VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Uživatelské rozhraní zabezpečovacího terminálu
User interface for security terminal

2016 Libor Chrástecký

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student:	**Libor Chrástecký**
Studijní program:	B2649 Elektrotechnika
Studijní obor:	2612R041 Řídicí a informační systémy
Téma:	Uživatelské rozhraní zabezpečovacího terminálu User Interface for Security Terminal
Jazyk vypracování:	čeština

Zásady pro vypracování:

1. Popis technologií SmartHome.
2. Technická specifikace TI BeagleBone.
3. Popis softwarových prostředků Linux a Java.
4. Návrh technického řešení uživatelského rozhraní.
5. Realizace a testování zhotoveného prototypu.
6. Testování zabezpečovacího terminálu.
7. Zhodnocení výsledků práce.

Seznam doporučené odborné literatury:

[1] SOBELL, Mark G. *Linux: praktický průvodce.* Vyd. 1. Praha: Computer Press, 1999, xxii, 946 s. Operační systémy. ISBN 80-7226-190-8.
[2] SCHILDT, Herbert. *Mistrovství - Java.* 1. vyd. Brno: Computer Press, 2014, 1224 s. Mistrovství. ISBN 978-80-251-4145-8.
[3] BEAGLEBOARD.ORG.*Beagleboard.org* [online]. 2015 [cit. 2015-09-29]. Dostupné z: http://beagleboard.org/BLACK.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Prauzek, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě dne 29. 4. 2016

Podpis autora: Chrástecký

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Michalu Prauzkovy, Ph.D za věcné rady a schovívavost, dále děkuji panu Ing. Jaromíru Konečnému, Ph.D za odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Ostravě dne 29. 4. 2016 Podpis autora: Chrástecký

Abstrakt

Cílem této práce je vytvořit návrh a realizaci zabezpečovacího terminálu na vývojové platformě BeagleBone Black. Práce se zabývá teorií kolem možností použití smarthome technologií, popisem softwarových prostředků operačního systému Linux a popisem softwarovými prostředky Java. Popisuje vlastnosti platformy BeagleBone Black a také poskytuje vlastnosti dalších platforem řady BeagleBone. Dále se práce zabývá vlastnostmi použitého rozšiřujícího modulu pro zabezpečovací terminál, kterým je modul dotykového displeje. V této práci se řeší návrh řídícího programu zabezpečovacího terminálu a realizaci toho zabezpečovacího terminálu. Realizace zabezpečovacího terminálu obsahuje přípravu platformy pro práci a sepsaní řídícího programu pro zabezpečovací terminál. Na závěr se práce zabývá testováním prototypu zabezpečovacího terminálu.

Klíčová slova

Zabezpečovací terminál, SmartHome, Linux, Java, BeagleBoard, BeagleBoard.org, BeagleBone, BeagleBone Black, Displej, Ethernet, UDP, BB-view

Abstract

The purpose of this thesis is design and realization of security terminal on development platform BeagleBone Black. The thesis is deal with theories about possibillity of using SmartHome technology, describing the software resources of operating system Linux and description Java software resources. It descibes the characteristics of BeagleBone Black platform and provides features of other platforms series BeagleBone. Futhermore, the thesis deals with the features of expansion module for secutity terminal, witch is a touch screen module. In this thesis solves a design of security terminal control program and realization of security terminál. Realization of security terminal includes preparing the platform for work and programming control program for security terminal. At the conclusion of the thesis is focused on testing a prototype security terminal.

Keywords

Security terminal, Smart Home, Linux, Java, BeagleBone, BeagleBoard.org, BeagleBone, BeagleBone Black, display, Ethernet, UDP, BB-view

Obsah

Seznam použitých symbolů a zkratek

GUI	Grafický uživatelský interface
SmartHome	Chytrý dům, inteligentní dům
UDP	User datagram protocol

Seznam použitých obrázků a tabulek

1. Úvod

Tato práce popisuje návrh řešení a realizací zabezpečovacího terminálu na platformě BeagleBone Black. Tento terminál umožní přihlášení do řídícího programu, kde bude uživatel mít možnost měnit stavy několika prvkům, nacházejících se v okruzích. Požadavky na změnu stavu budou posílány prostřednictvím UDP paketů na síti Ethernet. První část této práci popisuje teorií kolem možných použití technologií SmartHome. Následně se popíší možnosti operačního systému Linux, nejvíce možné obslužné programy, kterými tento operační systém disponuje pro práci prostřednictvím terminálu. Další část práce obsahuje seznam dostupných platforem od společnosti BeagleBoard.org a technické parametry použité platformy BeagleBone Black i s popisem použitého rozšiřujícího modulu BB-view. Poslední část teoretického rozboru práce popisuje možné softwarové prostředky Java. Následné části práce popisují praktické pracování zabezpečovacího terminálu. První část v praktickém zpracování popisuje návrh zabezpečovacího terminálu. Poté již následuje realizace zabezpečovacího terminálu. Realizace je rozdělená do dvou částí. První části popisuje přípravu platformy na práci, tím je myšleno, nainstalování požadovaných driveru pro komunikaci mezi platformou a počítačem, nainstalování driverů rozšiřujícího modulu a také nainstalování Javy. Součástí realizace je popis i dalších požadovaných postupů, například nastavení spouštění grafických aplikací na rozšiřujícím modulu. Druhá část realizace popisuje řídící program pro zabezpečovací terminál a jeho jednotlivé části. V závěru, práce popisuje testování prototypu zabezpečovacího terminálu a jeho jednotlivých částí. Těmito částmi jsou přihlašování a komunikace. Přihlašovací část popisuje práci přihlašování. Komunikační část popisuje schopnost komunikace zabezpečovacího terminálu, v této části se testuje posílání zpráv a přijímaní zpráv mezi pracovním počítačem, na kterém byla spuštěna testovací aplikace a zabezpečovacím terminálem.

2. SmartHome

V této kapitole budou popsány některé výhody technologiích SmartHome, a také obecný popis některých z technologií SmartHome, které lze použít ve spojení ze zabezpečovacím terminálem popřípadě k modifikaci zabezpečovacího terminálu.

2.1. Výhody inteligentních domů

Ovládání veškeré techniky

Díky propojení všech systémů do jednoho společného celku a možnosti programovat funkci každého vypínače lze zcela změnit ovládání v porovnání s běžným domem. Na rozdíl od klasického manuálního ovládání jednotlivých prvků lze místo toho vytvořit tzv. scény nebo režimy, tyto nám umožní předem definovat různé stavy budovy a pomocí jednoho či více stisků tlačítka přejít do jiné scény nebo režimu. [7]

Komfort a pohodlí

Tím, že můžeme ovládat veškerou technologii snadněji pomocí několika spínačů a také díky tomu, že můžeme upravovat, činnosti jednotlivých spínačů, má za výsledek zjednodušení prováděných akcí a tím i komfortu a pohodlí obyvatel. [7]

Bezpečnost

Právě díky jednoduchému ovládání a automatizace se zajistí, že bezpečnostní systém bude zapnut vždy, když bude zapotřebí. Může se zapínat samočinně při zamknutí vchodových dveří nebo před spánkem tlačítkem u postele. Alarm můžeme deaktivovat bezpečným způsobem zadáním vložením kódu pomocí dotykového panelu. Dům dokáže nasimulovat přítomnost lidí a samočinně ovládat světla, rolety a další zařízení. [7]

Úspory energií

Pomocí elektrické regulace topení a osvětlení se dá uspořit náklady za energie. Požadované teploty se dají nastavit zvlášť pro různé místnosti. Po odchodu z domu se sníží teplota a vypnout všechny zapomenutá světla. Po otevření okna se zastaví topení. Díky snímačům úrovně venkovního osvětlení je světlo rozsvíceno pouze na požadovanou intenzitu. Energeticky náročnější spotřebiče pracují v čase, kdy je nižší cena elektrické energie. [7]

2.2. Snížení nákladů za energie

Vytápění

Na rozdíl od běžných způsobu regulace, kdy je v domě pouze jeden termostat a ten řídí vytápění všech místností, je komfortnější a úspornější regulovat vytápění v každé místnosti zvlášť, nezávislé na ostatních místnostech. Abychom toto mohli provést, je nutné měřit teplotu v každé místnosti a místo klasických termoregulačních hlavic nebo ventilů použít elektricky ovládané. Samozřejmě je také nutné přizpůsobit topný systém k regulaci každé místnosti samostatně. [7]

Vytápění lze řídit automatickým nebo manuálním přepínáním, časovými programy, dálkovým ovládáním a speciálními akcemi, například zablokování při otevřeném okně.

Ventilace

Pro zajištění optimální výměny vzduchu a snížení energetické náročnosti je vhodné použít mechanické větrání. Můžeme tak využít rekuperaci (zpětné získávání tepla), kdy se vzduch přiváděný do budovy předehřívá teplým odpadním vzduchem. Teplý vzduch tak není odveden bez užitku ven, ale v rekuperačním výměníku předehřívá přiváděný vzduch. [7]

Instalováním snímačů C02, vlhkosti a přítomnosti osob lze automaticky regulovat množství vzduchu v místnosti. [7]

Klimatizace

Klimatizace v sobě zahrnuje veškeré úpravy vzduchu – filtrace, ohřev, chlazení, zvlhčování, odvlhčování. Dosažení úspor je v zásadě podobná jako u vytápění a to tedy měřením a regulací teploty individuálně v jednotlivých místnostech, využití snímačů přítomnosti osob, blokování klimatizace při otevřeném okně, časové programy. Zde se snažíme snížit teplotní zisky. Proto musí klimatizace pracovat s dalšími prvky domu, aby nedocházelo ke zbytečným ztrátám. [7]

Stínění

V letním období je ekonomičtější chránit osluněná okna před nežádoucími teplotními zisky, například použitím žaluzií, markýz, fólií apod., než se spoléhat pouze na energeticky náročnou klimatizaci. Stínění lze pomocí použití snímačů automaticky nastavovat v závislosti na teplotě v místnosti nebo intenzitě slunečního záření. [7]

Ohřev teplé vody

Snížení energie lze dosáhnout použitím solárních kolektorů. Dalším vhodným opatřením může být automatické řízení teploty výtokové teplé vody. Také lze zařídit cirkulaci teplé vody v potrubí. [7]

Osvětlení

Snížení spotřeby je možné zařídit zapojením automatického spínání světel v místnostech, kde se osoby zdržují jen krátkou dobu, například použitím snímačů pohybu nebo přítomnosti osob. Za to v místnostech, ve kterých se osoby pobývají déle lze úsporu zařídit automatickou regulaci intenzity světel. V noci je také vhodné například nechat světla zapínat pouze na nízkou intenzitu, aby mimo jiné nedošlo k oslnění. [7]

2.3. Technologie inteligentních domů

Vytápění

Měření teploty v každé teplotě zvlášť, neznamená, že musí být nutně v každé místnosti samostatný termostat. Je to sice možné, ale stačí v místnostech pouze měřit teplotu pomocí teplotního snímače. Tento snímač může být vestavěn ve vypínači nebo tlačítkovém panelu. Zjištění a nastavování teploty můžeme pak například pomocí dotykového panelu. V rámci rodinných domů je však vhodné ovládat teploty manuálně a proto lze můžeme přepnutím tlačítka přejít z automatické regulace na manuální, toto přinese určité výhody, ale také může ohrozit úsporu energie, neboť můžeme zapomenout zapnout automatický režim. Další možností je zapínat manuální režim pouze v určité časové době. [7]

Pokud je nastaveno automatické vypínání topení při otevřeném okně, je také vhodné nastavit minimální mez teploty, kterou když se překročí, by se mělo opět zapnout topení, aby nedošlo například zamrznutí vodovodního potrubí. U podlahového topení je vhodné měřit také teplotu podlahy a nejen teplotu vzduchu. [7]

Aby bylo možné řídit topení, je nutné například u běžně používaného teplovodního topení klasickými radiátory použít termoelektrické nebo elektromotorické hlavice na místo obvyklých termostatických hlavic. K ním musí být kabelem přivedeno napájení a ovládání. [7]

Krb

Použijeme-li místo klasického krbu spalujícího dřevo krb plynový, můžeme jeho ovládání připojit na řídicí systém domu, a získat tak integrované dálkové ovládání jeho činnosti. Možné je také připojení na termostat a udržovat tak předvolenou teplotu v místnosti. Další výhodu plynové krbu je snadné zapnutí pouhým tlačítkem. Kvůli bezpečnosti lze nainstalovat detektor úniku plynu a detektor oxidu uhličitého, které mohou automaticky vypnut krb a ostatní spotřebiče. [7]

Ventilace

Instalací motorového otevírání oken. Můžeme při potřebě větrání tyto okna automaticky nebo dálkově otevírat a v případě deště také zavírat. [7]

Ke zlepšení pocitu tepelné pohody napomůže stropní ventilátor. V horkém počasí působí pohyb vzduchu ochlazujícím dojmem a v zimě může pomoct k rozložení tepla rovnoměrně v místnosti. Stropní ventilátor můžeme plynule regulovat a automaticky jej zapínat podle přítomnosti osob v místnosti. [7]

Stínící technika

Mimo ochrany osluněných oken před nežádoucími tepelnými zisky slouží stínící technika k ochraně před UV zářením. Také lze použít stínící techniku k ochraně oken před poškozením v důsledku silného větru. Motoricky lze také ovládat i závěsy, které se mohou v závislosti na budíku samy roztáhnout. Mezi stínicí techniku můžeme také zařadit speciální sklo, které podle množství přiváděného elektrické proudu dokáže měnit zabarvení. [7]

Osvětlení

Hlavní výhodou inteligentních domů oproti běžným je možnost vytváření tzv. světelné scény, tedy možnost uložení do pamětí různé konfigurace osvětlení a poté je kdykoliv později vyvolat stiskem jednoho tlačítka. Scény jsou obvykle definovány podle různých aktivit, které v dané místnosti probíhají. Scéna nemusí být omezena na jednu místnost. Můžeme různě měnit intenzity osvětlení, aby nedošlo k oslnění a další. Kvalita osvětlení ovlivňuje, jak místnost vypadá a jak se v ní cítíme, takže více menších zdrojů světla působí lépe, než jen jedno centrální světlo. Lze také použít barevné světla a vhodnou řídící jednotkou měnit jejich barva a intenzita. [7]

2.4. Bezpečnostní systém

Elektrický zabezpečovací systém

Elektrický zabezpečovací systém je dnes běžné instalován téměř v každém domě. Ke své činnosti používá řadu různých snímačů a detektorů. [7]

- Pohybové snímače – kromě běžných pro montáž na zeď existují ve stopním nebo venkovním provedení. Speciální varianty dokážou ignorovat domácí zvířata, který přidávají ke klasickému infračervenému detektoru navíc mikrovlnný detektor pro potlačení falešných poplachů. Pro ochranu otevřených dveří balkónu a teras se nabízejí snímače s rozlišením směru pohybu.

