Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Ambientní inteligence v pracovním prostředí

Bakalářská práce

Autor: Jakub Kól
Studijní obor: Aplikovaná informatika, AI3

Vedoucí práce: Ing. Karel Mls, Ph.D.

Hradec Králové Duben 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 17.8.2016 Jakub Kól

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Karlu Mlsovi, Ph.D. za metodické vedení práce, vstřícnost při konzultacích a cenné rady, které mi pomohly tuto práci dokončit.

Anotace

Hlavním výzkumným cílem této práce je aplikování prvků ambientní inteligence v pracovním prostředí, za účelem zajištění autonomního chování tohoto pracovního prostředí. Aby bylo možno pracovní prostředí označit jako autonomní, musí být především chápáno jako prostředí inteligentní. Inteligentní prostředí je tvořeno senzory (teplotní, vlhkostní, apod.) a aktuátory (automatické ovládání oken, rolet, dveří, apod.). Diskutována byla řada metod a přístupů, z nichž byla nakonec vybrána metoda zahrnující poměrně známou techniku z oblasti strojového učení - umělé neuronové sítě (ANN). Následně byla vytvořena ANN, umožňující v daném prostředí hledat vzory chování člověka a pomocí nich pak predikovat budoucí stavy systému.

Annotation

Title: Ambient intelligence in workplaces

The main objective of this research is to apply elements of the ambient intelligence in a workplace to ensure the autonomous behaviour of the workplace. First, a workplace needs to be considered as an intelligent. The intelligent workplace includes sensors (such as thermal sensors, moisture sensors, etc.) and actuators (such as automatical controlling of windows, roller blinds, doors, etc.). Second, several related methods and approaches have been reviewed and subsequently an appropriate method has been selected. The method is associated with the field of machine learning and includes artificial neural networks (ANN). Finally, we created an ANN able to search for the human behavioral patterns in the given workplace. Using these patterns, ANN is able predict the future states of the workplace.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Ambientní inteligence je v současnosti předmětem intenzivního výzkumu a mnoha diskuzí. Lze ji chápat jako odvětví umělé inteligence, jehož předmětem je zkoumání a vývoj tzv. inteligentních prostředí. Inteligentním prostředím je myšleno prostředí, které je schopné aktivně působit nezávisle na uživateli a přizpůsobovat svůj stav jeho potřebám. Příkladem může být regulace teploty v místnosti na základě dat ze senzorů, které mohou monitorovat tělesnou teplotu uživatele či jiné důležité vlastnosti. Na základě těchto dat by pak systém měl být schopen učinit rozhodnutí a přizpůsobit stav daného prostředí tak, aby byl pro uživatele co nejlepší.

Nabízí se však otázka, jak toto autonomní chování inteligentního řídicího systému zajistit. Cílem této práce bylo nalezení odpovědi právě na tuto otázku. Zpočátku bylo zvažováno řešení pomocí multiagentního systému, které s sebou však nese zejména vysokou komplexnost. Po rozsáhlých diskuzích byl nakonec zvolen přístup, který je založen na strojovém učení.

Strojové učení je podoblastí umělé inteligence a zároveň ideální nástroj k vytváření stochasticky orientovaných modelů. Techniky strojového učení mohou být využity jak k predikci spojitých hodnot, tak i ke klasifikaci dat do předem definovaných tříd. Dále je třeba rozlišovat tzv. učení s učitelem a učení bez učitele. První technika je založena na formulování predikcí z již získaných a předpřipravených dat, druhá pak hledá v datech souvislosti bez dalších znalostí. V rámci práce je využit známý a velice efektivní přístup – tzv. umělé neuronové sítě, dále jen ANN. Data ze senzorů reprezentují vstupní hodnoty predikce a klasifikační třídy jsou reprezentovány jednotlivými aktuátory. S pomocí ANN jsme poté schopni formulovat predikce stavů prostředí. Celý tento koncept je doplněn deterministickým rozhodovacím modelem, který zajišťuje chod klíčových částí systému včetně řešení krizových situací.

2 Ambientní inteligence

Vývoj v oblasti informačních technologií a informatiky postupoval poměrně rychle již od svého počátku. Málokdo si dnes vzpomene na legendární sálové počítače, které sloužily výhradně pro složité výpočty ve vědě, či průmyslu. V současnosti se vývoj ubírá směrem co největší miniaturizace při zachování co největšího výkonu. Začínají se objevovat mobilní telefony, které disponují mikroprocesory čítající osm výpočetních jader, objevují se jednodeskové miniaturní počítače o velikosti kreditní karty - např. *Raspberry PI*[1]. Výpočty jsou prováděny všude kolem nás, aniž bychom to tušili.

Vzhledem k velkému množství zařízení, které jsou schopny provádět výpočty, vyvstává otázka, jak bychom mohli všudypřítomných zařízení, schopných výpočtů, využít. Dokázali bychom tato zařízení nějakým způsobem zkoordinovat za účelem kooperativní činnosti? Mohou nám tato zařízení v daném prostředí nějakým způsobem „ulehčit práci"? Podobnými otázkami se vědci zabývali již v roce 2001, kdy evropská skupina ISTAG popsala v [1] 4 možné scénáře inteligentních prostředí.

Tyto 4 scénáře se staly jakousi počáteční vizí ambientní inteligence. Autoři v nich popisují, jak by mohla inteligentní prostředí vypadat v roce 2010. Zaměřují se na klíčové aspekty těchto prostředí a zejména jaké služby by mohla uživatelům poskytovat. I přes stáří příspěvku však dnes můžeme s jistotou konstatovat, že vize představené v tomto příspěvku zatím nebyly v plné míře zrealizované. Konkrétně se jedná o tyto 4 scénáře popisující příslušná témata:

- 'Maria' – Road Warrior - návrh inteligentního podpůrného systému pro manažery
- 'Dimitrios' and the Digital Me' (D-Me) - analogie předchozího scénáře
- Carmen: traffic, sustainability & commerce - vize inteligentní domácnosti
- Annette and Solomon in the Ambient for Social Learning – představa inteligentních prostor pro výuku

[1] https://www.raspberrypi.org

2.1 Definice ambientní inteligence

Ambientní inteligence je založena především na tzv. všudypřítomných výpočtech. Zařízeními, která by byla těchto výpočtů schopná, se zabýval v 90. letech tým inženýrů americké výzkumné společnosti PARC, spadající pod společnost Xerox. Byl jím Mark Weiser, který ve svém příspěvku [4] v časopise Scientific American popsal vývoj zařízení, která by značně ulehčila práci moderního člověka. Taková zařízení přitom nejsou nikterak odlišná od běžných věcí. Autor zmiňuje elektronické tabule, počítače velikosti listu papíru formátu A4, aj. Důraz však klade na propojení těchto zařízení do sítě tak, aby byla schopná vzájemné spolupráce. Weiser tedy nastínil myšlenku integrace mikroprocesorů do rozličných předmětů denní potřeby. Taková zařízení volně pojmenoval jako tzv. „mizející počítače".

Tyto myšlenky společně formují dnešní definici ambientní inteligence. Můžeme ji chápat jako soustavu vzájemně propojených zařízení, které disponují schopností provádět výpočty prostřednictvím integrovaných mikroprocesorů. Tato soustava je pak integrována do běžných prostředí, kde působí nezávisle na člověku s cílem poskytovat mu co největší podporu při pracovních či běžných činnostech. Samozřejmostí je možnost komunikace člověka s takovým systémem pomocí důmyslných rozhraní. Může se jednat například o ovládání pomocí hlasu, či gest. Prostředí, které disponuje tímto inteligentním řídicím systémem, můžeme nazývat tzv. *chytrým prostředím*[2].

Formální definici ambientní inteligence formulovali v [3] pracovníci výzkumného sektoru firmy Philips:

„Digitální prostředí, které je adaptivní, responzivní a citlivé na přítomnost člověka.".

Autoři článku [2] definují ambientní inteligenci jako:

„Digitální prostředí, které proaktivně, ale v rozumné míře, podporuje lidi v jejich každodenním životě". Ve stejném článku jsou také definovány základní vlastnosti, kterými by se prostředí s inteligentním řídicím systémem mělo vyznačovat. Jedná se o následující vlastnosti, které budou dále popsány: citlivost, responzivnost, adaptivita, transparentnost, všudypřítomnost, inteligence.

[2] Přeloženo z anglického sousloví „smart environment".

- **Citlivost**
  Systém rozpozná činnost uživatele pomocí senzorů umístěných v prostředí. Příkladem může být citlivost na přítomnost uživatelů v určité místnosti.

- **Responzivnost**
  Systém je schopen pomocí rozhodovacích algoritmů reagovat na chování uživatele, případně na jeho požadavky vůči systému.

- **Adaptivita**
  Systém umožňuje jistou formu přizpůsobení. Toho může být dosaženo prostřednictvím uživatelských nastavení systému či využitím vybraných metod umělé inteligence a strojového učení.

- **Transparentnost**
  Systém je plně integrován do prostředí. Uživatel pak přítomnost systému považuje za přirozenou součást prostředí.

- **Všudypřítomnost**
  Systém plně pokrývá problémovou doménu.

- **Inteligence**
  Systém umí predikovat budoucí stavy prostředí na základě již získaných dat. V práci je diskutována aplikace umělých neuronových sítí pro prediktivní ovládání aktuátorů, kterými dané prostředí disponuje. Spojení ambientní inteligence a strojové učení diskutují autoři [6], [18], [19].

Shrneme-li veškeré dosavadní poznatky, je ambientní inteligence spojením různorodých technologií a přístupů. Chování systému je zajištěno metodami umělé inteligence, jednotlivé prvky systému pak spolu komunikují pomocí počítačových sítí. Stav systému je monitorován prostřednictvím soustavy senzorů, akce jsou pak prováděny díky soustavě aktuátorů. Veškerá činnost probíhá nezávisle na uživateli, případně je umožněna komunikace uživatele se systémem pomocí předem

definovaného komunikačního protokolu. Takový protokol může mít formu hlasové komunikace, či komunikace pomocí neverbálních gest.

2.2 Ambientní inteligence v kontextu pracovních prostředí

Pracovním prostředím je již od starověku místo, kde člověk stráví značnou část svého života. Je tedy zásadní, aby takové místo nabízelo pracujícím za daných podmínek dostatečný komfort a především bezpečnost práce. Mezi další klíčové faktory pak zajisté patří efektivita práce, která přímo souvisí s předchozími dvěma faktory. Vzhledem k velmi rozsáhlému vývoji v oblasti informačních technologií se nabízí otázka, zda bychom dokázali tyto technologie využít k vylepšení stávajících pracovních prostředí a tím i k zefektivnění pracovních procesů. Možným řešením, které je zároveň odpovědí na tuto otázku, je aplikování prvků ambientní inteligence do pracovních prostředí. Toho docílíme integrací inteligentních řídicích systémů do těchto prostředí.

Uveďme příklady vlastnostmi, kterými by inteligentní řídicí systém měl disponovat. Mikulecký v [16] zmiňuje následující vlastnosti.

- **Poskytování informací**
  Systém umožňuje přístup k interní znalostní databázi. Tato vlastnost je klíčová zejména u pracovních prostředí s větším počtem osob. Výhodou je snadná přístupnost databáze v rámci celého pracovního prostředí, což vede k vylepšení vzdělávacího procesu pracovníků.
- **Zajištění technické podpory**
  Systém monitoruje vybrané technické prostředky, které jsou využívány v rámci daných pracovních procesů. V případě jejich poškození pak o vzniklé situaci informuje uživatele a automaticky zajistí odborný servis, je-li to možné.

Vzhledem k předchozím vlastnostem lze s jistotou tvrdit, že inteligentní řídicí systém integrovaný do pracovního prostředí s sebou přináší markantní zkvalitnění a zefektivnění pracovních procesů. Zajišťuje bezpečnost práce, dodržování bezpečnostních zásad a pravidel daného pracoviště. Veškeré nadbytečné rutinní činnosti jsou plně automatizovány díky adaptivnímu charakteru systému.

V případě nutnosti získání dodatečných informací je uživateli poskytnuta možnost využití interní znalostní databáze. Klíčové součásti prostředí jsou monitorovány a jejich stav je pravidelně vyhodnocován. Všechny tyto aspekty lze považovat za jasné klady takového řídicího systému, musíme však počítat s možnými riziky.

2.2.1 Možná rizika inteligentních pracovních prostředí

Informační bezpečnost lze v informatice považovat za samostatný obor. V současnosti je právě tomuto oboru věnována značná pozornost. Je nutné zajistit v inteligentních řídicích systémech dostatečnou míru zabezpečení tak, aby nedošlo ke ztrátě důležitých informací či narušení klíčových součástí systému a tím i soukromí uživatele. Mikulecký v [5] uvádí jako jedno z největších rizik právě narušení soukromí uživatele. Další otázkou, na kterou je třeba odpovědět, je míra autonomnosti inteligentního prostředí a zda právě autonomnost řídicího systému nemůže uživatele ohrozit.

Každý řídicí systém bude připojen prostřednictvím počítačových sítí do Internetu. Je tedy nutné počítat s možnými bezpečnostními riziky a napadením systému. Systém je třeba rozvrstvit do více úrovní dle nutnosti připojení k internetu. Cílem takového přístupu je zajistit nezávislost klíčových součástí na Internetu. Útočník tak v případě úspěšného průniku do systému nebude mít možnost ovlivnit chod jádra systému a jeho škody tak budou značně minimalizovány. Veškerý síťový provoz musí být zabezpečen dle příslušných bezpečnostních norem.

V následujícím odstavci bude diskutována otázka autonomnosti řídicího systému. Jedním z dalších rizik inteligentního prostředí je paradoxně jeho „inteligence“. V případě systému, jehož rozhodovací model je částečně založen na pevně definovaných pravidlech, je nutno definovat základní sadu pravidel, podle kterých bude jednat v případě krizových či jiných nestandardních situací. Taková pravidla je třeba důkladně prodiskutovat se specialisty na bezpečnost práce. Tento přístup zajistí deterministické chování systému v případě výskytu právě takových, pro systém ve většině případů, nestandardních situací. Příkladem může být požár na pracovišti. Pracoval-li by systém zcela autonomně a nebyl na tuto událost připraven, mohlo by dojít k zásadním problémům při evakuaci osob a řešení celé

situace. Pevně definovaná pravidla pak v tomto případě zajistí optimální řešení situace. Podoba a rozsah sady pravidel je přímo závislý na konkrétním prostředí, do kterého je inteligentní řídicí systém integrován. Autoři [6] poukazují na absenci inteligentních rozhodovacích subsystémů, nicméně optimálním řešením je spojení pravidlového rozhodování, které je doplněno prvky umělé inteligence.

