Arduino platforma

Prekidi

Akvizicija i obrada podataka

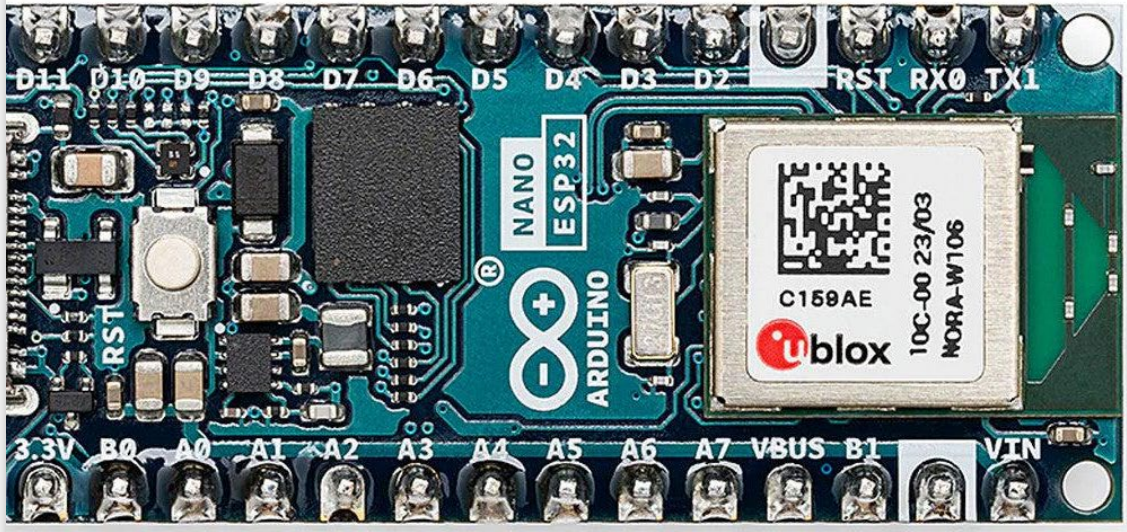
Najvažnije karakteristike:

- Programiranje u C/C++ okruženju.
- Direktan pristup digitalnim i analognim ulazima i izlazima.
- Podrška za veliki broj senzora i elektronskih modula.
- Mogućnost upravljanja motorima, relejima, LED sistemima i drugim aktuatorima.
- Podrška za serijsku, I2C, SPI i druge komunikacione protokole.
- Proširivanje funkcionalnosti putem biblioteka.
- Mogućnost povezivanja sa računarima, serverima i IoT platformama.

Prekidi (Interrupts)

- Prekidi su jedna od najvažnijih funkcija mikrokontrolera.
- Omogućavaju da Arduino reaguje na događaj u tačno određenom trenutku,
- nemapotrebe da neprekidno proverava da li se taj događaj dogodio.

``Suština prekida je da mikrokontroler privremeno prekida izvršavanje glavnog programa, obrađuje događaj, a zatim nastavlja tamo gde je stao.``



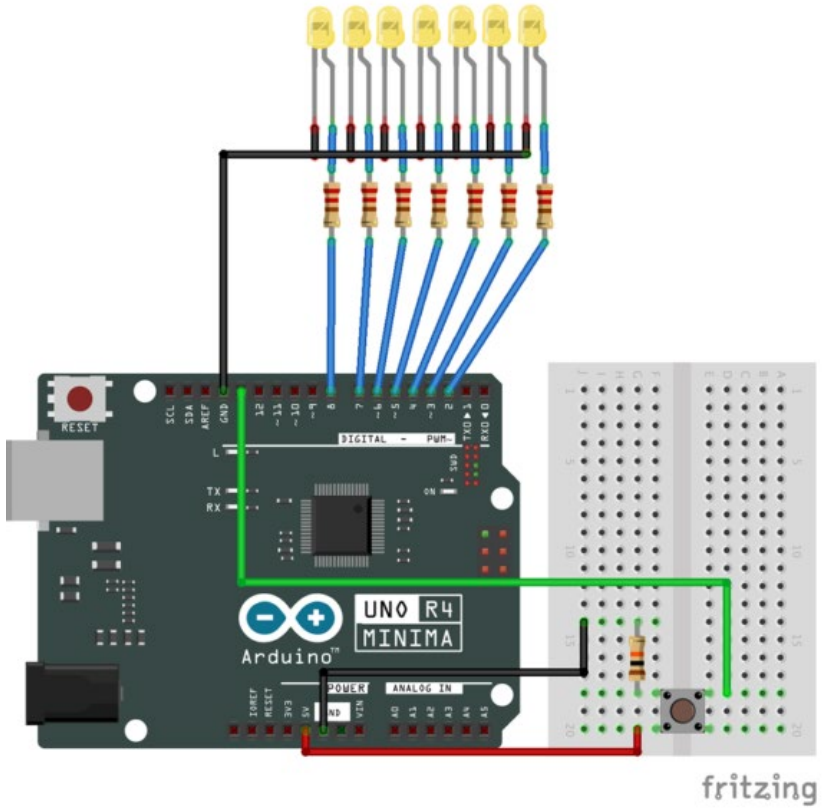
Zašto su nam potrebni prekidi?