- Dveřní a okenní kontakty – pro detekci otevření
- Snímače pro detekci tříštění
- Otřesový detektor – určený pro kovové trezory, kovové dveře a další. Podobný snímač je ve speciálním elektronickém zámku.
- Protipožární detektory – reagují na vznik požáru nebo nebezpečí výbuchu.
- Detektory úniku vody, plynu, oxidu uhelnatého

Připojením elektrického zabezpečovacího systému k řídicímu systému inteligentních budov získáme možnosti lépe ochránit dům. [7]

Ovládání mechanického zabezpečení – zámky, rolety, venkovní žaluzie

Pouze elektronický zabezpečovací systém nestačí a je nutné kombinovat s mechanickými zabezpečovacími prvky. Tyto prvky mohou být elektromotoricky ovládané. [7]

Například elektronický zámek nám může automaticky uzavírat dveře a také otevírat. Tyto zámky lze ovládat dálkově. [7]

Elektronický přístupový systém

V případě použití elektromotorického zámku je nutné zajistit, která osoba bude mít přístup a která ne. Ověření můžeme provést mnoha způsoby: [7]

- Zadáním číselného kódu nebo hesla na klávesnici
- Rádiovým ovládačem
- Využitím bezkontaktní karty nebo bezdrátového či kontaktního čipu
- Využitím biometrie – například otisky prstů nebo sítnice
- Kombinace předešlých

Kamerový systém

Lze použít nejen pro ochranu proti napadení, ale také může přinést se snímači zvuku jistou bezpečnost dětí. Kamerový systém nahrává a ukládá záběry do paměti, takže jsem schopni později si záznam přehrát. Kamery existují v různých variantách a velikostech. Kamery mohou být opatřeny o dodatečný zdroj, který bude kameru napájet v případě výpadku energie. [7]

Komunikace – video vrátný, interkom, telefon, videotelefon

U vstupu můžeme mít kromě zvonku mikrofon s reproduktorem pro komunikaci s příchozími. Propojíme-li branku s telefonní ústřednou v domě, lze stiskem zvonku vyvolat zvonění telefonů. Telefonní ústředna nám také umožní sdílení více linek pro příchozí a odchozí hovory. Lze komunikovat mezi místnostmi. Funkci interkomu a telefonu můžeme mít bez telefonních přístrojů, mnoho dotykových panelů má vestavěný mikrofon a reproduktor. [7]

2.5. Ostatní

Meteorologická stanice a místní předpověď počasí

Aktuální informace o počasí se používají pro automatickou reakci domu. Meteorologická stanice dokáže kromě běžných měření, jako jsou tlaku vzduchu, teploty, vlhkosti, rychlosti a směru větru, dokáže také měřit množství a intenzitu srážek a intenzitu slunečního záření včetně ultrafialového záření. [7]

Přesný čas a budík

Na čase obvykle závidí celá řada automaticky prováděných činností v domě, proto je pohodlnější nemuset jej nastavovat, ale nechat aby se nastavoval automaticky prostřednictvím rádiového signálu nebo pomocí internetu. [7]

2.6. Ovládání

Dotykové panely

Mohou být v nejrůznějších provedení. Výhodou je vytvoření uživatelského rozhraní vhodné pro určitý dům. Dotykový panel by měl obsahovat nejčastěji používané funkce. [7]

Dálkové ovladače

Nevýhodou je, že jsem schopni ovládat zařízení pouze v rámci jedné místnosti, neboť infračervený paprsek se odráží od zdi. To můžeme zařídit implementováním snímačů do stěny, které jsou připojeny na konkrétní zařízení. [7]

Tlačítkové panely s LCD displejem nebo bez něj

Nejčastěji bývají ve zdi nebo připevněné na zeď. Obsahují tlačítka a LED diody pro potvrzení stisku. Občas mívají LCD displej pro zobrazení popisu tlačítek nebo systémové menu. [7]

Vzdálený přístup

Pro vzdálený dohled, kontrolu a ovládání domu, využíváme zejména:

- Webový prohlížeč v počítači, mobilním telefonu nebo kapesním počítači.
- Telefon nebo mobilní telefon
- Zprávy SMS, MMS
- WAP na mobilním telefonu

3. Platforma BeagleBone Black

Platforma BeagleBone Black je produktem nadace BeagleBoard.org. Nadace BeagleBoard.org je nezisková společnost se sídlem v Anglii, která poskytuje možnosti výuky, propagaci designu, použití open-source softwaru a požití hardwaru ve vestavěných systémech. Nadále provozuje fórum pro vlastníky a vývojáře open-source softwaru a hardwaru k výměně nápadů, znalostí a zkušeností. Také nabízí podporu pro další zájemce o open-source software a hardware. [3]

3.1. Produkty BeagleBoard.org

Nadace BeagleBoard.org vyrábí desky, které jsou levné, jednočipové počítače založeny na výkonných procesorech od firmy Texas Instruments s jádry série ARM Cortex-A se všemi rozšířeními dnešních osobních počítačů, a to bez nadbytečného objemu, ceny a hluku osobního počítače. Z počátku byl vývoj zaměřen na rozšíření použití Linuxových distribucí s cílem zlepšit podporu pro ARM zařízení. S rostoucím úspěchem a podporou mnoha Linuxových distribucí, byl vývoj zaměřen na umožnění zjednodušeného fyzického computingu na pokročilých GUI a síťových zařízení a to co možno nejsnadněji bez nutnosti přílišných znalostí. [3]

Mezi produkty BeagleBoard.org patří platformy:

- BeagleBoard
- BeagleBoard-xM
- BeagleBone
- BeagleBone Black

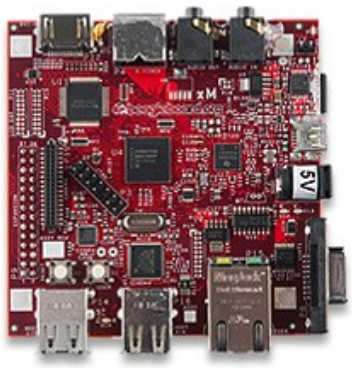
BeagleBoard

BeagleBoard je jednočipový počítač, napájený prostřednictvím USB, osazený procesorem OMAP3530 720MHz ARM Crotex-A8. Mezi rozšíření patří akcelerace NEON a VFP. Deska je opatřená rozhraními DVI-D, 2x audio výstup, SD slot, USB, S-VIDEO, které rozšíří možnosti použití platformy. [5]

Obr. 1 BeagleBoard (převzato z [5])

BeagleBoard-xM

BeagleBoard-xM je jednočipový počítač osazený procesorem AM37x 1GHz ARM. Deska obsahuje nízko výkonové pamětí typu RAM, které disponují velikosti 512MB, dále obsahuje 2D/3D grafické akcelerátory. Deska je opatřená rozhraními 4x USB port, MMC/SD konektor, DVI-D port, S-Video port, USB mini AB konektor, Ethernet, které rozšíří možnosti použití platformy. [6]



Obr. 2 BeagleBoard-xM (převzato z [6])

BeagleBone

Další z řady produktů společnosti BeagleBoard.org je BeagleBone. Jedná se o levnou a vysoce pokročilou platformu. Tato platforma je osazená o výkonné ARM procesory AM335x Cortex A8 od společnosti Texas Instruments. Platforma je podobná dřívějším BeagleBoard a může pracovat prostřednictvím rozhraní USB a Ethernet s dalšími platformy, například s dřívějšími platformy BeagleBoard a BeagleBoard-xM, nebo lze ji použít prostřednictvím dalších rozhraní ke komunikaci s dalšími moduly. BeagleBone umožňuje práci pomocí prostředí Clou9 IDE, který podporuje Node.JS nebo BoneScipt library. Deska obsahuje procesor AM335x 720MHz ARM Cortex-A8, 256MB DD2 RAM, 3D grafický akcelerátor, ARM Cortex-M3 pro řízení spotřeby, 2x PRU 32-b RISC CPU, USB klient: napájení, ladění a zařízení, USB host, Ethernet, 2x 46 pinů. [4]

Obr. 3 BeagleBone (převzato z [4])

BeagleBone Black

BeagleBone Black je nejnovější platformou řady BeagleBoard. Tato platforma je levná a vysoce pokročilá platforma řady BeagleBoard osazená levnými ARM procesory Sitara XAM3359AZCZ100 Cortex A8 od společnosti Texas Instruments. [3]

Platformy jsou vybaveny minimální sadou funkcí, které umožní uživateli využít plný potenciál procesoru a platformy. Platforma není zamýšlená jako plnohodnotnou vývojovou platformou s mnoha funkcemi. Rozhraní na desce BeagleBone Black neposkytuje přímou komunikaci s procesorem. Nejedná se o kompletní výrobek navrhnutý k plnění určité funkce. Je to základ pro experimentování a učení, jak programovat procesor a přistupovat k periferiím zařízení pro vytvoření vlastního softwaru a hardwaru. [3]

Obr. 4 BeagleBone Black (převzato z [8])

Tato platforma je podobná platformě BeagleBone. Rozdíly mezi BeagleBone Black a BeagleBone znázorňuje tabulka Tab. 1. Z tabulky je patrné, že platforma BeagleBone Black disponuje lepšími parametry jednotlivých prvků, například výkonnějším procesorem, větší velikostí paměti typu DRAM a další, také obsahuje některé rozšíření, které platforma BeagleBone neobsahuje, například výstup videa a další.

Tab. 1 Rozdíly mezi platformami BeagleBone a BeagleBone Black (převzato z [9])

	BeagleBone Black	BeagleBone
Procesor	AMR3358BZCZ100, 1GHz	AM3359ZCZ72, 720MHz
Výstup videa	HDMI	Žádný
Paměť DRAM	512MB DDR3L 800MHz	256MB DDR2 400MHz
Paměť Flash	4GB eMMC, uSD	uSD
Na desce JTAG	Volitelně	Přes USB
Seriál	Hlavička	Přes USB
PWR přídavný slot	Ne	Ano
Napětí	210-460 mA@5V	300-500 mA@5V

3.2. Technické specifikace BeagleBone Black

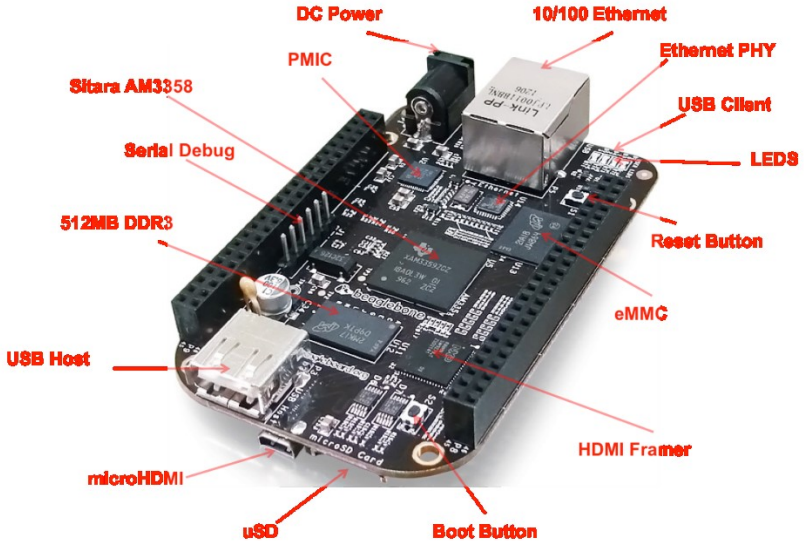
V následující tabulce Tab. 2 jsou zobrazeny hlavní technické specifikace, z tabulky se můžeme dozvědět například počty vstupů a výstupů, velikosti pamětí, jakými porty platforma disponuje a jejich použití a další informace.

Tab. 2 Technické specifikace BeagleBone Black (převzato z [9])

Technické specifikace		
Procesor	Sitara AM3358BZCZ100, 1GHz, 2000 MIPS	
Grafický engine	SGX530 3D, 20M polygons/S	
Paměť SDRAM	512MB DDR3L 800MHz	
Paměť FLASH	4GB, 8b vestavěná MMC	
PMIC	TPS65217C PMIC regulátor a přídavný LDO	
Nástroj ladění	Volitelný na desce 20 pinový CTI JTAG, Seriál Header	
Zdroj napájení	MiniUSB USB nebo DC jack	5VDC prostřednictvím vnějšího rozšiřujícího HEADER
PCB	3,4´´ x 2,1´´	6 vrstev
Indikátory	1-napětí, 2-Ethernet, 4-Uživatelské ovladatelné LED diody	
HS USB 2.0 Klient port	Přístup do USB0, klient modu skrz miniUSB	
HS USB 2.0 Host port	Přístup do USB1, typ A soket, 500mA LS/FS/HS	
Sériový port	UART0 přístup přes 6 pinů 3,3V TTL HEADER, HEADER je zabrán	
Ethernet	10/100, RJ45	
SD/MMC konektor	Mikro SD, 3,3V	
Uživatelské vstupy	Tlačítko reset, Tlačítko boot, Tlačítko power	
Video výstup	16b HDMI, 1280x1024 (MAX) 1024x768, 1280x720, 1440x900 ,1920x1080@24Hz w/EDID podpora	
Zvuk	Skrz HDMI rozhraní, Stereo	
Rozšiřující konektory	Power 5V, 3,3V, VDD_ADC (1,8V), 3,3V I/O na všechny signály McASP0, SPI1, I2C, CAN0, 4x sériový port, GPIO (69 max), LCD GPMC, MMC1, MMC2, 7x AIN (1,8V max), EHRPWM (0,2) XDMA přerušení, Tlačítko power, 4x časovač Rozšiřující desky ID (až 4 na sobě)	
Váha	39,68 gramů	

Design BeagleBone Black

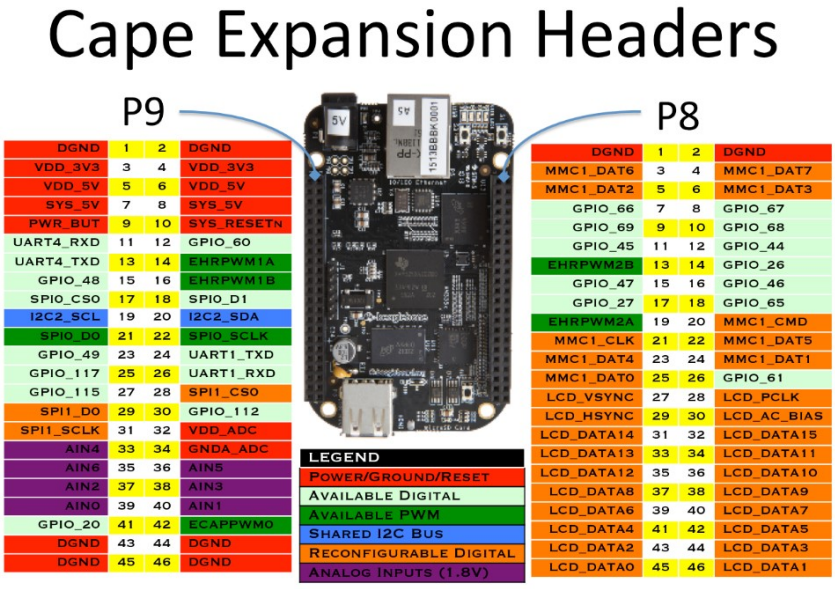
Na obrázku níže můžeme vidět, rozložení jednotlivých části BeagleBone Black. Pomocí tohoto obrázku můžeme identifikovat jednotlivé části platformy. Tento obrázek nám pomůže lépe se orientovat v rozmístěním jednotlivých portů BeagleBone Black.



Obr. 5 Rozložení jednotlivých prvků na BeagleBone Black (převzato z [12])

Piny pro rozšíření

Piny BeagleBone Black slouží pro možnosti rozšíření o další moduly (cape), nebo lze tyto piny použít jako samostatné vstupy a výstupy, popřípadě jako speciální piny pro komunikaci a další. Mezi tyto speciální piny patří například piny pro sériovou komunikaci, jako komunikace SPI, I2C, mezi další speciální piny patří analogové piny, které slouží pro vstup a výstup analogových hodnot, napájecí piny, které mohou být použity pro napájení zařízení, piny pro připojení grafického zařízení, k tomu slouží piny na GPIO a LCD. Výrobce nám u některých pinu ponechal i možnost je překonfigurovat.



Obr. 6 Popis rozšiřující pinů BeagleBone Black (převzato z [12])

3.3. BB-View Cape

BB-view cape je přenosný LCD rozšiřující modul (cape) s dotykovým displejem určený pro desky BeagleBone. Tento modul obsahuje 24 bitový LCD modul vybavený s TFT LCD (18bit) modulem. Tento modul je dostupný ve dvou velikostech: 4,3 palců a 7 palců. LCD displej velikostí 4,3 palců disponuje s rozlišením až 480x272 a displej velikostí 7 palců disponuje s rozlišením až 800x480. Obě verze displejů mají 4-vodičový odporový dotykový interface. [10]

Modul je navržen tak, aby bylo možné připojení modulu nad desku BeagleBone a po jeho připojení mít plný přístup ke všem vstupně/výstupným pinům skrze dva 46-pinové konektory, které se nachází na platformě BeagleBone a rozšiřujícím modulu BB-view. Modul je vybavený o pětici tlačítek (4 pro nastavení funkce vstupně/výstupní pinů a jedno určené pro bootování), na modulu se také nachází dvě uživatelsky modifikovatelné LED diody. Modul je napájen prostřednictvím desky BeagleBone, takže nepotřebuje vnější zdroj napájení. Modul by měl být již předkompilovaný s obrazem Linux QT demo pro snadnější a rychlejší nastavení. [10]

Obr. 7 Obrázek zapojeného BB-View převzato z [8]

Obsah sady BB-View

- Modul BB-View
- 4,3 nebo 7 palcový dotykový LCD displej
- Flexibilní FPC kabel
- Návod na rychlý start

3.4. Připojení a použití

V této části je popsáno možné připojení BeagleBone Black. Z možného druhu připojení vychází i základní použití platformy.

