Računalnik v pouku Fizike - zbirka vaj

Skripta je namenjena študentom UL-PeF študijske smeri Fizike z vezavami. V skripti so opisani osnovni principi vzorčenja fizikalnih količin s krmilnikom Arduino NANO.

dr. David Rihtaršič

2023-April

Kazalo

1 Merilni sistemi

Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer [[Wikipedia]]. Data acquisition systems, abbreviated by the acronyms DAS or DAQ, typically convert analog waveforms into digital values for processing. The components of data acquisition systems include: - Sensors, to convert physical parameters to electrical signals. - Signal conditioning circuitry, to convert sensor signals into a form that can be converted to digital values. - Analog-to-digital converters, to convert conditioned sensor signals to digital values.

Data acquisition applications are usually controlled by software programs developed using various general purpose programming languages such as Assembly, BASIC, C, C++, C#, Fortran, Java, LabVIEW, Lisp, Pascal, etc. Stand-alone data acquisition systems are often called data loggers[[Wikipedia]].

There are also open-source software packages providing all the necessary tools to acquire data from different hardware equipment. These tools come from the scientific community where complex experiment requires fast, flexible and adaptable software. Those packages are usually custom fit but more general DAQ package like the Maximum Integrated Data Acquisition System can be easily tailored and is used in several physics experiments worldwide[[Wikipedia]].

1.1 Importance of real-life experiments for students

- where theory finds its place
- greater motivation for learning
- more (learning) input –> beater understanding

- instant data processing
- explicit presentation of interdependent variables
- visualization of physical quantities that are detected by human nature

1.2 Parts of a DAQ System [[NationalInstruments]]

1.3 What Is a Sensor?

The measurement of a physical phenomenon, such as the temperature of a room, the intensity of a light source, or the force applied to an object, begins with a sensor. A sensor, also called a transducer, converts a physical phenomenon into a measurable electrical signal. Depending on the type of sensor, its electrical output can be a voltage, current, resistance, or another electrical attribute that varies over time. Some sensors may require additional components and circuitry to properly produce a signal that can accurately and safely be read by a DAQ device [[NationalInstruments]]. Kot v enačbi en. **??** in na sliki sl. **??**.

(“What Is Data Acquisition? - National Instruments”)

1.4 What Is a DAQ Device?

DAQ hardware acts as the interface between a computer and signals from the outside world. It primarily functions as a device that digitizes incoming analog signals so that a computer can interpret them. The three key components of a DAQ device used for measuring a signal are the signal conditioning circuitry, analog-to-digital converter (ADC), and computer bus. Many DAQ devices include other functions for automating measurement systems and processes. For example, digital-to-analog converters (DACs) output analog signals, digital I/O lines input and output digital signals, and counter/timers count and generate digital pulses[NationalInstruments].

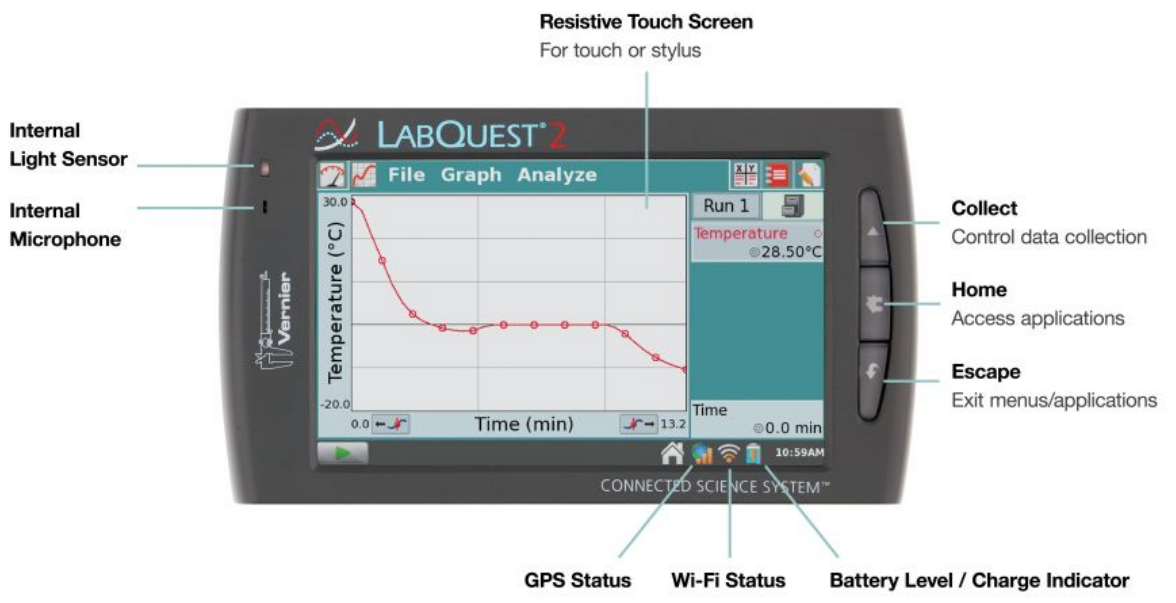
1.5 What Is a Computer’s Role in a DAQ System?

A computer with programmable software controls the operation of the DAQ device and is used for processing, visualizing, and storing measurement data. Different types of computers are used in different types of applications. A desktop may be used in a lab for its processing power, a laptop may be used in the field for its portability, or an industrial computer may be used in a manufacturing plant for its ruggedness (“What Is Data Acquisition? - National Instruments”).

2 Merilne naprave

Pri fizikalnih eksperimentih pogosto uporabljamo merilne naprave, s katerimi želimo potrditi teoretične principe in fizikalne pojave. V šolskem prostoru najpogosteje uporabljamo Vernier-jevo merilno napravo LabQuest Hardware “LabQuest® Hardware and Specifications | Vernier” , ki nam omogoča raznovrstne meritve.

2.1 Vernier’s DAQs



Slika 2.1: Vernier-jeva merilna naprava LabQuest2

Poglejmo si nekaj podatkov o napravi:

- **Display**
  - 11.2 cm x 6.7 cm (13.1 cm diagonal) screen

  - 00 x 480 pixel color display at 188 dpi
  - ED backlight
  - ortrait or landscape screen orientation
  - igh-contrast mode for outdoor visibility

- **Processor**
  - 800 MHz Application Processor

- **Connectivity**
  - Wi-Fi 802.11 b/g/n @ 2.4GHz
  - Bluetooth Smart for WDSS and Go Wireless Sensors

- **User Interface**
  - Resistive touch screen
  - Touch and stylus navigation for efficiency and precision

- **Data Acquisition**
  - 100,000 samples per second
  - 12-bit resolution
  - Built-in GPS, 3-axis accelerometer, ambient temperature, light, and microphone

- **Environmental Durability**
  - Operating Temperature: 0 + 45°C
  - Storage Temperature: -30 + 60°C
  - Splash resistant
  - Rugged enclosure designed to withstand a fall from lab bench

- **Size and Weight**
  - Size: 8.8 cm x 15.4 cm x 2.5 cm
  - Weight: 350 g

- **Ports**
  - 5 sensor channels
  - USB port for sensors, flash drives, and peripherals
  - USB mini port
  - DC power jack
  - MicroSD/MMC slot
  - Audio in and out

- **Storage**
  - 200 MB
  - Expandable with MicroSD and USB flash drive
- **Power**
  - Rechargeable, high-capacity battery
  - DC charging/powering through external adapter (included)
- **cena:**
  - $455

Seveda pa morate dokupiti še senzorje, ki tudi niso cenovno ugodni, saj se njihova cena giblje od $30 … naprej.

2.2 Arduino Data Acquisition System

On the market we can find different DAQ systems which are hi-end products and often expencive (from 100 € .. n k€). Buy we can use Arduino (Uno, nano, …) as low-cost data acqusition system if we do not need scientific accurate data (for pedagogical purposes).

<– Parts of DAQ[[NationalInstruments]] –>

Na plošči vsebuje mikrokrmilnik **Atmega328**, ki lahko opravi podobne naloge, kot smo jih opisali v poglavju **Merilne naprave**.

