UNIVERZA V NOVEM MESTU

FAKULTETA ZA STROJNIŠTVO

Tehnični uvod v Arduino

Navodila

Pripravil in uredil Gorazd Hlebanja

Novo mesto, 2022

KAZALO

1 UVOD

Arduino je odprtokodna elektronska platforma, ki temelji na strojni in programski opremi s poudarkom na njeni preprosti uporabi. Arduino plošče lahko berejo vhode, kot so svetloba, ki jo zazna senzor, prst na gumbu ali sporočilo na Twitterju, in jih spremenijo v izhode, kot so aktiviranje motorja, vklop LED, objave nečesa na spletu. Plošči lahko povemo, kaj naj naredi, tako da pošljemo nabor navodil mikro-krmilniku na plošči. Za to uporabimo programski jezik Arduino, ki temelji na prototipni platformi Wiring[1], in programski opremi Arduino (IDE), ki temelji na grafični knjižnici Processing[2]. V osnovi pa je programski jezik zasnovan na C++.

Skozi leta je Arduino postal jedro tisočerih projektov, od vsakdanjih objektov do kompleksnih znanstvenih instrumentov. Okoli te odprtokodne platforme se je zbrala svetovna skupnost ustvarjalcev - študentov, ljubiteljev, umetnikov, programerjev in profesionalcev, njihovi prispevki pa so prispevali k neverjetni količini dostopnega znanja, ki je lahko v veliko pomoč tako začetnikom kot strokovnjakom.

Arduino se bil ustvarjen v Ivrea Interaction Design Institute kot enostavno orodje za hitro izdelavo prototipov, namenjeno študentom brez predznanja v elektroniki in programiranju. Takoj, ko je plošča Arduino dosegla širšo skupnost, se je začela spreminjati, da bi se prilagodila novim potrebam in izzivom, ter ustvarila ponudbo od preprostih 8-bitnih plošč, do proizvodov za IoT aplikacije, prenosnih naprav, 3D tiskanja in vgrajenih okolij.

[1] **Wiring** is an open-source electronics prototyping platform composed of a programming language, an integrated development environment (IDE), and a single-board microcontroller. It was developed starting in 2003 by Hernando Barragán.

[2] Wiring builds on Processing, an open project initiated by Casey Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. **Processing** is a free graphical library and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

2 ZAKAJ JE ARDUINO UPORABEN?

Ker je koncept Arduina zasnovan na preprosti in dostopni uporabniški izkušnji je bil Arduino uporabljen v tisočih različnih projektih in aplikacijah. Programska oprema Arduino je enostavna za uporabo za začetnike in dovolj prilagodljiva za napredne uporabnike. Deluje v sistemih Mac, Windows in Linux. Učitelji in dijaki (pa tudi učenci zadnjih letnikov osnovnega izobraževanja) ga uporabljajo za izdelavo cenenih znanstvenih instrumentov, za dokazovanje kemijskih in fizikalnih principov ali za osnove programiranja in robotike. Oblikovalci in arhitekti gradijo interaktivne prototipe, glasbeniki in umetniki ga uporabljajo za instalacije in eksperimentirajo z novimi glasbili. Ustvarjalci ga seveda uporabljajo za gradnjo številnih projektov, ki so na primer razstavljeni na sejmu Maker Faire. Arduino je ključno orodje za učenje novih stvari. Kdorkoli – otroci, ljubitelji, umetniki, programerji – lahko začne s prvimi koraki preprosto, tako da sledi navodilom kompleta po korakih ali deli ideje na spletu z drugimi člani Arduino skupnosti.

Za fizično računalništvo je na voljo veliko drugih mikrokrmilnikov in mikrokrmilniških platform. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard in številni drugi ponujajo podobne funkcionalnosti. Vsa ta orodja zajamejo neurejene podrobnosti programiranja mikrokrmilnikov in jih zavijejo v paket, dovolj enostaven za uporabo. Arduino prav tako poenostavlja proces dela z mikrokrmilniki, vendar ponuja nekaj prednosti za učitelje, študente in zainteresirane amaterje pred drugimi sistemi:

- Poceni - plošče Arduino so razmeroma poceni v primerjavi z drugimi platformami mikrokrmilnikov. Najcenejšo različico modula Arduino je mogoče sestaviti ročno in celo vnaprej sestavljeni moduli Arduino stanejo manj kot 50 €.
- Programska oprema Arduino (IDE) deluje v operacijskih sistemih Windows, Macintosh OSX in Linux (cross-platform). Večina mikrokrmilniških sistemov je omejena na Windows.
- Enostavno in jasno programsko okolje - programska oprema Arduino (IDE) je enostavna za uporabo za začetnike, a dovolj prilagodljiva, da jo lahko izkoristijo tudi napredni uporabniki.
- Odprtokodna in razširljiva programska oprema - Programska oprema Arduino je objavljena kot odprtokodna orodja, ki so na voljo za razširitev s strani izkušenih programerjev. Jezik je mogoče razširiti s knjižnicami C++ in ljudje, ki želijo razumeti tehnične podrobnosti, lahko naredijo preskok z Arduina na programski jezik AVR C,

na katerem temelji. Podobno lahko dodajo kodo AVR-C neposredno v svoje programe Arduino, če želijo.

- Odprtokodna in razširljiva strojna oprema - Načrti plošč Arduino so objavljeni pod licenco Creative Commons, tako da lahko izkušeni oblikovalci vezij naredijo svojo različico modula, jo razširijo in izboljšajo. Celo razmeroma neizkušeni uporabniki lahko sestavijo različico modula, da bi razumeli, kako deluje, in prihranili denar.

3 KAKO SE LOTIMO DELA S PLATFORMO ARDUINO?

Začnemo z vodičem Kako začeti - Getting started guide. Če pa iščemo navdih, lahko najdemo mnogo različnih praktičnih vaj na Arduino Project Hub.
Tekst vodiča Arduino getting started guide je licenciran pod Creative Commons Attribution-ShareAlike 3.0 License. Primeri kod v vodiču pa so dani v javno domeno.

3.1 Strojna oprema

Arduino je doslej razvil več kot 100 elektronskih proizvodov: plošče, zaščitne vmesnike, razširitvene module, nosilce, komplete in druge pripomočke. V nadaljevanju bomo pregledali trenutno dobavljive (oktober 2022) skupine plošč, kot so Nano, MKR in klasična družina.

3.2 Arduino Pro

Arduino je razvil tudi plošče v skladu z industrijskimi standardi. Cilj teh izjemno zmogljivih proizvodov z varnostjo na industrijski ravni, ki uporabljajo ekosistem Arduino, je uvajanje zmogljivih algoritmov umetne inteligence in strojnega učenja.

3.3 Kompleti

Kompleti so odličen način za začetek dela z Arduinom. The klasični **Starter Kit** vsebuje Arduino UNO, nabor elektronskih component in knjigo s 15-imi poglavji v pomoč pri začetku
Komplet **Oplà IoT kit** vključuje komponente, ki so potrebne za izdelavo osupljivih projektov IoT[3], komplet **Sensor Kit** pa vsebuje niz odličnih samo-povezljivih (plug-and-play[4]) senzorjev in aktuatorjev. Ti kompleti imajo tudi svojo namensko vsebinsko platformo z več kul projekti, ki jim je treba slediti korak za korakom.

[3] The **Internet of things** (**IoT**) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.
[4] In computing, a **plug and play** (**PnP**) device or computer bus is one with a specification that facilitates the discovery of a hardware component in a system without the need for physical device configuration or user intervention in resolving resource conflicts.

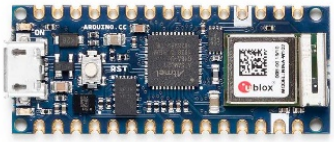
Arduino Starter Kit

Arduino Oplà IoT Kit

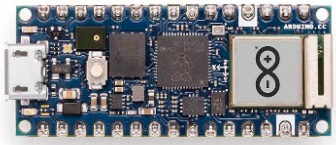
Arduino Sensor Kit

3.4 Nano družina

Družina Nano je zbirka plošč z majhnim odtisom, vendar z mnogimi funkcijami. Razpon je od poceni, osnovnega Nano Every do Nano 33 BLE Sense / Nano RP2040 Connect z več funkcijami, ki vsebujeta tudi module Bluetooth® / Wi-Fi. Te plošče imajo tudi nabor vgrajenih senzorjev, kot so temperatura/vlažnost, tlak, gibanje, mikrofon itd. Prav tako jih je mogoče programirati z MicroPython-om s podporo strojnega učenja.

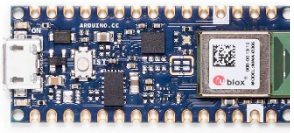


Arduino Nano 33 IoT



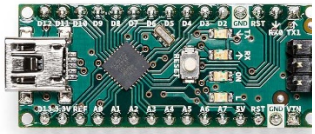
Arduino Nano RP2040 Connect

Arduino Nano 33 BLE Sense

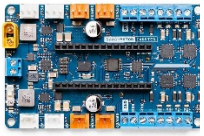


Arduino Nano 33 BLE

Arduino Nano Every



Arduino Nano



Arduino Nano Motor Carrier

3.5 MKR družina

MKR družina je serija plošč, vmesnikov in nosilcev, ki jih jih lahko kombiniramo za različne projekte brez dodatnih vezij. Vsaka plošča je opremljena z radijskim modulom (razen MKR

Zero), ki omogoča komunikacijo Wi-Fi, Bluetooth®, LoRa®, Sigfox, NB-IoT. Vse plošče v družini temeljijo na 32-bitnem procesorju Cortex-M0 SAMD21 z nizko porabo energije in so opremljene s kripto čipom za varno komunikacijo.

Vmesniki in nosilci MKR družine so zasnovani tako, da se poveča nabor funkcij same plošče, npr. kot so okoljski senzorji, GPS, Ethernet, krmilnik motorja in RGB matrika.

3.5.1 Platforme

Arduino MKR 1000 WiFi

Arduino MKR WiFi 1010

Arduino MKR FOX 1200

Arduino MKR WAN 1300

Arduino MKR WAN 1310

Arduino MKR GSM 1400

Arduino MKR NB 1500

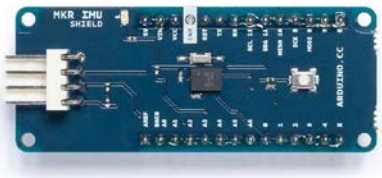
Arduino MKR Vidor 4000

Arduino MKR Zero

3.5.2 Vmesniki in nosilci

Arduino MKR ENV Shield Rev2

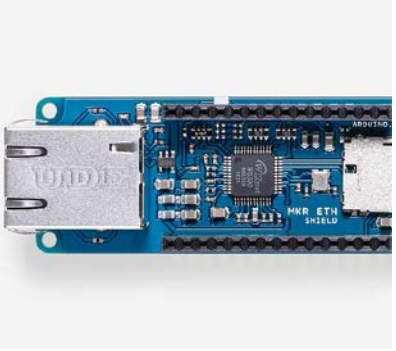
Arduino MKR GPS Shield



Arduino MKR IMU Shield

Arduino MKR RGB Shield

Arduino MKR THERM Shield



Arduino MKR ETH Shield

Arduino MKR 485 Shield

Arduino MKR CAN Shield

Arduino MKR MEM Shield

3.6 Klasična družina

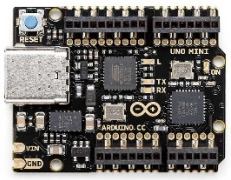
V klasični družini najdemo plošče, kot je legendarni Arduino UNO in druge klasične plošče, kot sta Leonardo in Micro. Te plošče tvorijo hrbtenico Arduino projekta in so izjemno popularne že vrsto let in bodo to tudi še naprej.

3.6.1 Platforme

Arduino UNO R3

Arduino Mega 2560 Rev3

Arduino Leonardo

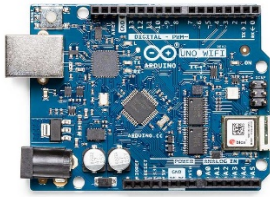


Arduino UNO Mini Limited Edition

Arduino Due

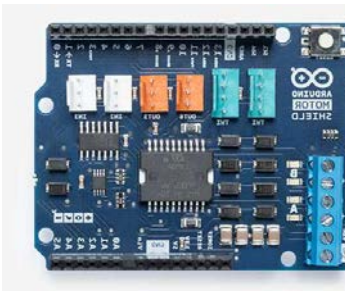
Arduino Micro

Arduino Zero

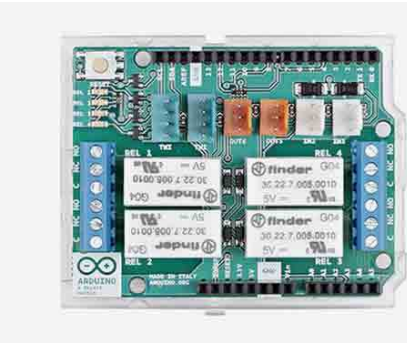


Arduino UNO WiFi Rev2

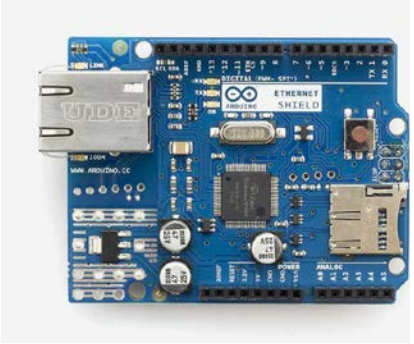
3.6.2 Vmesniki



Arduino Motor Shield Rev3



Arduino 4 Relay Shield



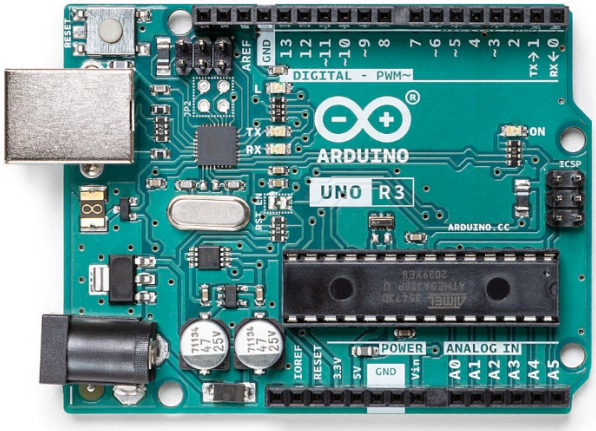
Arduino Ethernet Shield Rev2

4 ARDUINO UNO

Arduino Uno je plošča z mikrokrmilnikom zasnovana na 8-bitnem Atmel ATmega328P (podatkovni list) procesorju. Ima 14 digitalnih vhodno/izhodnih priključkov[5]. Od lahko 6

[5] Input/output pins

(posebej označenih) uporabljamo kot PWM[6] izhode. Vsebuje tudi 6 analognih vhodov, 16 MHz keramični resonator (CSTCE16M0V53-R0)[7], USB povezavo, napajalni priključek, ICSP header[8] in gumb za reset (ponastavljanje). Torej plošča vsebuje vse, kar je potrebno za podporo mikrokrmilniku. Enostavno priključimo jo na računalnik z USB kablom ali napajamo preko AC/DC adapterja ali baterije in naprava je pripravljena za delo. Glede na to, da plošča ni draga (24 €) tudi ni glavni problem, če gre kaj narobe. V najslabšem primeru jo nadomestimo z novo.