V souvislosti s tímto rizikem je také třeba definovat základní uživatelská práva v systému. Je nutné jasně specifikovat, k jakým částem systému může uživatel přistupovat. Příkladem jsou uživatelská práva pro obsluhu jednotlivých aktuátorů v systému. Disponuje-li uživatel dostatečnými právy, může danou součást systému ovládat a případně přenastavit dle jeho preferencí. Adaptivní a autonomní charakter systému je tak doplněn o uživatelem definovaná pravidla a uživatelské akce.

2.3 Konkrétní scénář – Inteligentní kancelář

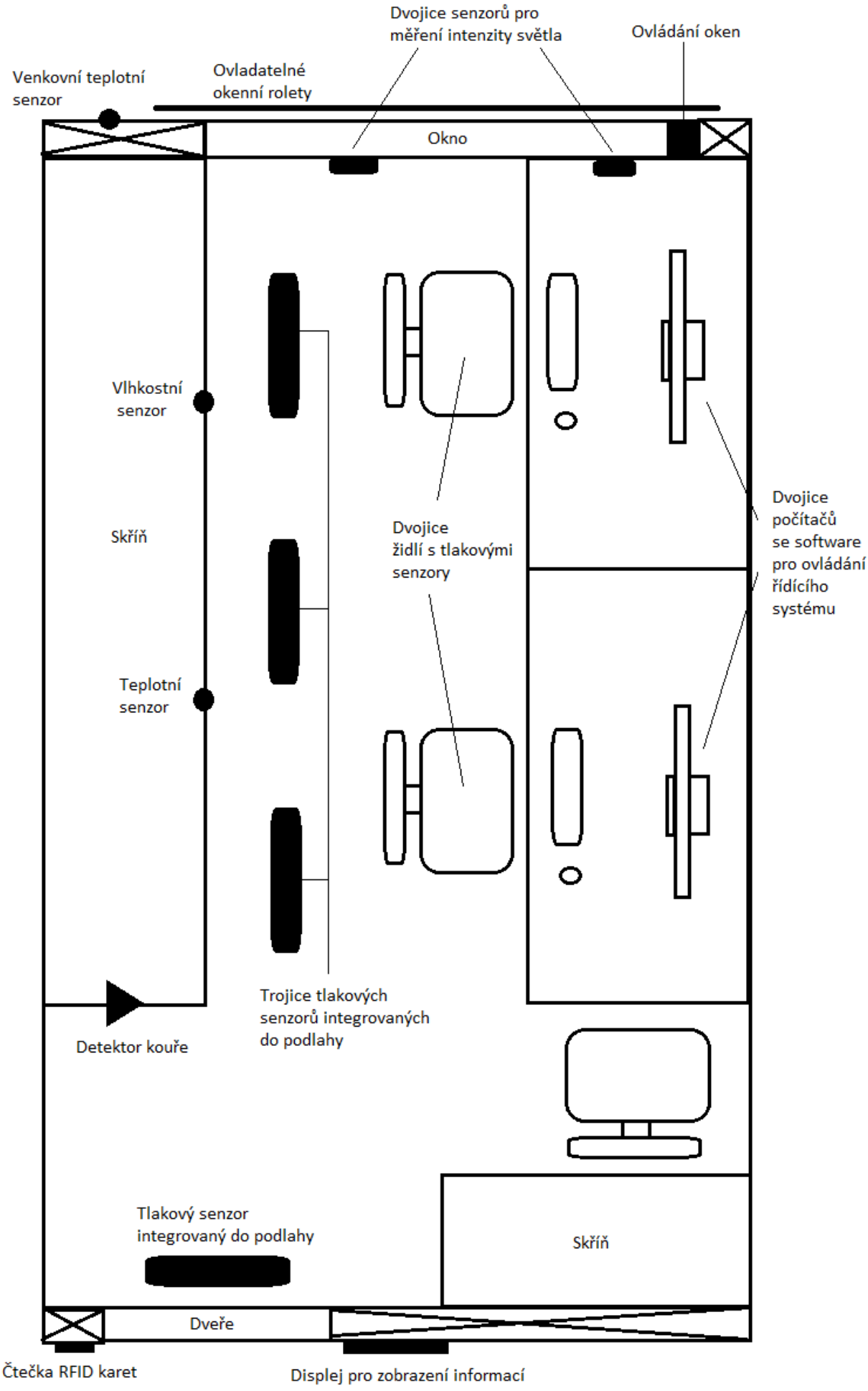
Ambientní inteligence může být integrována do téměř každého pracovního prostředí. V rámci konkrétního scénáře bude představena vize inteligentní kanceláře vysokoškolského pedagoga.

Pedagogové jsou základním prvkem každého vysokoškolského systému. Jejich práce není zaměřena pouze na vzdělávání studentů, ale značnou část pokrývá především vědecká činnost. Bylo by žádoucí, aby pracovní prostředí podporovalo právě tuto část pracovního procesu. Mezi funkce inteligentního řídicího systému, který je do kanceláře integrován, patří automatizace rutinních činností, analýza probíhajícího pracovního procesu či predikce návštěv studenty a pedagogy. Poslední funkce slouží především k nalezení optimálního času pro vědeckou činnost, tedy nalezení časového úseku s minimem návštěv vzhledem k rozvrhu pedagoga.

2.3.1 Senzory a aktuátory

Uvažujme nyní inteligentní kancelář, jejíž schéma je přiloženo na následující straně. V kanceláři je rozmístěna řada senzorů pro monitorování pohybu pracovníka. Konkrétně se jedná o autorizaci pomocí čtečky identifikačních RFID karet, tlakové senzory v židli a podlaze. Monitorování pracovníka v rámci budovy je pak zajištěno prostřednictvím „chytrého“ náramku. Z bezpečnostních senzorů se jedná o detektor kouře, teplotní a vlhkostní senzory.

Mezi aktuátory patří ovládání oken, okenních rolet, dveří a osvětlení. Veškeré aktuátory lze ovládat dálkově pomocí ovladače, webové aplikace běžící v interní bezdrátové síti, nebo prostřednictvím aplikace v mobilním telefonu. Posledním prvkem je displej instalovaný u vchodových dveří, který zobrazuje informace o aktuální dostupnosti pedagoga. Displej disponuje dotykovým ovládáním, které umožňuje studentům v případě nepřítomnosti pedagoga zobrazit jeho rozvrh a aktuální polohu v rámci budovy školy. Je-li pedagog přítomen, zobrazí displej informaci o tom, zda je zaneprázdněn a nepřijímá návštěvy, nebo je připraven konzultovat. Stav na displeji může pedagog měnit pomocí hlasového ovládání nebo nechat rozhodnutí na řídicím systému, ten pak rozhodne na základě výsledků analýz pracovního procesu a provede predikci stavu, který následně zobrazí na displeji.



Obrázek 1: Schéma inteligentní kanceláře
(zdroj: autor)

2.3.2 Modelový případ užití

Karel, vysokoškolský pedagog, právě přijíždí na parkoviště poblíž vysoké školy, kde působí. U vjezdu otevírá závoru pomocí identifikační karty. Čtecí zařízení ho úspěšně identifikuje, umožní mu vjezd na parkoviště a zašle signál do jeho kanceláře. Inteligentní řídicí systém, integrovaný v kanceláři, signál zpracuje a začíná připravovat kancelář pro uživatele, který je již na cestě.

Inicializace systému začíná kontrolním testem, který zkontroluje veškeré součásti systému a vybrané technické vybavení kanceláře. Na základě předchozích znalostí, které systém nashromáždil, pak adaptivní vrstva systému predikuje zapnutí Karlova počítače. Predikce je zhodnocena a počítač je následně zapnut. Identický proces pak řídicí systém provede pro všechny senzory a aktuátory v kanceláři. Po dokončení všech těchto dílčích procesů je kancelář plně nastavena dle Karlových preferencí a zvyklostí.

Karel vchází do budovy fakulty a chytrý náramek, který má na ruce, je zaznamenán řídicím systémem v kanceláři. Na displeji u dveří je aktualizován stav o jeho poloze a stavu - nedostupný / v budově. Jakmile přichází ke dveřím kanceláře, identifikuje se přes identifikační kartu prostřednictvím RFID čtečky, která je umístěna hned vedle dveří. Po vstupu do místnosti vidí, že kancelář je připravena přesně podle jeho zvyklostí, usedá tedy za pracovní stůl. Tlakový senzor na židli je aktivován a stav na displeji vedle dveří je ihned změněn na: zaneprázdněn / v budově.

Po vyřízení povinností pak manuálně, pomocí hlasového ovládání, mění svůj stav na: dostupný. Studenti tuto informaci vidí prostřednictvím webové aplikace a ví tedy, kdy přesně mohou svého pedagoga navštívit. Řídicí systém jejich návštěvy pravidelně zaznamenává a na základě těchto dat pak adaptivní vrstva Karlův stav v budoucnu automaticky predikuje. Po konzultacích Karel odchází na cvičení, která vede. Řídicí systém po jeho odchodu uzamyká kancelář a nastavuje veškeré aktuátory. V průběhu času, kdy vyučuje, je na displeji u dveří zobrazen jeho rozvrh. Po skončení výuky Karel odchází z budovy fakulty. Řídicí systém po jeho odchodu z kanceláře uzamyká dveře a přenastavuje aktuátory. U výjezdu z parkoviště se opět identifikuje prostřednictvím identifikační karty a čtecí zařízení opět zasílá

signál do jeho kanceláře. Řídicí systém signál zpracuje a provede pozastavení aktivní činnosti.

Řídicí systém opět zkontroluje veškeré své součásti a vybrané technické vybavení kanceláře. V případě nalezení závady tento nález ihned eviduje a předá záznam o závadě příslušnému oddělení, jež řešení těchto situací zajišťuje. Po ukončení kontrolních testů jsou veškeré aktuátory nastaveny do výchozích režimů a jejich činnost je pozastavena. Jediným aktivním prvkem je pak centrální modul, který má na starost řízení jednotlivých součástí systému. V úsporném bezpečnostním režimu pak čeká na další inicializační signál a celý scénář se opakuje.

3 Koncept inteligentního řídicího systému

V následující kapitole bude představen vlastní koncept inteligentního řídicího systému. Inteligentním řídicím systémem chápeme spojení hardware a software za účelem zajištění kompletního řízení inteligentního prostředí. Koncept je abstraktního charakteru, závěr kapitoly je věnován strojovému učení v kontextu ambientní inteligence, konkrétně problematice umělých neuronových sítí. V rámci praktické části pak byla umělá neuronová síť implementována a otestována.

3.1 Architektura systému

Architektura systému představuje širší pojem, který je primárně rozdělen na dva celky. Prvním je architektura systému z hlediska hardware, druhým celkem je softwarová, dále jen SW.

3.1.1 Dostupné technologie

Architektura systému představuje širší pojem, který je primárně rozdělen na dva celky. Prvním je architektura systému z hlediska hardware, druhým celkem je softwarová, dále jen SW, architektura systému.

V současnosti (2016) je situace v oblasti tzv. *embedded devices*[3] velice příznivá. Známé jsou především jednodeskové počítače, které umožňují realizaci zajímavých projektů. Taková zařízení dnes disponují dostatečnou výpočetní silou na to, aby zastávala klíčové role v řídicích systémech. Jednodeskový počítač Raspberry PI 2 je využit i v konceptu představeném v této práci, kde figuruje jako centrální výpočetní a řídící prvek. Z oblasti senzorů je třeba zmínit platformu Arduino, která je mezi vývojáři populární zejména díky cenové dostupnosti jednotlivých modulů a součástek. Vývojové desky Arduino umožňují připojení velkého množství senzorů a aktuátorů. Platformy Arduino je v tomto konceptu využito také.

Zaměříme-li se na SW architekturu systému, naskytne se nám poměrně rozsáhlý výčet technologií a s tím související důležitá rozhodnutí. Prvním je výběr programovacího jazyka pro implementaci jádra řídicího systému. Klíčovým

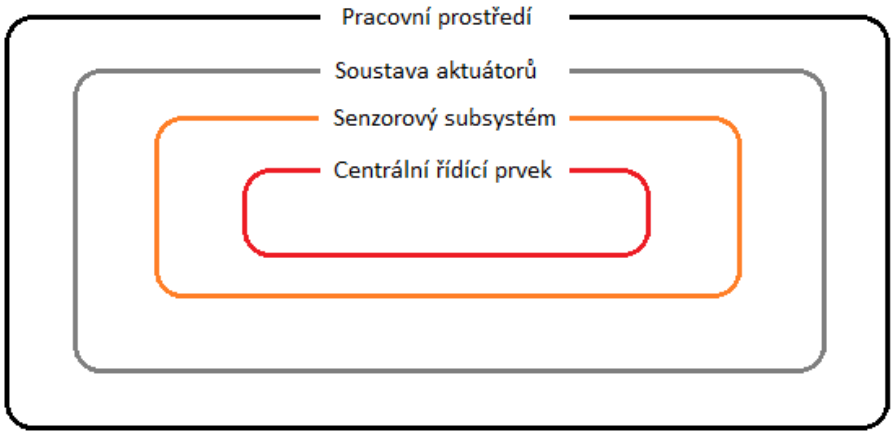
[3] Integrované zařízení s jasně definovaným primárním účelem.

aspektem je skutečnost, zda je daný jazyk multiplatformní a umožňuje agilní vývoj. Nemá téměř žádný smysl vyvíjet v úzce specializovaném jazyce, nemáme-li k tomu vážné důvody. Ze známých jazyků je mezi kandidáty objektově orientovaný programovací jazyk Java či Python. Vzhledem k zaměření řídicího systému na metody strojového učení je také nutné zvážit, zda existují pro daný jazyk nástroje pro implementaci algoritmů strojového učení. Oba zmíněné jazyky nabízí velice zajímavé nástroje. Konkrétně se jedná o framework *Neuroph*[4] pro programovací jazyk Java. Pro jazyk Python pak knihovna *pybrain*[5] [15]. Druhým rozhodnutím je výběr programovacího jazyka pro soustavu senzorů. Platforma Arduino poskytuje poměrně solidní nástroje pro programování obslužných skriptů. Jedná se o framework *Wiring*[6], který sestává z programovacího jazyka a integrovaného vývojového prostředí - *IDE*.