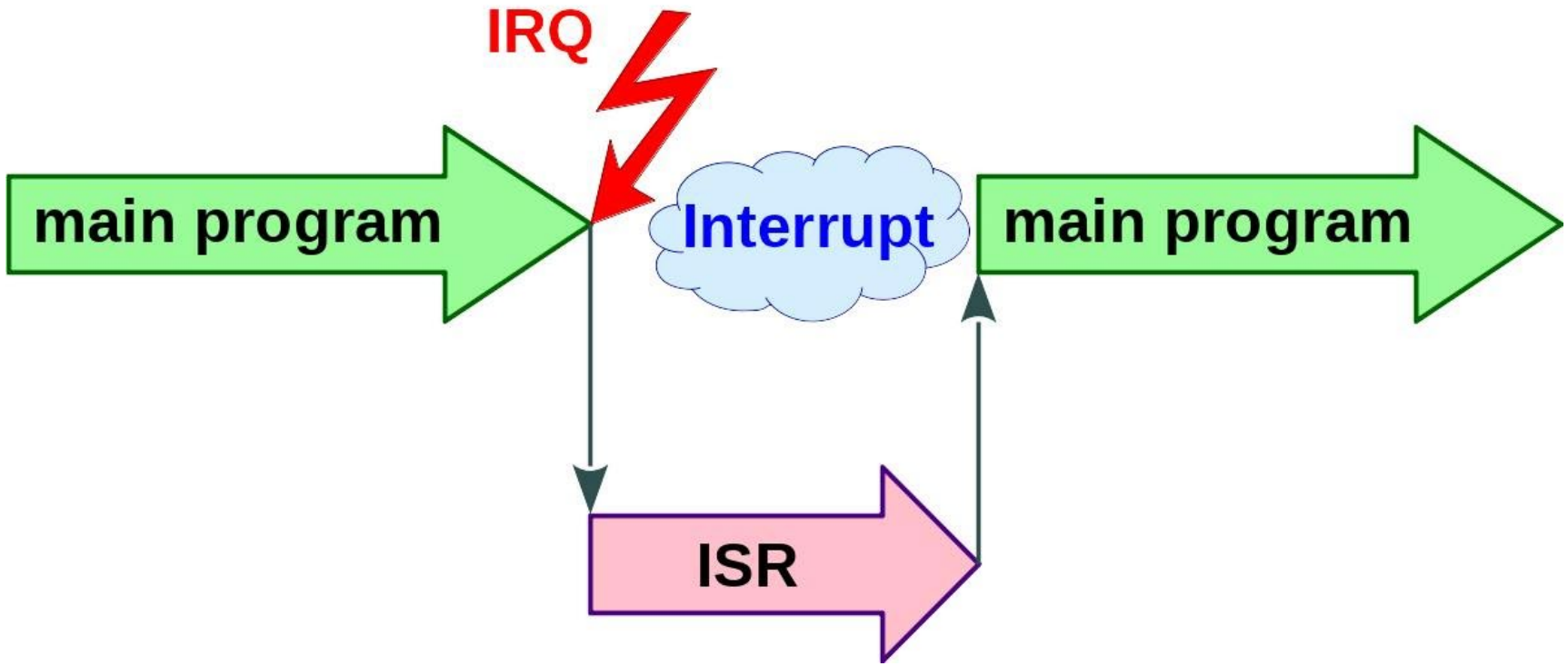
Primer: Arduino upravlja robotskim vozilom koje ima:

- dva elektromotora;
- enkodere za merenje broja obrtaja;
- ultrazvučni senzor;
- taster za hitno zaustavljanje;
- displej za prikaz podataka.

Ako koristimo samo standardnu funkciju loop(), Arduino mora periodično da proverava stanje svih uređaja.

Problem je što neki događaji mogu da se dogode veoma brzo i da budu propušteni dok se izvršava drugi deo programa.





