UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

Ľubomír Antoni, Peter Ševčík,
Iveta Zolotová, Peter Papcun,
Ján Vaščák, Miroslav Biňas,
Tomáš Kanócz, Jaroslav Porubän,
Matin Tomášek, Dominik Lakatoš,
Ján Skalka, Zoltán Balogh,
Peter Švec, František Galčík,
Miroslav Opiela, Peter Pisarčík,
František Jakab, Dušan Šveda

Internet vecí a jeho aplikácie

Národný výstup projektu IT Akadémia – vzdelávanie pre 21. storočie

Košice 2020, Univerzita Pavla Jozefa Šafárika v Košiciach

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

it akadémia

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

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

Internet vecí a jeho aplikácie

Spracované v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Ľubomír Antoni, Peter Ševčík, Iveta Zolotová, Peter Papcun, Ján Vaščák, Miroslav Biňas, Tomáš Kanócz, Jaroslav Porubän, Martin Tomášek, Dominik Lakatoš, Ján Skalka, Zoltán Balogh, Peter Švec, František Galčík, Miroslav Opiela, Peter Pisarčík, František Jakab, Dušan Šveda

Košice 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.
Spracované s finančnou podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie

Internet vecí a jeho aplikácie
Vysokoškolský učebný text

Autori:

RNDr. Ľubomír Antoni, PhD.	Univerzita Pavla Jozefa Šafárika v Košiciach
doc. Ing. Peter Ševčík, PhD.	Žilinská univerzita v Žiline
prof. Ing. Iveta Zolotová, CSc.	Technická Univerzita v Košiciach
Ing. Peter Papcun, PhD.	Technická Univerzita v Košiciach
doc. Dr. Ing. Ján Vaščák	Technická Univerzita v Košiciach
Ing. Miroslav Biňas, PhD.	Technická Univerzita v Košiciach
Ing. Tomáš Kanócz	Technická Univerzita v Košiciach
doc. Ing. Jaroslav Porubän, PhD.	Technická Univerzita v Košiciach
doc. Ing. Martin Tomášek, PhD.	Technická Univerzita v Košiciach
Ing. Dominik Lakatoš, PhD.	Technická Univerzita v Košiciach
RNDr. Ján Skalka, PhD.	Univerzita Konštantína Filozofa v Nitre
doc. Ing. Zoltán Balogh, PhD.	Univerzita Konštantína Filozofa v Nitre
PaedDr. Peter Švec, PhD.	Univerzita Konštantína Filozofa v Nitre
RNDr. František Galčík, PhD.	Univerzita Pavla Jozefa Šafárika v Košiciach
RNDr. Miroslav Opiela	Univerzita Pavla Jozefa Šafárika v Košiciach
RNDr. PhDr. Peter Pisarčík	Univerzita Pavla Jozefa Šafárika v Košiciach
doc. Ing. František Jakab, PhD.	Technická Univerzita v Košiciach
doc. RNDr. Dušan Šveda, CSc.	Univerzita Pavla Jozefa Šafárika v Košiciach

Recenzenti:

doc. Ing. Štefan Koprda, PhD.	Univerzita Konštantína Filozofa v Nitre
Mgr. Štefan Bocko	IBM Slovensko, spol. s r. o.

Umiestnenie: www.unibook.upjs.sk
Dostupné od: 20.10.2020

ISBN 978-80-8152-911-5 (e-publikácia)

Obsah

Úvod

Internet vecí a jeho rozvoj je jednou z najväčších výziev blízkej budúcnosti, ktorá má potenciál podporiť ekonomický rast a spoločenské zmeny vo svete. Hlavným cieľom tejto vysoko aktuálnej oblasti je vytvoriť sieť platforiem pre pripojenie zariadení a objektov k internetu, ktorá podporuje rôzne vertikálne orientované uzavreté systémy s dynamickými a adaptívnymi možnosťami konfigurácie. Príkladom je použitie internetu vecí pre inteligentné životné prostredie, inteligentné mestá a budovy, podnikové aplikácie, služby alebo obyvateľstvo. Súčasnou snahou výskumníkov, učiteľov a odborníkov z praxe je skúmať výhody a skúsenosti získané pri vývoji nových internetových ekosystémov a platforiem orientovaných na zákazníka a následne pretaviť tieto skúsenosti do vzdelávacieho procesu. Dôležitou úlohou je aj prekonanie fragmentácie vertikálne orientovaných uzavretých systémov a presun k otvoreným systémom a platformám, ktoré podporujú rôzne aplikácie.

Internet vecí pokrýva viacero oblastí (integrácia inteligentných systémov, kyber-fyzikálnych systémov, inteligentných sietí, veľkých dát), využíva viacero technológií (napríklad nano-elektronika, bezdrôtové siete, adaptívne a kognitívne systémy) a poskytuje aplikácie v rôznych doménach, akými sú napríklad zdravotníctvo, energetika, potravinárstvo, inteligentné transportné systémy, monitorovanie prostredia alebo logistika. Zaraďujeme tu aj tvorbu platforiem pre pripojenie smart objektov, ktoré integrujú budúce generácie zariadení a systémov a podporujú budúce sieťové technológie. Tieto platformy následne umožnia viacero nových aplikácií, ktoré nájdu využitie vo všetkých oblastiach života. Pri tvorbe platforiem, zariadení, architektúr a služieb je potrebné zabezpečiť podporu efektívnych bezpečnostných mechanizmov, ktoré sú charakterizované vlastnosťami ako je otvorenosť, dynamická škálovateľnosť, interoperabilita, spoľahlivosť, kognitívne schopnosti a distribuované rozhodovanie, energetická efektívnosť a používateľská prívetivosť.

V rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie sme otestovali a v praxi overili viacero aplikácií, ktoré vyplynuli z use-case analýz jednotlivých oblastí internetu vecí. Zamerali sme sa na efektívnu integráciu zariadení novej generácie a smart zariadení do adaptívnych a robustných prepojených sietí a platforiem. To zahŕňa aj výzvy z oblasti sieťového manažmentu na vyriešenie problémov týkajúcich sa konektivity obrovského počtu nových zariadení pripojených bezdrôtovo k internetu vecí. Táto aktuálna

oblasť poskytuje prínosné a zaujímavé úlohy na riešenie s množstvom pozitívnych synergických efektov.

Študenti sú pri riešení úloh motivovaní študovať poznatky do hĺbky, porozumieť fungovaniu hardvéru a softvéru na strojovej úrovni, riešiť použiteľnosť a používateľskú prívetivosť daného riešenia, jeho energetickú náročnosť, komunikačné obmedzenia a množstvo ďalších faktorov vplývajúcich na daný problém. Navyše, riešenia úloh z oblasti internetu vecí, napríklad inteligentné budovy alebo inteligentné mestá, je možné využiť aj pri popularizácii vedy a výskumu, čo má potenciál vzbudiť záujem širokej verejnosti a pritiahnuť k štúdiu informatiky kvalitných študentov.

Vo vysokoškolských učebných textoch prezentujeme základy internetu vecí, jeho vznik a technologický vývoj. Popisujeme aplikácie internetu vecí v zdravotníctve, osobnej a sociálnej oblasti, preprave a logistike, inteligentných mestách, či inteligentných budovách. Prinášame prehľad vysokoškolských predmetov v oblasti internetu vecí, ktoré sa vytvorili a inovovali na slovenských vysokých školách v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie. Prezentujeme rozšírené anotácie predmetov, aplikačné príklady a prípadové štúdie s použitím rôznych metód v oblasti internetu vecí. Vysokoškolské učebné texty sú určené pre študentov bakalárskych, magisterských, inžinierskych a doktorandských študijných programov informatických, ale aj neinformatických odborov.

1 Základy internetu vecí

Internet vecí (IoT) je novou paradigmou, ktorá sa rýchlo presadila na poli moderných bezdrôtových telekomunikačných sietí. Základnou myšlienkou tohto konceptu je všadeprítomnosť zariadení, tzv. vecí alebo objektov, senzorov, akčných členov, ktoré prostredníctvom jedinečných schém adresovania sú schopné navzájom interagovať a dosiahnuť spoločné ciele.

Inštitút elektrického a elektrotechnického inžinierstva IEEE vydal v roku 2015 publikáciu, v ktorej ponúka rôzne definície internetu vecí podľa použitých predpokladov a architektúry. Pred zavedením samotnej definície prezentujú základné charakteristiky tohto konceptu, medzi ktoré patrí (Minerva a kol., 2015):

- vzájomné prepojenie vecí,
- prepojenie vecí do internetu,
- jednoznačne identifikovateľné veci,
- všadeprítomnosť,
- schopnosť snímania a ovládania,
- schopnosť operatívnej komunikácie,
- konfigurovateľnosť heterogénnych zariadení,
- programovateľnosť.

Internet vecí v užšom zmysle *je definovaný ako sieť, ktorá spája jednoznačne identifikovateľné „veci“ s internetom. „Veci“ majú schopnosť snímania, ovládania a sú programovateľné. Pomocou využívania jednoznačnej identifikácie a snímania môžu byť informácie o „veciach“ zhromažďované a „veci“ môžu byť menené kdekoľvek, kedykoľvek a kýmkoľvek.*

Úroveň zložitosti systému založenom na internete vecí môže rásť, pričom veľké množstvo „vecí“ môže byť prepojených za účelom dodania komplexnej služby a podpory vykonávania komplexných procesov. V takom prípade spoločnosť IEEE používa nasledujúcu definíciu v širšom zmysle.

Internet vecí v širšom zmysle *je definovaný ako samo-konfigurovateľná, adaptívna a komplexná sieť, ktorá spája „veci“ s internetom prostredníctvom použitia štandardných komunikačných protokolov. Navzájom prepojené „veci“ majú fyzickú alebo virtuálnu reprezentáciu v digitálnom svete, schopnosť snímania a ovládania, programovateľnosti a sú jednoznačne identifikovateľné. Reprezentácia obsahuje informácie, ktoré obsahujú identitu „vecí“, stav, umiestnenie alebo ďalšie relevantné obchodné, sociálne alebo súkromné informácie. „Veci“ ponúkajú služby s ľudskou intervenciou alebo bez nej, s využitím jednoznačnej identifikácie, so získavaním a prenosom údajov a schopnosťou ovládania. Služby sú využívané prostredníctvom použitia inteligentných rozhraní a sú dostupné kdekoľvek, kedykoľvek a berúc do úvahy bezpečnosť čohokoľvek.*

V definícii je použitý pojem „vecí“ namiesto pojmu zariadenia. Pojem zariadenia je definovaný ako technický, fyzický komponent (hardvér) so schopnosťou prepojenia s inými systémami informačných technológií.

Súčasným rozoznávacím prvkom medzi definíciou internetu vecí v užšom zmysle a definíciou v širšom zmysle nie je iba ukazovateľ celkového počtu vecí, ale tiež pohľad na vlastníctvo a správu týchto vecí. Aj keď systémy internetu vecí môžu prináležať iba jednej osobe, napríklad pri zariadeniach alebo spotrebičoch v domácnosti, ich správa a administrácia je stále ohraničená celkovou logikou systému.

1.1 Vznik internetu vecí a technologický vývoj

Korene technológie rádiofrekvenčnej identifikácie (RFID) siahajú do obdobia druhej svetovej vojny. Nemci, Japonci, Američania aj Briti používali radary objavené v roku 1935 škótskym fyzikom Robertom Alexandrom Watson-Wattom. Slúžili na varovanie pred nepriateľskými lietadlami ešte v čase, keď boli vzdialené míle. Ale nebolo možné identifikovať, ktoré lietadlá patria nepriateľom a ktoré sa vracajú do vlastnej krajiny z bojovej misie.

Nemci zistili, že ak piloti rozkolíšu lietadlá predtým, ako sa vracajú do základne, rádiový signál v radarových systémoch sa zmení. Takýto postup upozornil obsluhu radarov na vlastné lietadlá. V skutočnosti to zodpovedalo prvého pasívnemu RFID systému.

Pokroky v radarových a rádiofrekvenčných komunikačných systémoch pokračovali aj v 50-tych a 60-tych rokoch 20. storočia. Výskumníci zo Spojených štátov amerických, Európy, Japonska skúmali, ako môže byť rádiofrekvenčná energia použitá na identifikáciu vzdialených objektov. Spoločnosti začali komercionalizovať výstražné systémy, ktoré používali rádiové vlny na určenie toho, či bola položka zaplatená alebo nie. Vznikli prvé RFID štítky, ktoré sa používali na sledovanie nukleárneho materiálu, alebo v poľnohospodárstve.

V roku 1990, inžinieri v IBM vyvinuli a patentovali ultra vysoko-frekvenčný (UHF) RFID systém, ktorý však nikdy nekomercionalizovali. Svoj patent predali poskytovateľovi systémov s čiarovými kódmi. Technológia bola v tom čase pomerne drahá.

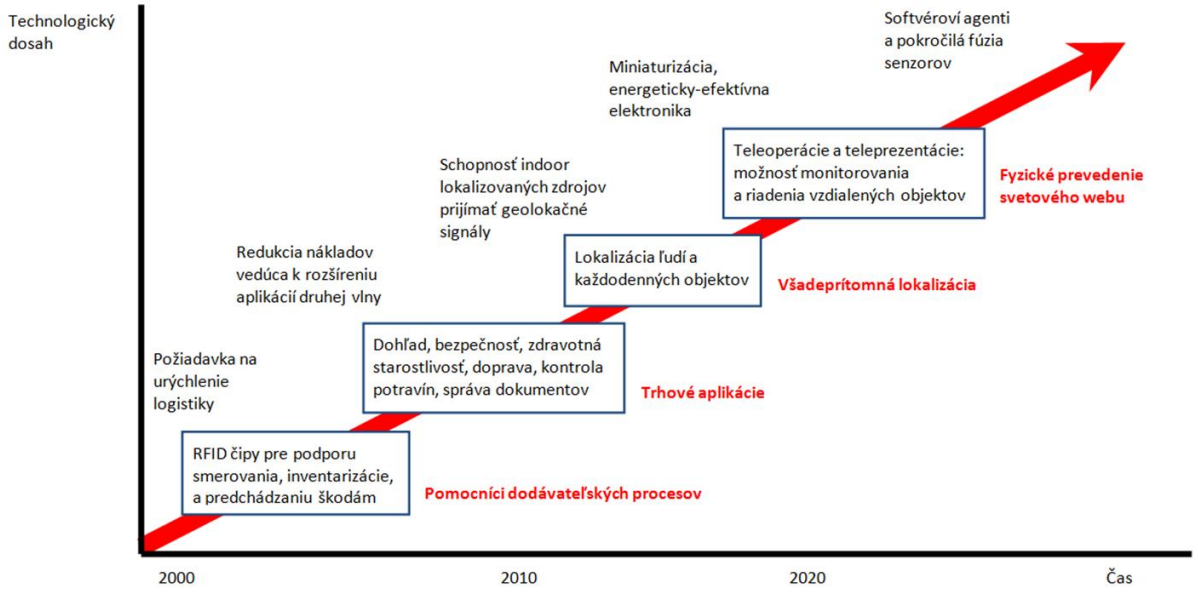
Ultra vysoko-frekvenčný (UHF) RFID systém sa uchytil až v roku 1999, keď vzniká globálna výskumná sieť akademických laboratórií Auto-ID Labs na Massachusettskom technologickom inštitúte MIT v meste Cambridge amerického štátu Massachusetts. Profesor Brock a profesor Sarma hľadania možnosti použitia nízkonákladových RFID štítkov na sledovanie výrobkov v zásobovacom reťazci.

Pri zakladaní tejto výskumnej siete stál aj britský technológ Kevin Ashton, ktorý pojem internetu vecí prvýkrát použil v jeho rovnomennej prezentácii počas jeho pracovného pôsobenia v spoločnosti Procter & Gamble. Toto slovné spojenie použil na označenie systému, v ktorom je internet prepojený s fyzickým svetom pomocou všadeprítomných senzorov. Spoločnosť Auto-ID zmenili RFID na sieťovú technológiu prepojením objektov s internetom prostredníctvom štítkov.

Ashton (2009) v časopise RFID Journal uvádza, že informačné technológie sú stále závislé na údajoch a informáciách, ktoré pochádzajú od ľudí. Dôraz do budúcna v súvislosti so zdieľaním údajov kladie na fyzické zariadenia, tzv. veci, ktoré majú rovnaký potenciál zmeniť svet, tak ako to dokázal internet. Dôležitým míľnikom vo vývoji Internetu vecí je rok 2008, kedy podľa odhadov prekročil počet zariadení pripojených k internetu veľkosť svetovej populácie.

V súčasnosti sa predpokladá, že k internetu je pripojených asi 50 miliárd zariadení. Na jedného človeka na zemi tak prislúcha približne 7 zariadení.

Na Obrázku 1 sú znázornené trendy vo vývoji internetu vecí a ich technologický dosah od roku 2000, teda od prvotných pomocníkov pre dodávateľské procesy, cez trhové aplikácie slúžiace pre účely zdravotnej starostlivosti, dopravy, kontroly potravín, bezpečnosti, následne všadeprítomnú lokalizáciu až pokročilú fúziu senzorov.

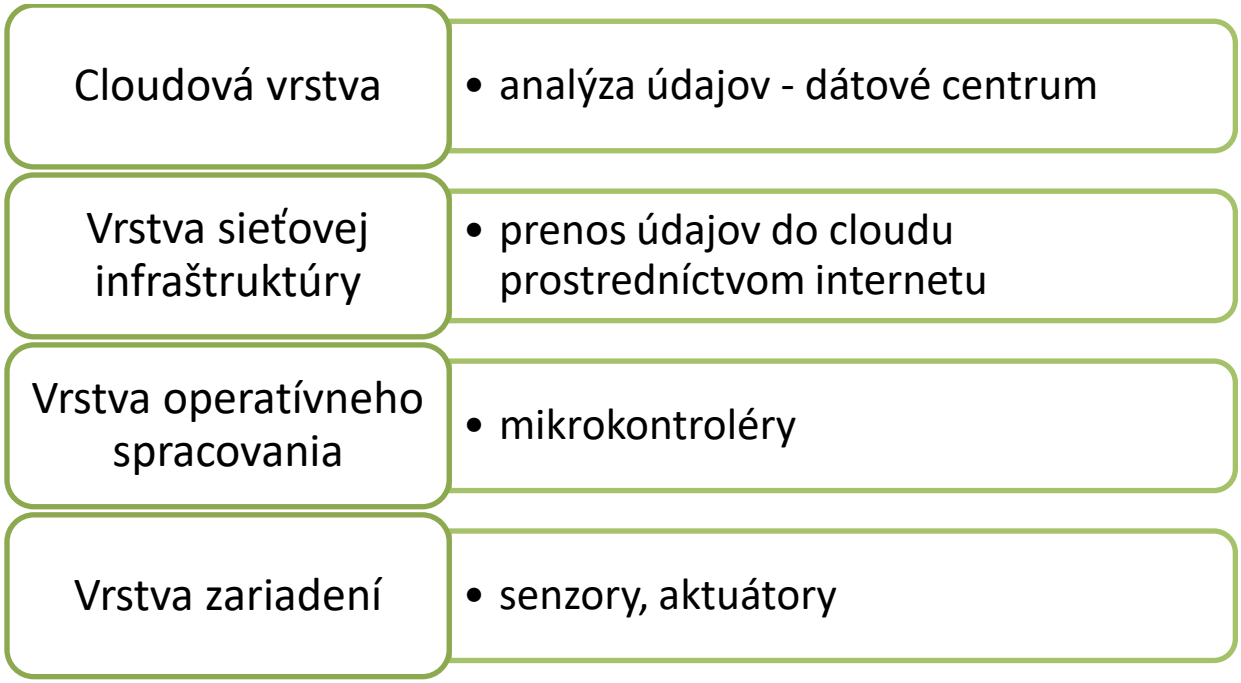


Zdroj: SRI Consulting Business Intelligence

Obrázok 1. Trendy vo vývoji internetu vecí a ich technologický dosah

1.2 Koncepcia Internetu vecí

V tejto časti prezentujeme základné pojmy, vrstvový model a prehľad základných protokolov a formátov dát pre oblasť internetu vecí.



Obrázok 2. Vrstvový model internetu vecí

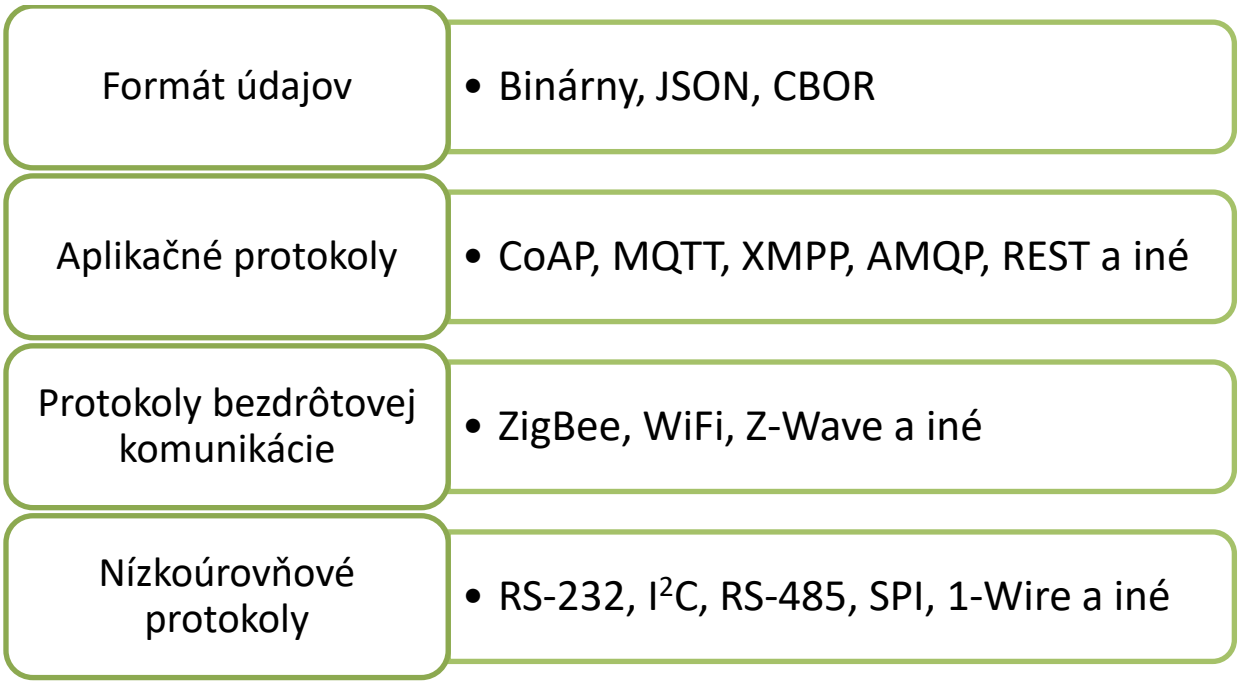
Senzory (alebo snímače) sú zariadenia, ktoré sa používajú na meranie fyzickej veličiny v prostredí. Takto získané informácie poskytujú iným zariadeniam. Senzory alebo snímače delíme na analógové alebo digitálne. Analógové senzory transformujú fyzikálnu veličinu na inú elektricky merateľnú veličinu (napríklad odpor alebo napätie). Napríklad termistor transformuje teplotu na odpor, pričom s vyššou teplotou klesá odpor. Svetelný senzor, nazývaný fotorezistor, transformuje intenzitu osvetlenia na odpor približne exponenciálne.

Aktuátory (ovládače, aktivátory, alebo akčné členy) sú zariadenia, ktoré menia elektrický signál na prislúchajúcu fyzikálnu veličinu, napríklad na pohyb, tlak a iné. Rozlišujeme aktuátory hydraulické, elektronické, elektromechanické a iné typy. Príkladom môže byť aktuátor žalúzií alebo garážovej brány.

Mikrokontroléry sú jednočipové počítače, ktoré sú programovateľné v jednom integrovanom obvode. Zodpovedajú za zhromažďovanie údajov zo senzorov

a za zabezpečenie sieťového spojenia. Údaje môže odoslať do vzdialených a výkonných počítačov na analýzu.

Vrstvové sieťové modely boli vyvinuté za účelom popisu fungovania siete. Existuje niekoľko modelov, ktoré pomáhajú uľahčiť návrh a tvorbu systémov internetu vecí. Vrstvový model znázornený na Obrázku 2 je založený na predpoklade, že senzory z vrstvy zariadení poskytujú údaje do mikrokontrolérov na úrovni operatívneho spracovania. Mikrokontrolér na základe vloženého programu vyberie údaje, ktoré sú určené na analýzu a tieto údaje posiela prostredníctvom vrstvy sieťovej infraštruktúry do cloudovej vrstvy. Príklady protokolov a používaných údajových formátov pri riešeniach internetu vecí sú znázornené na Obrázku 3.



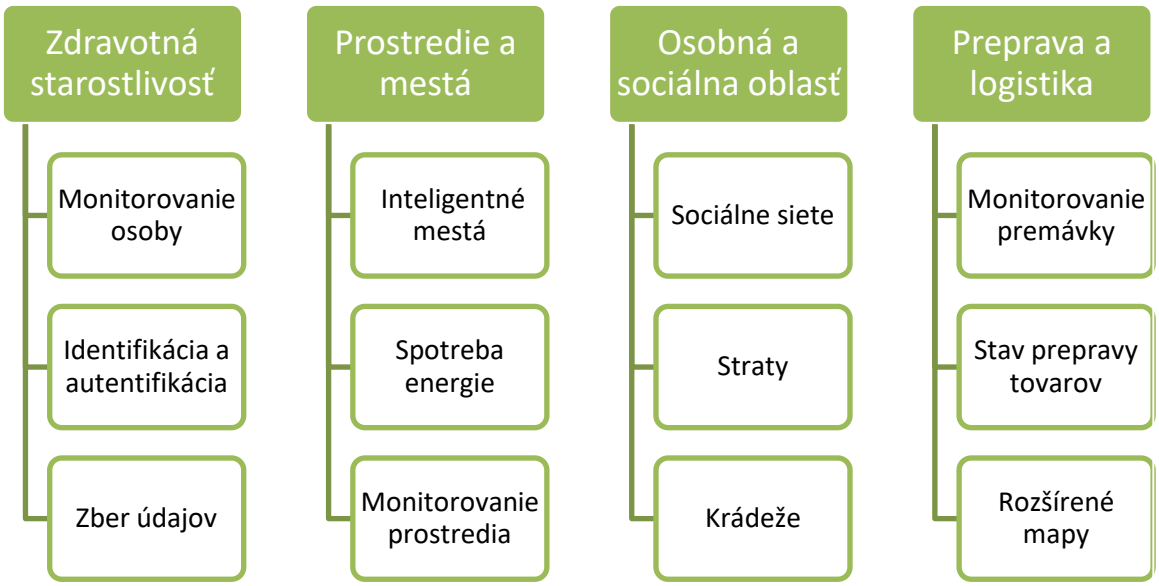
Obrázok 3. Príklady protokolov a používaného formátu dát pri riešeniach internetu vecí

Komponenty internetu vecí môžeme rozdeliť na hardvér, middleware (protokoly) a softvér. Medzi hardvér zaraďujeme všetky fyzické zariadenia, ktoré generujú údaje, prvky sieťovej a výpočtovej infraštruktúry, ale aj dátové úložiská, ktoré môžu byť súkromné, verejné, hybridné, či komunitné. Obľúbený jednodoskový počítač v oblasti Internetu vecí, ktorý bol vyvinutý na edukačné účely je Raspberry Pi.

2 Aplikácie internetu vecí

Potenciál, ktorý ponúka internet vecí, umožňuje vývoj veľkého množstva aplikácií. Týka sa to viacerých domén, v ktorých tieto aplikácie môžu zlepšiť kvalitu nášho života, napríklad v domácnosti, pri cestovaní, v súvislosti so zdravím, prácou, pohybovou aktivitou. V mnohých prostrediach je možné nasadiť veľmi širokú škálu aplikácií, ktoré môžeme zoskupiť do nasledujúcich domén (Atzori a kol., 2010; Zanella a Vangelista, 2014; Stankovič, 2014; Sissini a kol. 2018; Ashgari, 2019):

- zdravotná starostlivosť,
- inteligentné prostredie a mestá (domov, kancelária, spoločnosť),
- osobná a sociálna oblasť,
- preprava a logistika.



Obrázok 4. Príklad klasifikácie aplikácií v oblasti internetu vecí

V nasledujúcej časti si priblížime každú z týchto domén osobitne s identifikáciou rôznych príkladov aplikácií. Osobitnú pozornosť venujeme problematike inteligentných miest.

2.1 Zdravotná starostlivosť

Medzi aplikácie v oblasti zdravotnej starostlivosti patria napríklad monitorovanie pacientov a iných objektov, identifikácia a autentifikácia ľudí a objektov, či automatické zbieranie a snímanie údajov. Vzdialené monitorovanie zdravia pacientov v reálnom čase je vhodné napríklad na monitorovanie diabetikov či astmatikov, pacientov s dlhodobými ťažkosťami, seniorov. Viacerí pacienti už v súčasnosti využívajú mobilné aplikácie na správu vlastných zdravotných potrieb (Atzori a kol., 2010; Dimitrov, 2016).

Typickú nemocnicu založenú na internete vecí si môžeme predstaviť tak, že pacient s ochorením diabetes mellitus má ID kartu, ktorá po zoskenovaní ukladá na bezpečnom cloude údaje zo senzorov o glukóze, krvnom tlaku, teplote, uchováva všetky elektronické zdravotné záznamy, laboratórne výsledky, históriu zdravotných výkonov a históriu predpisov liekov. Lekári a zdravotné sestry môžu pristúpiť k týmto údajom na svojom tablete alebo počítači. Papierové záznamy, často písané nečitateľným rukopisom, sa ukladajú do kartoték, mimo dosahu výskumných pracovníkov alebo iných poskytovateľov zdravotnej starostlivosti. Udržaním všetkých dôležitých informácií na jednom mieste a ich ľahkej zdieľateľnosti je možné eliminovať neefektívnu správu zdravotníckych záznamov a zachrániť životy.

Identifikácia a autentifikácia môže byť použitá za účelom zníženia prípadov škodlivých pre pacientov, napríklad podanie nesprávneho lieku, podanie nesprávnej dávky, nesprávne určenie času, nesprávne určenie postupu, tiež pri komplexnej a aktuálnej správe elektronického lekárskeho záznamu v internej a ambulantnej podobe. Identifikácia novorodencov v nemocniciach môže slúžiť na zabránenie zámeny. Vo vzťahu k zamestnancom sa identifikácia a autentifikácia najčastejšie používa na zabezpečenie prístupu a na zlepšenie morálky zamestnancov zameraním na bezpečnosť pacientov. Identifikácia a autentifikácia objektov sa používa na splnenie požiadaviek bezpečnostných postupov, na zabránenie krádežiam alebo stratám dôležitých nástrojov alebo produktov.

Jedna z hlavných výziev pri implementácii internetu vecí súvisí aj so vzájomnou komunikáciou zariadení. Hoci veľa zariadení má v súčasnosti senzory na zhromažďovanie údajov, so serverom často komunikujú v ich vlastnom jazyku. Výrobcovia majú svoje vlastné protokoly, čo znamená, že senzory od rôznych výrobcov nemusia dokázať navzájom komunikovať. Toto softvérové prostredie spojené s obavami o ochranu súkromia a byrokratickou tendenciou zhromažďovať všetky informácie môže byť jednou z prekážok používania internetu vecí.

2.2 Inteligentné prostredie

V oblasti inteligentného prostredia, pod ktorým si môžeme predstaviť domácnosť, pracovisko, priemyselnú spoločnosť, ale aj prostredie voľnočasových aktivít, môžu riešenia internetu vecí skvalitniť život vo viacerých aspektoch:

- vykurovanie miestností je možné prispôsobiť našim vlastným preferenciám a počasiu,
- osvetlenie miestnosti je možné meniť podľa denného času,
- domácim incidentom je možné sa vyhnúť s príslušnými monitorovacími a výstražnými systémami,
- energiu je možné ušetriť automatickým vypínaním elektrických zariadení, ak to nie je potrebné.

Dodávatelia energií môžu využívať dynamicky sa meniace ceny energie, ktoré ovplyvňujú celkovú spotrebu energie tak, aby sa obmedzili vyťaženie siete v dobe najväčšieho zaťaženia. Automatizácia môže optimalizovať náklady na spotrebu energie tým, že sledujeme, v ktorých časových úsekoch dňa sú ceny uvedené externou webovou službou lacné a podľa toho nastavíme špecifické požiadavky každého spotrebiča v domácnosti (nabíjačky batérií, chladničky, rúry na pečenie a podobne).

Inteligentné prostredie môže tiež pomôcť pri zlepšovaní automatizácie v priemyselných spoločnostiach. Bezdrôtové senzory namontované na stroji môžu monitorovať stav výroby a v prípade prekročenia prahovej hodnoty vyvolať udalosť na okamžité zastavenie procesu (tzv. kontrola kvality). Manažér sleduje stav zariadenia a dokáže modelovať vedľajšie účinky oneskorenia linky z dôvodu nesprávnej činnosti výrobného zariadenia.

V oblasti voľnočasových aktivít, napríklad v múzeu, môžu expozície evokovať rôzne historické obdobia s rozdielnymi klimatickými podmienkami (doba ľadová a. Staroveký Egypt). Budova sa týmto podmienkam môže prispôsobiť a zobrať do úvahy aj vonkajšie podmienky. V telocvični môže osobný tréner nahrať na kartu profil cvičenia pre každého cvičiaceho, následne cvičebný stroj automatický rozozná nastavenie špecifické pre konkrétnu osobu. Zdravotné parametre môžu byť monitorované počas celého tréningu a zaznamenané hodnoty sa kontrolujú, aby sa predišlo pretrénovaniu alebo príliš zvoľnenému spôsobu cvičenia.

2.3 Osobná a sociálna oblasť

Do tejto oblasti patria aplikácie, ktoré umožňujú užívateľom vzájomné prepojenie s ďalšími osobami za účelom budovania a zachovania sociálnych vzťahov. Môžu pomôcť automaticky aktualizovať informácie o našich sociálnych aktivitách, našej polohe, historických záznamoch.

Aplikácia vyhľadávania osobných vecí je nástrojom, ktorý pomáha pri hľadaní predmetov, o ktorých si nepamätáme ich posledné umiestnenie. Používateľom umožňuje zobraziť posledné zaznamenané miesto pre ich označené objekty alebo vyhľadať umiestnenie konkrétneho objektu. Proaktívnejšie rozšírenie tejto aplikácie umožňuje notifikovať používateľov na posledné umiestnenie zaznamenaného objektu, ktoré zodpovedá podmienkam a požiadavkám užívateľa. Podobná aplikácia môže umožniť používateľovi zistiť, či sú niektoré objekty presunuté z vymedzeného priestoru (napríklad dom majiteľa alebo kancelária), čo by mohlo znamenať, že predmet je odcudzený. V takom prípade musí byť udalosť okamžite oznámená vlastníkovi, resp. bezpečnostným pracovníkom.

2.4 Preprava a logistika

Vyspelé autá, vlaky, autobusy, či bicykle na cestách alebo koľajniciach sú čoraz častejšie vybavené prístrojmi so senzormi, akčnými členmi a procesorovým výkonom. Cesty a prepravovaný tovar sú tiež vybavené značkami a senzormi, ktoré odosielajú dôležité informácie o premávke, kontrolných miestach, stave prepravovaného tovaru, dopravných prostriedkoch na lepšie smerovanie dopravy, príslušných dopravných informáciách pre turistov. Informácie o dopravných službách (stanice, počet pasažierov, cena, dostupnosť sedadiel, typ služieb a iné) môžu byť poskytované užívateľom prostredníctvom informačných panelov alebo aj priamo v mobilných telefónoch.

Tovary podliehajúce pokazeniu sa, napríklad ovocie, čerstvé rezané výrobky, mäso a mliečne výrobky sú dôležitou súčasťou našej výživy. Z miest ich produkcie na miesto spotreby prekonávajú tisíce kilometrov a počas prepravy je potrebné monitorovať stav ich konzervácie (teplota, vlhkosť a iné), aby sa predišlo zníženiu úrovne kvality a umožnilo správne rozhodnúť o ich distribúcii. Všadeprítomné výpočty a technológie snímačov ponúkajú veľký potenciál pre zlepšenie efektívnosti dodávateľského reťazca potravín.

2.5 Inteligentné mestá

V tejto časti uvádzame prehľad niektorých služieb internetu vecí v mestách, ktoré môžu zvýšiť kvalitu služieb ponúkaných občanom a zároveň priniesť ekonomické výhody pre správu mesta z hľadiska redukcie prevádzkových nákladov (Zanella a Vangelista, 2014; Albino a kol., 2015; Teng a kol., 2019).

2.5.1 Údržba historických budov

Správna údržba historických budov v meste si vyžaduje nepretržité monitorovanie skutočných podmienok každej budovy a identifikácie oblastí, ktoré sú najviac vystavené vonkajším vplyvom. Internet vecí v meste môže poskytovať distribuovanú databázu budov s meraniami ich štrukturálnej integrity pomocou vhodných senzorov umiestnených v budovách, ako napríklad senzory vibrácií a senzory deformácie na monitorovanie statiky budovy, senzorov atmosférických látok v okolitých oblastiach na monitorovanie úrovne znečistenia, senzorov teploty a vlhkosti na úplnú charakteristiku podmienok životného prostredia.

Takáto databáza môže umožniť zníženie potreby nákladných pravidelných statických skúšok, cielenú a proaktívnu údržbu a reštaurátorské práce. Je možné kombinovať vibrácie a seizmické údaje s cieľom lepšie porozumieť dopadu ľahkých zemetrasení na mestské budovy. Táto databáza môže byť verejne prístupná za účelom sprístupnenia informácií občanom pri starostlivosti o historické dedičstvo mesta.

Praktická realizácia tejto služby vyžaduje inštaláciu senzorov v budovách a okolitých oblastiach, ich vzájomné prepojenie s monitorovacím systémom, ktorého vytvorenie si môže vyžadovať počiatočné investície a potrebnú infraštruktúru.

2.5.2 Nakladanie s odpadmi

Nakladanie s odpadom je primárnym problémom v mnohých moderných mestách z dôvodu nákladov na službu a problémom so skladovaním odpadov na skládkach. Použitie riešení internetu vecí v tejto oblasti môže viesť k značných úsporám, ekonomickým a ekologickým výhodám.

Napríklad použitie inteligentných kontajnerov na odpad, ktoré umožňujú detekciu úrovne naplnenia a umožňujú optimalizáciu trasy nákladných vozidiel pri zbere odpadu, môže znížiť náklady na zber odpadu a vylepšiť zber z hľadiska kvality recyklácie. Na realizáciu služby riadenia inteligentného odpadu, zariadenia internetu vecí sú pripojené na kontajnery s odpadom, pričom optimalizačný softvér spracuje údaje a určí optimálnu správu vozového parku pre zber odpadov.

2.5.3 Kvalita vzduchu a monitorovanie hluku

Internet vecí v meste môže poskytovať sledovanie kvality vzduchu v preplnených oblastiach, parkoch alebo na turistických chodníkoch. Tieto informácie môžu byť prepojené na zdravotnícke aplikácie na zariadeniach chodcov, bežcov a podobne. Týmto spôsobom môžu nájsť obyvatelia najzdravšiu cestu pre exteriérové aktivity a môže byť nepretržite spojený s ich preferovanou aplikáciou osobného tréningu. Realizácia takejto služby si vyžaduje senzory kvality ovzdušia a senzory znečistenia, ktoré budú rozmiestnené po celom meste, pričom údaje z nich budú sprístupnené verejnosti.

Hluk je možné považovať za formu znečisťovania ovzdušia rovnako ako napríklad znečisťovanie oxidom uhličitý. Ide o akustický typ znečisťovania, pričom viaceré mestá už vydali konkrétne zákony na zníženie množstvo hluku v centre mesta v konkrétnych hodinách. Internet vecí v mestách môže ponúknuť službu monitorovania hluku v ktorúkoľvek hodinu na miestach, ktoré poskytujú služby. Okrem budovania časopriestorovej mapy znečistenia hlukom v danej oblasti možno takúto službu využiť aj na vynútenie verejnej bezpečnosti pomocou algoritmov detekcie zvuku, ktoré môžu rozoznať napríklad hluk skla alebo bitiek. Táto služba tak môže zlepšiť nočný pokoj v meste a dôveru vlastníkov verejných zariadení, aj keď inštalácia detektorov zvuku alebo mikrofónov je konfrontovaná s obavami o ochranu súkromia.

2.5.4 Dopravné zápchy

V rovnakej línii merania kvality vzduchu a hluku môže poskytovať internet vecí aj služby v oblasti monitorovania preplnenia ciest v meste. Napriek tomu, že kamerové systémy na sledovanie premávky už sú dostupné a nasadené v mnohých mestách, existujú riešenia s nižšou spotrebou energie a bohatším zdrojom informácií.

Monitorovanie dopravy je možné realizovať pomocou senzorov, GPS, spolu s kombináciou monitorovania kvality zvuku a úrovne hluku pozdĺž danej cesty. Tieto informácie majú pre veľký význam pre mestské úrady a obyvateľov mesta. Mestské úrady môžu monitorovať dopravný poriadok a vyslať hliadky na miesta, v ktorých je to najviac potrebné. Obyvatelia si môžu naplánovať cestu na pracovisko v predstihu, prípadne si môžu vhodne naplánovať cestu na nákupy do centra mesta.

2.5.5 Spotreba energie

Spolu so službou monitorovania kvality ovzdušia, môže internet vecí v mestách poskytovať službu monitorovania spotreby energie celého mesta a tým umožniť úradom a občanom získať jasný a podrobný prehľad o celkovom množstve energie požadovanej rôznymi službami, napríklad verejné osvetlenie, doprava, semafory, kamery, kúrenie a chladenie verejných budov a iné).

Takéto riešenie umožňuje identifikovať hlavné zdroje spotreby energie v meste a stanoviť priority s cieľom optimalizovať správanie týchto zdrojov. Všeobecne prijímané nariadenia rôznych inštitúcií nabádajú k zlepšeniu energetickej účinnosti.

Za účelom využívania tejto služby je potrebné integrovať monitorovacie zariadenia odberu energie do elektrickej siete v meste. Túto službu je možné rozšíriť aj o funkcionalitu na monitorovanie lokálnych štruktúr na výrobu energie, napríklad fotovoltaických panelov.

2.5.6 Inteligentné parkovanie

Služba inteligentného parkovania je založená na cestných senzoroch a inteligentných displejoch, ktoré usmerňujú motoristov pozdĺž najlepších ciest pre parkovanie v meste. Výhody plynúce z tejto služby sú rozmanité:

- rýchlejší čas na vyhľadanie parkovacieho miesta,
- menej emisií oxidu uhoľnatého z automobilu,
- menšie dopravné zápchy,
- šťastnejší obyvatelia.

Inteligentné parkovanie môže byť priamo integrované do mestskej infraštruktúry internetu vecí, pretože veľa spoločností za týmto účelom poskytuje trhové produkty pre túto aplikáciu internetu vecí. Využitím komunikačných technológií je možné realizovať overovací systém parkovacích preukazov vo vyhradených prevádzkových intervaloch pre obyvateľov alebo zdravotne postihnutých občanov. Toto riešenie ponúka občanom kvalitnejšie služby a efektívne nástroje v prípade zistenia porušenia.

2.5.7 Inteligentné pouličné osvetlenie

Optimalizácia účinnosti pouličného osvetlenia je dôležitou vlastnosťou z hľadiska šetrenia zdrojov energie. Touto službou môžeme optimalizovať intenzitu pouličného osvetlenia podľa času dňa, poveternostných podmienok a prítomnosti ľudí. Senzory monitorujú pohyb chodcov alebo vozidiel v okolí, pričom odosielajú informácie do riadiaceho centra. Následne je osvetlená iba oblasť s určitým počtom bodov, pričom tento počet môže byť identifikovaný aj na základe rýchlosti pohybu chodca alebo vozidla.

Na správne fungovanie služby je potrebné zahrnúť pouličné osvetlenie do konceptu internetu vecí v meste. Nad kontrolérmi pouličného osvetlenia je ľahko realizovateľný aj systém detekcie porúch.

2.5.8 Inteligentné budovy

Dôležitou aplikáciou technológií internetu vecí je monitorovanie spotreby energie a zdravého životného prostredia vo verejných budovách, teda v školách, úradoch, pracoviskách alebo v múzeách pomocou rôznych typov senzorov a akčných členov, ktoré riadia svetlo, teplota alebo vlhkosť.

Ovládaním týchto parametrov je možné skutočne zvýšiť úroveň komfortu osôb, ktoré sú prítomné v týchto budovách. Tieto riešenia môžu mať pozitívnu návratnosť z hľadiska produktivity pri súčasnom znižovaní nákladov na kúrenie alebo chladenie.

Hudec (2016) prezentuje prehľad automatizovaných systémov v prostredí interiéru budov za účelom pomoci seniorom pri ochoreniach pohybového aparátu, pri zrakových postihnutiach, ale aj zvýšení bezpečnosti pre starších ľudí. Tento autor článku Inteligentné budovy s asistentom pre nevidiacich je nevidiacim človekom, ktorý v univerzitnom prostredí vytvoril systém RUDO vybavený rôznymi technológiami na podporu obsluhy zónovej regulácie, vykurovacieho systému a používania počítačov v oblasti odbornej informatiky pre nevidiaceho človeka.

Minoli a kol. (2017) a Plageras a kol. (2018) uvádzajú príklady a techniky na správu energií pomocou internetu vecí napríklad v podobe inteligentných výťahov, interiérového osvetlenia, monitorovanie životného prostredia vrátane detekcie požiarov. Uvádzajú príklady systémov, v ktorých môžu byť použité riešenia internetu vecí na efektívnu správu spotreby energie:

- serverovne s klimatizáciami,
- priestory kancelárií,
- kotolne, chladenie, klimatizácia, kompresory,
- obchodné priestory a iné.

Ako riešenie implementujú systém, ktorý obsahuje senzory pre zaznamenávanie teploty, pohybu, svetla, vlhkosti, umožňuje komunikáciu medzi senzormi a užívateľmi.

3 Internet vecí v rámci národného projektu IT akadémia – vzdelávanie pre 21. storočie

Strategickým cieľom národného projektu IT Akadémia – vzdelávanie pre 21. storočie bolo vytvorenie modelu vzdelávania a prípravy mladých ľudí pre aktuálne a perspektívne potreby vedomostnej spoločnosti a trhu práce so zameraním na informatiku a IKT. Aktivity projektu boli rozdelené na dve časti, pričom jednu z aktivít predstavovalo vzdelávanie na vysokých školách pre informačnú spoločnosť. Táto aktivita bola zameraná na inováciu prípravy študentov vysokých škôl pre zamestnanie v IT sektore, štandardy digitálnej gramotnosti, osobnostného rozvoja a komunikačných kompetencií na vysokých školách, a tiež vytvorenie partnerstiev a sietí vysokých škôl a IT firiem.

Cieľovou skupinou v oblasti vzdelávania na vysokých školách pre informačnú spoločnosť bolo 20 vysokoškolských učiteľov a 3000 študentov vysokých škôl.

Jedným z hlavných výstupov projektu v oblasti inovácie prípravy študentov vysokých škôl pre zamestnanie v IT sektore, bolo **vybudovanie a aktívna činnosť kompetenčného centra pre vzdelávanie na vysokých školách** s účasťou všetkých partnerov projektu v štyroch oblastiach odborného zamerania – **Dátová veda (Data Science), Internet vecí (Internet of Things), Počítačové siete (Computer Networks) a Podnikovo-informačné systémy (Business Information Systems)**, a to s ohľadom na potreby praxe v IT sektore pri rešpektovaní kvalifikačných štandardov podľa Národnej sústavy kvalifikácií.