3.1.2 Hardwarová vrstva

HW vrstva je rozdělena na 3 základní celky:

- Centrální řídící prvek
- Senzorový subsystém
- Soustava akuátorů



Obrázek 2: Schéma rozvrstvení HW komponent
(zdroj: autor)

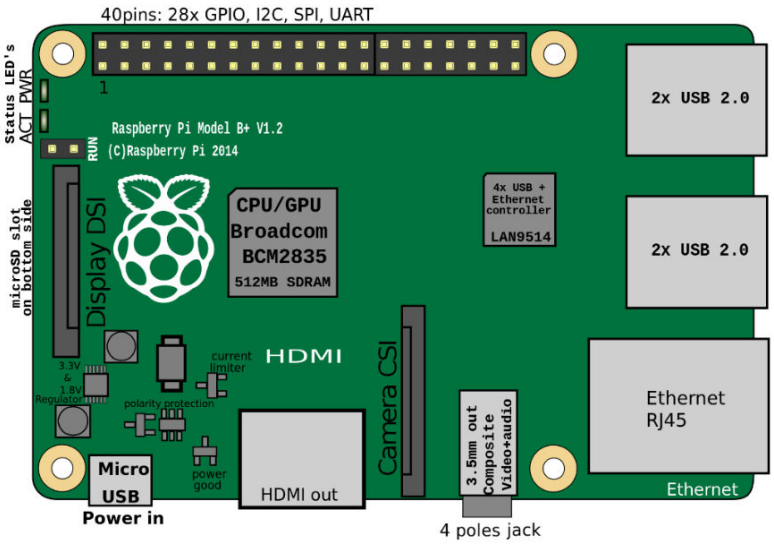
[4] http://neuroph.sourceforge.net
[5] http://pybrain.org
[6] http://wiring.org.co

Centrální řídící prvek

Jednodeskový počítač Raspberry Pi 2 je populární zejména díky výbornému poměru cena/výkon a rozsáhlé komunitě uživatelů a vývojářů. Pro účely centrálního prvku byl zvolen jednak díky příznivé ceně, ale především díky dostatečně vysokému výkonu.

Tabulka 1: Technické specifikace počítače Raspberry Pi 2, model B

CPU	ARM Cortex A7
Počet jader	4
Takt / jádro	900MHz, možno přetaktovat až na 1GHz
GPU	Videocore IV
RAM	1GB
Počet USB portů	4x USB 2.0
Slot na paměťové karty	microSD
Ethernet RJ45	10 / 100 MBit
Wifi	Ne, lze vyřešit přídavným Wifi modulem
OS	GNU/Linux
Rozměry	Šířka: 55mm, délka: 85mm



Obrázek 3: Ilustrace počítače Raspberry Pi B+. Rozmístění komponent na desce je totožné s Raspberry Pi 2, model B.
(zdroj: http://bitsnapper.com/wp-content/uploads/2015/09/Raspberry-Pi-2-Model-B-1GB-System-on-chip-diagram-circuit.jpg

Technické specifikace ukazují, o jak výkonné zařízení v poměru k jeho rozměrům se jedná. Mikroprocesor Cortex A7 disponuje čtyřmi výpočetními jádry, což ve výsledku znamená čtveřici vláken a zároveň možnost vícevláknového programování - paralelizace hraje u řídícího prvku významnou roli. O uchování dat se stará microSD karta. Ta však neslouží jako primární úložiště dat získaných ze senzorů, ale je vyhrazena výhradně pro potřeby centrálního prvku. K síti lze zařízení připojit pomocí klasického ethernetového portu o rychlosti 100Mbit. Jedná se o naprosto dostačující rychlost, která by neměla chod systému zásadně zpomalit. Pro připojení k bezdrátové síti Wifi lze využít přídavného modulu.

Operační systém Raspbian je distribucí operačního systému GNU/Linux, navrženou speciálně pro embedded zařízení. Jedná se o robustní a spolehlivý systém, který v rámci tohoto konceptu funguje v tzv. *headless módu*[7]. Připojení k zařízení v tomto módu je řešeno autorizovaným přístupem s využitím ssh protokolu. Samotné ovládání zařízení je pak realizováno prostřednictvím linuxového terminálu.

Primární role

Primární rolí centrálního prvku je kompletní správa inteligentního řídicího systému. Tu lze dále rozdělit na následující dílčí role:

- **Řízení soustavy aktuátorů**

  Veškeré aktuátory jsou prostřednictvím mikrokontrolerů ovládány skrze centrální prvek. Ten na základě rozhodnutí vytvořeného interním rozhodovacím modelem může měnit stav daných aktuátorů. Uživatelské rozhraní pro ovládání aktuátorů je řešeno webovou a mobilní aplikací.

- **Řízení sběru dat ze senzorového subsystému**

  Data ze senzorů jsou shromažďována a prostřednictvím centrálního prvku ukládána do relační databáze. Odtud jsou později využita pro účely predikce stavů soustavy aktuátorů. Hodnoty stavových proměnných jednotlivých senzorů lze získávat periodicky.

[7] Zařízení je připojeno do LAN sítě bez dodatečných periferií.

- **Správa neuronových sítí, provádění predikcí.**
  Umělé neuronové sítě jsou v uživatelem definovaných (případně výchozích) časových intervalech pravidelně přeučovány. Centrální prvek zajišťuje jak přeučení neuronových sítí, tak provádění predikcí pomocí naučených klasifikátorů.

- **Řízení testů pro kontrolu stavu připojených zařízení**
  Součástí centrálního prvku je provádění automatizovaných testů sloužících ke kontrole veškerého hardware řídicího systému. V případě nalezení závady je uživatel ihned informován a celá událost je zaznamenána.

Testování výkonu

V rámci testování výkonu CPU ARM Cortex A7 bylo zařízení Raspberry Pi 2 model B podrobeno zátěžovým testům. Pro účely tohoto testování byl napsán program v jazyce Java.

Ten umožňuje spuštění zátěžového testu zaměřeného na klasické maticové násobení, které je dáno následujícím vztahem (zde pro matice typu $2x2$):

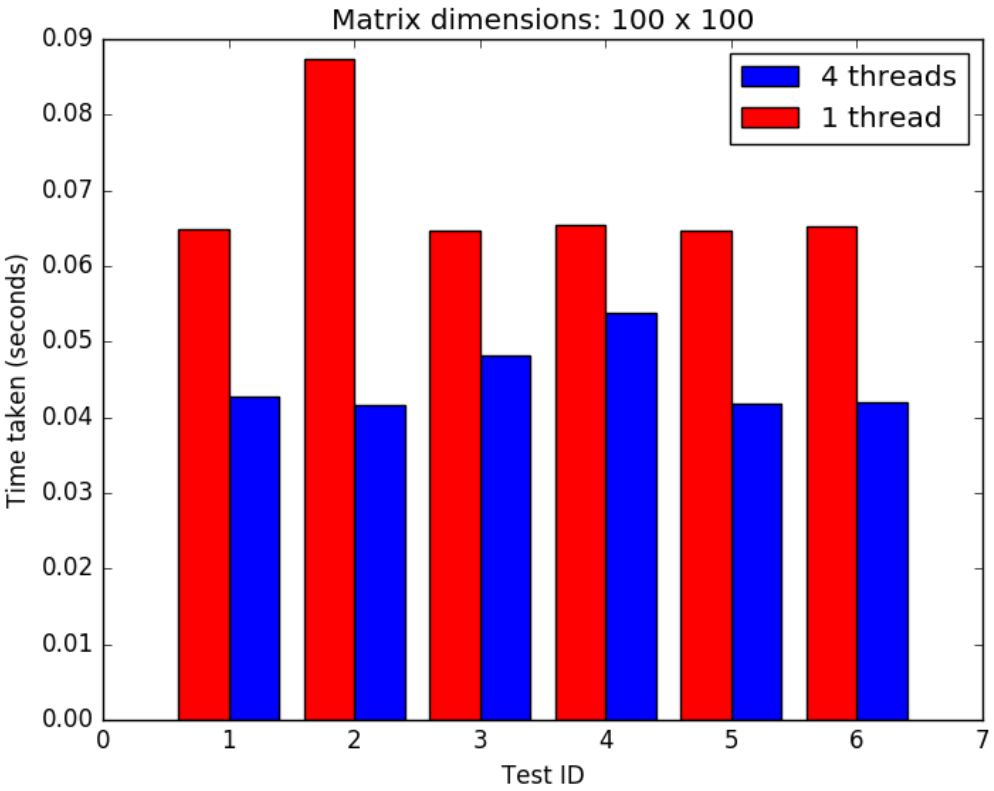
$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} * \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1} * b_{1,1} + a_{1,2} * b_{2,1} & a_{1,1} * b_{1,2} + a_{1,2} * b_{2,2} \\ a_{2,1} * b_{1,1} + a_{2,2} * b_{2,1} & a_{2,1} * b_{1,2} + a_{2,2} * b_{2,2} \end{bmatrix}$$

Testování bylo provedeno ve dvou fázích. První fáze byla cílena na paralelní výkon mikroprocesoru ARM Cortex A7. Pro tyto účely byla implementována paralelní verze standardního algoritmu maticového násobení.

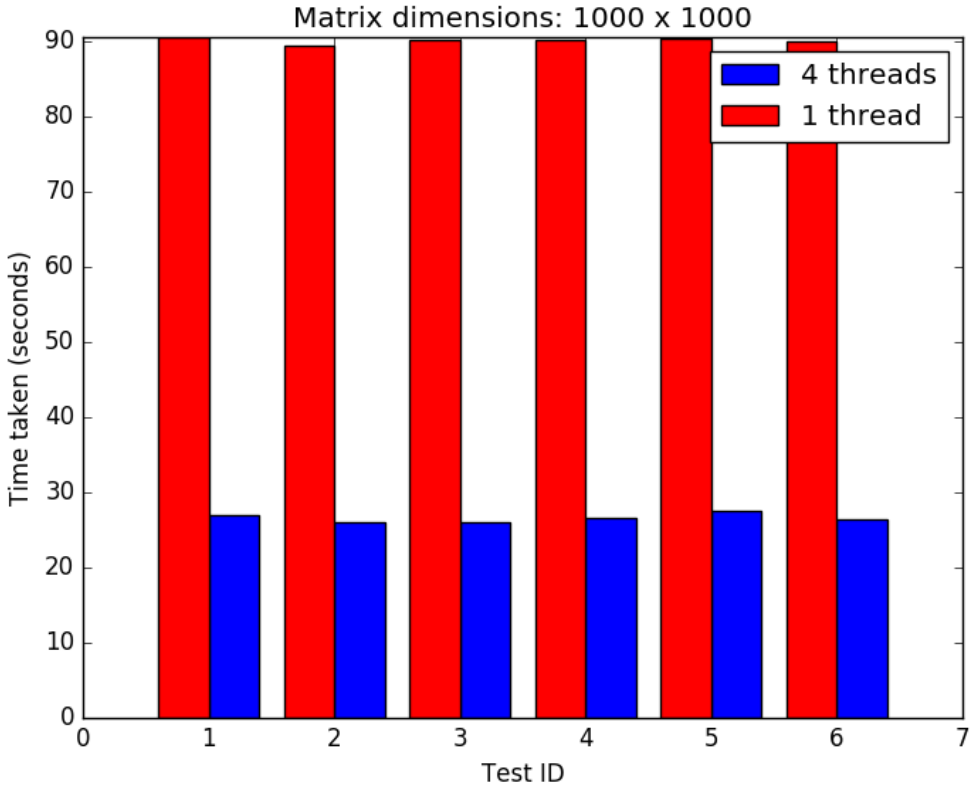
Celkem proběhlo 12 testů - 6 testů s náhodně generovanými vstupními maticemi o rozměrech 100 x 100 a následujících 6 s náhodně generovanými vstupními maticemi o rozměrech 1000 x 1000. Výsledky jsou vyjádřeny formou sloupcových grafů na další straně.

Tabulka 2: Parametry algoritmů zátěžového testování

Algoritmus mat. násobení	Standardní	Paralelizovaný
Asymptotická složitost	$O(n^3)$	$O(n^3)$
Datový typ test. dat	Čtvercové matice, prvky datového typu float	Čtvercové matice, prvky datového typu float
Rozsah hodnot	0.0f - 1000.0f	0.0f - 1000.0f
Počet alokovaných vláken	1	4
Teplota CPU (st. Celsia)	40 - 48	55 - 59



Obrázek 4: Výsledky zátěžových testů pro matice o rozměrech 100 x 100.
(zdroj: autor)



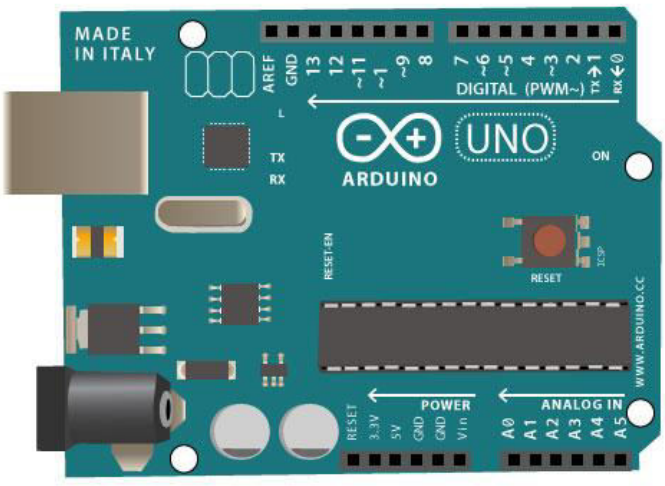
Obrázek 5: Výsledky zátěžových testů pro matice o rozměrech 1000 x 1000.
(zdroj: autor)

Senzorový subsystém, soustava aktuátorů

Pro účely sběru dat byla vybrána vývojová deska Arduino UNO, osazená mikrokontrolérem ATmega328P. Desku lze propojit s širokým spektrem senzorů pomocí vestavěných I/O pinů, či pomocí tzv. *shieldů*[8]. Výběr senzorů záleží na problémové doméně, tedy konkrétním prostředí, ve kterém je inteligentní řídicí systém nasazen.