"Uno" v italijanščini pomeni prvotno različico programske opreme, Arduino Software (IDE) 1.0. Uno plošča s programjem Arduino Software IDE verzije 1.0 sta referenčni verziji projekta Arduino, ki sta se razvili v novejše verzije. Uno plošča je prva v seriji množici USB Arduino plošč in je referenca za množico kasnejših in tudi že preteklih plošč.

"Uno" v italijanščini pomeni prvotno različico programske opreme, Arduino Software (IDE) 1.0. Uno plošča s programjem Arduino Software IDE verzije 1.0 sta referenčni verziji projekta Arduino, ki sta se razvili v novejše verzije. Uno plošča je prva v seriji množici USB Arduino plošč in je referenca za množico kasnejših in tudi že preteklih plošč.

[6] PWM – pulse width modulation – širina pulza pomeni vrednost izhoda (od 0 do 100 %).

[7] **A Ceramic Resonator is an electronic component consisting of a piece of a piezoelectric ceramic material with two or more metal electrodes attached**. When connected in an electronic oscillator circuit, resonant mechanical vibrations in the device generate an oscillating signal of a specific frequency.

[8] It is the **ICSP header** that allows the microcontroller to receive the firmware or program that does all the advanced functionalities that are desired. **ICSP** stands for In Circuit Serial Programming, it is a standard way to program AVR (Atmel) chips.

4.1 Tehnične specifikacije

Mikro-krmilnik	ATmega328P
Delovna napetost	5V
Vhodna napetost (priporočeno)	7-12V
Vhodna napetpst (maksimalno)	6-20V
Digitalni I/O priključki	14 (od tega jih 6 zagotavlja PWM izhod)
PWM Digitalni I/O priključki	6
Analogni vhodi	6
DC tok na I/O priključku	20 mA
DC tok na 3.3V priključku	50 mA
Flash spomin[9]	32 KB (ATmega328P) od tega uporablja 0.5 KB zagonski nalagalnik (bootloader)
SRAM[10]	2 KB (ATmega328P)
EEPROM[11]	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN (vgrajen LED)	13
Dolžina	68.6 mm
Širina	53.4 mm
Masa	25 g

4.2 Programiranje

Arduino plošče, tudi UNO, lahko programiramo na osebnem računalniku z Arduino

[9] Flash memory - Flash memory is a non-volatile memory chip used for storage and for transfering data between a personal computer (PC) and digital devices. It can be electronically reprogrammed and erased. It is often found in USB flash drives, MP3 players, digital cameras and solid-state drives.
[10] Static random-access *memory* (*static RAM* or *SRAM*) is a type of random-access *memory* (*RAM*) that uses latching circuitry (flip-flop) to store each bit.
[11] EEPROM (electrically erasable programmable read-only memory) is a user-modifiable ROM. It can be erased and reprogrammed (written to) repeatedly by applying an electrical voltage that is higher than normal. EEPROM is a type of non-volatile ROM that enables individual bytes of data to be erased and reprogrammed. That is why EEPROM chips are known as *byte erasable* chips. EEPROM is usually used to store small amounts of data in computing and other electronic devices.

Software (IDE). Arduino programje naložimo iz domače strani www.arduino.cc. Za uporabniški vmesnik lahko izberemo tudi slovenščino. V menu-ju Tools > Board (Orodja > Plošča) izberemo Arduino UNO: Nekaj več o programiranju kasneje, sicer pa so detajli na razpolago na reference in tutorials.

ATmega328 na Arduino Uno je vnaprej programiran s t.i. zagonskim nalagalnikom oz. »bootloader-jem[12]«, ki omogoča nalaganje nove kode v mikro-krmilnik brez uporabe zunanjega strojnega programatorja. Komunicira z uporabo originalnega STK500 protokola (reference).

Lahko pa tudi premostimo zagonski nalagalnik in programiramo mikro-krmilnik z ICSP (In-Circuit Serial Programming) header z uporabo zunanjega programatorja Arduino ISP ali podobnega vezja, gl. navodila za detajle.

ATmega16U2 (ali 8U2 na starejših ploščah) je programiran kot USB/serijski pretvornik. To je pomembno za prenos programa iz računalnika v Arduino UNO, pa tudi za serijsko prenašanje podatkov.

Arduino Uno ima varovalko (ponastavljivo), ki služi kot zaščita računalniških USB vrat pred kratkim stikom ali prevelikim tokom. Sicer ima računalnik sam notranjo zaščito, ta varovalka pa zagotavlja še dodaten nivo zaščite. Če je na USB tok večji od 500 mA, bo varovalka prekinila povezavo do odstranitve preobremenitve.

4.3 Napajanje

Arduino Uno ploščo lahko napajamo preko USB povezave ali z zunanjim napajanjem. Izbor vira napajanja je avtomatski.

Zunanje (ne-USB) napajanje je lahko preko AC/DC adapterja ali baterije. Adapter se priključi z 2,1 mm vtikačem (+ srednji, - zunanji), ki ga damo v vtičnico na plošči. Priključne kable (9V) baterije vstavimo v Vin (+) in GND (-) priključka na plošči.

Plošča deluje na zunanji napetosti med 6 in 20 V. Če je napetost pod 7V, lahko 5V priključek daje manj od 5V in plošča lahko postane nestabilna. Če pa se uporablja napetost večja od 12 V, lahko pride do pregrevanja napetostnega regulatorja, kar lahko poškoduje ploščo. Priporočena napetost je torej med 7V in 12V.

[12] The bootloader is a small piece of software that allows uploading of sketches onto the Arduino board. It comes preprogrammed on the microcontrollers on Arduino boards. Microcontrollers are usually programmed through a programmer unless you have a piece of firmware in your microcontroller that allows installing new firmware without the need of an external programmer. This is called a bootloader.

Napetostni priključki so torej:

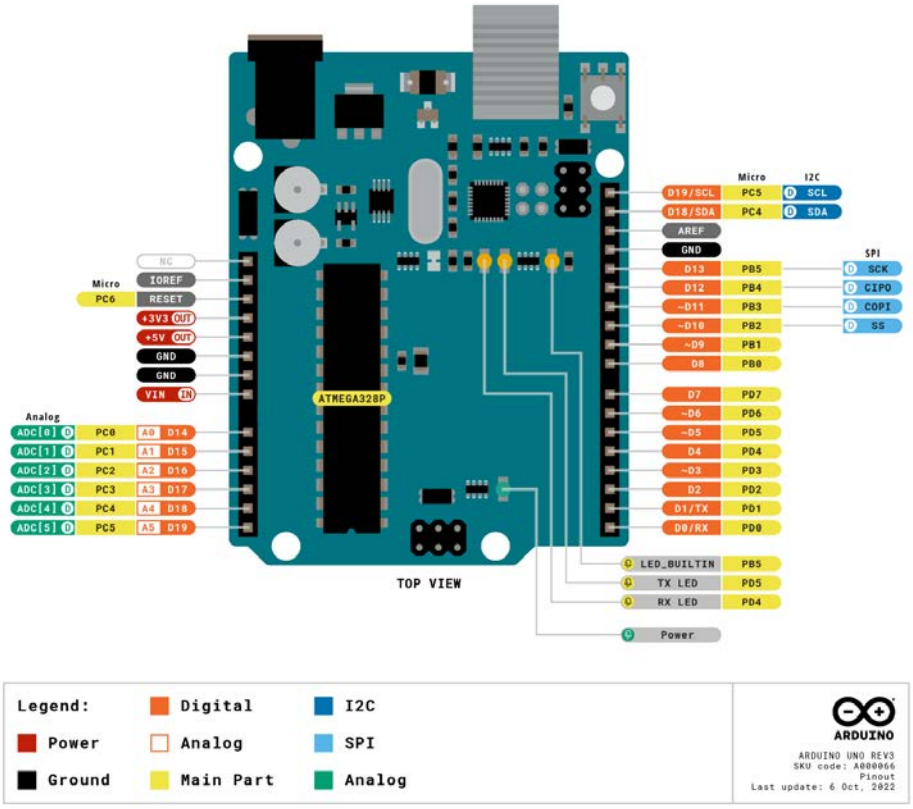
- Vin – pri uporabi zunanjega vira napetosti.
- 5V – namenjeno za reguliran izhod 5V (20 mA).
- 3V3 – reguliran izhod 3,3V (50 mA).
- GND – priključek (-) oz. ničla.
- IOREF – zagotavlja referenčno napetost, s katero mikro-procesor deluje. Vmesniki (nadgradne plošče) lahko berejo IOREF napetost in izberejo ustrezen vir napetosti ali omogočajo napetostnim pretvornikom na izhodih delo s 5V ali 3,3 V.

4.4 Spomin

ATmega328 ima 32 KB flash spomina (0.5 KB porabi zagonski nalagalnik - bootloader). Ima tudi 2 KB SRAM in and 1 KB of EEPROM. V EEPROM lahko beremo in pišemo z EEPROM library.

4.5 Razpored in nekatere specifikacije priključkov

Razpored priključkov med Arduinom in vrati ATmega 228P je prikazan na povezavi razpored priključkov. Razpored za procesorje Atmega8, 168, and 328 je enak.



Lokacije priključkov za Arduino UNO so prikazane na zgornji shemi. Prav tako so vse jasno označene na sami plošči.

Vsakega od 14-ih digitalnih priključkov na UNO plošči lahko uporabimo kot vhod ali izhod z uporabo funkcij pinMode(), digitalWrite(), in digitalRead(). Priključki delujejo na 5V. vsak od njih lahko zagotavlja ali sprejema 20 mA kot priporočen delovni pogoj. Prav tako imajo notranje "pull-up" upore (ti so v normalnem stanju izključeni) z 20-50 kΩ. Maksimalni tok je 40mA in ne sme biti presežen, da se izognemo trajni poškodbi mikro-krmilnika

Dodatno imajo nekateri priključki specializirane funkcije:

- Serijski: 0 (RX) in 1 (TX). Uporabljata se za sprejem (RX) in posredovanje (TX) TTL serijskih podatkov. Ta priključka sta povezana z odgovarjajočimi priključki ATmega8U2 vezja (USB-to-TTL Serial chip).
- Zunanje prekinitve: 2 in 3. Ta dva priključka lahko oblikujemo za sprožanje prekinitve na nizki (low) vrednosti, na dvigu ali padcu signala (a rising or falling edge) ali spremembi vrednosti. Za več detajlov glej funkcijo attachInterrupt().
- PWM: 3, 5, 6, 9, 10 in 11. Zagotavljajo 8-bitni PWM izhod s funkcijo analogWrite().
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ti priključki podpirajo SPI komunikacijo z uporabo SPI knjižnico.
- LED: 13. Vgrajen LED je na priključku 13. Kadar je na priključku vrednost HIGH, bo LED prižgan, kadar pa je vrednost LOW, je LED ugasnjen.
- TWI: A4 ali SDA in A5 ali SCL priključek. To je podpora TWI komunikaciji z uporabo Wire knjižnice.

Uno ima 6 analognih vhodov označenih z A0 do A5, vsak od njih pa zagotavlja 10-bitno ločljivost (resolucijo) oz. 2^{10} različnih vrednosti. Standardno je razpon napetosti od 0 (ground) do 5V. Možno pa je zgornjo mejo spreminjati z uporabo priključka AREF in funkcije analogReference().

Arduino plošče imajo torej večkanalni (UNO – 6-kanalov) 10-bitni A/D (analogno digitalni) pretvornik, tudi ADC[13]. Pri zgornji meji 5V in 1024 vrednosti je ločljivost 5V/1024 = 0,0049V = 4,9mV. To velja za večino klasičnih plošč, nekatere pa imajo 12-bitni A/D pretvornik.