Kompetenčné centrum pre vzdelávanie na vysokých školách vzniklo v roku 2016. Počas celého obdobia realizácie projektu (2016 – 2020) slúžilo ako zdroj inovácií pre vzdelávací proces v IT oblasti na vysokých školách v nasledujúcej forme:

1. Inovácia vysokoškolských predmetov a študijných programov pre informatikov a neinformatikov na vysokých školách.
2. Lektorovanie predmetov pre informatikov a pilotná výučba predmetov pre neinformatikov na vysokých školách.
3. Koordinácia obsahu a metód vzdelávania so súkromným IT sektorom – konzultácie s expertmi v IT firmách, stáže učiteľov v IT firmách.
4. Semináre, videokonferenčné semináre, účasť na konferenciách, bakalárske a diplomové práce.
5. Tvorba národného materiálu v troch oblastiach zamerania kompetenčného centra.

3.1 Inovácia vysokoškolských predmetov a študijných programov pre informatikov

Počas celej doby realizácie projektu prebiehala tvorba, revízia a aktualizácia obsahu vysokoškolských predmetov pre informatikov na piatich partnerských univerzitách:

a) Univerzita Pavla Jozefa Šafárika v Košiciach – UPJŠ,
b) Technická univerzita v Košiciach – TUKE,
c) Univerzita Konštantína Filozofa v Nitre – UKF,
d) Žilinská univerzita – UNIZA,
e) Univerzita Mateja Bela v Banskej Bystrici – UMB.

Účasť niektorých expertov z IT firiem na tvorbe predmetov bola zabezpečená s podporou Centra vedecko-technických informácií v Bratislave – CVTI.

Tvorba, revízia a aktualizácia obsahu vysokoškolských predmetov pre informatikov prebiehala vo všetkých plánovaných oblastiach odborného zamerania:

a) Dátová veda (Data Science),
b) Internet vecí (Internet of Things),
c) Počítačové siete (Computer Networks),
d) Podnikovo-informačné systémy (Business Information Systems).

Kvantitatívne rozdelenie predmetov podľa oblastí, v rámci ktorých sa tvorili študijné materiály, sú uvedené v Tabuľke 1.

Tabuľka 1. Prehľad tvorby študijných materiálov počas realizácie projektu

Názov oblasti informatického vzdelávania	**Počet nových a inovovaných predmetov s tvorbou študijných materiálov**
Oblasť Data Science	19 predmetov (8 UPJŠ + 3 TUKE + 5 UKF + 2 UMB + 1 UNIZA)
Oblasť Podnikovo-informačných systémov	14 predmetov (10 CVTI + 4 UPJŠ)
Oblasť Internetu vecí	13 predmetov (4 TUKE + 4 UKF + 3 UNIZA + 2 UPJŠ)
Oblasť Sieťová akadémia	14 predmetov (5 TUKE + 4 UPJŠ + 3 UNIZA + 1 UMB + 1 UKF)
Spolu	60 predmetov

V rámci uvedených predmetov sa tvorili aj študijné materiály, ktoré sú zamerané na prípravu absolventov neinformatických odborov pre ich pracovné uplatnenie v IT sektore (3 predmety v oblasti Dátovej vedy, 2 predmety v oblasti Sieťová akadémia, 1 predmet v oblasti Internet vecí, 4 predmety v oblasti Podnikovo-informačné systémy).

Prínosom národného projektu IT Akadémia – vzdelávanie pre 21. storočie vzhľadom na stanovený výstup je zaradenie nových a inovovaných predmetov do študijných programov vysokých škôl v aktuálnych a moderných trendoch informatického vzdelávania. **Príkladmi predmetov, ktoré by nevznikli bez podpory národného projektu IT Akadémia – vzdelávanie pre 21. storočie**, sú napríklad:

- NoSQL databázy, Prípadové štúdie dolovania údajov, Strojové učenie, Základy internetu vecí, Výpočty v prostredí SAP HANA (Univerzita Pavla Jozefa Šafárika v Košiciach),
- Vývoj aplikácií pre Internet vecí, Inteligentné kyberfyzikálne systémy (Technická univerzita v Košiciach),
- Programovanie aplikácií so senzormi, Edukačné dolovanie údajov (Univerzita Konštantína Filozofa v Nitre),
- Internet vecí (Žilinská univerzita v Žiline).

V rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie boli na partnerských univerzitách v rámci študijných programov implementované certifikované bloky „IT pre prax“, ktorých cieľom je zlepšiť vedomosti a praktické skúsenosti študentov neinformatických smerov v oblastiach IT zamerania. Certifikovaný interdisciplinárny kurz „IT pre prax“ na TUKE a UPJŠ pozostáva celkovo zo 4 predpísaných predmetov, konkrétne z dvoch povinných predmetov (Základy technológie SAP a Úvod do internetu vecí), z jedného voliteľného predmetu z bloku Data Science a z jedného voliteľného predmetu z bloku Počítačové systémy.

Jedným z dopadov projektu a tiež prínosom z hľadiska udržateľnosti je akreditácia nového bakalárskeho a magisterského študijného programu Analýza dát a umelá inteligencia na Prírodovedeckej fakulte UPJŠ v Košiciach, ktorý sa prvýkrát bude realizovať v akademickom roku 2020/2021. Garantom tohto študijného programu je prof. RNDr. Gabriel Semanišin, PhD., prorektor pre informatizáciu a riadenie kvality na UPJŠ v Košiciach a garant národného projektu IT Akadémia – vzdelávanie pre 21. storočie. Nový študijný program reaguje na zvýšený dopyt po odborníkoch v oblasti dátovej vedy na celom svete.

3.2 Lektorovanie predmetov pre informatikov a pilotná výučba predmetov pre neinformatikov na vysokých školách

Študenti partnerských univerzít projektu v rokoch 2016 až 2020 absolvovali nové a inovované predmety v jednotlivých oblastiach zamerania. Na piatich partnerských vysokých školách sa realizovala výučba 57 nových a inovovaných predmetov odborného zamerania – Dátová veda (Data Science), Internet vecí (Internet of Things), Počítačové siete (Computer Networks) a Podnikovo-informačné systémy (Business Information Systems). Údaje o počte lektorovaných predmetov na jednotlivých univerzitách sú uvedené v Tabuľke 2.

Tabuľka 2. Prehľad lektorovania predmetov počas realizácie projektu

Názov oblasti informatického vzdelávania	**Počet lektorovaných predmetov (výučba v rámci jedného semestra)**
Oblasť Data Science	19 predmetov (8 UPJŠ + 3 TUKE + 5 UKF + 2 UMB + 1 UNIZA)
Oblasť Podnikovo-informačných systémov	11 predmetov (7 CVTI + 4 UPJŠ)
Oblasť Internetu vecí	13 predmetov (4 TUKE + 4 UKF + 3 UNIZA + 2 UPJŠ)
Oblasť Sieťová akadémia	14 predmetov (5 TUKE + 4 UPJŠ + 3 UNIZA + 1 UMB + 1 UKF)
Spolu	57 predmetov

3.3 Koordinácia obsahu a metód vzdelávania so súkromným IT sektorom – konzultácie s expertmi v IT firmách, stáže učiteľov v IT firmách

Štruktúra predmetov a obsah študijných materiálov vznikali aj na základe konzultácií so zamestnancami IT firiem. Úroveň zapojenia IT firiem môžeme rozdeliť do 4 stupňov:

1. **Základná úroveň:** Komentáre a poznámky k študijným materiálom, ktoré pripravili vysokoškolskí učitelia pre všetky inovované predmety;

2. Pravidelné stretnutia na vedeckých seminároch a workshopoch, konzultácie s učiteľmi;
3. Stáže učiteľov v IT firmách;
4. **Najvyššia úroveň:** Intenzívna spolupráca pri tvorbe študijných materiálov (napríklad vo forme multimediálneho obsahu, skrípt).

Pre prvý a druhý stupeň spolupráce platilo, že pre každý nový a inovovaný predmet vysokej školy prebiehala konzultácia s **minimálne jedným expertom z IT firmy**. Experti z IT firiem konzultovali v prvej fáze projektu štruktúru a obsah študijných materiálov (roky 2016 – 2018). V rokoch 2019 až 2020 prebiehala záverečná fáza, v ktorej experti z IT firiem spracúvajú odborné recenzie na vytvorené študijné materiály. Každý predmet projektu mal minimálne jedného recenzenta študijných materiálov z IT firmy. Na základe týchto recenzií prebiehala aktualizácia študijných materiálov spracovaná učiteľmi vysokých škôl.

Počas realizácie projektu sa realizovalo 10 stáží učiteľov vysokých škôl v IT firmách. Každá stáž bola realizovaná v rozsahu 10 pracovných dní podľa vopred stanoveného harmonogramu. Skúsenosti získané na stáži boli úspešne aplikované pri tvorbe študijných materiálov a pri výučbe predmetov:

- Výpočty v prostredí SAP HANA,
- Strojové učenie,
- Prípadové štúdie dolovania údajov,
- Základy internetu vecí,
- Programovanie, algoritmy a zložitosť,
- Základy znalostných systémov,
- Počítačové siete (CCNA1/2),
- Aplikácie počítačových sietí (CCNA3/4).

V rámci národného projektu IT akadémia – vzdelávanie pre 21. storočie bolo vytvorených 26 partnerstiev, ktoré prepojili vysoké školy a podnikovú sféru. Do prípravy a realizácie nového akreditovaného študijného programu Analýza dát a umelá inteligencia na Prírodovedeckej fakulte UPJŠ v Košiciach sa zapojilo 8 IT spoločností.

3.4 Semináre, videokonferenčné semináre, účasť na konferenciách, bakalárske a diplomové práce

Konzultácia aktuálneho vzdelávacieho obsahu predmetov a výstupov a vzájomná výmena odborných poznatkov v jednotlivých oblastiach medzi partnerskými univerzitami prebiehala vo forme videokonferenčných seminárov a workshopov (Tabuľka 3). Videokonferenčné stretnutia sa konajú jedenkrát za semester za každú oblasť odborného zamerania. Stretnutia riešiteľov za oblasti Dátová veda a Podnikovo informačné systémy sa realizujú spoločne v rámci jedného videokonferenčného seminára.

Tabuľka 3. Prehľad videokonferenčných seminárov a workshopov

Názov oblasti informatického vzdelávania	**Termíny workshopov**
Oblasť Data Science a Podnikovo-informačné systémy	19. jún 2018, 4. február 2019, 4. jún, 2019, 4. február 2020, 9. september 2020
Oblasť Internetu vecí	25. jún 2018, 31. január 2019, 5. február 2020, 10. september 2020
Oblasť Sieťová akadémia	26. jún 2018, 4. február 2020, 8. september 2020

Účastníkmi niektorých workshopov počas realizácie projektu boli aj zástupcovia IT firiem. Obsahom seminárov a workshopov bola konzultácia aktuálneho vzdelávacieho obsahu predmetov vysokých škôl, prezentácia výstupov vo forme študijných materiálov a vzájomná výmena odborných poznatkov v jednotlivých oblastiach zamerania (Internet vecí, Dátová veda, Podnikovo-informačné systémy – SAP Akadémia, Sieťová akadémia).

4. júna 2019 sa s podporou národného projektu IT Akadémia – vzdelávanie pre 21. storočie realizoval workshop Dátovej vedy v Historickej aule na PF UPJŠ v Košiciach za osobnej účasti riešiteľov z partnerských univerzít. Viacerí riešitelia prezentovali výsledky dosiahnuté v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie aj na domácich a zahraničných konferenciách, napríklad ICETA, ITAT, DFRWS, ACIIDS, Informatics a iné.

V rámci projektu IT akadémia vzniklo viacero bakalárskych a diplomových prác aplikačného charakteru vo všetkých 4 oblastiach odborného zamerania – Dátová veda (Data Science), Internet vecí (Internet of Things), Počítačové siete (Computer Networks)

a Podnikovo-informačné systémy (Business Information Systems). Príkladom úspešnej bakalárskej práce je práca Bc. Tomáša Kekeňáka s názvom Objavovanie znalostí v časových údajoch pomocou databáz uložených v pamäti pod vedením Ing. Mirona Kuzmu, PhD., riešiteľa projektu za oblasť Podnikovo-informačných systémov. Rozvoj internetu totiž spôsobil, že v dnešnej dobe je potrebné veľké množstvo dát spracovať a analyzovať v reálnom čase. Dnešní používatelia požadujú nulové doby odozvy od aplikácií, ktoré používajú, aj v prípade, že ide o komplexné aplikácie pracujúce s veľkými množstvami dát. Práve tieto problémy a zlacnenie počítačovej pamäte viedli k vzniku novej generácie databáz, ktoré majú všetky dáta uložené v operačnej pamäti, čím sa extrémne zrýchli práca s danými dátami.

3.5 Tvorba národného materiálu v troch oblastiach odborného zamerania kompetenčného centra

Na základe vytvorených študijných materiálov bolo publikovaných 15 učebných materiálov s ISBN na všetkých partnerských školách.

Tri publikácie vznikli vo forme národného výstupu, konkrétne:

- Národný výstup – Počítačové siete, základné koncepty s konfiguráciou,
- Národný výstup – Dátová veda a jej aplikácie,
- Národný výstup – Internet vecí a jeho aplikácie.

Uvedené výstupy v rámci podaktivity 2.1 Inovácia prípravy študentov vysokých škôl pre zamestnanie v IT sektore národného projektu IT Akadémia – vzdelávanie pre 21. storočie sú prínosné z hľadiska dopadov projektu z pohľadu študenta, učiteľa ako aj IT firiem. Deklarujeme rozšírenie odborných kompetencií pre orientáciu žiakov na štúdium informatiky a IKT, spoluprácou univerzít s IT firmami sledujeme vytvorenie dlhodobého podnetného prostredia pre ďalší odborný rast, dlhodobý a udržateľný systém prípravy odborníkov pre aktuálne a budúce potreby IT sektora.

4 Internet vecí vo výučbe

V tejto kapitole prinášame prehľad rozšírených anotácií vysokoškolských predmetov, stručné osnovy, aplikačné príklady a prípadové štúdie v oblasti internetu vecí.

4.1 Inteligentné kyber-fyzikálne systémy a IoE, TUKE v Košiciach
(Iveta Zolotová, Peter Papcun, Ján Vaščák)

4.1.1 Anotácia a stručná osnova predmetu

Predmet predstavuje inovácie v oblasti IoE (Internet of Everything) pre štvrtú priemyselnú revolúciu. Študenti získajú prehľad o modeloch, metódach, architektúrach, platformách, scenároch IoT (Internet of Things) riešení, ktoré svojou výraznou inovatívnou mierou napomáhajú priemyslu rásť rýchlejšie, poskytovať lepšie dáta pre riadenie a rozhodovanie a ďalší vývoj. Predmet modeluje typické prepojenia a scenáre, snímače a akčné členy, protokoly, siete, služby, vnorené systémy, jednodoskové počítače a inteligentné uzly na niekoľkých prípadových štúdiách a projektoch.

Predmet Inteligentné kyber-fyzikálne systémy a IoE je ponúkaný a bol vyučovaný v druhom ročníku bakalárskeho štúdia v študijnom programe Inteligentné systémy v odbore Informatika (ako výberový predmet). Je vhodný pre rôzne študijné programy informatického a kybernetického zamerania.

Témy prednášok (cvičení) predmetu:

1. Úvod a motivácia (Úvod a motivácia)
2. Prepojený kybernetický svet (Prepojený kybernetický svet)
3. Prípadové štúdie (Prípadové štúdie)
4. Referenčné architektúry, ekosystém (Laboratórne prípadové štúdie)
5. Design thinking, SCRUM (RESTful API ako komunikačné IoT rozhranie 1)
6. SOA a dodávatelia cloudových služieb v priemysle (RESTful API ako komunikačné rozhranie 2)
7. Najväčší dodávatelia cloudových služieb (RESTful API ako komunikačné rozhranie 3)
8. IoE a Industry 4.0 (Machine Learning v IoE a IoT)

9. Operátor 4.0 (Robotika)
10. Internet vecí v robotike (Internet vecí v robotike)
11. Smart a inteligentné priestory pre robotické aplikácie (Smart a inteligentné priestory pre robotické aplikácie)
12. Aspekty robotiky v kontexte IoT (Aspekty robotiky v kontexte IoT)

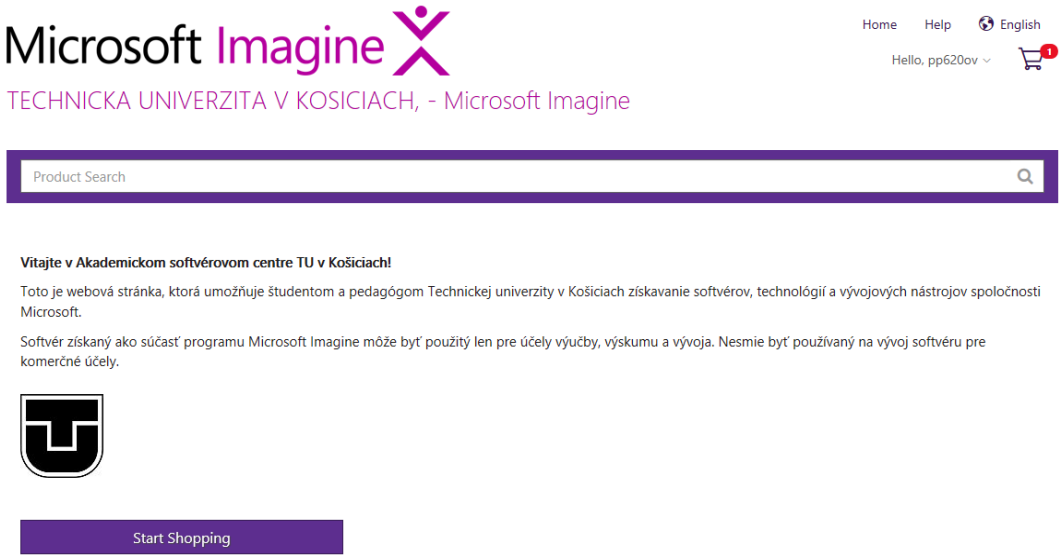
4.1.2 Aplikačný príklad

Ako aplikačný príklad sme si vybrali RESTful API ako komunikačné IoT rozhranie. Tri cvičenia sa zaoberajú pripojením IoT zariadení ku cloud-u pomocou aplikačného komunikačného rozhrania (API): RESTful API. Prvé cvičenie sa študenti zoznámia s prostredím Microsoft Azure a s vytváraním služieb ako napríklad Web App. Ďalšie cvičenie bude nadväzovať na predchádzajúce. Počas druhého cvičenia sa študenti zoznámia s prostredím Arduino IDE a s mikrokontrólerom ESP8266. Úlohou cvičenia bude prepojenie ESP s cloudovou službou. Tretie cvičenie sa zaoberá ukladaním dát do SQL databázy a spracovaním odpovede pre IoT zariadenie. Základná funkcionalita prenosu dát na cloud je vysvetlená na dvoch cvičeniach, preto v tomto materiály bude aplikačný príklad venovaný práve nim. Na týchto cvičeniach študenti pracujú samostatne (jednotlivá práca), počas práce sa môžu cvičiaceho pýtať prípadné otázky ku vypracovaniu, alebo téme aplikačného príkladu (skupinové odpovede na otázky). Prvé dve cvičenia zo série „RESTful API ako komunikačné IoT rozhranie“ sa zaoberajú týmito hlavnými témami:

- vytvorenie účtu na MS Azure,
- vytvorenie jednoduchej webovej aplikácie (WebApp),
- vysvetlenie a ukážka fungovania: RESTful API,
- prostredie Arduino IDE a následná konfigurácia dosiek a knižníc,
- naprogramovanie HTTP klienta v prostredí Arduino IDE,
- vytvorenie webovej aplikácie pre spracovanie dát z ESP8266.

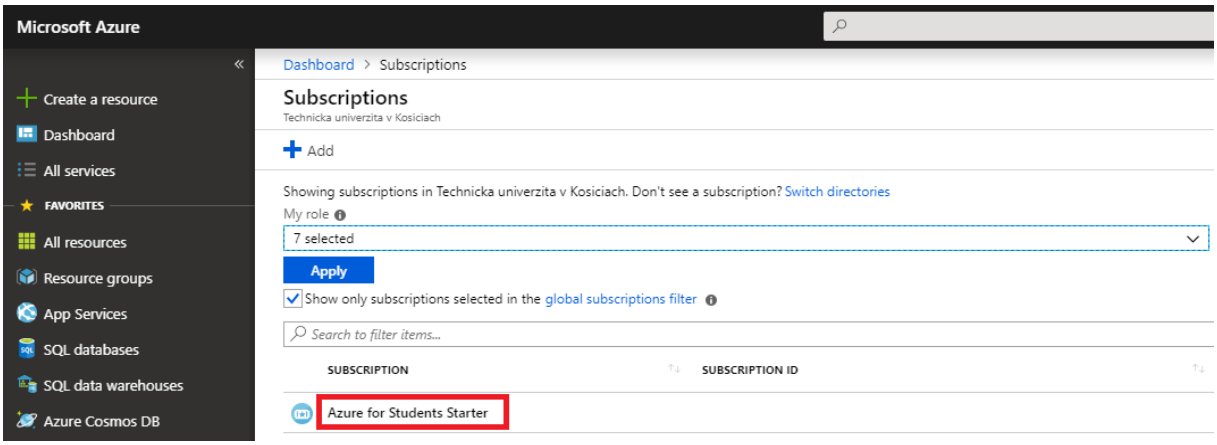
4.1.2.1 Vytvorenie účtu na MS Azure (30 min)

Na webovej stránke: https://nastavenia.tuke.sk/msdn/postup majú študenti prístup k voľnému softvéru a službám spoločnosti MS, po prihlásení cez školský účet budú presmerovaný na webstránku MS Imagine (Obrázok 5), kde do textového poľa „Product search“ zadajú „Azure Account“.



Obrázok 5. Web: Microsoft Imagine

Toto slovné spojenie vyhľadá produkt *New Microsoft Azure for students*. Tento produkt si zakúpia (zdarma). Následne vo faktúre kliknú na odkaz „Click here to access Microsoft Azure for Students“, ktorý ich presmeruje na formulár, ktorý je potrebné vyplniť (30 minút). Cvičiaci pri problémoch môže pomôcť študentom a vysvetliť, čo nesplnili podľa priloženého postupu. Na záver cvičiaci skontroluje pomocou záložky „Subscription“, či sa každému študentovi podarilo účet vytvoriť.

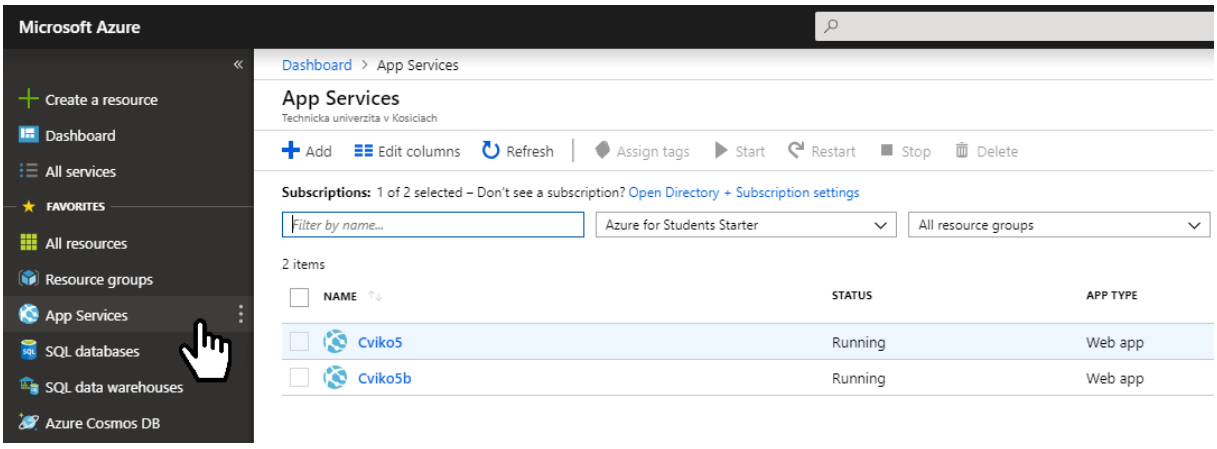


Obrázok 6. Záložka *Subscription* na portály MS Azure

Ak sa študentovi podarilo správne vytvoriť účet, tak na mieste červeného políčka (Obrázok 6) sa musí nachádzať text „Azure for Student“, pozor text „Azure for Student Starter“ nie je správny názov subskripcie.

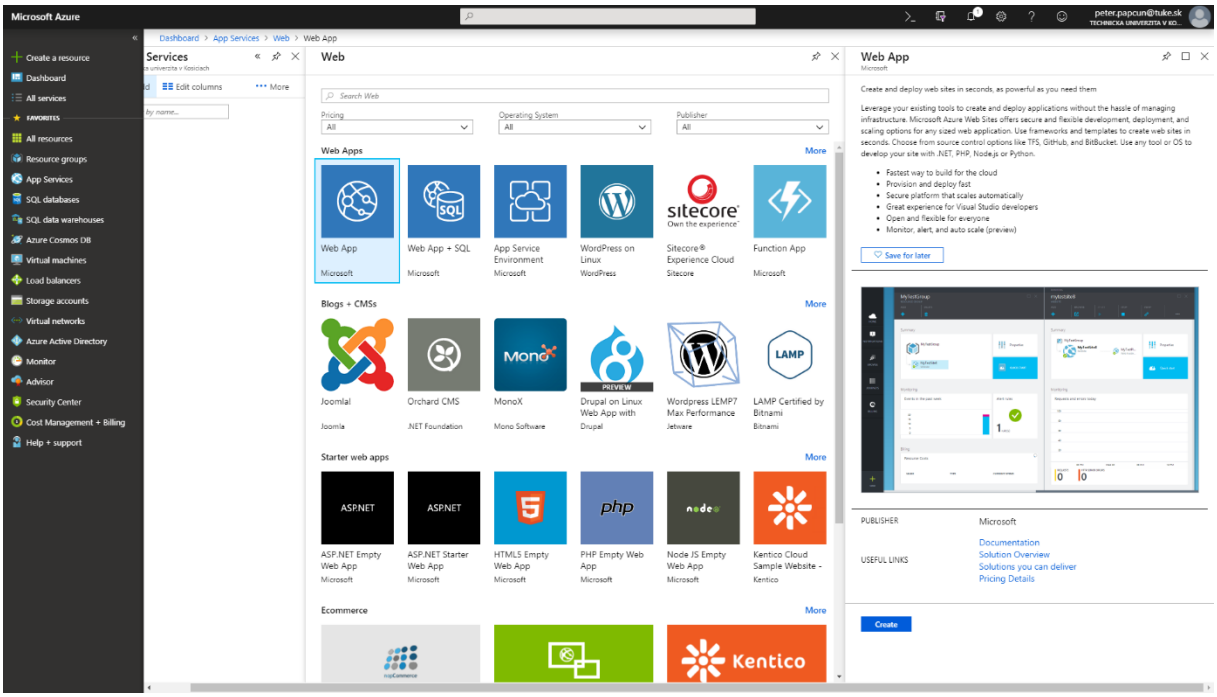
4.1.2.2 Vytvorenie webovej aplikácie (30 min)

Po vytvorení účtu si študenti vytvoria webovú aplikáciu pomocou portálu Azure. Existuje možnosť vytvorenia aplikácie pomocou IDE: Microsoft Visual Studio 2017, ale v tom prípade portál zvolí spoplatnenú verziu webovej aplikácie. Preto je odporúčané aplikáciu vytvoriť pomocou portálu. V portáli študenti kliknú na „App Services“ (Obrázok 7).



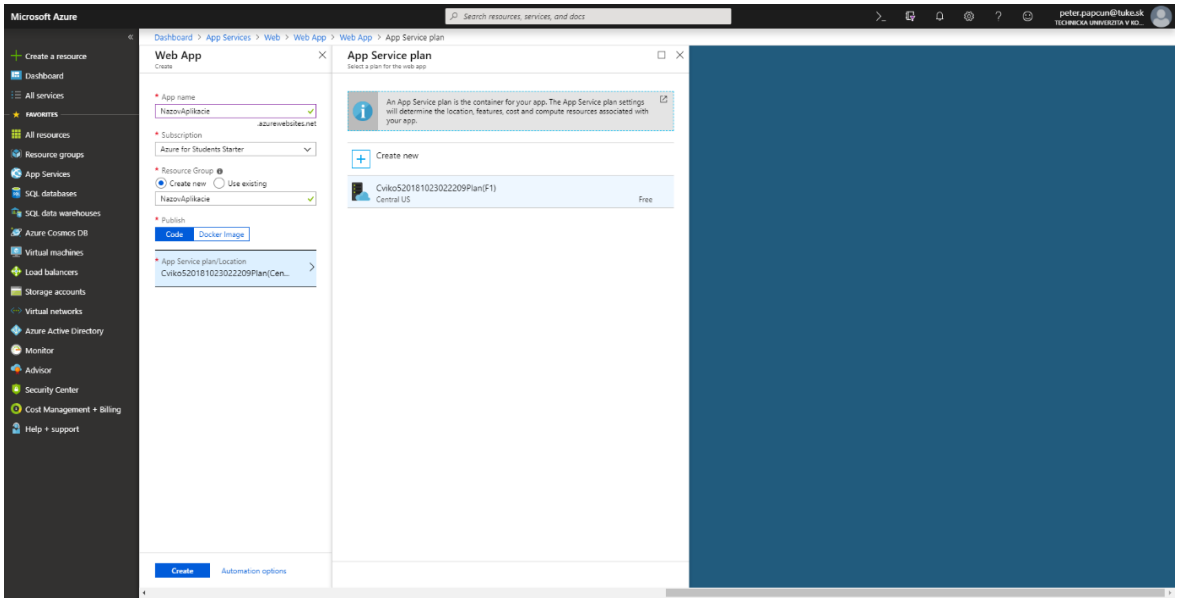
Obrázok 7. Otvorenie záložky App Services

Po otvorení záložky klikneme na možnosť „Add“ pri modrej ikone plus (Obrázok 7). Vyberieme si možnosť „Web App“ (Obrázok 8) a stlačíme tlačidlo „Create“ (Obrázok 8). Následne vyplníme formulár (Obrázok 9) a klikneme na tlačidlo „Create“.

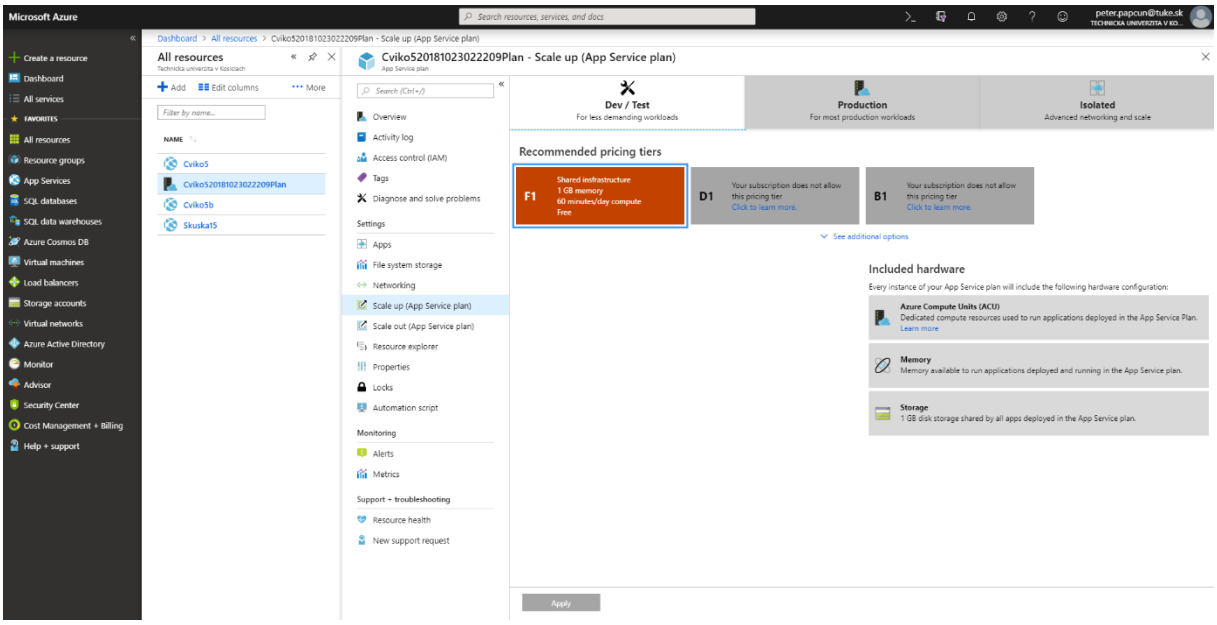


Obrázok 8. Vytvorenie Web App

Uistíme sa, že „App service plan“ je zadarmo (Free). Ak nie je zadarmo, treba ho nastaviť. Po vytvorení služby „Web App“ klikneme na záložku „All Resources“, kde si vyberieme náš „App service plan“.

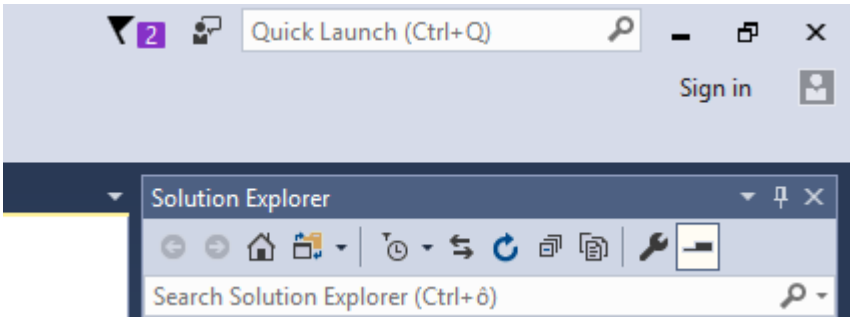


Obrázok 9. Formulár pre vytvorenie Web App



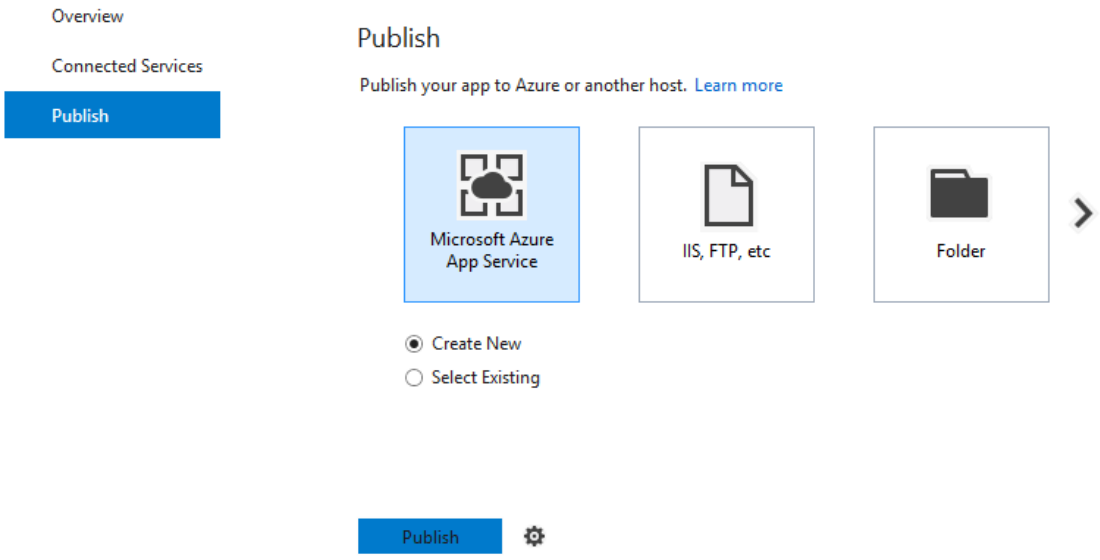
Obrázok 10. Nastavenie „App service plan“

Pomocou možnosti „Scale up“ alebo „Scale out“ nastavíme túto službu zadarmo (free, Obrázok 10). Teraz je webová aplikácia pripravená na používanie. Teraz je možné spustiť IDE: Microsoft Visual Studio 2017, kde sa spárujú IDE s vytvorenou službou na cloud-e, najskôr pomocou prihlásenia (Sign in, Obrázok 11) a potom aj pri vkladaní na cloud v možností „Publish“.



Obrázok 11. Prihlásenie MS Visual Studia 20O17 pomocou Microsoft konta

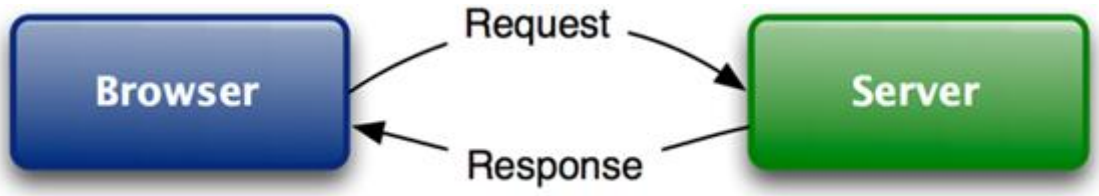
Po spárovaní si vytvoria novú webovú aplikáciu typu WebForm, kde vytvoria pomocou Designera jednoduchý formulár, ktorý vložia na cloud pomocou možnosti „Publish“, kde si vyberú možnosť „Select Existing“ a kliknú na tlačidlo „Publish“ (Obrázok 12). Potom budú študenti vyzvaný aby si vybrali službu ktorú si na cloud-e vytvorili.



Obrázok 12. Vloženie aplikácie na cloud

4.1.2.3 Vytvorenie a ukážka fungovania jednoduchej RESTful API (30 min)

Vysvetlenie fungovania RESTful API (get, post, delete, put, patch), a spôsob dopytovania pomocou http request-ov (Obrázok 13):



Obrázok 13. HTTP request

Najskôr sa študentom ukáže ako vedia programovo spracovať dodatočné informácie z URL adresy svojej aplikácie (GET Requst), vytvoria si tzv. server:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;

namespace DBREST
{
      public partial class REST1 : System.Web.UI.Page
      {
            protected void Page_Load(object sender, EventArgs e)
            {
                  string text1=Request.QueryString["meno"];
                  string text2=Request.QueryString["przv"];
                  Label1.Text = "Ahoj " + text1 + " " + text2;
            }
      }
}
```

Následne sa študentom ukáže, ako si môžu programovo vytvoriť klienta a tak vygenerovať akúkoľvek URL adresu. Pre daného klienta je veľmi dôležitá aj odpoveď servera, preto v kóde ktorý bude postupne študentom vysvetlený nachádza aj príjem odpovede:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Net;
using System.Text;
using System.IO;

namespace DBREST
{
      public partial class REST2 : System.Web.UI.Page
      {
            protected void Page_Load(object sender, EventArgs e)
            {}
            protected void Button1_Click(object sender, EventArgs e)
            {
            StringBuilder sb = new StringBuilder();
            byte[] buf = new byte[8192];
            HttpWebRequest request = (HttpWebRequest)
            WebRequest.Create("http://rt1.azurewebsites.net/REST1.asp
                  x?meno=Jan&przv=Dak");

            HttpWebResponse response = (HttpWebResponse)
            request.GetResponse();
```

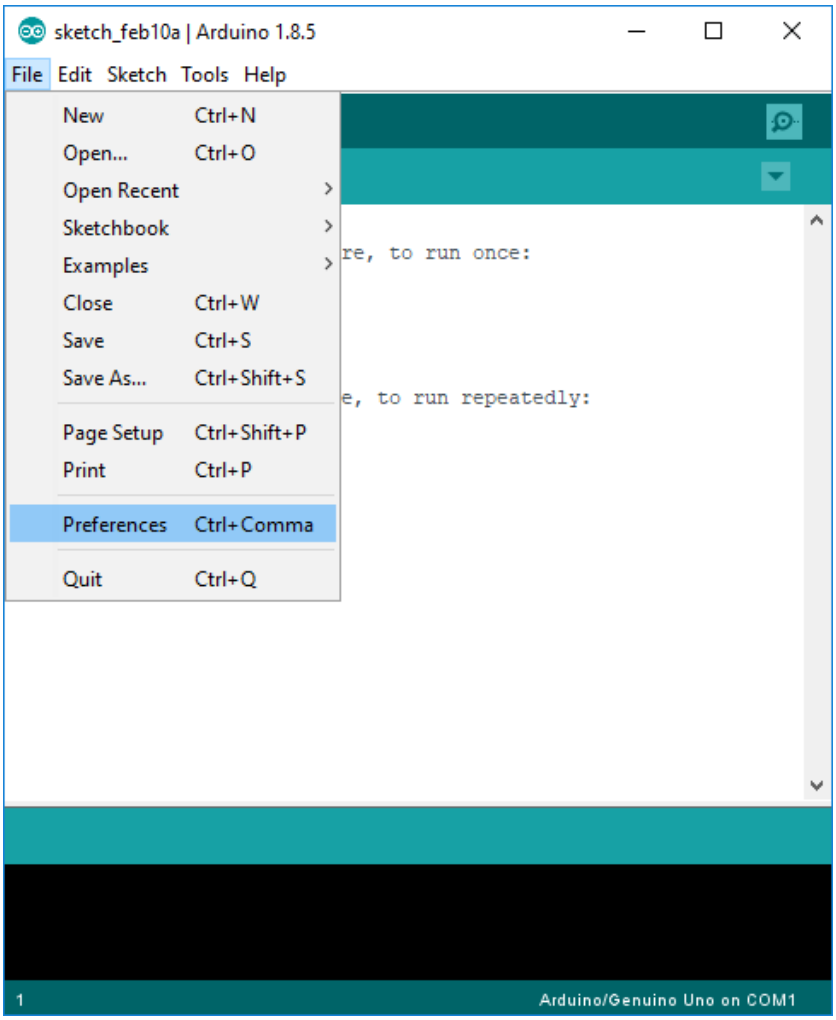
```
            Stream resStream = response.GetResponseStream();
            string tempString = null;
            int count = 0;

            do
                  {
                        count = resStream.Read(buf, 0, buf.Length);
                        if (count != 0)
                        {
                              tempString =
                        Encoding.ASCII.GetString(buf, 0, count);
                              sb.Append(tempString);
                        }
                  }
            while (count > 0);
            Response.Write(sb.ToString());
            TextBox1.Text = sb.ToString();

            resStream.Dispose();
            response.Dispose();
            }
      }
}
```

4.1.2.4 Prostredie Arduino IDE a následná konfigurácia dosiek a knižníc (40 min)

V úvode cvičenia sa študenti zoznámia s prostredím Arduino IDE a jeho možnosťami, konfiguráciou, monitorom sériovej linky, atď. Následne pridajú medzi programovateľné dosky ESP8255 a to kliknutím na položku *File* a z ponuky sa vyberie *Preferences* (Obrázok 14).

Otvorí sa okno *Preferences*, kde sa vypíše do položky *Additional Boards Manager URLs* nasledovná URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json.

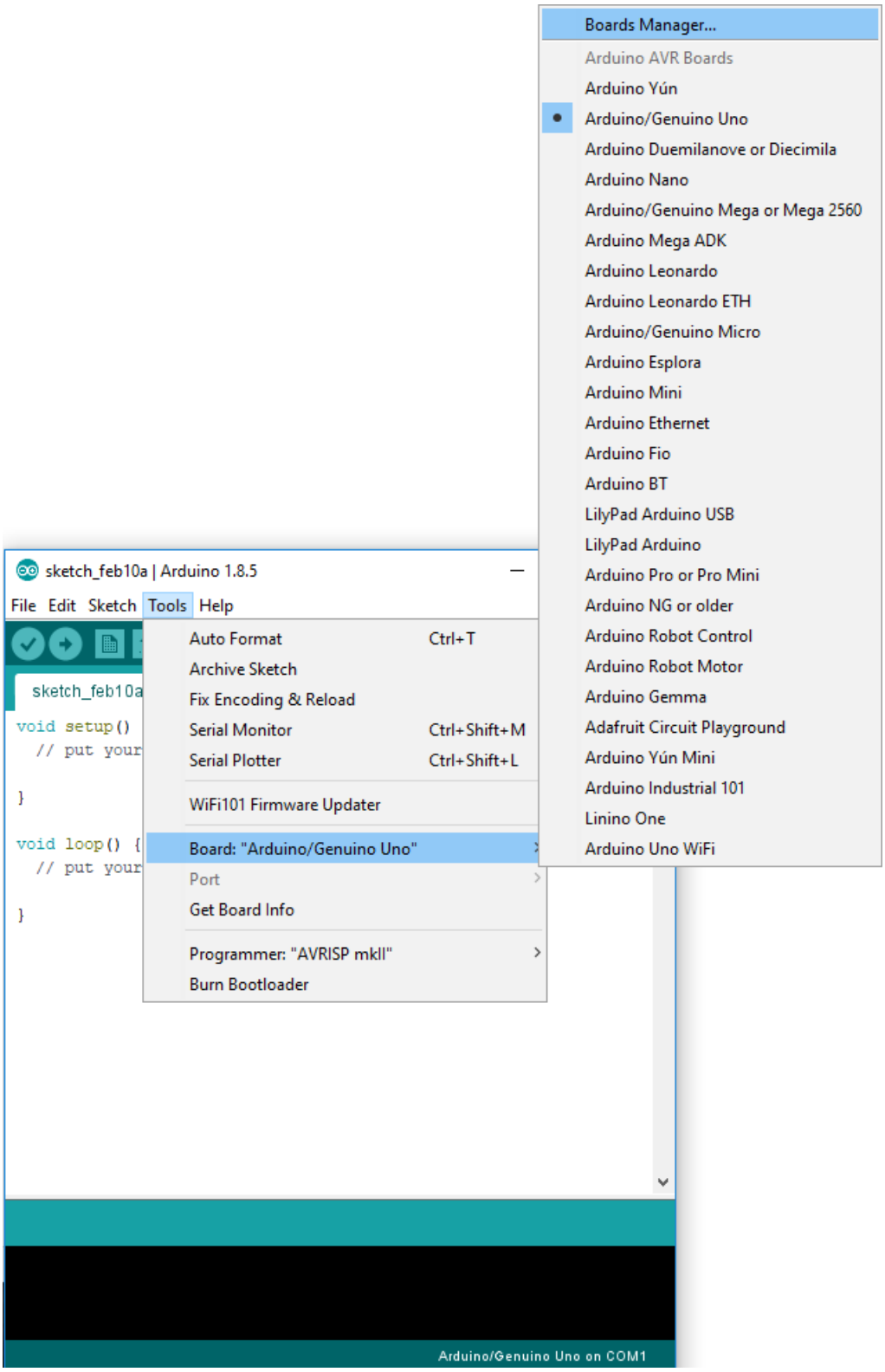
Napokon sa okno *Preferences* zatvorí potvrdením *OK*.