Tabulka 3: Technické specifikace vývojové desky Arduino UNO

Mikrokontrolér	ATmega328P
Takt	16MHz
Flash paměť	32KB
SRAM	2KB
EEPROM	1KB
Počet digitálních I/O pinů	14
Počet USB portů	1x USB 2.0
Ethernet RJ45	Ano - dodatečný modul
Wifi	Ano - dodatečný modul
Rozměry	Šířka: 54mm, délka: 69mm



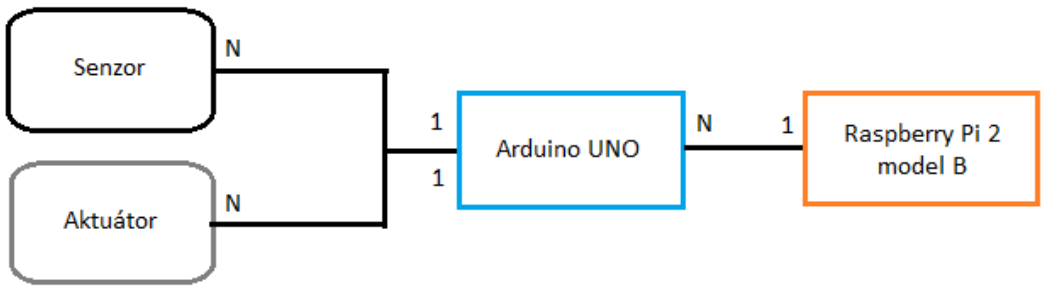
Obrázek 6: Ilustrace vývojové desky Arduino UNO
(zdroj: https://www.robomart.com/blog/wp-content/uploads/2015/09/arduino-uno.jpg)

[8] Doplňující modul rozšiřující vývojovou desku o dodatečnou funkcionalitu.

V technických specifikacích desky Arduino UNO lze vidět jasný smysl tohoto zařízení v kontextu inteligentního řídicího systému. Mikrokontrolér Atmega328P nedosahuje vysokého výpočetního výkonu. Je určen pro vývoj úzce specializovaných zařízení s nízkými nároky na výpočetní výkon, je tedy ideálním základem senzorového subsystému a soustavy aktuátorů.

Senzory je vhodné k desce připojit pomocí tzv. sensor shieldu - rozšiřujícího modulu, který desku doplňuje o dodatečné I/O porty. Samotná deska pak funguje jako „transportní prvek“ mezi senzorem a centrálním řídícím prvkem, který data dále zpracovává.

Připojení aktuátorů vyžaduje individuální přístup. Na trhu prozatím v roce 2016 není jednotná nabídka univerzálních ovládacích prvků. Je tedy nutné vzhledem ke konkrétnímu prostředí navrhnout a realizovat specifické ovládací prvky a ty následně prostřednictvím vhodných elektrotechnických součástek připojit k deskám Arduino. Konkrétní řešení ovládacích prvků je nad rámec této práce a nebude dále diskutováno.



Obrázek 7: Abstraktní schéma zapojení jednotlivých HW komponent systému.
(zdroj: autor)

Primární role

Primární role senzorového subsystému a soustavy aktuátorů byla již částečně naznačena. Každý senzor je stavovou proměnnou inteligentního prostředí. Na základě hodnot těchto proměnných jsou pak umělé neuronové sítě schopny provádět predikce budoucích stavů jednotlivých aktuátorů.

3.1.3 Softwarová vrstva

Obslužná aplikace centrálního řídícího prvku

Jádro inteligentního řídicího systému je tvořeno již zmiňovaným centrálním prvkem, pro ten bylo třeba navrhnout obslužný SW. V rámci konceptu byl pro tyto účely vybrán programovací jazyk Java. Ten je představitelem multiplatformního objektově orientovaného jazyka, umožňujícího vícevláknové programování. Existuje v několika verzích, pro zařízení Raspberry Pi 2 model B byla zvolena Java SE Embedded.

Paralelizace činností

Mikroprocesor ARM Cortex A7 podporuje paralelní běh až 4 výpočetních vláken. V rámci využití těchto prostředků byl navržen paralelní model rozvržení činností centrálního prvku. Následující tabulka ukazuje rozdělení činností systému mezi všechna 4 výpočetní vlákna:

Tabulka 4: Paralelizace činností

Výpočetní vlákno	Činnost
Thread #1	Řízení běhu systému, koordinace vláken 2 - 4
Thread #2	Rozhodování v systému
Thread #3	Řízení soustavy aktuátorů, realizace sběru dat ze senzorového subsystému
Thread #4	Správa umělých neuronových sítí, provádění predikcí

Základní moduly obslužné aplikace

Obslužná aplikace je členěna do modulů dle jejich zaměření. Každý z těchto modulů běží na separátním výpočetním vlákně - viz předchozí tabulka č. 4.

- **Modul pro řízení běhu systému**

  Tento modul je hlavní částí obslužné aplikace centrálního prvku. Jeho primárním úkolem je zejména koordinace činností v rámci centrálního prvku a dohled nad korektní funkčností systému, včetně základní diagnostiky HW komponent systému.

- **Modul pro podporu rozhodování**

  Rozhodovací model, který bude v této práci podrobněji diskutován, je soustředěn právě do modulu pro podporu rozhodování. V kompetenci tohoto modulu je kompletní rozhodování v rámci celého inteligentního řídicího systému, včetně řešení krizových situací.

- **Modul pro řízení sběru dat ze senzorů a ovládání aktuátorů**

  Již z názvu tohoto modulu je zřejmé jeho primární zaměření. Tím je realizace sběru dat ze senzorového subsystému. Data jsou poté pomocí persistentní vrstvy uchovávána v relační databázi pro účely strojového učení. Další činností, kterou tento modul realizuje je ovládání soustavy aktuátorů, úzce při tom spolupracuje s modulem pro správu umělých neuronových sítí. V rámci této spolupráce jsou pak aktuátory stochasticky ovládány pomocí predikcí jednotlivých stavů.

- **Modul pro správu umělých neuronových sítí**

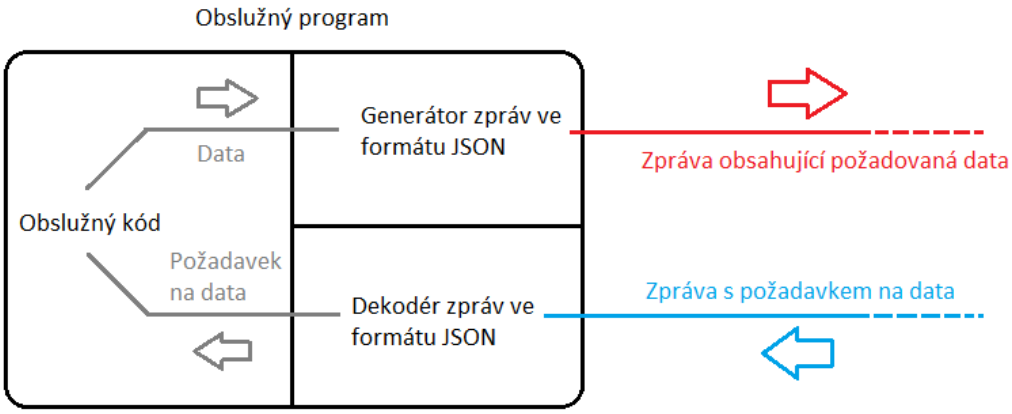
  Poslední z hlavních modulů zajišťuje správu umělých neuronových sítí, dále jen ANN, a realizaci predikcí stavů jednotlivých aktuátorů. Jak již bylo diskutováno v předchozím odstavci, úzce při tom spolupracuje s modulem č. 3. Problematice strojového učení pomocí ANN je v této práci věnována samostatná kapitola

Obslužné programy pro desky Arduino

Pro vývojové desky Arduino lze vyvíjet obslužné programy jak v jazyce C, tak především jazyce Wiring, který z něj vychází. V rámci tohoto konceptu bude využito právě frameworku Wiring, který se skládá jednak ze samotného jazyka, tak i vývojového prostředí pro snadné psaní kódu a obsluhu vývojových desek.

Pro každý senzor ze senzorového subsystému a každý aktuátor ze soustavy aktuátorů je nutno implementovat příslušný obslužný program. Struktura takového programu by měla být následující:

- První část: Kódování / dekódování zpráv ve formátu JSON.
- Druhá část: Obsluha senzorů / aktuátorů připojených k desce.



Obrázek 8: Schéma obslužného programu pro desky Arduino
(zdroj: autor)

Komunikace v systému

V předchozí kapitole věnované HW vrstvě systému bylo naznačeno abstraktní schéma zapojení HW komponent. Arduino UNO umožňuje sériovou komunikaci prostřednictvím integrovaného USB portu. Síťovou komunikaci samotná deska v základním provedení neumožňuje. Řešením je pak použití externího Wifi či ethernet modulu, kterých je na trhu řada. Široce používaný je například Wifi modul ESP8266. V rámci konceptu uvažujeme propojení jednotlivých komponent prostřednictvím bezdrátové sítě Wifi. Veškerá komunikace je delegována skrze

centrální prvek. Ten si uchovává vnitřní topologii sítě a zajišťuje zpracování JSON formátu včetně adresace zpráv v rámci celého systému.

Komunikační protokol

Pro účely interní systémové komunikace byl navržen komunikační protokol založený na technologii JSON - Javascript Object Notation. *JSON je na prostředky nenáročný, textový a platformě nezávislý formát pro výměnu dat, který je zároveň velice dobře čitelný jak člověkem, tak počítačem* [8]. JDK 8 obsahuje vlastní API pro zpracování formátu JSON, dostatečně je popsáno v [8]. Pro platformu Arduino existuje řada open-source knihoven, v rámci konceptu byla zvolena *Arduino JSON library*[9]. Autoři této knihovny ji nazývají *elegantní a účinnou JSON knihovnou pro embedded systémy.* Obslužné skripty napsané v jazyce Wiring s využitím této knihovny umožňují snadné kódování a dekódování dat. Senzory jiného typu, např. mobilní telefony, pak ve svých obslužných aplikacích používají příslušná API pro zpracování JSON formátu

Typy zpráv

Systémové zprávy jsou rozděleny do 3 základních skupin:

- Zprávy pro manipulaci s daty ze senzorového subsystému
- Zprávy pro obsluhu soustavy aktuátorů
- Zprávy pro diagnostiku HW komponent systému

[9] https://github.com/bblanchon/ArduinoJson

Zprávy pro manipulaci s daty ze senzorového subsystému

Tento druh zpráv je určen pro získávání dat ze senzorového subsystému. Dělíme je dle účelu na následující 2 typy:

- **SensorDataRequest**
  - Směr komunikace: Centrální prvek → senzor
  - Primární účel: Inicializace procesu sběru dat ze senzoru. Využití ilustruje následující příklad, který ukazuje zprávu s žádostí o zaslání dat ze senzoru teploty v místnosti na adresu 192.168.1.99.

```
{
  "message_type": "sensor_data_request",
  "target_sensor": "temperature_room",
  "source_ip_address": "192.168.1.99"
}
```

- **SensorDataRequestReply**
  - Směr komunikace: Senzor → centrální prvek
  - Primárním účel: Reakce na zprávu SensorDataRequest v podobě zaslání požadovaných dat. Využití je opět ilustrováno na příkladu. Položka „sensor_data“ reprezentuje teplotu ve stupních Celsia.

```
{
  "message_type": "sensor_data_request_reply",
  "target_sensor": "temperature_room",
  "sensor_data": 23
}
```

Zprávy pro obsluhu soustavy aktuátorů

Zprávy tohoto druhu jsou využívány pro ovládání soustavy aktuátorů. Jsou rozděleny na následující 4 typy:

- **ActuatorStateRequest**
  - Směr komunikace: Centrální prvek → aktuátor
  - Primárním účel: Inicializace procesu zjištění aktuálního stavu aktuátoru. Následující příklad ilustruje zprávu s žádostí o zaslání stavu aktuátoru pro ovládání oken na adresu 192.168.1.99.

```
{
  "message_type": "actuator_state_request",
  "target_actuator": "window_handler",
  "source_ip_address": "192.168.1.99"
}
```

- **ActuatorStateRequestReply**
  - Směr komunikace: Aktuátor → centrální prvek
  - Primárním účel: Reakce na zprávu ActuatorStateRequest v podobě zaslání aktuálního stavu daného aktuátoru. Na příkladu je demonstrována zpráva obsahující stav aktuátoru pro ovládání oken. Stavová proměnná nabývá hodnoty 1 - okno je otevřeno.

```
{
  "message_type": "actuator_state_request_reply",
  "target_actuator": "window_handler",
  "sensor_state": 1
}
```

- **ActuatorStateChangeRequest**
  - Směr komunikace: Centrální prvek → aktuátor
  - Primárním účel: Inicializace procesu modifikování stavu vybraného aktuátoru. Příklad reprezentuje zprávu s žádostí o modifikaci stavu aktuátoru pro ovládání oken. Stavová proměnná nabývá hodnoty 0 - okno má být zavřeno.

```
{
  "message_type": "actuator_state_change_request",
  "target_actuator": "window_handler",
  "desired_state": 0,
  "source_ip_address": "192.168.1.99"
}
```

- **ActuatorStateChangeRequestReply**
  - Směr komunikace: Aktuátor → centrální prvek
  - Primární účel: Reakce na zprávu ActuatorStateChangeRequest v podobě zaslání výsledku procesu modifikace stavu daného aktuátoru. Tento výsledek je reprezentován hodnotou 1, podařilo-li se stav modifikovat. V opačném případě pak výsledek nabývá hodnoty 0. Na příkladu je ukázána zpráva obsahující informaci o úspěšné modifikaci stavu aktuátoru pro ovládání oken.

```
{
  "message_type": "actuator_state_change_request_reply",
  "target_actuator": "window_handler",
  "state_change_successful": 1
}
```

3.2 Prediktivní ovládání aktuátorů s využitím strojového učení

Strojové učení je podoblastí umělé inteligence. Jedná se o vysoce multidisciplinární oblast. *Je založena na výsledcích z širokého spektra vědních disciplín. Spojuje umělou inteligence, teorii pravděpodobnosti, statistiku, teorii výpočetní složitosti, teorii informace, filosofii, psychologii, neurobiologii, aj.* [9]. V současnosti je strojové učení spjato především s analýzou dat a rozpoznáváním obrazu, předmětem výzkumu jsou však také například inteligentní vozidla, drony, apod.

Samotná idea strojového učení přišla spolu s otázkou, zda je počítač schopen samostatného učení. Výzkum v této oblasti začal rokem 1949, kdy americký vědec Arthur Samuel započal výzkum zaměřený na aplikaci umělé inteligence ve hře šachy. Tímto okamžikem byl odstartován intenzivní výzkum v této oblasti, pojmenované jako strojové učení.