Poleg ločljivost je pomembna tudi hitrost vzorčenja (sampling rate). UNO ima takt 16 MHz,

[13] ADC – analog digital converter

ADC ura pa je nastavljena na 16 MHz/128 = 125 kHz. Vsaka konverzija v AVR pa vzame 13 ADC taktov. Zato je maksimalna frekvenca vzorčenja 125 kHz/13 = 9615 Hz.
Dejanska hitrost vzorčenja v naši aplikaciji pa je odvisna tudi od intervala med naslednjimi klici na pretvorbo. Denimo, ko je signal prebran in poslan po serijskih vratih, dobimo dodatne zakasnitve, ki se povečujejo s padajočo hitrostjo signala (baud rate[14]) na vratih. Logično – manjša hitrost pomeni, da bo čas za pošiljanje enake količine podatkov (znakov) daljši, posledično pa bo kasnejši tudi klic na ADC pretvorbo.
Preostali priključki so:

- AREF. Referenčna napetost za analogne vhode. Uporablja se z analogReference().
- Reset. Ponastavljanje mikro-krmilnika je možno, če je vrednost LOW. Tipično se uporablja, da dodamo reset na razširitvene plošče, ki blokirajo reset na osnovni plošči.

4.6 Komunikacija

Arduino Uno ima na razpolago več različnih možnosti za komunikacijo z računalnikom, drugo Arduino ploščo ali drugimi mikro-krmilniki. ATmega328 zagotavlja UART TTL (5V) serijsko komunikacijo, ki je na razpolago na digitalnih priključkih 0 (RX) in 1 (TX). ATmega16U2 na plošči preusmeri to serijsko komunikacijo na USB[15] in deluje kot virtualna COM vrata za programje na računalniku. Strojno programska oprema 16U2 uporablja standardne USB COM gonilnike in ni potrebe po zunanjem gonilniku. Vendar je na Windowsih potrebna .inf datoteka. Arduino Software (IDE) vsebuje serijski prikazovalnik, ki dovoljuje prenos enostavnih tekstualnih podatkov v in iz plošče. RX in TX LEDa na plošči utripata ob prenosu podatkov preko USB-to-serial vezja in USB povezave na računalnik. To pa ne velja za serijsko komunikacijo preko priključkov 0 in 1.
SoftwareSerial library (programska knjižnica za serijsko komunikacijo) dovoljuje serijsko komunikacijo na kateremkoli od digitalnih priključkov UNO plošče.
ATmega328 podpira tudi I2C (TWI) in SPI komunikacijo. Arduino Software (IDE) vsebuje tudi Wire knjižnico za enostavno uporabo I2C vodila; detajli so v dokumentaciji. Za SPI komunikacijo uporabljamo SPI library oz. SPI knjižnico.

[14] baud rate je frekvenca, s katero podatki potujejo po komunikacijskem kanalu. Uporablja se zlasti v zvezi s serijsko komunikacijo in pomeni število bitov v sekundi. Torej 9600 baud = 9600 bit/s.
[15] USB – universal serial bus

4.7 Avtomatsko (programsko) ponastavljanje

Arduino UNO plošča ne zahteva, da fizično pritisnemo na reset gumb pred nalaganjem novega programa. Načrtovan je tako, da dovoljuje ponastavljanje s programom, ki je operative na priključenem računalniku. Ena od strojnih kontrolnih linij poteka (DTR) ATmega8U2/16U2 je povezana na reset linijo ATmega328 preko 100 nF kondenzatorja. Kadar je ta linija privzeta (low), bo na reset liniji dovolj dolg padec napetosti za ponastavljanje vezja. Arduino Software (IDE) uporablja to možnost za nalaganje kode z enostavnim klikom upload (naloži) gumba v, ki se nahaja na orodnem traku. To sicer pomeni da ima zagonski nalagalnik krajšo prekinitev, ker je lahko ničlanje DTR dobro koordinirano z začetkom nalaganja.

Ta postavitev ima druge posledice. Ko je Uno povezan z računalnikom z operacijskim sistemom Mac OS X ali Linux, se ponastavi vsakič, ko je z njim vzpostavljena povezava iz programske opreme (prek USB-ja). Naslednje pol sekunde ali več se na Uno izvaja zagonski nalagalnik. Medtem ko je programiran tako, da ignorira napačno oblikovane podatke (tj. vse razen nalaganja nove kode), bo prestregel prvih nekaj bajtov podatkov, poslanih na ploščo po vzpostavitvi povezave. Če skica, ki se izvaja na plošči, prejme enkratno konfiguracijo ali druge podatke, ko se prvič zažene, se prepričajmo, da programska oprema, s katero komunicira, počaka sekundo po odprtju povezave in preden pošlje te podatke.

Plošča Uno vsebuje vodnik (bakreno sled), ki jo lahko izrežemo, da onemogočimo samodejno ponastavitev. Spajkalna priključka na obeh straneh prekinjene sledi lahko kasneje premostimo s spajkanjem in s tem ponovno omogočimo ponastavljanje. To je označeno z "RESET-EN". Lahko pa onemogočimo samodejno ponastavitev tako, da priključimo upor 110Ω na priključka 5V in RESET; za podrobnosti je na razpolago povezava this forum thread.

4.8 Revizije

Revizija 3 (sedanja različica) UNO plošče ima naslednje nove lastnosti:

- Glede na razpored 1.0: dodana sta SDA in SCL priključka, ki sta locirana blizu AREF. Še dva nova priključka sta postavljena v bližini RESET. IOREF dovoljuje adaptacijo nadgradnih plošč na napetost, ki jo zagotavlja plošča. V prihodnosti bodo nadgradne plošče kompatibilne s ploščami, ki uporabljajo AVR in delujejo na 5V in tudi z

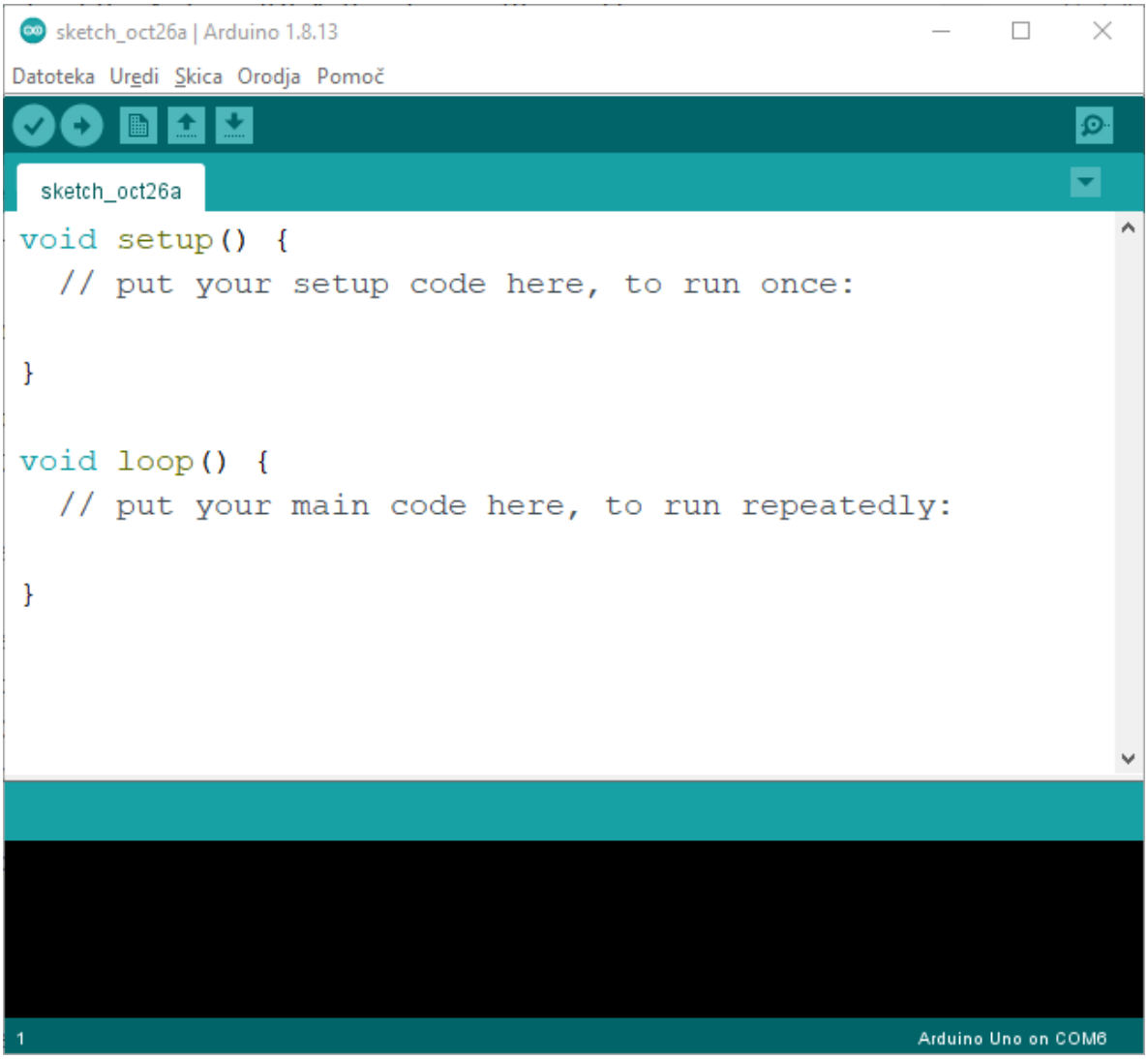
Arduino Due, ki deluje na 3,3V. Drugi priključek je slep in rezerviran za bodoče namene.

- Močnejše RESET vezje.
- Atmega 16U2 nadomešča 8U2.

5 PROGRAMSKA OPREMA - PREGLED

Trenutno se uporablja programsko okolje Arduino IDE 1, ki je klasični off-line programski urejevalnik (editor). Pravkar pa so že razvili izboljšano verzijo IDE 2. Prav tako se uporablja CLI ukazno orodje. Možno je uporabljati Arduino Cloud, da konfiguriramo, programiramo in povežemo naprave s Arduino IoT[16] Cloud servisom.

Integrirano razvojno okolje Arduino - ali programska oprema Arduino (IDE) - vsebuje urejevalnik besedila za pisanje kode, območje za sporočila, besedilno konzolo, orodno vrstico z gumbi za običajne funkcije in vrsto menijev. Povezuje se s strojno opremo Arduino za nalaganje programov in komunikacijo z njimi. Za delovni jezik lahko izberemo slovenščino.



[16] IoT – Internet of Things

5.1 Pisanje skic (sketch)

Programi, napisani s programsko opremo Arduino (IDE), se imenujejo skice, v angleški verziji sketches. Te skice so napisane v urejevalniku besedil in shranjene s pripono datoteke .ino.

Ime skice je je standardno sketch_oct26a (mesec in dan nastanka), več skic pa potem a, b, c, d, … Urejevalnik ima funkcije <ctrl C> <ctrl V> (kopiraj prilepi) in iskanje/zamenjavo besedila. Območje za sporočila daje povratne informacije med shranjevanjem in nalaganjem skic ter prikazuje tudi napake. Konzola prikazuje izpis besedila, ki ga ustvari Arduino (IDE), vključno s popolnimi sporočili o napakah in drugimi informacijami. V spodnjem desnem kotu okna sta prikazana konfigurirana plošča in oznaka serijskih vrat. Gumbi orodne vrstice vam omogočajo preverjanje in nalaganje programov, ustvarjanje, odpiranje in shranjevanje skic ter odpiranje serijskega monitorja.

 Gumb preveri (Verify) preveri napisano kodo za napake pri prevajanju.

 Gumb naloži (Upload) prevede našo kodo (program) in ga naloži na konfigurirano ploščo. Za detajle glej link uploading.

Opomba: Če za našo ploščo uporabljamo zunanji programator, držimo <Shift> tipko na računalniku in potem kliknemo gumb naloži. Tekst se bo spremenil v “Naloži z uporabo programatorja”

 Gumb nova (New file) ustvari novo skico.

 Gumb odpri (Open file) vklopi meni vseh skic v naši skicirki (sketchbook). Ko kliknemo na izbrano skico, se odpre novo okno s kodo te skice.

Opomba: zaradi hrošča v Javi ta meni ne prevrti navzdol. Če potrebujemo skico na dnu liste raje uporabimo **File | Sketchbook** (Datoteka | Skicirka) meni.

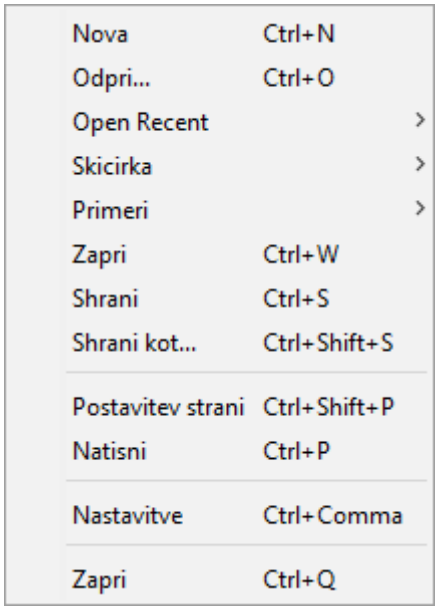
 Gumb shrani (Save file) shrani našo skico.

 Gumb serijski vmesnik (Serial Monitor) odpre serijski vmesnik (serial monitor).

Dodatne ukaze najdemo v 5-ih menijih: Datoteka, Uredi, Skica, Orodja, Pomoč (**File**, **Edit**, **Sketch**, **Tools**, **Help)**. Ti menuji so kontekstno senzitivni, kar pomeni, da so dosegljivi le ukazi pomembni za trenutno izvajanje dela.

