Merni sistemi u računarstvu, https://automatika.etf.bg.ac.rs/sr/13e053msr

Arduino programiranje II deo

Dr Nadica Miljković, redovna profesorka, kabinet 68, nadica.miljkovic@etf.bg.ac.rs

Prezentacija za ovu vežbu je delimično pokrivena knjigom Alan G. Smith, Introduction to Arduino: A piece of cake!, 2011, online: https://www.introtoarduino.com/downloads/IntroArduinoBook.pdf

Slika za naslovni slajd: https://www.arduino.cc/

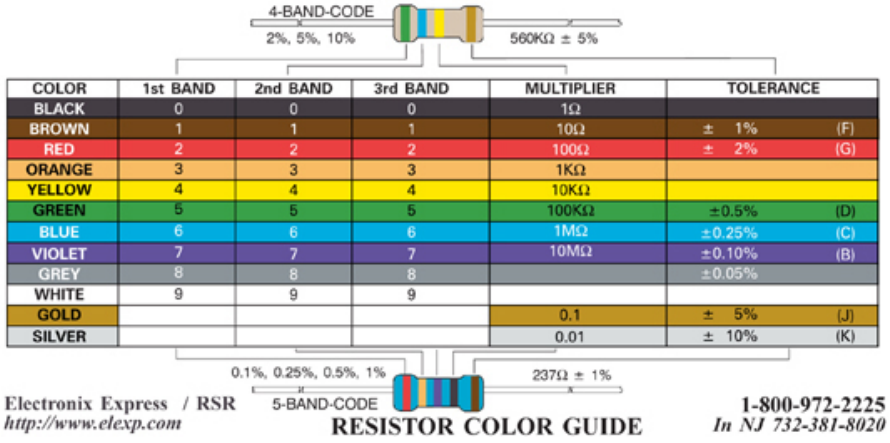
COLOR / COLOUR CODE

Color code

By Michael L. Umbricht - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=31024638.

- Elektronski *color code* se koristi kako bi se predstavile različite kvantitativne vrednosti električnih komponenti, https://en.wikipedia.org/wiki/Electronic_color_code
- Najčešće se koristi za otpornike.
- Može se koristiti i za kondenzatore, induktivnosti, diode i druge električne komponente.
- Razvijen je i uveden 20-ih godina prošlog veka od strane *Radio Manufacturers Association,* https://en.wikipedia.org/wiki/Electronic_Industries_Alliance

Color code otpornika

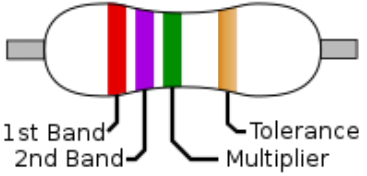


COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
BLACK	0	0	0	1Ω		
BROWN	1	1	1	10Ω	± 1%	(F)
RED	2	2	2	100Ω	± 2%	(G)
ORANGE	3	3	3	1KΩ		
YELLOW	4	4	4	10KΩ		
GREEN	5	5	5	100KΩ	±0.5%	(D)
BLUE	6	6	6	1MΩ	±0.25%	(C)
VIOLET	7	7	7	10MΩ	±0.10%	(B)
GREY	8	8	8		±0.05%	
WHITE	9	9	9			
GOLD				0.1	± 5%	(J)
SILVER				0.01	± 10%	(K)

Electronix Express / RSR
http://www.elexp.com

RESISTOR COLOR GUIDE

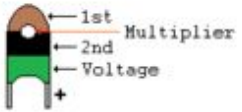
1-800-972-2225
In NJ 732-381-8020



- Na slici je prikazan *color code* iz MIEM udžbenika (Courtesy of Electronix Express, www.elexp.com).
- Zanimljiv *online* članak sa relativno velikim brojem primera se može naći na: http ://www.instructables.com/id/From-Resistors-to-ICs-Color-Codes/ (“From Resistors to Ics Color Codes” by Josehf Murchison), pristupljeno 16.10.2023.
- Najčešće postavljana pitanja:
  - Kako odrediti šta je na otporniku “levo”, a šta “desno”?
    - Kada se posmatra s leva na desno, poslednja linija odražava toleranciju i za nijansu je udaljena od ostalih linija (zeleni *multiplier* na slici gore desno, By jjbeard - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=823983).
  - Da li sam pravilno odredio/odredila boje?
    - Postoji standardna RAL šema koja koristi samo deo spektra (https://en.wikipedia.org/wiki/RAL_colour_standard).

Tantalum Capacitor Color Codes

Capacitance is in uF



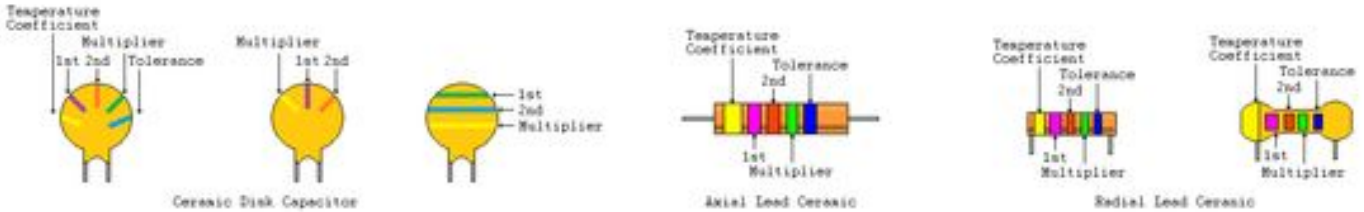
Color	1st	2nd	Multiplier	Voltage
Black	0	0		10
Brown	1	1	0	
Red	2	2	00	4
Orange	3	3		40
Yellow	4	4		6.3
Green	5	5		16
Blue	6	6		20
Violet	7	7	0.001	
Grey	8	8	0.01	25
White	9	9	0.1	3
Pink				35

