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

// ========================================================
// KUTOMJER S KALIBRACIJOM 0° I 180°
// Metodička verzija s hrvatskim nazivima i komentarima
// --------------------------------------------------------
// Logika rada:
// 1. Nakon uključenja, prve 2 sekunde moguće je ući u kalibraciju.
// 2. Ako se u tom vremenu pritisne tipkalo:
//    - prvi pritisak sprema položaj 0°
//    - drugi pritisak sprema položaj 180°
//    - zatim se kalibracija automatski završava
// 3. Ako se u prve 2 sekunde tipkalo ne pritisne,
//    program nastavlja normalan rad.
// 4. U normalnom radu kratki pritisak postavlja radnu nulu.
// 5. Kalibracijski podaci i radna nula spremaju se u EEPROM.
// ========================================================

// ================== KONFIGURACIJA ==================

// Vrsta displeja:
// 1 = common cathode (zajednička katoda)
// 0 = common anode   (zajednička anoda)
#define ZAJEDNICKA_KATODA 0

// Pinovi znamenki displeja slijeva nadesno
const uint8_t PIN_ZNAMENKE[4] = {12, 13, A2, A3};

// Pinovi segmenata A, B, C, D, E, F, G, DP
const uint8_t PIN_SEGMENTI[8] = {4, 5, 6, 7, 8, 9, 10, 11};

// Tipkalo je spojeno između A0 i GND
const uint8_t PIN_TIPKALO = A0;

// Vrijeme osvježavanja prikaza
const uint16_t PERIOD_OSVJEZAVANJA_MS = 100;   // 10 puta u sekundi
const uint16_t TRAJANJE_MUX_KORAKA_US = 800;  // vrijeme jedne znamenke

// Debounce tipkala
const unsigned long VRIJEME_DEBOUNCE_MS = 25;

// Koliko traje "prozor" za ulazak u kalibraciju nakon paljenja
const unsigned long VRIJEME_ULAZA_U_KALIBRACIJU_MS = 2000;

// Prag za prepoznavanje dugog pritiska
const unsigned long DUGI_PRITISAK_MS = 1000;
```

```
// ================== EEPROM PODACI ==================

// Prepoznatljiva oznaka po kojoj znamo jesu li podaci valjani
const uint32_t EEPROM_OZNAKA = 0x4B555431;   // "KUT1"

// Struktura podataka koju spremamo u EEPROM
struct PostavkeKutomjera {
  uint32_t oznaka;             // provjera valjanosti
  uint16_t sirovoNula;         // sirova vrijednost senzora za 0°
  uint16_t sirovoStoOsamdeset; // sirova vrijednost senzora za 180°
  uint16_t radnaNulaStotinke;  // korisnička nula u stotinkama stupnja
};

PostavkeKutomjera postavke;

// ================== GLOBALNE VARIJABLE ==================

AS5600 senzor;

// Polje za prikaz na displeju
uint8_t  okvir[4];        // segmentni kod po znamenki
bool     decimalnaTocka[4];
uint8_t  indeksMux = 0;

unsigned long zadnjiMuxTik = 0;
unsigned long zadnjeOsvjezenje = 0;

// Vrijeme pokretanja sustava
unsigned long vrijemePokretanja = 0;

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

// Stanja rada programa
enum StanjeRada {
  STARTNI_PROZOR,         // prve 2 sekunde nakon uključenja
  KALIBRACIJA_NULA,       // čeka spremanje položaja 0°
  KALIBRACIJA_180,        // čeka spremanje položaja 180°
  NORMALAN_RAD            // normalan rad uređaja
};

StanjeRada stanje = STARTNI_PROZOR;

// Mapa znamenki za 7-segmentni displej
// Redoslijed bitova: gfedcba, bit7 = DP
const uint8_t MAPA_ZNAMENKI[10] = {
```

```
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111  // 9
};

// ================== POMOĆNE FUNKCIJE ==================

// Uključivanje ili isključivanje svih znamenki
inline void postaviSveZnamenke(bool ukljuci) {
  const uint8_t ZNAMENKA_UKLJ = ZAJEDNICKA_KATODA ? LOW  : HIGH;
  const uint8_t ZNAMENKA_ISKLJ = ZAJEDNICKA_KATODA ? HIGH : LOW;

  for (uint8_t i = 0; i < 4; i++) {
    digitalWrite(PIN_ZNAMENKE[i], ukljuci ? ZNAMENKA_UKLJ :
ZNAMENKA_ISKLJ);
  }
}