Arduino izvršava program i u tom trenutku dobija impuls sa enkodera motora. Prekid omogućava da odmah registruje impuls, poveća brojač i vrati se prethodnom zadatku.
Bez prekida, impuls može biti propušten.

Kako funkcionišu prekidi?

- **Izvršavanje glavnog programa**

Arduino izvršava instrukcije u funkciji loop().

- **Pojava događaja**

Na ulaznom pinu pojavljuje se signal ili se dogodi drugi događaj koji može izazvati prekid.

- **Zaustavljanje trenutnog izvršavanja**

Mikrokontroler čuva stanje potrebno za povratak u program i prelazi na prekidnu rutinu.

- **Izvršavanje ISR rutine**

ISR (Interrupt Service Routine) obrađuje događaj.

- **Povratak u glavni program**

Izvršavanje se nastavlja od mesta na kojem je prekinuto.

Vrste prekida

A. Spoljašnji prekidi (External Interrupts)

- Aktiviraju se promenom signala na određenom fizičkom pinu.

Primeri:

- pritisak tastera;
- aktiviranje senzora;
- impuls sa enkodera;
- signal alarma;
- detekcija prolaska objekta.

Na Arduino Uno R3, standardni spoljašnji prekidi dostupni su na pinovima 2 i 3.

B. Prekidi tajmera (Timer Interrupts)

- Generišu se kada hardverski tajmer dostigne određenu vrednost ili događaj poređenja.

Koriste se za:

- precizno vremensko upravljanje;
- periodično očitavanje senzora;
- generisanje kontrolnih signala;
- upravljanje motorima;
- realizaciju sistema sa vremenski kritičnim zadacima.

C. Prekidi komunikacije

- Nastaju kada se dogodi komunikacioni događaj.

Na primer:

- prijem podataka preko UART-a;
- prijem podataka preko SPI-ja;
- promena stanja na I2C magistrali.
- Većinu ovih prekida Arduino biblioteke obrađuju u pozadini.

D. ADC prekidi

- Mogu se aktivirati kada se završi analogno-digitalna konverzija.
- Koriste se kada je potrebno efikasno prikupljanje analognih podataka

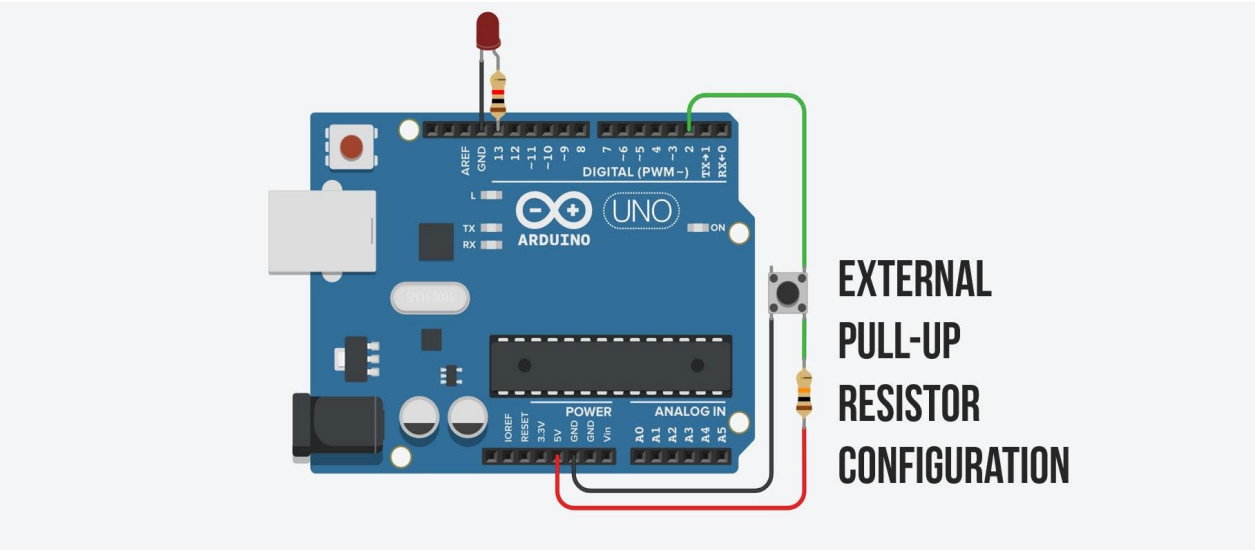
Prekidi (Interrupts)

```
void setup() {
  Serial.begin(9600);
  pinMode(2,
INPUT_PULLUP);
  attachInterrupt(
    digitalPinToInterrupt(2),
    buttonISR,
    FALLING
  );
}
void loop() {
  if (buttonPressed) {
    noInterrupts();
    buttonPressed = false;
    interrupts();

    Serial.println("Detektovan
pritisak tastera");
  }
}
```

Prekidne rutine treba da budu kratke. Ne treba u njima koristiti `delay()` ili složene operacije.

