FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Mobilní aplikace pro ovládání digitronových hodin (Nixie clock) přes Bluetooth
Student:	Dimitri Vizelka
Vedoucí:	Ing. Matěj Bartík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Vytvořte novou aplikaci pro ovládání digitronových hodin přes rozhraní Bluetooth.

Budoucí aplikace je určena pro mobilní telefony (nicméně podpora případných dalších platforem je možná a vítaná).

Implementační platforma by měla podporovat iOS nebo Android, zvolení univerzálního frameworku je preferováno.

Při návrhu doporučuji vyjít ze stávající aplikace pro Android (zejména z jejích funkcí). Funkce původní aplikace by měly být zachovány i v nové aplikaci, vznik nových funkcí je vítán, nikoliv vyžadován. Uživatelské rozhraní je možné přepracovat podle vlastního uvážení.

Při implementaci vezměte v potaz již existující specifikace hardwaru a firmwaru.

Výslednou aplikaci řádně otestujte a zdokumentujte.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 30. března 2022 v Praze.

Bakalářská práce

MOBILNÍ APLIKACE PRO OVLÁDÁNÍ DIGITRONOVÝCH HODIN (NIXIE CLOCK) PŘES BLUETOOTH

Dimitri Vizelka

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Matěj Bartík, Ph.D.
15. února 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval svému vedoucímu, Ing. Matěji Bartíkovi, Ph.D., za zajímavé a praktické téma, a také za konzultace a odborné vedení mé bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 15. února 2023 .....

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem, implementací a testováním multiplatformní aplikace pro ovládání digitronových hodin přes komunikační protokol Bluetooth Low Energy. Pro implementaci byly použity technologie ReactJS a Capacitor. Pro samotný zdrojový kód aplikace byl zvolen programovací jazyk TypeScript. Výsledkem je aplikace, kterou lze spustit jako nativní kód pro operační systémy iOS a Android, nebo jako progresivní webovou aplikaci ve webovém prohlížeči.

Klíčová slova multiplatformní aplikace, hodiny, digitron, Bluetooth, BLE, TypeScript, PWA, ReactJS, Capacitor

Abstract

This bachelor's thesis deals with the analysis, design, implementation and testing of a cross-platform application which controls a nixie clock via the Bluetooth Low Energy communication protocol. The application uses ReactJS and Capacitor libraries. The source code was created in TypeScript programming language. The implemented application runs natively on iOS and Android operating systems, but it can be also executed as a progressive web application in a web browser.

Keywords cross-platform application, clock, nixie tube, Bluetooth, BLE, TypeScript, PWA, ReactJS, Capacitor

Seznam zkratek

API	Application Programming Interface
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
DI	Dependency Injection
DOM	Document Object Model
DST	Daylight Saving Time
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HCI	Host Controller Interface
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated development environment
JSON	JavaScript Object Notation
L2CAP	Logical Link Control Adaptation Protocol
LED	Light-Emitting Diode
MAC	Media Access Control
MTU	Maximum Transmission Unit
MVVM	Model–view–viewmodel
NTP	Network Time Protocol
OOP	Object-oriented programming
OP	Operating System
PPCP	Peripheral Preferred Connection Parameters
PWA	Progressive Web Application
RTC	Real-time clock
SEO	Search Engine Optimization
SIG	Special Interest Group
SEČ	Středoevropský čas
SELČ	Středoevropský letní čas
SNTP	Simple Network Time Protocol
SPA	Single Page Application
SSL	Secure Sockets Layer
TWA	Trusted Web Activity
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UUID	Universally unique identifier

Úvod

Digitron (anglicky nixie tube) je elektronická součástka, která zobrazuje číslici či jiný znak (viz obrázek 1). Digitrony se používaly převážně v 50. a 60. letech 20. století. Byly součástí displejů různých elektronických zařízení jako jsou například: hodinky, kalkulačky nebo měřící přístroje. Kvůli vysoké výrobní ceně a krátké životnosti byly v průběhu 70. let nahrazeny LED diodami. V současné době se digitrony opět dostaly na trh. Vysoký zájem je především o digitronové hodiny, které vynikají svým originálním vzhledem.

Na FIT ČVUT v Praze vznikl a je dále rozvíjen projekt NixieClock, jehož cílem je vytvoření digitronových hodin s minimalistickým vzhledem a velkým množstvím doplňkových funkcí. Z těchto důvodů se upustilo od ovládání pomocí mechanických prvků (tlačítek) a hodiny komunikují výhradně bezdrátově pomocí protokolu BLE. Původní ovládací aplikace je dostupná pouze pro platformu Android, což omezovalo potencionální počet zákazníků/uživatelů. Je proto žádoucí, aby nová ovládací aplikace byla bez tohoto omezení.

Cílem této práce je vytvoření multiplatformní aplikace pro ovládání NixieClock přes protokol Bluetooth Low Energy. Tento protokol podporuje řada moderních zařízení jako jsou mobilní telefony, tablety či notebooky. To dává uživatelům možnost ovládání těchto hodin pomocí aplikace běžící na daných zařízeních.

Řešení zajistí pokrytí velkého množství různých zařízení na trhu. Dílčí cíle zahrnují rozbor protokolu Bluetooth Low Energy, protokolu NTP, analýzu požadavků na základě existující specifikace firmwaru a stávající Android aplikace. Dalšími dílčími cíli jsou výběr vhodných technologií, návrh uživatelského rozhraní, implementace a testování výsledné aplikace.

Obrázek 1 Digitron [1]

Kapitola 1
Analýza

Kapitola obsahuje analýzu požadavků na mobilní aplikaci pro digitronové hodiny. Analýza funkčních i nefunkčních požadavků je postavena na základě zadání, specifikace firmwaru a stávající Android aplikace. Dále kapitola pojednává o protokolu Bluetooth Low Energy, hodinách reálného času a protokolu NTP. Poté pojednává o způsobech pro tvorbu mobilních aplikací.

1.1 Požadavky

Mezi hlavní požadavky na aplikaci, které vyplývají ze zadání, patří kompatibilita s aktuálním firmwarem hodin NixieClock a funkčnost odpovídající nejméně stávající Android aplikaci. Klíčovým požadavkem je také tvorba jednoduchého řešení pro podporu více různých platforem. Kompletní popis firmwaru je k dispozici na [2]. Kompletní popis stávající Android aplikace je k dispozici na [3]. V následujících sekcích jsou blíže určeny konkrétní funkční i nefunkční požadavky.

1.1.1 Funkční požadavky

Funkční požadavky specifikují konkrétní možnosti, které má software nabízet. Na základě zadání, specifikace firmwaru a stávající Android aplikace byly definovány následující funkce:

- **F1. Připojení a odpojení**

  Aplikace by měla umožnit vyhledat hodiny a připojit se k nim. Též by měla poskytnout možnost odpojení aktuálně připojených hodin.

- **F2. Budíky**

  Aplikace by měla zobrazovat budíky nastavené na hodinách. Dále by měla umožnit úpravu času, vypnutí a zapnutí zvoleného budíku. Také by měla umět ztlumit zvonění pro všechny zapnuté budíky či nastavit délku daného zvonění.

- **F3. Stopky**

  Aplikace by měla zobrazovat aktuální čas a stav stopek. Zároveň by měla poskytovat možnost ovládání těchto stopek. To zahrnuje spuštění, pozastavení a vypnutí stopek.

- **F4. Časovač**

  Aplikace by měla zobrazovat aktuální čas a stav časovače. Zároveň by měla poskytovat možnost ovládání tohoto časovače. To zahrnuje zadání času, spuštění, pozastavení a vypnutí

časovače. Dále by měla umět ztlumit zvonění po doběhnutí aktuálně spuštěného časovače nebo nastavit délku daného zvonění.

- **F5. Datum a čas**

  Aplikace by měla být schopna zobrazovat a měnit datum a čas, který je nastaven na hodinách.

  - **F5.1. Volba způsobu zobrazení času**

    Aplikace by měla mít možnost nastavení frekvence blikání LED dvojteček při zobrazování času, případně dané blikání vypnout. Dále by měla poskytovat možnost vypnutí zobrazení času nebo nastavení délky daného zobrazení.
  - **F5.2. Volba způsobu zobrazení data**

    Hodiny podporují evropský a americký formát zobrazení data. Aplikace by tedy měla nabízet i možnost volby konkrétního formátu. Dále by měla poskytovat možnost vypnutí zobrazení data nebo nastavení délky daného zobrazení.

- **F6. Kalibrace hodin**

  Aplikace by měla umět odhalit desynchronizaci hodin a nabídnout možnost jejich kalibrace.

1.1.2 Nefunkční požadavky

Nefunkční požadavky doplňují funkční požadavky a určují, jak by měla výsledná aplikace fungovat.

- **N1. Podporované platformy**

  Aplikace by měla být multiplatformní. Podporovat by měla nejméně mobilní zařízení s operačními systémy Android a iOS. Případně je vítána i podpora webové platformy a desktopových operačních systémů – Windows, macOS a Linux.

  Zvolená verze Android by měla být v souladu s podmínkami pro publikaci do obchodu s aplikacemi Google Play. Pokryto by mělo být alespoň 85 % mobilních zařízení s operačním systémem Android.

  Zvolená verze iOS by měla být v souladu s podmínkami pro publikaci do obchodu s aplikacemi App Store. Pokryto by mělo být alespoň 85 % mobilních zařízení s operačním systémem iOS.

- **N2. Responzivita**

  Aplikace by měla být určena zejména pro mobilní zařízení. V případě podpory jiných platforem, a tedy i zařízení s větší velikostí obrazovky, by se aplikace měla přizpůsobit dané velikosti.

- **N3. Způsoby nastavení data a času**

  Uživatel by měl mít možnost na všech platformách nastavit datum a čas ručně nebo synchronizovat s danou platformou (zařízením). Možnost synchronizace s NTP serverem by měla být podporována nejméně pro operační systém Android.

- **N4. Správa budíků**

  Na hodinách může být najednou uloženo maximálně 10 budíků, proto lze zároveň mít nastaveno nejvýše 10 aktivních budíků. S daným omezením by měla počítat i budoucí aplikace. Budík by mělo být možné nastavit pouze na konkrétní hodinu a minutu.

- **N5. Lokalizace a vícejazyčnost**

  Aplikace by měla podporovat nejméně anglický a český jazyk. Zároveň by měla být zajištěna možnost pro případné budoucí rozšíření o jiné jazyky. Aplikace by se měla umět automaticky přizpůsobit aktuálně zvolenému jazyku daného zařízení. Též by měla mít možnost volby jazyka uživatelem.

- **N6. Barevné schéma**

  Aplikace by se měla umět automaticky přizpůsobit aktuálně zvolenému barevnému režimu daného zařízení. Též by měla mít možnost volby barevného schématu uživatelem. Podporovány by měly být nejméně 2 barevná schémata – světlé a tmavé.

- **N7. Interakce uživatele**

  Při interakcích uživatele, které vyžadují komunikaci s hodinami, by aplikace měla reflektovat stav čekání, úspěch a neúspěch provedené akce.

- **N8. Offline funkčnost**

  Aplikace jako celek by měla po instalaci fungovat i bez připojení k internetu. Tento požadavek zahrnuje zejména funkce vyžadující pouze komunikaci s hodinami. Nezahrnuje tedy funkce aplikace vyžadující připojení k internetu. Nastavení prostředí samotné aplikace (jazyka a barevného režimu) by mělo být přístupné i bez připojení k hodinám.

- **N9. Automatické opětovné připojení**

  V případě nečekaného odpojení by aplikace měla zvládnout se automaticky znovu připojit k hodinám v průběhu krátkého časového intervalu (nejvýše 5 sekund). V případě neúspěchu by bylo třeba nechat možnost znovupřipojení na uživateli.

- **N10. Komunikační protokol**

  Aplikace by měla být kompatibilní se službami definovanými v rámci existujícího firmwaru. Zároveň by měla počítat s případným budoucím přidáním nových funkcí. Úprava současného firmwaru však není součástí požadavků. S hodinami by měla komunikovat prostřednictvím protokolu Bluetooth Low Energy.

- **N11. Testování**

  Aplikace by měla být otestována manuálně vůči existujícímu hardwaru a firmwaru. Dále by měla být část aplikace, zodpovědná za konverzi dat, otestována jednotkovými (Unit) testy.

- **N12. Bezpečnost**

  Firmware neimplementuje bezpečné spojení. Kdokoliv se může k hodinám připojit a ovládat je. Komunikace přes BLE též není nijak šifrována. Aplikace by tedy neměla počítat s bezpečným spojením.

1.2 Bluetooth Low Energy

Architektura BLE se skládá ze 3 částí: Controller, Host a Application. Controller je zodpovědný za komunikaci a zpracování dat na fyzické úrovni zařízení. Controller zahrnuje linkovou vrstvu (Link Layer), která definuje strukturu paketů, způsob posílání paketů a mechanizmus pro udržení spojení. Host definuje protokoly a vrstvy vyšší úrovně. Obsahuje například Attribute Protocol a Generic Attribute Profile, které určují datové formáty a způsoby komunikace BLE zařízení. Dále zahrnuje Generic Access Profile, který určuje možnosti vyhledání a připojení jednotlivých BLE zařízení. Application reprezentuje aplikaci a definuje vysokoúrovňové koncepty užitečné pro koncového uživatele. [4]

1.2.1 Attribute Protocol

Základem protokolu ATT jsou atributy. Atribut reprezentuje adresovatelnou datovou strukturu. Skládá se ze 3 části: adresa (handle), typ (type) a hodnota (value). V rámci protokolu může existovat více atributů stejného typu, proto k jejich adresaci slouží unikátní 16bitové číslo (handle).

Platné adresy jsou v rozsahu od 0x0001 do 0xFFFF. Hodnotu 0x0000 nelze použít. Typ hodnoty je reprezentován unikátním 128bitovým číslem, které je označováno jako UUID. V praxi se používá jako 5 hexadecimálních čísel oddělených znakem '-'. Samotná data jsou uložena v hodnotě atributu. Hodnota může být libovolné velikosti od 0 do 512 bytů. Konkrétní formát hodnoty je určen typem atributu. [4]

Organizace Bluetooth SIG definovala seznam rezervovaných UUID pro specifické typy atributů. Tyto UUID mají stejný základ a navzájem se odlišují pouze v 16 bitech. Pro efektivní posílání těchto rezervovaných UUID mezi BLE zařízeními lze použít i 16bitovou zkrácenou verzi. Dle [4] se základ rezervovaných UUID nazývá Bluetooth Base UUID a vypadá následovně:

00000000-0000-1000-8000-00805F9B34FB

Zkrácené UUID se kombinuje s Base UUID tak, že se dosadí za poslední 2 byty první části Base UUID. Například 128bitová varianta zkráceného UUID 0x2A08 vypadá následovně:

00002A08-0000-1000-8000-00805F9B34FB

Protokol ATT definuje, jak mohou klient a server odesílat zprávy mezi sebou. Dle [4] je komunikace založena na následujících základních typech operací:

- **Request** – požadavek s odpovědí

  Klient pošle na server požadavek a očekává odpověď (Response). Pokud server požadavek úspěšně zpracuje, pošle klientovi validní odpověď. Pokud se však požadavek nezdaří, server vrátí odpověď s konkrétní chybou a jejím důvodem. V každém případě tedy klient obdrží odpověď.

- **Command** – požadavek bez odpovědi

  Klient pošle na server požadavek, ale žádnou odpověď nedostane. S pomocí této operace lze například požádat server o indikace či notifikace.

- **Indication** – pravidelné oznámení vyžadující potvrzení

  Klient může požádat server o indikace určitého atributu. Server poté pravidelně posílá klientovi indikace s aktuální hodnotou požadovaného atributu. Na každou indikaci očekává od klienta potvrzení o přijetí (Confirmation). Server tedy přestává posílat indikace, dokud nedostane potvrzení o přijetí poslední poslané indikace. Server může začít posílat indikace i bez požadavku od klienta.

- **Notification** – pravidelné oznámení nevyžadující potvrzení

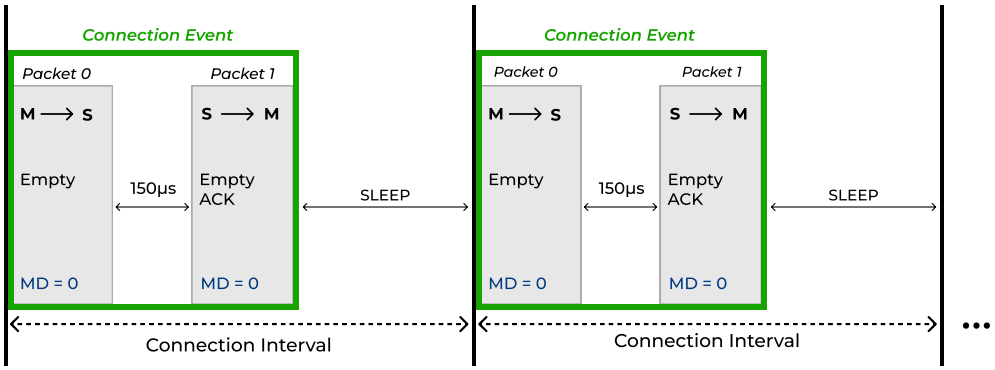
  Klient může požádat server o notifikace určitého atributu. Server poté pravidelně posílá klientovi notifikace s aktuální hodnotou požadovaného atributu. Potvrzení o přijetí notifikace od klienta však neočekává. Server může začít posílat notifikace i bez požadavku od klienta.

1.2.1.1 Connection Event

Zprávy mezi BLE zařízeními se posílají v paketech. Oboustranná komunikace mezi dvěma BLE zařízeními probíhá po jejich připojení. Příklad takové komunikace je zobrazen na obrázku 1.1. V rámci linkové vrstvy má klient roli Master (M) a server roli Slave (S). Klient iniciuje spojení se serverem a určuje Connection Interval. Tento interval definuje, jak často bude probíhat událost Connection Event. V rámci této události probíhá výměna datových paketů. I když nedochází k výměně dat, zařízení si musí pravidelně vyměňovat prázdné pakety, aby bylo zajištěno, že spojení je stále aktivní. Každá událost Connection Event tedy začíná posláním jednoho paketu od klienta. Pokud klient nepotřebuje v daný okamžik poslat žádný požadavek, tak se pošle prázdný paket (EMPTY). V rámci jedné události Connection Event se může poslat více paketů. [4]

Protokol BLE je spolehlivý, což znamená, že každý paket dat odeslaný z jedné strany musí být potvrzen druhou stranou. Toto potvrzení je implementováno na úrovni linkové vrstvy. Každý paket tedy musí být potvrzen příjemcem. Potvrzení může být v prázdném paketu (Empty ACK), nebo také součástí následujícího datového paketu. Mezi každým odeslaným paketem je povinné zpoždění 150 μs. Toto zpoždění se nazývá Inter Frame Space. [5]

Konec události Connection Event je určen bitem More Data (MD) v záhlaví paketu linkové vrstvy. Pokud je tento bit nastaven na 1, tak odesílatel daného paketu má aktuálně připravený minimálně jeden další paket, který chce poslat. Pokud je tento bit nastaven na 0, tak odesílatel již nemá připravený žádný další paket, který chce poslat. Pokud příjemce dostane paket s MD nastaveným na 0 a zároveň sám nechce poslat žádná data, pošle prázdný paket s potvrzením přijetí, který má též bit MD nastavený na 0. Tímto se Connection Event ukončí a zbytek času obě strany spí (SLEEP). Probouzí se až po ukončení aktuálního Connection intervalu, tedy začátkem nového intervalu. Tento mechanizmus slouží pro úsporu energie. [5, 6, 7]



Obrázek 1.1 Události Connection Event v rámci klient-server komunikace přes protokol BLE

Connection Interval může být nastaven na hodnotu v rozmezí od 7,5 milisekund do 4 sekund. Hodnotu intervalu určuje Master v požadavku na připojení (Connection Request). Dále Master určuje Slave Latency a Supervision Time-out. Supervision Time-out je maximální doba od posledního přijatého paketu, po které Master spojení ukončí. Tato hodnota může být v intervalu od 100 ms až do 3200 ms. Slave Latency určuje počet událostí Connection Event, které může Slave ignorovat, pokud nemá připravená žádná data k odeslání. Tato hodnota může být v intervalu od 0 až do 500. Pokud je nastavena na nulu, Slave musí reagovat na každou událost Connection Event. Parametry připojení smí nastavit pouze Master. Dle [8] musí mezi parametry připojení platit následující vztah:

$$(1 + SlaveLatency) \cdot ConnectionInterval < SupervisionTimeout$$

Při prvním připojení volí parametry připojení Master. Slave může doporučit hodnotu pro Connection Interval v rámci vysílaných discoverable advertising paketů. V případě úspěšného připojení může Slave doporučit jiné hodnoty parametrů. Aktualizovat parametry připojení však může pouze Master, který nemusí nastavit doporučené parametry. Pokud Master nenastaví vhodné parametry, Slave může ukončit spojení. Dle [9] existují 2 základní metody, pomocí kterých může Slave doporučit vhodné parametry připojení:

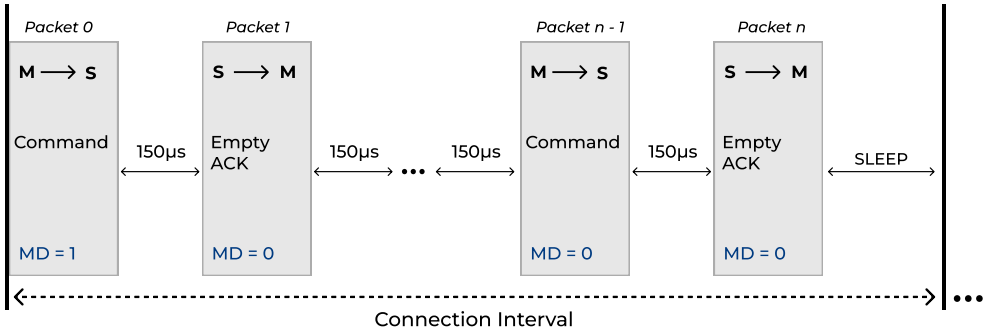
L2CAP Connection Parameter Update Request

Slave může poslat L2CAP paket s požadavkem pro aktualizaci parametrů připojení. Tento požadavek obsahuje minimální a maximální hodnotu pro Connection Interval, hodnotu pro Slave Latency a Supervision Timeout. Master může požadavek ignorovat, zvolit doporučené hodnoty nebo vybrat vlastní hodnoty. Volba konkrétních hodnot závisí na implementaci a může se u jednotlivých zařízení lišit.

- **Peripheral Preferred Connection Parameters**

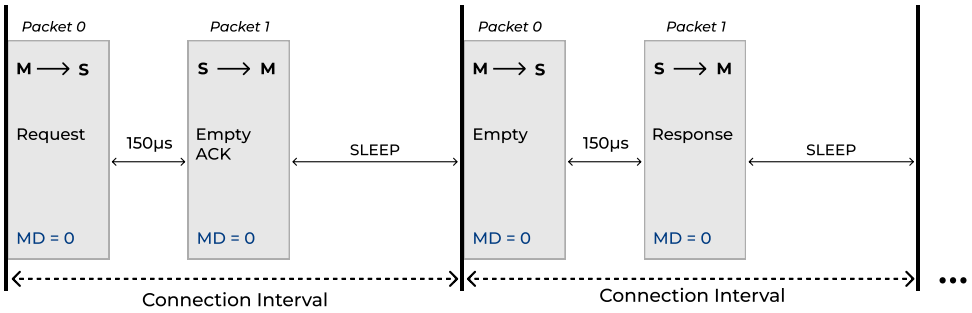
  Slave ukládá doporučené parametry do charakteristiky PPCP v rámci služby GAP Service. Po připojení si Master může tyto hodnoty přečíst. Na základě toho pak může aktualizovat parametry připojení. Tato možnost také závisí na konkrétní implementaci a může se u jednotlivých zařízení lišit.

Operace Command a Notification jsou asynchronní. Protistrana nečeká na odpověď, proto lze stihnout v rámci jedné události Connection Event použít tyto operace vícekrát za sebou. Potvrzení o přijetí paketu se však provede pokaždé v rámci linkové vrstvy. Protistrana ale neposílá samotnou odpověď na operaci, která se provádí v rámci aplikační vrstvy. Příklad použití několika operací Command v rámci jedné události Connection Event je zobrazen na obrázku 1.2. [10]



Obrázek 1.2 Operace Command v rámci události Connection Event protokolu BLE

Naopak operace Request je synchronní. Vyžaduje odpověď před provedením další operace Request. Server obvykle nedokáže zpracovat požadavek za méně než 150 μs, proto odpověď pošle až v další události Connection Event. Potvrzení o přijetí požadavku se však také musí provést v rámci linkové vrstvy, proto server potvrdí přijetí pomocí prázdného paketu. Příklad použití operace Request je zobrazen na obrázku 1.3. [10]



Obrázek 1.3 Operace Request v protokolu BLE

1.2.1.2 Zprávy protokolu ATT

Knihovny, umožňující implementaci BLE klienta pro mobilní a desktopové platformy, poskytují jednoduché rozhraní pro komunikaci s BLE serverem. Většinou umožňují rovnou získat objekty reprezentující servisy, charakteristiky a deskriptory (viz sekce 1.2.2). Dále zpřístupňují operace pro čtení, zápis, notifikace a indikace rovnou pomocí UUID atributu. V rámci protokolu ATT je však pro tyto operace použita adresa (handle) atributu. Knihovny tedy ukrývají operace

související s adresami atributů a proces objevování jednotlivých atributů. Tato sekce obsahuje popis některých zpráv, používaných v rámci protokolu ATT, které objasní procesy skryté za rozhraním těchto knihoven. [4]

- **Read Request**

  Vrátí hodnotu jednoho atributu na základě jeho adresy. [4]

- **Read Blob Request**

  Pokud se hodnota v odpovědi Read Request nevejde do jednoho paketu, pak server vrátí pouze část hodnoty. Pro zbytek hodnoty může klient požádat pomocí Read Blob Request, který navíc obsahuje offset. [4]

- **Read Multiple Request**

  Požadavek slouží k zjištění hodnot více atributů najednou. Klient pošle adresy atributů a v odpovědi dostane jejich hodnoty ve stejném pořadí. Je povoleno požádat o atribut s hodnotou proměnné délky pouze v případě, je-li uveden jako poslední v seznamu. Ostatní atributy musí mít hodnotu fixní délky. Hodnoty, které se nevejdou do velikosti paketu, jsou vynechány. [4]

- **Write Request**

  Klient pošle požadavek s adresou a hodnotou, kterou chce uložit v atributu s zadanou adresou. Server v odpovědi potvrdí, zda byla hodnota uložena. [4]

- **Read/Write Command**

  Read Command a Write Command fungují jako Read Request a Write Request. Rozdílem je pouze to, že server neposílá žádnou odpověď. Hodí se především pro dlouhé operace, protože klient nečeká na odpověď a neblokuje komunikaci. [4]

- **Find Information Request**

  Požadavek slouží k zjištění adresy (handle) a typu atributu. Klient pošle na server libovolný interval adres atributů. V případě, že existují atributy s adresou v požadovaném intervalu, pošle server v odpovědi adresy a typy těchto atributů. Odpověď však nemusí obsahovat hodnoty pro všechny atributy s adresami v zadaném intervalu, protože pakety mají omezenou velikost. Klient by tedy měl zkoušet měnit interval. V případě, že interval neobsahuje žádné atributy, nebo je daný interval neplatný, dostane klient odpověď typu Error Response. [4]

- **Find By Type Value Request**

  Požadavek slouží k zjištění adresy (handle) atributu podle jeho typu a hodnoty. Princip je stejný jako u Find Information Request. Rozdíl je v tom, že klient pošle navíc typ a hodnotu požadovaného atributu. V odpovědi pak dostane seznam adres odpovídajících atributů, nebo seznam intervalů adres v případě, že jde o skupinový atribut (viz sekce 1.2.2.1). [4]

- **Read By Type Request**

  Požadavek slouží k zjištění hodnoty atributu podle jeho typu v zadaném rozsahu adres. Odpověď zahrnuje seznam dvojic, obsahujících adresu a hodnotu. Tento požadavek může být použit například pro vyhledání všech charakteristik v servisu. [4]

- **Read By Group Type Request**

  Požadavek je obdobou Read By Type Request. Rozdíl je v tom, že klient může poslat jako typ pouze skupinový atribut. V odpovědi pak dostane kromě adresy a hodnoty zadaného skupinového atributu i adresu posledního atributu v rámci skupiny. [4]

- **The Exchange MTU Request**

  Attribute Protocol má výchozí maximální přenosovou jednotku (MTU) 23 bytů. Pokud chce zařízení posílat větší pakety, musí vyjednat vyšší velikost MTU. Pouze klient může tento požadavek iniciovat. V požadavku pošle svou hodnotu pro MTU. Jako odpověď dostane velikost MTU serveru. Výsledná hodnota MTU, kterou obě strany budou používat, je rovna minimu obou hodnot. [4]

1.2.2 Generic Attribute Profile

GATT je postaven na atributech a komunikačních operacích protokolu ATT. GATT definuje konkrétní typy atributů a jejich použití. Dále určuje vztahy mezi jednotlivými atributy, vytváří skupiny atributů, buduje hierarchii těchto atributů a definuje procedury pro jejich objevování klientem. V rámci GATT jsou představeny koncepty jako služba (Service) a charakteristika (Characteristic). Tyto koncepty jsou následně použity v rámci konkrétní specifikace aplikační vrstvy. Příkladem takové specifikace aplikační vrstvy je komunikační protokol hodin NixieClock, který je uvedený v příloze A.