3.2.1 Definice strojového učení

Počítačový program se učí ze zkušenosti E, vzhledem k množině úkolů T a výkonu P. Výkon při plnění úkolů z množiny T se zlepšuje v závislosti na zkušenostech E [9]. Uvedená definice bude ilustrována na následujícím příkladu, jehož předmětem bude klasická hra Dáma.
Uvažujme následující veličiny:

- E - Zkušenost získaná hraním velkého množství zápasů.
- T - Množina úkolů je zde pouze jednoprvková, obsahující elementární úkol - hraní hry Dáma.
- P - Výkon je reprezentován pravděpodobností, že program vyhraje následující zápas.

Vzhledem k definici pak můžeme formulovat následující tvrzení:
Čím více her program odehraje, tím více zkušeností získá a vzroste tedy jeho šance na výhru v dalším kole. Tento příklad jasně ilustruje definici uvedenou výše a zároveň reflektuje proces lidského učení.

3.2.2 Učení s učitelem

Dle [17] rozlišujeme 2 základní přístupy k samotnému procesu učení:

i. Učení s učitelem (*Supervised learning*)
   Tento přístup je založen na „učení se z dat“. Jinými slovy je algoritmům tohoto typu poskytnut tzv. *training set*, neboli datový soubor obsahující vlastnosti predikce a výslednou hodnotu, jež se k těmto vlastnostem váže. Cílem algoritmu je pak analýza datového souboru a vytvoření tzv. *hypotézy*, neboli funkce, pomocí níž jsou v budoucnu prováděny predikce.

ii. Učení bez učitele (*Unsupervised learning*)
   Učení bez učitele je „pravým opakem“ předchozího přístupu. Algoritmu je předložen datový soubor, ve kterém jsou následně hledány souvislosti, data jsou rozřazována do skupin. V rámci této práce bude dále diskutován první z přístupů, tedy učení s učitelem.

Množina trénovacích dat

Jak již bylo naznačeno, proces učení s učitelem je založen na tzv. „učení se z dat“. Učícímu se algoritmu je nutno dodat „množinu trénovacích dat“ - *training set*. Než bude zavedena formální definice, je třeba definovat klíčové pojmy.

Vektor vlastností predikce

Feature vector, neboli vektor vlastností predikce, ve svých složkách obsahuje hodnoty veličin, na jejichž základě jsou prováděny predikce.

Definice (3.1): Nechť R je n-rozměrný lineární prostor, kde n je počet vlastností predikce, nad tělesem reálných čísel. Pak vektor $\vec{x} \in R$, pro který platí $x_0 = 1,\ x_0 \in \vec{x}$, nazveme tzv. vektorem vlastností predikce. [10]

Jednotlivé složky vektoru v případě inteligentních prostředí reprezentují stavy senzorů. Vektor tohoto typu je ilustrován následujícím příkladem:

$$\vec{x} = \begin{bmatrix} 1 \\ t\_in \\ t\_out \\ hum \\ time \end{bmatrix}$$

Vlastnost predikce	Popis
t_in	teplota vzduchu v místnosti
t_out	venkovní teplota vzduchu
hum	vlhkost vzduchu v místnosti
time	aktuální čas

Výstupní vektor

Output vector, neboli výstupní vektor, ve svých složkách uchovává hodnoty veličin, které jsou předmětem predikce. Tyto hodnoty jsou využívány jako „správné“ výstupy predikce během procesu učení.

Definice (3.2): Označme písmenem D datový soubor o m záznamech. Nechť R je m-rozměrný lineární prostor nad tělesem reálných čísel. Pak vektor $\vec{y} \in R$ nazveme tzv. výstupním vektorem. [10]

V případě inteligentních prostředí jsou složky výstupního vektoru reprezentovány stavy jednotlivých aktuátorů. Ilustrujme nyní definici na příkladu:

Uvažujme vektor vlastností predikce $\vec{x}$ z předchozího příkladu. Definujme dále m-rozměrný výstupní vektor $\vec{y}$, jehož složky uchovávají stavy aktuátoru pro ovládání oken. V rámci příkladu budeme uvažovat 2 možné stavy:

- Okno je zavřeno - 0
- Okno je otevřeno - 1

Přiřadíme-li jednotlivé složky vektoru $\vec{y}$ k m různým vektorům vlastností predikce $\vec{x}$, dostaneme kompletní reprezentaci stavů daného inteligentního prostředí v kontextu aktuátoru pro ovládání oken. Na následujícím příkladu můžeme vidět

situaci, kdy teplota v místnosti dosahuje 22,3 st. Celsia, venkovní teplota 28,5 st. Celsia, vlhkost vzduchu v místnosti 50% v čase 14:00. Složka y_0 vektoru $\vec{y}$ pak indikuje, že okno je otevřeno.

$$\vec{x} = \begin{bmatrix} 1 \\ 22{,}3 \\ 28{,}5 \\ 50 \\ 1400 \end{bmatrix}, \vec{y} = [1]$$

Množina trénovacích dat - definice

Definice (3.3): Označme písmenem D datový soubor o m řádcích a k sloupcích. Sloupce 1 až $(k-1)$ reprezentují vlastnosti predikce, k-tý sloupec reprezentuje výstupní hodnoty. Nechť X je $(k-1)$ rozměrný lineární prostor transponovaných vektorů vlastností predikce - *feature space*. Nechť $\vec{y}$ je m-rozměrný výstupní vektor. Pak matici M o rozměrech m x k ve tvaru:

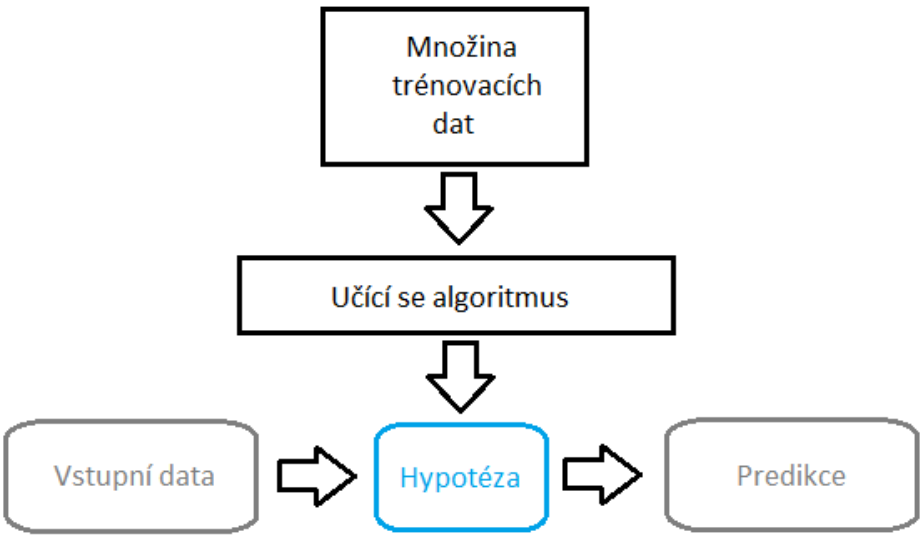
$$\begin{bmatrix} \vec{x}^{(1)} & y_1 \\ \vec{x}^{(2)} & y_2 \\ \vec{x}^{(m)} & y_m \end{bmatrix}; \ \vec{x}^{(i)} = [x_0, x_1, \dots \ x_{k-1}], i = 1, \dots, m; \ \vec{x}^{(i)} \in X$$

nazýváme množinou trénovacích dat v maticové reprezentaci. Uspořádanou dvojicí ve tvaru:

$$(\vec{x}^{(i)}, y_j^{(i)}), j = 1, \dots n$$

pak rozumíme i-tý trénovací záznam v množině trénovacích dat [10].

Průběh procesu učení s učitelem



Obrázek 9: Schéma procesu učení s učitelem
(zdroj: autor, námět: [10])

Průběh procesu učení lze zjednodušeně popsat následují sekvencí činností:

1. Vytvoření množiny trénovacích dat.
2. Předání množiny dat hlavnímu, učícímu se algoritmu, jehož výstupem je tzv. *hypotéza*, neboli funkce pomocí níž jsou prováděny predikce.

   Primárně rozlišujeme 2 způsoby predikce:

   i. Regrese = Predikce spojitých hodnot.

   ii. Klasifikace = Predikce diskrétních hodnot - rozdělení dat do tříd.

3.2.3 Strojové učení v kontextu inteligentních prostředí

Role senzorového subsystému

Senzorový subsystém je hlavním zdrojem dat. Jak již bylo řečeno, stavy jednotlivých senzorů tvoří složky vektorů vlastností predikce. Každý senzor představuje stavovou proměnnou příslušného prostředí.

V kapitole 2.3 byl diskutován konkrétní příklad inteligentního prostředí, jehož předmětem byla kancelář vysokoškolského pedagoga. V následující části využijeme soustavu senzorů, jež byla v tomto scénáři zavedena. Formulujme nyní obecnou formu vektoru vlastností predikce, dle definice 3.1, pro tento konkrétní scénář:

$$\vec{x} = \begin{bmatrix} 1 \\ RFID\_status \\ Pr\_sensor\_1 \\ Pr\_sensor\_2 \\ Pr\_sensors \\ user\_location \\ smoke\_sensor \\ temp\_in\_sensor \\ temp\_out\_sensor \\ hum\_sensor \end{bmatrix}$$

Tabulka 5: Specifikace stavových proměnných senzorů

Senzor	Stavová proměnná	Obor hodnot
Čtečka RFID čipů	RFID_status	ID uživatele : 0, 1, 2, … n
Tlakový senzor v židli	Pr_sensor_1	Nesepnuto / sepnuto: 0, 1
Tlakový senzor v židli	Pr_sensor_2	Nesepnuto / sepnuto: 0, 1
Tlakové senzory v podlaze	Pr_sensors	Nesepnuto / sepnuto: 0, 1
Poloha uživatele v budově	user_location	-
Detektor kouře	smoke_sensor	Kouř nedetekován / detekován: 0, 1
Teplota vzduchu v místnosti	temp_in_sensor	Stupně Celsia: float
Venkovní teplota vzduchu	temp_out_sensor	Stupně Celsia: float
Vlhkost vzduchu v místnosti	hum_sensor	Vlhkost v procentech: <0, 100>

Role soustavy aktuátorů

Soustava aktuátorů hraje v rámci adaptivního chování zásadní roli - stavy jednotlivých aktuátorů jsou hlavním předmětem predikce pro algoritmy strojového učení. Budeme-li schopni stavy aktuátorů efektivně predikovat, zajistíme tím adaptivní vlastnost inteligentního prostředí. Toho docílíme monitorováním uživatele a zkoumáním jeho chování. Hledáme tedy v daném prostředí *vzory chování jednotlivých uživatelů*. Tento přístup je diskutován v [18].

Nyní je již formulován vektor vlastností predikce pro konkrétní prostředí - inteligentní kancelář, formulujme analogicky pro stejný scénář vektory výstupů, dle definice 3.2:

$$\vec{y}_1 = [window1\_act]$$
$$\vec{y}_2 = [window2\_act]$$
$$\vec{y}_3 = [doors\_act]$$
$$\vec{y}_4 = [lights\_act]$$
$$\vec{y}_5 = [blinds\_act]$$

Tabulka 6: Specifikace stavových proměnných aktuátorů

Aktuátor	Stavová proměnná	Obor hodnot
Ovládání okna č. 1	window1_act	Zavřeno / otevřeno: 0, 1
Ovládání okna č. 2	window2_act	Zavřeno / otevřeno: 0, 1
Ovládání dveří	doors_act	Zavřeny / otevřeny: 0, 1
Ovládání světel	lights_act	Úrovně intenzity světla: 0, 1, 2, 3, 4, 5
Ovládání okenních rolet	blinds_act	Nezataženy / zataženy: 0, 1

Intenzita sběru dat

Data ze senzorů a aktuátorů jsou tedy uchovávána ve formě vektorů vlastností a vektoru výstupů v množině trénovacích dat. Je ovšem nutno definovat, v jakých časových intervalech tato data ukládat do databáze. Jedním z možných řešení je data shromažďovat periodicky v určitých časových intervalech, konkrétní délka tohoto intervalu je silně determinována prostředím - v prostředích s přísnými bezpečnostními pravidly bude třeba kontrolovat stavové proměnné častěji. Vzhledem k paralelizaci činností na centrálním prvku tento proces nebude mít na činnost řídicího systému zásadní vliv. Příchozí data jsou zároveň využita v rozhodovacím modulu obslužné aplikace, který je zhodnotí a navrhne možné změny stavů vybraných aktuátorů. Ilustrujme situaci opět na příkladu:

Inteligentní řídicí systém je integrován v kanceláři. Periodicky je monitorován stav na všech senzorech. Uživatel tohoto systému je zvyklý otevírat okno při 22 st. Celsia. S postupem času adaptivní vrstva tento vzor chování rozpoznala a naučila se ho rozeznávat. Je 12:30, pokojová teplota vzrostla na 22st. Celsia. Centrální prvek dostává data ze senzorového subsystému, podrobuje je analýze a zjišťuje, že je třeba upravit stav aktuátoru pro ovládání oken. Predikce je zhodnocena rozhodovacím modulem, přijata a příslušnému aktuátoru je zaslána zpráva s žádosti o změnu stavu. Obslužný skript zprávu přijímá, dekóduje, a pomocí obslužného zařízení otevírá okno. Na závěr zasílá centrálnímu prvku zprávu o úspěšném provedení daného úkolu.

3.2.4 Úvod do umělých neuronových sítí

Umělé neuronové sítě jsou poměrně známou technikou z oblasti strojového učení. *Učení pomocí metod umělých neuronových sítí poskytuje robustní přístup k aproximaci reálných, diskrétních a vektorových funkcí. Pro určité typy úloh, jako je například učení s cílem reprezentovat komplexní data ze senzorových systémů, jsou umělé neuronové sítě nejefektivnější technikou, jaká je nyní k dispozici* [9].

Koncept umělých neuronových sítí je inspirován biologickými principy procesu učení, fungování těchto sítí je velice blízké fungování lidského mozku.

Binární klasifikace dat

Binární klasifikace dat je přístup, kdy jsou data klasifikována do dvou, vzájemně disjunktních, tříd. Data jsou reprezentována standardní množinou trénovacích dat o m řádcích, dle definice 3.3. Dimenze vektoru vlastností predikce $\vec{x} \in R^n$ pak určuje dimenzi lineárního prostoru P, jenž reprezentuje daný klasifikační problém. Jednotlivé klasifikační třídy jsou reprezentovány složkami vektoru výstupů predikce $\vec{y} \in R^m$.