Poglejmo si nekaj karakteristik tega mikrokrmilnika (Atmel 2017):

- **Advanced RISC Architecture**
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- **High Endurance Non-volatile Memory Segments**
  - 32KBytes of In-System Self-Programmable Flash program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM

  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    * In-System Programming by On-chip Boot Program
    * True Read-While-Write Operation
  - Programming Lock for Software Security

- **Atmel® QTouch® Library Support**
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix® Acquisition
  - Up to 64 sense channels

- **Peripheral Features**
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - ADC
    * 8-channel 10-bit ADC in TQFP and QFN/MLF package
    * Temperature Measurement
    * 6-channel 10-bit ADC in PDIP Package
    * 10-bit Resolution
    * 0.5 LSB Integral Non-Linearity
    * ±2 LSB Absolute Accuracy
    * 13 - 260us Conversion Time
    * Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
    * Six Multiplexed Single Ended Input Channels
    * Two Additional Multiplexed Single Ended Input Channels (TQFP and VFQFN Package only)
    * Temperature Sensor Input Channel
    * Optional Left Adjustment for ADC Result Readout
    * 0 - VCC ADC Input Voltage Range
    * Selectable 1.1V ADC Reference Voltage
    * Free Running or Single Conversion Mode
    * Interrupt on ADC Conversion Complete
    * Sleep Mode Noise Canceler
  - Two Master/Slave SPI Serial Interface
  - One Programmable Serial USART

  - One Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - One On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change

- **Special Microcontroller Features**
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

- **I/O and Packages**
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

- **Operating Voltage:**
  - 1.8 - 5.5V

- **Temperature Range:**
  - -40°C to 105°C

- **Speed Grade:**
  - 0 - 4MHz @ 1.8 - 5.5V
  - 0 - 10MHz @ 2.7 - 5.5V
  - 0 - 20MHz @ 4.5 - 5.5V

- **Power Consumption at 1MHz, 1.8V, 25°C**
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1uA
  - Power-save Mode: 0.75uA (Including 32kHz RTC)

2.3 Requirements

(Checked is required, unchecked is optional)

Hardware:

☒ Computer

- ☒ Arduino board (Arduino UNO, Arduino LEONARDO, Arduino NANO, clone products)
- ☒ basic electronics components for sensors
- ☐ arduino set starter for example use this kit
- ☐ arduino sensors set for example use this kit

Software:

- ☒ Arduino IDE download here
- ☐ Python (>=3.0)
- ☐ pyserial
- ☐ Ms Excel

2.4 Pros & Cons

Pros	Cons
+ Price (Arduino ~3€, sensors ~2-5€)	- work in progress
+ Accessibility	- no plug&play solutions
+ Versatility	- DIY project
+ Easy importing data into MS Excel	- low sampling rate ~6kHz
+ točne časovne meritve	- 10-bi resolutuin

More about DAQ you can read in further reading…

> Kaj ko bi si lahko naredili svojo merilno napravo?

V ta namen smo ustvarili nekaj vsebin na portalu GitHub…

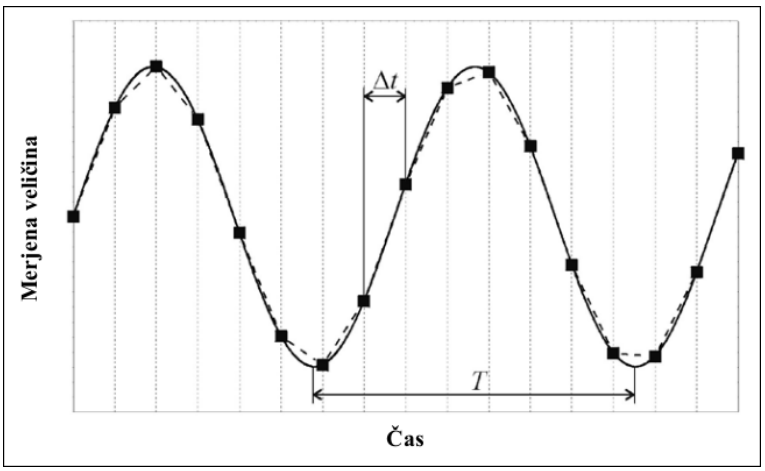
3 Teoretične osnove

3.1 Frekvenca vzorčenja

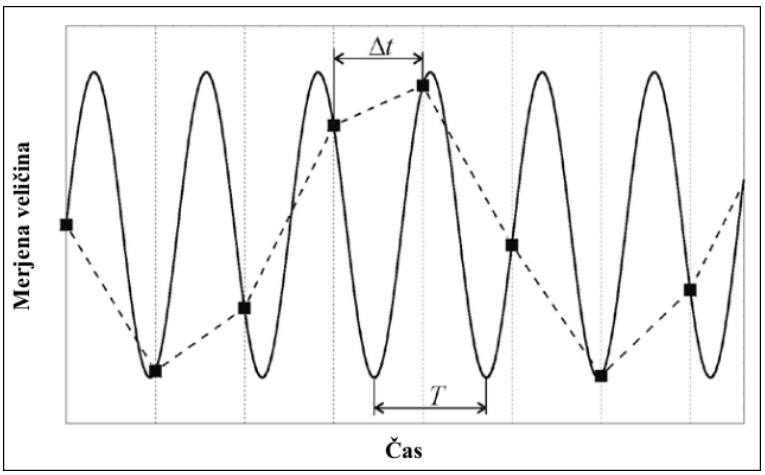
Pri vzorčenju signalov je zelo pomembna frekvenca vzorčenja f_{vz}. Upoštevati moramo Nyqistovo načela vzorčenja, ki pravi, da je potrebno periodične signale vzorčiti vsaj z 2x večjo frekvenco vzorčenja kot je frekvenca signala f_{sig} (Nyquist 1928), kot to predstavlja slika sl. 3.1.

$$f_{vz} = 2f_{sig} \tag{3.1}$$

V nasprotnem primeru laho dobimo nepravilno reprodukcijo merjenega signala (črtkana krivulja), kot to prikazuje slika sl. 3.2.



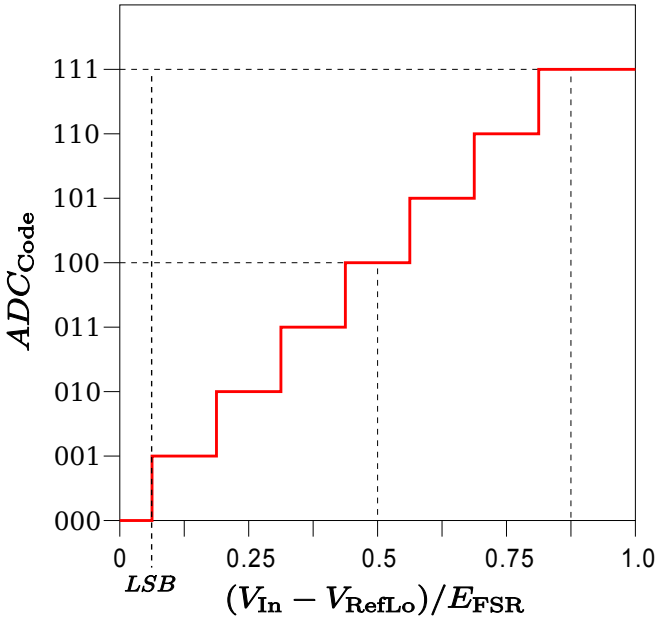
Slika 3.1: Pravilno vzorčenje sinusnega signala.



Slika 3.2: Primer reprodukcije (črtkana krivulja) podvzorčenega signala.

3.2 Digitalizacija

Merilni sistemi so opremljeni s t.i. analogno-digitalnimi pretvorniki (ang.: Analog-to-digital converter - ADC), ki pretvarjajo merjeno napetost v neko številsko vrednost. Zelo pogost primer je, ko zvezno napetostno območje od $0,0V - 5,0V$ pretvorimo v številske vrednosti od 0 - 1024. Pri tej pretvorbi ključno vlogo prevzame ADC in njegova **resolucija**. Grafični prikaz take transformacije je prikazan na sliki sl. 3.3.



Slika 3.3: Prenosna funkcija ADC pretvorbe (Contributors 2019a).

3.3 Resolucija in ločljivost

Resolucija AD pretvornikov je določena s številom vseh možnih stanj pretvorbe N. Ker so AD pretvorniki napreave prirejene digitalnim tehnologijam, se njihovi podatki izražajo v dvojiški obliki (binarno). Tako naprimer AD pretvornik z 10-bitno pretvorbo lahko prikaže:

$$N = 2^B \tag{3.2}$$

možnih stanj.

Ločljivost pa je najmanjša razlika med sosednjima digitaliziranima vrednostima merjene količine. Ta vrednost je odvisna tako od števila možnih stanj N, kakor tudi od območja, ki ga pretvarjamo. Zato bi lahko enačbo en. 3.3 zapisali:

$$Locljivost = \frac{Obmocje}{N} \tag{3.3}$$

3.3.1 NALOGA: Izračun frekvence, resolucije in ločljivosti AD pretvorbe

Glede na prejšnje podatke o mikrokrmilniku Atmega328 poiščite podatek o najvišji frekvenci vzorčenja f_{vz} analognih signalov in izračunajte najmanjši čas Δt med dvema vzorčenjema.

Izračunajte s kolikšno resolucijo lahko odčitavamo analogne signale z mikrokrmilnikom Atmega328.

Izračunajte kolikšna je ločljivost mikrokrmilnika Atmega328 pri odčitavanju analognih signalov.