Organizace Bluetooth SIG, která dohlíží na vývoj standardů Bluetooth, definovala v rámci rezervovaných UUID konkrétní nejčastěji používané služby a charakteristiky. Také pro tyto charakteristiky určila konkrétní formáty hodnot. Díky tomu jsou tyto služby a charakteristiky znovu použitelné. Pro takové služby lze tedy jednoduše napsat klienta, který bude fungovat pro více zařízení od různých výrobců. Také existuje řada generických klientů, které zvládnou prohledat zařízení a zjistit všechny jeho služby a charakteristiky. Dále je mohou zobrazit uživateli v seznamu a umožnit mu měnit hodnoty jednotlivých charakteristik.

1.2.2.1 Service

Služba (Service) je typ atributu, který reprezentuje skupinu charakteristik. Jde tedy o skupinový atribut (anglicky grouping attribute). Ekvivalentem služby ve světě OOP je třída, která implementuje imutabilní interface. Tato třída obsahuje datové položky a implementuje metody pro přístup k jednotlivým datovým položkám. [4]

1.2.2.2 Characteristic

Charakteristika (Characteristic) je typ atributu, který reprezentuje určitou hodnotu a typ této hodnoty. Dále určuje povolení pro čtení nebo zápis hodnoty a konfiguraci pro notifikace či indikace. Characteristic je stejně jako Service skupinovým atributem. Dle [4] se skládá z následujících 3 částí:

- **Declaration atribut**

  Atribut, který určuje začátek charakteristiky. Jako hodnotu obsahuje 3 položky: vlastnosti charakteristiky, adresu Value atributu a typ charakteristiky.

  Vlastnosti charakteristiky reprezentuje 8bitové číslo, kde jednotlivé bity určují, zda jsou povoleny read, write, notification, indication, broadcast. Další bit určuje, zda lze pro read a write použít kromě operace Request i operaci Command. Sedmý bit určuje, zda je pro přístup k hodnotě charakteristiky nutná autentizace. Poslední bit určuje, jestli jsou použity další rozšiřující vlastnosti, které se pak určují v deskriptorech.

  Typ charakteristiky je stejný jako typ Value atributu. Tato duplicitní hodnota slouží k optimalizaci komunikace při objevování charakteristik.

- **Value atribut**

  Atribut, který obsahuje hodnotu charakteristiky. Typem tohoto atributu je typ charakteristiky v podobě UUID.

- **Descriptor atribut**

  Atribut, který obsahuje dodatečnou informaci či konfiguraci charakteristiky. Charakteristika nemusí obsahovat žádný Descriptor atribut.

  Pro generické GATT klienty je například vhodné nastavit User Description Descriptor, který obsahuje popis charakteristiky. Nebo také je dobré přidat Presentation Format Descriptor, který určuje formát hodnoty dané charakteristiky.

  Pokud je ve vlastnosti charakteristiky povolena operace Notification nebo Indication, tak se musí přidat Client Characteristic Configuration Descriptor. Tento deskriptor upřesňuje, zda se pro charakteristiku použije buď jen notifikace, nebo jen indikace. Obě operace najednou používat nelze.

1.2.3 Generic Access Profile

GAP definuje způsoby, jakými se BLE zařízení navzájem vyhledávají a připojují. Určuje též způsob, jakým zařízení vytvářejí privátní spojení nebo permanentní spojení. [4]

1.2.3.1 GAP role

Dle [4] jsou v rámci GAP pro BLE zařízení definovány následující dvojice rolí:

- **Broadcaster a Observer**

  Broadcaster je zařízení, které pravidelně vysílá advertisement pakety. Tyto pakety mohou sledovat zařízení v roli Observer. Přitom žádné spojení mezi zařízeními vyžadováno není.

- **Central a Peripheral**

  Peripheral je zařízení, které vysílá discoverable nebo connectable advertisement pakety a čeká, dokud zařízení v roli Central neiniciuje spojení. Po připojení probíhá mezi těmito zařízeními klient-server komunikace, která je popsána v sekci 1.2.1.

Jedno BLE zařízení může mít více rolí najednou.

1.2.3.2 Vyhledání a připojení

Před připojením vysílají zařízení v roli Peripheral advertisement pakety. Peripheral vysílá buď discoverable, nebo connectable advertisement pakety. Discoverable advertisement pakety nesou informace o daném zařízení. Tyto pakety určují, že jakékoliv zařízení v roli Central může iniciovat spojení. Naopak connectable advertisement pakety určují, že se může znovu připojit zařízení v roli Peripheral, se kterým bylo po prvním připojení navázáno trvalé spojení. Toto trvalé spojení je výsledkem procedury zvané bonding. Využívá se pouze u těch zařízení, která vyžadují autentizaci. [4]

Zařízení v roli Central vyhledává zařízení, vysílající advertisement pakety, ve svém okolí. K tomu používá buď passive scanning, nebo active scanning. Passive scanning znamená, že poslouchá všechny advertisement pakety, které jakékoliv zařízení vysílá. Kvůli omezené velikosti paketu, se do discoverable advertisement paketu nemusí vejít všechny informace. Například se nemusí vejít celý název zařízení nebo UUID všech služeb, které jsou zpřístupněny. Active scanning navíc posílá pro každé nalezené zařízení požadavek na doplňující informace. Tento požadavek se označuje jako Scan Request. Odpovědí na daný požadavek je Scan Response, který obsahuje doplňující informace. Informace z advertisement paketů slouží zejména k tomu, aby klientská aplikace mohla uživateli zobrazit v seznamu nalezených zařízení jejich názvy nebo filtrovat zařízení podle poskytovaných služeb. [4]

1.2.3.3 Discoverable advertisement pakety

Discoverable advertisement pakety mohou zahrnovat úroveň signálu, název zařízení, seznam UUID poskytovaných služeb a další informace o zařízení. Pro název zařízení existují 2 formáty – pro celý název a pro zkrácený název. Stejně tak i pro seznam UUID služeb existuje více formátů. Pro seznam zkrácených 16bitových UUID a kompletních 128bitových UUID jsou definovány různé formáty. Zařízení může obsahovat velké množství různých služeb, které se do seznamu nevejdou. Z tohoto důvodu byly zavedeny 2 formáty pro neúplné seznamy UUID obou druhů. Zařízení v roli Central pak bude vědět, že jde o neúplný seznam nebo zkrácený název zařízení. Pro kompletní seznam pak může požádat pomocí požadavku Scan Request, pokud je v rámci Peripheral tato možnost implementována. Další možností, jak zjistit doplňující informace o zařízení, je po připojení použít GAP service. [4]

Dále mohou discoverable advertisement pakety obsahovat interval Slave Connection Interval Range. Tento interval určuje preferovanou hodnotu pro Connection Interval při připojení. [4]

1.2.3.4 GAP service

GAP definuje vlastní GATT službu. Tato služba také obsahuje informace o zařízení. Může obsahovat například charakteristiky pro název zařízení (Device Name), typ zařízení (Appearance) a doporučené parametry připojení (PPCP). [4]

1.2.4 Podporované platformy

Tato sekce se zabývá stanovením verzí požadovaných platforem podporujících technologii BLE. Dále obsahuje požadavky na implementaci, které jsou specifické pro konkrétní platformu.

1.2.4.1 Android

Android poskytuje podporu pro Bluetooth Low Energy od verze 4.3 (API 18). Od verze Android 6.0 (API 23) byla přidána možnost filtrování nalezených zařízení. [11]

Implementace nízkoúrovňové komunikace závisí na konkrétním výrobci Android telefonu. Například se může u jednotlivých výrobců lišit výběr hodnoty connection intervalu. Minimální hodnotou daného intervalu, kterou Android podporuje, je 7,5 milisekund. [7]

1.2.4.2 iOS, iPadOS

Podpora pro BLE začíná pro řadu mobilních telefonů iPhone od verze iOS 5.0. Všechny iOS telefony počínaje modelem iPhone 4s umožňují BLE komunikaci. Pro tablety řady iPad je minimální verzí iPadOS 5.0. [12]

GATT server může nabídnout aktualizaci parametrů BLE spojení pouze odesláním L2CAP požadavku ve vhodnou dobu. Klient tedy nebude číst ani používat parametry PPCP. Dle [13] může být požadavek na změnu parametrů připojení zamítnut, pokud nesplňuje následující podmínky:

- Slave Latency ≤ 30
- Min Connection Interval ≥ 15 ms (multiples of 15 ms)
- Min Connection Interval + 15 ms $\leq$ Max Connection Interval
- Max Connection Interval $\times$ (Slave Latency + 1) ≤ 2 seconds
- Max Connection Interval $\times$ (Slave Latency + 1) $\times 3 <$ Supervision Timeout
- 2 seconds $\leq$ Supervision Timeout ≤ 6 seconds

1.2.4.3 Desktop

V následujícím seznamu jsou určeny minimální verze nejpoužívanějších operačních systémů, které zajišťují podporu BLE:

- **Windows**

  Oficiální podpora BLE začíná od verze Windows 8 [14]. Windows 7 nemá vestavěné ovladače pro BLE, proto vyžaduje instalaci ovladače od výrobce Bluetooth adaptéru [15].

- **macOS**

  Minimální verzí s podporou BLE je OS X Yosemite (10.10) [12].

- **Linux**

  U konkrétní Linux distribuce je požadován Kernel 3.19+ a instalovaný balíček pro BLE komunikaci – BlueZ 5.41+. [16]

1.2.4.4 Web

Web Bluetooth je JavaScript API, které umožňuje interakci s BLE zařízeními prostřednictvím webových prohlížečů. Vývoj této technologie podporuje společnost Google. Prohlížeč Google Chrome zpřístupňuje rozhraní pro Web Bluetooth již od roku 2017. Předtím bylo možné komunikovat s BLE zařízeními pouze prostřednictvím nativních aplikací, používajících nativní BLE API specifické pro konkrétní platformu. [17]

Web Bluetooth API je stále označeno jako experimentální a nemá plnou podporu všech prohlížečů a operačních systémů. Dle [18] je většina funkcí daného API k dispozici v prohlížeči Google Chrome pro následující operační systémy:

- Android 6.0 a novější
- Windows 10 a novější (Google Chrome 70 a novější)
- macOS X Yosemite (10.10) a novější (Google Chrome 56 a novější)
- ChromeOS

Pro operační systém Linux a starší verze Windows je třeba v nastavení prohlížeče povolit experimentální webové technologie a Web Bluetooth. Pro Linux je třeba mít instalovaný BlueZ 5.41+, který umožňuje použití technologie Bluetooth. Řada Linux distribucí má tento balíček již předinstalovaný. [19]

Web Bluetooth API zpřístupňují také prohlížeče Microsoft Edge a Samsung Internet. Prohlížeč Opera vyžaduje v nastavení povolit experimentální webové technologie. Prohlížeč Firefox dané API nepodporuje vůbec. [19]

Pro operační systém iOS je nejpoužívanějším prohlížečem Safari. Jde o výchozí prohlížeč společnosti Apple, který preferuje více než 90 % uživatelů iOS [20]. V roce 2020 společnost Apple oznámila, že nebude implementovat 16 nových webových technologií (Web API) v Safari, protože představovaly hrozbu pro soukromí uživatelů [21]. Mezi tyto API patřil i Web Bluetooth. V roce 2022 prohlížeč Safari stále nepodporuje danou technologii. Druhým nejpoužívanějším prohlížečem na iOS zařízeních je Google Chrome, který však též nemá podporu pro Web Bluetooth na iOS. Aktuálně mezi prohlížeče podporující BLE patří Bluefy a WebBLE. Tyto prohlížeče byly implementovány za účelem podpory Web Bluetooth technologie na operačním systému iOS. Dle [22] lze Bluefy stáhnout zdarma. Dle [23] je cena WebBLE nyní 1,99 amerických dolarů.

Web Bluetooth API umožňuje implementaci GATT klientů. Základní funkce daného API umožňují vyhledat a připojit se k BLE zařízení. Dále lze číst a měnit charakteristiky, nebo poslouchat notifikace charakteristik. Také je zpřístupněna funkce, která detekuje stav připojení. Některé funkce jsou zatím experimentální. To znamená, že se jejich implementace může v budoucnu

změnit. Pro jejich použití musí uživatel v nastavení prohlížeče jednorázově povolit experimentální webové technologie. Pro tvorbu jednoduchého GATT klienta však stačí i základní funkce. Seznam klíčových funkcí, zpřístupněných jednotlivými prohlížeči, pro tvorbu GATT klienta je uveden v příloze B. [18]

Pro použití Web Bluetooth technologie musí webová stránka fungovat prostřednictvím zabezpečeného připojení. To znamená, že musí mít SSL certifikát a používat protokol HTTPS. Dalším bezpečnostním opatřením je to, že skenování zařízení může být spuštěno pouze uživatelským gestem, jako je dotyk obrazovky nebo kliknutí myší. Po každém otevření prohlížeče tedy uživatel musí nejprve spustit skenování zařízení v okolí. Poté musí vybrat zařízení, k němuž se chce připojit. Automatické opětovné připojení k určitému zařízení bez nutnosti skenování lze implementovat, ale pouze s použitím experimentální funkce. Uživatel tedy musí v nastavení prohlížeče povolit experimentální webové technologie. [18]

1.3 Specifikace hardwaru a firmwaru

Základem hodin NixieClock je procesor z řady STM32WB podporující komunikaci pomocí BLE verze 5.2. Pro udržení aktuálního času slouží RTC, který obsahuje oscilátor s frekvencí 32768 Hz. Hodiny zobrazují číslice od 0 do 9 pomocí šesti digitronů. Tyto digitrony jsou po dvojicích rozděleny pomocí LED diod. Dále hodiny obsahují akcelerometr, který detekuje vibrace a slouží jako fyzický ovládací prvek. Pro zvukové signály je k dispozici bzučák.

Jedním z klíčových požadavků je kompatibilita s existujícím firmwarem [2]. Níže jsou specifikovány pouze některé funkce, implementované ve firmwaru, které vyžadují doplňující popis. Jsou též objasněny některé vlastnosti hodin a jejich chování v nejednoznačných případech. Kompletní seznam nabízených funkcí je uveden v rámci komunikačního protokolu v příloze A.

- **Zobrazení data**

  Uživatel může na hodinách vidět datum ve 2 formátech. Americký formát je zobrazen ve tvaru 'MM.DD.RR'. Evropský formát je zobrazen ve tvaru 'DD.MM.RR'. Znak '.' v tomto formátu je reprezentován rozsvícenou LED diodou. Zobrazují se pouze 2 poslední číslice roku. [2]

- **Zobrazení času**

  Uživatel může na hodinách vidět čas pouze ve 24hodinovém formátu ve tvaru 'HH:MM:SS'. Znak ':' v tomto formátu je reprezentován dvojicí rozsvícených LED diod. [2]

- **LED diody**

  Displej hodin zvládne zobrazit 3 dvojice číslic od 0 do 9. Zobrazené hodnoty data a času se mohou někdy překrývat. K jejich rozlišení slouží právě LED diody. Při zobrazení času blikají LED diody v zadaném intervalu. Toto blikání lze vypnout. [2]

- **Budíky**

  Firmware podporuje uložení fixního počtu deseti budíků. Každý z nich je buď aktivován, nebo deaktivován. Při zvonění budíku dochází k blikání času na displeji, které je synchronizováno s frekvencí blikání LED diod. K synchronizaci dochází pouze v případě, že nejsou LED diody vypnuty. V opačném případě je zvolena fixní hodnota. [2]

- **Časovač**

  Časovač odpočítává čas od zadaného počtu sekund až do 0. Tento čas se zobrazuje ve formátu 'HH:MM:SS'. Maximální zobrazitelná hodnota je '99:59:59'. Po zadání nového času za běhu časovače dojde k jeho restartu. Odpočet začne znovu od nové hodnoty. Ke zvonění v tomto případě nedojde. [2]

- **Stopky**

  Stopky počítají čas od 0 sekund. Tento čas se také zobrazuje ve formátu 'HH:MM:SS'. Maximální zobrazitelná hodnota je též '99:59:59'. Po překročení této hodnoty však k zastavení stopek nedochází a na místě hodin se zobrazují poslední 2 číslice aktuální hodnoty. Celková hodnota času je posílána v sekundách a je reprezentována datovým typem uint32. K přetečení dojde za cca 136 let a není to nijak ošetřeno. [2]

- **Zvonění**

  Zvonění může být vyvoláno budíkem nebo doběhnutím časovače. Hodiny disponují akcelerometrem, který detekuje zrychlení. Vypnutí zvonění se realizuje jakýmkoliv pohybem hodin. [2]

- **Letní čas (DST)**

  Je implementováno automatické přepínání hodin mezi letním a skutečným pásmovým časem [2]. Přepnutí je harmonizováno direktivou Evropské komise. Přechod na letní čas se provádí poslední neděli v březnu. Po 01:59:59 SEČ (středoevropského času) se čas mění na 03:00:00 SELČ (středoevropského letního času). Letní čas končí poslední neděli v říjnu. Po 02:59:59 SELČ se hodiny posunou na 02:00:00 SEČ. [24]

  Některé země jako Rusko či Bělorusko letní čas nepoužívají vůbec. U jiných zemí jako například USA či Kanada dochází k přepnutí v odlišnou dobu než v Evropě. Firmware ovšem neumožňuje automatické přepínání upravit či vypnout [2].

- **Vícenásobné připojení**

  Firmware nepodporuje více než 1 připojení zároveň. [2]

- **Základní režim**

  Pokud nedochází ke zvonění, zobrazení stopek či časovače, potom jsou hodiny v základním režimu. V tomto režimu se střídá zobrazení času a data. Firmware umožňuje nastavení trvání těchto zobrazení. [2]

1.3.1 Komunikační protokol

Komunikace mezi hodinami a aplikací je postavena na architektuře klient-server. Komunikační protokol prošel 3 fáze vývoje. Nejprve byl navržen vedoucím bakalářské práce ve sdíleném dokumentu [25]. V rámci bakalářské práce, zabývající se tvorbou Android aplikace [3], byl zpřesněn a byla implementována klientská část využívající daný protokol. Nakonec byl v rámci bakalářské práce, zabývající se tvorbou firmwaru [2], doplněn o nové služby. Také byly specifikovány hodnoty pro provedení určitých akcí a chování v nejednoznačných případech.

Při tvorbě firmwaru nebylo nalezeno fungující interní řešení pro kalibraci RTC, proto bylo rozhodnuto přenechat kalibraci na klientskou aplikaci. Protokol byl tedy v poslední verzi rozšířen o možnost přesné kalibrace hodin. Podrobnější popis procesu kalibrace obsahuje sekce 1.4.

Pro doporučení parametrů připojení jsou jejich hodnoty uloženy v rámci charakteristiky PPCP ve službě GAP Service. Pro Connection Interval je nastavena minimální i maximální hodnota na 7,5 ms. Hodnota Slave Latency je nastavena na 0 a hodnota Supervision Timeout na maximální možnou hodnotu – 0xFFFF. [2]

Detailní popis stávajícího API, které serverová část poskytuje, lze najít v příloze A. Obecné vlastnosti protokolu jsou následující:

- Akceptovány jsou pouze validní hodnoty. Nevalidní hodnoty jsou ignorovány a vrátí se Error Response. [2]

- Pořadí bytů hodnot charakteristik posílaných v rámci protokolu je vždy big-endian, pokud se nejedná o typy hodnot definované organizací Bluetooth SIG. Například rok v rámci typu Date Time se posílá jako little-endian [26].
- Discoverable advertisement pakety nezahrnují UUID všech služeb. Obsahují pouze zkrácený seznam UUID služeb, který vždy obsahuje UUID služby hodin. [2]

1.4 Hodiny reálného času

Hodiny reálného času (RTC) jsou počítačové hodiny, které sledují aktuální čas. Používají se v osobních počítačích, serverech a systémech, které vyžadují přesné měření času. RTC mají často vlastní zdroj napájení. Mohou tedy uchovávat čas i v případě, že je primární zdroj napájení vypnutý. Většina RTC používá krystalový oscilátor. Frekvence krystalu je obvykle 32768 Hz. Vlivem teploty a stárnutí krystalů oscilátoru může dojít ke změně frekvence RTC, a tedy i časové nepřesnosti. [27]

Pro odstranění časové nepřesnosti používají hodiny NixieClock metodu přesné kalibrace (anglicky smooth calibration) [2]. Daná metoda umožňuje úpravu frekvence RTC v rozmezí od -511 až do +512 taktů za časový interval označovaný jako kalibrační okénko (anglicky calibration window) [28]. V hodinách NixieClock je nastaveno výchozí kalibrační okénko 32 sekundy [2]. Při ideální frekvenci RTC 32768 Hz tedy dojde k úpravě během 2^{20} taktů ($32768 \cdot 32 = 2^{20}$).

Pro nastavení množství skrytých či přidaných taktů slouží 2 registry – CALP a CALM. Hodnota v registru CALM odpovídá za skrytí zadaného množství taktů. CALM registr tedy nabývá hodnoty od 0 až do 511. Naopak hodnota v registru CALP odpovídá za přidání fixního množství 512 taktů. Pro přidání těchto taktů je třeba CALP registr nastavit na hodnotu 1. V opačném případě by měl obsahovat 0. [29]

1.4.1 Výpočet hodnot pro kalibraci

Vzorec pro výpočet frekvence RTC po kalibraci (f_{CAL}) pomocí hodnoty registru CALP (p) a registru CALM (m) při dané vstupní frekvenci (f_{RTC}) je následující [28]:

$$f_{CAL} = f_{RTC} \cdot \left(1 + \frac{p \cdot 512 - m}{2^{20} + m - p \cdot 512}\right)$$

Nechť se na časovém úseku t sekund hodiny odchylují o x sekund. Potom platí rovnost:

$$t \cdot f_{RTC} = (t + x) \cdot 32768$$

Z rovnosti lze vyjádřit skutečnou hodnotu frekvence krystalového oscilátoru RTC (f_{RTC}) jako:

$$f_{RTC} = 32768 \cdot \left(1 + \frac{x}{t}\right)$$

Cílem je dostat frekvenci po kalibraci (f_{CAL}) rovnou 32768 Hz. Hodnotu registru CALP (p) je třeba pro zpomalené hodiny nastavit na 1, pro zrychlené na 0. Požadovanou hodnotu registru CALM (m) lze pak z výše uvedených vzorců získat následovně:

$$m = 2^{20} \cdot \frac{x}{t} + p \cdot 512$$

1.5 Protokol NTP

Počítačové hodiny typu RTC nejsou zcela přesné [30]. K jejich synchronizaci slouží protokol NTP. Daný protokol vyžaduje přenos času z jednoho systému do druhého. K tomu se nejčastěji používá

síťový protokol UDP na portu 123. Při synchronizaci si klient a server neposílají přímo místní čas, ale převádí ho na koordinovaný světový čas (UTC). [30]

V rámci protokolu NTP je definována hierarchie serverů. Každý server v této hierarchii se nachází v určité vrstvě (anglicky stratum). Každá vrstva je identifikována číslem, které určuje její pořadí od referenčního zdroje přesného času. Tímto zdrojem mohou být například atomové hodiny. Referenční zdroj času je napojen přímo na server první vrstvy (stratum 1). Server vyšší vrstvy se synchronizuje se serverem bezprostředně předcházející vrstvy právě pomocí protokolu NTP. To znamená, že server druhé vrstvy (stratum 2) se synchronizuje se serverem první vrstvy (stratum 1) a tak dále. Důsledkem dané posloupnosti synchronizací je snižující se přesnost času na serverech vzdálenějších vrstev. To znamená, že čím vyšší je číslo vrstvy serveru, tím méně přesný je jeho čas. Servery jsou označeny jako nesynchronizované, pokud leží ve vrstvě 16 nebo vyšší. Řada organizací poskytuje přístup k veřejným NTP serverům. [30]

Protokol SNTP je zjednodušenou implementací protokolu NTP. Používá se v zařízeních a programech, které nevyžadují vysokou přesnost času. Oba protokoly používají pro přenos dat mezi serverem a klientem identický formát. Hlavní rozdíl je pouze ve způsobu zpracování přenesených dat. Klient SNTP tedy může synchronizovat čas s libovolným NTP serverem tak, jako by to byl server SNTP. [31]

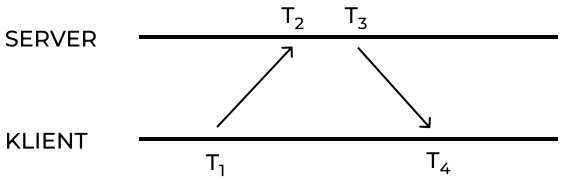
1.5.1 Popis algoritmu synchronizace

Klient pravidelně posílá požadavek na jeden nebo více serverů. Algoritmus pro synchronizaci, znázorněný na obrázku 1.4, probíhá v následujících krocích [30]:

1. Klient pošle svůj čas T_1 na server
2. Server dostane požadavek od klienta v čase T_2
3. Server dokončí jiné procesy, které na něm v danou chvíli běží
4. Server pošle klientovi v čase T_3 odpověď, která zahrnuje časy T_1, T_2 a T_3
5. Klient obdrží odpověď v čase T_4
6. Klient vypočítá časovou odchylku θ podle následujícího vzorce:

$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

V případě NTP se tyto kroky několikrát opakují. Dále následuje proces filtrování a statistické analýzy na základě získaných hodnot a údajů předchozích synchronizací. V případě SNTP je tento proces zjednodušen nebo vynechán [31].



Obrázek 1.4 Klient-server komunikace v protokolu NTP

1.6 Vývoj multiplatformních aplikací

Existují tři základní způsoby vývoje multiplatformních aplikací – nativní, hybridní a webový. Jednotlivé přístupy využívají různých technologií. Aplikace vytvořené těmito způsoby se navzájem odlišují hlavně v podporovaných platformách, funkcích a výkonu. Volba tedy závisí především na typu konkrétní aplikace a jejích funkcích. [32]

1.6.1 Nativní vývoj

Nativní vývoj vyžaduje znalost technologií a programovacích jazyků specifických pro cílovou platformu [32]. Pro podporu více platforem je tedy nutné vytvořit každou aplikaci zvlášť. V případě mobilních aplikací jde o dva nejpoužívanější operační systémy – Android a iOS. U těchto operačních systémů je třeba určit i podmínky pro publikaci do obchodů s aplikacemi. V případě desktopových aplikací jde o Windows, macOS a Linux.

Hlavní výhodou nativního vývoje je možnost lepší optimalizace pro jednu platformu. Výsledná aplikace má například rychlejší odezvu a zvládne vykreslování složitější grafiky. Navíc lze použít i takové komponenty uživatelského rozhraní a takové funkce, které jsou přístupné pouze pro danou platformu. [32]

1.6.1.1 Android

Platforma podporuje jazyky – Java a Kotlin. V roce 2019 byl Kotlin označen společností Google jako preferovaný jazyk pro vývoj Android aplikací. [33]

Při tvorbě aplikace se musí zvolit minimální a cílová verze operačního systému Android. Minimální verze je nejnižší podporovanou verzí, na které je aplikace spustitelná. Cílová verze určuje, pro kterou verzi je aplikace optimalizována. To znamená, že pokud v nové verzi Android vyjde nová funkce, která není kompatibilní s cílovou verzí, tak se v nové verzi použije starý způsob implementace dané funkce. [34]

Od 1. srpna 2022 musí nové aplikace cílit na verzi Android 12 (API 31) nebo vyšší [35]. Tato podmínka je zásadní pro možnost publikace aplikace do obchodu Google Play. Podpora BLE začíná od verze Android 4.3 (API 18) [11]. Minimální verze tedy nesmí být starší než Android 4.3.

1.6.1.2 iOS

Platforma podporuje jazyky – Objective-C a Swift. Jazyk Swift je vyvíjen společností Apple jako modernější alternativa k jazyku Objective-C. [36]

Při tvorbě aplikace se musí zvolit minimální verze operačního systému iOS. Pro publikaci do obchodu s aplikacemi App Store není žádné omezení související s minimální verzí. Od 25. dubna 2022 musí být aplikace pro iOS, iPadOS a watchOS pro publikaci do App Store sestaveny pomocí IDE Xcode 13 [37]. Podpora BLE začíná od verze iOS 5.0 [12]. Minimální verze tedy nesmí být starší než iOS 5.0.

1.6.2 Hybridní vývoj

Hybridní vývoj je postaven na společném rozhraní pro přístup k nativním funkcím jednotlivých platforem. Toto rozhraní poskytují hybridní frameworky. Implementace aplikace se provádí pomocí jediné technologie a programovacího jazyka. Výstupem jsou nativní aplikace pro jednotlivé platformy. [38]

Moderní frameworky nabízí 2 způsoby vývoje uživatelského rozhraní hybridních aplikací. První je založen na webových technologiích. Výsledná aplikace pak běží v nativní komponentě,

která dokáže načíst webovou stránku. Druhý způsob je založen na překladu univerzálních komponent do nativních. Výsledné nativní aplikace lze publikovat v obchodech s aplikacemi. [39]

Mezi výhody hybridního vývoje patří možnost sdílet komponenty uživatelského rozhraní, aplikační logiku a testy napříč platformami. Vývoj je tedy rychlejší a levnější. Mezi nevýhody se řadí pomalejší odezva aplikace a závislost vývoje na možnostech a omezeních hybridního frameworku. [38]

1.6.3 Webový vývoj

Pro webový vývoj se používá HTML, CSS a JavaScript. Výsledkem vývoje je webová aplikace, která se spouští ve webovém prohlížeči. Funkce takové aplikace tedy přímo závisí na možnostech prohlížeče. Moderní podobou webových aplikací jsou SPA a PWA. [40]

1.6.3.1 Single Page Application

SPA představuje aplikaci, která zobrazuje veškerý svůj obsah na jediné stránce. Při spuštění takové aplikace se načítá pouze jeden HTML dokument. Aktualizace obsahu tohoto dokumentu se poté provádí prostřednictvím jazyka JavaScript. Tento přístup odstraňuje nutnost načítání celých nových stránek ze serveru. [40]

1.6.3.2 Progressive Web Application

PWA je webová aplikace, která napodobuje nativní aplikaci. PWA může být zároveň i SPA aplikací. Základem PWA jsou technologie jako Manifest a Service Worker. [40]

Manifest webové aplikace je JSON soubor, který specifikuje vlastnosti PWA po instalaci. Tento soubor může obsahovat například název aplikace, ikonky aplikace nebo URL adresu, která by se měla otevřít při spuštění aplikace. [41]

Pro spuštění kódu, psaného v jazyce JavaScript, ve více vláknech v rámci prohlížeče slouží technologie Worker. Worker má 2 základní typy – Web Worker a Service Worker. Žádný z nich nemá přístup k modifikaci DOM modelu a oba mají omezený přístup k některým JavaScript API prohlížeče. Například není dostupné Web Bluetooth API. Web Worker může mít v rámci jedné aplikace několik instancí. Po zavření okna prohlížeče se procesy, které běží v rámci Web Worker, ukončí. [42]

Service Worker tvoří základ PWA. Oproti technologii Web Worker má určité rozdíly. V rámci jedné aplikace může existovat pouze jedna aktivní instance. Service Worker může běžet i po zavření okna prohlížeče a vykonávat různé operace na pozadí. Může například zpracovávat push notifikace, synchronizovat data aplikace či provádět aktualizace. Dále se chová jako síťový proxy a zajišťuje offline funkčnost. To znamená, že umožňuje zakešovat data a soubory aplikace, zachytit síťové požadavky a vrátit výsledek požadavku z keše. [42]

Dle [43] by měla PWA splňovat následující vlastnosti:

- **Instalovatelná a aktualizovatelná**

  Instalaci lze provést přímo pomocí webového prohlížeče. Aplikaci tedy není nutné publikovat do obchodů s aplikacemi. Aktualizaci provádí Service Worker na pozadí.