Cílem algoritmu pro binární klasifikaci je nalezení lineárního podprostoru k prostoru P, označme ho L. Dimenze L je rovna $n - 1$, kde n je dimenze prostoru P. Příkladem takového podprostoru může být přímka, která je jednorozměrným lineárním podprostorem libovolného dvourozměrného lineárního prostoru. Tento podprostor rozděluje prvky prostoru P do dvou klasifikačních tříd, hovoříme pak o něm jako o tzv. *decision boundary* [10], neboli hranici, která přesně vymezuje rozsah jednotlivých klasifikačních tříd. Binární klasifikace je realizovatelná s využitím jediného neuronu, tzv. Perceptronu.

Vícetřídní klasifikace dat

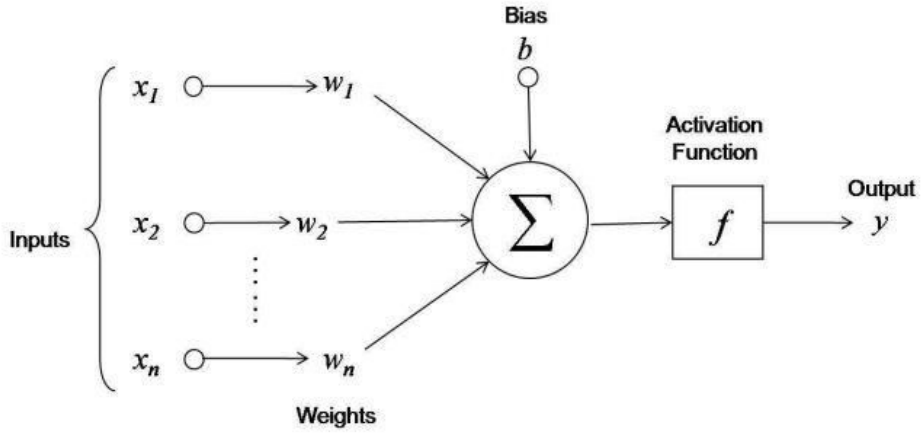
Binární klasifikace je ve většině případů nahrazena tzv. vícetřídní klasifikací, která umožňuje klasifikovat data do více než 2 klasifikačních tříd. Toho je dosaženo s využitím vícevrstvých neuronových sítí.

Architektura umělé neuronové sítě

Výzkum v oblasti umělých neuronových sítí byl částečně inspirován faktem, že biologické systémy, zajišťující proces učení, jsou tvořeny velice komplexními sítěmi vzájemně propojených neuronů [9]. Umělé neuronové sítě jsou analogií právě takových systémů, kdy každá síť je tvořena množinou vzájemně propojených umělých neuronů.

Model umělého neuronu

Základním stavebním kamenem umělé neuronové sítě je samotný neuron. Ten je tvořen n vstupy, které nabývají reálných hodnot, a jedním reálným výstupem. Každý z těchto vstupů je vážen reálnou konstantou, označme ji w. Jádrem umělého neuronu je tzv. aktivační funkce, jejímž úkolem je transformace n vážených vstupů na jediný výstup.



Obrázek 10: Model umělého neuronu
zdroj: [10]

Z ilustrace umělého neuronu lze jasně odvodit princip jeho fungování. Vstupní hodnoty proměnných $x_0 - x_n$ jsou lineárně zkombinovány tak, že platí: $z = \sum(w_i * x_i)$; $i = 1, \dots, n$. K proměnné z je přičtena hodnota tzv. *bias unit*, která je definována jako $b = 1$. Takto upravená lineární kombinace je následně aktivační funkcí zpracována do výsledného výstupu, platí tedy $y = f(z)$.

Aktivační funkce

Jak již bylo nastíněno, základním prvkem umělého neuronu je tzv. aktivační funkce. Klíčovým kritériem je pak volba konkrétní podoby této funkce. V umělých neuronových sítích se pro tento účel velice často používá speciální případ obecné logistické funkce - sigmoida [10].

Sigmoida je reálnou funkcí, jejíž definiční obor je množina reálných čísel. Oborem hodnot je interval <0, 1>. Limitní hodnoty oboru hodnot lze chápat jako klasifikační třídy, hodnoty mezi nimi pak jako pravděpodobnosti, vyjadřující pravděpodobnost klasifikace dat do třídy č. 1 [10]. Nechť $P(1)$ je pravděpodobnost klasifikace dat do třídy č. 1, pak $P(0) = 1 - P(1)$ je pravděpodobnost klasifikace dat do třídy č. 0.

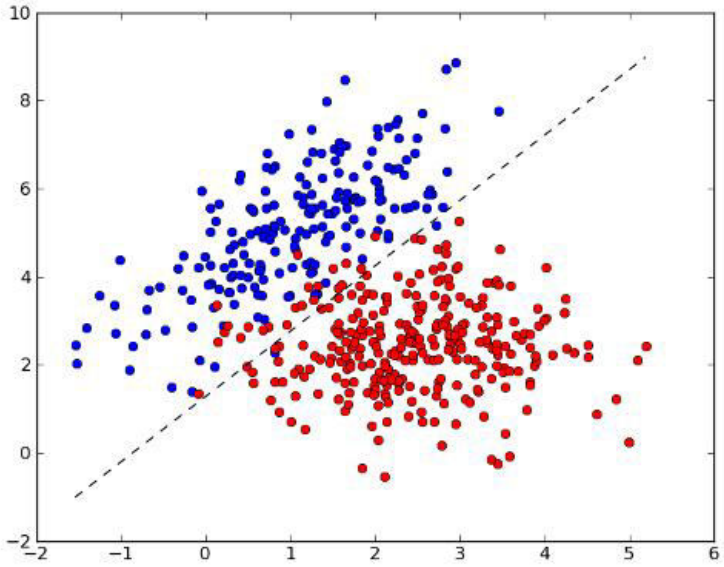
Formálně je sigmoida definována následujícím funkčním předpisem:

$$f(x): \frac{1}{1 + e^{-x}}, x \in R$$

Perceptron

Nejjednodušší aplikací umělého neuronu je tzv. Perceptron. Jedná se o přístup, kdy je ke klasifikaci dat použit pouze jediný umělý neuron.

Zásadní nevýhodou Perceptronu je skutečnost, že je schopen rozdělit lineární prostor P, jenž reprezentuje daný klasifikační problém, pouze na 2 podprostory. Provede tedy binární klasifikaci dat. Ilustrujme situaci vizualizací konkrétní situace.



Obrázek 11: Vizualizace výsledku činnosti perceptronu
zdroj: http://mlpy.sourceforge.net/docs/3.2/_images/perceptron1.png

Ilustrace ukazuje množinu dvourozměrných dat, které jsou klasifikovány do 2 klasifikačních tříd. Výsledkem činnosti perceptronu je lineární hranice pro vymezení klasifikačních tříd - v ilustraci zaznačena přerušovanou čarou.

Vícevrstvé neuronové sítě

Binární klasifikace pomocí Perceptronu je často nedostatečná. Vícevrstvá síť umožňuje vytvářet daleko komplexnější hypotézy.

V rámci této práce bude uvažována síť, která je tvořena třemi následujícími vrstvami:

1. Vstupní, tzv. *input layer*, vrstva. Jednotlivé prvky reprezentují složky vektoru $\vec{x}$.

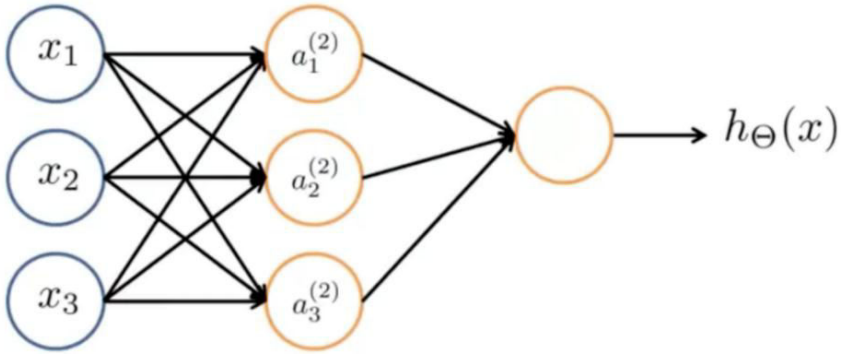
2. Skrytá, tzv. *hidden layer,* vrstva obsahuje k neuronů. Masters ve [12] formuloval následující vztah pro odvození počtu těchto neuronů v třívrstvých sítích:

$$k = \sqrt{n * m}$$

kde n je počet neuronů v první vrstvě a m pak počet neuronů vrstvy výstupní. Uvažujme opět prostor P, jenž reprezentuje daný klasifikační problém.

Každý z k neuronů v této vrstvě rozděluje prostor P na dva podprostory. Výsledkem je vytvoření k lineárních hranic pro vymezení klasifikačních tříd, které společně formují hlavní, nelineární, hranici.

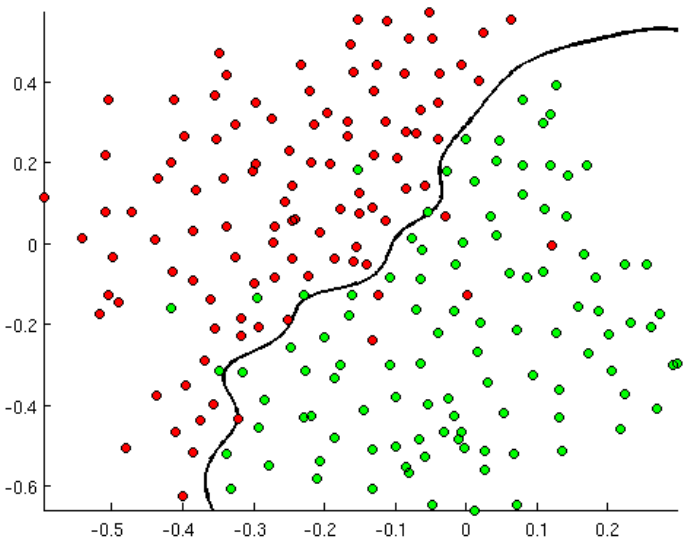
3. Výstupní, tzv. output layer, vrstva je reprezentována celkem l neurony. Počet těchto neuronů je závislý na typu klasifikace, Ng v [10] definuje pouze 1 výstupní neuron v případě binární klasifikace a k neuronů v případě klasifikace do k klasifikačních tříd. Výstupem poslední, třetí, vrstvy je již kompletní hypotéza.



Obrázek 12: Model třívrstvé umělé neuronové sítě (bez vyznačených *bias units*)
zdroj: [10]

Dle věty Universal approximation theorem jsou umělé neuronové sítě s jednou skrytou vrstvou, obsahující konečný počet neuronů, schopny aproximovat libovolnou spojitou funkci, definovanou na kompaktních podprostorech prostoru R^n [14]. Cybenko v [11] tuto větu formálně dokázal pro neuronové sítě, ve kterých jsou aktivačními funkcemi právě sigmoidy.

Následující ilustrace zobrazuje nelineární hranici pro vymezení klasifikačních tříd, která je výsledkem činnosti vícevrstvé sítě.



Obrázek 13: Vizualizace výsledku činnosti vícevrstvé neuronové sítě
zdroj: http://cs.stackexchange.com

Učení metodou zpětného šíření

Nejpoužívanější algoritmem pro učení umělých neuronových sítí je algoritmus zpětného šíření[10]. Ten je založen na stanovení tzv. chybové funkce, která vyjadřuje odchylku mezi skutečnou a požadovanou odezvou sítě. Cílem je pak tuto funkci nad danou trénovací množinou minimalizovat a najít vhodné hodnoty vah mezi jednotlivými neurony. Detailní popis průběhu algoritmu a jeho odvození je nad rámec této práce a nebude dále diskutováno. Obsáhlý popis algoritmu lze nalézt v [9], případně [10].

[10] Backpropagation algorithm

3.2.5 Umělé neuronové sítě v inteligentním řídicím systému

Adaptivní chování inteligentního řídicího systému zajišťuje soustava umělých neuronových sítí. Důvodem k dekompozici do více sítí je problematické zachycení možných stavů na jednotlivých aktuátorech v případě využití jedné umělé neuronové sítě. Pokud by adaptivní chování zajišťovala pouze jediná síť, bylo by nutné reprezentovat klasifikačními třídami jednotlivé stavy prostředí na globální úrovni, nikoliv z pohledu konkrétního aktuátoru.

Uveďme příklad, kdy inteligentní řídicí systém využívá jedinou univerzální umělou neuronovou síť. Uvažujme n senzorů, doplněných celkem k aktuátory, které z důvodu zjednodušení modelové situace nabývají pouze dvou stavů. Predikce sítě má následující vektorizovaný tvar:

$$\vec{p} = h_w(\vec{x}) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_k \end{bmatrix}, x \in R^n$$

Složky vektoru $\vec{p}$ reprezentují jednotlivé aktuátory, respektive jejich stavy. V rámci příkladu nabývají hodnot 0 nebo 1, které lze chápat následovně: 0 - aktuátor neaktivní, 1 - aktuátor aktivní. Síť tedy nepredikuje stav jednoho určitého aktuátoru, ale stavy daného pracovního prostředí v kontextu úplné soustavy aktuátorů. Zásadním problémem při použití tohoto přístupu je celkový počet klasifikačních tříd. Zachycujeme-li stavy prostředí s využitím dvoustavových aktuátorů, dosáhl by počet klasifikačních tříd čísla 2^k. Takové množství klasifikačních tříd je dáno tím, že každá třída jednoznačně reprezentuje unikátní stav celé soustavy aktuátorů. Zobecněním na aktuátory s l možnými stavy je pak počet klasifikačních tříd roven číslu l^k.

Uvažujme nyní opět scénář inteligentní kanceláře, představený v kapitole 2.3.

- Vektor vlastností predikce pro tento konkrétní příklad:

$$\vec{x} = \begin{bmatrix} 1 \\ RFID\_status \\ Pr\_sensor\_1 \\ Pr\_sensor\_2 \\ Pr\_sensors \\ user\_location \\ smoke\_sensor \\ temp\_in\_sensor \\ temp\_out\_sensor \\ hum\_sensor \end{bmatrix}$$

- Vektory výstupů:

$$\vec{y}_1 = [window1\_act]$$
$$\vec{y}_2 = [window2\_act]$$
$$\vec{y}_3 = [doors\_act]$$
$$\vec{y}_4 = [lights\_act]$$
$$\vec{y}_5 = [blinds\_act]$$

V prostředí se nachází celkem 5 aktuátorů, jejichž stav je předmětem predikce. Každému z těchto aktuátorů je přiřazena specializovaná umělá neuronová síť, celkem tedy adaptivní chování tohoto konkrétního inteligentního řídicího systému zajišťuje 5 separátních sítí. Predikce jsou prováděny na lokální úrovni, z pohledu konkrétního aktuátoru. V rámci konceptu jsou uvažovány standardní třívrstvé sítě.

