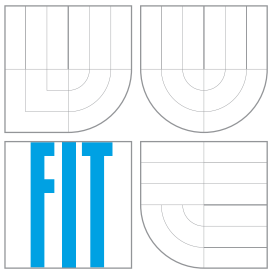
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DOCHÁZKOVÝ SYSTÉM S PŘÍSTUPOVÝM TERMINÁLEM

ATTENDANCE SYSTEM WITH ACCESS TERMINAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER PULIK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN MUSIL

BRNO 2016

Zadání bakalářské práce/17984/2015/xpulik00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Pulik Peter**
Obor: Informační technologie
Téma: **Docházkový systém s přístupovým terminálem**
Attendance System with Access Terminal
Kategorie: Vestavěné systémy

Pokyny:

1. Prozkoumejte současnou nabídku trhu v oblasti bezkontaktních přístupových terminálů a docházkových systémů a seznamte se s nejčastěji používanými technologiemi pro autentizaci příchozích osob.
2. Navrhněte realizaci docházkového terminálu založeného na některé ze snadno dostupných HW platforem (Například Raspberry Pi).
3. Realizujte docházkový terminál a vytvořte jednoduchou aplikaci pro správu docházky.
4. Otestujte výsledné řešení, zhodnoťte dosažené výsledky a navrhněte případné další pokračování práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Musil Martin, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá návrhom prístupového terminálu a dochádzkového systému. Popisuje návrh prístupového terminálu po hardwarovej stránke, implementáciu jeho riadiaceho programu, serverového programu slúžiaceho na výpočet dochádzky a komunikáciu medzi nimi. Celý systém je navrhnutý tak, aby ho bolo možné jednoducho rozširovať ako do počtu terminálov, tak novými možnosťami.

Abstract

This paper deals with the design of access terminal and attendance system. It describes access terminal hardware design, implementation of its control program and serverside attendance system program and the communication between these two programs. The whole system is designed to be easily scalable both in count of the access terminals used as well as in adding of new features.

Kľúčové slová

Prístupový terminál, dochádzkový systém, RFID, snímač odtlačkov prstov, Raspberry Pi, QT, Node.js

Keywords

Access terminal, attendance system, RFID, fingerprint scanner, Raspberry Pi, QT, Node.js

Citácia

PULIK, Peter. *Docházkový systém s přístupovým terminálem*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Musil Martin.

Docházkový systém s přístupovým terminálem

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána inžiniera Martina Musila.

.....
Peter Pulik
18. mája 2016

Poďakovanie

Rád by som poďakoval svojmu vedúcemu bakalárskej práce pánovi inžinierovi Martinovi Musilovi za jeho pomoc a rady pri návrhu tohoto prístupového terminálu. Taktiež by som rád poďakoval Ing. Petrovi Musilovi za to, že sa často zapájal do našich konzultácií a prispieval tak vlastnými radami a skúsenosťami.

Obsah

Kapitola 1

Úvod

Evidencia dochádzky je jednou z kľúčových záležitostí v každej firme. Je nie len nevyhnutná pre správny výpočet odpracovaných hodín, a teda aj mzdy zamestnancov, no je aj zo zákona povinná. Kontrola prístupu je taktiež nevyhnutná. Nie každá firma používa rôzne prístupové oprávnenia pre rôzne skupiny zamestnancov, no každá firma potrebuje oddeliť zákaznícku a pracovnú zónu a uistiť sa, že zákazníci sa nedostanú napríklad k výrobným linkám.

V dnešnej dobe existuje veľké množstvo rôznych dochádzkových systémov. Siahajú od tých najjednoduchších (pero a papier) až po tie najväčšie (prístupové turnikety) a najkomplexnejšie (skener očnej dúhovky, sken odtlačku dlane). Pri takomto rozsahu zariadení nie je prekvapivé, že existuje veľké množstvo rôznych senzorov a postupov určených na identifikáciu osôb. Niektoré senzory dokonca môžu fungovať na rôznych fyzikálnych princípoch, takže je možné vybrať si z viacerých variant, ktoré sa často odlišujú svojou veľkosťou, presnosťou identifikácie a cenou.

Práca je rozdelená na štyri kapitoly. Kapitola 2 Prístupové terminály je zameraná na analýzu trhu dostupných riešení, hardwarové periférie ktoré môžu používať a funkcionalitu ktorú ponúkajú. Na konci kapitoly je podkapitola 2.5 ktorá sa zameriava na rôzne spôsoby ako môže prístupový terminál komunikovať so serverom na ktorom beží dochádzkový systém.

Ďalšou kapitolou je 3 Návrh terminálu, kde sú postupne rozobrané možnosti implementácie jednotlivých podsystémov prístupového terminálu a dochádzkového systému. Súčasťou tejto kapitoly je aj podkapitola opisujúca použité riešenie komunikácie medzi terminálom a serverom a komunikačný protokol, ktorý používajú.

Predposlednou kapitolou je 4 Zostavenie a implementácia. V tejto kapitole je popísaná implementácia a fungovanie výsledného systému. Po tejto kapitole nasleduje záver, kde som popísal dosiahnuté ciele, možné vylepšenia a smer pokračovania práce.

Kapitola 2

Prístupové terminály

Základným princípom fungovania prístupových terminálov je overiť identitu zamestnanca a na základe toho vyhodnotiť, či by sa dvere mali otvoriť, alebo ostať zatvorené. Následne musí informáciu o (ne-)oprávnenom pokuse o vstup odoslať na server. Na to, aby bol prístupový terminál schopný vyhodnotiť prístupové práva, potrebuje mať prístup k databáze zamestnancov. Tá bude musieť byť uložená niekde v pamäti riadiacej jednotky. Z dôvodu zvýšenia spoľahlivosti systému je dobré, aby sa na prístupovom terminály lokálne ukladali taktiež záznamy o udalostiach na danom prístupovom terminály (napr. vstup zamestnanca, nedostupnosť servera).

2.1 Existujúce riešenia

Na súčasnom trhu existuje veľké množstvo riešení dochádzkových systémov a prístupových terminálov. Niektoré systémy fungujú ako aplikácia na klasickom počítači, kým iné ponúkajú špecializované prístupové systémy. Jednotlivé prístupové terminály sa líšia hlavne cenou, senzormi používanými pre identifikáciu zamestnanca a rozhraním pre komunikáciu s užívateľom.

Rozhranie pre komunikáciu s užívateľom

Prístupový terminál potrebuje rozhranie pre komunikáciu s užívateľom. Na trhu existujú terminály s rôznymi riešeniami tejto komunikácie. Pravdepodobne najjednoduchšie rozhranie používa prístupový terminál SCR100, ktorý používa len jednu RGB LED diódu [4]. Prepínaním farieb potom komunikuje s užívateľom:

- modrá farba = pohotovostný režim
- zelená farba = prístup povolený
- červená farba = prístup zamietnutý

Takéto rozhranie však umožňuje iba jednosmernú komunikáciu smerom k užívateľovi.

Obojsmernú komunikáciu umožňuje napríklad model DT1000, ktorý má monochromatický LCD display a klávesnicu [6]. Pomocou klávesnice je možné navoliť si napríklad dôvod odchodu z firmy (napr. doktor, obed, dovolenka). Tieto informácie sú dôležité pri výpočte odpracovaných hodín a ak by neboli zadané, museli by byť spätne ručne doplnené pred výpočtom. Informácie pre užívateľa sa následne zobrazujú na LCD display. Existujú aj

terminály ktoré majú farebný LCD display, no tie sa líšia len v spôsobe a možnostiach zobrazenia informácií.

Tretiu kategóriu tvoria terminály obsahujúce farebný dotykový display. Keďže je display dotykový, nepotrebujú mať klávesnicu. Akékoľvek vstupy sú totiž riešené formou ikon na obrazovke. Takéto riešenie má výhodu v podobe možnosti dynamicky meniť zobrazené ikony podľa potreby. Medzi zástupcov tejto kategórie patrí napríklad model S990 [1].

Obr. 2.1: Moderný dotykový dochádzkový terminál S990 s čítačkou RFID.[1]

Možnosti pripojenia

Pokiaľ ma dochádzkový systém spracovávať dochádzku zamestnancov, je potrebné preniesť do neho údaje o prístupoch zamestnancov z prístupového terminálu. Prístupový terminál teda musí disponovať rozhraním, cez ktoré sa k nemu je možné pripojiť. Dostupné riešenia ponúkajú širokú škálu rôznych rozhraní. Medzi tie najbežnejšie patria RS232, RS485, USB, pamäťová karta a Ethernet. Na obrázku 4.6 je možné vidieť rôzne komunikačné rozhrania moderného prístupového terminálu ATT-990. Ethernet a Wi-Fi sa používajú pre priebežný prenos dát medzi serverom a terminálom. RS232, RS485 a USB porty zväčša slúžia pre hromadné stiahnutie databázy záznamov do obslužnej aplikácie na PC.

2.2 Hardwarové platformy

Všetky procesy v prístupovom terminály musia byť niečím riadené. Dve hlavné kategórie, ktoré pre takúto aplikáciu prichádzajú do úvahy sú mikrokontrolér, alebo linuxové riešenie typu Single-Board Computer (SBC). Každé z týchto riešení má svoje výhody aj nevýhody.

Mikrokontrolér

Pre riešenie tejto úlohy by sa dal vhodne využiť mikrokontrolér. Hlavnou výhodou mikrokontroléru oproti linuxovému riešeniu je odbúranie medzivrstvy medzi programom a hardwarom terminálu v podobe operačného systému. Výsledný systém potom obsahuje menej

[1] Obrázok prevzatý z: http://www.alarmtel.net/DOCHADZKOVY-TERMINAL-S990-d142.htm
[2] Obrázok prevzatý z: http://dochadzkovy.system-is.com/dotykovy-dochadzkovy-terminal-att-990

Obr. 2.2: Ukážka rozhraní moderného prístupového terminálu.[2]

softwaru ktorý by mohol spôsobiť chybu, čo sa prejaví zvýšenou stabilitou systému a zrýchlením spustenia obslužného programu terminálu na hodnotu, keď je možné považovať ho za okamžité.

Nevýhody takéhoto riešenia však prevládajú nad jeho výhodami. Medzi jeho hlavné nevýhody je možné zaradiť napríklad sieťovú komunikáciu. Aj keď implementovať samotnú sieťovú komunikáciu v dostačujúcej forme pre potreby dochádzkového terminálu nie je veľmi zložité, bolo by potrebné implementovať prvky sieťovej bezpečnosti, ktoré sú na linuxových platformách už implementované. Najväčšou slabinou mikrokontrolérov je však vykresľovanie grafických elementov. Pre vytvorenie moderného užívateľského rozhrania by bolo potrebné implementovať pokročilé funkcie vykresľovania, ktoré by svojim rozsahom ďaleko presahovali rozsah tejto práce.

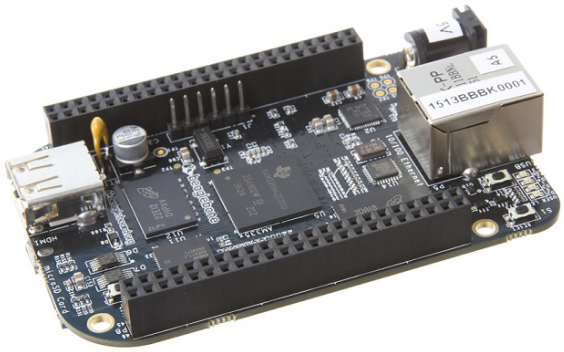
Linuxové riešenie Single-Board Computer

Linuxové riešenia so sebou prinášajú vlastnú zbierku nedostatkov. Predlžený štartovací čas, znížená stabilita a zvýšená zraniteľnosť systému pri útokoch sú len niektoré z nich. Vyriešená bezpečnosť sieťovej komunikácie, vykresľovanie aj tých najzložitejších grafických elementov, podpora pre súborové a databázové systémy a štandardizované systémové rozhranie umožňujúce jednoduchý prenosu kódu na iné linuxové zariadenie v budúcnosti však robia túto platformu ideálnou pre využitie v dochádzkovom termináli.

Na dnešnom trhu existuje veľké množstvo riešení SBC, ktoré sa líšia svojim výkonom, architektúrou rozhraniami a ďalšími vlastnosťami. Po preštudovaní dostupných možností som výber linuxovej platformy obmedzil na štyri zariadenia. Sú to Raspberry Pi, Beaglebone Black, UDOO Dual a Intel Edison. Existuje veľké množstvo ďalších SBC riešení, no vo väčšine prípadov sú veľmi podobné tým mnou vybraným a líšia sa hlavne množstvom pamäte RAM, použitým procesorom, prídavnými funkciami, ako napríklad infračervený prijímač a podobne.

Beaglebone Black

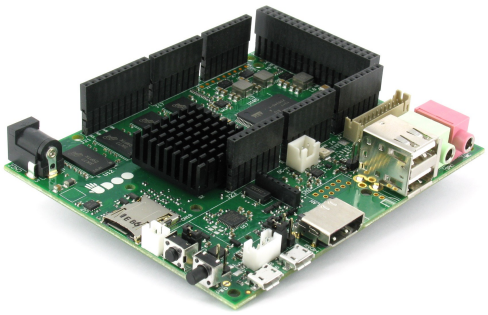
Beaglebone Black [3] patrí medzi najrozšírenejšie riešenia SBC. Má dostatočne veľkú užívateľskú základňu, ponúka vysoký výkon, dostatok portov GPIO, vstavanú pamäť eMMC a modernú architektúru ARMv7. Táto platforma spĺňa všetky zadané požiadavky. Hlavným nedostatkom tejto platformy však sú slabšie možnosti pripojenia displeja. Beaglebone Black z video portov totiž ponúka len HDMI. Taktiež však existuje možnosť pripojiť display cez GPIO porty. Takéto riešenie väčšinou nie je ideálne, pretože procesor sa musí starať o vykresľovanie dát na display, čo je veľmi náročné. Beaglebone Black však obsahuje dve PRU (Programmable Real-time Unit) jednotky, ktoré by boli schopné vykresľovať dáta na takýto display bez záťaže hlavného procesora.



Obr. 2.3: Beaglebone Black.[3]

UDOO Dual

Ďalšou zaujímavou možnosťou je UDOO Dual [18], ktoré sa od iných riešení typu SBC odlišuje hlavne prítomnosťou mikrokontroléru priamo na doske. Takéto riešenie teda ponúka možnosť využiť to najlepšie z oboch svetov, ktoré sa môžu navzájom dopĺňať. UDOO Dual spĺňa všetky požiadavky.



Obr. 2.4: UDOO Dual.[4]

[3] Obrázok prevzatý z: http://elinux.org/Beagleboard:BeagleBoneBlack

Intel Edison

Intel Edison [10] je na rozdiel od predošlých možností založený na architektúre x86. Vďaka tomu je kompatibilný s väčšinou softwaru dostupného na trhu. Intel Edison má v sebe zabudovaný mikrokontrolér, takže má rovnaké výhody ako UDOO Dual. Jeho miniatúrne rozmery, nízka spotreba a zabudované technológie Wi-Fi a Bluetooth z neho robia ideálnu voľbu pre vstavané systémy. Problémom platformy Intel Edison je absencia grafického jadra, čo znamená, že akékoľvek vykresľovanie by muselo prebiehať v procesore. Prítomnosť mikrokontroléru a nenáročnosť aplikácie by však takéto vykresľovanie umožnili. Jeho najväčším nedostatkom je, že je to nový produkt a teda má malú užívateľskú základňu a slabšiu podporu, než ostatné riešenia.

Obr. 2.5: Intel Edison - počítač o rozmeroch SD karty.[5]

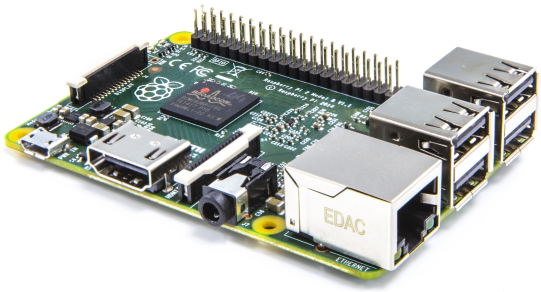
Raspberry Pi

Pravdepodobne najrozšírenejšie SBC riešenie na trhu, Raspberry Pi [14], má v porovnaní so zvyšnými tromi zariadeniami najväčšiu užívateľskú základňu s najväčšou podporou. Staršie modely majú nevýhodu v podobe staršej architektúry procesora ARMv6. Tá totiž už v dnešnej dobe nie je podporovaná modernými operačnými systémami a niektorým softwarom. Tento problém však do značnej miery rieši už spomínaná široká užívateľská základňa, pretože väčšina nekompatibilného softwaru bola upravená tak, aby bola schopná fungovať na Raspberry Pi. Riešenie tohto problému prináša aj nový model Raspberry Pi 2, ktorý je založený na modernejšej architektúre ARMv7 s 1GB RAM. Aj napriek tomu si však zachováva spätnú kompatibilitu so všetkým softwarom napísaným pre staršie verzie Raspberry Pi. Raspberry Pi 3 k vylepšeniam verzie 2 pridáva ešte zabudovaný Wi-Fi a Bluetooth komunikačný modul. Procesor pre Raspberry Pi 3 bol taktiež vymenený za novú 64-bitovú verziu.

2.3 Hardwarové periférie

Prístupový terminál môže pre identifikáciu osôb využívať rôzne senzory. Niektoré prístupové terminály používajú aj rôzne iné hardwarové periférie pre rozšírenie, alebo zlepšenie svojich funkcií, alebo pre zvýšenie spoľahlivosti systému ako takého.

[4]Obrázok prevzatý z: http://shop.udoo.org/eu/home/udoo-dual-basic.html
[5]Obrázok prevzatý z: http://www.intel.com/content/www/us/en/do-it-yourself/edison.html
[6]Obrázok prevzatý z: https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/



Obr. 2.6: Raspberry Pi 2.[6]

Používané senzory

Pre overenie identity môžu byť použité rôzne technológie ako napríklad jednoduché overenie PIN kódom, načítaním identifikačnej karty, alebo pomocou biometrického údaju. Najbežnejšou voľbou je použitie identifikačnej karty alebo čipu, v dnešnej dobe primárne na princípe RFID. Existujú snímače rôznych biometrických údajov, ako sú napríklad odtlačky prstov, sken očnej dúhovky, alebo rozpoznávanie tváre. Najdostupnejší a najjednoduchší na implementáciu je snímač odtlačkov prstov. Keďže sa však jedna o biometrický údaj a nie každý zamestnanec môže súhlasiť s jeho použitím, využitie odtlačku prsta pre identifikáciu je často voliteľné.

RFID

Technológia RFID umožňuje prenášať dáta, ako napríklad identifikačné značky, bezdrôtovo na krátke vzdialenosti. Výhodou tejto technológie je, že čipy, z ktorých sú dáta čítané, sú napájané energiou vyžarovanou z čítacieho zariadenia. Identifikačné karty teda nemusia obsahovať žiadne batérie, ktoré by bolo potrebné pravidelne vymieňať. Tento prenos môže fungovať na rôznych frekvenciách od nízkych až po mikrovlnné. Pre každú frekvenciu existujú špeciálne určené čipy a tieto čipy nie sú navzájom kompatibilné. Jednotlivé čítačky čipov sa odlišujú dosahom, ale aj používaným protokolom a cena čipov pre konkrétnu čítačku môže byť od niekoľko centov až po desiatky eur.

Senzor odtlačkov prstov

Úlohou snímača odtlačkov prstov je vytvoriť obraz priloženého prsta. Pri takom obraze nie je potrebné, aby bol farebný. Dôležité je len aby na ňom boli zachytené papilárne línie prsta. Tie sa následne spracujú algoritmom, ktorý nájde špecifické vzory papilárnych línií a informáciu o nich zakóduje do binárnej podoby. Nie je teda potrebné ukladať celý obraz odtlačku prsta. To taktiež prispieva k ochrane osobných údajov, pretože algoritmus, ktorý z línii vytvorí binárnu informáciu je jednosmerný. Nie je teda možné zo zakódovanej informácie spätne získať obraz odtlačku prsta [7].

Existujú rôzne metódy snímania odtlačkov prstov. Dostatočne spoľahlivé a jednoduché sú v súčasnosti však iba dva typy snímačov [5].

Optické snímače odtlačkov prstov fungujú na princípe vytvárania fotografií prsta. Pre vytvorenie tejto fotografie sa používajú klasické CCD alebo CMOS snímače používané aj v digitálnych fotoaparátoch a videokamerách. Tento typ snímača je možné jednoducho

oklamať priložením fotky alebo inej vernej napodobeniny prsta. Preto sa do nich občas pridávajú ďalšie kontrolné mechanizmy, ako napríklad senzor teploty, alebo detekcia pulzu pre odhalenie umelých napodobenín.

Kapacitné snímače odtlačkov prstov sú druhou často používanou metódou snímania odtlačkov prstov. Tento senzor meria kapacitu na povrchu senzora mriežkou kapacitných snímačov. Po priložení prsta na snímač sa namerané hodnoty zmenia. Keďže povrch papilárnych línii je bližšie k snímačom než povrch prsta medzi nimi, senzory pod papilárnymi líniami namerajú inú hodnotu kapacity než senzory medzi líniami. Z nameraných hodnôt je následne možné zostrojiť obraz priloženého prsta. Keďže kapacitné snímače vyžadujú fyzickú prítomnosť prsta, je omnoho ťažšie oklamať ich. Kapacitný snímač je však chúlostivejší na zmenu teploty, vlhkosti a podobne [5].

Watchdog

Watchdog je zariadenie slúžiace pre kontrolu správneho pracovania iného systému. Princíp jeho fungovania je jednoduchý. Kontroluje, či kontrolovaný systém pracuje, alebo nie. Ak systém pracuje, všetko je v poriadku. Ak však systém nepracuje - napríklad ak program zablúdi skokom na nesprávnu adresu, alebo uviazne v nekonečnej slučke, reštartuje ho. Na to, aby watchdog vedel vyhodnotiť, či je systém funkčný, potrebuje aby mu systém pravidelne posielal signál o tom, že je funkčný. Ak watchdog za určitý čas nedostane tento signál, predpokladá že systém prestal pracovať. Dĺžka času po ktorý bude watchdog čakať na signál pred tým, než systém reštartuje, je na väčšine watchdog zariadení nastaviteľná. Je úlohou programátora zabezpečiť odosielanie signálu o funkčnosti systému z programu dostatočne často.

Vstavaný watchdog Procesor použitý v Raspberry Pi obsahuje watchdog periférium. Je možné použiť ju pre kontrolu správneho fungovania programu. Má nastaviteľnú dobu medzi dvoma vyžadovanými pulzmi. Výhodou vstavaného watchdogu je že sa reštartuje priamo z programu špeciálnou inštrukciou. Nevýhodou však je, že dokáže reštartovať len procesor samotný. Zvyšok systému však pobeží ďalej.

Externý watchdog Výhodou externého watchdog zariadenia je to, že dokáže vypnúť a opäť zapnúť celý systém. Jeho miernou nevýhodou je však to, že je potrebné generovať výstupný signál na nejakom pine riadiaceho systému. Vzhľadom na vysoký počet pinov riadiacich systémov to však väčšinou nie je problém.

Display

Prístupové terminály často disponujú veľkými LCD obrazovkami. Takúto obrazovku je však potrebné nejakým spôsobom pripojiť k riadiacej jednotke. Existuje niekoľko spôsobov, ako ich navzájom prepojiť.

GPIO Piny GPIO[7] sú jedným zo spôsobov, ako je možné pripojiť LCD display k riadiacej jednotke. Takéto pripojenie sa používa hlavne s mikrokontrolérmi, ktoré na tento účel nemajú vlastné rozhranie. Pri linuxových SBC riadiacich jednotkách je však toto riešenie dosť nepraktické, pretože procesor musí riadiť prenos obrazu zo svojho obrazového bufera

[7] General Purpose Input/Output

cez GPIO piny. Pri pokusnom použití takéhoto typu pripojenia s Raspberry Pi 1 model B+ bolo využitie procesora v pokojovom stave 90-100%.

HDMI Raspberry Pi, rovnako ako Beaglebone Black, poskytujú pre video výstup rozhranie HDMI[8]. Na trhu existujú obrazovky pre vstavané aplikácie, ktoré obsahujú HDMI konektor. Nevýhodou týchto obrazoviek je však to, že potrebujú samostatnú riadiacu dosku, ktorá prijíma HDMI signál a riadi display. Tieto riadiace dosky sú však často dosť veľké, pretože obsahujú aj iné konektory aby boli čo možno najuniverzálnejšie. Takáto riadiaca doska spolu s HDMI prepojovacím káblom by prístupovému terminálu pridali objem.

DSI Raspberry Pi ponúka ešte možnosť pripojiť k nemu display cez DSI[9]. Toto rozhranie umožňuje pripojiť display pomocou plochého kábla. Medzi Raspberry Pi a display je ale taktiež nutné použiť prevodník. Ten je však menší, než bežné HDMI prevodníky a súčasne slúži ako radič pre dotykovú vrstvu obrazovky.

2.4 Podporné programy

Každý prístupový terminál potrebuje podporný počítačový program, ktorý umožňuje stiahnuť dochádzku z prístupového terminálu do počítača. Niektoré podporné programy nedokážu nič viac, než len to. Iné však dokážu zároveň plniť úlohu dochádzkového systému.

Dochádzkové systémy všetky plnia rovnakú základnú funkciu - výpočet odpracovaných hodín. Keďže ide o automatizovaný systém zberu dát, dochádzkový systém musí umožňovať dodatočnú úpravu uložených údajov. Tieto systémy taktiež umožňujú spravovať informácie o zamestnancoch a ich prístupové oprávnenia. Líšia sa však ďalšími nadštandardnými možnosťami, ktoré ponúkajú.

Niektoré dochádzkové systémy (napr. FLOWii [8]) umožňujú naplánovať udalosť do budúcnosti. Je napríklad možné zadať, že o dva dni pôjde zamestnanec na týždennú služobnú cestu. Systém si túto informáciu uloží a na konci mesiaca potom nie je potrebné manuálne túto skutočnosť do systému dopĺňať. Dôležitou funkciou je aj možnosť jednoducho vytvoriť mesačný výpis dochádzky zamestnancov vo vhodnom formáte (napr. PDF, Excel). Veľmi zaujímavá funkcia je zobrazenie celkového meškania zamestnanca za mesiac (systém AttendanceProW [2]). Tá síce nie je kľúčová, môže však slúžiť ako zaujímavosť pre spestrenie výpisu.

Mnohé dochádzkové systémy ponúkajú aj možnosť grafického zobrazenia dochádzky. Odpracované hodiny za deň, alebo mesiac je tak možné zobraziť v prehľadnom grafe.

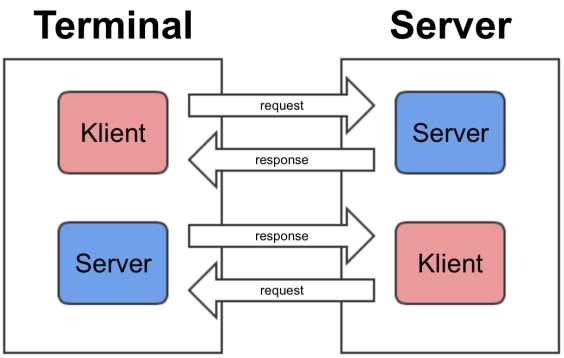
2.5 SW technológie

Pri použití Wi-Fi alebo Ethernet pripojenia musia prístupový terminál a dochádzkový systém byť schopné spolu komunikovať. Takáto komunikácia môže byť riešená rôznymi spôsobmi.

[8] High-Definition Multimedia Interface
[9] Display Serial Interface

Komunikácia server-server

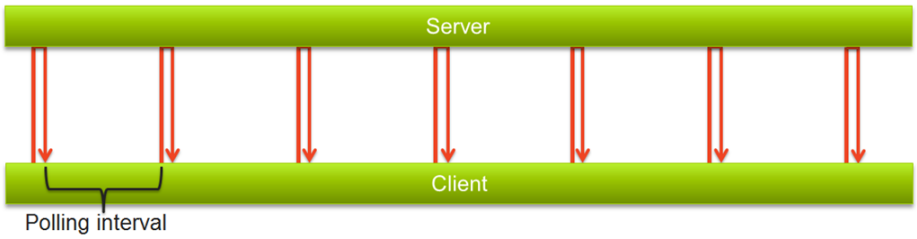
Najjednoduchším riešením komunikácie prístupového terminálu so serverom je vytvoriť webového klienta aj webový server na oboch stranách komunikácie. Ak by potom chcel server odoslať dáta klientovi, jednoducho by sa k nemu pripojil v roli klienta. Tento princíp je zobrazený na obrázku 2.7. Nevýhodou takéhoto riešenia je však zložitá implementácia a jeho neprehľadnosť.



Obr. 2.7: Znázornenie princípu komunikácie metódy server-server.

Polling

Polling je metóda pravidelného testovania. Klient v pravidelných časových intervaloch odosiela HTTP požiadavku na server. Ako odpoveď mu server pošle informáciu o tom, či sa nejaké dáta zmenili. Časový priebeh takejto komunikácie je dobre viditeľný na obrázku 2.8. Nevýhodou tejto metódy je zbytočné zaťažovanie zdrojov klienta a servera, rovnako ako aj zaťažovanie siete. Čím kratší je časový interval medzi dvomi po sebe nasledujúcimi požiadavkami klienta, tým väčšie je toto zaťaženie. Naopak ak je časový interval dlhý, zmena sa prejaví až s oneskorením.



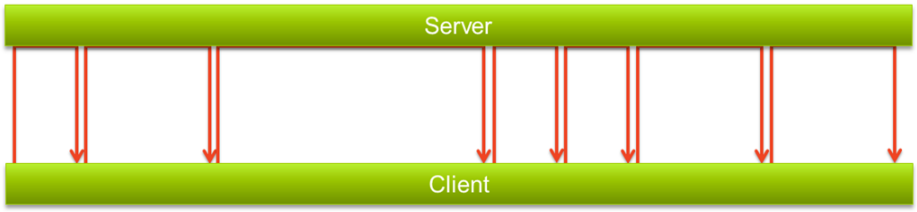
Obr. 2.8: Časový priebeh metódy server polling.[10]

Long polling

Long polling funguje podobne ako metóda polling. Aj pri metóde long polling klient odošle požiadavku na server. Server mu však neodošle odpoveď hneď. Miesto toho udržuje spojenie

[10]Obrázok prevzatý z: http://blog.maartenballiauw.be/image.axd?picture=image_154.png

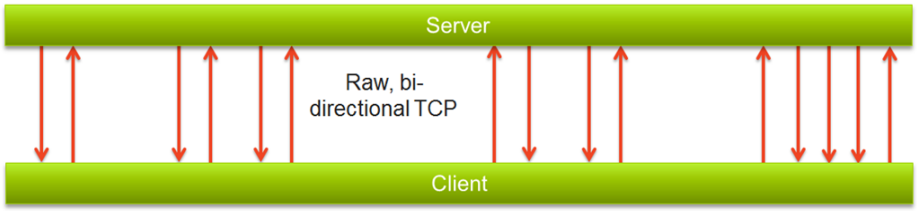
otvorené a odpoveď mu odošle až keď nastane zmena v dátach. Po prijatí odpovede zo servera klient spracuje výsledok a odošle na server novú long polling požiadavku. Časový priebeh metódy long polling je zachytený na obrázku 2.9.



Obr. 2.9: Časový priebeh metódy long polling.[11]

Socket

HTTP je komunikačný protokol. Problém tohto protokolu je však ten, že bol navrhnutý pre iné účely. Jednoduchým riešením problému komunikácie prístupového terminálu s dochádzkovým serverom je teda aj nepoužiť HTTP protokol, ale zísť o úroveň nižšie na úroveň socketov a implementovať si svoj vlastný komunikačný protokol. Sockety umožňujú nadviazať obojsmernú full-duplex komunikáciu medzi klientom a serverom. Časový priebeh komunikácie je zobrazený na obrázku 2.10. Taktiež zvyšujú rýchlosť prenosu a spracovania požiadaviek [16].



Obr. 2.10: Časový priebeh komunikácie cez sockety.[12]

[11] Obrázok prevzatý z: http://blog.maartenballiauw.be/image.axd?picture=image_155.png
[12] Obrázok prevzatý z: http://blog.maartenballiauw.be/image.axd?picture=image_156.png

Kapitola 3

Návrh terminálu

Po preskúmaní súčasnej ponuky prístupových terminálov a dochádzkových systémov bolo potrebné navrhnúť vlastný terminál, dochádzkový systém a komunikáciu medzi nimi. Taktiež je potrebné vybrať správny implementačný jazyk pre terminál, server pre komunikáciu s terminálom a webový server.

3.1 Prístupový terminál

Pred výberom vhodnej platformy je potrebné určiť si požiadavky. Z hardwarového hľadiska to je 2x UART alebo 1x USB, dva GPIO piny a ideálne aj vyvedený resetovací vstup. Taktiež je potrebný video výstup pre pripojenie displeja. Zo softwarového hľadiska je to hlavne podpora QT frameworku a čo možno najnovšia verzia operačného systému Linux.

Keďže už mám nejaké skúsenosti s vývojom takýchto aplikácií, viem že kvalitná dokumentácia a podpora sú pre vývoj veľmi dôležité. Preto som výber zúžil na Beaglebone Black, ktorý má kvalitnejšiu a podrobnejšiu dokumentáciu a Raspberry Pi, ktoré má väčšiu komunitu užívateľov. Počas môjho rozhodovania bol na trh uvedený display pre Raspberry Pi určený pre použitie vo vstavaných systémoch a neskôr bolo vydané aj Raspberry Pi 3 model B, ktoré prinieslo Wi-Fi konektivitu. To je dôvod, prečo som sa nakoniec rozhodol použiť Raspberry Pi.

3.2 Senzorová doska

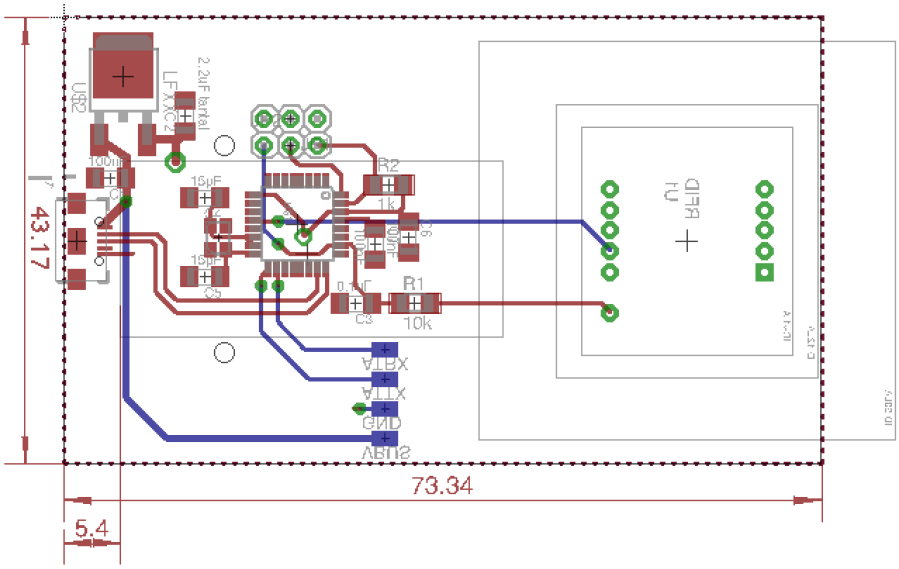
Čítačka RFID štítkov ID-12LA [9] ktorú som sa rozhodol v prístupovom termináli použiť má UART výstup. Čítačka samotná má v sebe zabudovanú určitú inteligenciu a tak napríklad informáciu o priloženom čipe odošle iba raz a potom automaticky vyčká, než je čip odstránený z dosahu, než umožní načítanie ďalšieho čipu. To znamená, že nie je potrebné softwarovo ošetrovať prípad ak je čip načítaný viackrát počas jedného priloženia.

Snímač odtlačkov prstov GT-511C1R ktorý som si vybral pre použitie v mojom prístupovom termináli má priamo na doske umiestnený mikrokontrolér, ktorý sa stará o vytvorenie a spracovanie obrazu odtlačku prsta a popisného reťazca. Nevýhodou tohoto snímača je ale fakt, že je potrebné manuálne zisťovať či je na snímacej ploche priložený prst a ak áno, tak je potrebné zaslať inštrukciu pre vytvorenie obrazu prsta, nasleduje inštrukcia pre vytvorenie popisného reťazca prsta a nakoniec inštrukcia pre spustenie procesu identifikácie.

Keďže oba tieto senzory komunikujú cez UART, pôvodne som chcel pripojiť oba k Raspberry Pi pomocou FTDI 2xUART na USB prevodníka. Po zistení, koľko obsluhy vyžaduje

snímač odtlačkov prstov, rozhodol som sa namiesto toho použiť mikrokontrolér, ktorý by slúžil ako prevodník, no zároveň by aj samostatne obsluhoval snímač odtlačkov, a správu do Raspberry Pi by poslal až po úspešnom, alebo neúspešnom pokuse o identifikáciu.

Rozhodol som sa teda vytvoriť dosku, na ktorú by sa osadili oba tieto senzory, mikrokontrolér a RGB LED indikačná dióda. Napájanie a komunikáciu celej tejto dosky s Raspberry Pi by zabezpečilo USB pripojenie. Po neúspešných pokusoch s mikrokontrolérom TM4C123GH6PM od firmy Texas Instruments som sa na riadenie dosky rozhodol použiť mikrokontrolér ATSAMD21E18A od spoločnosti Atmel. Výsledná doska navrhnutá pomocou návrhového softwaru Eagle je na obrázku 3.1.



Obr. 3.1: Návrh senzorovej dosky pre osadenie čítačkou RFID a čítačkou odtlačkov prstov.

Po návrhu dosky som si ju nechal vyrobiť na CNC fréze. Následne som dosku osadil súčiastkami. Po skontrolovaní dosky na skraty a skontrolovaní kontinuity prepojov som dosku pripojil k napájaniu. Indikačné LED na doske sa hneď po pripojení napájania rozsvietili, čo znamená, že napájanie na doske je v poriadku. Dosku som sa rozhodol programovať cez rozhranie SWD. Keďže však nemám k dispozícii SWD programátor, bol som pre programovanie nútený využiť Raspberry Pi, ktoré je schopné pomocou OpenOCD využívať GPIO piny pre napodobnenie SWD programátora. Viac informácii o tom, ako je možné využívať Linuxový počítač s prístupnými GPIO portmi v tomto režime je na stránke [15].

Po overení funkčnosti programátora som začal programovať čip Atmel ATSAMD21E18A použitý na senzorovej doske. Pre urýchlenie vývoja softwaru ponúka spoločnosť Atmel softwarovú knižnicu ASF. Z neznámych dôvodov mi však žiadny program využívajúci túto knižnicu nefungoval. Aj keď samotný preklad a programovanie boli úspešné, po naprogramovaní ukážkového programu ktorý s pauzami striedavo menil hodnotu na výstupe pinu z logickej “1” na “0” a späť som na danom pine s pripojeným osciloskopom nevidel žiadny priebeh a výstup mal konštantnú hodnotu. Následne som vyskúšal zmeniť používaný pin za iný, no s rovnakým výsledkom. Predpokladal som teda, že je problém buď so zdrojom hodinového signálu, alebo s mikrokontrolérom samotným. Skúsil som však ešte zmeniť časti kódu využívajúce funkcie knižnice ASF za priame zápisy do registrov mikrokontroléra. Po tejto zmene program začal fungovať. Mikrokontrolér, konkrétny výstup a zdroj hodinového

signálu teda boli v poriadku.

Keďže programovať USB komunikáciu a DMA prenosy bez akejkoľvek knižnice by bolo priveľmi zložité, pokúšal som sa sfunkčniť programy využívajúce knižnicu ASF. Ani po niekoľkých dňoch hľadania som však nebol schopný nájsť akýkoľvek dôvod, prečo funkcie knižnice ASF nefungovali. Nakoniec som teda musel celú senzorovú dosku zo svojho systému vypustiť.

3.3 Výber implementačného jazyka

Pred samotnou implementáciou je potrebné zvážiť aj to, v akom jazyku bude systém najvýhodnejšie programovať. Keďže celý systém sa skladá z viacerých častí komunikujúcich spolu, je možné pre každú časť využiť iný implementačný jazyk. Rád by som však použil čo možno najmenšie množstvo jazykov, pretože takým spôsobom sa z daného jazyka naučím viac a zároveň je často jednoduchšie vytvoriť komunikáciu medzi dvomi aplikáciami napísanými v rovnakom jazyku, než medzi dvomi rôznymi jazykmi.

Prístupový terminál

Program prístupového terminálu potrebuje zabezpečiť dve činnosti. Prvou je obsluha senzorov, vyhodnotenie prístupových práv, zabezpečenie patričných z toho plynúcich akcií ako je napríklad otvorenie dverí, vykreslenie grafického užívateľského rozhrania - GUI - a komunikácia s užívateľom. Druhou je komunikácia so serverom, aktualizácia databázy zamestnancov a odoslanie informácií o príchodoch na server.

Lokálna časť

Pred výberom implementačného jazyka je dobré ujasniť si požiadavky podobne ako som to robil pred výberom riadiacej jednotky. Pre potreby prístupového terminálu je potrebné ovládať dva GPIO piny, čítať dáta zo senzorov a komunikovať so serverom. Všetky tieto činnosti sú jednoducho implementovateľné v akomkoľvek programovacom jazyku. Pri komunikácii so senzormi je však potrebné si uvedomiť, že dáta z nich môžu prísť v akomkoľvek okamžiku. Jednou možnosťou ako to riešiť, je pravidelne v aplikácii zisťovať, či senzory majú nejaké nové dáta. Keďže však chcem aby odozva systému bola čo najlepšia, tieto dáta by som rád prijal a vyhodnotil hneď ako sú dostupné. To je opäť možné implementovať v akomkoľvek jazyku pomocou samostatného vlákna aplikácie. Ako logická voľba sa však javí použiť Node.js, čo je v podstate lokálne bežiaci JavaScript. Node.js je totiž udalosťami riadený jazyk s neblokujúcim spracovaním vstupno/výstupných požiadaviek [11]. Všetky podrutiny čakajú až nastane akcia, ktorá ich aktivuje. Zjednodušene povedané, obslužná podrutina spracúvajúca dáta zo senzorov sa napríklad automaticky vykoná pri príjme dát zo senzorov.

GUI

V tejto časti svojej práce som popísal dva spôsoby, ktorými je možné vytvoriť grafické užívateľské rozhranie prístupového terminálu. Existujú aj iné riešenia, tie som sa však z rôznych dôvodov rozhodol nepoužiť.

HTML/CSS Keďže sa Node.js javil ako najvhodnejší implementačný jazyk, ktorý by som mohol využiť pre implementáciu všetkých častí systému, začal som hľadať spôsob ako implementovať grafické užívateľské rozhranie pre aplikáciu v tomto jazyku. Tento jazyk je určený pre vytváranie serverových aplikácii a preto neposkytuje žiadnu možnosť ako priamo manipulovať s grafickými prvkami. Jeho primárnym určením je vytvoriť server, spracovať požiadavky klienta a generovať HTML kód, ktorý sa preposiela klientovi ako odpoveď na jeho požiadavky. Na strane klienta sa o vykresľovanie prijatých dát stará prehliadač. Veľmi zaujímavým riešením je ale NW.js [12]. Ten v pozadí spustí Node.js aplikáciu slúžiacu ako web server a následne otvorí okno prehliadača, ktoré sa na tento lokálny web server pripojí a zobrazí webovú stránku. Akékoľvek akcie v tejto aplikácii sa potom posielajú ako HTTP požiadavky na server, ktorý ich spracuje. Koncovému užívateľovi, ktorý nevie o existencii webového servera v pozadí, ani o odosielaní požiadaviek sa to celé javí ako bežná aplikácia. Vďaka tomu je teda možné implementovať aplikačnú logiku v Node.js a grafické užívateľské rozhranie vytvoriť v HTML a CSS.

QT Možnosť tvoriť GUI ponúka aj framework QT [13]. Je to však framework pre jazyk C++. Existuje aj wrapper pre jazyk Node.js, ten však nie je potrebný. QT totiž umožňuje využívať neblokujúce vstupno/výstupné inštrukcie. Tie fungujú obdobne ako v jazyku Node.js, kde je po dokončení požiadavky zavolaná callback funkcia. V tej je možné spracovať výsledky požiadavky. Tento framework (rovnako ako Node.js) umožňuje vytvárať a používať časovače. To je veľmi užitočná vlastnosť, pretože je možné takýto časovač použiť pre časovanie rôznych akcií. Keďže QT framework a Node.js oba ponúkajú vhodné prostriedky pre implementáciu aplikácie prístupového terminálu, no Node.js je interpretovaný jazyk a umožňuje tvorbu grafických aplikácii len okľukou, rozhodol som sa použiť QT v kombinácii s C++.

Server

Druhú časť mojej práce tvorí dochádzkový systém, ktorý bude bežať na firemnom servery. Tento systém bude spracovávať dochádzku zamestnancov a bude umožňovať vykonávať dodatočné úpravy. Dochádzkový systém nepotrebuje priamo komunikovať s prístupovými terminálmi, môže pracovať len s dátami uloženými v databáze. To znamená, že komunikáciu s prístupovými terminálmi a vkladanie dát do databázy môže zabezpečovať samostatná aplikácia. Obe tieto časti teda môžu byť implementované v rôznych jazykoch. Radšej by som však obe tieto časti implementoval v rovnakom jazyku a pokiaľ to nebude mať výrazný dopad na prehľadnosť kódu, alebo implementačnú náročnosť, rád by som ich spojil do jednej aplikácie.

PHP

PHP je pravdepodobne najrozšírenejší jazyk pre tvorbu dynamických stránok. Je to však bezstavový jazyk, čo znamená, že si nedokáže udržať trvalé spojenie s terminálom cez sockety. Aplikáciu by som teda musel rozdeliť na dve časti. PHP má však vysokú mieru podpory na serveroch a taktiež existuje veľké množstvo frameworkov pre tento jazyk (napr. Laravel), ktoré ho rozširujú a zjednodušujú prácu s ním. Jeho výhodou je aj to, že je to časom overené riešenie.

Python

Python umožňuje vytvoriť sockety a komunikovať tak s terminálmi. Existuje na neho veľké množstvo modulov a frameworkov (napr. Django, Twisted) ktoré rozširujú jeho funkcie. Podobne ako PHP je to časom overený jazyk.

Node.js

Node.js je v podstate JavaScript bežiaci na strane servera. Aj napriek tomu, že je to pomerne nové riešenie, už teraz existuje veľké množstvo rôznych modulov a frameworkov rozširujúcich základnú funkčnosť celého jazyka. K Node.js je pribalený aj program npm slúžiaci na jednoduché doinštalovanie týchto modulov do projektu. Rovnako ako Python, aj Node.js umožňuje vytvárať spojenie s terminálom cez socket.

3.4 Komunikácia terminál-server

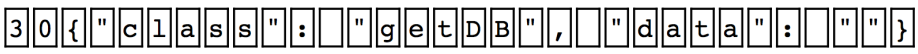
Opäť je dobré začať ujasnením si požiadaviek. Prístupový terminál musí pri každom vstupe zamestnanca informovať dochádzkový systém. Musí si taktiež byť schopný vyžiadať zaslanie najnovšej databázy zamestnancov. Terminál bude na server a späť prenášať aj rôzne iné informácie. Všetky tieto informácie by bolo možne prenášať pomocou HTTP požiadaviek. Pre potreby prístupového terminálu je však potrebné vytvoriť obojsmernú komunikáciu. To je možné riešiť ľubovoľným zo štyroch spôsobov popísaných v kapitole 2.

Vytvorenie samostatného webového klienta a servera na oboch stranách by bolo zbytočne komplikované riešenie, ktoré by bolo zo všetkých riešení výpočtovo najnáročnejšie. V porovnaní s ostatnými riešeniami nemá táto alternatíva žiadne výhody, takže som ju z výberu vylúčil. Druhá alternatíva bola metóda polling. Táto metóda je opäť náročnejšia na výpočtový výkon než ďalšie zostávajúce varianty. Server je zbytočne zaťažovaný množstvom požiadaviek a medzi vznikom zmeny v dátach na servery a ich prenosom na terminál vzniká oneskorenie, ktoré síce nie je pre potreby prístupového terminálu podstatné, je však zbytočné. Tretia alternatíva je využiť metódu long polling. Táto metóda odstraňuje nedostatky predošlej metódy, je však mierne zbytočná (nie je dôvod využívať pre prenos HTTP požiadavky) a nepraktická na implementáciu. Najlepšou možnosťou teda je použiť pripojenie pomocou socketov, ktoré sú najrýchlejším spôsobom komunikácia a spĺňajú aj požiadavky na obojsmernú komunikáciu.

Keďže pri použití socketov neprijme prijímateľ dáta ako ucelený blok, ale tie môžu prísť vo viacerých častiach, je potrebné vedieť, aká veľká je správa pred tým, než sa ju prijímateľ pokúsi prečítať. V opačnom prípade by totiž hrozilo, že sa pokúsi správu interpretovať kým je ešte neúplná, alebo by mohol prijať aj časť následujúcej správy, čím by znemožnil interpretáciu oboch správ. Prvou informáciou, ktorá sa musí preniesť teda je dĺžka samotných dát. Najjednoduchšie je určiť si pevný počet bajtov, v ktorých bude dĺžka zakódovaná. Ja som si zvolil dĺžku päť bajtov. Prvých päť bajtov každej vymenenej správy bude teda obsahovať dĺžku správy.

Následne sa prijímateľ musí dozvedieť, ako ma dáta interpretovať. Telo každej správy preto bude tvoriť JSON objekt. Tento objekt bude vždy obsahovať len dve položky, a to “class” a “data”. V položke “class” je uložený typ správy (napr. “getDB”). Po prečítaní typu správy príjemca zistí o aké dáta sa jedná. Samotné dáta sú potom uložené v položke “data”. Takýto komunikačný protokol je jednoducho rozšíriteľný a umožňuje komunikovať aj zariadeniam s rôznymi verziami tohto protokolu. Ak napríklad bude mať jedno zariadenie

novšiu verziu programu a k dátam pripojí viac informácií, starší program túto správu bez problémov príjme a vyberie si z nej len informácie ktorým rozumie. Názorná ukážka správy ktorú odosiela prístupový terminál na server pre vyžiadanie najnovšej verzie databázy je na obrázku 3.2.



Obr. 3.2: Znázornenie správy posielanej medzi serverom a terminálom cez socket.

Kapitola 4

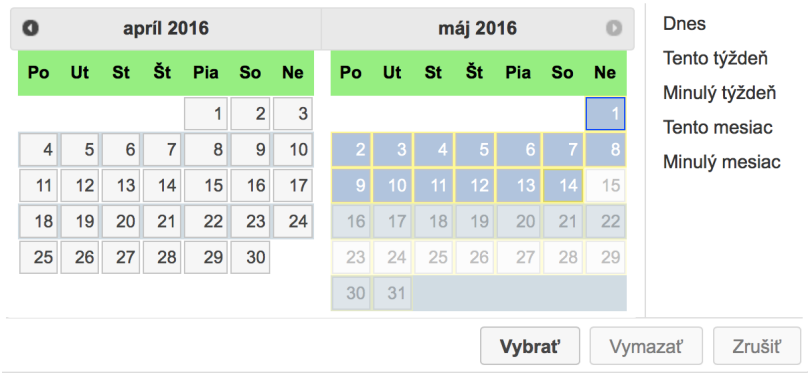
Zostavenie a implementácia

4.1 Webový server

Webový server je naprogramovaný v jazyku Node.js. Pre tvorbu serverovej časti bol použitý framework Total.js [17]. Je to MVC framework, čo znamená, že kód servera je rozdelený na model (Model), zobrazenie (View) a kontrolér (Controller). Model je zdrojový súbor obsahujúci JavaScript funkcie zabezpečujúce prácu s modelom (tabuľkou) v databáze -napr. zamestnanec. Zobrazenie (view) obsahuje HTML kód, ktorý môže byť odoslaný klientovi ako odpoveď na jeho požiadavku. Framework však umožňuje aj mnoho ďalších vylepšení, ako je napríklad použitie šablón, čo sú špeciálne vyznačené jednoduché príkazy jazyka JavaScript, ktoré vyhodnotí server pred odoslaním súboru a môžu samotný súbor upraviť. Je teda možné napísať podmienku a HTML kód sa do súboru doplní na základe jej pravdivosti, alebo je možné veľmi výhodne využiť cyklus foreach pre vloženie všetkých prvkov poľa do nejakého elementu (tabuľka, zoznam). Poslednou časťou MVC frameworku je kontrolér (controller), ktorý vyhodnocuje samotné požiadavky od klientov, reaguje na ne a odosiela odpovede. Total.js k týmto súborom pridáva ešte rôzne ďalšie - napríklad definície (Definitions) kde sú uložené nastavenia ktoré sa majú nastaviť pre všetky modely a kontroléry, alebo moduly (Modules) kde sa nachádzajú rozširujúce moduly webového servera ako napr. overovanie identity užívateľa.

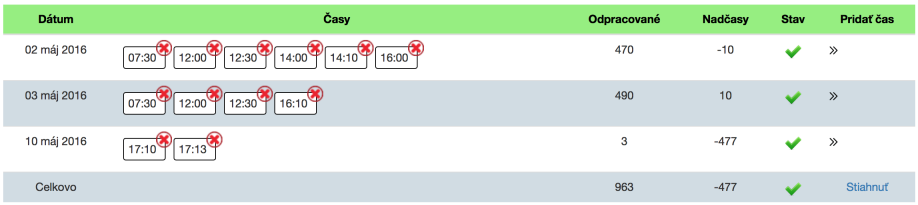
Stránka sa skladá z dvoch hlavných podstránok. Prvou sú vstupy. Na tejto stránke je možné vybrať si zamestnanca zo zoznamu a rozsah dátumov ktorými majú byť vypísané dáta ohraničené. Výber je možné uskutočniť buď pomocou prednastavených makier (napr. dnes, tento týždeň, tento mesiac) alebo priamym určením počiatočného a konečného dňa intervalu. Rozhranie pre výber rozsahu dátumov je možné vidieť na obrázku 4.1.

Dáta vstupov sú vypisované formou tabuľky. Jednotlivé dni tvoria riadky tabuľky. Do prvého stĺpca sa vypíše dátum. Do druhého stĺpca sa vloží vnorená tabuľka do ktorej sa vložia jednotlivé časy. Takýto postup bol zvolený pretože to je najefektívnejšie riešenie ako zabezpečiť aby boli všetky riadky s časmi rovnako široké a aby tak všetky nasledujúce stĺpce hlavnej tabuľky ostali zarovnané, ako aj to, aby boli jednotlivé časy od seba programovo jednoducho odlíšiteľné, čo je potrebné pre ďalšiu funkcionalitu a zjednodušuje to aj tvorbu vzhľadu časov. V nasledujúcom stĺpci sa vypíše dĺžka nadčasov v minútach za daný deň. Ak zamestnanec odpracoval napríklad o 10 minút viac než 8 hodín, zobrazí sa +10, naopak ak odpracuje o 10 minút menej, zobrazí sa -10. Keďže dôležitou úlohou je aby boli tieto údaje jednoducho pochopiteľné, tabuľka obsahuje aj stĺpec znázorňujúci stav daného dňa. Ak server zistí, že v danom dni je uložený nepárny počet záznamov daného zamestnanca, vyhodnotí riadok tohto dňa ako obsahujúci chybu a do stĺpca so stavom sa vykreslí veľký



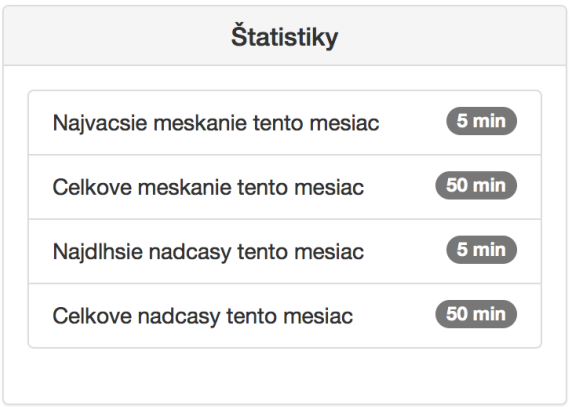
Obr. 4.1: Výber rozsahu dátumov pre zobrazenie vstupov a výstupov. Na pravej strane sú prednastavené hodnoty.

červený krížik. Ak sú naopak dáta v poriadku, do stĺpca so stavom sa vykreslí sa zelená ikona symbolizujúca všetko je v poriadku. Celá tabuľka je zobrazená na obrázku 4.2.



Obr. 4.2: Prístupy zamestnanca sú zobrazené v tabuľke.

Za samotnou tabuľkou sú nakoniec zobrazené ešte štatistiky. Tieto štatistiky sú však skôr informatívneho charakteru a sú tu uvedené hlavne pre zaujímavosť. Obsahujú informácie ako maximálne a celkové nadčasy zamestnanca za zobrazené obdobie, alebo maximálne a celkové meškanie zamestnanca za dané obdobie.



Obr. 4.3: Zaujímavé štatistiky zamestnanca.

Prihlasovanie

Vstup do webovej aplikácia dochádzkového systému musí byť zabezpečený. Pre prístup bude preto potrebné zadať prihlasovacie meno a heslo. Tie musia byť uložené v databáze. Nie je však vhodné, aby boli hesla uložené len tak, v textovej forme. Preto je potrebné z hesla pred uložením do databázy pomocou špeciálneho algoritmu vyrobiť hash. Tento algoritmus musí byť jednosmerný, aby nebolo možné z hashu spätne získať heslo a musí pre každé dve rôzne hesla generovať rôzne hashe, aby neboli heslá zameniteľné. Takéto hashovanie hesiel má aj ďalšiu výhodu, a to tú, že vzniknuté hashe majú konštantnú dĺžku bez ohľadu na dĺžku vstupnej hodnoty, takže ich uloženie v databáze je jednoduchšie. Po hashovaní hesla je z bezpečnostného hľadiska dobré k hashu pripojiť ešte tzv. soľ (anglicky salt), čím sa zvýši zložitosť prelomenia takéhoto hesla.

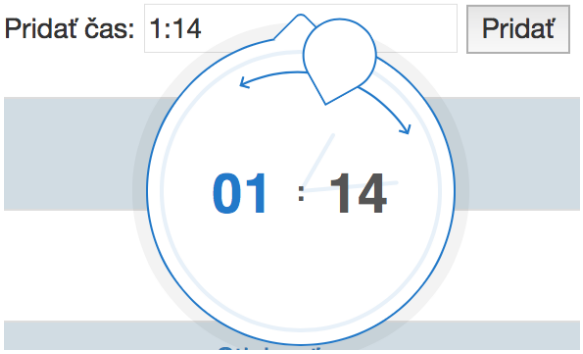
Pre Node.js existuje knižnica s názvom Bcrypt, ktorá ponúka jednoduché API. Táto knižnica umožňuje hashovať vstupné heslo a automaticky pridať soľ. Výsledok je potom možné bezpečne uložiť do databázy. Táto knižnica ponúka taktiež aj možnosť zadať heslo a hash pre porovnanie. V takom prípade knižničná funkcia vytvorí z hesla hash, pridá soľ a porovná výsledok s hashom zadaným v argumente. Návratovou hodnotou je potom premenná typu bool s hodnotou true ak sú heslá rovnaké, alebo false ak nie sú.

Úprava uložených údajov

Dáta z prístupových terminálov je občas potrebné pred finálnym spracovaním upraviť. Je totiž potrebné pridať chýbajúce údaje, poprípade odstrániť duplicitné hodnoty. Preto sa v pravom hornom rohu každého zobrazeného času vždy nachádza malé X, ktoré umožňuje daný časový údaj pohodlne odstrániť. Takéto riešenie bolo vybrané pre svoju intuitívnosť a užívateľskú oboznámenosť - takéto riešenie používajú mnohé súčasné počítačové systémy. Pred samotným odstránením času je však užívateľ vždy požiadaný o potvrdenie vymazania. Systém vyhodnocuje formát uložených dát pre jednotlivé dni. Pokiaľ je v dátach niektorého dňa objavená chyba, na konci riadku sa zobrazí možnosť pridať do daného dňa čas. Ak je formát dát v poriadku, na konci riadku sa zobrazí len malá šípka, ktorá slúži na zobrazenie možnosti pridať čas. To plní dvojitú úlohu. Po prvé to zvyšuje prehľadnosť stránky, pretože užívateľ sa tak jednoduchšie zorientuje ku ktorému dňu pridanie času patrí. Po druhé to znižuje nároky stránky na počítač užívateľa, pretože každý vstup času funguje cez JavaScript.