- **Přístupná Offline**

  PWA není závislá na připojení a zajišťuje tím offline funkčnost. To znamená, že je spustitelná a použitelná i bez připojení k internetu.

- **Progresivní a responzivní**

  Uživatelské rozhraní napodobuje nativní. Aplikace se umí přizpůsobit různým velikostem obrazovek.

- **Dohledatelná a dostupná**

  Pomocí souboru manifest.json určuje prohlížeči, že jde o PWA. Aplikace je dostupná ve webovém prohlížeči pomocí URL adresy. Instalace tedy není nutná.

- **Bezpečná**

  Aplikace používá zabezpečené spojení pomocí HTTPS.

Klíčovou vlastností pro PWA je instalovatelnost. Tato možnost je podporována pouze v některých prohlížečích a operačních systémech. Prohlížeče Google Chrome (od verze 73) a Microsoft Edge (od verze 79) zajišťují plnou podporu PWA pro Android a desktopové operační systémy – Windows, macOS, Linux a Chrome OS. Pro operační systém Android nabízí plnou podporu PWA prohlížeč Samsung Internet (od verze 18). Pro operační systémy iOS a ipadOS lze instalovat PWA pouze pomocí prohlížeče Safari (od verze 11.3). [44]

Safari na operačním systému iOS má omezenou podporu pro PWA. Například při otevření aplikace v okně prohlížeče nenabízí uživateli dialog s možností instalace. Uživatel musí pro instalaci otevřít menu prohlížeče. Safari nepodporuje Web Push notifikace a synchronizaci na pozadí. Dále umožňuje použití pouze některých vlastností souboru Manifest. Některé chybějící vlastnosti lze doplnit pomocí speciálních meta značek v HTML dokumentu. To zahrnuje například značku 'apple-touch-startup-image', která nastavuje ikonku zobrazovanou při otevření aplikace na úvodní obrazovce (splash screen). V novějších verzích prohlížeče Safari a nových verzích iOS se stále přidávají chybějící vlastnosti, které nahrazují existující meta značky. [44]

Aplikaci PWA nelze přímo publikovat do obchodů s aplikacemi, ale je nutné aplikaci zabalit do nativní komponenty. V případě OS Android je třeba pro publikaci do Google Play Store zabalit PWA do nativní komponenty TWA. Pro tento proces nabízí společnost Google nástroj Bubblewrap, který vše provede automaticky. V případě OS iOS je také třeba pro publikaci do App Store zabalit PWA do nativní komponenty. Společnost Apple však nenabízí pro tento proces žádné nástroje. [44]

Mezi hlavní výhody PWA patří to, že představuje multiplatformní řešení, nabízí přístup i instalaci přes prohlížeč, zvládá automatické aktualizace na pozadí a zabírá méně paměti než nativní aplikace. Mnoho společností, jako například Twitter nebo AlliExpress, zaznamenalo po přechodu z nativní aplikace na PWA vyšší počet návštěv či větší konverzi uživatelů [45]. Hlavní nevýhodou je omezení možností aplikace, která může mít přístup pouze k rozhraním poskytovaným prohlížečem. Navíc zatím nelze zajistit plnou podporu pro operační systém iOS. [44]

1.7 Stávající Android aplikace

Zdrojové kódy stávající Android aplikace jsou zveřejněné na [46]. Aplikace je implementována v jazyce Kotlin. Minimální podporovanou verzí je Android 6.0 (API 23). Na moment tvorby aplikace daná minimální verze zajistila pokrytí nejméně 85 % zařízení s operačním systémem Android. Cílovou verzí je Android 11 (API level 30). Tato verze však již nesplňuje podmínky pro publikaci do obchodu Google Play Store. [3]

Uživatelské rozhraní je realizováno pomocí XML a jsou použity Material Design komponenty. Jde o hotové nastavitelné stavební bloky aplikace, splňující zásady Material Design od společnosti Google. Například výběr data a času se realizuje pomocí komponent Date Picker a Time Picker (viz příloha C). Jazyk a barevné schéma aplikace se umí přizpůsobit systému. Popis uživatelského rozhraní této aplikace, včetně snímků všech obrazovek, je součástí bakalářské práce o tvorbě dané Android aplikace. Tato práce je publikována na [3]. Autor navíc zveřejnil videoukázku použití aplikace [47]. Z těchto důvodů není třeba popisovat všechny použité UI komponenty, ale je vhodné aplikaci otestovat a určit, zda odpovídá popisu a neobsahuje chyby či nedostatky.

Architektura aplikace je MVVM, což je zkratka pro Model, View a ViewModel. Jde o třívrstvou architekturu. View je prezentační vrstvou aplikace. Model reprezentuje interní a externí zdroj dat. ViewModel propojuje View a Model tak, že umožňuje obousměrný tok dat. Pro komunikaci

s hodinami přes BLE byly implementovány servisní třídy, které jsou z hlediska architektury součástí modelu. Dané servisní třídy nemají definované rozhraní (interface). To může ztížit případné budoucí změny implementace. Není použita žádná knihovna pro vkládání závislostí (dependency injection). Instance servisních tříd jsou vytvářeny voláním konstruktoru přímo v jednotlivých třídách typu ViewModel.

V souboru Manifest.xml jsou uvedena oprávnění (permissions) aplikace. Pro komunikaci přes protokoly SNTP a BLE jsou definována oprávnění pro internet a Bluetooth. Není však jasný důvod použití oprávnění pro nahrávku zvuku 'android.permission.RECORD_AUDIO'. GATT klient je implementován pomocí knihovny Kable. Tato knihovna poskytuje multiplatformní rozhraní pro interakci se zařízeními Bluetooth Low Energy. Mezi podporované platformy knihovny patří Android, iOS, macOS a web (Web Bluetooth API) [48]. Do implementace je také zahrnutý open source SNTP klient [49]. Tento klient je napsaný v jazyce Java. Jako NTP server je použít time.google.com.

Pro testování aplikace jsou použity lokální jednotkové (Unit) testy, napsané pomocí knihovny JUnit. Testovány jsou funkce pro konverzi dat z interní reprezentace do formátu posílaného přes BLE a naopak. Testy pokrývající funkce souboru 'ByteArrayConverters.kt' nejsou však funkční. Tyto funkce jsou implementovány jako rozšiřující funkce (extension functions) konvertovaných datových typů. Testovány jsou však tak, jako kdyby se jednalo o statické metody, které jako argument přijímají hodnotu konvertovaného datového typu. Po úpravě volání těchto funkcí všechny testy prochází úspěšně. Autor aplikace pravděpodobně po změně jejich implementace zapomněl upravit odpovídající testy.

1.7.1 Testování stávající aplikace

Původní aplikace byla otestována při tvorbě firmwaru a bylo zjištěno nekorektní chování některých funkcí. Při ručním nastavení data aplikace nastavuje měsíc na hodnotu o 1 menší. Aplikace také neodesílá nové hodnoty charakteristik ve službě nastavení (settings service), ale přitom tyto nové hodnoty zobrazuje. Tlačítko zobrazující stav stopek se po změně hodnoty neaktualizuje správně. Uživatel musí přepnout na jinou obrazovku a zpět, aby si vynutil aktualizaci. Obrazovka časovače má podobný problém. [2]

Aplikace byla vytvořena v době, kdy autor neměl k dispozici samotné hodiny. Z tohoto důvodu pro účely testování vytvořil vlastní emulátor hodin. Filtrování zařízení při skenování realizoval na základě MAC adresy tohoto emulátoru. Při testování s hodinami NixieClock však aplikace kvůli filtrování nemohla vyhledat žádné zařízení. Dle specifikace [2] vysílají hodiny NixieClock v discoverable advertisement paketech zkrácený seznam 128bitových UUID, obsahující pouze UUID služby hodin (Clock Service). Filtr dle MAC adresy byl tedy vyměněn na filtr na základě UUID. Po úpravě filtrování již bylo možné aplikaci otestovat na stávajícím firmwaru.

Pro odhalení nedostatků existujícího uživatelského rozhraní bylo provedeno testování aplikace, kterého se zúčastnilo 5 osob. Důvodem účasti více osob byla potřeba získat objektivní pohled na UI. Testování proběhlo s každým účastníkem individuálně. Aplikace byla vyzkoušena na následujících zařízeních a verzích operačního systému Android:

- Xiaomi Mi 9T Pro (Android 11)
- Samsung Galaxy A40 (Android 11)
- Samsung Galaxy S8 (Android 9) – 3 účastníci

Úkolem účastníků bylo postupně si vyzkoušet všechny možnosti, které aplikace nabízí. Cílem účastníka bylo například připojit se k hodinám, zapnout budík na požadovaný čas, vypnout zobrazení data a podobně. Na základě testování bylo potvrzeno výše popsané nekorektní chování aplikace. Dále byly zjištěny nedostatky uživatelského rozhraní, popsané v sekci 1.7.1.1.

1.7.1.1 Zjištěné nedostatky stávající aplikace

Během testování byly zjištěny následující nedostatky uživatelského rozhraní stávající aplikace:

- **Připojení**

  Po zapnutí aplikace se objevuje obrazovka s tlačítkem pro skenování zařízení. Toto tlačítko není výrazné a nachází se v pravém dolním rohu obrazovky. Z uživatelského hlediska není hned zřejmé, co je třeba dělat po zapnutí aplikace. Jedná se tedy o nevhodné umístění tlačítka.

- **Budíky**

  Aplikace zobrazuje všech 10 budíků na 1 obrazovce (viz obrázek 1.5b). U každého budíku zobrazuje jeho pořadí, čas zvonění a jeho stav. Číslování budíků je však redundantní. Neurčuje totiž pro uživatele žádnou užitečnou informaci. Budíky se načítají pomalu, po jednom a bez žádné animace, což působí zastarale a neodpovídá moderním přístupům, na které jsou uživatelé zvyklí.

- **Časovač**

  Pro zadání startovacího času časovače je použito pole formuláře (viz obrázek 1.5c). Z uživatelského hlediska však není jasný formát požadované hodnoty. Při zadání nevalidní hodnoty aplikace též neupozorní na nevalidní formát a časovač se nespustí. Dle videoukázky dané aplikace je akceptován formát 'HH:MM:SS'. Bohužel některé telefony na numerické klávesnici neumožňují zadat znak ':' (viz obrázek 1.5d). Z tohoto důvodu je možné na takových zařízeních zadat pouze 1 číslo v rozmezí 1–59, které reprezentuje sekundy. Po spuštění časovače se klávesnice neskrývá automaticky (viz obrázek 1.5e).

- **Nastavení**

  Hodnoty v nastavení, zadávané přes formulářové pole, nemají jasně určený formát (viz obrázek 1.5f). Jednotka zadávané hodnoty též není určena. Dle popisu jde o celočíselné hodnoty v milisekundách. Uživatel však není schopen danou informaci dohledat přímo v aplikaci. Navíc je například zadání délky zvonění v milisekundách pro uživatele nepraktické. Aktuálně nastavené hodnoty nejsou zobrazeny přímo v nastavení jednotlivých vlastností. Pro zjištění těchto hodnot je potřeba otevřít editaci příslušné vlastnosti, kde se její hodnota zobrazí v poli formuláře.

- **Načítání a notifikace**

  Aplikace neindikuje stav načítání. Z uživatelského hlediska není jasné, zda se požadavek načítá, nebo skončil. Aplikace neurčuje stav úspěchu či neúspěchu provedené akce. Aplikace neupozorňuje na chyby v případě zadání nevalidních hodnot.

- **Ukončení spojení**

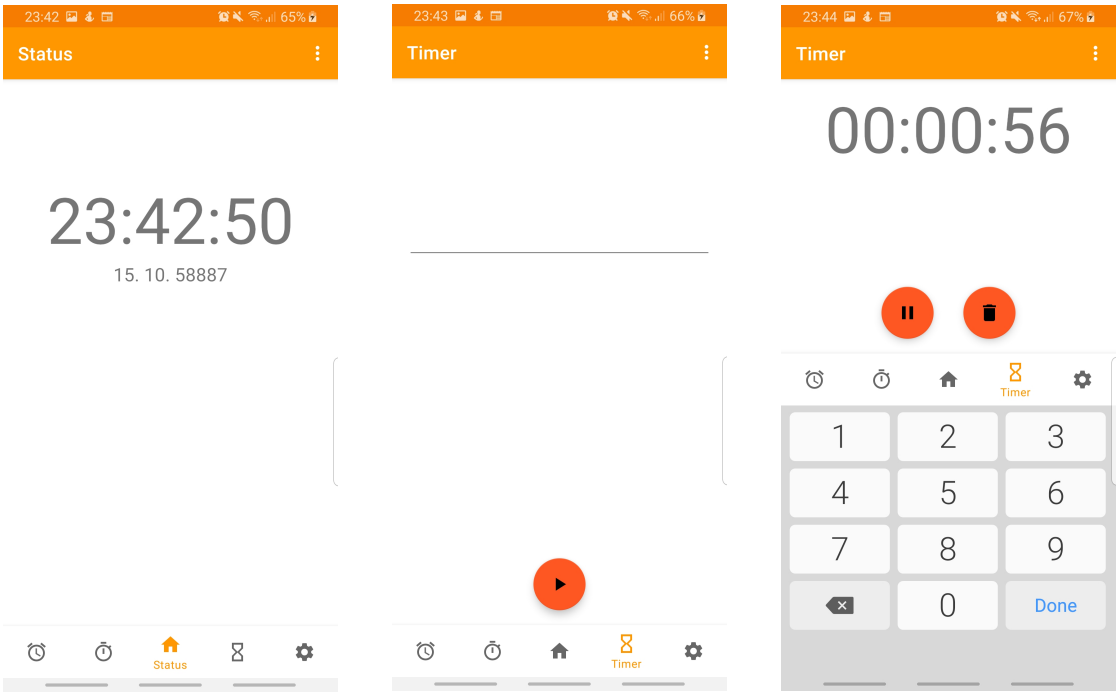
  Ukončení spojení může být vyvoláno například vypnutím hodin nebo větší vzdáleností mezi hodinami a mobilním telefonem. Po ukončení spojení aplikace po krátké době reaguje přepnutím na stránku skenování. V případě provedení určité akce před reakcí aplikace na odpojení, došlo několikrát k tomu, že mobilní aplikace spadla. Toto chování lze vysvětlit tím, že některé funkce aplikace, zajišťující BLE komunikaci, nemají ošetřené výjimky.

1.7.2 Splnění aktuálních požadavků

Stávající aplikace obsahuje nedostatky uživatelského rozhraní, zjištěné v průběhu testování. Implementace není v souladu s aktuálními funkčními i nefunkčními požadavky v následujících bodech:

- Aplikace podporuje pouze operační systém Android. (N1)

- Cílová verze Android již nesplňuje podmínky pro publikaci do obchodu Play Store. (N1)
- Chybí možnost kalibrace hodin. (F6)
- Není implementováno automatické připojení bez nutnosti skenování. (N9)
- Není implementováno automatické znovupřipojení v případě ztráty spojení. (N9)
- Aplikace nereflektuje do uživatelského rozhraní stav komunikace s hodinami. (N7)
- Vlastnosti v nastavení, zadávané přes formulářové pole, se mění pouze lokálně a požadavek se neodesílá na hodiny NixieClock. (částečně F2–F5)
- Je vyžadována úprava jednotkových testů metod servisních tříd. (N11)
- Nelze zvolit jazyk a barevný režim v nastavení aplikace. (N6)



(a) Datum a čas **(c)** Časovač **(e)** Časovač – po spuštění

(b) Budíky **(d)** Časovač – zadání času **(f)** Nastavení – formulář editace

Obrázek 1.5 Snímky Android aplikace NixieClock [46] na telefonu – Samsung Galaxy S8

Kapitola 2
Návrh

Kapitola popisuje výběr vhodných technologií pro tvorbu mobilní aplikace NixieApp. Dále pojednává o návrhu architektury a uživatelského rozhraní dané aplikace.

2.1 Technologie pro tvorbu aplikace

Dle zadání je preferována tvorba aplikace pomocí univerzálního frameworku, proto není vhodné zvolit nativní způsob vývoje. PWA a Web Bluetooth mají omezenou podporu pro operační systém iOS. Je tedy potřeba vybrat technologii pro hybridní způsob vývoje, která zajistí podporu nejméně pro iOS a Android, případně i pro web a desktopové platformy. Výběr konkrétní technologie závisí na možnostech pro implementaci GATT klienta, způsobech pro tvorbu uživatelského rozhraní a množství podporovaných platforem.

2.1.1 Technologie pro hybridní vývoj

Tato sekce obsahuje popis nejpoužívanějších technologií pro hybridní vývoj, které podporují tvorbu GATT klienta. Výběr se realizuje na základě množství podporovaných platforem a možnostech pro tvorbu UI. To zahrnuje jak existenci hotových UI komponent, tak i tvorbu vlastních komponent.

2.1.1.1 .NET MAUI

.NET MAUI je multiplatformní framework pro tvorbu nativních mobilních a desktopových aplikací pomocí jazyka C#. Framework je produktem společnosti Microsoft. Podporuje vývoj pro Android, iOS, macOS a Windows. .NET MAUI umožňuje vytvářet aplikace pro více platforem pomocí jednoho projektu, ale v případě potřeby lze přidat zdrojový kód specifický pro konkrétní platformu. [50]

Framework nabízí jednotné rozhraní pro funkce a uživatelské rozhraní jednotlivých platforem. UI komponenty se tvoří pomocí jazyka XAML, který má syntaxi podobnou HTML. Tyto komponenty jsou univerzální a transformují se na nativní UI komponenty jednotlivých platforem. [50]

2.1.1.2 React Native

React Native je multiplatformní framework, vytvořený společností Facebook. Pro vývoj umožňuje použít JavaScript i TypeScript. Podporuje tvorbu aplikací pro Android, iOS, Windows,

macOS a web. Tento framework také poskytuje možnost psát nativní kód pro jednotlivé platformy. Umožňuje tvorbu univerzálních komponent a nabízí řadu hotových rozhraní pro přístup k nativním funkcím jednotlivých platforem. [51]

2.1.1.3 Flutter

Flutter je produktem společnosti Google. Jde o framework pro tvorbu multiplatformních aplikací v jazyce Dart. Používá se na vývoj aplikací pro Android, iOS, Windows, MacOS, Linux a web. Funguje na stejném principu jako React Native a .NET MAUI. [52]

2.1.1.4 Apache Cordova

Apache Cordova je nástroj, který dokáže zabalit webovou aplikaci do nativního kontejneru, který má přístup k funkcím zařízení na několika platformách. Tyto funkce jsou zpřístupněny prostřednictvím jednotného rozhraní, které umožňuje snadno napsat jednu sadu kódu pro téměř každý telefon nebo tablet na současném trhu. Konkrétně jde o podporu operačních systémů – Android a iOS. Toto jednotné rozhraní je implementováno v jazyce JavaScript s možností použít TypeScript. [53]

Tento nástroj vyžaduje existenci hotové webové aplikace, vytvořené pomocí webových technologií – JavaScript, HTML a CSS. Rozšiřuje tedy možnosti tvorby uživatelského rozhraní aplikace o řadu existujících webových knihoven a frameworků. Díky tomu umožňuje v rámci jednoho řešení zahrnout i PWA. [53]

2.1.1.5 CapacitorJS

CapacitorJS je obdobou nástroje Apache Cordova. Též umožňuje zabalit webovou aplikaci do nativního kontejneru pro operační systémy iOS a Android. Také poskytuje jednotné rozhraní pro jednotlivé platformy v jazyce JavaScript s podporou pro TypeScript. Navíc má oficiální podporu pro většinu moderních webových API, potřebných pro tvorbu PWA. Mezi tyto API patří i Web Bluetooth API. CapacitorJS nabízí dokonce kompatibilitu s většinou pluginů vytvořených pro Apache Cordova. [54]

Pro tvorbu webové aplikace poskytují vývojáři tohoto nástroje hotové UI komponenty v rámci produktu Ionic Framework. Použití tohoto frameworku však není nutné a pro tvorbu PWA lze použít libovolnou webovou technologii. [54]

2.1.1.6 Zvolená technologie

Všechny uvedené hybridní frameworky nabízí možnost tvorby GATT klienta, použití hotových UI komponent či tvorbu vlastních. Dle tabulky 2.1 zajistí nejlepší podporu technologie – Cordova a CapacitorJS. Volbou v tomto případě je CapacitorJS, protože jde o modernější alternativu s podporou pro většinu moderních webových API. Navíc je zpětně kompatibilní s většinou nástrojů vytvořených pro technologii Cordova. Oproti alternativám nabízí navíc podporu pro web, který pokrývá i desktopové operační systémy – Windows, macOS a Linux. Dále umožní plnou podporu webové verze pro Android v podobě PWA a částečnou podporu pro iOS prostřednictvím prohlížeče Bluefy. Též zajistí podporu pro Android a iOS v podobě nativních aplikací. Nativní Android aplikace bude přístupná pro verzi Android 5.1 (API 22) a novější [55]. To zajistí podporu více než 95 % Android zařízení na trhu [55]. Nativní iOS aplikace bude přístupná pro verzi iOS 13 a novější. To taktéž zajistí podporu více než 95 % iOS zařízení na trhu [56]. Daná volba hybridní technologie tedy splňuje nefunkční požadavek N1 o podporovaných platformách (viz sekce 1.1.2).

Cílem knihovny 'bluetooth-le' je podporovat stejné funkce na všech platformách. Poskytuje API, které vychází primárně z možností Web Bluetooth API. To může zjednodušit případný budoucí přechod na webové API. Případně díky zpětné kompatibilitě s technologií Cordova lze

použít i knihovnu 'cordova-plugin-ble-central'. Tato knihovna nabízí větší množství funkcí, které jsou ovšem specifické pro konkrétní platformu. Například pro Android umožňuje poslat MTU Request či pracovat s L2CAP komunikačním kanálem. Z komunikačního protokolu (viz příloha A) však plyne, že si aplikace NixieApp vystačí se základními GATT operacemi, které knihovna 'bluetooth-le' podporuje na všech platformách.

V případě plné budoucí podpory technologie Web Bluetooth ve všech operačních systémech, umožní daná volba hybridního frameworku jednoduchý přechod pouze na webovou platformu. Také potenciálně zjednoduší budoucí vývoj, jelikož nebude nutné řešit nativní prostředí iOS a Android. Vývojář si díky tomu vystačí se znalostí webových technologií. Rozhraní Web Bluetooth API je stále ve vývoji a je označováno jako experimentální, a tedy je třeba počítat i s negativním scénářem. V případě odstranění tohoto rozhraní z prohlížečů, bude stále možné podporovat iOS a Android nativně díky technologii CapacitorJS.

Framework	GATT Knihovna	Android	iOS	Web	Windows	macOS	Linux
.NET MAUI	BluetoothLE [57]	+	+	-	-	-	-
React Native	react-native-ble-plx [58]	+	+	-	-	-	-
	react-native-ble-manager [59]	+	+	-	-	-	-
Flutter	Flutter_blue [60]	+	+	-	-	+	-
	Flutter_BLE_lib [61]	+	+	-	-	-	-
	Flutter_reactive_BLE [62]	+	+	-	-	-	-
	flutter_web_bluetooth [63]	-	-	+	-	-	-
Cordova	cordova-plugin-ble-central [64]	+	+	+	-	-	-
CapacitorJS	bluetooth-le [65]	+	+	+	-	-	-

Tabulka 2.1 Hybridní frameworky – podpora BLE

2.1.2 Technologie pro tvorbu PWA

Jako technologie pro hybridní vývoj byl zvolen nástroj CapacitorJS. Tento nástroj poskytuje rozhraní pro komunikaci přes BLE, specifické pro jednotlivé platformy. Umožňuje zabalit již hotovou webovou aplikaci do nativního kontejneru. Nenabízí však možnosti pro tvorbu uživatelského rozhraní samotné webové aplikace. Tato sekce se tedy zabývá výběrem technologií pro tvorbu webové aplikace a knihovny s hotovými UI komponentami. Výběr ovlivní podpora pro tvorbu PWA. Dalším faktorem výběru je existence a možnosti hotových UI komponent pro výběr data a času. Je vhodné, aby tyto komponenty byly optimalizovány jak pro mobilní, tak i pro desktopová zařízení.

2.1.2.1 Ionic Framework

Ionic Framework nabízí řadu hotových komponent uživatelského rozhraní. Neposkytuje ovšem prostředky pro tvorbu aplikační logiky, uložení stavu aplikace a zpracování interakce uživatele. Hotové komponenty lze použít rovnou v JavaScript kódu bez použití webového frameworku, nebo také jsou oficiálně podporovány populární webové frameworky a knihovny – Angular, Vue

a React. Dané nástroje zjednodušují tvorbu vlastních UI komponent a nabízí způsoby pro zpracování interakce uživatele. Dále poskytují možnosti pro implementaci aplikační logiky a určují architekturu aplikace, čímž usnadňují její tvorbu. V rámci Ionic Framework existuje komponenta pro výběr data a času. Tato komponenta je však optimalizována hlavně pro dotykové obrazovky mobilních telefonů. Je tedy vhodné najít alternativní varianty UI knihoven s hotovými komponentami. [66]

2.1.2.2 Angular

Angular je frontend framework vytvořený společností Google. Je postaven na jazyce TypeScript. Stejně jako většina moderních nástrojů pro tvorbu UI uplatňuje koncept znovupoužitelných komponent. Komponenta má oddělenou prezentační a funkční část. Angular staví na principech OOP a poskytuje nástroje pro DI. Používá obousměrný tok dat (anglicky two-way binding) mezi datovou a UI vrstvou. Tuto funkcionalitu zajišťuje knihovna RxJS, která poskytuje implementaci pro reaktivní programování. Angular se hodí pro větší aplikace, které vyžadují škálovatelnost. Angular má podporu pro tvorbu PWA aplikací. Mezi nejpoužívanější UI knihovny, podporující Angular, patří – Angular Material, PrimeNG a Ng Bootstrap. [67]

2.1.2.3 React

React je knihovna pro tvorbu uživatelského rozhraní vytvořená společností Facebook. Podporuje jazyk JavaScript i Typescript. React staví na principech funkcionálního programování. Vykreslování UI aplikace optimalizuje pomocí virtuálního DOM modelu. Srovnává předchozí a aktuální stav všech komponent a aktualizuje pouze ty elementy skutečného DOM modelu, které byly změněny. Funkcionální komponenta v knihovně React je čistá funkce, která přijímá parametry a vrací JSX kód. JSX je rozšířením jazyka JavaScript, které umožňuje psát komponenty v podobné syntaxi jako HTML značky. React se nejčastěji používá v kombinaci s knihovnou Redux, která slouží k implementaci aplikační logiky a komunikaci s datovou vrstvou. Tato kombinace uplatňuje principy architektury FLUX, jejíž podstatou je jednosměrný tok dat (anglicky unidirectional data flow) mezi datovou a prezentační vrstvou. React má podporu pro tvorbu PWA aplikací. Mezi nejpoužívanější UI knihovny, vytvořené pro knihovnu React, patří – Material UI, Ant Design a React Bootstrap. [68]

2.1.2.4 Vue

Vue je frontend framework vytvořený vývojářem jménem Evan You. Podporuje jazyk JavaScript i TypeScript. Podobně jako React, používá virtuální DOM model. Vue podporuje jak JSX, tak i čisté HTML. Mezi datovou a prezentační vrstvou používá obousměrný tok dat. Součástí frameworku je knihovna Vuex, která slouží pro správu stavu aplikace (anglicky State Management). Vue má podporu pro tvorbu PWA aplikací. Mezi nejpoužívanější UI knihovny, podporující Vue, patří – BootstrapVue, Quasar, Vuetify a Vue Material. [69]

2.1.2.5 TypeScript

TypeScript je objektově orientovaný programovací jazyk vyvinutý společností Microsoft Corporation. Jde o nadmnožinu jazyka JavaScript. Kód psaný v jazyce TypeScript nemohou prohlížeče přímo interpretovat, proto se překládá do jazyka JavaScript. Tento proces je známý jako transpilace. Mezi výhody tohoto jazyka patří statická typizace, existence rozhraní (anglicky interface) a interoperabilita s jazykem JavaScript. Usnadňuje vývoj tím, že upozorňuje na chyby kompilace. Snižuje tedy pravděpodobnost chyb za běhu. [70]

2.1.2.6 Zvolená technologie

Angular, React i Vue podporují možnosti pro tvorbu PWA aplikací. Ze seznamu pro výběr však lze rovnou vyloučit Angular, jelikož se hodí zejména pro velké aplikace, vyžadující škálovatelnost, což není případ aplikace NixieApp. Pro React a Vue existují knihovny, které obsahují hotové responzivní komponenty pro výběr data a času, optimalizované pro dotykové obrazovky i desktopová zařízení. V případě knihovny React, jde o knihovnu Material UI. V případě frameworku Vue, jde o knihovny – Quasar a Vuetify. Mezi výhody knihovny React oproti Vue patří podpora velké společnosti a vyšší aktivita na StackOverflow [71], což může zjednodušit řešení případných problémů při vývoji aplikace. Z těchto důvodů je volbou React.