Color code kondenzatora

Slika (pristupljeno 2019. godine), *Fair Use*, https://cdn.instructables.com/F0J/AG43/IFSJNABK/F0JAG43IFSJNABK.MEDIUM.jpg

Ceramic Capacitor Color Code With Temperature Coefficient

Capacitance is in Picofarad

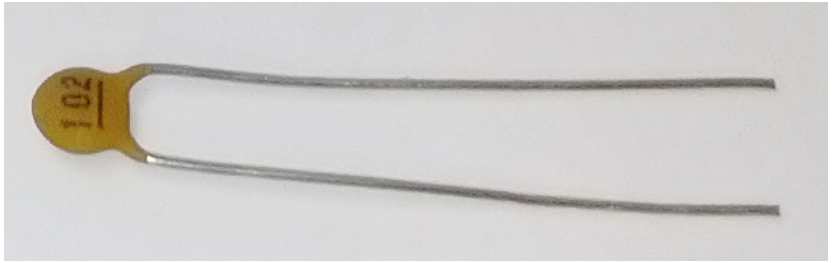


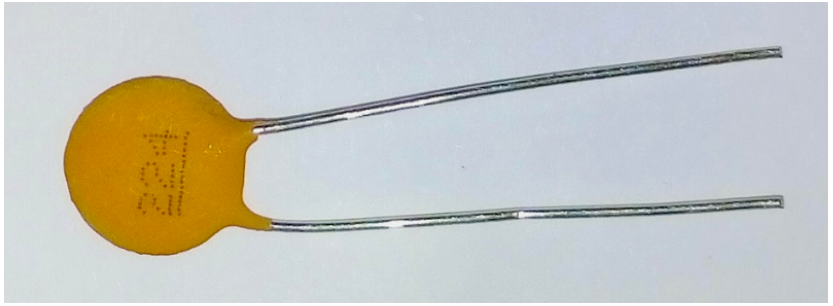
Color	1st	2nd	Multiplier	Tolerance 10pf+	Tolerance 10pf-	Temperature Coefficient
Black	0	0		20%	2.0pF	0
Brown	1	1	0	1%	0.1pF	-30
Red	2	2	00	2%	0.25pF	-80
Orange	3	3	000	3%		-150
Yellow	4	4	0,000	4%		-220
Green	5	5	00,000	5%	0.5pF	-330
Blue	6	6				-470
Violet	7	7				-750
Grey	8	8	0.01	+80% -20%	0.25pF	+30
White	9	9	0.1	10%	1.0pF	+120 to -750 (EIA) +500 to -330 (JAN)
Gold			0.1	5%		Bypass or Coupling
Silver			0.01	10%		+100 (JAN)
None				20%		

Color code kondenzatora

Slika (pristupljeno 2019. godine), *Fair Use*, https://cdn.instructables.com/F0J/AG43/IFSJNABK/F0JAG43IFSJNABK.MEDIUM.jpg

Oznake na keramičkim kondenzatorima



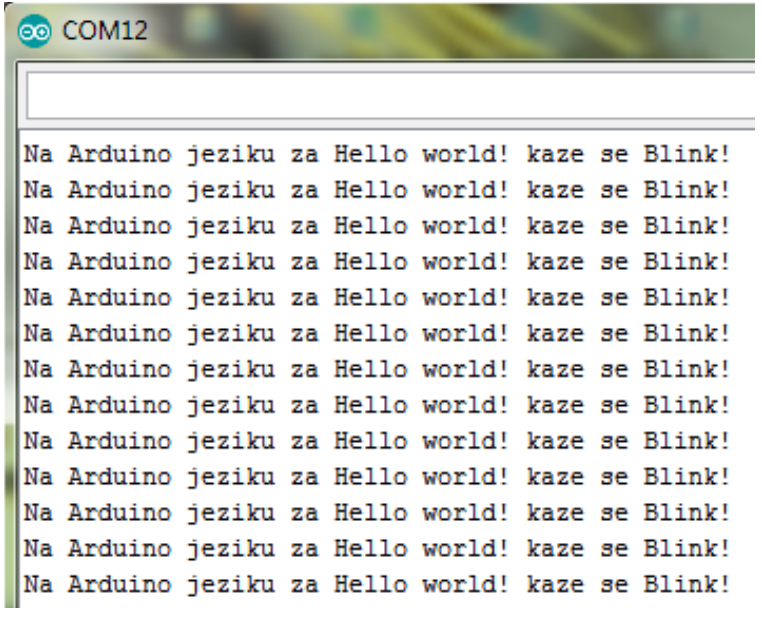


- Kod označavanja kapacitivnosti kondenzatora, koriste se osim *color code*-a i oznake koje su najčešće predstavljene kao brojevi. Neke od oznaka su: 108, 158, 228, 338, 478, 102, 222, 224.
- Korisna tabela sa oznakama je dostupna na: http://grathio.com/assets/capacitor_tags.pdf, pristupljeno 16.10.2023. Primeri kondenzatora sa numeričkim oznakama su prikazani na slici.