// Upis stanja svih segmenata
inline void upisiSegmente(uint8_t maskaSegmenata) {
  const uint8_t SEGMENT_UKLJ = ZAJEDNICKA_KATODA ? HIGH : LOW;
  const uint8_t SEGMENT_ISKLJ = ZAJEDNICKA_KATODA ? LOW : HIGH;

  for (uint8_t i = 0; i < 8; i++) {
    bool ukljuci = maskaSegmenata & (1 << i);
    digitalWrite(PIN_SEGMENTI[i], ukljuci ? SEGMENT_UKLJ : SEGMENT_ISKLJ);
  }
}

// Osvježavanje displeja multipleksiranjem
void osvjeziDisplej() {
  unsigned long sada = micros();
  if ((sada - zadnjiMuxTik) < TRAJANJE_MUX_KORAKA_US) return;
  zadnjiMuxTik = sada;

  // Najprije ugasi sve znamenke
  postaviSveZnamenke(false);

  // Pripremi kod trenutačne znamenke
  uint8_t kod = okvir[indeksMux];
  if (decimalnaTocka[indeksMux]) kod |= 0b10000000;
```

```
  // Upiši segmente
  upisiSegmente(kod);

  // Uključi samo jednu znamenku
  const uint8_t ZNAMENKA_UKLJ = ZAJEDNICKA_KATODA ? LOW : HIGH;
  digitalWrite(PIN_ZNAMENKE[indeksMux], ZNAMENKA_UKLJ);

  // Sljedeći put prelazimo na iduću znamenku
  indeksMux = (indeksMux + 1) % 4;
}

// Čitanje sirove 12-bitne vrijednosti senzora (0..4095)
uint16_t ocitajSirovuVrijednost() {
  return (senzor.rawAngle() & 0x0FFF);
}

// Kružna razlika između dvije sirove vrijednosti
uint16_t kruznaRazlika(uint16_t pocetak, uint16_t kraj) {
  return (kraj + 4096 - pocetak) & 0x0FFF;
}

// Spremanje podataka u EEPROM
void spremiPostavke() {
  EEPROM.put(0, postavke);
}

// Učitavanje podataka iz EEPROM-a
void ucitajPostavke() {
  EEPROM.get(0, postavke);

  // Ako nema valjanih podataka, upisujemo početne vrijednosti
  if (postavke.oznaka != EEPROM_OZNAKA) {
    postavke.oznaka = EEPROM_OZNAKA;
    postavke.sirovoNula = 0;
    postavke.sirovoStoOsamdeset = 2048;
    postavke.radnaNulaStotinke = 0;
    spremiPostavke();
  }
}

// Računanje kalibriranog broja sirovih koraka za puni krug
uint16_t dohvatiKalibriraniPuniKrug() {
  uint16_t polaKruga = kruznaRazlika(postavke.sirovoNula,
postavke.sirovoStoOsamdeset);

  // Zaštita od loše ili slučajne kalibracije
  if (polaKruga < 1000 || polaKruga > 3000) {
    polaKruga = 2048;
  }
```

```
  uint16_t puniKrug = 2 * polaKruga;
  if (puniKrug == 0) puniKrug = 4096;

  return puniKrug;
}

// Pretvorba sirove vrijednosti u kalibrirani kut u stotinkama stupnja
// npr. 1234 znači 12,34°
uint16_t izracunajKalibriraniKutStotinke(uint16_t sirovo) {
  uint16_t relativnoOdNule = kruznaRazlika(postavke.sirovoNula, sirovo);
  uint16_t puniKrug = dohvatiKalibriraniPuniKrug();

  uint32_t kutStotinke = (uint32_t)relativnoOdNule * 36000UL / puniKrug;
  kutStotinke %= 36000UL;

  return (uint16_t)kutStotinke;
}

// Izračun kuta za prikaz, uz uvažavanje radne nule
uint16_t izracunajKutZaPrikazStotinke(uint16_t sirovo) {
  uint16_t kalibriraniKut = izracunajKalibriraniKutStotinke(sirovo);

  if (kalibriraniKut >= postavke.radnaNulaStotinke) {
    return kalibriraniKut - postavke.radnaNulaStotinke;
  } else {
    return kalibriraniKut + 36000 - postavke.radnaNulaStotinke;
  }
}

// Priprema znamenki za prikaz kuta s jednom decimalom
// Primjer: 123,4
void pripremiPrikazIzStotinki(uint16_t stotinke) {
  int z0 = stotinke / 10000;        // stotice
  int z1 = (stotinke / 1000) % 10;  // desetice
  int z2 = (stotinke / 100) % 10;   // jedinice
  int z3 = (stotinke / 10) % 10;    // prva decimala