V kombinaci s knihovnou React je vhodné použít i Redux. Důvodem je možnost jednoznačným způsobem rozdělit aplikační a prezentační vrstvu. Tato kombinace navíc poskytne navazujícím vývojářům známou a časem prověřenou architekturu aplikace s množstvím dostupné dokumentace, což může usnadnit orientaci v projektu a urychlit práci. Jako programovací jazyk je zvolen TypeScript, jelikož má od knihovny React oficiální podporu a oproti jazyku JavaScript nabízí řadu výhod (viz sekce 2.1.2.5).

2.2 Architektura aplikace

React umožňuje vytvářet aplikace i bez použití knihovny Redux. Každá komponenta pak interně spravuje svůj vlastní stav a oznamuje změny potomkům přes parametry, nebo rodičovským komponentám přes poskytnutou callback funkci. Sdílení stavu mezi více komponentami však může být komplikované. Redux je právě řešením tohoto problému.

Pro tvorbu webových aplikací na straně klienta používá React ve spojení s Redux principy architektury Flux. Flux je architektonický vzor vytvořený společností Facebook. Jeho základním principem je jednosměrný tok dat mezi jednotlivými vrstvami aplikace. Redux ovšem některé přístupy modifikuje a implementuje vlastním způsobem [72]. Pochopení architektury, kterou poskytuje knihovna Redux, je nezbytné pro implementaci aplikace. Z tohoto důvodu je v následující sekci popsána přímo architektura, kterou nabízí knihovna Redux.

2.2.1 Redux

Redux poskytuje centralizovanou správu stavu aplikace. Zvládne přijmout a zpracovat požadavky na změnu stavu. Následně oznamuje tyto změny stavu pouze těm UI komponentám, které tato data používají. Tento proces je znázorněn na obrázku 2.1. Dle [73] jsou v rámci architektury této knihovny definovány následující části (architektonické komponenty):

- **View**

  View je uživatelské rozhraní, které zobrazuje data aplikace uložená v komponentě Store. Skládá se z UI komponent knihovny React. Knihovna React tedy reprezentuje View. View reaguje na interakce uživatele a odesílá (anglicky Dispatch) požadavky (Action) do komponenty Reducer. [73]

- **Action**

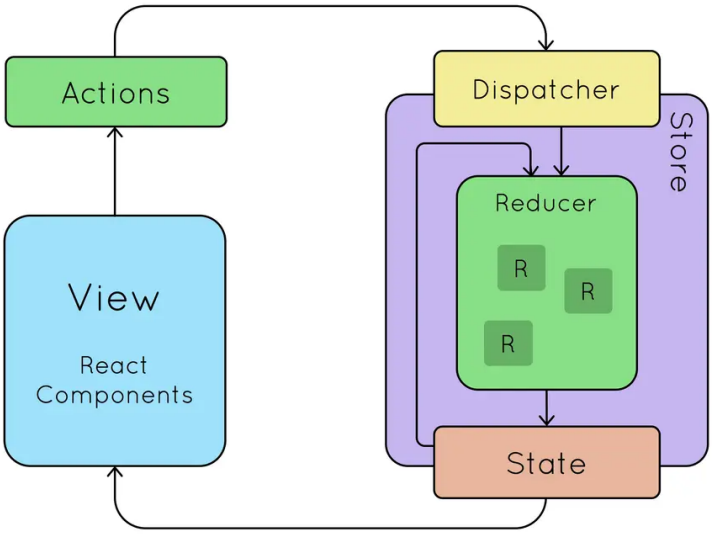
  Action je objekt obsahující 2 položky – type a payload. Type obsahuje typ akce. Payload obsahuje data potřebná k provedení daného typu akce. [73]

- **Store**

  Store je jediný zdroj pravdy (anglicky Single source of truth). Obsahuje data celé aplikace. Všechny změny v datech oznamuje těm React komponentám z View, které dané změny odebírají. [73]

- **Reducer**

  Reducer je čistá funkce (anglicky pure function), která přijme aktuální stav (anglicky State) aplikace a požadavek na provedení akce (anglicky Action). Následně zpracuje tento požadavek a vrátí nový stav aplikace. [73]



Obrázek 2.1 Architektura React/Redux [74]

2.2.2 Redux Thunk

Pro modifikaci dat, uložených v komponentě Store, slouží funkce Reducer. Jde o čistou funkci, která nesmí provádět žádné vedlejší účinky (anglicky side effect). Mezi vedlejší účinky patří například komunikace s GATT serverem či ukládání dat do lokální paměti. V takovém případě je vhodné použít Redux Thunk. [75]

Redux Thunk je middleware pro zpracování asynchronního kódu pomocí funkce nazývané thunk. Tato funkce může obsahovat jak asynchronní, tak i synchronní logiku. Dokáže interagovat s komponentou Store. Zvládne získat aktuální data uložená v komponentě Store či tato data modifikovat odesláním (Dispatch) požadavku (Action) do komponenty Reducer. [75]

Funkce typu thunk mohou obsahovat libovolnou logiku a lze je použít pro různé účely. Dle [75] patří mezi nejběžnější případy použití:

- Přesun složité logiky z React komponenty
- Tvorba asynchronní logiky
- Odeslání více požadavků pro modifikaci dat (uložených v komponentě Store)
- Je vyžadován přístup k aktuálním datům komponenty Store

Redux Thunk middleware se registruje při inicializaci komponenty Store. Při registraci lze middleware přizpůsobit přidáním 'extra' argumentu. Middleware bude poté automaticky vkládat daný argument do všech funkcí typu thunk. To se může hodit pro injektáž závislostí (anglicky Dependency Injection). [75]

2.3 Návrh uživatelského rozhraní

Figma je nástroj, který umožňuje návrh uživatelského rozhraní a tvorbu interaktivního prototypu aplikace. Výhodou tohoto nástroje je existence online i offline verze. V sekci 2.1.2.6 byla pro tvorbu UI zvolena knihovna React, pro kterou existuje knihovna s hotovými UI komponentami Material UI (MUI). Vývojáři MUI nabízí pro nástroj Figma takzvaný UI Kit, který obsahuje návrhy komponent poskytovaných pro React. Komunitní verze tohoto UI Kitu obsahuje všechny základní komponenty a je k dispozici zdarma na [76].

Díky zvolené hybridní technologii, popsané v sekci 2.1.1.6, bude aplikace podporovat Android, iOS a web. Dle nefunkčního požadavku N2 (viz sekce 1.1.2) má být aplikace optimalizována primárně pro mobilní zařízení. To znamená, že při návrhu je vhodné použít takzvaný Mobile First přístup. Jak naznačuje termín, jde o přístup, ve kterém se nejprve navrhne design aplikace pro mobilní zařízení. Poté se návrh rozšíří o zařízení s větší velikostí obrazovky, kde mohou být volitelně přidány doplňkové funkce. Mobilní telefony mají omezenou plochu pro zobrazení a ovládání. Hlavní funkce aplikace se tedy musí vejít na obrazovku a umožnit jednoduché ovládání, jak dotykem, tak i přes virtuální klávesnici. Webová verze aplikace bude funkční i na desktopových zařízeních, které nemají dotykovou obrazovku. Při návrhu UI je tedy potřeba počítat i s podporou těchto zařízení a případně nabídnout možnosti pro ovládání přes fyzickou klávesnici. Pro větší velikosti obrazovek se však aplikace téměř nezmění. Pouze dojde k roztažení některých bloků a zvětšení textů.

Před samotným návrhem je třeba určit minimální podporovanou velikost obrazovky. Nejmenší doporučenou šířkou obrazovky mobilního telefonu, kterou je třeba podporovat je 360 px [77]. Mobilní telefony s šířkou obrazovky 320 px však ke konci roku 2021 používalo 3 % uživatelů [78]. Zbývá určit, zda dané telefony podporují BLE a požadované operační systémy. Pro iOS jde o iPhone 4, iPhone 4s, iphone 5 a iPhone SE 1st gen [79]. Pro Android jde například o Galaxy S2 a Galaxy S3 mini [80]. V případě iOS tyto modely telefonů v roce 2022 již nedostávají aktualizace softwaru a počet jejich uživatelů se s každým rokem blíží 0 [81]. V případě Android výše jmenované starší modely telefonů též nedostávají aktualizace softwaru a jejich verze os již nesplňuje požadavky zvolené hybridní technologie – CapacitorJS. Z těchto důvodů bude návrh vytvořen pro minimální šířku 360 px.

Uživatelské rozhraní aplikace NixieApp vychází především z rozhraní existující aplikace pro Android, které je doplněno o nové prvky a nový design. Dále se inspiruje UI komponentami použitými v aplikacích s funkcí budíku pro mobilní telefony. Rozbor stávající Android aplikace, včetně nedostatků uživatelského rozhraní je součástí sekce 1.7. UI komponenty ostatních aplikací, které sloužily jako předloha při návrhu aplikace NixieApp, jsou k dispozici v příloze D.

2.3.1 Komponenty

Po připojení k hodinám bude struktura uživatelského rozhraní aplikace rozdělena na 2 části:

- Hlavní stránky

  Hlavní stránky budou zobrazovat data a poskytovat základní funkce pro ovládání hodin. Jde o zobrazení data a času, budíky, stopky a časovač.

- Vedlejší stránky (podstránky)

  Vedlejší stránky budou poskytovat sekundární funkce. Jde o interní nastavení samotné aplikace NixieApp a také o nastavení vlastností hodin.

Tato struktura bude použita při definici některých komponent.

2.3.1.1 Horní navigace

Horní navigační panel bude realizován pomocí komponenty AppBar. Tento panel bude vždy obsahovat název aktuální stránky. Dále bude vpravo volitelně obsahovat tlačítka s možnostmi, které souvisí s aktuální stránkou.

Všechny hlavní stránky budou obsahovat tlačítko pro přechod do nastavení a podnabídku s dalšími možnostmi, které přímo souvisí s aktuální stránkou (viz obrázek 2.2a). Podnabídka bude vždy obsahovat možnost odpojení od hodin.

Podstránky budou v navigačním panelu vlevo od názvu stránky vždy obsahovat tlačítko pro přechod na předchozí stránku (viz obrázek 2.2b).

(a) Hlavní stránka **(b)** Podstránka

Obrázek 2.2 Návrh horní navigace aplikace NixieApp

2.3.1.2 Dolní navigace

Dolní navigace bude realizována pomocí komponenty BottomNavigation, která může obsahovat nejvýše 5 položek. Do této navigace budou umístěny odkazy na hlavní stránky, proto bude obsahovat 4 položky – datum a čas, budíky, stopky a časovač (viz obrázek 2.3). Oproti původní aplikaci bude položka nastavení přesunuta do horní navigace. Dolní navigace tedy bude vypadat stejně jako nativní aplikace pro iOS a Android (Samsung) (viz příloha D). Uživatelům obou operačních systémů tedy budoucí aplikace nabídne takové UI dolní navigace, na které jsou zvyklí. Navigace bude zobrazena pouze v případě, že jsou hodiny připojeny.

Obrázek 2.3 Návrh dolní navigace aplikace NixieApp

2.3.1.3 Načítání

Při komunikaci s hodinami přes BLE se data mohou přenášet s menším zpožděním. Tuto skutečnost je třeba indikovat pro aktuální obrazovku. To znamená, že když dochází k přenosu dat, která jsou obsažena na aktuální stránce, bude uživateli zobrazen stav načítání. Uživatel díky tomu pochopí, že nedošlo k chybě a že by měl počkat. Pro tento účel bude použita komponenta LinearProgress (viz obrázek 2.4a), která bude umístěna v horním okraji aplikace. Pro indikaci načítání dat v rámci určité komponenty bude použita komponenta Skeleton (viz obrázek 2.4b). Jde o šedý čtyřúhelník, který se přizpůsobí velikosti načítaných dat a bude dočasně vykreslen na jejich místě.

(a) LinearProgress **(b)** Skeleton

Obrázek 2.4 Návrh komponent pro indikaci načítání

2.3.1.4 Oznámení

Aplikace bude u každé akce, vyvolané uživatelem, určovat její výsledek. Neúspěch provedené akce bude uživateli vždy oznámen prostřednictvím notifikace. Úspěch provedené akce bude oznámen pouze v případě, není-li to patrné z příslušné UI komponenty, pomocí které uživatel danou akci vyvolal. Toto oznámení se bude zobrazovat v podobě zprávy v dolní části obrazovky. Pro tento účel bude použita komponenta SnackBar (viz obrázek 2.5). Oznámení zmizí po uplynutí nastavené doby zobrazení, nebo kliknutím na tlačítko.

(a) Úspěch **(b)** Neúspěch

Obrázek 2.5 Návrh komponenty SnackBar

2.3.1.5 Tlačítko LoadingButton

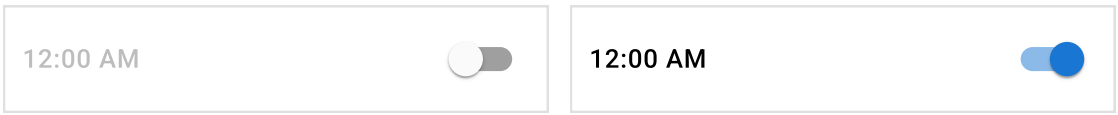
Tlačítko LoadingButton (viz obrázek 2.6) bude mít 2 stavy – zobrazení dat a zpracování požadavku. Ve stavu zobrazení bude tlačítko klikatelné. Ve stavu zpracování tlačítko klikatelné nebude a nad tlačítkem se zobrazí vrstva obsahující animovanou komponentu CircularProgress. Toto tlačítko bude použito v případě, že vyvolává akci, jejíž zpracování může trvat delší dobu. Například se může jednat o komunikaci s hodinami, kdy firmware nestihne požadavek zpracovat za krátkou dobu či došlo k přerušení spojení a probíhá pokus o opětovné připojení.

(a) Zobrazení **(b)** Zpracování

Obrázek 2.6 Návrh komponenty pro tlačítko LoadingButton

2.3.1.6 Budík

Budík (viz obrázek 2.7) bude tvořen komponentou Paper, což je kontejner ohraničený rámečkem. Uvnitř budou 2 části. Levou část bude tvořit tlačítko, zobrazující aktuálně nastavený čas budíku. Po kliknutí na toto tlačítko se zobrazí okno obsahující komponentu pro výběr času. Při uložení nové hodnoty času se okno zavře a tlačítko bude ve stavu načítání, dokud hodiny nevrátí odpověď s potvrzením o úspěchu. Pravou část bude tvořit komponenta Switch, která bude zodpovědná za změnu stavu budíku. Když je budík zapnutý, bude čas zvonění zvýrazněn tmavší barvou.



(a) Vypnutý **(b)** Zapnutý

Obrázek 2.7 Návrh komponenty pro budík

2.3.1.7 Výběr data a času

Pro výběr času bude použita Material Design komponenta TimePicker. Tato komponenta podporuje 24h formát i AM/PM formát času. Pro výběr času nabízí 2 režimy. První režim umožňuje zadat čas posunutím ručiček na ciferníku, což je vhodné pro ovládání přes dotykovou obrazovku i pomocí kurzoru myši. Druhý režim umožňuje zadat čas do textového pole pomocí klávesnice.

Pro výběr data bude použita Material Design komponenta DatePicker. Výběr roku se realizuje výběrem z rozevíracího seznamu. Pro volbu měsíce a dne slouží kalendář, zobrazující všechny dny ve vybraném měsíci. Přechod o jeden měsíc dopředu i dozadu se realizuje pomocí 2 tlačítek ve tvaru šipky.

Knihovna MUI nabízí navíc komponentu DateTimePicker, která slouží pro výběr data i času zároveň. Tato komponenta bude použita pro manuální nastavení data a času na hodinách. Všechny 3 komponenty zvládnou zajistit dostatečnou podporu pro mobilní i desktopová zařízení.

2.3.2 Stránky

Tato sekce obsahuje popis návrhu uživatelského rozhraní pro jednotlivé stránky. Obrázky 2.9 a 2.10 obsahují návrhy jednotlivých stránek.

2.3.2.1 Připojení

Z bezpečnostních důvodů je mobilními operačními systémy a webovými prohlížeči vyžadováno, aby skenování BLE zařízení bylo vyvoláno uživatelským gestem. Z tohoto důvodu bude obrazovka připojení obsahovat tlačítko, které bude spouštět proces skenování. Po kliknutí se v seznamu budou postupně zobrazovat nalezená zařízení. Proces skenování se ukončí buď automaticky po uplynutí časového intervalu, nebo manuálně kliknutím na tlačítko. V případě, že hodiny nebudou nalezeny, bude uživateli umožněno opakovat skenování stisknutím tlačítka. Pro připojení uživatel klikne na nalezené zařízení. Pokud se připojení nepovede, zobrazí se oznámení o neúspěchu. V opačném případě se zobrazí stránka s datem a časem.

2.3.2.2 Datum a čas

Úvodní stránka, která se objeví po připojení, bude zobrazovat následující údaje:

- **Čas**

  Stránka bude zobrazovat aktuálně nastavený čas na hodinách. Aplikace bude podporovat 2 formáty zobrazení času – AM/PM a 24h formát. Hodiny NixieClock podporují pouze 24h formát. Z tohoto důvodu bude možné nastavit formát zobrazení času pouze v rámci aplikace NixieApp. Formát zobrazení času na této stránce tedy bude záviset pouze na nastavení aplikace.

- **Datum**

  Stránka bude zobrazovat aktuálně nastavené datum na hodinách. Aplikace bude podporovat 2 formáty zobrazení data – americký a evropský. Hodiny též podporují oba formáty.

- **Den v týdnu**

  Stránka bude navíc zobrazovat den v týdnu, což hodiny NixieClock neukazují. Tento údaj bude určen slovně v jazyce nastaveném v aplikaci.

2.3.2.3 Budíky

Hodiny ukládají v paměti přesně 10 budíků. Neposkytují však žádnou informaci o tom, zda byl budík vytvořen a nastaven uživatelem aplikace, nebo jde o výchozí či náhodnou hodnotu v paměti. Zároveň tuto skutečnost není vhodné ukládat lokálně do paměti aplikace, protože uživatel může danou aplikaci používat na více různých zařízeních. Nelze tedy dynamicky měnit počet budíků. To znamená, že není potřeba vytvářet funkcionalitu pro přidání či smazání budíku. Na obrazovce se vždy zobrazí 10 budíků, u kterých bude možné upravit čas zvonění a stav.

2.3.2.4 Stopky

Stopky počítají čas od 0 sekund, který uběhne po startu. Zobrazení aktuálního počtu sekund bude ve formátu 'HH:MM:SS'. Pod tímto časovým údajem bude umístěn ovládací panel, který bude obsahovat tlačítka pro ovládání stopek. Stopky mají následující 3 stavy:

- **Vynulované (Reset)**

  Když jsou stopky vynulované, odpočet neprobíhá a na stránce bude zobrazeno '00:00:00', což představuje 0 sekund. Ovládací panel bude obsahovat tlačítko pro odstartování stopek.

- **Běžící (Running)**

  Když stopky běží, dochází každou sekundu k inkrementaci celkového počtu sekund od startu. Doba běhu stopek je omezena pouze velikostí datového typu uint32, a tedy může dosáhnout až 136 let (viz sekce 1.3). Hodiny ovšem nedokážou zobrazit hodnotu rovnou 100 hodin či větší. Navíc se stopky běžně nepoužívají na tak velkém časovém úseku. Z těchto důvodů bude nejvyšší zobrazitelnou jednotkou hodina. To znamená, že se po překročení 100 hodin nebudou zobrazovat dny, měsíce či roky. Ovládací panel bude obsahovat tlačítko pro pozastavení stopek a tlačítko pro vynulování stopek.

- **Pozastavené (Paused)**

  Když jsou stopky pozastavené, k inkrementaci počtu sekund nedochází. Na stránce tedy bude staticky zobrazena poslední hodnota, která byla před pozastavením. Ovládací panel bude obsahovat tlačítko pro pokračování inkrementace a tlačítko pro vynulování stopek.

Přepínání mezi jednotlivými stavy vyžaduje komunikaci s hodinami, což může způsobit zpoždění. V tomto případě je tedy vhodné použít tlačítko LoadingButton, které bude indikovat načítání, dokud aplikace nepřijme od hodin zprávu o úspěšné změně stavu stopek. Po obdržení této zprávy dojde k přechodu do nového stavu i v aplikaci. V případě neúspěchu se uživateli zobrazí oznámení s chybou a stopky v aplikaci zůstanou ve stejném stavu.

2.3.2.5 Časovač

Časovač provádí odpočet od zadaného počtu sekund až do 0. Zobrazení aktuálního počtu sekund bude též ve formátu 'HH:MM:SS'. Časovač má následující 3 stavy:

- **Vynulovaný (Reset)**

  Když je časovač vynulovaný, odpočet neprobíhá a na stránce bude zobrazena komponenta pro výběr startovacího času. Tato komponenta bude ve výchozím stavu zobrazovat 0 sekund. Pro zadání startovacího času nebyla v knihovně MUI nalezena žádná vhodná komponenta. Jedinou možností by bylo použití formulářového vstupu tak, jako to má stávající aplikace. Pouze by byl přidán formát vstupu a automatické doplnění znaku ':', což by zajistilo podporu zařízení bez tohoto znaku na numerické klávesnici. Mnohem lepší variantou je vytvoření vlastního řešení, které by bylo podobné komponentě použité v aplikaci Clock od společnosti Samsung (viz obrázek D.1b).

Nová komponenta pro výběr startovacího času bude označována jako NumberPicker. Tato komponenta bude nabízet možnost zadání startovacího času pomocí klávesnice. Ve výchozím stavu bude komponenta zobrazovat '00:00:00'. Po kliknutí na číslici, reprezentující určitou časovou jednotku (hodiny, minuty, sekundy), dojde k jejímu zvýraznění pomocí barevného pozadí a aplikace bude očekávat uživatelský vstup. V případě mobilního zařízení se pro editaci startovacího času otevře virtuální klávesnice. Po zadání určité hodnoty komponenta zajistí automatické přepnutí na hodnotu bezprostředně navazující. To znamená, že po zadání hodnoty s větší jednotkou, komponenta tento stav detekuje a přepne na hodnotu s menší jednotkou.

Pro zjednodušení výběru startovacího času bude navíc vytvořena vlastní komponenta, která se bude označovat jako PresetPicker. Komponenta nabídne zadání startovacího času časovače výběrem z přednastavených hodnot. Každá přednastavená hodnota bude reprezentována tlačítkem. Tlačítka budou umístěna v seznamu horizontálně. Pokud se všechny hodnoty nevejdou na stránku, bude možné posouvat jednotlivé položky seznamu.

- **Běžící (Running)**

  Když časovač běží, dochází každou sekundu k dekrementaci celkového počtu sekund od zadaného startovacího času. Podobně jako v nativních aplikacích Clock pro Android (viz obrázek D.1c) i iOS (viz obrázek D.3c) bude proces dekrementace doprovázen animací, kterou zajistí komponenta CircularProgress. Dekrementace se zastaví, když hodnota dosáhne 0 sekund. Poté automaticky dojde k vynulování časovače. Ovládací panel bude obsahovat tlačítko pro pozastavení časovače a tlačítko pro vynulování časovače.

- **Pozastavený (Paused)**

  Když je časovač pozastavený, k dekrementaci počtu sekund nedochází. Na stránce tedy bude staticky zobrazena poslední hodnota, která byla před pozastavením. Taktéž dojde k pozastavení animace komponenty CircularProgress. Ovládací panel bude obsahovat tlačítko pro pokračování odpočtu a tlačítko pro vynulování časovače.

2.3.2.6 Přednastavené časovače

Tato stránka umožní správu přednastavených časovačů. Na stránce bude zobrazen seznam přednastavených hodnot, které bude možné přidat, odstranit či editovat. Přednastavené hodnoty mají usnadnit zadání startovacího času. Budou se hodit zejména pro uživatele, kteří budou často používat funkci časovače pro stejné hodnoty. Přednastavené hodnoty by měly usnadnit zadávání startovacího času a nemělo by jich být tedy hodně. V opačném případě by uživatel musel dlouho hledat hodnotu v seznamu. Z tohoto důvodu umožní aplikace uložení nejvýše 8 přednastavených hodnot.

2.3.2.7 Nastavení

Nastavení bude obsahovat seznam položek, rozdělený dle kategorií. Jednotlivé kategorie budou reprezentovat vzájemně související funkce hodin či aplikace samotné. Každá kategorie bude mít zobrazený název a seznam tlačítek, které budou určovat aktuálně nastavené hodnoty či umožní modifikaci těchto hodnot. Konkrétně budou na stránce nastavení přístupny následující kategorie funkcí:

- **Hodiny**

  Nastavení data a času bude probíhat na podstránce (viz sekce 2.3.2.8), na kterou se uživatel dostane po kliknutí na tlačítko umístěné v této kategorii.

  Dále zde bude umístěno tlačítko pro výběr formátu zobrazení data na hodinách. Po kliknutí na toto tlačítko se zobrazí dialogové okno, obsahující formulář pro výběr formátu. Tento formulář

bude reprezentován pomocí komponenty Toggle Button, která bude obsahovat 2 položky pro výběr – americký a evropský formát data.

Délka zobrazení data a času je v rámci komunikačního protokolu odesílána v milisekundách. Pro dané hodnoty je mnohem rozumnější, aby je uživatel mohl zadávat v sekundách. Důvodem je zcela nepatrný rozdíl v zobrazení mezi dvěma hodnotami s rozdílem menším než 1 sekunda. Uživatel by tedy musel zbytečně doplňovat nuly při zadávání hodnot v milisekundách. Případně bude možné sekundy doplnit o desetinná místa pro specifické účely, mezi které může patřit například testování hardwaru. Pro vypnutí či zapnutí zobrazení bude použita komponenta Switch, která bude plnit funkci přepínače. Při vypnutí zobrazení si aplikace uloží do paměti poslední použitou hodnotu a aktuální hodnotu nastaví na 0 sekund. Při zapnutí zobrazení aplikace nastaví délku zobrazení na předchozí hodnotu z paměti, pokud je přítomna. V opačném případě bude použita výchozí hodnota. Toto přepínání zajistí snadný přechod mezi nočním a denním režimem bez nutnosti zadávání číselných hodnot. Noční režim znamená vypnutí zobrazení, a tedy zhasnutí digitronů.

Poslední položkou bude nastavení intervalu blikání LED diod, reprezentujících oddělovače při zobrazení data i času. Tato hodnota se též bude zadávat v jednotkách sekund s možností rozšíření o 3 desetinná místa. Vypnutí a zapnutí bude realizováno též pomocí komponenty Switch.

- **Časovač**

  Časovač má nastavitelnou délku zvonění. Tato hodnota se též bude zadávat v sekundách s možností přepínání. Dále zde bude tlačítko pro přechod na podstránku pro editaci přednastavených časovačů.

- **Budíky**

  Hodiny mají nastavitelnou délku zvonění. Tato hodnota se též bude zadávat v sekundách s možností přepínání.

- **Aplikace**

  V rámci aplikace bude nastavitelný jazyk a barevný režim. Obě hodnoty budou zadávány výběrem ze seznamu možností, který bude reprezentovat komponenta Radio Group. Pouze tyto možnosti v rámci nastavení budou přístupny i v případě, že jsou hodiny odpojeny. Dále bude tato kategorie obsahovat volbu formátu zobrazení času (AM/PM nebo 24 h). Také bude kategorie zahrnovat možnost pro zapnutí automatického připojení při spuštění aplikace.

2.3.2.8 Nastavení data a času

Nastavení data a času bude obsahovat následující kategorie:

- **Manuální nastavení**

  Pro manuální nastavení data a času bude k dispozici možnost volby hodnoty pomocí MUI komponenty DateTimePicker (viz sekce 2.3.1.7).

- **Automatické nastavení**

  V rámci automatického nastavení data a času bude na všech platformách k dispozici možnost synchronizace se zařízením, na němž aplikace běží. Tato operace se bude realizovat kliknutím na příslušné tlačítko. Případně zde bude i možnost synchronizace pomocí protokolu NTP.

- **Kalibrace**

  Uživateli bude k dispozici tlačítko pro manuální ověření přesnosti hodin. V opačném případě se mu zobrazí dialog s možností pro kalibraci. Dále zde bude umožněno zapnout automatickou kontrolu přesnosti hodin při spuštění aplikace.

2.3.3 Testování a výsledný návrh

Pro názornější ukázku uživatelského rozhraní aplikace byl návrh, vytvořený v nástroji Figma, rozšířen o interaktivní prototyp. Důvodem byla možnost otestovat UI ještě před implementací, což by potenciálně mohlo ušetřit čas při vývoji.