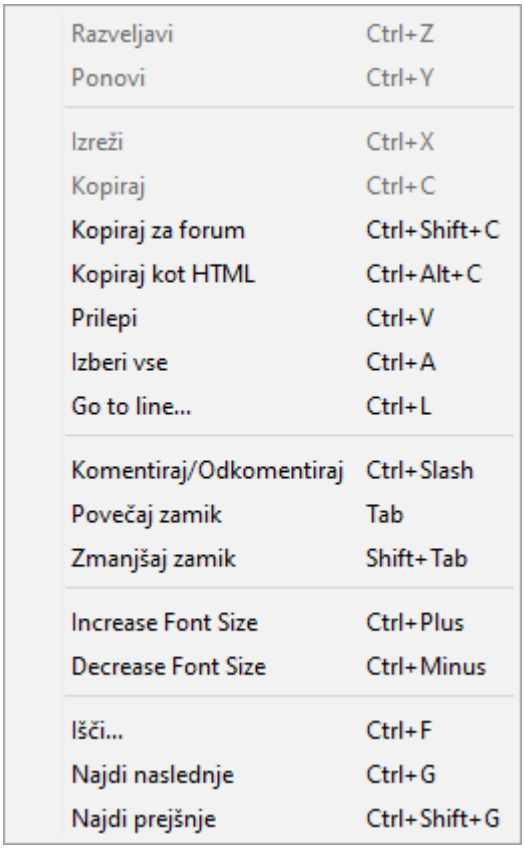
5.2 File (Datoteka)

- *New/Nova* kreira novo pojavno okno urejevalnika z že pripravljeno minimalno strukturo skice.
- *Open/Odpri* naloži skico z iskanjem po računalniških diskih in mapah.
- *Open Recent (odpri nedavne)* daje kratko listo najnovejših skic, ki so pripravljene za odpiranje.
- *Sketchbook/skicirka* pokaže skice v mapni strukturi skicirke, klik na katerokoli ime odpre skico v novem oknu urejevalnika.
- *Examples/Primeri* v tem ukazu se prikažejo vsi vprimeri, ki so na razpolago v Arduino Software (IDE) ali v knjižnici. Primeri so strukturirani v drevesno strukturo, da jih lažje poiščemo po temah.
- *Close/Zapri* zapre okno Arduino Software IDE, iz katerega smo kliknili.
- *Save/Shrani* shrani skico s trenutnim imenom. Če ime ni bilo definirano prej preidemo na okno Save as.../Shrani kot…
- *Save as.../Shrani kot…* omogoča, da shranimo skico z drugim imenom.



- *Page Setup/Postavitev strani* pokaže okno za postavitev strani z namenom tiskanja.
- *Print/Natisni* pošlje trenutno skico v printer v skladu z nastavitvami, definiranimi s *Page Setup/Postavitev strani*
- *Preferences/Nastavitve* odpre okno z nastavitvami, kjer lahko prilagodimo nekatere nastavitve IDE, tudi jezik, ki ga IDE vmesnik uporablja.
- *Quit/Zapri IDE* zapre vsa IDE okna. Skice, ki smo jih zaprli na ta način se avtomatsko odprejo naslednjič, ko poženemo IDE.

5.3 Edit (Uredi)



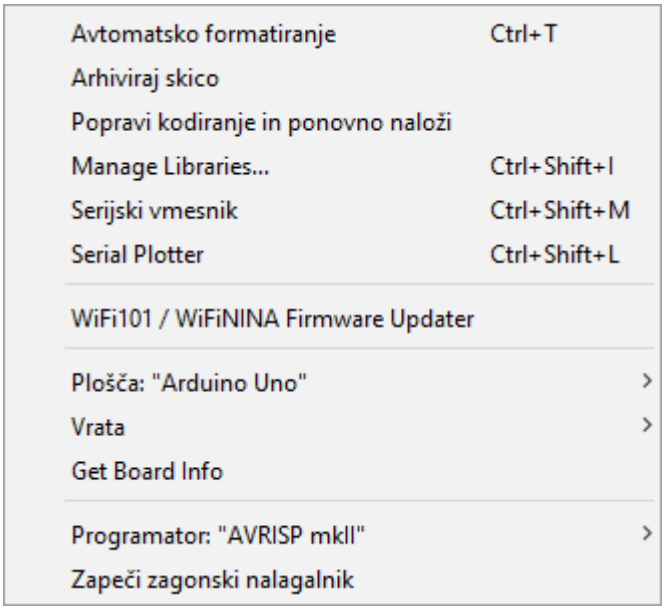
- *Undo/Razveljavi* gre nazaj en ali več korakov, opravljenih med urejanjem.
- *Redo/Ponovi* Tisto kar smo razveljavili, lahko tudi ponovimo s tem ukazom.
- *Cut/Izreži* odstrani izbrani tekst iz urejevalnika in ga prestavi v odložišče – clipboard.
- *Copy/Kopiraj izbrani tekst v urejevalniku in ga prenese v odložišče.*
- *Copy for Forum/Kopiraj za forum* kopira kodo naše skice v odložoišče v obliki, primerno za objavo na forumu skupaj s kontekstno obarvano sintakso.
- *Copy as HTML/Kopiraj kot HTML* kopira kodo v odložišče v obliki HTML, ki je primerna za vgradnjo v omrežne strani.
- *Paste/Prilepi* položi vsebino odložišča na trenutni lokaciji kazalca v urejevalniku
- *Select All/Izberi vse* izbere in označi celotno vsebino urejevalnika.
- *Comment/Uncomment|Komentiraj/Odkomentiraj* postavi ali odstrani oznako za komentar na začetek vsake izbrane vrstice.
- *Increase/Decrease Indent|Povečaj/Zmanjšaj zamik* doda ali odvzame prostor na začetku vsake od izbranih vrstic; s tem se tekst pomika v desno ali v levo.
- *Find/Išči…* odpre okno *Najdi in zamenjaj* kjer lahko specificiramo tekst za iskanje znotraj trenutne skice po več kriterijih.
- *Find Next/Najdi naslednje* označi naslednjo pojavitev (če obstaja) specificiranega teksta relativno na lego kazalca.
- *Find Previous/Najdi prejšnje* označi prejšnjo pojavitev (če obstaja) specificiranega teksta relativno na lego kazalca.

5.4 Sketch (Skica)

- *Verify/Compile (Preveri/Prevedi) pregleda skico in odkrije morebitne napake pri prevajanju; poroča o porabi pomnilnika za kodo in spremenljivke v konzolnem delu (črni del spodaj) urejevalnika.*
- *Upload/Naloži* prevede in naloži binarno datoteko na konfigurirano ploščo prek konfiguriranih vrat.
- *Upload Using Programmer/Naloži s programatorjem* prepiše zagonski nalagalnik na plošči. Za obnovitev le tega moramo uporabiti Orodja > Zapiši zagonski nalagalnik in skico lahko znova naložimo na serijska vrata USB. Vendar zgornji ukaz omogoča, da za svojo skico uporabimo celotno zmogljivost flash pomnilnika. Ta ukaz NE ponastavi varovalk. To lahko naredimo z ukazom Tools/Orodja -> Burn Bootloader/Zapiši zagonski nalagalnik.
- *Export Compiled Binary (prenesi prevedeno binarno datoteko)* shrani .hex datoteko, ki jo lahko uporabimo za arhiv ali pošljemo na ploščo z drugimi orodji.
- *Show Sketch Folder/Pokaži mapo skice* odpre mapo trenutne skice.
- *Include Library (vključi knjižnico)* doda knjižnico(e) skici s stavki #include na začetku kode. Za več detajlov glej link libraries. Dodatno lahko s tem ukazom dosegamo t.i. “Library Manager” in uvozimo nove knjižnice iz .zip datotek.
- *Add file/Dodaj datoteko...* doda pomožno datoteko skici (kopirana bo s trenutne lokacije). Datoteka je shranjena v `data` podmapi skice in namenjena dokumentaciji. Vsebina v `data` mapi se ne prevaja, tako ne postane del programa.

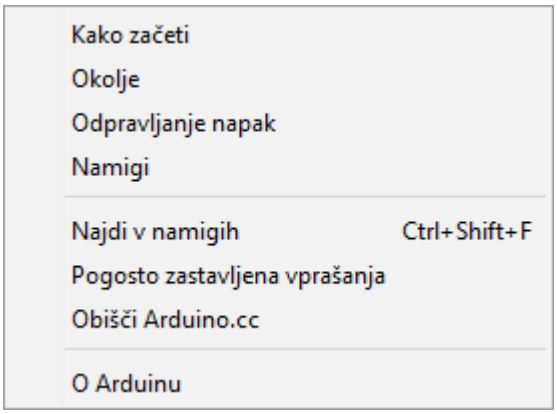
5.5 Tools (Orodja)

- *Auto Format/Avtomatsko formatiranje* lepo oblikuje kodo: tj. zamakne jo tako, da se levi in desni zaviti oklepaji poravnajo in da so stavki znotraj zavitih oklepajev bolj zamaknjeni.
- *Archive Sketch/Arhiviraj skico* arhivira kopijo trenutne skice v formatu .zip. Arhiv je v isti mapi kot skica.Archives a copy of the current sketch in .zip format.



- *Fix Encoding & Reload/Popravi kodiranje in ponovno naloži* Odpravi možne razlike med kodiranjem v skladu s kodno tabelo znakov v urejevalniku in kodnimi tabelami znakov drugih sistemov (npr. operacijskega Sistema).
- *Serial Monitor/Serijski vmesnik* odpre okno serijskega vmesnika in sproži izmenjavo podatkov s katero koli povezano ploščo na trenutno izbranih vratih. To običajno ponastavi ploščo, če plošča podpira ponastavitev prek odpiranja serijskih vrat.
- *Board/Plošča* Izbere ploščo, ki jo uporabljamo. V povezavi so opisi različnih plošč.
- *Port/Vrata* Ta meni vsebuje vse serijske naprave (realne ali virtualne) na računalniku. Osveži se vsakič, ko odpremo meni orodij najvišje ravni.
- *Programmer/Programator* za izbiro strojnega programatorja pri programiranju plošče ali čipa brez uporabe vgrajene serijske povezave USB. Običajno tega ne potrebujemo, če pa zapečete zagonski nalagalnik na nov mikro-krmilnik, je pa to nujno.
- *Burn Bootloader/Zapeči zagonski nalagalnik* elementi v tem meniju omogočajo, da zapečemo (zapišemo) zagonski nalagalnik na mikro-krmilnik na plošči Arduino. Pri običajni uporabi plošče tega ne rabimo (zagonski nalagalnik je že na flash spominu). Je pa koristno, če kupimo nov mikro-krmilnik ATmega (ki je običajno brez zagonskega nalagalnika). Prepričati se moramo, da smo v Boards/Plošča meniju izbrali pravo ploščo in šele potem zapečemo zagonski nalagalnik na ciljno ploščo. Ta ukaz nastavi tudi prave varovalke (SW).

5.6 Help (Pomoč)



Tu najdemo enostaven dostop do številnih dokumentov, ki so priloženi programski opremi Arduino (IDE). Imamo dostop do Getting Started (Kako začeti), Reference (Referenčni priročnik), vodiča za IDE in drugih dokumentov lokalno, brez internetne povezave. Dokumenti so lokalna kopija spletnih in lahko vodijo nazaj na Arduino spletno stran.

- *Find in Reference/Najdi v namigih* je edina interaktivna funkcija v tem meniju: direktno izbere relevantno stran v lokalni kopiji Referenčnega priročnika (Reference) za funkcijo ali ukaz na kazalcu.

5.7 Sketchbook (Skicirka)

Programska oprema Arduino oz. Arduino Software (IDE) uporablja koncept skicirke, ki je standardno mesto za shranjevanje programov (ali skic). Skice v skicirki lahko odpremo v meniju Datoteka > Skicirka **(File > Sketchbook)** ali z gumbom Odpri v orodni vrstici. Ko prvič zaženemo programsko opremo Arduino, bo samodejno ustvarila imenik za skicirko. Lokacijo skicirke si lahko ogledamo ali spremenimo v pogovornem oknu Nastavitve (Preferences).
Od različice IDE 1.0 se datoteke shranjujejo s pripono .ino. **Starejše** različice uporabljajo pripono .pde. Še vedno lahko odpremo datoteke s pripono.pde v različicah 1.0 in novejših, programska oprema bo samodejno preimenovala pripono v .ino.

5.8 Zavihki (Tabs), več datotek in prevajanje

Omogoča upravljanje skic z več kot eno datoteko (od katerih je vsaka prikazana v svojem zavihku). To so lahko običajne kodne datoteke Arduino (brez vidne pripone), datoteke C (pripona .c), datoteke C++ (pripona .cpp) ali datoteke vodilne (header) datoteke (.h).
Pred prevajanjem skice so vse običajne datoteke Arduino kode te skice (.ino, .pde) združene v eno datoteko po vrstnem redu, v katerem so prikazani zavihki. Druge vrste datotek ostanejo takšne, kot so.

New Tab/Nov zavihek je na razpoplago z gumbom  na desni strani v pasici zavihkov ali

<Ctrl><Shift> N

5.9 Uploading (Nalaganje)

Pred nalaganjem skice je treba izbrati prave elemente v menijih Orodja > Plošča (Tools > Board) in Orodja > Vrata (Tools > Port). Plošče so na kratko opisane kasneje v tem tekstu. V Macu so serijska vrata zelo verjetno opisane kot sledi:

- /dev/tty.usbmodem241 (za Uno ali Mega2560 ali Leonardo) ali
- /dev/tty.usbserial-1B1 (za Duemilanove ali starejšo ploščo USB) ali
- /dev/tty .USA19QW1b1P1.1 (za serijsko ploščo, povezano z vmesnikom Keyspan USB-to-Serial).

V sistemu Windows so serijska vrata običajno COM1 ali COM2 (za serijsko ploščo) ali COM4, COM5, COM7 ali višje (za USB ploščo). Ta informacija je na razpolago v upravitelju naprav Windows, v razdelku Vrata (COM in LPT) in krmilniki USB. V Linuxu imamo običajno /dev/ttyACMx, /dev/ttyUSBx ali podobno. Ko izberemo prava serijska vrata in ploščo, pritisnemo gumb za nalaganje v orodni vrstici ali izberemo Skica > Naloži. Sedanje plošče Arduino se bodo samodejno ponastavile in začele nalaganje. Pri starejših ploščah (pred Diecimilo), ki nimajo samodejne ponastavitve, moramo gumb za ponastavitev na plošči tik pred začetkom nalaganja. Na večini plošč med nalaganjem skice vidimo, da lučki LED RX in TX utripata. Programska oprema Arduino (IDE) prikaže sporočilo, ko je nalaganje končano, ali prikaže napako.

Pri nalaganju skice uporabljamo zagonski nalagalnik Arduino, majhen program, ki je vnaprej naložen (zapečen) v mikro-krmilnik na naši plošči. Ta program omogoča nalaganje kode brez uporabe dodatne strojne opreme. Zagonski nalagalnik je aktiven nekaj sekund, ko se plošča ponastavi; potem pa zažene skico, ki je bila zadnja naložena v mikro-krmilnik. Ko se zagonski nalagalnik aktivira utripa LED na plošči (priključek 13), plošča pa se ponastavi.

5.10 Libraries (Knjižnice)

Knjižnice nudijo dodatno funkcionalnost za uporabo v skicah, bodisi da je to delo s strojno opremo ali manipuliranje s podatki. Torej, če želimo knjižnico uporabiti v skici, jo izberemo v meniju ˙Skica > Include Library[17] (Sketch > Import Library). Tako bo vstavljen eden ali

[17] Ni slovenskega prevoda v sicer slovenskem meniju.

več stavkov #include na vrh skice in knjižnica prevedena v trenutno skico. Ker so knjižnice naložene na ploščo skupaj s skico, povečajo porabo prostora, ki ga ta zavzame. Če v skici knjižnice niso več potrebne, stavke #include teh knjižnic izbrišemo z vrha kode.
V referenci je seznam knjižnic. Nekatere knjižnice so vključene v programsko opremo Arduino. Druge lahko prenesemo iz različnih virov ali prek upravitelja knjižnice. Od različice 1.0.5 IDE lahko uvozimo knjižnico iz datoteke zip in jo uporabimo v odprti skici. Namestimo lahko tudi knjižnice drugega izvora. Če želimo napisati svojo knjižnico, so na razpolago kratka navodila.

5.11 Strojna oprema drugih proizvajalcev

Podporo za strojno opremo drugih proizvajalcev lahko dodamo v mapo strojne opreme, ki se nahaja v mapi skicirke. Tam nameščene platforme lahko vključujejo definicije plošč (ki se prikažejo v meniju plošče), osnovne knjižnice, zagonske nalagalnike in definicije programatorja. Za namestitev ustvarimo mapo strojne opreme, nato pa razpakiramo platformo drugega proizvajalca v njeno lastno podmapo. (Ne smemo uporabljati "arduino" kot ime podmape, ker bomo sicer razveljavili vgrajeno platformo Arduino.) Za odnamestitev take platforme preprosto izbrišite njeno mapo. Za podrobnosti o ustvarjanju kompletov strojne opreme drugih proizvajalcev je primeren ogled specifikacije platforme Arduino.

5.12 Serijski vmesnik

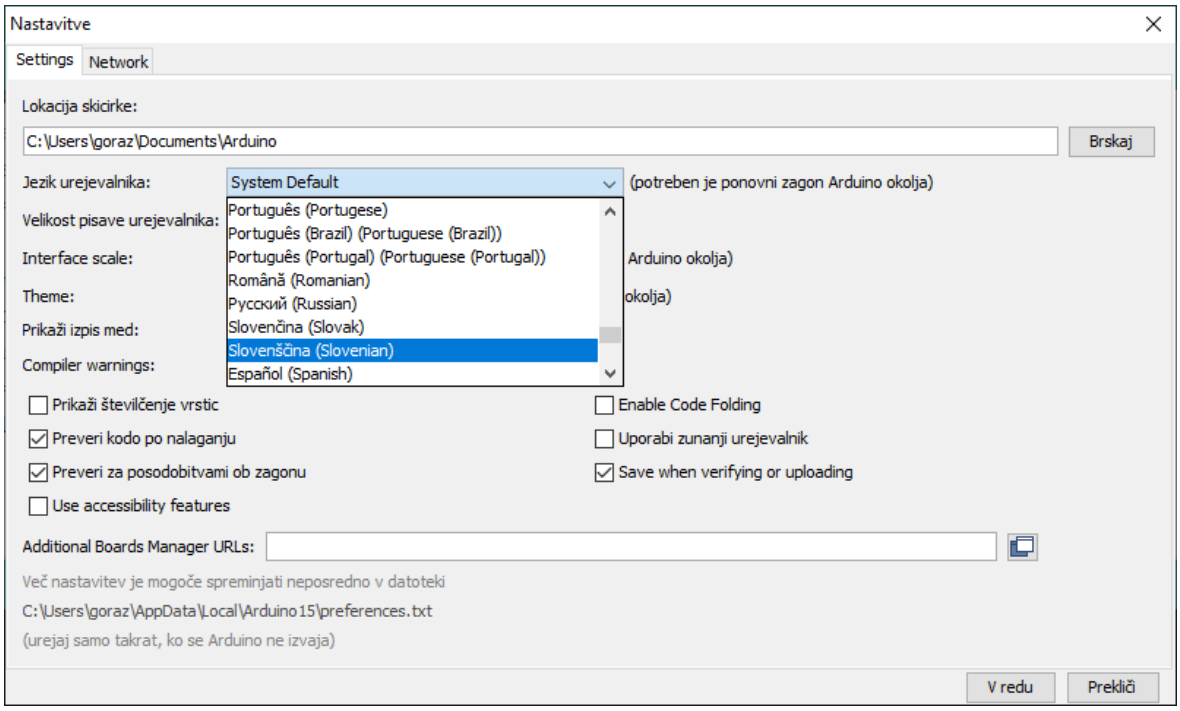
Serijski vmesnik služi prikazovanju podatkov (na računalniku), ki jih posreduje Arduino plošča preko USB ali serijskega priključka. Da bi poslali podatke na ploščo v serijskem vmesniku vnesemo besedilo in kliknemo na gumb Pošlji ali pritisnemo <ENTER>. V spustnem meniju izberemo hitrost prenosa, ki se ujema s hitrostjo, posredovano Serial.begin v naši skici. Upoštevajmo, da se v sistemih Windows, Mac ali Linux plošča ponastavi (znova zažene našo skico), ko se povežemo s serijskim vmesnikom. Upoštevajmo, da serijski vmesnik ne obdeluje kontrolnih znakov; če naša skica potrebuje popolno upravljanje serijske komunikacije s kontrolnimi znaki, lahko uporabimo zunanji terminalski program in ga povežemo s COM vrati, ki so dodeljena naši plošči Arduino.

5.13 Preference (Nastavitve)

Nekatere preference lahko nastavimo v dialogu Nastavitve (pod **Arduino** menijem na Mac-

u, ali **File/Datoteka** na Windows'ih in Linux-u). Preostale najdemo v datoteki nastavitev, katere lokacija je podana v dialogu nastavitev (na dnu)

5.14 Jezik urejevalnika (Language Support)



Od verzije 1.0.1 , je bil Arduino Software (IDE) preveden v mnoge jezike (več kot 30). Standardno se IDE naloži v jeziku, ki je bil izbran v operacijskem sistemu (Na Windows-ih in običajno na Linux-u je to določeno z nastavitvijo lokalizacije, ki določa valuto in format datuma in ure in ne z jezikom, v katere je prikazan operacijski sistem).

Če pa želimo sami spremeniti jezik IDE, poženemo Arduino Software (IDE) in odpremo okno **Preference/Nastavitve**. Ob **Editor Language/Jezik urejevalnika** je spustni meni z vsemi podprtimi jeziki. Po izboru jezika iz menija je potreben ponoven zagon IDE. Če IDE jezika ne podpira, se nastavi v angleščino-

Programje lahko vrnemo v standardno jezikovno nastavitev s klikom na System Default. Če pride do nove lokalizacije operacijskega sistema, je potrebno Arduino Software (IDE) ponovno startati, da se vzpostavi prednastavljen jezik.

5.15 Boards (Plošče)

Izbira plošče ima dva učinka: nastavitev parametrov (npr. hitrost procesorja in hitrost prenosa), ki se uporabljajo pri prevajanju in nalaganju skic; in nastavitve datoteke in

varovalk, ki jih uporablja ukaz Burn Bootloader/Zapeči zagonski nalagalnik. Nekatere definicije plošč se razlikujejo samo v slednjem. Torej, tudi, če smo uspešno naložili z določeno izbiro, jo bomo preverili, preden bi zapekli zagonski nalagalnik.

Arduino Software (IDE) vsebuje vgrajeno podporo za plošče iz spodnje liste, ki vse bazirajo na AVR jedru (AVR Core). Boards Manager je vključen v standardno instalacijo in omogoča podporo rastočemu številu novih plošč, ki bazirajo na različnih jedrih, npr. Arduino Due, Arduino Zero, Edison, Galileo itd.

Ime platforme – mikro-krmilnik – delovna frekvenca – možnost ponastavljanja – analogni vhodi – digitalni I/O – PWM
Arduino Yún An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
Arduino Uno An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Diecimila or Duemilanove w/ ATmega168 An ATmega168 running at 16 MHz with auto-reset.
Arduino Nano w/ ATmega328P An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
Arduino Mega 2560 An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Mega An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Mega ADK An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
Arduino Leonardo An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
Arduino Micro An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
Arduino Esplora An ATmega32u4 running at 16 MHz with auto-reset.
Arduino Mini w/ ATmega328P An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
Arduino Ethernet Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

Arduino Fio An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino BT w/ ATmega328P ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board Bluetooth® module, 6 Analog In, 14 Digital I/O and 6 PWM..
LilyPad Arduino USB An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
LilyPad Arduino An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino NG or older w/ ATmega168 An ATmega168 running at 16 MHz*without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
Arduino Robot Control An ATmega328P running at 16 MHz with auto-reset.
Arduino Robot Motor An ATmega328P running at 16 MHz with auto-reset.
Arduino Gemma An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

6 OSNOVNI UKAZI PROGRAMSKEGA JEZIKA ARDUINO

Na koncu pa še pomembno vprašanje: Kako napišemo program za Arduino?

Arduino podpira jezik, ki ga imenujemo Arduino Programski jezik ali kar Arduino jezik.

Ta jezik je zasnovan na Wiring razvojni platformi, ki je zgrajena na Processing grafični knjižnici in integriranem razvojnem okolju (IDE), tudi p5.js interpretacija Processing. Projekti, ki so zasnovani in zgrajeni na drugih projektih v skladu z odprtokodnim (Open Source) načinom imajo dolgo zgodovino. Tudi Arduino IDE bazira na Processing IDE in Wiring IDE na vrhu..

Ko delamo za Arduino ploščami, običajno uporabljamo Arduino IDE, ki je na razpolago za vse glavne računalniške platforme (macOS, Linux, Windows). Arduino IDE nam da dve stvari: 1) programski urejevalnik z vgrajeno podporo knjižnic in 2) način za enostavno prevajanje in nalaganje Arduino programov (skic) na ploščo, ki je povezana z računalnikom.

Arduino Programski jezik je v osnovi zgrajen na C++. Sicer lahko argumentiramo, da pravzaprav niti ni pravi programski jezik v običajnem pomenu.

Program v napisan v Arduino jeziku se imenuje **sketch/skica**. Skico običajno shranimo s pripono `.ino` (iz Arduino).

Glavna razlika med "normalnim" C ali C++ je da vso svojo kodo ogrnemo v dve glavni funkciji. Lahko jih je sicer več, vendar mora Arduino program imeti najmanj ti dve.

Ena funkcija se imenuje `setup()`in druga `loop()`. Ob izvajanju programa se `setup()` izvaja enkrat na začetku, druga pa se ponavlja ves čas izvajanja programa.

Torej nimamo `main()` funkcije, ki je običajna v C/C++ kot vhodna točka v program. Ko prevedemo našo skico, IDE zagotovi, da je končni rezultat pravilna C++ koda in IDE doda manjkajočo vez.

Vse ostalo je normalna C++ koda in kot je C++ nadgradnja C, bo tudi vsaka C koda veljavna Arduino koda.

Ena od razlik, ki bi nam lahko povzročila težave, je ta, da morajo biti vse te datoteke v isti mapi, čeprav lahko svoj program ustvarimo v več datotekah. Morda bi bila lahko to omejitev, če program postane velik, vendar se bo na tej točki enostavno premakniti na izvorno nastavitev C++.

Del Arduino programskega jezika so vgrajene knjižnice, ki omogočajo enostavno integracijo funkcionalnosti, ki jih zagotavlja Arduino plošča.

V običajnih programskih jezikih je prva naloga narediti program, ki npr. na ekran zapiše

“Hello, world”. V primeru Arduina bo to program ki prižge ali ugasne led. Za to nalogo rabimo funkcije `pinMode()`, `delay()` in `digitalWrite()` skupaj s konstantami `HIGH`, `LOW`, `OUTPUT`. In rezultat:

```
#define LED_PIN 13

void setup() {
    // Configure pin 13 to be a digital output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // Turn on the LED
    digitalWrite(LED_PIN, HIGH);
    // Wait 1 second (1000 milliseconds)
    delay(1000);
    // Turn off the LED
    digitalWrite(LED_PIN, LOW);
    // Wait 1 second
    delay(1000);
}
```

Ta program prižge led na vratih 13 za 1s in ga nato ugasne za 1s. Program ima zanko, ki se ponavlja, dokler plošče ne izklopimo fizično. Vse to je del programskega jezika Arduino, ki bi ga morda bolje imenovali zbirka ali knjižnica.

6.1 Podpora drugim jezikom

Naj spomnimo, nismo omejeni na uporabo Arduino jezika in IDE za programiranje Arduina. Med drugim obstajajo projekti, ki omogočajo izvajanje kod v drugih jezikih: Node.js s pomočjo projekta Johnny Five Johnny Five, kode Python z uporabo pyserial in kode Go z Gobot-om, vendar je programski jezik Arduino zagotovo tisti, na katerem bazira večina kode in vadnic, saj je to izvorni in kanonični način dela z Arduino ploščami.

6.2 Vgrajene konstante

Arduino ima dve nastavitveni dve konstanti, ki ju lahko uporabimo:

`HIGH` pomeni visoko raven napetosti, ki se lahko razlikuje glede na strojno opremo (>2V na 3,3V ploščah, kot je Arduino Nano, >3V na 5V ploščah, kot je Arduino Uno). `LOW` pomeni je enaka nizki ravni napetosti. Tudi tu je točna vrednost je odvisna od uporabljene plošče.

Nato imamo 3 konstante, ki jih lahko uporabimo v kombinaciji s funkcijo `pinMode()`:

- `INPUT` nastavi priključek kot vhodni priključek
- `OUTPUT` nastavi pin kot izhodni priključek
- `INPUT_PULLUP` nastavi priključek v način notranjega dvižnega upora.

Naslednja konstanta je `LED_BUILTIN`, ki ima številsko vrednost testnega priključka za led in je običajno 13.

Poleg tega imam še C/C++ konstanti `true` in `false`.

6.2.1 Matematične konstante

- `M_PI` π=3.14159265358979323846
- `M_E` e=2.71828182845904523536
- `M_LN10` naravni logaritem števila 10.
- `M_LN2` naravni logaritem števila 2.
- `M_LOG10E` logaritem e z bazo 10.
- `M_LOG2E` logaritem e z bazo 2.
- `M_SQRT2` koren iz 2.
- `NAN` NAN (not a number).

6.3 Vgrajene funkcije

V tem razdelku se bomo v glavnem sklicevali na vgrajene funkcije, ki jih ponuja programski jezik Arduino.

6.3.1 Programski cikel

- `setup()` je funkcija ki se izvaja samo ob zagonu programa in v primeru, ko je Arduino izklopljen in se ga potem ponovno aktivira.

- `loop()` je funkcija, ki jo Arduino program vseskozi (ponavlja) kliče dokler je Arduino plošča aktivna.

6.3.2 Vhodno izhodne (I/O) funkcije

Te funkcije pomagajo pri rokovanju z vhodi/izhodi v/iz naprave Arduino.

6.3.2.1 Digitalni I/O

- `digitalRead()` bere vrednost iz digitalnega priključka. Številka priključka je parameter. Funkcija vrne `HIGH` ali `LOW` vrednost.
- `digitalWrite()` zapiše `HIGH` ali `LOW` na digitalni izhodni priključek. Parametra sta številka priključka in digitalna vrednost (`HIGH` ali `LOW`).
- `pinMode()` nastavi priključek kot vhod ali izhod (input ali output). Parametra sta številka priključka in `INPUT` ali `OUTPUT`.
- `pulseIn()` na danem priključku bere digitalni pulz iz `LOW` v `HIGH` in nato spet v `LOW`, ali iz `HIGH` v `LOW` in nato spet v `HIGH` na danem priključku. Če je drugi parameter `HIGH` bo funkcija čakala, da vrednost na priključku iz `LOW` preide v `HIGH`, nato začne meriti čas in čaka do takrat ko signal pade na `LOW` in ustavi štoparico. Vrne dolžino pulza v µs ali 0, če v danem času ni bil izvršen kompleten pulz.
- `pulseInLong()` je enak kot `pulseIn()`, s tem, da daje bolj zanesljive rezultate pri daljših pulzih.
- `shiftIn()` bere byte (8 bitov) podatkov po en bit naenkrat iz podatkovnega priključka, drugi priključek daje takt. Tretji parameter pa je vrstni red branja bitov `MSBFIRST` ali `LSLFIRST`. Lahko rečemo, da je ta funkcija programski pretvornik iz serijskega v paralelni način.
- `shiftOut()` zapiše podatkovni byte po en bit naenkrat na podatkovni priključek. Parametri so še priključek za takt, vrstni red bitov in vrednost. Ta funkcija deluje kot pretvornik iz paralelnega v serijski način.
- `tone()` pošlje na priključek pulzni signal z dano frekvenco in 50% gostoto, ki ga uporabimo, če hočemo, da zvočnik oddaja ton. Specificiramo priključek in frekvenco, lahko pa tudi trajanje.
- `noTone()` ustavi `tone()`, generiran na priključku.

6.3.2.2 Analog I/O

- `analogRead()` bere vrednost iz analognega priključka.
- `analogReference()` konfigurira vrednost, ki je bila uporabljena za največjo vrednost na vhodu analognega vhoda in je prednastavljena na 5V na 5V ploščah in na 3.3V na 3.3V ploščah.
- `analogWrite()` zapiše analogno vrednost na priključek.
- `analogReadResolution()` omogoča, da spremenimo prednastavljeno ločljivost (resolucijo) za `analogRead()`, ki je standardno 10 bitov. Deluje pa samo na določenih Arduino ploščah (Arduino Due, Zero and MKR)
- `analogWriteResolution()` omogoča, da spremenimo prednastavljeno ločljivost (resolucijo) za `analogWrite()`, ki je standardno 10 bitov. Deluje pa samo na določenih Arduino ploščah (Arduino Due, Zero and MKR)

6.3.3 Časovne funkcije

- `delay()` ustvari programski premor, ki traja določeno število ms, kar je specificirano kot parameter.
- `delayMicroseconds()` prekine program za toliko µs, kolikor je specificirano kot parameter.
- `micros()` število µs od začetka programa. Ponastavitev je po ~70 min zaradi prekoračenja.
- `millis()` število ms od začetka programa. Ponastavitev je po ~50 dneh zaradi prekoračenja.

6.3.4 Matematične funkcije

- `abs()`absolutna vrednost
- `constrain()` omeji število, da je v dovoljenem obsegu, glej uporabo
- `map()` pretvori (preslika) število iz enega obsega v drugega, glej uporabo
- `max()` maksimum dveh števil
- `min()`minimum dveh števil
- `pow()` vrednost števila na potenco
- `sq()` kvadrat števila

- `sqrt()` kvadratni koren števila
- `cos()` kosinus kota
- `sin()` sinus kota
- `tan()` tangens kota

Opomba: vgrajenih matematičnih funkcij je sicer več, dokumentirane so v naslednjem linku.

6.3.5 Delo z alfanumeričnimi znaki

- `isAlpha()` preveri ali je znak alpha (črka)
- `isAlphaNumeric()` preveri ali je znak alfanumeričen (črka ali številka)
- `isAscii()` preveri ali je znak ASCII znak
- `isControl()` preveri ali je znak krmilni znak
- `isDigit()` preveri ali je znak število
- `isGraph()` preveri ali je znak natisljiv ASCII znak in ima vsebino (npr. ni presledek)
- `isHexadecimalDigit()` preveri ali je znak heksadecimalna številka (A-F 0-9)
- `isLowerCase()` preveri ali je znak mala črka
- `isPrintable()` preveri ali je znak natisljiv ASCII znak
- `isPunct()` preveri ali je znak ločilo (vejica, podpičje, narekovaj itd.)
- `isSpace()` preveri ali je znak presledek, , form feed `\f`, newline `\n`, carriage return `\r`, horizontalni tabulator `\t` alo vertikalni tabulator `\v`.
- `isUpperCase()` preveri ali je znak velika črka
- `isWhitespace()` preveri ali he znak presledek ali horizontalni tabulator `\t`

6.3.6 Generiranje naključnih števil

- `random()` generira psevdo naključno število
- `randomSeed()` inicializira psevdo naključni generator s poljubnim začetnim številom.

V Arduinu, kot to velja za večino jezikov je nemogoče dobiti dejanska naključna števila. Sekvenca je vselej enaka, tako da je seme trenutni čas ali (v primeru Arduina) lahko beremo vhod iz analognih vrat oz. priključka.

6.3.7 Delo z biti in byti (zlogi)

- `bit()` izračuna vrednost bita (0 = 1, 1 = 2, 2 = 4, 3 = 8…)
- `bitClear()` postavi na 0 clear bit numerične vrednosti (števila). Parametra sta število in vrstni red bita z začetkom na desni
- `bitRead()` bere bit numerične vrednosti (števila). Parametra sta število in vrstni red bita z začetkom na desni
- `bitSet()` nastavi bit števila na 1. Parametra sta število in vrstni red bita z začetkom na desni
- `bitWrite()` zapiše vrednost 1 ali 0 v določen bit danega števila. Parametra sta število, vrstni red bita začenši z desne in vrednost, ki naj jo zapišemo (0 or 1)
- `highByte()` pridobi high-order (najbolj levi) byte (zlog) spremenljivke velikosti beseda (2 byta/zloga)
- `lowByte()` pridobi low-order (najbolj desni) byte (zlog) spremenljivke velikosti beseda (2 byta/zloga)

6.3.8 Prekinitve (Interrupts)

- `noInterrupts()` onemogoči prekinitve
- `interrupts()` ponovno omogoči prekinitve, potem ko so bile onemogočene
- `attachInterrupt()` omogoči, da postane digitalni vhodni priključek prekinitev. Različne plošče imajo različne dovoljene priključke, preveriti je treba uradno dokumentacijo.
- `detachInterrupt()` onemogoči prekinitev, ki je bila prej dovoljena z `attachInterrupt()`

6.4 Strukture

Strukture jezika so dokumentirane v naslednjih linkih in so:

- break
- continue
- do...while
- else
- for

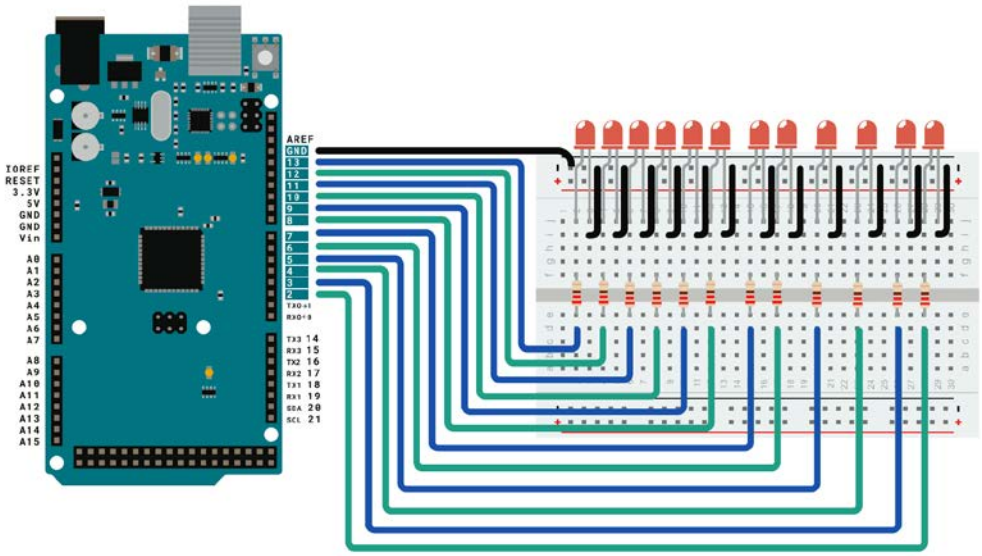
- goto
- if
- return
- switch...case
- while

Celotna specifikacija konstant, operatorjev, podatkovnih tipov spremenljivk, struktur in sistematični pregled ukazov je na razpolago na domači strani Language Reference.

7 NEKAJ PRIMEROV

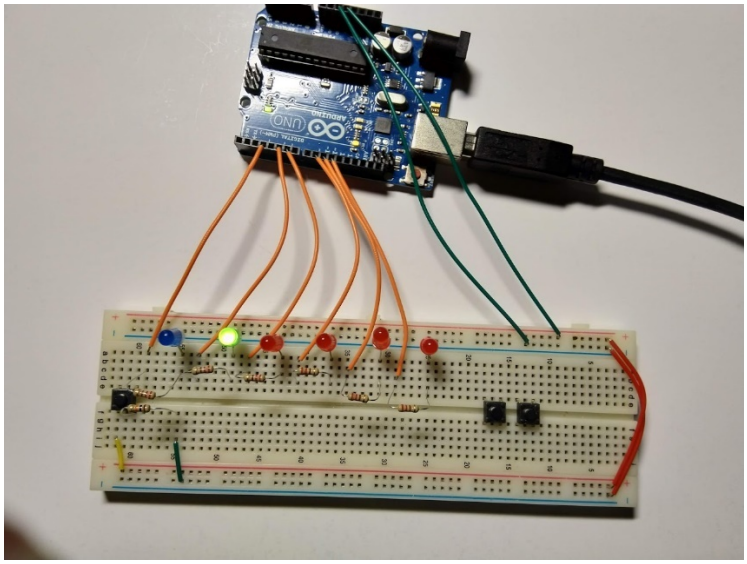
7.1 Analog Write

Originalni program je pripravljen za večjo ploščo Arduino Mega, ki ima 12 priključkov za PWM izhod. Arduino Uno ima takih priključkov samo 6. Torej od strojne opreme poleg Arduino Uno rabimo še 6 ledic, 6 220 Ω uporov, prototipno ploščo in vezice. Vezalna shema je praktično enaka tisti na spodnji sliki (s tem, da imamo samo 6 priključkov).



Program zaporedoma zvezno prižiga in ugaša 6 ledic, kolikor jih lahko uporabimo z analognim PWM v UNO platformi. Najprej definiramo zaporedje pinov, ki so sposobni ustvariti PWM izhod. To pa so {3, 5, 6, 9, 10, 11}.

- Potem te pine nastavimo kot izhodne.
- In v zanki jih zaporedoma posamično prižigamo od 0 do 254 in na enak način ugašamo.



```
/*
  Uno analogWrite() test

  This sketch fades LEDs up and down one at a time on UNO PWM pins.
  Original sketch was written for the Arduino Mega, and will not work on other boards.

  The circuit:
  - LEDs attached from pins 3, 5, 6, 9, 10, 11 to ground.

  created 8 Feb 2009
  by Tom Igoe
  update for UNO 6 Mar 2018 by gh

  Original example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/AnalogWriteMega
*/

  int myPins[] = {3, 5, 6, 9, 10, 11};

void setup() {
  // set PWM pins as outputs:
  for (int i=0; i <= 5; i++) {
    pinMode(myPins[i], OUTPUT);
  }
}

void loop() {
  // iterate over the pins:
for (int i=0; i <= 5; i++) {
    // fade the LED on myPins[i] from off to brightest:
  for (int brightness = 0; brightness < 255; brightness++) {
      analogWrite(myPins[i], brightness);
      delay(5);
    }
    // fade the LED on myPins[i] from brightest to off:
    for (int brightness = 255; brightness >= 0; brightness--) {
      analogWrite(myPins[i], brightness);
      delay(5);
    }
    // pause between LEDs:
    delay(100);
  }
}

```

Postopek pa je:

- Naredimo vezje (pazimo na polariteto). Ledice zaporedno vežemo z upori in priključimo na PWM izhode Arduino UNO plošče.
- V delovnem okolju Arduino IDE ustvarimo program (ali naredimo ustrezne spremembe k obstoječemu programu).
- Program preko USB povezave prelijemo v UNO ploščo.
- Če nismo priključili zunanjega napajanja, mora biti UNO ves čas povezan z računalnikom (zaradi napajanja).
- Program deluje, dokler ga ne nadomesti drug program oz. do prekinitve napajanja.

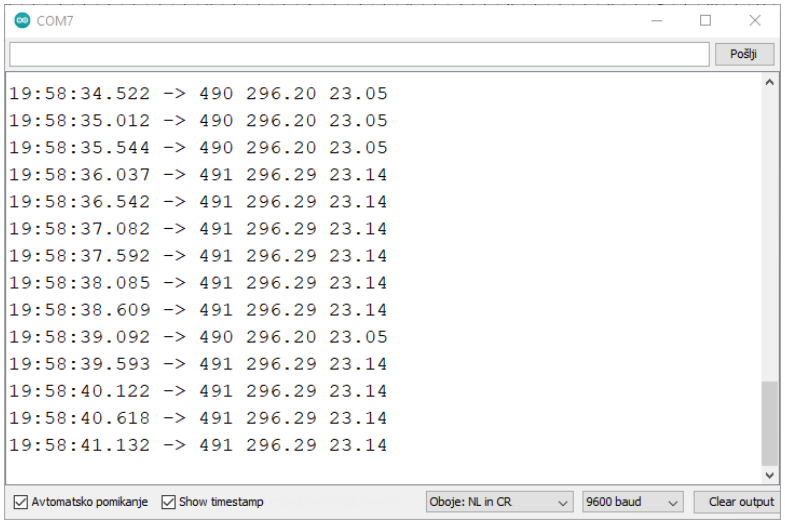
7.2 Termometer

Program uporablja 2x16 LCD monitor za prikaz temperature, ki jo izmerimo s termistorjem. Uporabljamo NTC termistor, ki se mu z naraščanjem temperature upornost zmanjšuje. Temperaturo na osnovi spremembe upornosti se računa s Steinhart-Hartovo enačbo, kjer moramo poznati karakteristiko uporabljenega termistorja.
Za LCD je potrebna posebna knjižnica, ki omogoča prikaz na monitorju.
Dodali smo ukaze „Serial“, ki isto informacijo (odčitek na analognem priključku »tempReading«, »tempK«, »tempC«), prikazuje na računalniku.
Uporabili smo:

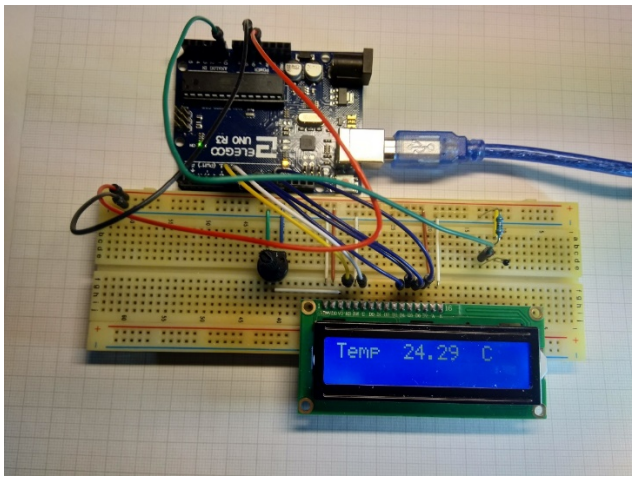
- UNO platformo (neoriginalno – Elegoo UNO)
- LCD monitor 2x16
- Potenciometer (obkrožen z rumeno) za nastavljanje svetlosti LCD
- Termistor (obkrožen z rdečo)

Uporabili smo naslednje pine:

- AI0 za merjen signal
- Digitalne pine 7, 8, 9, 10, 11, 12 za prikaz na LCD monitorju.



Prikaz na računalniku



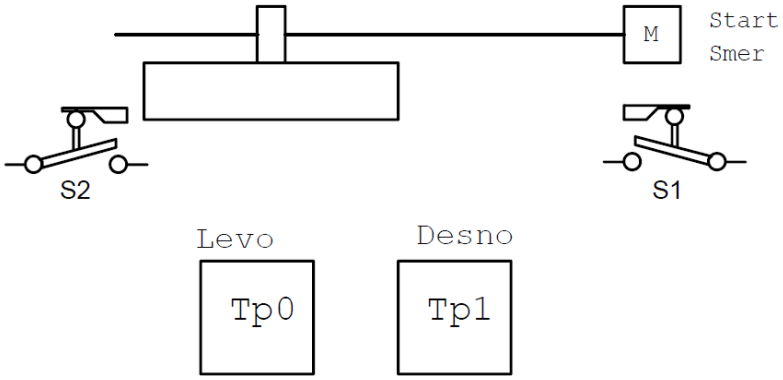
Vezava in prikaz na LCD monitorju.

```
//www.elegoo.com
//2016.12.9
//2021.4.11 serial communication added - displaying analogRead, tempK, tempC
//          possibility of another thermistor acc. to Steinhart-Hart eq.

#include <LiquidCrystal.h>  // knjižnica za LCD 2x16
int tempPin = 0;
//                BS  E  D4 D5  D6 D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup()
{
  lcd.begin(16, 2);    //nastavitev LCD
  Serial.begin(9600);  //baud rate serijske komunikacije 9600 b/s (počasno)
}
void loop()
{
  int tempReading = analogRead(tempPin); //branje iz analognega pina AI0

  // This is OK
  double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));  //upornost
  tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK * tempK )) *
  tempK );
// Izračun temperature (v Kelvinih!) na osnovi karakteristike termistorja
// tempK na desni strani zgornjega izraza se nanaša na upornost,
// taka uporaba pa je primerna zaradi varčevanja s spominom
//  tempK = 1 / (0.001213 + (0.000253 * tempK ));       //  Temp Kelvin for the
thermistor type 4k7
  float tempC = tempK - 273.15;            // Convert Kelvin to Celcius
  float tempF = (tempC * 9.0)/ 5.0 + 32.0; // Convert Celcius to Fahrenheit
  /*  replaced
    float tempVolts = tempReading * 5.0 / 1024.0;
    float tempC = (tempVolts - 0.5) * 10.0;
    float tempF = tempC * 9.0 / 5.0 + 32.0;
  */
  Serial.print(tempReading); // na ekran računalnika pišemo analogni signal
  Serial.print(" ");
  Serial.print(tempK); // temperatura v K na osnovi Steinhart-Hartove enačbe
  Serial.print(" ");
  Serial.println(tempC); // pretvorba v C
  // Display Temperature in C
  lcd.setCursor(0, 0); // nastavitev zapisa LCD
  lcd.print("Temp         C  "); // zapis na LCD
  // Display Temperature in F
  //lcd.print("Temp         F  ");
  lcd.setCursor(6, 0); // nastavitev 2. zapisa na LCD
  // Display Temperature in C
  lcd.print(tempC); // zapis temperature v C
  // Display Temperature in F
  //lcd.print(tempF);
  delay(500); // zakasnitev 500 ms
}
```

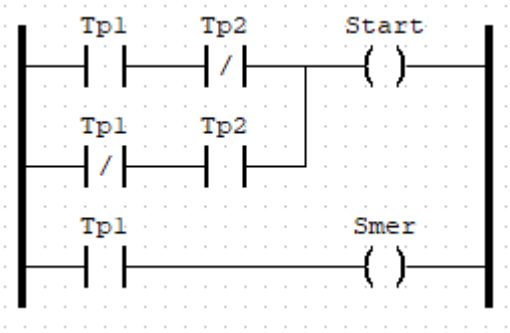
7.3 Simulacija PLC

Na spodnji sliki je prikazana delovna miza, ki jo premika motor preko krogličnega vretena in matice. Leva in desna meja gibanja sta določeni s končnima stikaloma S2 in S1. Tipki določata smer pomika mize. Če pritisnemo obe tipki hkrati, se miza ne premakne.

Da bi rešili to nalogo, moramo opisati delovanje krmilnika, definirati vhode in izhode, določiti pravilnostno tabelo in zapisati in preveriti lestvični diagram.



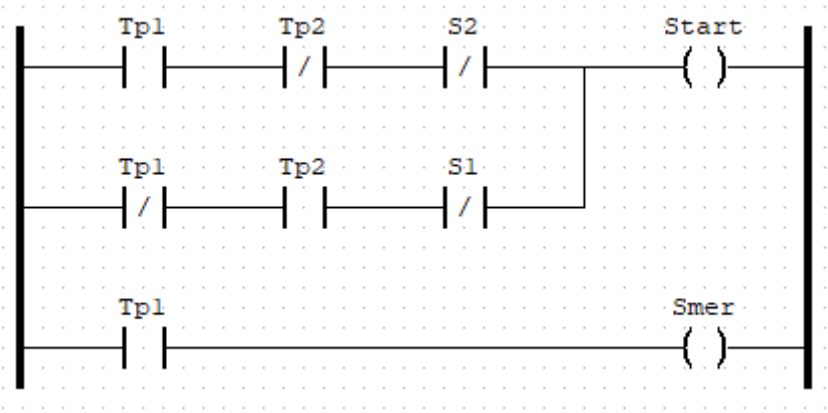

```
1  LDN TP1
2  AND TP2
3  ST M0
4
5  LD TP1
6  ANDN TP2
7  OR M0
8  ST Start
9
10 LD Tp1
11 ST Smer
```



Tp1	Tp2	Start	Smer
0	0	0	-
0	1	1	0
1	0	1	1
1	1	0	-

Najprej pa rešimo preprost primer, ko motor omogoča zgolj vrtenje v levo ali v desno. Ne rabimo končnih stikal, pač pa zgolj krmilna signala: start in smer. Za to nalogo bo primerna XOR logika (pravilnostna tabela na desni), z vezavo, ki je programirana v zgornjem lestvičnemu diagramu LD (na sredini) in IL (na levi).

```
1  LDN Tp1
2  AND Tp2
3  ANDN S2
4  ST M0
5
6  LD Tp1
7  ANDN Tp2
8  ANDN S1
9  OR M0
10 ST Start
11
12 LD Tp1
13 ST Smer
```



Potem pa uvedemo obe končni stikali (S1 na desni in S2 na levi). Obe morata biti normally closed. Torej S2 ustavi vrtenje motorja, ko se miza pomika v levo in S1 pri pomiku v desno. Lestvični diagram in IL sta prikazana na sliki na prejšnji strani.

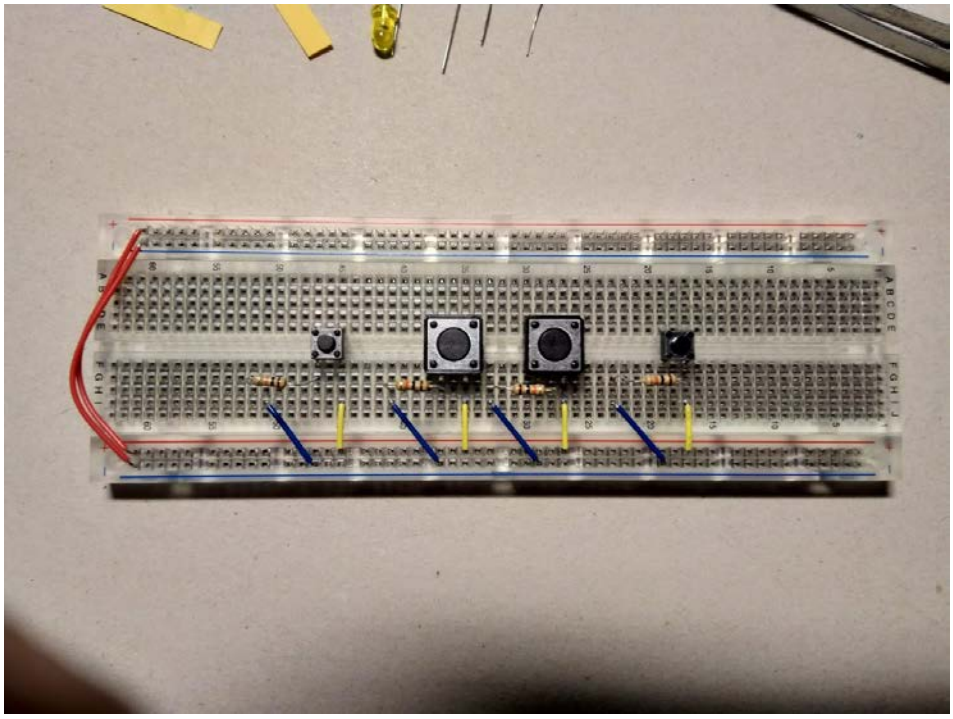
Za delo z Arduino UNO moramo najprej postaviti elemente na prototipno ploščo. Za začetek smo naredili postavitev za preprosto simulacijo s tipkami in ledicami. To ustreza tudi PLC

simulatorju. Nato pa smo v Arduino IDE platformi naredili preprost program z logiko IL oz. LD (spodnja slika na prejšnji strani), ki je prikazan spodaj.

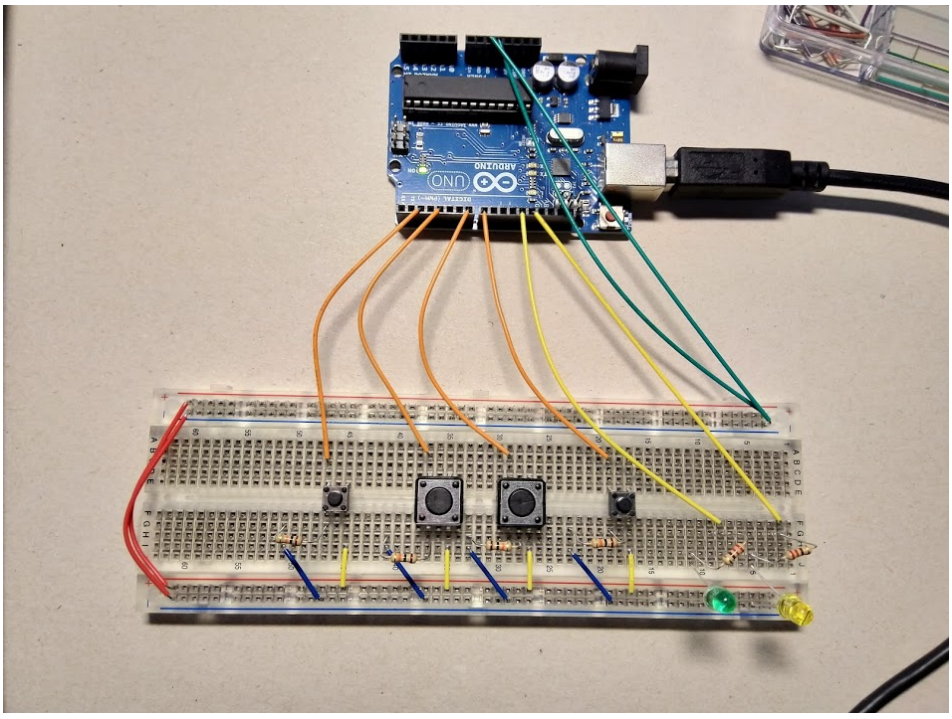
```
int buttonPin[] = {2, 4, 7, 8};
int ledPin[] = {12, 13};

void setup() {
  // put your setup code here, to run once:
  bool T0 = false;
  bool T1 = false;
  bool S1 = false;
  bool S2 = false;
  bool ON = false;
  bool SMER = false;
  for  (int i=0; i <= 3; i++){
    pinMode(buttonPin[i], INPUT);
  }
  for  (int i=0; i <= 1; i++){
    pinMode(ledPin[i], OUTPUT);
  }

}

void loop() {
  // put your main code here, to run repeatedly:
  bool T0 = digitalRead(buttonPin[1]); //Tp1
  bool T1 = digitalRead(buttonPin[2]); //Tp2
  bool S1 = digitalRead(buttonPin[3]); //S1
  bool S2 = digitalRead(buttonPin[0]); //S2

  bool M1 = T0 && !T1 && !S2;
  bool M2 = !T0 && T1 && !S1;
  bool ON = M1 || M2;
  bool SMER = T0;
  digitalWrite(ledPin[0], ON);
  digitalWrite(ledPin[1], SMER);
}
```

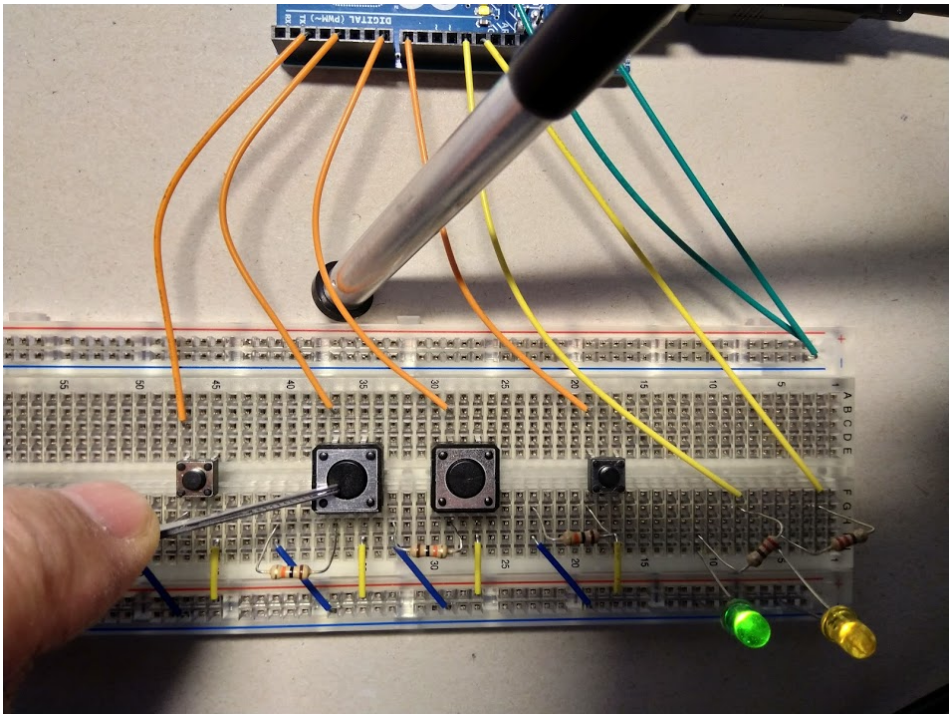
Tako kot prej, moramo tudi zdaj najprej definirati digitalne priključke. Nato definiramo začetne vrednosti spremenljivk (boolean) in digitalne priključke definiramo bodisi kot vhodne ali kot izhodne. Zanka je neskončna (ne ustavi se s programskim pogojem). V

vsakem preteku zanke se preveri stanje tipk. Potem pa se z logičnimi funkcijami ustvari ustrezno stanje izhodov (v tem primeru ledic). Delovanje z motorjem ne bi bilo bistveno drugačno. Motor bi se vrtel v eno ali drugo smer ali pa bi miroval.

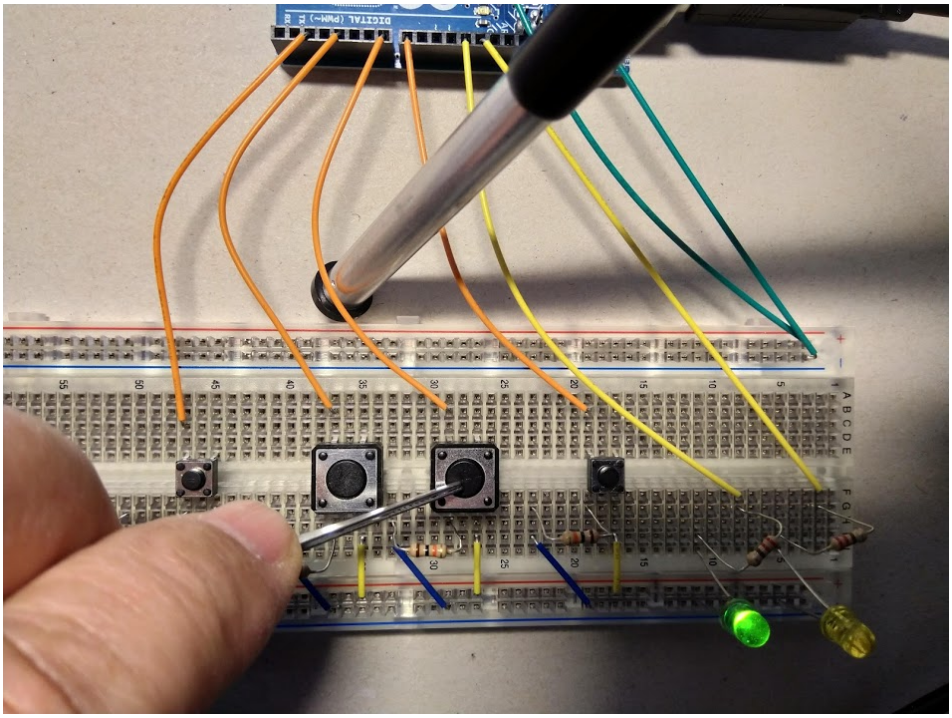


Najprej pripravimo elemete na prototipni plošči. Veliki tipki sta za TP0 in TP1 (start/levo, desno), mali pa za S2 in S1 (levo in desno končno stikalo). Tipke in ledice so serijsko vezane z ustreznimi upori.





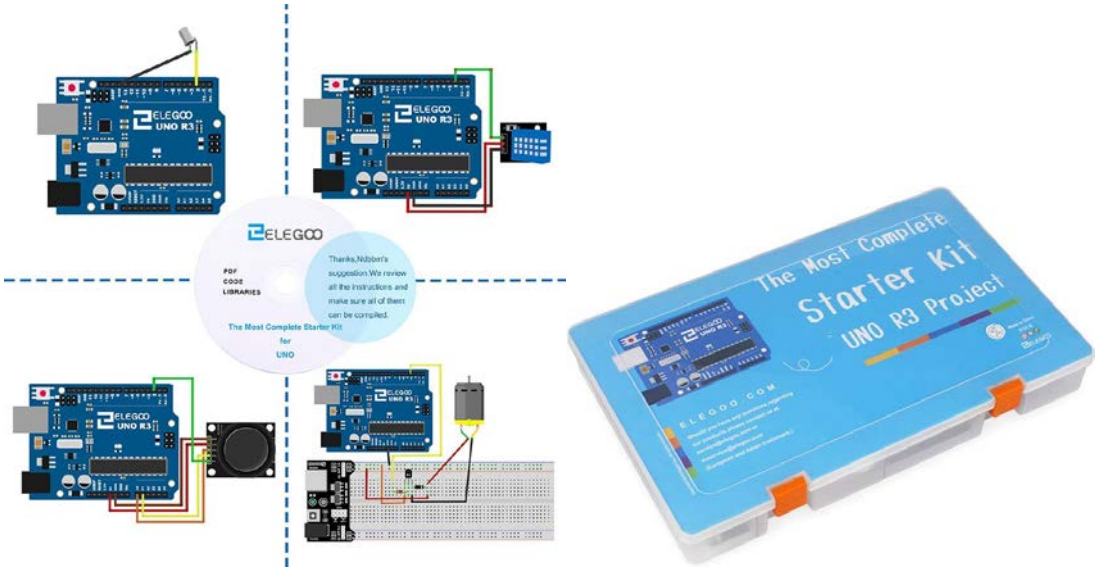
Če pritisnemo TP0 imamo prižgani ledici za start (motor se vrti) in za smer (levo).



Če pa pritisnemo TP1, deluje ledica za start (motor se vrti v drugo smer), ledica za smer pa je ugasnjena (torej smer desno). Če se pritisne še prava tipka za končno stikalo (v tem primeru S1) bo motor ugasnil (ledica start ugasne).

8 UČNI IN RAZVOJNI KOMPLETI (KOMPATIBILNI)

Poleg kompletov, ki jih ponuja Arduino, obstaja cela vrsta bogatih kompatibilnih kompletov, npr. Elegoo, slika spodaj. Na Amazon.de (nemški) je tak komplet na razpolago za 49,34 € + 11,99 € za poštnino. Link: Elegoo Set/Kit for Arduino UNO Project The Complete Ultimate Starter Kit with German Tutorial, UNO R3 Microcontroller and Many Accessories for Arduino UNO R3: Amazon.de: Computer & Accessories



Elegoo ponuja tudi mobilne robote, ki so cenovno zelo ugodni. (79,99 € + 11,99 €)

https://www.amazon.de/-/en/Electric-Construction-Tracking-Ultrasonic-Smartphone/dp/B07474MMB5/ref=sr_1_12?crid=10IG4AYGZ0DQX&keywords=elegoo+arduino&qid=1670329009&sprefix=%2Caps%2C92&sr=8-12


ELEGOO
UNO R3
MPU6050
SMART ROBOT CAR V4.0 WITH CAMERA
Assembly Tutorial