  // Gašenje vodećih nula
  okvir[0] = (z0 == 0) ? 0 : MAPA_ZNAMENKI[z0];
  okvir[1] = ((z0 == 0) && (z1 == 0)) ? 0 : MAPA_ZNAMENKI[z1];
  okvir[2] = MAPA_ZNAMENKI[z2];
  okvir[3] = MAPA_ZNAMENKI[z3];

  // Decimalna točka između jedinica i prve decimale
  decimalnaTocka[0] = false;
  decimalnaTocka[1] = false;
  decimalnaTocka[2] = true;
  decimalnaTocka[3] = false;
```

```
}

// Jednostavan ispis u serijski monitor
void ispisiStanje(uint16_t sirovo, uint16_t kalibriraniKut, uint16_t
prikazKut) {
  Serial.print("Očitana sirova vrijednost kuta: ");
  Serial.print(sirovo);

  //Serial.print(" | Kalibrirani kut: ");
  //Serial.print(kalibriraniKut / 100.0, 2);

  Serial.print(" | Kut za prikaz: ");
  Serial.println(prikazKut / 100.0, 2);

  //Serial.print(" | Kal 0: ");
  //Serial.print(postavke.sirovoNula);

  //Serial.print(" | Kal 180: ");
  //Serial.print(postavke.sirovoStoOsamdeset);

  //Serial.print(" | Radna nula: ");
  //Serial.println(postavke.radnaNulaStotinke / 100.0, 2);
}

// Kratki treptaj svih znamenki kao potvrda
void potvrdaTreptajem(uint8_t brojPonavljanja, uint16_t trajanjeMs) {
  for (uint8_t i = 0; i < brojPonavljanja; i++) {
    okvir[0] = MAPA_ZNAMENKI[8];
    okvir[1] = MAPA_ZNAMENKI[8];
    okvir[2] = MAPA_ZNAMENKI[8];
    okvir[3] = MAPA_ZNAMENKI[8];
    decimalnaTocka[0] = false;
    decimalnaTocka[1] = false;
    decimalnaTocka[2] = false;
    decimalnaTocka[3] = false;

    unsigned long start = millis();
    while (millis() - start < trajanjeMs) {
      osvjeziDisplej();
    }

    okvir[0] = 0;
    okvir[1] = 0;
    okvir[2] = 0;
    okvir[3] = 0;

    start = millis();
    while (millis() - start < trajanjeMs) {
      osvjeziDisplej();
```

```
    }
  }
}

// ================== OBRADA TIPKALA ==================

// Funkcija vraća:
// 0 = ništa
// 1 = kratki pritisak
// 2 = dugi pritisak
uint8_t ocitajDogadajTipkala() {
  bool stanjeTipkala = digitalRead(PIN_TIPKALO);
  unsigned long sada = millis();

  // Ako se stanje promijenilo, pokrećemo debounce
  if (stanjeTipkala != prethodnoStanjeTipkala) {
    vrijemePromjeneTipkala = sada;
    prethodnoStanjeTipkala = stanjeTipkala;
  }

  // Pričekaj da prođe debounce
  if ((sada - vrijemePromjeneTipkala) <= VRIJEME_DEBOUNCE_MS) {
    return 0;
  }

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

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

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

  // Otpuštanje tipkala = mogući kratki pritisak
  if (!pritisnuto && biloPritisnuto) {
    unsigned long trajanje = sada - vrijemePritiskaTipkala;

