Državna razina natjecanja mladih tehničara

Automatika

Pripreme

Wraparound

Kalibracija

Pohrana referentnih vrijednosti

Školska godina 2025. / 2026.

Radni materijal za pripremu učenika osnovnih škola

Hrvoje Vrhovski, Hrvatska zajednica tehničke kulture

Sadržaj

Wraparound

Problem prijelaza preko nule (wraparound)

Magnetni senzor AS5600 daje vrijednosti u rasponu od **0 do 4095**, što predstavlja puni krug od **360°**.

Važno je razumjeti da se radi o **kružnom**, a ne linearnom mjerenju, dakle vrijednosti se nakon prolaska punog kruga ponavljaju.
To znači da nakon najveće vrijednosti ponovno slijedi najmanja vrijednost:

👉 4095 → 0

Ova pojava naziva se **wraparound (prijelaz preko nule)**.

Upravo taj prijelaz može uzrokovati pogreške u izračunu ako se ne obradi pravilno.

Ako u programu jednostavno oduzimamo dvije vrijednosti, rezultat može biti potpuno pogrešan.

Primjer problema

Pretpostavimo da je početna vrijednost:

```
pocetna = 4000;
```

a trenutno očitana vrijednost:

```
trenutna = 100;
```

Ako računamo ovako:

```
int razlika = trenutna - pocetna;
```

dobivamo:

```
100 - 4000 = -3900
```

Takav rezultat nema smisla jer je pomak zapravo bio vrlo mali, samo malo preko nule.

Do problema dolazi zato što senzor ne mjeri pomak po ravnoj liniji, nego po kružnici.

Sličnu situaciju nalazimo i kod običnog sata: 10 → 11 → 12 → 1

Vrijeme prolazi, minute rastu, sati rastu; nakon 10 sati dolazi 11 pa 12 pa 1 i tako redom.
Na primjer, bilo je 10 sati, a sad je 11, dakle, staro vrijeme je 10, novo vrijeme je 11.

Razlika „novo vrijeme“ – „staro vrijeme“ iznosi 1 sat jer je 11 – 10 = 1.

Nakon 11 sati dolazi 12 i opet staro vrijeme je 11, novo vrijeme je 12 i razlika je opet 1 sat jer je „novo vrijeme“ – „staro vrijeme“ 1 sat, 12 – 11 = 1.

To funkcionira dok god je novo vrijeme već od starog vremena.

No kad bude 1 sat, ne možemo ovakvim računanjem dobiti stvaran protok vremena jer je staro vrijeme 12, a novo vrijeme je 1 pa 1 – 12 = -11.

Ne možemo računati:

1 - 12 = -11

Dakle računica nam je kriva, no ako u ovom slučaju na novo vrijeme dodamo puni krug (12 sati), računica će biti ispravna:

(„novo vrijeme“ + 12) – „staro vrijeme“ = (1 + 12) – 12 = 13 – 12 = 1

Jednostavno rješenje

Ako je trenutna vrijednost manja od početne, to može značiti da je došlo do prijelaza preko nule.
U tom slučaju potrebno je dodati puni raspon senzora:

```
if (trenutna < pocetna) {
  trenutna += 4096;
}

int razlika = trenutna - pocetna;
```

Primjer

```
pocetna = 4000;
trenutna = 100;

if (trenutna < pocetna) {
  trenutna += 4096;   // 100 + 4096 = 4196
}

int razlika = 4196 - 4000;   // 196
```

Sada je rezultat ispravan.

Općenitije rješenje

Još je bolje koristiti kružno oduzimanje:

```
uint16_t kruznaRazlika(uint16_t pocetak, uint16_t kraj) {
  return (kraj + 4096 - pocetak) & 0x0FFF;
}
```

Na taj način dobivamo ispravnu razliku unutar raspona senzora bez obzira na prijelaz preko nule.

Primjer uporabe:

```
uint16_t razlika = kruznaRazlika(pocetna, trenutna);
```

Ovakav pristup je pouzdaniji i uredniji, pa se često koristi u konačnom programu.

Važna napomena

Bez wraparound obrade:

vrijednosti mogu “skakati”
prikaz na displeju može biti pogrešan

Wraparound je obavezan dio rješenja

Kalibracija

Iako smo pravilnom obradom wraparounda riješili problem prijelaza preko nule, time još uvijek nismo osigurali potpuno točno mjerenje kuta.

