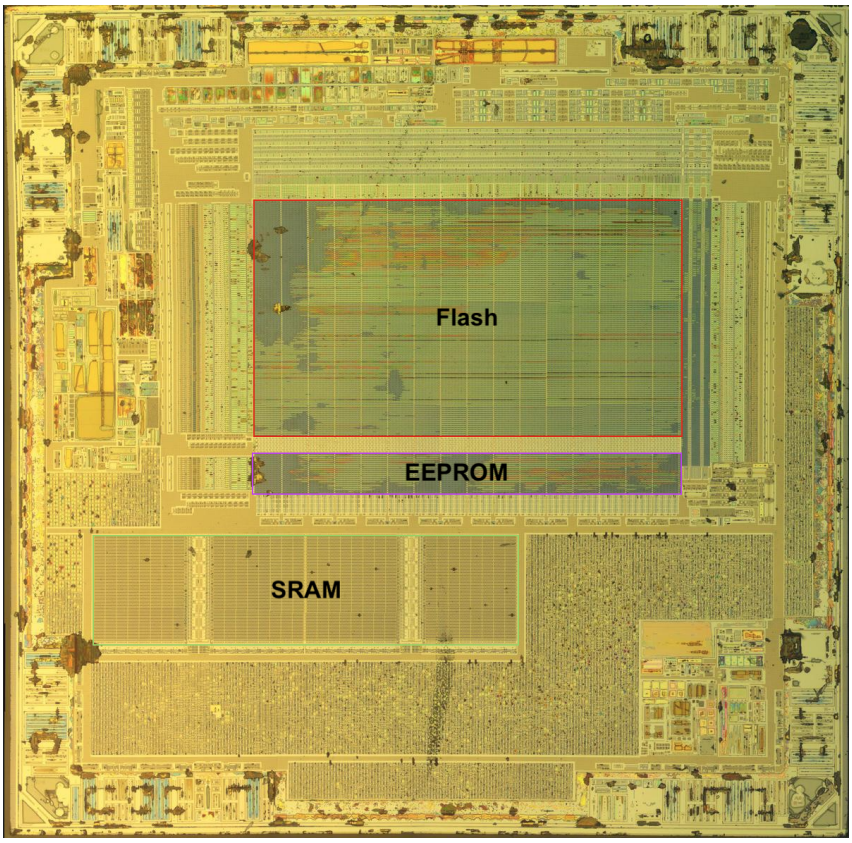
# 1 UPORABA MIKROKRMILNIKOV

Za projekte, ki vključujejo programabilno elektroniko, pogosto uporabljamo že izdelane krmilnike iz družine Arduino. Na teh vezjih lahko najdemo mikrokrmilnike proizvajalca Atmel. Najbolj pogosto uporabljena krmilnika (Arduino Uno in Arduino NANO) temeljita na mikrikrmilniku Atmega328p. Blokovna shema tega mikrokrmilnika je prikazana na sl. 2.