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

```
    }
  }

  biloPritisnuto = pritisnuto;
  return 0;
}

// ================== LOGIKA RADA ==================

void obradiLogikuStanja() {
  uint8_t dogadajTipkala = ocitajDogadajTipkala();
  unsigned long sada = millis();

  switch (stanje) {

    case STARTNI_PROZOR:
      // Tijekom prve 2 sekunde korisnik može ući u kalibraciju
      if (dogadajTipkala == 1 || dogadajTipkala == 2) {
        stanje = KALIBRACIJA_NULA;
        Serial.println("Ulazak u kalibraciju.");
        Serial.println("Postavi krak na 0° i pritisni tipkalo.");
        potvrdaTreptajem(1, 120);
      }
      else if ((sada - vrijemePokretanja) >=
VRIJEME_ULAZA_U_KALIBRACIJU_MS) {
        stanje = NORMALAN_RAD;
        Serial.println("Normalan rad.");
      }
      break;

    case KALIBRACIJA_NULA:
      // Čekamo da korisnik postavi krak na 0° i pritisne tipkalo
      if (dogadajTipkala == 1) {
        postavke.sirovoNula = ocitajSirovuVrijednost();
        Serial.print("Spremljeno 0°: ");
        Serial.println(postavke.sirovoNula);

        stanje = KALIBRACIJA_180;
        Serial.println("Postavi krak na 180° i pritisni tipkalo.");
        potvrdaTreptajem(2, 100);
      }
      break;

    case KALIBRACIJA_180:
      // Čekamo da korisnik postavi krak na 180° i pritisne tipkalo
      if (dogadajTipkala == 1) {
        postavke.sirovoStoOsamdeset = ocitajSirovuVrijednost();

        // Nakon završene kalibracije radna nula se vraća na 0
```

```
        postavke.radnaNulaStotinke = 0;

        spremiPostavke();

        Serial.print("Spremljeno 180°: ");
        Serial.println(postavke.sirovoStoOsamdeset);

        Serial.println("Kalibracija zavrsena.");
        potvrdaTreptajem(3, 80);

        stanje = NORMALAN_RAD;
      }
      break;

    case NORMALAN_RAD:
      // U normalnom radu:
      // kratki pritisak = postavi radnu nulu
      if (dogadajTipkala == 1) {
        uint16_t sirovo = ocitajSirovuVrijednost();
        postavke.radnaNulaStotinke =
izracunajKalibriraniKutStotinke(sirovo);
        spremiPostavke();

        Serial.println("Postavljena nova radna nula.");
        potvrdaTreptajem(1, 80);
      }
      break;
  }
}

// ================== SETUP ==================

void setup() {
  // Postavljanje pinova segmenata i znamenki kao izlaza
  for (uint8_t i = 0; i < 8; i++) pinMode(PIN_SEGMENTI[i], OUTPUT);
  for (uint8_t i = 0; i < 4; i++) pinMode(PIN_ZNAMENKE[i], OUTPUT);

  // Tipkalo koristimo s internim pull-up otpornikom
  pinMode(PIN_TIPKALO, INPUT_PULLUP);

  // Početno gašenje segmenata i znamenki
  const uint8_t SEGMENT_ISKLJ = ZAJEDNICKA_KATODA ? LOW : HIGH;
  for (uint8_t i = 0; i < 8; i++) digitalWrite(PIN_SEGMENTI[i],
SEGMENT_ISKLJ);
  postaviSveZnamenke(false);

  // Pokretanje I2C komunikacije i senzora
  Wire.begin();
  senzor.begin();
```

```
  // Serijska komunikacija
  Serial.begin(9600);
  delay(300);

  // Učitavanje spremljenih podataka
  ucitajPostavke();

  // Početne vrijednosti prikaza
  okvir[0] = 0;
  okvir[1] = 0;
  okvir[2] = 0;
  okvir[3] = 0;

  decimalnaTocka[0] = false;
  decimalnaTocka[1] = false;
  decimalnaTocka[2] = false;
  decimalnaTocka[3] = false;

  // Bilježimo vrijeme pokretanja
  vrijemePokretanja = millis();
  stanje = STARTNI_PROZOR;

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

// ================== LOOP ==================

void loop() {
  // Displej treba osvježavati stalno
  osvjeziDisplej();

  // Obrada stanja i tipkala
  obradiLogikuStanja();

  // Osvježavanje prikaza i serijskog ispisa
  unsigned long sada = millis();
  if ((sada - zadnjeOsvjezenje) >= PERIOD_OSVJEZAVANJA_MS) {
    zadnjeOsvjezenje = sada;

    uint16_t sirovo = ocitajSirovuVrijednost();

    if (stanje == KALIBRACIJA_NULA) {
      // U kalibraciji 0° prikazujemo 0.0
      pripremiPrikazIzStotinki(0);
    }
```

```
    else if (stanje == KALIBRACIJA_180) {
      // U kalibraciji 180° prikazujemo 180.0
      pripremiPrikazIzStotinki(18000);
    }
    else {
      // U normalnom radu prikazujemo izmjereni kut
      uint16_t kalibriraniKut = izracunajKalibriraniKutStotinke(sirovo);
      uint16_t kutZaPrikaz = izracunajKutZaPrikazStotinke(sirovo);

      pripremiPrikazIzStotinki(kutZaPrikaz);
      ispisiStanje(sirovo, kalibriraniKut, kutZaPrikaz);
    }
  }
}
```