Primer -Prekid pomoću tastera



Cilj: kada pritisnemo taster, Arduino registruje događaj pomoću prekida.

```
volatile bool buttonEvent =
false;
void buttonISR() {
  buttonEvent = true;
}
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(
   digitalPinToInterrupt(2),
    buttonISR,
    FALLING
  );
}
void loop() {
  if (buttonEvent) {
    noInterrupts();
    buttonEvent = false;
    interrupts();
    Serial.println("Taster
je pritisnut!");
  }
  // Glavni program
nastavlja rad
}
```

OSVRT

- attachInterrupt() povezuje pin sa prekidnom rutinom.
- digitalPinToInterrupt(2) određuje izvor prekida.
- FALLING znači da se prekid aktivira pri prelasku sa HIGH na LOW.
- volatile govori kompajleru da se promenljiva može promeniti van uobičajenog toka programa.
- buttonISR() predstavlja prekidnu rutinu.

Napomena:

kod mehaničkih tastera postoji problem odbijanja kontakata (debouncing). Jedan pritisak može proizvesti više električnih prelaza. Zato je potrebno dodati softversko ili hardversko filtriranje.

Gde se prekidi koriste u stvarnim sistemima?

Enkoderi elektromotora

- Svaki impuls predstavlja informaciju o pomeranju osovine. Prekidi omogućavaju precizno brojanje impulsa i izračunavanje brzine.

Sigurnosni sistemi

- Detekcija alarma, otvaranja zaštitnih vrata i zahteva za zaustavljanje mašine. Za stvarne sigurnosne funkcije potreban je odgovarajući namenski sigurnosni hardver, a ne samo Arduino prekid.

Merači protoka

- Brojanje impulsa senzora protoka omogućava određivanje količine protekle tečnosti.

GPS i navigacioni sistemi

- Prijem i obrada podataka, vremensko označavanje događaja i sinhronizacija. Serijska komunikacija i namenski hardver često obavljaju veliki deo posla nezavisno od glavne petlje.

Brzi senzori

- Registrovanje kratkotrajnih impulsa, prekida svetlosnog snopa i promena stanja senzora.

Pravila

Pravilo	Zašto je važno?
ISR mora biti kratka	Da se program brzo vrati glavnom zadatku
Koristiti `volatile`	Da kompajler pravilno pristupa promenljivim koje menjaju prekidi
Izbegavati `delay()`	Može blokirati obradu i izazvati probleme
Ne koristiti `Serial.print()` u ISR	Serijska komunikacija može blokirati ili poremetiti rad
Pažljivo deliti promenljive	Sprečava nekonzistentno čitanje podataka
Voditi računa o prioritetima	Neki prekidi mogu biti blokirani dok se obrađuje drugi

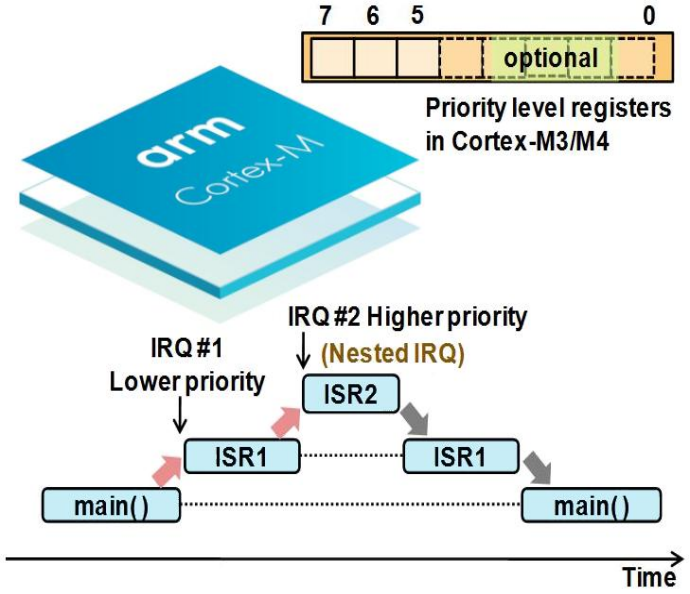
Prekidi u odnosu na polling

Osobina	Polling	Interrupt
Princip	Stalna provera stanja	Reakcija na događaj
Opterećenje procesora	Može biti veliko	Često manje
Reakcija na događaj	Zavisi od učestalosti provere	Po nastanku događaja
Mogućnost propuštanja impulsa	Veća kod sporog polling-a	Manja
Složenost programa	Jednostavnije	Zahteva pažljivije programiranje