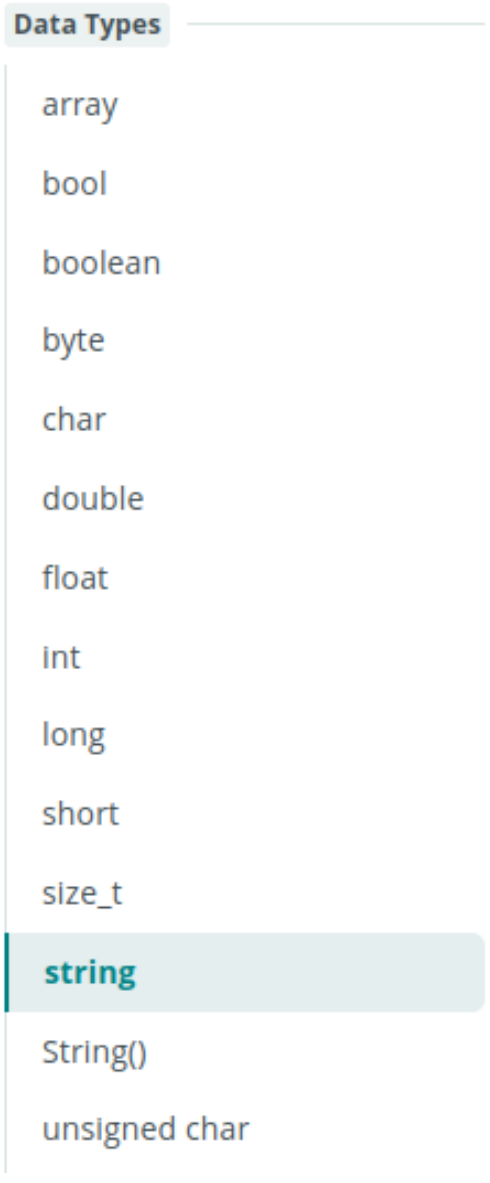
STRINGOVI U ARDUINO PROGRAMSKOM OKRUŽENJU

Primer sa prethodnog časa

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("Na Arduino jeziku za Hello world! kaze se Blink!");
  delay(100);
}
```



- Na slici je prikazan primer ispisa poruke na serijskom portu.
- Dve osnovne funkcije za rad sa stringovima su *Serial.print()* i *Serial.println()*.
- Šta bi bilo da je iskorišćena funkcija *Serial.print()*?



String() i *string*

- Prikazana je tabela sa (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/en/variables/data-types/string/
- Postoji objekat tipa *string*: na slici prikazan kao *String()*. Primer korišćenja ovog objekta je dat na sledećem slajdu.
- Postoji i podatak tipa *string*, koji je sastavljen iz niza karaktera, gde je poslednji karakter jednak nuli (eng. *null-terminate*).
- Osnovna razlika između *string*-ova i karaktera je što se *string*-ovi definišu između dvostrukih znaka navoda (“ABC”), a karakteri između jednostrukih znaka navoda (‘A’).
- Najčešće se stringovi koriste kao nizovi.

Objekat tipa *String()*

```
String stringOne = "Hello String";                         // using a constant String
    String stringOne = String('a');                        // converting a constant cha
    String stringTwo = String("This is a string");         // converting a constant str
    String stringOne = String(stringTwo + " with more");   // concatenating two strings
    String stringOne = String(13);                         // using a constant integer
    String stringOne = String(analogRead(0), DEC);         // using an int and a base
    String stringOne = String(45, HEX);                    // using an int and a base (
    String stringOne = String(255, BIN);                   // using an int and a base (
    String stringOne = String(millis(), DEC);              // using a long and a base
    String stringOne = String(5.698, 3);                   // using a float and the dec
```

- Na slici su dati primeri korišćenja objekata tipa *String()*.
- Slika je preuzeta sa sajta (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/en/variables/data-types/stringObject/

Primer

```
char* myStrings[]={"This is string 1", "This is string 2", "This is string 3",
"This is string 4", "This is string 5","This is string 6"};

void setup(){
Serial.begin(9600);
}

void loop(){
for (int i = 0; i < 6; i++){
   Serial.println(myStrings[i]);
   delay(500);
   }
}
```

- Primer niza *string*-ova je dat na slici. Kod prikazan na slici je preuzet sa sajta: https://www.arduino.cc/reference/en/language/variables/data-types/string/, pristupljeno 16.10.2023.
- Najčešće se ovakvi nizovi stringova prave kada se sa Arduino ili sličnim hardverom koristi LCD (eng. *Liquid-Crystal Display*, https://en.wikipedia.org/wiki/Liquid-crystal_display).
- Šta znači * pored *char* u prvom redu koda sa slike? Koja je funkcija koda sa slike?

Karakteri

Characters

isAlpha()

isAlphaNumeric()

isAscii()

isControl()

isDigit()

isGraph()

isHexadecimalDigit()

isLowerCase()

isPrintable()

isPunct()

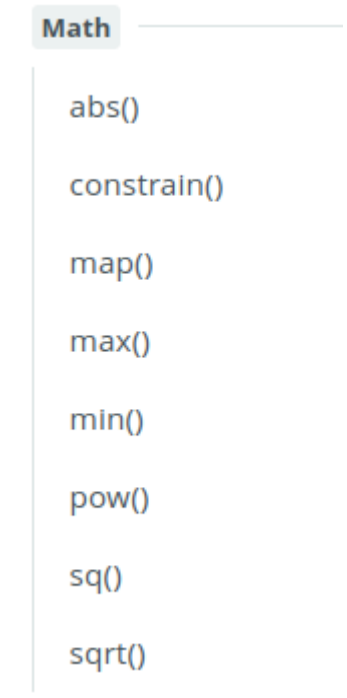
isSpace()

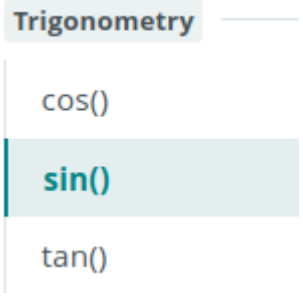
isUpperCase()

isWhitespace()

- Funkcije za rad sa karakterima su prikazane u tabeli levo.
- Tabela je preuzeta sa (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/#functions
- Za rad sa *string*-ovima i karakterima, postoje i ugrađeni primeri (u folderu/fascikli “08.Strings”), kao što su: *CharacterAnalysis.ino*, *StringAdditionOperator.ino*, *StringCaseChanges.ino* i *StringCharacters.ino*.
  - Idealno za zadatak za ispit!

JOŠ FUNKCIJA ZA RAD U ARDUINO OKRUŽENJU





Trigonometrijske i matematičke funkcije

- Trigonometrijske i matematičke funkcije su prikazane na slici – dostupne su na (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/en/functions/trigonometry/sin/
- Lista nije kompletna, a za specifične funkcije, moguće je koristiti i odgovarajuće biblioteke.
- *constrain()* se najčešće koristi za vrednosti, koje su učitane sa analognih senzora i služi da ograniči prikaz između granica *a* i *b* – pogledati (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/en/functions/math/constrain/
- Funkcija *sq()* računa kvadrat nekog broja (eng. *square*), a više na (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/en/functions/math/sq/

Pseudoslučajni brojevi

Random Numbers

random()

randomSeed()

- Postoje funkcije *random()* i *randomSeed()* koje se koriste za generisanje “slučajnih” brojeva. Pogledati više na (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/#functions
- O njima ćemo više, kada budemo radili mernu nesigurnost tipa A i tipa B.
- Zašto sam napisala pseudoslučajni, a ne slučajni?
- *Seed* se koristi u slučaju kada postoji potreba za generisanjem ponovljenih sekvenci pseudoslučajnih brojeva → veoma korisno, posebno zbog računarske reproducibilnosti (eng. *computational reproducibility*).

Tajmer

Notes and Warnings

While it is easy to create a blinking LED with the `delay()` function, and many sketches use short delays for tasks such as switch debouncing, the use of `delay()` in a sketch has significant drawbacks. No other reading of sensors, mathematical calculations, or pin manipulation can occur during the delay function, so, in effect, it brings most other activity to a halt.

For alternative approaches to controlling timing, see the **Blink Without Delay** sketch, which loops, polling the **millis()** function until enough time has elapsed. More knowledgeable programmers usually avoid using `delay()` for timing events longer than 10s of milliseconds, unless the Arduino sketch is straightforward.

Certain things do occur while the `delay()` function is controlling the chip because the delay function does not disable interrupts. Serial communication that appears at the RX pin is recorded. PWM (**analogWrite**) values and pin states are maintained, and interrupts (**attachInterrupt**) will function as expected.

Time

delay()

delayMicroseconds()

micros()

millis()

- Funkcije koje se koriste za merenje vremena su prikazane na slici (https://docs.arduino.cc/language-reference/#functions, pristupljeno 19.11.2025. godine). Do sada, na lab. vežbama je korišćen primer *delay()*, a od danas *millis()* i *micros()*.
- Pogledati na slici ispod upozorenje kod korišćenja *delay()* funkcije: https://docs.arduino.cc/language-reference/en/functions/time/delay/, pristupljeno 16.10.2023.

millis() i *micros()* funkcije

micros()

Last revision • 06/05/2025

Description

Returns the number of microseconds since the Arduino board began running the current program. This number will overflow (go back to zero) after approximately 70 minutes.

millis()

Last revision • 06/05/2025

Description

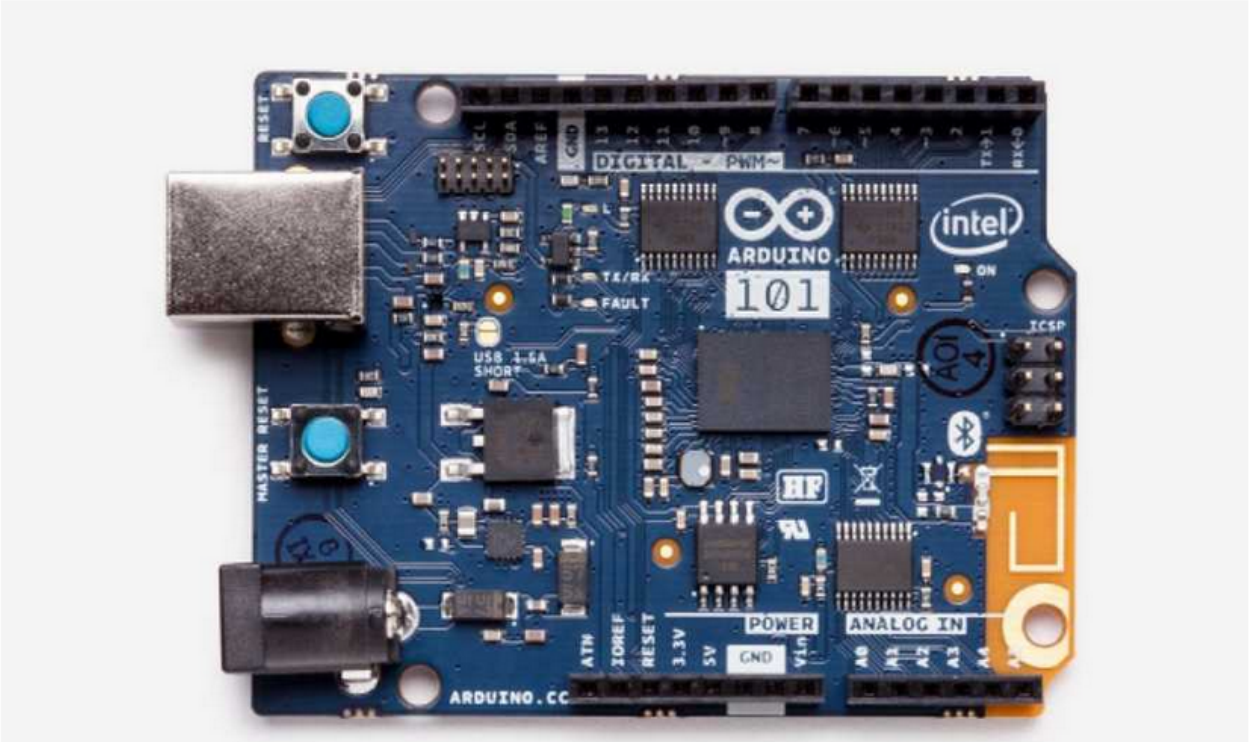
Returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

- Za UNO R3 (Mega328P) *clock* frekvencija je 250 kHz (pristupljeno 16.10.2023. godine): https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- Dodatno, 32 b je rezervisano za *timer*.
- Pogledati detaljno uputstvo za ove dve funkcije pre primene na Arduino sajtu (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/#functions

Arduino 101

Arduino 101 combine the ease-of-use of the classic boards with the latest technologies. The board recognises gestures, and features a six-axis accelerometer and gyroscope. Control your projects with your phone over Bluetooth® connectivity!

Last revision • 03/14/2024



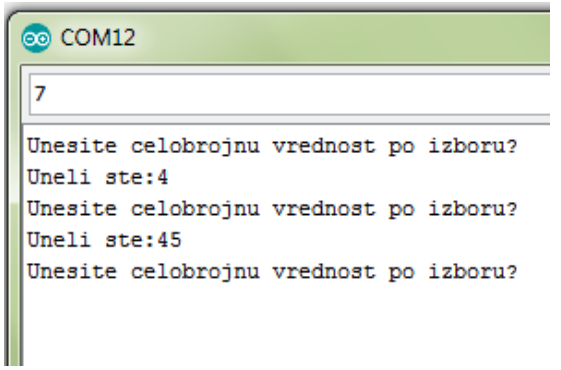
Druge Arduino biblioteke

- Arduino okruženje može biti “prošireno” korišćenjem biblioteka.
- Kako bi se uvezla neka biblioteka koristi se padajući meni *Sketch* i opcija *Import Library*.
- Za one koji to žele, postoje i uputstva kako napisati biblioteku za Arduino (pristupljeno 16.10.2023. godine): https://www.arduino.cc/en/Hacking/LibraryTutorial
- Jedan savet: Nikada nemojte u potpunosti “verovati” nekoj biblioteci, proverite šta tačno radi koja funkcija, pre nego što je iskoristite. Nekada brza rešenja nisu najbolja rešenja.
- Slika (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/retired/boards/arduino-101-619/

NEKE KORISNE FUNKCIJE

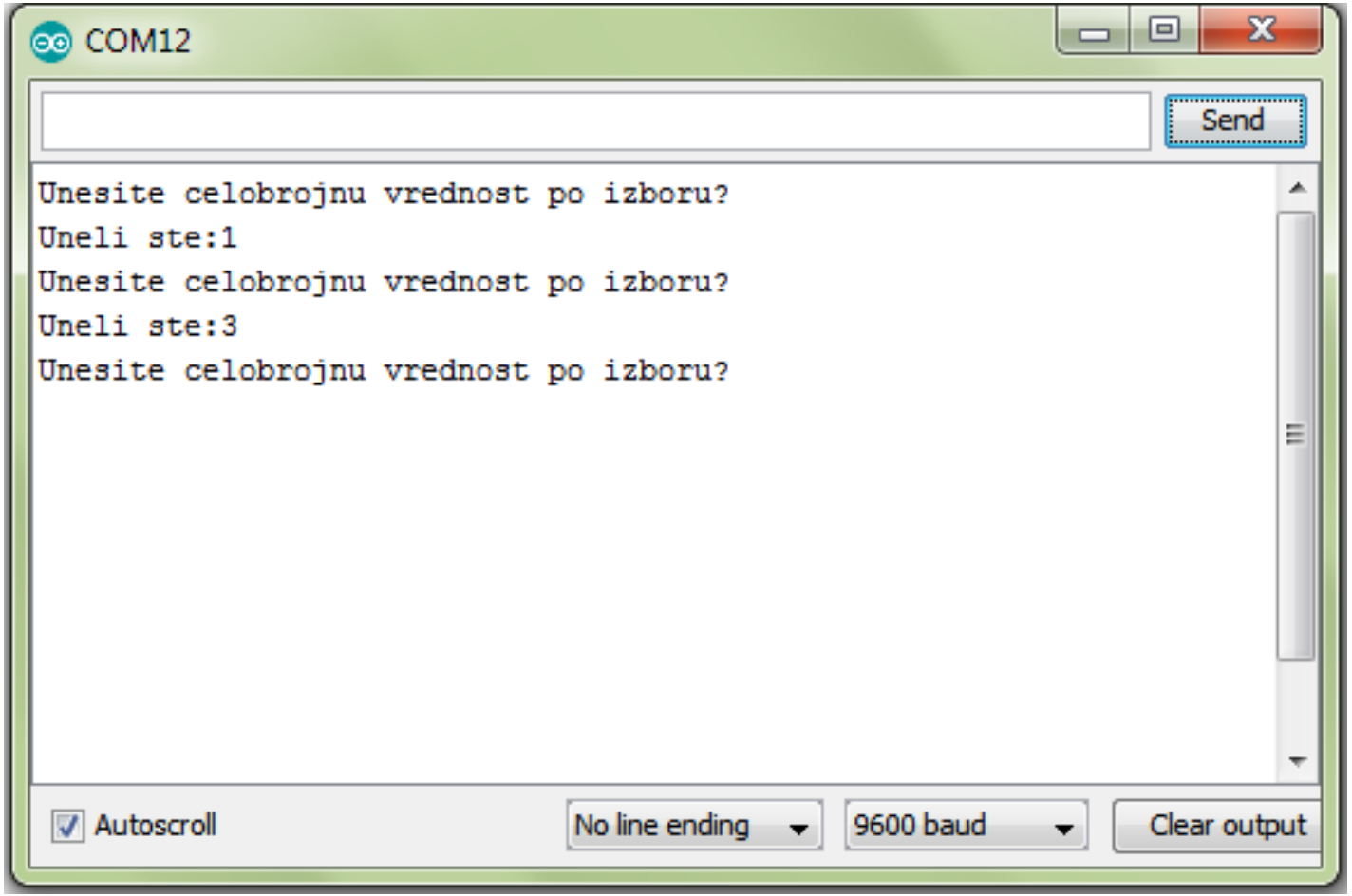
Korisnički unos u Arduino programu

```
/*
  Primer korisničkog unosa parametara.
*/
int broj;  // celobrojna vrednost po izboru korisnika

void setup() {
  // serijski port se koristi za unos broja
  Serial.begin(9600);
}

void loop() {
  // stampanje pitanja na serijskom portu (Serial monitor)
  Serial.println("Unesite celobrojnu vrednost po izboru?");
  while(Serial.available() == 0) { // ceka se unos korisnika
  }
  broj = Serial.parseInt();  // ocitava se vrednost koju unosi korisnik
  // prikazuje se sta je uneo korisnik na monitoru
  Serial.print("Uneli ste: ");
  Serial.println(broj);
}
```



- Moguće je realizovati takav program da korisnik unosi parametre preko serijskog porta.
- Koristi se funkcija *Serial.parseInt()* za koju uputstvo možete pogledati na: https://www.arduino.cc/en/Serial/ParseInt, pristupljeno 16.10.2023.
- Primer koda i izgled serijskog monitora dati su na slici. Kako bi ste dodali *timeout*? Šta se desi ako se unese 1.1?

Unos 1.3


COM12
Send
Unesite celobrojnu vrednost po izboru?
Uneli ste:1
Unesite celobrojnu vrednost po izboru?
Uneli ste:3
Unesite celobrojnu vrednost po izboru?
Autoscroll
No line ending
9600 baud
Clear output

O MERENJU KAPACITIVNOSTI

Biće na lab. vežbama

$$\Delta t = RC\ln 9$$

- Merenje kapacitivnosti je određeno rezolucijom tajmera.
- Ako je R = 1 MΩ i ako je kapacitivnost, koja se meri C = 470 pF (primer sa lab. vežbi), koliko je onda Δt? (1.03 ms)
- Kolika je najmanja vrednost, koju može da meri Arduino kod? (Obzirom da je za merenje Δt korišćena Arduino funkcija *millis()*, koja može da meri najmanju vrednost od 1 ms).
- Kako je moguće povećati ovu rezoluciju?
  - Korišćenjem većeg otpornika?
  - Korišćenjem tajmera?

Arduino tajmeri

- Arduino UNO kao i UNO R3 mikrokontrolerske pločice imaju tri tajmera: Timer0, Timer1 i Timer2. Prilikom korišćenja, potrebno je voditi računa o rezoluciji tajmera, jer se one razlikuju.
- *millis()* funkcija je povezana sa Timer0, koji omogućava realizaciju interapta / prekida svake milisekunde.
- Tajmeri su zapravo brojači koji rade na frekvenciji sistemskog sata (eng. *system clock*), koja je jednaka 16 MHz.
- Ako se *millis()* funkcija reinicijalizuje (vraća na 0) posle oko 50 dana, kolika je rezolucija ovog tajmera?
  - 50*24*60*60*1000 = 4 320 000 000 ≈ 2^{32}

ARDUINO PROGRAMIRANJE
PREKIDI I FUNKCIJE

Prekidi

External Interrupts

attachInterrupt()

detachInterrupt()

digitalPinToInterrupt()

Interrupts

interrupts()

noInterrupts()

- U Arduino programskom okruženju postoje eksterni/spoljni i interni/unutrašnji prekidi/interapti.
- Prekidi su generalno jednostavan način da se “reaguje” na događaje u realnom vremenu.
- Drugim rečima, ove funkcije služe da procesor “brzo” odgovara na “važne” događaje.
- Spisak funkcija je preuzet sa sajta (pristupljeno 19.11.2025. godine): https://docs.arduino.cc/language-reference/#functions
- Najjednostavniji primer kada koristiti prekide?

```
void setup(){
  Serial.begin(9600);
}

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // k now contains 6
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```

```
int ReadSens_and_Condition(){
  int i;
  int sval = 0;

  for (i = 0; i < 5; i++){
    sval = sval + analogRead(0);    // sensor on analog pin 0
  }

  sval = sval / 5;    // average
  sval = sval / 4;    // scale to 8 bits (0 - 255)
  sval = 255 - sval;  // invert output
  return sval;
}
```

Jedan prekid u ovoj prezentaciji

- Ako postoji 1) kod koji se ponavlja i/ili 2) potreba za modularnom organizacijom koda.
- Po pravilu, “dužina” koda bi trebalo da bude jednaka visini ekrana, a “širina” koda bi trebalo da bude jednaka širini ekrana.
- Primer dve funkcije koje korisnik definiše su date na slici (https://www.arduino.cc/en/Reference/FunctionDeclarationpristupljeno 16.10.2023).
- Koja je funkcija kodova sa slike?

Prekidi – primer

```
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin =  13; // the number of the LED pin

// variables will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH); // turn LED on
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

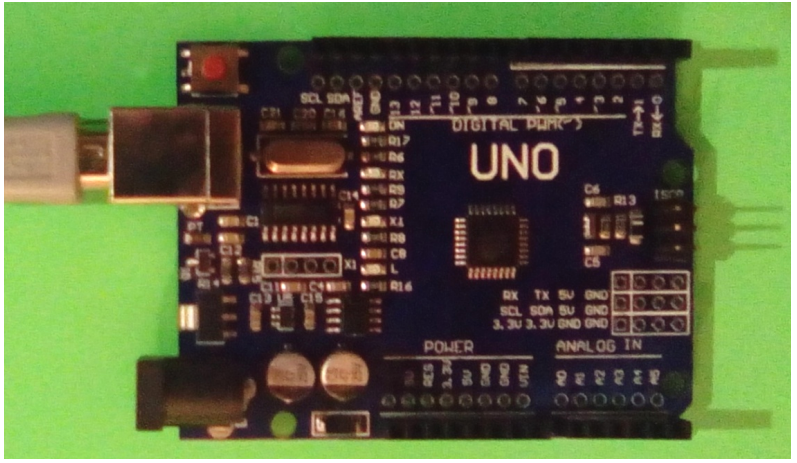
- Na slici je prikazan primer sa sajta: https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-arduino pristupljeno 16.10.2023.
- Šta nije u redu sa primerom sa slike? Pa, sve!
- Da bi se oslobodilo vreme procesora i da se ne bi proveravalo stanje *buttonPin*-a u svakoj iteraciji petlje, dodaje se prekid u kodu.
- Neka ideja?

Prekid

```
const int buttonPin = 2; // the number of the pushbutton pin
const int ledPin =  13; // the number of the LED pin

// variables will change:
volatile int buttonState = 0; // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // Attach an interrupt to the ISR vector
  attachInterrupt(0, pin_ISR, CHANGE);
}