## 1.1 Shema mikrokrmilnika ATmega238

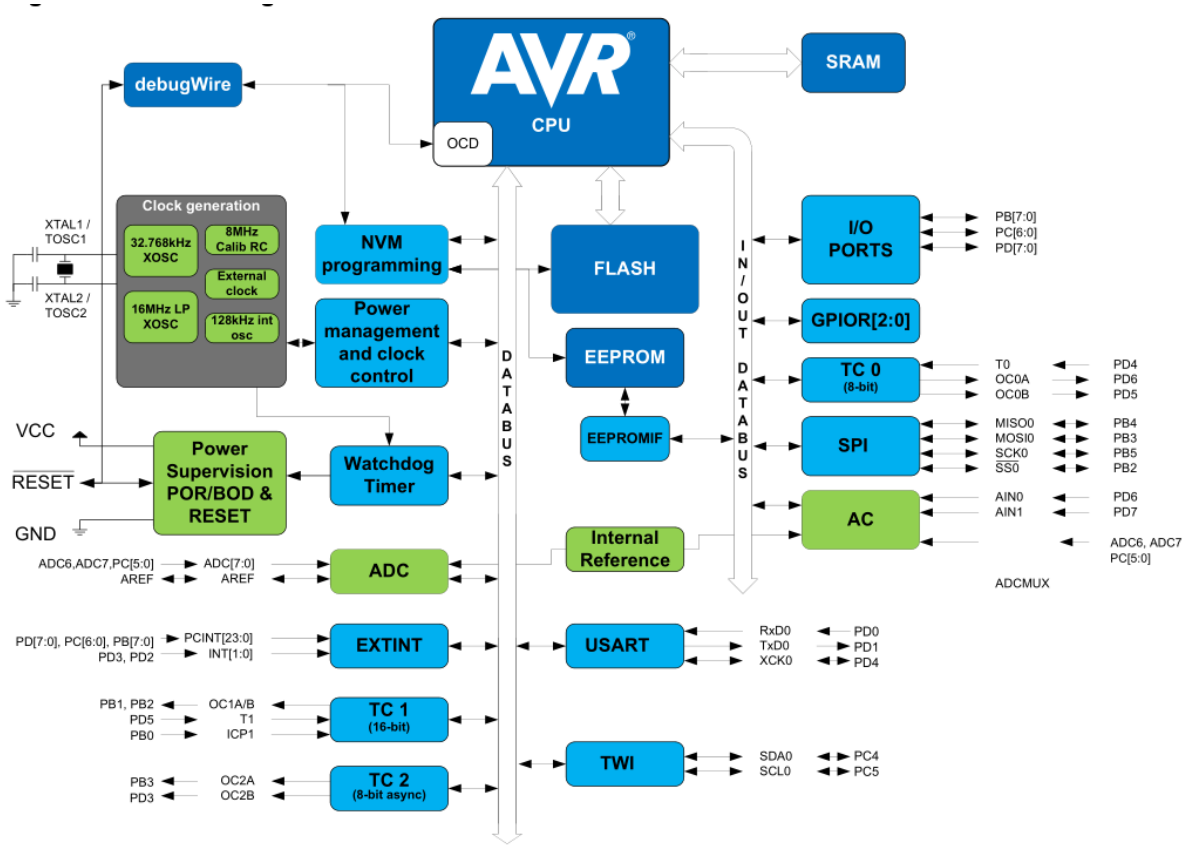
Mikrokrmilniki so integrirana vezja, z zelo kompleksno notranjo strukturo. Sestavlja jih na milijone tranzistorjev, ki s povezavami in ostalimi osnovnimi elementi sestavljajo smiselne logične sklope. Povečana slika dejanskega integriranega vezja mikrokrmilnika ATmega238 je na sl. 1.




**Slika 1:** Slika vezja mikrokrmilnika ATmega238 na plošči SiO2

Iz sl. 1 je nemogoče razbrati posamezne dele integriranega vezja. Opazimo lahko le večje enake sklope, ki so namenjene spominskim funkcijam. Še bolj podrobno sliko pa si lahko ogledate na povezavi ATmega238-SiO2.

Pri tako kompleksnih vezjih je bolj smiselno, da posamezne logične sklope predstavimo z blokovno shemo. Tako shemo lahko najdemo v navodilih za uporabo mikrokrmilnika ATmega238 in je prikazana na sl. 2.




**Slika 2:** Blokovna shema mikrokrmilnika ATmega238.

### 1.1.1 NALOGA: Glavni deli krmilnika

1. Na blokovni shemi označi pomembne dele krmilnika in zapiši njihov glavni namen (funkcijo). Glavni sestavni deli so: generator delovnega takta (ura), centralno procesna enota, začasni (delovni) spomin, trajni spomin, vhodno-izhodne enote, komunikacijske enote …

Mikrokrmilniki na krmilnikih Arduino so že opremljeni s programom (angl. »boot loader«), ki poskrbi za ustrezno prepisovanje programske vsebine, ki jo računalnik pošlje preko USB vodila. Tako lahko enostavno programiramo mikrokrmilnike, ki so na ploščah krmilnikov Arduino.

#### 1.1.2 NALOGA: Osnovne nastavitve in testni program

1. Iz programskega okolja Arduino IDE prepišite nastavitve programatorja ter izpolni tbl. 1:

**Tabela 1:** Nastavitveni parametri programatorja.

| Nastavitveni parameter | Vrednost nastavitvenega parametra |
|---|---|
| Board (Plošča) | |
| ~~Processor~~ (Mikrokrmilnik) | |
| Port (vrata) | |
| Programmer (programator) | |

2. Iz primerov, ki so vključeni v programskem okolju Arduino IDE izberite Blink.ino in ga preskusite.

### 1.2 Programiranje krmilnikov Arduino

Za programiranje mikrokrmilnika skrbi odprtokodna programska koda - avrdude, ki se izvaja v ozadju programskega okolja Arduino IDE. Proces programiranja lahko bolj natančno spremljamo tako, da vključimo

```
File -> Preferences:

Shov verbose output during: [x] compilation [x] upload
```