Obrázok 14. Otvorenie okna „Preferences“

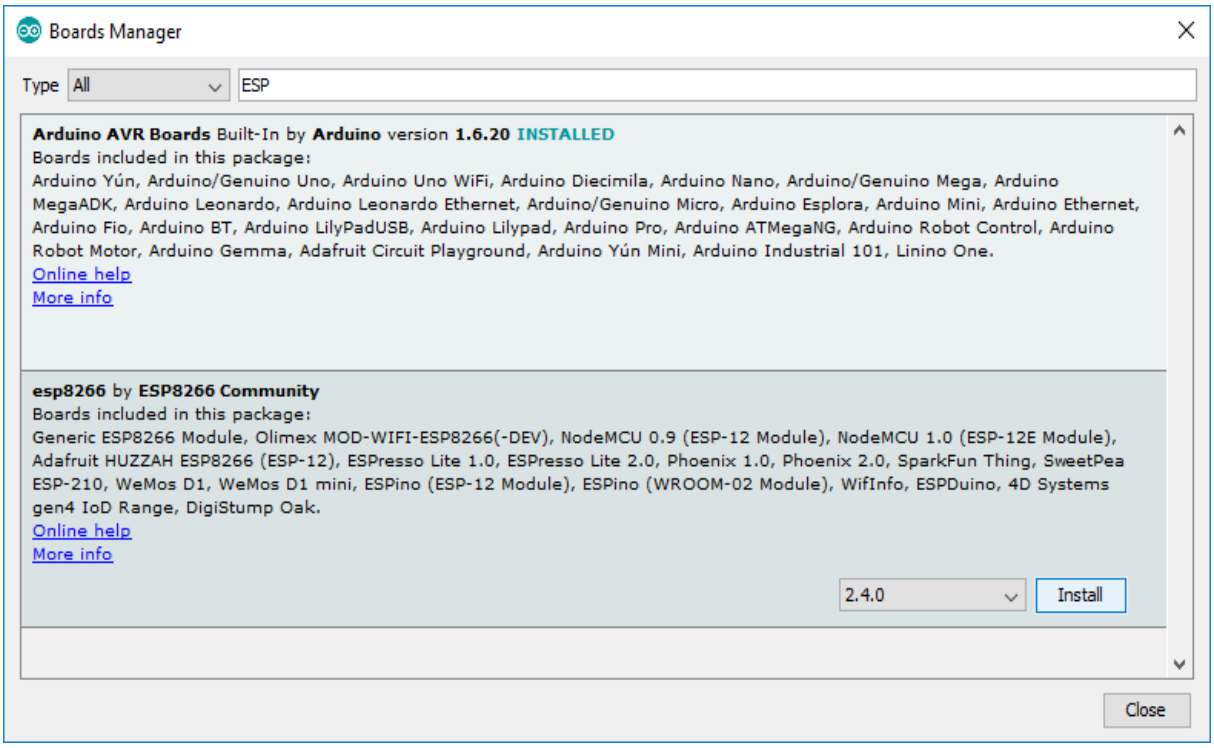
Ďalším krokom je otvorenie okna *Boards Manager* (Obrázok 15). V okne *Boards Manager* sa nainštaluje doska (mikrokonróler) ESP 8266, inštalácia sa hľadá pomocou kľúčového slova ESP (Obrázok 16).

Po inštalácii je už možný výber dosky NodeMCU 1.0 v ponuke dosiek (*Tools*/*Board*). Po vybratí dosky sa nastaví prenosová rýchlosť sériovej linky pri nahrávaní programu do čipu a port cez ktorý je doska pripojená k počítaču.



Obrázok 15. Otvorenie okna „Board Manager“

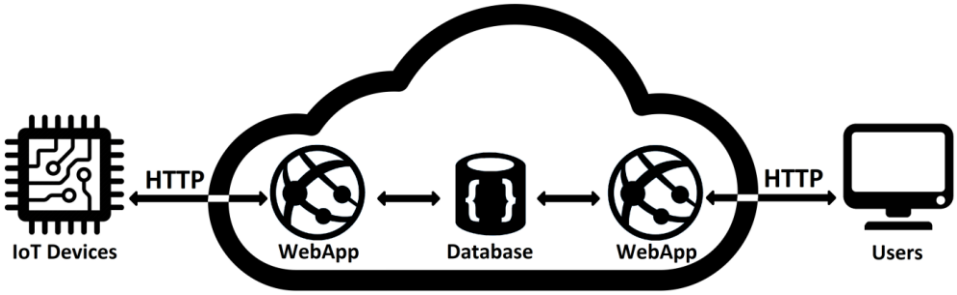
V prípade potreby implementácie nových knižníc do prostredia Arduino IDE postupujeme podobne, ale nepoužijeme okno *Board Manager*, ale okno *Manage Libraries*, ktoré sa otvorí pomocou možnosti *Sketch* v menu prostredia a v ňom sa vyberie položka *Include Library*.



Obrázok 16. Okno „Board Manager“

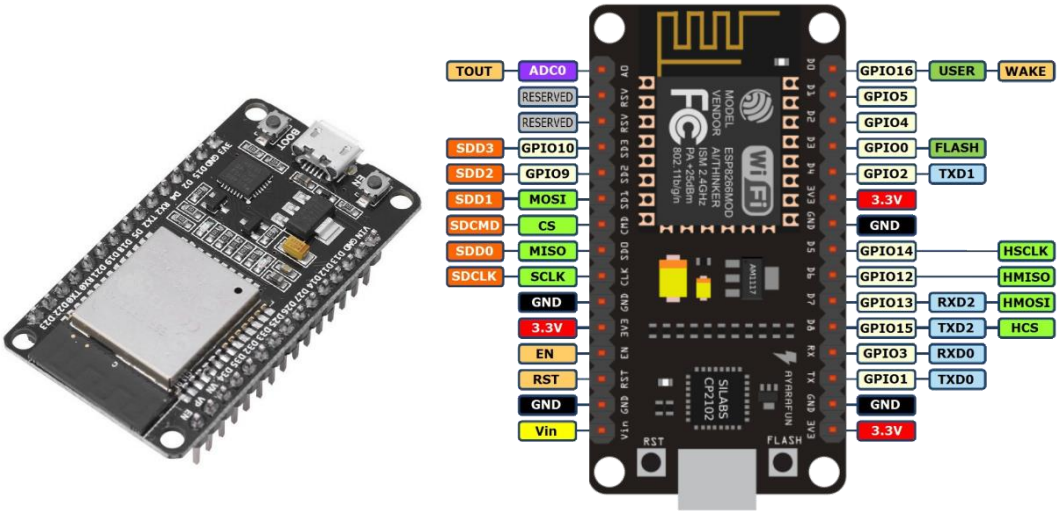
4.1.2.5 Naprogramovanie HTTP klienta v prostredí Arduino IDE (30 min)

Architektúra komunikačného rozhrania RESTful API, ktorá bude implementovaná na tomto cvičení je na Obrázku 17.



Obrázok 17. Architektúra komunikačného rozhrania RESTful API s využitím cloudových služieb

Pred programovaním sa ešte bližšie študenti oboznámia s možnosťami dosky ESP8266 (Obrázok 18).



Obrázok 18. ESP8266

Po konfigurácií prostredia Arduino IDE sa môže začať programovať. Následný kód bude študentom vysvetlený:

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "Industry4";
const char* password = "Industry4";

void setup()
{
      Serial.begin(115200);
```

```
	WiFi.begin(ssid, password);

	while (WiFi.status() != WL_CONNECTED)
	{
		delay(1000);
		Serial.print("Connecting..");
	}
}

void loop()
{
	//Kontrola pripojenia k Wi-Fi
	if (WiFi.status() == WL_CONNECTED)
	{
		//Deklarácia objektu triedy HTTPClient
		HTTPClient http;

		//Vyskladanie URL
		http.begin("http://..../.../...aspx?...");

		//Request na server
		int httpCode = http.GET();

		//Dostali sme odpoveď?
		if (httpCode > 0)
		{
			//Načítanie odpovede do String-u
			String	payload = http.getString();

			//Výpis odpovede na sériovú linku
			Serial.println(payload);
		}
		//Zatvorenie linky so serverom
		http.end();
	}
	//Ďalší request sa pošle o 30 sekúnd
	delay(30000);
}
```

Následne sa otestuje funkčnosť aplikácie.

4.1.2.6 Vytvorenie webovej aplikácie pre spracovanie dát z ESP8266 (20 min)

Samostatná práca: na základe učiva z minulého a tohto cvičenia (obe cvičenia sú zahrnuté v tomto materiály) vytvorte aplikácie, ktoré budú:

- vizualizácia dát z ESP8266 vo webovej aplikácií,
- webová aplikácia bude načítavať 2 náhodné čísla z ESP8266, ktoré následne spočíta.

4.2 Internet vecí, UNIZA v Žiline (Peter Ševčík)

4.2.1 Anotácia a stručná osnova predmetu

Po úspešnom absolvovaní predmetu Internet vecí študent rozumie základným princípom, na ktorých je postavený svet internetu vecí. Študent je schopný vybrať vhodné technické prostriedky (platformu, senzory, akčné členy) a použiť ich vo svojej aplikácii. Získané dáta dokáže predpracovať a následne poslať, uložiť a prezerať vo vybranej cloudovej službe.

Stručná osnova predmetu:

Prednášky:

1. Úvod do sveta internetu vecí.
2. Základné pojmy.
3. Prehľad platforiem pre internet vecí – Arduino.
4. Prehľad platforiem pre internet vecí – Raspberry Pi.
5. Senzorický subsystém.
6. Možnosti ovládania akčných členov.
7. Komunikačný subsystém.
8. Možnosti ukladania a spracovania nameraných dát.
9. Ochrana súkromia a bezpečnosť v aplikáciách internetu vecí.
10. Zdroje napájania aplikácií internetu vecí.
11. Vybrané aplikácie internetu vecí (Smart city).
12. Vybrané aplikácie internetu vecí (Inteligentné budovy).
13. Vybrané aplikácie internetu vecí (Priemysel 4.0).

Cvičenia:

1. Zoznámenie sa prácou v IT Science laboratóriu.
2. Základné elektrotechnické zručnosti (meranie vybraných veličín, identifikácia jednotlivých elektronických komponentov).
3. Arduino.
4. Raspberry Pi.

5. Snímanie vybranej fyzikálnej veličiny na platforme Arduino.
6. Snímanie vybranej fyzikálnej veličiny na platforme Raspberry Pi.
7. Ovládanie vybraných akčných členov.
8. Komunikácia a posielanie dát.

9.-13. Práca na semestrálnej práci.

4.2.2 Aplikačný príklad

V aplikačnom príklade prezentujeme rozširujúcu dosku plošných spojov pre modul ESP32-WROOM-32U. Návrh tejto dosky je v nasledujúcej podobe:

1. Základné údaje o module ESP32:
    - Výrobca: Espressif
    - Bezdrôtová komunikácia: WiFi, Bluetooth LE
    - Napájanie: 3 – 3,6 VDC
    - Konektor antény: U.FL
    - Kapacita pamäte flash: 4MB
    - Počet jadier procesora: 2
    - Komunikačné rozhrania: GPIO, I2C, IR, SD, SPI, I2S, UART, SDIO
    - Programovanie prostredníctvom rozhrania: UART

2. Stručný popis jednotlivých častí:

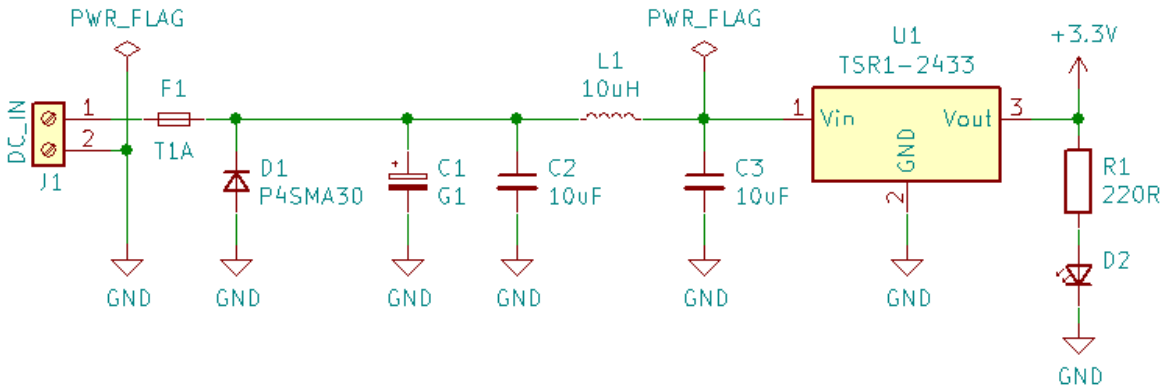
Rozširujúca doska je navrhnutá s ohľadom na využitie tohto modulu či už vo výučbovom procese predmetu Internet vecí alebo v priemyselnej aplikácii ako riadiaci automatizačný modul. Z tohto dôvodu je modul navrhnutý s ohľadom na:

- Napájanie dosky: 12 – 24 VDC
- Optické oddelenie vstupno-výstupných obvodov s ochranou proti prepätiu
- Optickou indikáciou stavu jednotlivých vstupov a výstupov
- Akustickou indikáciou a optickou indikáciou RGB LED diódou
- Rozširujúcim konektorom pre nadstavbové aplikácie
- Konektorom pre pripojenie senzoru DHTxx

V ďalšej časti bude opísaný vždy len jeden kanál zapojenia, ostatné sú zapojené identicky, ak nie je uvedené inak.

2.1. Napájací subsystém:

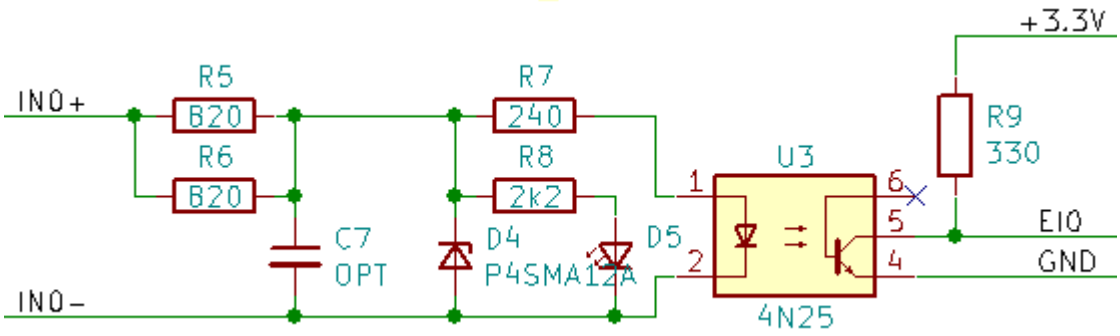
Napájací subsystém je tvorený spínaným step-down meničom TRACO TSR1 – 2433 (Obrázok 19). Rozsah vstupných napätí je 4,7 – 36VDC s pevným výstupom 3,3VDC a maximálnym prúdom 1A. Menič obsahuje integrované ochrany proti skratu a preťaženiu. Napájací zdroj je doplnený o ochranu proti prepätiu a prepólovaniu poistkou F1 a transilom D1. LC filter na vstupe (C2, L1, C3) zabraňuje spätnému šíreniu rušenia z meniča do rozvodov napájacieho napätia.



Obrázok 19. Napájací obvod modulu

2.2. Vstupné obvody

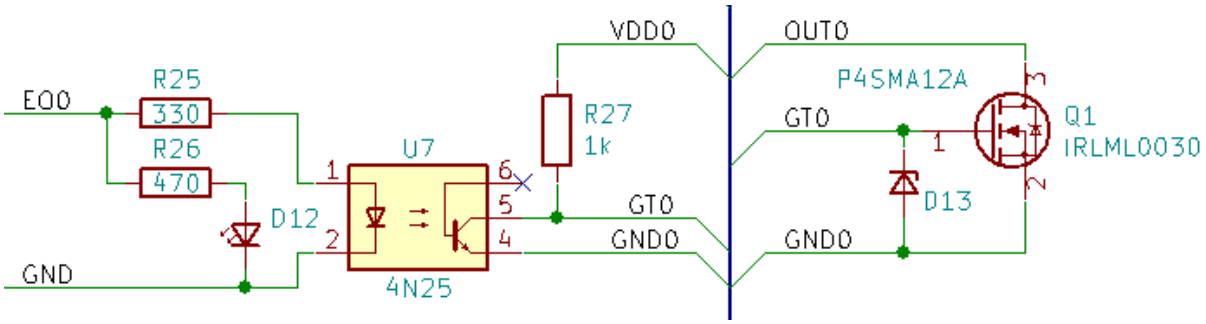
Vstupný obvod (Obrázok 20) je navrhnutý tak, aby spoľahlivo pracoval so vstupnými úrovňami signálu v rozsahu 5 – 24 VDC. Zároveň je možné upraviť frekvenčné parametre vstupného signálu dolnopriepustným filtrom tvoreným súčiastkami (R5, R6, C7). Vstupný signál privedený na vstupný konektor je galvanicky oddelený od napájacích i logických obvodov modulu ESP32 optočlenom U3. Správne napäťové úrovne výstupného tranzistora optočlenu zabezpečuje pull-up rezistor R9. Tento obvod je identický pre všetky vstupné kanály okrem IN3, kde pull-up rezistor pre optočlen sa neosadzuje a musí byť zapnutý interný rezistor v module ESP32 programovo.



Obrázok 20. Zapojenie vstupného obvodu

2.3. Výstupný obvod

Zapojenie výstupného obvodu je na Obrázku 21. Výstupný tranzistor optočlenu U7 ovláda výstupný mosfet IRLM0030 a tým je zabezpečená napäťová i prúdová nezávislosť výstupu výstupného obvodu jednak od modulu ESP32 a jednak jednotlivých výstupov medzi sebou navzájom. Výstupný obvod priamo môže spínať prúdy do veľkosti 4A.



Obrázok 21. Zapojenie výstupného obvodu

2.4. Akustická indikácia a optická indikácia RGB LED diódou

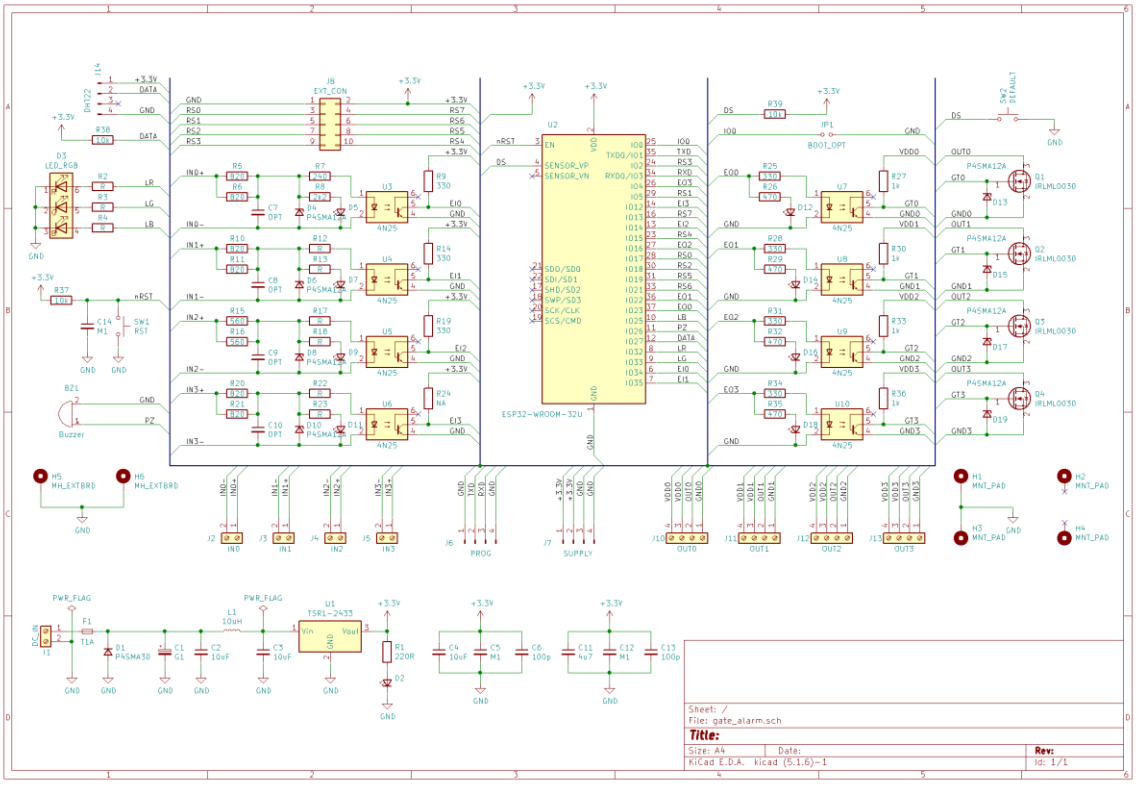
Na indikáciu prevádzkových alebo poruchových stavov obsahuje doska RGB LED diódu D3 a bzučiak BZ1 pripojené priamo na výstupy modulu ESP32.

2.5. Rozširujúci konektor J8

Na doske samozrejme nie je možné integrovať všetky periférie, ktoré by užívateľ mohol v budúcnosti potrebovať. Preto je na doske osadený rozširujúci konektor J8, na ktorom je prístupné jednak napájanie pre nadstavbovú dosku a jednak 8 dátových liniek modulu ESP32. Nadstavbovú dosku je možné aj mechanicky upevniť pomocou montážnych otvorov H5 a H6.

2.6. Konektor pre pripojenie senzoru DTHxx

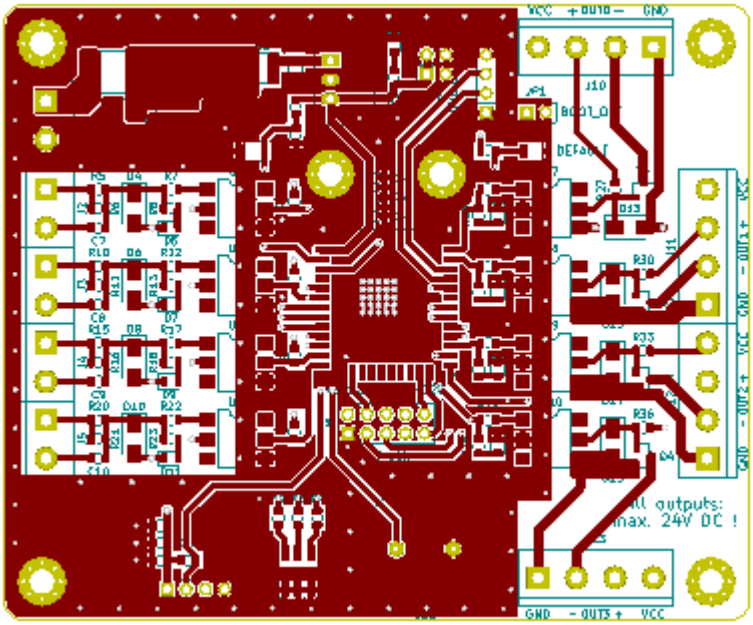
Na rozširujúcej doske je pripravené miesto pre priame pripojenie integrovaného senzoru vlhkosti a teploty typu DHTxx.



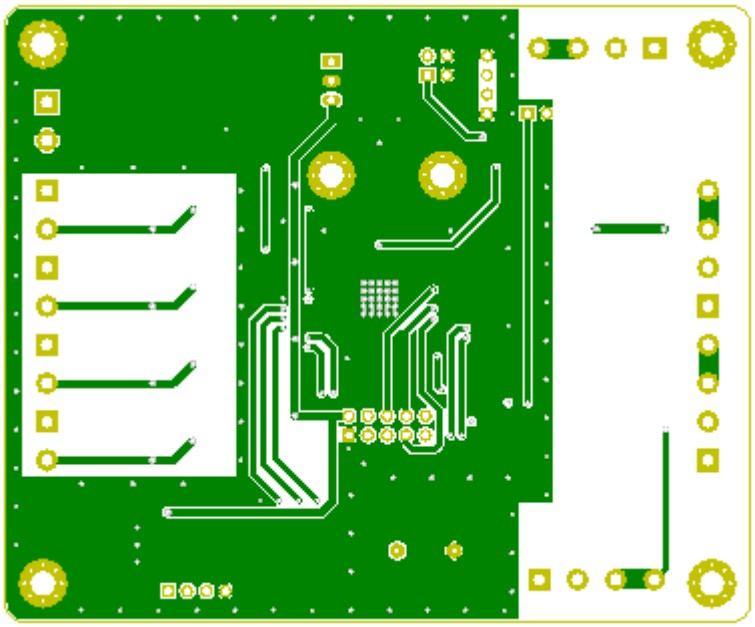
Obrázok 22. Celková schéma zapojenia rozširujúcej dosky

Schéma rozširujúcej dosky a samotný návrh dosky plošných spojov bol realizovaný v návrhovom programe KiCad (Obrázok 22). KiCad je multiplatformový voľne dostupný program určený na návrh schém a dosiek plošných spojov. V rámci projektu IT akadémia prebehli aj on-line webináre zamerané na podporu tvorby dosiek plošných spojov v tomto programe.

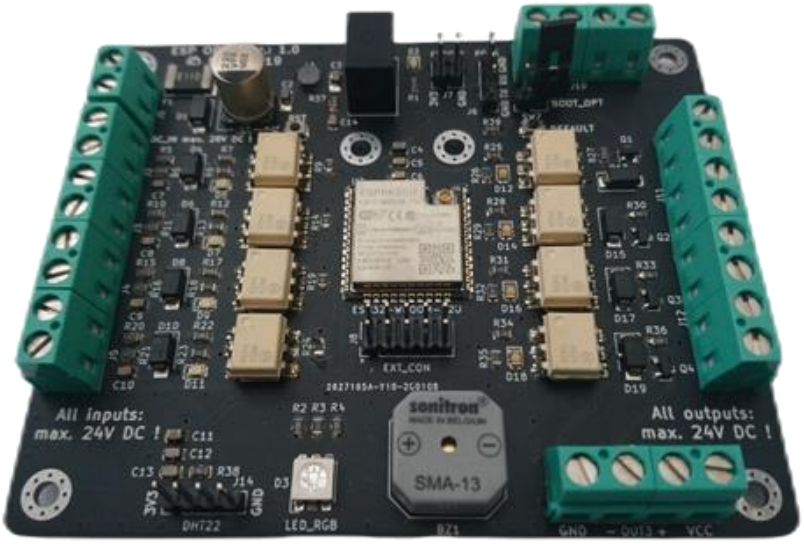
Doska plošných spojov je navrhnutá s ohľadom na jej stanovené využitie. Je koncipovaná ako obojstranná (Obrázok 23, Obrázok 24), so súčiastkami osadenými na strane TOP. Využíva zmiešanú montáž (SMD i vývodové súčiastky). Je dodržané striktné oddelenie modulu ESP od vstupno-výstupných obvodov (napr. i izolačnou vzdialenosťou plochy rozliatej zeme od vstupno-výstupných obvodov). Doska bola osadená, oživená a následne bola overená jej funkčnosť. Fotografia hotovej dosky ja na Obrázku 25.



Obrázok 23. Motív dosky plošných spojov zo strany TOP



Obrázok 24. Motív dosky plošných spojov zo strany BOTTOM



Obrázok 25. Fotografia hotovej rozširujúcej dosky

4.3 Kybernetika, kognitívne systémy a Internet vecí, UKF v Nitre
(Zoltán Balogh)

4.3.1 Anotácia a stručná osnova predmetu

Študent získa znalosti z oblasti princípov a metód používaných pri nasadzovaní, prevádzke a údržbe informačných a riadiacich systémov, zariadení prevádzkovaných v súlade so všeobecnými princípmi informačného zabezpečenia, rozhodovania a riadenia výrobných a organizačných systémov. Absolvent bude prakticky zručný v zavádzaní moderných informačných a sieťových technológií do riadenia a organizácie systémov, v udržiavaní systémov priemyselnej informatiky, vo využívaní funkčných a prevádzkových možností prostriedkov informačných a rozhodovacích systémov, získa základné vedomosti o kognitívnych robotických systémoch, senzorických sieťach a Internetu vecí.

Pojem inteligentný systém sa často používa v odbornej literatúre aj v popularizačných časopisoch. Fenomén Internetu vecí (Internet of Things, v skratke IoT) sa objavuje všade okolo nás s rôznymi interpretáciami a definíciami. Internet veci je sieť fyzických objektov, zariadení, strojov a iných predmetov s pokročilou elektronikou, ktorý obsahuje softvér, senzory a pripojenie k sieti, umožňuje zber a výmenu údajov. Internet veci umožňuje objekty snímať a ovládať na diaľku cez existujúce sieťové infraštruktúry, vytvára príležitosti pre ďalšiu priamu integráciu fyzického sveta do počítačových systémov, ktorých výsledkom je zvýšenie efektivity, presnosti a má ekonomický prínos. IoT je rozšírená pomocou senzorov a akčných členov čím sa technológia stane kyber-fyzikálnym systémom, ktorý zahŕňa technológie ako sú inteligentné siete, inteligentné domy, inteligentnej dopravy, inteligentné mestá a v neposlednom rade inteligentné vzdelávanie.

Základným cieľom predmetu je poukázať na význam a špecifiká IoT pri realizácii vyššej úrovne automatizácie, resp. analyzovať predpoklady na realizáciu tzv. inteligentných zariadení ako systémov s rozmanitými technickými, ekonomickým a ekologickými požiadavkami a v neposlednom rade požiadavkami na používateľský komfort či kvalitu života používateľov vo všetkých oblastiach života. Predmet ponúka stručný prierez oblasťou internetu vecí (IoT) a jej základnými komponentami od fyzických senzorov a aktuátorov, cez lokálnu komunikáciu a spracovanie až k službám v cloude a strojovému učeniu. Predmet rozoberá problematiku automatizácie, modelovania a navrhuje samotné využívanie hardvéru

(mikrokontroléra Arduino alebo mikropočítača Raspberry PI), ktorý môže tvoriť jednu z častí inteligentných systémov a byť tak plne súčasťou IoT. Určujúce faktory pri tvorbe koncepcie vyššej úrovne automatizácie inteligentných systémov, podobne ako pri priemyselných procesoch pritom tvoria:

- vlastnosti objektu riadenia,
- vlastnosti informačných, riadiacich a komunikačných systémov,
- ciele riadenia a optimalizácie.

Zámerom je uviesť viaceré metódy používané v oblasti IoT, zhrnúť ich základné charakteristiky, poukázať na silné a slabé stránky týchto metód. Študent získa kľúčové vedomosti z oblasti využitia Internetu vecí na monitorovanie reálneho aktuálneho problému. Predmet sa bude venovať návrhom, implementácii a testovaniam systémov, v ktorom budú použité zariadenia napr. Arduino, Raspberry Pi, Sigfox, Zigbee atď. Študent získa širokospektrálny prehľad o architektúre Internetu vecí, jeho funkčných stavebných blokoch, senzoroch, akčných členoch, softvérovom programovaní a integrovaní s fyzických svetom, lokálnom spracovaní na hranici siete, bezpečnom a efektívnom prenose dát cez rôzne sieťové protokoly, ukladaní a spracovaní dát v cloude, riadení na základe dát, ako aj podnikateľské nápady v danej oblasti. V predmete sa študent naučí kreatívne navrhnúť šikovné systémy od jedného konca k druhému pre Internet vecí a prepojiť fyzický svet so softvérovým svetom metódou rýchleho prototypovania. Predmet sa bude venovať definovaniu pravidiel pre zber údajov (frekvenciu, formát, hraničné stavy sledovaných veličín, atď.) Počas semestra bude študent súčasťou výskumného tímu, ktorého výstupom bude projekt. Výstupný projekt môže byť prototypom pre systém, ktorý bude monitorovať zvolený reálny problém z praxe.

4.3.2 Aplikačný príklad 1

V prvom aplikačnom príklade si predstavíme prípadovú štúdiu detekcie pohybu pomocou HD kamery mikropočítača Raspberry Pi. Počítačové videnie ako vedná a technologická disciplína sa zaoberá schopnosťou elektronických zariadení získať informáciu z digitálneho obrazu - "pochopiť situáciu" a na základe nej urobiť rozhodnutie či vykonať zadanú úlohu. Patrí medzi technológie využívané vo všetkých oblastiach priemyslu a výskumu.

V posledných rokoch sa rapídne zvýšila dostupnosť a presnosť technológie rozpoznávania tváre. Existuje už relatívne veľké množstvo systémov, ktoré sú schopné detekovať a rozpoznávať tváre. Niektoré z týchto systémov sú: Picasa, OpenCV, iPhoto. Picasa a iPhoto sú podobné a primárne sa používajú na detekciu tvárí na fotkách. Oproti tomu OpenCV je vytváraná komunitou a obsahuje najčastejšie používané metódy a algoritmy pre detekciu a rozpoznávanie tvárí. Treba si uvedomiť, že tento nástroj nie je klasickým systémom, ale jedná sa skôr o zbierku metód, ktoré sa môžu použiť na takéto účely. Toto nás inšpirovalo k použitiu tejto knižnice na mikropočítači Raspberry Pi. Samozrejme pre návrh komplexného funkčného systému od hardvéru až k softvéru, bolo potrebné najprv analyzovať najčastejšie používané mikropočítače a porovnať ich vlastnosti.

Vstavané počítačové videnie je veľmi praktický odbor počítačového videnia, ktorý sa zaujíma o rozvoj alebo modifikáciu algoritmov počítačového videnia, ktoré bežia na vstavaných systémoch malých mobilných počítačov. Dva hlavné dôvody pre systémy vstavaného počítačového videnia sú rozumné využívanie výpočtového výkonu a nízka spotreba batérie. Ako príklad takéhoto vstavaného systému, si môžeme priblížiť Raspberry Pi. Je to malý open-source mikropočítač, ktorý rýchlo získal obľubu medzi fanúšikmi a výskumníkmi kvôli jeho jednoduchej obsluhe, všestranných schopností a prekvapivo nízkej cene aj napriek dobrej rozširovatelnosti a kvalite (Brahmbhatt, 2013).

Raspberry Pi je mikropočítač osadený procesorom ARM použiteľný aj ako plnohodnotné PC. Ako taký iste nie je prvý, od ostatných sa však odlišuje svojou cenou. Ako operačný systém je tu možné použiť niektorú z linuxových distribúcií upravených pre ARM procesory - napríklad Debian Squeeze, Arch Linux ARM, Fedora Remix, Raspbian alebo aj Windows 10. Počítač je vyvíjaný neziskovou organizáciou Raspberry Pi Foundation, ten je mimo iného podporovaná univerzitou Cambridge a Broadcomem. Projekt si kladie za cieľ rozšírenie počítačovej vedy do škôl. Myšlienka malého a lacného počítača pre deti

prišla v roku 2006. Na školách sa učia hlavne prácu s Word, Excel, tvorbu webových stránok, chýbal však prístupný produkt, na ktorom by sa mohlo vyučovať programovanie, teda náhrada za počítače Amiga, BBC Micro, pectrum ZX a Commodore 64, na ktorých sa programovanie učili minulé generácie (Vujović a Maksimović, 2015).

Najzaujímavejší spôsob ako používať Raspberry Pi počítač, je ako platforma projektu. Vďaka svojej veľmi nízkej spotrebe môže slúžiť aj ako riadiaca jednotka inteligentného domu. Zoznam úloh, ktoré možno dosiahnuť, závisí len od predstavivosti a tvorivosti užívateľa. Okrem toho GPIO vývody, ďalšie porty a konektory umožňujú prevádzku a riadenie všetkých elektronických periférií vrátane senzorov, motorov, tlačidiel a podobne. Digitálne vstupy Raspberry Pi môžu byť použité na čítanie analógových senzorov veličín ako je teplota, vzdialenosť, úroveň osvetlenia alebo stavu senzorov pohybu. Takéto úlohy možno zrealizovať pomocou prevodníka analógového signálu na digitálny (ADC). Ostatné periférne zariadenia (CNC stroje, roboty) môžu byť pripojené k GPIO vývodom priamo na Raspberry Pi dosku alebo cez lokálnu, bezdrôtovú sieť. Je možné pridať bezdrôtové pripojenie k internetu pomocou jednoduchého USB Wi-Fi adaptéra. USB Wi-Fi adaptéry zvyčajne potrebujú veľa energie, takže je lepšie ich pripájať cez externe napájaný USB hub (Horáček, 2012).

Raspberry Pi sa dodáva ako samostatná doska s niekoľkými konektormi. Pre spustenie je potrebné nahrať distribúciu na microSD pamäťovú kartu (tá tu nahrádza pevný disk), pripojiť napájanie v podobe microUSB a prípadne ďalšie periférie. Myš a klávesnicu môžeme pripojiť do USB portov, pre pripojenie monitora je k dispozícii HDMI konektor, podporujúci aj prenos audia. Ako primárny audio vstup / výstup slúži 3,5 mm jack konektor (Balogh, Koprda, & Turcani, 2019; Krushinitskiy & Sziebig, 2013).

Keďže táto práca je zameraná na vizuálnu detekciu, pre náš systém sme potrebovali kvalitnú kameru. Tvorcovia Raspberry Pi vydali vlastnú dosku kamery pre spracovanie obrazu s aplikáciu počítačového videnia. Doska je malá a váži len 3 gramy, ale obsahuje 5 megapixelový CMOS snímač. Pre pripojenie sa používa plochý kábel s konektorom CSI (Brahmbhatt, 2013). Jedná sa o kameru s HD rozlíšením. Rozlíšenie sa dá samozrejme nastaviť, pri nižších hodnotách dokáže snímať obraz s vyšším FPS. Má 5 megapixelový objektív a jej cena sa pohybuje okolo 24€. Pomocou kamery a dostupných knižníc dokážeme použiť náš mikropočítač na vizuálnu detekciu.

Vizuálna detekcia

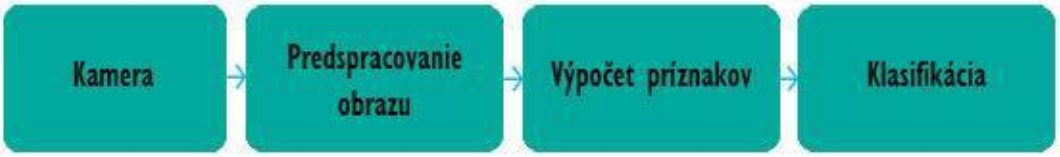
Vizuálna detekcia a rozpoznávanie objektov patrí v poslednom desaťročí medzi najväčšie výzvy počítačového videnia. Potenciálne uplatnenie systémov detekcie a rozpoznávania objektov je veľmi široké, od bezpečnostných systémov (napr. identifikácia oprávnených osôb), cez medicínske techniky (napr. detekcia tumorov), priemyselné aplikácie (napr. vizuálna inšpekcia výrobkov), robotiku (napr. navigácia robota v nedostupnom teréne) až po rozšírenú realitu a mnohé ďalšie. Úloha automaticky vyhľadať objekt sa spravidla formuluje ako úloha segmentácie objektu, prípadne iba ako detekcia opísaného štvoruholníka, alebo ako rozpoznanie samotného objektu (Šikudová a kol., 2013).

Obrázok 26. Detekcia objektov

K úlohe detekcie môžeme pristupovať na rôznych stupňoch zovšeobecnenia. Buď rozpoznávame konkrétne inštancie jednotlivých objektov, alebo celú kategóriu objektov. Vizuálna detekcia a rozpoznávanie objektov je veľmi komplexný problém. Príkladom sú:

- nehomogénne a premenlivé osvetlenie scény,
- meniace sa spektrálne charakteristiky osvetlenia a následne i farba snímaného objektu,
- objekty, ktoré sa prekrývajú,
- rôzne uhly snímania objektov kamerou,
- v prípade rozpoznávania kategórií je samozrejme veľkou výzvou i rôznorodosť objektov v rámci jednej kategórie.

Základný reťazec operácií systému detekcie rozpoznávania môžeme znázorniť pomocou diagramu na Obrázku 27.



Obrázok 27. Diagram vizuálnej detekcie

Na vstupe uvedeného reťazca je digitálna kamera, z ktorej získame signál obrazu, či už farebný alebo čiernobiely. Úlohou predspracovania obrazu je odstránenie nežiaducich a pre ďalšie spracovanie rušivých javov ako sú napr. šum v signáli, zvýraznenie informácie (napríklad kontúr objektov), ktorá je relevantná a naopak potlačenie informácie, ktorá nie je relevantná z pohľadu danej úlohy. Dobrým príkladom je filtrácia pre odstránenie šumu, hornopriepustná filtrácia pre odstránenie nehomogenity osvetlenia, prevod do iných farebných priestorov alebo morfologické spracovanie. Cieľom ďalšieho stupňa spracovania je získanie príznakov, ktoré budú dobre reprezentovať obrazovú informáciu dostatočne reprezentatívnym a diskriminatívnym spôsobom. Ak je našou úlohou napríklad detekcia ľudskej tváre v obraze, tak celý obraz budeme postupne prechádzať tzv. kĺzavým oknom. V každej jeho pozícii bude úlohou klasifikátora rozhodnúť sa medzi dvoma triedami – je alebo nie je tvár v tej časti obrazu, ktorú práve vyšetruje. Vstupom pre klasifikátora je práve vektor príznakov na základe ktorého sa môže klasifikátor rozhodnúť. V procese návrhu môžeme vygenerovať vektor príznakov podstatne väčší, než ktorý bude použitý vo výslednom klasifikátore. Vhodné (relevantné) príznaky určíme metódami selekcie príznakov. V ďalšom rozhodovacom stupni sa klasifikátor na základe vektora príznakov rozhodne o príslušnosti k triede (Šikudová et al., 2013).

Postupy rozpoznávania tvárí

Rozpoznávania tvárí je jednou z najdôležitejších schopností, ktoré používame v našom každodennom živote. Existuje niekoľko dôvodov pre rastúci záujem o automatické rozpoznávanie tváre, vrátane narastajúcimi obavami ohľadom verejnej bezpečnosti, nutnosť overenia totožnosti, a potreba analýzy a modelovacích techník v správe multimediálnych dát a digitálnej zábavy. Výskum v automatickom rozpoznávaní tvárí bola zahájená v roku 1960. V posledných rokoch došlo k významným pokrokom v tejto oblasti a bolo vyvinutých a následne nasadených množstvo systémov pre rozpoznávanie tváre a modelovanie systémov. Avšak, presné a robustné rozpoznávanie tvárí stále obsahuje rad problémov pre počítačové videnie a rozpoznávanie vzorov, obzvlášť v premenlivom prostredí (Jain a Li, 2011).

Rozpoznávanie tváre je úloha, ktorú ľudia vykonávajú rutinne a bez námahy v každodennom živote. Široká dostupnosť silných a lacných hardvérových riešení a integrovaných mikropočítačov vytvorila obrovský záujem o automatické spracovanie digitálnych obrazov v celej rade aplikácií. Ako napríklad biometrická autentizácia, zabezpečenie, interakcia človeka s počítačom, a ovládanie multimédií. Rozpoznávanie tvárí má niekoľko výhod oproti iným biometrickým metódam, ako sú odtlačky prstov a skenovanie dúhovky. Okrem toho, že je to prirodzenejšie, najdôležitejšou výhodou tváre je to, že môže byť zachytený na diaľku skrytým spôsobom. Rozpoznávanie tvárí, ako jeden z hlavných biometrických technológií, sa stáva čoraz dôležitejším vzhľadom na rýchle pokroky v digitalizácii obrazu (sledovacie kamery, kamera v mobile), dostupnosť veľkého množstva snímok tvárí na webe, a zvýšené požiadavky na vyššiu bezpečnosť. Technológia rozpoznávania tvárí značne pokročila od doby, kedy bola navrhnutá metóda Eigenface. V obmedzených situáciách, napríklad kde vieme kontrolovať osvetlenie, pozíciu, vystúpenie a mimiku automatické rozpoznávanie tvárí môže prekonať ľudské rozhodovanie, a to najmä keď databáza (galéria) obsahuje veľký počet obrázkov. Automatické rozpoznávanie tvárí stále čelí mnohým výzvam, napríklad v prípade keď sú obrazy tvárí získané v pomocou automatizovaného procesu (Jain a Li, 2011). Scenáre rozpoznávania tvárí možno rozdeliť do dvoch typov, overenie tváre a identifikácia tváre. V publikácií Face Recognition Vendor Test (FRVT) 2002, bol definovaný ďalší scenár pod menom „watch list“.

Overenie tváre je porovnávanie s metódou jedna ku jednej, ktorá porovnáva obraz tváre dotazu zo šablónou obrazu tváre, ktorej totožnosť sa žiada. Metóda pre vyhodnotenie verifikačného procesu, ktorý sa znázorňuje počtom oprávnených prístupov (prístup bol poskytnutý oprávneným osobám) a počtom neoprávnených prístupov (prístup dostali neoprávnené osoby), sa nazýva ROC krivka. Dobrý overovací systém by mal vyvážiť tieto dve hodnoty na základe prevádzkových potrieb.

Identifikácia tváre je proces zodpovedajúci porovnávaniu jedna k viacerým, ktorý porovnáva obraz tváre so všetkými šablónami obrázkov v databáze tvárí, pre stanovenie identity. Identifikácia skúšobného obrazu sa vykonáva s nájdením obrazu v databáze tvárí, ktorá má najvyššiu podobnosť s testovacím obrazom. Skúšobné predmety (obrázky) sú porovnávaný s ostatnými v databáze systému a pri každom porovnaní je k nim pridelené číslo, ktoré opisuje mieru podobnosti. Tieto takzvané skóre podobnosti sú potom zoradené v zostupnom poradí. Percento prípadov, kedy najvyššie skóre podobnosti je správny pre

všetky prípady, je označovaný ako maximálne skóre porovnávania. Ak skóre podobnosti zodpovedá testovaného subjektu, považuje sa ako správny výsledok.

Watch list je metóda, ktorá testuje každého jednotlivca či sa nachádza v databáze systému. Osoba sa porovnáva s ostatnými v databáze systému a je k nemu pridelené skóre podobnosti pre každé porovnanie. Tieto skóre sú zoradené tak, aby najvyššie skóre podobnosti bolo prvé. Ak je skóre podobnosti vyššia ako vopred stanovená prahová hodnota, spustí sa alarm. Keď sa spustí alarm, systém si myslí, že jednotlivec sa nachádza v databáze systému. Existujú dve základné hodnoty watch list aplikácií. Prvým z nich je percento prípadov, keď systém spustí alarm a správne identifikuje osobu v databáze. Tomu sa hovorí percento detekcie a identifikácie. Druhé je percento prípadov, keď systém spustí alarm pre jednotlivca, ktorý sa nenachádza v databáze. To sa nazýva percento falošných poplachov (Xiaoguang a kol., 2003).

Faktory ovplyvňujúce rozpoznávanie tvárí

Ľudská tvár má veľmi veľké množstvo potenciálnych intraindividuálných odchýliek spôsobené faktormi ako:

- Pozícia hlavy
- Osvetlenie v interiéri alebo v exteriéri
- Výraz tváre
- Prekrývanie objektov, napríklad šatky, okuliare a podobne
- Ochlpenie tváre
- Starnutie

Na druhej strane, medzi subjektmi sú variácie malé vzhľadom k podobnosti jednotlivých vystúpení. Príklady vzhľadových variantov jedného subjektu sú uvedené v práci Xiaoguang (2003).

Predspracovanie obrazu

V spracovaní obrazu segmentácia je technika používaná na oddelenie objektu záujmu od pozadia v obraze. Segmentácia v statickom obraze je často zakladaná na detekcii hrán a následnom vyhľadávaní hraníc objektu. Segmentácia v dynamickom obraze sa líši od segmentácie v statickom obraze. Na segmentáciu objektov v dynamickom obraze sa často používa metóda nazývaná odčítanie pozadia (Background Subtraction).

Odčítanie pozadia je technika, ktorá sa používa pri analýze dynamického obrazu. Jedná sa o segmentáciu pohybujúcich sa objektov v obraze. Úlohou odstránenia pozadia je určiť príslušnosť každého pixelu v obraze k pozadiu scény alebo k objektu záujmu. Pokročilé techniky odčítania pozadia dokážu identifikovať aj tieň objektu v obraze a následne ho potlačiť. Odčítaním pozadia z obrazu získame objekty záujmu. K odčítaniu pozadia dochádza medzi snímkou z videa a modelom pozadia, ktorý môže byť statický alebo adaptívny. Statický model pozadia je vhodný v prípade, že scéna je konštantne osvetlená a nedochádza v nej k náhodným javom ako napríklad pád listu zo stromu. Odčítaním pozadia so statickým modelom by bol list vždy identifikovaný ako objekt. Adaptívne modely pozadia sú výhodnejšie do exteriérových aplikácií, kde dochádza k náhodným rušivým vplyvom a zmenám osvetlenia scény. Objavenie sa náhodného statického objektu v scéne sa prejaví do modelu pozadia a za nejaký čas sa objekt stane súčasťou pozadia. Časový úsek, ktorý je potrebný na adaptáciu takejto situácie sa líši od konkrétnej aplikácie. Model pozadia musí byť adaptívny voči nasledujúcim vplyvom:

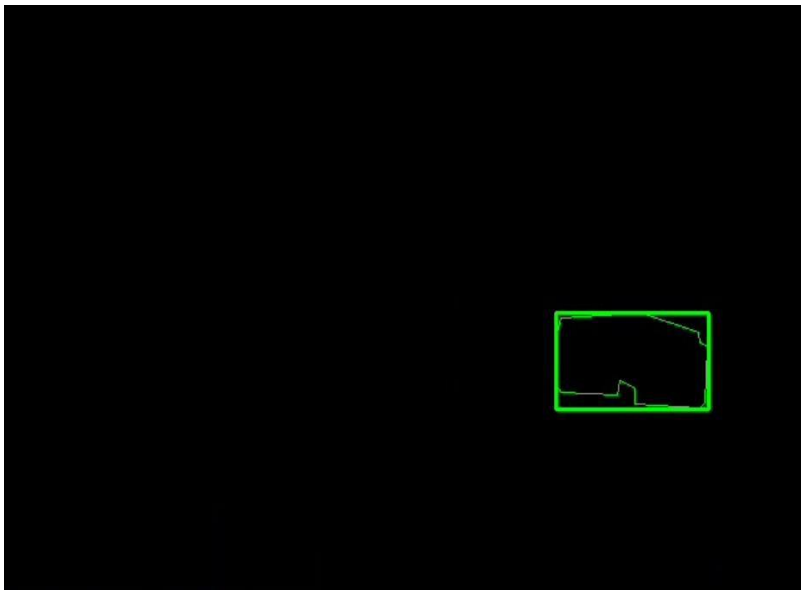
- Zmena osvetlenia
- Mechanické oscilácie kamery
- Vniknutie statického objektu do scény

Výsledkom operácie odčítanie pozadia je binárny obraz, takzvaná maska popredia (Foreground Mask). Na odstránenie pozadia sa v praxi používajú rôzne techniky (Baggio a kol., 2012).

Najjednoduchším spôsobom segmentácie dynamického obrazu je aplikovanie diferencie snímok. Dve po sebe nasledujúce snímky sa od seba odčítajú. Výsledkom sú hrany pohybujúcich sa objektov.

Obrázok 28. Detekcia pohybu objektu

Na Obrázku 28 je zobrazená snímka z kamery zobrazujúca premávku. Na snímke je vidieť vozidlo v jazdnom pruhu, ale aj zaparkované vozidlá. Po aplikácii diferencie snímok zostali vo výslednom obraze len pixely, na ktorých bola zaznamenaná medzi snímkami zmena jasu. V praxi sa teda jedná o hrany pohybujúceho sa objektu, ktoré sú zobrazené na Obrázku 29.



Obrázok 29. Hrany pohybujúceho sa objektu

Použitie tejto techniky na reálnom videozázname z kamery je sporné z dôvodu zmien osvetlenia a mechanických oscilácii kamery. Tieto rušivé vplyvy je možné čiastočne odstrániť

naprahovaním výsledného pozadia. Pixely, ktorých hodnota sa zmenila len v malej miere sú stále označené ako pozadie. Pixely presahujúce túto hodnotu sú označené za objekty (Pornpanomchai a kol., 2008).

OpenCV

OpenCV (Open-source Computer Vision, opencv.org) je švajčiarsky nožík počítačového videnia. Má celý rad modulov, pomocou ktorých si vieme poradiť s mnohými problémami počítačového videnia. Ale možno najužitočnejšia časť OpenCV je jeho architektúra a správa pamäte. To nám poskytuje rámec, v ktorom môžeme pracovať s obrázkami a videom s akýmkoľvek spôsobom. S použitím algoritmov OpenCV alebo bez nich, bez toho aby sme sa museli starať o pridelenie a zrušenie pridelenia pamäte pre obrázky. OpenCV bol oficiálne predstavený ako výskumný projekt v rámci „Intel Research to advance technologies in CPU-intensive applications“. Ciele tohto projektu boli uvedené ako:

- Pokročilý výskum počítačového videnia tým, že poskytuje nielen otvorený, ale aj optimalizovaný kód pre základnú infraštruktúru počítačového videnia.
- Šíriť vedomosti počítačového videnia tým, že poskytuje spoločnú infraštruktúru, ktorú zostavia vývojári, takže kód bude ľahšie čitateľné a prenositeľné.
- Pokročilé aplikácie na báze počítačového videnia, ktoré budú prenositeľné,
- Optimalizovaný kód, ktorý bude k dispozícii zadarmo, s licenciou, ktorá nevyžaduje aby výsledná aplikácia bola open source alebo zadarmo.

Prvá alfa verzia OpenCV bola predstavená pre verejnosť na konferencii „IEEE Conference on Computer Vision and Pattern Recognition“ v roku 2000. V súčasnej dobe OpenCV je vo vlastníctve neziskovej nadácie s názvom „OpenCV.org“.

Vstavané moduly OpenCV sú výkonné a dostatočne univerzálne na riešenie väčšinu problémov počítačového videnia, pre ktoré sú k dispozícii dobre zavedené riešenie. Môžeme orezať obrázky, zvýšiť ich jas, ostrosť a kontrast, odhaliť tvary v nich, detegovať pohybujúce sa objekty vo videu, rozpoznať známe objekty, odhadnúť pohyb robota zo spätnej väzby kamery, používať stereo kamery a získať 3D pohľad na svet. Ak však chceme vyvinúť algoritmus pre počítačové videnie, pre ktoré tieto moduly samy o sebe nie sú úplne postačujúce, OpenCV nám stále veľa pomôže svojou architektúrou, manažovaním pamäte a podporou GPU. Vlastné algoritmy, ktoré pracujú spolu s vysoko optimalizovanými modulmi OpenCV môžu byť veľmi efektívne. Hlavný aspekt OpenCV modulov, ktorú

je potrebné zdôrazniť je, že sú vysoko optimalizované. Sú určené pre použitie v reálnom čase a navrhnuté tak, aby pracovali veľmi rýchlo na rôznych počítačových platformách od MacBookov cez osobné počítače s operačným systémom Windows až po mikropočítače s operačným systémom Linux (Brahmbhatt, 2013).

Metódy rozpoznávania tvárí v OpenCV

Algoritmy na rozpoznávanie tváre, ktoré sú dostupné v knižnici OpenCV sú nasledujúce:

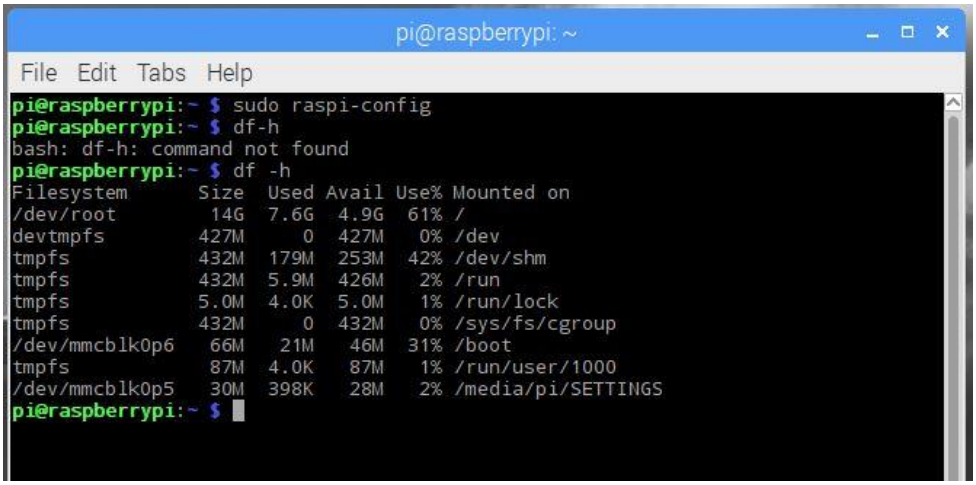
- FaceRecognizer.Eigenfaces: Eigenfaces, označovaný aj ako PCA, najprv to použili Turk a Pentland v roku 1991.
- FaceRecognizer.Fisherfaces: Fisherfaces, označovaný aj ako LDA, vynašiel to Belhumeur, Hespanha a Kriegman v roku 1997.
- FaceRecognizer.LBPH: Local Binary Pattern Histograms, vynájdený trojicou Ahonen, Hadid a Pietikäinen v roku 2004 (Baggio a kol, 2012).

Rozhodli sme sa použiť metódu Fisherfaces preto, lebo je založený na algoritme LDA (Rashmi a Namrata, 2015). Vo svojom článku pri porovnávaní rôznych algoritmov sme dosiahli s algoritmom LDA až 95.3%-nú úspešnosť a pritom čas, ktorý potrebuje na rozpoznávanie je priemerný o proti ostatným algoritmom. Metóda Fisherfaces učí sa z transformačnej matice triedy, takže nezaznamenáva intenzitu osvetlenia ako metóda Eigenfaces. Diskriminačná analýza nájde tvárové rysy k potrebné k porovnávaniu osôb. Je dôležité spomenúť, že výkonnosť Fisherfaces v značnej miere ovplyvňujú aj vstupné dáta. Prakticky povedané, ak učenie Fisherfaces prebehne na dobre osvetlených obrázkoch tak pri pokusoch o rozpoznaní tvárí so zlým osvetlením sa pravdepodobne bude vracať vyšší počet chybných výsledkov.

Takéto riešenia sa používajú v rôznych priemyselných aplikáciách, v projektoch pre výučbu, ale čoraz viac si nájdu uplatnenie v inteligentných domácnostiach. Rozpoznávanie tvárí a detekciu pohybu sa môže použiť pri stavbe systému do inteligentných riešení v domácnosti, ktorú môžeme použiť či už ako samostatnú funkčnú jednotku alebo ako prvok väčšieho systému prepojenou pomocou technológie Internet of Things (IoT).

Inštalácia OpenCV 3 na Raspberry Pi 3

Pomocou príkazu *sudo raspi-config* otvoríme nastavenia a zvolíme možnosť Expand File System (rozšíriť súborový systém) a stlačíme Enter na klávesnici a potom reštartujeme Raspberry Pi. Po reštarte bude súborový systém rozšírený čo znamená že bude dostupný na všetkých miestach na SD karte.



Obrázok 30. Rozšírenie súborového systému

Inštalácia OpenCV 3 trvá približne 90 minút. Prvým krokom je aktualizovať všetky existujúce balíčky s príkazom *sudo apt-get update* a *sudo apt-get upgrade*. Potom musíme nainštalovať niektoré vývojárske nástroje, vrátane cmake, čo nám pomáha konfigurovať OpenCV. Nainštalujeme pomocou príkazu: *sudo apt-get install build-essential cmake pkg-config*. Ďalej musíme nainštalovať nejaké I/O balíčky, ktoré nám umožnia zaviesť rôzne formáty obrazových súborov z disku s príkazom: *sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev*. Príklady takýchto formátov súborov zahŕňajú JPEG, PNG, TIFF, atď. Musíme tiež nainštalovať I/O balíčkov pre video formátov (*sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev*). Tieto knižnice nám umožňujú čítať rôzne formáty video súborov z disku, ako aj pracovať priamo s video streamom. Knižnica OpenCV je dodávaná modulom s názvom highgui, ktorá sa používa na zobrazenie obrázkov na naše obrazovky a budovať základné GUI. Za účelom zostavenia modulu highgui, musíme nainštalovať knižnicu GTK (*sudo apt-get install libgtk2.0-dev*). Mnoho operácií vnútri OpenCV (maticové operácie) možno ďalej optimalizovať tým, že nainštalujeme niekoľko ďalších modulov (*sudo apt-get install libatlas-base-dev gfortran*). Tieto optimalizácie sú dôležité najmä pre zariadenie s obmedzenými zdrojmi, ako napríklad Raspberry Pi. Potom je potrebné

nainštalovať aj Python 2.7 a Python 3 pomocou príkazu: *sudo apt-get install python2.7-dev python3-dev*.

Teraz, keď máme všetko nainštalované môžeme stiahnuť OpenCV 3.1.0 od oficiálneho OpenCV úložiska pomocou nasledujúcich príkazov:

- `cd ~`
- `wget-Oopencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip`
- `unzip opencv.zip`
- `wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip`
- `unzip opencv_contrib.zip`

Najprv musíme nainštalovať PIP (správcu balíčkov Python)a až potom budeme môcť začať zostavovať OpenCV na našom Raspberry Pi 3,

- `wget https://bootstrap.pypa.io/get-pip.py`
- `sudo python get-pip.py`

Potom nainštalujeme virtualenv a virtualenvwrapper balíčky. Keď programujeme v Pythone tak je veľmi vhodné používať virtuálne prostredie. Virtuálne prostredie je špeciálny nástroj používaný k udržaniu závislostí rôznych projektov v rôznych miestach tým, že vytvorí izolované, nezávislé Python prostredie pre každú z nich. Stručne povedané, rieši "Projekt X závisí na verzii 1.x, ale Projekt Y potrebuje 4.x" dilemu. Nainštalujeme pomocou nasledujúcich príkazov:

- `sudo pip install virtualenv virtualenvwrapper`
- `sudo rm -rf ~/.cache/pip`

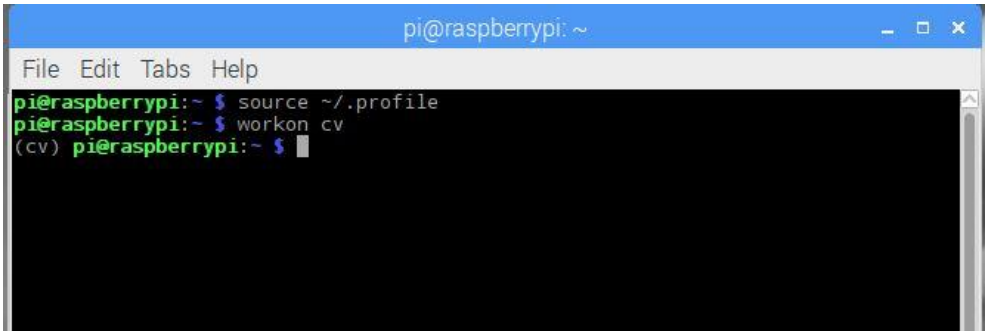
Teraz, keď sú nainštalované oba virtualenv a virtualenvwrapper, musíme aktualizovať ~ / .profile súbor, aby obsahoval nasledujúce riadky na koniec súboru:

- # virtualenv and virtualenvwrapper
- export WORKON_HOME=$HOME/.virtualenvs
- source /usr/local/bin/virtualenvwrapper.sh

Ďalej vytvoríme Python virtuálne prostredie, ktorý budeme používať pre vývoj počítačového videnia:

- `mkvirtualenv cv -p python2`
- `mkvirtualenv cv -p python3`

Ak chceme používať virtuálne prostredie, tak musím použiť príkaz workon.



Obrázok 31. Virtuálne prostredie

Potom je potrebné nainštalovať Python balíček NumPy, ktoré sa používa pre numerické spracovanie: „`pip install numpy`". Teraz už môžeme nainštalovať OpenCV. Najprv treba skontrolovať či sme vo virtuálnom prostredí, a potom nainštalujeme pomocou nasledujúcich príkazov:

- `cd ~/opencv-3.1.0/`
- `mkdir build`
- `cd build`
- `cmake -D CMAKE_BUILD_TYPE=RELEASE \`
- `-D CMAKE_INSTALL_PREFIX=/usr/local \`
- `-D INSTALL_PYTHON_EXAMPLES=ON \`
- `-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \`
- `-D BUILD_EXAMPLES=ON ..`

Keď to už máme hotové, je potrebné kompilovať OpenCV pomocou príkazu: `make -j4`. Príkaz –j4 znamená počet jadier ktoré pracujú pri kompilovaní OpenCV 3.

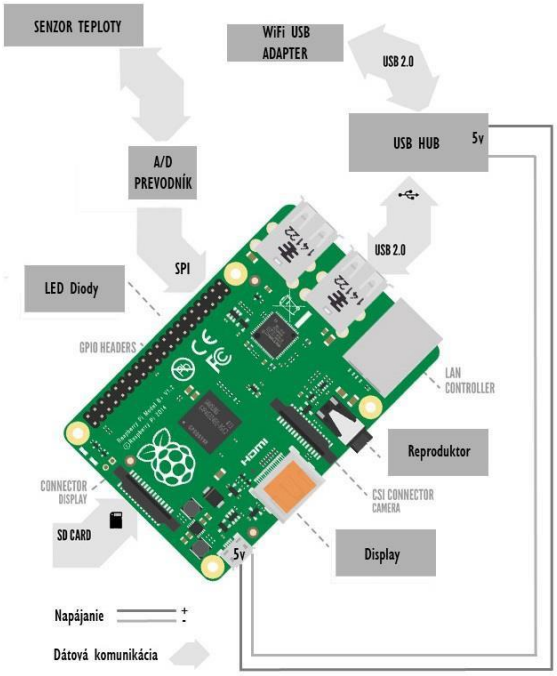
Raspberry Pi 3 má štyri jadrá, preto sme dali hodnotu 4, tým pádom Raspberry Pi pracuje rýchlejšie. Teraz už môžeme písať programy, ktoré používajú knižnicu OpenCV 3 (Rosebrock, 2016).

Metodika riešenia

Postup riešenia práce môžeme rozdeliť na dve hlavné časti, zostrojenie hardvéru s potrebnou konfiguráciou a vytvorenie ovládacieho programu. Hlavná časť kamery je doska Raspberry Pi a jeho modul HD kamery. Vytvorená aplikácia rieši image processing.

Zostrojenie hardvérových častí

Náš systém sa skladá zo šiestich hlavných hardvérových častí. Hlavná jednotka systému je mikropočítač Raspberry Pi . V našom systéme boli použité nasledujúce komponenty: Základná doska Raspberry Pi, Modul kamery, USB HUB, Wi-Fi adaptér, A/D menič, Senzor teploty, Reproduktor, Napájací adaptér, Signalizačné LED diódy. Schéma zapojenia je znázornená na Obrázku 32.



Obrázok 32. Schéma zapojenia hardvérových súčiastok

Hlavnou jednotkou nášho riešenia je bezpochybne mikropočítač Raspberry Pi v spojení s modulom HD kamery. Pre Raspberry Pi sme potrebovali vybrať operačný systém

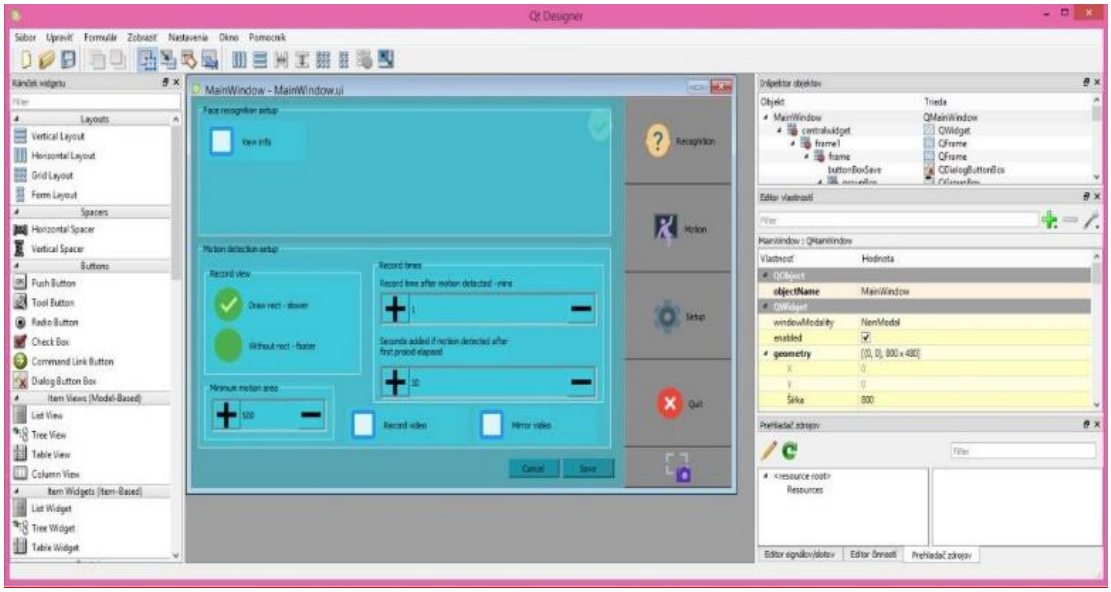
ešte pred samotným konfigurovaním. Je potrebné vytvoriť systém, ktorý čo najmenej zaťažuje procesor a celkovo počítač, preto že spracovanie obrazu v reálnom čase je sám o sebe náročný na výkon. Voľba padla teda na najviac používaný OS Raspbian. Tento operačný systém, je špeciálne upravená a optimalizovaná verzia Debianu pre Raspberry Pi.

Návrh softvéru pre Raspberry Pi

Existuje mnoho jazykov, ktoré sa môžu použiť pri vývoji pre mikropočítače. Najrozšírenejšie sú asi Python a C++. Pre Raspberry Pi je odporúčaný Python aj kvôli jednoduchosti tohto jazyka. Jedná sa o scriptovací jazyk, na ktorých sa aj vyučuje programovanie pre tento mikropočítač, keďže sa skladá z anglických slov, ktorým deti ľahšie porozumejú. Naša aplikácia je vytvorená v programovom jazyku C++.

1) GUI program

Program pre návrh grafického rozhrania sa volá Qt Designer, a prostredie v ktorom sa dajú navrhovať grafické časti programu vidíme na Obrázku 33.



Obrázok 33. Vývojové prostredie Qt Designer

Softvér nám ponúka základné prvky ktoré môžeme osadiť do rôznych vrstiev. Vieme si naimportovať obrázky a vytvárať si vlastné grafické prvky. Pridávať vlastné prvky si vieme samozrejme aj zo samotného kódu, ako aj modifikovať všetky navrhnuté časti priamo z kódu. Program po uložení návrhu vytvorí xml súbor s príponou *.ui* a jeden súbor pre zdroje ako sú obrázky, s príponou *.qrc*. Súbor s vlastnosťami a nastaveniami projektu Qt má príponu *.pro*. Ostatnú časť projektu tvoria klasické súbory s funkciami a hlavičkové súbory.

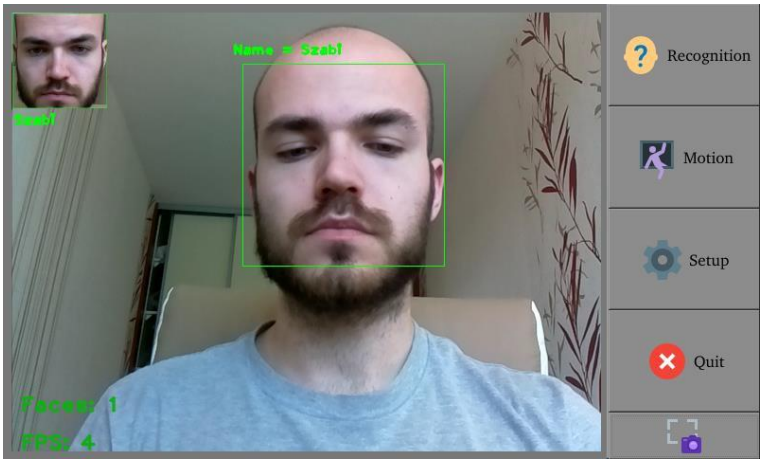
Pre samotné GUI sme zvolili jednoduchý dizajn, po pravom okraji okna máme hlavné ovládacie prvky. Prvé tlačidlo slúži pre spustenie detekcie tvárí. Ďalšie tlačidlo pod ním slúži pre aktiváciu detekcie pohybu, tlačidlo Setup nám otvorí polopriesvitnú vrstvu nad aktuálne zvoleným režimom. Nastavenia máme rozdelené do dvoch hlavných častí kde si môžeme nastaviť jednotlivé parametre a funkcie oboch režimov. Samozrejme tu máme tlačidlo na vypnutie programu a pre posledné tlačidlo si vieme zvoliť dva režimy, buď nám spraví screenshot celej obrazovky alebo fotku aktuálne zobrazenej scény z kamery.

2) Funkcie programu

Vytvorený program má viacero funkcií, detekciu osôb a detekciu pohybu. Detekcia pohybu spočíva v tom, že keď program zaznamená pohyb pomocou kamery, označí nám na obrazovke oblasť, kde pohyb nastal. Označuje to vykreslením zeleného štvorca okolo oblasti. Jej veľkosť závisí od veľkosti oblasti kde bol zaznamenaný pohyb. Samozrejme tu máme možnosť nahrávania videa v prípade, že bol nejaký pohyb zaznamenaný. Táto možnosť sa dá zapnúť z nastavení programu, kde môžeme si nastaviť aj parametre nahrávania. Nastaviť sa dá dĺžka nahrávania po rozpoznaní, po uplynutí tejto doby sa program rozhoduje či má nahrávanie pokračovať alebo zastaviť. Program sa rozhodne predĺžiť nahrávku v prípade, že stále zaznamenáva pohyb. Dĺžku predĺženia si vieme tiež nastaviť. V ľavej spodnej časti videa sa nachádza takzvaný timestamp, kde sa zobrazuje dátum a čas, a v pravej časti sa počas nahrávania zobrazuje ikona ktorá ukazuje že aktuálny obraz sa nahráva. Medzi nastaveniami nájdeme aj možnosť zrkadlenia obrazu, aby sme v prípade že kameru máme natočenú smerom na pozorovateľa, videli obraz reálne ako aj v zrkadle. Nahrávanie videa sa dá tiež vypnúť ako aj označovanie pohybu počas nahrávania. Keď túto možnosť vypneme môžeme dosiahnuť vyššie hodnoty FPS, lebo program nerozpoznáva pohyb počas celej nahrávky iba keď vyprší vopred nastavený čas. Posledná možnosť v nastaveniach pre túto časť programu je možnosť meniť minimálnu plochu pohybu, ktorý sa deteguje. Po ukončení nahrávky program skontroluje či má Raspberry Pi aktívne pripojenie na internet, v prípade že áno tak nahrá uložené video na cloudové úložisko Dropbox. Detekciu pohybu môžeme vidieť na Obrázku 34.

Obrázok 34. Detekcia pohybu

Ďalšia funkcia riešenia je detekcia tvárí a rozpoznávanie osôb. Aj pre túto časť si vieme nastaviť zrkadlenie obrazu. Okrem toho tu máme možnosť zobraziť informácie na obrazovke, konkrétne aktuálny FPS a počet detekovaných tvárí. Program rozpoznáva všetky tváre ktoré deketoval, ale berie do úvahy len najväčšiu z nich. Logicky ten, kto zaberá väčšiu plochu na obraze, stojí bližšie ku kamere. Po rozpoznaní osoby v prípade, že osoba sa nachádza v databáze program zobrazí nasnímanú tvár v ľavom hornom rohu. Pod obrázkom sa zobrazí meno osoby a Raspberry Pi ho pozdrav. Na obrazovke počas snímania každá tvár sa označuje štvorcom, obyčajne červenej farby ktoré neboli rozpoznané. Tváre ktoré boli rozpoznané sa označujú rôznymi farbami. Detekciu tváre vidíme na Obrázku 35.



Obrázok 35. Detekcia tváre

3) Cloudové úložisko – Dropbox

Screenshoty a videá ktoré sa uložia na SD kartu mikropočítača, program automaticky nahrá do vopred definovaného priečinka. Prvým krokom je vytvoriť účet na portály www.dropbox.com, a následne si musíme vytvoriť aplikáciu, aby sme mohli súbory nahrávať do nášho úložiska. Aplikácie vieme vytvárať na stránke www.dropbox.com/developers/apps, ako prvé si vyberieme API. V našom prípade klasické Dropbox API, ďalej si zvolíme prístupové práva programu, aby sme mohli nahrávať súbory len do jedného priečinka. Následne už stačí zadať názov nášho programu, po tomto kroku už máme vytvorenú aplikáciu. Posledný krok je vygenerovať takzvaný acess token pre pristupovanie do aplikácie bez dodatočnej autorizácie. Pomocou tohto tokenu náš program dokáže nahrávať súbory do priečinka ktorý bol vytvorený pre našu Dropbox aplikáciu. V zdrojovom kóde nášho programu len zostavíme jednoduchý script, ktorý následne zavoláme pomocou funkcie *system()*. V zdrojovom kóde to vyzerá nasledovne:

```
void OOpenCVWidget::uploadVideo(string pathAndName)
{
	if (system("ping -c1 -s1 www.google.com"))
	{
		cout << "There is no internet connection   \n";
	}
	else
	{
		std::string str;
		str.append("curl -H ");
		str.append(1u, '"');
		str.append("Authorization: Bearer rBoj...bgTGXcub9s_UbkHa6UR-7DV");
		str.append(1u, '"');
		str.append(" https://api-content.dropbox.com/1/files_put/auto/ -T ");
		str.append(pathAndName);
		system(str.c_str());
		cout << str.c_str();
	}
}
```

Obrázok 36. Zdrojový kód programu

Výsledky riešenia a ich zhodnotenie

Výsledkom práce je funkčný systém pre rozpoznávanie osôb a na detekciu pohybu. Nami vytvorené riešenie je kompletne zostrojené a vhodné na použitie napríklad v inteligentných domoch. Systém pracuje spoľahlivo a počas testov sme nezaznamenali žiadne zlyhanie softvéru ani hardvéru. Počas testov sa preukázalo že detekcia pohybu zaznamená všetky pohyby a v prípade označovania oblastí dosahuje 4-5 FPS, bez označenia

až 8 FPS. Keďže detekcia osôb je náročnejší proces aj dosahované výsledky pri zobrazovaní sú nižšie, okolo 4-5 FPS alebo menej. V cieľoch sme uviedli aj to aby po detekcií sa vykonali isté akcie. Pri detekcií pohybu program automaticky začne ukladať nahrávku a po ukončení nahrá na cloudové úložisko. V prípade detekcie tvárí, program pozdraví rozpoznanú osobu. Na čelnej strane LED diódy signalizujú či aktuálne je rozpoznaná nejaká osoba alebo nie. Pre názornú ukážku vstupov sme použili senzor teploty ktorý je zobrazený na obrazovke. Medzi výsledkami by sme rád spomenuli cenu tohto open source riešenia, kde celková cena nášho zariadenia je iba 160 €. Našim cieľom bol vytvoriť sofistikovaný systém ktorý je ľahko rozšíriteľný pri nízkych nákladoch na súčiastky, čo sme aj splnili. Počas vývoja a testovania tohto riešenia sme si všimli aj slabé stránky nami navrhnutého systému. Podobný systém by sme neodporúčali použiť pre autentifikáciu osôb pri vstupe do budovy, alebo v podobných situáciách. Náš systém slúži hlavne pre zvýšenie komfortu pri inteligentných riešeniach kde určitá hladina chybnej detekcie osôb je prípustné. V prípade povolenia vstupu do budovy sú nároky na presnosť detekcie oveľa vyššie, a to tieto systémy v tejto podobe nedokážu splniť. Pre takýto typ autentifikácie je už potrebná presná detekcia kde nemôžeme dovoliť aby systém sa dal oklamať jednoduchou fotkou osoby. Pre riešenie tohto problému navrhujeme použitie 3D detekciu tváre, kde tvár osoby sníma viacero kamier, obraz z kamier sa spracuje a systém vytvorí 3D model tváre osoby. Tento spôsob autentifikácie už dokáže rozpoznať rozdiel medzi fotkou a skutočnou tvárou.

Záver

Podarilo sa nám uskutočniť rozpoznávanie tvárí aj detekciu pohybu pomocou nášho navrhnutého systému. Podobné systémy sa používajú napríklad v inteligentných domoch. Naše riešenie je cenovo priaznivejšia alternatíva pre komerčne dostupné systémy. Je ľahko prispôsobiteľný osobným požiadavkám používateľov. Vytvorený systém dosahuje dobré výsledky a dokazuje možnosť využitia balíka OpenCV aj na menej výkonných zariadeniach. Prínosom tejto práce bolo poukázať na možnosti vizuálnej detekcie s využitím mikropočítača Raspberry Pi. Prínosom tejto práce bolo aj to že sme poukázali na možnosti využitia počítačového videnia na lacnejších mikropočítačoch. Preukázali sme aj to, že aj pri menej výkonných systémoch sa dá používať knižnica OpenCV pri vhodne zvolených kritériách optimalizácie. Plán do budúcna čo sa týka softvéru je hlavne rozšírenie možností nastavenia programu napríklad o možnosť pridávania osôb a profilov priamo z obrazovky, alebo cez webové rozhranie. Bolo by vhodné vyskúšať vytvorený softvér na viacerých verziách

mikropočítača pre porovnanie výkonu a možností optimalizovania vizuálnej detekcie. Pomocou zabudovaného Wi-Fi modulu si môžeme zariadenie pripojiť do domácej siete alebo ho prepojiť s ostatnými zariadeniami pomocou IoT. Pri inteligentných domoch by mohol slúžiť aj ako webserver cez ktorý by sme vedeli ovládať všetky zariadenia ktoré sú zapojené do siete. Veríme že riešenie bude príkladom aj pre ostatných, a počet podobných riešení bude rásť.

4.3.3 Aplikačný príklad 2

V druhom aplikačnom príklade prezentujeme prípadovú štúdiu monitorovania priestorovej aktivity využitím systému Raspberry PI a RFID.

Priestorová aktivita drobných živočíchov si v dnešnej dobe kladie vysoké nároky na presnosť. Zastarané metódy fyzického monitoringu prinášajú do pozorovania vysokú chybovosť, ktorá môže zmeniť výsledok vedeckej práce. V dnešnej dobe viac ako 80 % všetkých testovaných cicavcov sú myši. Nové technológie dovoľujú automatizáciu procesu monitorovania a znižujú nepresnosti merania. Technológie ako GPS, alebo Argos ponúkajú nové možnosti v odvetví monitorovania pohybu zvierat. Tieto dáta je možne zberať nepretržite a s minimálnou odchýlkou nameraných dát. Nevýhodou týchto technológií je však ich vysoká cena na realizáciu. V prípade implementácie na monitoring pohybu hlodavcov sa takisto stretávame s problémom rozmerov sledovacieho zariadenia. Vysielače týchto technológií sú príliš veľké na umiestnenie na telo hlodavca.

Jedným zo systémov, ktorý si nachádza uplatnenie v monitoringu hlodavcov je technológia RFID. Technológia RFID ponúka praktické riešenie monitoringu hlodavcov pri priestorovej a semipriestorovej aktivite. RFID transpondéry o veľkosti 11 x 2 milimetre sú dostatočne malé na implantáciu do tela hlodavcov. V prípade použitia pasívnych tagov táto technológia takisto zaručuje časovo neobmedzené označkovanie. Pasívne tagy nepoužívajú integrovanú batériu, ale získavajú energiu pomocou indukcie prúdu z poľa antény čítacieho RFID modulu. Životnosť takýchto tagov vysoko presahuje očakávanú dĺžku života pozorovaných hlodavcov. Dôležitou súčasťou autonómneho merania pohybu v teréne je prenos dát. S rozvojom internetu vecí a pokrokom technológií získavame nové možnosti zberu dát z okolia. Takmer úplné pokrytie sieťou internet umožňuje neustálu komunikáciu

s autonómnymi stanicami. Táto neustála komunikácia dovoľuje výskumníkom sústrediť sa viac na výskum ako na zber a kategorizovanie nazberaných dát.

Neoddeliteľnou súčasťou vzdialeného autonómneho zberu dát je optimalizácia výkonu a spotreby monitorovacieho zariadenia. Vývoj v oblasti batérii a solárnych panelov umožňuje nepretržitú prácu meracích zariadení i pri malých energetických optimalizáciách. Optimalizácia spotreby energie však stále zostáva dôležitou otázkou pre ochranu životného prostredia a minimalizáciu celkovej ceny navrhovaného riešenia.

Návrh optimálneho monitorovacieho systému drobných hlodavcov si vyžaduje analyzovať metódy používané v praxi. Technologický vývoj za posledné roky priniesol množstvo nových technológií na monitorovanie zvierat vo voľnej prírode. Medzi hlavné kritéria moderných monitorovacích metód patrí autonómnosť a schopnosť práce bez potreby zásahu človeka, spotreba a výdrž batérie, vzdialenosť dosahu a spôsob prenosu nazbieraných dát vedcom na spracovanie.

Technológia RFID

Za posledných 30 rokov si technológia RFID (Rádio Frekvenčná identifikácia) našla uplatnenie v monitorovaní, medicíne, výrobe, zásobovaní a mnohých iných odvetviach. Systém RFID je zložený z dvoch komponentov: RFID transpondéru a čítacieho modulu. Zloženie RFID transpondéru sa mení v závislosti na jeho type a použitý. Štandardne však obsahuje anténu slúžiacu na komunikáciu a mikročip s uloženými dátami. Čítací modul komunikuje s RFID čipom pomocou rádiových vĺn na frekvencií v závislosti od použitého RFID tagu. RFID čipy sa dajú rozdeliť podľa internej štruktúry na aktívne, poloaktívne a pasívne čipy. RFID tagy je takisto možné rozdeliť podľa ich operačnej frekvencie na nízko-frekvenčné, vysoko-frekvenčné, ultra- vysoko-frekvenčné a mikrovlnne.

RFID	EPC trieda	Pamäť	Frekvencia	Bity slova	Zdroj energie	Vzdialenosť čítania (m)
Pasívne	0	ROM	125 KHz 134 MHz	64	EMF	0,04 - 3
Aktívne	4	ROM	13,85 MHz	64	Batéria	3 -10
Pasívne Programovateľné	1	EEPROM	125 KHz 132 MHz	96 128	EMF	0,04 - 3
Aktívne Programovateľné	2, 3, 4,	EEPROM	138 KHz 13,85 MHz	>128	Batéria	3 - 10
Dátový tag	2, 3, 4	CMOS FLASH	13,85 MHz 985 MHz	>128	Batéria	3 -10
RF lokátor	-	EEPROM CMOS	303 MHz 2.4/5.8 GHZ	64	Batéria	1 - 100

Obrázok 37. Typy RFID tagov podľa použitia

Aktívne RFID transpondéry využívajú na svoj chod vlastný zdroj elektrickej energie v podobe integrovanej batérie. Vďaka použitiu internej batérie, aktívne tagy dokážu ukladať väčšie množstvo dát ako pasívne RFID čipy. Aktívny RFID čip nepotrebuje pre svoj chod komunikovať s čítacím modulom. Môže pracovať ako maják, ktorý v určitých časových intervaloch alebo pri nastavení určitej udalostí vyšle dáta do svojho okolia. Tento vyslaný signál môže biť zachytený ďalšími aktívnymi tag-mi bez nutnosti prechodu signálu cez čítací modul. Aktívny RFID čip môže obsahovať senzor a dokáže lokálne spracovať dáta pred ich odoslaním. Niektoré aktívne tagy dokážu vysielať na dvoch frekvenciách pričom nižšiu frekvenciu využívajú len na prijímanie dát. Najväčšou výhodou Aktívneho tagu je jeho dosah, ktorý môže byť až do vzdialenosti 300 metrov. Nevýhodou aktívnych RFID tagov je ich cena, veľkosť a obmedzená životnosť ktorá sa pohybuje v rozsahu 10 rokov.

Poloaktívne RFID čipy obsahujú batériu bez aktívneho vysielača. Poloaktívny RFID čip nikdy neinicializuje komunikáciu, len odpovedá na odozvy. Interná batéria dovoľuje omnoho väčšiu vzdialenosť komunikácie, väčšie množstvo uložených dát a možnosť pridania senzorov. Vďaka prítomnosti batérie takisto dokáže odpovedať na slabí signál čítacieho modulu. Poloaktívny RFID tag majú väčšiu cenu a rozmery ako pasívne RFID čipy. Priemerná dĺžka života batérie je v rozsahu 2 až 10 rokov v závislosti na čase použitia a frekvencií antény.

Pasívne RFID čipy neobsahujú batériu a vysielač. Elektrickú energiu získavajú z rádiových vĺn čítacieho modulu. Táto energia je dostatočnou len na odoslanie alebo zápis správy o minimálnej veľkosti. Vďaka neprítomnosti batérie, pasívne RFID tagy dokážu komunikovať len na malé vzdialenosti v rozmedzí niekoľkých centimetrov. Táto krátka vzdialenosť komunikácie môže biť v niektorých prípadoch výhodou. Príkladom takéhoto využitia sú napríklad kreditné karty. Pasívne RFID čipy majú malú hmotnosť a rozmery. Ich životnosť sa pohybuje rádovo v desiatkach rokov .

	Frekvencia	Dáta a ich rýchlosť	Dosah	Anténa
Nízko frekvenčné	125kHz 134kHz	Malá rýchlosť	do 1,5m	Indukčná cievka
Vysoko frekvenčné	13,56MHz	Stredná rýchlosť	0,3 až 1m	Indukčná cievka
Veľmi vysoko frekvenčné	433MHz	Vysoká rýchlosť	do 30m	Špecifický dizajn
Ultra vysoko frekvenčné	860MHz 960Mhz	Vysoká rýchlosť	1 až 60m	Jeden-dva dipólová
Mikrovlnné	2,45GHz 5,4GHz	Vysoká rýchlosť	0,3 až 100m	Jeden dipól

Obrázok 38. Rozdelenie RFID čipov podľa ich frekvencie

Nízko frekvenčné RFID transpondéry majú vyznačenú hranicu frekvencií od 30 kHz do 300 kHz. V praxi sa však používajú len frekvencie o veľkosti 125 kHz a 134.2KHz. Táto hranica bola určená pre monitorovanie zvierat od roku 1979. Všetky nízko frekvenčné tagy sú pasívnymi tagmi z čoho vyplýva, že majú dosah len niekoľko centimetrov. Nízko frekvenčné tagy nemajú žiadne proti kolízne opatrenie, čo znamená že nie je možné čítať viacero tagov v ten istý čas. Vďaka svojej komplexnejšej anténe je možné tieto tagy čítať i cez pevné prekážky.

Vysoko frekvenčné transpondéry majú vyznačenú hranicu frekvencií od 30 do 300 MHz. V praxi sa využíva len frekvencia 13,56MHz. Vysoko frekvenčné tagy patria do kategórie pasívnych tagov. Ich anténa predstavuje len ~5 otočení cievky čo zapríčiňuje ich slabý prienik cez prekážky. Vysokofrekvenčné tagy ponúkajú vyšší dátový transfer ako nízkofrekvenčné tagy. Vysokofrekvenčné tagy je možne čítať cez pevné prekážky. Výskyt kovu v týchto prekážkach však môže zapríčiniť nespoľahlivé čítanie.

Ultra vysoko frekvenčné transpondéry majú vyhraničené frekvencie medzi hodnotami 300 až 1000MHz. Aplikácie však využívajú len frekvencie medzi 860 a 960 MHz a frekvenciu 433 MHz. Frekvencia 433MHz je používaná aktívnymi tagmi a frekvencie medzi 860 až 960 MHz sú zvyčajne pridelené pasívnym a poloaktívnym tagom. Všetky ultra vysoko frekvenčné transpondéry majú ochranu proti kolízií vďaka čomu je možné čítať viacero tagov v ten istý čas. Ultra vysoko frekvenčné transpondéry sú jednoducho vyrobiteľné a rozmerovo malé, pričom celková hrúbka tagu môže biť menšia ako 100 mM. Tieto transpondéry nie je možné čítať v prípade, ak sú pripevnené na objekte obsahujúcom vodu alebo na tele zvieraťa, pretože voda absorbuje ultra vysoko frekvenčné vlny.

Mikrovlnné transpondéry majú vymedzenú frekvenciu v rozmedzí medzi 1 až 1GHz. V praxi sú využívané len frekvencie 2.45 a 5.8GHz z čoho takmer všetky tagy používajú frekvenciu 2.45 GHz. Do tejto kategória patria aktívne a polo aktívne transpondéry. Poloaktívne mikrovlnné tagy sú zriedka používané. Ich najväčším používateľom na svete je Japonsko.

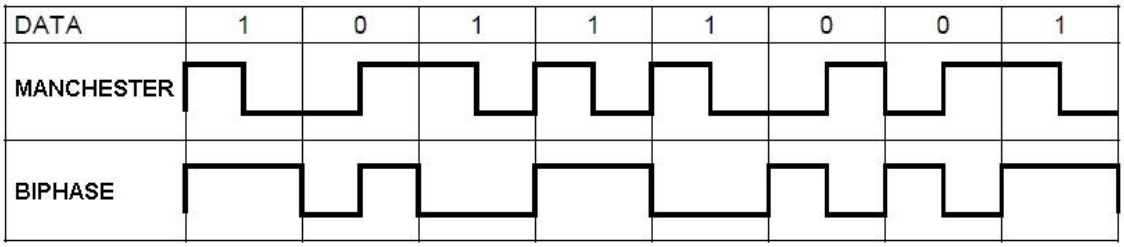
Obrázok 39. Podkožný pasívny zvierací transpondér

Zvieracie identifikačné protokoly RFID čipov

HDX a FDX-B sú protokoly formálne používané v transpondéroch aplikácií vyžadujúcim zvieraciu identifikáciu. Oba protokoly sú popísane normami ISO 11784 & 11785. Hlavným rozdielom medzi týmito dvoma protokolmi je ich spôsob komunikácie. FDX-B transpondér (Full Duplex) posiela svoje dáta v tom istom čase ako RFID modul vysiela magnetické pole zo svojej antény. Tento spôsob komunikácie dovoľuje FDX-B transpondéru byť stále pod prúdom v prítomnosti magnetického poľa antény.

Nevýhodou tejto metódy je, že silné magnetické pole prehlušuje signál transpondéru čo zapríčiňuje zhoršenú čitateľnosť (Design, 2007).

FDX-B používa na svoju komunikáciu 134,2kHz frekvenciu bipfázovým kódovaním. Bipfázová kódovacia schéma moduluje RF pole a vytvára jednoznačný prechod na začiatku každého bitu. Logická nula má prechod v strede bitovej periódy. Na rozdiel od nuly, logická jednotka nemá žiaden prechod v bitovej perióde. Pre bitovú dĺžku FDX-B korešponduje 32 cyklov aktívneho poľa čítačky.



Obrázok 40. Manchesterské a Bipfázové kódovanie protokolu FDX-B

FDX-B protokol nesie 128 bitov dát pri štruktúre:

- 11 bitov hlavičky (10000000000),
- 64 identifikačných bitov s 8 kontrolnými bitmi,
- 16 bitov CRC s dvoma kontrolnými bitmi,
- 24 rozšírených bitov dát s 3 kontrolnými bitmi.

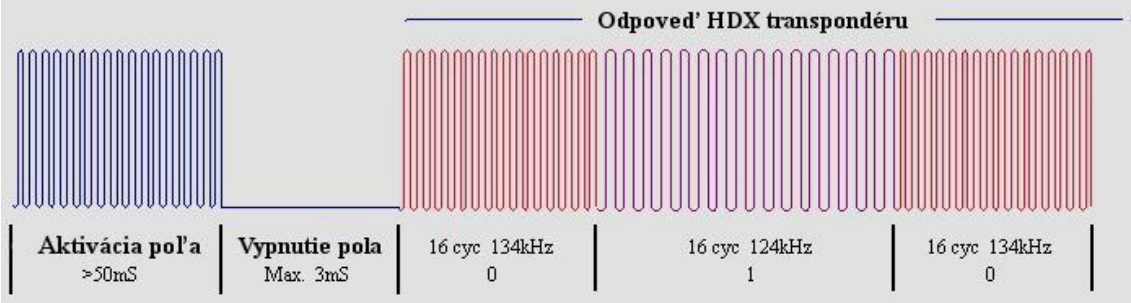
Transpondér HDX (Half duplex) je rovnako ako FDX-B napájaným magnetickým polom antény no namiesto okamžitej komunikácie čaká pokým RFID modul neprestane komunikovať. To znamená, že transpondéry HDX dokážu komunikovať s RFID modulom pri väčšej vzdialenosti.

HDX pracuje na rovnakej frekvencií ako transpondér FDX-B s uplatnením kľúčového fázového posunu pri odosielaní svojich dát. Pre logickú jednotku vyšle signál o 16 cykloch pri frekvencií 124kHz a pre logickú nulu vyšle signál o 16 cykloch pri frekvencií 134kHz (Vuza & Frosch, 2010).

HDX protokol nesie 112 bitov dát pri štruktúre:

- 8 bitov hlavičky (01111110),
- 64 identifikačných bitov,
- 16 bitov CRC,
- 24 rozšírených bitov dát.

Aktivačné pole vytvárané čítačkou je zvyčajne o veľkosti 50 milisekúnd. Počas tohto času čítačka vyhľadá prítomnosť transpondéru s protokolom FDX-B. V prípade, ak ho nenájde čítačka, vypne svoje pole a dovolí transpondéru s protokolom HDX odoslať svoje dáta. V prípade, ak RFID modul v čase 3 milisekundy nezachytí odpoveď, znova sa pokúsi o spojenie s FDX-B transpondérom.



Obrázok 41. Práca HDX protokolu

Monitorovanie drobných živočíchov pomocou RFID tagov

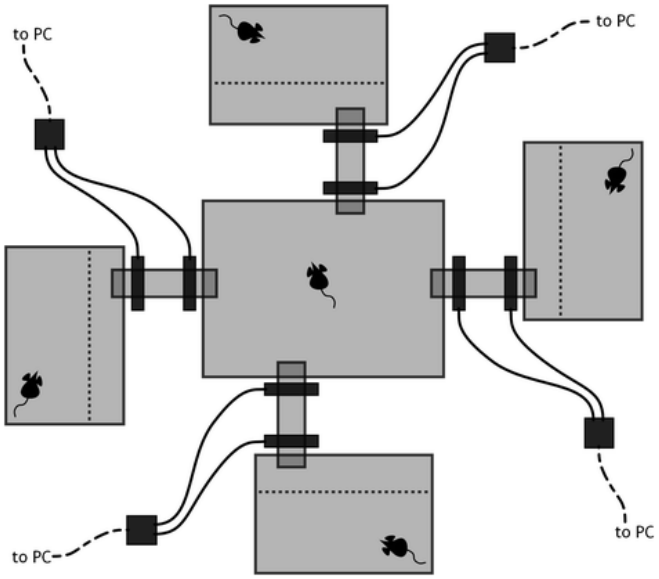
Zber dát o pohybe a správaní hlodavcov sa po dlhé roky vykonával manuálne fyzickým pozorovaním. Pozorovateľ bol pri takomto zbere informácií natrénovaný na rozpoznávanie jedinečných vzorov správania a pohybu hlodavcov. Jednotlivé hlodavce boli zároveň označkované pomocou farebných vzorov na ich tele alebo amputáciou častí končatín. Nevýhodou takéhoto zberu dát bola nespoľahlivosť pozorovateľa o rozpoznanie a záznam dát. Túto chybovosť zapríčiňovala nálada a únava pozorovateľa. Takisto každý nový pozorovateľ si so sebou prináša svoju vlastnú zaujatosť do zoznamovacieho procesu (Kritzler a kol., 2007). Keďže myši sú nočné zvieratá, zber dát o ich pohybe sa takisto zhoršil z dôvodu zlej viditeľnosti.

Automatické riešenia monitoringu hlodavcov je často krát označované ako metóda na zlepšenie štandardného testovania. Automatické systémy ponúkajú exaktné údaje a znižujú nároky na fyzickú manipuláciu zo zvieratami.

Z nástupom RFID technológie do poľa monitoringu zvierat si našla táto technológia uplatnenie i pri zaznamenávaní pohybu a správania drobných hlodavcov. Metódy s použitím RFID tagov sú určené pre prácu v laboratórnych podmienkach a v uzatvorených priestoroch. Tieto systémy zvyčajne pracujú na systéme sieti klietok, ktoré sú medzi sebou poprepájané pomocou tunelov obsahujúcich RFID modul. V prípade, ak myš prejde jedným z tunelov, počítač zaznamená tento údaj a uloží ho do svojej databázy.

Takéto systémy môžu byť obohatené o kamerový systém, infračervené senzory, telemetriu a systémami na kvantifikáciu vibrácií jednotlivých klietok. Tieto rozšírenia doplňujú detekciu správania sa hlodavcov a ich stavu v závislosti na skúmanom jave. Hlavnou nevýhodou komerčne ponúkaných systémov monitorovania hlodavcov pomocou RFID tagov je ich vysoká cena a neflexibilita.

Príkladom takéhoto systému je systém použitý Miriam Linnenbrink v roku 2007 na testovanie kritérií výberu partnerov hlodavcov. Použitý systém bol zložený z piatich klietok, pričom štyri okrajové klietky boli pripojene k centrálnej klietke pomocou tunelov. Každý tunel obsahoval dve RFID čítačky, ktoré zaznamenávali pohyb myší. Každá okrajová klietka bola zároveň rozdelená mriežkou s účelom zabránenia páreniu testovaných myší (Linnenbrink a von Merten, 2017).



Obrázok 42. Schéma rozloženia komponentov RFID monitorovacieho systému

RFID transpondéry sú do tela hlodavcov implantované pomocou striekačky, alebo tvorbou malého nárezu skalpelom pri použití anestézie. Implantované transpondéry sú zvyčajne vkladané do tela hlodavca na dĺžku pre optimalizáciu čítania transpondéru. V laboratórnych podmienkach bývajú hlodavce takisto farebne označené pre ich jednoduché rozpoznanie pozorovateľom bez nutnosti ďalšej manipulácie (Howerton a kol., 2012)

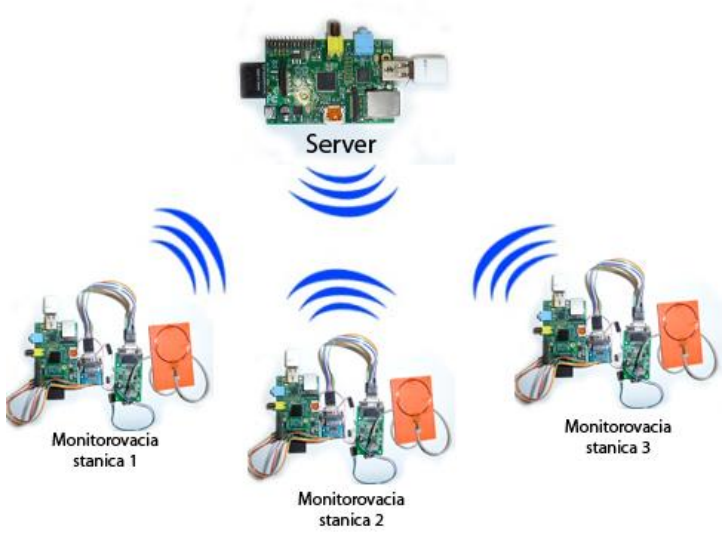
Návrh riešenia monitoringu živočíchov pomocou RFID

Navrhovaný systém zberu dát z priestorovej aktivity sa skladá z dvoch častí založených na mikropočítači Raspberry Pi.

Provu časťou riešenia je monitorovacia stanica, určená na zber dát o pohybe hlodavcov. Tieto dáta sú zberané pomocou RFID modulu RFIDRW-E-USB, ktorý zaznamenáva dáta o implantovaných tagoch hlodavcov v jej okolí. V prípade, ak hlodavec nemá implantovaný RFID tag, stanica je stále schopná zaznamenať prítomnosť hlodavca pomocou tlakového spínača. Tieto dáta o pohybe sú následne zaznamenané a odoslané spolu s aktuálnym časom a označením stanice na spracovanie. V prípade, ak tieto dáta nie sú z akéhokoľvek dôvodu odoslané, stanica ich uloží a pokúsi sa o odoslanie znovu. Stanica používa na komunikáciu WiFI anténu a dokáže pracovať z batérie.

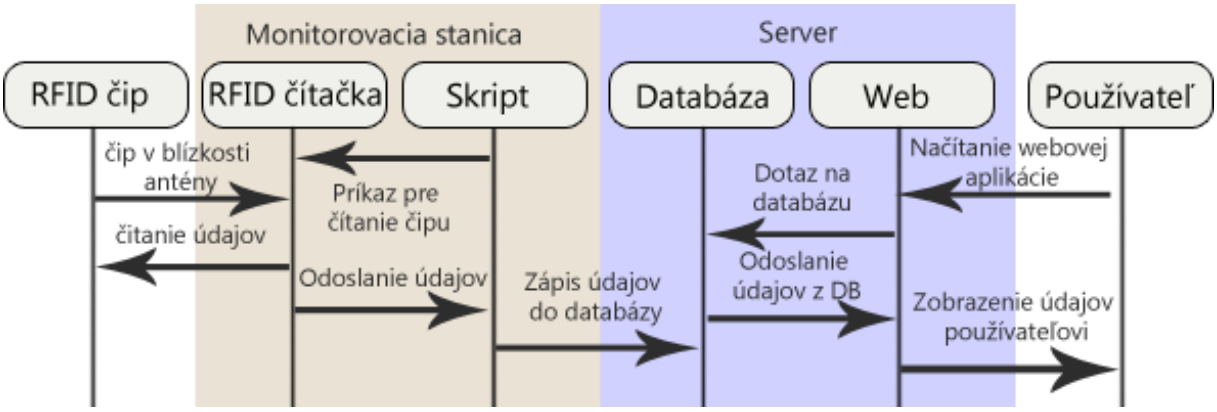
Druhou časťou monitorovacieho systému je server. Server dokáže prijímať dáta o pohybe hlodavcov odoslané monitorovacími stanicami a spracovať ich. Server tieto dáta následne zobrazuje na webovej stránke. ktorú hosťuje. Táto webová stránka takisto dokáže zaznamenávať momentálny stav staníc a dovoľuje export nazberaných dát.

Vybranými komponentmi jadra monitorovacieho systému sú mikropočítač Raspberry Pi a RFID modul RFIDRW-E-USB. Mikropočítač Raspberry Pi spĺňa potrebné hardwarové a sieťové požiadavky serveru i stanice a rovnako zaručuje jednoduchú a rýchlu inštaláciu. RFID modul RFIDRW-E-USB bol vybraný z dôvodu jeho schopnosti pracovať s HDX a FDX-B zvieracími transpondérmi a jeho zaručenou dostupnosťou (Balogh a Baláž, 2020).



Obrázok 43. Monitorovací systém

Server bude obsahovať webový server, ktorom bude umiestnená webová aplikácia pre prístup k údajom v relačnej databáze. Server bude obsahovať DHCP server, aby nebolo potrebné nastavovať IP adresu na každej monitorovacej stanici zvlášť. Server bude taktiež obsahovať DNS server, pre zjednodušenie prístupu k webovej aplikácii. DNS server sa bude odkazovať na verejný DNS server od Google pre prípadné pripojenie do siete Internet. Údaje načítané monitorovacími stanicami sa budú ukladať do MySQL databázy zriadenej na serveri. Kompletná komunikácia v systéme je zobrazená na Obrázku 44.

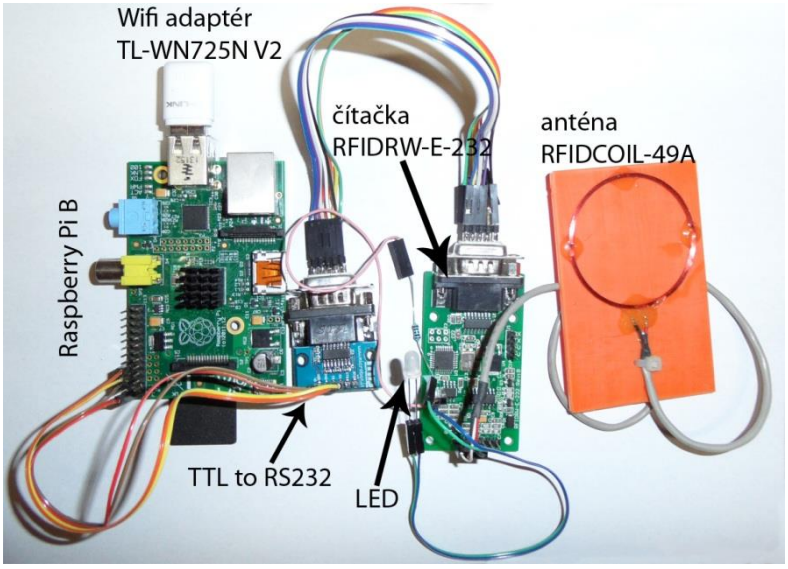


Obrázok 44. Komunikácia monitorovacieho systému

Následne bude možné k údajom pristupovať pomocou pripojenia sa do wifi siete vysielanej serverom a navštívení URL adresy http://pasce.ukf. Webová aplikácia bude umožňovať prezeranie záznamov uložených v databáze, ich export a prípadné filtrovanie.

Monitorovacia stanica

Monitorovacia stanica (Obrázok 45) sa skladá z riadiacej jednotky – minipočítač Raspberry Pi, čítačky RFIDRW-E-232, signalizačnej dvojfarebnej led, 1kΩ rezistor, wifi adaptéra (TL-WN725N V2) a prekladača TTL logiky na RS232 logiku.

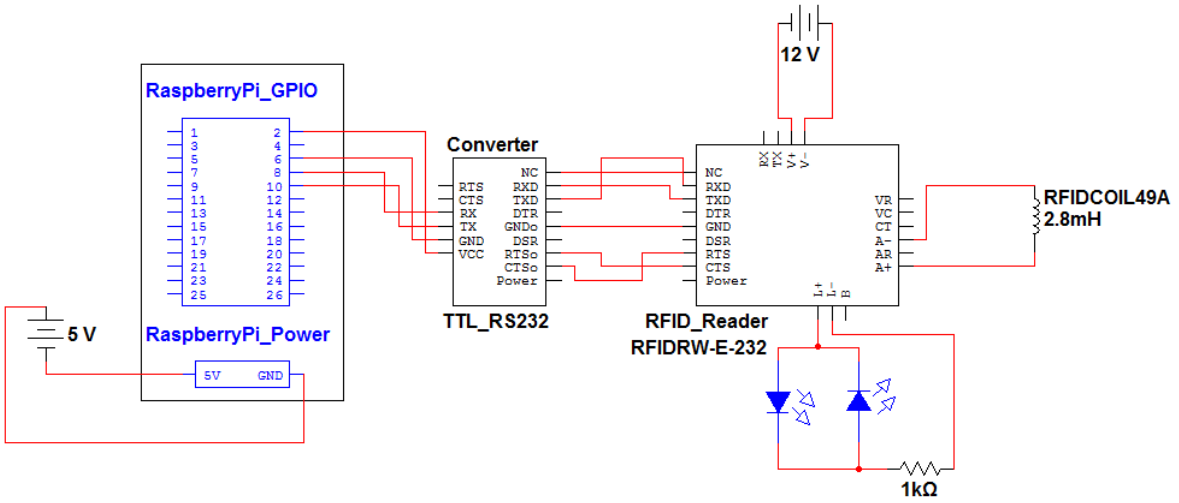


Obrázok 45. Monitorovacia stanica

Komunikácia prebieha prostredníctvom dvoch UART pinov:

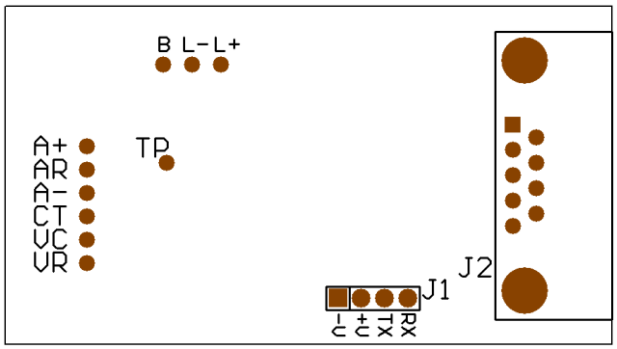
- RXD – pin 10 na Raspberry Pi,
- TXD – pin 7 na Raspberry Pi.

UART piny na Raspberry Pi pracujú na 3.3V logike a čítačka RFIDRW-E-232 pracuje na 5V logike. Kvôli tejto nezhode musí komunikácia prechádzať cez prekladač TTL (3.3V) na RS232 (-25V do 25V). Komunikácia medzi pinom RXD a pinom TXD musí byť prekrížená, to znamená, že pin RXD na Raspberry Pi prepojíme s pinom TXD na čítačke a pin TXD na Raspberry Pi prepojíme s pinom RXD na čítačke (Obrázok 46).



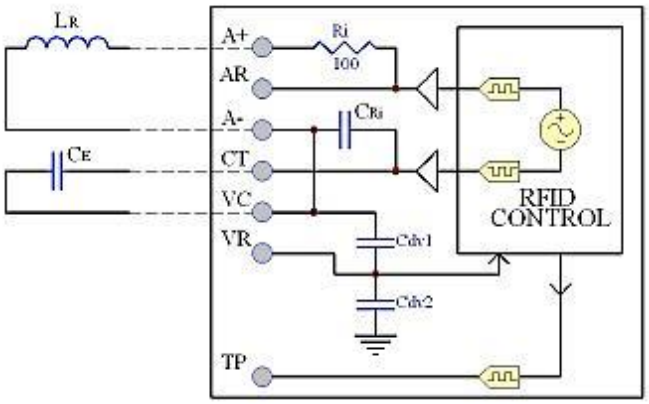
Obrázok 46. Schéma zapojenia monitorovacej stanice

Napájacie napätie na Raspberry Pi je 5V a odporúčaný prúd je 1A. Prevodník potrebuje napájanie 5V, ktoré nám následne poskytuje Raspberry Pi z pinu číslo 2., zem je možné napojiť na pin číslo 6 na Raspberry Pi.



Obrázok 47. Rozloženie pinov na RFID čítačke

Čítačka potrebuje napájanie 12V a minimálne 38mA. Na čítačku je možné pripojiť dvojfarebnú LED diódu, ktorá signalizuje čítanie a zápis na čip. Dióda sa zapojí na piny L+ L-. Na čítačku je taktiež možné pripojiť bzučiak na pin B.



Obrázok 48. Všeobecné zapojenie antény na čítačke

Čítačka disponuje možnosťou nastavenia frekvencie antény pomocou kondenzátoru. Pre výpočet kapacity kondenzátora pre danú frekvenciu je možné použiť nasledovné rovnice:

$$f_0 = \frac{1}{2 * \pi * \sqrt{L_R * C_R}}$$

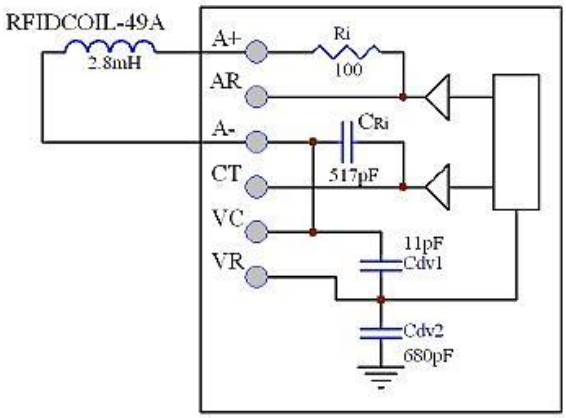
$$C_R = C_E + C_{Ri}$$

pričom f_0 = operačná frekvencia,

L_R = indukčnosť antény,

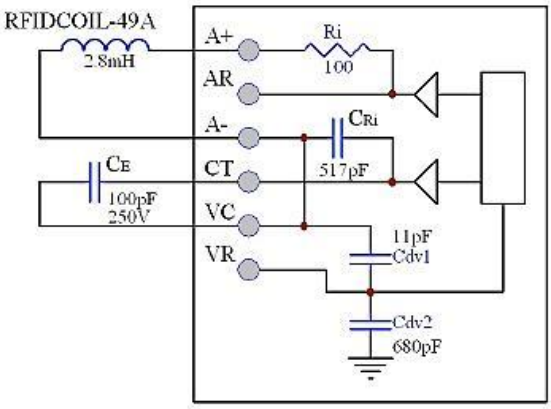
C_R = kapacita kondenzátora.

Ako príklady si môžeme uviesť použitie antény RFIDCOIL-49A na frekvencii 134kHz a 125kHz.



Obrázok 49. Anténa s frekvenciou 134kHz

Čítačka podporuje anténu RFIDCOIL-49A na frekvencii 134kHz bez akéhokoľvek prídavného kondenzátora. O nastavenie frekvencie sa starajú len kondenzátory a odpor v čítačke.



Obrázok 50. Anténa s frekvenciou 125kHz

Ak potrebujeme aby anténa pracovala na frekvencii 125kHz, tak po dosadení hodnôt do rovnice

$$125 = \frac{1}{2 * 3.14 * \sqrt{2800 * C_R}}$$

$$C_R = 0.560714\ nF$$

$$C_E = 560\ pF - 517\ pF \qquad C_E \sim 100\ pF$$

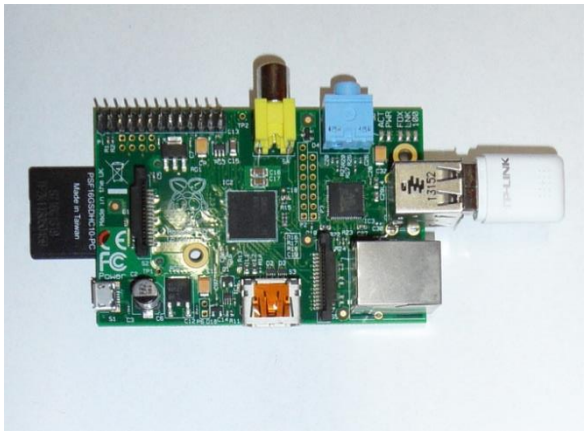
dostaneme výslednú hodnotu kapacity kondenzátora C_E približne 100 pF. Po pripojení tohto kondenzátora bude čítačka pracovať na frekvencii 125 kHz.

Princíp komunikácie RFID čipu s čítačkou

RFIDRW-E čítačka generuje magnetické pole pomocou externej antény, zvyčajne na frekvencii 125 kHz alebo 134 kHz. Pasívne čipy majú integrovanú anténu, ktorá je naladená na rovnakú frekvenciu. Keď je čip v dosahu magnetického poľa vysielaného externou anténou čítačky, čip je schopný čerpať dostatok energie z elektromagnetického poľa na napájanie vlastnej elektroniky. V momente ako je čip napájaný, je schopný modulovať magnetické pole, ktoré je zachytávané čítačkou. Týmto spôsobom je možné, aby čipy prenášali svoje dáta k čítačke. Existuje mnoho rôznych typov čipov, ktoré môžu pracovať na rôznych frekvenciách. Pomocou výberu správnej antény a nastavenia kapacity, môže byť čítačka RFIDRW-E naladená na rovnakú frekvenciu ako skenovaný čip.

Server

Server (Obrázok 51) zabezpečuje WIFI sieť pre monitorovacie stanice, relačnú databázu, DHCP server, DNS server a web server. Server sa skladá z minipočítača Raspberry Pi a wifi adaptéra (TL-WN725N V2). Jednotlivé monitorovacie stanice sa pri zapnutí ihneď automaticky prihlásia do siete vytvorenej serverom. Pre prístup na webovú aplikáciu, ktorá bude spustená na webovom serveri bude tiež potrebné pripojiť notebook do siete a zadať URL adresu webovej aplikácie.



Obrázok 51. Server

Softvér pre monitorovací systém

Softvér pre monitorovací systém sa skladá z troch väčších častí. Prvá časť je spoločná pre celý monitorovací systém (monitorovacie stanice a server). Skladá sa z operačného systému Raspbian, Java development kit-u a rôznych nastavení operačného systému. Druhá časť obsahuje softvér len pre monitorovacie stanice. Pri monitorovacích staniciach je potrebné vykonať nasledujúce kroky:

- Uvoľniť sériový port. Pre komunikáciu s čítačkou je potrebné uvoľniť sériový port. Štandardne je sériový port v Raspberry Pi využívaný pre poskytovanie informácií o boote alebo slúži pre prihlásenie do systému.
- Nainštalovať knižnicu Pi4j. Pi4j je knižnica, pomocou ktorej môžeme pristupovať k GPIO pinom z programovacieho jazyku Java, je závislá na knižnici WiringPi.
- Spustiť vytvorenú aplikáciu, ktorá bude zaznamenávať čipy, ktoré sa pohybujú v blízkosti monitorovacej stanice a odosielať dáta do databázy na server.

Tretia časť obsahuje nasledujúci softvér pre server:

- MySQL databáza. Umožňuje uchovávanie dát z jednotlivých monitorovacích staníc a umožňuje jednoduchý prístup k dátam webovej aplikácii.
- Tomcat 8 webserver. Na tomto webovom serveri bude spustená naša webová aplikácia.
- Unbound DNS. Umožní nám pristupovať k webovej aplikácii zadaním zvolenej doménovej adresy namiesto IP adresy. Medzi jeho hlavné výhody patrí jednoduchá konfigurácia.
- Udhcpd a nastavenia siete. Udhcpd nám poskytne dhcp server pre prideľovanie IP adries jednotlivým monitorovacím staniciam, ktoré sa budú pripájať do siete, ktorú vytvorí server.

Naprogramovaný softvér pre monitorovací systém sa skladá z dvoch častí. Prvá časť zabezpečuje komunikáciu s čítačkou RFID čipov a zápis načítaných údajov do relačnej databázy. Druhú časť tvorí webová aplikácia, ktorá zabezpečuje prístup k dátam uložených v databáze.

Vytvorený softvér pre monitorovaciu stanicu

Komunikácia s čítačkou prebieha pomocou odosielania reťazcov zakončených špeciálnym znakom (CR - posunutie kurzoru na začiatok riadku) cez sériový port. Komunikáciu cez sériový port a GPIO zabezpečuje knižnica pi4j. Komunikácia cez sériový port sa iniciuje pomocou riadku:

```
serial.open("/dev/ttyAMA0",9600);
```

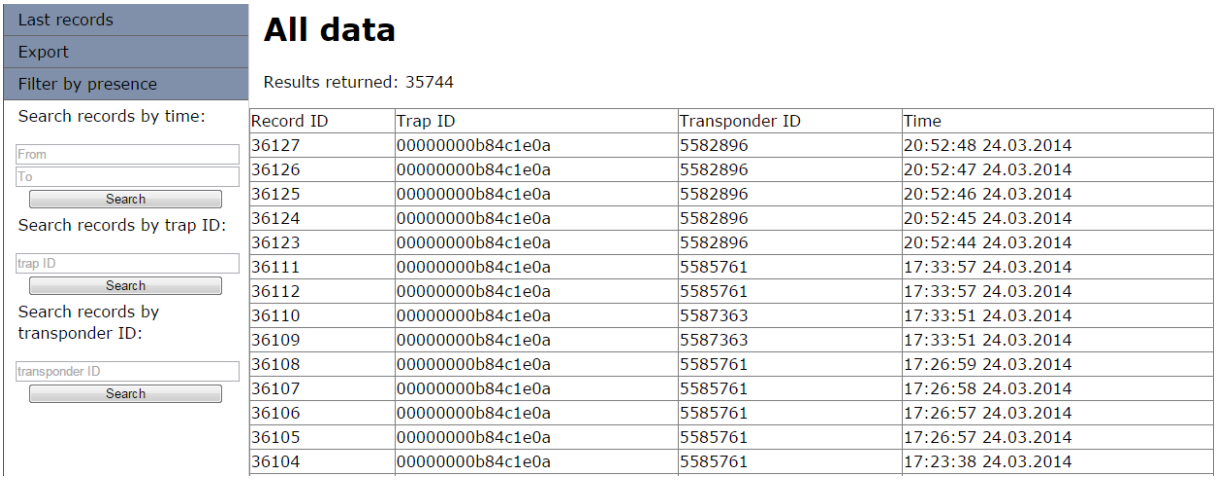
Sériový port sa nachádza na ceste /dev/ttyAMA0 a číslo 9600 udáva baudrate (rýchlosť komunikácie).

Po otvorení sériového portu sa aplikácie pokúsi nadviazať spojenie s databázou, ktorá sa nachádza na serveri. Ak sa spojenie nepodarí, aplikácia počká 10 sekúnd a opätovne sa pokúsi nadviazať spojenie, až kým spojenie nie je aktívne. Následne sa k sériovému portu pridá listener, aby bolo možné zachytiť údaje, ktoré naspäť odosiela RFID čítačka. Metóda dataReceived (SerialDataEvent event) zabezpečuje spracovanie všetkých údajov, ktoré prídu cez sériový port. Kvôli možným chybám pri prenose sa kontroluje správnosť údajov skontrolovaním ich dĺžky. Ak sú údaje správne, do databázy sa uloží identita čipu a identita čítačky, ktorá čip prečítala. Pri ukladaní údajov do databázy sa k záznamu pridá aj aktuálny čas uloženia záznamu. Metóda run() zabezpečuje pravidelné dotazovanie sa na RFID čítačku, či sa v jej blízkosti nachádza RFID čip. Prípadné chyby sa počas behu programu zapisujú do logovacieho súboru logy.log. O vytvorenie logovacieho súboru a spustenie triedy RFIDreader sa stará trieda Trap. Trieda Trap pri spustení čaká 15 sekúnd, aby mal server dostatok času pre spustenie databázy a aby sa predišlo neúspešným pokusom o pripojenie k databáze.

Použité príkazy pre RFID čítačku:
SL0 – červená dióda skenovanie/zelená čítanie
SL4 – dióda vypnutá
SL5 – zelená dióda
SL6 – červená dióda
SRA – aktivovať RFID čítačku
ST2 – nastaviť čítačku pre čipy FDX-B/HDX
RAT – prečítať FDX-B/HDX čip

Vytvorený softvér pre server

Pre server sme museli vytvoriť webovú aplikáciu (Obrázok 52), ktorá umožňuje prístup k dátam. Dáta je možné prezerať, exportovať do formátu CSV a XML. Exportovať dáta je možné podľa zadaného dátumu alebo podľa zadaných x posledných záznamov. Exportované dáta je ďalej možné spracovať v akomkoľvek programe, ktorý podporuje import dát v daných formátoch. Webová aplikácia taktiež umožňuje filtrovať zobrazené výsledky podľa dátumu, identifikátoru monitorovacej stanice, identifikátoru čipu a zobraziť v akej dobe sa daný čip nachádzal v blízkosti monitorovacej stanice.



Record ID	Trap ID	Transponder ID	Time
36127	00000000b84c1e0a	5582896	20:52:48 24.03.2014
36126	00000000b84c1e0a	5582896	20:52:47 24.03.2014
36125	00000000b84c1e0a	5582896	20:52:46 24.03.2014
36124	00000000b84c1e0a	5582896	20:52:45 24.03.2014
36123	00000000b84c1e0a	5582896	20:52:44 24.03.2014
36111	00000000b84c1e0a	5585761	17:33:57 24.03.2014
36112	00000000b84c1e0a	5585761	17:33:57 24.03.2014
36110	00000000b84c1e0a	5587363	17:33:51 24.03.2014
36109	00000000b84c1e0a	5587363	17:33:51 24.03.2014
36108	00000000b84c1e0a	5585761	17:26:59 24.03.2014
36107	00000000b84c1e0a	5585761	17:26:58 24.03.2014
36106	00000000b84c1e0a	5585761	17:26:57 24.03.2014
36105	00000000b84c1e0a	5585761	17:26:57 24.03.2014
36104	00000000b84c1e0a	5585761	17:23:38 24.03.2014

Obrázok 52. Webová aplikácia

O nadviazanie spojenia s databázou sa stará trieda ConnectionFactory. Všetky konštanty používané aplikáciou sú uložené v triede WebConstants, čo umožňuje jednoduchú zmenu hlavných nastavení programu. O komunikáciu s databázou sa stará trieda DataBean. Táto trieda zabezpečuje načítanie dát pre každú časť webovej aplikácie.

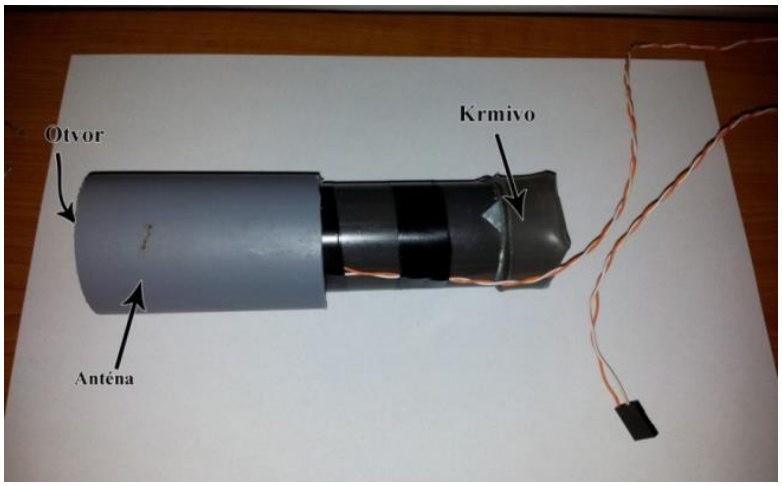
Webová aplikácia obsahuje aj zobrazenie dát podľa zdržania sa pri monitorovacej stanici. Táto funkcia zobrazí dáta v rozsahu v akom bol čip prítomný pri monitorovacej stanici.

Testovanie a vyhodnotenie monitorovacieho systému

Monitorovací systém sme testovali v laboratórnych podmienkach z rôznych hľadísk ako spoľahlivosť systému, výdrž pri napájaní z batérie a aktivity RFID antény.

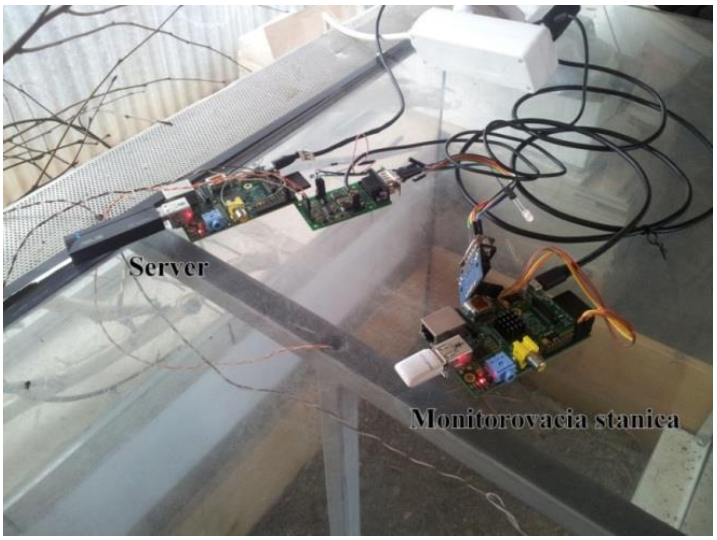
Testovanie spoľahlivosti monitorovacieho systému

Testovanie systému prebiehalo v laboratórnych podmienkach. Pre testovanie sa čip vložil do dvoch myší, ktoré boli umiestnené do veľkého terária. Monitorovací systém pozostával z jedného servera a jednej monitorovacej stanice. Pascu predstavovala umelohmotná trubka, ktorá bola otvorená len z jednej strany, kde bola umiestnená aj anténa pre načítavanie RFID čipov (Obrázok 53). Do druhej strany pasce bolo umiestnené krmivo pre myši.



Obrázok 53. Pasca

Monitorovací systém bol umiestnený na teráriu. Monitorovacia stanica bola prepojená s pascou pomocou dvojmetrového kábla.

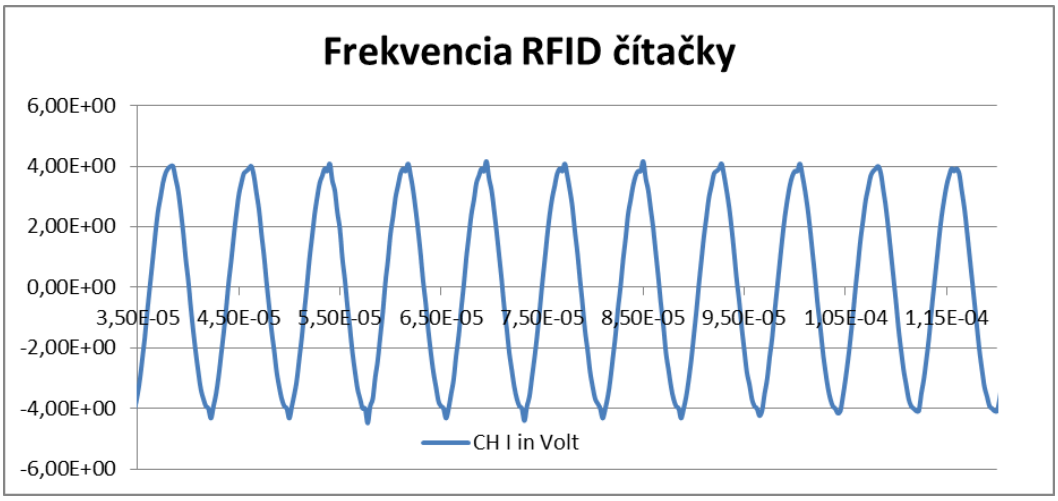


Obrázok 54. Monitorovací systém

Počas testovania boli server aj monitorovacia stanica napájané z elektrickej siete (Obrázok 54). Testovanie prebiehalo 7 dní a počas testovania nedošlo k žiadnej chybe alebo zlyhaniu. Počas testovania však bolo treba vymeniť jednej z myší čip pretože bol nesprávne implantovaný a strácala ho. Pri tomto testovaní bolo nahromadených dokopy 35744 záznamov do relačnej databázy a systém sa ukázal ako spoľahlivý (Balogh a kol., 2016; Balogh a Turcani, 2016).

Sledovanie aktivity RFID antény

Pri sledovaní spoľahlivosti systému sme zistili, že anténa neregistruje RFID čipy z požadovanej vzdialenosti. Táto skutočnosť môže mať dva dôvody, a to príliš veľký odpor kábla, ktorým je anténa pripojená k čítačke alebo nesprávne naladená frekvencia. Aktivitu antény sme sledovali pomocou osciloskopu a počítačového softvéru. Pomocou osciloskopu sme zmerali presnú frekvenciu, na ktorej anténa vysiela a porovnali ju s frekvenciou 134 kHz vypočítanou podľa vzorca. Osciloskop bol schopný zapamätať si a odoslať do počítača 2046 meraní v rôznych intervaloch. Pre zmeranie frekvencie čítačky sme použili vzorkovanie na úrovni 250 nanosekúnd (Obrázok 55). Tieto dve funkcie sú takmer totožné (Obrázok 56). Z tejto skutočnosti vyplýva, že anténa má slabý dosah kvôli veľkému odporu kábla.



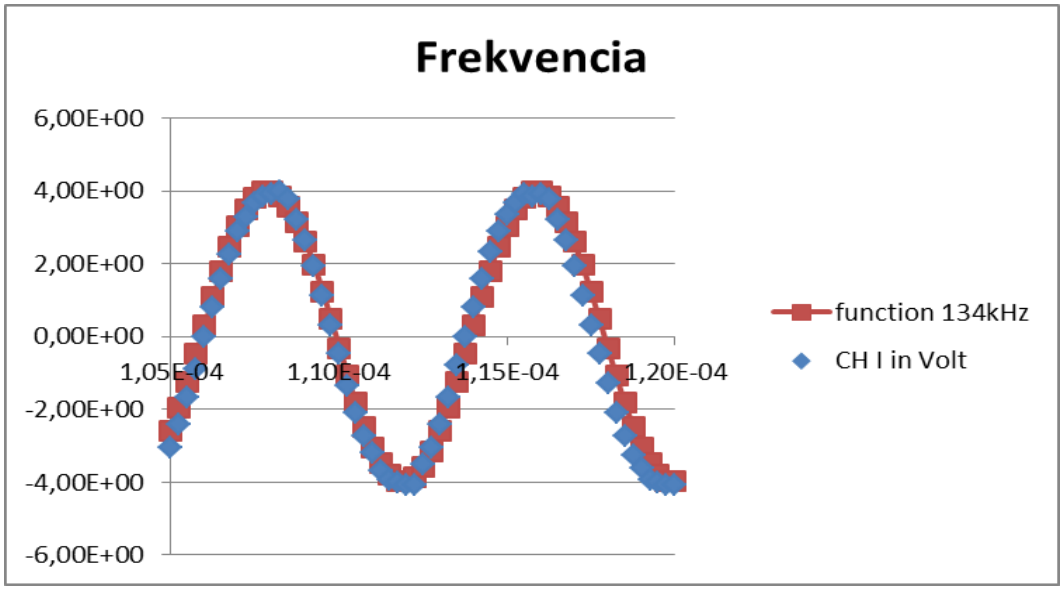
Obrázok 55. Nameraná frekvencia RFID čítačky

Výslednú hodnotu frekvencie 134 kHz sme dostali po dosadení hodnôt do vzorca:

$$f = A * \sin(\boldsymbol{\omega} * \mathbf{T} + \mathbf{f_i})$$

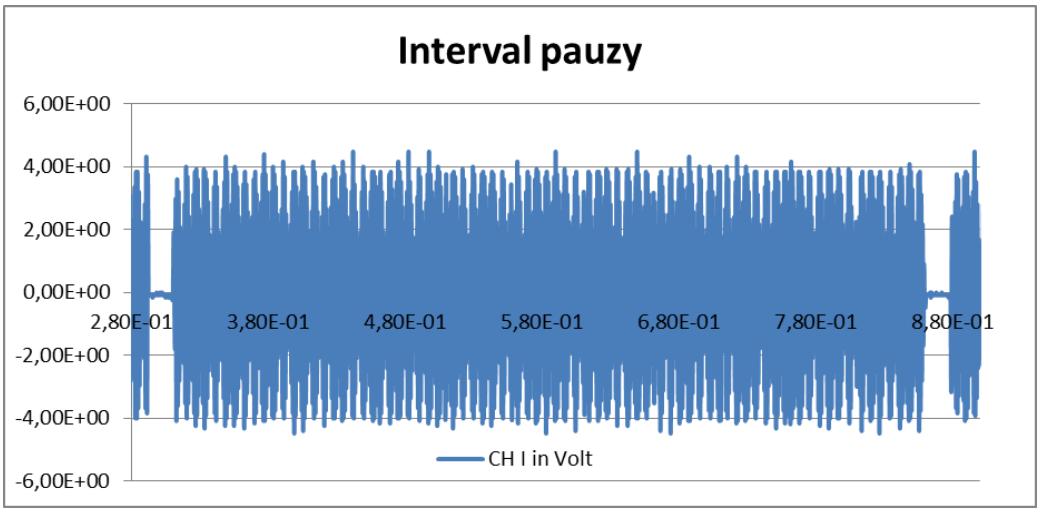
pričom

- f je výsledná funkcia, ktorá popisuje frekvenciu 134 kHz,
- A je veľkosť amplitúdy,
- ω je uhlová rýchlosť,
- T je čas,
- f_i je posunutie.

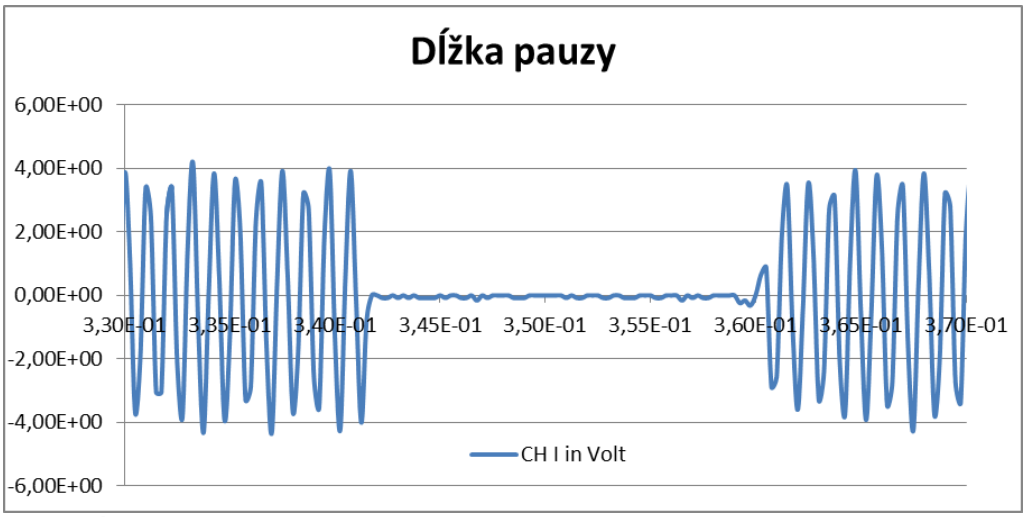


Obrázok 56. Výsledná frekvencia RFID čítačky s porovnaním vypočítanej frekvencie 134kHz

Pri sledovaní aktivity antény sme si všimli, že čítačka nie je aktívna po celý čas, ale vysielanie prerušuje. Po nameraní pauzy sme zistili, že RFID čítačka prerušuje vysielanie každých 0,5 sekúnd na dobu 0,02 sekundy (Obrázok 57, Obrázok 58). Tento interval predstavuje jeden cyklus, kým čítačka sníma okolie pre prítomnosť RFID čipu. Interval pauzy bol meraný na úrovni vzorkovania 0,5 milisekúnd a dĺžka pauzy na úrovni 0,25 milisekúnd.



Obrázok 57. Interval pauzy



Obrázok 58. Dĺžka pauzy

Keďže dĺžka pauzy je 0,02 sekundy a veľkosť okruhu v ktorom anténa dokáže zaznamenať 11mm čip je 5 cm. Rýchlosť akou by sa zver musela pohybovať, aby ju čítačka nezaznamenala, je viac ako 2,5 metra za sekundu.

Výrobné náklady systému

Pre fungujúci systém je potrebné vlastniť minimálne jeden Server a jednu Monitorovaciu stanicu. Výsledná výrobná cena Servera je $47.5:

- Raspberry Pi $35,
- WIFI adaptér TL-WN723N $12.5.

Výsledná výrobná cena pre jednotlivé Monitorovacie stanice je $90.5:

- RFID čítačka $40.00,
- Raspberry Pi $35,
- RS232 to TTL $1.50,
- Wifi adaptér TL-WN723N $12.5,
- Káble $0.5,
- Drobné komponenty $1.

Cena bežných RFID monitorovacích staníc sa pohybuje okolo $2000 a vyššie. Napríklad čítačka od spoločnosti Oregon RFID Single Antenna HDX Reader w/ EasyTuner začína na cenovke $2095. Dokáže čítať čipy používajúce normu ISO 11784, logovať a ukladať záznamy až do 10 miliónov. A jej nominálna vzdialenosť záznamu pre 23mm čip je 30cm. Cena nášho systému sa pohybuje okolo $140. Ak ale chceme rozšíriť monitorované územie stačí, ak dokúpime ďalšiu monitorovaciu stanicu za $90, nemusíme kupovať celý systém.

Záver

Prípadová štúdia popisuje možnosti ako vytvoriť systém, ktorý bude schopný monitorovať pohyb hlodavcov a inej zveri s využitím RFID technológie a mikropočítača Raspberry Pi. Systém, ktorý je prezentovaný, dokáže zautomatizovať monitorovanie priestorovej aktivity a dáta zhromažďuje v relačnej databáze umiestnenej na servery. Výsledný systém sa skladá z Monitorovacej stanice a Servera. Raspberry Pi slúži aj ako server, ale aj ako mikropočítač, ktorý komunikuje s RFID čítačkou a odosiela údaje na server (Monitorovacia stanica). Server sa skladá z Raspberry Pi a wifi prijímača. Server obsahuje DHCP server pre prideľovanie IP adries monitorovacím staniciam, relačnú databázu pre ukladanie zaznamenaných aktivít, DNS server pre ľahšie pripájanie k webovej aplikácii a web server, na ktorom je spustená webová aplikácia.

4.4 Programovanie aplikácií so senzormi, UKF v Nitre (Ján Skalka)

4.4.1 Anotácia a stručná osnova predmetu

Predmet Programovanie aplikácií so senzormi predstavuje nadstavbový materiál pre vývoj mobilných aplikácií vyučovaný u študentov bakalárskeho štúdia študijného programu aplikovaná informatika. Vývoj mobilných aplikácií patrí medzi základné zručnosti moderného programátora. Predmet ponúka rozšírenie základného kurzu vývoja mobilných aplikácií o prácu s grafikou, prácu so senzormi a ich využitie v najrozličnejších typoch aplikácií. V úvode predmetu sú zhrnuté základné znalosti z vývoja mobilných aplikácií pre Android, ktoré môžu tvoriť základnú bázu znalostí aj pre študenta, ktorý zatiaľ s vývojom mobilných aplikácií pre Android do kontaktu neprišiel.

Predmet špirálovito prechádza od základných znalostí práce až do hĺbky operačného systému Android tak, aby sa v prvom rade orientoval na využívanie rôznych typov senzorov. Predmet je rozdelený do ucelených častí, ktoré pokrývajú prácu s grafikou, prácu so senzormi a náročnejšie témy vyžadujúce znalosť architektúry OS Android. Každá časť poskytuje okrem základného popisu teoretických princípov aj štruktúrované zdrojové kódy zabezpečujúce pochopenie problematiky.

V úvode sa predmet venuje poskytnutiu základných informácií pre komunikáciu medzi aktivitami, tvorbu zoznamov a menu a následne prechádza ku grafickému spracovaniu údajov. Grafické princípy vysvetľujú témy venované vykresľovaniu 2D objektov, obrázkov a tvorbe vlastného grafického editora ovládaného pohybom prsta po displeji. Tematika animácie je pokrytá vykresľovaním a pohybom 2D objektov i grafických obrázkov. V rámci aplikácií sa vytvára niekoľko jednoduchých počítačových hier a sú prezentované koncepty rôznych typov ovládania, identifikácie kolízií a programovania inteligentných objektov. Na ovládanie počítačových hier nadväzuje využitie senzorov, kde najmä akcelerometer možno využiť ako alternatívny prvok pre určovanie smeru a rýchlosti objektov. Po prezentovaní rôznych možností použitia akcelerometra sa prechádza na využitie krokomera a vytvorenie vlastnej aplikácie merajúcej aktivitu používateľa. Jednoúčelové využitie predstavuje práca so svetelným senzorom a naopak multifunkčným prvkom je magnetický senzor využívaný v aplikáciách vyžadujúcich zistenie orientácie mobilného zariadenia v priestore.

Callbacková komunikácia predstavuje pri komunikácii zariadenia s iným zariadením nevyhnutný prvok získavania informácií. Postupnosť kapitol *Broadcast Receiver* a *Bluetooth komunikácia* poskytujú študentovi základné informácie pre využívanie komunikačných princípov v rámci komunikácie s dlhšou odozvou. Lokalizácia prostredníctvom GPS tvorí samostatnú kapitolu a poskytuje plynulý prechod k tvorbe aplikácií využívajúcich mapy. Na tento účel sú predstavené dva prístupy – využívanie *Google Maps* a *MapBox*-u.

Výsledky vzdelávania

Študent pozná situácie a techniky typické pre využívanie senzorov (najmä mobilných zariadení). Študent aplikuje znalosti z vývoja mobilných aplikácií do situácií, v ktorých sa využívajú senzory. Študent dokáže využívať grafiku, animáciu a spracovanie dotykov pri grafických aplikáciách. Študent dokáže vytvoriť komplexnú mobilnú aplikáciu využívajúcu všetky potrebné komunikačné rozhrania (bluetooth, internet, senzory, gestá a pod.) v operačnom systéme Android. Študent pozná a vie riešiť problémy spojené s bezpečnosťou komunikácie a oprávneniami v OS Android.

4.4.2 Aplikačný príklad – práca s mapovými podkladmi

V rámci aplikačného príkladu sú predstavené dva rôzne prístupy k práci s mapovými podkladmi. Prvý (práca s mapami Google) predstavuje univerzálny štandard spolu s princípmi využívania služieb Google, druhý (ako lacnejšia alternatíva) je orientovaný na *MapBox*, ktorý je v súčasnosti hojne používaný v mnohých aplikáciách a je zvyčajne prvou voľbou pri tvorbe záverečných prác.

Práca s mapovými podkladmi prestáva byť pre poskytovateľa údajov v určitom momente záležitosťou charity a stáva sa produktom predaja – po určitom počte dotazov je potrebné za vybavovanie dotazov našej aplikácie platiť poskytovateľovi obsahu. Na tento účel sa používa identifikátor aplikácie a používateľa, ktorý jednak umožňuje používateľovi sledovať využiteľnosť aplikácie (princíp Google analytics) a jedna poskytovateľovi sledovať, či už aplikácia neprekročila bezplatný limit.

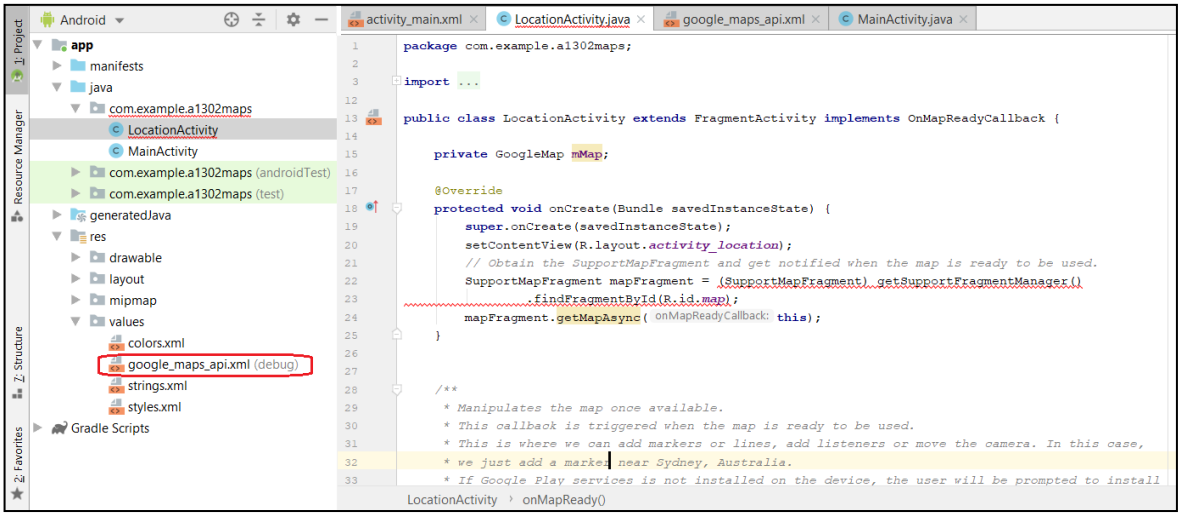
Google Maps

Prístup k mapám získame najjednoduchšie vytvorením samostatnej aktivity s mapou, ktorú pridáme ako aktivitu do aplikácie, v ktorej budeme prezentovať jednotlivé funkcionality Google máp.

Na to, aby mohla naša aplikácia pristupovať k mapám, potrebuje komunikovať so službou *Google Maps*, ktorá mapy poskytuje, aby Google vedel:

- ktorá to je aplikácia,
- komu patrí,
- či už neprekročila bezplatný limit.

Z týchto dôvodov sa vyžaduje pri komunikácii so službou identifikátor – **API key**. Postup k získaniu API key je uvedený v súbore *google_maps_api.xml*.



Obrázok 59. Postup k získaniu API key

Kľúč získame vo vývojárskej konzole Google buď priamym zadaním vygenerovaného linku, ktorý má podobu napr.:
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=58:8E:F5:42:E0:25:85:87:11:11:EC:C8:11:8C:B4:16:9A:97:D3:10%3Bcom.example.a1302maps
alebo cez https://console.developer.google.com a následný prechod po položkách.

Získaný kľúč vložíme do súboru so stringovými hodnotami (values), odkiaľ si ho aplikácia vytiahne na miesta, kde potrebuje.

Do manifestu pridáme oprávnenie pre prácu s Internetom, ďalšie dve oprávnenia sú vhodné, ale nepovinné.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.a1302maps">

<!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
Google Maps Android API v2, but you must specify either coarse or
fine location permissions for the 'MyLocation' functionality.
-->
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
...
```

Google maps ponúka viaceré funkcionality, ktorých použitie prezentujeme na ďalších riadkoch. Vytvorme jednu aktivitu, z ktorej bude možné volať jednotlivé funkcionality. Na to bude využívaná stále tá istá aktivita s mapou, do ktorej budú posielané rôzne požiadavky.

Kľúčovou metódou je *onMapReady()*, v ktorej sa údaje, ktoré prídu do aktivity získajú cez *getIntent* a použijú vtedy, keď je mapa pripravená. Jednotlivé úlohy rozlíšime pomocou čísla úlohy definovaného v intente.

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    LatLng myPosition;
    Intent vstupIntent = getIntent();
    Bundle data = vstupIntent.getExtras();
    int uloha = data.getInt("uloha");

    switch (uloha) {
        case 1 : ...
                 break;
        case 2 : ...
                 break;
     ...
```

Pozícia na mape

Prvú úlohu vyriešime zadaním pozície pri kliknutí na tlačidlo v *main* aktivite a načítaním v *map* aktivite:

Vyžiadajme si zobrazenie napríklad pozície UKF, údaje o polohe pribalíme do intentu pre prvé tlačidlo:

```
public void onClickLocation(View view) {
    Intent mapIntent = new Intent(this, LocationActivity.class);
    // vytvorím polohu
    LatLng myPosition = new LatLng(48.308526, 18.091698);
    mapIntent.putExtra("uloha", 1); // definujem číslo úlohy
    mapIntent.putExtra("position", myPosition); // pribalím pozíciu
    startActivity(mapIntent); // otvorím aktivitu
}
```

V *onMapReady()* doplníme riešenie pre úlohu typu 1:

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    LatLng myPosition;
    Intent vstupIntent = getIntent();
    Bundle data = vstupIntent.getExtras();
    int uloha = data.getInt("uloha");

    switch (uloha) {
        case 1 : myPosition = (LatLng) data.get("position");
            mMap.addMarker(new

MarkerOptions().position(myPosition).title("UKF"));

mMap.moveCamera(CameraUpdateFactory.newLatLng(myPosition));
            break;
     ...
```

Postup je nasledovný:

- v prvom kroku sa prečíta poradové číslo úlohy,
- v druhom kroku sa vyberie kód pre úlohu a prečítajú sa parametre,
- na danej pozícii sa vytvorí marker s textom UKF,
- napokon sa na danú pozíciu presunie kamera.

Obdĺžniková oblasť

Na mape možno v rámci samostatných vrstiev vykresliť ľubovoľný grafický útvar. Pre vykreslenie obdĺžnika použijeme príkazy na vykreslenie n-uholníka. Tento postup sa používa na ohraničenie vybranej oblasti. Príkaz *PolylineOptions* umožňuje definovať pomocou vrcholov ľubovoľnú postupnosť vrcholov.

Príprava bodov v hlavnej aktivite:

```
public void onClickRectangle(View view) {
    PolylineOptions rectOptions = new PolylineOptions()
            .add(new LatLng(48.2, 18))
            .add(new LatLng(48.2, 18.2))
            .add(new LatLng(48.4, 18.2))
            .add(new LatLng(48.4, 18))
            .add(new LatLng(48.2, 18));

    Intent mapIntent = new Intent(this, LocationActivity.class);
    mapIntent.putExtra("uloha",2);
    mapIntent.putExtra("data",rectOptions);
    startActivity(mapIntent);
}
```

Zobrazenie v mapovej aktivite:

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    LatLng myPosition;
    Intent vstupIntent = getIntent();
    Bundle data = vstupIntent.getExtras();
    int uloha = data.getInt("uloha");
         switch (uloha) {
      ...
     case 2 : PolylineOptions rectOptions = (PolylineOptions)
     data.get("data");
        mMap.addPolyline(rectOptions);
             mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(
                       rectOptions.getPoints().get(0), 10));
        break;
     ...
```

Najprv sa údaje načítajú z intentu. Pri spracovaní je kľúčové použitie funkcie *addPolyline*, ktorá pridá zoznam bodov do vrstvy mapy slúžiacej na kreslenie a vykreslí n-uholník.

Napokon vo forme animácie zabezpečíme aj desaťnásobné priblíženie mapy s centrom v prvom bode zoznamu.

Vykreslenie kruhu

Kruh sa opäť používa na ohraničenie vybranej oblasti a vykresľuje sa vo vrstve nad mapovými údajmi. Pre kruh definujeme stred a polomer, pričom polomer sa definuje v metroch.

Pre pripravenie údajov použijeme v hlavnej aktivite metódu:

```
public void onClickCircle(View view) {
    Intent mapIntent = new Intent(this, LocationActivity.class);
    LatLng myPosition = new LatLng(48.264300, 18.453336);
    mapIntent.putExtra("uloha",3);
    mapIntent.putExtra("position",myPosition);
    mapIntent.putExtra("polomer1",20000);
    mapIntent.putExtra("polomer2",40000);
    startActivity(mapIntent);
}
```

Pre vykreslenie najprv vytvoríme nastavenia pre kružnice (dvoma rôznymi farbami) a až následne ich pridáme do mapy.

```
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    LatLng myPosition;
    Intent vstupIntent = getIntent();
    Bundle data = vstupIntent.getExtras();
    int uloha = data.getInt("uloha");
         switch (uloha) {
           ...
           case 3: myPosition = (LatLng) data.get("position");
        int r1 = data.getInt("polomer1");
        int r2 = data.getInt("polomer2");
        CircleOptions circleOptions1 = new CircleOptions()
            .center(myPosition)
            .radius(r1)
            .strokeColor(Color.RED); // In meters

        CircleOptions circleOptions2 = new CircleOptions()
            .center(myPosition)
            .radius(r2)
            .strokeColor(Color.BLUE); // In meters
```

```
            mMap.addCircle(circleOptions1);
            mMap.addCircle(circleOptions2);

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(myPosition,
8));
            break;
          ...
```

Nájdenie cesty

Poslednou úlohou je nájsť cestu medzi dvoma bodmi.

Riešenie bude spočívať v nasledovnej postupnosti operácií:

- získanie údajov o polohe dvoch bodov,
- vytvorenie URL dotazu na získanie postupnosti bodov (na uliciach),
- odoslanie požiadavky,
- získanie odpovede v podobe json,
- rozparsovanie získaných údajov a prevod do pozícií,
- prekreslenie do mapy.

V prípade služieb Google však narazíme po určitom čase na informáciu:

> **Reminder:** To use the Directions API, you must include an API key with all API requests and you must enable billing on each of your projects.

Pre používanie *Directions* (služby vracajúcej cestu) je potrebné prejsť na vyšší level a umožniť systému pracovať s našou kreditnou kartou.

Táto služba nie je na úrovni *Google Maps* k dispozícii v žiadnej podobe zdarma, preto sme nútený nájsť inú alternatívu. Všetky vyššie uvedené postupy sú univerzálne a teda nám budú na osoh aj pri iných službách, prípadne ich dokážeme využívať aj v jednoduchých aplikáciách.

MapBox

Google Maps je len jednou z existujúcich služieb na poskytovanie mapového obsahu. Je možno majoritnou, avšak politika spoplatnenia v ostatných rokoch tlačí programátorov k iným prevádzkovateľom.

	OsmAnd	Mapbox	JawgMaps	HERE	GraphHopper	Google Maps
Free request/month	unlimited	50,000	50,000	250,000	15,000	28,000
Paid plans (from)	$1,63	$5	$250	$449	$48	$7/each 1000
Types of maps	tile, vector	tile, vector	tile, vector	tile, vector	vector only	tile, vector, satelite
Open-source data	✔	✔	✔	✘	✔	✘
Business geocoding	✔	✘	✔	✔	✔	✔
Navigation	✔	✔	✔	✔	✔	✔
Voice guidance	✔	✘	✘	✔	✘	✔
Traffic Insights	✘	✔	✘	✔	✘	✔
Layers	✔	✔	✘	✔	✔	✔
GPX tracks	✔	✔	✘	✔	✔	✔
Street view	✔	✘	✘	✔	✘	✔

Obrázok 60. Ponuka služieb pre prácu s mapovými podkladmi (marec 2020)

Okrem výberu z existujúcich prevádzkovateľov existujú aj možnosti hostingu vlastných „streamovacích“ serverov s open-source mapovými zdrojmi.

Pripojenie

Ako alternatívnu službu sme si pre potrebu tvorby mobilných aplikácií vybrali *MapBox*. Dôvody výberu tejto služby:

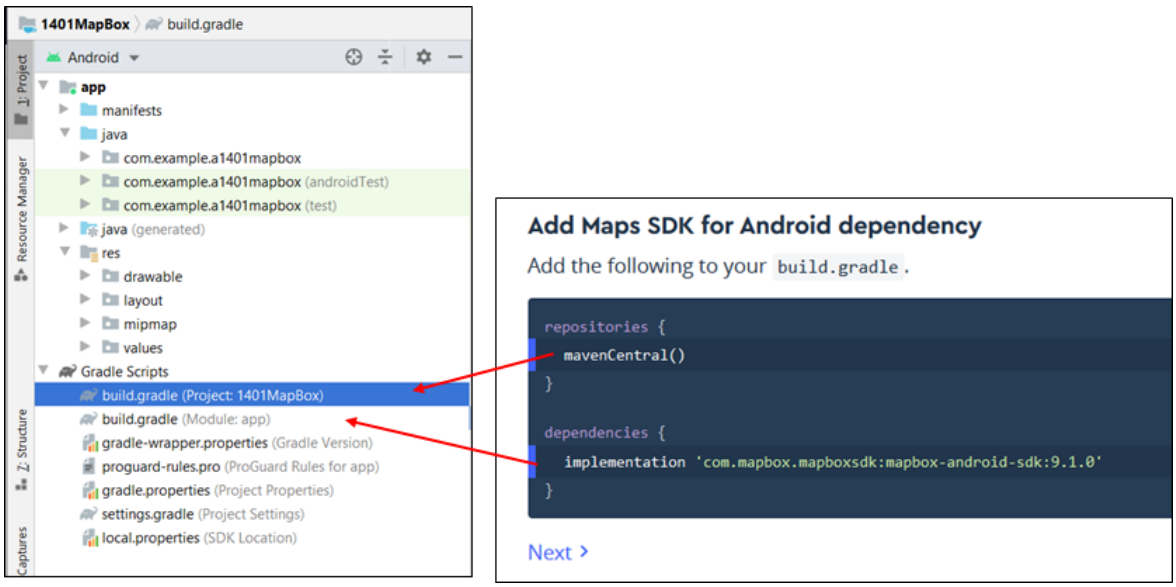
- relatívne najlacnejší,
- podpora satelitného aj mapového zobrazovania,
- podpora Android, iOS, web, Unity,

- popularita,
- jednoduchá integrácia do aplikácie,
- rozvíjajúca sa platforma,
- dobre spracovaný tutoriál + príklady.

Princíp používania všetkých externých služieb je veľmi podobný. Rovnako ako v prípade *Google Maps*, aj v prípade *MapBox*-u potrebujeme najprv získať API kľúč.

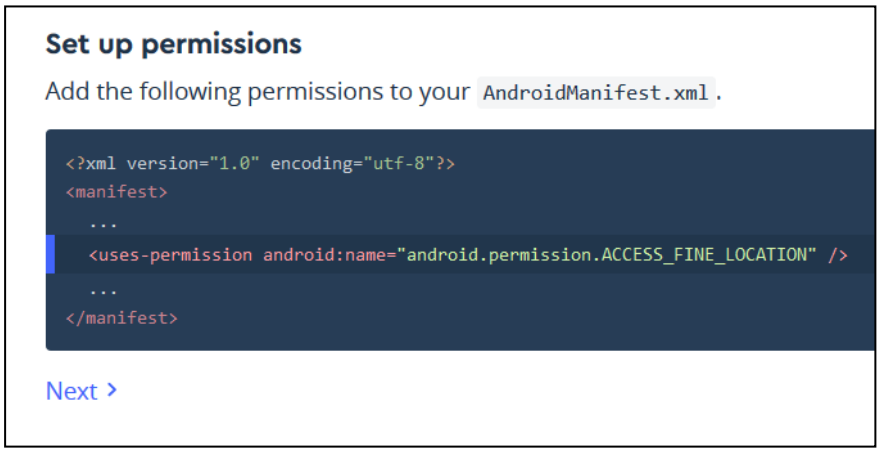
Po získaní tokenu pre použitie v Androide sú potrebné 3 kroky / úpravy aplikácie. Aktuálny popis nájdete aj na https://docs.mapbox.com/android/maps/overview/. V texte používame obrázky z tejto stránky s popisom, aby sme ukázali, že postup je naozaj veľmi jednoduchý.

Prvým krokom je napojenie projektu na adresu s *MapBox*-om. Pripojíme sa vložením dvojice závislostí do súborov *gradle*. Umiestnenie textu do sekcií vidíme na obrázku.



Obrázok 61. Zápis závislostí do buildovacích súborov

Do manifestu doplníme oprávnenie na prístup k polohe (Obrázok 62).



Obrázok 62. Doplnenie oprávnení do manifestu

Posledný krok postupu berme len ako orientačný - existujú rôzne spôsoby použitia *MapBox*-u a ukážeme si aj iné postupy.

Oživenie

Zobrazovacím prvkom mapy je prvok *MapView*, ktorý môžeme umiestniť do aktivity. Nastavíme mu rozmer na celú šírky a výšku aktivity, v ktorej je vložený a pomenujeme ho aby bol identifikovateľný v java kóde.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapActivity1">

    <com.mapbox.mapboxsdk.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"                    />

</androidx.constraintlayout.widget.ConstraintLayout>
```

V aktivite fungujeme na veľmi podobnom princípe ako s mapami *Google*. Callback v tomto prípade pridáme priamo do metódy *onCreate()*, nebudeme ho definovať ako samostatnú metódu – nemusíme používať rozhranie nad aktivitou.

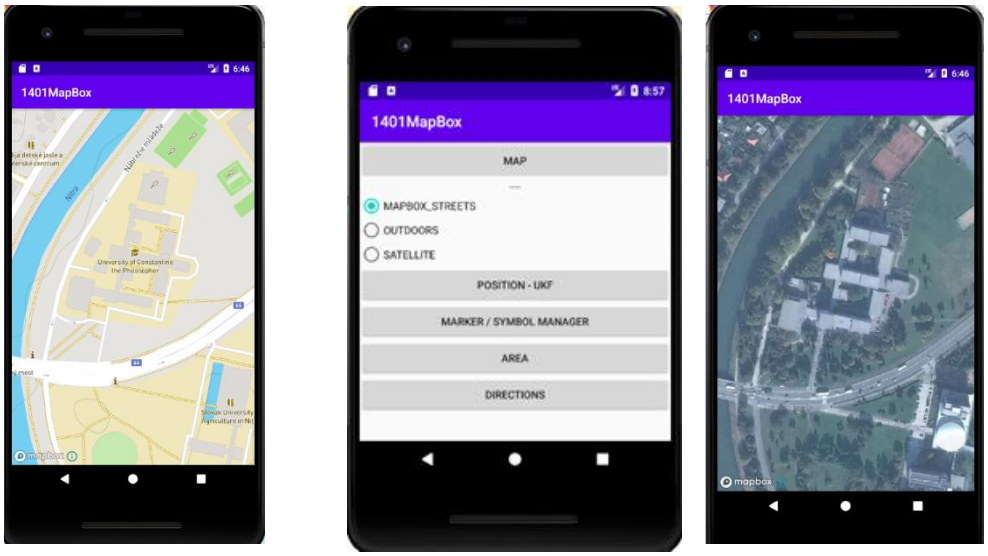
```
public class MapActivity1 extends AppCompatActivity {
    private MapView mapView;  // komponent pre prácu s mapou
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // pred načítaním xml sa vytvorí inštancia pre prácu s
           mapou – malým je API token/key
        Mapbox.getInstance(this, "pk.eyJ1IjoiaxxxxxxxxMu-
           1vUHckPANv5ccCq8Lw");
        setContentView(R.layout.activity_map1); // načítame
           dizajn
        mapView = findViewById(R.id.mapView);
        mapView.onCreate(savedInstanceState); // volanie
                          udalosti pri vytvorení mapy
        // obdoba async tasku, ktorá načíta mapu a po jej
           skončení volá onMapReady
        mapView.getMapAsync(new OnMapReadyCallback() {
          @Override
          public void onMapReady(@NonNull MapboxMap mapboxMap){
            mapboxMap.setStyle(Style.MAPBOX_STREETS,
                                new Style.OnStyleLoaded() {
              @Override
             // obdoba async tasku, pre načítanie štýlu,
               // po skončení vykoná kód tela
               public void onStyleLoaded(@NonNull Style style){
                // Mapa je pripravená, vzhľad mapy je načítaný
               }
            });
        }
    });
  }
}
```

Prvá aplikácia je hotová a dokonca funguje aj v emulátore.

Pozícia na mape

Budeme prezentovať rôzne spôsoby zobrazenie údajov v mape. Nastavíme pohľad na pozíciu UKF. Opäť si vytvoríme hlavnú aktivitu, z ktorej budeme spúšťať jednotlivé úlohy, ktoré v rámci prezentácie funkcionalít *MapBox*-u ešte pribudnú.

Pre prvú úlohu budeme prezentovať tri rôzne štýly vzhľadu, ktoré pošleme do zobrazovacej aktivity na základe výberu radiobuttonmi. Na základe štýlu sa vyberú typ mapových údajov, ktorý sa pošle používateľovi do aplikácie.



Obrázok 63. Obrazovky prvej aplikácie

Rôzne vzhľady mapy

Kliknutie na tlačidlo MAP spúšťa proces prípravy podkladov, ktoré sa následne v aktivite pre mapu preberú a obsah sa podľa nich prispôsobí.

```
public void onMapClick2(View view) {
    Intent i = new Intent(this, MapActivity2.class);
    String mapType = Style.MAPBOX_STREETS;
    RadioButton rb = (RadioButton)
     findViewById(R.id.radioButton2);
    if (rb.isChecked()) mapType = Style.OUTDOORS;
    rb = (RadioButton) findViewById(R.id.radioButton3);
    if (rb.isChecked()) mapType = Style.SATELLITE;
    i.putExtra("style", mapType);
    startActivity(i);
}
```

Style je trieda, ktorá definuje jednotlivé štýly ako konštanty typu *String*. Vybraný štýl odošleme a v mapovej aktivite teda len prečítame. V tomto prípade použijeme dynamické vytvorenie obsahu aktivity – podobne ako sme vytvárali *Canvas* => xml aktivitu v zásade ani nepotrebujeme.

```
public class MapActivity2 extends AppCompatActivity {
    private MapView mapView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Mapbox.getInstance(this, "pk.xxx");
        // nastavíme kameru na našu pozíciu a zazoomujeme
        // robíme len prípravu nastavení
       MapboxMapOptions options =
           MapboxMapOptions.createFromAttributes(this, null)
           // používame builder a „.“
                .camera(new CameraPosition.Builder()
                     .target(new LatLng(48.308526, 18.091698))
                     .zoom(16)
```

```
                        .build());
            // vytvoríme mapview na základe nastavení
            mapView = new MapView(this, options);
            mapView.onCreate(savedInstanceState);
            // asynchrónnym taskom načítame obsah mapy
            mapView.getMapAsync(new OnMapReadyCallback() {
                @Override
                public void onMapReady(@NonNull MapboxMap
                     mapboxMap) {
                    Intent i = getIntent();
                    String mapStyle = i.getStringExtra("style");
                    mapboxMap.setStyle(mapStyle, new
                           Style.OnStyleLoaded() {
                        @Override
                        public void onStyleLoaded(@NonNull Style
                                    style) {
                            // po načítaní môžeme vykonávať ďalšie
                                 operácie
                        }
                    });
                }
            });
            // nastavíme mapu ako obsah vytváranej aktivity
            setContentView(mapView);
        }
    }
```

Prostredníctvom intentu pošleme do aktivity vzhľad (štýl) mapy. Naším cieľom je nastaviť a nazoomovať mapu na konkrétnu pozíciu. Toto nastavenie môžeme urobiť predtým ako načítame mapu a pri načítaní ho aplikovať - výsledkom bude efektívnejšie (=rýchlejšie) načítanie požadovaného obsahu.

Opäť máme asynchrónne metódy, v rámci ktorých reagujeme na:

- situáciu, keď je mapa pripravená (*onMapReady*),
- situáciu, keď je nahraný/pripravený štýl (*onStyleLoaded*).

Markery

Jednoduchosť používania markerov v *MapBox*-e skončila v predchádzajúcich verziách Androidu, aktuálne je k dispozícii skupina Managerov, ktoré sa starajú o zobrazovanie rôznych objektov a grafiky v rôznych vrstvách:

- pozitíva: môžeme na jednotlivých vrstvách robiť čokoľvek (vlastné ikony, vlastná grafika),
- negatíva: kód je o čosi dlhší a občas ťažšie pochopiteľný.

Aby sme mohli používať modernejší prístup, je potrebné pridať knižnicu s anotáciami do dependencies v *build.gradle(Module.app)*.

```
...
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.mapbox.mapboxsdk:mapbox-android-
          sdk:9.1.0'
    implementation 'com.mapbox.mapboxsdk:mapbox-android-plugin
          -annotation-v9:0.8.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation
          'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso
          -core:3.2.0'
}
```

V layoute v tomto prípade nepoužijeme dynamické vytváranie v kóde, ale v xml súbore. View pre zobrazenie mapy vložíme do xml (**com.mapbox.mapboxsdk.maps.MapView**) a budeme k nemu pristupovať z aktivity – môžeme nastaviť aj pozíciu a zoom mapy.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:mapbox="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapActivity3">
    <com.mapbox.mapboxsdk.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        mapbox:mapbox_cameraTargetLat="48.308526"
        mapbox:mapbox_cameraTargetLng="18.091698"
        mapbox:mapbox_cameraZoom="14"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Pokiaľ používame v komponente nastavenia patriace *MapBox-u* – nie *android:* ale *mapbox:*, je nutné uviesť definíciu *MapBox*-u v nastaveniach layoutu (5. riadok).

Pre písanie java kódu si tiež ukážeme alternatívny prístup:

- metódu pre callback nebudeme vkladať do *onCreate()*, vytvoríme ju ako samostatnú v rámci aktivity,
- je potrebné implementovať na aktivitu rozhranie *OnMapReadyCallback*.

```
public class MapActivity3 extends AppCompatActivity implements
OnMapReadyCallback {
    private MapView mapView;
    // private MapboxMap mapboxMap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        Mapbox.getInstance(this, "pk.xxx");
        setContentView(R.layout.activity_map3);
        // Inicializácia MapView-u
        mapView = findViewById(R.id.mapView);
        mapView.onCreate(savedInstanceState);
        mapView.getMapAsync(this);
    }
    @Override
    public void onMapReady(@NonNull final MapboxMap mapboxMap)
 {
        ...
    }
}
```

Po pripravení mapy môžeme:

- nastaviť štýl a po nastavení štýlu,
- použiť zastaranú metódu *addMarker*,
- na zobrazenie markera s titulkom UKF na pozíciu UKF.

```
@Override
public void onMapReady(@NonNull final MapboxMap mapboxMap) {
     mapboxMap.setStyle(Style.MAPBOX_STREETS, new
       Style.OnStyleLoaded() {
         @Override
         public void onStyleLoaded(@NonNull Style style) {
             mapboxMap.addMarker(new MarkerOptions()
                     .position(new LatLng(48.308526, 18.091698))
                     .title("UKF"));
         }
      }
}
```

Alebo po novom: použijeme *SymbolManager* na vytvorenie ikon na mape. Vytvorením sa ikona umiestni na mapu, ktorú sme zadali pri vytváraní managera.

```
public void onMapReady(@NonNull final MapboxMap mapboxMap) {
   mapboxMap.setStyle(Style.MAPBOX_STREETS, new
      Style.OnStyleLoaded() {
       @Override
       public void onStyleLoaded(@NonNull Style style) {
          SymbolManager symbolManager = new SymbolManager(
                                    mapView, mapboxMap, style);
          // nastavíme, aby si ikony navzájom neprekážali
          symbolManager.setIconAllowOverlap(true);
          symbolManager.setIconIgnorePlacement(true);
          // pridáme ikonu požiarnej stanice na pozíciu s daným
             zväčšením
          Symbol symbol = symbolManager.create(new
             SymbolOptions()
                      .withLatLng(new LatLng(48.308520, 18.093))
                      .withIconImage("fire-station-15")
                     .withIconSize(2.0f));
       }
  });
}
```

Ohraničenie oblasti

Ohraničenie oblasti predstavuje bežný prvok práce s mapami – jeho aplikácia je pomerne jednoduchá.

Pohľad na mapu v aktivite zostane nastavený a nazoomovaný na náš centrálny bod.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:mapbox="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapActivity4">
    <com.mapbox.mapboxsdk.maps.MapView
```

```
            android:id="@+id/mapView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            mapbox:mapbox_cameraTargetLat="48.308526"
            mapbox:mapbox_cameraTargetLng="18.091698"
            mapbox:mapbox_cameraZoom="14"
            />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

Inicializácia aktivity je rovnaká ako v predchádzajúcom prípade pridáme len zoznam bodov ohraničenia.

```
public class MapActivity4 extends AppCompatActivity implements
OnMapReadyCallback {
            private MapView mapView;
            List<Point> myArea;
            @Override
            protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            Mapbox.getInstance(this, "pk.xxx");
            setContentView(R.layout.activity_map4);
            naplnBody();
            // Inicializácia mapView-u
            mapView = findViewById(R.id.mapView);
            mapView.onCreate(savedInstanceState);
            mapView.getMapAsync(this);
        }
            private void naplnBody() {
                // vytvor list
                myArea = new ArrayList();
                myArea.add(Point.fromLngLat(18.088, 48.307));
                myArea.add(Point.fromLngLat(18.088, 48.3095));
                myArea.add(Point.fromLngLat(18.095, 48.3095));
                myArea.add(Point.fromLngLat(18.095, 48.307));
                myArea.add(Point.fromLngLat(18.088,48.307));
            }
```

Po nahratí štýlu sa v callbacku vytvorí nová vrstvu, pre ktorej obsah nastavíme vlastnosti.

```
@Override

public void onMapReady(@NonNull final MapboxMap mapboxMap) {
   mapboxMap.setStyle(Style.MAPBOX_STREETS, new
           Style.OnStyleLoaded() {
    @Override
    public void onStyleLoaded(@NonNull Style style) {
         // nová vrstva s čiarkovanou čiarou
        style.addLayer(new LineLayer
              ("linelayer", "line-source").withProperties(
        PropertyFactory.lineDasharray(new Float[]
                                          {0.1f, 2f}),
        PropertyFactory.lineCap(Property.LINE_CAP_ROUND),
        PropertyFactory.lineJoin(Property.LINE_JOIN_ROUND),
         PropertyFactory.lineWidth(5f),
         PropertyFactory.lineColor(Color.parseColor("#e55e5e"))
        ));
        ...

        Do tejto vrstvy umiestnime čiary – vytvoríme objekt GeoJsonSource, ktorý je zdrojom pre objekty vykresľované do mapy – vytvorené pole sa transformuje na geo-dáta.

        ...
        // transformácia poľa na geodata
        style.addSource(new GeoJsonSource("line-source",
             FeatureCollection.fromFeatures(new Feature[]
                    {Feature.fromGeometry(
                       LineString.fromLngLats(myArea)
        )})));
    }
});
```

Directions

Služby typu *directions* poskytujú nástroj na hľadanie cesty medzi dvoma pozíciami často s možnosťou prechodu cez ďalšie body.

Vyhľadávanie cesty patrí medzi pokročilé funkcie, ktoré si vyžadujú strojový čas na počítanie a vytvorenie výsledku podľa aktuálnych požiadaviek používateľa. Je k dispozícii v balíku Directions, ktorý využíva inú API sadu, preto je potrebné pridať ho do dependencies v *build.gradle(Module.app)*.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.mapbox.mapboxsdk:mapbox-android
     -sdk:9.1.0'
    implementation 'com.mapbox.mapboxsdk:mapbox-android-plugin
     -annotation-v9:0.8.0'
    Implementation 'com.mapbox.mapboxsdk:mapbox-sdk
     -services:5.1.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation
     'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso
     -core:3.2.0'
}
```

Pri vytváraní aplikácie môžete naraziť na problém s verziami týchto SDK a verziou Javy, pričom bolo uvádzané, že sú podporované len verzie od N vyššie. Doplnením nastavení pre kompilátor (v *build.gradle(Module.app)*) sa problém vyrieši.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
```

V prvom kroku definujeme konštanty pre vrstvy a základné objekty na prácu s mapou.

```
public class MapActivity5 extends AppCompatActivity implements
OnMapReadyCallback {
        private static final String ROUTE_LAYER_ID = "route-layer
         -id";
        private static final String ROUTE_SOURCE_ID = "route
         -source-id";
        private static final String ICON_SOURCE_ID = "icon-source
         -id";
        private MapView mapView;
        private DirectionsRoute currentRoute;
        private MapboxDirections client;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            Mapbox.getInstance(this, "pk.xxx");
            setContentView(R.layout.activity_map5);
            // map view
            mapView = findViewById(R.id.mapView);
            mapView.onCreate(savedInstanceState);
            mapView.getMapAsync(this);
        }
```

Po získaní mapy definujeme definuje začiatok a koniec cesty. Dokumentácia uvádza že možno definovať aj viac ako 20 prechodových bodov. Pripravíme formát pre dotaz a vrstvu pre vykreslenie.

```
@Override
public void onMapReady(@NonNull final MapboxMap mapboxMap) {
    mapboxMap.setStyle(Style.MAPBOX_STREETS, new
     Style.OnStyleLoaded() {
        @Override
        public void onStyleLoaded(@NonNull Style style) {
            // štart a cieľ
            Point origin = Point.fromLngLat(18.090272,
            48.307561);
            Point destination = Point.fromLngLat(18.094539,
            48.312398);
            // priprav geodáta na získanie cesty
            initSource(style, origin, destination);
            // priprav vrstvu na kreslenie
            initLayers(style);
            // získaj cestu z Mapbox Directions API
            getRoute(mapboxMap, origin, destination);
        }
    });
}
```

Metóda *initSource()* zabezpečí, že do premennej (objektu) *loadedMapStyle* sa umiestnia potrebné nastavenia údaje o pozícii začiatku a konca.

```
private void initSource(@NonNull Style loadedMapStyle, Point
 origin, Point destination) {
    loadedMapStyle.addSource(new
      GeoJsonSource(ROUTE_SOURCE_ID));
    GeoJsonSource iconGeoJsonSource = new
      GeoJsonSource(ICON_SOURCE_ID,
    FeatureCollection.fromFeatures(new
          Feature[] {
```

```
        Feature.fromGeometry(Point.fromLngLat(origin.longitude(),
origin.latitude())),
Feature.fromGeometry(Point.fromLngLat(destination.longitude(),
destination.latitude()))}));
            loadedMapStyle.addSource(iconGeoJsonSource);
        }
```

Metóda *initLayers()* definuje novú vrstvu a v nej parametre pre vykresľovanú čiaru, resp. vrstvu čiar. Informácie opäť prejdú do nastavení mapy cez objekt *loadedMapStyle*.

```
    private void initLayers(@NonNull Style loadedMapStyle) {
        LineLayer routeLayer = new LineLayer(ROUTE_LAYER_ID,
         ROUTE_SOURCE_ID);
        // pridá LineLayer do map. Thistáto vrstva zobrazí
           directions route.
        routeLayer.setProperties(
                lineCap(Property.LINE_CAP_ROUND),
                lineJoin(Property.LINE_JOIN_ROUND),
                lineWidth(5f),
                lineColor(Color.parseColor("#FF0000"))
        );
        loadedMapStyle.addLayer(routeLayer);
    }
```

Metóda *getRoute()* je kľúčová pre získanie potrebných informácií. Vytvorí požiadavku na návrat údajov a na tomto mieste možno konfigurovať či má ísť o trasu pre peších, cyklistov alebo automobily.

```
private void getRoute(final MapboxMap mapboxMap, Point origin, Point
destination) {
        client = MapboxDirections.builder()
                .origin(origin)
                .destination(destination)
                .overview(DirectionsCriteria.OVERVIEW_FULL)
                .profile(DirectionsCriteria.PROFILE_DRIVING)
```

```
                .accessToken(getString(R.string.access_token))
                .build();
...
```

Asynchrónnou požiadavkou si vypýta údaje. Ak príde prázdny výsledok alebo výsledok bez pozícií, skončíme, inak je výsledkom cesta – *currentRoute*, u ktorej vieme zistiť napr. aj dĺžku.

```
client.enqueueCall(new Callback<DirectionsResponse>() {
    @Override
    public void onResponse(Call<DirectionsResponse> call,
                           Response<DirectionsResponse> response) {
        // ak nevráti žiaden výsledok, koniec
        if (response.body() == null) {
            return;
        } else if (response.body().routes().size() < 1) {
            return;
        }
        // východiskový bod
        currentRoute = response.body().routes().get(0);
        // na ilustráciu - vzdialenosť
        Toast.makeText(MapActivity5.this, "dlzka: " +
           currentRoute.distance(),
        Toast.LENGTH_SHORT).show();
```

Ak je všetko v poriadku, t.j. *mapboxMap* stále existuje, tak po načítaní štýlu získame prístup k zdroju pre kreslenie čiar a čiary z *CurrentRoute* doň pošleme na vykreslenie

```
if (mapboxMap != null) {
    mapboxMap.getStyle(new Style.OnStyleLoaded() {
        @Override
        public void onStyleLoaded(@NonNull Style style) {
            // Načíta a aktualizuje zdroj zobrazujúci trasu
                trasy
            GeoJsonSource source =
```

```
                    style.getSourceAs(ROUTE_SOURCE_ID);
                // Vytvorí postupnosť LineString s geometriou trasy
                // resetuje zdroj GeoJSON pre zdroj trasy LineLayer
                if (source != null) {
                    source.setGeoJson(LineString.fromPolyline(
                        currentRoute.geometry(),
                            PRECISION_6));
                }
            }
        });
    }
```

Napokon musíme implementovať metódu *onFailure* pre *client.enqueuCall*.

```
            @Override
            public void onFailure(Call<DirectionsResponse> call,
                                  Throwable throwable) {
                Toast.makeText(MapActivity5.this, "Error: " +
                               throwable.getMessage(),
                    Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Záver

S využitím *MapBox*-u je možné vytvoriť rovnaké aplikácie ako s použitím *Google Maps*, pričom najmä v prípade hľadania trasy je postup omnoho jednoduchší.

MapBox poskytuje množstvo ďalších funkcií, v mnohých ohľadoch je práca s ním aktuálne jednoduchšia ako v s mapami Google. Okrem iného umožňuje vytvárať a vkladať do mapového zdroja aj rôzne typy objektov, ktoré môžu byť zdieľané s komunitou, podporuje aj web a je tak vhodným nástrojom pre tvorbu vlastných aplikácií nad viacerými platformami.

4.5 Základy internetu vecí, UPJŠ v Košiciach (František Galčík, Miroslav Opiela)

4.5.1 Anotácia a stručná osnova predmetu

Predmet Základy internetu vecí ponúka stručný prierez oblasťou internetu vecí a jej základnými komponentami od fyzických senzorov a aktuátorov, cez lokálnu komunikáciu a spracovanie, až k službám v cloude a strojovému učeniu. Dôraz je kladený na to, aby predmet nadväzoval na znalosti a skúsenosti z iných predmetov študijného programu, prípadne tieto predmety dopĺňal. Tomu sú prispôsobené príklady, miera detailu, ale aj použité nástroje. Popri nízkoúrovňových protokoloch, aplikačných protokoloch a softvérových riešeniach sa predstavujú aj hardvérové súčasti, ako sú napríklad jednodoskové počítače, mikrokontroléry, senzory a aktuátory.

Základom pre absolvovanie predmetu sú základné vedomosti z fyziky a základná znalosť programovania. Predpokladá sa znalosť programovania v jazyku C/C++ v prípade programovania mikrokontrolérov a v jazyku Java pri ďalších demonštráciách. Jazyk Java bol zvolený ako jazyk, ktorý je študentom najviac blízky s ohľadom iné predmety v študijnom programe. V spolupráci s IT sektorom je priestor venovaný prípadovým štúdiám a aktuálnym trendom v tejto oblasti. Workshopy aj praktické úlohy počas kurzu poskytujú možnosť vyskúšať si reálne aplikácie nadobudnutých poznatkov.

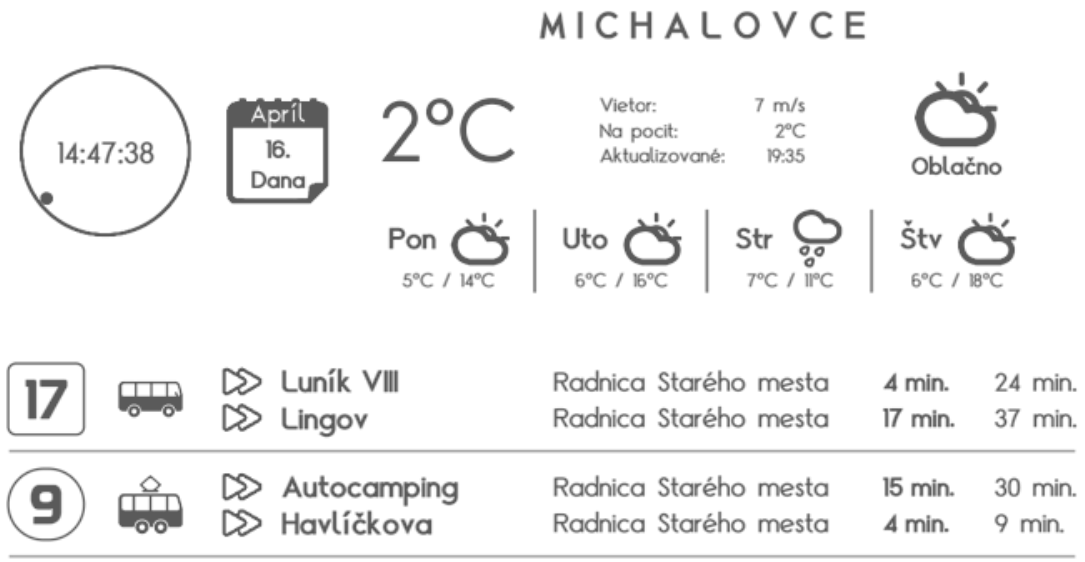
V úvode kurzu sa predstavuje prehľad základných pojmov a niektorých aplikácii z oblasti internetu vecí doplnený o pripomenutie znalostí z fyziky týkajúcich sa jednosmerného prúdu a práce s multimetrom. V prvej časti kurzu sa pracuje s programovateľným zariadením s mikrokontrolérom Arduino pokrývajúc témy digitálnych a analógových senzorov a aktuátorov, programovanie Arduino zariadenia, prácu s pinmi, PWM, pull-up a pull-down rezistory, asynchrónnu a synchrónnu sériovú komunikáciu, UART, RS-232, RS-485, komunikáciu s digitálnymi senzormi a protokoly SPI, I2C a 1-Wire.

Druhá časť kurzu sa venuje lokálnemu spracovaniu s hardvérovým zameraním na jednodoskový počítač Raspberry Pi, jeho konfiguráciu a komunikáciu s pinmi. Okrem toho sú predstavené aplikačné protokoly pre IoT CoAP a MQTT a softvérový nástroj Node-RED s dôrazom na spracovanie dát a vizualizáciu meraní zo senzorov. Predstavenie základných konceptov a produktov výpočtov v cloude, konkrétne na platforme Amazon Web Services, dopĺňa detailnejšie vysvetlenie a demonštrácia IoT aplikácie v cloude vrátane autentifikácie

a autorizácie, registrácie a reprezentácie vecí a nastavovania pravidiel na ďalšie spracovanie, napr. vo forme notifikácií. Okrem základných pojmov a modelov strojového učenia, je priestor venovaný strojovému učeniu v cloude, aplikačným príkladom a práci s dátami.

4.5.2 Aplikačný príklad

V aplikačnom príklade predstavujeme využitie viacerých konceptov IoT v študentskom projekte Zirro. Ide o smart zrkadlo, ktoré ponúka digital signage riešenie v podobe obrazovky na zobrazovanie informácii zabudovanej v zrkadle a reagujúce na podnety z okolitého prostredia získavané prostredníctvom senzorov. Príkladom použitia je situácia, keď človek opúšťa vybranú budovu a pri odchode sa pozrie do zrkadla, v ktorom pozoruje okrem seba aj viaceré podstatné informácie, ako je aktuálna predpoveď počasia, odchody autobusov z najbližšej zastávky a pod.



Obrázok 64. Ukážka poskytnutých informácii na obrazovke zabudovanej v zrkadle

Hardvérové požiadavky na takéto riešenie zahŕňajú obrazovku, špeciálne polopriepustné zrkadlo a rám s konštrukciou. Vo vnútri je jednodoskový počítač Raspberry Pi s napojením na obrazovku, kameru a mikrokontrolér na prácu so senzormi. Použilo sa viacero senzorov: senzor odtlačkov prstov, senzor vzdialenosti, senzor pohybu, RFID čítačka a ďalšie. Na implementáciu softvérovej časti sa zvolila Java a JavaFX. Výmena dát medzi softvérovými komponentami riešenia (zobrazovacia aplikácia, scraper údajov z webových zdrojov, riadiaca aplikácia, atď.) je realizovaná protokolom MQTT, pričom ako broker sa využil projekt mosquitto. Zo softvérového hľadiska je riešenie navrhnuté tak, aby umožnilo prepojenie, spoluprácu a jednotné manažovanie viacerých zobrazovacích zariadení (napríklad zrkadiel, ale aj monitorov).

V základnej verzii poskytuje zrkadlo zobrazovanie všeobecných užitočných informácii. Tie sú v pravidelných intervaloch sťahované z príslušných zdrojov, napr. shmu.sk pre informácie o počasí a imhd.sk pre odchody mestskej hromadnej dopravy. Dáta sú následne spracované, upravené a zobrazené v príslušnej forme na obrazovke. Zobrazenie je viazané k umiestneniu zrkadla, teda zobrazuje odchody autobusov z najbližších zastávok, počasie v danom meste a pod. Rozloženie komponentov sa dá jednoducho meniť, čo umožňuje pridávanie informácii v ďalších fázach projektu. Monitor sa automaticky prepína do a z úsporného režimu na základe údajov z detektora pohybu. To zároveň aj aktivuje napájanie senzora odtlačkov prstov. Na základe merania vzdialenosti objektu pred zrkadlom (detekcia osoby pred zrkadlom) sa mení režim zobrazovania obsahu. Prototyp takéhoto zariadenia v rámci projektu Zirro a jeho nasadenie získali pozitívnu spätnú väzbu od návštevníkov budovy, v tomto prípade študentov a zamestnancov univerzity.

Ďalšie vylepšenia aplikácie zahŕňajú senzory zabudované v konštrukcii zrkadlového rámu. Údaje z kamery sú využiteľné pre študentské aplikácie a projekty z oblasti počítačového videnia, napríklad z oblasti rozpoznávania gest, pomocou ktorých sa umožní upravovať zobrazovaný obsah, resp. iná forma prirodzenej interakcie so zobrazovacou aplikáciou. Prototyp zrkadla tak má potenciál aj na rozvoj medzipredmetových väzieb. Hlavným prínosom a zároveň výzvou je prispôsobený obsah pre konkrétneho používateľa. Príkladom je využitie ISIC karty alebo odtlačku prsta na autentifikáciu a následne zobrazenie personalizovaného obsahu (napríklad na zobrazenie aktuálneho rozvrhu pre daného študenta).

4.6 Systémové programovanie, UPJŠ v Košiciach (Peter Pisarčík)

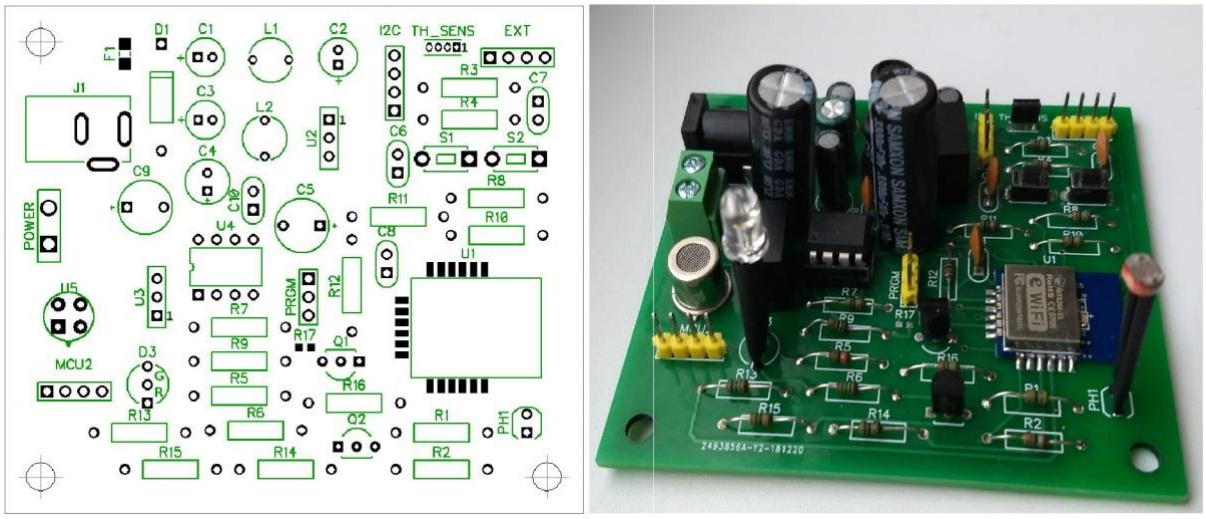
4.6.1 Anotácia a stručná osnova predmetu

Fenomén internetu vecí posúva hranice počtu pripojených zariadení k internetu. Zariadenia sa vzájomne spájajú častokrát aj bez využitia bezpečnostných protokolov. Informačná bezpečnosť je preto jedným z kľúčových faktorov pri komunikácii zariadení prostredníctvom internetu. Kurz poskytuje informácie o základných komunikačných protokoloch, integrácii zariadení, aplikačných projektoch, zelených riešeniach, smart domácnostiach a podobne.

Obsah predmetu Systémové programovanie je zameraný aj na získanie vedomostí o nízkoúrovňových aplikačných rozhraniach OS Windows a Linux. Popisujeme vývojové nástroje pre internet vecí. Prezentujeme systémové volanie a glibc, súborové operácie, súborový systém, signál, program a proces, vlákno a synchronizačné objekty, rozšírené IPC, socket a šifrovanie, zariadenia, virtuálne súborové systémy, dynamický modul jadra OS Linux.

4.6.2 Aplikačný príklad

Pre účely výučby predmetu Systémové programovanie bola navrhnutá vývojová doska, ktorá umožňuje výučbu programovania mikrokontrolérov. Táto doska je navrhnutá tak, aby poskytla študentom získanie skúseností pri programovaní 8-bitových a 32-bitových mikrokontrolérov a súčasne, aby sa študenti naučili využívať štandardné rozhrania vstupu a výstupu, pracovať so systémom prerušení, ovládať časovače a pochopili časovú distribúciu v mikrokontroléroch, ako aj nízkoúrovňové komunikačné protokoly, ako je napr. UART, I2C, OneWire a rovnako vysokoúrovňové protokoly, t. j. komunikácia prostredníctvom TCP/IP cez WiFi. Súčasne doska poskytuje výučbu programovania v prostredí Real-Time operačných systémov, konrétne FreeRTOS.



Obrázok 65. Vývojová doska na výučbu programovania mikrokontrolérov

Vývojovú dosku je možné napájať prostredníctvom dvoch konektorov: jedného štandardného [J1] (vidlica,5,5/2,1mm) a jednej svorkovnice [POWER]. Konektory sú priamo na doske prepojené, preto je možné ich využiť na prepojenie viacerých dosiek z jedného napájacieho zdroja. Dosku je možné napájať napätím v rozsahu od 6.5 do 32 voltov. Na doske sú osadené dva napäťové meniče (DC/DC), ktoré zo vstupného napätia vyrábajú stabilizovaných 5 voltov a 3.3 volta. Päť voltov je určených pre ATTiny mikrokontrolér a napájanie plynového senzora. 3.3V je určených pre napájanie WiFi modulu s integrovaným obvodom od spoločnosti Espressif ESP8266ex a pre napájanie senzora vlhkosti a teploty, senzora jasu, a riadenie dvojfarebnej diódy. Doska je na vstupe chránená diódou a 1A rýchlou poistkou.

Ako bolo uvedené vyššie, jadro vývojovej dosky tvorí modul s integrovaným obvodom ESP8266ex. Ide o WiFi modul, ktorý obsahuje 32 bitový ARM mikrokontrolér. Tento mikrokontrolér je možné programovať prostredníctvom SDK od spoločnosti Espressif. Pre účely výučby bola zvolená varianta SDK s podporou a Real-Time operačného systému konkrétne freeRTOS.

FreeRTOS je navrhnutý s cieľom, aby bol jednoducho implementovateľný a súčasne efektívny. Jadro tohto operačného systému je napísané v jazyku C, avšak z hľadiska rýchlosti je nevyhnutné, aby časti kódu boli napísané v assembleri. FreeRTOS operačný systém

poskytuje volania pre tvorbu vlákien alebo úloh, synchronizačných mechanizmov (semafór, mutex), poskytuje tvorbu softvérových časovačov a podobne.

FreeRTOS umožňuje vytvárať úlohy (vlákna) s rôznou prioritou. To znamená, že úlohy s vyššou prioritou majú vždy prednosť pred štandardnými úlohami (t. j. úlohami s nižšou prioritou), čo nazývame prioritné plánovanie. Štandardne sú jednotlivé objekty (úloha, mutex a iné) vo freeRTOS alokované (prideľované) dynamicky (statická alokácia je samozrejme tiež možná), to napríklad znamená, ak sa vytvára úloha, jadro freeRTOS sa postará o to, aby úloha bola vytvorená v určitom pamäťovom mieste a zase následne po jej ukončení, aby bola z tohto pamäťového miesta odstránená. Vo freeRTOS existuje päť základných schém pre dynamickú alokáciu pamäte. Jednotlivé schémy sa od seba líšia spôsobom ako je pamäť alokovaná a či je následne aj uvoľnená.

Je potrebné súčasne podotknúť, že FreeRTOS nie je ako iné operačné systémy určené pre každodennú prácu, ako je napríklad Linux, MacOS, či Windows. V tomto operačnom systéme nenájdeme štandardné funkcie, ktoré väčšinou obsahujú vyššie uvedené operačné systémy, ako sú ovládače zariadení, rozšírená správa pamäte, používateľské účty, či správa siete. Dôraz v Real-Time operačných systémoch je kladený na rýchlosť vykonávania úloh a ich presné načasovanie s determinovaním času štartu a ukončenia vykonávania konkrétnych úloh.

FreeRTOS implementuje viaceré obsluhy prerušení, ktoré umožňujú paralelné vykonávanie úloh, takzvaný multitasking. Túto úlohu má v operačnom systéme na starosti plánovač. Vo freeRTOS je možné ako plánovanie kooperatívne, tak aj plánovanie preemprívne. Zabudovaný plánovač úloh automaticky na základe časového intervalu prepína beh jednotlivých úloh prostredníctvom round-robin plánovacej schémy (časové okno pre prepínanie kontextu tzv. quantum je zväčša jedna milisekunda alebo 10 milisekúnd).

Medzi základné výhody freeRTOS patrí:

- softvér s otvoreným zdrojovým kódom a teda aj viac recenzentov kódu,
- výborná dokumentácia a dostupná literatúra,
- malá pamäťová stopa (tzv memory footprint a overhead),
- rýchle a presné vykonávanie úloh,
- možnosť využitia v aplikáciách na batériu,
- vhodný pre hobby, aj pre profesionálov, ktorí vytvárajú komerčné produkty,
- plánovač úloh umožňuje preemptívne aj kooperatívne vykonávanie úloh.

4.7 Základy internetu vecí, TUKE v Košiciach (Miroslav Biňas, Tomáš Kanócz)

4.7.1 Anotácia a stručná osnova predmetu

Cieľom predmetu je oboznámenie sa s konceptom Internetu vecí. Študent získa širokospektrálny prehľad o architektúre Internetu vecí, jeho funkčných stavebných blokoch, senzoroch, akčných členoch, softvérovom programovaní a integrovaní s fyzických svetom, lokálnom spracovaní na hranici siete, bezpečnom a efektívnom prenose dát cez rôzne sieťové protokoly, ukladaní a spracovaní dát v cloude, riadení na základe dát, ako aj podnikateľské nápady v danej oblasti. Študent sa naučí kreatívne navrhnúť šikovné systémy od jedného konca k druhému pre Internet vecí a prepojiť fyzický svet so softvérovým svetom metódou rýchleho prototypovania.

4.7.2 Aplikačný príklad

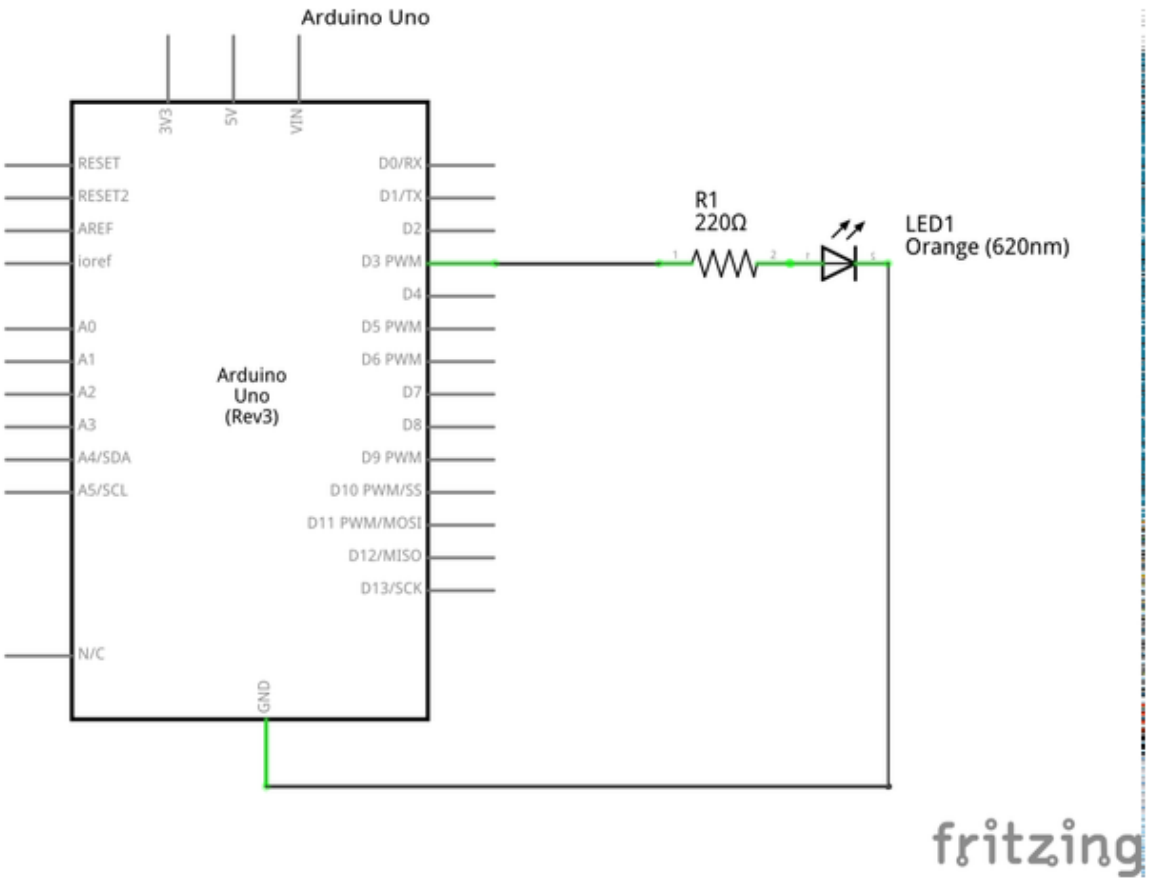
V aplikačnom príklade si predstavíme úlohu, ktorá je súčasťou jedného z prvých cvičení tohto predmetu. Cieľom je vytvoriť a naprogramovať semafor, ktorým sa riadi premávka na križovatkách. Semafor pracuje v dvoch režimoch. V nočnom režime svieti len oranžové svetlo a v dennom režime svietia všetky tri farby. K prepnutiu režimu dochádza po stlačení príslušného tlačidla. Študenti môžu využívať Arduino, alebo použiť niektorý z dostupných online nástrojov, akým je napríklad circuits.io.

Riešenie úlohy začína rozblikaním jednej oranžovej LED diódy. Blikanie oranžového svetla bude reprezentovať nočný režim semaforu. Cez rezistor pripojíme na PIN č. 3 oranžovú LED diódu. Pri zapájaní je potrebné dbať na správnu polaritu diódy a tiež nezabudnúť na umiestnenie odporu pred samotnú diódu.

Vo funkcii `setup()` je potrebné inicializovať PIN č. 3 na výstupný. Používanie globálnych premenných sa neodporúča. Na premenovanie PIN č.3 je vhodné použiť makro.

```
#define LED_ORANGE  3

void setup() {
    pinMode(LED_ORANGE, OUTPUT);
}
```

Vytvoríme funkciu `night_mode()`, po zavolaní ktorej dôjde k spusteniu nočného režimu. Funkcia bude reprezentovať jeden cyklus nočného režimu, pričom na jednu sekundu sa svetlo rozsvieti a následne na 1 sekundu svetlo zhasne. Volanie tejto funkcie umiestnime do funkcie `loop()` sketch-u. Táto funkcia nebude mať žiadne parametre, ani nebude vracať žiadnu hodnotu.



Obrázok 66. Schéma zapojenia semafora pre nočný režim

```
void night_mode(){
    digitalWrite(LED_ORANGE, HIGH);
    delay(1000);
    digitalWrite(LED_ORANGE, LOW);
    delay(1000);
}

void loop() {
    night_mode();
}
```

Ak sme postupovali správne, po nahratí programu do Arduina sa oranžová LED dióda v sekundových intervaloch rozsvecuje a zhasína. V prípade, ak nastane problém s prenesením programu do Arduina, je potrebné skontrolovať správne nastavenie portu a zvolenie správneho typu dosky.

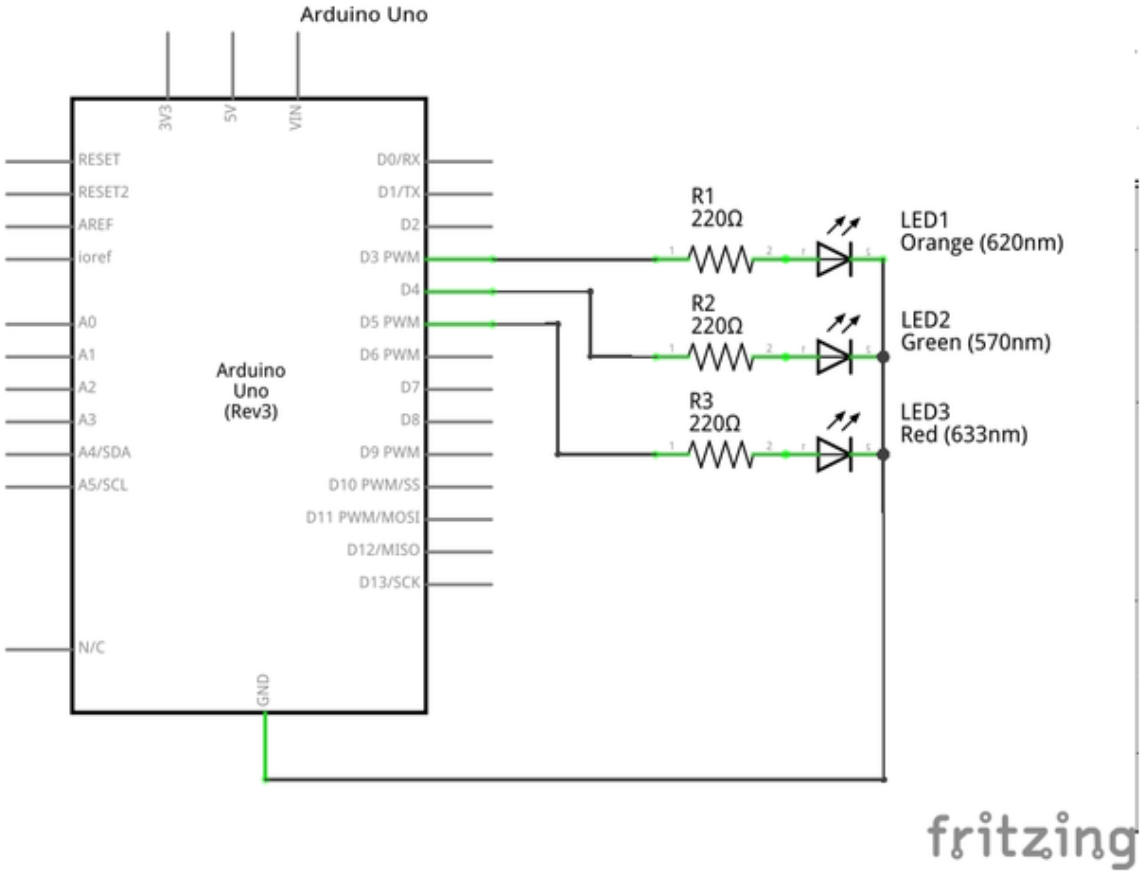
V ďalšom kroku sa presunieme na usmerňovanie premávky pomocou všetkých troch farieb. K oranžovej LED dióde zapojíme podobným spôsobom aj zelenú (PIN č. 4) a červenú (PIN č. 5) diódu.

Inicializujme PIN č. 4 a PIN č. 5 vo funkcii `setup()`.

```
#define LED_ORANGE  3
#define LED_GREEN   4
#define LED_RED     5

void setup() {
    pinMode(LED_ORANGE, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    pinMode(LED_RED, OUTPUT);
}
```

Vytvoríme funkciu `day_mode()`, po zavolaní ktorej sa semafor prepne do denného režimu. Funkcia bude reprezentovať jeden cyklus denného režimu. Tento režim bude mať nasledujúce správanie:

- Po zapnutí, resp. zavolaní funkcie (v čase 0 sekúnd) bude na semafore svietiť len červená LED dióda.
- V čase 5 sekúnd začne okrem červenej LED diódy svietiť aj oranžová.
- V čase 7 sekúnd začne svietiť len zelená LED dióda.
- V čase 12 sekúnd začne svietiť len oranžová LED dióda.
- V čase 14 sekúnd sa ukončí cyklus denného režimu zhasnutím oranžovej LED diódy.



Obrázok 67. Schéma zapojenia semafora pre denný režim

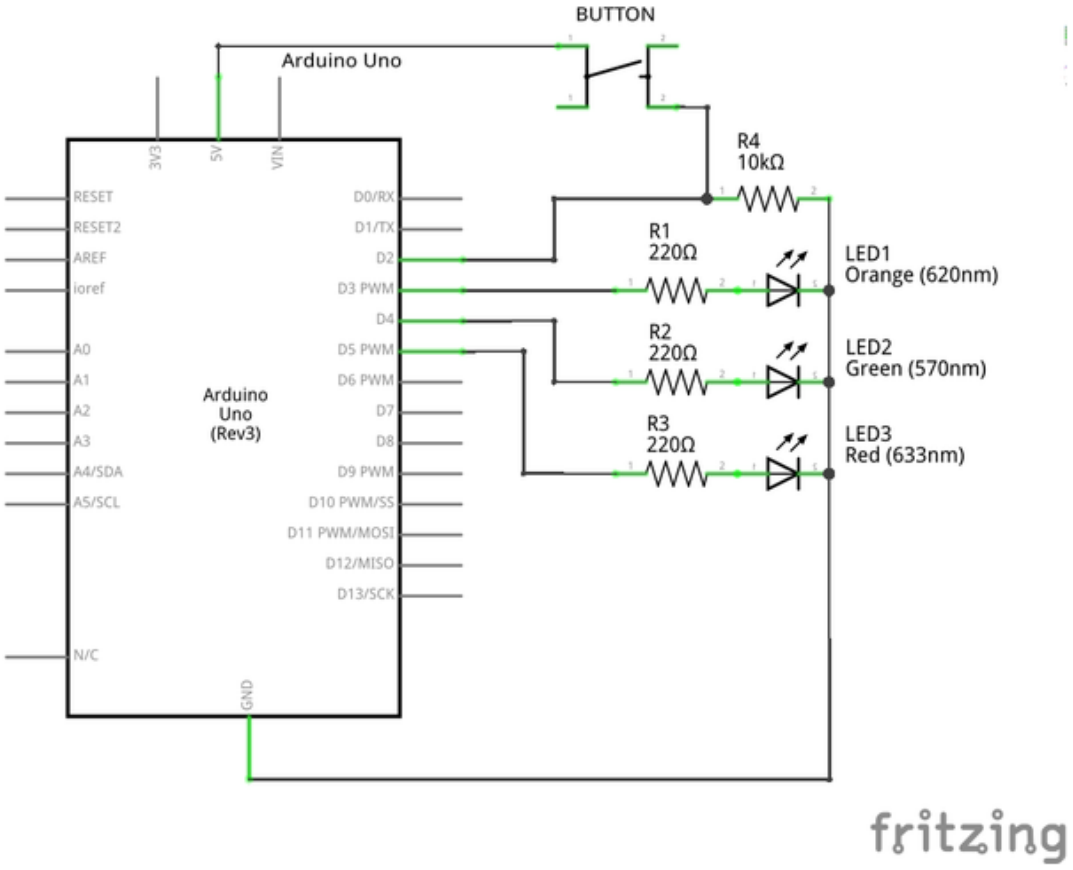
Funkcia nemá žiadne vstupné parametre a tiež nevracia žiadnu hodnotu. Overíme naše riešenie. Vytvorenú funkciu zavoláme zvnútra funkcie `loop()`. Ak sme postupovali správne, semafor začne pracovať v dennom režime.

```
void day_mode(){
    digitalWrite(LED_RED, HIGH);
    delay(5*1000);

    digitalWrite(LED_ORANGE, HIGH);
    delay(2*1000);

    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_ORANGE, LOW);
    digitalWrite(LED_GREEN, HIGH);

    delay(3*1000);

    digitalWrite(LED_GREEN, LOW);
    digitalWrite(LED_ORANGE, HIGH);

    delay(2*1000);
    digitalWrite(LED_ORANGE, LOW);
}

void loop(){
    night_mode();
}
```

Zmenu nočného režimu na denný a opačne je možné riešiť časovačom alebo svetelným senzorom. V našom prípade využijeme jedno tlačidlo.

Na PIN č. 2 pripojíme tlačidlo, po stlačení ktorého bude možné na tomto PIN-e odčítať napätie reprezentujúce úroveň logickej jednotky.

-



Obrázok 68. Schéma zapojenia semafora spolu s tlačidlom pre prepínanie režimov

Pri zapájaní je potrebné mať na pamäti, že ak na PIN nie je privedené žiadne napätie, tzn. tlačidlo nie je stlačené, neznamená to automaticky, že sa na ňom nachádza 0 voltov. Je potrebné zvážiť použitie Pull-Up alebo Pull-Down rezistorov.

Vo funkcii `loop()` implementujeme kód, pomocou ktorého budeme vedieť prepnúť režim práce semaforu z denného na nočný a opačne. Vzhľadom na súčasnú implementáciu je možné prepnúť režim práce semaforu až po ukončení jeho aktuálneho pracovného cyklu. Po zmene pracovného cyklu sa tento cyklus musí opakovať, až kým tlačidlo znova nestlačíme. Stav semafora uchovávajme opäť bez použitia globálnych premenných.

Vzhľadom na aktuálnu implementáciu bude možné prepnúť režim práce semaforu až po ukončení jeho aktuálneho pracovného cyklu. Čo je však dôležité - po zmene pracovného cyklu sa tento cyklus musí neustále opakovať, až pokiaľ nestlačíte tlačidlo znova.

```
#define PIN_BUTTON 2

enum mode {
    NIGHT_MODE,
    DAY_MODE
};

void loop(){
    enum mode mode = NIGHT_MODE;

    for(;;){
        // check button first
        bool state = digitalRead(PIN_BUTTON);

        // evaluate the button
        if(state == HIGH){
            if(mode == NIGHT_MODE){
                mode = DAY_MODE;
            }else{
                mode = NIGHT_MODE;
            }
        }

        // play mode
        if(mode == NIGHT_MODE){
            night_mode();
        }else{
            day_mode();
        }
    }
}
```

Ak sme postupovali správne, stlačením tlačidla dôjde k prepnutiu režimu práce semafora z nočného na denný alebo opačne. Semafor by mal v tomto režime pracovať aj vtedy, keď sa zvolený cyklus skončí a spustí sa znova.

Ako doplňujúce úlohy môžeme vytvoriť rozšírenie o semafor pre chodcov, ktorý bude synchronizovaný s našim semaforom pre autá. Môžeme sa tiež pokúsiť upraviť riešenie tak, aby bolo možné prejsť z jedného režimu do druhého stlačením tlačidla kedykoľvek, aj počas práve prebiehajúceho cyklu.

4.8 Vývoj aplikácií pre internet vecí, TUKE v Košiciach

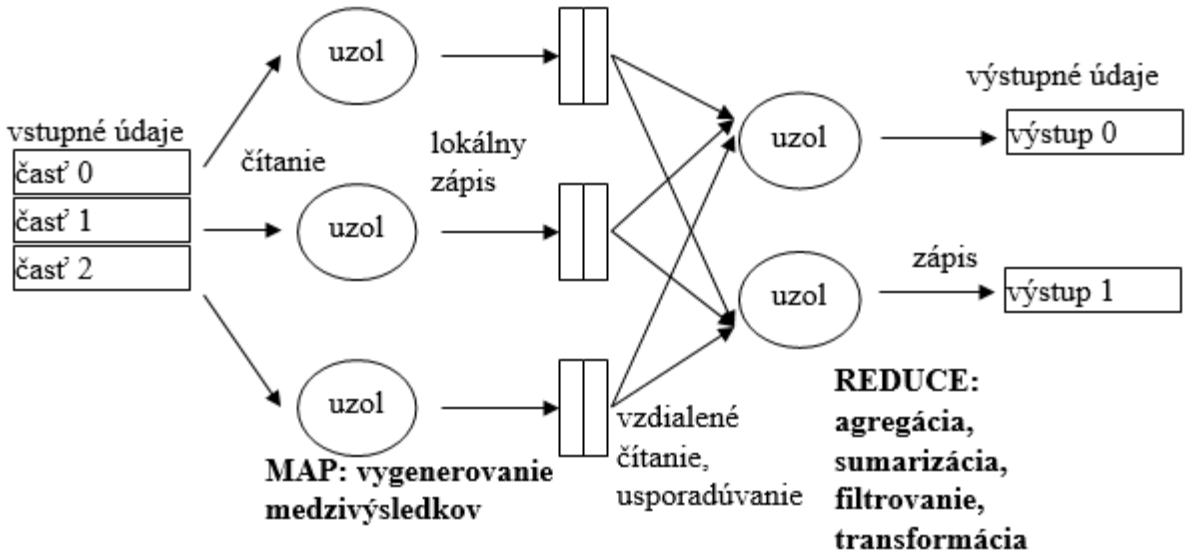
(Jaroslav Porubän, Martin Tomášek, Dominik Lakatoš)

4.8.1 Anotácia a stručná osnova predmetu

Obsahom predmetu Vývoj aplikácií pre internet vecí sú základné pojmy cloudového počítania, webových služieb, riziká cloudového počítania, výpočtové platformy. Popisujeme modely cloudových služieb, princípy verejného, súkromného a hybridného cloudu, paralelné výpočty pomocou modelu MapReduce, cloudové úložiská, spôsob výmeny informácií medzi rôznymi aplikáciami. Významnú časť tvorí aj problematika znakov architektúry cloudových aplikácií s ohľadom na komponenty aplikácie, zdieľanie aplikácie alebo integráciu prostredí.

4.8.2 Aplikačný príklad

V aplikačnom príklade si predstavíme výpočet početnosti jednotlivých slov v texte užívateľa pomocou modelu distribuovaného výpočtu MapReduce pre spracovanie veľkých dátových sád. Tento model bol navrhnutý v spoločnosťou Google, pričom bol inšpirovaný funkcionálnym programovaním a stratégiou rozdeľuj a panuj.



Obrázok 69. Schéma modelu distribuovaného výpočtu MapReduce

Motiváciou pre takéto modely je spracovanie veľkého množstva údajov, napríklad Google spracováva približne 24 petabajtov dát denne. Jeden stroj nedokáže obslúžiť také veľké množstvo údajov, z tohto dôvodu je potrebné použiť distribuovaný systém na uskladnenie a spracovanie údajov paralelným spôsobom. Riešenie úlohy pomocou programovacieho modelu MapReduce môže mať schému, ktorá je znázornená na obrázku.

Vstupom je množina dvojíc `(kľúč, hodnota)` s rovnakým typom všetkých kľúčov a rovnakým typom všetkých hodnôt. Našim cieľom je spočítať početnosti jednotlivých slov v texte užívateľa. Vstupom pre funkciu `map()` budú dvojice

$$(\text{queryID}, \text{QueryText}),$$

napríklad jedna z dvojíc môže mať nasledujúci tvar:

> (Q1,“The teacher went to the store. The store was closed;
> the store opens in the morning. The store opens at 9am.”).

Výstup funkcie `map()` je v tvare

> (The, 1) (teacher, 1) (went, 1) (to, 1) (the, 1) (store,1) (the, 1) (store, 1) (was, 1) (closed, 1) (the, 1) (store,1) (opens, 1) (in, 1) (the, 1) (morning, 1) (the 1) (store, 1) (opens, 1) (at, 1) (9am, 1).

Funkcia `reduce ()` má na vstupe výstup funkcie `map()` a následne agreguje hodnoty na základe kľúča. Pre predchádzajúci výstup funkcie `map()`, dostávame nasledujúci výstup funkcie `reduce()`:

> (The, 6) (teacher, 1) (went, 1) (to, 1) **(store, 4)** (was, 1) (closed, 1) (opens, 1) (morning, 1) (at, 1) (9am, 1).

Zdrojový kód programu na výpočet početnosti jednotlivých slov v texte užívateľa pomocou modelu distribuovaného výpočtu MapReduce v programovacom jazyku Java môže mať nasledujúci tvar:

```
public class WordCount {
 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
 }}}
 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
 }}

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);
        p
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
}}
```

Obrázok 70. Program na výpočet početnosti jednotlivých slov v texte užívateľa pomocou modelu distribuovaného výpočtu MapReduce v programovacom jazyku Java

4.9 Úvod do internetu vecí, TUKE v Košiciach (Tomáš Kanócz, František Jakab)

4.9.1 Anotácia a stručná osnova predmetu

Obsahom predmetu Úvod do internetu vecí, ktorý je určený aj pre študentov neinformatických odborov, je prehľad základných pojmov a architektúre internetu vecí, jeho funkčných stavebných blokoch, senzoroch, akčných členoch, softvérovom programovaní a integrovaní s fyzických svetom, lokálnom spracovaní na hranici siete, bezpečnom a efektívnom prenose dát cez rôzne sieťové protokoly, ukladaní a spracovaní dát v cloude, riadení na základe dát, ako aj podnikateľské nápady v danej oblasti.

Prezentujeme pojmy vecí, pripojení, stavebných blokov systému internetu vecí, procesy v riadených systémoch, modely komunikácie, vrstvy pripojení, vplyv pripojení na súkromie a bezpečnosť. Popisujeme spájanie vecí, programovanie, Raspberry PI, používanie operačného systému Linux, Python na Raspberry PI, Budovanie modulov internetu vecí v Packer Tracer. Zaoberáme sa snímačmi, aktuátormi a mikrokontrolérmi. Prezentujeme základné a pokročilé elektronické terminológie a koncepty, schématické diagramy, testovacie dosky. Pri mikrokontroléroch popisujeme súpravu vynálezcu, jednoduché obvody, snímanie prostredia pomocou senzorov, ovládače, aktivátory a relé. Prezentujeme pripojenie vecí k sieti, úlohy siete, bezdrôtové technológie, Fog a Cloud služby, Big Data, bezpečnostné obavy v Internete vecí. Obsahom predmetu sú aj osvedčené postupy pri tvorbe riešení pomocou internetu vecí, pri aplikáciách internetu vecí v podnikaní,

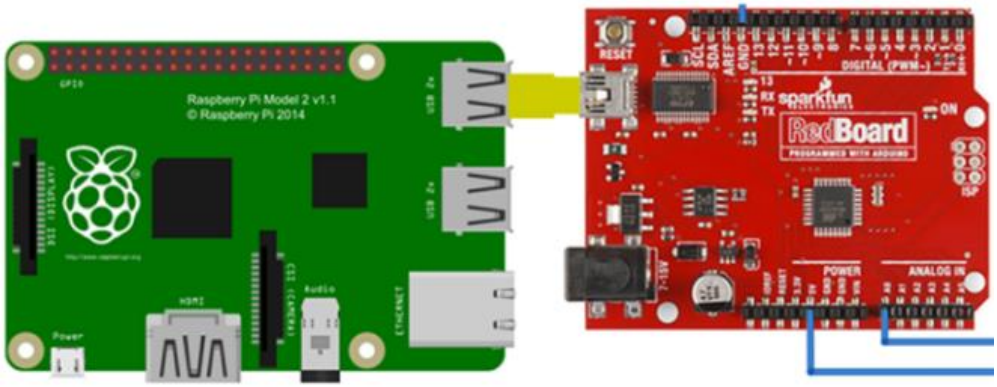
4.9.2 Aplikačný príklad

V tejto časti predstavíme odporúčaný postup pri tvorbe zariadení internetu vecí a priebežne modelujeme príklad, v ktorom tvoríme zariadenie internetu vecí, ktoré sníma množstvo svetla a na základe toho určuje východ a západ slnka.

Na začiatku každého riešenia je potrebné vytvoriť prehľad projektu, v ktorom identifikujeme problém, ktorý je možné vyriešiť pomocou zariadenia internetu vecí. Keďže elektronické komponenty majú špecifické požiadavky na napájanie, polaritu a pripojenie, je potrebné identifikovať usporiadanie okruhu a opísať požiadavky.

Postup pri vytvorení nového zariadenia internetu vecí by sme mohli zhrnúť do nasledujúcich krokov:

1. Identifikácia problému, ktorý je možné vyriešiť pomocou zariadenia internetu vecí. Napríklad v našom príklade o východe a západe slnka sa používajú internetové služby IFTTT, Raspberry Pi, Arduino a fotorezistor IoT.
2. Vytvorenie rozloženia obvodu. Príklad o snímaní množstva svetla vyžaduje rozdeľovač napätia, ktorý produkuje výstupné napätie. Výstupné napätie je zlomkom jeho vstupného napätia, rozdeľovaním vstupného napätia medzi komponenty deliča. rozloženie obvodu.
3. Vytvorenie požiadaviek na programovanie – výber programovacieho jazyka, API REST. Aplikácie REST API používajú metódy HTTP na výmenu údajov medzi systémami alebo aplikáciami.
4. Vytvorenie vývojových diagramov a elektronických schém, ktoré je potrebné pridať do dokumentácie. Vývojové diagramy sú diagramy reprezentujúce usporiadané procesy a pracovné postupy. Elektronické schémy reprezentujú komponenty a prepojenia schémy zapojení pomocou medzinárodne štandardizovaných symbolov.
5. Vytvorenie diagramu sekvencií, ktorý slúži na vyjadrenie interakcií medzi entitami na časovej osi.
6. Tvorba zdrojového kód programu. V príklade východu a západu slnka môžeme použiť programovací jazyk Python. Arduino je pripojený k Raspberry Pi. Programovanie sa vykonáva na Raspberry Pi. Firmata, všeobecný protokol pre komunikáciu s mikrokontrolérmi, slúži na komunikáciu medzi Arduino a Raspberry Pi.



Obrázok 71. Raspberry Pi a doska Sparkfun

4.10 Princípy počítačových sietí, UKF v Nitre (Peter Švec)

4.10.1 Anotácia a stručná osnova predmetu

Predmet je zameraný na základné princípy fungovania počítačových sietí z pohľadu používateľa, ale aj z pohľadu ich administrátora. Prezentujeme základné prenosové média používané v počítačových sieťach a popisujeme ich podstatné vlastnosti. V predmete predstavujeme históriu vzniku a vývoja počítačových sietí a ich protokolov, pričom sa zameriavame na vývoj referenčných vrstvových modelov používaných pre popis počítačových sietí. Popis funkcií na jednotlivých vrstvách dopĺňame o názorné príklady, hry a iný multimediálny obsah. V závere sa venujeme bezpečnosti počítačových sietí a nástrahám, ktoré ich používanie prináša.

4.10.2 Aplikačný príklad

V aplikačnom príklade sa budeme venovať orientácii v Mikrotik Router OS a základnej konfigurácii zariadenia. Zariadenia Mikrotik sú obľúbenými v malých a stredných podnikoch najmä z dôvodu ich cenovej dostupnosti, jednoduchému a intuitívnemu ovládaniu a spoľahlivosti. Na vyskúšanie si príkazov nie je potrebné vlastniť Mikrotik zariadenie, môžete použiť demo dostupné na adresách *demo.mt.lv* alebo *demo2.mt.lv* (predvolené meno používateľa je *demo*, bez hesla).

Router OS

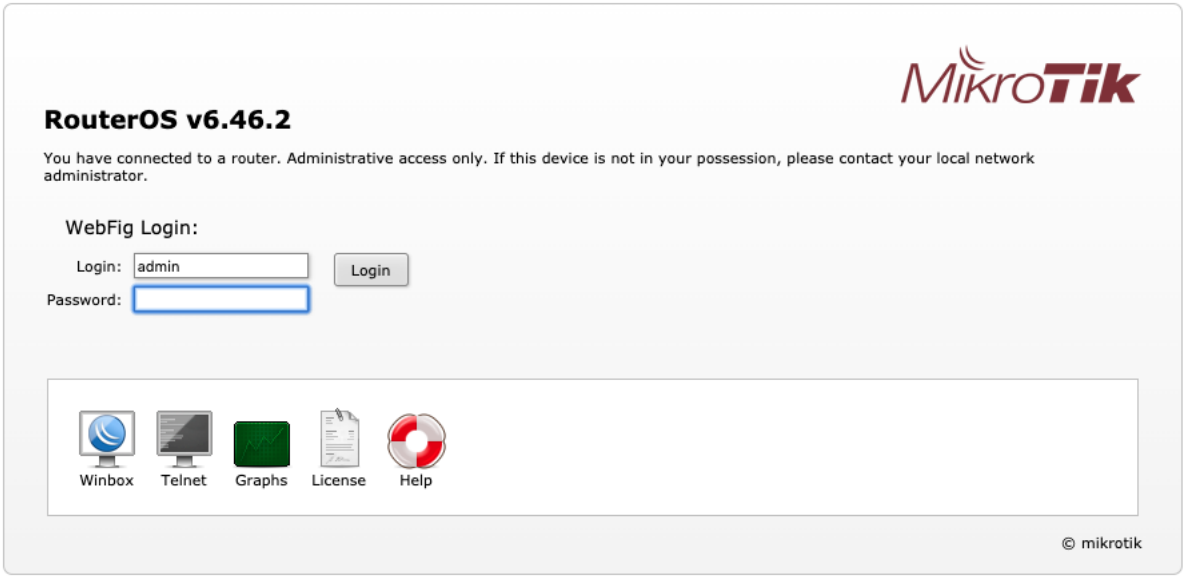
Operačný systém je postavený na linuxovom jadre a je možné ho používať nielen so zariadeniami typu Router Board ale aj na bežnom počítači, ktorý bude plniť úlohu smerovača. Zariadenie môžeme pripojiť konfigurovať niekoľkými spôsobmi.

- Pomocou webového rozhrania (WebFig).
- Pomocou špeciálneho softvéru WinBox (dostupného pre Windows, avšak kompatibilného s Wine)
- Pomocou príkazového riadku (CLI) dostupnom cez Telnet, SSH, sériový kábel alebo dokonca klávesnicou s monitorom ak má zariadenie grafickú kartu.

Každý smerovač má od výroby predkonfigurovanú IP adresu 192.168.88.1/24 na porte ether1. Predvolené prihlasovacie meno je *admin*, heslo nastavené nie je. Po prvom prihlásení odporúčame vytvoriť nového používateľa s plným prístupom a vymazať používateľa *admin*.

Webfig

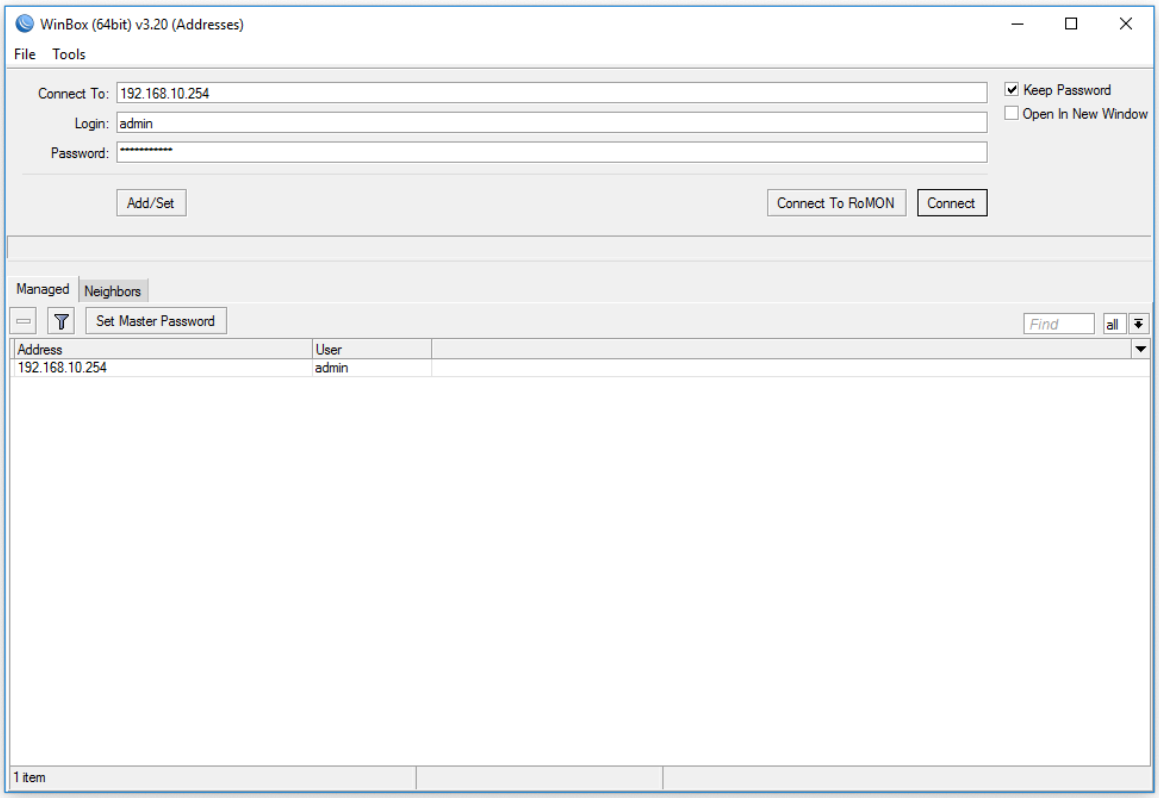
Webové rozhranie predstavuje najjednoduchší spôsob ako manažovať Mikrotik zariadenie. Po zadaní IP adresy v prehliadači dostávame možnosť stiahnuť si aplikáciu Winbox zo stránku výrobcu, sledovať grafy rozhraní alebo si pozrieť licenciu alebo manuál.



Obrázok 72. Prihlásenie v RouterOS

Winbox

Softvérový nástroj Winbox je najjednoduchšou možnosťou, akou sa pripojiť k Mikrotik zariadeniu. Pre jeho spustenie potrebujeme počítač s operačným systémom Windows.



Obrázok 73. Prihlásenie vo WinBox

V prípade, že sme na lokálnej sieti, Winbox dokáže nájsť dostupné zariadenia s RouterOS (záložka Neighbors). Ak nie, môžeme zadať IP adresu manuálne. Po úspešnom prihlásení sa si Winbox stiahne zo smerovača zásuvné moduly.

Príkazový riadok (CLI)

Príkazový riadok slúži pre konfiguráciu pomocou textových príkazov. Keďže RouterOS je postavený na linuxovom jadre, aj príkazový riadok sa podobá linuxového shellu. Príkazový riadok tiež umožňuje vytvárania rôznych skriptov.

Pre prístup k príkazovému riadku môžeme použiť terminál vo Winboxe, Telnet, SSH alebo sériový kábel. Ak máme nainštalovaný RouterOS na počítači, tak môžeme použiť klávesnicu a monitor. Parametre pre pripojenie cez sériový port sú *115200bit/s, 8 data bits, 1 stop bit, no parity, flow control=none*.

Predvolený používateľ má meno **admin** a heslo nie je nastavené. Príkazy sú rozdelené do skupín, čo pripomína hierarchické menu. Názov úrovne menu prezrádza kontext príkazov, ktoré v menu nájdeme, napr. `/ip route print`

```
[admin@MikroTik] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m -
     mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY
     DIS INTE...
 0 A S  0.0.0.0/0                          r 10.0.3.1
     1   bridge1
 1 ADC  1.0.1.0/24         1.0.1.1
     0   bridge1
 2 ADC  1.0.2.0/24         1.0.2.1
     0   ether3
 3 ADC  10.0.3.0/24        10.0.3.144
     0   bridge1
 4 ADC  10.10.10.0/24      10.10.10.1
     0   wlan1
[admin@MikroTik] >
```

K rovnakému výsledku sa dopracujeme aj keď sa budeme nachádzať v menu ip route a zadáme príkaz `print`.