void loop() {
  // Nothing here!
}

void pin_ISR() {
  buttonState = digitalRead(buttonPin);
  digitalWrite(ledPin, buttonState);
}
```

- Na slici je prikazan izmenjen kod.
- Koje su osnovne izmene u odnosu na kod prikazan na poslednjem slajdu?

Pažnja!



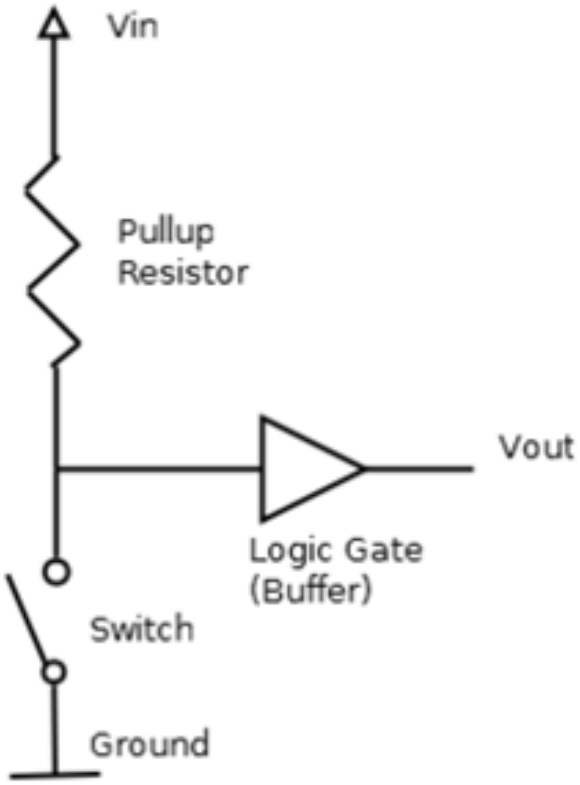
- Prekidi bi, po pravilu, trebalo da budu “kratkog trajanja”, odnosno ne bi trebalo na “duže” prekidati glavnu petlju:
  - Podaci sa serijskog porta mogu biti izgubljeni tokom izvršavanja prekida. Ni tajmeri neće raditi tokom prekida.
- Prekidi nemaju ulazne i izlazne promenljive kao funkcije. Sve promene moraju biti izvršene na globalnim promenljivim.
- Na UNO R3 mikrokontrolerskoj pločici na digitalnim pinovima 2 i 3 se mogu generisati prekidi, koji odgovaraju interapt vektorima 0 i 1.
- Pored promene (CHANGE) prekidi mogu da reaguju na RISING, FALLING i LOW.
- Još detalja na: https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-arduino/, pristupljeno 16.10.2023.

Volatile int vs. int

- *Volatile* (promenljiv) je podtip podataka (eng. *Qualifier*), koji se dodaje pre definicije promenljive i oznaka je kompajleru.
- Dodatno, kompajler je softver koji prevodi C/C++ kod u mašinski kod, koji sadrži “prave” instrukcije za ATMega čip. Ovo je jedna od osnovnih funkcija kompajlera. Ima ih još (*).
- *Volatile* je oznaka kompajleru da će se ta promenljiva koristiti u prekidima i da je potrebno smestiti tu promenljivu u RA memoriji.
  - Voditi računa o tome da je RA memorija mikrokontrolera ograničena (https://www.arduino.cc/en/Tutorial/Memory, pristupljeno 16.10.2023).
- (*) Kompajleri su zaduženi za “čišćenje/brisanje” (eng. *pruning*) nekorišćenih promenljivih.

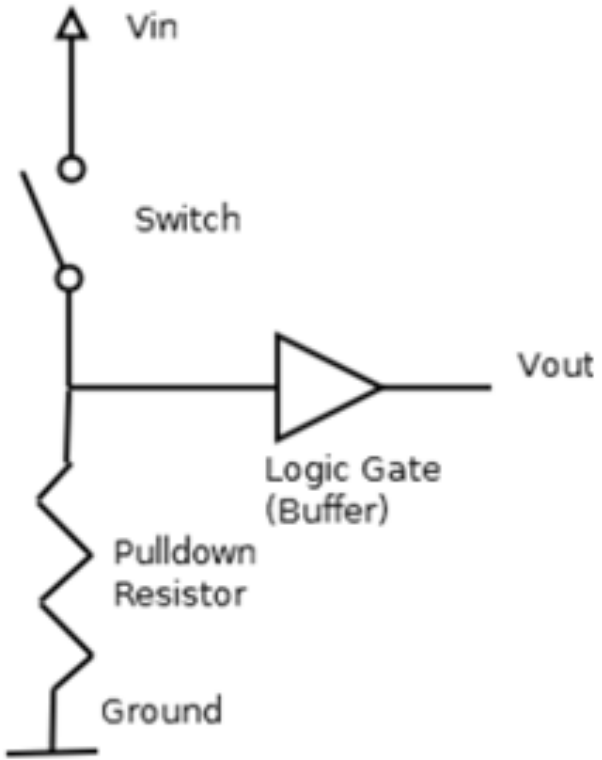
PULL-UP I *PULL-DOWN* OTPORNIK

Čemu služe i kada se koriste?



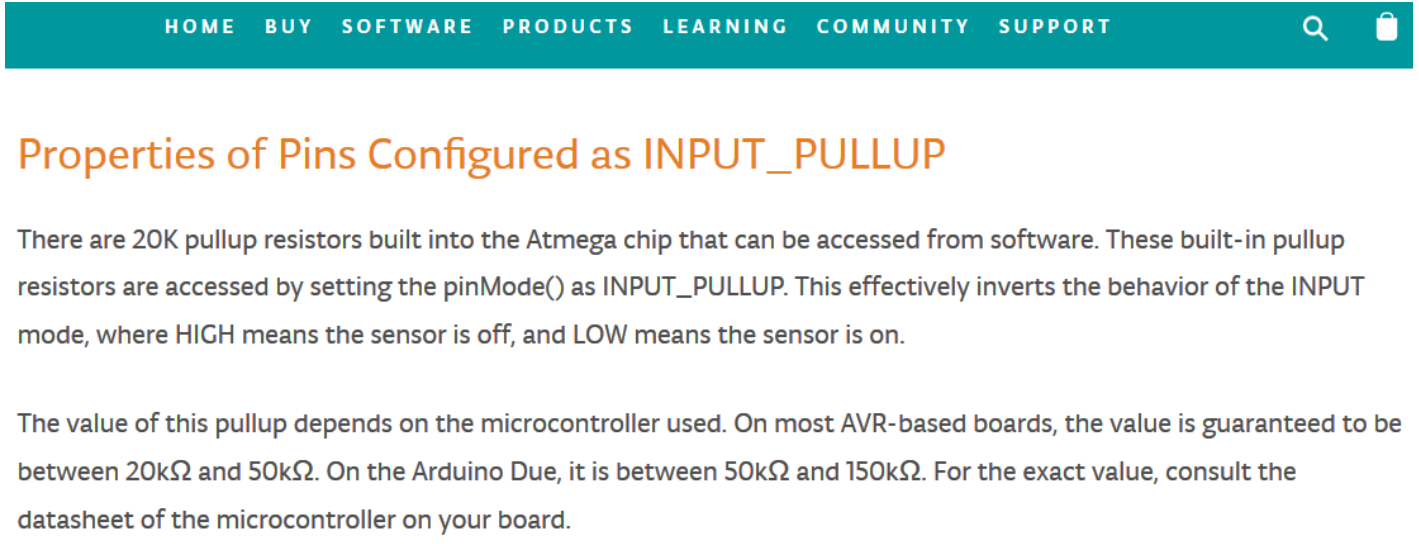
- *Pull-up* otpornici, https://en.wikipedia.org/wiki/Pull-up_resistor se dodaju na ulazu u Arduino pinove (digitalni ulazi), kako bi se omogućilo da logički nivo ostane na logičkoj jedinici za slučaj da je na primer pin nepovezan.
- Slika: https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/Pullup_Resistor.png/220px-Pullup_Resistor.png
- Ovaj otpornik “povlači” (eng. *pulls*) napon ka napajanju za koji je povezan, ako su druge komponente u kolu neaktivne.
- *Pull-up* otpornik omogućava da postoji definisan logički nivo na ulazu u Arduino, ako nijedan uređaj nije povezan.
- Postavlja ulaz na logičku vrednost 1, ako ništa nije povezano, odnosno ako je prekidač na slici otvoren.

Čemu služe i kada se koriste?



- *Pull-down* otpornik ima analognu funkciju, kao i *pull-up* otpornik sa razlikom što je umesto za napajanje povezan za masu (slika).
- On postavlja logički nivo na 0, kada nijedan uređaj nije povezan (kada je prekidač sa slike otvoren).
- Vrednost *pull-down* i *pull-up* otpornika zavisi od uređaja koji se koristi.
- Slika: https://upload.wikimedia.org/wikipedia/commons/thumb/3/3b/Pulldown_Resistor.png/220px-Pulldown_Resistor.png
- Generalno, ovi otpornici se koriste u kombinaciji sa bilo kojim mikrokontrolerom i nisu karakteristični samo za Arduino hardver.
- Na ovom i prethodnom slajdu su prikazani baferi ka *Vout*, odnosno ka digitalnom pinu na MCU.

Kako odabrati otpornik?



- Otpornici koji imaju relativno nižu otpornost se nazivaju *strong pull-up* (*strong* jer veća struja protiče kroz njih).
- Otpornici koji imaju relativno višu vrednost se nazivaju *weak pull-up* (*weak* jer kroz njih protiče manja struja).
- Kod ATmega328 kontrolera postoji INPUT_PULLUP pin mod (https://www.arduino.cc/en/Tutorial/DigitalPins, pristupljeno 16.10.2023).
- Praktično, ove vrednosti se biraju da budu oko 10 i više puta manje od ulazne impedanse.
- Treba imati na umu da velike otpornosti mogu izazvati kašnjenja zbog kapacitivnih efekata žica. Kako?