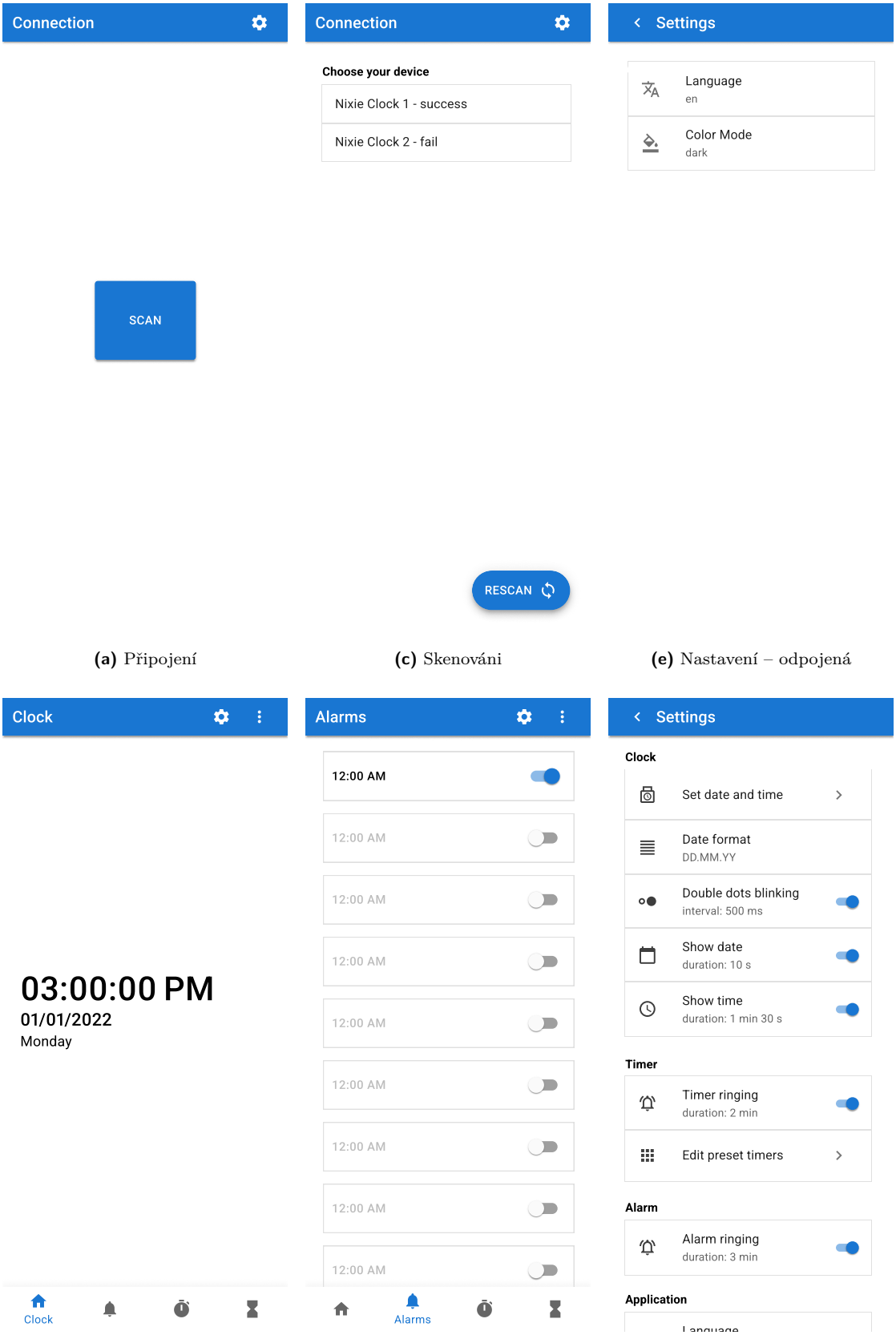
Testování se zúčastnily stejné osoby, které se podílely na testování stávající Android aplikace (viz sekce 1.7.1). Všechny změny, uvedené v prototypu nové aplikace, byly vnímány pozitivně. Na základě zpětné vazby byly do prototypu aplikovány následující úpravy:

- Na stránce časovače byla do podnabídky horní navigace přidána možnost přechodu na stránku pro editaci přednastavených časovačů. Dříve by bylo nutné přejít nejprve do nastavení a až poté na stránku editace. Tato možnost bude též zachována.
- Bylo zvětšeno písmo u komponent, zobrazujících čas na stopkách i časovači.
- Byl změněn formát zobrazení hodnot nastavení, zadávaných v sekundách. Takové hodnoty byly dříve zobrazovány v jednotkách sekund. V upraveném návrhu se hodnota transformuje na minuty, sekundy a milisekundy.

Výsledný návrh uživatelského rozhraní aplikace NixieApp, včetně interaktivního prototypu, je přístupný online na odkaze 2.8.

Obrázek 2.8 QR kód s odkazem na návrh UI aplikace [82]

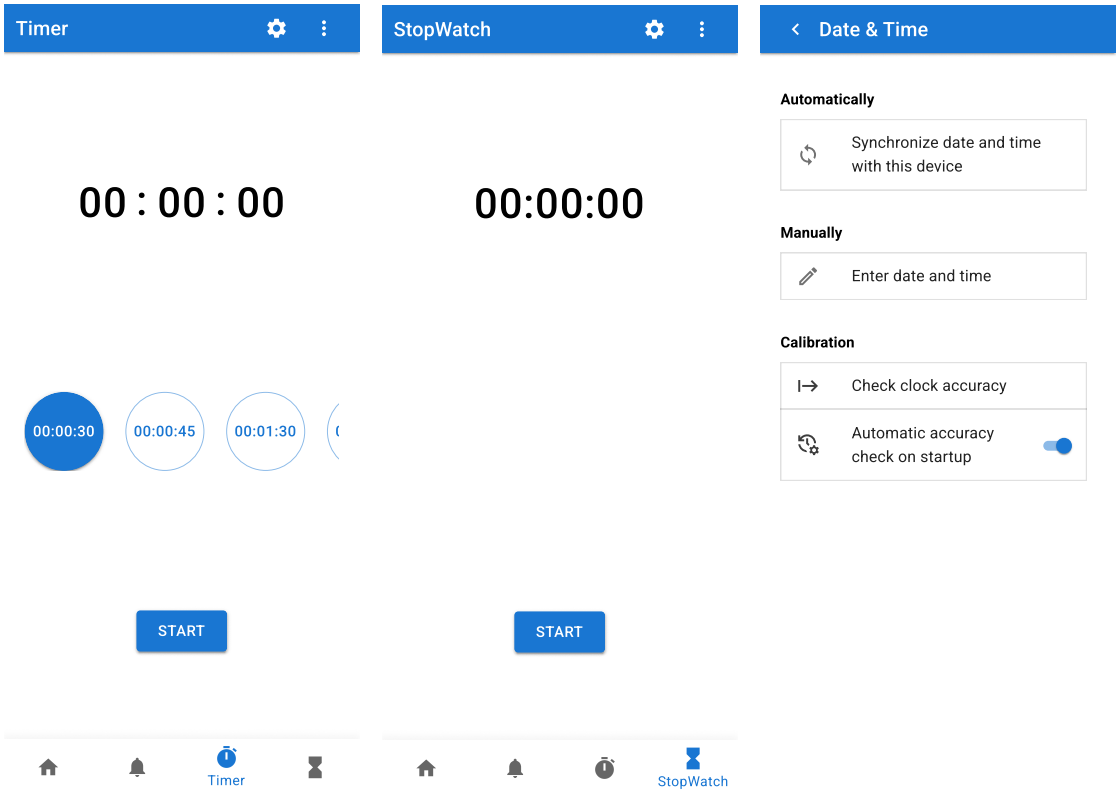
Exportovaná offline verze tohoto návrhu je k dispozici v příloze této práce ve formátu FIG. Tento soubor lze importovat do nástroje Figma, což se může hodit pro navazujícího vývojáře v případě návrhu nových funkcí či úpravy stávajících UI komponent.



(a) Připojení **(c)** Skenováni **(e)** Nastavení – odpojená

(b) Datum a čas **(d)** Budíky **(f)** Nastavení – připojená

Obrázek 2.9 Návrh uživatelského rozhraní aplikace NixieApp



(a) Časovač – zadání času **(c)** Stopky **(e)** Nastavení – datum a čas

(b) Časovač – běžící **(d)** Stopky – běžící **(f)** Přednastavené časovače

Obrázek 2.10 Návrh uživatelského rozhraní aplikace NixieApp

Kapitola 3
Implementace

Kapitola se zabývá implementací aplikace NixieApp. Nejprve pojednává o použitých nástrojích v průběhu implementace. Následně jde o popis implementace uživatelského rozhraní PWA, GATT klienta a proces tvorby nativních aplikací pro Android a iOS. Dále kapitola popisuje tvorbu emulátoru hodin pro Android.

3.1 Nástroje pro implementaci a příprava projektu

Tato sekce obsahuje popis použitých nástrojů v průběhu implementace. Dále se zabývá založením projektu a uvedením jeho struktury.

3.1.1 Vývojové prostředí

Pro implementaci aplikace bylo použito vývojové prostředí – Visual Studio Code. Pro zjednodušení práce s vybranými technologiemi byla použita následující rozšíření (anglicky extensions):

- **ESLint**

  ESLint staticky analyzuje TypeScript kód a upozorňuje na případné chyby.

- **Auto Import** – ES6, TS, JSX, TSX

  Toto rozšíření zajišťuje automatický import závislostí.

- **Prettier**

  Prettier je nástroj pro automatické formátování kódu. Díky tomu je kód zobrazen konzistentně a přehledně.

Pro úpravu kódu, sestavení a spuštění nativní iOS aplikace v emulátoru byl použit xCode. Pro Android aplikaci bylo použito Android Studio.

Na začátku vývoje bylo potřeba instalovat následující nástroje:

- **NodeJS**

  NodeJS je prostředí, které umožňuje spouštět kód, napsaný v jazyce JavaScript, mimo prohlížeč. Slouží pro sestavení aplikací vytvořených pomocí knihovny React a transpilaci jazyka TypeScript v jazyk JavaScript.

- **npm**

  npm je výchozí správce balíčků pro NodeJS. Provádí instalaci požadovaných verzí balíčků definovaných v rámci konfiguračního souboru 'package.json'.

3.1.2 Verzovací systém

Pro verzování zdrojových kódů aplikace NixieApp byl použit systém správy verzí Git. Jako vzdálený repozitář sloužil GitLab. Nástroj GitLab Pages umožňuje publikovat statické webové stránky přímo z úložiště v GitLabu. K zprovoznění této služby je třeba v kořenovém adresáři vytvořit konfigurační soubor '.gitlab-ci.yml'. Tento soubor obsahuje popis pro sestavení spustitelné aplikace a její nasazení do služby Gitlab Pages. Vývojáři tedy stačí nahrát aktuální verzi zdrojového kódu na Gitlab repozitář do specifikované větve (anglicky branch) a vše se provede automaticky. Pro použití technologie Web Bluetooth a pro možnost instalace PWA je prohlížečem Google Chrome vyžadováno, aby webová aplikace běžela přes protokol https. Výhodou nástroje Gitlab Pages je skutečnost, že nasazená webová aplikace může operovat přes protokol https s certifikátem podepsaným autoritou Let's Encrypt. Díky tomu bylo možné spouštět webovou aplikaci na různých zařízeních po celou dobu vývoje.

3.1.3 Založení projektu

Utilita Create React App poskytuje jednoduchý způsob vytvoření základní struktury React aplikace, včetně všech potřebných závislostí. Pro tvorbu aplikace byla zvolena šablona zahrnující podporu pro PWA a TypeScript. Základ aplikace byl vytvořen pomocí následujícího příkazu:

```
npx create-react-app nixie-app --template cra-template-pwa-typescript
```

3.1.3.1 Instalace balíčků

Základ aplikace byl doplněn o následující knihovny, které byly instalovány pomocí správce balíčků npm:

- **React MUI**

  Knihovna poskytuje hotové komponenty uživatelského rozhraní, splňující pravidla Material Design, pro knihovnu React.

- **Capacitor, bluetooth-le**

  Capacitor zabaluje webovou aplikaci do nativního kontejneru pro Android a iOS. Umožňuje tedy tvorbu nativních mobilních aplikací. Knihovna bluetooth-le, představující zásuvný modul (anglicky plugin) pro nástroj Capacitor, poskytuje multiplatformní rozhraní pro implementaci GATT klienta.

- **React Router DOM**

  Knihovna implementuje navigaci pro SPA aplikace. Taková navigace nevyžaduje přenačtení stránky a aktualizace obsahu se provádí prostřednictvím jazyka JavaScript.

- **Redux, Toolkit**

  Redux nabízí možnost pro tvorbu centralizované správy stavu aplikace. Redux Toolkit usnadňuje práci s knihovnou Redux tím, že umožňuje vytvoření stejné funkcionality s menším množstvím kódu.

- **Redux Thunk**

  Redux Thunk je knihovna, která poskytuje middleware pro zpracování asynchronních požadavků. V rámci aplikace byla použita při implementaci GATT komunikace.

- **RxJS**

  Knihovna bluetooth-le, stejně jako Web Bluetooth, poskytuje rozhraní, kde požadavky jsou zpracovány v rámci poskytované callback funkce. Knihovna RxJS nabízí implementaci pro reaktivní programování. V rámci aplikace byla například GATT operace Notification, oznamující notifikace prostřednictvím poskytované callback funkce, namapována na návrhový vzor Observable.

3.2 Implementace webové aplikace

Tato sekce popisuje implementaci webové aplikace. Nejprve se zabývá adresářovou strukturou a architekturou aplikace. Dále popisuje rozhraní datové vrstvy, implementaci uživatelského rozhraní a GATT klienta.

3.2.1 Struktura a architektura webové aplikace

Aplikace je rozdělena na prezentační, aplikační a datovou vrstvu. Prezentační vrstvu reprezentují UI komponenty knihovny React. Aplikační vrstvu představuje Redux. Datová vrstva zajišťuje práci s daty samotné aplikace i komunikaci s hodinami.

Vývojáři knihovny Redux doporučují organizovat logiku uživatelského rozhraní aplikace do souborů seskupených dle společné funkcionality v rámci jedné složky. Každá složka tedy reprezentuje jednu funkcionalitu (anglicky feature) aplikace. Funkcionalitu představují UI komponenty knihovny React a aplikační logika, vytvořená pomocí knihovny Redux. Na obrázku 3.1 je znázorněna adresářová struktura webové aplikace. Složka 'features' obsahuje funkce aplikace jako budíky, časovač, stopky a další.

Datová vrstva je přístupná přes rozhraní, což zjednoduší změnu implementace v případě budoucího přechodu na jinou knihovnu pro implementaci GATT klienta. Pro každou funkcionalitu bylo definováno vlastní rozhraní (viz 3.2.2).

```
src
├── index.tsx ..... vstupní bod aplikace
├── app ..... obsahuje globální nastavení celé aplikace
│   ├── store.ts ..... inicializace komponenty Store knihovny Redux
│   ├── rootReducer.ts ..... inicializace kořenové komponenty Reducer
│   └── App.tsx ..... kořenová React komponenta aplikace
├── common ..... generické UI komponenty, pomocné funkce
├── data ..... datová vrstva aplikace
│   ├── layer ..... definice datové vrstvy
│   │   ├── domain ..... doménové objekty datové vrstvy
│   │   ├── services ..... definice rozhraní datové vrstvy
│   │   └── exceptions ..... výjimky reprezentující chyby datové vrstvy
│   ├── impl ..... skutečná implementace datové vrstvy
│   └── stub ..... testovací implementace datové vrstvy
└── features ..... prezentační a aplikační vrstva aplikace
    ├── timer ..... funkcionalita časovače
    │   ├── timerSlice.ts ..... Aplikační logika časovače (Redux)
    │   └── Timer.tsx ..... UI komponenta reprezentující časovač (React)
    └── ...
```

Obrázek 3.1 Adresářová struktura webové aplikace

3.2.2 Definice datové vrstvy

Prvním krokem implementace byla definice rozhraní datové vrstvy. Tato vrstva zahrnuje metody pro GATT komunikaci, které vychází z komunikačního protokolu (viz příloha A). Operace pro GATT komunikaci jsou asynchronní. Z tohoto důvodu byl pro operace čtení a zápis zvolen generický datový typ Promise, který je součástí jazyka JavaScript. Pro operaci notifikace byla zvolena generická třída Observable, kterou zpřístupňuje knihovna RxJs. V případě nevalidní hodnoty či chyby v komunikaci jsou vyhozeny výjimky (anglicky exception). Výpis kódu 3.1 obsahuje ukázku rozhraní s definicemi metod pro čtení, zápis a notifikace hodnoty data a času.

Výpis kódu 3.1 Rozhraní DateTimeService

```
export interface DateTimeService {
    readDateTime(): Promise<ClockDateTime>
    streamDateTime(): Observable<ClockDateTime>
    stopStreamDateTime(): Promise<void>
    writeDateTime(dateTime: ClockDateTime): Promise<void>
}
```

Metody pro správu stavu připojení mezi aplikací a hodinami definuje rozhraní ConnectionService. Rozhraní nabízí následující 2 způsoby pro skenování:

- První způsob zajišťuje výpis nalezených zařízení v hotovém UI dialogu.
- Druhý způsob umožňuje streamování nalezených zařízení. To znamená, že při nalezení nového zařízení, je o tom informován každý zaregistrovaný posluchač (anglicky subscriber) této události. Díky tomu je možné vytvořit vlastní UI pro zobrazení seznamu nalezených zařízení. Pro streamování zařízení na webové platformě je však vyžadováno povolení experimentálních webových technologií v nastavení prohlížeče. Z tohoto důvodu byla tato možnost použita pouze v nativních aplikacích.

Základní metody GATT klienta jsou definovány v rozhraní GattClient. Implementace daného rozhraní, která byla uskutečněna pomocí knihovny 'bluetooth-le', je použita v rámci implementace jednotlivých servisních tříd. Rozhraní NtpClient zpřístupňuje metody pro získání aktuálního času přes protokol NTP. Metody, zajišťující lokální perzistenci dat, jsou definovány v rozhraní AppStorage.

3.2.3 Implementace uživatelského rozhraní

Díky existenci rozhraní datové vrstvy bylo možné uskutečnit vývoj uživatelského rozhraní nezávislé na datové vrstvě. Po definici požadovaných metod pro GATT komunikaci byly vytvořeny jejich testovací implementace, které simulují chování hodin NixieClock. Díky tomu byl vývoj uživatelského rozhraní proveden bez skutečného připojení ke GATT serveru. Po tvorbě testovací implementace datové vrstvy již bylo možné přejít k implementaci uživatelského rozhraní.

Tato sekce se zabývá popisem klíčových částí implementace uživatelského rozhraní aplikace NixieApp, včetně ukázek zdrojového kódu. Implementace uživatelského rozhraní byla provedena dle návrhu v sekci 2.3, kde jsou popsány jednotlivé UI komponenty a stránky. Z tohoto důvodu je zde popsána především tvorba vlastních UI komponent či úprava hotových komponent.

3.2.3.1 Injektáž závislostí

Injektáž závislostí (anglicky dependency injection) v rámci redux thunk middleware lze realizovat pomocí funkce withExtraArgument [75]. Daná funkce přijímá objekt s vybranými implementacemi rozhraní datové vrstvy. Middleware poté vloží všechny závislosti v podobě třetího argumentu jednotlivých thunk funkcí.

3.2.3.2 Navigace

Pro navigaci byla zvolena komponenta HashRouter, která ukládá aktuální umístění do hash části adresy URL. Každá stránka má vlastní identifikátor, který je definován v komponentě Route. V případě, že jsou hodiny odpojeny, je k dispozici stránka nastavení a jakákoliv jiná stránka v rámci aplikace je přesměrována na stránku připojení. Přechod na určitou stránku lze uskutečnit pomocí komponenty Link, nebo použitím funkce useNavigate.

3.2.3.3 NumberPicker a PresetPicker

Komponenta NumberPicker zajišťuje výběr startovacího času časovače. Byla vytvořena pro zajištění jednotného způsobu pro zadání startovacího času přes dotykovou obrazovku i fyzickou klávesnici (viz sekce 2.3.2.5).

Komponenta PresetPicker nabízí zadání startovacího času časovače výběrem z přednastavených hodnot. Tlačítka s přednastavenými hodnotami jsou umístěna v bloku, který lze horizontálně posouvat. Na desktopových zařízeních se toto posunutí realizovalo pomocí rolovací lišty, což nebylo uživatelsky přívětivé. Z tohoto důvodu byla přidána knihovna 'react-use-draggable-scroll', která umožňuje posunutí tažením bloku pomocí kurzoru v horizontálním směru.

3.2.3.4 Dialogové okno pro kalibraci hodin

Uživatel má možnost zkontrolovat přesnost hodin manuálně kliknutím na tlačítko nebo zapnout automatickou kontrolu přesnosti při spuštění aplikace. Při prvním spuštění si aplikace uloží aktuální čas nastavený na zařízení a na hodinách. Tyto hodnoty pak slouží jako referenční a aktualizují se při každé změně data a času. Kontrola přesnosti se realizuje porovnáním aktuálních hodnot s referenčními. V případě, že je odchylka větší než 5 sekund, zobrazí se dialogové okno obsahující následující možnosti:

- **Kalibrovat**

  Aplikace načte z hodin aktuální hodnoty registrů pro kalibraci. Dále provede výpočet nových hodnot na základě zjištěné odchylky dle vzorce odvozeného v sekci 1.4.1. Nové hodnoty pak nastaví do odpovídajících charakteristik v rámci GATT služby pro kalibraci.

- **Ignorovat**

  Uživatel může používat aplikaci z více různých zařízení. Může tedy změnit datum a čas z jednoho zařízení. Na jiném zařízení však aplikace není schopna zjistit, že k této změně došlo. Z tohoto důvodu byla přidána možnost ignorovat zjištěnou nepřesnost. Při této volbě dojde k aktualizaci referenčních hodnot.

- **Zavřít a rozhodnout se později**

  Při této volbě dojde k zavření dialogového okna a staré referenční hodnoty jsou zachovány.

3.2.3.5 Barevný režim

Aplikace podporuje 2 barevné režimy: tmavý a světlý. Navíc nabízí automatickou detekci barevného režimu na základě aktuálního nastavení systému. Tato možnost byla realizována pomocí JS rozhraní 'matchMedia', které umožňuje zjistit hodnotu vlastnosti 'prefers-color-scheme'. Tato vlastnost udává, zda aktuálně nastavené barevné schéma systému je světlé či tmavé. Pro případ změny barevného režimu systému za běhu aplikace byl zaregistrován posluchač této události (anglicky event listener).

3.2.3.6 Vícejazyčnost

Pro podporu vícejazyčnosti byla přidána knihovna 'react-i18next'. Překlady do jednotlivých jazyků jsou uloženy jako soubory ve formátu JSON. Do aplikace byly přidány 3 jazyky: angličtina, čeština a ruština. V nastavení aplikace lze zvolit buď konkrétní jazyk, nebo nastavit automatickou detekci jazyka. Automatická detekce byla vybrána jako výchozí možnost, která se nastaví při prvním spuštění aplikace. Knihovna totiž nabízí možnost automatické detekce jazyka na základě nastavení prohlížeče uživatele. Interně jde o použití vlastnosti 'language' zpřístupněné JS rozhraním 'Navigator'. Jako výchozí jazyk byla zvolena angličtina. Pokud je tedy detekován takový jazyk, který není aplikací podporován, použije se anglický jazyk. Použití této knihovny poskytne navazujícím vývojářům jednoduchý způsob pro doplnění nových jazyků.

3.2.4 PWA

Šablona, zvolená při založení projektu (viz sekce 3.1.3), zahrnuje podporu pro PWA. Service Worker, který tvoří základ PWA a je zodpovědný za offline funkčnost, je implementován v souboru 'service-worker.ts'. Logika pro jeho registraci je obsažena v souboru 'serviceWorkerRegistration.ts'. Samotná registrace se provádí v rámci kořenové React komponenty, umístěné v souboru 'index.ts'.

Nativní aplikace, vytvořené pomocí nástroje Capacitor, obsahují celou webovou aplikaci lokálně a nevyžadují tedy komunikaci s webovým serverem. Navíc nativní komponenty, zodpovědné za vykreslení obsahu webové stránky, nepodporují technologii Service Worker. Kvůli tomu by byla vyhozena výjimka a aplikace by mohla spadnout. Z tohoto důvodu se provádí kontrola platformy, na které aplikace běží, a k registraci dochází pouze na webové platformě.

Do souboru 'manifest.json' byly přidány všechny údaje o PWA. To zahrnuje například název aplikace, počáteční URL či barvu pozadí startovací obrazovky. Různé formáty ikonek pro PWA byly vygenerovány pomocí specializovaného nástroje (viz sekce 3.3.3).

3.2.5 Implementace GATT klienta

Tato sekce se zabývá implementací jednotlivých možností GATT klienta. Dále specifikuje rozdíl v možnostech webové a nativní aplikace. V průběhu implementace byly jednotlivé funkce testovány vůči emulátoru hodin NixieClock, což usnadnilo vývoj. Popis emulátoru obsahuje sekce 3.4.

3.2.5.1 Základní GATT operace

Operace pro čtení, zápis a notifikace byly namapovány na příslušné metody knihovny 'bluetooth-le'. Implementace těchto operací splňuje rozhraní datové vrstvy (viz sekce 3.2.2).

3.2.5.2 Skenování

Pro skenování byly v rámci datové vrstvy definovány 2 metody (viz sekce 3.2.2). V případě webové platformy bylo skenování realizováno pomocí dialogu implementovaného v rámci prohlížeče. V nativních aplikacích byla pro skenování vytvořena vlastní komponenta, zobrazující seznam nalezených zařízení.

3.2.5.3 Automatické připojení

Během analýzy technologie Web Bluetooth (viz sekce 1.2.4.4) bylo zjištěno, že daná technologie nabízí možnost pro implementaci automatického připojení bez nutnosti skenování. Pro použití této možnosti je však potřeba povolit experimentální webové technologie v nastavení prohlížeče.

Během implementace však bylo objeveno, že se po prvním připojení zařízení zařadí správně do seznamu připojitelných. Pokusy o automatické připojení však na jednotlivých platformách vždy skončily neúspěchem. Dle [83] je tato funkcionalita stále ve fázi vývoje a dle [84] byl tento problém již nahlášen vývojářům projektu Chromium. Z tohoto důvodu byla tato možnost z webové aplikace odstraněna a lze ji použít pouze na nativních platformách, kde funguje korektně.

V případě ztráty spojení se na webové platformě taktéž neprovádí pokusy o znovupřipojení. Uživatel je v takovém případě přesměrován na stránku skenování, kde se mu zobrazí zpráva o ztrátě spojení. Na nativní platformě se pokus o znovupřipojení provede 3krát. V případě úspěchu dojde k aktualizaci všech načtených dat z GATT serveru, jelikož v průběhu odpojení se mohly změnit. V případě neúspěchu též dojde k přesměrování na stránku skenování.

3.2.6 Nastavení data a času

V aplikaci byla implementována možnost pro manuální nastavení data a času. Dále byla přidána možnost synchronizace se zařízením.

Dle zadání platí, že by měly být zachovány všechny funkce staré aplikace. To zahrnuje i možnost synchronizace času pomocí NTP. Webová platforma ovšem nepodporuje protokol UDP, na kterém je postaven protokol NTP pro synchronizaci času. Z tohoto důvodu nebylo možné přidat tuto funkci do PWA. Moderní mobilní operační systémy pravidelně synchronizují čas na telefonu. Nepřesnost může vzniknout pouze na starších Android zařízeních, u nichž k synchronizaci dochází pouze při startu zařízení. Z tohoto důvodu byla synchronizace času pomocí NTP zachována pouze v nativní Android aplikaci. Pro tento účel byl implementován SNTP plugin pro Capacitor. Stejně jako v původní Android aplikaci byl jako SNTP server zvolen time.google.com. Plugin vznikl s použitím open source zdrojových kódů OS Android, které byly převzaty ze SNTP pluginu pro Apache Cordova, publikovaného na [85].

3.3 Tvorba nativních aplikací

Po implementaci webové aplikace byly vytvořeny nativní aplikace pro Android a iOS pomocí nástroje Capacitor. Tato sekce popisuje proces jejich tvorby a provedené úpravy.

3.3.1 Android aplikace

Nativní Android aplikace byla vygenerována pomocí následujícího příkazu:

```
npx cap add android
```

3.3.1.1 Oprávnění

Pro použití knihovny 'bluetooth-le' pro tvorbu GATT klienta bylo třeba přidat několik oprávnění do souboru 'AndroidManifest.xml'. Pro verzi Android 11 (API 30) a starší jsou pro skenování zařízení vyžadována následující oprávnění pro polohu:

```
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
```

Pro novější verze Android (API 31+) bylo přidáno oprávnění pro skenování, které navíc specifikuje, že aplikace nikdy neodvozuje fyzickou polohu z výsledků skenování.

```
<uses-permission
    android:name="android.permission.BLUETOOTH_SCAN"
    android:usesPermissionFlags="neverForLocation"
    tools:targetApi="s" />
```

3.3.2 iOS aplikace

Nativní iOS aplikace byla vygenerována pomocí následujícího příkazu:

```
npx cap add ios
```

Do souboru Plist.Info bylo třeba přidat klíč NSBluetoothAlwaysUsageDescription. Jeho hodnotou je zpráva, která uživateli sdělí, proč aplikace potřebuje přístup k Bluetooth. Tento klíč je vyžadován, pokud aplikace používá rozhraní Bluetooth. V opačném případě by aplikace spadla při spuštění.

Dále bylo zjištěno, že text stavové lišty (anglicky status bar) je ve výchozím stavu černé barvy. Při zapnutí tmavého režimu v aplikaci NixieApp nebyl tento text viditelný. Z tohoto důvodu byl do souboru Plist.Info přidán klíč UIViewControllerBasedStatusBarAppearance s hodnotou 'NO'. Tento klíč udává, zda se vzhled stavové lišty nastavuje programově či nikoliv. Dále byl přidán klíč UIStatusBarStyle s hodnotou Light Content. Tento klíč určuje styl stavové lišty pro celý program. Zvolená hodnota nastavuje text lišty na bílou barvu. Díky tomu je viditelný jak ve světlém, tak i ve tmavém režimu aplikace NixieApp.