Vstupem každé z těchto sítí je vektor vlastností predikce $\vec{x}$, ten může být dále rozdělen do tzv. „subvektorů“, s ohledem na primární zaměření konkrétní sítě. V praxi to znamená využití těch vlastností, které mají zásadní vliv na výslednou predikci. Vlastnosti predikce, které takový vliv na výsledek nemají, nejsou využity. Tímto optimalizačním krokem docílíme snížení dimenze vstupního vektoru a s tím souvisejícího snížení počtu vstupních neuronů dané sítě.

Výstupem každé sítě je m-rozměrný vektor, který reprezentuje výsledný stav aktuátoru. Dimenze tohoto vektoru je určena počtem stavů, kterých může aktuátor nabývat. V případě aktuátoru pro ovládání osvětlení dostáváme následující pětici vektorů:

$$\overrightarrow{p_1} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}, \overrightarrow{p_2} = \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}, \overrightarrow{p_3} = \begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}, \overrightarrow{p_4} = \begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}, \overrightarrow{p_5} = \begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}$$

Tyto vektory jsou vektorizovanou reprezentací klasifikačních tříd, které stavy aktuátoru zastupují. Ostatní aktuátory jsou dvoustavové, výstupem sítí je skalár, představující konkrétní klasifikační třídu. V tomto případě se jedná o standardní binární klasifikaci dat.

Všechny sítě jsou učeny algoritmem zpětného šíření, popsaným v minulé kapitole. Vzhledem k maticovému charakteru úlohy lze uvažovat o implementaci s využitím GPGPU přístupu, který je založen na paralelizaci výpočtů s využitím grafických karet. Všechny sítě by tak byly zpracovávány paralelně a došlo by k markantnímu zrychlení běhu řídicího systému.

Vliv dynamiky prostředí na kvalitu predikce

Dynamika pracovního prostředí hraje v procesu učení významnou roli. Aby byla adaptivní vrstva schopna provádět účinné predikce, je třeba trénovací datovou sadu udržovat aktuální a sítě pravidelně přeučovat. Je třeba definovat, v jakých časových intervalech budou jednotlivé sítě přeučovány. Délka těchto intervalů je determinována typem konkrétního prostředí. V případě inteligentní kanceláře, diskutované v kapitole 2.3, je jednou z možností přeučovat sítě v době, kdy kancelář není intenzivně využívána. Sítě je tedy možno přeučovat pravidelně o víkendech, či v jiných, uživatelem definovaných časových úsecích.

3.3 Rozhodovací model

V následující kapitole je diskutován návrh rozhodovacího modelu. Autoři [6] zdůvodňují důležitost role, jakou problematika rozhodování v rámci inteligentních prostředí hraje. Mnohdy je inteligentním prostředím nazýváno prostředí, které je pouze částečně automatizováno, a veškeré činnosti jsou explicitně iniciovány uživatelem.

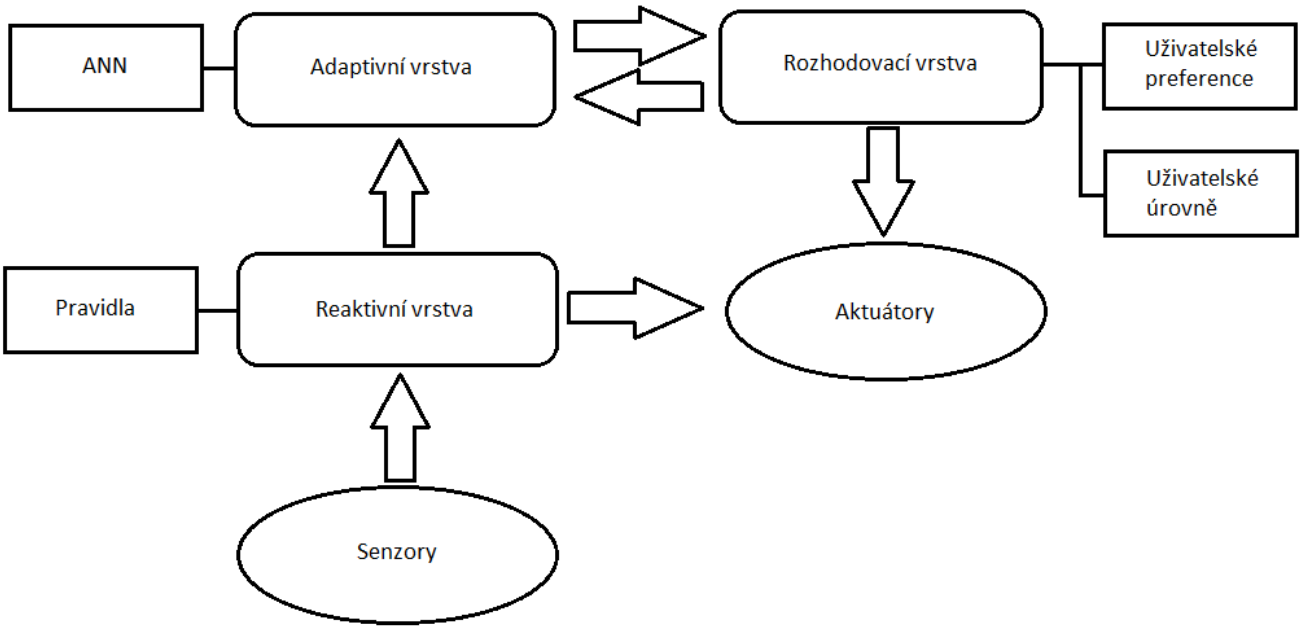
Model představený v této práci je založen na pravidlovém rozhodování, které je podpořeno rozhodováním pomocí umělých neuronových sítí, zajišťujících adaptivní chování inteligentního řídicího systému. Rozhodování v rámci modelu tedy můžeme primárně rozdělit na 2 hlavní části:

- Deterministické pravidlové rozhodování
- Stochastické rozhodování s využitím umělých neuronových sítí

Deterministické pravidlové rozhodování je založeno na pevně definované množině pravidel. S aktuátory je pak manipulováno s ohledem právě na tuto množinu základních pravidel. Rozhodování stochastického charakteru je podloženo predikcemi stavů na jednotlivých aktuátorech. Umělé neuronové sítě v kontextu inteligentních prostředí jsou diskutovány například v [13], kde jsou využity k zefektivnění spotřeby energií v domácnosti. Konjunkcí těchto dvou přístupů pak získáme vícevrstvý rozhodovací model, který dělíme do následujících tří vrstev:

1. Reaktivní vrstva
2. Adaptivní vrstva
3. Rozhodovací vrstva

Ucelené schéma rozhodovacího modelu je znázorněno v následující ilustraci. Šipky vyznačují směr hlavní komunikace. Z HW hlediska je komunikace obousměrná.



Obrázek 14: Schéma rozhodovacího modelu
zdroj: autor

3.3.1 Reaktivní vrstva

Základní vrstvou rozhodovacího modelu je tzv. Reaktivní vrstva. Proces rozhodování na této vrstvě je čistě deterministický, řízený pevně definovanými pravidly. Ty definují mezní hodnoty stavů senzorového subsystému a příslušné reakce v případě překročení těchto mezí. Příkladem takového pravidla může být monitorování kvality vzduchu - je-li senzorovým subsystémem detekován podíl kouře, který je vyšší než povolená mez, dojde k příslušné reakci ve formě evakuace místnosti či objektu. Úprava základních pravidel vyžaduje autorizovaný přístup prostřednictvím účtu hlavního administrátora řídicího systému.

Reaktivní vrstva je přímo spojena se senzorovým subsystémem a soustavou aktuátorů, je tak jedinou vrstvou, která je připojena k senzorovému subsystému, díky čemuž je schopna monitorovat tok dat z jednotlivých senzorů.
Veškerá data, která na vrstvu ze senzorů dorazí, jsou podrobena analýze s využitím základních pravidel.

Na základě výsledku analýzy je následně provedena jedna ze dvou základních typů reakcí:

1. Okamžitá reakce

   Okamžitá reakce nastává v případě, kdy je překročena mezní hodnota stavu některého ze senzorů. Dojde-li k takové situaci, reaguje systém provedením příslušné reakce přímou manipulací se senzorovým subsystémem. Data ze senzoru tedy nejsou delegována do vyšších vrstev.

2. Delegovaná reakce

   Delegovaná reakce je standardním typem reakce. Nastává v případech, kdy jsou data vyhodnocená analýzou označena jako bezriziková. V takovém případě jsou data delegována do vyšších vrstev rozhodovacího modelu a výsledná reakce je provedena až tzv. Rozhodovací vrstvou.

3.3.2 Adaptivní vrstva

Další vrstvou rozhodovacího modelu je tzv. Adaptivní vrstva. Ta zajišťuje obsluhu umělých neuronových sítí. V její kompetenci je provádění příslušných predikcí a udržování sítí v aktuálním stavu, čehož je docíleno periodickým přeučováním sítí nad pravidelně aktualizovanou množinou trénovacích dat. Adaptivní vrstva je úzce spojena s vrstvou Rozhodovací - jednotlivé predikce stavů aktuátorů nemají za následek přímou manipulaci s aktuátory, ale jsou podrobeny dodatečnému zhodnocení právě vrstvou Rozhodovací. Reakce stochastického charakteru jsou Rozhodovací vrstvou vyhodnoceny a případně označeny jako použitelné. V takovém případě vyústí v manipulaci s příslušným aktuátorem a nastavení jeho stavu na predikovanou hodnotu.
Činnost adaptivní vrstvy lze popsat následující sekvencí událostí:

1. Přijetí dat ze senzorového subsystému skrze Reaktivní vrstvu
2. Transformace dat na vektor vlastností predikce
3. Provedení predikcí na jednotlivých neuronových sítích
4. Předání výsledků predikcí Rozhodovací vrstvě

3.3.3 Rozhodovací vrstva

Poslední, třetí, vrstvou je tzv. Rozhodovací vrstva, jejíž činnost již byla částečně nastíněna. Obdobně jako je tomu u vrstvy Reaktivní, i zde je obsažena množina pravidel. Tato pravidla jsou však definována jednotlivými uživateli a reprezentují jejich preference v rámci inteligentního prostředí. Každý uživatel tak má možnost systému sdělit, jaké mezní hodnoty na vybraných senzorech jsou pro něj komfortní. S uživatelskými preferencemi jsou spojené uživatelské úrovně. Ty reprezentují váhu uživatelských preferencí - čím vyšší vahou preference disponuje, tím vyšší je na ni kladen při rozhodování důraz.

Činnost rozhodovací vrstvy opět ilustrujme sekvencí činností:

1. Přijetí predikce stavů aktuátorů z Adaptivní vrstvy
2. Zhodnocení predikce s ohledem na uživatelské preference, včetně uživatelských úrovní
3. Jestliže je výsledek kladný, pak:
   Nastavení stavů aktuátorů na predikované hodnoty.
4. Jinak:
   Ignorování predikce a následné podání zprávy adaptivní vrstvě

4 Implementace a testování umělých neuronových sítí

V rámci návrhu rozhodovacího modelu byla provedena implementace třívrstvé umělé neuronové sítě, predikující stav aktuátorů pro ovládání oken. Roli senzorového subsystému pak zajišťuje senzor venkovní teploty a dvojice senzorů, které detekují přítomnost osob v daném prostředí.

Největším problémem se pak stala samotná data. Vzhledem k faktu, že se jedná o poměrně úzce specializovaný projekt, bylo problémem najít vhodná, volně dostupná data, která by zahrnovala použitelnou množinu senzorů a aktuátorů a zároveň by obsahovala dostatečné množství záznamů pro trénování umělé neuronové sítě. Na základě četných konzultací byla situace vyřešena vygenerováním kompletní množiny dat, která v dostatečné míře reflektuje vzory lidského chování v pracovním prostředí.

4.1 Specifikace dat pro testování

Jak již bylo nastíněno, učení umělé neuronové sítě bylo provedeno nad vygenerovanou množinou dat. Celkem bylo vygenerováno 5000 trénovacích záznamů, které byly následně rozděleny do 2 skupin: 4000 záznamů bylo použito pro trénování, zbylých 1000 záznamů posloužilo pro určení kvality predikce. Uveďme nyní formálněji vektor vlastností predikce a vektor výstupní.

$$\vec{x} = \begin{bmatrix} 1 \\ t\_out \\ presence\_1 \\ presence\_2 \end{bmatrix}$$

Tabulka 7: Specifikace senzorového subsystému

Senzor	Stavová proměnná	Obor hodnot
Venkovní teplota vzduchu	t_out	Stupně Celsia: < 18, 29 >
Senzor pro detekci přítomnosti osoby č. 1	presence_1	Nepřítomen / přítomen: {0, 1}
Senzor pro detekci přítomnosti osoby č. 2	presence_2	Nepřítomen / přítomen: {0, 1}

$$\vec{y} = [window1\_act]$$

Tabulka 8: Specifikace soustavy aktuátorů

Aktuátor	Stavová proměnná	Obor hodnot
Ovládání oken	window1_act	Zavřeno / otevřeno: {0, 1}

Samotná data byla vygenerována na základě teplotního schématu, které zahrnuje 2 osoby s odlišnými teplotními nároky. Tabulka č. 9 pak zachycuje intervaly teplot, při kterých jsou dané osoby zvyklé otevírat okno.

V datech byly dále vytvořeny situace, které zohledňují úroveň uživatelských preferencí (viz tabulka č. 10) obou uživatelů. Čím vyšší je úroveň preferencí uživatele, tím více systém zohledňuje jeho přítomnost. Ilustrujme situaci na příkladu, založeném na hodnotách z tabulky č. 9 a č. 10: osoba č. 1 má vyšší úroveň uživatelských preferencí než osoba č. 2. Venkovní teplota vzduchu dosahuje hodnoty 24,5 st. Celsia. Ačkoliv je osoba č. 2 zvyklá při této hodnotě otevírat okno, vzhledem k přítomnosti osoby č. 1 je situace vyhodnocena právě vzhledem k osobě č. 1, tedy osobě s vyšší úrovní uživatelských preferencí. Výslednou predikcí je stav 0 - okno zavřeno.