Napredne tehnike prekida u Arduino sistemima

Nested Interrupts – ugnježđeni prekidi

- Ugnježđeni prekidi predstavljaju situaciju u kojoj se tokom izvršavanja jedne prekidne rutine aktivira drugi prekid.
- Mikrokontroler privremeno prekida obradu prvog događaja kako bi obradio drugi događaj, a zatim nastavlja prethodnu prekidnu rutinu.



Gde se koriste?

•Industrijski kontroleri.

•Robotski sistemi.

•Sistemi za upravljanje motorima.

•Uređaji sa vremenski kritičnim alarmima.

•Sistemi za prikupljanje podataka velikom brzinom.

Primer: Mikrokontroler koji upravlja industrijskim uređajem:

- Prekid 1: prijem podataka sa senzora.
- Prekid 2: detekcija kritičnog alarma.
- Prekid 3: vremenski signal tajmera.

Ako se tokom obrade senzora pojavi kritični alarm, sistem može biti projektovan tako da obradi alarm pre nego što završi obradu senzora.

Gde se koriste?

•Industrijski kontroleri.
•Robotski sistemi.
•Sistemi za upravljanje motorima.
•Uređaji sa vremenski kritičnim alarmima.
•Sistemi za prikupljanje podataka velikom brzinom.

Kako to funkcioniše na Arduino Uno?

- ATmega328P po ulasku u standardnu prekidnu rutinu automatski isključuje globalno dozvoljene prekide.
- Drugi prekidi standardno ne izvršavaju dok traje prva ISR rutina.
- Ugnježdeni prekidi mogu se omogućiti eksplicitnim ponovnim uključivanjem globalnih prekida,
- Pažljivo projektovanje.

```
ISR(TIMER1_COMPA_vect) {

  // Obrada prvog
prekida

  sei();  // Ponovno
dozvoljavanje prekida

  // Nastavak obrade
}
```

Interrupt Priority – prioriteti prekida

Zašto su nam potrebni prioriteti?

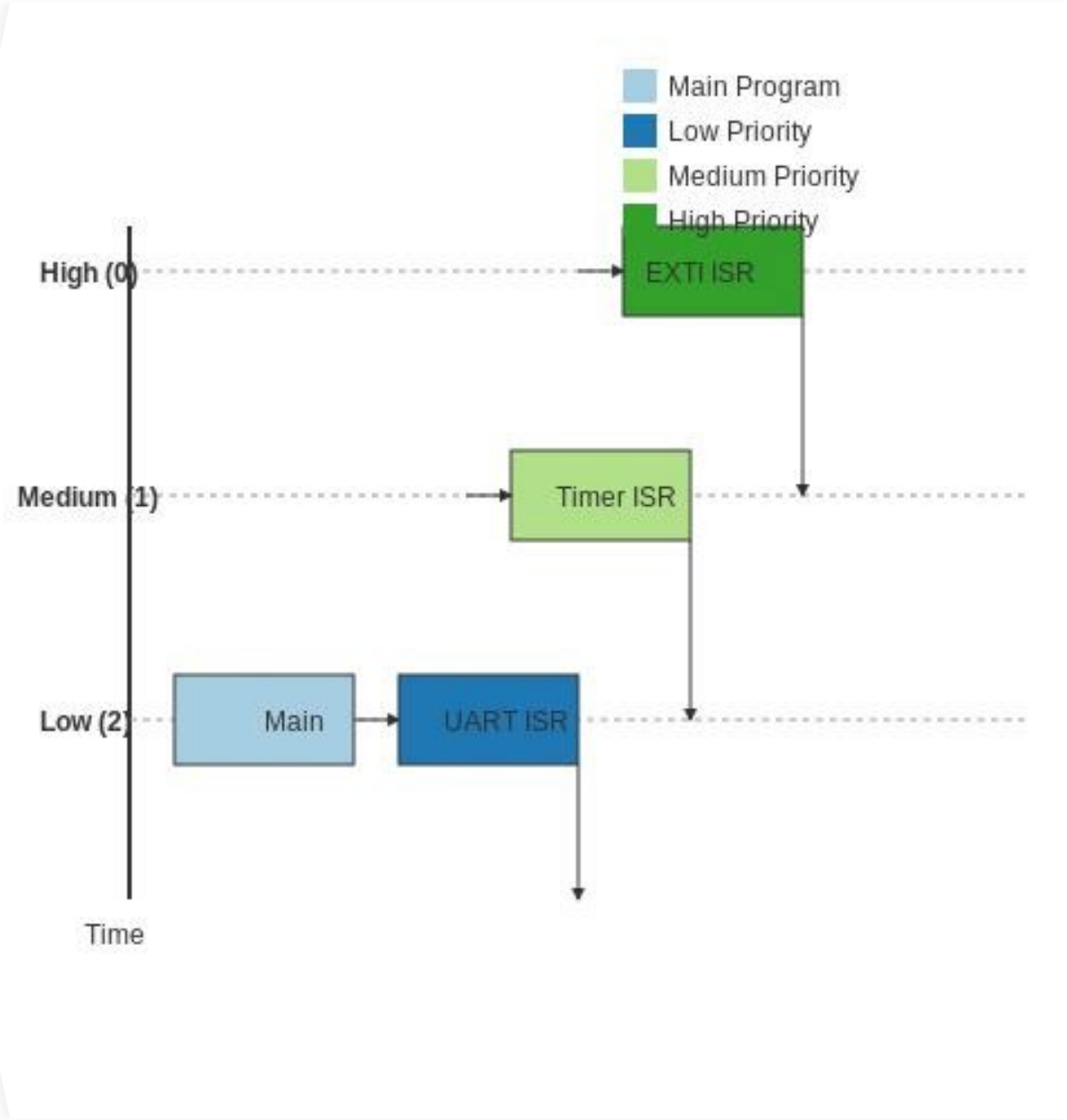
Zamislimo sistem koji istovremeno prima tri događaja:

- Kritični sigurnosni signal.
- Impuls enkodera motora.
- Zahtev za osvežavanje displeja.

Da li sva tri događaja imaju isti značaj?

- Ne. Kritični sigurnosni događaj treba da ima prednost u obradi.

Prioriteti omogućavaju određivanje redosleda obrade prekida.



- Kod ATmega328P postoji fiksni redosled prioriteta hardverskih izvora prekida. On se određuje pozicijom odgovarajućeg vektora u tabeli prekida.
- Arduino Uno nema opšti sistem u kome programer jednostavno dodeljuje proizvoljne numeričke prioritete prekidima.
- Kod ESP32, koji koristi Xtensa ili RISC-V arhitekturu zavisno od modela, postoje naprednije mogućnosti upravljanja prioritetima i nivoima prekida.

Primer praktične organizacije

Događaj	Zahtev za reakciju
Kritični alarm	Najviši
Enkoder motora	Visok
Tajmer za merenje	Srednji
Osvežavanje displeja	Nizak

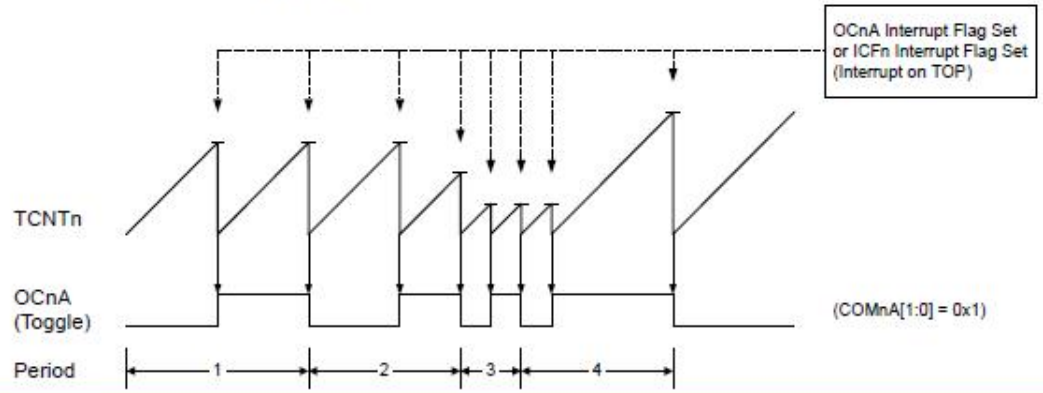
Timer Compare Match – prekidi poređenja tajmera

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNT1) increases until a Compare Match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 21-6. CTC Mode, Timing Diagram