3.3.3 Ikonky a úvodní obrazovka

Pro nástroj Capacitor existuje modul 'capacitor-assets'. Tento modul slouží pro generování ikonek a úvodních obrazovek (anglicky splash screen) všech požadovaných rozměrů a formátů jak pro PWA, tak i pro jednotlivé nativní platformy. Generování probíhá nejméně z jednoho souboru s logem. Volitelně lze přidat i tmavou variantu daného loga a barevné pozadí. V případě PWA jsou vygenerované ikonky automaticky přidány do souboru 'manifest.json'. Ikonka aplikace, stejně jako uživatelské rozhraní, byla vytvořena v nástroji Figma.

3.4 Emulátor hodin NixieClock

Autor stávající Android aplikace pro testovací účely z důvodu nedostupnosti hodin NixieClock implementoval emulátor hodin pro vývojovou sadu "STM32WB55 Nucleo". Během vývoje nové aplikace byly hodiny k dispozici pouze v hardwarové laboratoři. Bylo jasné, že možnost testovat aplikaci i mimo laboratoř by jistě usnadnila vývoj. Jelikož nebyl dostupný ani hardware pro použití existujícího emulátoru, bylo přijato rozhodnutí vytvořit vlastní emulátor hodin v podobě Android aplikace. To navíc umožní navazujícím vývojářům dostupnou možnost pro testování klientské aplikace, neboť je mnohem větší pravděpodobnost, že vývojář vlastní Android telefon či tablet. Další výhodou tohoto řešení je možnost sledovat aktuálně uložené hodnoty v reálném čase. Ve starém emulátoru totiž musel autor používat aplikaci třetí strany pro ověřování uložených hodnot.

3.4.1 Založení a konfigurace projektu

Jako vývojové prostředí bylo použito Android Studio. Nová aplikace, emulující hodiny NixieClock, byla pojmenována jako NixieEmulator. Při založení projektu byla zvolena šablona "No Activity", která neobsahuje žádné výchozí prvky uživatelského rozhraní. Cílovou verzí operačního systému byl vybrán Android 12 (API 32). Od verze Android 5.0 (API 21) může zařízení fungovat v roli Peripheral. To znamená, že od dané verze lze na Android zařízení implementovat GATT server. Minimální podporovanou verzí byl zvolen Android 8 (API 26). Tato volba zajistila podporu více než 90 % Android zařízení na trhu.

3.4.1.1 Oprávnění

Oprávnění aplikace slouží k ochraně soukromí uživatelů. Omezují přístup k datům uživatele či stavu systému. Dále omezují akce, jako je nahrávání zvuku, přístup ke kameře či právě použití Bluetooth. Oprávnění aplikace jsou obsažena v souboru 'AndroidManifest.xml'. Obecně v operačním systému Android existují 2 základní typy oprávnění:

- **Install-time oprávnění**

  Tento typ oprávnění umožňuje aplikaci přístup k takovým funkcím zařízení, které minimálně ovlivňují systém. Jde například o připojení k internetu. Když si uživatel aplikaci nainstaluje, systém automaticky udělí aplikaci oprávnění.

- **Nebezpečná (Runtime) oprávnění**

  Tento typ oprávnění poskytuje aplikaci přístup k takovým funkcím zařízení, které podstatně ovlivňují systém. Jde například o kontaktní údaje či informaci o poloze. Uživatele je třeba o tento typ oprávnění požádat za běhu aplikace. To znamená, že je třeba pokaždé programově kontrolovat, zda bylo oprávnění již uděleno a v opačném případě o něj požádat.

Pro tvorbu GATT serveru na operačním systému Android bylo třeba uvést několik oprávnění. Pro podporu starších zařízení s verzí Android 11 (API 30) a nižší bylo třeba přidat následující install-time oprávnění, která jsou nezbytná k provedení jakékoliv komunikace přes BLE:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Dále byly přidány 2 runtime oprávnění pro zařízení s verzí Android 12 (API 31) a vyšší. První oprávnění povoluje možnost připojení k okolním zařízením přes BLE. Druhé oprávnění povoluje vysílání Advertisement paketů, což je nezbytné k tomu, aby bylo dané zařízení vyhledatelné pro okolní zařízení. Tato oprávnění vypadají následovně:

```
android.permission.BLUETOOTH_CONNECT
android.permission.BLUETOOTH_ADVERTISE
```

3.4.1.2 Technologie pro tvorbu GATT serveru

V rámci Android Bluetooth API jsou odpovědi na všechny asynchronní požadavky vráceny do jednotlivých metod, obsažených v implementaci rozhraní BluetoothGattServerCallback. Například odpověď na požadavek pro čtení libovolné charakteristiky přichází do implementace metody onCharacteristicReadRequest. Poté je nutné dle UUID charakteristiky, která spolu s odpovědí přichází jako parametr metody, zjistit, o která data se jedná a na základě toho je zpracovat. Evidentně by bylo vhodné poskytnout vývojářům takové rozhraní, které by bylo pro implementaci jednodušší. V případě tvorby GATT klienta je dané API velice podobné. Z tohoto důvodu také vznikla řada knihoven třetích stran pro implementaci GATT klienta.

GATT server, běžící na operačním systému Android, se zřejmě používá pouze ve specifických scénářích. Existuje totiž pouze malé množství různých nástrojů třetích stran pro jeho implementaci. Nalezena byla pouze 1 knihovna – RxAndroidBleServer [86]. Tato Knihovna podporuje jazyk Java a staví na principech reaktivního programování. Knihovna byla naposledy aktualizována v listopadu roku 2020.

Emulátor byl vytvářen pouze pro Android. Použití knihovny třetí strany v tomto případě nenabízelo takové výhody, jako v případě implementace multiplatformního řešení. Nebylo rozhodně žádoucí objevit v průběhu vývoje problém související s takovou knihovnou. Z tohoto důvodu bylo přijato rozhodnutí použít přímo Android Bluetooth API. Rozhodnutí bylo též ovlivněno skutečností, že z dlouhodobého hlediska představuje daná volba stabilnější řešení.

3.4.1.3 Návrh uživatelského rozhraní emulátoru

V rámci operačního systému Android existují následující komponenty:

- **Activity**

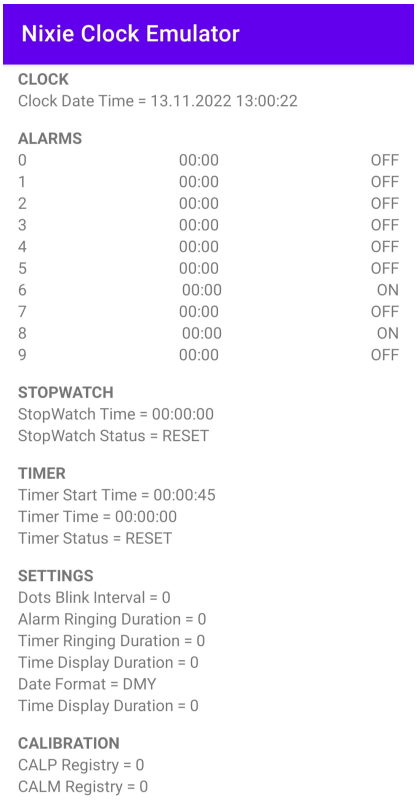
  Tato komponenta představuje vstupní bod aplikace. Každá aktivita musí být registrována v souboru 'AndroidManifest.xml'. Daná komponenta zajišťuje interakci s uživatelem a je reprezentována v podobě jednoho okna. Může obsahovat mnoho komponent typu Fragment.

- **Fragment**

  Tato komponenta představuje část uživatelského rozhraní aplikace. Fragment definuje a spravuje své vlastní rozvržení, má svůj vlastní životní cyklus a může zpracovávat své vlastní vstupní události. Daná komponenta může existovat pouze v rámci aktivity a je reprezentována v podobě jedné stránky (záložky).

Aplikace NixieEmulator je určena pouze pro testovací účely a nejedná se o produkt pro koncového uživatele. Z tohoto důvodu bylo rozhodnuto vytvořit jednoduché uživatelské rozhraní, skládající se pouze z 1 stránky, na které se zobrazují aktuální data (viz obrázek 3.2). To znamená, že aplikace obsahuje jednu aktivitu a jeden fragment. Pro tvorbu uživatelského rozhraní bylo použito XML.

Pro programový přístup k jednotlivým prvkům uživatelského rozhraní, definovaného v XML, bylo dříve nutné požadovaný prvek vyhledat na základě jeho identifikátoru v podobě řetězce. K tomu je určena funkce 'findViewById'. Nyní lze použít takzvaný view binding, který na základě XML generuje třídu obsahující reference na všechny prvky s identifikátorem. Výhodou je typová bezpečnost a null safety. View Binding byl aktivován v souboru 'build.gradle'.



Obrázek 3.2 Snímek obrazovky aplikace NixieClockEmulator

3.4.1.4 Návrh jazyka a architektury aplikace

Pro implementaci byl zvolen jazyk Kotlin a architektura MVVM. Tato volba byla uskutečněna na základě doporučení společnosti Google pro tvorbu Android aplikací.

Kromě základních knihoven, které byly přítomny v nově založeném projektu, byla navíc přidána závislost na knihovnu Koin. Tato knihovna slouží pro vkládání závislostí (anglicky Dependency Injection) v rámci aplikace.

3.4.2 Implementace emulátoru

Na obrázku 3.3 je znázorněna adresářová struktura Android aplikace.

```
src
├── App.kt ..... vstupní bod aplikace
├── MainActivity.kt ..... hlavní aktivita aplikace
├── data ..... datová vrstva aplikace
│   ├── features ..... jednotlivé funkcionality aplikace
│   │   ├── layer ..... definice rozhraní jednotlivých funkcionalit datové vrstvy
│   │   └── impl ..... lokální implementace jednotlivých funkcionalit datové vrstvy
├── domain ..... definice doménových objektů
└── presentation ..... prezentační vrstva aplikace
    ├── ble ..... GATT server
    └── ui ..... uživatelské rozhraní aplikace
        ├── AllDataFragment.kt ..... fragment obsahující veškerá data z features
        ├── BlePermissionChecker.kt ..... pomocná třída pro kontrolu BLE oprávnění
        └── features ..... UI komponenty a aplikační logika jednotlivých funkcionalit
            ├── alarms ..... UI komponenta a logika pro budíky
            └── ...
```

Obrázek 3.3 Adresářová struktura Android aplikace NixieClockEmulator

Aplikace se skládá z datové, aplikační a prezentační vrstvy. Datová vrstva emuluje jednotlivé funkce hodin jako časovač, stopky, budík a další. Spravuje tedy perzistenci a přístup k aktuálním datům daných funkcionalit. Datová vrstva je oddělena rozhraními. Asynchronní operace pro čtení a zápis jsou definovány jako suspend funkce. Pro pravidelné notifikace o změně hodnot byl použit generický datový typ Flow, který představuje obdobu návrhového vzoru observable. Jde o implementaci principů reaktivního programování přímo v jazyce Kotlin. Ukázka kódu 3.2 obsahuje definici rozhraní datové vrstvy pro časovač.

Výpis kódu 3.2 Rozhraní TimerDataSource

```
interface TimerDataSource {
    fun streamTimerTime(): Flow<UInt>
    fun streamTimerStatus(): Flow<TimerStatus>
    fun streamTimerStartTime(): Flow<UInt>

    suspend fun setTimerTime(time: UInt)
    suspend fun setTimerStatus(status: TimerStatus)
    suspend fun getTimerStatus(): TimerStatus

    fun streamTimerRingingDuration(): Flow<UInt>
    suspend fun getTimerRingingDuration(): UInt
    suspend fun setTimerRingingDuration(value: UInt)
}
```

Implementace rozhraní jednotlivých funkcionalit v rámci datové vrstvy jsou lokální a zajišťují perzistenci dat pouze po dobu běhu programu. Při vypnutí aplikace jsou tedy všechny nastavené hodnoty ztraceny.

Všechny závislosti aplikace jsou definovány v rámci modulu 'clockModule', kde jsou inicializovány jednotlivé implementace datové vrstvy, komponenty viewModel a GATT server. Tyto závislosti jsou zabaleny do DI komponent knihovny Koin. Pro implementace datové vrstvy a GATT serveru byla vybrána komponenta singleton. Deklarování komponenty singleton znamená, že kontejner Koin bude uchovávat jedinou instanci objektu, inicializovaného v rámci dané komponenty. Pro viewModel byla použita stejnojmenná komponenta viewModel, která zjednodušuje jeho inicializaci a dokáže ho svázat s životním cyklem Android komponenty (jako je například Aktivita či Fragment). Ukázka kódu 3.3 obsahuje příklad definice DI komponent pro funkcionalitu časovače.

Výpis kódu 3.3 Vkládání závislostí

```
single<TimerDataSource> {
    InMemoryTimerDataSource()
}

viewModel {
    TimerViewModel(dataSource = get())
}
```

Uživatelské rozhraní bylo realizováno v podobě jediné Android komponenty Fragment, obsahující aktuální data všech funkcionalit. Dle SOLID principu jedné odpovědnosti musí každý objekt být zodpovědný za 1 konkrétní funkčnost. Z tohoto důvodu je každá funkcionalita reprezentována vlastní třídou, mapující data do XML prvků uživatelského rozhraní, a dědicem třídy viewModel, která představuje aplikační vrstvu a zajišťuje přenos dat mezi prezentační a datovou vrstvou.

Soubor 'NixieClockProfile.kt' obsahuje definici UUID všech servisů a charakteristik. Jednotlivé UUID jsou reprezentovány jako řetězce. Pro definici GATT struktury v rámci programu byly vytvořeny následující třídy:

- **BleCharacteristic**

  Obaluje třídu BluetoothGattCharacteristic z Android Bluetooth API. Zjednodušuje její inicializaci a rozšiřuje ji o funkce, které mají být volány při události pro čtení či zápis dané charakteristiky. Operace pro čtení vrací aktuální hodnotu z datové vrstvy. Operace pro zápis ukládá novou hodnotu do datové vrstvy.

- **BleService**

  Obaluje třídu BluetoothGattService z Android Bluetooth API. Zjednodušuje její inicializaci a přidává charakteristiky jako seznam objektů třídy BleCharacteristic.

- **NixieClockServer**

  Definuje Gatt strukturu hodin NixieClock pomocí třídy BleService a BleCharacteristic. Mapuje Android Bluetooth API na vlastní reprezentaci servisů a charakteristik a naopak.

Třída AndroidLeAdvertiser zajišťuje vysílání discoverable advertisement paketů. Toto vysílání slouží k tomu, aby byl GATT server nalezen klientem v průběhu skenování a bylo možné navázat připojení. Pro filtrování zařízení během skenování na straně klienta bylo do těchto vysílaných paketů přidáno UUID služby data a času. Na zařízeních s delším názvem však vysílání selhávalo s kódem 1 (ADVERTISE_FAILED_DATA_TOO_LARGE). Vysílaná data totiž nemohou být větší než 31 bytů. Z tohoto důvodu byl název zařízení programově změněn na 'Nixie'. Poté však bylo zjištěno, že se název zařízení někdy neměnil a vysílání selhávalo. Proto byl název zařízení z vysílaných paketů zcela vynechán.

V implementaci abstraktní třídy BluetoothGattServerCallback byly přetíženy metody, do kterých přicházejí požadavky na čtení, zápis či notifikace hodnot jednotlivých charakteristik. Za přepínání notifikací je zodpovědný deskriptor Client Characteristic Configuration Descriptor v rámci požadované charakteristiky.

3.4.3 Testování emulátoru

V rámci testování emulátoru byly provedeny jednotkové a manuální testy. Jednotkové testy pokrývají metody zajišťující konverzi mezi doménovými objekty a binárními daty, používanými v rámci BLE komunikace s klientskou aplikací.

Manuální testy byly provedeny s použitím aplikace nRF Connect [87]. Jde o implementaci generického GATT klienta. Pomocí této aplikace byla ověřena funkčnost operací pro čtení, zápis a notifikace hodnot jednotlivých charakteristik.

V průběhu testování bylo zjištěno, že GATT Server dostává při každém zapnutí Bluetooth jiný identifikátor. Jako důsledek nebylo možné v klientské aplikaci, v případě restartování Bluetooth, provést automatické připojení bez skenování. Při testování automatického připojení klientské aplikace tedy nesmí být Bluetooth restartován.

Dále byl při manuálním testování aplikace na různých zařízeních zjištěn problém s připojením. Klientská aplikace se nedokázala připojit ke GATT serveru. V některých případech bylo možné realizovat připojení až po párování na úrovni operačního systému v nastavení Bluetooth. Tento problém se projevil na zařízení Galaxy S8 s verzí Android 9. Tabulka 3.1 znázorňuje jednotlivá zařízení a verze operačních systémů, na kterých byl spouštěn GATT klient při testování připojení k zařízení Galaxy S8. Sloupec 'Připojení' určuje, zda se připojení povedlo (+), bylo vyžadováno párování (P) či se připojení nepovedlo (-). Při testování emulátoru na zařízení Galaxy A40 (Android 11) či Galaxy tab S8 (Android 13) se připojení vždy povedlo. Pouze na operačním systému Windows bylo nutné provést párování. Problém s připojením byl zřejmě způsoben konkrétním zařízením či starou verzí Android. Jelikož se připojení na novějších verzích operačního systému Android vždy povedlo a emulátor měl sloužit pouze k testovacím účelům, nebyl tento problém dále zkoumán.

Zařízení (GATT klient)	Operační systém	Připojení
iPad Pro M1 (2021)	iPadOS 16.2	-
iPad 10.2 (2021)	iPadOS 16.1	+
Galaxy A40 (2019)	Android 11	+
Galaxy tab S8	Android 13	+
MacBook Pro (2021)	macOS 13.0.01	P
Lenovo IdeaPad 3	Ubuntu 22.04.5	+
Acer Aspire 5	Windows 11	+

Tabulka 3.1 Zařízení použitá k testování aplikace NixieAppEmulator, běžící na zařízení Galaxy S8

3.4.4 Dokumentace

Dokumentační komentáře byly vytvořeny pomocí nástroje KDoc. Pro generování dokumentace byl použit nástroj Dokka. Vygenerovaná dokumentace ve formátu HTML je k dispozici v příloze této práce.

Kapitola 4

Testování

Kapitola se zabývá testováním aplikace NixieApp. Nejprve pojednává o manuálním testování PWA a nativních aplikací vůči existujícímu hardwaru a firmwaru. Dále se zabývá testováním PWA a uživatelského rozhraní aplikace. Následně jde o popis implementace jednotkových (unit) testů.

4.1 Testování vůči hodinám NixieClock

Nativní aplikace a PWA byly otestovány manuálně vůči hodinám NixieClock na následujících zařízeních a operačních systémech:

Zařízení	Operační systém	Bluetooth	Aplikace
Huawei P10 lite	Android 8	4.1	PN
Galaxy S8	Android 9	5.0	PN
Galaxy A40 (2019)	Android 11	5.0	PN
Xiaomi Mi 9T Pro (2019)	Android 11	5.0	PN
Galaxy tab S8	Android 13	5.2	PN
iPhone X	iOS 16.2	5.0	N
Lenovo Legion 5	Windows 10	5.0	P
Acer Aspire 5	Windows 11	4.2	P
Lenovo IdeaPad 3	Ubuntu 22.04.5 (+ Bluez 5.53)	5.1	P
iPad 10.2 (2021)	iPadOS 16.1	4.2	PN
MacBook Pro (2021)	macOS Ventura 13.0.1	5.0	PN

Tabulka 4.1 Zařízení použitá k testování aplikace NixieApp (P = PWA, N = nativní aplikace)

Nativní aplikace byly testovány na mobilních operačních systémech: Android a iOS. Progresivní webová aplikace byla testována v prohlížeči Google Chrome na operačních systémech: Android, Windows, Ubuntu a macOS. Na operačním systému Windows bylo provedeno testování i v prohlížeči Microsoft Edge. Na Galaxy S8 proběhlo navíc testování i v prohlížeči Samsung Internet. Na tabletu iPad 10.2 byl k testování použit prohlížeč Bluefy, protože Google Chrome nepodporuje technologii Web Bluetooth pro tento operační systém (viz sekce 1.2.4.4).

4.1.1 Problém s připojením

PWA i nativní aplikace se dokázaly připojit k hodinám z mobilních operačních systémů: Android a iOS. Negativně však dopadly všechny pokusy o připojení PWA k hodinám z desktopových operačních systémů: Windows, Ubuntu a macOS. V konzoli prohlížeče se vypisovala zpráva vyhozené výjimky o neúspěšném připojení (viz 4.1).

Výpis kódu 4.1 Zpráva výjimky vyhozené při pokusech o připojení

```
DOMException: GATT Server is disconnected.
Cannot retrieve services. (Re)connect first with ‘device.gatt.connect‘.
```

4.1.1.1 Identifikace chyby

Nejprve bylo třeba chybu identifikovat. Tento proces probíhal následovně:

1. **Hledání možného důvodu na internetu**

   Po prohledání internetu nebyla nalezena jednoznačná příčina pro vyhození této výjimky. Například dle [88] se tento problém projevil pouze na konkrétním zařízení s operačním systémem Windows 10. Důvod odpojení však nebyl stanoven. Ostatní nalezené případy byly též bez jasné odpovědi.

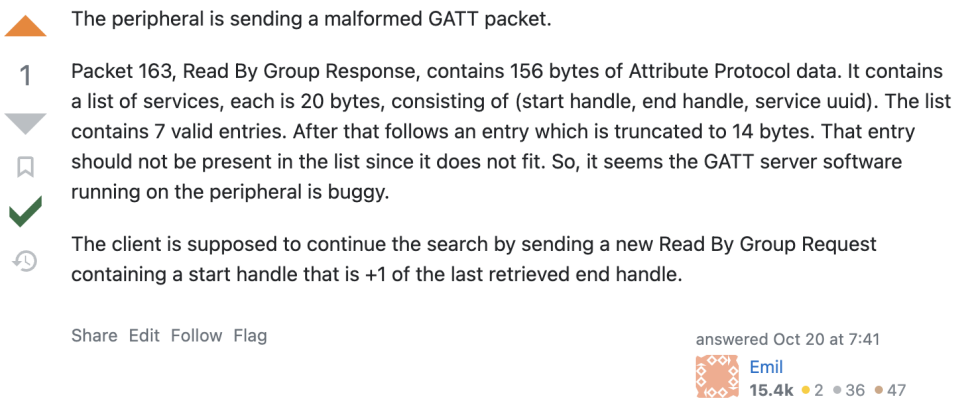
2. **Kontrola implementace knihovny 'bluetooth-le'**

   Nejprve bylo třeba zjistit, zda problém není způsoben případnou chybou v knihovně použité pro implementaci GATT klienta. Z tohoto důvodu bylo třeba provést pokus o připojení přímo s použitím Web Bluetooth API. K tomuto účelu byla použita oficiální ukázka implementace daného API, přístupná na odkaze [89]. Všechny pokusy o připojení dopadly negativně a byla vyhozena stejná výjimka.

3. **Analýza BLE komunikace**

   Následně bylo rozhodnuto analyzovat BLE komunikaci mezi desktopovými zařízeními a hodinami. Z tohoto důvodu byly odchyceny přijímané a odesílané pakety v průběhu BLE komunikace na operačních systémech Windows a Ubuntu. Pro odchycení paketů na operačním systému Windows byl použit nástroj USBPcap. Na operačním systému Ubuntu k tomuto účelu sloužila utilita tcpdump. Odchycené pakety byly uloženy v podobě souborů pcapng a jsou přístupny v příloze této práce. Pro vizualizaci těchto souborů sloužil program WireShark. V obou případech došlo k odpojení v průběhu objevování jednotlivých servisů a charakteristik. Na Ubuntu nebyl specifikován iniciátor odpojení ani jeho důvod. Odpojení na Windows bylo iniciováno ze strany klienta s důvodem: "Remote user terminated connection (0x13)". Dle [90] jde o generickou chybu a nenese žádnou užitečnou informaci k odhalení problému.

   Jelikož ani po delší analýze nebyl identifikován problém, bylo přijato rozhodnutí se obrátit za pomocí na portál StackOverflow [91]. Na základě odpovědi z daného portálu (viz obrázek 4.1) bylo zjištěno, že odpojení způsobovala chyba na straně GATT serveru. Nevalidní paket je znázorněn na obrázku 4.2. Modrou barvou je označen poslední validní servis, který obsahuje přesně 20 bytů. Červeně je označeno posledních 14 bytů, obsahujících pouze část informací o daném servisu. Validní servis by měl být popsán pomocí 20 bytů. Správně by tedy měl být obsažen až v rámci následujícího paketu. To znamená, že šlo o chybu v implementaci firmwaru.



The peripheral is sending a malformed GATT packet.

1

Packet 163, Read By Group Response, contains 156 bytes of Attribute Protocol data. It contains a list of services, each is 20 bytes, consisting of (start handle, end handle, service uuid). The list contains 7 valid entries. After that follows an entry which is truncated to 14 bytes. That entry should not be present in the list since it does not fit. So, it seems the GATT server software running on the peripheral is buggy.

The client is supposed to continue the search by sending a new Read By Group Request containing a start handle that is +1 of the last retrieved end handle.

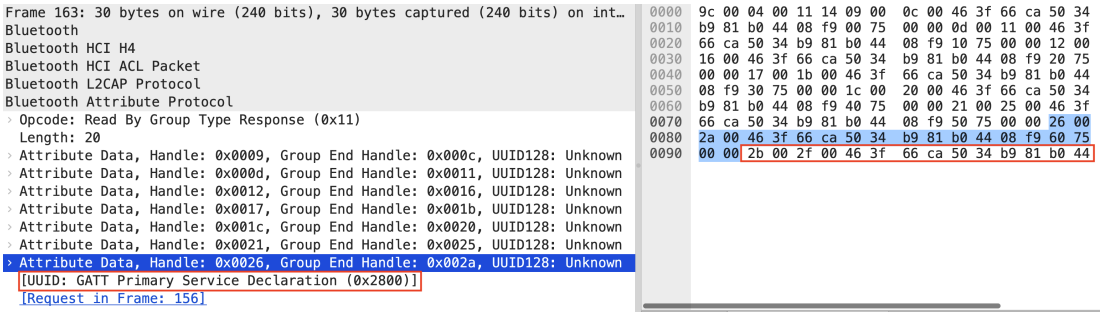
Share Edit Follow Flag

answered Oct 20 at 7:41

Emil

15.4k • 2 • 36 • 47

Obrázek 4.1 StackOverflow – určení chyby v BLE komunikaci [91]



```
Frame 163: 30 bytes on wire (240 bits), 30 bytes captured (240 bits) on int…
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
 › Opcode: Read By Group Type Response (0x11)
   Length: 20
 › Attribute Data, Handle: 0x0009, Group End Handle: 0x000c, UUID128: Unknown
 › Attribute Data, Handle: 0x000d, Group End Handle: 0x0011, UUID128: Unknown
 › Attribute Data, Handle: 0x0012, Group End Handle: 0x0016, UUID128: Unknown
 › Attribute Data, Handle: 0x0017, Group End Handle: 0x001b, UUID128: Unknown
 › Attribute Data, Handle: 0x001c, Group End Handle: 0x0020, UUID128: Unknown
 › Attribute Data, Handle: 0x0021, Group End Handle: 0x0025, UUID128: Unknown
 › Attribute Data, Handle: 0x0026, Group End Handle: 0x002a, UUID128: Unknown
    [UUID: GATT Primary Service Declaration (0x2800)]
    [Request in Frame: 156]
```

```
0000  9c 00 04 00 11 14 09 00  0c 00 46 3f 66 ca 50 34
0010  b9 81 b0 44 08 f9 00 75  00 00 0d 00 11 00 46 3f
0020  66 ca 50 34 b9 81 b0 44  08 f9 10 75 00 00 12 00
0030  16 00 46 3f 66 ca 50 34  b9 81 b0 44 08 f9 20 75
0040  00 00 17 00 1b 00 46 3f  66 ca 50 34 b9 81 b0 44
0050  08 f9 30 75 00 00 1c 00  20 00 46 3f 66 ca 50 34
0060  b9 81 b0 44 08 f9 40 75  00 00 21 00 25 00 46 3f
0070  66 ca 50 34 b9 81 b0 44  08 f9 50 75 00 00 26 00
0080  2a 00 46 3f 66 ca 50 34  b9 81 b0 44 08 f9 60 75
0090  00 00 2b 00 2f 00 46 3f  66 ca 50 34 b9 81 b0 44
```

Obrázek 4.2 WireShark – chyba v paketu způsobující odpojení klienta

4.1.1.2 Nalezení možného řešení

Desktopové operační systémy posílají na začátku spojení požadavek Exchange MTU REQUEST s hodnotou 527. Tento požadavek slouží ke změně velikosti posílaných dat v bytech v rámci protokolu ATT. Hodiny následně vracely odpověď Exchange MTU Response s hodnotou 156. V rámci Attribute Protocol bylo v nevalidním paketu posláno přesně 156 bytů (viz obrázek 4.2). Po analýze zdrojových kódů firmwaru bylo zjištěno, že část kódu odpovídající za sestavování paketů nebyla vytvořena autorem firmwaru, ale byla vygenerována nástrojem STM32CubeMX. Z uživatelského kódu však lze nastavit maximální hodnotu pro MTU, která se tehdy rovnala přesně 156 bytům.

Na základě výše uvedených poznatků bylo navrženo řešení v podobě nastavení maximální hodnoty MTU na hodnotu, kterou používaly mobilní telefony při komunikaci s hodinami NixieClock. Zbývalo tedy analyzovat BLE komunikaci mezi Android zařízením a hodinami. Na operačním systému Android je třeba pro zachycení BLE komunikace v možnostech pro vývojáře povolit protokol Bluetooth HCI a restartovat Bluetooth. Na Galaxy S8 je třeba navíc zařízení restartovat. Dále bylo provedeno připojení k hodinám pro zachycení komunikace a vypnut protokol Bluetooth HCI. Následně byl komunikační log exportován do počítače s operačním systémem Ubuntu pomocí nástroje androiddump. Tento soubor je obsažen v příloze této práce a lze jej též vizualizovat v programu WireShark.