Tabulka 9: Komfortní teploty pro otevření okna

Teplotní schéma												
Teplota (st. Celsia):	18	19	20	21	22	23	24	25	26	27	28	29
Osoba č. 1												
Osoba č. 2												

Tabulka 10: Úrovně uživatelských preferencí

	Úroveň
Osoba č. 1	2
Osoba č. 2	1

4.2 Testování na embedded zařízení Raspberry PI 2

Samotné testování bylo realizováno na zařízení Raspberry PI verze 2. Implementace umělé neuronové sítě a testovacího modulu byla provedena v programovacím jazyce Python s využitím knihovny pybrain [15], která disponuje robustním a srozumitelným aplikačním rozhraním.

Celkem bylo provedeno 10 nezávislých testů, lišících se v počtu iterací Backpropagation algoritmu. Výsledky jednotlivých testů jsou zaznamenány v následující tabulce.

Tabulka 11: Výsledky testování na embedded zařízení Raspberry PI 2

Pořadové číslo testu	Počet iterací BP alg.	Časové nároky	Přesnost predikce
1	1	0,000221 s	65,7 %
2	5	101,31 s	84,8 %
3	10	203,63 s	84,7 %
4	15	305,19 s	96,3 %
5	20	407,08 s	95 %
6	25	508,81 s	96,8 %
7	30	611,36 s	98,2 %
8	35	712,24 s	99,3 %
9	40	814,06 s	99,7 %
10	45	921,25 s	99,8 %

5 Závěr

Ačkoliv se nepodařilo prakticky implementovat kompletní a plně funkční řešení inteligentního pracoviště, vymezila se především cesta, po které by bylo možné v dalším výzkumu pokračovat. Práce poskytuje abstraktní náhled na řešení problematiky inteligentního pracovního prostředí a nastiňuje možná řešení jak hardwarové, tak softwarové vrstvy. Poukazuje na *open source* nástroje a jejich využití v rámci softwarové vrstvy řídicího systému. Zejména praktická implementace a testování umělé neuronové sítě s využitím knihovny *pybrain* v dostatečné míře poukazuje na obrovský potenciál metod strojového učení v kontextu ambientní inteligence. I když z technik strojového učení byly diskutovány pouze umělé neuronové sítě, nabízí se řada dalších přístupů (např. *Support vector machines*), které by byly v rámci inteligentních řídicích systémů použitelné.

Největším problémem, který v průběhu tvorby práce vyvstal, byl velice znatelný nedostatek volně dostupných dat. I přes existenci řady datových repositářů nebyla nalezena vhodná a zejména dostatečně rozsáhlá data. Problém byl vyřešen vygenerováním kompletních datových sad, včetně vytvoření mírného šumu v těchto datech. Tento problém jasně indikuje potřebu praktické implementace inteligentního prostředí, z kterého by bylo možné získat dostatečné množství reálných dat.

6 Seznam použitých zdrojů

[1] BOGDANOWICZ, M., et al. ISTAG: *Scenarios for Ambient Intelligence in 2010.* European Commission, Luxembourg, 2001

[2] COOK, Diane J., Juan C. AUGUSTO and Vikramaditya R. JAKKULA. 2009. Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 2009, 5.4:277–298

[3] PHILLIPS RESEARCH. Ambient intelligence: Changing lives for the better, 2007. www.research.phillips.com.

[4] WEISER, M. The Computer for the 21st Century. Scientific American 265 (1991), s. 94-104. http://www.scientificamerican.com/article/the-computer-for-the-21st-century/

[5] MIKULECKÝ, P. "AmI at Workplaces – What Are the Problems?" *AmI at Workplaces – What Are the Problems?* http://www.academia.edu/953690/AmI_at_Workplaces_What_Are_the_Problems

[6] RAMOS C., AUGUSTO J.C., SHAPIRO D., Ambient intelligence the next step for artificial intelligence, IEEE Intelligent Systems 23 (2008), 15–18.

[7] RASPBERRY PI FOUNDATION. Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/

[8] KOTAMRAJU J. Java API for JSON Processing: An Introduction to JSON. 2013. http://www.oracle.com/technetwork/articles/java/json-1973242.html

[9] MITCHELL, T. M. Machine Learning. New York: McGraw-Hill, 1997. ISBN: 0070428077 9780070428072.

[10] NG, A. Machine Learning. Online výukový kurz, dostupné online: https://www.coursera.org/learn/machine-learning

[11] CYBENKO, G. Approximations by superpositions of sigmoidal functions. Mathematics of Control, Signals, and Systems, 2 (4), 303-314, 1989.

[12] MASTERS, T. Practical neural network recipes in C++. Morgan Kaufmann, 1993.

[13] MOZER, M. C. The neural network house: An environmen that adapts to its inhabitants. AAAI Spring Symp. Intelligent Environments, 1998.

[14] BALÁZS C. C.. Approximation with Artificial Neural Networks. Faculty of Sciences. Eötvös Loránd University, Hungary.

[15] SCHAUL T., BAYER J., WIERSTRA D., SUN Y., FELDER M., SEHNKE F., RUCKSTIEß T., SCHMIDHUBER J. PyBrain. The Journal of Machine Learning Research, 11:743–746, 2010.

[16] MIKULECKÝ P.: Notes on Smart Workplaces. Workshop on Smart Office and Other Workplaces, International Conference on Intelligent Environments 2009, Barcelona.

[17] ABU-MOSTAFA Y. S., MAGDON-ISMAIL M., AND LIN H.-T.. Learning from Data: A Short Course. AMLBook, 2012.

[18] AZTIRIA A., IZAGUIRRE A., AUGUSTO J.C. Learning patterns in ambient intelligence environments: a survey. Artif Intell Rev, 34: 35, 2010.

[19] MOZER M.C., Lessons from an adaptive home, in: D.J. Cook, S.K. Das (Eds.), Smart Environments: Technology, Protocols, and Applications, Wiley, 2004, pp. 273–298.

7 Přílohy

1) Zdrojový kód testovacího modulu **anntest.py**

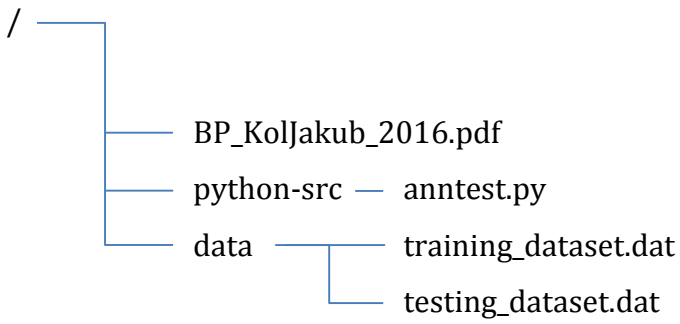
   Příloha obsahuje kompletní zdrojový kód testovacího modulu. Veškeré metody jsou opatřeny popisnými *docstringy*.

```
import time
import numpy as np
from pybrain.datasets import ClassificationDataSet
from pybrain.supervised.trainers.backprop import BackpropTrainer
from pybrain.tools.shortcuts import buildNetwork
from pybrain.structure import SigmoidLayer

class Ann:
    """
       Třída reprezentuje třívrstvou umělou neuronovou síť. Vzhledem k úzké
       specializaci implementace není vhodné tuto třídu používat pro obecné
       případy. Třída je součástí bakalářské práce 'Ambientní inteligence v
       pracovním prostředí' a slouží čistě k testování prováděném v rámci této
       práce.
       Autor: Jakub Kól
    """

    def __init__(self, pathToTrainingDataset, pathToTestingDataset):
        """
        Specifikace funkce:
              Konstruktor.
        Parametry:
              pathToTrainingDataset - relativní cesta k datovému souboru
              sloužícímu k trénování sítě.
              pathToTestingDataset - relativní cesta k datovému souboru
              sloužícímu k testování přesnosti predikce.
        """
        self.pathToTrainingDataset = pathToTrainingDataset
        self.pathToTestingDataset = pathToTestingDataset
        self.dataset = None # Training set
        self.tempMax = None # Max temperature
        self.tempMin = None # Min temperature
        self.neuralnet = buildNetwork(3, 4, 1, bias = True, outclass =
                          SigmoidLayer, hiddenclass = SigmoidLayer) # ANN

    def parse_training_dataset(self, pathToDataset = None):
        """
        Specifikace funkce:
              Funkce sloužící ke zpracování datového souboru pro trénování sítě.
        Parametry:
              pathToDataset - relativní cesta k datovému souboru sloužícímu k
                              trénování sítě. Výchozím datovým souborem pro
                              trénování je ten, jehož cesta byla předána v
                              konstruktoru.
        """
        if(pathToDataset is None):
            data = np.loadtxt(self.pathToTrainingDataset)
        else:
```

```
            data = np.loadtxt(pathToDataset)
        self.tempMin = min(data[:, 0])
        self.tempMax = max(data[:, 0])
        data_scaled = [(value - self.tempMin) / (self.tempMax - self.tempMin) for
                       value in data[:, 0]]
        data_scaled = np.column_stack((data_scaled, data[:,(1, 2)]))
        self.dataset = ClassificationDataSet(3, nb_classes = 2);
        for i in range(len(data)):
            self.dataset.appendLinked(data_scaled[i, 0:3], data[i, 3])

    def train(self, epochCount = 50, isVerbose = True):
        """
        Specifikace funkce:
              Funkce sloužící k trénování sítě s využitím algoritmu zpětného
              šíření.
        Parametry:
              epochCount - počet iterací trénovacího algoritmu. Zvýšení počtu
                           iterací ve většině případů znamená zlepšení kvality
                           predikce, vede však ke značnému zpomalení procesu
                           učení.
              isVerbose - je-li předána funkci hodnota True, dochází k
                          sekvenčnímu vypisování celkové chyby do konzole.
        """
        start = time.time() # Timer - start
        trainer = BackpropTrainer(self.neuralnet, self.dataset, verbose =
                                  isVerbose, learningrate = 0.03)
        trainer.trainEpochs(epochCount)
        end = time.time() # Timer - stop
        if(isVerbose is True):
            print("Time elapsed: " + str(end - start))

    def predict_output(self, sensorData):
        """
        Specifikace funkce:
              Funkce sloužící k provádění predikcí na základě dat předaných v
              parametru.
        Parametry:
              sensorData - vektor vlastností predikce ve formátu seznamu (list)
                           dle následující specifikace:
                         [temperature, presence sensor #1, presence sensor #2]
        Návratová hodnota:
              Desetinné číslo vyjadřující pravděpodobnost, s jakou se bude
              aktuátor nacházet ve stavu č. 1 - okno je otevřeno.
        """
        sensorData[0] = self.scale_input(sensorData[0])
        return self.neuralnet.activate(sensorData)

    def scale_input(self, x):
        """
        Specifikace funkce:
              Pomocná funkce sloužící k úpravě hodnot (scaling) z teplotního
              senzoru. Hodnoty jsou upraveny dle následujícího vztahu:
              x_scaled = (x - minTemperature) / (maxTemperature -
                         minTemperature).
        Parametry:
              x - vstupní hodnota z teplotního senzoru.
        Návratová hodnota:
              upravená hodnota 'x'.
        """
        return ((x - self.tempMin) / (self.tempMax - self.tempMin))
```

```
def measure_accuracy(self):
    """
    Specifikace funkce:
          Pomocná funkce sloužící k měření přesnosti predikce.
    Návratová hodnota:
          Desetinné číslo reprezentující přesnost predikce v %.
    """
    errorCount = 0
    data = np.loadtxt(self.pathToTestingDataset)
    for val in data:
        prediction = self.predict_output(val[0:3])
        if((prediction < 0.5) and (val[3] == 1)):
            errorCount += 1
        elif((prediction > 0.5) and (val[3] == 0)):
            errorCount += 1
    return (1 - (errorCount / 1000.0)) * 100

def test_network(self, epochCount):
    """
    Specifikace funkce:
          Pomocná funkce sloužící k automatizovanému testování sítě.
    """
    print('Starting automated ANN testing...')
    print('Training dataset: ' + self.pathToTrainingDataset)
    print('Testing dataset: ' + self.pathToTestingDataset)
    print('Starting learning process...')
    self.parse_training_dataset()
    self.train(epochCount)
    print('Learning process finished.')
    print('Measuring prediction accuracy...')
    print('Prediction accuracy: ' + str(self.measure_accuracy()))
```

2) Přiložené CD obsahující text práce ve formátu PDF, zdrojový kód testovacího modulu, trénovací a testovací sadu dat.



Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Kól Jakub	Zvole 189, Zvole	I1201493

TÉMA ČESKY:

Ambientní inteligence v pracovním prostředí

TÉMA ANGLICKY:

Ambient Intelligence in workplaces

VEDOUCÍ PRÁCE:

Ing. Karel Mls, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce:
Navrhnout a realizovat rozšiřitelný (modulový) systém AmI se zaměřením na podporu pracovních činností.

Osnova:
Úvod
Cíl práce, volba metodologie, způsob řešení
Vlastní text práce
- Rešerše stávajícího stavu problematiky
- Návrh konkrétního systému
- Testování, ověřování návrhu
- Shrnutí výsledků

Závěry a doporučení
Seznam použité literatury

SEZNAM DOPORUČENÉ LITERATURY:

BOGDANOWICZ, Marc, et al. Scenarios for ambient intelligence in 2010. Office for official publications of the European Communities, 2001.

COOK, Diane J.; AUGUSTO, Juan C.; JAKKULA, Vikramaditya R. Ambient intelligence: Technologies, applications, and opportunities. Pervasive and
Mobile Computing, 2009, 5.4: 277-298.

NAKASHIMA, Hideyuki; AGHAJAN, Hamid; AUGUSTO, Juan Carlos. Handbook of ambient intelligence and smart environments. Springer Science & Business Media, 2009.

DE PAOLA, Alessandra, et al. Sensor 9 k: A testbed for designing and experimenting with WSN-based ambient intelligence applications. Pervasive and Mobile Computing, 2012, 8.3: 448-466.

STAVROPOULOS, Thanos G., et al. aWESoME: A web service middleware for ambient intelligence. Expert Systems with Applications, 2013, 40.11: 4380-4392.

TAPIA, Dante I., et al. Integrating hardware agents into an enhanced multi-agent architecture for Ambient Intelligence systems. Information Sciences, 2013, 222: 47-65.

FORTINO, Giancarlo, et al. Middlewares for smart objects and smart environments: Overview and comparison. In: Internet of Things Based on Smart Objects. Springer.

Podpis studenta:  Datum: .....

Podpis vedoucího práce: ..... Datum: .....