FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Mobilní aplikace pro ovládání digitronových hodin (Nixie clock) přes Bluetooth
Student:	Ivan Alekhin
Vedoucí:	Ing. Matěj Bartík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

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. Michal Valenta, Ph.D. dne 27. září 2022 v Praze.

Bakalářská práce

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

Ivan Alekhin

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

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat svému vedoucímu za to, že vymyslel tento úkol a poskytl mi podporu. A mé rodině a přátelům, kteří vydrželi nespočet nocí, kdy jsem pracoval na dokončení této 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 4. ledna 2023 .....

Abstrakt

Cílem této bakalářské práce bylo vyvinout mobilní aplikaci pro ovládání digitálních nixie hodin. Aplikace měla běžet na platformách Android i iOS a byla navržena tak, aby podporovala řadu funkcí a vlastností, které uživatelům umožní nastavit čas a datum, nastavit budíky, ovládat stopky a časovač a měnit různé možnosti na nixie hodiny. Při implementaci aplikace jsem vyhodnotil dostupné frameworky a jazyky dostupné pro vytvoření multiplatformní mobilní aplikace a prozkoumal jsem komunikační protokol Bluetooth Low Energy používaný k navázání spojení mezi hodinami a aplikací. K otestování aplikace jsem použil kombinaci testování jednotek a ručního testování a byl jsem schopen prokázat, že aplikace je stabilní, spolehlivá a schopná splnit funkční požadavky, které byly stanoveny na začátku projektu. Výsledkem této práce je mobilní aplikace pro ovládání digitálních nixie hodin pomocí komunikačního protokolu Bluetooth Low Energy.

Klíčová slova Bluetooth, Android, iOS, digitron, BLE, hodiny, Flutter, Dart

Abstract

The goal of this bachelor thesis was to develop a mobile application for controlling a digital nixie clock. The app was intended to run on both Android and iOS platforms, and was designed to support a range of functions and features that would allow users to set the time and date, set alarms, control a stopwatch and timer, and change various options on the nixie clock. To implement the app, I evaluated available frameworks and languages available for creating a crossplatform mobile app, and researched Bluetooth Low Energy communication protocol used to establish a connection between the clock and the app. In order to test the app, I used a combination of unit testing and manual testing, and I was able to demonstrate that the app was stable, reliable, and capable of meeting the functional requirements that had been set out at the outset of the project. The result of this thesis is the mobile app for controlling a digital nixie clock using Bluetooth Low Energy communication protocol.

Keywords Bluetooth, Android, iOS, nixie tube, BLE, clock, Flutter, Dart

Seznam zkratek

BLE	Bluetooth Low Energy
GATT	Generic Attribute Profile
IDE	Integrated Development Environment
SDK	Software Development Kit
IoT	Internet of things
UUID	Universally Unique Identifier
UI	User Interface
API	Application Programming Interface

Úvod

Nixie tubes jsou digitrony, které se běžně používaly v 50. a 60. letech k zobrazování číselných informací. Skládají se z řady katod ve tvaru číslic nebo jiných symbolů a anody a fungují tak, že se na anodu přivede vysoké napětí, což způsobí, že mezi anodou a jednou z katod vznikne elektrický výboj, který rozsvítí odpovídající číslo nebo symbol.

Nixie hodiny jsou digitální hodiny, které používají nixie tubes k zobrazení času. Tyto hodiny byly v minulosti oblíbené, ale od té doby se přestaly používat kvůli dostupnosti levnějších a efektivnějších zobrazovacích technologií. V poslední době však zaznamenaly opětovný nárůst popularity díky jejich výrazné retro estetice[1].

Jedním z nedostatků tradičních nixie hodin je jejich nepříliš pohodlné ovládání. Většina nixie hodin vyžaduje ruční nastavení času nebo změnu nastavení. To může ztížit používání hodin v moderním prostředí.

Mobilní aplikace, kterou jsem vyvinul, si klade za cíl tyto problémy řešit tím, že uživatelům umožňuje ovládat své nixie hodiny přes Bluetooth. To umožňuje větší pohodlí a flexibilitu při používání hodin, protože uživatelé mohou snadno nastavit čas a další parametry ze svých mobilních zařízení.

V této bakalářské práci popíšu proces implementace této mobilní aplikace, včetně popisu komunikační technologie Bluetooth, analýzy komunikačního protokolu používaného hodinami nixie, výběru frameworku a programovacího jazyka pro implementaci a případných další relevantní aspekty projektu. Výslednou aplikaci také otestuji a zdokumentuji, abych se ujistil, že je spolehlivá a snadno se používá.

Obrázek 1 Jeden z prototypů nixie hodin

Kapitola 1
Analýza

V této části rozeberu technologie a požadavky, které byly použity a zohledněny při vývoji mobilní aplikace pro ovládání nixie hodin přes Bluetooth. To bude zahrnovat podrobný úvod do komunikačního protokolu Bluetooth a toho, jak se používá k umožnění komunikace mezi mobilním zařízením a hodinami nixie. Budu také diskutovat o požadavcích, které byly na aplikaci kladeny, včetně konkrétních případů použití a funkčnosti, na které bylo během vývoje zaměřeno. Tato analýza poskytne základ pro návrh a implementaci aplikace v dalších částech práce.

1.1 Programovací jazyk a framework

Existuje několik programovacích jazyků a frameworků, které mohly být použity k implementaci mobilní aplikace pro ovládání hodin nixie přes Bluetooth.

1.1.1 Java a Android SDK

Java je oblíbený programovací jazyk pro vývoj Androidu a Android Studio je oficiální integrované vývojové prostředí (IDE) pro Android. Výhodou používání Java a Android Studio je, že jsou dobře zavedené a široce podporované a existuje velká komunita vývojářů, kteří mohou poskytnout pomoc a zdroje. Nevýhodou je, že jej lze použít pouze pro vývoj aplikací pro Android. Java jako jazyk lze použít pro vytváření aplikací pro iOS, ale bylo by potřeba použít samostatnou sadu SDK, což z ní dělá prakticky samostatnou aplikaci s jinou kódovou základnou[2].

1.1.2 Kotlin a Android Studio

Kotlin je programovací jazyk, který je plně kompatibilní s Javou a lze jej použít k vytváření aplikací pro Android. Je to oficiálně podporovaný jazyk pro vývoj Android a lze jej používat ve spojení s Android Studio. Výhodou použití Kotlinu je, že jde oproti Javě o modernější a výstižnější jazyk, který může usnadnit psaní a údržbu kódu. Nevýhodou je, že Kotlin, stejně jako Java, je zaměřen na Android. Podporuje určitou úroveň interoperability s Swift, což umožňuje kombinovat kód Swift specifický pro iOS s kódem Kotlin specifickým pro Android. To by však stále efektivně znamenalo vytvoření dvou samostatných aplikací s oddělenými kódovými bázemi[2].

1.1.3 Swift

Swift je vyvinutý programovací jazyk od společnosti Apple speciálně pro vytváření aplikací pro iOS a macOS. Xcode je oficiální IDE pro vývoj iOS a macOS. Výhodou použití Swift a Xcode

je, že jsou optimalizovány pro vytváření vysoce výkonných nativních aplikací pro platformy iOS a macOS. Nevýhodou je, že jsou dostupné pouze na platformách Apple[3].

1.1.4 Dart a Flutter

Dart je programovací jazyk vyvinutý společností Google, který je navržen tak, aby byl rychlý, flexibilní a snadno se naučil. Flutter je framework pro vývoj mobilních aplikací, který používá Dart jako svůj primární programovací jazyk. Výhodou použití Dart a Flutter je, že umožňují vývoj multiplatformních aplikací, které lze spustit na Androidu i iOS. Flutter má také funkci hot reload, která umožňuje vývojářům vidět změny, které v kódu provedou, v reálném čase, což může být velmi užitečné pro ladění a testování. Nevýhodou je, že Flutter je relativně nový framework, takže může být k dispozici méně zdrojů a podpory komunity ve srovnání se zavedenějšími technologiemi[4].

1.1.5 JavaScript a React Native

JavaScript je populární programovací jazyk, který se běžně používá pro vývoj webových aplikací, a React Native je framework pro vývoj mobilních aplikací, který umožňuje vývojářům vytvářet nativní aplikace pro Android a iOS pomocí JavaScriptu a knihovny React. Výhodou použití JavaScriptu a React Native je, že umožňují vývoj multiplatformních aplikací, které lze spustit na Androidu i iOS, a mají velkou vývojářskou komunitu a širokou škálu dostupných zdrojů. Nevýhodou je, že aplikace React Native nemusí mít stejnou úroveň výkonu jako nativně vyvinuté aplikace[5].

1.2 Bluetooth

Bluetooth je bezdrátová komunikační technologie, která umožňuje zařízením propojovat se a vyměňovat si data na krátké vzdálenosti. Byl vyvinut v 90. letech minulého století jako náhrada datových kabelů RS-232, které se běžně používaly pro připojení zařízení jako klávesnice, myši a tiskárny k počítačům. Bluetooth pracuje ve frekvenčním pásmu 2,4 GHz a k přenosu dat mezi zařízeními využívá vysokofrekvenční spojení[6].

Technologie Bluetooth ušla od svého vzniku dlouhou cestu. Vyvinul se z původní specifikace Bluetooth 1.0, která měla maximální rychlost přenosu dat 1 Mbps, k aktuální specifikaci Bluetooth 5.2, která má maximální rychlost přenosu dat 2 Mbps a dosah až 400 metrů. Bluetooth byl také upraven tak, aby podporoval širokou škálu aplikací a zařízení, včetně mobilních telefonů, notebooků, reproduktorů, sluchátek a dokonce i lékařských přístrojů a domácích spotřebičů.

Kromě využití ve spotřební elektronice se technologie Bluetooth používá také v průmyslových a lékařských aplikacích. Může být například použit pro připojení senzorů a dalších zařízení v továrně nebo skladu pro účely monitorování a kontroly nebo pro připojení lékařských zařízení, jako jsou monitory hladiny glukózy v krvi a inzulínové pumpy, k chytrým telefonům pro vzdálené monitorování a ovládání.

Celkově se technologie Bluetooth stala nedílnou součástí moderního života a poskytuje pohodlný a spolehlivý způsob připojení a ovládání široké škály zařízení a systémů. Jak se technologie neustále vyvíjí, je pravděpodobné, že najde ještě širší využití v různých aplikacích.

1.2.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) je varianta technologie Bluetooth, která je speciálně navržena pro aplikace s nízkou spotřebou. Byl představen ve specifikaci Bluetooth 4.0 a od té doby se stal oblíbenou volbou pro aplikace, které vyžadují dlouhou výdrž baterie nebo nízkou spotřebu energie[6].

Jedním z klíčových rozdílů mezi BLE a tradiční technologií Bluetooth je způsob, jakým řídí napájení. BLE používá jiný vysokofrekvenční protokol než tradiční Bluetooth, což umožňuje přenášet data s mnohem menší spotřebou energie. Díky tomu je ideální pro aplikace, kde je problémem spotřeba energie, jako jsou nositelná zařízení, chytré domácí spotřebiče a další zařízení internetu věcí (IoT).

BLE je také navrženo tak, aby podporovalo širokou škálu konfigurací zařízení a případů použití. Může pracovat v různých režimech, včetně režimu vysílání, který umožňuje zařízení odesílat data do jakéhokoli jiného zařízení BLE v dosahu, a režimu připojení, který umožňuje dvěma zařízením vytvořit zabezpečené obousměrné spojení. Díky tomu je BLE flexibilní a všestranná technologie, kterou lze použít v různých aplikacích.

V kontextu této práce lze BLE použít k ovládání nixie hodin vytvořením spojení mezi mobilní zařízení a hodiny a odesílání řídicích příkazů přes linku BLE. To umožňuje uživatelům nastavovat čas, měnit nastavení zobrazení a provádět další akce na hodinách ze svých mobilních zařízení. Použití BLE pro tento účel umožňuje pohodlné a energeticky efektivní ovládání hodin, protože spojení BLE lze udržovat s minimální spotřebou energie.

Celkově je BLE výkonná a všestranná technologie, která se dobře hodí pro aplikace, které vyžadují nízkou spotřebu energie a flexibilní připojení zařízení. Jde o důležitý nástroj umožňující vývoj chytrých propojených zařízení, která lze snadno ovládat a monitorovat z mobilního zařízení.

1.2.2 Technické detaily BLE

Data a funkce, které jsou k dispozici na zařízení BLE, jsou reprezentovány charakteristikami a službami[7].

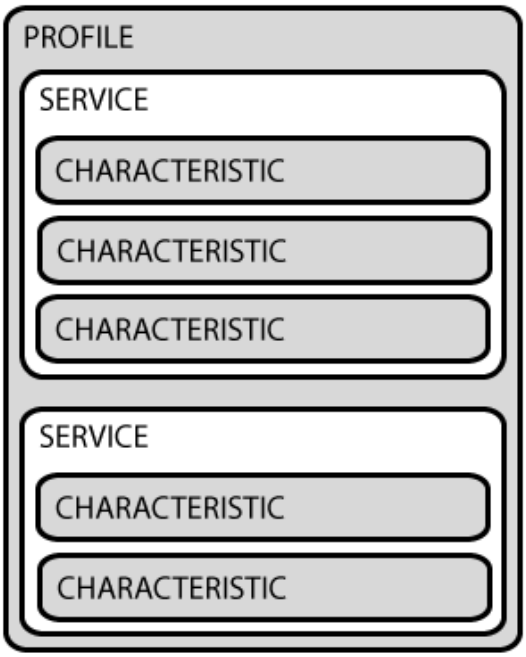
Charakteristika je hodnota, kterou lze číst nebo zapisovat zařízením BLE. Má jedinečný identifikátor, hodnotu a sadu vlastností, které definují, jak lze k charakteristice přistupovat a jak ji používat. Některé příklady charakteristik zahrnují data senzoru (např. teplota, vlhkost), řídicí parametry (např. úroveň jasu) a informace o zařízení (např. název výrobce, číslo modelu).

Služba je soubor jedné nebo více charakteristik, které souvisejí s konkrétní funkcí nebo funkcí zařízení BLE. Služby se používají k seskupování souvisejících charakteristik a usnadňují ostatním zařízením zjišťování a přístup k datům a funkcím, které jsou na zařízení BLE dostupné. Některé příklady služeb zahrnují službu baterie, která může zahrnovat vlastnosti pro čtení úrovně baterie a povolení upozornění, když se úroveň baterie změní, a službu srdeční frekvence, která může zahrnovat vlastnosti pro čtení srdeční frekvence a povolení oznámení při změně srdeční frekvence.

Charakteristiky a služby jsou na zařízení BLE uspořádány do hierarchie. Na nejvyšší úrovni má zařízení jeden nebo více profilů, které definují celkovou strukturu a účel zařízení. Každý profil obsahuje jednu nebo více služeb a každá služba obsahuje jednu nebo více charakteristik. Tato hierarchie umožňuje dalším zařízením strukturovaným a organizovaným způsobem zjišťovat a přistupovat k datům a funkcím, které jsou na zařízení BLE dostupné.

Z technického hlediska jsou charakteristiky a služby reprezentovány pomocí datových struktur GATT (Generic Atribute Profile). Datová struktura GATT se skládá z popisovače atributu, hodnoty atributu a sady oprávnění, která definují, jak lze k atributu přistupovat a jak jej používat. Popisovač atributu je jedinečný identifikátor, který se používá k odkazování na atribut, a hodnota atributu je skutečná data nebo funkce, které atribut představuje. Oprávnění definují, zda lze atribut číst, zapisovat nebo oznamovat, a zda je atribut povinný nebo volitelný[7].

Obrázek 1.1 Struktura GATT[8]



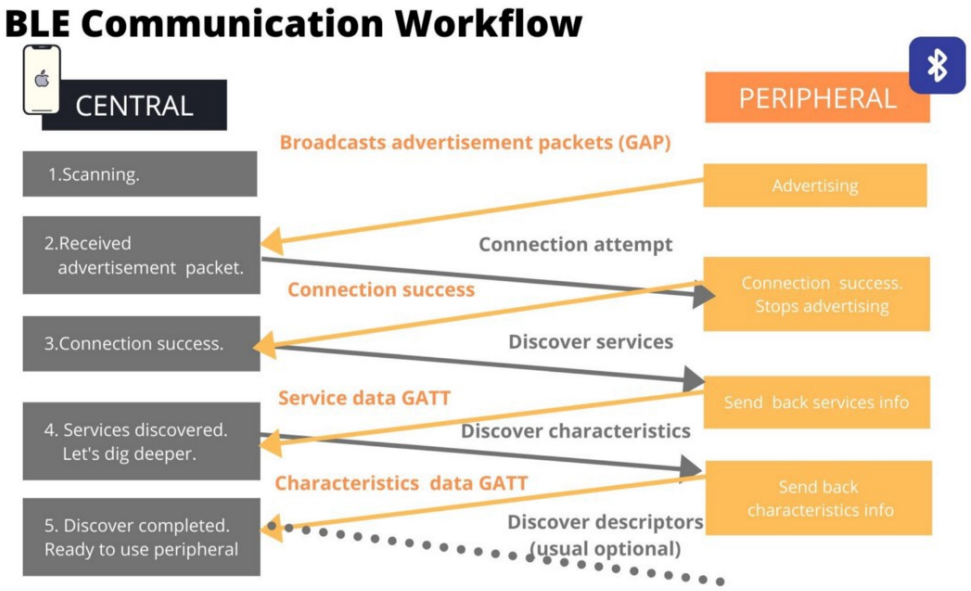
Každý atribut se skládá ze tří částí:

- **128-bitové UUID**, označující typ atributu[9],
- **16-bitový identifikátor**, který slouží k vyhledání a identifikaci atributu v tabulce atributů,
- **Hodnota**

Server vytváří jeden nebo více atributů, které ukládá v tabulce atributů a klient, který se připojuje k serveru, může číst tyto atributy a upravovat je pomocí příkazů:

- **Write Command** – zápis bez potvrzení,
- **Write Request** – zápis s potvrzením,
- **Read Request** – čtení

Obrázek 1.2 Pracovní postup protokolu BLE[10]



Stručně řečeno, vlastnosti a služby jsou slouží k reprezentaci dat a funkcí, které jsou k dispozici na zařízení BLE. Jsou organizovány do hierarchie profilů, služeb a charakteristik a jsou reprezentovány pomocí datových struktur GATT. To umožňuje ostatním zařízením strukturovaným a organizovaným způsobem zjišťovat a přistupovat k datům a funkcím, které jsou na zařízení BLE dostupné.

1.3 Funkční požadavky

Mobilní aplikace vyvíjená v rámci této bakalářské práce je určena pro ovládání digitálních nixie hodin. Samotný hardware nixie hodin byl vyvinut před touto prací a funkčnost hodin je předem určena a vystavena mi prostřednictvím rozhraní BLE[11].

Existující hardwarové funkce jsou:

- Zobrazení času - hodiny si pamatují nastavený čas a správně jej změní.
- Zobrazení data - podobně si hodiny umí datum zapamatovat a s postupem času ho správně upraví.
- Funguje jako stopky - režim zobrazení hodin se změní na zobrazení sekund uplynulých od začátku odpočítávání.
- Funguje jako časovač - podobně mohou hodiny odpočítávat od určitého počtu sekund a přehrají zvuk upozornění, když dosáhne nuly (jehož délku lze upravit)
- Funguje jako budík - hodiny si mohou zapamatovat až 10 časových razítek a může přehrát zvuk upozornění, když čas dosáhne.

Poté očekávané funkční požadavky aplikace jsou:

- Připojení k hodinám
- Nastavení času a data na hodinách

- Ovládání funkce stopek na hodinách
- Ovládání funkce časovače hodin
- Nastavení až 10 budíků na hodinách
- Změna různých nastavení hodin, jako je režim zobrazení.

Pro podporu těchto schopností bude muset mobilní aplikace implementovat řadu komunikačních funkcí BLE, včetně schopnosti vyhledávat zařízení BLE v okolí, připojit se k zařízení BLE, číst a zapisovat charakteristické hodnoty a povolit a zakázat upozornění.

Kromě komunikačních funkcí BLE bude mobilní aplikace také muset implementovat uživatelské rozhraní, které uživateli umožní provádět různé kontrolní úkoly uvedené výše. To bude pravděpodobně zahrnovat řadu obrazovek a ovládacích prvků pro nastavení času a data, nastavení budíků, spuštění a zastavení stopek a časovače a změnu možností na hodinách.

Kapitola 2
Návrh

Návrh mobilní aplikace pro ovládání digitálních nixie hodin byl kritickým aspektem procesu vývoje, protože určoval celkovou strukturu a funkčnost aplikace. V této části bakalářské práce popíšu klíčová designová rozhodnutí, která byla učiněna, a vysvětlím, jak přispěla k úspěšné implementaci aplikace.

2.1 Zvolený framework a jazyk

Při návrhu mobilní aplikace pro ovládání digitálních nixie hodin byla jedním z klíčových hledisek volba programovacího jazyka a vývojového frameworku. Pro vývoj mobilních aplikací je k dispozici několik možností, z nichž každá má své vlastní jedinečné přednosti a omezení.

Jednou z hlavních motivací pro výběr konkrétního jazyka a frameworku byla potřeba podporovat platformy Android i iOS, byť jen teoreticky. Mnoho dostupných možností podporuje pouze jednu platformu, což by omezilo potenciální uživatelskou základnu aplikace. Aby aplikaci mohlo používat co nejvíce lidí, bylo důležité vybrat jazyk a framework, které by mohly podporovat Android i iOS.

Po zvážení řady možností se objevili dva kandidáti: React Native a Flutter. Oba tyto frameworky umožňují vývoj multiplatformních aplikací, které mohou běžet na Androidu i iOS, což z nich udělalo ideální kandidáty pro tento projekt.

Nakonec padlo rozhodnutí jít s Flutterem. Jedním z klíčových faktorů byla skutečnost, že Flutter je kompilovaný jazyk, což znamená, že je rychlejší a efektivnější než mnoho jiných možností. To může vést k lepšímu výkonu a nižší spotřebě baterie aplikace[12], což jsou důležitá hlediska pro jakoukoli mobilní aplikaci.

Kromě výkonových výhod nabízí Flutter také vysokou úroveň bezpečnosti paměti a snadné použití. Framework používá typově bezpečný jazyk, který pomáhá předcházet běžným programovým chybám a zlepšuje spolehlivost kódu. Zahrnuje také širokou škálu vestavěných widgetů a nástrojů, které vývojářům usnadňují vytváření uživatelských rozhraní a implementaci komplexních funkcí.

Obrázek 2.1 Pořadí programovacích jazyků podle energetické účinnosti[12]

	Energy (J)
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

2.1.1 Použité knihovny

Jednou z klíčových výzev při vývoji aplikace byla potřeba použít knihovnu pro usnadnění komunikace BLE. Zatímco Flutter, framework, který byl vybrán pro tento projekt, obsahuje řadu vestavěných nástrojů a widgetů pro vytváření uživatelských rozhraní a implementaci různých typů funkcí, neposkytuje vestavěný způsob snadného použití BLE z krabice.

Aby bylo možné toto omezení překonat, bylo nutné použít knihovnu třetí strany, která by mohla poskytnout potřebné komunikační nástroje a funkce BLE. Po zvážení řady možností bylo rozhodnuto použít knihovnu Flutter Reactive BLE.

Knihovna Flutter Reactive BLE je open source, stabilní, dobře zdokumentovaná knihovna, kterou vyvinula společnost Phillips speciálně pro použití s jejími produkty Phillips Hue. Poskytuje řadu funkcí a tříd, které usnadňují vyhledávání zařízení BLE, připojení k zařízení BLE, čtení a zápis charakteristických hodnot a povolení a zakázání upozornění. Zahrnuje také podporu pro zpracování běžných chyb a výjimek BLE a také pro správu stavu připojení BLE[13].

Jedním z hlavních důvodů pro výběr knihovny Flutter Reactive BLE byla skutečnost, že ji vyvíjí a udržuje velká renomovaná společnost. To poskytuje vysokou úroveň důvěry v kvalitu a spolehlivost knihovny, protože je pravděpodobné, že bude dobře testována a podporována. Navíc skutečnost, že se knihovna stále aktivně vyvíjí a zdokonaluje, naznačuje, že pravděpodobně zůstane životaschopnou možností pro budoucí projekty.

2.2 Komunikační protokol

Při návrhu mobilní aplikace bylo nutné zohlednit komunikační protokol mezi aplikací a hodinami. Hardware a protokol pro hodiny nixie byly předdefinovány a mobilní aplikace musela být navržena tak, aby fungovala s těmito již existujícími specifikacemi.

Komunikační protokol používaný pro nixie hodiny je založen na profilu Generic Attribute (GATT), který definuje sadu služeb a charakteristik, které lze použít k reprezentaci různých typů dat a funkcí[11].

2.2.1 Hodiny

Service UUID - 00007500-f908-44b0-81b9-3450ca663f46

2.2.1.1 Čas na hodinách

Characteristics UUID	00002a08-0000-1000-8000-00805f9b34fb
Typ	DateTime
Čtení	Ano
Zápis	Ano
Oznámení	Ano

2.2.2 Budík

Service UUID - 000075[**X**]0-f90-44b0-81b9-3450ca663f46, kde X je číslo budíku v hex (1-a)

2.2.2.1 Čas na budíku

Characteristics UUID	000075b1-f908-44b0-81b9-3450ca663f46
Typ	DateTime
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.2.2 Stav budiku

Characteristics UUID	00007511-f908-44b0-81b9-3450ca663f46
Typ	bool
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.3 Stopky

Service UUID - 000075b0-f908-44b0-81b9-3450ca

2.2.3.1 Čas na stopkách

V sekundách.

Characteristics UUID	000075b1-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ne
Oznámení	Ano

2.2.3.2 Stav stopek

0 - vypnuté/restartované, 1 - běžící, 2 - pozastavené

Characteristics UUID	000075b2-f908-44b0-81b9-3450ca663f46
Typ	uint8
Čtení	Ano
Zápis	Ano
Oznámení	Ano

2.2.4 Časovač

Service UUID - 000075c0-f908-44b0-81b9-3450ca663f46

2.2.4.1 Aktuální čas na časovači

V sekundách.

Characteristics UUID	000075c1-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ne
Oznámení	Ano

2.2.4.2 Čas na který byl časovač nastaven

V sekundách.

Characteristics UUID	000075c2-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ne
Zápis	Ano
Oznámení	Ne

2.2.4.3 Stav

0 - vypnuté/restartovaný, 1 - běžící, 2 - pozastavený.

Characteristics UUID	000075c3-f908-44b0-81b9-3450ca663f46
Typ	uint8
Čtení	Ano
Zápis	Ano
Oznámení	Ano

2.2.5 Obecné

Service UUID - 000075d0-f908-44b0-81b9-3450ca663f46

2.2.5.1 Frekvence blikání dvojteček

V milisekundách.

Characteristics UUID	000075d1-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.5.2 Délka zobrazení času

V milisekundách.

Characteristics UUID	000075d2-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.5.3 Délka zobrazení data

V milisekundách.

Characteristics UUID	000075d3-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.5.4 Délka zvonění budíku

V milisekundách.

Characteristics UUID	000075d4-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.5.5 Délka zvonění časovače

V milisekundách.

Characteristics UUID	000075d5-f908-44b0-81b9-3450ca663f46
Typ	uint32
Čtení	Ano
Zápis	Ano
Oznámení	Ne

2.2.5.6 Formát data

0 pro americký, 1 pro evropský.

Characteristics UUID	000075d6-f908-44b0-81b9-3450ca663f46
Typ	uint8
Čtení	Ano
Zápis	Ano
Oznámení	Ne

Pomocí těchto služeb je možné komunikovat s hodinami nixie a provádět řadu funkcí, včetně nastavení času a data, nastavení budíků, ovládání stopek a časovače a změny různých možností na hodinách. Mobilní aplikace byla navržena tak, aby pracovala s těmito předdefinovanými službami a charakteristikami, aby poskytovala požadovanou úroveň ovládání a funkčnosti.

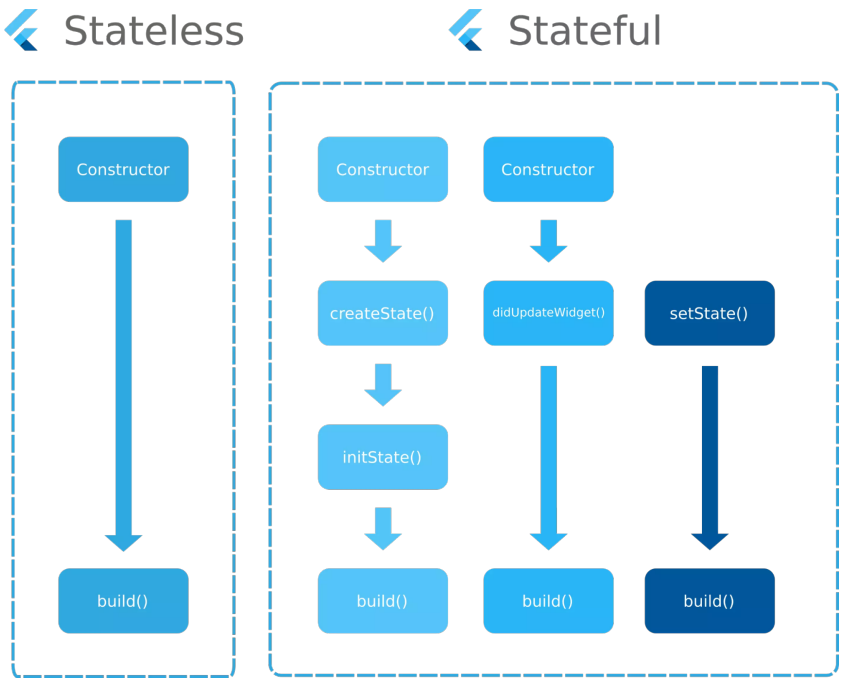
2.3 Architektura aplikace

Při návrhu mobilní aplikace byla celková architektura aplikace založena na principech frameworku Flutter.

Flutter je moderní vývojový framework, který je navržen tak, aby usnadnil vytváření vysoce kvalitních, citlivých a výkonných mobilních aplikací pro řadu platforem, včetně Androidu a iOS. Jádrem Flutteru je koncept widgetů, což jsou malé, samostatné jednotky kódu, které lze kombinovat a vytvářet uživatelské rozhraní a základní chování aplikace.

Ve Flutteru jsou dva typy widgetů: stavové a bezstavové. Stavové widgety si udržují svůj vlastní vnitřní stav a mohou měnit svůj vzhled a chování na základě vstupu uživatele nebo jiných událostí. Na druhou stranu bezstavové widgety si neudržují svůj vlastní stav a jednoduše se používají k zobrazení informací a zpracování uživatelského vstupu.

Obrázek 2.2 Typy widgetů ve Flutteru[14]



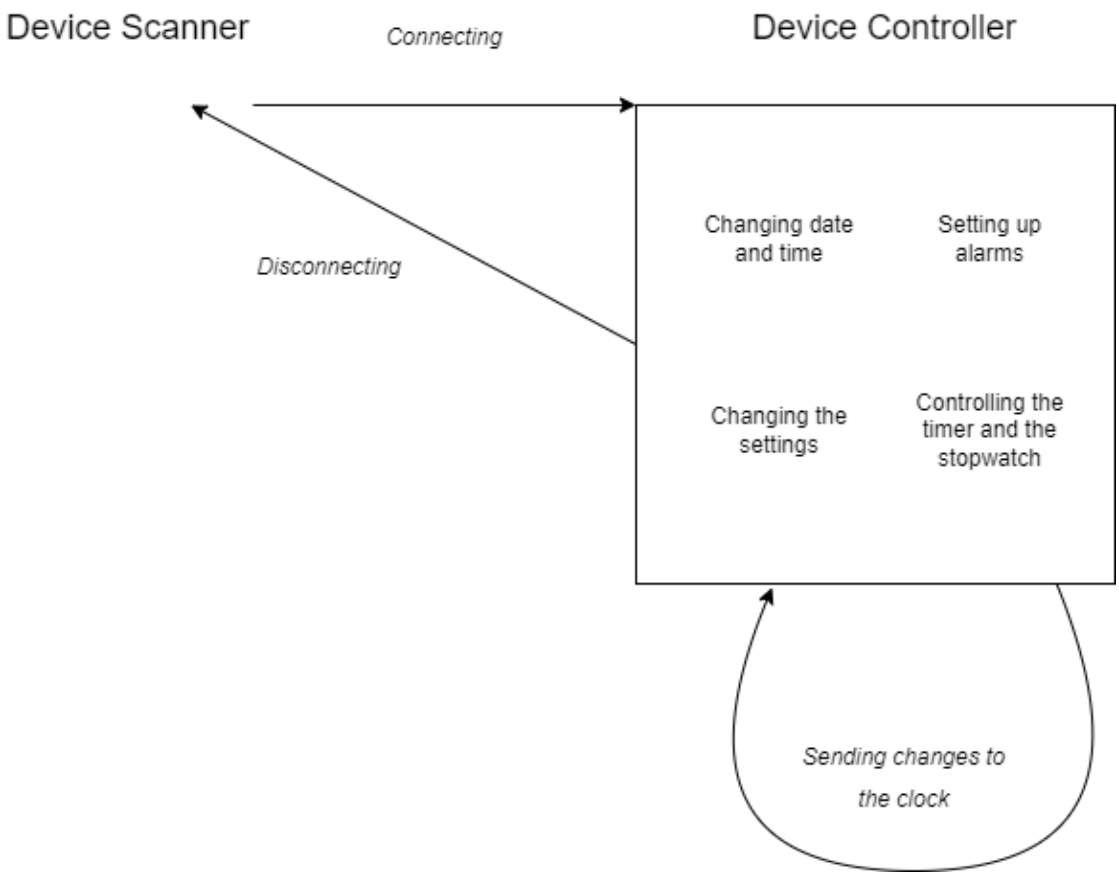
V případě mobilní aplikace pro ovládání digitálních nixie hodin byl vytvořen stavový widget pro každou z hlavních funkcí aplikace (např. nastavení času a data, nastavení budíků, ovládání stopek a časovače atd.). Tento přístup umožnil snadnou synchronizaci stavu aplikace se stavem hodin nixie prostřednictvím BLE, což zajistilo, že aplikace vždy přesně odrážela aktuální stav hodin.

Celková architektura aplikace byla navržena tak, aby byla flexibilní a modulární, přičemž každá z hlavních funkcí je implementována jako samostatný stavový widget. To umožnilo snadné přizpůsobení a rozšíření aplikace a také lepší opětovné použití kódu a údržbu.

Kromě stavových widgetů aplikace také obsahovala řadu bezstavových widgetů, které byly použity k zobrazení informací a zpracování uživatelského vstupu. Tyto widgety byly zkombinovány se stavovými widgety k vytvoření celkového uživatelského rozhraní a chování aplikace.

2.4 Logický diagram

Obrázek 2.3 Logický tok aplikace



Existuje několik klíčových případů použití, které byly zvažovány při návrhu mobilní aplikace pro ovládání digitálních nixie hodin. Tyto případy použití jsou založeny na očekávaných funkcích aplikace, které zahrnují nastavení času a data, nastavení budíků, ovládání stopek a časovače a změnu různých možností na hodinách.

Aplikace se spustí se skenerem BLE, který se zobrazí zařízením v okolí uživatele. Jakmile se

uživatelé rozhodnou připojit ke konkrétnímu zařízení, aplikace přejde na samostatnou obrazovku, která odhalí hlavní ovládání hodin.

Jedním z klíčových případů použití je nastavení času a data na nixie hodinách. Aby aplikace zvládla tento případ použití, musí obsahovat stavový widget, který je zodpovědný za zobrazení uživatelského rozhraní pro nastavení času a data, jakož i za zpracování uživatelského vstupu a komunikaci s hodinami nixie přes BLE za účelem nastavení času a data. Když uživatel provede změny času a data, widget bude muset aktualizovat svůj vnitřní stav, aby odrážel tyto změny, a poté odeslat příslušné příkazy do nixie clock prostřednictvím BLE, aby aktualizoval čas a datum.

Dalším klíčovým případem použití je nastavení budíků na nixie hodinách. Aby aplikace zvládla tento případ použití, musí obsahovat stavový widget, který je zodpovědný za zobrazení uživatelského rozhraní pro nastavování budíků a také za manipulaci s uživatelským vstupem a komunikaci s hodinami nixie přes BLE za účelem nastavení budíků. Když uživatel provede změny v alarmech, widget aktualizuje svůj vnitřní stav, aby odrážel tyto změny, a poté odešle příslušné příkazy do hodin nixie prostřednictvím BLE, aby se alarmy aktualizovaly.

Podobně musí aplikace také obsahovat stavové widgety pro ovládání stopek a časovače, které si poradí s případy použití spouštění, zastavování a resetování stopek a časovače a také pro zobrazování aktuálního času na těchto funkcích. Tyto widgety budou zodpovědné za aktualizaci svého vnitřního stavu a komunikaci s hodinami nixie přes BLE, aby odrážely aktuální stav stopek a časovače.

Nakonec musí aplikace obsahovat stavový widget pro změnu různých možností na nixie hodinách, jako je frekvence blikání dvojtečky, délka zobrazení času a délka zobrazení dat. Tento widget bude zodpovědný za aktualizaci svého vnitřního stavu a komunikaci s hodinami nixie prostřednictvím BLE, aby odrážel aktuální možnosti na hodinách.

Celkově je architektura mobilní aplikace navržena tak, aby tyto klíčové případy použití zvládla flexibilně a efektivně a poskytla řadu výkonných a pohodlných nástrojů pro ovládání digitálních hodin nixie prostřednictvím BLE.

Kapitola 3
Implementace a testování

Implementace a testování mobilní aplikace pro ovládání digitálních nixie hodin byly zásadní fáze procesu vývoje, protože nám umožnily uvést aplikaci do života a zajistit, aby fungovala správně a spolehlivě. V této části bakalářské práce popíšu klíčové kroky, které byly podniknuty k implementaci a testování aplikace.

Implementace aplikace zahrnovala nastavení vývojového prostředí, vytvoření uživatelského rozhraní a pomocných tříd, které byly potřeba k tomu, aby byla aplikace funkční. To vyžadovalo důkladné pochopení frameworku Flutter a také nástrojů a knihoven, které byly používány. Pečlivým dodržováním osvědčených postupů a používáním systematického přístupu jsem byl schopen aplikaci implementovat efektivně a efektivně a poskytnout potenciálním uživatelům vysoce kvalitní produkt.

Testování bylo důležitou součástí implementačního procesu, protože mi umožnilo identifikovat a opravit vady nebo problémy, které se mohly objevit během vývoje. Použili jsme řadu testovacích metod, včetně testování jednotek a ručního testování, abychom zajistil, že aplikace funguje podle očekávání a splňuje funkční požadavky, které byly stanoveny na začátku projektu.‘

3.1 Vývoj

Vývojářské prostředí pro mobilní aplikaci pro ovládání digitálních hodin nixie byl založen na frameworku Flutter, který je navržen tak, aby usnadnil vytváření, sestavování a testování mobilních aplikací pro řadu platforem.

Obecně se projekt Flutter skládá ze sady souborů zdrojového kódu, které definují uživatelské rozhraní a chování aplikace, a také sady konfiguračních souborů, které řídí sestavení a spouštění aplikace. Soubory zdrojového kódu jsou napsány v programovacím jazyce Dart, což je moderní, objektově orientovaný jazyk, který se dobře hodí pro vytváření mobilních aplikací.

K sestavení a spuštění mobilní aplikace v této práci byly použity nástroje příkazového řádku Flutter ke kompilaci zdrojového kódu a vytvoření spustitelného balíčku pro aplikaci. Aplikace by pak mohla být spuštěna na fyzickém zařízení nebo v emulátoru a testovat a ladit chování aplikace.

Při vývoji mobilní aplikace byl k úpravám a správě souborů zdrojového kódu použit textový editor Visual Studio Code. Visual Studio Code je populární a výkonný editor kódu, který poskytuje řadu funkcí pro práci s kódem, včetně zvýraznění syntaxe, dokončování kódu a kontroly chyb.

Celkově bylo vývojářské prostředí pro mobilní aplikaci v této práci navrženo tak, aby bylo flexibilní a efektivní a poskytovalo řadu nástrojů a zdrojů pro vytváření, testování a ladění aplikace. To umožnilo hladký a efektivní proces vývoje, jehož výsledkem byla vysoce kvalitní a

spolehlivá aplikace.

3.2 Uživatelské rozhraní

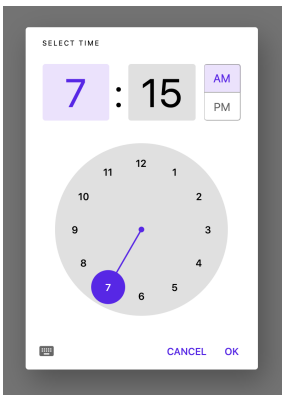
Uživatelské rozhraní (UI) mobilní aplikace pro ovládání digitálních hodin nixie je vizuální a interaktivní aspekt aplikace, který je prezentován uživateli. V kontextu této práce bylo uživatelské rozhraní navrženo a implementováno pomocí frameworku Flutter, který poskytuje řadu výkonných a flexibilních nástrojů pro vytváření moderních, responzivních a výkonných uživatelských rozhraní mobilních aplikací.

Ve Flutteru je uživatelské rozhraní aplikace vytvořeno pomocí hierarchie widgetů, přičemž každý widget představuje malou samostatnou jednotku kódu, kterou lze kombinovat a vytvořit tak celkový vzhled a chování aplikace. K implementaci uživatelského rozhraní pro mobilní aplikaci v této práci byla použita řada různých widgetů, včetně stavových i bezstavových.

3.2.1 Time Picker

Time picker umožňuje uživatelům zadat konkrétní časovou hodnotu. Jsou zobrazeny v dialozích a lze je použít k výběru hodin, minut a časového období. V mé aplikaci ji používám pro nastavení času nixie hodin.

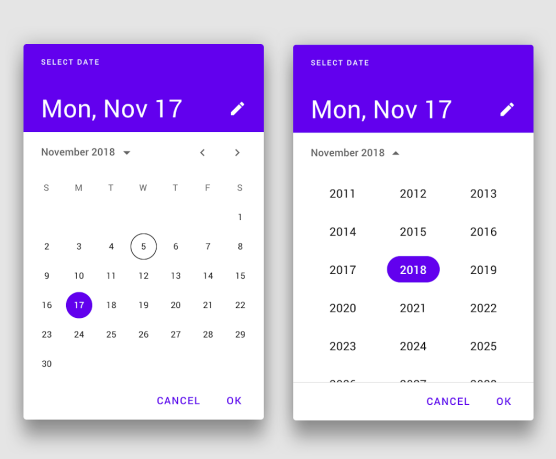
Obrázek 3.1 Příklad nástroje pro výběr času



3.2.2 Date Picker

Date picker umožňuje uživatelům vybrat datum nebo rozsah dat. Ve své aplikaci ji používám pro nastavení data hodin nixie.

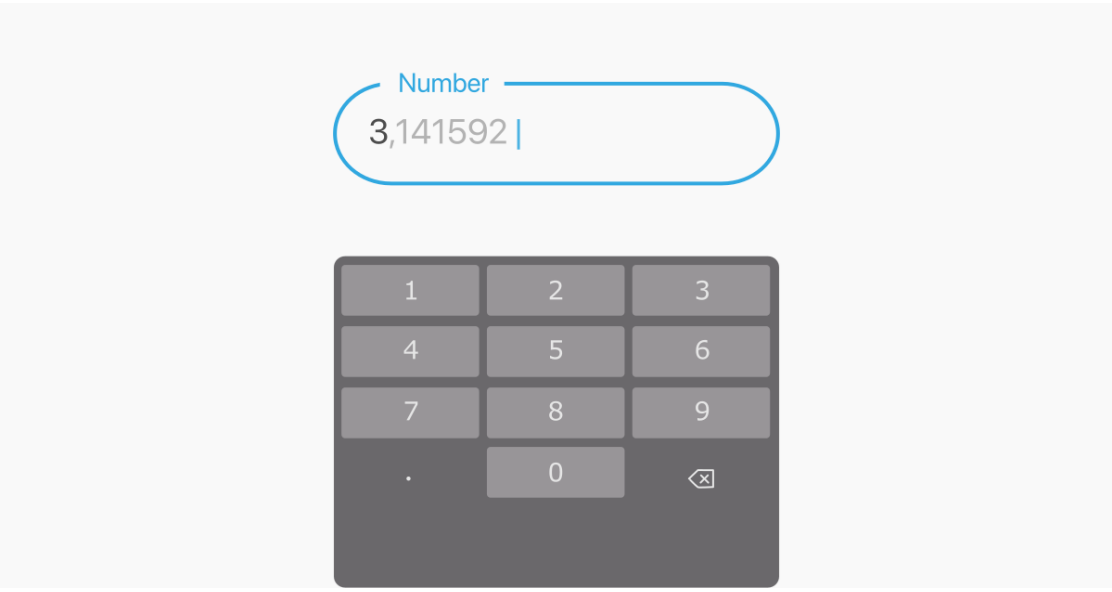
Obrázek 3.2 Příklad nástroje pro výběr data



3.2.3 Numeric Input Field

Flutter technicky nemá numerické vstupní pole. V tomto případě se předpokládá použití TextInput se vstupem omezeným na numerickou klávesnici. Ve své aplikaci jej používám pro nastavení číselných hodnot, jako je interval blikání teček, interval vyzvánění budíku nebo hodnota odpočítávání časovače.

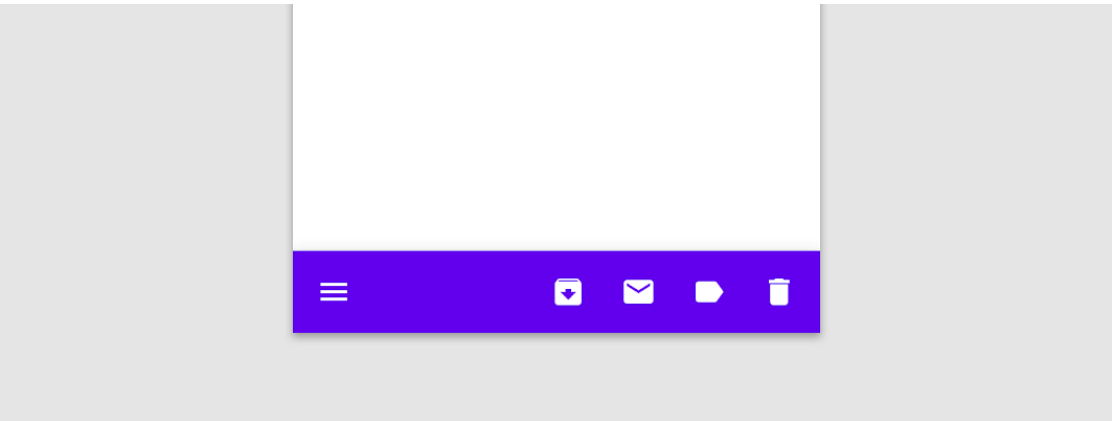
Obrázek 3.3 Příklad číselného vstupního pole



3.2.4 Bottom Navigation Bar

Bottom Navigation Bar zobrazuje navigaci a klíčové akce ve spodní části mobilních obrazovek. Bottom Navigation Bar poskytuje přístup ke spodnímu navigačnímu šuplíku a až čtyřem akcím, včetně plovoucího akčního tlačítka. Ve své aplikaci jej používám pro přepínání mezi menu pro konkrétní funkce hodin.

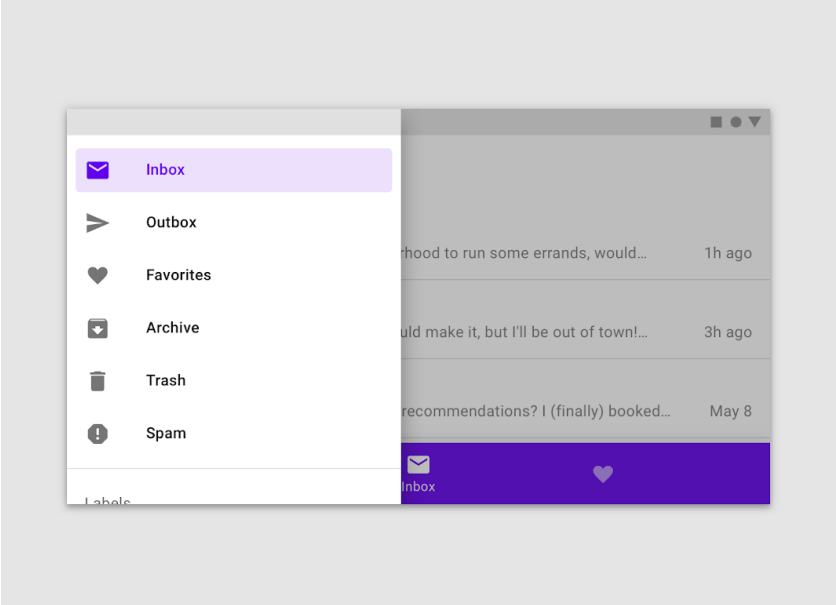
Obrázek 3.4 Příklad spodního navigačního panelu



3.2.5 Drawer

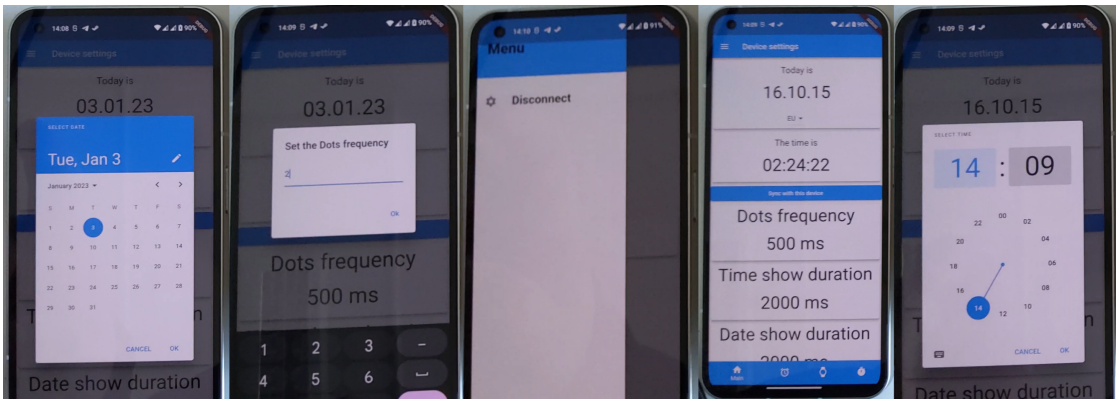
Drawer poskytuje přístup k cílům a funkcím aplikací, jako je přepínání účtů. Mohou být buď trvale na obrazovce, nebo je lze ovládat pomocí ikony navigační nabídky. Ve své aplikaci ji používám k odpojení od připojených hodin.

Obrázek 3.5 Příklad zásuvky



Pomocí těchto a dalších widgetů jsem byl schopen vytvořit uživatelské rozhraní, které bylo intuitivní, responzivní a vizuálně přitažlivé a které uživatelům aplikace poskytovalo řadu funkcí a funkcí.

Obrázek 3.6 Kombinace widgetů ve finálním uživatelském rozhraní



3.3 Další nástroje

Kromě hlavních widgetů, které tvoří uživatelské rozhraní a funkčnost aplikace, byla implementována řada dalších pomocných tříd jako prosté třídy Dart, které tyto widgety podporují a poskytují další nástroje.

Jednou z takových pomocných tříd byl převodník dat, který byl zodpovědný za převod dat mezi různými typy a formáty. Konkrétně byla tato třída použita k převodu dat z datových typů Dart (jako jsou celá čísla, třída DateTime a booleans) do sekvencí bajtů, které bylo možné přenášet přes BLE, a k rozbalení těchto bajtů zpět do datových typů Dart po přijetí. To byl důležitý úkol, protože protokol BLE vyžaduje přenos dat ve specifickém formátu a převodník dat poskytl pohodlný a spolehlivý způsob, jak tento proces převodu zvládnout.

Další pomocnou třídou, která byla implementována, byl generátor charakteristik, který byl použit k sestavení typu QualifiedCharacteristics z charakteristik služby UUID a charakteristik atributů UUID. Tato třída poskytla pohodlný a efektivní způsob, jak vytvořit a spravovat vlastnosti, které aplikace používala ke komunikaci s hodinami nixie prostřednictvím BLE.

Nakonec byla implementována třída představující připojení nixie clock, aby abstrahovala operace čtení a zápisu za jednoduchými funkcemi, které bylo možné volat z widgetů aplikace. Tato třída poskytovala rozhraní na vysoké úrovni pro interakci s hodinami nixie, což usnadňovalo widgetům komunikaci s hodinami a provádění jejich funkcí.

Tyto pomocné třídy poskytovaly řadu užitečných a pohodlných nástrojů a zdrojů pro widgety aplikace a podporovaly funkčnost a výkon aplikace.

Výpis kódu 3.1 Příklad kódu, který převádí třídu DateTime na sekvenci bajtů, které lze přenášet přes BLE[9]

```
List<int> dateTimePacked = List.filled(7, 0);
dateTimePacked[1] = (value.year / 256).floor().toUnsigned(8);
dateTimePacked[0] = (value.year - dateTimePacked[1] * 256).toUnsigned(8);
dateTimePacked[2] = value.month.toUnsigned(8);
dateTimePacked[3] = value.day.toUnsigned(8);
dateTimePacked[4] = value.hour.toUnsigned(8);
dateTimePacked[5] = value.minute.toUnsigned(8);
dateTimePacked[6] = value.second.toUnsigned(8);
return dateTimePacked;
```

3.4 Testování

V kontextu vývoje softwaru se testování týká procesu hodnocení softwarového systému nebo komponenty, aby se zjistilo, zda splňuje stanovené požadavky a funguje tak, jak bylo zamýšleno. Testování je kritickou součástí procesu vývoje softwaru, protože pomáhá zajistit, aby byl software vysoce kvalitní, spolehlivý a stabilní a fungoval podle očekávání.

V případě mobilní aplikace pro ovládání digitálních nixie hodin bylo testování důležitým krokem v procesu vývoje, protože pomohlo zajistit, aby aplikace byla schopna komunikovat s nixie hodinami a provádět své různé funkce správně. To zahrnovalo testování uživatelského rozhraní aplikace, jejího propojení s hodinami a její schopnosti nastavovat čas a datum, nastavovat budíky, ovládat stopky a časovač a měnit možnosti hodin.

V této kapitole bakalářské práce popíšu proces testování, který byl pro mobilní aplikaci proveden, včetně vytvořených testovacích případů, použitých nástrojů a metod a dosažených výsledků.

3.4.1 Unit testy

Unit testy je typ testování, které se zaměřuje na jednotlivé jednotky nebo komponenty softwarového systému s cílem ověřit, zda každá jednotka nebo komponenta funguje správně a jak bylo zamýšleno. Testování jednotek je obvykle prováděno vývojáři během implementační fáze softwarového projektu a je navrženo tak, aby zachytilo chyby a defekty v rané fázi vývojového procesu, než mají šanci se rozšířit a způsobit vážnější problémy.

V případě mobilní aplikace pro ovládání digitálních nixie hodin se většina aplikace skládá z výchozích widgetů, které jsou vzájemně propojeny relativně jednoduchým a přímočarým způsobem. Výsledkem bylo, že nebylo mnoho kódu, který by bylo třeba testovat na jednotku. Jednou z klíčových součástí aplikace, která však vyžadovala unit testy, byla pomocná třída převodníku dat, která byla zodpovědná za převod dat mezi různými typy a formáty.

K otestování třídy převodníku dat bylo vytvořeno několik jednotkových testů, které prováděly metody dané třídy a kontrolovaly výsledky, aby se zajistilo, že data byla převedena správně. Tyto testy zahrnovaly případy, kdy byly různé hodnoty převáděny tam a zpět a výsledky byly kontrolovány, aby se ujistil, že hodnoty zůstaly nezměněny. Ověřením správného chování třídy převodníku dat jsem si mohl být jistý, že funguje správně a při použití aplikací nezpůsobí žádné problémy.

Celkově sehrálo testování jednotek důležitou roli při vývoji mobilní aplikace a pomohlo zajistit, že aplikace bude spolehlivá, stabilní a vysoce kvalitní. Důkladným testováním různých součástí aplikace jsme byli schopni zachytit a opravit jakékoli závady nebo problémy, které mohly nastat, a dodat uživatelům stabilní a spolehlivou aplikaci.

Výpis kódu 3.2 Příklad unit testu, který testuje konverzi data

```
test('DateTime conversion', () {
    DateTime timestamp = DateTime.now();
    DateTime original = DateTime(timestamp.year,
                                 timestamp.month,
                                 timestamp.day,
                                 timestamp.hour,
                                 timestamp.minute,
                                 timestamp.second);
    List<int> converted = Converter.toByteArray<DateTime>(original);
    DateTime restored = Converter.fromByteArray<DateTime>(converted);
    expect(restored, original);
});
```

3.4.2 Manuální testování

Kromě testování jednotek bylo důležitou součástí procesu vývoje mobilní aplikace také manuální testování. Vzhledem k tomu, že primárním účelem aplikace je komunikovat s hodinami nixie a ovládat jejich různé funkce, ruční testování bylo efektivním způsobem, jak zajistit, aby aplikace fungovala správně a splňovala zadané požadavky.

Pro ruční testování byly jako testovací lůžko použity hardwarové nixie hodiny a aplikace byla nainstalována na mobilní zařízení. Aplikace byla poté cvičena k provádění řady funkcí, včetně připojení k hodinám, nastavení času a data, nastavení budíků, ovládání stopek a časovače a změny možností hodin. V každém případě bylo pozorováno chování aplikace a výsledky byly zkontrolovány, aby bylo zajištěno, že funkce fungují podle očekávání a že jsou splněny všechny zadané případy použití.

Celkově hrálo při vývoji mobilní aplikace zásadní roli manuální testování, které pomáhalo zajistit, aby byla aplikace spolehlivá, stabilní a vysoce kvalitní. Důkladným testováním funkčnosti a výkonu aplikace jsem byl schopen zachytit a opravit jakékoli závady nebo problémy, které mohly nastat. Poskytnutím důkladného a komplexního popisu procesu ručního testování se snažím prokázat robustnost a spolehlivost mobilní aplikace.

Seznam zařízení používaných pro testování:

Tabulka 3.1 Testovací zařízení

Zařízení	Nothing Phone 1	Samsung Galaxy Tab S6
Typ	Smartphone	Tablet
Operační systém	Android 12	Android 11

Obrázek 3.7 Manuální testování s jedním z prototypů

Kapitola 4

Zaver

Závěrem lze říci, že mobilní aplikace pro ovládání digitálních nixie hodin přes Bluetooth Low Energy byla úspěšně vyvinuta a otestována a splnila funkční cíle, které byly stanoveny na začátku projektu.

Aplikace byla implementována pomocí frameworku Flutter a programovacího jazyka Dart, což poskytuje řešení pro více platforem, které lze potenciálně spustit na zařízeních Android i iOS. Použití Flutter umožnilo rychlý vývoj aplikace díky bohaté sadě widgetů a nástrojů a také její schopnosti nativně kompilovat pro více platforem.

Pro usnadnění komunikace s hodinami nixie byla použita knihovna třetí strany s názvem Flutter Reactive BLE, která poskytovala sadu API a nástrojů pro interakci se zařízeními BLE. Architektura aplikace byla navržena tak, aby byla modulární a škálovatelná, se samostatnými widgety a třídami, které zvládají různé funkce a odpovědnosti. Díky tomu bylo snadné spravovat stav aplikace a synchronizovat ji s hodinami nixie přes BLE.

Během procesu vývoje byla použita řada testovacích metod k zajištění kvality a spolehlivosti aplikace. Testování jednotek bylo použito k ověření správnosti jednotlivých komponent, zatímco ruční testování bylo použito k procvičení funkcí aplikace a kontrole správného fungování. Důkladným testováním aplikace se mi podařilo zachytit a opravit jakékoli závady nebo problémy, které mohly nastat, a poskytnout potenciálním uživatelům stabilní a spolehlivou aplikaci.

Celkově byl vývoj mobilní aplikace pro ovládání digitálních hodin nixie přes BLE úspěšný a přínosný projekt a jsem přesvědčen, že aplikace poskytne hodnotný a pohodlný nástroj pro uživatele, kteří chtějí ovládat své hodiny nixie ze svého mobilu.

Bibliografie

1. ICASIANO, Aurelio. Nixie clocks: The future that never was. *Mantle Magazine* [online]. 2019 [cit. 2022-12-01]. Dostupné z: https://www.mantlemagazine.com/index.php/2019/05/26/nixie-clocks-the-future-that-never-was/.
2. LLC, GOOGLE. *Application Fundamentals for Android developers* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://developer.android.com/guide/components/fundamentals.
3. APPLE, Inc. *About Swift* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://www.swift.org/about/.
4. LLC, GOOGLE. *Dart overview* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://dart.dev/overview.
5. META PLATFORMS, Inc. *Introduction to React* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://reactnative.dev/docs/getting-started.
6. BLUETOOTH SIG, INC. *Bluetooth Technology Overview* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.
7. BLUETOOTH SIG, INC. *A Developer's Guide To Bluetooth* [online]. 2016. [cit. 2022-12-03]. Dostupné z: https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%5C%20UUID%5C%20Numbers%5C%20Document.pdf.
8. TOWNSEND, Kevin. *Introduction to Bluetooth Low Energy* [online]. 2014. [cit. 2022-12-03]. Dostupné z: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt?view=all.
9. BLUETOOTH SIG, INC. *16-bit UUID Numbers Document* [online]. 2021. [cit. 2022-12-03]. Dostupné z: https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%5C%20UUID%5C%20Numbers%5C%20Document.pdf.
10. ALEXIOS. *Mastering the fundamentals of BLE for iOS using Swift.* [online]. 2021. [cit. 2022-12-03]. Dostupné z: https://alimovlex.medium.com/mastering-the-fundamentals-of-ble-in-ios-with-swift-73a43ffe44dc.
11. BARTÍK, Matěj; VERNER, David. *Digitronové hodiny (Nixie clock)- specifikace pro účely BP* [online]. 2021. [cit. 2022-12-03]. Dostupné z: https://tinyurl.com/r9yxwwej.
12. PEREIRA, Rui; COUTO, Marco; RIBEIRO, Francisco; RUA, Rui; CUNHA, Jácome; FERNANDES, João Paulo; SARAIVA, João. Ranking programming languages by energy efficiency. *Science of Computer Programming*. 2021, roč. 205, s. 102609. ISSN 0167-6423. Dostupné z DOI: https://doi.org/10.1016/j.scico.2021.102609.
13. HOLDING, Signify. *Flutter reactive BLE library* [online]. 2022. [cit. 2022-12-03]. Dostupné z: https://github.com/PhilipsHue/flutter_reactive_ble.

14. GERKEN, Marc. *StatelessWidget vs. StatefulWidget* [online]. 2020. [cit. 2022-12-03]. Dostupné z: https://www.flutterclutter.dev/flutter/basics/statelesswidget-vs-statefulwidget/2020/1195/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `apk` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `media`
  - `demo.mp4` ..... video, které ukazuje použití aplikace