Na základě analýzy komunikace bylo zjištěno, že se požadavek Exchange MTU REQUEST neposílal. V tomto případě se použila výchozí hodnota pro MTU 23 bytů. Pro vyřešení problému bylo tedy navrženo nastavit maximální hodnotu MTU na 23 byty.

4.1.1.3 Oprava a aktualizace firmwaru

Zdrojové kódy firmwaru byly převzaty z repozitáře publikovaného na [92]. Konkrétně byla použita verze z větve legacy. Po úpravě maximální hodnoty MTU na 23 byty bylo potřeba nahrát upravený firmware na hodiny NixieClock. Aktualizace firmwaru byla realizována pomocí programátoru STLINK-V3MINI a vývojového prostředí STM32CubeIDE. Připojení PWA k hodinám NixieClock pak dopadlo úspěšně i na desktopových operačních systémech. Upravené zdrojové kódy byly nahrány do forku repozitáře publikovaného na [93].

4.1.2 GATT notifikace v prohlížeči Bluefy

Při testování webové aplikace v prohlížeči Bluefy se objevil problém s notifikacemi charakteristik. Operace pro čtení a zápis byly funkční. Nejprve bylo třeba zjistit, zda problém nebyl způsoben případnou chybou v knihovně použité pro implementaci GATT klienta. Z tohoto důvodu bylo třeba otestovat notifikace přímo s použitím Web Bluetooth API. K tomuto účelu byla použita oficiální ukázka implementace notifikací pomocí daného API, přístupná na odkaze [89]. Bylo zjištěno, že notifikace byly funkční, což znamenalo, že chyba byla obsažena v implementaci knihovny 'bluetooth-le'. Po identifikaci problému bylo vytvořeno issue, popisující daný problém, v rámci github repozitáře této knihovny. Dané issue je publikováno na [94].

4.1.3 Zjištěné problémy hardwaru a firmwaru

Během testování byly objeveny následující problémy hardwaru a firmwaru:

- **Vypnutí zobrazení data a času**

  Při vypnutí zobrazení data a času zároveň by mělo dojít k zhasnutí všech digitronů i LED diod. Nedochází ovšem k zhasnutí všech digitronů. Na druhém digitronu zleva se stále zobrazuje číslo 5.

- **Zvonění časovače a budíku**

  Doba vyzvánění časovače někdy neodpovídá nastavené hodnotě. Zvonění v takovém případě trvá jen krátkou dobu. Jde o hodnotu menší než 1 sekunda. Stejný problém se projevuje i u zvonění budíku.

- **Persistence budíků**

  Po restartování hodin se mažou všechny uložené budíky. Každý budík je poté nastaven na čas 00:00 a je ve vypnutém stavu.

- **LED diody**

  Někdy se LED diody nepřepínají do režimu zobrazení data. To znamená, že při zobrazení data pokračují svítit všechny LED diody. Datum se tedy zobrazuje ve tvaru 'DD:MM:YY'. Místo toho by mělo být ve tvaru 'DD.MM.YY'.

- **Objevitelnost**

  Některá zařízení při skenování nebyla schopna objevit GATT server hodin NixieClock. Konkrétně se tento problém projevil na zařízení Huawei P10 lite (Android 8 – API 26). Ke skenování byly použity aplikace: NixieApp (PWA i nativní aplikace), nRF Connect [87] a BLE Scanner [95]. Při testování vůči emulátoru hodin k tomuto problému nedochází.

- **Přechod na letní/zimní čas**

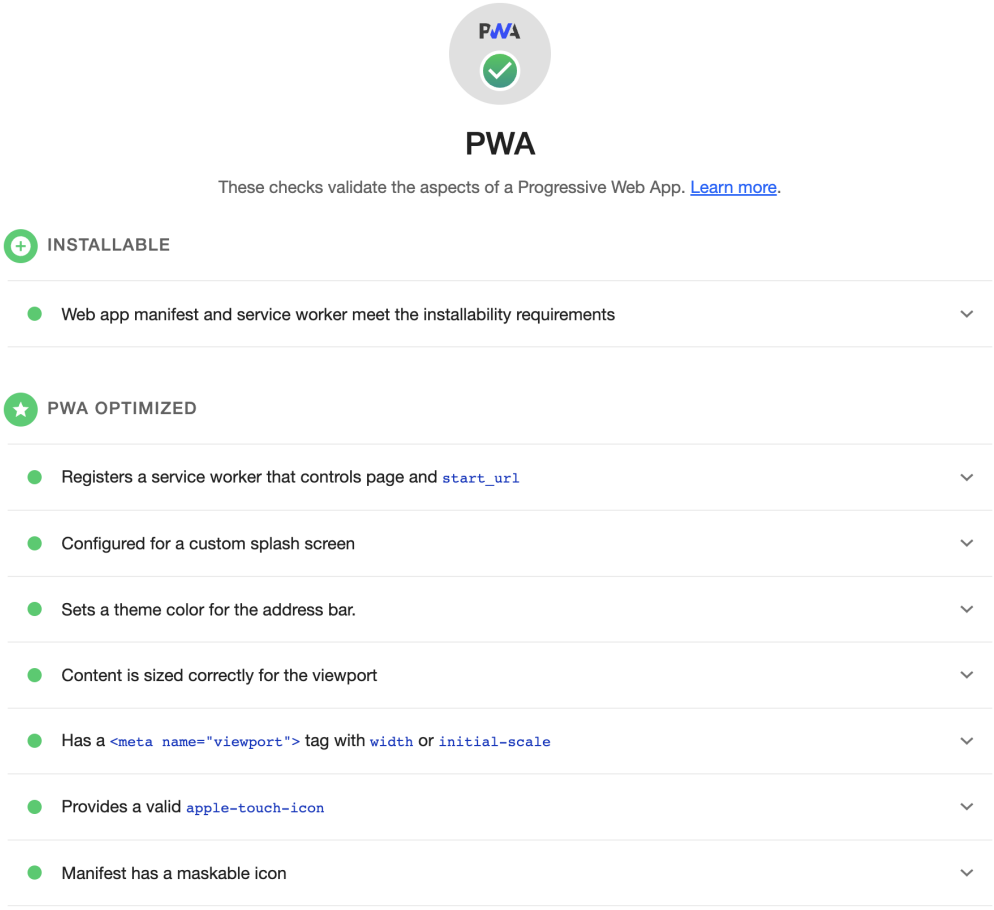
  Vedoucím práce bylo sděleno, že hodiny neuskutečnily přechod na zimní čas v roce 2022. Daný problém se projevil i při manuálním testování pro roky 2022 a 2023. Testován byl jak přechod na zimní, tak i letní čas.

4.2 Testování PWA

Pro testování PWA byly použity automatické i manuální testy.

4.2.1 Automatické testování

Automatické testování PWA proběhlo pomocí nástroje LightHouse, který slouží pro zlepšení kvality webových stránek. Provádí audity pro výkon, dostupnost, PWA, SEO a další. Neúspěšné body v auditu pak slouží jako indikátory, jak stránku vylepšit. Nástroj Lighthouse je přístupný v rámci Chrome DevTools. Exportovaný report auditu PWA ve formátu JSON je obsažen v příloze této práce a náhled je znázorněn na obrázku 4.3.



Obrázek 4.3 LightHouse – report z testování PWA

4.2.2 Manuální testování

Automatické testování prověřuje pouze hlavní koncepty PWA, potřebné především k tomu, aby prohlížeč umožnil instalaci aplikace. V rámci reportu nástroje LightHouse bylo doporučeno aplikaci otestovat manuálně v následujících bodech:

- **Každá stránka má vlastní URL**

  Identifikátor stránky v rámci aplikace je uložen do hash části adresy URL. Otevřením všech stránek aplikace bylo ověřeno, že každá stránka má v rámci aplikace jednoznačné URL.

- **Přechod mezi stránkami je plynulý**

  Každá stránka v aplikaci se zobrazuje okamžitě. To znamená, že se stránka vykreslí i bez dat z GATT serveru hodin NixieClock. Jednotlivé komponenty, zobrazující data, jsou v takovém případě ve stavu načítání.

- **Stránky fungují ve více různých prohlížečích a operačních systémech**

  Během testování byla ověřena funkčnost PWA v prohlížeči Google Chrome na operačních systémech Android, Windows, Ubuntu a macOS. Na operačním systému iOS (iPadOS) v prohlížeči Bluefy. Dále v prohlížeči Samsung Internet na Samsung zařízeních s operačním systémem Android. Na operačním systému Windows v prohlížeči Microsoft Edge.

4.2.2.1 Instalace

Instalace PWA, včetně ověření funkčnosti, byla provedena v následujících prohlížečích a operačních systémech:

- **Google Chrome**

  Instalace proběhla úspěšně na operačních systémech: Android, Windows, Ubuntu a macOS. Manuální testování jednotlivých funkcí aplikace na těchto operačních systémech dopadlo též úspěšně.

- **Samsung Internet**

  Při testování instalované aplikace na zařízení Galaxy S8 bylo zjištěno, že se někdy nezobrazuje dialog pro skenování BLE zařízení. Při spuštění webové aplikace přímo v prohlížeči tato funkce fungovala vždy. Dialog pro skenování se zobrazil v PWA vždy v případě, že bezprostředně před skenováním byla webová aplikace otevřena přímo v prohlížeči. Z tohoto důvodu je na operačním systému Android lepší volbou pro instalaci PWA prohlížeč Google Chrome.

- **Microsoft Edge**

  Instalace proběhla úspěšně a žádné specifické problémy zjištěny nebyly.

- **Bluefy**

  Dle [22] nabízí prohlížeč Bluefy podporu pro PWA na operačních systémech iOS a iPadOS. Při testování bylo zjištěno, že nenabízí možnost instalace aplikace.

4.3 Testování uživatelského rozhraní

Uživatelské rozhraní bylo testováno manuálně. Jednotlivé testovací případy jsou popsány v následujících podsekcích.

4.3.1 Uživatelské testování

Uživatelské testování bylo provedeno již ve fázi návrhu pomocí interaktivního prototypu, vytvořeného v nástroji figma (viz sekce 2.3.3). Testování se zúčastnily stejné osoby, které se podílely na testování stávající Android aplikace (viz sekce 1.7.1). Všechny změny, uvedené v prototypu nové aplikace, byly vnímány pozitivně. Na základě zpětné vazby byly do prototypu aplikovány následující úpravy:

- Na stránce časovače byla do podnabídky horní navigace přidána možnost přechodu na stránku pro editaci přednastavených časovačů. Dříve by bylo nutné přejít nejprve do nastavení a až poté na stránku editace. Tato možnost bude též zachována.
- Bylo zvětšeno písmo u komponent, zobrazujících čas na stopkách i časovači.
- Byl změněn formát zobrazení hodnot nastavení, zadávaných v sekundách. Takové hodnoty byly dříve zobrazovány v jednotkách sekund. V upraveném návrhu se hodnota transformuje na minuty, sekundy a milisekundy.

Při testování bylo zkontrolováno, že výsledná aplikace odpovídá prototypu a všechny výše uvedené body splňuje.

4.3.2 Obrazovka s výřezem

Mobilní zařízení mohou obsahovat na obrazovce výřez pro kameru či nějaké senzory. Uživatelské rozhraní nativní i webové aplikace by se mělo přizpůsobit takovým obrazovkám.

Na operačním systému Android byl výřez displeje simulován na zařízeních: Galaxy S8, Galaxy A40 a Galaxy tab S8. Simulaci výřezu lze nastavit v nastavení telefonu v možnostech vývojáře. Celkově byly testovány všechny 3 nabízené typy výřezů: rohový, dvojitý a vysoký. Ve všech případech se aplikace zobrazovala korektně.

Nativní aplikace pro operační systém iOS byla testována na telefonu iPhone X, který obsahuje na obrazovce výřez. Všechny stránky byly zobrazeny správně. Dále byla nativní iOS aplikace testována v xCode simulátoru zařízení iPhone 14 Pro, iPhone 13 a iPhone 12. I v tomto případě bylo uživatelské rozhraní zobrazeno korektně.

4.3.3 Vícejazyčnost

V rámci nastavení aplikace byla otestována automatická detekce jazyka. Testování proběhlo v prohlížeči Google Chrome na desktopových operačních systémech: Windows, macOS a Ubuntu. V nastavení prohlížeče byl postupně měněn seznam preferovaných jazyků. Změna jazyka se projevovala až po přenačtení stránky a pro všechny možnosti bylo chování aplikace korektní.

4.3.4 Barevné schéma

V rámci nastavení aplikace byla otestována automatická detekce barevného režimu dle aktuálního nastavení systému. Testování této možnosti v rámci PWA proběhlo v prohlížeči Google Chrome na operačních systémech: Android, iOS a macOS. Dále byla tato funkce ověřena i v rámci nativních aplikací. Ve všech případech se změna barevného režimu v systému korektně promítla do aplikace.

4.3.5 Výchozí hodnoty

Aplikace byla spuštěna v anonymním režimu, kde nebyly uloženy žádné hodnoty v lokální paměti prohlížeče (LocalStorage). Pomocí toho bylo ověřeno, že při prvním spuštění aplikace se nastavuje automatická detekce pro jazyk a barevné schéma. Dále bylo ověřeno, že se pro délku zvonění či zobrazení použijí výchozí hodnoty v případě, že daná nastavení byla na hodinách vypnuta.

4.3.6 Uživatelský vstup z klávesnice

Pro číselné vstupní hodnoty v rámci nastavení bylo použito formulářové pole. Na všech zařízeních, uvedených v tabulce 4.1, bylo možné zadat všechny povolené hodnoty. Na mobilních zařízeních bylo ověřeno, že se použije virtuální klávesnice určená pro číselné hodnoty.

4.4 Jednotkové testy

Jednotkové testy (anglicky Unit tests) individuálně a nezávisle na celém programu ověřují správnost menších částí kódu. Pro implementaci jednotkových testů byla použita knihovna jest. Testovány byly funkce pro konverzi doménových objektů a binárních dat posílaných přes protokol BLE. Příklad testu pro převod data a času z binární reprezentace na doménový objekt, který se používá v rámci aplikační vrstvy, obsahuje výpis kódu 4.2.

Výpis kódu 4.2 Test funkce toClockDateTime

```
test('toClockDateTime', () => {
    const input = new DataView(new ArrayBuffer(7))

    input.setUint16(0, 2000)
    input.setUint8(2, 7)
    input.setUint8(3, 22)
    input.setUint8(4, 3)
    input.setUint8(5, 4)
    input.setUint8(6, 5)

    const output = toClockDateTime(input)
    expect(output).toEqual({
        date: { year: 2000, month: 7, day: 22 },
        time: { hour: 3, minute: 4, second: 5 },
    })
})
```

Dále byly vytvořeny testy pro ověření správnosti funkcí pro výpočet hodnot pro kalibraci hodin. Ukázka 4.3 ověřuje výpočet v případě, že hodiny spěchají o 1 sekundu za den.

Výpis kódu 4.3 Test funkce calcSmoothCalibrationValues

```
test('Clock is 1 second fast per day', async () => {
    const { calp, calm } =
        calcSmoothCalibrationValues(dayInMs,
                                    dayInMs + secondInMs,
                                    { calp: 0, calm: 0 })
    expect(calp).toBe(0)
    expect(calm).toBe(12)
})
```

Soubor videí, pořízených během testování aplikace, je přístupný online na serveru Youtube na následujícím odkaze:

Obrázek 4.4 QR kód s odkazem na videoukázky aplikace [96]

Kapitola 5
Závěr

Cíl této práce – tvorba multiplatformní aplikace pro ovládání digitronových hodin – se podařilo úspěšně splnit. Řešením je aplikace, kterou lze spustit nativně na iOS a Android, ale také na webu jako progresivní webovou aplikaci. Přes webovou platformu je navíc zajištěna podpora desktopových operačních systémů: Windows, macOS a Linux.

Nejprve byly stanoveny požadavky na novou aplikaci na základě zadání, specifikace firmwaru a původní Android aplikace. Dále práce obsahuje analýzu protokolů BLE, NTP a kalibrace RTC. Následně byl rozebrán současný stav firmwaru, hardwaru a staré Android aplikace. Na základě testování byly odhaleny problémy v uživatelském rozhraní i některých funkcích staré Android aplikace.

Uživatelské rozhraní nové aplikace, včetně interaktivního prototypu, bylo navrženo pomocí nástroje Figma. Aplikace byla implementována jako progresivní webová aplikace, používající Web Bluetooth API. Tato experimentální webová technologie je stále ve vývoji a proto nemá plnou podporu na všech operačních systémech. Z tohoto důvodu byly vytvořeny nativní aplikace pro iOS a Android pomocí nástroje Capacitor, který nabízí jednotné rozhraní pro funkce jednotlivých platforem. Tyto nativní aplikace tedy používají Bluetooth API příslušných platforem a rozšiřují webovou aplikaci o další možnosti.

Nad rámec původního zadání byl implementován emulátor digitronových hodin pro operační systém Android v jazyce Kotlin, protože přípravek NixieClock byl dostupný pouze ve fakultní laboratoři. Předchozí emulátor GATT serveru nebylo možné použít z důvodu chybějícího hardwaru. Vytvořením nového emulátoru se usnadnil vývoj a bylo možné odhalit chyby v průběhu implementace GATT klienta.

Část aplikace byla otestována pomocí jednotkových testů. Testování progresivní webové aplikace bylo provedeno pomocí automatického nástroje LightHouse. Dále proběhlo manuální testování, které odhalilo problémy ve firmwaru a hardwaru. Byla identifikována a opravena chyba v implementaci GATT serveru, která znemožňovala připojení z desktopových zařízení.

Progresivní webová aplikace byla publikována pomocí nástroje gitlab pages. Odkaz vedoucí na spustitelnou aplikaci je zobrazen na obrázku 5.1.

Obrázek 5.1 QR kód s odkazem na webovou aplikaci – NixieApp [97]

5.1 Návrhy na rozšíření a vylepšení

Při implementaci aplikace byl kladen důraz na rozšiřitelnost a jednoduchou změnu implementace GATT klienta. Díky tomu může být v budoucnu zajištěn jednoduchý přechod přímo na Web Bluetooth API v případě, že daná technologie bude mít plnou podporu na všech požadovaných platformách. Firmware i aplikaci by bylo vhodné rozšířit o následující možnosti:

- **Budíky**

  Hodiny NixieClock zvládnou uložit přesně 10 budíků. Bylo by vhodné umožnit dynamický počet budíků. Budíky jsou nastavitelné pouze na konkrétní čas pro každý den. Bylo by dobré přidat možnost nastavení budíku na vybranou kombinaci dnů v týdnu či na konkrétní datum.

- **Zvonění budíku a časovače**

  Zvonění budíku je realizováno bzučákem na fixní frekvenci. Bylo by dobré vytvořit několik základních melodií vyzvánění a umožnit uživateli volbu melodie.

  Dále je vhodné přidat možnost pro deaktivaci zvonění budíku či časovače do komunikačního protokolu.

- **Letní čas (DST)**

  V rámci firmwaru je realizováno automatické přepínání mezi pásmovým a letním časem, které se řídí pravidly Evropské unie. Existují ovšem země, ve kterých se letní čas nepoužívá vůbec, nebo přechod nastává v jinou dobu než v zemích Evropské unie. Například pro USA či Kanadu platí, že letní čas začíná dříve a končí později než v zemích Evropské unie. Bylo by tedy vhodné umožnit uživateli tuto funkci vypnout, nebo nabídnout podporu automatického přepínání pro více různých regionů.

- **Aktualizace firmwaru**

  Bylo by vhodné přidat možnost aktualizace firmwaru přes mobilní aplikaci. Tato možnost by byla vhodná hlavně pro koncového uživatele. Aktualizaci by bylo možné realizovat též přes protokol BLE přímo z telefonu. Díky tomu by mohly být přidávány nové funkce či opravovány případné chyby v implementaci bez nutnosti použití specializovaného softwaru a hardwaru.

Příloha A

Komunikační protokol

Příloha obsahuje specifikaci komunikačního protokolu, převzatou z bakalářské práce pro tvorbu firmwaru [2].

Custom UUID template: 0000075[ID]-f908-44b0-81b9-3450ca663f46

- Clock Service (Hodiny)
  - ID: 00
  - Date Time Characteristic
    - SIG UUID: 0x2A08
    - Data type: Date Time [26]
    - Properties: read, write, notify
- Alarm Service (Budíky)
  - ID: X0, where X corresponds to the alarm number in hexadecimal notation. The alarms are numbered 1–10 (i.e., 0x1–0xA).
  - Alarm Time characteristic
    - SIG UUID: 0x2A08
    - Data type: Date Time [26]
    - Value: Time (Hour, Minute, Second) should be filled with zeros
    - Properties: read, write
  - Alarm status characteristic
    - ID: 11
    - Data type: uint8
    - Value: (0 = off, 1 = on)
    - Properties: read, write
- Stopwatch Service (Stopky)
  - ID: b0
  - Stopwatch current time characteristic
    - ID: b1
    - Data type: uint32
    - Value: seconds
    - Properties: read, notify

- Stopwatch status characteristic
  - ID: b2
  - Data type: uint8
  - Value: (0 = pause, 1 = running, 2 = reset)
  - Properties: read, write

- Timer service (Časovač)
  - ID: c0
  - Timer current time characteristic
    - ID: c1
    - Data type: uint32
    - Value: seconds
    - Properties: read, notify
  - Timer set time characteristic
    - UUID: c2
    - Data type: uint32
    - Value: seconds
    - Properties: write
  - Timer status characteristic
    - ID: c3
    - Data type: uint8
    - Value: (0 = pause, 1 = running, 2 = reset)
    - Properties: read, write, notify

- Settings service (Nastavení)
  - Date format characteristic
    - ID: d0
    - Data type: uint8
    - Value: (0 = DMY, 1 = MDY)
    - Properties: read, write
  - LED blink interval characteristic
    - ID: d1
    - Data type: uint32
    - Value: miliseconds (0 = LEDs off)
    - Properties: read, write
  - Time display duration
    - ID: d2
    - Data type: uint32
    - Value: miliseconds (0 = Time not showing)
    - Properties: read, write

- Date display duration
  - ID: d3
  - Data type: uint32
  - Value: miliseconds (0 = Date not showing)
  - Properties: read, write
- Alarm ringing duration
  - ID: d4
  - Data type: uint32
  - Value: miliseconds (0 = ringing off)
  - Properties: read, write
- Timer ringing duration
  - ID: d5
  - Data type: uint32
  - Value: miliseconds (0 = ringing off)
  - Properties: read, write

- Calibration service (Kalibrace)
  - ID: e0
  - CALP characteristic
    - ID: e1
    - Data type: uint8
    - Value: The CALP register is a single bit, so only the lowest bit of the characteristic is used. Any unused bits should be set to 0.
    - Properties: read, write
  - CALM characteristic
    - ID: e2
    - Data type: uint16
    - Value: The CALM register is 10 bits long, so only the lowest 10 bits of the characteristic are used. Any unused bits should be set to 0.
    - Properties: read, write

- GAP Service
  - Device name characteristic
    - Properties: read
  - Appearance characteristic
    - Properties: read
    - Value: 256 (Generic Clock category)

Příloha B

Web Bluetooth API

Web Bluetooth API je experimentální technologií. Je tedy stále ve vývoji a nemá plnou podporu všech prohlížečů. V tabulce B.1 jsou uvedeny základní GATT operace a jejích podpora v rámci jednotlivých prohlížečů. Specifikovány jsou minimální verze prohlížečů, podporujících uvedenou funkci. Znak 'X' udává, že prohlížeč danou funkcionalitu nepodporuje vůbec. Znak 'F' znamená, že daná funkcionalita vyžaduje povolení experimentálních webových technologií v nastavení prohlížeče.

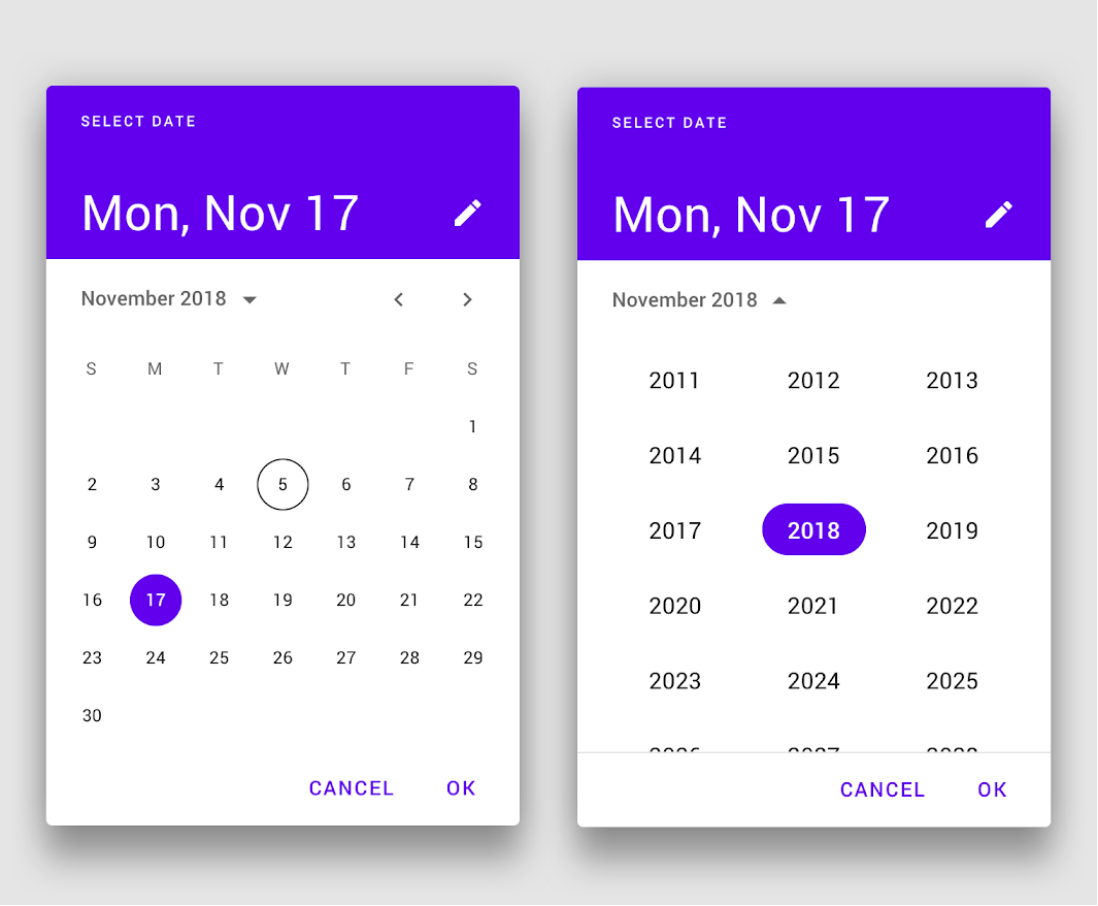
API Function	Google Chrome	Microsoft Edge	Samsung Internet
readValue	56	79	6.0
startNotifications	56	79	6.0
stopNotifications	56	79	6.0
getCharacteristic	56	79	6.0
connect	56	79	6.0
getCharacteristic	56	79	6.0
disconnect	56	79	6.0
getPrimaryService	56	79	6.0
requestDevice	56	79	6.0
writeValueWithoutResponse	85	85	14.0
writeValueWithResponse	85	85	14.0
getDevices	85 (F)	85 (F)	X
Scanning API	85 (F)	85 (F)	X

Tabulka B.1 Web Bluetooth API – podporované verze prohlížečů [18]

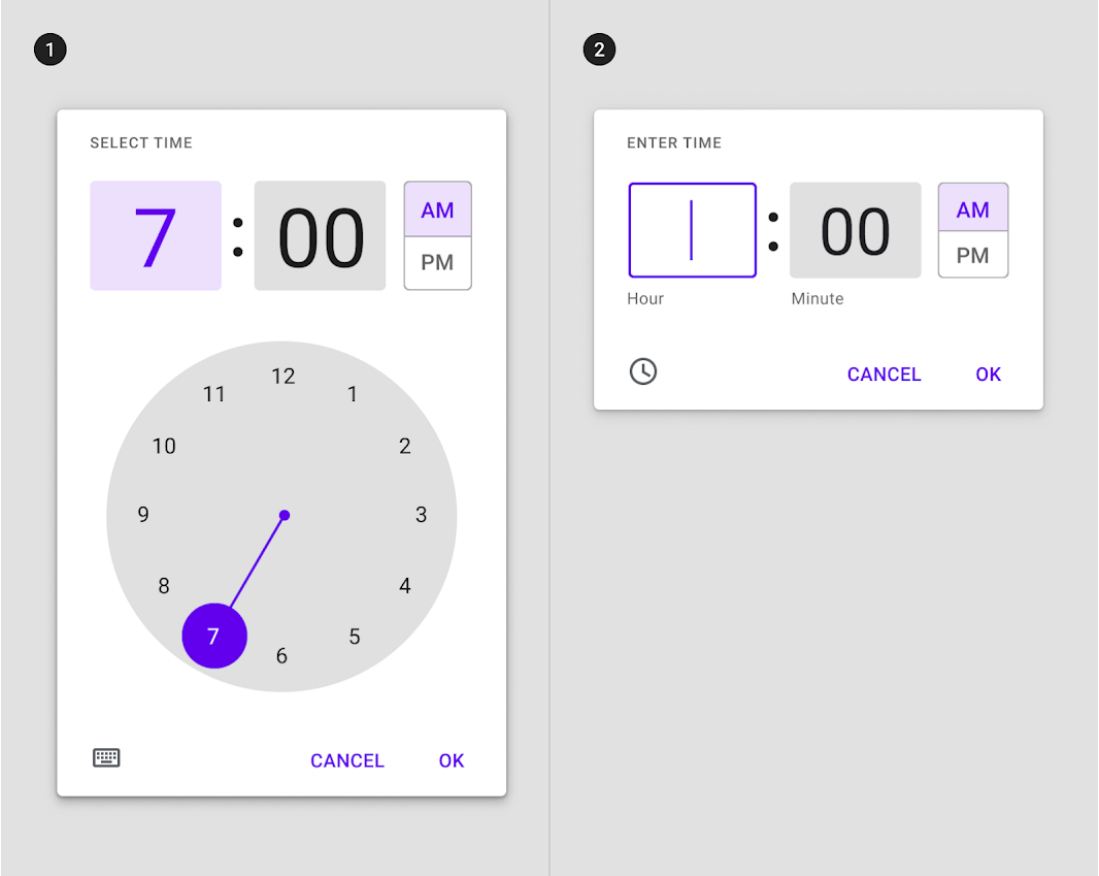
Příloha C

Komponenty Material Design

Příloha obsahuje příklady Google Material Design komponent pro výběr data a času. Seznam a popis všech komponent lze nalézt na [98].