- Mikrokontroler poseduje hardverske tajmere koji broje taktne impulse.
- Kada vrednost brojača dostigne unapred zadatu vrednost, nastaje događaj Compare Match i može se aktivirati prekid.

```
  Serial.begin(9600);
}
void loop() {
static unsigned long previous = 0;
unsigned long current;
noInterrupts();
 current = milliseconds;
 interrupts();
  if (current - previous >= 1000) {
  previous = current;
  Serial.println("Prosla je jedna
sekunda");
  }
}
```

Koristimo prescaler 64:

$$f_{timer} = \frac{16\,000\,000}{64} = 250\,000\ Hz$$

Vreme jednog taktnog impulsa:

$$T = \frac{1}{250\,000} = 4\ \mu s$$

Za 1 ms potrebno je:

$$N = \frac{1000}{4} = 250$$

Zato je vrednost registra:

$$OCR1A = 250 - 1 = 249$$

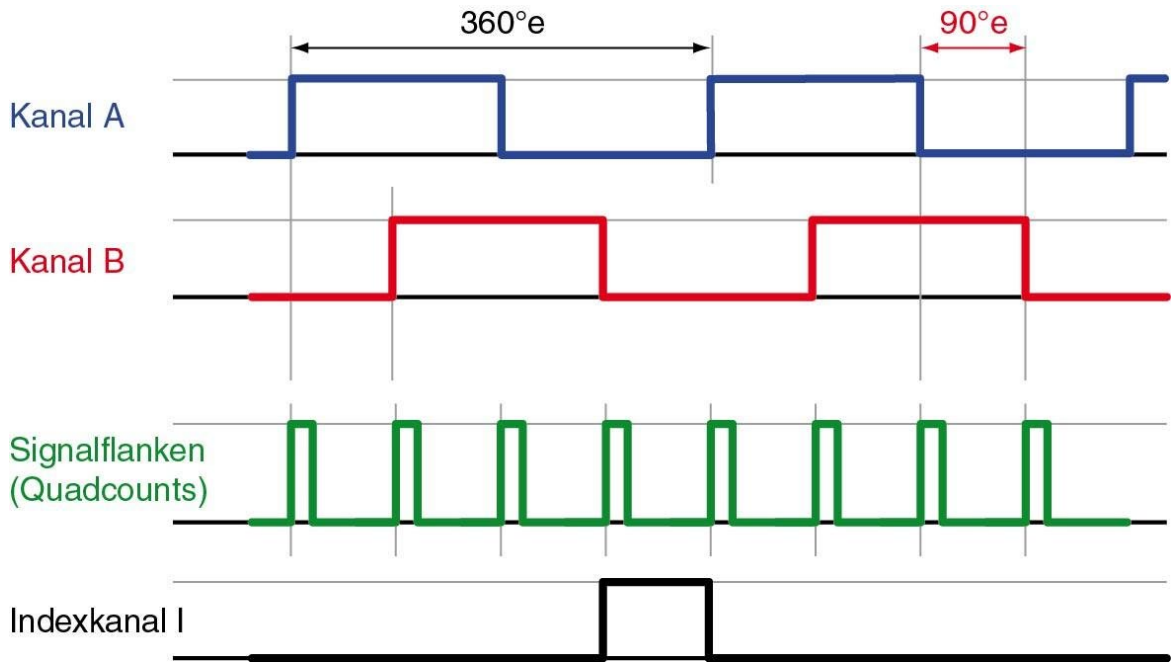
Rezultat je prekid svakih 1 ms.

Quadrature Encoder – dekodiranje A/B signala

Šta je kvadraturni enkoder?

Kvadraturni enkoder je senzor koji generiše dva digitalna signala, označena kao A i B.

Ta dva signala imaju fazni pomeraj od 90 električnih stepeni.



Na osnovu redosleda promena signala A i B možemo utvrditi:

- Da li se osovina okreće u jednom ili drugom smeru.
- Koliko se osovina okrenula.
- Kolika je brzina obrtanja.
- Da li je došlo do promene smera.

Kako se određuje smer?