3.4 Točnost in natančnost

Točnost (v različnih virih je poimenovana različno, ang.: validity) je lastnost merilnega sistema, ki predstavlja ustreznost prestavljene meritve glede na njeno realno merjeno vrednost. Navadno jo izražamo kot relativno napako ϵ v procentualni obliki (en. 3.4):

$$\epsilon = \frac{(\mu - \bar{x})}{\mu} \tag{3.4}$$

Kjer je μ realna merjena vrednost (to je parameter) in $\bar{x}$ povprečna izmerjena vrednost.

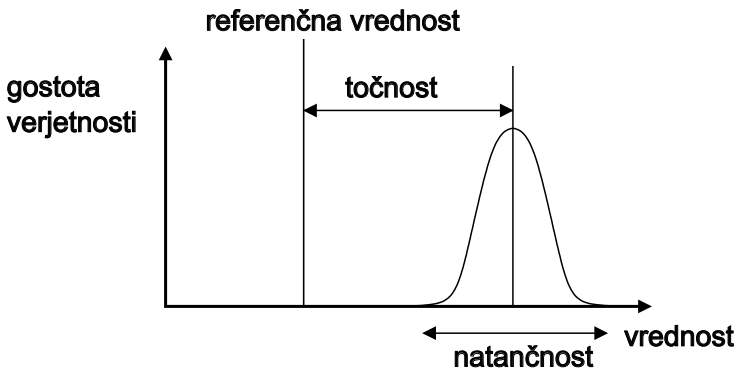
Lahko pa točnost izrazimo tudi v absolutni obliki (en. 3.5):

$$e = \mu - \bar{x} \tag{3.5}$$

kjer je $\bar{x}$ povprečna vrednost meritev in je tako statistično izmerjena količina (ni parameter).

Natančnost oz. Preciznost (zopet v različnih literaturah poimenovana različno, ang.: reliability) je sposobnost merilnega sistema reprodukcije iste merjene (referenčne) vrednosti z enakimi izmerjenimi vrednostmi. V mnogih primerih se izkaže, da gre v tem primeru za naključno napako merjenja in to vrednost lahko ponazarjamo s standardnim odklonom merilnega postopka (en. 3.10). V nekaterih primerih to vrednost podajamo tudi z intervalom zaupanja, pri katerem podamo tudi verjetnost meritve (en. 3.11).

V splošnem bi lahko točnost in natančnost predstavili z grafom na sliki [sl. 3.4](Contributors 2019c).



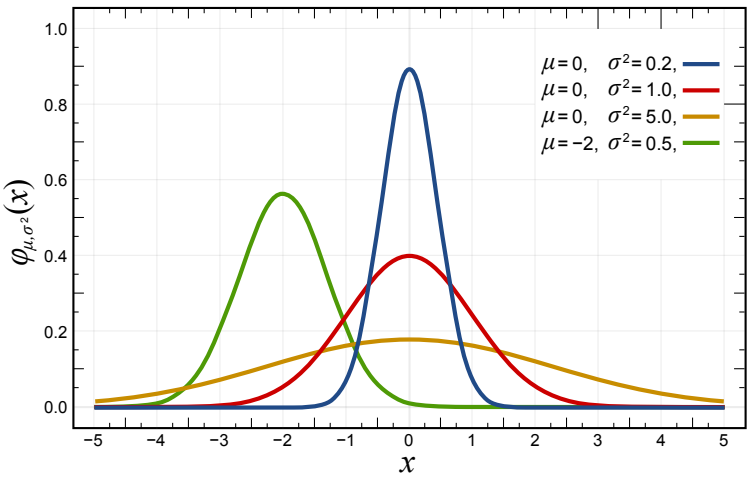
Slika 3.4: Točnost in natančnost mreitev.

3.5 Normalna porazdelitev

Kadar imamo v merilnem sistemu opravka z naključnimi napakami, meritve lahko predstavimo s krivuljo normalne porazdelitve - v splošenm imenovnane Gaussova porazdelitev. Zapišemo jo v obliki en. 3.6.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \tag{3.6}$$

Kjer je μ povprečna vrednost populacije in σ standardni odklon populacije. Nekaj različnih krivulj lahko vidimo na sliki sl. 3.5 (Contributors 2019b).



Slika 3.5: Primeri normalne verjetnostne porazdelitve.

Koeficienta o sploščenosti in premaknjenosti normalne porazdelitve lahko izračunamo tudi z različnimi računalniškimi programi za obdelavo razpredelnic, kot sta na primer Microsoft Excel ali LibreOffice Calc.

Sploščenost

Pričakovan koeficient sploščenosti je okoli 0. Če je vrednost izven območja $-2 < k < +2$ privzamemo, da porazdelitev ni normalno sploščena.

```
=KURT(Range)
```

Premaknjenost

Pričakovana vrednost premaknjenosti je okoli 0. Če je vrednost > 0.5 govorimo o pozitivni premaknjenosti in je porazdelitev vzorca nagnjena v levo (in obratno).

```
=SKEW(Range)
```

Povprečna vrednost populacije (en. 3.7) in vzorca (en. 3.8)

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} \tag{3.7}$$

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{n} \tag{3.8}$$

```
=AVERAGE(Range)
```

Standardni odklon populacije (en. 3.9) in vzorca (en. 3.10)

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{3.9}$$

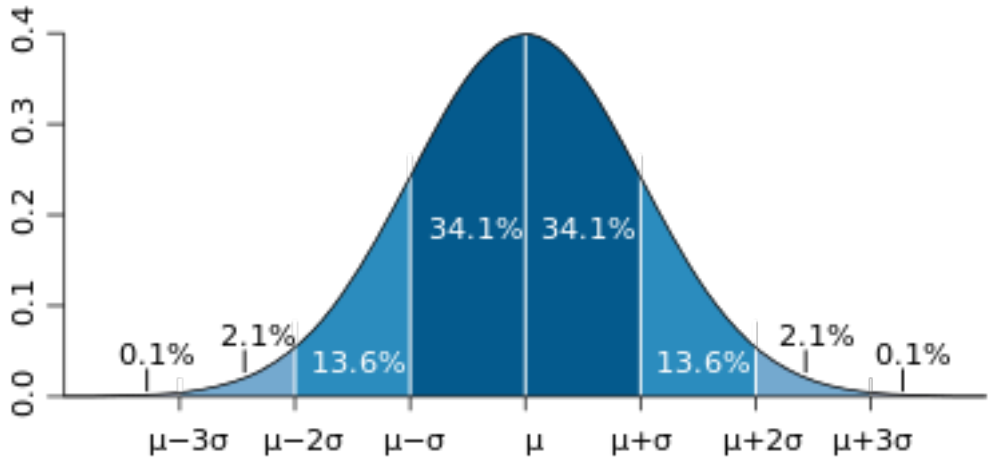
$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{3.10}$$

```
1    =STDEV(Range)
```

** .**

3.6 Ocenjevanje nepoznanega parametra μ

Iz grafa normalne porazdelitve, ki je prikazan na sl. 3.6, so razvidni deleži vsebovanih meritev v določenih območjih standardnih odkonov ($1\sigma, 2\sigma$ in 3σ) za celotno populacijo.



Slika 3.6: Graf normalne porazdelitve z vključujočimi deleži meritev.

Na primer izkaže se, da je v območju $\mu \pm 1\sigma$ kar 68% vseh meritev, v območju $\mu \pm 2\sigma$ jih je 95% in v območju $\mu \pm 3\sigma$ celo 99,7%.

Zato te iste verjetnosti veljajo tudi pri vzorčenju manjših vzorcev. Tako s **standardno napako ocene povprečne vrednosti ($s_{\bar{x}}$)** naših meritev lahko ocenimo interval ($\bar{x} \pm \alpha s_{\bar{x}}$) v katerem se dejanski parameter μ nahaja z neko verjetnostjo. Standardno napako ocene povprečne vrednosti lahko izračunamo po en. 3.11:

$$s_{\bar{x}} = \frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}}, \tag{3.11}$$

```
STDEV(A2:A6)/SQRT(COUNT(A2:A6))
```

kjer je sicer σ standardni odklon celotne populacije, ki ga pogosto ne poznamo in ga zato nadomestimo s standardnim odklonom vzorca s (en. 3.10). Korekturni faktor $\sqrt{\frac{N-n}{N-1}}$ uporabljamo le, če poznamo N celotne populacije in pri izredno velikih vzorcih ($n > \frac{N}{100}$).