```
[admin@MikroTik] > ip route
[admin@MikroTik] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m -
     mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY
     DIS INTE...
 0 A S  0.0.0.0/0                          r 10.0.3.1
     1   bridge1
 1 ADC  1.0.1.0/24         1.0.1.1
     0   bridge1
 2 ADC  1.0.2.0/24         1.0.2.1
     0   ether3
```

```
 3 ADC  10.0.3.0/24         10.0.3.144
      0   bridge1
 4 ADC  10.10.10.0/24       10.10.10.1
      0   wlan1
[admin@MikroTik] ip route>
```

Do vyššej úrovne menu sa dostaneme po zadaní .., prípadne môžeme zadať / aby sme sa dostali na najvyššiu úroveň. Samozrejmosťou je dopĺňanie príkazov po stlačení klávesy TAB.

Mnoho z príkazov pracuje s zoznamami údajov ako sú rozhrania, cesty, používatelia a pod. Položky takého zoznamu sú číslované (napr. číslo rozhrania) alebo pomenované (napr. názov rozhrania) a toto číslo alebo názov môžeme používať v ďalších príkazoch. Pri písaní skriptov preferujeme pomenovanie, keďže je viac špecifické. Číslo je závislé od príkazu print a je viazané na dané sedenia. Znamená to, že pri rôznych prihláseniach k systému môže byť výstup z print príkazu iný. Ak použijeme číslo bez predchádzajúceho zavolania príkazu print, systém vykoná print na pozadí.

```
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1500
  1  R ether2              ether            1500
  2  R ether3              ether            1500
  3  R ether4              ether            1500
[admin@MikroTik] > interface set 0,1,2 mtu=1460
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1460
  1  R ether2              ether            1460
  2  R ether3              ether            1460
  3  R ether4              ether            1500
[admin@MikroTik] >
```

Základná konfigurácia - konfigurácia IPv4 a IPv6 adries na rozhraní

Vo väčšine prípadov stačí zadať adresu, masku siete a argumenty rozhrania. Sieťový prefix a adresa pre broadcast sa počítajú automaticky. Na rozhranie je možné pridať viac IP adries alebo nechať rozhranie aj bez priradených adries.

MikroTik RouterOS má nasledujúce typy adries:

- Statické – manuálne nastavené používateľom
- Dynamické – automaticky nastavené pomocou DHCP servera alebo PPP spojením

IPv4 adresu nastavíme nasledovne:

```
[admin@MikroTik] ip address> add address=10.10.10.1/24
     interface=ether2
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS            NETWORK         BROADCAST
     INTERFACE
  0    2.2.2.1/24         2.2.2.0         2.2.2.255
     ether2
  1    10.5.7.244/24      10.5.7.0        10.5.7.255
     ether1
  2    10.10.10.1/24      10.10.10.0      10.10.10.255
     ether2
```

Podobne nastavíme aj IPv6 adresu:

```
[admin@MikroTik] > ipv6 address add address=fc00:3::/64
     interface=ether3 eui-64=yes
[admin@MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global,
     L - link-local
 #     ADDRESS                                     INTERFACE
     ADVERTISE
 ...
 5  G fc00:3::20c:42ff:fe1d:3d4/64                 ether3
     yes
```

```
[admin@MikroTik] > ipv6 address add address=2001:DB8::1/64
     interface=ether1 advertise=no
[admin@ MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global,
     L - link-local
 #     ADDRESS                                      FROM-POOL
     INTERFACE     ADVERTISE
 0  G 2001:db8::1/64
     ether1        no
 3 DL fe80::219:d1ff:fe39:3535/64
     ether1        no
```

Konfigurácia IPv4 DHCP servera

Router podporuje vytvorenie samostatného servera na každom rozhraní typu ethernet. DHCP server podporuje základné funkcie poskytujúce každému žiadajúcemu klientovi IP adresu / sieťovú masku, predvolenú bránu, názov domény, servery DNS a servery WINS (pre klientov Windows).

Aby server DHCP fungoval, musia byť nakonfigurované aj rozsahy IP. V rozsahu IP adries nesmieme uviesť IP adresu samotného servera. RouterOS má zabudovaný príkaz, ktorý umožní ľahko nastaviť server DHCP. Povedzme, že chceme nakonfigurovať server DHCP v rozhraní ether1 na prenájom adries od 192.168.0.2 do 192.168.0.254, ktoré patria do siete 192.168.0.0/24. Brána a server DNS je 192.168.0.1. V ponuke / ip dhcp-server spustíme príkaz setup a postupujte podľa pokynov:

```
[admin@MikroTik] ip dhcp-server> setup
Select interface to run DHCP server on

dhcp server interface: ether1
Select network for DHCP addresses

dhcp address space: 192.168.0.0/24
Select gateway for given network

gateway for dhcp network: 192.168.0.1
```

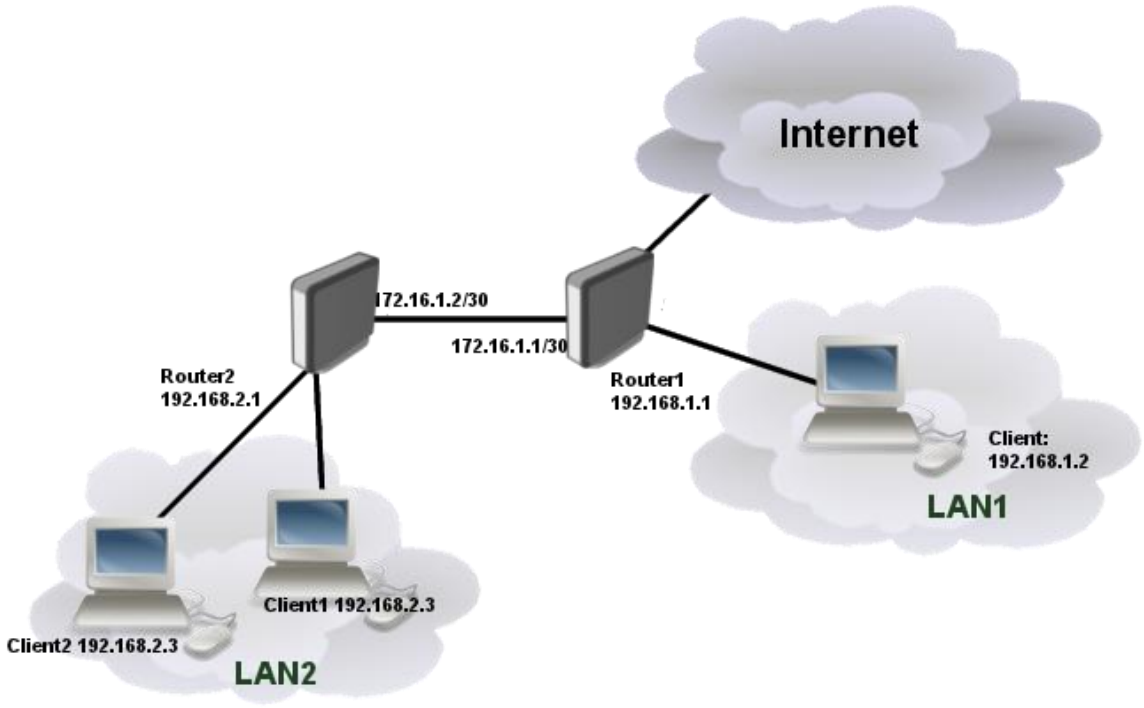
```
Select pool of ip addresses given out by DHCP server

addresses to give out: 192.168.0.2-192.168.0.254
Select DNS servers

dns servers: 192.168.0.1
Select lease time

lease time: 3d
[admin@MikroTik] ip dhcp-server>
```

Sprievodca urobil nasledujúcu konfiguráciu na základe vyššie uvedených odpovedí:

```
[admin@MikroTik] ip dhcp-server> print
Flags: X - disabled, I - invalid
  #    NAME            INTERFACE RELAY           ADDRESS-
     POOL LEASE-TIME ADD-ARP
  0    dhcp1           ether1    0.0.0.0         dhcp_pool1
     3d         no

[admin@MikroTik] ip dhcp-server> network print
  # ADDRESS            GATEWAY         DNS-SERVER
     WINS-SERVER     DOMAIN
  0 192.168.0.0/24        192.168.0.1        192.168.0.1

[admin@MikroTik] ip dhcp-server> /ip pool print
  # NAME                                          RANGES
  0 dhcp_pool1
     192.168.0.2-192.168.0.254

[admin@MikroTik] ip dhcp-server>
```

IP pool sa používajú na definovanie rozsahu adries IP, ktoré sa používajú pre servery DHCP a Point-to-Point. Na vytvorenie rozsahu s názvom ip-pool s rozsahom adries 10.0.0.1-10.0.0.125 s výnimkou adresy brány 10.0.0.1 a adresy servera 10.0.0.100 a druhej oblasti dhcp-pool s adresou 10.0.0.200-10.0.0.250 použijeme nasledovné príkazy:

```
[admin@MikroTik] ip pool> add name=ip-pool ranges=10.0.0.2-
     10.0.0.99,10.0.0.101
10.0.0.126
[admin@MikroTik] ip pool> add name=dhcp-pool
     ranges=10.0.0.200-10.0.0.250
[admin@MikroTik] ip pool> print
  # NAME                                    RANGES
  0 ip-pool                                 10.0.0.2-10.0.0.99
                                            10.0.0.101-10.0.0.126
  1 dhcp-pool                               10.0.0.200-10.0.0.250

[admin@MikroTik] ip pool>
```



Obrázok 74. Statické smerovanie

Ether1 na Router1 je pripojený k ISP a bude bránou našich sietí. Router2 je pripojený k ether2 Router1 a bude slúžiť ako brána pre klientov pripojených k nemu z LAN2. Router1 tiež pripája jedného klienta k ether3. Naším cieľom je vytvoriť nastavenie tak, aby klienti z LAN1 mohli komunikovať s klientmi z LAN2 a všetci sa mohli pripojiť k internetu.

Predpokladajme, že ISP nám pridelil adresu 10.1.1.2/30 a brána je 10.1.1.1.

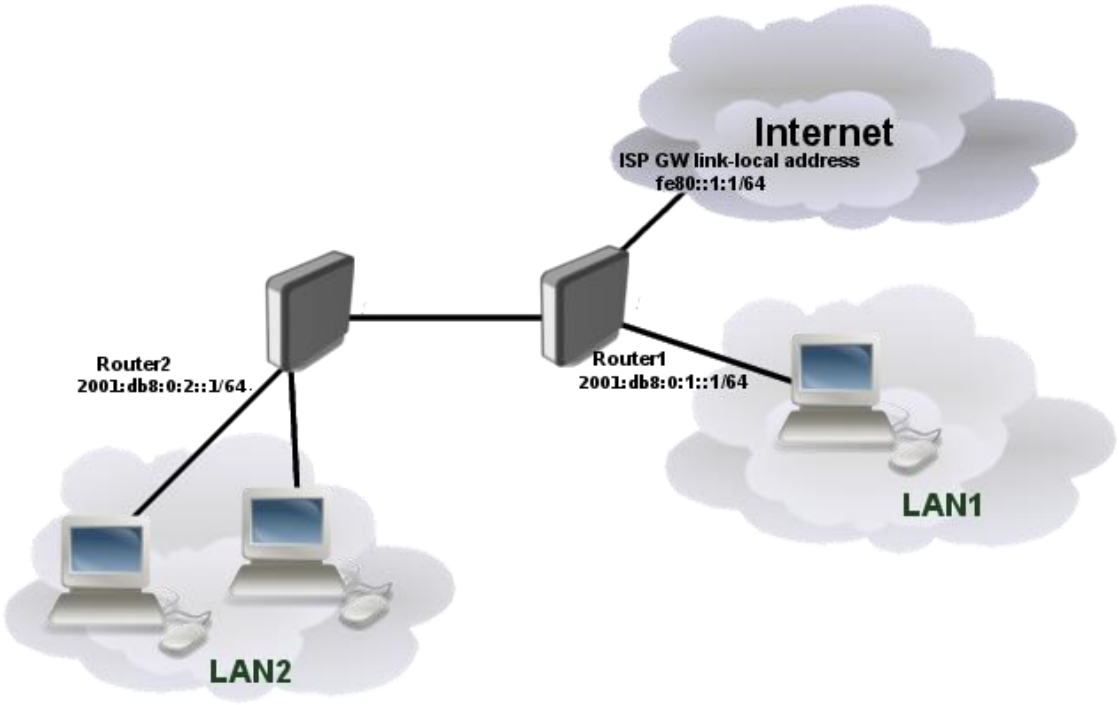
Router1:

```
/ip address
add address=10.1.1.2 interface=ether1
add address=172.16.1.1/30 interface=ether2
add address=192.168.1.1/24 interface=ether3

/ip route
add gateway=10.1.1.1
add dst-address=192.168.2.0/24 gateway=172.16.1.2
```

Router2:

```
/ip address
add address=172.16.1.2/30 interface=ether1
add address=192.168.2.1/24 interface=ether2

/ip route
add gateway=172.16.1.1
```

Ak sa pozrieme na konfiguráciu, uvidíme, že na Router1 sme pridali cestu do cieľa 192.168.2.0/24. Je potrebné, aby klienti z LAN1 boli schopní osloviť klientov v LAN2. V Router2 sa takáto cesta nevyžaduje, pretože LAN1 sa dá dosiahnuť predvolenou cestou.



Obrázok 75. Statické smerovanie 2

Predpokladajme, že ISP nám pridelil prefix 2001:db8::/62 and prefix je ku nám smerovaný pomocou link-local adresy (fe80::1:1).

Rozhranie Ether1 na Router1 je pripojený k ISP a bude bránou našich sietí. Router2 je pripojený k rozhraniu ether2 na Router1 a bude slúžiť ako brána pre klientov pripojených k nemu z LAN2. Router1 tiež pripája jedného klienta k ether3. Naším cieľom je vytvoriť nastavenie tak, aby klienti z LAN1 mohli oslovovať klientov z LAN2 a všetci sa mohli pripojiť k internetu.

Konfigurácia

Najprv musíme zistiť, aké link-local adresy sú na na smerovači Router1 a na smerovači Router2. Smerovanie pomocou protokolu IPv6 môžeme vykonať bez globálnych IPv6 adries, čím nebudeme plytvať adresami. V nižšie zobrazenom nastavení nie sú definované žiadne globálne adresy a to ani medzi poskytovateľom internetových služieb a našou bránou.

```
[admin@R1] /ipv6 address> print
```

```
Flags: X - disabled, I - invalid, D - dynamic, G - global,
     L - link-local
 #    ADDRESS    FROM-POOL INTERFACE     ADVERTISE
 0 DL fe80::219:d1ff:fe00:3511/64     ether1        no
 1 DL fe80::219:d1ff:fe00:3512/64     ether2        no
 1 DL fe80::219:d1ff:fe00:3513/64     ether3        no
[admin@R2] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global,
     L - link-local
 #    ADDRESS    FROM-POOL INTERFACE     ADVERTISE
 0 DL fe80::219:d1ff:fe39:3535/64     ether1        no
 1 DL fe80::219:d1ff:fe39:3536/64     ether2        no
```

Postup pre konfiguráciu je nasledovný:

Router1

```
/ipv6 address
add address=2001:db8:0:1::1/64 interface=ether3
     advertise=yes

/ipv6 route
add gateway=fe80::1:1%ether1
add dst-address=2001:db8:0:2::/64
     gateway=fe80::219:d1ff:fe39:3535%ether2
```

Router2

```
/ipv6 address
add address=2001:db8:0:2::1/64 interface=ether2
     advertise=yes

/ipv6 route
add gateway=fe80::219:d1ff:fe00:3512%ether1
```

Pridané globálne adresy sú označené príznakom advertise, čo znamená, že sa použije RA na automatickú konfiguráciu adresovania IPv6 na klientskych počítačoch.

Konfigurácia RIP protokolu vyzerá nasledovne.

```
[admin@MikroTik] > interface print
Flags: X - disabled, D - dynamic, R - running
  #    NAME                TYPE             MTU
  0  R ether1              ether            1500
  1  R ether2              ether            1500
[admin@MikroTik] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST
      INTERFACE
  0   10.0.0.174/24      10.0.0.174      10.0.0.255
      ether1
  1   192.168.0.1/24     192.168.0.0     192.168.0.255
      ether2
[admin@MikroTik] > ip route print
Flags: X - disabled, I - invalid, D - dynamic, J -
     rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
     #    DST-ADDRESS        G GATEWAY         DISTANCE
      INTERFACE
     0 DC 192.168.0.0/24     r 0.0.0.0         0
      ether2
     1 DC 10.0.0.0/24        r 0.0.0.0         0
      ether1
[admin@MikroTik] >
```

Potrebná konfigurácia všeobecných nastavení RIP je nasledovná:

```
[admin@MikroTik] routing rip> set redistribute-
     connected=yes
[admin@MikroTik] routing rip> print
       redistribute-static: no
```

```
    redistribute-connected: yes
         redistribute-ospf: no
          redistribute-bgp: no
             metric-static: 1
          metric-connected: 1
               metric-ospf: 1
                metric-bgp: 1
              update-timer: 30s
             timeout-timer: 3m
             garbage-timer: 2m

[admin@MikroTik] routing rip>
```

Minimálna požadovaná konfigurácia rozhrania RIP znamená len zapnutie siete asociovanej s rozhraním ether1.

```
[admin@MikroTik] routing rip network> add
     address=10.0.0.0/2
[admin@MikroTik] routing rip network> print
  # ADDRESS
  0 10.0.0.0/24

[admin@MikroTik] routing rip network>
```

Na rozhraní ether2 nemusíme spúšťať RIP, pretože na tomto rozhraní to nie je potrebné. Zoznam ciest získaných cez RIP protokol zobrazíme z menu /routing rip route.

```
[admin@MikroTik] routing rip> route print
Flags: S - static, R - rip, O - ospf, C - connect, B - bgp
  0 R dst-address=0.0.0.0/0 gateway=10.0.0.26 metric=2
     from=10.0.0.26
  1 C dst-address=10.0.0.0/24 gateway=0.0.0.0 metric=1
     from=0.0.0.0
  2 C dst-address=192.168.0.0/24 gateway=0.0.0.0 metric=1
     from=0.0.0.0
```

```
  3 R dst-address=192.168.1.0/24 gateway=10.0.0.26 metric=1
     from=10.0.0.26
  4 R dst-address=192.168.3.0/24 gateway=10.0.0.26 metric=1
     from=10.0.0.26
[admin@MikroTik] routing rip>
```

Smerovacia tabuľka je:

```
[MikroTik] routing rip> /ip route print
Flags: X - disabled, I - invalid, D - dynamic, J -
     rejected,
C - connect, S - static, R - rip, O - ospf, B - bgp
     #    DST-ADDRESS        G GATEWAY         DISTANCE
      INTERFACE
     0  R 0.0.0.0/0          r 10.0.0.26       120
      ether1
     1  R 192.168.3.0/24     r 10.0.0.26       120
      ether1
     2  R 192.168.1.0/24     r 10.0.0.26       120
      ether1
     3 DC 192.168.0.0/24     r 0.0.0.0         0
      ether2
     4 DC 10.0.0.0/24        r 0.0.0.0         0
      ether1
[admin@MikroTik] routing rip>
```

VPN spojenie

Zariadenia Mikrotik umožňujú vytvoriť niekoľko typov VPN sietí. Najjednoduchším typom je PPTP, bezpečný tunel na prenos IP prenosov pomocou PPP (Point-to-Point). PPTP zapuzdruje PPP do virtuálnych spojení, ktoré bežia cez IP. Na vytvorenie šifrovaného spojenia spája PPTP dva protokoly PPP a MPPE (Microsoft Point to Point Encryption). Účelom tohto protokolu je vytvoriť bezpečné spojenia medzi smerovačmi alebo medzi smerovačov a PPTP klientmi (klienti sú k dispozícii pre takmer všetky operačné systémy).

Tabuľka 4. Prehľad protokolov

Názov protokolu	**Vrstva OSI**	**Max MTU**	**Protokol**	**Most**	**Topológia**	**Bezpečnosť**	**Verzia Mikrotik**	**Použitie**
EoIP	L3	1500	TCP	áno	PtP	nie	> 2.9	Prepojenie sietí cez rôznych ISP
IP tunnel	L3	1480	TCP	nie	PtP	nie	> 2.9	
PPtP	L2	1420	GRE, TCP	áno (BCP)	PtMP	áno	> 2.9	Pripojenie klientov na server
L2tP	L2	1420	UDP	áno (BCP)	PtMP	áno	> 2.9	Pripojenie klientov na server
SSTP	L2	1500	TCP	áno (BCP)	PtMP	áno	> 5.0	Pripojenie klientov na server

Podporovaný je aj Multilink PPP (MP), ktorý MRRU je schopnosť prenášať pakety s plnou veľkosťou 1500 alebo väčšie) a umožňuje tiež vytvoriť premostenie cez PPP spojenie (pomocou Bridge Control Protocol (BCP)). Dosiahneme tým odosielanie ethernetových rámcov cez PPP spojenie). Týmto spôsobom je možné nastaviť premostenie bez nutnosti použiť EoIP. Prepojenie by malo obsahovať buď administratívne nastavenú MAC adresu alebo vytvorené virtuálne rozhranie typu ethernete, pretože spojenia PPP nemajú MAC adresy.

PPTP obsahuje overovanie a účtovanie PPP pre každé pripojenie PPTP. Úplnú autentifikáciu a účtovanie každého pripojenia je možné dosiahnuť prostredníctvom RADIUS klienta. Šifrovanie je MPPE 128bit RC4.

PPTP prevádzka využíva TCP port 1723 a IP protokol GRE (Generic Routing Encapsulation, IP protokol ID 47). Protokol PPTP sa môže používať s väčšinou firewallov a smerovačov povolením TCP portu 1723 a protokolu 47. Vytvoreniu a prevádzke PPTP môže brániť NAT zariadenie v sieti.

V nasledujúcom príklade vytvoríme PPTP clienta s menom „pptp-test“ a heslom „pass123“. Server bude spustený na adrese 10.1.101.100.

```
/interface pptp-client add name=pptp-hm user=pptp-test
password=pass123 connect-to=10.1.101.100 disabled=no
```

Na nasledovnom výpise vidíme detail vytvoreného klienta

```
/interface pptp-client print detail
Flags: X - disabled, R - running
 0    name="pptp-hm" max-mtu=1460 max-mru=1460 mrru=disabled
      connect-to=10.1.101.100                user="pptp-test"
password="pass123"
      profile=default-encryption  add-default-route=no  dial-
on-demand=no
      allow=pap,chap,mschap1,mschap2
```

Pre každý vytvorený tunel sa automaticky vytvára rozhranie. V konfigurácii servera PPTP existujú dva typy rozhraní.

Statické rozhrania sa pridávajú administratívne, ak je potrebné uviesť odkaz na konkrétny názov rozhrania (v pravidlách brány firewall alebo inde) vytvorený pre konkrétneho používateľa.

Dynamické rozhrania sa do tohto zoznamu pridávajú automaticky vždy, keď je sa pripojí používateľ s používateľské menom, ktoré sa nezhoduje so žiadnym existujúcim statickým záznamom (alebo v prípade, že je záznam už aktívny, pretože nemôžu existovať dve samostatné tunelové rozhrania s rovnakým menom).

Dynamické rozhrania sa objavia, keď sa používateľ pripojí a zmiznú, keď sa odpojí. Nie je teda možné používať názov dynamického rozhrania v konfigurácii smerovača (napríklad vo firewalle). Ak pre daného používateľa potrebujeme pravidlá, musíme pre tohto používateľa vytvoriť statický záznam. V opačnom prípade je bezpečné používať dynamickú konfiguráciu.

Konfigurácia servera prebieha nasledovne.

```
/interface pptp-server server set enabled=yes
```

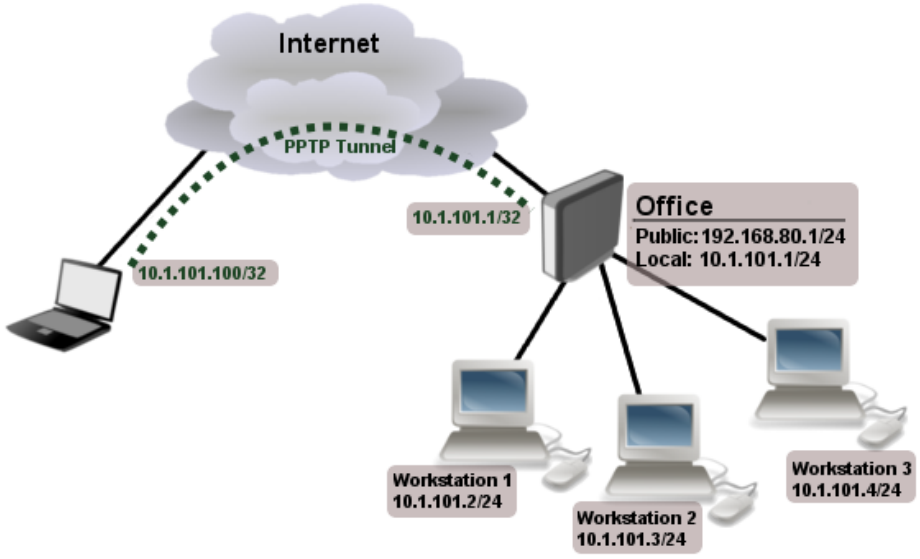
Detaily sú

```
/interface pptp-server server print
             enabled: yes
             max-mtu: 1460
             max-mru: 1460
                mrru: disabled
      authentication: mschap2,mschap1
  keepalive-timeout: 30
    default-profile: default
```

Pre zobrazenie stavu spojenia môžeme použiť príkaz monitor

```
/interface pptp-client monitor 0
      status: "connected"
      uptime: 7h24m18s
   idle-time: 6h21m4s
    encoding: "MPPE128 stateless"
         mtu: 1460
         mru: 1460
```

Celú konfiguráciu si môžeme ukázať na nasledovnom príklade:



Obrázok 76. Schéma smerovania

Smerovač Office je pripojený do internetu cez rozhrania ether1. Pracovné stanice (Workstation) sú pripojené na rozhranie ether2. Laptop sa nachádza v sieti internet a k smerovaču Office sa vie pripojiť cez jeho verejnú IP adresu, aj keď v našom príklade použijeme ako verejnú adresu 192.168.80.1.

Prvým krokom je vytvorenie používateľa. Nazveme ho „laptop“ a heslo bude mať „pass123“.

```
/ppp secret add name=Laptop service=pptp password=pass123
local-address=10.1.101.1 remote-address=10.1.101.100

/ppp secret print detail
Flags: X - disabled
0    name="Laptop"          service=pptp          caller-id=""
     password="pass123" profile=default
      local-address=10.1.101.1     remote-address=10.1.101.100
routes==""
```

Všimnime si, že PPTP lokálna adresa je rovnaká ako adresa smerovača na jeho lokálnom rozhraní a vzdialená adresa je z rovnakého rozsahu aký je v lokálnej sieti (10.1.101.0/24). Po vytvorení používateľa vytvoríme server.

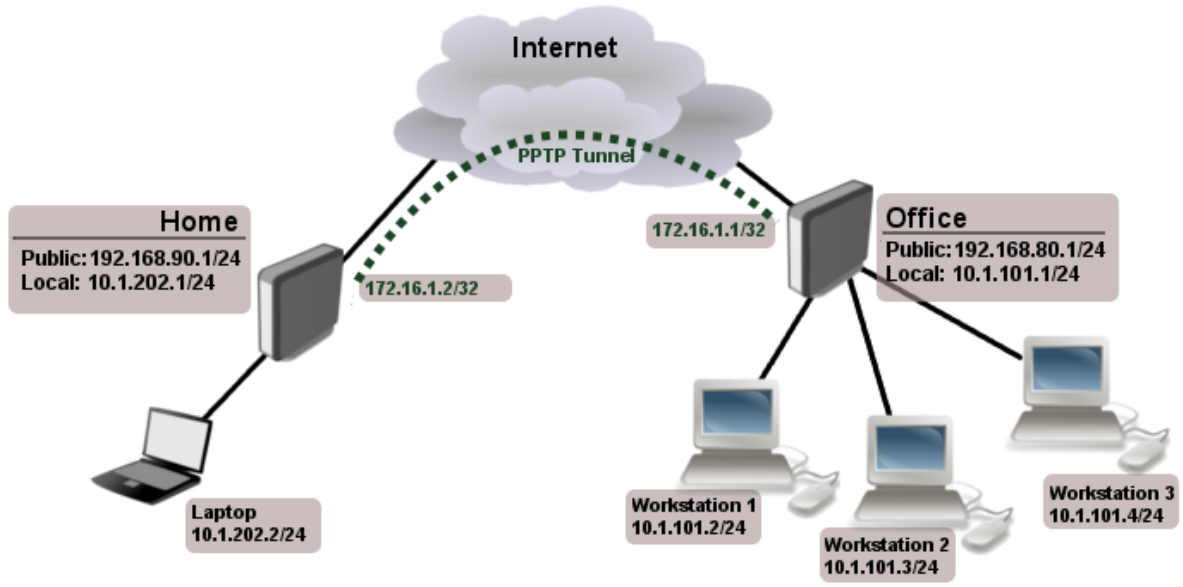
```
/interface pptp-server server set enabled=yes
/interface pptp-server server print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
  keepalive-timeout: 30
    default-profile: default
```

PPTP klient sa bude pripájať na verejnú adresu smerovača 192.168.80.1. Spôsob, akým sa vytvorí PPTP spojenie v danom operačnom systéme, v tomto príklade neuvádzame.

Predpokladajme, že spojenie sa úspešne nadviazalo. Ak by sme skúsili vykonať príkaz ping na niektorú z pracovných staníc, budeme neúspešný. Je to z dôvodu, že sa neodosiela ARP komunikácia. Riešením je zapnutie ARP proxy na lokálnom rozhraní Office.

```
/interface ethernet set Office arp=proxy-arp
/interface ethernet print
Flags: X - disabled, R - running
  #    NAME                 MTU   MAC-ADDRESS         ARP
  0  R ether1               1500  00:30:4F:0B:7B:C1 enabled
  1  R ether2               1500  00:30:4F:06:62:12 proxy-arp
```

Ďalším typickým príkladom je prepojenie dvoch lokálnych sietí. Schematicky sú znázornené na Obrázku 77.



Obrázok 77. Prepojenie dvoch lokálnych sietí

Smerovače Office a Home sú pripojené do internetu cez rozhranie ether1, pracovné stanice a laptop sú pripojené na rozhranie ether2. Sieťová komunikácia z oboch lokálnych sietí je smerovaná cez PPTP klienta, preto nie sú v rovnakej broadcast doméne. Ak by sme ich chceli mať v rovnakej broadcast doméne, museli by sme použiť protokol BCP a následne nakonfigurovať premostenie (bridge) medzi PPTP tunelom a lokálnym rozhraním.

Prvým krokom je vytvorenie používateľa.

```
/ppp secret add name=Home service=pptp password=123 local-
     address=172.16.1.1              remote-address=172.16.1.2
     routes="10.1.202.0/24 172.16.1.2 1"
/ppp secret print detail
Flags: X - disabled
  0     name="Home" service=pptp caller-id="" password="123"
     profile=default
      local-address=172.16.1.1       remote-address=172.16.1.2
     routes=="10.1.202.0/24 172.16.1.2 1"
```

Po pripojení používateľa sa pridá do smerovacej tabuľky nový záznam.

```
/interface pptp-server server set enabled=yes
/interface pptp-server server> print
            enabled: yes
            max-mtu: 1460
            max-mru: 1460
               mrru: disabled
     authentication: mschap2
  keepalive-timeout: 30
    default-profile: default
```

Ďalším krokom je zapnutie PPTP servera na Office strane a konfigurácie klienta na Home strane.

```
/interface pptp-client add user=Home password=123 connect-
     to=192.168.80.1 disabled=no
/interface pptp-client print
Flags: X - disabled, R - running
 0             name="pptp-out1"   max-mtu=1460   max-mru=1460
     mrru=disabled connect-to=192.168.80.1 user="Home"
       password="123" profile=default-encryption add-default-
     route=no dial-on-demand=no
       allow=pap,chap,mschap1,mschap2
```

Posledným krokom je pridanie statickej cesty. Inak by Home sieť nebola dostupná.

```
/ip route add dst-address=10.1.101.0/24 gateway=pptp-out1
```

Záver

Potenciál, ktorý ponúka internet vecí, umožňuje vývoj veľkého množstva aplikácií. Týka sa to viacerých domén, v ktorých tieto aplikácie môžu zlepšiť kvalitu nášho života, napríklad v domácnosti, pri cestovaní, v súvislosti so zdravím, prácou, prostredím. Fenomén internetu vecí posúva hranice počtu pripojených zariadení k internetu. Zariadenia sa vzájomne spájajú častokrát aj bez využitia bezpečnostných protokolov. Informačná bezpečnosť je preto jedným z kľúčových faktorov pri komunikácii zariadení prostredníctvom internetu.

V týchto vysokoškolských učebných textoch sme prezentovali základy internetu vecí, jeho charakteristiky, vznik a technologický vývoj. Popísali sme vzťahy medzi internetom vecí a inými disciplínami, sumarizovali aplikácie internetu vecí. V druhej časti učebnice sme priniesli prehľad rozšírených anotácií predmetov, aplikačných príkladov a prípadových štúdií s použitím rôznych metód v oblasti internetu vecí, ktoré sme vytvorili a inovovali na slovenských vysokých školách v rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie.

V rámci národného projektu IT Akadémia – vzdelávanie pre 21. storočie sme otestovali a v praxi overili viacero aplikácií, ktoré vyplynuli z analýz prípadových štúdií jednotlivých oblastí internetu vecí. Zamerali sme sa na efektívnu integráciu zariadení novej generácie a smart zariadení do adaptívnych a robustných prepojených sietí a platforiem. To zahŕňa aj výzvy z oblasti sieťového manažmentu na vyriešenie problémov týkajúcich sa konektivity obrovského počtu nových zariadení pripojených bezdrôtovo k internetu vecí. Okrem aktuálnosti, táto oblasť poskytuje aj veľmi zaujímavé úlohy na riešenie s množstvom pozitívnych synergických efektov.

Použitá literatúra

ALBINO, V., BERARDI, U., & DANGELICO, R. M. (2015). Smart cities: Definitions, dimensions, performance, and initiatives. Journal of urban technology, 22(1), 3-21.

ASHTON, K. (2009). That ‘internet of things’ thing. RFID journal, 22(7), 97-114.

ASGHARI, P., RAHMANI, A. M., & JAVADI, H. H. S. (2019). Internet of Things applications: A systematic review. Computer Networks, 148, 241-261.

ATZORI, L., IERA, A., & MORABITO, G. (2010). The Internet of Things: A survey, Computer Networks, 54(15), 2787-2805.

BAGGIO, D., EMAMI, S., ESCRIVÁ, D., a kol. (2012). Mastering OpenCV with practical computer vision projects: Packt Publishing Ltd.

BALOGH, Z., & BALÁŽ, I. (2020) Optimizing of spatial activities monitoring using the raspberry Pi and RFID system. In: Vol. 1031 AISC. 4th International Conference on Intelligent Computing, Communication and Devices, ICCD 2018 (pp. 615-622): Springer.

BALOGH, Z., BIZIK, R., TURCANI, M., & KOPRDA, S. (2016). Proposal for Spatial Monitoring Activities Using the Raspberry Pi and LF RFID Technology. In Q. A. Zeng (Ed.), Wireless Communications, Networking and Applications, Wcna 2014 (Vol. 348, pp. 641-651).

BALOGH, Z., KOPRDA, S., & TURCANI, M. (2019). Motion detection using HD camera of microcomputer raspberry Pi. Paper presented at the 11th IEEE International Conference on Application of Information and Communication Technologies, AICT 2017.

BALOGH, Z., & TURCANI, M. (2016). Complex Design of Monitoring System for Small Animals by the Use of Micro PC and RFID Technology. In A. ElOualkadi, F.

Choubani, & A. ElMoussati (Eds.), Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015, Vol 1 (Vol. 380, pp. 55-63).

BRAHMBHATT, S. (2013). Practical OpenCV: Apress.

CHUI, M., LOFFLER, M., & ROBERTS, R. (2010). The internet of things. McKinsey Global Institute.

DA XU, L., HE, W., & LI, S. (2014). Internet of things in industries: A survey. IEEE Transactions on industrial informatics, 10(4), 2233-2243.

DESIGN, P. (2007). FDX-B Animal Identification Protocol description Retrieved from http://www.priority1design.com.au/fdx-b_animal_identification_protocol.html

DIMITROV, D. V. (2016). Medical internet of things and big data in healthcare. Healthcare informatics research, 22(3), 156-163.

DZUBUR, E., LI, M., KAWABATA, K., a kol. (2015). Design of a smartphone application to monitor stress, asthma symptoms, and asthma inhaler use. Ann Allergy Asthma Immunol. 2015;114(4):341–342.e2.

HORÁČEK, P. (2012, 2013). Raspberry π I. – Úvod. Retrieved from http://www.linuxsoft.cz/article.php?id_article=1937

HOWERTON, C. L., GARNER, J. P., & MENCH, J. A. (2012). A system utilizing radio frequency identification (RFID) technology to monitor individual rodent behavior in complex social settings. Journal of Neuroscience Methods, 209(1), 74-78.

HUDEC, M. (2016). Inteligentné budovy s asistentom pre nevidiacich. Acta Informatica Pragensia, 5(1), 4-17.

JAIN, A. K., & LI, S. Z. (2011). Handbook of face recognition (Vol. 1): Springer.

KRITZLER, M., LEWEJOHANN, L., & KRÜGER, A. (2007). Analysing movement and behavioural patterns of laboratory mice in a semi natural environment based on data collected via RFID-technology. Paper presented at the CEUR Workshop Proceedings.

KRUSE, C.S., KOTHMAN, K., ANEROBI, K., a kol. (2016). Adoption factors of the electronic health record: a systematic review. JMIR Med Inform. 4(2):e19

KRUSHINITSKIY, P., & SZIEBIG, G. (2013). Review of open source computing devices for iSpace in production workshops. Paper presented at the 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom).

LINNENBRINK, M., & VON MERTEN, S. (2017). No speed dating please! Patterns of social preference in male and female house mice. Frontiers in Zoology, 14(1).

MINERVA, R., BIRU, A., & ROTONDI, D. (2015). Towards a definition of the Internet of Things (IoT). IEEE Internet Initiative, 1(1), 1-86.

MINOLI, D., SOHRABY, K., & OCCHIOGROSSO, B. (2017). IoT considerations, requirements, and architectures for smart buildings—Energy optimization and next-generation building management systems. IEEE Internet of Things Journal, 4(1), 269-283.

MIORANDI, D., SICARI, S., De PELLEGRINI, F., & CHLAMTAC, I. (2012). Internet of things: Vision, applications and research challenges. Ad hoc networks, 10(7), 1497-1516.

PLAGERAS, A. P., PSANNIS, K. E., STERGIOU, a kol. (2018). Efficient IoT-based sensor BIG Data collection–processing and analysis in smart buildings. Future Generation Computer Systems, 82, 349-357.

PORNPANOMCHAI, C., LIAMSANGUAN, T., & VANNAKOSIT, V. (2008). Vehicle detection and counting from a video frame. Paper presented at the 2008 International Conference on Wavelet Analysis and Pattern Recognition.

RASHMI, R., & NAMRATA, D. (2015). A Review on Comparison of Face Recognition Algorithm Based on Their Accuracy Rate. International Journal of Computer Sciences and Engineering, 3(2), 40-44.

RAY, P. P. (2018). A survey on Internet of Things architectures. Journal of King Saud University-Computer and Information Sciences, 30(3), 291-319.

ROSEBROCK, A. (Producer). (2016, 4 18). pyimagesearch.com. Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3. Retrieved from http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/

SALAZAR, J., SILVESTER, S. (2015). Internet vecí. Praha. České vysoké učení technické v Praze. 33 s.

SISINNI, E., SAIFULLAH, A., HAN, S. a kol. (2018). Industrial internet of things: Challenges, opportunities, and directions. IEEE Transactions on Industrial Informatics, 14(11), 4724-4734.

STANKOVIČ, J. A. (2014). Research directions for the internet of things. IEEE Internet of Things Journal, 1(1), 3-9.

ŠIKUDOVÁ, E., ČERNEKOVÁ, Z., BENEŠOVÁ, W. a kol. (2013). Počítačové videnie. Detekcia a rozpoznávanie objektov. 397.

TAN, L., & WANG, N. (2010). Future internet: The internet of things. In 2010 3rd international conference on advanced computer theory and engineering (ICACTE) (Vol. 5, pp. V5-376). IEEE.

TENG, H., LIU, Y., LIU, A., a kol. (2019). A novel code data dissemination scheme for Internet of Things through mobile vehicle of smart cities. Future Generation Computer Systems, 94, 351-367.

VERMESAN, O., FRIESS, P., GUILLEMIN, P., a kol. (2011). Internet of things strategic research roadmap. Internet of things-global technological and societal trends, 1(2011), 9-52.

VUJOVIĆ, V., & MAKSIMOVIĆ, M. (2015). Raspberry Pi as a Sensor Web node for home automation. Computers & Electrical Engineering, 44, 153-171. doi:http://dx.doi.org/10.1016/j.compeleceng.2015.01.019

VUZA, D. T., & FROSCH, R. (2010). RFID readers for the HDX protocol — Design, simulation and testing. Paper presented at the 2010 IEEE 16th International Symposium for Design and Technology in Electronic Packaging (SIITME).

WORTMANN, F., & FLUCHTER, K. (2015). Internet of things. Business & Information Systems Engineering, 57(3), 221-224.

ZANELLA, A., BUI, N., CASTELLANI, A., a kol. (2014). Internet of things for smart cities. IEEE Internet of Things journal, 1(1), 22-32.

XIA, F., YANG, L. T., WANG, L., a kol.. (2012). Internet of things. International journal of communication systems, 25(9), 1101.

XIAOGUANG, L., YUNHONG, W., & JAIN, A. K. (2003). Combining classifiers for face recognition. Paper presented at the Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on.

Internet vecí a jeho aplikácie
Vysokoškolský učebný text

Autori: RNDr. Ľubomír Antoni, PhD., doc. Ing. Peter Ševčík, PhD., prof. Ing. Iveta Zolotová, CSc., Ing. Peter Papcun, PhD., doc. Dr. Ing. Ján Vaščák, Ing. Miroslav Biňas, PhD., Ing. Tomáš Kanócz, doc. Ing. Jaroslav Porubän, PhD., doc. Ing. Martin Tomášek, PhD., Ing. Dominik Lakatoš, PhD., RNDr. Ján Skalka, PhD., doc. Ing. Zoltán Balogh, PhD., PaedDr. Peter Švec, PhD., RNDr. František Galčík, PhD., RNDr. Miroslav Opiela, RNDr. PhDr. Peter Pisarčík, doc. Ing. František Jakab, PhD., doc. RNDr. Dušan Šveda, CSc.

Vydavateľ: Univerzita Pavla Jozefa Šafárika v Košiciach
Vydavateľstvo ŠafárikPress

Rok vydania: 2020
Počet strán: 166
Rozsah: 6,3 AH
Vydanie: prvé

ISBN 978-80-8152-911-5 (e-publikácia)