Kod jednog smera rotacije redosled promena je:

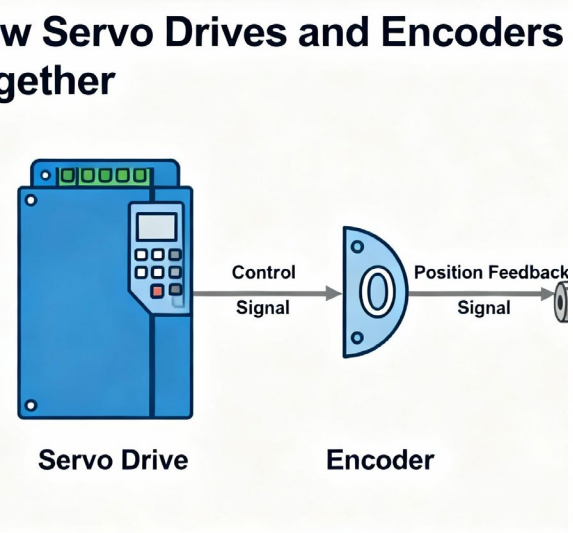
A → B → A → B

Kod suprotnog smera:

B → A → B → A

U stvarnosti se smer određuje na osnovu stanja oba signala, a ne samo njihovog pojedinačnog prelaska.





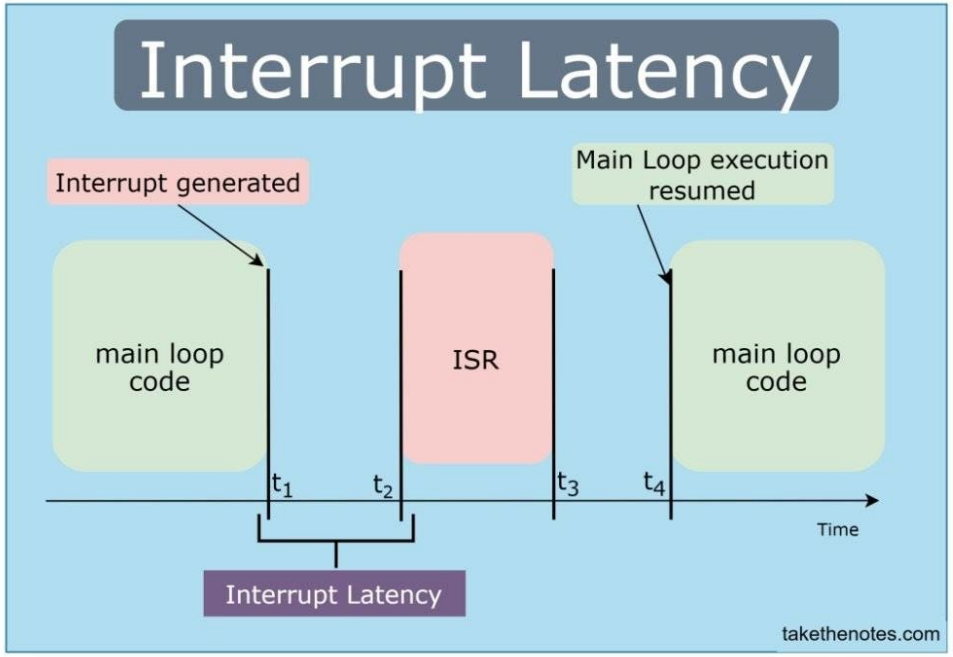


Gde se koristi?

- Robotska vozila – određivanje pređenog rastojanja.
- CNC mašine – precizno pozicioniranje.
- Servo sistemi – povratna informacija o položaju.
- Industrijski motori – kontrola brzine i smera.
- Autonomni sistemi – odometrija i procena kretanja.

Interrupt Latency – kašnjenje prekida

Interrupt Latency - vreme koje protekne od nastanka događaja koji zahteva prekid do početka izvršavanja odgovarajuće prekidne rutine.



- Senzor detektuje prepreku u trenutku 10,000 ms.
- Mikrokontroler počinje izvršavanje ISR u trenutku 10,008 ms.

Kašnjenje je:

$$L = 10,008 - 10,000 = 0,008\ ms$$

Odnosno:

$$L = 8\ \mu s$$

Od čega zavisi kašnjenje?

- Arhitektura mikrokontrolera.
- Frekvencija procesora.
- Trenutna instrukcija koja se izvršava.
- Da li su drugi prekidi aktivni.
- Da li je globalna obrada prekida privremeno isključena.
- Dužina prethodne prekidne rutine.

Kod ATmega328P, minimalno hardversko vreme odziva za standardni prekid iznosi nekoliko taktnih ciklusa, ali stvarno kašnjenje zavisi od stanja procesora i izvršavanja programa.

Kako se smanjuje Interrupt Latency?

- Smanjiti trajanje ISR rutina.
- Izbegavati nepotrebno isključivanje prekida.
- Izbegavati blokirajuće funkcije.
- Koristiti odgovarajuće hardverske tajmere.
- Pravilno projektovati prioritete događaja.
- Izbegavati nepotrebne operacije unutar prekidne rutine.