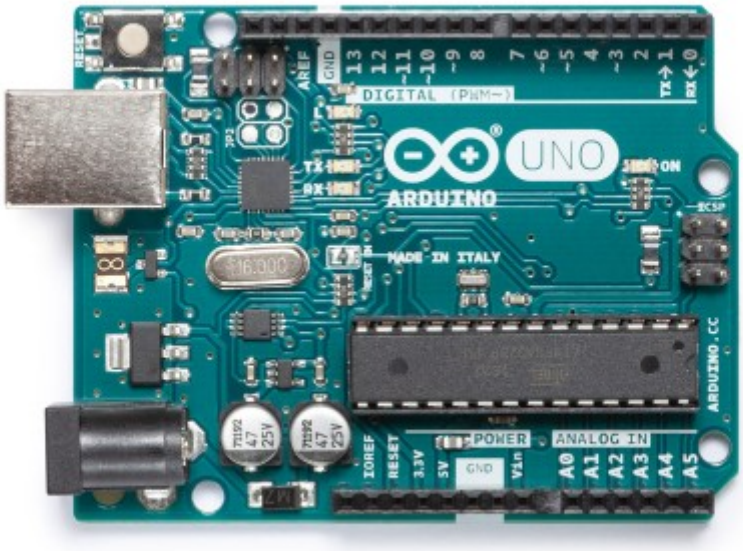
Tako območje $\bar{x} \pm \alpha s_{\bar{x}}$ imenujemo območje zaupanja. Najpogosteje se v praksi uporablja območje zaupanja s koef. $\alpha = 1,96$, s katerim pričakujemo 95,00% gotovost, da naša izmerjena povprečna vrednost $\bar{x}$ ustreza dejanskemu parametru μ.

```
=CONFIDENCE(Signif., Std.Dev., Sample Size)
```

4 Arduino UNO

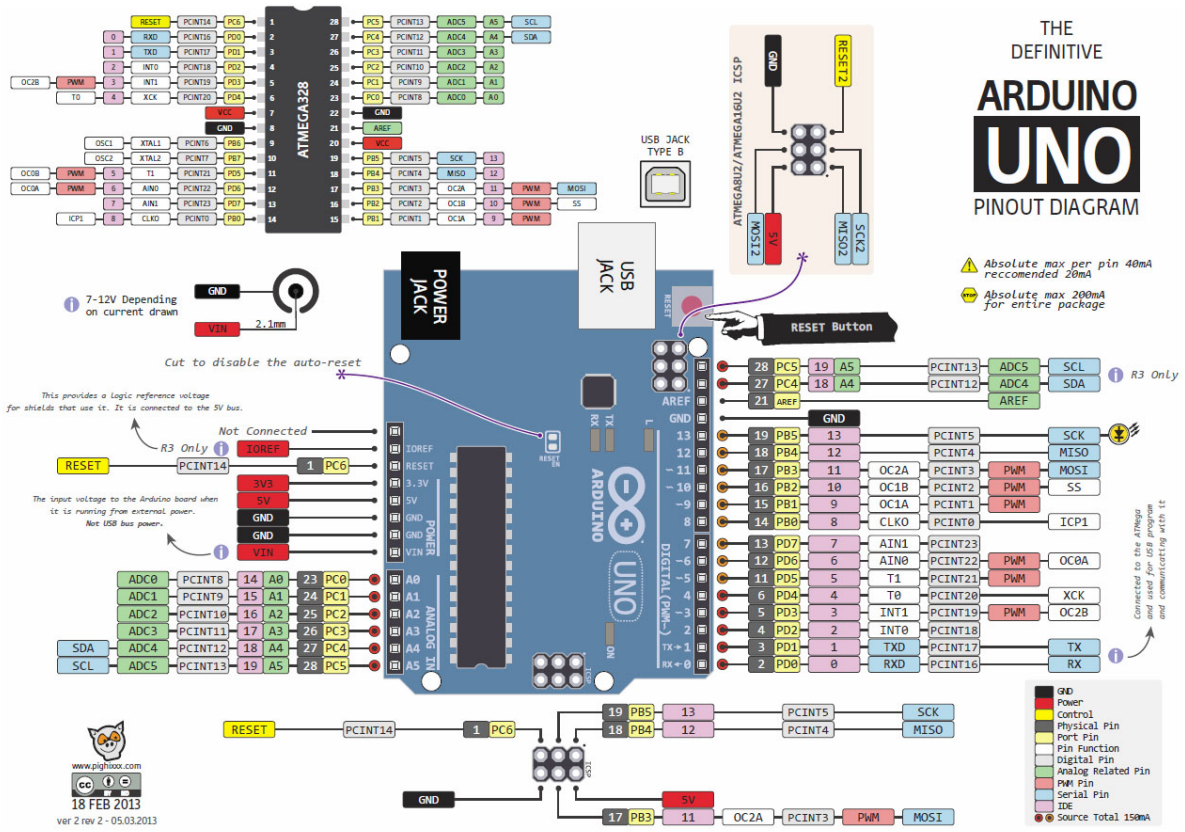
Arduino Uno je:

> The UNO is the best board to get started with electronics and coding. If this is your first experience tinkering with the platform, the UNO is the most robust board you can start playing with. The UNO is the most used and documented board of the whole Arduino family. “Arduino Uno Rev3”

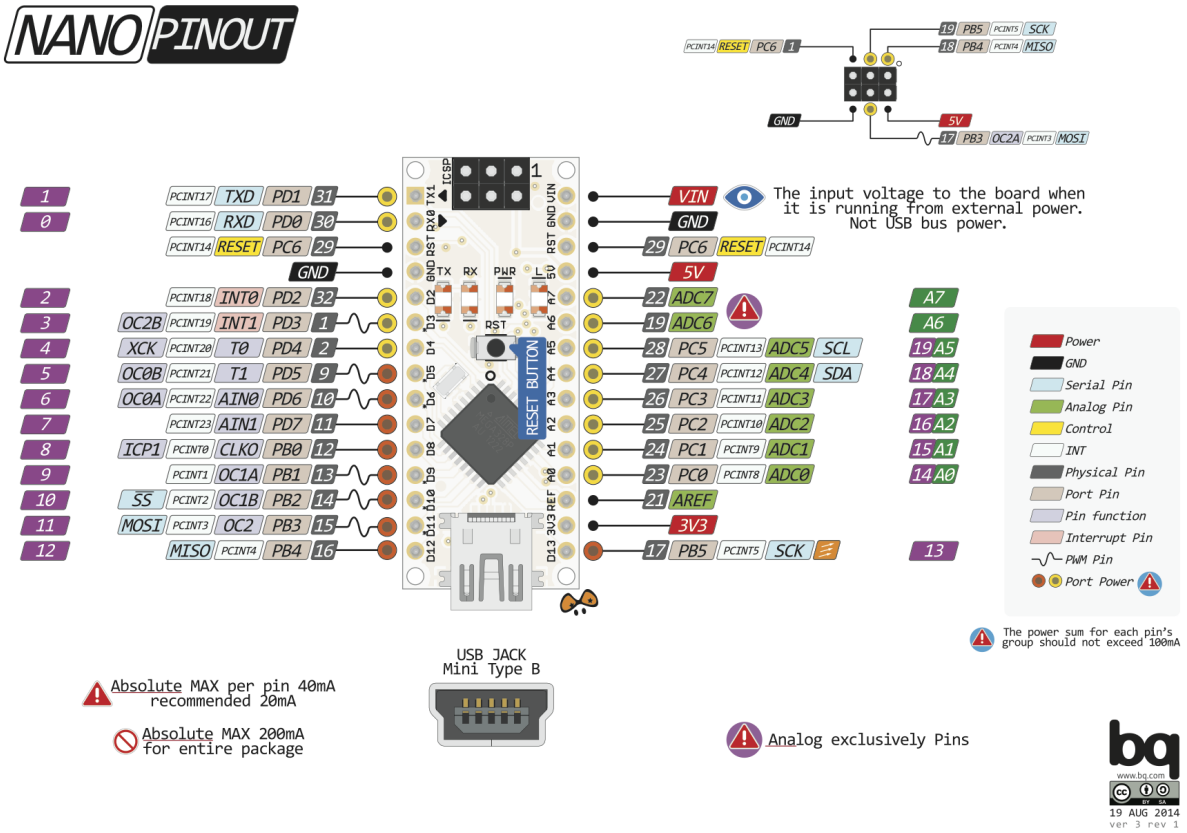


Slika 4.1: Arduino Uno

4.1 Arduino UNO/NANO pinout



Slika 4.2: Razporeditev priključkov na krmilniku Arduino UNO (Theman 2015).



Slika 4.3: Razporeditev priključkov na krmilniku Arduino NANO (Theman 2015).

Kot lahko opazimo, so funkcije priključkov na sliki sl. 4.2 in sliki sl. 4.3 enaki. To ni nenavadno, saj sta krmilnika po električni zgradbi enaka, razlikujeta se le v fizični izvedbi.

4.2 Osnove programiranja krmilnika

4.2.1 NALOGA: Nastavitve Arduino IDE programa

Pravilno priključite krminlnik Arduino na računalnik in nastavite programsko okolje Aruino IDE ter sprogramirajte krmilnik s testnim programom **BLINK.INO**. Iz nastavitev programskega okolja prepišite vaše nastavitve za:

1. krmilnik:_____,
2. komunikacijska vrata: _____,
3. mikrokrmilnik: _____.

Testni program BLINK.INO

Nastavite in delovanje krmilnika lahko preverimo s testnim programom **blink.ino**. Na krmilniku Arduino UNO je na priključek **13** priključena svetleča dioda prav za ta namen.

Program 4.1: Vzorčni program za krmilnik Arduino NANO.

```
1  void setup() {
2    pinMode(LED_BUILTIN, OUTPUT);
3  }
4
5  void loop() {
6    digitalWrite(LED_BUILTIN, HIGH);
7    delay(1000);
8    digitalWrite(LED_BUILTIN, LOW);
9    delay(1000);
10 }
```

4.2.2 NALOGA: Berljivost programske kode

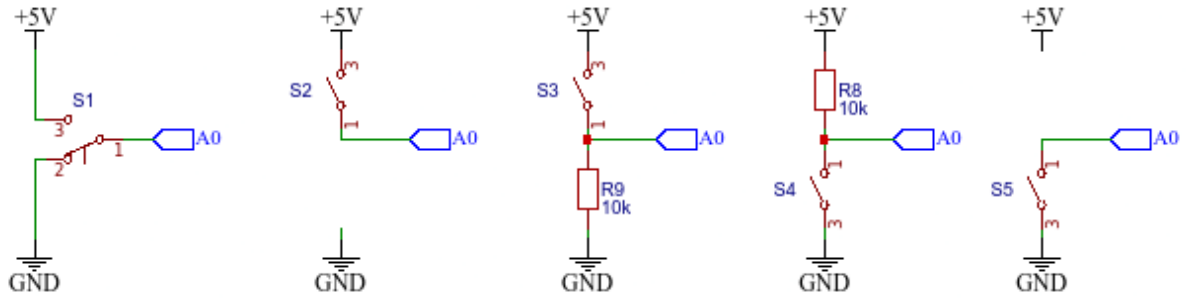
Spremenite prog. 4.1 tako, da bo čim bolj berljiv. Pri berljivosti programske kode lahko upoštevate neka smernic:

1. Skupek programskih ukazov združujte v funkcije, ki jih smiselno poimenujte.
    1. Iz programske strukture naj se opazi čemu je program namenjen.
    2. Če funkcija `void loop()` bolje opisuje namen programa, jo postavite pred `void setup()`.
    3. Funkcije naj vračajo in sprejemajo smiselne parametre.
2. Uporabljajte `razlagalne spremenljivke in konstante` namesto surovih številčnih vrednosti.
    1. Uporabljene priključke vedno poimenujte npr.: `int LED_PIN = 13;`.
    2. Vmesne rezultate in vrednosti smiselno poimenujte npr.: `int vrednost_senzorja = analogRead(SENZOR_PIN);`
3. Uporabljajte komentarje LE tam, kjer je to ZARES potrebno.
    1. Imena spremenljivk, konstant in funkcij naj pomagajo pri opisovanju kode, npr.: funkcija `delay(1000)` ne potrebuje nobenega komentarja.
    2. Komentarji lahko postanejo dodatni element, ki ga bo potrebno vzdrževati…

5 Uporaba digitalnih vhodov

Za digitalne spremembe kot so naprimer: pritisk na tipko, prehod predmeta mimo svetlobnih vrat… najpogosteje uporabljamo digitalne vhodne priključke krmilnika. Le-te lahko najdemo na priključkih **D0..D13** in tudi na **A0..A7**. Primer enostavne vezave tipke na krmilnik prikazuje slika sl. 5.1.

5.1 Različne možnosti vezave tipke



Slika 5.1: Vezava tipke na digitalni vhod krmilnika Arduino.

Program 5.1: Preverjanje stanja tipke (digitalnega vhoda).

```
const int TIPKA = A0;
const int LED = 13;

void loop(){
    bool tipka_je_pritisnjena = digitalRead(TIPKA);
    if (tipka_je_pritisnjena == 1)
        digitalWrite(LED, HIGH);
    else
        digitalWrite(LED, LOW);
}
void setup(){
    pinMode(TIPKA, INPUT);
    pinMode(LED, OUTPUT);
}
```

5.1.1 NALOGA: Zaznavanje digitalnih vrednosti

1. Preskusite delovaje vseh vezav tipke, ki so prikazani na shemi in komentirajte (ne-)delovanje.
2. Napišite program s katerim boste lahko zaznali spremembo digitalnega vhoda (pomagajte si s funkcijo, ki jo prikazuje prog. 5.2).

Program 5.2: Primer funkcije za zaznavanja spremembe na digitalnem vhodu.

```
1  void waitForInputChange(int INPUT_PIN){
2      bool zacetno_stanje = digitalRead(INPUT_PIN);
3      bool trenutno_stanje = zacetno_stanje;
4      while (trenutno_stanje == zacetno_stanje)
5          trenutno_stanje = digitalRead(INPUT_PIN);
6  }
```

6 Časovne meritve

Za preproste časovne meritve lahko uporabimo funkcijo `millis()`. Funkcija `millis()` vrne število milisekund od začetka zagona programa na krmilniku Arduino Uno.

Program 6.1: Beleženje časa.

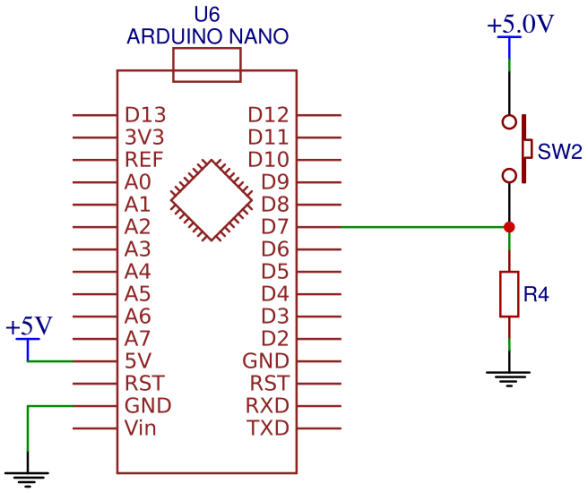
```
unsigned long time;

void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("Time: ");
  time = millis();

  Serial.println(time); //prints time since program started
  delay(1000);          // wait a second so as not to send massive
     amounts of data
}
```

Preverite tudi:

```
time = micros();
```

6.1 Časovni interval med dvema digitalnima spremembama

V povezavi s časovnimi meritvami pogosto uporabljamo digitalne vhode. Le-te lahko najdemo na priključkih **D0..D13** in tudi na **A0..A7**. Primer enostavne vezave tipke na krmilnik prikazuje slika sl. 6.1.



Slika 6.1: Vezava tipke na digitalni vhod krmilnika Arduino.

6.1.1 NALOGA: Merjenje časovnih intervalov

Napišite program za merjenje hitrosti človeškega odziva in zvežite vezje, ki ga prikazuje slika sl. 6.1. Po naključnem času naj zasveti lučka na krmilniku ArduinoUNO. Nato pa naj program nemudoma shrani trenutni čas v $start\_time$ in ko uporabnik pritisne tipko naj si program ponovno shrani čas v $stop\_time$. Program naj izračuna razliko časov in ga prikaže v na računalniku.
Napredno: Če znate program popravite tako, da ne bo omogočal goljufanja.

Program 6.2: Merjenje časovnega intervala.

```
const int TIPKA = 7;
void randomly_turn_LED_on();
void start_timing_user_respond();
void reset_the_game();

void loop() {
  randomly_turn_LED_on();
  start_timing_user_respond();
  reset_the_game();
}

void setup() {
  pinMode(LED_BUILTIN ,OUTPUT);
  pinMode(TIPKA       ,INPUT_PULLUP);
  Serial.begin(9600);
  Serial.println("Start...");
  randomSeed(analogRead(0));
}

void randomly_turn_LED_on(){
  digitalWrite(LED_BUILTIN,LOW);
  delay(random(5000,10000));
  digitalWrite(LED_BUILTIN,HIGH);
}

void start_timing_user_respond(){
  unsigned long start_time = micros();
  unsigned long stop_time = 0;

  while (digitalRead(TIPKA) == 0){
    stop_time = micros();
  }

  unsigned long time_div = stop_time - start_time;
  Serial.println(time_div);
}

void reset_the_game(){
  while (digitalRead(TIPKA) == 1){
    delay(200);
    Serial.println("spusti tipko...");
  }
  Serial.println("Start...");
}
```

6.2 Hitrost

6.2.1 NALOGA: Merjenje hitrosti predmeta (svetlobna vrata)

Uporabite ali sestavite dvojna svetlobna vrata. Nato napišite program, ki bo izmeril razliko v času ko se svetlobni snop na enih in drugih svetlobnih vratih prekine. Izmerite razdaljo med vratoma in izračunajte hitrost.
Napredno: Če zante program popravite tako, da bo občutljiv na “spremembo” vhodnega signala.

Program 6.3: Merjenje hitrosti predmeta.

```
const int VRATA_1 = 7;
const int VRATA_2 = 8;
void wait_for_gate_change(int input_pin);
void print_time_difference(unsigned long start_time,
                           unsigned long stop_time);
void loop() {
  wait_for_gate_change(VRATA_1);
  unsigned long start_time = micros();
  wait_for_gate_change(VRATA_2);
  unsigned long stop_time = micros();

  print_time_difference(start_time, stop_time);
  Serial.println("Nova maritev...");
}

void setup() {
  pinMode(VRATA_1, INPUT);
  pinMode(VRATA_2, INPUT);
  Serial.begin(9600);
  Serial.println("Start...");
}

void wait_for_gate_change(int input_pin){
  bool zacetna_vrednost  = digitalRead(input_pin);
  bool trenutna_vrednost = zacetna_vrednost;

  while (trenutna_vrednost == zacetna_vrednost){
    trenutna_vrednost = digitalRead(input_pin);
  }
}
void print_time_difference(unsigned long start_time,unsigned long
    stop_time){
  unsigned long time_diff = stop_time - start_time;
  Serial.println(time_diff);
}
```

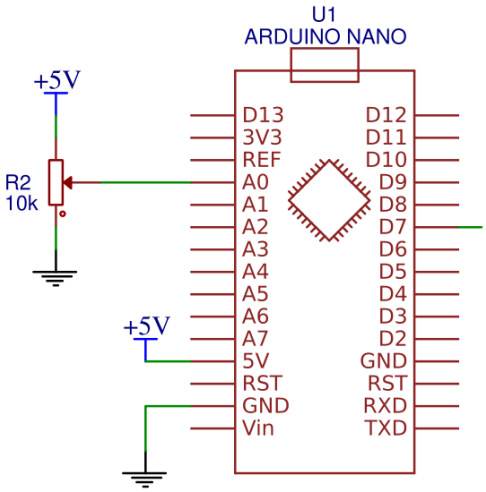
6.3 Pospešek

7 Merjenje napetostnih potencialov

Krmilnik Atmega328 ima vgrajen AD pretvornik, s katerim lahko odčitavamo analogne napetostne potenciale v območju $U_{ADC} = [0..5]V$.

7.1 Priključki za analogno odčitavanje napetostnih potencialov

Analogne napetostne potenciale lahko odčitavamo na priključkih krmilnika, ki so označeni z **A0..A7**. Zato moramo senzorje priključiti tako kot prikazuje slika sl. 7.1.



Slika 7.1: priključitev potenciometra na analogni priključek krmilnika.

7.1.1 NALOGA: Merjenje napetostnega potenciala

Zvežite vezje po shemi na sliki sl. 7.1 in sprogramirajte krmilnik tako, da boste na ekran računalnika izpisovali izmerjene vrednosti.
Napišite v kakšnem intervalu so bile izmerjene vrednosti:

ADC = [_____,_____]_.

7.2 Analogno-digitalni pretvornik

Program 7.1: Odčitavanje napetostnih potencialov z analogno-digitalnim pretvornikom.

```
const int POTENCIOMETER = 0;
void setup() {
    Serial.begin(9600);
}

void loop() {
    int adc_value = analogRead(POTENCIOMETER);
    Serial.println(adc_value);
    delay(100);
}
```

7.3 Izračun napetosti

7.3.1 NALOGA: Preračun ADC vrednosti v napetost

Napišite program za merjenje napetosti z ADC vmesnikom. V spodnji prostor pa vpišite le programske vrstice, ki ste jih uporabili za izračun napetosti.

7.4 Normalna porazdelitev meritev

7.4.1 NALOGA: Koeficienti normalne porazdelitve

Z Arduino krmilnikom izmerite 100 meritev neke poljubne napetosti. Nato te meritve preverite še z volt-metrom, kar naj predstavlja vašo referenčno vrednost. Meritve vnesite v program za delo s tabelami in z ustreznimi funkcijami izračunajte, rezultat pa vpišite na črte:

- izmerjena referenčna vrednost: _____,

- povprečno vrednost meritev : _____,
- točnost predstavite z absolutno napako :_____,
- preciznost meritev pa podajte s standardno napako
  ocene povprečne vrednosti za 95% verjetnost :_____.

Program 7.2: Izračun napetosti.

```
const int POTENCIOMETER = 0;
void get_100_Samples();

void setup() {
  Serial.begin(9600);
  get_100_Samples();
}

void loop() {
}

void get_100_Samples(){
  for (int i = 0; i < 100; i++){
    int adc_value = analogRead(POTENCIOMETER);
    float voltage = (float)adc_value * 5 / 1023;
    Serial.println(voltage, 4);
    delay(10);
  }
}
```

8 Izpis podatkov

Prikazovanje podatkov je ena od ključnih nalog vseh merilnih sistemov. To lahko naredimo na različne načine:

1. v obliki tabele na zaslon računalnika,
2. v grafični obliki,
3. na zaslon samostojne merilne naprave,
4. shranimo podatke na SD spominsko kartico …

8.1 Serijski izpis

Spodnji primer programske kode prikazuje kako lahko z Arduino krmilniki izpisujemo osnovne podatke neke meritve. Izpisovali bomo dva podatka v ločena stolpca. Stolpca bosta ločena s tabulatorjem, kar predstavlja označba **\t**. V prvem stolpcu bomo izpisovali trenutni čas (**millis();**), v drugega pa vrednost analogne pretvorbe (**analogRead(0);**).

Program 8.1: Serijski izpis podatkov.

```
1  void setup() {
2    Serial.begin(9600);
3    Serial.println("t[ms]\tADC");
4  }
5
6  void loop() {
7    Serial.print(millis());
8    Serial.print("\t");
9    Serial.println(analogRead(0));
10   delay(100);
11 }
```

8.2 Grafični izris

Zelo nazorno je spremljati izris podatkov v grafičnem načinu. Do neke mere nam to funkcijo ponuja programsko okolje Arduino IDE, vendar v zelo okenjeni obliki. Preskusite naslednji program:

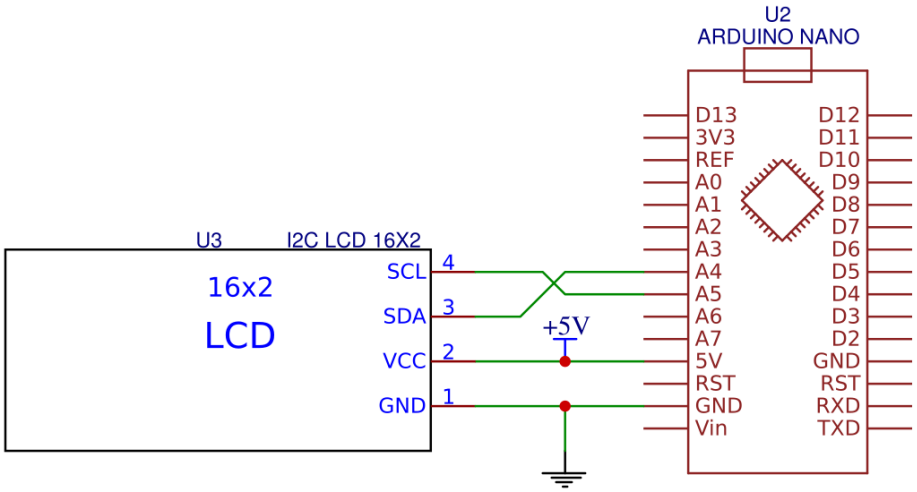
Program 8.2: Izrisovanje vrednosti s serijskim grafikonom.

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("0,1023,");
  Serial.println(analogRead(A0));
  delay(10);
}
```

Kot lahko opazimo računalniku pošiljamo 3 različne števila: 0, 1023 in ADC vrednost. Prvi dve nam slušita kot območje, saj bi v nasprotnem primeru program sam prilagajal skalo na y osi (ang.: autofit).

8.3 Izpis na LCD

Če želimo narediti merilno napravo, ki bo delovala samostojno, bomo verjetno potrebovali dodaten ekran za prikaz izmerjenih vrednosti. Zato ga moramo priključiti na krmilnik Arduino, kot kaže slika sl. 8.1.



Slika 8.1: Vezava I^2C LCD-ja na krmilnik Arduino.

Ena naj enostavnejših rešitev je, da krmilniku Arduino UNO dodamo LCD ekranček z I^2C vodilom. Ker je I^2C vodilo zasnovano tako, da nanj lahko priključimo več naprav, moramo za vsako napravo poznati njen naslov. Le tega lahko pridobimo od proizvajalca naprave, ga nastavimo sami ali pa z ustreznim programom pregledamo vse naslove priključenih naprav (programska koda(Arduino 2019)).

Za uporabo LCDja na podatkovnem vodilu I^2C bomo potrebovali knjižnico **LiquidCrystal_I2C**, ki si jo lahko presnamete s spletne strani johnrickman. Knjižnico lahko presnamete tako, da kliknete na gumb `Code \bigtriangledown` in nato še `Download zip`. V Arduino IDE pa `Tools` -> `Library Manager` -> `Add zip Library` in dodate pravkar presneto zip datoteko.

Program 8.3: Izpis podatkov na LCD prikazovalnik.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C DaqLcd(0x27, 16, 2);

void setup() {
  DaqLcd.init();
  DaqLcd.backlight();
  DaqLcd.clear();
}

void loop() {
  int i = 0;
  long ch = 0;
  for (i=0;i<64;i++){
    ch += analogRead(7);
  }

  float val = ch/64.0;
  DaqLcd.clear();
  DaqLcd.print(val);
}
```

8.3.1 NALOGA: Izpis podatkov

Prikažite podatke na različne načine:
1. v obliki tabele,
2. v grafični obliki in
3. na LCD zaslon merilne naprave.

8.4 Shranjevanje podatkov na SD spominsko kartico

Za krmilnike Arduino Uno (in druge) lahko dokupimo module z režami za SD spominsko kartico. En takih je modul za vzorčenje in beleženje meritev je (angl.:) Data Logger Shield. Modul vsebuje vtičnico za SD kartico, kot tudi bolj točen časovni oscilator (angl.: RTC), s katerim lahko merimo čas na nekaj sekund točno na mesec.

Pri vnašanju podatkov v datoteko je smiselno, da izberemo način vnosa podatko, kjer so podatki ločeni z vejico. Tako oblikovane podatke lahko shranimo v datoteko s končnico `csv` (angl.: comma separated values). Primer programske za vzorčenje in shranjevanje podatkov s pomočjo programske knjižnice `SD.h` je relativno enostaven.

```
#include <SD.h>
#include <SPI.h>
// define constants for the pin numbers
#define chipSelect 10
#define dataPin 11
void setup() {
 // Initialize serial port for communication with PC
 Serial.begin(9600);
 // Initialize SD card
 Serial.print("Initializing SD card...");
 if (!SD.begin(chipSelect)) {
  Serial.println("initialization failed!");
  return;
 }
 Serial.println("initialization done.");
 // create a new file
 File dataFile = SD.open("datalog.csv", FILE_WRITE);
 // write to the file
 if (dataFile) {
 dataFile.println(\"Data Logging Start!");
 dataFile.close();
 }
}

void loop() {
 // log data
 int data = analogRead(dataPin);
 Serial.println(data);
 // open the file. note that only one file can be open at a time,
 // so you have to close this one before opening another.
 File dataFile = SD.open("datalog.txt", FILE_WRITE);
 // if the file is available, write to it:
 if (dataFile) {
  dataFile.print(millis());dataFile.print(",");dataFile.println(data);
  dataFile.close();
 }
 delay(1000);
}
```

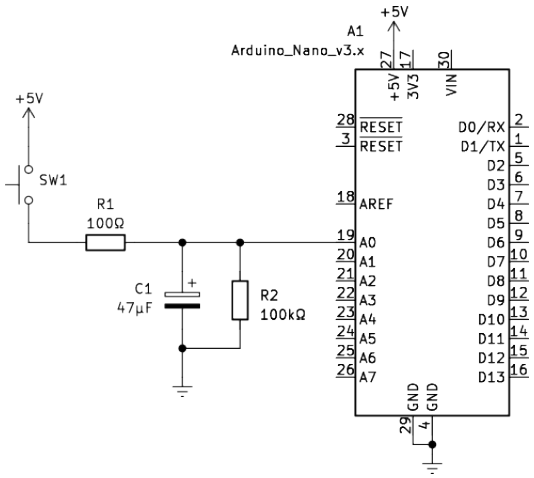
9 Linearizacija meritev (praznenje kondenzatorja)

Linearizacija je postopek, ki se uporablja za preoblikovanje ne-linearnega sistema v linearno obliko. To je uporabno za merjenje in analizo signalov, saj je lažje zaznati in napovedati linearne modele kot ne-linearne. Linearizacija omogoča uporabnikom, da preprosto in natančno določijo vrednosti signalov in njihovih karakterističnih konastank, ki jih vključujejo.

9.1 Vezje in zajem podatkov

Izvedli bomo poskus, pri katerem bomo lahko meritve odvisne spremenljivke (napetost) opravili pri različnih vrednostih neodvisne spremenljivke (čas). Iz rezultatov bomo skušali izluščiti kapaciteto kondenzatorja (kot iskano konstatnto sistema). Ker je fizikalni pojav nelinearen, ga bomo najprej linearizirali, in šele nato izračunali iskano konstanto (kapaciteto kondenzatorja). Tako lahko v izračun enostavno vključimo vse meritve…

Vezje sestavimo po sliki sl. 9.1:



Slika 9.1: Vezava vezja za zajem napetosti na kondenzatorju.

9.2 Linearizacija

Naloga linearizacijskega postopka je, da enačbo, ki opisuje dogajanje v sistemu (npr: en. 9.2) preoblikujemo v lineatrno obliko z en. 9.1.

$$f(x) = kx + n \tag{9.1}$$

$$U_C(t) = U_0 e^{-\frac{t}{RC}} \tag{9.2}$$

$$lnU_0 = -\frac{1}{RC}t + lnU_C \tag{9.3}$$

ali ...

$$ln(\frac{U_C}{U_0}) = -\frac{1}{RC}t \tag{9.4}$$

9.2.1 NALOGA: Linearizacija meritev (praznjenje kondenzatorja)

Izvedite poizkus praznjenja kondenzatorja in izmerite $U_C(t)$. Podatke linearizirajte tako, da jim boste lahko dodelili regresijsko premico.
Izračunajte:
- regresijski koeficient: $R\check{s}$:_____,
- enačba smernega koeficienta lienaricacije:
- vrednost smernega koeficienta:
- vrednost kapacitete kondenzatorja:

9.3 Koeficienti regresijske premice

Smerni koeficient premice

```
1    =SLOPE(y-Range,x-Range)
```

Začetna vresnost

```
1    =INTERCEPT(y-Range,x-Range)
```

Pearsonov korelacijski koeficient

Predstavlja velikost linearne povezanosti spremenljivk x in y, merjenih na istem predmetu preučevanja.

```
=PEARSON(y-Range,x-Range) //ali
=CORREL(y-Range,x-Range)
```

Determinacijski koeficient

Predstavlja delež pojasnjene variance spremenljivke za primer linearne regresije (r^2)

```
=RSQ(y-Range,x-Range)
```

Standardna napaka predvidenih vrednosti

Vrne standardno napako predvidenih vrednosti y za vsak x v regresiji. Standardna napaka je mera za obseg napake v napovedi y-a za posamezni x.

```
STEYX(y-Range,x-Range)
```

10 Senzorji

- toleranca uporov

10.1 Občutljivost

Ob predpostavki, da je določena meritev linearna na celotnem merilnem področju merilnega instrumenta, lahko izrazimo občutljivost instrumenta kot kvocient med spremembo izmerjene in spremembo merjene spremenljivke. Občutljivost analognega instrumenta določa razmerje med linearnim pomikom indikatorja lege in spremembo merjene spremenljivke, ki omenjeni pomik povzroči.

10.2 Delilniki napetosti s spremenljivim uporom

Pri takih vrstah senzorjev je izrednega pomena, da izberemo primeren referenčni upor R_{Ref}, saj le-ta vpliva na občutljivost senzorja. Izhodno napetost za temperaturni senzro s termistorjem in referenčnim uporom lahko izračunamo po enačbi en. 10.1.

$$U_{izh} = \frac{U_0 R_{Ref}}{R_{Ref} + R_{NTC}} \tag{10.1}$$

Če bomo ta temperaturni senzor uporabljali v temperaturnem intervalu $T \in [T_{min}, T_{max}]$, se bo njegova upornost spreminjala v intervalu $R_{NTC} \in [R_{NTC_{max}}, R_{NTC_{min}}]$. Ker želimo, da bi senzor imel v tem območju kar največjo občutljivost lahko sestavimo enačbo en. 10.2, za spremembo izhodne napetosti pri intervalu T.

$$\Delta U_{izh} = U_{izh}(T_{max}) - U_{izh}(T_{min}) \tag{10.2}$$

Če poiščemo maksimum funkcije enačbe en. 10.2 za R_{Ref} se izkaže, da je največja občutljivost senzorja pri referenčnem uporu, ki ga podaja enačba en. 10.3:

$$R_{Ref} = \sqrt{R_{NTC_{min}} R_{NTC_{max}}} \tag{10.3}$$

Delilnik napetosti z notranjim uporom

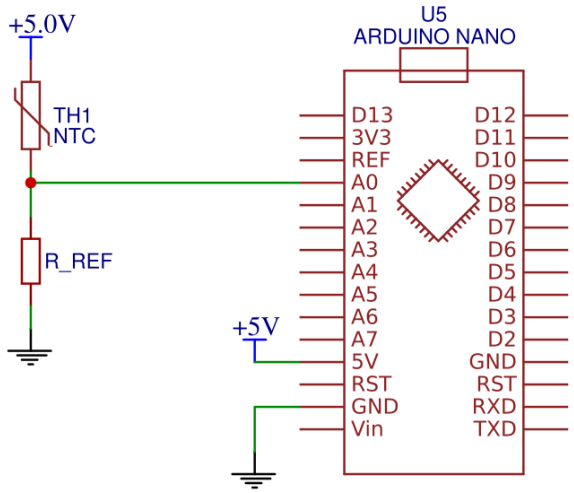
Delilnik napetosti lahko Naredimo tudi z uporom, ki ga mikrokrmilnik že vsebuje. Ker je ta upor vezan proti napajanju ga imenujemo **pull-up resistor**. V našem primeru ima ta upor upornost $R_{PULL-UP} \approx 10k\Omega$. Ker njegova vrednost ni točna je manj primeren za uporabo pri analognih senzorjih, vendar za izdelavo digitalnih lahko nadomesti nepotrebne vezave.

10.3 Načrtovanje temperaturnega senzorja

- glej poglavje o delilnikih
- naredite temp. senzor, katerega odziv bo največji za temperaturno območje $T_{NTC} = [15, 95]\check{r}C$.

10.3.1 NALOGA: Načrtovanje temp. senzorja

Podrobno si poglejte sec. 10 in dimenzionirajte temperaturn sentor z NTC termistorjem za območje $T = [15, 95]\check{r}C$. Izračunajte ali izmerite ključne vrednosti za dimenzioniranje senzorja:
$R_{NTC,T=15\check{r}C} =$_____,
$R_{NTC,T=95\check{r}C} =$_____,
$R_{REF} =$_____.



Slika 10.1: Vezava temperatunega senzorja.

11 Umerjanje (temperaturnega) senzorja

Zaradi različnih dejavnikov je potrebno senzorje predhodno umeriti. Umeritvene postopke pogosto podajo proizvajalci senzorjev, lahko pa ga storite sami. Preprost in univerzalen postopek je, da si zabeležite pare meritev vrednosti senzorja z merjenimi vrednostmi.

11.1 Umeritveni postopek

11.1.1 NALOGA: Umeritveni postopek

Naredite umeritveni postopek in predstavite tabelo s podatki.

T[°C] | **U[V]**

Program 11.1: Umerjanje senzorja.

```
const int TEMPERATURE_SENSOR = A0;
void wait_for_user_to_entry_data();
void print_average_value_of(int input_pin);

void loop() {
  wait_for_user_to_entry_data();
  print_average_value_of(TEMPERATURE_SENSOR);
}

void setup() {
  Serial.begin(9600);
  Serial.println("##########################################");
  Serial.println("# INSTRUCTIONS:                          #");
  Serial.println("#   1. Insert temperature (example: 22.4) #");
  Serial.println("#   2. then press ENTER                  #");
  Serial.println("#   3. repeat steps 1. and 2. ...        #");
  Serial.println("# When you are done with calibration:    #");
  Serial.println("#   4. Copy data to spreadsheet.         #");
  Serial.println("#   5. Add some polynome to the points.  #");
  Serial.println("#   6. Insert coeficients into file:     #");
  Serial.println("#       'temperature.ino'                #");
  Serial.println("##########################################");
  Serial.print("T°[C]");Serial.println("\tADC Value");
}

void wait_for_user_to_entry_data(){
  bool data_entry_is_complete = false;
  while (!data_entry_is_complete){
    if (Serial.available()){
      char inChar = Serial.read();
      if (inChar == '\n')
        data_entry_is_complete = true;
      else
        Serial.print(inChar);
    }
  }
}

void print_average_value_of(int input_pin){
  Serial.print("\t");
  long ADC_value = 0;
  for (int i = 0; i < 128; i++){
    ADC_value += analogRead(input_pin);
    delay(1);
  }
  float avg_ADC_val = (float)ADC_value/128;
  Serial.println(avg_ADC_val);
}
```

12 Interpolacija

Interpolácija je matematični postopek s katerim določimo približno vrednost funkcije znotraj obsega znanih vrednosti dveh neodvisnih spremenljivk.

12.1 Prirejanje polinoma

12.1.1 NALOGA: Interpolacija meritev

Dobljene podatke vnesite v programsko orodje za delo s tabelami in podatke izrišite v grafični obliki. Nato dodajte ustrezen trend in izpišite enačbo prenosne funkcije.
Prenosna funkcija:

12.2 Izpis temperature (stand-alone DAQ)

12.2.1 NALOGA: Prirejanje polinoma n-te stopnje

V programskem orodju za tabele (Microsoft Excel ali LibreOffice Calc) podatkom priredimo ustrezno krivuljo in odčitamo koeficiente… Nato napišemo program:

Program 12.1: Izračun vrednosti polinomske funkcije.

```
//              k0        k1        k2        k3        k4      k5
float k[6] = { -74.9, 531E-3, -1.68E-3, 3.25E-6, -3.12E-9, 1.22E-12};
const int TEMPERATURE_SENSOR = A0;
float get_average_value_of(int input_pin);
float calculate_temperature(float avg_ADC_val);

void loop() {
  float avg_ADC = get_average_value_of(TEMPERATURE_SENSOR);
  float avg_tmp = calculate_temperature(avg_ADC);
  Serial.println(avg_tmp);
  delay(1000);
}

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

float get_average_value_of(int input_pin){
  long ADC_value = 0;
  for (int i = 0; i < 128; i++) {
    ADC_value += analogRead(input_pin);
    delay(1);
  }
  result = (float)ADC_value / 64.0;
}

float calculate_temperature(float avg_ADC_val){
  float Temperature = 0;
  for (int i = 0; i <= 5; i++) {
    Temperature += k[i] * pow(avg_ADC_val, i);
  }
  result = Temperature;
}
```

13 Projektna naloga

> Sestavite poljuben fizikalni eksperiment, ga opremite s primernimi senzorji in podatke zajemite s krmilnikom Arduino. V primernim programskem okolju podatke obdelajte in jih številsko in grafično prikažite.
> Dokumentirajte:
> 1. kratek opis eksperimenta,
> 2. vezje,
> 3. postavitev eksperimenta,
> 4. programsko kodo,
> 5. meritve,
> 6. obdelavo podatkov in
> 7. predstavitev rezultatov.

13.1 Točkovanje

- Vse naloge od 1 .. 15 se točkujejo s $t_{1..15} = [0, 1, 2]$ točke.

- Naloga 16 pa se točkuje s $t_{16} = [0, 5, 10]$ točkami za vsako od postavk:
  - Naloga: **16.1** (kratek opis eksperimenta) =
  - Naloga: **16.2** (vezje) =
  - Naloga: **16.3** (postavitev eksperimenta) =
  - Naloga: **16.4** (programsko kodo) =
  - Naloga: **16.5** (meritve) =

- Naloga: **16.6** (obdelavo podatkov in) =

- Naloga: **16.7** (predstavitev rezultatov =

13.2 Točke:

$t_{1..15} =$..... $/30$
$t_{16} =$..... $/70$
Skupaj:
$t_{RPF} =$..... $/100$

Viri in literatura

Arduino. 2019. “Arduino Playground - I2cScanner.” 2019. https://playground.arduino.cc/Main/I2cScanner/.

“Arduino Uno Rev3.” https://store.arduino.cc/usa/arduino-uno-rev3.

Atmel. 2017. “ATMEGA328P.” 2017. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

Contributors, Wikipedia. 2019a. “Analog-to-Digital Converter.” 2019. https://en.wikipedia.org/wiki/Analog-to-digital_converter.

———. 2019b. “Normal Distribution.” 2019. https://en.wikipedia.org/wiki/Normal_distribution.

———. 2019c. “Točnost in Natančnost.” 2019. https://sl.wikipedia.org/wiki/To%C4%8Dnost_in_natan%C4%8Dnost.

“LabQuest® Hardware and Specifications | Vernier.” https://www.vernier.com/products/interfaces/labq/hardware/.

Nyquist, H. 1928. “Certain Topics in Telegraph Transmission Theory.” *Transactions of the American Institute of Electrical Engineers* 47 (2): 617–44. https://doi.org/10.1109/t-aiee.1928.5055024.

Theman, Dan. 2015. “Can’t Get I2C to Work on an Arduino Nano? (Pinout Diagrams).” 2015. https://bigdanzblog.wordpress.com/2015/01/30/cant-get-i2c-to-work-on-an-arduino-nano-pinout-diagrams.

“What Is Data Acquisition? - National Instruments.” http://www.ni.com/data-acquisition/what-is/.