Připojení k počítači pomocí USB kabelu

Při připojení počítače a BeagleBone přes USB kabel, není potřeba dalších kabelů. Všechny napájení pro BeagleBone poskytuje počítač pomocí USB kabelu. V případě, že napětí poskytnuté počítačem nebude dostatečné, je možné použití vnějšího napájecího zdroje napětí. V tomto případě zapojení může být BeagleBone použit jako paměť nebo RNDIS ethernetové připojení. [11]

Samostatný počítač

BeagleBone pracuje jako samostatný počítač, bez nutnosti připojení k jinému počítači. Toto nám umožňuje vytvářet vlastní kód k ovládání desky, kdykoliv budeme potřebovat. K tomu aby mohl sloužit BeagleBone jako samostatný počítač je nutné k němu připojit vhodné periférie jako displej, klávesnici myš a také vnější napájecí napětí. [11]

4. Linux

Linux je operační systém založený na Linuxovém jádru, který se šíří v podobě distribucí. Díky své licenci se jedná o open-source software, který lze volně používat, upravovat a dále distribuovat. [1]

4.1. Historie

Linux vytvořil Linus Torvalds na univerzitě v Helsinkách s pomocí programátorů z celého světa. Linux byl odvozen operačního systému Unix, který byl navržen v roce 1970 ve firmě Bell Laboratories. [1]

4.2. Distribuce

Vlivem možnosti upravovat Linux, vzniklo již mnoho linuxových distribucí, které používají velké i malé firmy, dokonce i jednotlivci z celého světa. Linuxová distribuce je, upravený a nakonfigurovaný Linux s výběrem mnoha běžných linuxových aplikací a s pár aplikacemi navíc, které v jiných distribucí nenalezneme. [1]

4.3. Vlastnosti Linuxu

Operační systém Linux má spoustu jedinečných a výkonných funkcí. Linux je, podobně jako ostatní operační systémy, především řídícím programem pro mikropočítače. Obsahuje množinu obslužných programů a sadu nástrojů. [1]

Linux má programátorské rozhraní pro komunikaci s jádrem. Jádro lze považovat za základní modul operačního systému Linux, který je odpovědný za využívání zdrojů počítače a plánování uživatelských úloh. Programy komunikují s jádrem operačního systému prostřednictvím tzv. systémového volání. Programátor může pomocí jediného systémového volání komunikovat s různými zařízeními. [1]

4.4. Příkazový procesor

Příkazový procesor je interpret, jenž zprostředkovává rozhraní mezi uživatelem a operačním systémem. Příkazový procesor vykonává příkazy zadané prostřednictvím terminálu. V systému Linux existuje spousta příkazových procesorů. Nejpopulárnější z nich jsou: [1]

- Bourne Again Shell (bash) je vylepšenou verzí příkazového procesoru Bourne Shell, který se byl v původních verzích operačního systému UNIX.
- TC Shell (tcsh), který představuje vylepšenou verzi příkazového procesoru C Shell.
- Z Shell (zsh), který zahrnuje spoustu vlastností z ostatních příkazových procesorů.
- Public Korn Shell (pdksh), představuje podmnožinu příkazového procesoru Korn Shell.

Kromě toho může být příkazový procesor použit i jako programátorský jazyk vyšší generace. Příkazy příkazového procesoru mohou být uvedeny v souboru, jenž pak může být spuštěn jako samostatný program. [1]

Příkazový řádek

Aby, mohl příkazový procesor něco vykonávat, musíme mu v příkazovém řádku zadat nějaký příkaz. Příkazový řádek nazýváme řádek, jenž obsahuje vlastní příkaz včetně argumentů. [1]

4.5. Standartní adresáře a soubory

Struktura adresářů v operačním systému Linux je dána standardem FSSTND. [1]

Tab. 3 Některé důležité adresáře operačního systému Linux (převzato z [1])

Adresář	Účel adresáře a obsah adresáře
/	Kořenový adresář je přítomen ve všech strukturách operačního systému Linux.
/bin	Důležité spustitelné soubory. Tento adresář obsahuje soubory důležité pro zavádění operačního systému.
/boot	Statické soubory programu pro zavádění systému
/dev	Ovladače zařízení. Všechny soubory, které reprezentují periferní zařízení jako diskové řadiče, terminály a tiskárny jsou umístěny v tomto adresáři.
/etc/X11	Strojově závislá konfigurace systému X Window
/home	Domovský adresář pro uživatele.
/lib	Sdílené knihovny
/mnt	Připojovací bod pro dočasné diskové oblasti
/proc	Virtuální souborový systém s informacemi o jádrech a procesorech.
/root	Domovský adresář pro uživatele root
/sbin	Důležité systémové spustitelné soubory. Obsahuje soubory a programy potřebné v průběhu zavádění systému.
/tmp	Dočasné a pracovní soubory. Většina programů používá tento adresář pro dočasné a pracovní soubory.
/usr	Druhý důležitý strom. Tento adresář tradičně obsahuje soubory, které obsahují informace používané systémem.
/usr/bin	Většina uživatelských příkazů. Tento adresář obsahuje standartní obslužné programy operačního systému Linux, které nejsou potřebné v jednouživatelském módu.
/usr/sbin	Soubory a programy pro správu systému, které nejsou životně důležité.
/usr/src	zdrojové soubory.
/var	Proměnná data. Soubory, jejichž obsah se při běhu operačního systému mění.

4.6. Obslužné programy

Obslužné programy umožňují operačnímu systému Linux realizovat různé úlohy, například umožňují manipulovat se soubory. Obslužné programy se mohou brát jako takové příkazy Linuxu. Tyto obslužné programy jsou použity pro práci s operačním systémem Linux v příkazovém řádku neboli terminálu. Obslužné programy lze dělit do různých oblastí, dle použití.

Obslužné programy pro zobrazování soborů a manipulaci s nimi

Zde se nachází výpis jednotlivých obslužných programů určeny pro zpracování souborů, jako jsou hledání v souboru, a různé druhy zobrazení, také se zde nacházejí obslužné programy pro manipulaci se soubory, jako například kopírování souboru, přesun souboru a další.

Tab. 4 Výpis obslužných programů pro zpracování souborů a manipulaci s nimi (převzato z [1])

Obslužný program	Funkce
afio	vytvoření archivního souboru a obnova souboru z archivu
awk	vyhledání a zpracování vzoru v souboru
cat	spojování a zobrazování souborů
cmp	zjištění rozdílů mezi dvěma soubory
comm	porovnání setříděných souborů
cp	kopírování jednoho nebo více souborů
cpio	vytvoření archivního souboru a obnova souboru z archivu
cut	výběr znaků nebo položek ze vstupních řádků
dd	kopírování souboru z jednoho zařízení na druhé
diff	zobrazení rozdílu mezi dvěma soubory
find	vyhledání souborů na základě různých kritérií
fmt	jednoduché formátování souborů
grep	vyhledání regulárního výrazu v souboru
gzip	komprimace a dekomprimace souboru
head	zobrazení začátku souboru
Ispell	kontrola překlepů v souboru
less	zobrazení textových souborů po stránkách
ln	vytvoření odkazu na soubor
lpr	tisk souboru
ls	zobrazení informaci o jednom nebo více souborech
man	zobrazení dokumentace k příkazům
mkdir	vytvoření adresáře

Obslužný program	Funkce
mv	přesouvání (přejmenování) souboru
od	výpis obsahu souboru
paste	spojení odpovídajících řádků ze souboru
pr	stránkování souboru při tisku
rm	odstranění souboru (zrušení symbolického odkazu)
rmdir	odstranění adresáře
sed	editování souboru (neinteraktivního)
sor	třídění a /nebo spojování souboru
tail	zobrazení posledních řádků souboru
tar	ukládání souboru do archivu a obnovování souboru z archivu
touch	aktualizace času poslední modifikace souborů
uniq	zobrazení unikátních řádků souboru
wc	zobrazení počtu slov, řádků a znaků v souboru

Obslužné programy pro práci v síti

V této části můžeme nalézt obslužné programy pro práci v síti, jako například přenášení souborů po síti, připojování se ke vzdálenému počítači a další.

Tab. 5 Výpis obslužných programů pro práci v síti (převzato z [1])

Obslužný program	Funkce
ftp	Přenos souborů prostřednictvím počítačové sítě
rcp	Kopírování jednoho nebo více souborů ze vzdáleného počítače
rlogin	Přihlášení se ke vzdálenému počítači
rsh	Realizace příkazu na vzdáleném počítači
rwho	Zobrazení jmen uživatelů na počítačích připojených k síti
telnet	Připojení ke vzdálenému počítači prostřednictvím sítě

Obslužné programy pro komunikaci

Obslužné programy pro komunikaci jsou nejčastěji používány pro umožnění komunikace s jinými uživateli.

Tab. 6 Výpis obslužných programů pro komunikaci (převzato z [1])

Obslužný program	Funkce
mail	odesílání a přijímání zpráv elektronické pošty
mesg	umožnění / znemožnění přijímání zpráv
pine	odesílání a přijímání zpráv elektronické pošty
write	odesílání zprávy jinému uživateli

Obslužné programy, které zobrazují nebo mění stav

Obslužné programy, které zobrazují nebo mění stav, jsou do jisté míry podobné obslužným programům pro zobrazení souboru a manipulaci s nimi, neboť zde se také do jisté míry manipuluje ze soubory, například se mění přístupové práva souboru. Hlavním úkolem těchto obslužných programů je komplexnější pracování v operačním systému Linux.

Tab. 7 Výpis obslužných programů, které zobrazují nebo mění stav (převzato z [1])

Obslužný program	Funkce
cd	změna pracovního adresáře
chgrp	změna skupiny spojené se souborem
chmod	změna přístupových práv souboru
chown	změna vlastníka souboru
date	zobrazení a nastavení času a data
df	zobrazení diskového prostoru
du	zobrazení informací o používání disku
file	zobrazení klasifikace souboru
finger	zobrazení podrobných informací o uživateli
kill	ukončení procesu
nice	změna priority příkazu
nohup	spuštění programu, který poběží i po odhlášení se od systému
ps	zobrazení informací o procesech
sleep	vytvoření procesu, který bude po stanovenou dobu neaktivní
stty	zobrazení parametru terminálu

Obslužný program	Funkce
top	zobrazení stavu běžících programů
umask	nastavení masky pro vytváření souborů
W	zobrazení informací o uživatelích systému
which	zobrazení informací o umístění souboru
who	zobrazení jmen uživatelů

Programové nástroje

Tyto programové nástroje jsou především určení ke zpracovávání a vytváření programu v jazyce C.

Tab. 8 Programové nástroje (převzato z [1])

Obslužný program	Funkce
configure	automatická konfigurace zdrojového kódu programového vybavení
gcc	překladač programů v jazyku C (C++)
make	sestavování programů ze zdrojového kódu
patch	aktualizace zdrojových souborů

Obslužné programy pro správu zdrojových kódů

Obslužné programy pro správu zdrojových kódů doplňují programové nástroje, tím že umožňují různé nastavení atribut, možnosti nastavení přístupů a také možnost rekonstrukce zdrojového kódu z revize.

Tab. 9 Výpis obslužných programů pro správu zdrojových kódů (převzato z [1])

Obslužný program	Funkce
ci	vytvoření záznamu o změnách v souboru RCS
co	rekonstrukce zadané revize ze souboru RCS
cvs	správa konkurenčních přístupů ke zdrojovým souborům
rcs	vytváření a změna atributů souboru RCS
rlog	zobrazení souhrnných informací o historii souboru RCS

Další pomocné obslužné programy

V této sekci můžeme nalézt řadu dalších obslužných programů, které mají jinou funkci a nezapadají do jiných kategorií obslužných programů, které byly vypsány.

Tab. 10 Výpis dalších obslužných programů (převzato z [1])

Obslužný program	Funkce
at	realizace skriptu ve stanoveném čase
cal	zobrazení kalendáře
crontab	plánování úloh a jejich spouštění v zadaných intervalech
echo	zobrazení zprávy
expr	vyhodnocení výrazu
fsck	kontrola a oprava souborového systému
mtools	sada příkazů ve stylu MS-DOS pro manipulaci se soubory
tee	kopírování standartního vstupu na standartní výstup a do jednoho nebo více souborů
test	vyhodnocení výrazu
tr	substituce specifikovaných znaků
tty	zobrazení jména cesty k terminálu
xargs	převod standartního výstupu z jednoho programu na argumenty pro jiné program

4.7. Počítačové sítě

Ethernetová síť

Tento typ sítě bývá zpravidla používaný v lokálních počítačových sítích. Počítače komunikují v síti pomocí jednoznačně přiřazených adres, které přiřazuje síťové programové vybavení. Zpráva odesílaná počítačem, tzv. paket, obsahuje adresu cílového počítače (zrovna tak jako zpětnou adresu odesílatele). V ethernetové síti každý počítač ověřuje cílovou adresu v každém paketu odesílaném přes síť. Pokud nějaký počítač zjistí, že cílová adresa je jeho vlastní, přijme paket a vhodně zpracuje. [1]

5. Java

Java je jeden ze světově nejdůležitějších a nejrozšířenějších používaných počítačových jazyků. Java vychází z C++, který je přímým následovníkem programovacího jazyku C. Mnoho z prvků Javy implementuje prvky z těchto dvou jazyků. Z programovacího jazyku C odvozuje syntaxi. Mnoho z Java objektově orientovaných doplňků vychází z C++. [14]

5.1. Vznik Javy

Javu vymysleli James Gosling, Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan ve společnosti Sun Microsystems, Inc roku 1991. Vývoj první pracující verze trval 18 měsíců. Původní název byl Oak, následně roku 1995 byl přejmenován na Java. [14]

5.2. Souvislost s C#

Rozsah a síla Javy je nadále cítit ve světovém rozvoji počítačových jazyků. Mnoho inovačních doplňků Javy byly použity pro základ dalších programovacích jazyků. Nejdůležitějším následovníkem Javy je programovací jazyk C#. Jazyk C# byl vytvořen společností Microsoft k podpoře .NET Framework. Jazyk C# je podobný Javě, oba tyto jazyky sdílí stejnou základní syntaxi, podporují distribuované programování a využívají stejný objektový mód. I když jazyk C# byl vyvinut z Javy, stále obsahuje odlišnosti. [14]

5.3. Balík java.lang

V této kapitole se nachází tabulky třídy a interface definované v balíku java.lang. Tento balík je automaticky importovaný do všech programů. Jedná se o nejpoužívanější balík. V balíku jsou obsažené třídy pro práci ze základními daty, objekty a vlákny. [14]

Třídy obsažené v java.lang

Následující třídy popsány v tabulce Tab. 11 jsou obsaženy v balíku java.lang. Každá třída v tabulce níže, může obsahovat velkou řadu dalších metod a jejich popsání je za hranici této práce.

Tab. 11 Příklady tříd v java.lang (převzato z [14])

Boolean	Enum	Process	String
Byte	Float	ProcessBuilder	StringBuffer
Character	InheritableThreadLocal	ProcessBuilder.Redirect	StringBuilder
Character.Subset	Integer	Runtime	System
Character.unicodeBlock	Long	RuntimePermission	Thread
Class	Math	SecurityManager	ThreadGroup
ClassLoader	Number	Short	ThreadLocal
ClassValue	Object	StackTraceElement	Throwable
Compiler	Package	StrictMath	Void
Double			

Interface obsažené v java.lang

V následující tabulce Tab. 12 jsou obsažené interface v java.lang.

Tab. 12 Příklady interface v java.lang (převzato z [14])

Appendatle	Cloneable	Readable
AutoCloseable	Comparable	Runnable
CharSequence	Iterable	Thread.UncaughtExceptionHandler

5.4. Balík java.util

V této kapitole bude popsán balík java.util. Balík obsahuje velké množství tříd a interface pro rozšíření funkcionality. Balík má třídy pro generování pseudonáhodných čísel, pracuje s daty a časem. Balík obsahuje Framework pro kolekci, která standardizuje způsob, manipulace ze skupinou objektů prostřednictvím programu. [14]

Třídy obsažené v java.util

Z důvodu, že java.util obsahuje velké množství pole funkcionality, tabulka níže obsahuje pouze list části tříd. Každá třída v tabulce Tab. 13, obsahuje velkou řadu dalších metod a jejich popsání je za hranici této práce.

Tab. 13 Příklady tříd v java.util (převzato z [14])

AbstractCollection	FormattableFlags	Properties
AbstractList	Formatter	PropertyPermission
AbstractMap	GregorianCalendar	PropertyResourceBundle
AbstractQueue	HashMap	Random
AbstractSequentialList	HashSet	ResoourceBundle
AbstractSet	Hashtable	Scanner
ArrayDeque	IdentityHashMap	ServiceLoader
ArrayList	IntSummatyStatistics	SimpleTimeZone
Array	LinkedHashMap	Spliterators
Base64	LinkedHashSet	SplitableRandom
BitSet	LinkedList	Stack
Calendar	ListResourceBundle	StringJoiner
Collections	Locale	StringTokenizer
Currency	LongSummaryStatistics	Timer
Date	Objects	TimerTask

Pokračování Tab. 13.

Directionary	Observable	TimeZone
DoubleSummaryStatistics	Optinal	TreeMap
EnumMap	OptionalDouble	TreeSet
EnumSet	OptionalInt	UUID
EventListenerProxy	OptinalLong	Vector
EventObject	PriorityQueue	WeakHashMap

Interface obsažené v java.util

V následující tabulce Tab. 14 se nachází ukázka obsažených interface v balíku java.util.

Tab. 14 Příklady interface v java.util (převzato z [14])

Collection	Map.Entry	Set
Comparator	NavigableMap	SortedMap
Deque	NavigableSet	SortedSet
Enumeration	Observer	Spliterator
EventListener	PrimitiveIterator	Spliterator.OfDouble
Formatttable	PrimitiveIterator.OfDouble	Spliterator.OfInt
Iterator	PrimitiveIterator.OfInt	Spliterator.OfLong
List	PrimitiveIterator.OfLong	Spliterator.OfPrimitive
ListIterator	Queue	
Map	RandomAccess	

5.5. Balík java.io

Balík java.io poskytuje podporu pro I/O (vstupně/výstupní) operace, reprezentované jako Java I/O systém, zahrnující základní techniky pro čtení a zápis souborů. Řešení I/O exception a zavíraní souboru. Díky tomuto balíku lze například manipulovat se síťovým připojením, paměťovým bufferem nebo se soubory na disku. I když jsou tyto zařízení fyzicky odlišná, umožňují tyto zařízení manipulaci se stejným abstraktem stream. [14]

Třídy obsažené v java.io

Tabulce Tab. 15 jsou obsažené některé ze tříd nacházejících se v balíku java.io, tyto funkce slouží pro vstup a výstup, jako je čtení nebo zápis souboru.

Tab. 15 Příklady tříd v java.io (převzato z [14])

BufferedinputStream	FileWriter	PipedOutputStream
BufferedOutputStram	FilterInputStream	PipedReader
BufferedReader	FilterOutputStream	PipedWriter
BufferedWriter	FilterReader	PrintStream
ByteArrayInputStream	FilterWriter	PrintWriter
ByteArrayOutputStream	InputStream	PushbackInputStream
CharArrayReader	InputStreamReader	PushbackReader
CharArrayWriter	LineNumberReader	RandomAccesFile
Console	ObjectsInputStream	Reader
DataInputStream	ObjectsInputStream.GetField	SequenceInputStream
DataOutputStream	ObjectOutputSteam	SerializablePermission
File	ObjectSteamClass	StringTokenizer
FileDesciptor	ObjectStreamClass	StringReader
FileInputStream	ObjectStreamField	StringWriter
FileOutputStream	ObjectStream	Writer
FilePermission	OutputStreamWriter	
FileReader	PipedInputStream	

Interface obsažené v java.io

V následující tabulce Tab. 16 se nachází ukázka obsažených interface v balíku java.io.

Tab. 16 Příklady interface v java.io (převzato [14])

Closeable	FileFilter	ObjectInputValidation
DataInput	FilenameFilter	ObjectOutput
DataOutput	Flushable	ObjectStreamConstatns
Externalizable	ObjectInput	Serializable

5.6. Balík java.nio

Začátkem verze 1.4, Java poskytuje druhý I/O systém jménem NIO první je typický I/O systém. Toto podporuje kanálově-založený přístup k I/O operacím. [14]

5.7. Balík java.net

Balík java.net slouží pro možnost Java programu komunikovat prostřednictvím internetu. Síťová komunikace v Javě podporuje koncept soketů. Soket je definován jako koncový bod v síti. Soket komunikace pracuje dle určeného protokolu. [14]

Internetový protokol IP je nejnižším propojovacím protokolem, který rozdělí data na malé pakety a pošle je na adresu přes síť, není zaručeno, že příjemce zpráv, zprávu dostane. Transmission Control Protocol (TCP) je nejvyšší propojovací protokol. User Datagram Protocol (UDP), leží vedle TCP a může být použit přímo pro podporu rychlého posílání zpráv. [14]

Třídy obsažené v java.net

Java podporuje TCP/IP tak, že rozšíří stream I/O interface. Java podporuje protokoly TCP a UDP. Protokol TCP je použit pro spolehlivý přenos přes síť. Protokol UDP podporuje snadnější, tedy rychlejší, point-to-point datagram orientovaný model. [14]

Tab. 17 Příklady tříd v java.net (převzato z [14])

Authenticator	InetAddress	SocketAddress
CatcheRequest	InetSocketAddress	SocketImpl
Catcheresponse	InterfaceAddress	SocketPermission
ContentHandler	JarURLConnection	StandardSocketOption
CookieHandler	MulticastSocket	URI
CookieManager	NetPermission	URL
DatagramPacket	NetworkInterface	URLClassLoader
DatagramSocket	PasswordAuthentication	URLConnection
DatagramSocketImpl	Proxy	URLDecoder
HttpCookie	ProxySelector	URLEncoder
HttpURLConnection	ResponseCache	URLPremission
IDN	SecureCacheResponse	URLStreamHandler
Inet4Address	ServerSocket	
Inet6Address	Socket	

Interface obsažené v java.net

V následující tabulce Tab. 18 se nachází ukázka obsažených interface v balíku java.net.

Tab. 18 Příklady interface v java.net (převzato z [14])

ContentHandlerFactory	FileNameMap	SocketOptions
CookiePolicy	Protocolfamily	URLStreamHandlerFactory
CookieStore	SocketImplFactory	
DatagramSocketImplFactory	SocketOption	

5.8. Popis applet

Balík java.applet poskytuje detailnější kontrolu nad spouštěním appletu. Applety se dělí na dva typy. Prvním typem je založen přímo na Applet třídě. Tento typ používá Abstract Window Toolkit (AWT) pro poskytnutí grafického uživatelského rozhraní. Druhý typ apletů je založený na Swing třídě JApplet, která dědí třídu Applet. Swing applety poskytují Swing třídy pro poskytování GUI. Swing nabízí bohatší a často i lehčí použití uživatelského rozhraní než AWG. Součástí tříd Applet a JApplet jsou metody pro tvorbu a správu uživatelského grafického rozhraní. [14]

5.9. Popis Event Handling

Řešení událostí je základ pro Java programování, protože se jedná o nedílnou součást řady aplikací, počítaje s appletem až po ostatní typy graficky založených aplikací. Napsání grafické aplikace není prakticky možné bez příkazu pro řešení událostí. Události jsou podporovány řadou balíků, počínaje java.util, java.awt a java.awt.event. Většina událostí jsou generovány, když uživatel interaguje s grafickým uživatelským interface. Existuje mnoho tříd, určené pro řešení událostí. Vyvolat událost dané třídy, mohou vyvolat akce na prvky obsažené na Appletu, například stisk tlačítka, nebo akce na myši a další. Ke každé třídě, která řeší událost, existuje interface, ve kterém jsou obsaženy metody na řešení událostí. [14]

5.10. Popis AWT

Abstract Window Toolkit (AWT) je první GUI Framework a byl součástí Javy od první verze. Obsahuje řadu tříd a metod, které umožňují vytvářet okna a jednoduše je kontrovat. Je důležité uvést, že jen zřídka se vytváří GUI založené výhradně na AWT, protože existují silnější GUI frameworky, jako jsou Swing a JavaFX, které byly vytvořeny pro Javu. I přes tento fakt je AWT důležitou částí Javy. [14]

Frameworky nejčastěji používají Swing. Protože Swing poskytuje bohatší a více flexibilnější GUI Framework než má AWT. Ale pro lepší pochopení Swing je důležité správné pochopení AWT. Nejnovějším frameworkem, který Java poskytuje je JavaFx. [14]

Součástí AWT jsou různé druhy kontrolu. Tyto kontroly jsou například popisky, tlačítka, různé políčka, seznamy a různé úpravy textu. [14]

5.11. Obrázky

Součástí balíku java.awt.image jsou třídy provázející podporu pro práci ze soubory, jako jsou zobrazování a manipulaci s grafickými obrázky. Obrázky jsou klíčovou komponentou pro webový návrh. [14]

Obrázky jsou objekty třídy Image, která je částí balíku java.awt. Obrázky jsou manipulovány prostřednictvím tříd nacházejících se v java.awt.image. Součástí tohoto balíku je řada tříd a interface určený pro práci s obrázky. Z toho důvodu zobrazím pouze část tříd a interface určených pro práci s obrázky. [14]

5.12. Nástroje pro souběžnost

Ze začátku Java poskytovala podporu pro vytváření více vláknové aplikace a synchronizaci. Například nové vlákno může být vytvořeno implementací Runnable a nebo rozšířením o Thread. Synchronizace je možná použitím klíčového slova synchronized a mezi vláknovou komunikací podporováno metodami wait() a notify(). Souběžnost je součástí balíku java.util.concurrent a dvou pod balíku java.util.concurrent.atomic a java.util.concurrent.locks. Třídy pro synchronizaci jsou definování v java.util.concurrent. [14]

Tab. 19 Příklady třídy pro synchronizaci vláken (převzato z [14])

Semaphore	Implementuje klasický semafor
CountDownLatch	Čeká, dokud nenastane specifické číslo událostí
CyclicBarrier	Umožňuje skupině vláken čekat na předdefinovaným bodem.
Exchanger	Vyměňuje data mezi dvěma vlákny
Phaser	Synchronizuje vlákna, tak vytváří řadu fází v operacích.

6. Návrh zabezpečovacího terminálu

Zabezpečovací terminál bude obsahovat možnost přihlášení pro dva uživatele. V zabezpečovacím terminálu budou obsaženy dva různé okruhy, ve kterých budou obsažené prvky, které budou moct být uzamčeny popřípadě odemčeny. Možnost odemčení nebo uzamčení budou mít oba dva uživatelé. Stav prvků bude signalizovaný v zabezpečovacím terminálu, prostřednictvím signalizačních prvků. Stisk tlačítka vyvolá odeslání zprávy s žádosti ke změně stavu prvku. Na tuto změnu odpoví zařízení odesláním zprávy zpět do terminálu. Tato zpráva způsobí změnu signalizačního prvku příslušného objektu. Z důvodu možnosti změny stavu prvky i prostřednictvím zařízení, bude muset zabezpečovací terminál být schopen získat tuto změnu.

Návrh zabezpečovacího terminálu je rozdělen na dvě hlavní části. První část se zabývá zprovozněním platformy BeagleBone Black. A druhá část se zabývá vytvořením řídícího programu pro řízení funkce zabezpečovacího terminálu. Jednotlivé části návrhu budou popsány následujících částech této kapitoly a pro zvýšení přehlednosti budou rozděleny na další částí.

6.1. Návrh platformy

První krok návrhu je zprovoznění platformy BeagleBone Black, tím se myslí doplnění platformy o příslušné programy a nástroje, které umožní jeho následnou práci jako zabezpečovací terminál.

Platforma bude schopná komunikace s počítačem. Na platformu bude nainstalován nový operační systém, který bude zaváděn z vnitřní paměti platformy. Následně bude platforma schopná pracovat společně s rozšiřujícím modulem BB-view. Bude možné spouštět Java aplikací na platformě a také spouštět grafické Java aplikace na rozšiřujícím modulu platformy.

6.2. Návrh řídícího programu

Druhý krok návrhu je vytvoření řídícího programu pro zabezpečovací terminál. Řídící program bude vytvořen pomocí programovacího jazyku Java a k jeho vývoji bude použito vývojové prostředí NetBeans IDE verze 8.0.2. Řídící program se bude skládat ze dvou hlavních funkčních částí. První částí řídícího programu bude přihlašovací systém a druhou částí řídícího programu bude komunikace prostřednictvím UDP paketů v síti Ethernet.

Přihlašovací část

Přihlašovací systém, bude obsahovat pole pro vkládání číslic a bude opatřen o klávesnice určenou pro vkládání číselného kódu do pole. Klávesnice bude z důvodu omezené velikosti displeje obsažena na panelu určeném pro přihlášení společně s textovým polem. Klávesnice bude obsahovat tlačítka pro vložení číselného kódu, tyto tlačítka budou označeny čísly 0 až 9 a jejich stisk vyvolá akci, která připíše hodnotu stisknutého tlačítka do pole pro vložení kódu. Součástí klávesnice budou také tlačítka pro vstup a mazaní hodnoty pole. Posledním prvkem přihlašovací částí bude popisek, který bude zobrazovat hlášky o stavu kódu. K přihlášení může dojít pouze při zadání správného kódu. Uživatelské kódy budou uložené v souboru. Při spuštění musí přihlašovací část vždy zkontrolovat, zda existuje tento soubor a případně tento soubor vytvořit. Uživatelské přihlašovací údaje mají tvar *„JmenoUživatele_KÓD*.

Komunikace

Zabezpečovací terminál bude schopen komunikovat prostřednictvím UDP paketů po síti Ethernet. Terminál bude schopen posílat žádosti o uzamčení popřípadě odemčení objektu. Toto se vždy provede po stisku příslušného tlačítka daného objektu, který chceme změnit. Zabezpečovací terminál následně bude umět přijímat UDP pakety, při příchodu paketu dojde k vyvolání akce, která se projeví změnou signalizačního prvku daného objektu, který se změní. Následně bude zabezpečovací terminál posílat zprávu na dotazování stavu vždy v pravidelném intervalu.

7. Realizace

V této části se budeme zabývat potřebnými úkony v rámci realizace zprovoznění platformy BeagleBone Black a následně i popisem řídícího kódu. Popsané úkony byly potřebné a použité pro samotnou realizaci, vše v souladu s návrhem popsaným v kapitole návrh zabezpečovacího terminálu.

7.1. Zprovoznění platformy BeagleBone Black

Pro zprovoznění platformy BeagleBone Black je nutné provést několik důležitých úkonů, jako jsou zprovoznění komunikace mezi platformou a počítačem, nainstalování nového image operačního systému, nainstalování Javy na platformě a nainstalování driverů pro rozšiřující modul. V této části jsou i popsány několik dalších úkonů pro správnou funkci platformy. Ke všem těmto postupům je nutné znát Linuxové příkazy a také mít terminál pro komunikaci přes síť. Použité terminály byly terminál Putty a terminál Psftp.

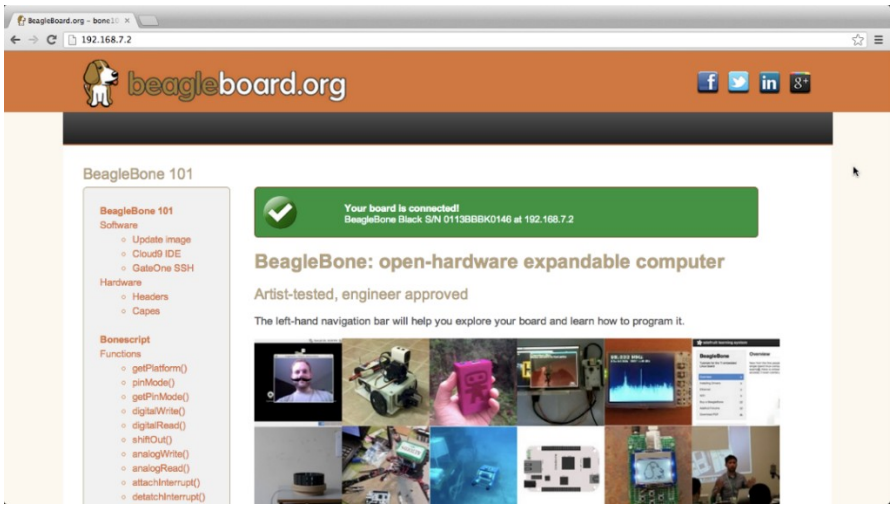
Instalace platformy

Prvním krokem zprovoznění platformy BeagleBone je připojení platformy s počítačem a zprovoznění komunikace mezi platformou a počítačem, neboť bez této úpravy není možné s platformou vůbec pracovat. Platforma a počítač se propojí prostřednictvím USB kabelu, tímto zajistíme napájení platformy. Po připojení se spustí platforma tím, že se zavede operační systém Linux z paměti eMMC. Systém může být zaveden také z mikro SD karty. V paměti platformy BeagleBone Black se nachází základní dokumentace a stažené ovladače.

Ovládače pro počítač můžeme nalézt buď na internetových stránkách výrobce nebo přímo v paměti BeagleBone, tyto ovladače slouží pro vytvoření sítě přes USB. Na stránkách od výrobce můžeme také nalézt i další ovladače například pro informace o virtuální ethernetové síti pomocí USB, nebo rozhraní USB a sériového portu. [15]

Dokončení instalace BeagleBone Black

Pokud jsme provedli potřebnou instalaci platformy, propojení a instalaci ovladačů, již stačí otestovat, zda tato instalace proběhla správně. Toto zjistíme tak, že propojíme platformu a počítač prostřednictvím USB kabelu a v internetovém prohlížeči Chrome nebo Firefox zadáme adresu 192.168.7.2, která odpovídá adrese platformy. Pokud používáme připojení prostřednictvím sítě Ethernet tak zadáme do internetového prohlížeče odpovídající adresu platformy. [15]



Obr. 8 Ukázka stránky po úspěšné instalaci (převzato z [15])

Zobrazí se nám stránka, na kterém nám stav správně nainstalované platformy udává zelené popisové pole. Na této stránce můžeme nalézt různé návody, rady a nápady jak používat platformu, jako jsou parametry jednotlivých platforem, můžeme zde nalézt také fórum, kde uživatelé přidávání rady a návody pro různé použití platformy. Také se zde nachází ukázka funkcí napsaných v BoneScriptu. Jedná se o knihovnu speciálně vytvořených funkcí určených pro práci na platformách od společnosti BeagleBoard.org, pro práci s těmito funkcemi můžeme na této stránce, také nalézt také vývojové prostředí Cloud9 IDE. Nachází zde také odkazy na různé Linuxové distribuce, tzv. image a mnoho dalších užitečných informací.

7.2. Nový operačního systému

Jakmile máme propojený počítač a platformu, přistoupíme k přehrání paměti o nový image, neboť je vhodnější, když je platforma doplněná o novější verzi operačního systému. Pro instalaci nového image je nutné nejprve stáhnout jednu z možných image určených pro platformu. Můžeme nalézt různé druhy image, mezi nejrozšířenější patří image Linuxových distribucí, ale můžeme nainstalovat na platformu i jiné image, například od společnosti Windows nebo android. Stažení nových image můžeme provést ze stránek výrobce, pro realizaci práce jsem zvolil jednu z Linuxových distribucí jménem Debian, jednalo se o tento image Debian 7.8 (BeagleBone, BeagleBone Black - 4GB SD) 2015-03-01.

Programy pro nahrání nového operačního systému na mikro SD kartu

Abychom byli schopni naistalovat nový image na platformu je nutné provést několik úkonů a je také zapotřebí několika programů, například je zde zapotřebí programu 7-zip, pomocí kterého se následně provede dekomprimace staženého souboru do souboru s příponou .img. Dalším potřebným programem, který bude potřebný je Image Writer, tento program umožní nahrání souboru s příponou .img na SD kartu.

Zavedení nového operačního systému z mikro SD kartu

K nahrání nového image musíme připojit SD kartu do počítače, k tomu je nutné použít adapteru na mikro SD kartu. Dalším krokem je stažení příslušného image a jeho dekomprese prostřednictvím programu 7-zip, toto vytvoří soubor, který bude mít příponu .img. Následně se tento soubor nahraje na SD kartu, k tomu použijeme program Image Writer. Jakmile máme již příslušný soubor nahraný, můžeme vysunout SD kartu z počítače a vytáhnout mikro SD kartu z adapteru. Mikro SD kartu následně vložíme do platformy. Po vložení mikro SD karty můžeme spustit platformu, buď připojením vnějšího napájecího zdroje anebo můžeme napájet platformu prostřednictvím USB kabelu připojeného v počítači. Před zapojením napětí, podržíme tlačítko User, nacházející se na platformě. Tyto kroky nám způsobí, zavedení nového image z mikro SD karty. [15]

Zavedení nového operačního systému z eMMC

K zavedení nového image můžeme použít přímo interní paměť eMMC místo zavádění prostřednictvím mikro SD karty. Abychom byli schopni toto provést, musíme tento image nahrát do příslušné paměti. K nahrání do paměti musíme upravit obsah souboru /boot/uEnv.txt. K tomu použijeme terminál Putty, kterým se připojíme z počítače na platformu, to nám umožní ovládání platformy prostřednictvím linuxových obslužných příkazů.

Připojení se provede tak, že vyplníme obsah textové pole Host Name adresou platformy BeagleBone Black tedy buď adresou 192.168.7.2, pokud používáme připojení přes USB kabel anebo příslušnou adresou platformy při komunikaci po síti. Po otevření spojení nás požádá platforma BeagleBone Black o přihlášení pod uživatelským účtem. Pro nahrání a instalaci image do paměti musíme zvolit uživatelský účet jménem root, tento účet patří administrátorovi a neobsahuje žádné přihlašovací heslo.

Po přihlášení můžeme přejít k úpravě textu v souboru /boot/uEnv.txt. Pro úpravu textových souborů obsahuje Linux dva primární editační programy vi a nano. Pro úpravu tohoto souboru zvolíme editačního programu nano. [15]

nano /boot/uEnv.txt

Pokud nastane problém s přístupem a oprávněním čtení a zápisu tohoto souboru musíme tedy vhodně upravit toto oprávnění. K úpravě nastavení oprávnění poskytuje Linux obslužný program chmod. Díky tomuto příkazu a vhodně zvoleným parametrům jsme schopni nastavit oprávnění souboru. Hodnotu parametrů jsem zvolil na 777, díky toho budu mít oprávnění provádět z daným souborem jakékoliv úpravy.

chmod 777 /boot/uEnv.txt

V souboru /boot/uEnv.txt můžeme nalézt následující řádky.

```
##enable BBB: eMMC Flasher:
#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Obr. 9 Text v souboru uEnv.txt před změnou

Pro možnost nahrání a instalace image do paměti eMMC musíme upravit druhý řádek a to tím, že odstraníme ze začátku druhého řádku znak mřížky. Tímto umožníme, aby došlo k nahrání a instalaci nového image z mikro SD karty nebo jiného zdroje na paměť.

```
##enable BBB: eMMC Flasher:
cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Obr. 10 Text v souboru uEnv.txt po změně

Po úpravě obsahu dokumentů, provedeme uložení tohoto dokumentu a následně provedeme reboot platformy. Pokud jsme provedli předchozí úpravy správně po spuštění platformy, dojde k začátku instalace nového image, který se nacházel na mikro SD kartě na paměť eMMC. Instalace nového image potrvá asi 40 minut, během instalace dochází k blikání uživatelských LED diod umístěných na platformě. Dokončení instalace indikuje rozsvícení uživatelských LED diod. Následně se odpojí platforma od zdroje napájení a vytáhne se mikro SD karta z BeagleBone Black, aby nedošlo k tomu, že po připojení napájení začne nová instalace image do paměti. Pokud již jsme vytáhli mikro SD kartu, připojíme zdroj napájení. Po spuštění, se platforma BeagleBone Black spustí pod novým imagem.

7.3. Instalace Java

Aby byla platforma možná pracovat a spouštět Java aplikace, je nutné nainstalovat na platformu příslušný doplněk Java. Pro instalaci jsem zvolil verzi Linux ARM v6/v7 Hard Float ABI, která je určena pro procesory typu ARM a operační systém Linux. Pro instalaci existuje několik způsobů. Pro nahrání a instalaci doplňku Java na platformu jsem zvolil postup při, kterém se provede stažení příslušného souboru a pomocí terminálu Psftp se provede přenos tohoto souboru na platformu a následně pomocí terminálu Putty se provede instalace daného souboru na tuto platformu.

Přenesení instalačního souboru na BeagleBone Black

K přenosu příslušného instalačního souboru se použije terminál Psftp s následujícím postupem. Pomocí příkazu open se otevře spojení mezi počítačem a platformou. K otevření spojení musíme k příkazu open dodat adresu zařízení, se kterým toto spojení chceme uskutečnit. Pokud je zařízení platforma BeagleBone Black bude tato adresa 192.168.7.2.. Po otevření spojení budeme požádání o přihlášení se pod uživatelským účtem. Po instalaci image Debian, máme možnost přihlášení pod dvěma defaultními účty, prvním z účtů je administrátorský účet root a druhý z účtů je běžný účet jménem debian, který osahuje heslo temppwd.

```
psftp: no hostname specified; use "open host.name" to connect
psftp> open 192.168.7.2
login as: root
Debian GNU/Linux 7

BeagleBoard.org Debian Image 2015-03-01

Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:temppwd]

Remote working directory is /root
psftp>
```

Obr. 11 Ukázka z terminálu Psftp po přihlášení

Po přihlášení provedeme zadání příkazu lcd s cestou, ve které se nachází stažený instalační soubor doplňku Java. Tento příkaz provede změnu aktuálního adresáře za adresář, ve kterém se nachází instalační soubor.

```
psftp> lcd "C:\Users\Public\Downloads"
New local directory is C:\Users\Public\Downloads
psftp> _
```

Obr. 12 Ukázka z terminálu Psftp pro změnu adresáře

Jakmile se nacházíme v adresáři s instalačním souborem, můžeme provést přenesení instalačního souboru na platformu. K tomu slouží příkaz mput s parametrem. Parametrem příkazu je jméno souboru, který chceme přenést. Přenášení instalačního souboru trvá v závislosti na velikosti souboru několik vteřin.

```
psftp> mput jdk*
local:jdk-8u60-linux-arm32-vfp-hflt.gz => remote:/root/jdk-8u60-linux-arm32-vfp-
hflt.gz
psftp>
```

Obr. 13 Ukázka z terminálu Psftp pro nahrání souboru na BeagleBone Black

Pokud již přenos instalačního souboru skončil, můžeme zavřít teminál Psftp. Následně musíme spustit terminál Putty, ve kterém provedeme instalaci doplňku Java.

Instalace Java pomocí terminálu Putty

Po spuštění terminálu Putty a následného připojení na platformu, musíme zadat uživatelský účet, na kterém se přenesl instalační soubor. Instalační soubor by se měl nacházet přímo na pomyslné ploše platformy, tedy v adresáři home. Pro kontrolu zda se nachází instalační soubor v příslušném adresáři provedeme příkaz ls popřípadě ls –al, který zobrazí všechny soubory a adresáře, které se nachází v adresáři.

```
root@beaglebone:~# ls
jdk-8u60-linux-arm32-vfp-hflt.gz
```

Obr. 14 Ukázka z terminálu PuTTy po nahrání souboru

Jakmile víme, že instalační soubor je obsažený v adresáři, vytvoříme nový adresář, do kterého následně přeneseme instalační soubor. Vytvořený adresář má jméno java a byl vytvořen v adresáři usr.

mkdir /usr/java/

Přenos instalačního souboru umožní příkaz mv s parametrem. Parametrem tohoto příkazu je jméno souboru a cesty k cílovému adresáři, do kterého chceme instalační soubor přenést.

mv jdk-8u60-linux-arm32-vfp-hflt.gz /usr/java

Po provedení přenosu následuje extrahování instalačního souboru v adresáři. Tato operace je časově náročná a závisí na velikosti souboru.

tar xzf jdk-8u60-linux-arm32-hflt.gz

Po dokončení extrahování souboru musíme vytvořit cestu k extrahovaným adresářům, díky následujícím dvěma příkazům. Pro zapsání těchto příkazů již bude možné spouštět Java aplikace.

export PATH=$PATH:/usr/java/jdk1.8.0_60/bin

export JAVA_HOME=/usr/java/jdk1.8.0_60/

Po rebootování platformy dochází k tomu, že opět nelze spouštět Java aplikace. Toto lze opravit tím, že opakujeme zápis předchozích dvou příkazů pro vytvoření cesty k extrahovaným adresářům. Toto řešení není moc praktické a z tohoto důvodu jsem zapsal předchozí příkazy na vytváření cesty do souboru .profile, který se čte vždy při startu operačního systému. Pro úpravu tohoto souboru použiji editačního programu nano. Po zapsání příkazů do souboru, již nedochází k problému se spouštěním Java aplikací po rebootu nebo po spuštění.

nano .profile

7.4. Instalace Element14 BB View_43 4.3inch LCD display cape

Dalším částí zprovoznění je nainstalování ovladačů pro rozšiřující modul BB-view. Ovladače pro rozšiřující modul můžeme nález na stránkách výrobce nebo prodejce platformy. Stažení ovladačů bylo provedeno ze stránky prodejce. Pro instalaci těchto ovladačů se musí nahrát tyto soubory na platformu, toto nahrání lze provést několika různými způsoby.

Postup instalace ovladačů byl následující. Prvně se provedlo nahrání souborů na flash disk. Po nahrání ovladačů na flash disk vytáhne se flash disk z počítače a vloží se do platformy. Po vložení flash disku do platformy spustíme terminál Putty a zde provedeme několik příkazů, které provedou nahrání souboru z flash disku na platformu a následně jejích instalaci. [16]

Po připojení flash disku a otevření komunikace prostřednictvím terminálu Putty. Přistoupíme k operaci, která vytvoří na platformě adresář udisk, který se bude nacházet v adresáři media.

mkdir /media/udisk

Jakmile vytvoříme příslušný adresář, použijeme příkazu mount, který umožní platformě pracovat s flash diskem, následně provede kopírování obsahu flash disku do vytvořeného adresáře.

mount /dev/sda1 /media/udisk

Po připojení flash disku, zkopíruji soubory ovladače, nacházející se na flash disku do systémových adresářů na platformě.

cp –f /media/udisk/zImage /boot/vmlinuz-3.8.13-bone70

cp –f /media/udisk/.dtb /boot/dtbs/3.8.13-bone70*

Jakmile jsou tyto soubory převedené, přistoupíme k extrahování souboru v adresáři.

tar –xvf /media/udisk/kernel_modules.tar.gz –C /

Následně jsou extrahované soubory kopírovány do systémového adresáře, ve kterých se nachází informace o displeji.

cp –f /media/udisk/xorg.conf /etc/X11/

Pro dokončení kopírování souborů provedeme příkazu sync. Tím se vyprázdní zásobník diskové vyrovnávací paměti.

sync

Jakmile jsou již soubory zkopírované v systémových adresáři, musíme ještě provést náhradu souborů určených pro BB-view za již existující v paměti platformy. Tyto příkazy jsou určeny pro nainstalování ovladačů pro 4-palcový typ displeje.

Nejprve se změní adresář, ve kterém se nacházím na jeden ze systémových adresářů.

cd /boot/dtbs/3.8.13-bone70

V tomto adresáři přepíšeme soubor, který se zde nachází za soubor, který jsme zde kopírovali předchozími příkazy.

cp am335x-boneblack-lcd4.dtb am335x-boneblack.dtb

Opět použijeme příkazu sync a tím ukončíme instalaci dipleje.

sync

Jakmile provedeme tyto příkazy, musíme platformu rebootovat a pokud byly provedené akce správně provedeny, mělo by dojít ke správné funkci rozšiřujícího modulu.

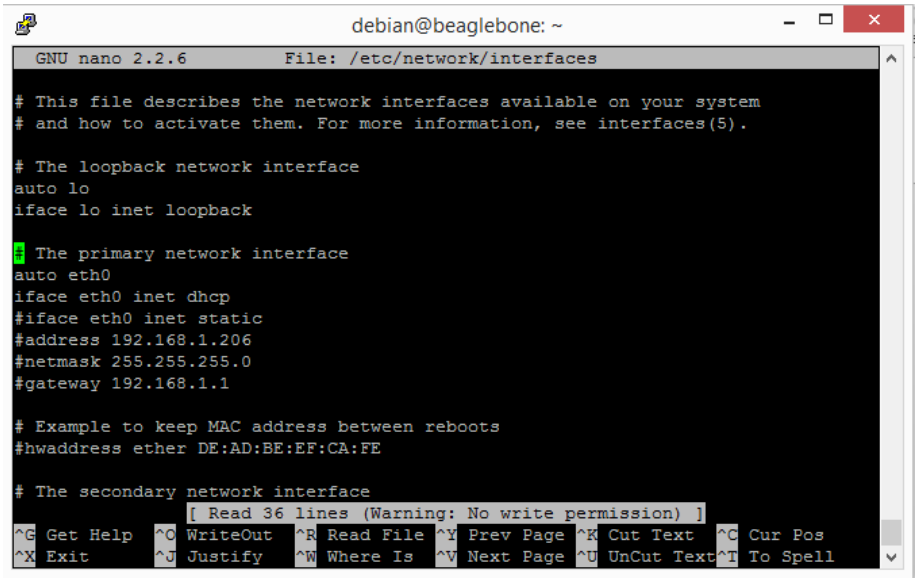
7.5. Zprovoznění komunikace

Po instalaci rozšiřujícího modulu, tedy displeje dojde k tomu, že nelze použít Ethernetového portu ke komunikaci. Proto je nutné jej znovu zprovoznit, neboť součástí této práce je nutné, aby platforma byla schopná komunikovat po síti Ethernet s dalším zařízením. Ke znovu zprovoznění komunikace prostřednictvím sítě Ethernet není potřeba stahování nebo instalování dalších souborů, stačí pouze pozměnit obsah textu v souboru interfaces nacházejícím se na /etc/network/interfaces.

K tomu opět použijeme terminálu PuTTy a připojíme se na BeagleBone Black. Po přihlášení, použijeme příkazu nano k přepisu obsahu souboru.

nano /etc/network/interfaces

V tomto souboru musíme nalézt řádky, které obsahují #auto eth0 a #iface eth0 inet dhcp, které se nachází u The primary network interface. U těchto dvou řádků musíme smazat znaky mřížky a uložit soubor. Pro úpravu musíme být přihlášeny pod uživatelem root, jinak nebudeme mít požadované oprávnění provést tyto úpravy, ale pouze jen číst tento soubor. Po této úpravě bude již platforma schopná komunikovat po síti Ethernet a také bude ji automaticky přidělována adresa z DHCP serveru. V tomto souboru lze také nastavit statickou IP adresu. Při nastavení statické adresy musíme mít na paměti, aby byla v síti jedinečnou adresou, aby nedocházelo ke kolizi adres v síti.



Obr. 15 Stav souboru po provedení úprav pro zprovoznění komunikace

7.6. Spouštění grafické aplikace na rozšiřujícím modulu BB-view

Z důvodu toho, že aplikace má běžet a pracovat s dotykovým displejem, tedy s rozšiřujícím modulem BB-view, musí se prostřednictvím terminálu Putty zapsat příkaz, pro spouštění grafických aplikací na displeji. Kdyby k tomu nedošlo, tak by platforma nevěděla, že se aplikace má spouštět na displeji a zkoušela by spustit aplikaci jinak. Toto z důvodu, že aplikace obsahuje prvky Java Jframe a form, které jsou grafickými, způsobí to chybu a aplikaci nebudeme možné spustit.

export DISPLAY=:0

Z důvodu toho abychom nemuseli zadávat tento příkaz při každé spuštění zapíšeme jej do inicializačního souboru jménem .profile systému Debian. Zapsání se prování prostřednictvím jednoho z příkazů pro editaci souborů vi nebo nano.

nano .profile

7.7. Problémy s logy

Platforma BeagleBone Black má problémy s řízením logů, tedy nedochází k mazání starých logů, ale pouze k ukládání nových, z tohoto důvodů dochází k tomu, že paměť platformy se přeplní daty z logů a to vede k tomu, že operační systém nenaběhne správně a na rozšiřujícím modulu se zobrazí pouze úvodní ikona a následně zůstane modul prázdný.

Řešením je ruční mazání těchto logů. K tomu budeme opět potřebovat použít terminál Putty.

cat /dev/null > /var/log/messages

cat /dev/null > /var/log/kern.log

cat /dev/null > /var/log/syslog

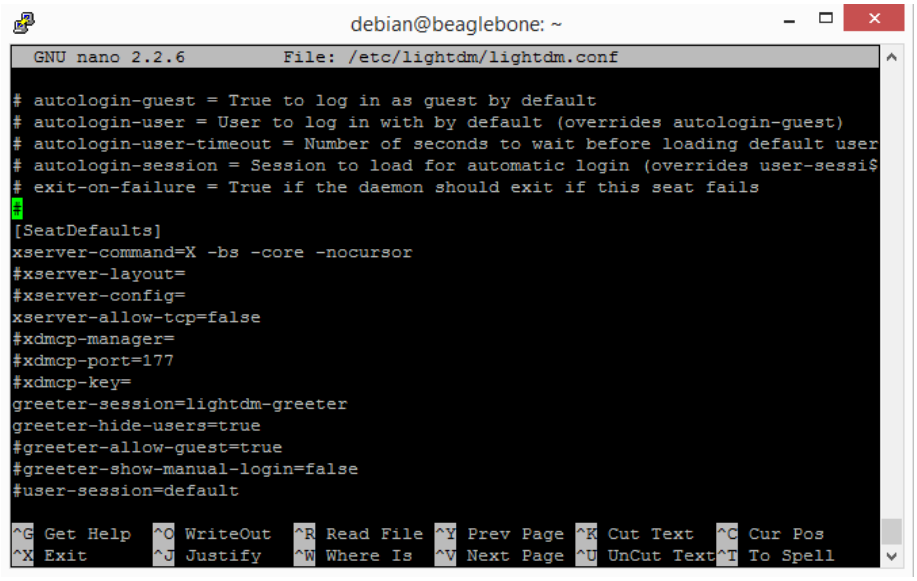
Tyto příkazy provedou smazání obsahu uvedených souborů v adresáři /var/log/. [18]

7.8. Nezobrazování kurzoru

Neboť zabezpečovací terminál má běžet jako aplikace nad operačním systémem je zde absolutně nepotřebné, aby byl zobrazen kurzor. Pro nezobrazování kurzoru je nutné provést změnu části textu, který se nachází v souboru /etc/lightdm/lightdm.conf, úprava se provede následovně. [17]

Nano /etc/lightdm/lightdm.conf

Zde musíme najít část textu začínající s [SeatDefaults] a vyhledat řádek ve kterém je obsaženo # server-command=x. Pro schování kurzoru je nutné zde dopsat část –bs –core –nocursor, po dopsání a smazání znaku mřížky před příslušným řádkem, stačí provést reboot platformy a po načtení nebude kurzor zobrazován.

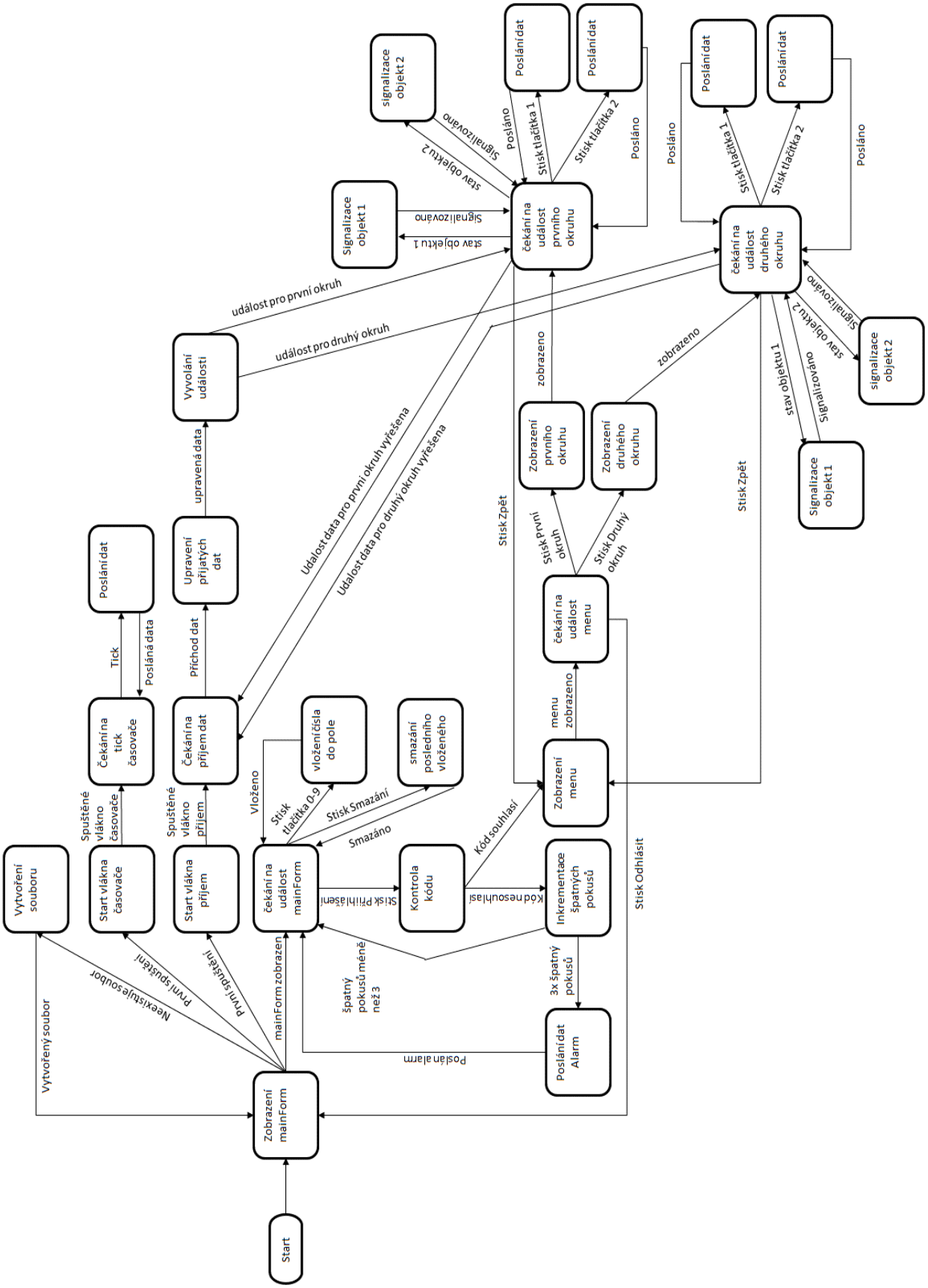


Obr. 16 Stav souboru po provedení úprav pro nezobrazování kursoru

7.9. Realizace řídícího programu

Realizace řídícího programu zabezpečovacího terminálu se skládá z realizace přihlašovací části a realizace komunikace. V následujících kapitolách bude popsán podrobnější popis funkce řídícího programu a bude zobrazen stavový diagram řídícího programu doplněný o jednoduchý popis.

Stavový diagram



Obr. 17 Stavový diagram

7.10. Popis stavového diagramu

Start programu vyvolá akci, která zobrazí panel mainForm, následně se v případě neexistence souboru, který obsahuje uživatelské údaje, tento soubor vytvoří. Pokud se jedná o spuštění po startu programu, spustí se dvojice vláken, vlákno časovače a vlákno přijímaní dat. Po zobrazení panelu očekává program událost na příslušném panelu mainForm. Událostí může být stisk tlačítek. Každé tlačítko má vlastní událost, na kterou bude řídící program reagovat. Stisk tlačítka 0-9 vyvolá akci, která přidá hodnotu tlačítka do pole obsahující vložený kód. Stisk tlačítka Smazání smaže poslední hodnotu pole. A stisk tlačítka Přihlášení zkontroluje stav textového pole a stav obsahuje souboru. Po kontrole a vyhodnocení kódu jako správný, dojde k zobrazení panelu menu. Po zobrazení panelu menu se následně opět čeká na událost stisk tlačítek nacházejících se na tomto panelu. Tlačítka na panelu menu jsou První okruh, Druhý okruh a Odhlášení. Tlačítka První a Druhý okruh zobrazí panel, na kterých se čeká na událost, událostí může být stisk tlačítka, která pošle zprávu nebo událost na změnu stavu prvku, která způsobí změnu signalizačního prvku. Součástí panelů První okruh a Druhý okruh je tlačítko Zpět, které slouží pro návrat z těchto panelů na panel předchozí, tedy menu. Součástí panelu menu je tlačítko Odhlášení, které umožní návrat na panel mainForm.

Vlákno časovače čeká na tick časovače, jakmile nastane tento tick, součástí vlákna se pošlou data. Po poslání dat se opět čeká na další tick.

Vlákno přijímání dat čeká na příjem dat z portu, jakmile tyto data přijdou, musí se pro další práci upravit, upravená data vyvolají příslušnou událost. Události mohou být inicializovat událost v prvním nebo druhém okruhu, tyto události změní stav signalizačních prvků, dle příslušné události.

7.11. Popis řídicího programu

V této části kapitoly se popíšu podrobněji funkci řídícího programu zabezpečovacího terminálu. Popis bude rozdělen na jednotlivé části řídícího programu.

Start programu

Po startu programu, program zobrazí panel *BPmainForm.* V konstruktoru panelu jsou metody inicializace, startUp a statusFile. V metodě inicializace se provede inicializace komponentů formy a také zaregistrování událostí na stisk tlačítek, toto je provedeno prostřednictvím přidáním tlačítka do listu událostí. Po provedení metody inicializace je volána metoda *startUp,* v této metodě se spouští nové vlákna, prvním vláknem je *BPreceive*, toto vlákno slouží pro příjem UDP paketů, dalším vláknem vytvořeným a spuštěním v této metodě je *BPtimerSend*, toto vlákno slouží pro opakované posílání zpráv. Součástí metody startUp je také zaregistrování události na příjem paketů. Toto zaregistrování se provádí přidáním instance *BPcircuitFirst* a *BPcircuitSecond* do vytvořeného listu událostí, který se nachází ve vlákně pro příjem paketů *BPrecive*. Po provedení metody *startUp* je volána metoda *statusFile*, v této metodě se kontroluje, zda existuje soubor jménem *Heslo.bk* ve kterém jsou obsaženy uživatelské údaje. Pokud tento soubor neexistuje tak se prostřednictvím této metody vytvoří a zapíší se do tohoto souboru uživatelské údaje o dvou uživatelích. Vytvoření souboru má relativní cestu a tedy podle toho odkud je program volán se vždy vytvoří adresář Hesla a do něj se následně vytvoří tento soubor jménem *Heslo.bk*.

Panel mainForm

Po kompletní inicializaci a zobrazení panelu *BPmainForm*, vyčkává řídící program na událost, kterou může být stisk tlačítka. Na tomto panelu se nachází pole pro vkládání přihlašovacího kódu s oznamovacím popiskem a vytvořená klávesnice, která umožní vkládání uživatelského přihlašovacího kódu. Jakmile dojde ke stisku tlačítek na klávesnici označené hodnotami od 0 do 9, nastane událost na stisk tohoto tlačítka. Po rozpoznání správné události se zavolá metoda *setPasswordValue* s parametrem odpovídající číslu stisknutého tlačítka, získané pomocí metody *getText* z textu napsaného uvnitř tlačítka, metoda *setPasswordValue* přidá do obsahu textového pole hodnotu, odpovídajícího parametru. Jakmile nastane stisk tlačítka označeného Smazat, zavolá se metoda *deletePasswordLastValue*, která smaže z obsahu textového pole poslední hodnotu. Obsahem panelu *mainForm* je také tlačítko Přihlášení, které bude popsáno v následující kapitole přihlášení.

Přihlášení

Jakmile uživatel stiskne tlačítko přihlášení, které se nachází na panelu *mainForm*, zkontroluje se stav textového pole, zda textové pole obsahuje nějaké data. Pokud textové pole neobsahuje žádný znak, požádá řídící program uživatele, aby zadal heslo. Toto požádání je formou oznámení a je zprostředkované zobrazením textu Vlož kód v popisku nacházející se na panelu *BPmainForm.* Pokud textové pole obsahuje nějaký znak, provede se přečtení obsahu toho pole. Přečtení se provádí prostřednictvím metody *getPassword*, přečtená data jsou ve formě pole znaků a proto je nutné provést převod na *string*. Převod na *string* se realizuje zavolání metody *getStringFromCharArray*. Vstupním parametrem této metody jsou data ve formě pole znaků a výstupem je odpovídající textový řetězec *string*. Po převedení obsahu textového pole se vytvoří instance třídy *BPsoubor* a zavolá se metoda *testLogUser* se vstupním parametrem konvertovaným obsahem textového pole. V této metodě se najde soubor zadaný relativní cestou Hesla/Heslo.bk a zkontroluje se, zda existuje tento soubor, pokud existuje, vytvoří se reader pro tento soubor, který přečte obsah dat v souboru. Ze souboru se vždy přečte jeden řádek textu a provede se jeho úprava, tato úprava rozdělí přečtený řádek na dvě části na část, která bude obsahovat *JménoUživatele* a na část, která bude obsahovat *Kód*. Rozdělení se provádí pomocí metody *split*, která rozdělí přečtený řádek podle zadaného znaku, tímto znakem je „_“. Po upravení dat se přejde k testování kódu pomocí metody *matches*, která otestuje, zda přečtená část odpovídající uživatelskému kódu je totožná zadanému kódu od uživatele, pokud nesouhlasí kódy, přečte se obsah dalšího řádku souboru, za podmínky, že další řádek existuje v souboru. Pokud se nalezne shoda kódu v souboru a zadaného kódu, zavolá se metoda *setloggedUser*, ve které se zapíše *JmenoUživatele* pro zadaný kód. A následně se vrátí zpět z metody *testLogUser* ve třídě *BPsoubor*. Zde se v případě nálezu získá *JménoUživatele* a zobrazí se nový panel *BPmenu*. Pokud nebyla, naleze shoda, napíše se do popisku na panelu oznamovací zpráva špatné heslo a zavolá se metoda *setBadPasswordCounter*, ve které se počítá počet špatně zadaných pokusů. Pokud počet zadaných pokusů dovrší počtu 3, zavolá se metoda *makeAlarm*, ve které se prostřednictvím metody *sendMessege*, která se nachází ve třídě *BPsend,* pošle zpráva.

Panel menu

Při vytváření instance *BPmenu* musíme předat instance *BPcircuitFirst* a *BPcircuitSecond*, neboť tyto instance byly použity při zaregistrování událostí na příchod paketů, aby zabezpečovací terminál správně vyhodnocoval událost na změnu signalizačního prvku jednoho z panelu. V konstruktoru této instance se provede zaregistrování události na akci od tlačítek a následně se v popisku, který se nachází na v této instanci, zobrazí oznamovací hláška s aktuálně přihlášeným uživatelem. Na tomto panelu se nachází tři tlačítka *První okruh*, *Druhý okruh* a *Odhlášení*. Tyto tlačítka slouží pro větvení programu.

Při stisku tlačítek *První Okruh* a *Druhý Okruh* se vyvolá událost, ve které se nejprve předá instance a jméno přihlášeného uživatele a následně se zobrazí příslušná instance podle stisknutého tlačítka, tedy pro *První okruh BPcircuitFirst* a pro *Druhy okruh BPcircuitSecond*. Tlačítko Odhlášení slouží pro navrácení se do *BPmainForm* a tedy pro možnost přihlášení jiného uživatele.

Panel první a druhý

V instanci *BPcircuitFirst* nebo *BPcircuitSecond* se nachází tlačítka pro zamknutí popřípadě odemknutí prvků a příslušné signalizační prvky. Při stisku tlačítka se vytvoří instance *BPsend* a prostřednictvím její metody *sendMessege* se pošle zpráva o zamknutí nebo odemknutí. Změna stavu signalizační prvku je způsobená až po příchodu požadovaného paketu. Změna signalizačního prvku je provedena v *handleru* na událost příchodu dat. Tyto *handlery* jsou *makeFirstOneObject* pro první objekt a *makeFirtsTwoObject* pro druhý objekt, oba tyto objekty se nachází v BPcircuitFirst. A následně *makeSecondOneObject* pro první objekt a *makeSecondTwoObject* pro druhý objekt, oba tyto objekty se nachází na *BPcuircuitSecond*.

Komunikace

Součástí komunikace je posílání a přijímaní UPD paketů, které se nachází ve třídách *BPsend* a *BPreceive*. Třída *BPsend* obsahuje metodu *sendMessege*, ve které se vytvoří instance třídy *SenderThread*, pro zavolání metody *sendMessege* musíme zadat data, která chceme poslat, adresu na kterou chceme data poslat a číslo portu na který budeme posílat zprávu. Následně se vytvoří vlákno, které bude implementovat třídu *SenderThread*. Po spuštění vlákna se zavolá metoda run, která se nachází v implementované třídě *SenderThread*. V této metodě se získají údaje pro datagram paket ve správném tvaru. Nejprve se převede adresa zadaná ve *stringu* na adresu ve formátu *InetAddress* pomocí metody *getByName* s parametrem zadané adresy ve *stringu*. Následně se vytvoří datagram socket. Neboť součástí datagram paktu je velikost zprávy, získám velikost zprávy a následně převedu obsah zprávy na pole bajtů. Po těchto přípravách již vytvořím datagram *packet*, který bude obsahovat zprávu, velikost zprávy, adresu na kterou se posílá a číslo portu na který se posílá. Již je možné zavolat metodu *sendToInterface*, která pošle datagram paket a uzavře datagram soket. Tato metoda je *synchronized*, toto zajistí, že nedojde k možnosti, aby se tato metoda zavolala znovu, dokud není metoda již zavolaná zcela provedena.

Třída *BPreceive* implementuje *Runnable*, tedy možnost aby tato třída mohla být vláknem. Při spuštění daného vlákna pro tuto třídu. Spustí se metoda run, v této metodě se vytvoří datagram *socket* doplněný o port, na kterém bude číst příchod paketů. Následně se vytvoří datagram paket, kterému se přiřadí pole bajtů pro zprávu a velikost tohoto pole. A následně se zavolá metoda *receive* doplněná o vytvořený datagram paket, zavolání této metody se čeká na příchozí paket. Po příchodu paketů se zkontroluje obsah příchozí zprávy. Pokud obsah zprávy odpovídá jedné z očekávajících obsahu, dojde k vyvolání příslušné události, která je následně vyřešená v příslušném panelu prvním nebo druhém okruhu. Obsluha těchto událostí vede ke změně stavu signalizačního prvku, které následně odpovídá stavu zamknuto nebo odemknuto.

Událost na příchod paketů

Pro vytvoření události na příchod paketů je zapotřebí nejprve vytvořit interface, tedy *listener* na požadovanou událost. Tento interface byl vytvořen ve třídě *BPreceiveEventListener*. Součástí tohoto *listeneru* jsou deklarace metod, které bude řešit událost. Následně je nutné vytvořit třídu s metodou, která umožní přenos zdroje události mezi dalšími třídami. Poté se ve třídě, ve které vniká zdroje události, tedy ve třídě *BPreceive*, vytvořit list *listenerů* na událost. Následně se vytvoří dvě metody ve třídě BPreceive, tyto dvě metody slouží pro zaregistrování a odregistrování událostí, *addEventListener* a *removeEventListener*. Následně musíme napsat metody, které budou upozorňovat instance *BPcircuitFirst* a *BPcircuitSecond* na vyvolanou událost, těmito metody jsou *fireMakeFirstOneObject*, *fireMakeFirstTwoObject*, *fireMakeSecondOneObject* a *fireMakeSecondTwoObject*. Každá z těchto metod vyvolá v instanci jinou metodu, která bude řešit danou událost.

Jakmile máme vytvořenou událost, musíme přiřadit, tedy zaregistrovat událost pro instance třídy *BPcircuitFirst* a *BPcircuitSecond* pomocí *addEventListener*, aby toto bylo možné, musí tyto třídy implementovat vytvořený interface *BPreceiveEventListener* a musí obsahovat abstraktní třídy tohoto interface.

8. Testování

Testování zabezpečovacího terminálu je rozděleno na dvě různé části testování. První část se orientuje na testování prototypu zabezpečovacího terminálu, tedy na testování při vývoji řídícího programu pro zabezpečovací terminál. V této části se popisuje, testování jednotlivých částí zabezpečovacího terminálu, tedy přihlašování a komunikace. V druhá část se bude zabývat testováním již kompletního zabezpečovacího terminálu, tedy zkompletovaného řídícího programu na platformě BeagleBone Black.

8.1. Testovaní prototypu

Testování prototypu jak již bylo zmíněno, bude obsahovat testování jednotlivých částí řídícího programu při jeho vývoji, tedy přihlašovací části a komunikace. Testování přihlašovací části obsahovalo, testování vytvoření souboru jeho přečtení a následně vyhodnocení správnosti obsahu vloženého kódu uživatelem a přečteného kódu nacházejícího se ve vytvořeném souboru. Testování komunikace bude obsahovat, schopnost prototypu posílat a přijímat data ve formě UDP paketů prostřednictvím sítě Ethernet.

Příprava pro testování

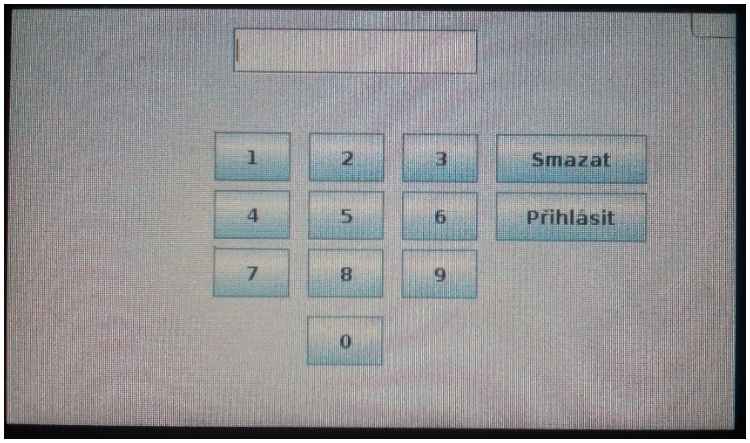
Pro možnost testování bylo nutné připojit platformu BeagleBone Black k pracovnímu počítači. Toto připojení bylo realizováno nejprve prostřednictvím USB kabelů, později bylo nahrazeno za propojení prostřednictvím sítě Ethernet, neboť při testování posílání dat docházelo, že platforma posílala data, prostřednictvím USB, která také vytváří síť, místo požadované sítě Ethernet. Po propojení následovalo nahrání programu do platformy BeagleBone Black. Nahrání programu bylo provedeno prostřednictvím samotného vývojového prostředí NetBeans IDE, které umožňuje vytvoření Java platformy pro vzdálené zařízení. Následně se program přenese na platformu.

Spuštění aplikace

Po nahrání aplikace prostřednictvím vývojového prostředí NetBeans IDE, se tato aplikace musí spustit. Spuštění se u prototypu řešilo prostřednictvím terminálu Putty, kde se prostřednictvím příkazu cd přešlo na adresář, ve kterém se nacházel program ve tvaru soubor.jar, jakmile se dostaneme do adresáře zapíše se příkaz java –jar jménosouboru.jar. Následně se testovalo spuštění aplikace prostřednictvím příkazu java –jar /cesta/k/souboru/jmenosouboru.jar

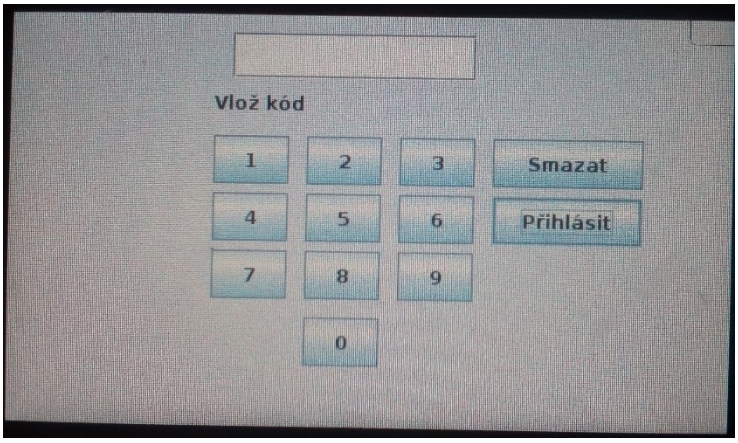
Testování přihlašovací části

Přihlašovací část se testovala tak, že se prostřednictvím klávesnice, která byla vytvořena na příslušném panelu, zadávaly čísla do textového pole. Stisk tlačítka klávesnice zapíše hodnotu tohoto tlačítka do pole. Po stisku tlačítka Smazat smaže poslední hodnotu v textovém poli.



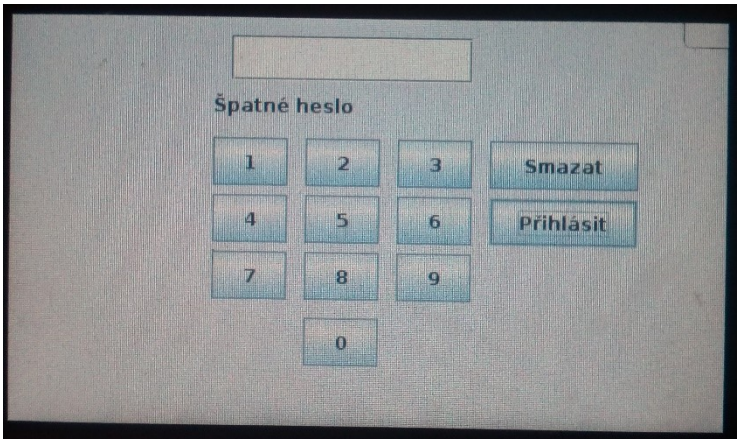
Obr. 18 Obrazovka pro přihlášení

Stisk tlačítka Přihlásit zkontroluje obsah textového pole. Pokud textové pole neobsahuje žádné údaje, objeví se na panelu s klávesnicí oznamovací popisek Vlož kód.



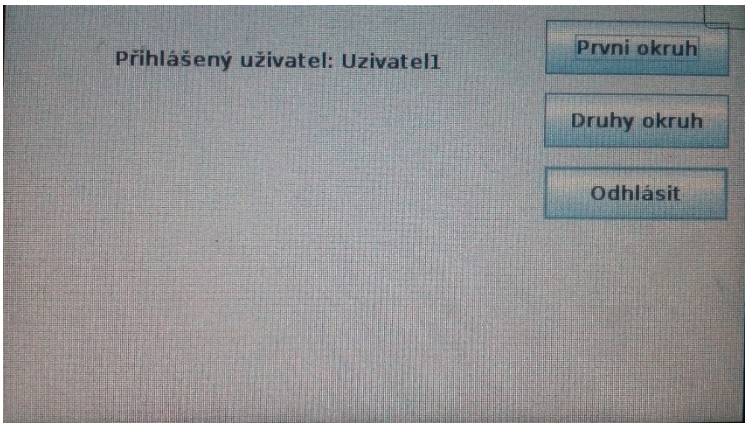
Obr. 19 Obrazovka pro přihlášení s popiskem Vlož kód

Pokud při stisku Přihlásit, obsahuje textové pole znaky. Zkontroluje se tento obsah s obsahem v souboru Heslo.bk tento soubor je vytvořen v závislosti odkud je program volán, buď je /NetBeansProjects/BakalarskaPrace/dist/ a vytvořený adresář Hesla nebo se soubor vytváří ve složce Hesla, která se nachází v adresáři home. Pokud obsahy nesouhlasí, objeví se v popisku na panelu hláška špatné heslo.



Obr. 20 Obrazovka pro přihlášení s popiskem Špatné heslo

Při shodném kódu dojde k zobrazení panelu menu, na tomto panelu lze vidět oznámení o jménu přihlášeného uživatele a lze následně dále pokračovat.



Obr. 21 Obrazovka menu se zobrazeným přihlášeným uživatelem

Testování komunikační části

V této části testování se řešila schopnost zabezpečovacího terminálu komunikovat prostřednictvím UPD paketů s jiným zařízením nacházejícím se v síti Ethernet. K testování komunikace se zprvu použilo programu WireShark, který odchytával příchozí a odchozí pakety. Program WireShark je sice užitečný, ale pro testování není příliš vhodný, neboť odchytává všechny pakety a kvůli tomu je v něm nepřehledné hledání potřebného paketů. Dalším problémem programu WireShark je, že neobsahuje možnost posílat zprávy, ale pouze číst pakety. Z toho důvodů bylo zapotřebí stáhnout terminál Hercules. Tento terminál umožní připojit se na adresu od platformy, nastavit port, na kterém se bude číst a port, na kterém se bude poslouchat. Terminál Hercules je schopný zobrazit příchozí zprávy a také je možné posílat z terminálu zprávy.

Testování posílání zpráv

Testování komunikace jsem rozdělil na dvě části na část posílání a na část příjem. Testování posílání probíhalo následovně, nejprve se v kódu programu určeného pro posílání zpráv cílovou IP adresu na 192.168.1.101, tato IP adresa odpovídala adrese pracovního počítače. Následně se v kódu také nastaví port pro odesílání zpráv na 4445. K ověření, zda komunikace probíhá správně, musí se nastavit v terminálu Hercules IP adresu na 192.168.1.103, která odpovídala adrese platformy, a následně také port, na kterém se má poslouchat na 4445.

Zde se nejdříve objevil problém, že se poslané zprávy nezobrazovaly v terminálu Hercules. Toto bylo zvláštní, neboť při použití programu WireShark se poslané pakety zobrazovaly. Z toho důvodu se muselo upravit část kódu pro posílání zpráv, neboť zřejmě v této části kódu docházelo k tomu, že se stále otevírat soket i když již byl otevřený. Po úpravě kódu se již tato chyba neprojevovala a poslané data již byly zobrazené v terminálu Hercules.

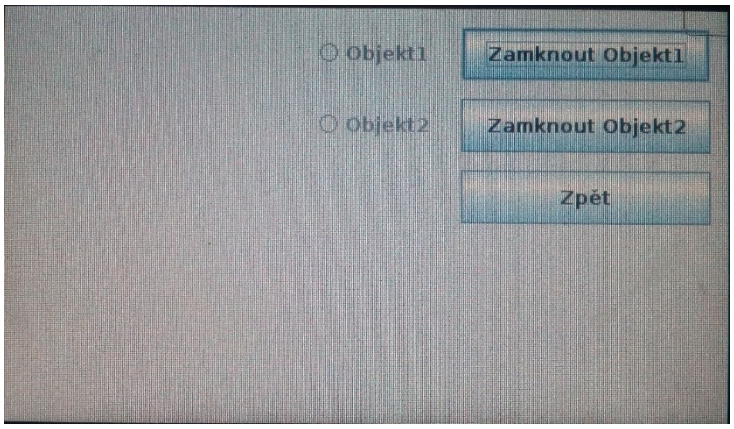
Testování přijímaní zpráv

Testování příjmu zpráv probíhal obdobně jako posílání, jen zde se prostřednictvím terminálu Hercules posílaly zprávy na zadanou IP adresu platformy 192.168.1.103 a na port 4023, toto číslo portu odpovídalo číslu portu, na kterém část kódu určená pro příjem očekávala příchod zpráv. Prostřednictvím terminálu Hercules se poslaly zprávy. Na tyto zprávy čekal program a při obdržení zprávy došlo k vyvolání události, která upozornila na příchod paketu.

Testování komunikace

Když již byla otestovaná schopnost programu pro posílání zpráv a programu pro příjem zpráv komunikovat. Byl změněn obsah zprávy pro jednotlivé objekty a testovalo se propojení komunikačních částí a přihlašovací části. Tedy pokud po stisku tlačítka nacházejícího se na jednom ze dvou okruhů, dojde k vyslání zprávy a zároveň se testovalo, zda při příchodu zprávy dojde k vyvolání události, která změní hodnotu signalizačního prvku nacházejícího se na jednom ze dvou okruhů.

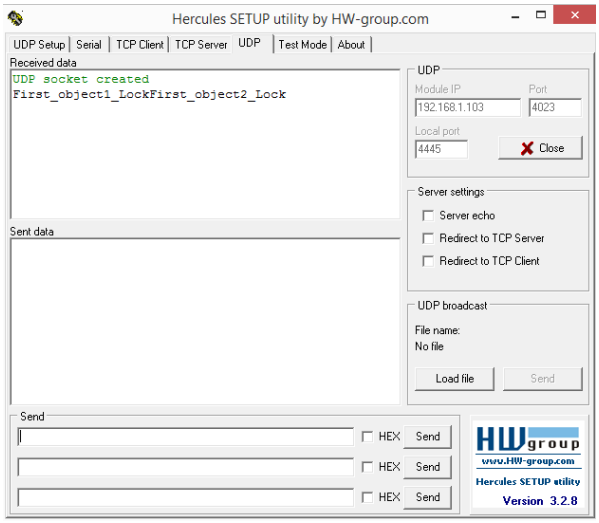
Součástí testování komunikace bylo i testování časovače, který po uplynutí 5s pošle zprávu ve tvaru Object_Status, která je určena pro zjištění stavu prvku.



Obr. 22 Obrazovka okruhů

Formát odeslaných dat

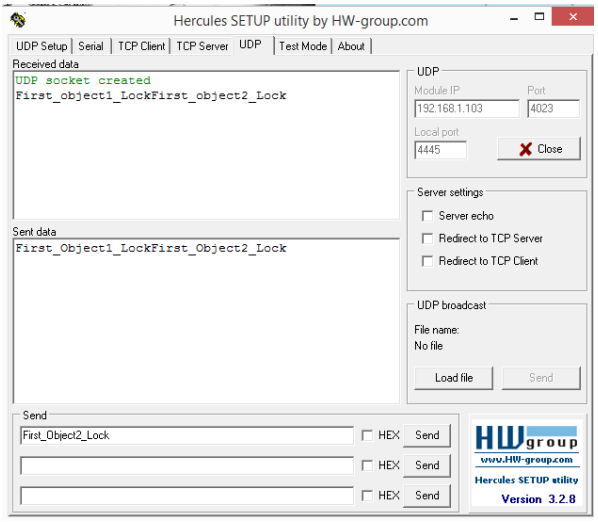
Při stisku tlačítka objekt v prvním okruhu zařízení odešle paket s daty ve tvaru First_objekt s číslem objektu a akcí Lock nebo Unlock, podle stavu prvku, tedy First_objekt1_Lock, First_objekt1_Unlock a First_objekt2_Lock a First_objekt2_Unlock. Stejná akce se provede při stisku tlačítka objektu v druhém okruhu s tím rozdílem, že formát poslaných dat je ve formátu Second_objekt1_Lock, Second_object1_Ulock a Second_object2_Lock , Second_object2_Unlock.



Obr. 23 Ukázka formátu přijatých zpráv

Formát přijímaných dat

Pro zamknutí požaduje zabezpečovací terminál zprávu ve tvaru First_Object1_Lock, First_Object1_Unlock a First_Object2_Lock, First_Object2_Unlock pro změnu stavu signalizačního prvku v prvním okruhu. Stejný formát zprávy obsahuje i signalizační prvky druhého okruhu, tedy Second_Object1_Lock, Second_Object2_Unlock a Second_Object1_Lock, Second_Object2_Unlock.



Obr. 24 Ukázka formátu zpráv odeslaných zpráv

8.2. Testování zabezpečovacího terminálu

Testování zkompletovaného zabezpečovacího terminálu obsahovalo jeho připojení k sítí prostřednictvím UTP kabelu. Po připojení následovalo spuštění řídícího programu zabezpečovacího terminálu na platformě. Po spuštění aplikace se objevil panel, na kterém se zadával přihlašovací kód prostřednictvím klávesnice na panelu.

Pro otestování správné činnosti zabezpečovacího terminálu bylo nutné sepsání testovací aplikace. Tato aplikace byla následně spuštěná na jiném zařízení, toto zařízení byl pracovní počítač. Testovací aplikace byla sepsaná, tak aby se na ní nacházely signalizační prvky objektů a tlačítka, která měnila stavy signalizačních prvků. Aplikace byla takto vytvořena z důvodu simulace změny stavu prvku i jinak než prostřednictvím zabezpečovacího terminálu.

Testovací aplikace

Úkolem testovací aplikace je přijímat zprávy ze zabezpečovacího terminálu. Zabezpečovací terminál odešle zprávu s příkazem o změně prvku, tedy zamknutí nebo odemčení. Po získání příslušné zprávy provede testovací aplikace akci, která zamkne prvek, změnou signalizačního prvku a odešle odpověď o příslušné akci zpět do zabezpečovací terminálu. V testovací aplikaci lze také provést manuální změnu prvků, aby bylo možné testování, zda je zabezpečovací terminál schopen získat tento stav. Stav prvků se získá odesláním zprávy s Object_Status ze zabezpečovacího terminálu, na kterou testovací aplikace zareaguje a pošle zprávy o stavech jednotlivých prvků.

Obr. 25 Testovací aplikace

Testování přihlášení

Testování přihlášení probíhalo obdobně jako při testování prototypu. Testovala se schopnost zabezpečovacího terminálu spouštět správně vyhodnotit vložený kód, tedy zda při správně zadaném kódu dojde k zobrazení panelu menu a při nesprávně zadaném kódu dojde k zobrazení oznamovací hlášky.

Při pokusu přihlášení, při kterém bylo prázdné pole pro přihlašovací kód se testovalo, zda dojde k zobrazení hlášky, vložte kód. Při zkoušce přihlásit se, když je vložen nesprávný kód, dojde k vymazání obsahu pole pro přihlašovací kód a zobrazení oznamovací hlášky nesprávný kód. Při vložení jednoho ze dvou uživatelských kódů dojde k zobrazení panelu menu a zobrazení jména uživatele, kterému tento kód patří.

Testování komunikace

Testování změny stavu prvku se provádělo po přihlášení na panelu prvního okruhu. Při stisku tlačítka došlo k poslání zprávy do testovací aplikace. Testovací aplikace tuto zprávu přijala a zpracovala. Následně došlo k požadované změně prvku v testovací aplikaci, která následně poslala zpět zabezpečovacímu terminálu zprávu se stavem prvku, kterou zabezpečovací terminál přečetl a nastavil signalizaci toho prvku.

Zároveň také se zde objevuje testování timeru, který každých 5s pošle zprávu do testovací aplikace, kterou žádá o stav prvku, ta na něj odpoví a pošle zpět zprávy o stavech prvků.

9. Závěr

Dosažené výsledky práce jsou zprovoznění platformy BeagleBone Black pro umožnění práce na této platformě. Umožnění platformy BeagleBone Black práci s Java aplikacemi a spouštění grafických aplikací na rozšiřujícím modulu BB-view. Následně vytvoření řídícího programu zabezpečovacího terminálu dle návrhu. Tedy řídící program zabezpečovacího terminálu umožňuje rozpoznání vloženého kódu pro přihlášení. Následně je řídící program zabezpečovacího terminálu schopen posílat zprávy ve formě UDP paketů o žádost ke změně prvků, také je schopný přijímat zprávy ve formě UDP paketů o změnách stavu prvků. Řídící program zabezpečovacího terminálu je schopen pravidelně odesílat zprávy a tím získat stav prvků.

Možností rozšíření zabezpečovacího terminálu jsou možné propojení s řídící jednotkou domu a technologiemi SmartHome. Další možností rozšíření je úprava řídícího programu, který bude umožňovat změnu kódu prostřednictvím terminálu. Popřípadě lze upravit řídící program zabezpečovacího terminálu a vytvořit ovládací terminál pro ovládání SmartHome technologií.

Při vypracování práce se objevila řada problémů s platformou, například platforma má problémy s logy, které způsobují zaplnění vnitřní paměti platformy a následně neschopnost platformy správně pracovat. Dalším problémem platformy je po připojení rozšiřujícího modulu dochází k tomu, že kurzor na rozšiřujícím modulu se samostatně přesunuje na pravou stranu modulu.

Na závěr bych dodal, že při vypracování této práce jsem si rozšířil své dosavadní znalosti, o znalosti s práci z platformou BeagleBone Black, vytváření Java aplikací pro platformu BeagleBone Black a také spouštění těchto aplikací na platformě a rozšiřujícím modulu. A zároveň mi práce umožnila rozšířit znalost v komunikaci prostřednictvím sítě Ethernet.

10. Doporučená literatura

[1] SOBELL, Mark G. *Linux: praktický průvodce*. Vyd. 1. Praha: Computer Press, 1999, xxii, 946 s. Operační systémy. ISBN 80-7226-190-8.

[2] SCHILDT, Herbert. *Mistrovství - Java*. 1. vyd. Brno: Computer Press, 2014, 1224 s. Mistrovství. ISBN 978-80-251-4145-8.

[3] BEAGLEBOARD.ORG.*Beagleboard.org* [online]. 2015 [cit. 2015-09-29]. Dostupné z: http://beagleboard.org/BLACK.

[4] BeagleBone. *Beagleboard.org* [online]. 2016 [cit. 2016-03-27]. Dostupné z: http://beagleboard.org/bone-original

[5] BeagleBoard. *Beagleboard.org* [online]. 2016 [cit. 2016-03-27]. Dostupné z: http://beagleboard.org/beagleboard

[6] BeagleBoard-xM. *Beagleboard.org* [online]. 2016 [cit. 2016-03-27]. Dostupné z: http://beagleboard.org/beagleboard-xm

[7] VALEŠ, Miroslav. *Inteligentní dům*. 1. vyd. Brno: ERA, 2006, viii, 123 s. 21. století. ISBN 80-736-6062-8.

[8] Beagleboard: Main Page. *ELinus* [online]. 2014 [cit. 2016-03-27]. Dostupné z: http://elinux.org/Beagleboard:Main_Page

[9] Beagleboard:BeagleBoneBlack. *ELinux* [online]. [cit. 2016-04-21]. Dostupné z: http://elinux.org/Beagleboard:BeagleBoneBlack

[10] *BB-View Cape* [online]. 2. vydání. Anglie: element14, 2014 [cit. 2016-04-21]. Dostupné z: http://www.farnell.com/datasheets/1824482.pdf

[11] *Element14 BeagleBone Black: System Reference Manual* [online]. 1. vydání. 2014 [cit. 2016-04-21]. Dostupné z: https://www.element14.com/community/servlet/JiveServlet/previewBody/54165-102-5-291579/e14%20BBB_SRM_rev%200.9.pdf

[13] BeagleBone 101. *Beagleboard.org* [online]. 2016 [cit. 2016-04-21]. Dostupné z: http://beagleboard.org/support/bone101

[14] SCHILDT, Herbert. *Java: the complete reference* [online]. Ninth edition. New York: McGraw-Hill Education, 2014 [cit. 2016-04-21]. ISBN 978-007-1808-552. Dostupné z: http://staff.cs.psu.ac.th/iew/cs344-481/Java%20The%20Complete%20Reference%20Ninth%20Edition.pdf

[15] Getting Started with BeagleBone & BeagleBone Black. *BeagleBoard.org* [online]. 2016 [cit. 2016-03-27]. Dostupné z: http://beagleboard.org/getting-started

[16] Beaglebone Black with the BBView and the new Beaglebone Debian image. *Element14 Community*[online]. Velká Británie, 2015 [cit. 2016-04-27]. Dostupné z: https://www.element14.com/community/community/designcenter/single-board-computers/next-gen_beaglebone/blog/2015/05/20/beaglebone-black-with-the-bbview-and-the-new-beaglebone-debian-image

[17] How to hide the mouse cursor. *Askubuntu* [online]. 2015 [cit. 2016-04-27]. Dostupné z: http://askubuntu.com/questions/157134/how-to-hide-the-mouse-cursor

[18] *Exploring BeagleBone: tools and techniques for building with embedded Linux* [online]. 1. Indianapoli, IN: John Wiley, 2015, s. 77 [cit. 2016-04-27]. ISBN 1118935128. Dostupné z: https://books.google.cz/books?id=RHq8BQAAQBAJ&printsec=frontcover&hl=cs#v=onepage&q&f=false

11. Seznam příloh

Příloha na CD/DVD 1 - Řídící_program

Příloha na CD/DVD 2 - Testovací_aplikace