Pridanie času

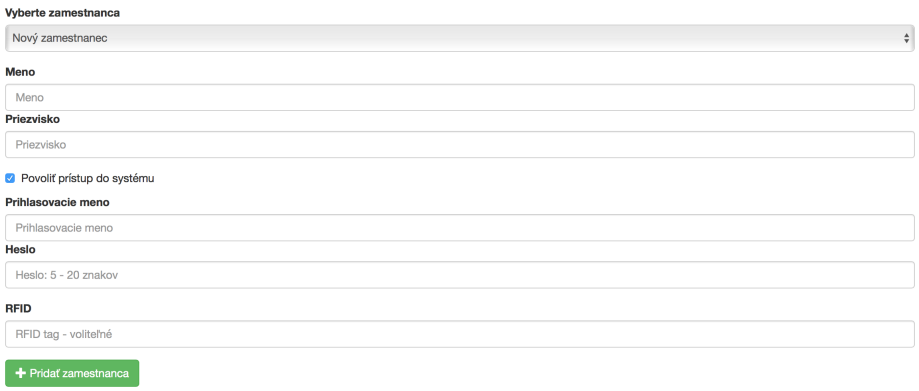
Po kliknutí na vstup času sa objaví okrúhly ciferník vyobrazený na obrázku 4.4, na ktorom je možné ťahaním ručičky po obvode navoliť hodiny a následne je možné rovnakým spôsobom zvoliť aj minúty. Kliknutím na hodiny, resp. minúty je možné prepínať medzi nastavovaním jedného alebo druhého. Je možné zapnúť aj možnosť automatického prepínania, kedy sa po navolení hodín automaticky prepne na voľbu minút a naopak, no túto možnosť som po chvíli testovania vypol, aby zbytočne nemiatla užívateľa. Takýto typ vstupu bol zvolený preto, aby bol čo možno najintuitívnejší a jednoducho ovládateľný aj z tabletu. Po potvrdení navoleného času sa čas zapíše do vstupného políčka času. Kliknutím na tlačidlo pridať sa čas odošle na server kde sa pridá do databázy.



Obr. 4.4: Rozhranie pre navolenie času príchodu alebo odchodu.

Správa zamestnancov

Druhou hlavnou podstránkou je správa zamestnancov. Na tejto podstránke je možné pridať nového zamestnanca, alebo upraviť údaje už existujúceho zamestnanca. Taktiež je možné zaškrtnutím možnosti "povoliť prístup do systému" určiť, či bude mať daný zamestnanec prístup do webového rozhrania dochádzkového systému. Po zaškrtnutí tejto možnosti sa objavia vstupné polia pre zadanie prihlasovacích údajov pre zamestnanca. Nezaškrtnutím tejto možnosti sa prístup do systému pre daného zamestnanca zakáže, aj ak ho mal pred prevedením zmien povolený.



Obr. 4.5: Vstupný formulár pre pridanie zamestnanca do systému.

4.2 Komunikácia terminál-server

Rovnako ako webový server, aj server pre komunikáciu s terminálom je naprogramovaný v jazyku Node.js. Do hlavnej aplikácie je pripojený ako modul. Takéto riešenie umožnilo prepojiť oba servery do jednej aplikácie, takže spolu môžu navzájom jednoducho komunikovať a využívať spoločné časti kódu. Zároveň sú však oba kódy oddelené, čo zlepšuje prehľadnosť kódu. Modul pre komunikáciu s terminálom využíva "net" modul jazyka Node.js pre vytvorenie oddeleného servera od webového servera, ktorý prijíma spojenia na porte 6969. Hlavnú časť tohto servera tvorí switch, ktorý na základe prijatej požiadavky vykoná očakávanú reakciu.

4.3 Prístupový terminál

Aplikácia pre prístupový terminál je implementovaná v jazyku C++ s použitím frameworku QT. Celá aplikácia je objektovo orientovaná. Tvoria ju štyri základné objekty - network, database, GUI a sensorReader, ktoré sú spolu prepojené pomocou systému QT signálov a slotov. Je to alternatíva ku klasickému zasielaniu správ medzi objektami, no objekty na seba nemusia mať navzájom referenciu. Jednotlivé signály a sloty je možné navzájom prepojiť volaním funkcie "QMetaObject::Connection QObject::connect(const QObject *sender, const char *signal, const QObject *receiver, const char *method, Qt::ConnectionType type = Qt::AutoConnection)". Po vyvolaní signálu je potom automaticky zavolaná pripojená metóda (slot) a sú jej predané všetky parametre zadané signálu.

Objekt network má na starosti sieťovú komunikáciu. Akékoľvek požiadavky na server sú posielané práve cez tento objekt, ktorý udržuje konštantne otvorené spojenie na úrovni socketu s časťou servera pre terminály. Po prijatí odpovede na požiadavku o tejto udalosti informuje objekt, ktorý požiadavku zaslal, alebo objekt, pre ktorý je odpoveď určená, ak sa nejedná o ten istý objekt.

Objekt sensorReader slúži pre komunikáciu prístupového terminálu s pripojenými senzormi RFID čipov a senzorom odtlačkov prstov. Po prijatí správy od senzora prepošle načítané údaje objektu database, ktorý vyhodnotí vstupné oprávnenia a pokračuje v spracovaní, čím sa úloha objektu sensorReader končí.

Objekt database sám o sebe nevykonáva žiadne akcie. Miesto toho čaká a reaguje na správy od iných objektov. Jednou takou správou je napríklad prijatie databázy od sieťového objektu. V takom prípade objekt databázu uloží. V prípade správy od objektu sensorReader o načítaní kódu z priloženého čipu sa objekt pokúsi nájsť kód v databáze zamestnancov. Ak je kód nájdený, prístup je povolený a objekt informuje objekt network o tejto udalosti. Sieťový objekt následne zabezpečí zápis tejto udalosti do databázy servera. V prípade, ak načítaný kód v databáze nie je nájdený, považuje sa pokus o vstup za neoprávnený. V oboch prípadoch je odoslaná správa do objektu GUI pre vykreslenie informácie o (ne-)rozpoznaní zamestnanca na display.

Užívateľské rozhranie aplikácie je zobrazené na obrázku 4.6. Zachytené bolo v momente, keď som k čítačke RFID čipov priložil čip uložený v databáze v mojom osobnom zázname. Zariadenie úspešne overilo identitu, odoslalo požiadavku na uloženie prístupu do databázy na server a na obrazovke vypísalo meno zamestnanca, teda moje meno.

Obr. 4.6: Užívateľské rozhranie terminálu zobrazuje aktuálny čas a aktuálny príchod zamestnanca.

Kapitola 5

Záver

Cieľom tejto bakalárskej práce bolo preskúmať súčasnú ponuku prístupových terminálov a dochádzkových systémov na trhu, preskúmať možnosti rôznych senzorov, ktoré sa používajú pre identifikáciu osôb, navrhnúť a implementovať terminál založený na Raspberry Pi, navrhnúť a implementovať dochádzkový systém pre zber a správu dochádzky a otestovať výsledný systém.

Podarilo sa mi splniť všetky body zadania. Požiadavkou bolo výsledný systém aj otestovať. Systém ako celok som preto pred odovzdaním práce chcel nasadiť do plnej prevádzky. Z časových dôvodov sa mi však v tomto systéme nepodarilo implementovať všetky súčasti, ktoré som chcel implementovať. Aj preto som systém nemohol nasadiť do prevádzky. Počas vývoja som však niektoré časti systému konzultoval s konečnými užívateľmi a niektoré časti som následne upravil tak, aby lepšie spĺňali ich požiadavky.

Výsledný systém je možné použiť ako prístupový terminál a dochádzkový systém. Stále mu však chýbajú niektoré pokročilejšie súčasti, ktoré by umožnili nasadiť systém do plnej prevádzky. Príkladom takýchto súčastí, je napríklad senzorová doska, ktorú som pre túto prácu navrhol, no ktorú som nakoniec nebol schopný pred odovzdaním dokončiť.

Literatúra

[1] AlarmTel S990.
URL http://www.alarmtel.net/DOCHADZKOVY-TERMINAL-S990-d142.htm

[2] AttendanceProW.
URL http://www.biometric.sk/dochadzkovy-system

[3] Beagleboard official webpage.
URL http://beagleboard.org

[4] BIOMETRIC SCR100.
URL http://www.dochadzka.net/terminaly/kontrola-vstupu/scr100

[5] Biometricke riesenia Aktion.
URL
http://www.aktion.sk/sk/el-kontrola-vstupu/biometricke-riesenia.html

[6] DMR-System DT1000.
URL http://www.dmrsystem.sk/www/sk/dochadzkovy-terminal-dt1000-2/

[7] Fingerprint analysis.
URL http://computer.howstuffworks.com/fingerprint-scanner4.htm

[8] FLOWii.
URL https://www.flowii.com

[9] ID-12LA RFID reader.
URL http://www.id-innovations.com/httpdocs/Modules(non%20write).htm

[10] Intel Edison official webpage.
URL http://www.intel.com/content/www/us/en/do-it-yourself/edison.html

[11] Node.js official page.
URL https://nodejs.org/en/

[12] NW.js official page.
URL http://nwjs.io

[13] QT.
URL http://www.qt.io

[14] Raspberry Foundation official page.
URL https://www.raspberrypi.org

[15] Raspberry Pi SWD programátor.
URL https://petervanhoyweghen.wordpress.com/2015/10/11/burning-zero-bootloader-with-beaglebone-as-swd-programmer/

[16] Sockets vs HTTP speed comparsion.
URL http://blog.arungupta.me/rest-vs-websocket-comparison-benchmarks/

[17] Total.js Node.js framework.
URL https://www.totaljs.com

[18] UDOO official webpage.
URL http://www.udoo.org

Zoznam príloh

Príloha A

Obsah CD

- Aplikácia pre prístupový terminál
- Webová aplikácia pre dochádzkový systém
- Súbory návrhu senzorovej dosky pre program Eagle

Príloha B

Schéma zapojenia senzorovej dosky