC modula za čitanje četiri kanala, proračun regresionih modela na mikrokontroleru i slanje rezultata preko serijskog porta. Projekat je zamišljen kao edukativni primer obrade podataka i realizacije algoritama optimizacije u ograničenom hardverskom okruženju.

Hardverske komponente

Za realizaciju projekta korišćene su sledeće hardverske komponente:

STM32 Nucleo-C031C6 razvojna ploča

Glavni mikrokontroler koji obavlja akviziciju podataka sa senzora (ACC), njihovu obradu, računanje linearne i polinomne regresije, i komunikaciju sa računarom putem UART-a.



Slika 1 - Nucleo C031C6

Akcelerometar (ACC modul)

Analogni senzor koji na svojim izlazima X i Y daje napon proporcionalan ubrzanju u određenim osama. Signali sa ACC modula su povezani na analogne ulaze mikrokontrolera.



Slika 2 - Akcelerometar

USB kabl (micro-USB – USB-A)

Koristi se za napajanje ploče i serijsku komunikaciju sa računarom.



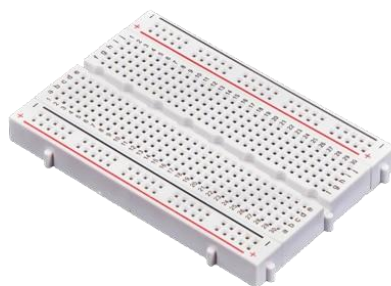
Slika 3 - micro USB kabl

Breadboard i žice (jumper wires)

Omogućavaju povezivanje ACC senzora sa ulazima Nucleo ploče (pinovi PA0 i PA1).

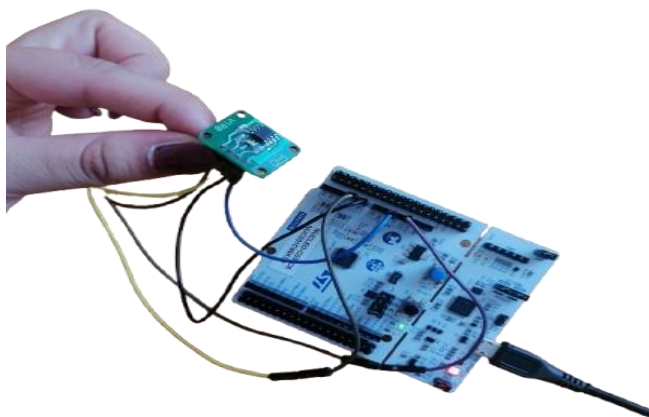


Slika 4 – Žice



Slika 5 – Proto-ploča

Sve hardverske komponente povezane



Slika 6 - Povezane hardverske komponente

Povezane komponente

Pored osnovnih hardverskih delova, za povezivanje i funkcionisanje sistema korišćene su sledeće komponente:

UART komunikacija (USART2)

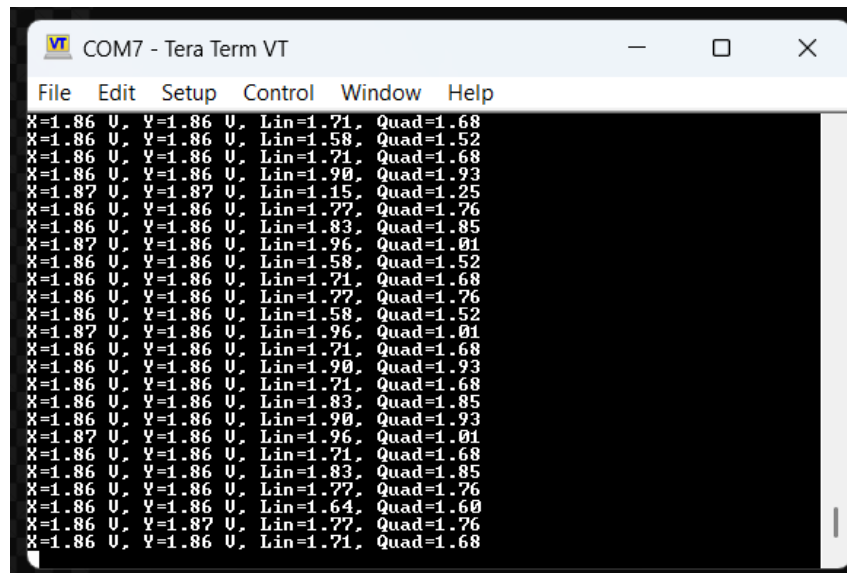
Povezuje mikrokontroler i računar preko ST-Link Virtual COM porta. Omogućava slanje podataka o očitanim vrednostima i rezultatima regresije ka terminal programu (Tera Term ili PuTTY).

ADC (Analog-to-Digital Converter)

Ugrađen u STM32 mikrokontroler. Pretvara analogne izlaze sa senzora (X i Y) u digitalne vrednosti koje se dalje koriste za izračunavanje regresionih modela.

Tera Term softver

Softverski terminal na računaru koji prima i prikazuje podatke poslati preko UART veze. Koristi se za monitoring i verifikaciju rezultata.



Slika 7 - Izlaz na Tera Term-u sa realnim vrednostima i predikcijama

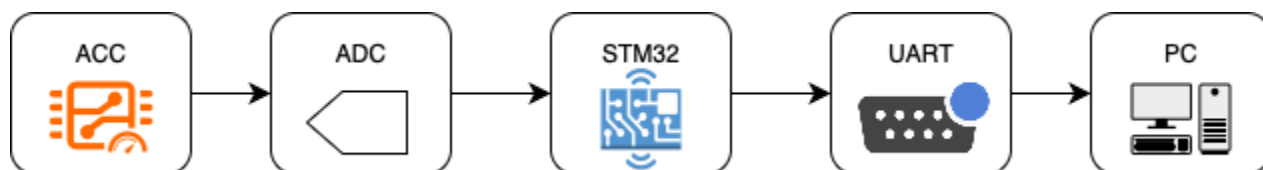
Opis rada sistema

Sistem funkcioniše tako što akcelerometar (ACC) generiše analogne naponske vrednosti na svojim izlazima X i Y. Te vrednosti predstavljaju ubrzanje u odgovarajućim osama. Analogni signali se zatim dovode na ulazne pinove mikrokontrolera STM32 Nucleo-C031C6, konkretno na PA0 (ADC1_IN0) i PA1 (ADC1_IN1).

Unutar mikrokontrolera, ADC (Analogno-digitalni konvertor) pretvara te napone u digitalne vrednosti. Nakon digitalizacije, podaci se koriste za računanje linearne i kvadratne (polinomne) regresije. Na taj način sistem pravi predikciju odnosa između X i Y signala.

Rezultati (realna očitana vrednost, kao i predikcije modela) šalju se preko USART2 periferije kroz ST-Link Virtual COM Port ka računaru. Na računaru je pokrenut terminalski program (Tera Term), koji omogućava da korisnik vidi očitane i izračunate vrednosti u realnom vremenu.

Na ovaj način, sistem kombinuje akviziciju podataka, digitalnu obradu signala i vizuelizaciju rezultata, što čini osnovu za eksperimente sa metodama mašinskog učenja u realnom vremenu.



Slika 8 - Blok dijagram toka podataka

```
TeraTerm

X=1.84 V, Y=1.85 V, Lin=1.70, Quad=1.77
X=1.84 V, Y=1.84 V, Lin=1.46, Quad=1.53
X=1.84 V, Y=1.84 V, Lin=1.38, Quad=1.45
X=1.84 V, Y=1.84 V, Lin=1.62, Quad=1.69
X=1.84 V, Y=1.84 V, Lin=1.54, Quad=1.61
X=1.84 V, Y=1.84 V, Lin=1.62, Quad=1.69
X=1.84 V, Y=1.84 V, Lin=1.46, Quad=1.53
X=1.84 V, Y=1.84 V, Lin=1.46, Quad=1.53
X=1.84 V, Y=1.84 V, Lin=1.70, Quad=1.77
X=1.84 V, Y=1.84 V, Lin=1.62, Quad=1.69
X=1.84 V, Y=1.84 V, Lin=1.06, Quad=1.13
X=1.84 V, Y=1.84 V, Lin=1.54, Quad=1.61
X=1.84 V, Y=1.84 V, Lin=1.62, Quad=1.69
X=1.83 V, Y=1.83 V, Lin=1.90, Quad=1.96
```

Slika 9 - Primer Tera Term izlaza sa realnim vrednostima i predikcijama

ADC

ADC (Analog-to-Digital Converter) je ugrađen modul u STM32 mikrokontroleru koji pretvara kontinualni analogni signal (npr. napon sa senzora) u digitalnu vrednost koja se može obraditi programom.

U ovom projektu ADC je podešen na rezoluciju od **12 bita**, što znači da se ulazni napon (0–3.3V) mapira na vrednosti od **0 do 4095**. Na primer:

- 0 V → 0
- 1.65 V → oko 2048
- 3.3 V → 4095

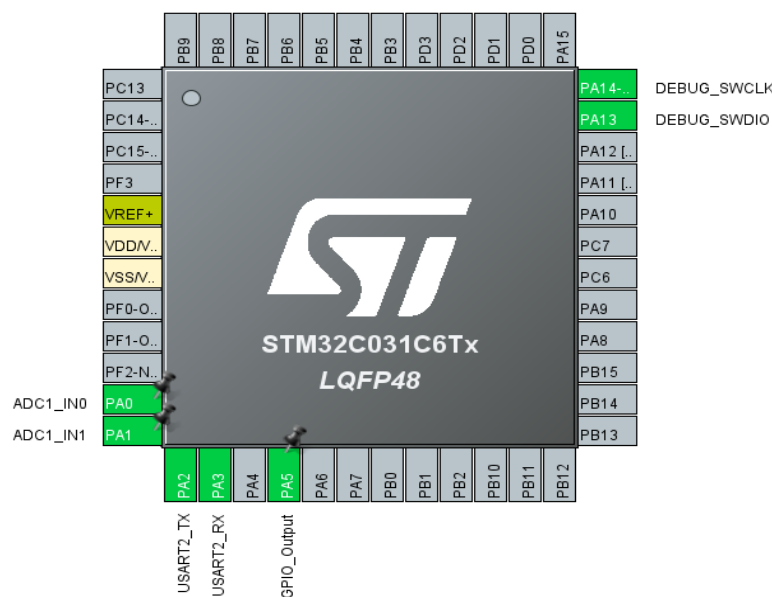
Postupak očitavanja je sledeći:

1. Mikrokontroler preko HAL biblioteke startuje ADC konverziju.
2. ADC uzorkuje trenutnu vrednost napona na pinovima (PA0, PA1).
3. Dobijena digitalna vrednost se čuva i dalje koristi za računanje regresije.

Za lepši prikaz korisniku, digitalne vrednosti se mogu konvertovati nazad u napon pomoću formule:

$$U = 4095 \text{ADC_value} \times 3.3[V]$$

Ovaj princip omogućava da sa običnim analognim senzorima radimo precizna digitalna merenja i na njihovim vrednostima implementiramo algoritme kao što su linearna i polinomna regresija.



Slika 10 - Povezanih pinovi u STM32CubeMX

Definisanje handlera i promenljivih:

```
ADC_HandleTypeDef hadc1;  
uint32_t adcX, adcY;
```

Kod 1 - definisanje handlera i promenljivih

- hadc1 – handle za ADC1 periferiju.
- adcX i adcY – promenljive u koje se čuvaju očitane vrednosti sa ADC kanala.

Inicijalizacija ADC1

```
Inicijalizacija ADC1  
  
static void MX_ADC1_Init(void)  
{  
    ADC_ChannelConfTypeDef sConfig = {0};  
  
    hadc1.Instance = ADC1;  
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV1;  
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;  
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;  
    hadc1.Init.ScanConvMode = ADC_SCAN_SEQ_FIXED;  
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;  
    hadc1.Init.ContinuousConvMode = DISABLE;  
    hadc1.Init.NbrOfConversion = 1;  
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;  
    if (HAL_ADC_Init(&hadc1) != HAL_OK) { Error_Handler(); }  
  
    sConfig.Channel = ADC_CHANNEL_0;  
    sConfig.Rank = ADC_RANK_CHANNEL_NUMBER;  
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);  
  
    sConfig.Channel = ADC_CHANNEL_1;  
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);  
}
```

Kod 2 - inicijalizacija ADC1

- ADC konfiguracija: 12-bitna rezolucija, desno poravnanje.
- Kanali 0 i 1 su podešeni za čitanje (npr. PA0 i PA1).
- Konverzija se startuje softverski (ADC_SOFTWARE_START).

Čitanje sa ADC kanala

```
Čitanje sa ADC kanala

// --- Čitanje X ---
HAL_ADC_ConfigChannel(&hadc1, &(ADC_ChannelConfTypeDef){

    .Channel = ADC_CHANNEL_0,
    .Rank = 1,
    .SamplingTime = ADC_SAMPLETIME_12CYCLES_5
});
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
adcX = HAL_ADC_GetValue(&hadc1);

// --- Čitanje Y ---
HAL_ADC_ConfigChannel(&hadc1, &(ADC_ChannelConfTypeDef){

    .Channel = ADC_CHANNEL_1,
    .Rank = 1,
    .SamplingTime = ADC_SAMPLETIME_12CYCLES_5
});
HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
adcY = HAL_ADC_GetValue(&hadc1);
```

Kod 3 - Čitanje sa ADC kanala

- HAL_ADC_ConfigChannel() – konfiguracija kanala pre svakog čitanja.
- HAL_ADC_Start() – start konverzije.
- HAL_ADC_PollForConversion() – čekanje da konverzija završi.
- HAL_ADC_GetValue() – očitavanje digitalne vrednosti ADC.

Realizacija linearne regresije

Za realizaciju linearne regresije korišćena je metoda najmanjih kvadrata, kojom se pronalaze koeficijenti pravca a i b za jednačinu:

$$Y = aX + b$$

Podaci za regresiju prikupljeni su pomoću ADC modula sa dva analogna ulaza (ADC_CHANNEL_0 i ADC_CHANNEL_1), koji predstavljaju X i Y vrednosti. Nakon što je prikupljeno N = 20 uzoraka, izračunate su sume potrebne za formulu:

$$a = (N * \sum XY - \sum X * \sum Y) / (N * \sum X^2 - (\sum X)^2)$$
$$b = (\sum Y - a * \sum X) / N$$

Koeficijenti a i b se izračunavaju u programu, a rezultat se šalje preko USART2 interfejsa na računar. Na taj način je omogućeno praćenje izračunatih parametara u realnom vremenu.

```

// Broj uzoraka
#define N 20
uint32_t sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

// Sakupljanje podataka
for (int i = 0; i < N; i++) {
    // Čitanje X
    HAL_ADC_ConfigChannel(&hadc1, &(ADC_ChannelConfTypeDef){
        .Channel = ADC_CHANNEL_0, .Rank = 1, .SamplingTime =
ADC_SAMPLETIME_12CYCLES_5});
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adcX = HAL_ADC_GetValue(&hadc1);

    // Čitanje Y
    HAL_ADC_ConfigChannel(&hadc1, &(ADC_ChannelConfTypeDef){
        .Channel = ADC_CHANNEL_1, .Rank = 1, .SamplingTime =
ADC_SAMPLETIME_12CYCLES_5});
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    adcY = HAL_ADC_GetValue(&hadc1);

    // Akumulacije
    sumX += adcX;
    sumY += adcY;
    sumXY += (adcX * adcY);
    sumX2 += (adcX * adcX);

    HAL_Delay(100);
}

// Izračunavanje koeficijenata
float a_lin = ((N * sumXY) - (sumX * sumY)) / (float)((N * sumX2) - (sumX * sumX));
float b_lin = ((sumY - (a_lin * sumX)) / (float)N);

// Slanje rezultata
sprintf(msg, "Linear regression: Y = %.3f * X + %.3f\r\n", a_lin, b_lin);
HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);

```

Kod 4 - Računanje suma potrebnih za linearnu regresiju

Realizacija polinomne regresije

Kvadratna regresija je specijalni slučaj polinomne regresije koji koristi polinom drugog stepena. Za izračunavanje koeficijenata kvadratne funkcije korišćeno je Cramerovo pravilo i determinante.

Koeficijenti su zatim formatirani i poslani preko UART komunikacije.

```
// --- Kvadratna regresija (Cramer) ---
float A[3][3] = {
    {N,      sumX,      sumX2},
    {sumX, sumX2,      sumX3},
    {sumX2, sumX3,      sumX4}
};
float B[3] = {sumY, sumXY, sumX2Y};

float detA =
    A[0][0]*A[1][1]*A[2][2]-A[1][2]*A[2][1]
    - A[0][1]*A[1][0]*A[2][2]-A[1][2]*A[2][0]
    + A[0][2]*A[1][0]*A[2][1]-A[1][1]*A[2][0];

float detC =
    B[0]*A[1][1]*A[2][2]-A[1][2]*A[2][1]
    - A[0][1]*B[1]*A[2][2]-A[1][2]*B[2]
    + A[0][2]*B[1]*A[2][1]-A[1][1]*B[2];

float detB =
    A[0][0]*B[1]*A[2][2]-A[1][2]*B[2]
    - B[0]*A[1][0]*A[2][2]-A[1][2]*A[2][0]
    + A[0][2]*A[1][0]*B[2]-B[1]*A[2][0];

float detA2 =
    A[0][0]*A[1][1]*B[2]-B[1]*A[2][1]
    - A[0][1]*A[1][0]*B[2]-B[1]*A[2][0]
    + B[0]*A[1][0]*A[2][1]-A[1][1]*A[2][0];

float c_quad = detC / detA;
float b_quad = detB / detA;
float a_quad = detA2 / detA;

// Slanje rezultata, sada za polinomnu
int aQ=(int)(a_quad*1000);
int bQ=(int)(b_quad*1000);
int cQ=(int)(c_quad*1000);

sprintf(msg, "Quadratic: Y = %.03d*X^2 + %.03d*X + %.03d\r\n",
    aQ/1000,
    bQ/1000,
    abs(bQ%1000),
    cQ/1000,
    abs(cQ%1000));
HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
```

Kod 5 - Računanje suma potrebnih za (polinomnu) kvadratnu regresiju

UART2

UART2 komunikacija na STM32 Nucleo ploči omogućava jednostavan serijski prenos podataka između mikrokontrolera i računara. Na Nucleo C031 ploči UART2 je obično povezan preko ST-Link USB interfejsa, što znači da nije potrebno dodatno spajanje kablova za serijsku komunikaciju – USB kabl koji povezuje ploču sa računarom istovremeno omogućava napajanje i prenos podataka. Kada se USB poveže, mikrokontroler emulira COM port (virtuelni serijski port) na računaru, što omogućava slanje i primanje podataka u realnom vremenu preko terminal programa (kao što su PuTTY, Tera Term ili STM32CubeMonitor). Ovaj način povezivanja je praktičan za debugovanje, praćenje senzorskih podataka i prikaz rezultata računskih operacija, kao što su linearna i kvadratna regresija u ovom projektu.

Prednosti korišćenja UART-a u ovom projektu

- Omogućava jednostavno praćenje rada sistema i predikcija.
- Ne zahteva dodatni ekran ili display.
- Može se koristiti za debug i testiranje modela regresije u realnom vremenu.
- Podaci se mogu čuvati ili procesirati na računaru.

Inicijalizacija UART2



```
static void MX_USART2_UART_Init(void)
{
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    huart2.Init.ClockPrescaler = UART_PRESCALER_DIV1;
    huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
    if (HAL_UART_Init(&huart2) != HAL_OK) { Error_Handler(); }
}
```

Kod 6 - Inicijalizacija UART2

- BaudRate = 115200: brzina komunikacije u bitovima po sekundi.
- WordLength = 8B: dužina jednog podatkovnog bajta.
- StopBits = 1: signal za kraj bajta.
- Parity = NONE: bez pariteta.
- Mode = TX_RX: omogućeno slanje i prijem.
- HAL_UART_Init: funkcija HAL biblioteke koja inicijalizuje UART perifernu funkciju.

Primer funkcije za slanje ispisa poruke:

```
Slanje podataka preko UART-a

void uartPrint(char *msg) {
    HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
}
```

Kod 7 – Funkcija za transmitovanje poruka

- HAL_UART_Transmit: HAL funkcija za slanje bajtova.
- huart2: pokazivač na UART periferiju koju koristimo.
- strlen(msg): broj bajtova za slanje.
- HAL_MAX_DELAY: funkcija čeka dok se svi bajtovi ne pošalju.

Primer slanja podataka u kodu:

```
Slanje podataka preko UART-a, primer u kodu

sprintf(msg, "X=%d, Y_real=%d, Y_lin_pred=%d.%02d, Y_quad_pred=%d.%02d\r\n",
        adcX, adcY, yL_int/100, abs(yL_int%100), yQ_int/100, abs(yQ_int%100));
uartPrint(msg);
```

Kod 8 - Ispisivanje podataka preko UART-a

Analiza rezultata

Tokom faze implementacije i testiranja realizovani su i **linearni** i **polinomni regresioni modeli** na razvojnoj ploči **STM32 Nucleo-C031C6**, koristeći analogne ulaze sa akcelerometra. Merenja su prikupljana putem ADC konvertora na dva kanala:


- **X (nezavisna promenljiva)** – očitavanje na ADC kanalu PA0, predstavlja napon sa X-ose akcelerometra.
- **Y (zavisna promenljiva)** – očitavanje na ADC kanalu PA1, predstavlja napon sa Y-ose akcelerometra.

Dobijene sirove vrednosti (0–4095 za 12-bitni ADC) korišćene su direktno za računanje regresionih koeficijenata. Kao dodatak, implementirano je i pretvaranje u napone radi bolje interpretacije od strane korisnika.

$$V = \frac{ADC}{4095.0} * 3.3V$$

Linearna regresija

Linearni model u obliku $Y = aX + b$ uspešno opisuje **opšti trend** zavisnosti između X i Y, ali su predikcije imale uočljivu sistematsku grešku. Odstupanja su se najčešće kretala između **20 i 40 jedinica ADC-a**, što predstavlja nekoliko desetina milivolta.



```
Primer izlaza
X=2081, Y_real=2079, Y_lin_pred=2108.06 → greška = +29
```

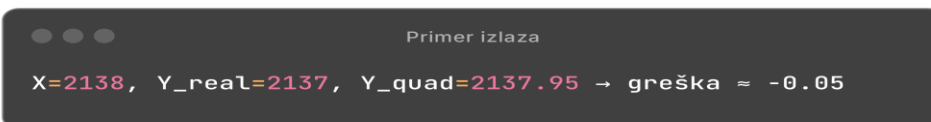
Slika 11 - Primer izlaza linearne regresije i greške

Ovo pokazuje da linearna regresija može da posluži za grubu aproksimaciju, ali ne daje dovoljnu preciznost za detaljnije predviđanje.

Polinomna regresija

Kvadratni model $Y = aX^2 + bX + c$ izračunat je pomoću viših suma ($\sum X^2$, $\sum X^3$, $\sum X^4$, $\sum X^2 Y$) i rešen Cramerovim pravilom. Rezultati su pokazali gotovo savršeno uklapanje sa realnim podacima.

Greške su u proseku bile manje od ± 1 ADC jedinice, što ukazuje na izuzetnu preciznost kvadratnog modela.



```
Primer izlaza
X=2138, Y_real=2137, Y_quad=2137.95 → greška = -0.05
```

Slika 12 - Primer izlaza polinomne regresije

Ovo potvrđuje da polinomna regresija značajno nadmašuje linearnu u tačnosti predviđanja.

Opšta ocena

- **X** → ulazni signal sa jedne ose akcelerometra (ADC0).
- **Y** → ulazni signal sa druge ose akcelerometra (ADC1).
- **Y_real** → stvarna vrednost dobijena direktno sa ADC-a.
- **Y_lin_pred** → predikcija dobijena linearnim modelom.
- **Y_quad_pred** → predikcija dobijena kvadratnim modelom.

Analiza pokazuje da linearna regresija može da posluži za demonstraciju i jednostavne proračune, dok polinomna regresija daje daleko bolju tačnost i praktično se poklapa sa realnim vrednostima.

Zaključak

Rezultati potvrđuju da je moguće realizovati regresione modele direktno na ugrađenom mikrokontroleru. Linearna regresija je računski jednostavnija i pokazuje osnovni princip, dok polinomna regresija daje znatno preciznije rezultate. Ovo jasno prikazuje kompromis između složenosti modela i tačnosti predviđanja, što je važan aspekt primene mašinskog učenja na uređajima sa ograničenim resursima.

Za budući rad predviđa se testiranje viših stepena polinoma (treći, četvrti red), računanje metrika greške (MSE, RMSE), kao i predstavljanje rezultata u fizičkim jedinicama (napon ili ubrzanje u g) radi bolje interpretacije.