U stvarnom sustavu očitanja senzora često odstupaju od stvarnog položaja. Najčešći razlozi su:

- magnet nije savršeno centriran iznad senzora
- magnet nije potpuno paralelan sa senzorom
- udaljenost magneta od senzora nije idealna
- mehanička konstrukcija ima određene tolerancije

Zbog toga očitana vrijednost senzora često ne odgovara potpuno stvarnom kutu.

To znači da uređaj može raditi, ali bez dodatne prilagodbe neće biti dovoljno precizan.

Zbog toga je potrebno provesti **kalibraciju**.

Kalibracija je postupak kojim se mjerni sustav prilagođava stvarnim uvjetima rada kako bi dobiveni rezultati što točnije odgovarali stvarnim vrijednostima.
U našem slučaju to znači da određujemo poznate položaje (npr. 0° i 180°) i na temelju njih prilagođavamo izračun kuta.

Na taj način ne mijenjamo rad senzora, nego prilagođavamo program stvarnom sustavu.

Kalibracija senzora

Kalibracijom prilagođavamo stvarni, nesavršeni sustav tako da daje točnije rezultate.

Ideja je jednostavna:

- u jednom poznatom položaju spremimo vrijednost koja predstavlja **0°**
- u drugom poznatom položaju spremimo vrijednost koja predstavlja **180°**

Te dvije vrijednosti postaju referentne točke sustava.

Na temelju njih možemo izračunati stvarni kut.

Referentne vrijednosti

Tijekom kalibracije spremamo:

```
uint16_t minVal;   // položaj 0°
uint16_t maxVal;   // položaj 180°
```

Ako se raspon nalazi preko prijelaza 4095 → 0, i dalje možemo pravilno računati pomoću kružne razlike.

Izračun raspona

```
uint16_t raspon = kruznaRazlika(minVal, maxVal);
```

Primjer

Ako je:

```
minVal = 4000;
maxVal = 300;
```

onda vrijedi:

```
raspon = (300 + 4096 - 4000) & 0x0FFF = 396
```

Dakle, stvarni raspon između 0° i 180° iznosi 396 koraka senzora.

Izračun trenutnog pomaka

Za svako novo očitanje računamo pomak od referentne nule:

```
uint16_t diff = kruznaRazlika(minVal, raw);
```

Pretvorba u kut

Ako znamo da diff predstavlja trenutačni pomak, a raspon puni kalibrirani raspon od 0° do 180°, tada kut računamo ovako:

```
float kut = (float)diff * 180.0 / raspon;
```

Po potrebi ga možemo ograničiti:

```
if (kut < 0) kut = 0;
if (kut > 180) kut = 180;
```

Primjer koda za kalibraciju

```
uint16_t kruznaRazlika(uint16_t pocetak, uint16_t kraj) {
  return (kraj + 4096 - pocetak) & 0x0FFF;
}

uint16_t minVal = 4000;
uint16_t maxVal = 300;
uint16_t raw    = 100;
```

```
uint16_t raspon = kruznaRazlika(minVal, maxVal);
uint16_t diff   = kruznaRazlika(minVal, raw);

float kut = (float)diff * 180.0 / raspon;

if (kut < 0) kut = 0;
if (kut > 180) kut = 180;
```

Kalibracijom dakle ne popravljamo senzor, nego prilagođavamo program stvarnim uvjetima rada.

Česta pogreška

Korištenje običnog oduzimanja
Ignoriranje wraparounda

Zaključak

Kalibracijom:

- ✔ prilagođavamo sustav
- ✔ povećavamo točnost

Pohrana referentnih vrijednosti u EEPROM

Nakon što smo proveli kalibraciju i odredili referentne vrijednosti (0° i 180°), postavlja se važno pitanje:

👉 što se događa kada isključimo uređaj?

Ako te vrijednosti spremimo samo u obične varijable, one će se izgubiti nakon isključivanja napajanja.
To znači da bi bilo potrebno svaki put ponovno provoditi kalibraciju, što je nepraktično. Kalibracija se ne bi trebala raditi svaki put iznova.

Zbog toga koristimo **EEPROM**.

EEPROM je memorija koja zadržava podatke i nakon isključivanja napajanja.

To znači da uređaj nakon ponovnog uključenja može učitati prethodno spremljene referentne vrijednosti i odmah nastaviti s radom.

Bez EEPROM-a:

- nakon gašenja uređaja gubi se kalibracija
- potrebno je ponovno postavljati 0° i 180°

S EEPROM-om:

- referentne vrijednosti ostaju spremljene
- uređaj je spreman za rad odmah nakon uključivanja

Primjer spremanja podataka

```
#include <EEPROM.h>

uint16_t minVal = 1234;
uint16_t maxVal = 2876;

EEPROM.put(0, minVal);
EEPROM.put(2, maxVal);
```

Primjer učitavanja podataka

```
uint16_t minVal;
uint16_t maxVal;

EEPROM.get(0, minVal);
EEPROM.get(2, maxVal);
```

Važna napomena

EEPROM ima ograničen broj zapisa.
Zbog toga podatke ne treba spremati stalno, nego samo kada je to potrebno, primjerice:

- nakon kalibracije
- nakon postavljanja radne nule

Cjelovit primjer programa

Slijedi cjelovit primjer programa koji:

- očitava senzor AS5600
- obrađuje wraparound
- omogućuje kalibraciju 0° i 180°
- omogućuje postavljanje radne nule
- sprema referentne vrijednosti u EEPROM
- prikazuje podatke u serijskom monitoru

Ovaj primjer je prilagođen za **Dasduino CORE / Arduino Nano (ATmega328P)**.

```
#include <Wire.h>
#include <AS5600.h>
#include <EEPROM.h>

AS5600 senzor;

// tipkalo
const uint8_t PIN_TIPKALO = A0;

// vrijeme
const unsigned long DEBOUNCE_MS = 25;
const unsigned long DUGI_PRITISAK_MS = 1000;
const unsigned long STARTNI_PROZOR_MS = 2000;
const unsigned long PERIOD_ISPISA_MS = 100;

// EEPROM oznaka
const uint32_t EEPROM_OZNAKA = 0x4B555431;   // "KUT1"

// struktura podataka za EEPROM
struct Postavke {
  uint32_t oznaka;
  uint16_t minVal;            // 0°
  uint16_t maxVal;            // 180°
  uint16_t radnaNulaStotinke;   // korisnička nula
};

Postavke postavke;

// stanja rada
enum StanjeRada {
  STARTNI_PROZOR,
```

```
  KALIBRACIJA_NULA,
  KALIBRACIJA_180,
  NORMALAN_RAD
};

StanjeRada stanje = STARTNI_PROZOR;

// tipkalo
bool prethodnoStanjeTipkala = HIGH;
unsigned long vrijemePromjeneTipkala = 0;
unsigned long vrijemePritiskaTipkala = 0;
bool dugiPritisakObraden = false;

// vrijeme rada
unsigned long vrijemePokretanja = 0;
unsigned long zadnjiIspis = 0;

// ------------------------------------------------
// POMOĆNE FUNKCIJE
// ------------------------------------------------

uint16_t ocitajSirovuVrijednost() {
  return (senzor.rawAngle() & 0x0FFF);
}

uint16_t kruznaRazlika(uint16_t pocetak, uint16_t kraj) {
  return (kraj + 4096 - pocetak) & 0x0FFF;
}

void spremiPostavke() {
  EEPROM.put(0, postavke);
}

void ucitajPostavke() {
  EEPROM.get(0, postavke);

  if (postavke.oznaka != EEPROM_OZNAKA) {
    postavke.oznaka = EEPROM_OZNAKA;
    postavke.minVal = 0;
    postavke.maxVal = 2048;
    postavke.radnaNulaStotinke = 0;
    spremiPostavke();
  }
}

uint16_t izracunajKalibriraniKutStotinke(uint16_t raw) {
  uint16_t raspon = kruznaRazlika(postavke.minVal, postavke.maxVal);

  // zaštita od neispravne kalibracije
  if (raspon < 10) raspon = 2048;

  uint16_t diff = kruznaRazlika(postavke.minVal, raw);

  uint32_t kut = (uint32_t)diff * 18000UL / raspon;
```

```
  if (kut > 18000UL) kut = 18000UL;

  return (uint16_t)kut;
}

uint16_t izracunajKutZaPrikazStotinke(uint16_t raw) {
  uint16_t kalibriraniKut = izracunajKalibriraniKutStotinke(raw);

  if (kalibriraniKut >= postavke.radnaNulaStotinke) {
    return kalibriraniKut - postavke.radnaNulaStotinke;
  } else {
    return 0;   // za ovaj kutomjer prikazujemo 0° do 180°
  }
}

// ---------------------------------------------------
// OBRADA TIPKALA
// 0 = ništa
// 1 = kratki pritisak
// 2 = dugi pritisak
// ---------------------------------------------------

uint8_t ocitajDogadajTipkala() {
  bool stanjeTipkala = digitalRead(PIN_TIPKALO);
  unsigned long sada = millis();

  if (stanjeTipkala != prethodnoStanjeTipkala) {
    vrijemePromjeneTipkala = sada;
    prethodnoStanjeTipkala = stanjeTipkala;
  }

  if ((sada - vrijemePromjeneTipkala) <= DEBOUNCE_MS) {
    return 0;
  }

  static bool biloPritisnuto = false;
  bool pritisnuto = (stanjeTipkala == LOW);

  if (pritisnuto && !biloPritisnuto) {
    vrijemePritiskaTipkala = sada;
    dugiPritisakObraden = false;
  }

  if (pritisnuto && !dugiPritisakObraden && (sada - vrijemePritiskaTipkala >=
DUGI_PRITISAK_MS)) {
    dugiPritisakObraden = true;
    biloPritisnuto = pritisnuto;
    return 2;
  }

  if (!pritisnuto && biloPritisnuto) {
    unsigned long trajanje = sada - vrijemePritiskaTipkala;
```

```
   if (trajanje >= DEBOUNCE_MS && trajanje < DUGI_PRITISAK_MS) {
    biloPritisnuto = pritisnuto;
    return 1;
   }
  }

  biloPritisnuto = pritisnuto;
  return 0;
}

// ------------------------------------------------
// LOGIKA PROGRAMA
// ------------------------------------------------

void obradiStanja() {
  uint8_t dogadaj = ocitajDogadajTipkala();
  unsigned long sada = millis();

  switch (stanje) {

    case STARTNI_PROZOR:
     if (dogadaj == 1 || dogadaj == 2) {
      stanje = KALIBRACIJA_NULA;
      Serial.println("Ulazak u kalibraciju.");
      Serial.println("Postavi krak na 0° i pritisni tipkalo.");
     }
     else if ((sada - vrijemePokretanja) >= STARTNI_PROZOR_MS) {
      stanje = NORMALAN_RAD;
      Serial.println("Normalan rad.");
     }
     break;

    case KALIBRACIJA_NULA:
     if (dogadaj == 1) {
      postavke.minVal = ocitajSirovuVrijednost();
      Serial.print("Spremljeno 0°: ");
      Serial.println(postavke.minVal);

      stanje = KALIBRACIJA_180;
      Serial.println("Postavi krak na 180° i pritisni tipkalo.");
     }
     break;

    case KALIBRACIJA_180:
     if (dogadaj == 1) {
      postavke.maxVal = ocitajSirovuVrijednost();
      postavke.radnaNulaStotinke = 0;
      spremiPostavke();

      Serial.print("Spremljeno 180°: ");
      Serial.println(postavke.maxVal);
      Serial.println("Kalibracija zavrsena.");

      stanje = NORMALAN_RAD;
```

```
    }
    break;

  case NORMALAN_RAD:
    if (dogadaj == 1) {
      uint16_t raw = ocitajSirovuVrijednost();
      postavke.radnaNulaStotinke = izracunajKalibriraniKutStotinke(raw);
      spremiPostavke();

      Serial.println("Postavljena nova radna nula.");
    }
    break;
 }
}

// ------------------------------------------------
// SETUP
// ------------------------------------------------

void setup() {
 pinMode(PIN_TIPKALO, INPUT_PULLUP);

 Wire.begin();
 senzor.begin();

 Serial.begin(9600);
 delay(300);

 ucitajPostavke();

 vrijemePokretanja = millis();

 Serial.println("Kutomjer pokrenut.");
 Serial.println("U prve 2 sekunde moguc je ulazak u kalibraciju.");
 Serial.println("Kratki pritisak u normalnom radu postavlja radnu nulu.");
}

// ------------------------------------------------
// LOOP
// ------------------------------------------------

void loop() {
 obradiStanja();

 unsigned long sada = millis();
 if ((sada - zadnjiIspis) >= PERIOD_ISPISA_MS) {
  zadnjiIspis = sada;

  uint16_t raw = ocitajSirovuVrijednost();

  if (stanje == KALIBRACIJA_NULA) {
    Serial.println("Kalibracija: cekam spremanje 0°");
  }
  else if (stanje == KALIBRACIJA_180) {
```

```
    Serial.println("Kalibracija: cekam spremanje 180°");
   }
   else {
    uint16_t kalibriraniKut = izracunajKalibriraniKutStotinke(raw);
    uint16_t kutZaPrikaz = izracunajKutZaPrikazStotinke(raw);

    Serial.print("Ocitana sirova vrijednost kuta = ");
    Serial.println(raw);

    Serial.print("Preracunata vrijednost u stupnjevima = ");
    Serial.println(kutZaPrikaz / 100.0, 2);

    Serial.print("Kalibrirani kut = ");
    Serial.println(kalibriraniKut / 100.0, 2);

    Serial.println();
   }
  }
}
```

Završna napomena

U ovom programu spojeni su svi važni elementi rada sustava:

- obrada prijelaza preko nule (**wraparound**)
- kalibracija prema stvarnom mehaničkom sustavu
- pohrana referentnih vrijednosti u **EEPROM**
- postavljanje radne nule tipkalom
- prikaz podataka u serijskom monitoru

Na taj način dobiva se sustav koji nije samo funkcionalan, nego i prilagođen stvarnim uvjetima rada.

Najčešće greške

Nema wraparound obrade
Koristi se samo oduzimanje
Kalibracija nije spremljena
EEPROM se koristi u svakoj petlji
Vrijednosti “skaču”

Završni zaključak

Za ispravan rad uređaja potrebno je:

Razumjeti kružnu prirodu mjerenja
Pravilno obraditi wraparound
Provesti kalibraciju
Spremiti vrijednosti u EEPROM

Tek kombinacijom svih ovih koraka dobiva se s**tabilan i precizan uređaj.**

Kratki test – Wraparound, kalibracija i EEPROM

1. (razumijevanje pojma)

Što znači pojam **wraparound** kod senzora AS5600?

a) Greška senzora
b) Prijelaz s maksimalne na minimalnu vrijednost
c) Kalibracija senzora
d) Pohrana podataka u memoriju

2. (računski zadatak)

Zadano je:

pocetna = 4000;
trenutna = 100;

Izračunaj razliku:

a) bez wraparound obrade
b) s wraparound obradom

3. (razumijevanje problema)

Zašto obična operacija:

razlika = trenutna - pocetna;

može dati pogrešan rezultat?

4. (kod – dopuni)

Dopuni kod za ispravnu wraparound obradu:

```
if (trenutna < pocetna) {
   _____;
}
```

5. (koncept kalibracije)

Što predstavljaju varijable:

minVal
maxVal

6. (razumijevanje kalibracije)

Zašto je kalibracija potrebna kod ovog sustava?

7. (EEPROM – osnovno)

Što je EEPROM?

a) Privremena memorija
b) Memorija koja briše podatke nakon gašenja
c) Memorija koja trajno pohranjuje podatke
d) Vrsta senzora

8. (primjena)

Kada je ispravno koristiti EEPROM?

a) U svakoj petlji programa
b) Samo nakon kalibracije ili promjene postavki
c) Nikada
d) Samo za ispis podataka

9. (analiza)

Što će se dogoditi ako se kalibracijske vrijednosti ne pohrane u EEPROM?

10. (napredno – razumijevanje)

Zašto se koristi funkcija:

(kraj + 4096 - pocetak) & 0x0FFF

Što ona osigurava?

☑ Rješenja

1. b
2. a) -3900
   b) 196
3. Jer senzor radi kružno (wraparound)
4. trenutna += 4096;
5. 0° i 180° referentne vrijednosti
6. Zbog mehaničkih odstupanja i netočnosti
7. c
8. b
9. Kalibracija se gubi nakon gašenja
10. Osigurava kružno računanje (wraparound)