Obrázek C.1 Material Date Picker [99]

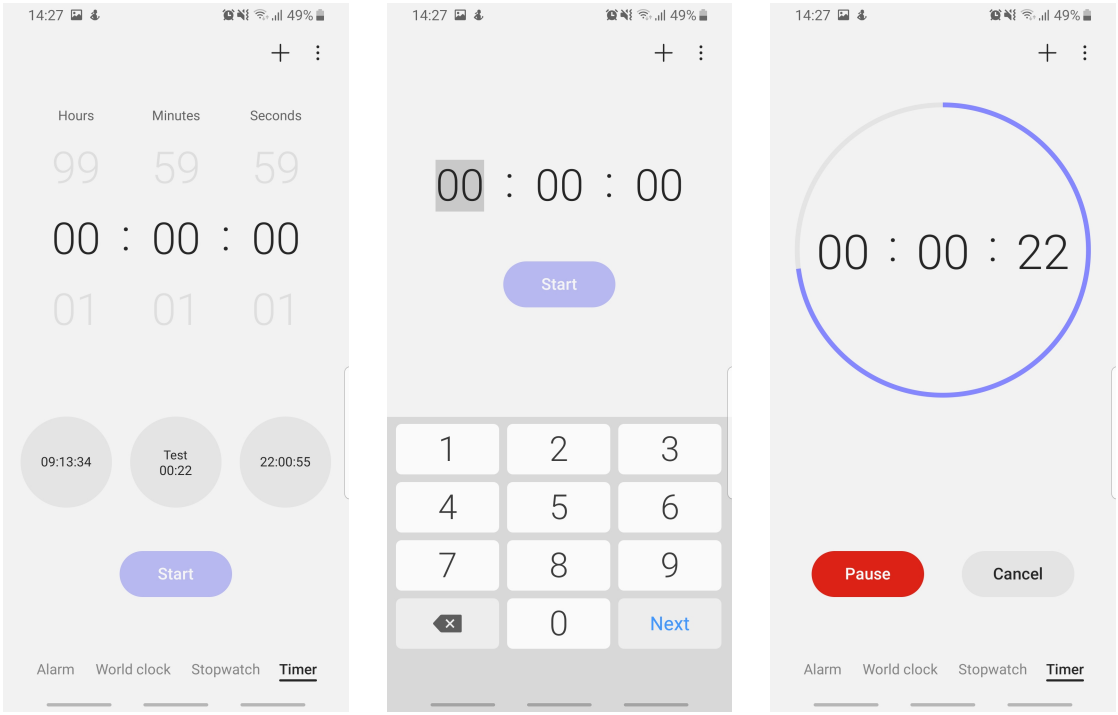


Obrázek C.2 Material Time Picker [100]

Příloha D

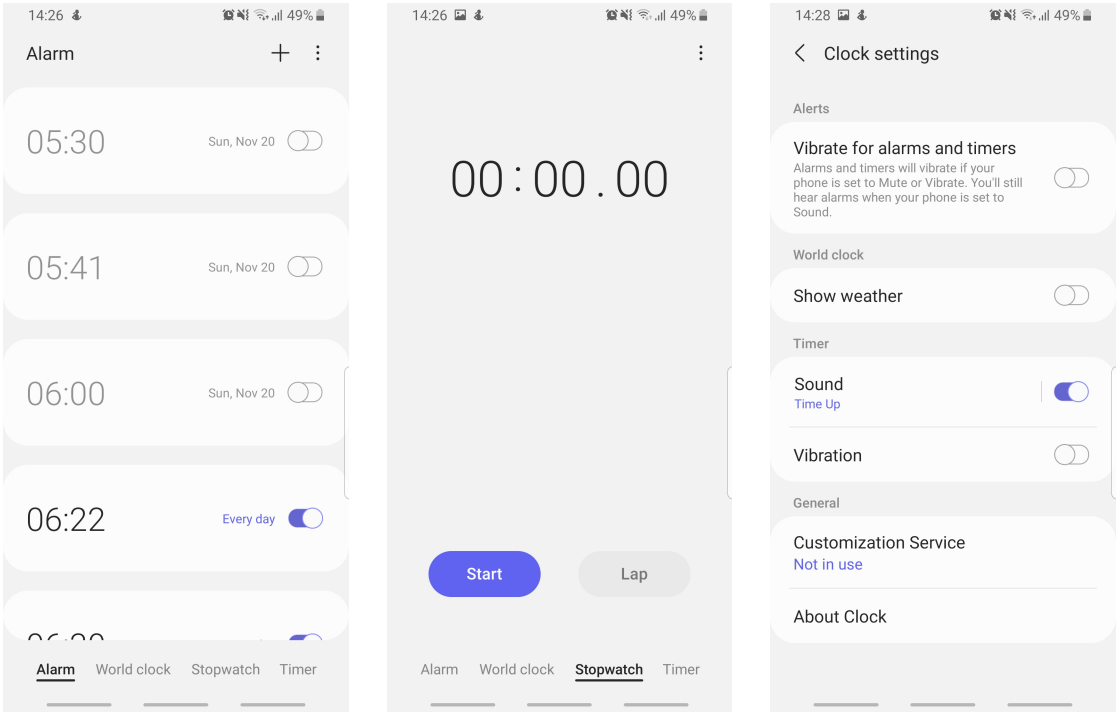
Příklady uživatelského rozhraní

Příloha obsahuje příklady uživatelského rozhraní aplikací s funkcí budíku, stopek a časovače. Komponenty těchto UI sloužily jako inspirace při návrhu UI aplikace NixieApp.



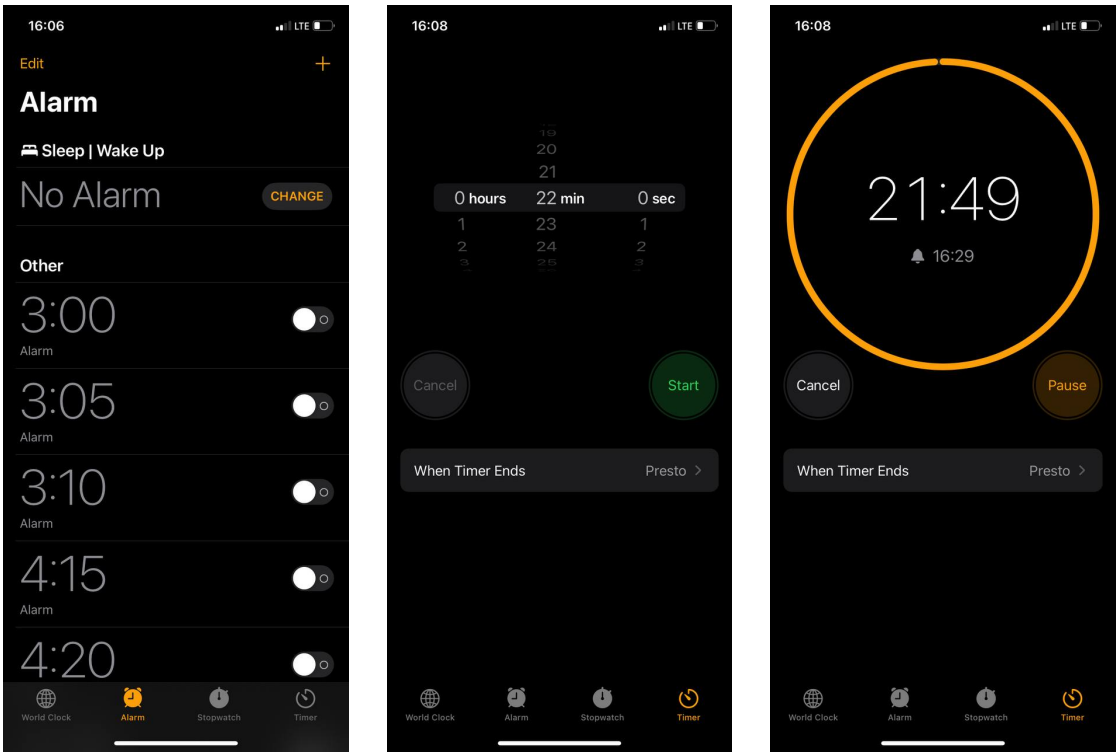
(a) Časovač – výchozí stav **(b)** Časovač – výběr času **(c)** Časovač – běžící stav

Obrázek D.1 Snímky časovače v nativní aplikaci Clock na telefonu – Samsung Galaxy S8



(a) Budíky **(b)** Stopky **(c)** Nastavení

Obrázek D.2 Snímky nativní aplikace Clock na telefonu – Samsung Galaxy S8



(a) Budíky **(b)** Časovač – výběr času **(c)** Časovač – běžící

Obrázek D.3 Snímky nativní aplikace Clock na operačním systému – iOS 16

Bibliografie

1. LAY, Georg-Johann. *Nixie tube Telefunken ZM1210 (without coating) operating* [online]. [cit. 2022-06-22]. Dostupné z: `https://commons.wikimedia.org/wiki/File:ZM1210-operating.jpg`.
2. BECHYŇOVÁ, Martina. *Software pro digitronové hodiny*. Praha, 2022. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
3. VERNER, David. *Mobilní aplikace pro ovládání digitronových hodin*. Praha, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
4. HEYDON, Robin. *Bluetooth Low Energy: The Developer's Handbook*. Upper Saddle River: Pearson Education, 2013. ISBN 978-0-13-288836-3.
5. COLEMAN, Chris. *A Practical Guide to BLE Throughput* [online]. [cit. 2022-09-05]. Dostupné z: `https://interrupt.memfault.com/blog/ble-throughput-primer`.
6. COLEMAN, Chris. *A Practical Guide to BLE Throughput* [online]. [cit. 2022-09-05]. Dostupné z: `https://www.btframework.com/connparams.htm`.
7. THROUGH, Punch. *Maximizing BLE Throughput on iOS and Android* [online]. [cit. 2022-09-05]. Dostupné z: `https://punchthrough.com/maximizing-ble-throughput-on-ios-and-android`.
8. TEXAS INSTRUMENTS. *Generic Access Profile (GAP)* [online]. [cit. 2022-09-04]. Dostupné z: `https://software-dl.ti.com/lprf/simplelink_cc2640r2_latest/docs/blestack/ble_user_guide/html/ble-stack-3.x/gap.html`.
9. JACOBT_81. *Connection Parameter Update Requests* [online]. [cit. 2022-09-03]. Dostupné z: `https://community.infineon.com/t5/Resource-Library/Connection-Parameter-Update-Requests/ta-p/246944`.
10. DERHGAWEN, Ashish. *Maximizing BLE Throughput Part 4: Everything You Need to Know* [online]. [cit. 2022-09-05]. Dostupné z: `https://punchthrough.com/ble-throughput-part-4/`.
11. GOOGLE LLC. *Bluetooth Low Energy* [online]. [cit. 2022-08-28]. Dostupné z: `https://source.android.com/docs/core/connect/bluetooth/ble`.
12. APPLE INC. *Core Bluetooth* [online]. [cit. 2022-09-03]. Dostupné z: `https://developer.apple.com/documentation/corebluetooth`.
13. APPLE INC. *Using the correct Bluetooth LE Advertising and Connection Parameters for a stable connection* [online]. [cit. 2022-09-05]. Dostupné z: `https://developer.apple.com/library/archive/qa/qa1931/_index.html`.

14. MICROSOFT CORPORATION. *Bluetooth version and profile support in previous Windows versions* [online]. [cit. 2022-09-03]. Dostupné z: https://docs.microsoft.com/en-us/windows-hardware/drivers/bluetooth/bluetooth-support-in-previous-windows-versions.
15. INTEL CORPORATION. *Intel Wireless Bluetooth for Windows 7* [online]. [cit. 2022-09-03]. Dostupné z: https://www.intel.com/content/www/us/en/download/16806/intel-wireless-bluetooth-for-windows-7.html.
16. JANC, Szymon. *Doing Bluetooth Low Energy on Linux* [online]. [cit. 2022-09-03]. Dostupné z: https://elinux.org/images/3/32/Doing_Bluetooth_Low_Energy_on_Linux.pdf.
17. BEAUFORT, François. *Communicating with Bluetooth devices over JavaScript* [online]. [cit. 2022-09-03]. Dostupné z: https://web.dev/bluetooth/.
18. MOZILLA CORPORATION. *Web Bluetooth API* [online]. [cit. 2022-09-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API.
19. WEBBLUETOOTHCG. *Implementation Status* [online]. [cit. 2022-09-03]. Dostupné z: https://github.com/WebBluetoothCG/web-bluetooth/blob/main/implementation-status.md.
20. GEORGIEV, Deyan. *60 Browser Usage Statistics, Facts, and Trends* [online]. [cit. 2022-09-06]. Dostupné z: https://review42.com/resources/browser-usage-statistics.
21. CIMPANU, Catalin. *Apple declined to implement 16 Web APIs in Safari due to privacy concerns* [online]. [cit. 2022-09-05]. Dostupné z: https://www.zdnet.com/article/apple-declined-to-implement-16-web-apis-in-safari-due-to-privacy-concerns/.
22. PNN SOFT. *Bluefy – Web BLE Browser* [online]. [cit. 2023-01-03]. Dostupné z: https://apps.apple.com/us/app/bluefy-web-ble-browser/id1492822055.
23. GREENPARKSOFTWARE. *WebBLE* [online]. [cit. 2022-09-06]. Dostupné z: https://apps.apple.com/us/app/webble/id1193531073.
24. *Směrnice Evropského parlamentu a Rady 2000/84/ES ze dne 19. ledna 2001 o úpravě letního času* [online]. [cit. 2023-01-16]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/?qid=1477733776604&uri=CELEX:32000L0084.
25. BARTÍK, Matěj. *Digitronové hodiny (Nixie clock) - specifikace pro účely BP* [online]. [cit. 2022-04-20]. Dostupné z: https://docs.google.com/document/d/1xetzb_nKhZtPHte_wYI1smfKd7ueFHZZWfDHvDGd1Fs/edit?usp=sharing.
26. BLUETOOTH SIG, INC. *GATT Specification Supplement 6* [online]. [cit. 2023-01-16]. Dostupné z: https://www.bluetooth.com/specifications/specs/gatt-specification-supplement-6/.
27. ALI, Iqram. *Role of RTC(Real Time Clock) in Embedded Devices* [online]. [cit. 2023-01-18]. Dostupné z: https://iqramali.medium.com/role-of-rtc-real-time-clock-in-embedded-devices-35dbd2e8f9e7.
28. STMICROELECTRONICS. *Using the hardware real-time clock (RTC) in STM32 F0, F2, F3, F4 and L1 series of MCUs* [online]. [cit. 2022-07-20]. Dostupné z: https://www.st.com/resource/en/application_note/an3371-using-the-hardware-realtime-clock-rtc-in-stm32-f0-f2-f3-f4-and-l1-series-of-mcus-stmicroelectronics.pdf.
29. BENSCH, Peter. *How to convert Accuracy(ppm) to CALP&CALM values when calibrate RTC* [online]. [cit. 2022-07-21]. Dostupné z: https://community.st.com/s/question/0D53W00001GiKi8SAF/how-to-convert-accuracyppm-to-calpcalm-values-when-calibrate-rtc-.

30. MILLS, David L. *Computer network time synchronization: the network time protocol*. Boca Raton: CRC Press, 2006. ISBN 9780849358050.

31. STAFF, Editorial. *SNTP: All You Need To Know About Simple Network Time Protocol* [online]. [cit. 2022-06-20]. Dostupné z: https://timetoolsltd.com/ntp/sntp-overview/.

32. SHARMA, Neeraj. *Native, Web, Hybrid, or Cross-Platform – How to Choose the Right Mobile App for Your Business?* [online]. [cit. 2022-06-15]. Dostupné z: https://www.kellton.com/kellton-tech-blog/how-to-choose-the-right-mobile-app-for-your-business.

33. GOOGLE LLC. *Android's Kotlin-first approach* [online]. [cit. 2022-06-16]. Dostupné z: https://developer.android.com/kotlin/first.

34. SADOWSKA, Paulina. *CompileSdkVersion and targetSdkVersion — what is the difference?* [online]. [cit. 2022-07-10]. Dostupné z: https://proandroiddev.com/compilesdkversion-and-targetsdkversion-what-is-the-difference-b4227c663ba8.

35. GOOGLE LLC. *Požadavky na cílovou úroveň rozhraní API pro aplikace na Google Play* [online]. [cit. 2022-08-27]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/11926878?hl=cs.

36. APPLE INC. *Swift* [online]. [cit. 2022-07-20]. Dostupné z: https://developer.apple.com/swift/.

37. APPLE INC. *App Store submission requirement starts April 25* [online]. [cit. 2022-09-05]. Dostupné z: https://developer.apple.com/news/?id=2t1chhp3.

38. DESIGNRUSH. *The Ultimate Guide To Hybrid App Development* [online]. [cit. 2022-06-15]. Dostupné z: https://www.designrush.com/agency/mobile-app-design-development/trends/hybrid-app-development.

39. MAHAJAN, Parth. *Hybrid Application: Native-like Experience with WebView* [online]. [cit. 2022-06-15]. Dostupné z: https://medium.com/1mgofficial/hybrid-application-native-like-experience-with-webview-9896f61881cb.

40. MORTKA, Adam. *PWA vs SPA vs Native Mobile Application — What's Right?* [online]. [cit. 2022-06-17]. Dostupné z: https://medium.com/pgs-software/pwa-vs-spa-vs-native-mobile-application-whats-right-47f1b071ad09.

41. LEPAGE, Pete; STEINER, Thomas; BEAUFORT, François. *Add a web app manifest* [online]. [cit. 2022-06-18]. Dostupné z: https://web.dev/add-manifest/.

42. RENZULLI, Demian; GUAN, Andrew. *Workers overview* [online]. [cit. 2022-06-19]. Dostupné z: https://web.dev/workers-overview/.

43. MOZILLA CORPORATION. *Introduction to progressive web apps* [online]. [cit. 2022-06-19]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction.

44. GOOGLE LLC. *Introduction to progressive web apps* [online]. [cit. 2022-06-24]. Dostupné z: https://web.dev/learn/pwa/progressive-web-apps/.

45. BRAINHUB. *PWA Examples: 5 Well-known Companies that Succeed with PWAs* [online]. [cit. 2022-06-19]. Dostupné z: https://brainhub.eu/library/progressive-web-apps-examples.

46. VERNER, David. *Mobilní aplikace pro ovládání digitronových hodin* [online]. [cit. 2022-09-04]. Dostupné z: https://dspace.cvut.cz/handle/10467/95573.

47. VERNER, David. *NixieClock - použití* [online]. [cit. 2022-11-17]. Dostupné z: https://www.youtube.com/watch?v=q8VVkQbQOqA.

48. JUUL LABS, INC. *Kable* [online]. [cit. 2022-09-03]. Dostupné z: https://github.com/JuulLabs/kable.
49. ANVER, Aslam. *SNTPClient for Android* [online]. [cit. 2022-09-04]. Dostupné z: https://github.com/aslamanver/sntp-client-android.
50. MICROSOFT CORPORATION. *Co je .NET MAUI?* [online]. [cit. 2022-11-06]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/maui/what-is-maui.
51. META PLATFORMS, INC. *React Native* [online]. [cit. 2022-11-18]. Dostupné z: https://reactnative.dev/.
52. GOOGLE LLC. *Flutter* [online]. [cit. 2022-11-18]. Dostupné z: https://flutter.dev/.
53. APACHE SOFTWARE FOUNDATION. *Cordova Overview* [online]. [cit. 2022-11-18]. Dostupné z: https://cordova.apache.org/docs/en/11.x/guide/overview/index.html.
54. CAPACITOR. *A cross-platform native runtime for web apps.* [online]. [cit. 2022-11-18]. Dostupné z: https://capacitorjs.com/.
55. CAPACITOR. *Capacitor Android Documentation* [online]. [cit. 2023-01-17]. Dostupné z: https://capacitorjs.com/docs/android.
56. BELINSKI, Eugene. *iOS version usage* [online]. [cit. 2023-01-17]. Dostupné z: https://iosref.com/ios-usage.
57. STEVENS, Adrian. *Monkey.BluetoothLE* [online]. [cit. 2022-11-05]. Dostupné z: https://github.com/xamarin/Monkey.BluetoothLE.
58. MCIECHANOWICZ. *React Native BLE library* [online]. [cit. 2022-11-05]. Dostupné z: https://github.com/dotintent/react-native-ble-plx.
59. SINIGAGLIA, Marco. *React Native BLE Manager* [online]. [cit. 2022-11-05]. Dostupné z: https://github.com/innoveit/react-native-ble-manager.
60. PAULDEMARCO.COM. *FlutterBlue* [online]. [cit. 2022-11-05]. Dostupné z: https://pub.dev/packages/flutter_blue.
61. POLIDEA.COM. *FlutterBleLib* [online]. [cit. 2022-11-05]. Dostupné z: https://pub.dev/packages/flutter_ble_lib.
62. MEETHUE.COM. *Flutter reactive BLE library* [online]. [cit. 2022-11-05]. Dostupné z: https://pub.dev/packages/flutter_reactive_ble.
63. DROPMEIRL.COM. *Flutter Web Bluetooth* [online]. [cit. 2022-11-05]. Dostupné z: https://pub.dev/packages/flutter_web_bluetooth.
64. PEITSCH, Philip. *BLE Central Plugin for Apache Cordova* [online]. [cit. 2022-11-05]. Dostupné z: https://github.com/don/cordova-plugin-ble-central.
65. WESPI, Patrick. *Capacitor plugin for Bluetooth Low Energy* [online]. [cit. 2022-11-05]. Dostupné z: https://github.com/capacitor-community/bluetooth-le.
66. IONIC. *Introduction to Ionic* [online]. [cit. 2022-11-18]. Dostupné z: https://ionicframework.com/docs.
67. GOOGLE LLC. *What is Angular?* [online]. [cit. 2022-11-18]. Dostupné z: https://angular.io/guide/what-is-angular.
68. META PLATFORMS, INC. *React* [online]. [cit. 2022-11-18]. Dostupné z: https://reactjs.org/.
69. YOU, Evan. *The Progressive JavaScript Framework* [online]. [cit. 2022-11-18]. Dostupné z: https://vuejs.org/guide/introduction.html.
70. MICROSOFT CORPORATION. *TypeScript for JavaScript Programmers* [online]. [cit. 2022-11-18]. Dostupné z: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html.

71. STACK OVERFLOW. *2022 Developer Survey* [online]. [cit. 2023-01-17]. Dostupné z: https://survey.stackoverflow.co/2022/.
72. ASIRI, Sidath. *Flux and Redux* [online]. [cit. 2023-01-17]. Dostupné z: https://medium.com/@sidathasiri/flux-and-redux-f6c9560997d7.
73. ABRAMOV, Dan. *Redux Fundamentals* [online]. [cit. 2023-01-17]. Dostupné z: https://redux.js.org/tutorials/fundamentals/part-1-overview.
74. SYMPHONY. *Redux Toolkit: Redux with Less Code Part I* [online]. [cit. 2022-11-15]. Dostupné z: https://symphony.is/media/img2.png.
75. ABRAMOV, Dan. *Writing Logic with Thunks* [online]. [cit. 2023-01-20]. Dostupné z: https://redux.js.org/usage/writing-logic-thunks.
76. MUI. *MUI for Figma v5.9.0 (Community)* [online]. [cit. 2022-11-14]. Dostupné z: https://www.figma.com/community/file/912837788133317724.
77. ANDERSON, Shaun. *What are the best screen sizes for responsive web design in 2022?* [online]. [cit. 2023-01-17]. Dostupné z: https://www.hobo-web.co.uk/best-screen-size/.
78. PRESTON, Lee. *Mobile screen sizes for 2022 based on data from 2021* [online]. [cit. 2023-01-17]. Dostupné z: https://worship.agency/mobile-screen-sizes-for-2022-based-on-data-from-2021.
79. *iOS Resolution* [online]. [cit. 2023-01-17]. Dostupné z: https://www.ios-resolution.com/.
80. *Screen Sizes* [online]. [cit. 2023-01-17]. Dostupné z: https://screensiz.es/phone.
81. RICHTER, Felix. *How Long Does Apple Support Older iPhone Models?* [online]. [cit. 2023-01-17]. Dostupné z: https://www.statista.com/chart/5824/ios-iphone-compatibility/.
82. VIZELKA, Dimitri. *NixieApp* [online]. [cit. 2023-01-03]. Dostupné z: https://www.figma.com/file/jZrV9XzicD8WdGktfKQzqc/NixieApp?node-id=5571%5C%3A70215&t=3FRvVHtL%5C-rZGZahxP-1.
83. CHROMIUM. *Issue 974879: Implement persistent Web Bluetooth permissions* [online]. [cit. 2023-01-14]. Dostupné z: https://bugs.chromium.org/p/chromium/issues/detail?id=974879.
84. CHROMIUM. *Issue 1173186: DOMException: Bluetooth Device is no longer in range.* [online]. [cit. 2023-01-14]. Dostupné z: https://bugs.chromium.org/p/chromium/issues/detail?id=1173186.
85. RUAS, Rômulo; NAGY, Gergely. *cordova-plugin-sntp* [online]. [cit. 2022-08-20]. Dostupné z: https://github.com/hub9/cordova-plugin-sntp.
86. SCHULTZ, Stephan; MIRTSCHIN, Marvin; NIKLAS. *RxAndroidBleServer* [online]. [cit. 2022-10-22]. Dostupné z: https://github.com/neXenio/RxAndroidBleServer.
87. NORDIC SEMICONDUCTOR ASA. *nRF Connect for Mobile* [online]. [cit. 2023-01-19]. Dostupné z: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp.
88. KOVAČIČ, Damjan. *[Bug] GATT Server is disconnected. Cannot retrieve services. (Re)connect first with device.gatt.connect.* [online]. [cit. 2022-10-12]. Dostupné z: https://github.com/pybricks/support/issues/312.
89. BEAUFORT, François. *Web Bluetooth / Automatic Reconnect Sample* [online]. [cit. 2022-10-12]. Dostupné z: https://googlechrome.github.io/samples/web-bluetooth/automatic-reconnect.html.

90. ARGENOX TECHNOLOGIES LLC. *UNDERSTANDING BLE DISCONNECTIONS* [online]. [cit. 2022-10-13]. Dostupné z: `https://www.argenox.com/blog/understanding-ble-disconnections/`.
91. VIZELKA, Dimitri. *Web Bluetooth - Windows 11 disconnection* [online]. [cit. 2022-12-30]. Dostupné z: `https://stackoverflow.com/questions/74116580/web-bluetooth-windows-11-disconnection`.
92. BECHYNOVA, Martina. *Control Software for Nixie Clock* [online]. [cit. 2023-01-01]. Dostupné z: `https://gitlab.com/Martina-Bechynova/control-software-for-nixie-clock`.
93. VIZELKA, Dimitri. *Control Software for Nixie Clock* [online]. [cit. 2023-01-01]. Dostupné z: `https://gitlab.com/vizeldim/control-software-for-nixie-clock`.
94. VIZELKA, Dimitri. *Bluefy - notifications support* [online]. [cit. 2023-01-02]. Dostupné z: `https://github.com/capacitor-community/bluetooth-le/issues/470`.
95. BLUEPIXEL TECHNOLOGIES. *BLE Scanner (Connect and Notify)* [online]. [cit. 2023-01-19]. Dostupné z: `https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner`.
96. VIZELKA, Dimitri. *NixieClock* [online]. [cit. 2023-01-03]. Dostupné z: `https://www.youtube.com/playlist?list=PLGEnzSvFWC4rWSqGWES65Qs4IXXvgObDn`.
97. VIZELKA, Dimitri. *NixieApp* [online]. [cit. 2023-01-19]. Dostupné z: `https://vizeldim.gitlab.io/nixieapp/`.
98. GOOGLE LLC. *Material Components* [online]. [cit. 2022-09-04]. Dostupné z: `https://material.io/components`.
99. GOOGLE LLC. *Date pickers* [online]. [cit. 2022-09-04]. Dostupné z: `https://material.io/components/date-pickers`.
100. GOOGLE LLC. *Time pickers* [online]. [cit. 2022-09-04]. Dostupné z: `https://material.io/components/time-pickers`.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `app` ..... zdrojové kódy implementace aplikace NixieApp
  - `emulator` ..... zdrojové kódy implementace emulátoru
  - `sntp` ..... zdrojové kódy implementace SNTP pluginu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `media`
  - `figma` ..... návrh uživatelského rozhraní aplikace NixieApp
  - `screenshots` ..... snímky obrazovky aplikace NixieApp
  - `video` ..... videoukázky aplikace NixieApp
  - `links.txt` ..... odkazy na návrh a videoukázky NixieApp
- `bleCaptures` ..... adresář se zachycenou BLE komunikací
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF