UNIVERZITA KONŠTANTÍNA FILOZOFA
V NITRE

Programovanie vývojovej dosky Arduino

Zbierka úloh pre stredné odborné školy

Peter Kuna
Miloš Palaj

Názov: Programovanie vývojovej dosky Arduino
Zbierka úloh pre stredné školy

Autori: Ing. Mgr. Peter Kuna, PhD.
Mgr. Miloš Palaj, PhD.

Recenzenti: Prof. Ing. Veronika STOFFOVÁ, CSc.
doc. Ing. Štefan Koprda, PhD.

Publikácia bola vydaná v rámci projektu KEGA 017UKF-4/2020

ISBN 978-80-558-1827-6

OBSAH

Úvod

Prvé dve dekády tretieho tisícročia sa nesú v znamení rozvoja informačných technológií a ich aplikačného rozmachu do oblastí bežného života človeka. V prvej vlne to bola výpočtová technika, ktorá už úplne nahradila ostatné kancelárske nástroje, čím sa od základov zmenili metódy a postupy spracovania informácií. V druhej vlne to bol nástup mobilných technológií. Súčasné inteligentné (smart) telefóny už plnia skôr funkciu osobného dátového centra ako zariadenia na telefonickú komunikáciu. Ďalšie kroky rozvoja v oblasti aplikácie informačných technológií sú definované v dokumente Industry 4.0. Tento dokument sa podrobne venuje výskumným a vývojovým cieľom v širokom rozsahu použitia, ale so spoločným menovateľom – aplikácia informačných technológií, automatizácie a kybernetiky do každej oblasti nášho života. Jednou z oblastí, ktorej sa dokument Industry 4.0 venuje je aplikácia systémov IoT (Internet of Things) do reálneho života. Systémy IoT aplikujú internetové dátové prepojenie technických zariadení v takej hĺbke a rozsahu, ktoré doteraz zodpovedali len obsahu Sci-Fi literatúry. Prvé lastovičky v podobe smart hodiniek, smart televízií či inteligentnej domácnosti sa v súčasnosti stávajú už synonymom dnešnej doby. Systémy IoT však majú oveľa širšie aplikačné možnosti. Ako príklad uvádzame sledovanie naplnenia smetných nádob a podľa toho automatické privolanie služby na odvezenie, či automatický zber dát a ich posielanie ošetrujúcemu lekárovi v prípade srdcových strojčekov a podobne.

Samozrejme, že všetky nové technológie so sebou prinášajú aj riziká. Týmto témam sa venuje morálka a výchova. Nijako tento rozmer aplikácie informačných technológií nepodceňujeme ale našim primárnym cieľom, teda učiteľov technických odborných predmetov, je reflektovanie výučbového procesu požiadavkám aplikačnej praxe. Tomuto cieľu sme venovali aj učebnicu programovania vývojovej dosky Arduino.

Systém Arduino je celosvetovo úspešným univerzitným projektom, ktorý slúži na podporu vzdelávania programátorov v oblasti procesorovej techniky. Vývojová doska Arduino a jej rozšírenia spĺňajú v plnom rozsahu všetky technické požiadavky systémov IoT. Nezanedbateľná je aj cenová dostupnosť jednotlivých súčastí pre školy a dokonca i jednotlivcov.

Vývojová doska Arduino sa tak stáva silným nástrojom pre výučbu programovania IoT systémov.

Cvičenie č. 1: Technické prostriedky Arduina

Zoznamujeme sa s Arduinom

Predstavenie prvej vývojovej dosky Arduino, ktorá obsahuje mikrokontrolér ATmega od firmy Atmel, začala ešte v roku 2005 v talianskom Interaction Design Institute. Ich cieľom bolo vytvoriť jednoduchý a lacný vývojový nástroj, najmä pre žiakov, ktorí nechceli pracovať s vývojovou doskou BASIC Stamp. Medzi študentmi sa Arduino ujalo, a tak sa tvorcovia rozhodli poskytnúť ho ako prevažne učebnú pomôcku celému svetu, nakoľko ide o tzv. Opensource projekt, je teda voľne a zdarma šíriteľný medzi programátormi a technickými nadšencami.

Obrázok 1: Arduino Uno

Programová časť pre Arduino bola založená na tzv. Processingu, čo je v podstate programovací jazyk s vlastným editorom (Arduino IDE), určený k výučbe programovania (v súčasnosti je možné programovať arduino aj v iných vývojových prostrediach). V dnešnej dobe sa predalo už niekoľko miliónov dosiek Arduino a za uplynulé roky vývoja vzniklo veľa rôznych typov tejto vývojovej dosky. Keďže ide o opensource projekt, vznikalo spoločne s hlavnou líniou projektu aj veľa ďalších, neoficiálnych typov a klonov.

Arduino Uno je v súčasnej dobe asi najčastejšie používaný typ dosky. Je priamym pokračovateľom hlavnej vývojovej línie, ktorá začala prvým Arduinom so sériovým portom namiesto USB, pokračujúce cez Arduino Extreme, NG, Diecimila a Duemilanove až k dnešnému typu Uno. Na doske nájdeme procesor ATmega328 a už klasické USB. Z tejto hlavnej línie sa vyvinuli aj ďalšie dve špeciálne dosky. Prvá z nich je Arduino Ethernet, ktorá

má rovnakú výbavu ako Uno. Namiesto USB portu tu ale nájdeme Ethernet port pre pripojenie k sieti. Druhou doskou je Arduino Bluetooth. Ako už názov napovedá, miesto USB tu nájdeme bluetooth modul pre bezdrôtovú komunikáciu. Veľmi odľahčenou verziou Arduino Uno je Arduino Pro. Tu nie je USB port, a preto je nutné ho programovať externým prevodníkom. Je určené skôr k pevnému zabudovaniu do určitého projektu.

Mikroprocesor	ATmega328P
Max. napätie I/O pinov	5V
Napájanie dosky (doporučené)	7-12V
Napájanie dosky (min/max)	6-20V
Digitálne I/O piny	14 (z toho je 6 PWM výstupov)
PWM piny (8-bit)	6 (D3, D5, D6, D9, D10, D11)
Analógové piny (10-bit)	6 (A0-A5)
DC prúd na I/O pin	20 mA (max 40mA)
DC prúd z 3.3V pin	50 mA
Flash pamäť	32 KB z toho 0.5 KB pre bootloader
SRAM	2 KB
EEPROM	1 KB
Frekvencia mikroprocesora	16 MHz
LED_BUILTIN	13 (integrovaná LED na pine)
Dĺžka	68.6 mm
Šírka	53.4 mm
Váha	25 g

Tabuľka 1: Technické parametre vývojovej dosky Arduino UNO

Arduino Uno sa líši od všetkých predchádzajúcich dosiek v tom, že nepoužíva čip FTDI (USB-to-serial), namiesto neho je použitý čip Atmega16U2

(Atmega8U2 u verzie R2), naprogramovaný ako prevodník USB-to-séria1.U klonov môže byť nahradený prevodníkom CH340.

Napájanie

Arduino Uno doska môže byť napájaná cez USB konektor alebo externým DC napájaním cez konektor. Napájací zdroj je vybraný automaticky. Doska môže pracovať s externým napájaním od 6 do 20 voltov. Ak je napájanie menej ako 7V, tak na výstupnom 5V pine môže byť menej ako 5V a doska môže byť nestabilná. Ak použijete viac ako 12V, regulátor (stabilizátor) napätia sa môže prehriať a poškodiť dosku. Odporúčaný rozsah je teda 7 až 12 voltov. Napájacie piny sú nasledovné:

- ***Vin:*** Vstupné napájanie Arduino dosky (doporučovaných 7-12V), keď je použité externé napájanie (nie z USB). Ak je napájanie cez jack konektor, na tomto pine sa objaví toto napätie a môžete ho použiť na napájanie iných obvodov.
- ***5V:*** Na tomto pine je výstup 5V z regulátora napájania osadeného rovno na doske, ktorý je napájaný buď z napájacieho konektora DC (7-12V) alebo z pinu Vin (7-12V) alebo taktiež pri napájaní z konektora USB (5V). Napájať dosku rovno cez 5V pin (alebo 3.3V pin) sa nedoporučuje, nakoľko môže dôjsť k poškodeniu regulátora, napájania alebo obvodov na doske.
- ***3V3:*** Na tomto pine je výstup 3,3V generované z regulátora napájania osadeného tiež na doske. Maximálny odber prúdu je 50 mA.
- ***GND:*** Uzemnenie.
- ***IOREF:*** Tento pin na Arduino doske poskytuje referenčné napätie 5V, s ktorým aktuálne doska na I/O pinoch pracuje (Arduino Due tu má 3.3V). Je trvalo napojená na pin 5V. Napríklad správne nakonfigurovaný shield vie z tohoto pinu vyčítať napätie a vyberie príslušný zdroj napájania shieldu alebo bude podľa potreby používať konvertor TTL úrovní (*level shift converter*) na výstupe/vstupe.

Pamäť

Integrovaný čip ATmega328 má 32 KB flash pamäti pre program, z toho je použitých 0.5 KB pre bootloader. Veľkosť SRAM je 2 KB a má 1 KB EEPROM.

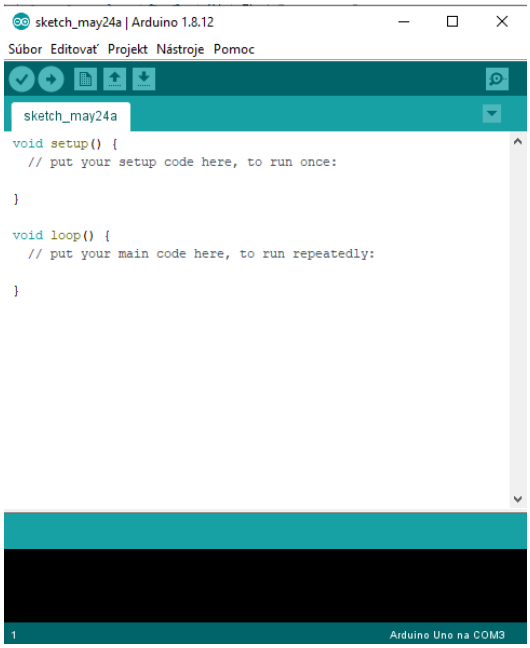
Vstupy a výstupy

Každý zo 14 digitálnych pinov na Arduino UNO doske možno použiť ako vstup alebo výstup, pomocou pinMode(), digitalWrite() a digitalRead() funkcií. Pracovné napätie pinov je 5V. Každý pin môže poskytovať alebo prijímať 20mA. Rovnako obsahuje aj interný pull-up rezistor 20-50kOhm (v predvolenom nastavení je odpojený). Maximálna hodnota prúdu je 40mA, ktorá nesmie byť prekročená na akomkoľvek I/O pine, aby sa zabránilo trvalému poškodeniu mikrokontroléra.

Avšak niektoré piny na doske majú špecializované funkcie:

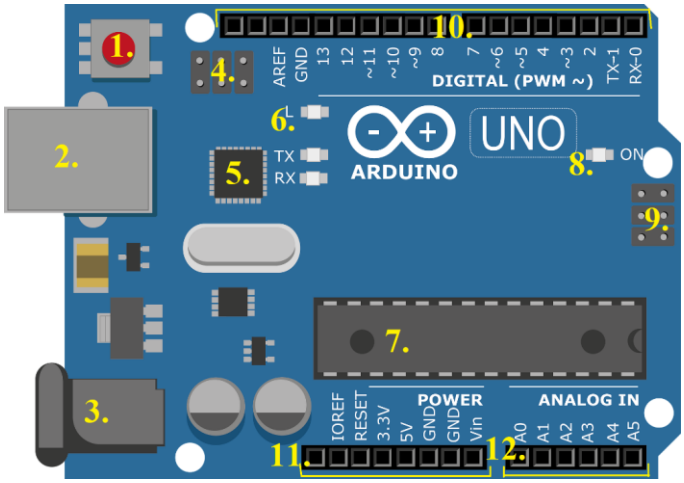
- ***Seriová linka:*** 0 (RX) a 1 (TX). Používa sa pre príjem (RX) a prenos (TX) TTL dát. Tieto piny sú pripojené na zodpovedajúce piny čipu ATmega8U2 (USB-to-TTL Sérial). Komunikácia je indikovaná pomocou integrovaných LED diód (RX LED a TX LED).
- ***Externé prerušenie:*** Pin 2 a 3. Tieto piny môžu byť konfigurované tak, aby spustili prerušenie pri nízkej hodnote 0V (low) alebo pri nábežnej hrane z 0V na 5V (rising), alebo pri klesajúcej hrane z 5V na 0V (falling), alebo pri zmene hodnoty (change). Viac informácií nájdete vo funkcii attachInterrupt().
- ***PWM:*** Piny 3, 5, 6, 9, 10 a 11 poskytujú 8-bitový PWM výstup funkciou analógWrite().
- ***SPI:*** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Tieto piny podporujú hardwarovú SPI komunikáciu.
- ***LED:*** Pin 13. Na doske je zabudovaná LED dióda do série s rezistorom a je pripojená k digitálnemu pinu 13. Keď je na pine hodnota high, tak LED svieti, keď je na pine hodnota low, tak LED nesvieti.
- ***Analógové vstupy:*** 6 analógových vstupov, z ktorých každý poskytuje 10 bitové rozlíšenie (t.j. 1024 rôznych hodnôt). Referenčné napätie pre analógové vstupy je štandardne 5V, ale dá sa zmeniť použitím funkcie analógReference(). Piny A4 a A5 navyše majú špecializovanú funkciu pre komunikáciu po I2C zbernici.
- ***I2C:*** Používajú sa analógové piny A4 (SDA) a A5 (SCL). Podporujú komunikáciu I2C (TWI) pomocou knižnice Wire.
- ***AREF:*** Pin sa používa ako referenčné napätie pre analógové vstupy. Používa sa s funkciou analógReference().

- ***Reset:*** Tento pin pri hodnote low resetuje čip. Zvyčajne sa používa na pridanie resetovacieho tlačidla alebo mikrospínača do shieldov. Jedno takéto nájdete aj na tejto doske.



Obrázok 2: Vývojové prostredie Arduino IDE

Popis jednotlivých častí vývojovej dosky Arduino UNO



Obrázok 3: Vývojová doska Arduino s číselným označením jednotlivých častí

1. Pod číslom jeden sa nachádza resetovacie tlačidlo. Používa sa v prípade, pokiaľ potrebujeme náš program spustiť odznova. Pri rôznych typoch Arduina sa môžeme stretnúť aj s inými umiestneniami tohto tlačidla. Väčšinou je toto tlačidlo popísané nápisom Reset.
2. USB konektor typu B. V prípade najstaršieho modelu Arduina nájdeme miesto USB portu tzv. sériový port. V prípade novších modelov sa môžeme stretnúť aj s micro USB. Niektoré vývojové dosky tento port vôbec neobsahujú, nakoľko obsahujú iný spôsob pripojení (Ethernet, BT). Rovnako sa môže využívať na programovanie aj externý programátor.
3. Napájací konektor. Využíva sa v prípade, ak Arduino nenapájame prostredníctvom USB portu.
4. ICSP hlavica pre externé programovanie USB-sérial prevodníkov. Avšak bežný používateľ túto možnosť nevyužije.
5. USB-sérial prevodník. Slúži na komunikáciu medzi hlavným čipom a PC. Zastáva úlohu prekladateľa. V prípade verzie Arduina bez prevodníka alebo s prevodníkom, integrovaným v hlavnom čipe, tento prevodník nenájdeme.
6. Indikačná LED (L, Rx, Tx). Dióda s označením L je často využívaná a je prepojená s výstupom pinu 13. S jej pomocou sa dá otestovať blikanie aj bez pripojenia externej LED a odporu. Niektoré typy Arduina ju však neobsahujú. Diódy s označením Rx a Tx indikujú prebiehajúcu sériovú komunikáciu.
7. Hlavný čip celej vývojovej dosky. V rôznych podobách a typových vyhotoveniach ho nájdeme vo všetkých Arduino doskách.
8. Indikačná LED ON. Svieti v prípade, ak je pripojené napájanie.
9. ICSP hlavica pre externé programovanie hlavného čipu. Túto hlavicu využívajú niektoré nadstavby („shieldy“) pre Arduino.
10. Digitálne piny. Do týchto dierok sa pripájajú konkrétne obvody a ich časti. Vývody označené vlnovkou, podporujú pulznú šírkovú moduláciu (PWM).
11. Výstupy slúžiace pre napájanie Arduina.
12. Analógové piny. Sem pripájame vodiče, na ktorých budeme merať analógovú veličinu. Tieto piny je možné využiť aj ako digitálne vstupy.

Zaujímavé projekty, riadené prostredníctvom vývojovej dosky Arduino

Oskenuj QR kód alebo skopíruj url adresu do svojho internetového prehliadača, pre inšpiráciu a ukážku zaujímavých projektov, ktoré môžeš vytvoriť prostredníctvom vývojovej dosky Arduino, ktorá obsahuje mikrokontrolér Atmel.

https://www.youtube.com/watch?v=Lp8eSCuQqb0&ab_channel=ViralHattrix

Cvičenie č. 2: Programovanie jednoduchých aplikácií s LED

V druhom cvičení si ukážeme praktický príklad toho, ako zapojiť a naprogramovať vývojovú dosku Arduino UNO. V našom prípade to bude „Arduino blikanie LED“. Blikať LED možno bez ďalších súčiastok s takmer každou vývojovou doskou Arduino. V tomto príklade použijeme Arduino UNO.

Čo budeme potrebovať pre blikanie s LED?

Na blikanie LED budeme potrebovať vývojovú dosku napr. Arduino UNO, prepojovací USB kábel a počítač s nainštalovaným programom Arduino IDE. Rovnako bude vhodné vyskúšať si blikanie nielen s integrovanou LED, ale aj s externou LED. K tomu bude potrebná aspoň jedna farebná LED a rezistor s hodnotou 220Ω a tzv. Breadboard, (to je prepojovacie kontaktné pole).

Zdôvodnenie hodnoty predradeného odporu 220 Ω

Arduino, ku ktorému pripájame LED, poskytuje obmedzený maximálny prúd, ktorý je schopné dodať na výstup. Maximálny prúd, ktorý prechádza jedným pinom, je 40mA. Odporúčaná hodnota záťaže na jeden pin je **do** 20mA. Preto môžeme LED priamo pripojiť na výstup Arduina, ale napr. žiarovku alebo elektromotor nie, nakoľko majú zvyčajne oveľa väčšiu prúdovú spotrebu, (preto používame Arduino len ako riadiaci člen, ktorý posiela riadiace signály). Na obmedzenie pretekajúceho prúdu obvodom používame rezistor. Čím vyššiu hodnotu odporu rezistor má, tým menší prúd prechádza diódou a ona bude o

to slabšie svietiť. Takže pri hodnote rezistoru 220Ω bude mať prúd hodnotu 11mA (záleží na farbe diódy), pri hodnote rezistoru 1kΩ je hodnota pretekajúceho prúdu 3mA. Avšak aj pri takto malom prúde bude LED jemne svietiť. LED však nepripájame k Arduinu bez rezistoru, pretože môže prísť k preťaženiu pin-u Arduina prúdom väčším ako 40mA a môže dôjsť k poškodeniu LED. Zvýšenú pozornosť je potrebné venovať aj tomu, že Arduino má obmedzený aj maximálny prúd pre všetky piny, a teda dohromady max 200mA**.** Môžeme teda pripojiť maximálne 10 súčasne svietiacich LED s prúdom 20mA, alebo výstup pinu zaťažíme nižším prúdom vďaka rezistoru s vyššou hodnotou odporu, a týmto spôsobom zvýšime počet LED v obvode, ktoré môžu súčasne svietiť. Pre prúd 10mA ich teda môžeme pripojiť 20. Samozrejme toto platí nielen pre Arduino, ale aj iné integrované obvody, kde ich maximálne výstupné prúdy nájdeme v katalógovom liste (tzv. datasheete).

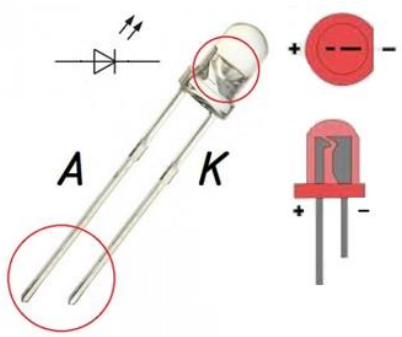
Zapojenie rezistoru a LED

Nakoľko sú obe súčiastky zapojené do série, tečie oboma súčiastkami rovnaký prúd. V takom prípade je týmto súčiastkam jedno, v akom poradí budú zapojené. Uvádzaný poznatok sa využíva pri návrhu umiestnenia súčiastok na plošnom spoji.

Správne zapojenie LED

Na rozdiel od žiaroviek, u ktorých nezáleží na polarite napájacieho napätia, a sú teda schopné pracovať na striedavé napätie, LED zapojená nesprávnym spôsobom svietiť nebude. Pre určenie správnej polarity diódy je potrebné pozrieť sa do jej katalógového listu (tzn. do datasheetu). Ďalším spôsobom, ako zistiť polaritu LED, je zmerať ju meracím prístrojom (napr. multimetrom).

Ďalšie spôsoby určenia polarity LED uvádza tabuľka s obrázkom:

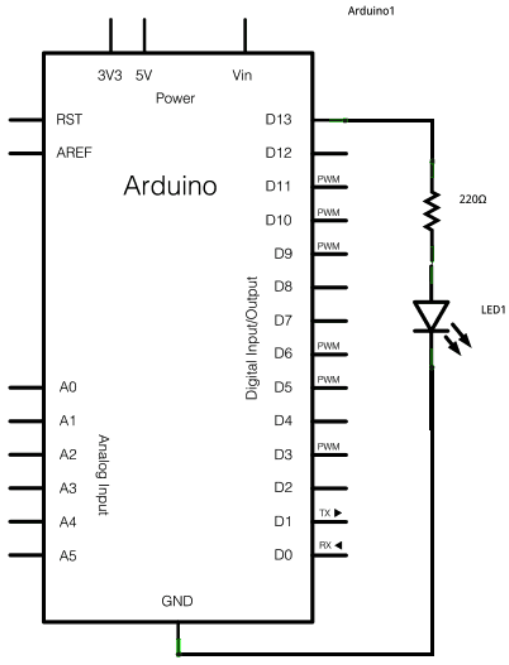


znamienko:	+	–
polarita:	kladná	záporná
výstup:	anóda (A)	katóda (K)
vývod:	dlhý	krátky
vo vnútri puzdra:	menšie	väčšie

Obrázok 4: Spôsoby určenia polarity LED

Základné zapojenie Arduino a LED

Každá súčiastka má svoju elektrotechnickú značku, kde sú označené jej vývody, prípadne hodnoty. Schéma zapojenia potom vyzerá nasledovne:



Obrázok 5: Schéma zapojenia Arduina s LED

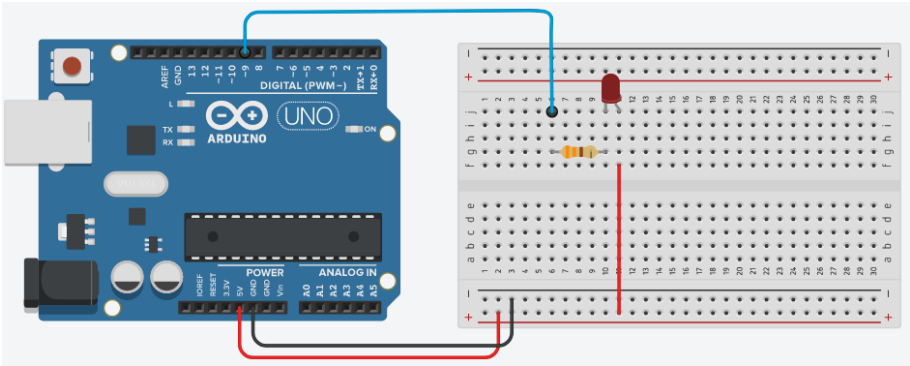
Uvádzaná schéma predstavuje typické zapojenie integrovanej LED, ktorá je umiestnená priamo na Arduino doskách. Niektoré vývojové dosky môžu mať LED aj na inom pine, alebo ju dokonca vôbec nemajú (napr. Arduino DUE). Na uvádzanej schéme môžeme vidieť veľký obdĺžnik, ktorý predstavuje Arduino s popisom pinov. K nemu je do série zapojená LED s rezistorom 220Ω medzi pin D13 a GND (GND je zem alebo mínusový pól). LED je elektronický prvok, do ktorej privádzame prostredníctvom Arduina prúd, a tým ovplyvňujeme jej stav. V najjednoduchšom prípade len prúd zapneme alebo vypneme – LED je v aktívnom stave alebo v pokoji (svieti / nesvieti). Na to, aby sme mohli prostredníctvom Arduina prúd riadiť, potrebujeme k zapojeniu napísať program. Najčastejšie býva používaný jazyk Wiring (v kombinácii s processing), ktorý je podobný jazyku C (C++).

Program pre riadenie vstavanej LED

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Pripojenie externých elektronických súčiastok k Arduinu

Veľmi často sa však využíva Arduino pre ovládanie externých súčiastok a zariadení. Preto si ukážeme, ako pripájať jednotlivé súčiastky prostredníctvom kontaktného poľa (tzv. Breadboard). Externú LED pripojíme spolu s rezistorom podľa nasledovného obrázka. Všimnite si, že LED je teraz pripojená na pin D9.



Obrázok 6: Zapojenie Arduina s LED pomocou kontaktného poľa

Nezabudnite si po pripojení Arduina skontrolovať číslo COM portu a nastavenie správnej vývojovej dosky. Tieto informácie sa nachádzajú vo

vývojovom prostredí Arduino IDE, v záložke Nastavenia. Zostáva potom už len testovanie, kompilácia programu a jeho nahratie do Arduina.

Činnosť riadiacich signálov

Uvádzaný program sa pokúsime analyzovať a následne aplikovať určité modifikácie. Chceme, aby blikala externá LED. Celý program pre Arduino sa skladá z dvoch častí – funkcií. Vo funkcii void setup sa nachádzajú príkazy, ktoré sa vykonajú len raz po zapnutí (alebo po resete), v tejto časti definujeme jednotlivé prvky, ktoré bude Arduino používať. Následne Arduino začne opakovane vykonávať príkazy, uvádzané vo funkcii void loop. Prvé príkazy, ktoré popíšeme, budú pre ovládanie jednotlivých pinov (nožičiek) Arduina. V prvom prípade nám budú stačiť len dva stavy (vypnuté alebo zapnuté, High/LOW) a teda digitálne signály (logické signály) pre ovládanie Arduina. Ako prvé musíme nastaviť to, či daný pin bude vstupný (signál bude do Arduina privádzaný zvonka) alebo výstupný (Arduino bude niečo riadiť). K tomu slúži funkcia (príkaz) pinMode(pin, mode).

pinMode(pin, mode)

- *pin ... určuje, ktorý pin (D0-D13) sa má nastaviť*
- *mode ... určuje, či to bude vstup alebo výstup (INPUT, OUTPUT)*

Funkcia pinMode sa nachádza v časti void setup, pretože nám ju stačí vykonať len raz pri spustení programu. LED ešte nerozsvietime, len povieme Arduinu, že tento pin nastavujeme ako výstupný. Zápis bude potom vyzerať takto:

pinMode (13, OUTPUT); // Nastavenie pinu 13 na výstup

pinMode (12, INPUT); // a pinu 12 na vstup

Lomítka za jednotlivými časťami programu predstavujú poznámky programátora a text za nimi kompilátor ignoruje. Druhú z funkcií, ktorú budeme potrebovať, je nastavenie hodnoty na výstupnom pine a k tomu nám poslúži ďalší príkaz *digitalWrite(pin, value)*.

digitalWrite(pin, value)

- *pin ... určuje, ktorý pin (D0-D13) sa má nastaviť*
- *value ... určuje hodnotu výstupu 0/1 LOW/HIGH 0V/5V*

Tento príkaz sa už nachádza vo funkcii loop. To je časť programu, ktorý sa nám bude neustále opakovať. Zápis tohto príkazu vyzerá takto:

digitalWrite (9, HIGH); // na výstup 9 nastaviť 1 (5V) LED svieti

digitalWrite (9, LOW); // na výstup 9 nastaviť 0 (0V) LED nesvieti

Posledným príkazom je delay(1000). Príkaz delay(1000) zastaví vykonávanie programu na 1000 milisekúnd, čo je jedna sekunda. Preto vidíte, že LED bliká v sekundových intervaloch, pretože je medzi dvomi príkazmi digitalWrite.

delay(time)

- *time ... čas čakania v milisekundách*

Celý program bude vyzerať nasledovne:

```
void setup() {
  pinMode(9, OUTPUT);
}
void loop() {
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}
```

Popis funkcie programu

Nastavíme najskôr pin 9 ako výstupný. Ďalej v slučke loop nastavíme tento pin na HIGH, čím LED rozsvietime (privedieme naň signál). Program potom čaká 1s, následne diódu zhasne a čaká znovu jednu sekundu. Potom program začína znova slučkou loop od príkazu na nastavenie pinu na HIGH a stále dookola.

Ako postupovať v prípade zmeny pin-u Arduina?

Pri prepisovaní čísla 9 na iné číslo pinu nastáva potenciálne nebezpečenstvo, že to niekde zabudnete. Hlavne pri dlhších programoch. Preto je vhodnejšie zaviesť takzvané symbolické pomenovanie namiesto konkrétneho čísla pinu, a potom ho používať v programe. To realizujeme tzv. direktívou preprocesora. Podobne, ako je to s konštantou LED_BUILTIN.

#define LED 10

- *LED ... symbolické meno*

- *10 ... hodnota, ktorou sa symbolické meno v programe nahradí*

Takéto pomenovanie si teraz vytvoríme a rovno dve rôzne. Potrebujeme totiž meniť aj čas.

```
#define LED   10
#define TIME  500
void setup()
{
  pinMode(LED, OUTPUT);   // nastavenie pinu LED ako výstup
}
void loop()
{
  digitalWrite(LED, HIGH);  // rozsvietime LEDku
  delay(TIME);              // čakanie TIME milisekúnd
  digitalWrite(LED, LOW);   // zhasneme LEDku
  delay(TIME);              // čakanie TIME milisekúnd
}
```

Teraz fyzicky pripojíme LED na pin 10, nahráme program a LED bude blikať na pine 10 ale rýchlejšie. Hodnotu 500 zmeňte na nižšiu. Vyskúšajte si zadefinovať dva rôzne časy – jeden na dĺžku svietenia a druhý na dĺžku nesvietenia. Napr. 2 sekundy LED svieti a pol sekundy nesvieti.

Zaujímavosť

Funkcia delay má jeden parameter a to je čas čakania v milisekundách. Rozsah parametra je spravidla od 0 do 4.294.967.295 (unsigned long). Veľkou nevýhodou funkcie delay() a aj funkcie času delayMicroseconds() je to, že dôjde k zastaveniu takmer celej činnosti programu (pozastavenie čítania hodnôt zo senzorov, matematické operácie, nemožnosť ovládať logické hodnoty na pinoch atď.). Avšak nedôjde k zastaveniu tých funkcií, ktoré nie sú priamo závislé na programe procesoru. Ide najmä o prerušenia, príjmu informácií z Rx linky, kde spracovanie nastane až po skončení funkcie delay a takisto o funkciu

analogWrite() či generovanie signálu PWM, ktorá prebieha mimo hlavný procesorový blok.

V prípade ak realizujeme zapojenie, kde sa nachádzajú rôzne komponenty, odporúčame používať funkciu millis(). Použitím funkcie millis() je možné určiť hodnotu uloženú v internom časovači procesora. Nachádzajú sa tu uložené informácie o dĺžke behu programu od jeho spustenia. Táto funkcia nepotrebuje žiadny parameter, pretože vráti počet milisekúnd od začiatku behu programu. Toto číslo však nie je nekonečné. Maximálna návratová hodnota je 4.294.967.295 (unsigned long), čo je po prepočet 1193 hodín alebo 49,7 dní. Po prekročení tohto času dôjde k tzv. pretečeniu časovača a začne sa odpočítavanie od začiatku. Funkcia millis() sa používa napríklad vtedy, ak potrebujete čakať, ale nie je žiadúce prerušiť chod celého programu.

Ukážka programu s použitím millis() bez použitia funkcie delay()

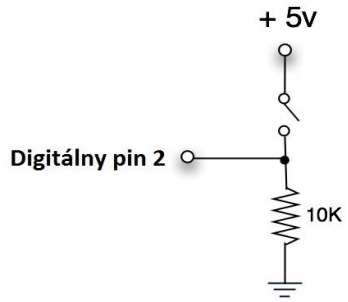
```
/* Vo výsledku to znamená, že program môže vykonávať aj iné výpočty
 bez toho, aby bol prerušený čakaním medzi bliknutiami LED. */
const int ledPin = 13;  // číslo pinu pre LED
int ledState = LOW;     // ledState sa používa k nastavovaniu stavu LED
// premenné sú typu „unsigned long“, nakoľko je čas meraný v milisekundách
//hodnoty rýchlo narastajú na vysoké čísla a premenná int, by nepostačovala
unsigned long previousMillis = 0; // posledný čas aktualizácie LED
unsigned long interval = 1000;   // časový interval blikania v milisek
void setup() {
pinMode(ledPin, OUTPUT);        // nastavenie pinu ako výstup:
}
void loop()  {
 // Sem vložíte kód, ktorý sa bude neustále opakovať.
unsigned long currentMillis = millis();
 if(currentMillis - previousMillis > interval) {
   previousMillis = currentMillis;   // uložíme čas, posledného bliknuitia s LED
  if (ledState == LOW)        // ak je LED vypnutá, zapneme ju a naopak:
     ledState = HIGH;
```

```
    else
      ledState = LOW;
    digitalWrite(ledPin, ledState); // nastavíme LED podľa ledState
} }
```

Cvičenie č. 3: Programovanie jednoduchých s tlačidlom

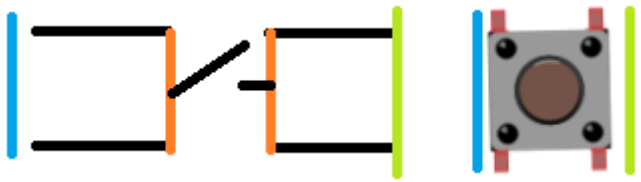
V tomto cvičení sa budeme zameriavať na programovanie tlačidla a neskôr na programovanie troch LED. Akékoľvek tlačidlo, spínač, kontaktný nárazník a pod. je spravidla pripojený na digitálny pin Arduina, ktorý je nastavený ako vstupný. Môže nadobúdať iba dve hodnoty, buď je tlačidlo stlačené alebo stlačené nie je. Počet digitálnych pinov závisí od vývojovej dosky Arduina.

Schematické zapojenie spínača je nasledovné:



Obrázok 7: Schéma zapojenia tlačidla

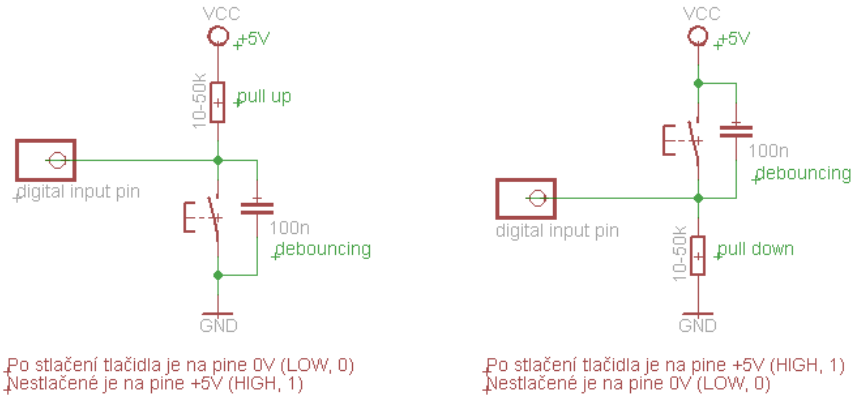
Na ďalšom obrázku môžeme vidieť vnútornú štruktúru tlačidla a jeho prepojenie vývodov.



Obrázok 8: Vnútorná štruktúra zapojenia tlačidla

Digitálny pin nastavíme ako vstupný príkazom pinMode(číslo_pinu, INPUT); (v prípade, že používame externý pull up alebo pull down rezistor), prípadne príkazom pinMode(číslo_pinu, INPUT_PULLUP); (ak nepoužívame

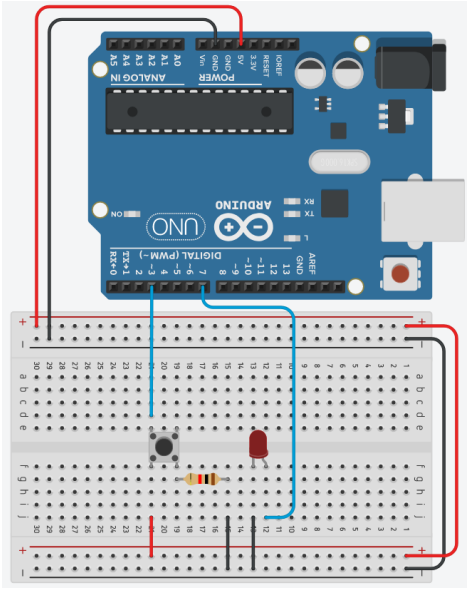
externý pull up rezistor a chceme, aby po stlačení tlačidla bola na pine hodnota LOW (0V, GND)) na začiatku programu, najčastejšie vo funkcii void setup(). Pri stlačení tlačidla (a tiež pri jeho uvoľnení) vznikajú vo veľmi krátkom čase viacnásobné zopnutia a rozopnutia, zákmity tlačidla (tzv. bouncing). Odstránenie tohto javu (tzv. debouncing) je možné buď softvérovo (viacnásobné čítanie vstupu) alebo hardvérovo (RC obvod, klopný obvod). Čiastočne pomôže aj samostatný kondenzátor pripojený paralelne k tlačidlu.



Obrázok 9: Schéma zapojenia s ošetrením zákmitov

Úloha 1: Načítanie tlačidla a prenos stavu na výstup LED

Na realizáciu tejto úlohy potrebujeme Arduino, prepojovaciu dosku (breadboard), LED diódu, rezistor a tlačidlo (spínač). Tento program je o jednoduchom čítaní statusu pinov a zapisovaní výsledku v podobe rozsvietenia LED diódy.



Obrázok 10: Zapojenie Arduina a tlačidla

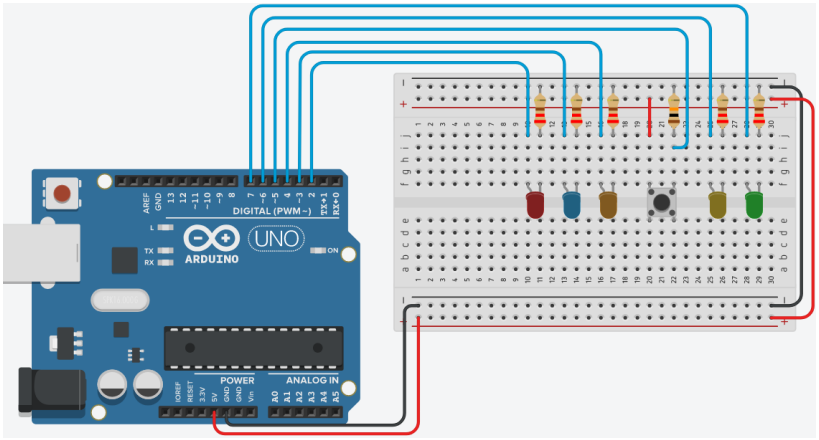
Výsledný program pre úlohu č. 1

```
int čítanie;
int led = 7;
int tlačidlo = 3;
void setup()  {
pinMode(led, OUTPUT);
pinMode(tlačidlo, INPUT);
}
void loop() {
čteni = digitalRead(tlačidlo);
digitalWrite(led, čteni);
}
```

Úloha 2: Semafor cez Arduino a LED

V tejto časti cvičenia sa budeme venovať zapojeniu LED v praktickom príklade semaforu. LED semafor patrí k jednoduchším projektom, ktorý bude fungovať tak, že sa po stlačení tlačidla zapne červená LED pre autá a zelená pre

chodcov. Po chvíli sa opäť vráti do pôvodného stavu - toto sa stane rovnako ako pri reálnom semafore.



Obrázok 11: Arduino a LED v zapojení semafor

Použité súčiastky:

arduino doska, 2x zelená LED, 2x červená LED, 1x žltá LED, 1x tlačidlo, 5x 220Ω rezistor, 1x 10k rezistor, prepojovacie vodiče.

Výsledný program pre úlohu č. 2

Na začiatku sa piny 2, 3, 4, 6 a 7 nastavia ako vstupné (input) a pin 5 ako výstupný (output), tiež sa zapne pre autá zelená a pre chodcov červená. Ďalej sa neustále kontroluje, či je pin 5 zapnutý (tlačidlo je stlačené) a ak áno, tak sa zapne sekvencia (opakovanie) semaforu.

```
int cervena = 2 ;  // pin 2 je červená
int zlta = 3 ;  // pin 3 je žltá
int zelena = 4 ;  // pin 4 je zelená
int zelenal = 7 ;  // pin 7 je zelená pre chodcov
int červenal = 6 ;  // pin 6 je červená pre chodcov
int tlacitko = 5 ;  // pin 5 je tlačidlo
void  setup () {
  pinMode (cervena, OUTPUT );  // červená je výstup
 pinMode (zlta, OUTPUT );  // žltá je výstup
```

```
 pinMode (zelena, OUTPUT );  // zelená je výstup
 pinMode (zelenal, OUTPUT );  // zelená pre chodcov je výstup
 pinMode (červenal, OUTPUT );  // červená pre chodcov je výstup
 pinMode (tlacitko, INPUT );  // tlačidlo je vstup
 digitalWrite (zelena, HIGH );  // zapnúť zelenú
 digitalWrite (červenal, HIGH );  // zapnúť červenú chodcom
}
void  loop () {
  if ( digitalRead (tlacitko) == HIGH ) {   // ak je tlačidlo zapnuté
   digitalWrite (zelena, LOW );  // vypnúť zelenú
   digitalWrite (zlta, HIGH );  // zapnúť žltú
   delay ( 500 );  // počkať 0.5 sekúnd
   digitalWrite (zlta, LOW );  // vypnúť žltú
   digitalWrite (cervena, HIGH );  // zapnúť červenú
   delay ( 500 );  // počkať 0.5 sekúnd
   digitalWrite (červenal, LOW );  // vypnúť červenú pre chodcov
   digitalWrite (zelenal, HIGH );  // zapnúť zelenú pre chodcov
   delay ( 5000 );  // počkať 5 sekúnd
   digitalWrite (zelenal, LOW );  // vypnúť zelenú pre chodcov
   digitalWrite (červenal, HIGH );  // zapnúť červenú pre chodcov
   delay ( 500 );  // počkať 0.5 sekúnd
   digitalWrite (zlta, HIGH );  // zapnúť žltú
   delay ( 500);  // počkať 0.5 sekúnd
   digitalWrite (cervena, LOW );  // vypnúť červenú
   digitalWrite (zlta, LOW );  // vypnúť žltú
   digitalWrite (zelena, HIGH );  // zapnúť zelenú
 }}
```

Cvičenie č. 4: Programovanie analógových vstupov.

S pojmami ako digitálny vstup a výstup sme sa už stretli v predchádzajúcich cvičeniach. Čo ak ale potrebujeme pracovať so signálmi, ktoré môžu nadobudnúť viac ako dve hodnoty? V takomto prípade má Arduino vo výbave užitočné funkcie.

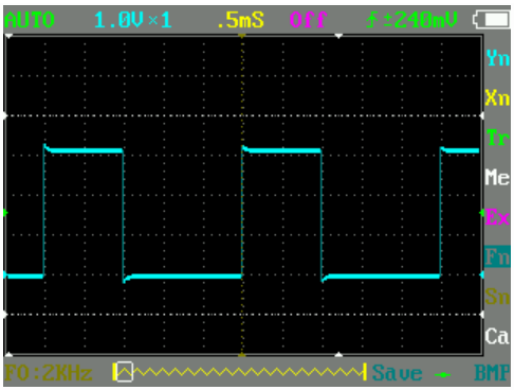
Arduino dokáže okrem digitálnych hodnôt spracovávať aj analógové signály. Analógový signál je ten, ktorý môže na rozdiel od digitálneho signálu, nadobudnúť viac ako dve hodnoty (digitálny signál HIGH a LOW). Na meranie hodnoty analógových signálov má Arduino zabudovaný analógovo-digitálny prevodník (ADC). Tento prevodník prevedie analógovú hodnotu (napätie) na digitálnu hodnotu. Funkcia, ktorú používa pre získanie hodnoty analógového signálu, je analogRead (pin). Slúži na čítanie analógových hodnôt na vstupoch Arduina s označením A0-A5. Toto čítanie využívame napríklad pri práci s rôznymi senzormi (napr. teplota, vlhkosť, atď.). Väčšina Arduino dosiek má schopnosť čítať analógovú veličinu v rozsahu 10 bitov (210). Inak povedané, táto funkcia prepočíta hodnotu napätia na analógovú vstupnú hodnotu a vráti digitálnu hodnotu od 0 po 1023 vzhľadom na referenčnú hodnotu. Štandardná hodnota je 5V, no môžeme sa stretnúť aj s hodnotou 3,3V.

Arduino však nemá zabudovaný vlastný digitálno/analógový prevodník (DAC). Dokáže ale pracovať s PWM (pulse width modulation, pulzná šírková modulácia) signálom, a tak môže simulovať niektoré funkcie analógového výstupu. Funkcia použitá pre výstup signálu PWM má podobu analogWrite (pin, hodnota). Pin je číslo, používané pre výstup PWM, hodnota je číslo úmerné pracovnému cyklu signálu. Ak je hodnota = 0, signál je vždy vypnutý. Keď je hodnota = 255, signál je vždy zapnutý. Na väčšine dosiek Arduino je funkcia PWM dostupná na pinoch 3, 5, 6, 9, 10 a 11. Frekvencia signálu PWM na väčšine pinov je 490 Hz. Na Uno a podobných doskách majú kolíky 5 a 6 frekvenciu približne 980 Hz. Ak chceme mapovať hodnotu analógového vstupu, ktorá sa pohybuje v intervale od 0 po 1023 na výstupný signál PWM, ktorý sa pohybuje v rozmedzí od 0 do 255, môžeme použiť tzv. mapu. Táto funkcia má päť parametrov, jedna je premenná, v ktorej je uložená analógová hodnota, zatiaľ čo ostatné sú 0, 1023, 0 a 255.

Princíp PWM

Mikrokontroléry dokážu spínať svoje piny buď k zemi (GND) alebo k napájaciemu napätiu. Tým sa vytvorí stále napätie 0 alebo 5V(3.3V). V prípade, že mikrokontrolér bude tieto stavy rýchlo striedať a na takýto výstup pridáme

filter vysokých frekvencii, zbavíme sa kolísania napätia a na výstupe dostaneme napätie v rozmedzí medzi 0-5V. Uvádzané napätie závisí od šírky tohto pulzu, v porovnaní so svojou periódou. Ak potrebujeme, aby mikrokontrolér vytvoril napätie 1V, musíme brať do úvahy, že je to 1/5 napájacieho napätia, a tým pádom šírka pulzu musí byť 1/5 periódy = 20%. Čiže naprogramujeme náš mikrokontrolér , aby svoj pin priviedol na 5V, napríklad na 1ms, a potom ho necháme uzemniť- tento pin na 4ms (1ms+4ms=5ms=1 perióda).



Obrázok 12: Priebeh PWM signálu na osciloskope

V prípade, ak chceme vytvoriť konkrétne napätie, môžeme použiť pomerne jednoduchý vzorec.

$$U_{out} = t_{on}/t_p * U_{in}$$

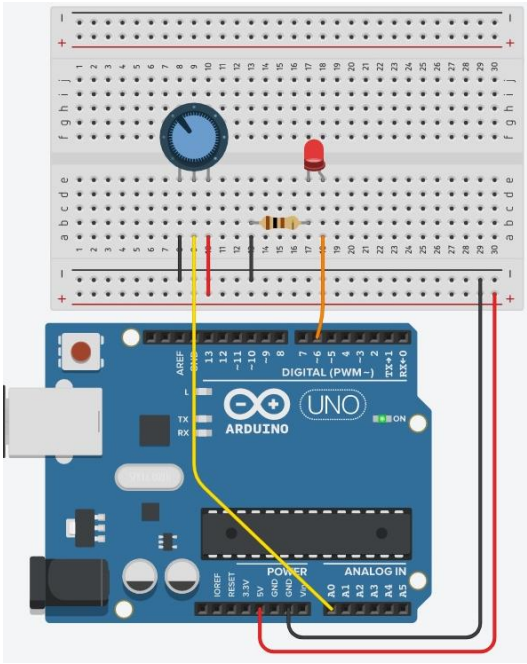
Kde t_on/t_p je strieda, čiže pomer času t_on, počas ktorého má pin napájacie napätie k času celej periódy t_p (t_p = t_on + t_off), toto číslo býva vyjadrené spravidla v percentách. V predchádzajúcom prípade bola táto hodnota 20% = 0.2 =1ms/(1ms+4ms) = 1ms/5ms. U_in je napájacie napätie mikrokontroléra alebo iné napätie, ktoré sa strieda so zemou (0V), aby sa vytvoril analógový signál, niekedy označované ako U_max.

Napríklad, ak potrebujem na výstupe 3.3V z 5V pomocou PWM , v takom prípade dosadíme tie hodnoty, ktoré poznáme a zistíme, akú striedu musíme použiť (D znamená strieda) 3.3V=D*5V

$$D=3.3V/5V=0.66=66\%$$

Týmto spôsobom generujeme zvuky, stmievame LED, regulujeme napätie, riadime motory, generujeme signály pre transformátory, meniče a iné. Program možno upraviť podľa zadaných parametrov.

Úloha: Regulácia jasu LED prostredníctvom arduina a potenciometra



Obrázok 13: Zapojenie Arduina, LED a potenciometra

V programe si musíme dať pozor na rozsah hodnôt, s ktorými funkcia pracuje. Z funkcie analogRead() dostaneme hodnoty v rozmedzí 0 až 1023, avšak funkcia analogWrite() pracuje s rozsahom hodnôt 0 až 255. Z uvádzaného dôvodu musíme zabezpečiť správny prevod týchto hodnôt. V tomto prípade to je jednoduché, nakoľko hodnota 256 (28) sa nachádza v hodnote 1024 (210) štyrikrát. Najjednoduchší spôsob je teda vydeliť hodnoty vo funkcii analogRead() štyrmi (existuje aj elegantnejší spôsob prevodu hodnôt, o ktorom si povieme ďalej).

```
byte led = 6;
byte pot = A0;
int val;
void setup() {
}
void loop() {
```

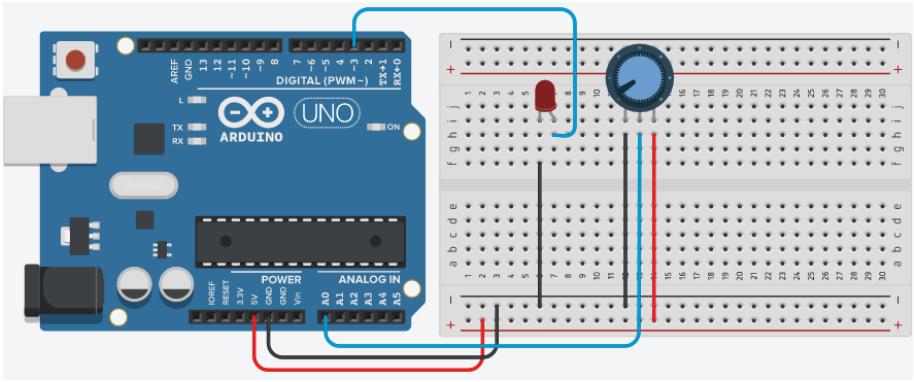
val = analogRead(pot)/4;

analogWrite(led, val);

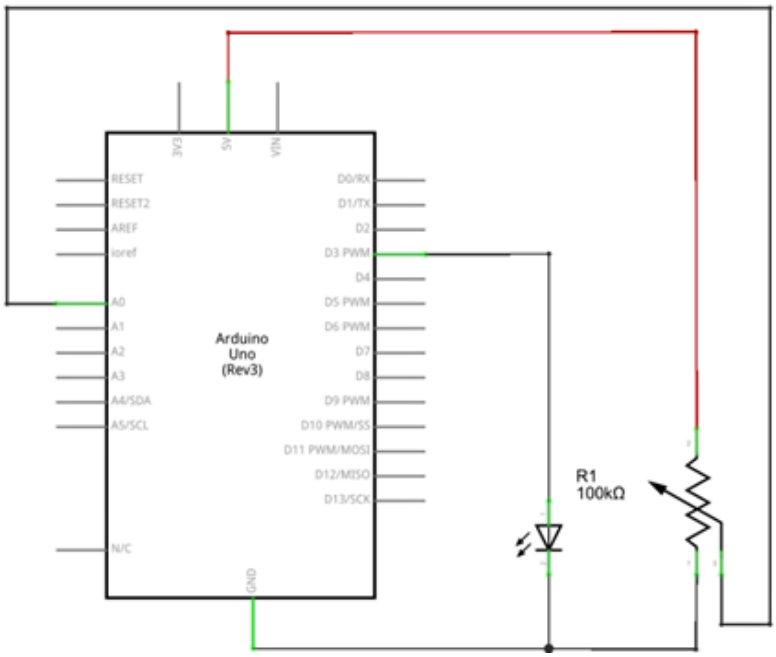
}

Ďalší spôsob využitia analógovej veličiny v spojitosti so sériovou komunikáciou

Ako ďalší príklad uvádzame riadenie jasu LED prostredníctvom už spomínanej PWM (pulse width modulation, pulzná šírková modulácia). Pre správne riadenie výstupu je nutné mapovať nameranú hodnotu z potenciometra na výstupnú analógovú hodnotu napätia. Hodnoty namerané pomocou funkcie analogRead sú v rozsahu 0 až 1023, zatiaľ čo na výstup PWM pinov musíme nastavovať napätie v rozsahu 0 až 255. Tento prepočet zabezpečí funkcia map. Táto funkcia sa skladá z niekoľkých častí. Na prvom mieste je zdrojová premenná z potenciometra, na druhom a treťom rozsah zdrojovej premennej a na posledných dvoch miestach rozsah cieľovej premennej. ledProm = map(potProm, 0, 1023, 0, 255); Dôležité je pripomenúť si, že výstupné napätie je možné nastavovať len na pinoch, ktoré sú označené ako PWM. Arduino UNO má tieto piny 3, 5, 6, 9, 10 a 11. Vo funkcii setup() sa nachádza príkaz pre samotné zapnutie a nastavenie sériovej komunikácie. Ďalej sa pre riadenie jasu LED odosielajú namerané hodnoty z potenciometra cez sériovú linku do pripojeného počítača. Okno pre zobrazenie zaslaných hodnôt si môžeme aktivovať po nahraní programu cez ikonu lupy v pravom hornom rohu či v ponuke- Nástroje/Sériový monitor. Nutné je tiež nastaviť v sériovom monitore rovnakú rýchlosť, akú máme nastavenú v programe a to 9600 bps.



Obrázok 14: Zapojenie Arduino Uno a potenciometer



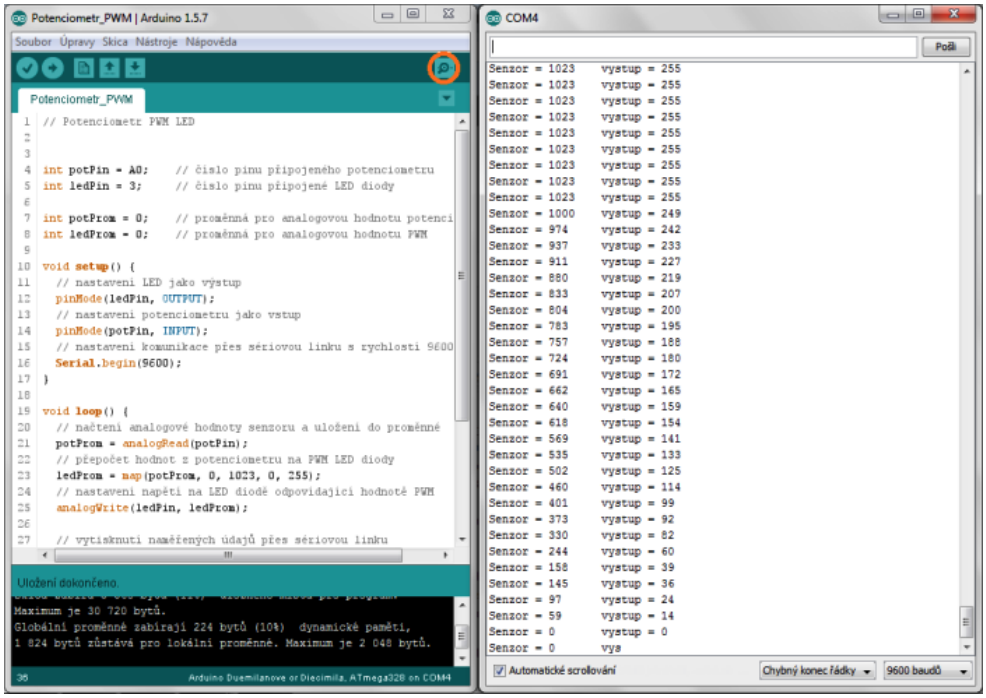
Obrázok 15: Schéma zapojenia pre Arduino Uno a potenciometer

Výsledný program pre reguláciu jasu LED prostredníctvom potenciometra

```
int potPin = A0;    // číslo pinu pripojeného potenciometra
int ledPin = 3;     // číslo pinu pripojenej LED diódy
int potProm = 0;    // premenná pre analógovú hodnotu potenciometra
int ledProm = 0;    // premenná pre analógovú hodnotu PWM


void setup() {
 pinMode(ledPin, OUTPUT); // nastavenie LED ako výstup
 pinMode(potPin, INPUT);  // nastavenie potenciometra ako vstup
 Serial.begin(9600); // nastavenie sériovej linky s rýchlosťou 9600 bps
}
void loop() {
// načítanie analógovej hodnoty snímača a uloženie do premennej
 potProm = analogRead(potPin);   //prepočet hodnôt z potenc. na PWM LED
 ledProm = map(potProm, 0, 1023, 0, 255); // nastavenie U na LED pre PWM
 analogWrite(ledPin, ledProm);
 Serial.print("Senzor = " ); // výpis nameraných hodnôt  cez serial linku
 Serial.print(potProm);
 Serial.print("\t vystup = ");
 Serial.println(ledProm);
  delay(2);  // pauza programu na 2 ms pre ustálenie
}
```

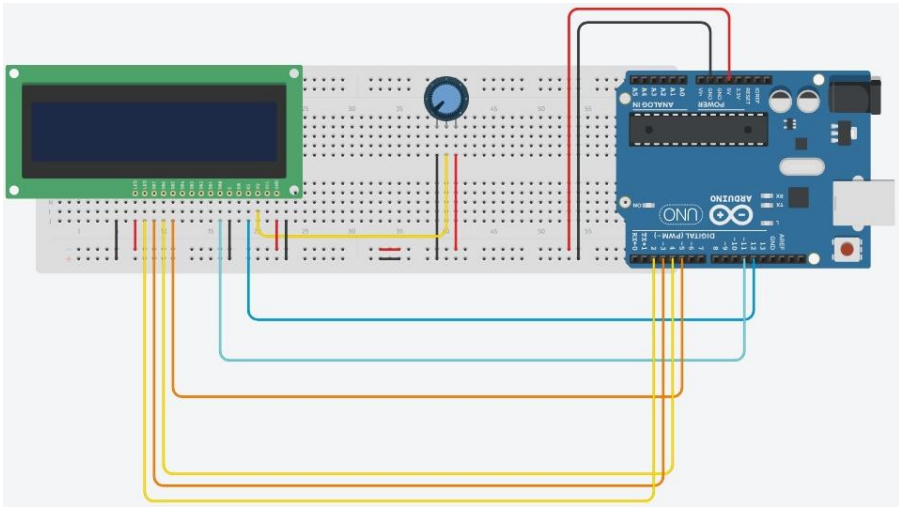
Po správnom zapojení a naprogramovaní dostaneme výsledky zobrazené v sériovom monitore ako na nasledujúcom obrázku.



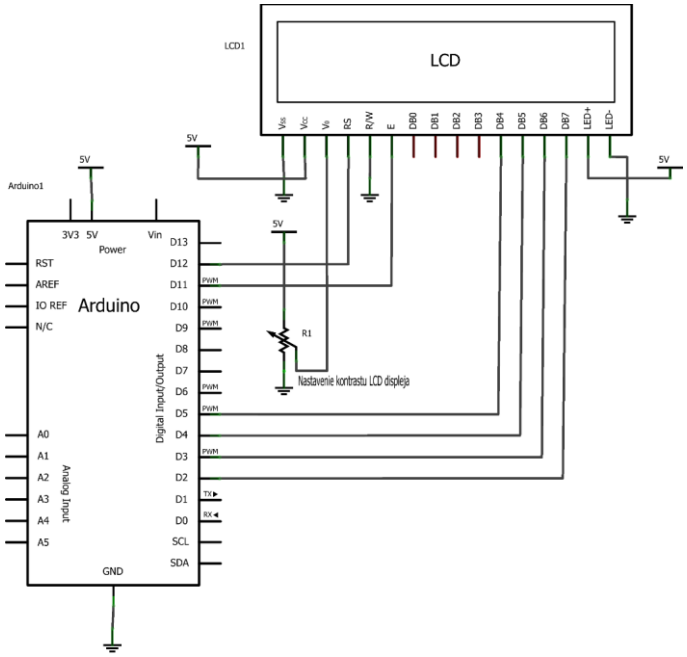
Obrázok 16: Arduino IDE a monitor sériového portu

Cvičenie č. 5: Programovanie LCD a zložitejších aplikácií

V uvádzanom cvičení si ukážeme, ako zapojiť a použiť LCD (Liquid crystal display). LCD je veľmi známy a obľúbený typ displeja. V nasledujúcich úlohách budeme využívať LCD na zobrazovanie jednoduchých informácií, ako sú teplota, vlhkosť, signalizácia zopnutia LED a podobne.



Obrázok 17: Zapojenie Arduino Uno s LCD



Obrázok 18: Schéma zapojenia pre Arduino Uno s LCD

LCD možno používať ako lacné riešenie, prostredníctvom ktorého môžeme zobrazovať údaje a nie je pritom limitovaný pripojením k počítaču. Na obrázku je zapojený displej, ktorý má v sebe nahraný program „Hello world“. Tento príklad sa nachádza priamo v prostredí Arduino IDE v záložke súbor. Použité súčiastky sú tri. Dva rezistory tvoria tzv. delič napätia, ktorý definuje kontrast displeja. Vhodnejšie je použiť potenciometer a nastaviť hodnotu napätia experimentálne. Posledný rezistor je pripojený k podsvieteniu a limituje prúd cez LED, a tým aj celkové podsvietenie. Tento rezistor vyžaduje schéma zapojenia a výrobca v katalógových údajoch (tzv. datasheet) displeja. Niektoré displeje tento rezistor nemusia vyžadovať a v takom prípade používajú priamo 5V napájanie.

Popis pinov LCD

Piny sú na displeji označené a obvykle sa nachádzajú na ľavom hornom okraji displeja. Označenie:

- VSS – Na tento pin sa pripája GND (uzemnenie) z Arduina.
- VDD - Na tento pin sa pripája 5 V z Arduina.

- Vo - Na tento pin sa pripája potenciometer, ktorý nastavuje kontrast displeja. Ten je možné nastaviť potenciometrom alebo spomínaným deličom napätia. Spravidla ale stačí nastaviť výstupné napätie na 3V.
- RS - High - Data. Low – Inštrukcie (napríklad na digitálny pin 12 Arduina).
- R/W - Zapisovanie alebo čítanie dát. Zvyčajne zapisujeme, preto sem pripojíme GND.
- E - Enable signal (napríklad na digitálny pin 11 Arduina).
- DB0 - Data. V obvyklom štvorbitovom móde býva nezapojený.
- DB1 - Data. V obvyklom štvorbitovom móde býva nezapojený.
- DB2 - Data. V obvyklom štvorbitovom móde býva nezapojený.
- DB3 - Data. V obvyklom štvorbitovom móde býva nezapojený.
- DB4 - Data. V obvyklom štvorbitovom móde býva zapojený. Napr. digitálny pin 5
- DB5 - Data. V obvyklom štvorbitovom móde býva zapojený. Napr. digitálny pin 4
- DB6 - Data. V obvyklom štvorbitovom móde býva zapojený. Napr. digitálny pin 3
- DB7 - Data. V obvyklom štvorbitovom móde býva zapojený. Napr. digitálny pin 2
- A - Kladný pól podsvietenia displeja. Tento parameter je dobré nastaviť presne podľa katalógových údajov, nakoľko existujú dve možnosti:
- niektoré displeje sem umožňujú pripojiť priamo 5V a napájajú LED cez predradený rezistor,
- iné displeje tento rezistor nemajú a bez predradeného odporu by LED zničili.
- K - Záporný pól podsvietenia. Sem sa pripojí GND.

Knižnica na prácu s LCD

Na prácu s LCD sa vo vývojom prostredí Arduino IDE najčastejšie používa knižnica LiquidCrystal. Táto knižnica je už súčasťou prostredia a netreba ju doinštalovať. Stačí, ak ju v programe zadefinujeme a budeme používať. Uvádzaná knižnica má mnoho príkazov a my sa s nimi postupne oboznámime

pomocou vzorových príkladov. Treba poznamenať, že existujú aj iné knižnice na prácu s LCD, ale tie nie sú také rozšírené.

Program pre základné ovládanie LCD

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);          //Vyberieme piny pre LCD
void setup()                    //Základné nastavenia programu
{
 //Spustenie komunikácie s LCD displejom
   lcd.begin(16, 2); //Čísla 16 a 2 značia, že ide o 16 znakový displej s 2 radmi
}
void loop()   //Opakovanie programu
{
 lcd.setCursor(0,0);   //Funkcia "setCursor" slúži na určenie začiatku textu
 lcd.print("Arduino"); //Funkcia "print" zapíše text, hodnoty a pod.
lcd.setCursor(0,1);   //Funkcia "setCursor" slúži na určenie začiatku textu
lcd.print("Ako sa mas?"); //Funkcia "print" zapíše text, hodnoty a pod.
}
```

V prvej časti programu sa nachádza príkaz, ktorý zahŕňa knižnicu potrebnú pre prácu s LCD a taktiež obsahuje aj príkaz LiquidCrystal. Táto funkcia deklaruje určitý objekt (programový objekt) s konkrétnym názvom, pod ktorým ho budeme ďalej volať v našom programe. Zároveň v tejto funkcii definujeme aj to, na ktoré vývody (piny) Arduina je LCD pripojený.

Vývody sa definujú v tomto poradí: RS, EN, D4, D5, D6, D7. Je teda zrejmé, že táto funkcia používa zapojenie pinov podľa tabuľky č. 2 (v prípade, že sme použili už hotový LCD shield). Ale ako sme už spomenuli, nič nám nebráni použiť iné zapojenie a v tejto funkcii nadefinovať správne piny. Takže ak pripojíme LCD displej podľa schémy na obrázku 21, deklarácia objektu bude LiquidCrystal lcd(12, 11, 5, 4, 3, 2) V časti setup() sa nachádza iba jeden príkaz lcd.begin(16, 2). Metóda begin objektu lcd aktivuje displej a definuje, koľko má stĺpcov (znakov na riadok) a riadkov. V našom prípade máme dvojriadkový displej so 16 znakmi na riadok. Keby sme mali pripojený iný displej, napríklad 4-riadkový s 20 znakmi na riadok, použili by sme príkaz lcd.begin(20, 4)). Existujú aj 2-riadkové displeje s 20 znakmi a jednoriadkové so 16 znakmi.

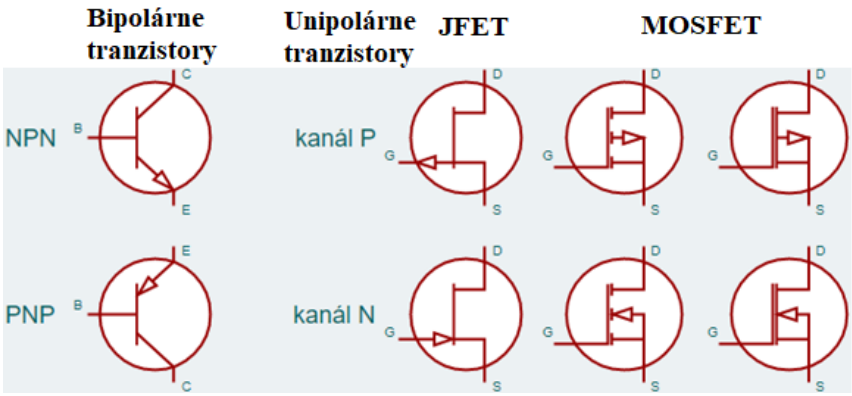
V slučke loop() príkaz lcd.SetCursor nastavuje pozíciu kurzora displeja. Pozíciu číslujeme od ľavého horného rohu displeja a to tak, že prvý znak má poradové číslo 0 (nula) a prvý riadok takisto číslo 0 (nula). Zápis lcd.setCursor(0,0) teda nastaví kurzor na pozíciu prvého znaku prvého riadka (obdobne príkaz lcd.setCursor(0,1) nastaví kurzor na pozíciu prvého znaku druhého riadka. Príkaz lcd.print("Arduino") vypíše text „Arduino“ na nastavenom riadku. Ďalším príkazom setCursor presunieme kurzor na druhý riadok a posledným príkazom lcd.print vypíšeme text: Ako sa mas?

Cvičenie č. 6: Arduino a spínacie prvky relé a tranzistor

V tomto cvičení sa budete zaoberať spínaním prostredníctvom vývojovej dosky Arduino Uno. V prípade, ak budeme spínať pomocou Arduina, narazíme na jeden veľký problém - pin má výstupný prúd len 20mA, v krátkodobých záťažiach 40mA. V takýchto prípadoch používame externé súčiastky, ktoré nám toto spínanie umožnia. V tomto cvičení si ukážeme dve možnosti spínania, a to prostredníctvom tranzistora a relé. Obe súčiastky majú svoje výhody aj nevýhody. Na ich ukážku použijeme dva kódy - Blink z Examples (každú sekundu blikne s LEDkou) a Melody.

Teoretický rozbor : Tranzistor

Tranzistor je základným konštrukčným prvkom takmer každého elektronického zariadenia. Základom bipolárneho tranzistora je kryštál polovodiča s dvoma priechodmi PN. Polovodičové priechody tranzistora vytvárajú štruktúru zodpovedajúcej spojenie dvoch polovodičových diód v jednej súčiastke. Avšak väčšinu vlastností tranzistorov sa nedá nahradiť touto dvojicou diód. Pri zhotovovaní tranzistorov sa používajú rôzne technológie. Podľa princípu činnosti sa tranzistory delia na bipolárne a unipolárne.



Obrázok 19: Schematické značky bipolárnych a unipolárnych tranzistorov

Typy tranzistorov

Každý tranzistor má (najmenej) tri vývody, ktoré sa u bipolárnych tranzistorov označujú ako kolektor (C alebo K), báza (B) a emitor (E). V prípade unipolárnych sa tieto vývody označujú drain (D), gate (G) a source (S).

Tranzistory môžeme rozlišovať na základe niekoľkých kategórií:

a) Základné typy tranzistorov podľa vnútornej štruktúry:

- Bipolárne:

  (BJT – Bipolar Junction Transistor) sú riadené prúdom vchádzajúcim do bázy tranzistora. Tu rozlišujeme dve podkategórie a to PNP a NPN tranzistory.

- Unipolárne:

  (FET – Field Effect Transistor) sú riadené napätím (elektrostatickým poľom) na riadiacej elektróde (gate). V tomto prípade rozlišujeme nasledovné podkategórie:

  - JFET – (Junction FET) riadiaca elektróda je tvorená záverne polarizovaným priechodom PN.
  - MESFET – (Metal Semiconductor FET) riadiaca elektróda je tvorená záverne polarizovaným priechodom kov-polokov.
  - MOSFET – (Metal Oxide Semiconductor FET) riadiaca elektróda je izolovaná od zvyšku tranzistora oxidom. Ich výkonnostné varianty majú medzi Drain a Source takzvanú body diódu, ktorá im pomáha zvládať napäťové špičky opačného napätia,

spôsobené rýchlym odpájaním induktora (napr. Motora, kde sa MOSFET na riadenie často používa).

- MISFET – (Metal Insulated Semiconductor FET) obecný názov pre tranzistor s izolovanou riadiacou elektródou.

b) Rozdelenie tranzistorov podľa výkonu:

- Bežné tranzistory:

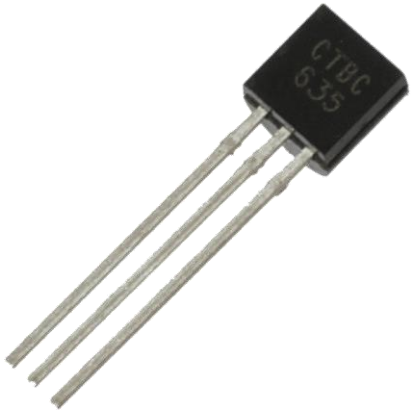
  Slúžia pre spracovanie signálu (či už ako jednotlivé "diskrétne" súčiastky alebo ako súčiastky v integrovaných obvodov) a sú dnes základným prvkom spotrebnej elektroniky (televízory, rádiá, počítače, mobilné telefóny, atď). Bežné tranzistory zvyčajne spracúvajú signál v jednotkách voltov, pričom ich prúd býva nameraný v rádoch mA (miliampérov). Od počiatku vývoja tranzistorov je snaha o minimalizáciu oboch elektrických veličín, taktiež strát energie v súčiastke. Z toho vyplýva aj snaha o čo možno najväčšiu efektivitu spracovania informácie.

- Výkonové tranzistory:

  Sú kľúčovým prvkom používaným vo výkonovej elektronike, napríklad v oblasti spínaných zdrojov alebo frekvenčných meničov. Výkonová elektronika je taktiež kľúčová pri realizácii súčasných zdrojov svetla (úsporné žiarovky, LED), moderných trakčných vozidiel s asynchrónnymi motormi, hybridných automobilov a elektromobilov. Súčasné výkonové tranzistory (IGBT) sú schopné v spínacom režime pracovať s napätím až niekoľko kilovoltov a s prúdmi v ráde stoviek alebo tisícok ampérov.

- Stredne výkonné tranzistory:

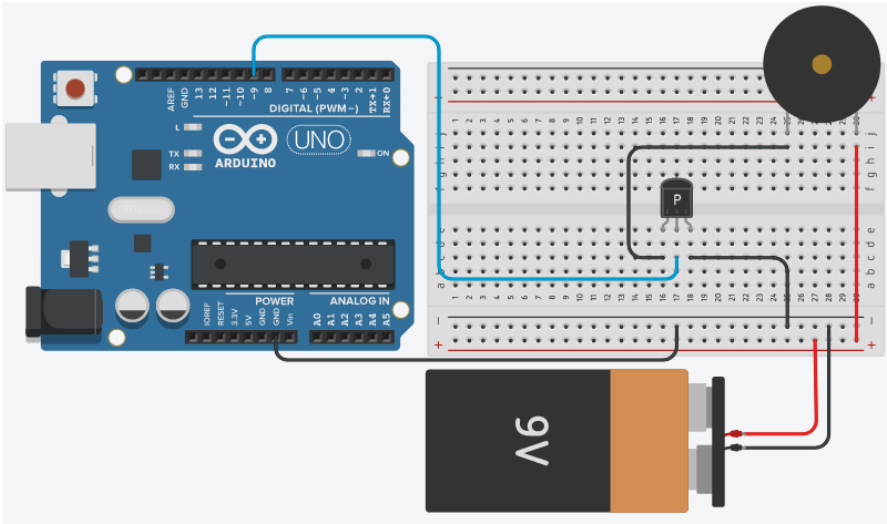
  Sú niekde medzi bežnými a výkonovými tranzistormi (často ako parametrami, tak aj fyzickou funkciou). Prevádzkované sú v lineárnom režime a používajú sa napríklad pre lineárne regulátory napätia alebo pre výkonové stupne zvukového zosilňovača.



Obrázok 20: Tranzistor

Prvou možnosťou, ako spínať obvod, je použiť tranzistor. V tomto prípade ide o polovodičovú bipolárnu súčiastku s prechodom PNP alebo NPN, no my použijeme len PNP tranzistor. Ten by sme mohli opísať ako "vypínač ovládaný prúdom". Má 3 vývody - emitor, bázu a kolektor (úplný popis sme absolvovali v druhom ročníku na predmete elektronika a praktické cvičenia z elektroniky). Vďaka polovodičom možno spínať extrémne rýchlo, teda tranzistor zvládne aj použitie a zapojenie ako "zosilňovač" pri hraní melódie.

Ako príklad použijeme program s názvom Melody. Budeme ho ale chcieť prehrať na väčších reproduktoroch, aby bol zvuk hlasnejší. Tento úkon má hneď niekoľko problémov - Arduino nemá dostatočný výkon pre väčšie reproduktory, a navyše by sa mohlo zničiť. Z tohto dôvodu pripojíme do obvodu tranzistor, ktorý použijeme ako "zosilňovač" pre reproduktory, ktoré budú napájané napätím 12V. Tranzistor použijeme z kategórie bipolárnych, a to NPN typu BC337, BC327 alebo 2N2222, 2N3904. Emitor tranzistora pripojíme na GND (voči zemi) spínaného zariadenia, kolektor na GND (zem) napájania a bázu na pin Arduina. V prípade, ak by došlo k prerazeniu tranzistora, je vhodné medzi Arduino a tranzistor umiestniť diódu tak, aby nám do pinu Arduina neprešiel veľký prúd. Batéria slúži len pre napájanie obvodu s reproduktorom a samotné Arduino budeme napájať prostredníctvom USB alebo z iného externého zdroja. Dôležité ale je, aby boli obvody pripojené na rovnaké GND (zem).



Obrázok 21: Zapojenie Arduina s tranzistorom a reproduktorom

Program Melody

```
int speakerPin = 9;
int length = 15; // počet tónov
char notes[] = "ccggaagffeeddc "; // medzera na konci predstavuje pauzu
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;
void playTone(int tone, int duration) {
for (long i = 0; i < duration * 1000L; i += tone * 2) {
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
delayMicroseconds(tone);
}}
void playNote(char note, int duration) {
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
for (int i = 0; i < 8; i++) {
```

```
if (names[i] == note) {
playTone(tones[i], duration);
}}}
void setup() {
pinMode(speakerPin, OUTPUT);
}
void loop() {
for (int i = 0; i < length; i++) {
if (notes[i] == ' ') {
delay(beats[i] * tempo); // rest
} else {
playNote(notes[i], beats[i] * tempo);
}
delay(tempo / 2);
}}
```

Spínanie prostredníctvom Relé

Ďalšou možnosťou, ako spínať väčšie záťaže, je použiť Relé. Ide tiež o elektronicky ovládaný spínač, avšak na celkom inom princípe. Vyrábajú sa rôznymi spínacími napäťovými úrovňami (5V, 12V, 24V, 230V). Pre Arduino sú vyrábané 5V Relé moduly, ktorého vývody stačí pripojiť GND na GND (zem), VCC na 5V Arduina, a potom vstupom IN (prípadne IN1, IN2,atď, u modulov s viacerými Relé v rámci jedného obvodu) ho prostredníctvom Arduina spínať. Spôsob programovania Relé je taký, že funkcia digitalWrite () má dva stavy – HIGH (logická 1 (5V)) a LOW (logická 0 (0V)). Stav HIGH privedie na IN Relé napätie, takže je tento pin nastavený digitalWrite (pin, HIGH). V tomto prípade Relé zopne. V prípade ak je privádzaný stav LOW, tak Relé rozopne svoje kontakty.

Obrázok 22: Modul s Relé

Výhodou tejto súčiastky je to, že prostredníctvom Arduina ním môžeme spínať prakticky ľubovoľné zariadenie. Pomocou výstupu Relé môžeme spínať aj sieťové napätie 230V (avšak musíme dodržať limity Relé, ktoré sú určené výrobcom a to spravidla 10 – 16A). Naproti tranzistoru má Relé ale jednu nevýhodu. Relé nie je schopné spínať vo vyššej frekvencii. Spína len niekoľkokrát za sekundu, v inom prípade hrozí jeho poškodenie. V prípade zapojení ako prehranie melódie je nepoužiteľné.

Výstup Relé obsahuje 3 piny (zvyčajne svorky) - NC, NO, COM.

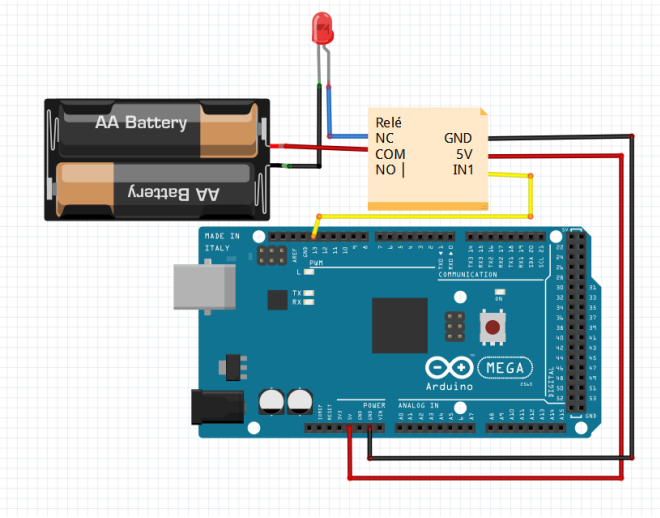
- COM - Označuje vstup (tento výstup pripájame vždy)
- NO – Normal Open. - Relé je v polohe rozopnuté a spíname ho privedením signálu na IN.

- NC - Normal Closed – Relé je v polohe zopnuté a rozopíname ho privedením signálu na IN.

Program pre spínanie relé v sekundových intervaloch

```
//Tento program funguje tak, že každú sekundu zopne výstup Relé
int led = 13;
void setup() {
  pinMode(led, OUTPUT);  //Nastaví digitálny pin arduina ako výstupný.
}
void loop() {
  digitalWrite(led, HIGH);   // privedie signál na výstup digitálneho pinu
  delay(1000);               // počká 1000ms a teda 1 sekundu
  digitalWrite(led, LOW);    // vypne signál, ktorý prichádza na digitálny pin
  delay(1000);               // počká 1000ms a teda 1 sekundu
}
```

V ďalšom kroku pripojíme Relé a na jeho výstup obvod s napájaním, ktorý budeme spínať. Môžeme použiť napr. LED pás, ktorý je napájaný sieťovým napätím alebo zariadenie, ktoré je napájané jednosmerným napätím 24V, atď. Vybrané zariadenie pripojíme tak, ako by malo svietiť trvalo, len na jeden z vodičov (spravidla rozdeľujeme fázový vodič v prípade sieťového napätia alebo vodič s + pólom v prípade jednosmerného obvodu) pripojíme Relé, ktoré nám tento obvod bude spínať (alebo rozpínať). Z USB budeme napájať Arduino a spínané zariadenie budeme napájať externým zdrojom napätia. Zapojenie realizujeme podľa schémy:



Obrázok 23: Zapojenie RElé a LED s externým zdrojom pre LED

Cvičenie č. 7: Riadenie DC motora

Funkcia a popis činnosti DC motora.

V prvom kroku cvičení je potrebné predstaviť princíp činnosti prostredníctvom videa, kde nám bude názorným spôsobom demonštrovaná činnosť DC motora.

https://www.youtube.com/watch?v=I7IFsQ4tQU8&ab_channel=HowToMechatronics

https://www.youtube.com/watch?v=sOz41WQF7wE&ab_channel=eTechTom

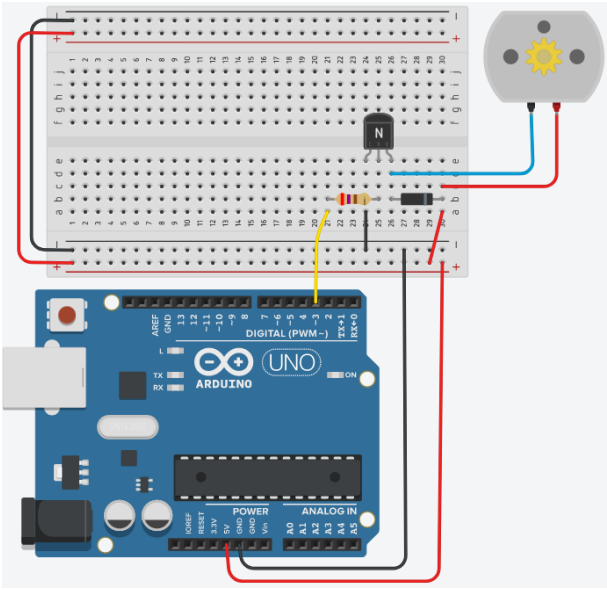
Obrázok 24: QR kódy pre načítanie web stránky s príslušnými videami

Poznámka:

Motorček nepripájame priamo k vývojovej doske Arduino, nakoľko by tento spôsob mohol dosku zničiť. Z tohto dôvodu je potrebné použiť pomocný riadiaci obvod ako napr. tranzistor alebo iný integrovaný obvod, ako tzv. H-most (súbor tranizistorov).

Pre uvádzaný obvod budeme potrebovať nasledujúce súčiastky:

- 1x Arduino UNO
- 1x PN2222 Tranzistortor
- 1x 6V DC Motor
- 1x 1N4001 diódu
- 1x 270 Ω odpor



Obrázok 25: Zapojenie DC motorčeka prostredníctvom tranzistora NPN

Cvičenie č. 8: Riadenie servomotora

Funkcia a popis činnosti servomotorčeka.

V prvom kroku cvičení je opäť potrebné predstaviť princíp činnosti prostredníctvom videa, kde nám bude názorným spôsobom demonštrovaná činnosť servomotorčeka.

https://www.youtube.com/watch?v=LXURLvga8bQ&ab_channel=HowToMechatronics

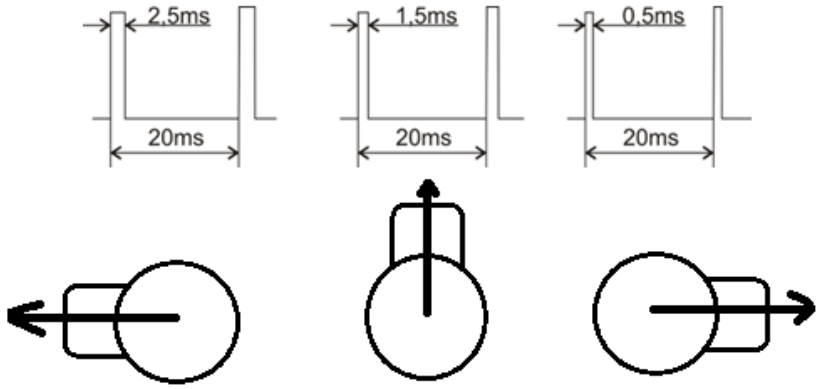
https://www.youtube.com/watch?v=J8atdmEqZsc&ab_channel=GreatScott%21

Obrázok 26: QR kódy pre načítanie web stránky s príslušnými videami

Popis činnosti:

Servomotory sú navrhnuté tak, aby sa ich rameno nastavilo do určitej polohy, ktorú nastavíme. Následne v tejto polohe zotrvá nastavený čas. Jednosmerné servomotory majú všestranné využitie ako napr. ovládanie robotickej ruky, riadenie kormidla pri leteckých alebo lodných modeloch, či riadenie ramena rampy na parkovisku. Ich hlavnou výhodou je malý rozmer a malá hmotnosť s relatívne veľkou silou.

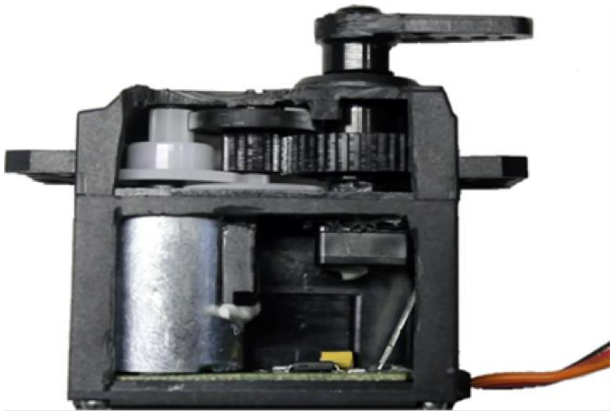
Tieto motory obvykle neumožňujú otáčavý pohyb okolo svojej osi, ale udržujú nastavený uhol natočenia. Uhol servomotora sa pohybuje najčastejšie v rozsahu od 0° až po 180°. Niektoré servomotory umožňujú natáčanie ramena aj v uhle 360°. Nastavenie tohto uhla sa vykonáva poslaním impulzu určitej dĺžky. Neutrálna poloha (90°) zodpovedá zvyčajne dĺžke impulzu 1,5ms. Dĺžka 0,5ms zodpovedá uhlu 0° a impulz dĺžky 2,5ms nastaví uhol 180°. Tieto impulzy Arduino posiela do riadiaceho obvodu motorčeka každých 20ms.



Obrázok 27: Princíp činnosti servomotorčeka

Vnútorná štruktúra servomotorčeka:

Vo vnútri servomotorčeka sa nachádza jednosmerný motorček (DC motorček, ktorý sme sa učili zapájať a riadiť v predchádzajúcom cvičení). Výstup DC motorčeka je zapojený cez prevody a riadiaci obvod, ktorý je vyvedený na výstupné rameno serva. Na výstupe obvodu prebieha meranie polohy pomocou integrovaného potenciometra. Ten sa môže otáčať iba v určitom rozsahu, a to je aj hlavný dôvod, prečo nemožno otáčať servom dookola (aj keď pri 360° servomotore to môže tak vyzerať). Nachádza sa tu aj riadiaca elektronika, ktorá prijíma impulzy a porovnáva ich dĺžku s nastaveným odporom potenciometra.



Obrázok 28: Prierez servomotora

Použitý potenciometer v servomotorčeku má celkový odpor 1800Ω. Dĺžka impulzu 0,5ms by podľa teórie zodpovedala odporu 0Ω, pre impulz o dĺžke 1,5ms by táto hodnota bola 900Ω a pre impulz 2,5ms by hodnota zodpovedala

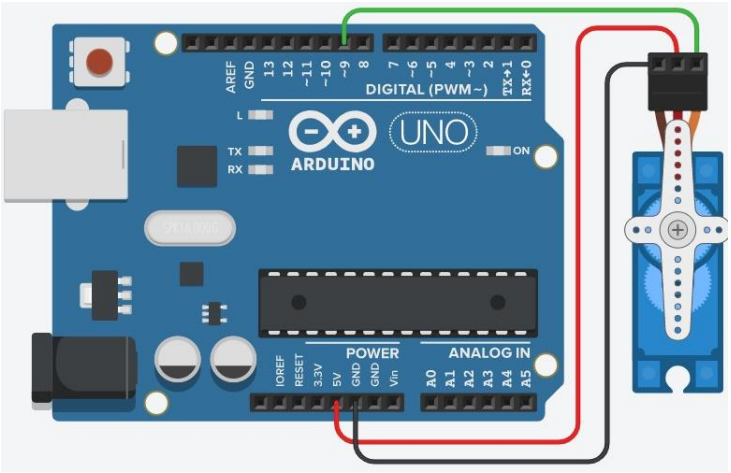
odporu celých 1800Ω. V prípade, ak bude motor natočený v uhle 90 °, jeho odpor bude 900Ω. V prípade, ak dostane impulz s dĺžkou 2,5ms, potom elektronika vie, že cieľový odpor má byť 1800Ω a privedie do motorčeka prúd v kladnom smere. Prúd do servomotorčeka prúdi po príchode impulzu vždy pár milisekúnd. Tým sa o krok priblíži k cieľovej hodnote. S ďalším impulzom dĺžky 2,5ms elektronika znovu zmeria odpor potenciometra a pokiaľ bude menší ako 1800Ω, potom znovu privedie do motorčeka prúd v kladnom smere.

V prípade, že privedieme do servomotorčeka riadiaci impulz s dĺžkou 0,5ms, elektronika vie, že cieľový odpor 0Ω je menší než aktuálna hodnota a privedie do motorčeka prúd v opačnom smere. Tým sa servo natočí o krok smerom k uhlu 0 °. Čím viac sa bude odpor potenciometra blížiť k cieľovej hodnote, tým kratšiu dobu bude elektronika do motorčeka privádzať prúd. Ak bude na výstupné rameno motorčeka pôsobiť len malá sila, ktorá bude servo vychyľovať od cieľového uhla, bude mať motorček tiež malú spotrebu.

Z teoretických vedomostí vieme, že uhol natočenia v servomotorčekoch môže byť rozdielny. Záleží hlavne na tom, aký potenciometer má servomotorček integrovaný. Ako sme uvádzali vyššie, niektoré servomotorčeky majú rozsah 180°, 270 °, alebo dokonca celých 360 °. Súčasťou konštrukcie servomotorčka je aj tzv. doraz, ktorý bráni pretočeniu rozsahu potenciometra cez maximálny uhol, a tým zabráni jeho poškodeniu. Práve preto sa neodporúča vykonávať otáčavý pohyb, a tým narážať do koncových dorazov, nakoľko v tom momente pôsobí na prevody relatívne veľká sila a častým narážaním by došlo k jeho poškodeniu.

Zapojenie servomotorčeka

Jednosmerný servomotorček obsahuje spravidla tri vývody. Jeden pre napájanie (zvyčajne červený), druhý pre GND (čierny alebo hnedý) a tretí pre riadiace impulzy (žltý alebo oranžový). Napájací vývod pripájame na 5V, uzemnenie na GND a riadiaci vývod na digitálny pin (napr.D9).



Obrázok 29: Zapojenie arduino a servomotorček

Popis programu

Pre riadenie servomotrčeka obsahuje Arduino IDE pripravenú knižnicu, ktorá významným spôsobom uľahčuje jeho riadenie. Pre testovanie uvádzaného zapojenia použijeme ukážku, ktorá sa nachádza vo vývojovom prostredí Arduino IDE v záložke Súbor/Príklady/Servo/. Máme na výber z možností „Knob“ alebo „Sweep“. Tieto programy postupne menia uhol natočenia z 0° až k 180° a späť.

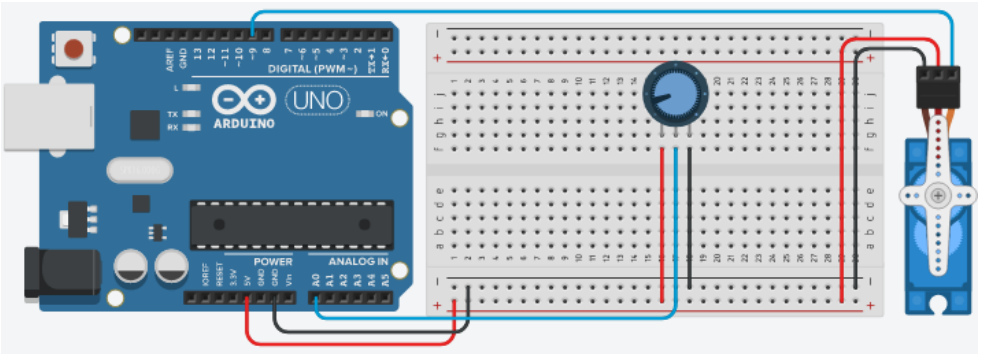
```
# include  <Servo.h>     // zahrnutie knižnice pre ovládanie servo motora
Servo myservo;          // každý motor má svoju inštanciu triedy Servo
int pos = 0 ;          // premenná obsahujúca pozíciu motora (uhol natočenia)
void  setup ()
{
 myservo. attach ( 9 );       // tento motor je pripojený na pin 9
}
void loop ()
{ for (pos = 0 ; pos <= 180 ; pos += 1 )       // natočenie uhla od 0 po 180

 {
  myservo. write (pos);     // natočenie motora na aktuálny uhol
  delay ( 15 );                          // čakanie
```

```
 } for (pos = 180 ; pos>= 0 ; pos -= 1 )          // natočenie uhla od 180 po 0
 {
   myservo. write (pos);  // natočenie motora na aktuálny uhol
delay ( 15 );         // čakanie
 } }
```

Otáčanie servomotorčeka potenciometrom

V tomto príklade použijeme rovnako knižnicu Servo v programovacom prostredí Arduino IDE. Avšak k tomu obvodu pridáme navyše externý potenciometer a nahráme program „Knob“.



Obrázok 30: Zapojenie Arduino, potenciometer a servomotorček

Program pre servomotorčeka potenciometrom

```
# include  <Servo.h>     // zahrnutie knižnice pre ovládanie servo motora
Servo myservo;        // každý motor má svoju premennú typu Servo
int potpin = 0 ;       // pin, ku ktorému je pripojený potenciometer
int val;              // premenná pre načítanie a nastavenie uhlu
void  setup ()
{
  myservo. attach ( 9 );   // tento motor je pripojený na pin 9
} void loop ()
{
```

```
  val = analogRead (potpin);   // napätia na potenciometra (0 až 1023)
  val = map (val, 0 , 1023 , 0 , 180 ); // prevod z 0 až 1023 na 0 až 180
  myservo. write (val);       // polohovania podľa potenciometra
  delay ( 15 );               // chvíľka čakania, kým sa motor natočí
}
```

Popis programu

Na rozdiel od predchádzajúceho programu sa tu nachádza navyše premenná int potpin = 0; V tejto je uložené číslo analógového pinu, ku ktorému je pripojený potenciometer. Pin musí byť analógový, aby na ňom bolo možné merať napätie pomocou val = analogRead (potpin); Týmto spôsobom sa do premennej val uloží hodnota v rozsahu 0 až 1023 podľa polohy potenciometra. Ak natočíme potenciometer po ľavý doraz, bude nameraná hodnota 0, ak natočíme potenciometer na stred, nameriame hodnotu 512 a na pravom doraze to bude hodnota 1023.

Nakoľko funkcia write akceptuje iba rozsah 0 až 253, je potrebné nameraný rozsah 0 až 1023 prepočítať na rozsah 0 až 253. Môžeme to realizovať jednoduchým vydelením nameranej hodnoty číslom 4 alebo využitím univerzálnej funkcie map. Táto funkcia prepočíta hodnotu z ľubovoľného vstupného rozsahu na zodpovedajúcu hodnotu v rozsahu výstupu.

Dĺžky impulzov pre natočenie na určitý uhol sa môžu líšiť podľa rôznych modelov a výrobcov servomotorčekov. Knižnica Servo umožňuje nastaviť vlastné hodnoty pre minimálny a maximálny uhol natočenia. Funkcia attach (pin, min, max) preberá voliteľné parametre, ktoré udávajú minimálnu a maximálnu dĺžku impulzu v mikrosekundách. Uhol natočenia možno tiež zadať priamo pomocou dĺžky impulzu funkciou writeMicroseconds (us).

Cvičenie č. 9: Riadenie krokového motora

Funkcia a popis činnosti krokového motorčeka.

V prvom kroku si pozrieme princíp činnosti prostredníctvom videa, kde nám bude názorným spôsobom demonštrovaná jeho činnosť.

https://www.youtube.com/watch?v=TWMai3oirnM&ab_channel=HowToMechatronics

https://www.youtube.com/watch?v=eyqwLiowZiU&t=173s&ab_channel=LearnEngineering

Obrázok 31: QR kódy pre načítanie web stránky s príslušnými videami

Riadenie krokového motorčeka (Stepper)

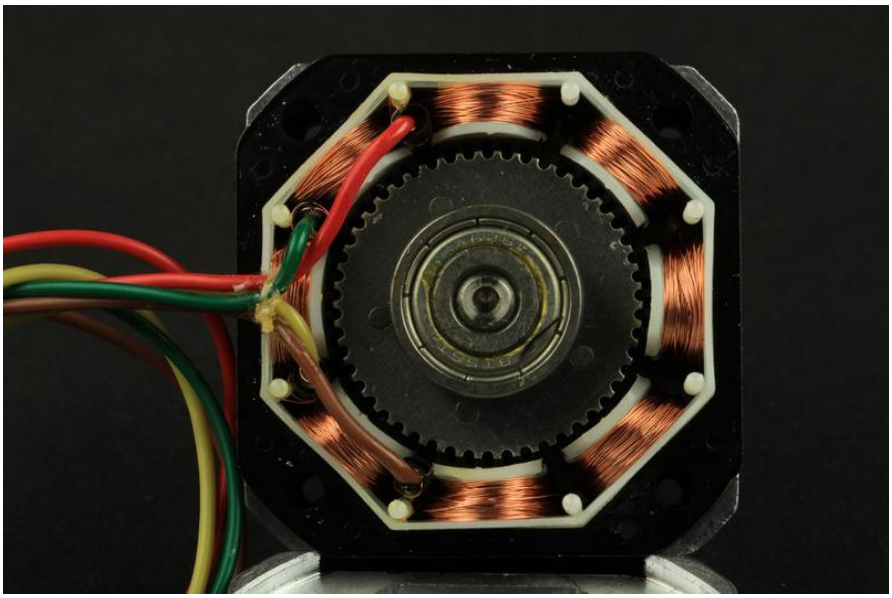
Krokové motory sa používajú všade tam, kde je potrebné presné riadenie otáčok, či už v automatizačnej technike alebo robotike.

Obrázok 32: Krokový motor

Princíp činnosti

Krokový motor je synchrónny jednosmerný motor (rotor sa točí rovnakou rýchlosťou ako točivé magnetické pole v statore). Točivé magnetické pole nie je vytvárané striedavým prúdom, ale postupným zapínaním jednotlivých cievok statora. Stator motora sa skladá z niekoľkých dvojíc cievok (zvyčajne 4 dvojice), ktoré môžu byť rôzne zapojené (vyvedené obe strany cievky, dve a dve cievky spojené jednou stranou vinutia, všetky cievky so spoločnou jednou).

Rotor je zväčša tvorený valčekom z magneticky mäkkého alebo tvrdého materiálu s pólmi.



Obrázok 33: Prierez krokového motora

Poloha A - Motor je v prvej polohe, pretože prúd tečúci cievkami spôsobuje magnetický tok, ktorý prechádza miestom s najnižším magnetickým odporom, a teda cez rotor krokového motora. Ostatnými cievkami nepreteká žiadny prúd.

Poloha B - Prepnutím aktívnej cievky sa vytvorí magnetický tok na inom mieste. Rotor sa teda natočí tak, aby kládol čo najnižší magnetický odpor, teda o 60 ° doľava. Rýchlym a postupným prepínaním jednotlivých dvojíc cievok sa zaistí rotácia rotora. Motor je možné ovládať rôznymi druhmi riadenia, kde možno aktivovať vždy dve susedné cievky - rotor sa teda natočí medzi dva

pólové body statora (dve aktívne cievky spôsobia takmer dvojnásobný krútiaci moment).

Ak je rotor z magneticky tvrdého materiálu (magnetický), môže jedna cievka tlačiť, druhá ťahať.

Počet krokov je možné aj zvýšiť (tzv. mikrokrokovať) postupným zapínaním jednotlivých cievok - motor má následne plynulejší chod. Práve posledné uvádzané sa využíva veľmi často.

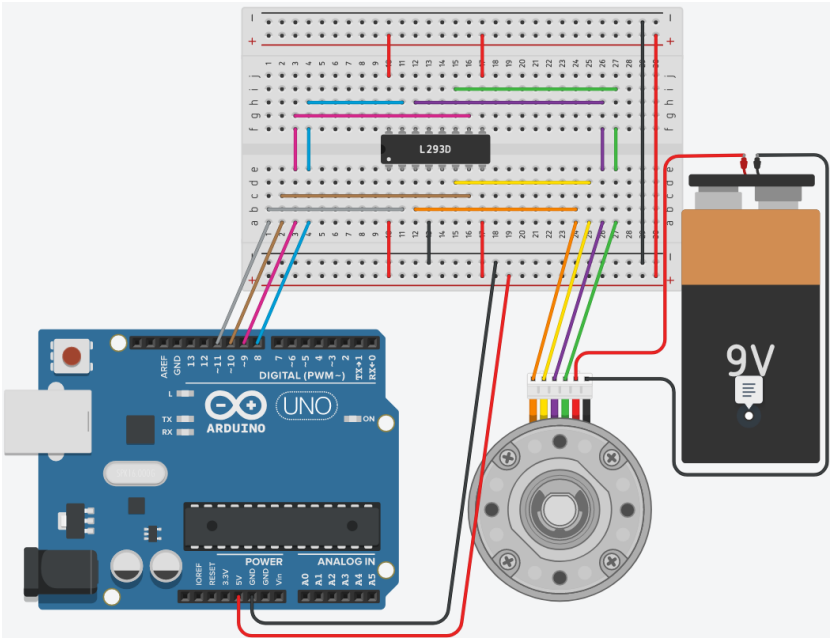
Ak cievky medzi sebou prepínajú veľmi rýchlo, prejavuje sa tzv. strata kroku a znižuje sa aj účinnosť. Tento jav nastáva v prípade, ak sa točivé magnetické pole rotora nedokáže dostatočne rýchlo otočiť. Strata kroku môže nastať taktiež pri veľkom mechanickom zaťažení motora. V bežnej praxi sa stretávame s krokovými motormi riadenými prostredníctvom riadiacich obvodov "driverov" alebo tiež tzv. H-mostov (H-bridge).

Popis činnosti riadiacich obvodov pre krokové motory (driver):

Väčšina krokových motorov funguje len prostredníctvom vyššie uvedeného ovládacieho obvodu. Je to preto, že riadiaci modul (v našom prípade Arduino) nie je schopné poskytovať dostatočný prúdu zo svojich výstupov (pinov), ktoré by dokázali zabezpečiť prevádzku motora. Ako ovládač krokového motora sa teda používa externý modul, ako napr. ULN2003. Existuje mnoho typov ovládacích modulov. Ich vlastnosti sa menia na základe typu použitého motora.

Obrázok 34: Príklad krokového motora s riadiacim obvodom

Pre úspešné prepojenie krokového motora s Arduinom je nutné v prvom rade prepojiť motor s driverom pomocou konektora (zväčša päť pinov). V ďalšom kroku musíme prepojiť driver a Arduino šiestimi prepojovacími pinmi. Pripojíme IN1 na D8, IN2 na D9, IN3 na D10 a následne z napájacích pinov - na GND a + na 5V. Napájanie je však vhodné pri väčšej záťaži pripojiť prostredníctvom externého zdroja, napríklad batérie.



Obrázok 35: Zapojenie Arduino a krokový motor

Uvádzaný program obsahuje prácu s krokovým motorom. Po spustení programu sa vykoná otočenie hriadeľa krokového motora o plných 360 stupňov. Na začiatku kódu nastavíme čísla pinov pre digitálne výstupy a vytvoríme premennú, pomocou ktorej sa nastavuje rýchlosť - 1 zodpovedá najvyššej rýchlosti a s väčšími číslami (doba medzi spínaním) sa rýchlosť znižuje a tiež vytvoríme premennú na nastavenie uhla otočenia hriadeľa. V časti programu setup nastavíme všetky digitálne piny ako výstupné. V časti programu void loop potom pomocou funkcie for vykonáme otáčku o nastavený uhol v smere hodinových ručičiek, počkáme jednu sekundu a vykonáme rotáciu proti smeru hodinových ručičiek. V ďalšej časti podprogramu sa nastavia jednotlivé kroky a ich postupnosti tak, ako udáva výrobca motora. Arduino krokový motor s driverom, (riadiacou doskou), je vhodným kompletom pre najrôznejšie projekty, ktoré vyžadujú riadený mechanický pohyb.

Nižšie uvedený postup je konkrétne zameraný na krokový motor 28BYJ-48 a driver s tranzistorovým zapojením ULN2003. Spomínaný motor je unipolárny

a ide o elektromechanické zariadenie, ktoré premieňa elektrické impulzy na mechanický pohyb. Medzi výhody tohto krokového motora patrí jednoduché otočenie o ľubovoľný uhol, veľká sila aj v pokojnej polohe či okamžitá odozva pri zastavení aj spúšťaní motora.

Program riadenia krokového motora

```
const  int in1 =   8 ;
const  int in2 =   9 ;
const  int IN3 = 10 ;
const  int IN4 = 11 ;
int rychlost = 1 ;
int uhol = 360 ;

void  setup () {
 pinMode (in1, OUTPUT );
 pinMode (in2, OUTPUT );
 pinMode (IN3, OUTPUT );
 pinMode (IN4, OUTPUT );
}
void  loop () {
 for ( int i = 0 ; i <(uhol * 64 / 45 ); i ++) {
   rotacePoSmeru ();
 }
   delay ( 1000 );

 for ( int i = 0 ; i <(uhol * 64 / 45 ); i ++) {
   rotaceProtiSmeru ();
 }
   delay ( 1000 );
}
void rotacePoSmeru () {
 Krok1 ();
 Krok2 ();
```

```
  Krok3 ();
  Krok4 ();
  Krok5 ();
  Krok6 ();
  Krok7 ();
  Krok8 ();
}
void rotaceProtiSmeru () {
  Krok8 ();
  Krok7 ();
  Krok6 ();
  Krok5 ();
  Krok4 ();
  Krok3 ();
  Krok2 ();
  Krok1 ();
}
void Krok1 () {
   digitalWrite (in1, HIGH );
  digitalWrite (in2, LOW );
  digitalWrite (IN3, LOW );
  digitalWrite (IN4, LOW );
  delay (rychlost);
}
void Krok2 () {
   digitalWrite (in1, HIGH );
  digitalWrite (in2, HIGH );
  digitalWrite (IN3, LOW );
```

```
  digitalWrite (IN4, LOW );
  delay (rychlost);
}
void Krok3 () {
   digitalWrite (in1, LOW );
  digitalWrite (in2, HIGH );
  digitalWrite (IN3, LOW );
  digitalWrite (IN4, LOW );
  delay (rychlost);
}
void Krok4 () {
   digitalWrite (in1, LOW );
  digitalWrite (in2, HIGH );
  digitalWrite (IN3, HIGH );
  digitalWrite (IN4, LOW );
  delay (rychlost);
}
void Krok5 () {
   digitalWrite (in1, LOW );
  digitalWrite (in2, LOW );
  digitalWrite (IN3, HIGH );
  digitalWrite (IN4, LOW );
  delay (rychlost);
}
void Krok6 () {
   digitalWrite (in1, LOW );
  digitalWrite (in2, LOW );
  digitalWrite (IN3, HIGH );
```

```
  digitalWrite (IN4, HIGH );
  delay (rychlost);
}
void Krok7 () {
   digitalWrite (in1, LOW );
  digitalWrite (in2, LOW );
  digitalWrite (IN3, LOW );
  digitalWrite (IN4, HIGH );
  delay (rychlost);
}
void Krok8 () {
   digitalWrite (in1, HIGH );
  digitalWrite (in2, LOW );
  digitalWrite (IN3, LOW );
  digitalWrite (IN4, HIGH );
  delay (rychlost);
}
```

Po nahratí programu do Arduina môžeme pozorovať opakujúce sa otáčanie hriadeľa. Následne môžeme skúsiť meniť rýchlosť otáčania pomocou premennej, rýchlosť alebo uhol otočenia hriadeľa pomocou nastavenia počtu priebehov prostredníctvom príkazu for. Tento uhol vychádza z katalógových údajov výrobcu.

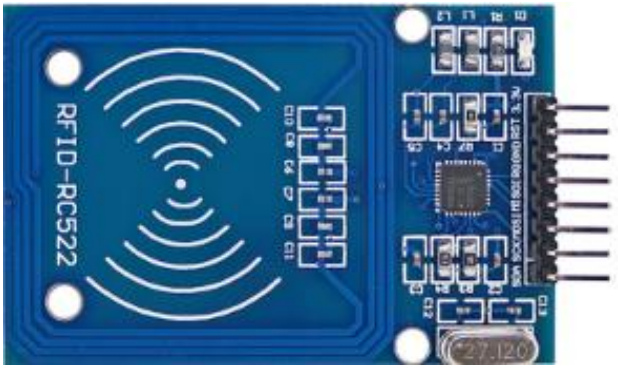
Cvičenie č. 10: Arduino a bezdrôtové zariadenie NFC

Skratka NFC pochádza z anglického spojenia Near Field Communication, a teda komunikácia na krátke vzdialenosti. Ide o prenos údajov, ktorý je podobný technológii Bluetooth alebo staršiemu štandardu infračerveného prenosu, kde jedno zariadenie informácie sprostredkúva inému prostredníctvom bezdrôtového prenosu. Avšak ak sa na NFC pozrieme bližšie, zistíme, že pre správne pochopenie potrebujeme mať znalosti z elektroniky. Táto technológia potrebuje na prenos údajov energiu v podobe elektromagnetického vlnenia.

Toto vlnenie je pri prenose dodávané cievkou, ktorá sa nachádza v prijímateľovi, teda v smartfóne, tablete alebo inom zariadení. Cievka vysiela elektromagnetické vlny o určitej frekvencii, ktoré dokáže pasívny prvok, opäť formou cievky prijať a premeniť na energiu. Ak sa zariadenie dostatočne priblíži k NFC štítku „tagu“ alebo inému pasívnemu NFC prvku, cievka v ňom elektromagnetické vlnenie pohltí a premení na energiu. Pomocou energie potom môže odosielateľ vyslať uloženú informáciu prijímateľovi. Komunikačné zariadenia musia byť veľmi blízko pri sebe, zvyčajne pár cm, ale rozsah sa môže líšiť v závislosti od zariadenia, ktoré vysiela a takisto aj od veľkosti štítku. Značky NFC nevyžadujú žiadny príkon. Používajú magnetickú indukciu medzi dvoma malými anténami. Zariadenia dokážu prenášajú od 96 až po 4 096 bajtov informácií. Pre viac informácií o činnosti tejto technológie môžete pozrieť aj https://www.mojandroid.sk/nfc-co-to-je-ako-funguje/ .

Arduino a NFC

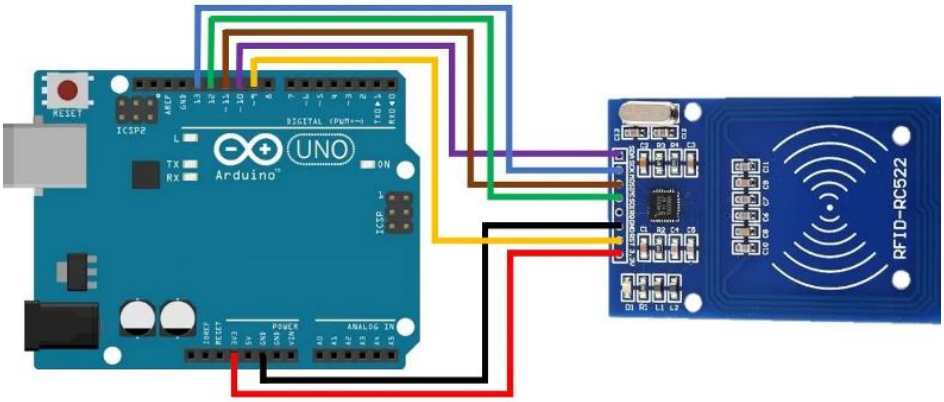
NFC čítačku tagov považujeme za vstupný modul pre Arduino. Obsahuje obvod PN532, ktorý pracuje na frekvencii 13.56 MHz a podporuje takzvané NFC tagy, kde sa za tagy považujú napr. RFID karty, žetóny, samolepky atď. Obvod komunikuje pomocou troch rôznych rozhraní a to HSU, I2C a SPI. Prepínanie medzi jednotlivými zbernicami zabezpečujú dva prepínače, ktoré sa nachádzajú na module. Pomocou ich kombinácie zapnuté / vypnuté volíme danú zbernicu. Pre naše cvičenie sme zvolili komunikáciu I2C. Pre napájanie NFC je možné použiť napätie 3,3V, aj 5V. Prúdový odber sa môže dostať pri aktívnej komunikácii až na 150 mA.



Obrázok 36: Ukážka NFC modulu pre Arduino

Pre funkčné prepojenie NFC čítačky s vývojovou doskou Arduino stačí zapojiť celkom štyri vodiče, v prípade, ak ide o verziu PN532. Prepojíme pin

čítačky GND so zemou Arduino, VCC s 3V3 Arduino, SDA s pinom A4 a SCL s pinom A5. Pre zvolenie komunikácie I2C nastavíme prepínače, umiestnené na module na kombináciu "1 | 0". Pre piny SDA a SCL musíme vybrať vždy I2C piny na nami vybranej doske, kde pre Arduino UNO sú to práve piny A4 a A5. V prípade verzie RC522 je zapojenie možné vidieť na nasledujúcom obrázku.



Obrázok 37: Zapojenie NFC, verzia RC522

Pre správnu činnosť je potrebné stiahnuť a importovať knižnicu PN532.Vzorový program obsahuje na svojom začiatku pripojenie potrebných knižníc. Ďalej je nutné vytvoriť tzv. inštanciu I2C komunikácie pre čítačku PN532. Následne vytvoríme inštanciu NFC modulu. V podprograme setup v prvom kroku zahájime komunikáciu po sériovej linke (serial begin) a potom komunikáciu s NFC modulom. V ďalšom kroku uložíme verziu dosky do lokálnej premennej, ktorá zároveň slúži ako kontrola pripojeného modulu, ako je možné vidieť v podmienke if. Pokiaľ je verzia načítaná, vypíšeme si informáciu na sériovú linku a pokračujeme v programe. Ďalším príkazom nastavíme maximálny počet pokusov o načítanie tagu v okolí, ktorý s hodnotou 0xFF zodpovedá približne jednej sekunde hľadania. Po tomto nastavení už len nakonfigurujeme NFC modul pre čítanie a vypíšeme na sériovú linku hlásenie. Na začiatku slučky loop si vytvoríme tri lokálne premenné, do ktorých sa budú ukladať výsledky meraní. A za nimi nasleduje príkaz pre čítanie, ktorý svoj výsledok uloží do všetkých nastavených premenných. Po ukončení čítania teda skontrolujeme premennú, ktorá nám v prípade hodnoty 1 označuje úspešné čítanie. Ak sa teda tak stane, vypíšeme postupne získané informácie, ako je dĺžka, teda počet bitov UID a samotné UID v HEX formáte. A za výpisom informácií nasleduje už len pauza (delay) po dobu jednej sekundy pred novým cyklom slučky. V prípade, že čítanie úspešné nebolo, postupujeme do časti programu s funkciou else, kde vypíšeme informačné hlásenie o neúspešnom čítaní.

Program pre PNC532

```
// pripojenie potrebných knižníc
# include  <PN532_I2C.h>
# include  <PN532.h>
# include  <NfcAdapter.h>
 // vytvorenie inštancie I2C komunikácie pre PN532
PN532_I2C pn532i2c ( Wire );
// vytvorenie inštancie PN532 NFC modulu
PN532 rsguys (pn532i2c);

void  setup () {
 Serial . begin ( 9600 );    // zahájenie komunikácie po sériovej linke

  rsguys. begin ();  // zahájenie komunikácie s NFC modulom
  uint32_t versiondata = nfc.getFirmwareVersion ();
  // kontrola načítané verzia NFC modulu
  if (! Versiondata) {
    Serial . println ( "Nebol nájdený PN53x modul!" );
   // zastavenie programu
   while ( 1 );
  }
  else {
    // vypíše informáciu o pripojenom module
   Serial . print ( "Nájdený modul PN5" ); Serial . println ((versiondata >> 24 ) & 0xFF , HEX);
   Serial . print ( "Firmware verzia" ); Serial . print ((versiondata >> 16 ) & 0xFF , DEC);
   Serial . print ( '.' ); Serial . println ((versiondata >> 8 ) & 0xFF, DEC);
  }
```

```
  // nastavenie maximálneho počtu pokusov o čítanie NFC tagu,
  // zodpovedá cca jednej sekunde
  nfc.setPassiveActivationRetries ( 0xFF );
  // konfigurácia NFC modulu pre čítanie tagov
  nfc.SAMConfig ();
  // informácia o začiatku čítania
  Serial . println ( "PN532 modul nastavený, prilozte tag ..." );
 }

 void  loop () {
  // vytvorenie premenných, do ktorých sa budú ukladať výsledky čítania
  boolean uspech;                    // úspešné čítanie
  uint8_t uid [] = { 0 , 0 , 0 , 0 , 0 , 0 , 0 }; // unikátne ID tagu
  uint8_t uidLength;                 // dĺžka ID tagu
  // čítanie tagov v okolí, výsledok sa uloží do nastavených premenných
  uspech = nfc.readPassiveTargetID (PN532_MIFARE_ISO14443A, & uid [ 0 ],
& uidLength);
  // ak je čítanie úspešné, vypíšeme získané informácie
  if (uspech) {
    Serial . println ( "Nájdený NFC tag!" );
   Serial . print ( "UID dlzka:" ); Serial . print (uidLength, DEC); Serial . println
( "bytu" );
   Serial . print ( "UID hodnoty:" );
   for ( uint8_t i = 0 ; i <uidLength; i ++) {
     Serial . print ( "0x"); Serial . print (uid [i], HEX);
   }
   Serial . println ( "" );
   delay ( 1000 );
  }
```

```
  else {
    // vytlačenie informácie o vypršaní času čítanie
    Serial . println ( "počas čítania nebol priložené žiadne tag." );
  }
}
```

Po nahratí ukážkového kódu do Arduino dosky s pripojenou NFC čítačkou PN532 dostaneme napríklad tento výsledok:

Nájdený modul PN532

Firmware verzia 1.6

PN532 modul nastavený, priložte tag ...

Počas čítania nebol priložené žiadne tag.

Nájdený NFC tag!

UID dlzka: 4 bytu

UID hodnoty: 0xF2 0xE9 0xB2 0xC3

Nájdený NFC tag!

UID dlzka: 4 bytu

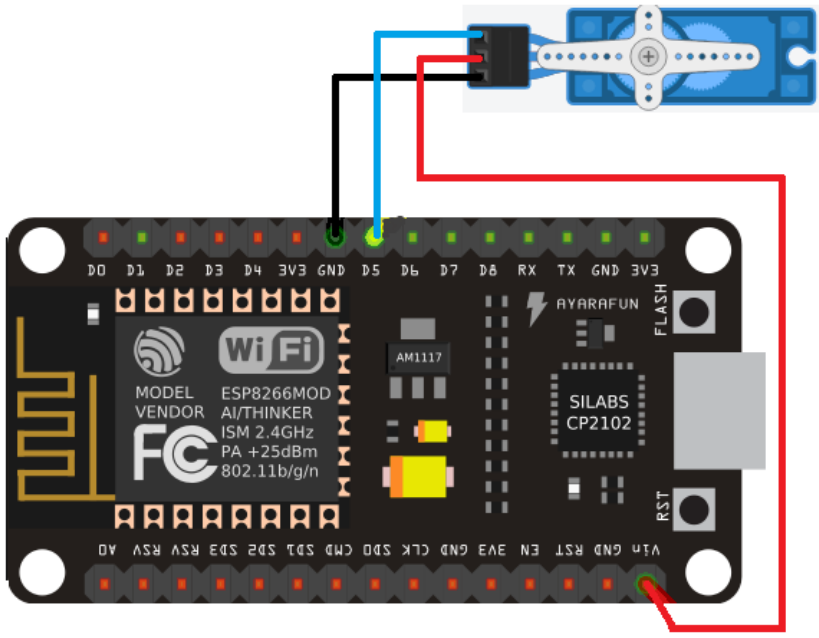
UID hodnoty: 0x56 0xAC 0x28 0x1F

Počas čítania nebol priložené žiadne tag.

Cvičenie č. 11: Arduino a bezdrôtové zariadenie wifi

V uvádzanom cvičení si ukážeme, ako pracovať s vývojovou doskou ESP8266 (NodeMCU). Ide o veľmi podobný integrovaný obvod, ako je práve Arduino UNO, avšak obsahuje navyše aj WiFi modul pre bezdrôtovú komunikáciu. V tomto cvičení budeme názorným spôsobom demonštrovať to, ako riadiť servomotor pomocou webového servera (prehliadača v rámci rovnakej WiFi siete). O tom, ako funguje servomotor, sme sa zaoberali v cvičení č.8 (riadenie servomotora), kde pre ovládanie servomotora je potrebné iba Arduino, servomotor a potenciometer. V závislosti od polohy potenciometra sa mení hodnota PWM signálu na vstupe servomotora a následne sa mení aj natočenie hriadeľa servomotora. Pre naše potreby vytvoríme jednoduchú webovú stránku (HTML), ktorá bude obsahovať grafické znázornenie natočenia serva o určitý uhol. Táto webová stránka je prístupná či už z notebooku alebo

mobilného telefónu, ktorý je pripojený k rovnakej sieti WiFi ako ESP8266. Pre riadenie servomotora prostredníctvom siete WiFi, môžeme používať ako riadiaci prvok vstup z pripojeného potenciometra alebo z webovej stránky prostredníctvom bežca. Nastavené hodnoty (uhly) tak odosiela a prijíma ESP8266 (NodeMCU). Mikrokontrolér po prijatí hodnoty vyšle príslušný PWM signál servomotorčeku, ktoré následne zmení. Celý obvod ale funguje len v prípade, že ESP8266, ktoré funguje ako server a laptop (alebo mobilný telefón), ktorý plní úlohu klienta, musia byť pripojené k rovnakej WiFi sieti.



Obrázok 38: Zapojenie ESP8266 a servo

Použité súčiastky :

- ESP8266
- Servomotor
- Prepojovacie vodiče
- PC, Notebook

Funkcia obvodu :

Hlavný program zodpovedný za všetko, tj. pripojenie ESP8266 k WiFi, získavanie informácií z webu a nakoniec ovládanie serva. Pin D5 na NodeMCU poskytne potrebný signál PWM do serva (oranžový). Jeho ďalšie piny sú piny napájacieho zdroja a sú pripojené na 5 V a GND (červený a hnedý).

Program pre bezdrôtové zariadenie WIFI

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <Servo.h>
#include "index.h";
#define LED 2
#define ServoPin 14  //D5 ako GPIO14

const char *ssid = "your-ssid";        //Konfigurácia WiFi pripojenia
const char *password = "your-password";                    //Konfigurácia WiFi pripojenia
Servo myservo;  // vytvorenie objektu servo, pre riadenie serva
ESP8266WebServer server(80);
//=================================================
void handleServo(){
 String POS = server.arg("servoPOS");
 int pos = POS.toInt();
 myservo.write(pos);  // nastav servo do určenej pozície
 delay(15);
 Serial.print("Servo Angle:");
 Serial.println(pos);
 digitalWrite(LED,!(digitalRead(LED))); //Toggle LED
 server.send(200, "text/plane","");
}

void handleRoot() {
 String s = MAIN_page;           //čítanie HTML obsahu
 server.send(200, "text/html", s);                  //odošle a zobrazí vytvorenú webstránku
```

```
}

//=============================================
//        Setup
//=============================================

void setup() {
 delay(1000);
 Serial.begin(115200);
 Serial.println();
 pinMode(LED,OUTPUT);
 myservo.attach(ServoPin);
 WiFi.begin(ssid, password);    //pripojenie na lokálny roater
 Serial.println("");

  while (WiFi.status() != WL_CONNECTED) {          // čakanie na pripojenie
  delay(500);
  Serial.print(".");
 }
 Serial.println("");                //ak je pripojenie úspešné, ukáž IP adresu na sériovom monitore
 Serial.print("Connected to ");
 Serial.println(ssid);
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());

 server.on("/",handleRoot);        //Inicializácia webservera
 server.on("/setPOS",handleServo);                //Nastaví pozíciu serva na požadovanú
```

```
 server.begin();
}
//=============================================
//    LOOP
//=============================================
void loop() {
 server.handleClient();
}
//=============================================

Program pre vytvorenie webstránky
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE html>
<html>
<head>
<title>ESP8266 Servo | Circuits4you.com</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<style>
.angle{
  width: 79px;
  height: 50px;
  position: absolute;
  vertical-align: middle;
  margin-top: 50px;
  margin-left: -114px;
  border: 0px none;
  background: rgba(0, 0, 0, 0) none repeat scroll 0% 0%;
```

```
  font: normal normal bold normal 20px Arial;
  text-align: center;
  color: rgb(34, 34, 34);
  padding: 0px;
}
.spd{
  width: 79px;
  height: 50px;
 position: absolute;
  vertical-align: middle;
  margin-top: 50px;
  margin-left: -114px;
  border: 0px none;
  background: rgba(0, 0, 0, 0) none repeat scroll 0% 0%;
  font: normal normal bold normal 50px Arial;
  text-align: center;
  color: rgb(34, 34, 34);
  padding: 0px;
}

.imageDiv{
    padding: 5%;
 }

.flx{
  display: flex;
}
```

```
</style>
<body>
<div style="width:100%;">
<div style="width:50%;  margin: 0 auto;">
 <h3>Circuits4you.com</h3>
 <h4>ESP8266 Servo Motor Control Demo</h4>
</div>
</div>

<div style="width: 50%; margin: 0 auto;" class="flx">
 <svg viewBox="0 0 500 500" width="250" height="250" id="mySVG" style="background:#fff; border: 1px solid black">
 <path fill="none" stroke="#30D8D9" stroke-width="50" d="M 376.79805300444093 404.6682053761632 A 200 200 0 1 0 121.44247806269212 403.2088886237956"></path>
 <path id="arc1" fill="none" stroke="#00A8A9" stroke-width="50" style="stroke-linecap: round;"/>
 <text x="230" y="260" fill="#777" id="angle" class="spd">0</text>
 <text x="200" y="300" fill="#777" class="angle">Servo Angle</text>
 </svg>
</div>

<script>
function sendData(pos) {
 var xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   console.log(this.responseText);
  }
```

```
  };
  xhttp.open("GET", "setPOS?servoPOS="+pos, true);
  xhttp.send();
}

function polarToCartesian(centerX, centerY, radius, angleInDegrees) {
  var angleInRadians = (angleInDegrees-90) * Math.PI / 180.0;

  return {
    x: centerX + (radius * Math.cos(angleInRadians)),
    y: centerY + (radius * Math.sin(angleInRadians))
  };
}

function describeArc(x, y, radius, startAngle, endAngle){

    var start = polarToCartesian(x, y, radius, endAngle);
    var end = polarToCartesian(x, y, radius, startAngle);

    var largeArcFlag = endAngle - startAngle <= 180 ? "0" : "1";

    var d = [
        "M", start.x, start.y,
        "A", radius, radius, 0, largeArcFlag, 0, end.x, end.y
    ].join(" ");

    return d;
}
```

```
window.onload = function() {
 document.getElementById("arc1").setAttribute("d",  describeArc(250,  250,
200, 220, 210));
};

var svg  = document.getElementById("mySVG");
  pt  = svg.createSVGPoint(),

svg.addEventListener('mousedown',function(evt){
 var loc = cursorPoint(evt);
 var degrees = Math.atan2(loc.x-250,loc.y-250)*180/Math.PI + 90;

 var offset = 220;

 degrees = (degrees + 90)
 degrees = degrees + offset;
 if(degrees > 360)
 {
  degrees = degrees - 360;
 }
 degrees = 360 - degrees;
 angle = degrees + offset;

 console.log(degrees, angle);

 if(degrees<281)
 {
```

```
    document.getElementById("arc1").setAttribute("d", describeArc(250, 250, 200, offset, angle));
    var servoAng = Math.round(((angle - 220)/280) * 100);
    document.getElementById("angle").innerHTML=servoAng;
    sendData(servoAng);
  }
});

// Get point in global SVG space
function cursorPoint(evt){
  pt.x = evt.clientX; pt.y = evt.clientY;
  return pt.matrixTransform(svg.getScreenCTM().inverse());
}

</script>

</body>
</html>
)=====";
```

Po nahratí kódu môžeme po otvorení monitora sériového portu vidieť stav WiFi modulu ESP8266. Po vykonaní všetkých inicializačných krokov (nastavenie do režimu Stanica, pripojenie ESP8266 k WiFi a spustenie webového servera. IP adresu z ESP8266 získame z monitora sériového portu a otvoríme ju prostredníctvom ľubovoľného webového prehliadača. Ak je všetko zapojené správne a ESP8266 funguje, tak pri zmene polohy bežca (alebo potenciometra) zmeníme aj polohu natočenia servomotora.

Literatúra

ARDUINO IDE 2014 - ARDUINO IDE Dostupné na internete: https://arduino.cz/arduino-ide/&gt

BOŽÍK, 2010. Arduino užívateľská príručka, Dostupné na internete: https://sk.wikipedia.org/wiki/Arduino&gt

ELEKTRORAJ, 2018 – Fritzing [online] 2018. Dostupné na internete: http://www.elektroraj.cz/2014/01/21/fritzing-vyrabejte-elektroniku/&gt

MARGOLIS, M. 2011 Arduino cookbook. O´ Reilly Media, Sebastopol, 2011. ISBN 978-0-596-80247-9.

BLEMINGS, H. – OXER, J. 2009. Practical Arduino Cool Projects for Open Source Hardware. Apress, Berkley, 2009 ISBN 978-1-4302-2477-8.

ŠÁŠIK, R. Arduino história Dostupné na internete: https://www.arduinoposlovensky.sk/hardware/historia/&gt

www.arduino.cz. 2019 Arduino. [online]. Dostupné na internete: https://arduino.cz/arduino-ide/&gt

www.arduinoposlovensky.sk. 2019. Arduino. [online]. 2019. Dostupné na internete: https://www.arduinoposlovensky.sk/hardware/historia/&gt

Názov:	Programovanie vývojovej dosky Arduino Zbierka úloh pre stredné školy
Autori:	Ing. Mgr. Peter Kuna, PhD. Mgr. Miloš Palaj, PhD.
Recenzenti:	Prof. Ing. Veronika STOFFOVÁ, CSc. doc. Ing. Štefan Koprda, PhD.
Jazyková korektúra:	Mgr. Ľubica Arpášová
Cover design:	Mgr. Miroslav Šebo, PhD.
Motív na obálke:	Mgr. Miroslav Šebo, PhD.
Technická úprava:	Ing. Mgr. Peter Kuna, PhD.
Vydanie:	prvé
Rok:	2021
Rozsah:	77 s.
Vydavateľ:	UKF v Nitre
Tlač:	online / CD
Náklad:	100 ks

Publikácia bola vydaná v rámci projektu KEGA 017UKF-4/2020

ISBN 978-80-558-1827-6