#### 1.2.1 NALOGA: AVRDUDE - program za prenos strojne kode

1. Prepišite ukazno vrstico programa avrdude za premos strojne kode in
2. opišite pomen parametrov.

Več o parametrih lahko najdete na spletni strani avrdude

### 1.3 Uporaba vhodno-izhodnih priključkov na kmilniku Arduino

### 1.3.1 NALOGA: Shema krmilnika Arduino NANO

1. Oglejte si shemo krmilnika Arduino NANO in
2. skušaj ugotoviti na kateri priključek IO enote je priključena LED, ki je na krmilniku.
3. Izpolni tbl. 2

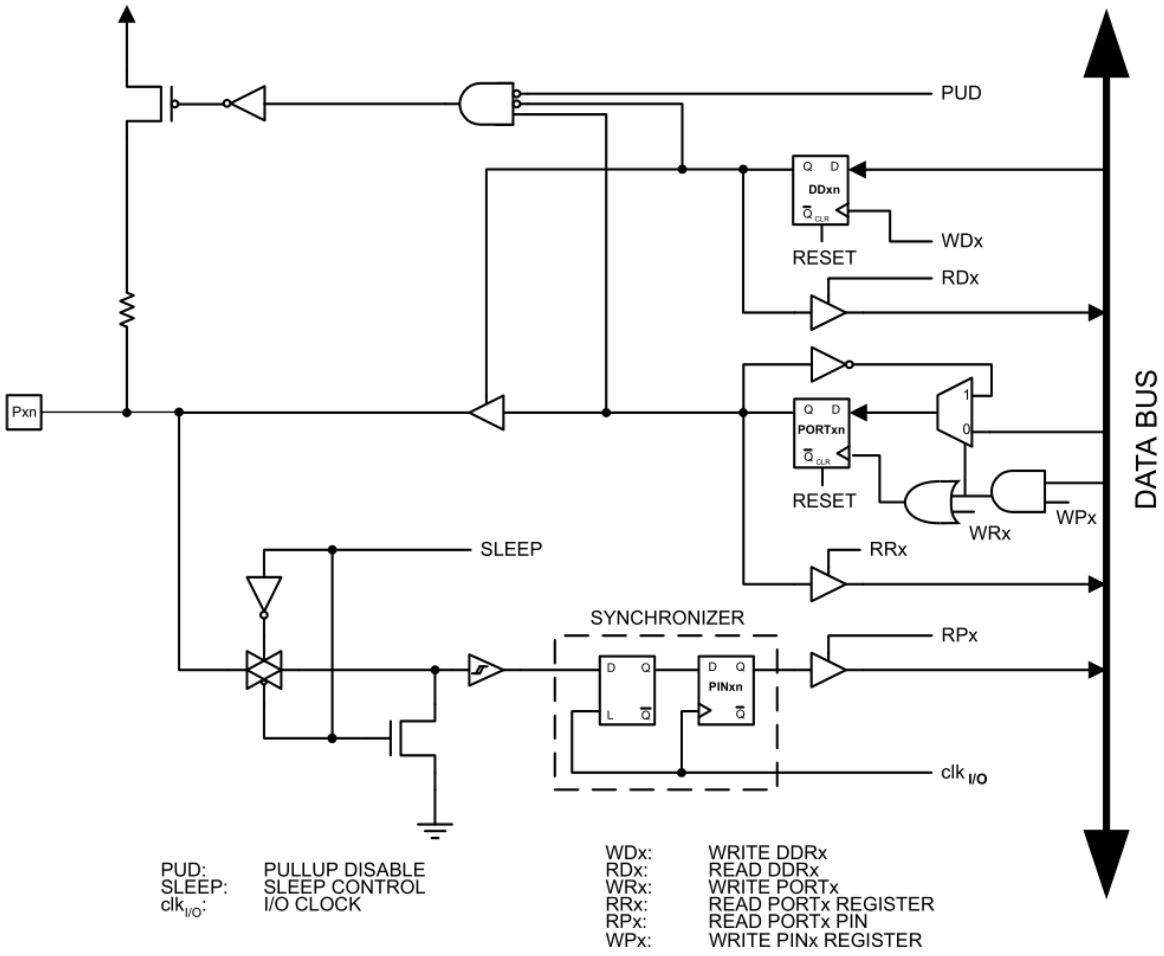
**Tabela 2:** Razporeditev priključkov na krmilniku in mikrikrmilniku.

| Funkcija krmilnika | Arduino NANO[a] | ATmega238[b] |
|---|---|---|
| LED na plošči | | |
| TxD | | |
| RxD | | |

[a] Oznaka priključka na krmilniku Arduino NANO.
[b] Oznaka priključka na mikrokrmilniku ATmega238.

## 1.4 Vhodno-izhodne enote mikrokrmilnika

Vhodno-izhodne enote mikrokrmilnika so dvo-smerne z možnostjo nastavitve upora proti napajanju. Stikalna shema enega priključka na neki vhodno-izhodni enoti je prikazana na sl. 3.




**Slika 3:** Shema priključka n na vhodno-izhodni enoti x.

Čeprav je shema na sl. 3 nekoliko bolj kompleksna, lahko ugotovimo, da za nastavitev delovanja vhodno-izhodne enote potrebujemo le dva signala:

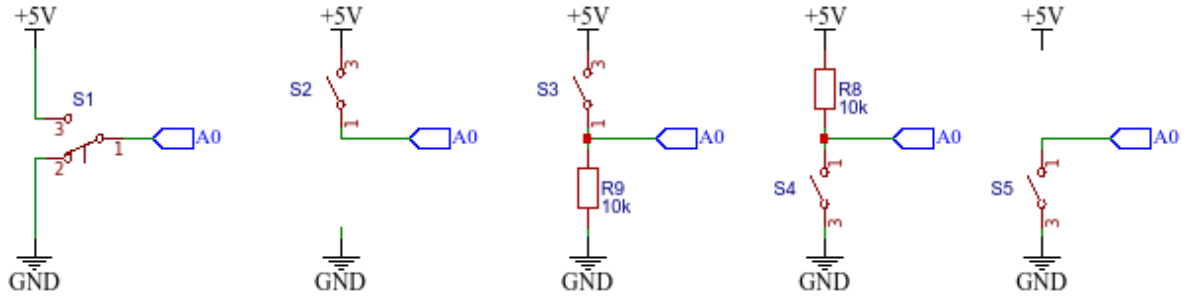
- WDx - (DDRx) - za določanje vhodne oz. izhodne funkcije priključka
- WPx - (PORTx) - za določanje vrednosti logičnega stanja na priključku x.

### 1.4.1 NALOGA: Krmilni registri mikrokrmilnika

1. Preoblikuj program Blink.ino tako, da boš krmilil vklop in izklop LED z nastavitvijo krmilnikh registrov.
2. Predstavite programsko kodo.

## 1.5 Branje napetostnega potenciala

Branje napetostnega potenciala je najbolj pogosto uporabljeno pri vzorčenju pritiska tipke. Poglejmo si tak primer.




**Slika 4:** Različne možnosti priključitve tipke na digitalni vhod.

### 1.5.1 NALOGA: Branje napetostnega potenciala

1. Sestavite prve tri različice priključitve tipke na digitalni vhod krmilnika A0 in preskusite spodnji program.
2. Razložite (ne-)delovanje.

```
void setup(){
    pinMode(13, OUTPUT);
    pinMode(A0, INPUT);
}

void loop(){
    int stikalo_je_sklenjeno = digitalRead(A0);
    if ( stikalo_je_sklenjeno == HIGH)
        digitalWrite(13, HIGH);
    else
        digitalWrite(13, LOW);
}
```

### 1.5.2 NALOGA: Uporaba upora proti napajanju

1. Spremenite delilnih napetosti tipka - upor tako, da bo upor vezan proti napajanju in nato spremenite program tako, da bo delovanje ostalo enako prejšnji nalogi. Priložite stikalno

shemo vezja in preoblikovan program.
2. Odstranite upor in preverite delovanje vezja. Kaj opazite, razložite delovanje.
3. Vključite notranji upor proti napajanju, ki se nahaja v mikrokrmilniku ob vsakem vhodno-izhodnem priključku (glej sl. 3). Priložite programsko spremembo.

## 1.6 Prehodni stiki stikal

Fizični preklopni elementi (kot je tipka) imajo tudi fizikalne lastnosti kot so trdota, trdnost, prožnost, elastičnost… Zaradi vseh teh lastnosti je preklop tipke iz enega položaja v drugega lahko tudi nekoliko nepredvidljiv.

### 1.6.1 NALOGA: Večkratni preklopi

1. Preskusite spodnji program in opišite njegovo delovanje.
2. Opišite zaznane težave.
3. Z osciloskopom ujemite enega od prehodov (0->1 ali 1->0) in
4. problem rešite:
    - programsko ter
    - elektronsko. Obe rešitvi dokumentirajte.

```
void setup()
{
    const int LED = 13;
    pinMode(LED, OUTPUT);
    pinMode(A0, INPUT_PULLUP);
}

int i=0;

void loop()
{
    int tipka_je_pritisnjena = !digitalRead(A0);
    if (tipka_je_pritisnjena)
    {
      i++;
      while (tipka_je_pritisnjena)
        tipka_je_pritisnjena = !digitalRead(A0);
      int i_je_liho = i % 2;
      if (i_je_liho)
        digitalWrite(13, HIGH);
      else
        digitalWrite(13, LOW);
    }
}
```