STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Zavedení IoT a AR do školství

Lukáš Šafránek
Královéhradecký kraj

Hradec Králové 2023

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Zavedení IoT a AR do školství

Introduction of IoT and AR to education system

Autoři: Lukáš Šafránek
Škola: Střední škola a vyšší odborná škola aplikované kybernetiky s.r.o., Hradecká 1151/9, Hradec Králové 500 03
Kraj: Královéhradecký kraj
Konzultant: Ing. Matěj Lang

Hradec Králové 2023

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Hradci králové dne 27. 2. 2023 ……Lukáš Šafránek……

Poděkování

Na tomto místě bych rád poděkoval Ing. Matěji Langovi za odbornou pomoc v oblasti programování a za zapůjčení brýlí pro rozšířenou realitu Hololens 2.

Též nemohu opomenout Ing. Jakuba Návesníka PhD, který mě na toto téma přivedl a též s ochotou poskytl Vernier senzory. Jeho zpětná vazba na celý projekt je uvedená v sekci Diskuse o možnostech rozšíření a vylepšení systému

Anotace

Tato práce se zaměřuje na využití technologie IoT (internet věcí) při výuce chemie, fyziky, elektroniky a biologie. Pomocí bezdrátových senzorů značky Vernier, které využívají technologii BLE (Bluetooth Low Energy), jsou shromažďována data z měření. Tyto naměřené hodnoty jsou následně seskupeny a odeslány na centrální server. Za tímto účelem jsem vyvinul vlastní protokol postavený na TCP pro rovnováhu mezi minimální latencí a maximální spolehlivostí. Zde jsou data zpracována a uložena pro další použití. Studenti mohou pomocí veřejně dostupného připojení SignalR přistoupit k serveru a získat data pro svá vlastní měření. Tento způsob umožňuje zobrazování dat na různých zařízeních, včetně mobilních zařízení a speciálních brýlí pro rozšířenou realitu Hololens 2 od společnosti Microsoft. Výsledkem se pak stává snadná a efektivní manipulace s měřenou soustavou.

Celkově tedy tato práce přináší řadu výhod. Patří mezi ně efektivní shromažďování dat, otevřené připojení pro studenty a snadné zobrazení dat na různých zařízeních. Díky tomu mají studenti snadný přístup k datům a mohou je snadno analyzovat a lépe porozumět měřené soustavě.

Klíčová slova

IoT; vizualizace dat; Hololens 2; analýza dat; výuka

Annotation

This thesis focuses on the use of IoT (Internet of Things) technology in teaching chemistry, physics, electricity and biology. Measurement data is collected using Vernier wireless sensors using BLE (Bluetooth Low Energy) technology. These measurements are then aggregated and sent to a central server. For this purpose, I developed a custom protocol based on TCP to balance between minimum latency and maximum reliability. Here the data is processed and stored for further use. Students can use the publicly available SignalR connection to access the server and retrieve data for their own measurements. This allows the data to be displayed on a variety of devices, including mobile devices and Microsoft's special Hololens 2 augmented reality glasses. The result is easy and efficient manipulation of the measured system.

All in all, there are a number of benefits to this work. These include efficient data collection, open connectivity for students and easy display of data on different devices. As a result, students can easily access and analyze the data to better understand the measured system.

Keywords

IoT; data visualization; Hololens 2; data analysis; education

Obsah

1 Úvod

1.1 Motivace pro využití IoT v praktických hodinách výuky

Hlavní motivací pro vývoj systému bylo zjednodušení práce pro studenty a vylepšení intuitivního chápání měření. Tento cíl byl vytyčen na základě požadavku Střední průmyslové školy chemické v Pardubicích, která chtěla vylepšit výuku v těchto oborech pomocí moderních technologií. Hlavním požadavkem bylo využití rozšířené reality poskytovanou Hololens 2. Tímto zařízením škola disponuje, avšak postrádá využití pro nedostatek volně dostupného softwaru.

1.2 Cíle práce

Cílem této práce je navrhnout a implementovat systém využívající IoT technologie pro praktické výukové hodiny chemie, fyziky, elektrotechniky a biologie, který umožní snadný sběr a analýzu dat pomocí bezdrátových senzorů Vernier. Systém bude využívat mikroprocesor ESP32 pro sběr, seskupení a následné odeslání dat z jednotlivých senzorů pomocí TCP protokolu na centrální server. Na serveru budou data zpracována a uložena pro následné využití v aplikacích pro mobilní zařízení. Cílem je poskytnout studentům efektivní nástroj pro manipulaci s měřenou soustavou a pomoci jim lépe porozumět fyzikálním a chemickým principům. Součástí práce je také vyhodnocení přínosů pro výuku a navrhnutí dalších možností rozvoje a vylepšení.

1.3 Struktura práce

Práce se skládá z několika částí, které jsou nezbytné pro správnou funkci celého systému.

První část se zaměřuje na Vernier GO senzory a jejich protokol. Následně implementaci komunikace s nimi v jazyce C++.

Dále se práce věnuje síťové komunikaci a jednotlivým protokolům, jako jsou vyhledávací protokol, obecný protokol, protokol pro kontrolu funkčnosti, příkazový protokol, protokol událostí a datový protokol.

Další část je věnována ESP32, které slouží pro sběr dat z Vernier GO senzorů. Popisuje se zde připojení k síti Wi-Fi, implementace vlastních protokolů, včetně Vernier protokolu, zajištění integrity dat, seskupování dat a odesílání dat.

V další části se věnuji serveru, který slouží k abstrakci připojení, abstrakci dat, uložení dat a zpřístupnění dat pro klienty.

Poslední část popisuje realizaci klienta, který komunikuje se serverem, implementuje vyhledávací protokol, získává data, zpracovává je a zobrazuje vizualizaci zpracovaných dat. Klient také poskytuje funkce pro přidružené výpočty.

2 TECHNICKÁ ČÁST

2.1 Použité technologie

2.1.1 BLE

BLE (Bluetooth Low Energy) je bezdrátová technologie s nízkou spotřebou energie, která umožňuje přenos dat mezi zařízeními s kompatibilními vlastnostmi. Je vhodná pro senzory, hodinky, fitness náramky a další přenosná zařízení, která mohou být propojena s chytrými telefony, tablety a dalšími zařízeními. BLE využívá frekvenci 2,4 GHz, ale používá různé modulace a modulační indexy pro dosažení co nejnižší spotřeby energie. Zařízení s BLE mohou být buď v režimu "central" nebo "peripheral". BLE se používá v IoT a aplikacích jako jsou chytré domácnosti, inteligentní města a zdravotní zařízení. [1]

2.1.2 TCP

TCP (Transmission Control Protocol) je spolehlivý a bezpečný protokol pro přenos dat v sítích, který se používá pro citlivá data, jako jsou e-maily, webové stránky, přenos souborů a vzdálené připojení k počítačům. TCP zajistí doručení dat bez ztráty a v pořadí, využívá tokovou kontrolu a optimalizuje odesílání dat tak, aby nedošlo k zahlcení sítě. TCP se používá v situacích, kdy je důležité, aby data byla doručena bez chyb a v pořadí, v jakém byla odeslána. [2]

2.1.3 UDP

UDP je bezspojový a rychlý protokol pro přenos dat v síťových aplikacích, jako jsou hry, hlasová a video komunikace a IoT aplikace. UDP nepoužívá mechanismy pro udržení spojení mezi odesílatelem a příjemcem, což zajišťuje rychlý přenos dat, ale také může vést ke ztrátě dat. UDP je vhodný pro přenos méně citlivých dat a umožňuje zasílání dat více adresátům najednou a v rámci jednoho paketu. UDP se často používá v kombinaci s protokolem TCP pro zlepšení rychlosti a efektivity komunikace v některých aplikacích. [3]

2.1.4 C++

C++ je často používaným jazykem pro vývoj embedded aplikací díky své efektivitě, podpoře OOP a možnosti efektivní práce s omezenými prostředky embedded zařízení. C++ umožňuje programátorům efektivně organizovat kód do tříd, dědit a používat polymorfismus. C++ také podporuje knihovny a nástroje pro embedded systémy, což usnadňuje integraci s existujícím kódem a umožňuje programátorům využívat rozsáhlé knihovny funkcí pro řízení hardware a komunikaci se senzory a dalšími zařízeními. Pro úspěšný vývoj embedded systémů je důležité pečlivě navrhovat kód a minimalizovat používání paměti a procesoru. [4]

2.1.5 C#

C# je moderní, objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Kombinuje prvky z jiných jazyků a umožňuje psát bezpečné a spolehlivé kódy s minimálním

počtem chyb. .NET Core 7.0 je open-source, multi-platformový framework pro vývoj webových, desktopových a mobilních aplikací. Poskytuje vývojářům výkonné a škálovatelné funkce, jako jsou asynchronní I/O operace a integrace s kontejnery. Kombinace C# a .NET Core 7.0 je ideální pro vývoj spolehlivých a výkonných aplikací na různých platformách. [5]

2.1.6 SignalR

SignalR je open-source knihovna pro vývoj webových aplikací, která umožňuje v reálném čase komunikovat mezi klientem a serverem pomocí technologií WebSockets, Server-Sent Events a dalších.

SignalR je multiplatformní a poskytuje programátorům API pro jednoduchou a efektivní komunikaci mezi webovými aplikacemi. SignalR řeší problém, kdy by v případě klasického HTTP protokolu by bylo nutné neustále obnovovat stránku, aby se zobrazily nové informace. SignalR umožňuje tuto komunikaci přenést na server a také ji propojit s klasickým HTTP protokolem. [6]

2.1.7 UWP a XAML

UWP (Universal Windows Platform) je platforma pro vývoj aplikací pro různá zařízení s Windows 10 nebo 11 pomocí jednoho kódu. XAML (Extensible Application Markup Language) je jazyk pro popis uživatelského rozhraní pro platformu .NET, který umožňuje snadné oddělení logiky od návrhu uživatelského rozhraní. Spojení UWP a XAML umožňuje vývojářům snadno vytvářet uživatelské rozhraní pro aplikace s různými zařízeními a propojovat je s kódem aplikace pomocí datových vazeb a událostí. [7]

2.1.8 Hololens 2

Hololens 2 je zařízení od společnosti Microsoft, které umožňuje rozšířenou realitu (AR) v kombinaci s virtuální realitou (VR) a inteligentními cloudy[1]. Hololens 2 poskytuje vývojářům řadu funkcí, jako jsou rozpoznávání hlasu, gest a okolního prostředí, senzory pro sledování pohybu hlavy a rukou a možnost vytvářet vlastní holografické aplikace a scény.

Vývojáři mohou vytvářet aplikace pro Hololens 2 pomocí různých nástrojů, včetně Unity[2] a Visual Studio[3]. Tyto nástroje umožňují vývojářům vytvářet holografické scény a aplikace pomocí objektů a prostorových transformací. Vývojáři mohou také vytvářet vlastní interakce pro gesta a hlasové příkazy.

Hololens 2 podporuje vývoj aplikací v jazycích C++ a C# a vývojáři mohou také používat různé knihovny a frameworky, jako jsou DirectX, OpenGL a OpenCV. Vývojáři mohou také využít

[1] Cloud: služba běžící na vzdáleném serveru, zpravidla v nějakém datacentru

[2] Unity: vývojové prostředí pro tvorbu 3D her pomocí jazyku C#

[3] Visual Studio: vývojové prostředí podporující řadu jazyků, například C# nebo C++

řadu služeb a technologií poskytovaných cloudem, jako jsou Azure Cognitive Services, Azure Spatial Anchors a Azure Digital Twins. [8]

2.2 Vernier GO senzory

Vernier GO senzory nabízejí několik výhod pro praktickou výuku fyziky a chemie. Mezi nejdůležitější patří:

1. Snadné použití: Senzory jsou velmi snadno použitelné a přístupné pro studenty různých věkových kategorií.

2. Vysoká přesnost: nabízejí vysokou přesnost měření, což je klíčové pro správnou interpretaci výsledků experimentů.

3. Rozmanitost: jsou k dispozici v široké škále konfigurací, což umožňuje měření různých fyzikálních a chemických veličin.

4. Bezdrátové připojení: Senzory jsou bezdrátově připojitelné k zařízení pomocí Bluetooth.

5. Hromadné nabíjení: Výrobce nabízí i hromadnou nabíječku na několik senzorů naráz.

Hlavní nevýhodou zde je hlavně cena, která je téměř astronomická. V dalších pracích bych se tedy rád věnoval i vývoji vlastních senzorů.

Celkově lze říci, že Vernier GO senzory jsou velmi užitečným nástrojem pro praktickou výuky již v základu. Chybí jim však efektivní připojení mnoha senzorů najednou, či sdílení naměřených hodnot do více zařízení. Hlavním úskalím se pak stává zobrazení dat v rozšířené realitě. Oficiální aplikace od výrobce totiž brýle Hololens vůbec nepodporuje. Tento fakt mě přiměl k prozkoumání protokolu, kterým sondy komunikují, a navrhnutí vlastní implementace.

Vycházel jsem z knihoven pro komunikaci napsanou v Pythonu[4] a JavaScriptu[5], které jsem našel volně dostupné na internetu [10] [11]. Tyto knihovny nelze využít pro cíl této práce, jelikož ani jeden jazyk není určen k rychlé a spolehlivé komunikaci na nízké úrovni. Případné využití těchto knihoven by znamenalo zásadní omezení v počtu potencionálně připojených zařízení. Knihovny jsem shledal velice čitelně napsané, a proto přepsání do jazyka C++ nebylo nijak časově náročné.

[4] Python: skriptovací jazyk určen převážně k analýze dat

[5] JavaScript: skriptovací jazyk určen převážně k použití na webu

2.2.1 Porozumění Vernier protokolu

Vernier protokol není veřejně zdokumentován. V rámci této části se proto věnuji jeho popisu s cílem pomoci ostatním vývojářům, kteří se s ním setkají, a ušetřit jim čas.

Senzory komunikují pomocí Bluetooth Low Energy (BLE) charakteristik. Tyto charakteristiky jsou endpointy[6], ze kterých lze číst data, ale zároveň do nich lze data zapisovat. Každé zařízení má vedle standardních charakteristik, jako například na čtení stavu baterie, dvě speciální, proprietární charakteristiky. Jedna slouží k čtení a druhá k zápisu dat ve formě paketů. Tyto charakteristiky jsou součástí Vernier protokolu a jsou používány pro přenos dat a příkazů mezi senzorem a aplikací.

Obecná struktura paketu je následující:

Název	Typ paketu	Délka	Pořadí	Kontrolní součet	Typ příkazu	Parametry příkazu
Délka [Byte]	1	1	1	1	1	0 až (Délka - 5)

Tabulka 1: Vnitřní struktura Vernier paketu

Paket se tedy skládá z:

1. Typu
   - 0x58 – příkaz
   - 0x20 – naměřené hodnoty
   - 0xXX[7] - odpověď na příkaz
2. Délky
   - počet bytu od začátku po konec
3. Pořadí
   - Každý odesílaný paket musí mít tento údaj o jedno menší, než předchozí odeslaný
4. Kontrolní součet
   - Součet všech hodnot paketu pro následné ověření integrity dat
5. Typ příkazu

Hodnota	**Popis**	**Parametry**
0x10	Zjistit status	-
0x18	Zahájit měření	Fixní parametr: 0xFF, 0x01 Dynamický parametr: bitová maska senzorů: uint32 (4 byty)

[6] Endpoint: datový přístupový bod aplikace

[7] 0xXX symbolizuje jakoukoliv hodnotu od 0x00 po 0xFF vyjímaje 0x58 a 0x20

Hodnota	Popis	Parametry
0x19	Ukončit měření	Fixní parametr: 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 0xFF
0x1A	Zahájit komunikaci	Fixní parametr: 0xa5, 0x4a, 0x06, 0x49, 0x07, 0x48, 0x08, 0x47, 0x09, 0x46, 0x0a, 0x45, 0x0b, 0x44, 0x0c, 0x43, 0x0d, 0x42, 0x0e, 0x41 *Poznámka:* *počítadlo je potřeba nastavit na hodnotu 0xFE*
0x1B	Nastavit periodu měření	Fixní parametr: 0xFF, 0x00 Dynamický parametr: perioda: uint64 (8 bytu)
0x50	Zjistit info o senzoru	Dynamický parametr: index: uint8 (1 byte)
0x51	Zjistit dostupné senzory	-
0x54	Odpojit	-
0x55	Zjistit info o zařízení	-
0x56	Zjistit výchozí senzory	-

Tabulka 2: Typy příkazů Vernier protokolu

Odesílání probíhá segmentovaně. Maximální délka jednoho zápisu je 20 bytů.

```
void VernierDevice::sendPacket(PacketBase *packet) {
    uint8_t *data = packet->toArray();
    uint8_t lenRemaining = packet->getLength();
    uint8_t offset = 0;

    while (lenRemaining > 0) {
        uint8_t chunk = min(lenRemaining, (uint8_t) 20);
        _commandCharacteristic->writeValue(data + offset, chunk);
        lenRemaining -= chunk;
        offset += chunk;
    }
}
```

Zdrojový kód 1: Odesílání Vernier paketu

Pro zajištění párovatelnosti požadavků a odpovědí má přijatý paket stejnou hodnotu sekce “Pořadí” jako paket odchozí.

2.2.2 C++ implementace komunikace

Při implementaci jsem se zaměřil hlavně na balanc mezi jednoduchou škálovatelností a rychlostí. Využil jsem pro tento cíl OOP paradigma. Struktura je následující.

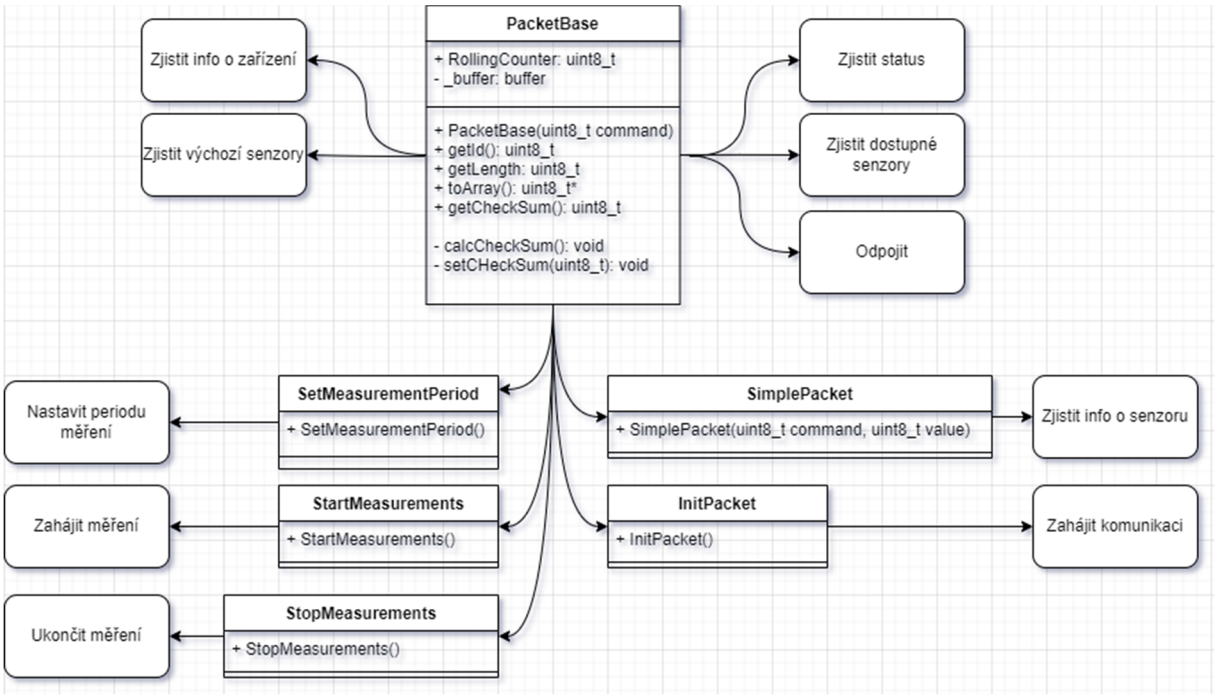


Diagram 1: Implementace Vernier protokolu

Realizaci jednotlivých tříd můžete najít ve zdrojovém kódu, soubor Packets.h: Příloha 1: Zdrojový kód.

Zde přikládám pouze na ukázku kód třídy InitPacket:

```
struct InitPacket : public PacketBase
{
   InitPacket() : PacketBase(EVernierCommand::Init)
   {
       RollingCounter = 0xFF;
       _buffer.data[2] = --RollingCounter;

       static uint8_t data[] = {
               0xa5, 0x4a, 0x06, 0x49,
               0x07, 0x48, 0x08, 0x47,
               0x09, 0x46, 0x0a, 0x45,
               0x0b, 0x44, 0x0c, 0x43,
               0x0d, 0x42, 0x0e, 0x41};

       _buffer.write(data, 20);
   }
};
```

Zdrojový kód 2: Implementace struktury InitPacket

Vzhledem k tomu, že IoT zařízení mají omezené prostředky, využití kolekcí z knihovny std je dosti komplikované. Jejich použití s sebou nese vyšší náročnost na procesor a paměť. Můžete si proto všimnout proměnné, která je datového typu s názvem "buffer". Jedná se o můj vlastní typ pro ukládání dat, u kterých předem neznám jejich délku. Struktura Buffer je velice jednoduchá čili v důsledku i nenáročná. Její zdrojový kód zároveň považuji za další ideální příklad jazyka C++.

```
#define BUFFER_SIZE 256

struct buffer {
   uint8_t data[BUFFER_SIZE];
   uint8_t length;

   void write(uint8_t value) {
       if (length + 1 < BUFFER_SIZE) data[length++] = value;
   }

   void write(void *values, size_t count) {
       if (length + count <= BUFFER_SIZE) {
           memcpy(data + length, values, count);
           length += count;
       }
   }

   void clear() {
       length = 0;
   }

   buffer() { clear(); }
};
```

Zdrojový kód 3: Implementace struktury buffer

2.3 Síťová komunikace

2.3.1 Vyhledávací protokol

Pro usnadnění připojení nových zařízení k serveru byl navržen jednoduchý protokol nálezu. Server vysílá UDP pakety na zvolenou multicast[8] adresu 239.244.244.224 a port 2442. Každé zařízení, které chce s tímto serverem komunikovat, naslouchá na této adrese a portu a čeká na příchozí pakety. Tento protokol umožňuje rychlé a jednoduché připojení nových zařízení k serveru bez nutnosti ruční konfigurace síťových parametrů.

Paket má též velice jednoduchá pravidla. Pokud datová část paketu začíná textem "VernierMasterNode", jedná se o paket validní.

[8] Multicast: doručí data více adresátům naráz

2.3.2 Obecný princip vlastního protokolu

Všechny mé protokoly využívají TCP protokol s vlastní nadstavbou s výjimkou vyhledávacího protokolu, který využívá UDP komunikaci. Všechny protokoly jsou umístěny v aplikační vrstvě TCP/IP[9] modelu, což je na úrovni velmi známého a využívaného HTTP. Ten se bohužel vyznačuje vysokou náročností na zdroje zapříčiněnou přílišnou sofistikovatelností, proto nebyl vhodným kandidátem pro mé využití.

Každý protokol se řídí jednoduchými pravidly: nejprve je navázáno TCP spojení, následně je odeslán 12 bytový unikátní identifikátor zařízení a dále se pravidelně kontroluje, zda je spojení stále aktivní. Pokud spojení funguje správně, data jsou odesílána a přijímána bez problémů. V případě poruchy spojení se postupuje od znovunavázání TCP spojení a odeslání identifikátoru. Díky unikátnímu identifikátoru server rozpozná relaci a umožní komunikaci bez dalších problémů.

2.3.3 Protokol kontroly funkčnosti

Pro zajištění správné funkce serverového systému je nezbytné pravidelné monitorování dostupnosti klientů a sběrných zařízení. Pro tento účel je nutné, aby každé zařízení periodicky (s frekvencí 5 sekund) odesílalo zprávu o své připravenosti ke komunikaci na server. Pokud server neobdrží tuto zprávu po více než 5 pokusů za sebou, automaticky se zařízení odpojí od všech služeb a již s ním nebude možné komunikovat do dalšího připojení. Tento postup zajišťuje bezproblémový chod systému.

Vyhradil jsem pro tento protokol port 4224.

2.3.4 Příkazový protokol

Ovládání jednotlivých sběrných zařízení je umožněno pomocí příkazového protokolu (port 4444) a protokolu událostí (port 2222). Tedy podobné paradigma, jenž jsme viděli již v sekci Porozumění Vernier protokolu.

Strukturu paketu příkazu však realizuji mnohem jednodušeji:

Název	Délka	Příkaz	Parametry
Velikost [Byte]	1	1	0 až (Délka - 2)

Tabulka 3: Vnitřní struktura příkazového paketu

Hlavním rozdílem vůči Vernieru je absence typu, kontrolního součtu a pořadí.

Jelikož protokol slouží pouze pro odesílání a přijímání příkazů, informace Typ se stává nepotřebnou.

[9] TCP/IP: Neboli TCP přes internetový protokol, strukturalizace datových toků do jednotlivých úrovní od fyzické po aplikační

Vzhledem k tomu, že komunikace probíhá přes TCP spojení, lze vypustit Kontrolní součet, protože správnost dat zajišťuje TCP samotné.

Zbavíme se tedy i Pořadí, neboť se jedná o účelový protokol, kde server zasílá požadavek na vykonání akce a příjemce následně odešle aktuální stav požadovaného úkonu protokolem událostí. Vše je tedy zpracováno v synchronním módu, čímž je dosažena vysoká účinnost a rychlost komunikace.

Příkazů máme hned několik:

Hodnota	**Popis**	**Parametry**
0x01	Začít BLE sken	-
0x02	Ukončit BLE sken	-
0x03	Připojit se k zařízení	uint64 deviceId
0x04	Odpojit se od zařízení	uint64 deviceId
0x05	Zahájit sběr dat	uint64 deviceId, uint32 sensorId
0x06	Ukončit sběr dat	uint64 deviceId

Tabulka 4: Možné příkazy příkazového protokolu

Myslím, že v tomto kontextu je nutné zdůraznit výhodu jednoduchého designu, který umožňuje rychle a spolehlivě zpracovávat příkazy. Jednoduchý přístup se v tomto případě ukázal jako klíčový prvek úspěchu.

2.3.5 Protokol událostí

Jak již bylo zmíněno v sekci Příkazový protokol, zpětná vazba tvoří druhou polovinu komunikačního kanálu mezi serverem a sběrným bodem. Paket události kopíruje stejné datové rozpoložení a paradigma jako u paketu příkazu.

Název	Délka	Událost	Parametry
Velikost [Byte]	1	1	0 až (Délka - 2)

Tabulka 5: Vnitřní struktura paketu událostí

Typy událostí dle následující tabulky:

Hodnota	**Popis**	**Parametry**
0x01	Sken zahájen	-

Hodnota	Popis	Parametry
0x02	Sken ukončen	-
0x03	Zařízení připojeno úspěšně	uint64 deviceId, uint32 sensorMask, uint32[] sensorIds
0x04	Zařízení připojení selhalo	uint64 deviceId
0x05	Zařízení odpojeno	uint64 deviceId
0x06	Zařízení nalezeno	uint64 deviceId
0x07	Měření zahájeno	uint64 deviceId, uint32 sensorId
0x08	Měření ukončeno	uint64 deviceId
0x09	Info o senzoru	uint64 deviceId, SensorInfo sensorInfo

Tabulka 6: Možné typy událostí

2.3.6 Datový protokol

Přenos naměřených hodnot zajišťuje protokol na portu 2224. Jedná se o sofistikovanější provedení, než jsme mohli vidět u předchozích, avšak stále držící se zásady co nejjednodušší struktury dat.

Obecné Info	Info o zařízeních	Info o Senzorech			Naměřené hodnoty
Počet zřízení	Počet senzorů	Senzor ID	Je celé číslo	Počet hodnot	Hodnoty
2	2	302	TRUE	3	11
					12
					13
		Senzor ID	Je celé číslo	Počet hodnot	Hodnoty
		303	FALSE	2	24.1

Obecné Info	Info o zařízeních	Info o Senzorech			Naměřené hodnoty
					24.7
	Počet senzorů	Senzor ID	Je celé číslo	Počet hodnot	Hodnoty
	1	421	FALSE	2	1126
					1208

Tabulka 7: Strukturovaný pohled na datový paket

Můžeme zde pozorovat již náznak seskupování sbíraných údajů, které je do detailu probrané v sekcích Implementace vlastních protokolů a Odesílání dat.

Ukázkový paket by tedy dle předchozí tabulky vypadal následovně:

2	2	302	TRUE	3	11	12	13	303	FALSE	2	24.1	24.7	1	421	FALSE	2	1126	1208

Tabulka 8: Výsledný ukázkový datový paket

Nejdříve se nám vyskytuje obecné informace udávající počet zařízení. Následuje pole informací o zařízení o délce u právě zde uvedené.

Každé zařízení obsahuje počet senzorů, pro které jsou dostupné hodnoty a následuje přesně tak dlouhé pole informací o senzorech

Informace o senzorech tvoří již více kolonek.

1. Senzor ID: Identifikátor typu senzoru
2. Zda je hodnota celé číslo: Hodnota může být celé číslo (int) nebo také číslo reálné (float), zde se rozhoduje, jak bude s následujícími daty nakládáno.
3. Počet naměřených hodnot
4. Pole hodnot o délce z kolonky číslo 3

2.4 Mikroprocesor ESP32

Mikroprocesor ESP32, varianta S3, od firmy Espressif je zásadním komponentem celé infrastruktury. Zkoušel jsem i jiné varianty např. C3 ta však trpěla občasnou ztrátou připojení. Příčina defektu mi není známa.

Jeho úkol činí hlavně sběr dat z podružných senzorů. Tato data následně seskupovat a odesílat na řídící server.

Zvolil jsem právě toto zařízení, jelikož již v základu disponuje Wi-Fi a BLE konektivitou, dvěma CPU jádry a mnoho RAM paměti, konkrétně 520KB interní a možností až 8 MB externí. Jeho cena je taktéž velmi přívětivá, okolo 150 Kč, tedy ideální pro studenta na seberozvoj.

2.4.1 Připojení k síti Wi-Fi

Připojení k Wi-Fi nečinilo žádná úskalí, jelikož jsem použil vestavěnou knihovnu výrobce.

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
...
Serial.print("Connected with local address: ");
Serial.println(WiFi.localIP());
```

Zdrojový kód 4: Připojení k síti Wi-Fi

V případě nasazení do produkčního prostředí nastává ale jiný problém, adaptace na místní infrastrukturu. Za tímto účelem by bylo možno nastavit aktuální název Wi-Fi a heslo přes web rozhraní přímo mikroprocesoru, což by ale znamenalo značné vytížení prostředků. Mnohem elegantnějším se tedy nabízí například nastavení přes BLE a mobilní aplikaci.

2.4.2 Implementace vlastních protokolů

Jednotlivé protokoly jsem se rozhodl aplikovat ve formě “služeb”. Separátních procesů, které mezi sebou komunikují. Výsledkem celého orchestru se pak stává vysoce elastické a škálovatelné řešení.

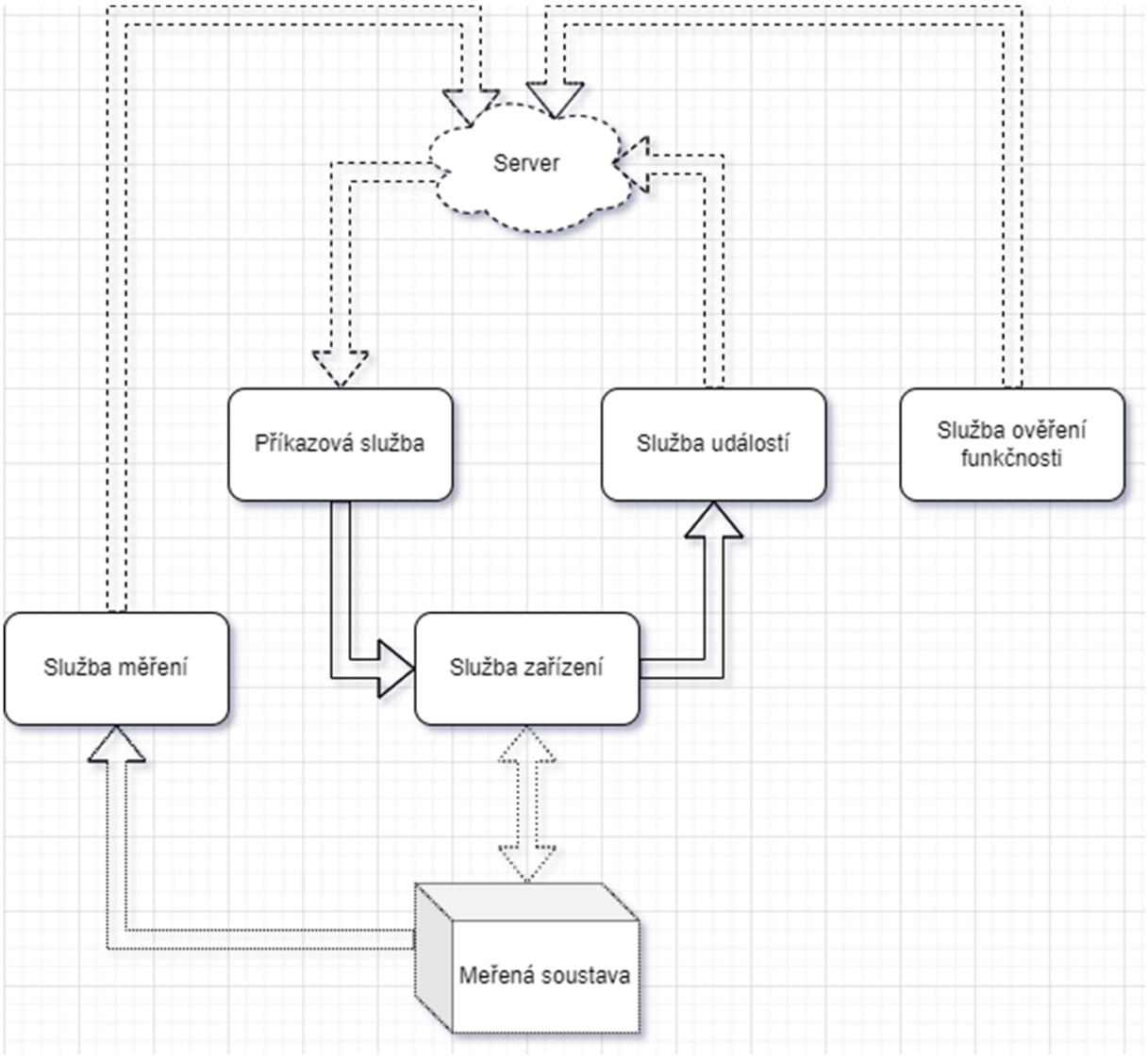


Diagram 2: Služby na mikrokontroleru

Služba ověřování funkčnosti zastává práci taktéž práci vyhledávacího protokolu, pro nalezení dostupných serverů na lokální síti. Po získání validního spojení, zpřístupní IP adresu serveru i zbytku aplikace.

Příkazová služba a služba událostí zde mají podobnou roli jakož tomu je u Vernier sond. Komunikují s nadřazeným zařízením, vykonávají požadované úkony a odpovídají asynchronně skrz separátní kanál.

Služba zařízení se pak stará o samotné komunikování s Vernier senzory. Instrukce mu uděluje příkazová služba. Výsledek následně předává zpět službě událostí. jestliže načte vernier paket, který je typu "Naměřené hodnoty", uloží ho pomocí dual_lockable do mezi paměti.

Služba měření pak shromáždí uložené hodnoty a odešle je ve strukturované podobě na přidružený server.

Žádná služba není víc ani míň důležitá, jedna bez druhé by zkrátka nefungovaly.

2.4.3 Implementace Vernier protokolu

Jak již bylo zmíněno v kapitole Porozumění Vernier protokolu, realizace komunikace náleží jazyku C++. Využil jsem proto OOP (objektově orientovaného programování) a pro jednotlivé typy paketů vytvořil samostatné struktury. Všechny dědí ze struktury PacketBase. Většina komunikace si s touto strukturou vystačí, jelikož nevyžadují žádné parametry, pouze jejich typ. Ty používanější či s větším počtem parametrů, se dočkali jmenovité implementace, např. InitPacket. Avšak ty, které vyžadují pouze jeden parametr, například "Zjistit info o senzoru" využívají třídu SimplePacket. Viz schéma C++ implementace komunikace.

Samotné odesílání a následné přijetí paketů je řešeno asynchronně, což bývá často problematické. Charakteristika odpovědí má naštěstí možnost naslouchání na změnu hodnoty, tudíž se implementace dosti zjednodušuje.

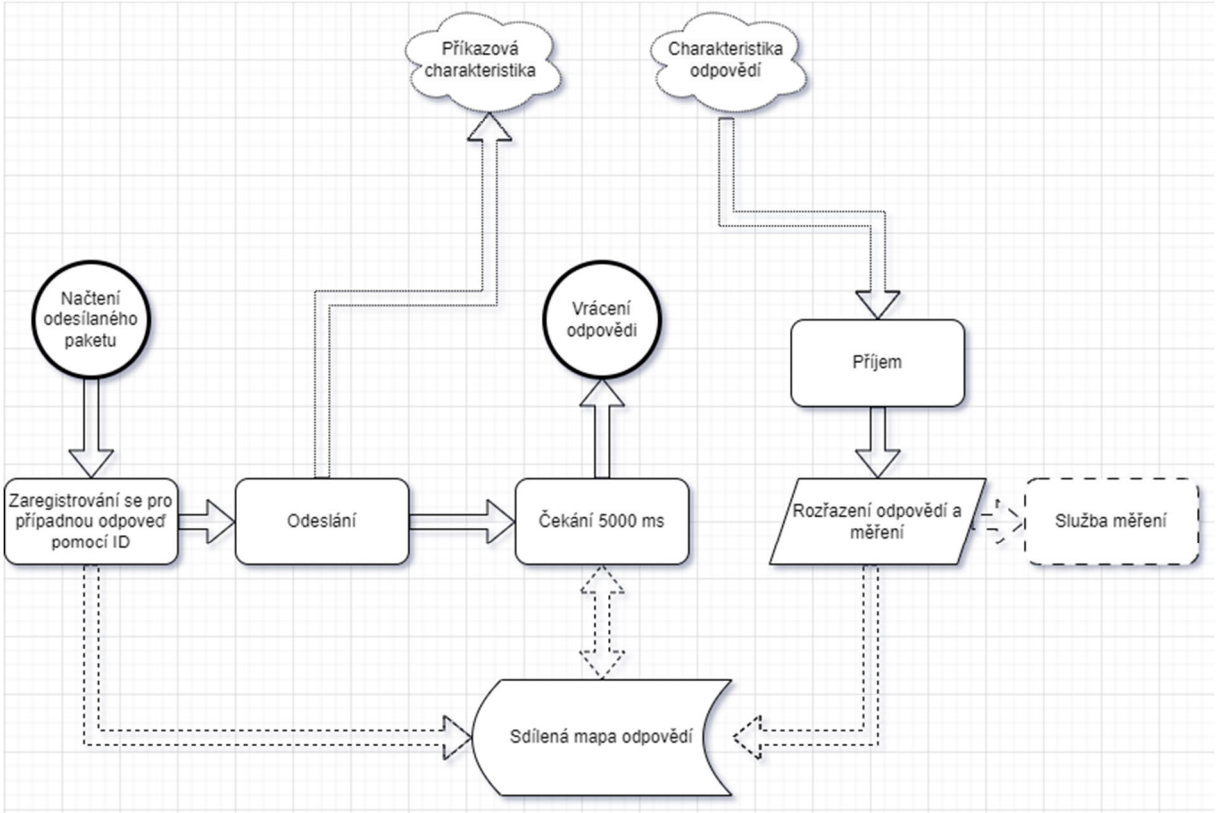


Diagram 3: Vnitřní implementace Vernier protokolu na mikrokontroleru

Skládá se tedy ze dvou částí, odeslání a příjmu.

Nejdříve se paket zpracuje, vypočte se kontrolní součet a přiřadí sekvenční identifikátor.

Odesílací část se "zaregistruje" tím, že specifikuje ID a místo v paměti, kam se má odpověď zapsat. Tudíž, jakmile přijde odezva, přijímací část zapíše data do specifikovaného místa.

Na toto již první část čeká a vrátí uživateli danou hodnotu. Kdyby se náhodou nepodařilo paket přijmout, je nastavena maximální doba čekání a to 5000 ms.

2.4.4 Zajištění integrity dat

Tím, že je systém založený na fragmentaci do služeb, vzniklo mnoho komplikací s problematikou více vláknových aplikací.

Jako řešení se ukázalo vytvořit jeden zásadní datový typ a to "dual_lockable". Jedná se o generický typ, který implementuje dvouzámek pro objekty typu T[10]. Tento dvouzámek umožňuje dvěma vláknům nebo procesům přistupovat k objektům typu T bez konfliktů, takže mohou být bezpečně používány v multithreadovém nebo paralelním prostředí. Struktura v mém použití drží vždy dva objekty.

Tato třída obsahuje následující metody:

- **lock()**: metoda pro získání exkluzivních práv na data. Blokuje tím jakékoliv jiné vlákno od přístupu k datům, aby je nemohlo pozměnit.
- **unlock()**: metoda pro uvolnění dat, pro použití někým dalším.
- **swap()**: metoda prohodí aktivní data, a vrátí data aktuálně nevyužívána.
- **current()**: metoda pro získání aktuálního objektu typu T.

Následuje názorná ukázka použití při seskupování naměřených hodnot.

Ukázka vlákna sběru, vlákno vždy získá přístup k aktuálnímu bufferu a zapíše své hodnoty:

```
...
SensorInfo *info = sensorInfos[i];
uint8_t *data = response->getData(offset++);
uint8_t dataCount = response->getDataCount();

info->values_buffer.lock();
buffer *buff = info->values_buffer.current();

buff->write(data, dataCount);
info->isInts = response->isInts();
info->values_buffer.unlock();
...
```

Zdrojový kód 5: Ukázka zápisu dat do bufferu ze separátního vlákna

Ukázka vlákna zpracování, po získání práv prohodí aktivní data a hned umožní přístup předešlému vláknu, nyní již neaktivní data projde, zpracuje a vymaže:

[10] Typ T – libovolný datový typ

```
...
SensorInfo *info = device->sensorInfos[i];
info->values_buffer.lock();
buffer *buff = info->values_buffer.swap();
info->values_buffer.unlock();

...
//Zpracování dat
...

buff->clear();
...
```

Zdrojový kód 6: Ukázka čtení dat z bufferu ze separátního vlákna

2.4.5 Odesílání dat

Poté, co jsou hodnoty senzorů zaznamenány, nastává otázka, jak je odeslat na server co nejefektivněji. Odesílat každou hodnotu zvlášť se možná jeví jako nejjednodušší řešení, avšak tento způsob bývá velice náročný, jak na výpočetní výkon, tak na vytížení datového spojení.

Proto jsem se rozhodl data seskupit a odeslat v určitém intervalu. Data jsou odesílána frekvencí 10 Hz, tedy v intervalu po 100ms. Pro snížení latence lze toto nastavení změnit. Po celou dobu testování a zkoumání jsem zpoždění nijak znatelně nezpozoroval, tudíž jsem necítil potřebu vzorkovací frekvenci nijak měnit.

2.5 Server

Zařízení nazvané jako “Server” může být jakýkoliv počítač s možností připojení k lokální síti a schopností spustit dotnet 7.0 runtime.

2.5.1 Abstrakce připojení

Pro jednodušší práci s celou sítí sběrných bodů a senzorů je zapotřebí správně nastavit míru abstrakce.

Tím, že nevyužívám standardního protokolu HTTP, nebo jeho varianty, nýbrž vlastní komunikaci narazil jsem na mnohé problémy. Jazyk C# sice nabízí předpřipravené třídy na komunikaci se sítí (např. TcpListener nebo TcpClient), ale žádná z nich neumožňuje svižnou manipulaci se surovými daty. Tento problém jsem částečně obešel využitím třídy Socket, která nabízí nejnižší možný přístup v C#. Pomocí ní jsem si vytvořil třídu “TcpService”, ze které ostatní služby protokolů vycházejí. TcpService implementuje Obecný princip vlastního protokolu. Čeká na otevření nového TCP spojení na specifikovaném portu, přečte z něj unikátní identifikátor a zařadí. Veškerá jiná, aplikačně specifická, funkcionalita je nadále implementována až konkrétními službami.

2.5.2 Abstrakce dat

Abstrakce dat je důležitá nejen pro jejich ukládání, ale hlavně pro jejich zpracovávání a následnou analýzu. Proto jsem vytvořil třídy EspDevice, VernierDevice a VernierSensor.

2.5.2.1 EspDevice

Obsahuje informace jakožto:

- Unikátní identifikátor
- Připojená zařízení
- Zařízení v okolí
- Zda je zapnutý BLE sken
- Datum posledního ohlášení

Zároveň dává možnost přihlásit se k události “SensorValuesUpdated”, sloužící k asynchronnímu ohlášení o přijatých hodnotách.

Pro interní fungování třída též obsahuje frontu hodnot na zpracování, více informací naleznete v sekci Zpracování dat.

2.5.2.2 VernierDevice

Veřejně viditelné informace:

- Název
- Unikátní identifikátor
- Senzory

Slouží hlavně jako seskupovač a identifikátor fyzických zařízení.

2.5.2.3 VernierSensor

Slouží hlavně k uchovávání informací o senzoru dané výrobcem:

- Zda se jedná o reálné či celé číslo
- Pořadí
- Identifikátor typu
- Vzorkovací mód
- Popis
- Jednotky měření
- Nejistota měření
- Minimální hodnota měření
- Maximální hodnota měření
- Minimální perioda měření

- Maximální perioda měření
- Typická (doporučená) perioda měření
- Zda je zakázáno měření s nějakým dalším senzorem naráz

Používám tento typ také jako úložiště naměřených hodnot.

2.5.3 Zpracování dat

Formát dat není nijak složitý, viz Datový protokol. Avšak jejich zpracování v dostatečné rychlosti tak jednoduché není.

Po přijetí balíku dat od mikroprocesoru, data nejsou ihned zpracována, nýbrž předána dál. Převedení dat do abstraktní podoby je oddáleno tím, že se nejdříve zařadí do fronty. Zde vyčkávají, než je zpracují dedikovaná vlákna. Počet dedikovaných vláken se dynamicky mění dle aktuální vytíženosti každých 100 ms. Pro lepší demonstraci přikládám diagram zpracování.

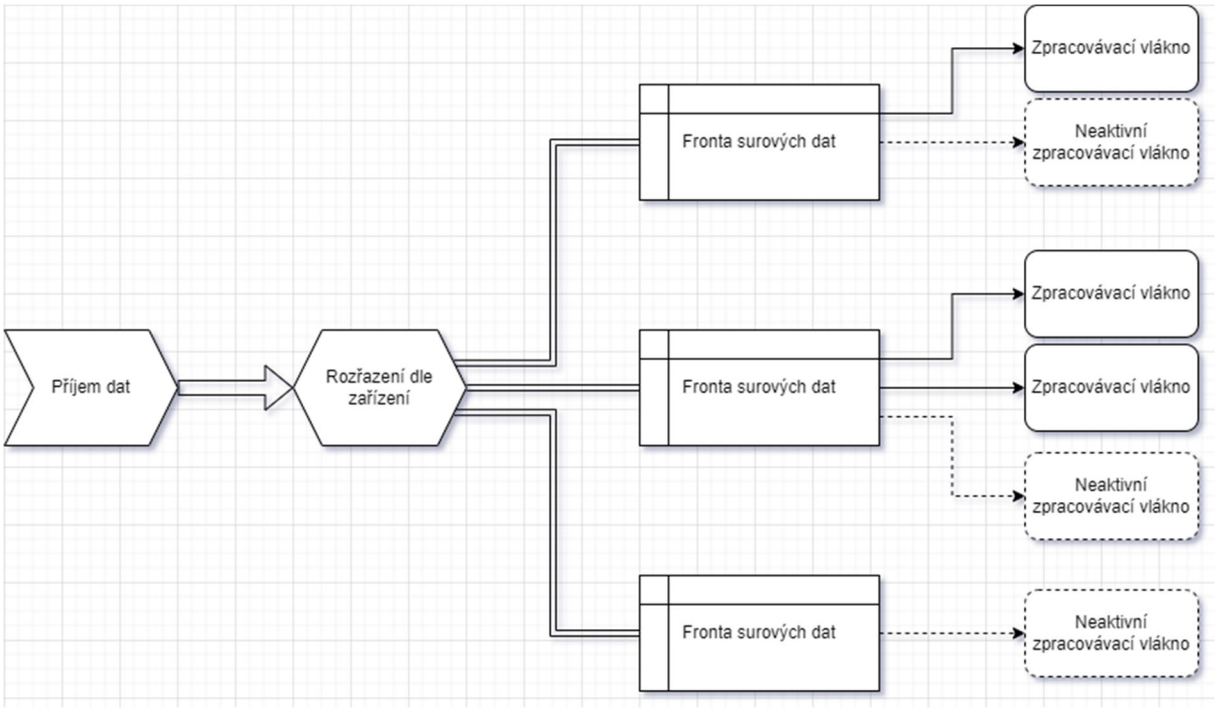


Diagram 4: Zpracování dat do abstraktní podoby více vláknovým způsobem

Po úspěšném zpracování je s daty nakládáno pouze v jejich abstraktní podobě. V tomto kroku tedy ztrácí podobu pouhých sekvencí hodnot a stávají se z nich vhodní kandidáti k uložení, analýzu a vizualizaci.

2.5.4 Uložení dat

Pro retrospektivní analýzu je samozřejmě potřeba hodnoty někam ukládat. Zvolil jsem metodu úchovy do databáze pomocí Entity Framework. Tato technologie nám umožní velice jednoduchou práci nad databází. Způsob “code first”, kdy celé schéma databáze píšeme

v programovacím jazyce, v mém případě c#, a následně pustím “generátory”, které za nás databázi vytvoří, se mi zdá zdaleka nejrychlejší pro vývoj. Datové schéma objektů je stejné jako v sekci Abstrakce dat.

Ukládání dat není v aplikaci implementováno, pouze plánováno, jelikož se aplikace nachází ve fázi stálého vývoje.

2.5.5 Zpřístupnění dat

Naměřené, zpracované a uložené hodnoty by nám ve výsledku k ničemu nebyly, kdybychom je nemohli též nějak zužitkovat. Komunikační knihovna SignalR je naprosto ideální pro náš cíl nízké latence a vysoké rychlosti. Funguje na principu server-klient architektury. Server, takzvaný “HUB” umožňuje klientovi se připojit a spouštět různé metody a získat zpět jejich výsledek.

V našem případě “RealtimeHub” nabízí následující:

- Získej všechny připojené EspDevice
- Zaregistruj se pro události přicházející ze specifického EspDevice
- Zruš registraci událostí od specifického EspDevice
- Začni sken na EspDevice
- Ukonči sken na EspDevice
- Připoj se k VernierDevice
- Odpoj se od VernierDevice
- Začni sběr hodnot na VernierSensor
- Ukonči sběr hodnot z VernierSensor

Na klientu je poté možné volat tyto metody:

- Sken byl zapnut na EspDevice
- Sken byl ukončen na EspDevice
- Připojení k VernierDevice proběhlo úspěšně
- Připojení k VernierDevice se nezdařilo
- VernierDevice odpojeno
- VernierDevice nalezeno
- Sběr hodnot aktivní na VernierSensor
- Sběr hodnot ukončen na VernierSensor
- Nová hodnota na VernierSensor
- Informace o VernierSensor
- EspDevice připojeno
- EspDevice odpojeno

2.6 Klient

Klientská aplikace je realizována pomocí technologie UWP a XAML. Lze ji tedy bez ebemenších problému na Hololens. Hlavní požadavek, který celou tuto práci motivoval.

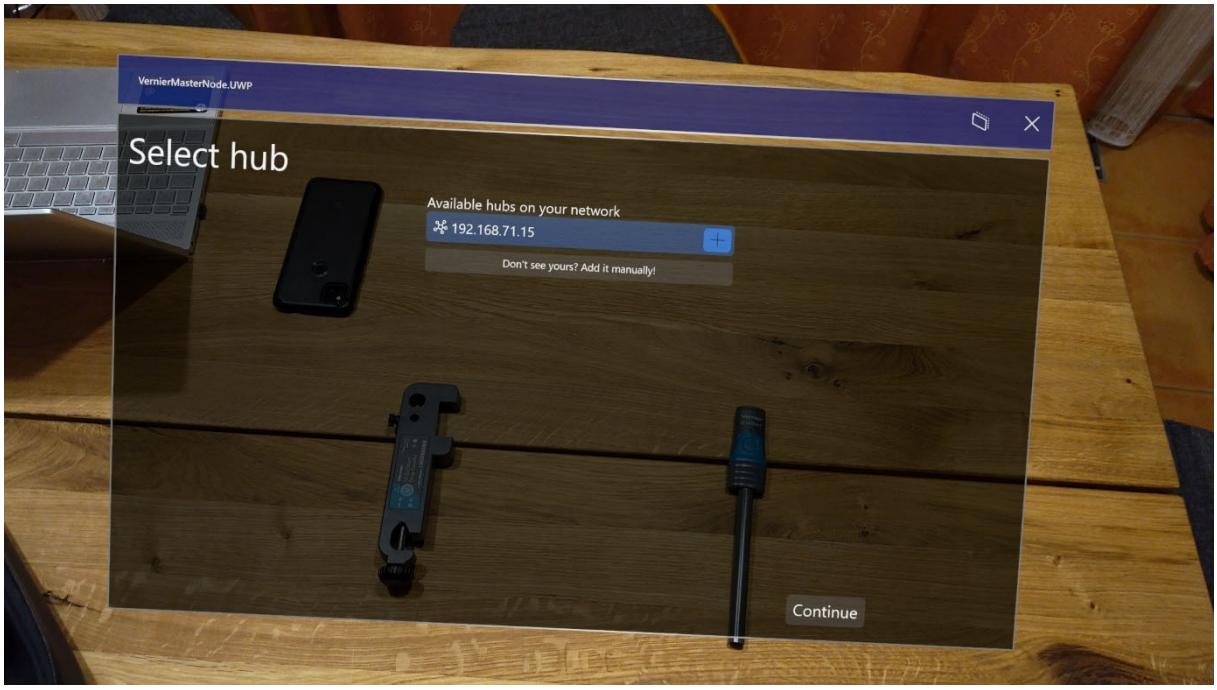
2.6.1 Implementace vyhledávacího protokolu

Implementace vyhledávacího protokolu bylo mnohem složitější, než tomu bylo u mikrokontroleru. Čím dále se vzdalujeme od programování na nízké úrovni, tím se zvětšuje náročnost implementace protokolů. Jelikož je ale tento protokol velice přínosný, je také jediný, který se používá ve všech vrstvách systému i přes úskalí, na které jsem v průběhu narazil.

Jedním takovým je například fakt, že i když explicitně povolíte aplikaci přistup do lokální sítě, Windows blokuje UDP komunikaci na uživatelem definovaných portech. Toto lze jednoduše obejít bez úprav firewallu tak, že si nastavíme aktuálně připojenou síť jako domácí. Tato jednoduchá oprava mi zabrala déle než hodinu zkoumání. Celkově hledání chyb při vývoji nativních Windows aplikací bylo velice zdlouhavé a úmorné.

Třída “HubDiscoveryService” se tedy stará o ukládání nalezených serverů na síti, kontroluje, zda jsou stále k dispozici, a umožňuje k nim přístup z ostatních částí aplikace.

Uživatel má tedy nakonec velice jednoduché rozhraní pro výběr, ke kterému lokálnímu serveru se připojit.

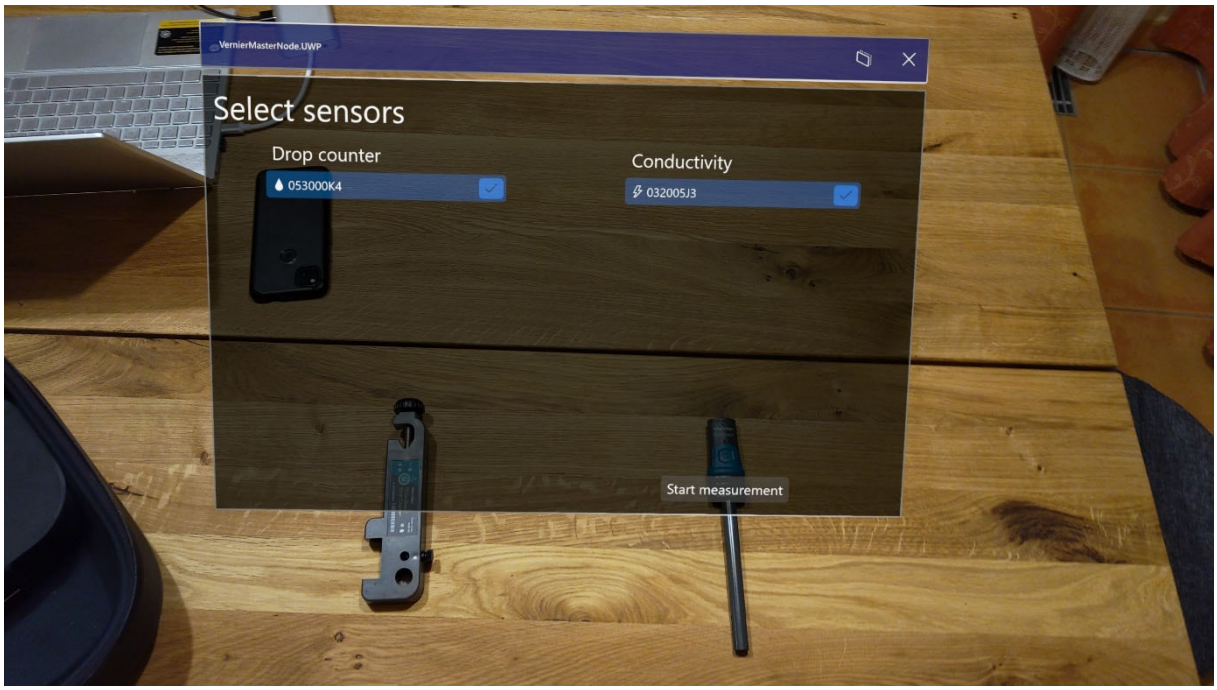


Obrázek 1: Výběr lokálního server

2.6.2 Získání dat

Po úspěšném připojení k serveru následuje výběr požadovaných senzorů.

Veliká výhoda kombinace C# a SignalR spočívá v její multiplatformnosti. Mohu tedy sdílet stejné návrhy tříd mezi serverem a klientem. Důsledkem je velice jednotné předávání si dat bez nutnosti redundantní deklarace tříd do více jazyků.

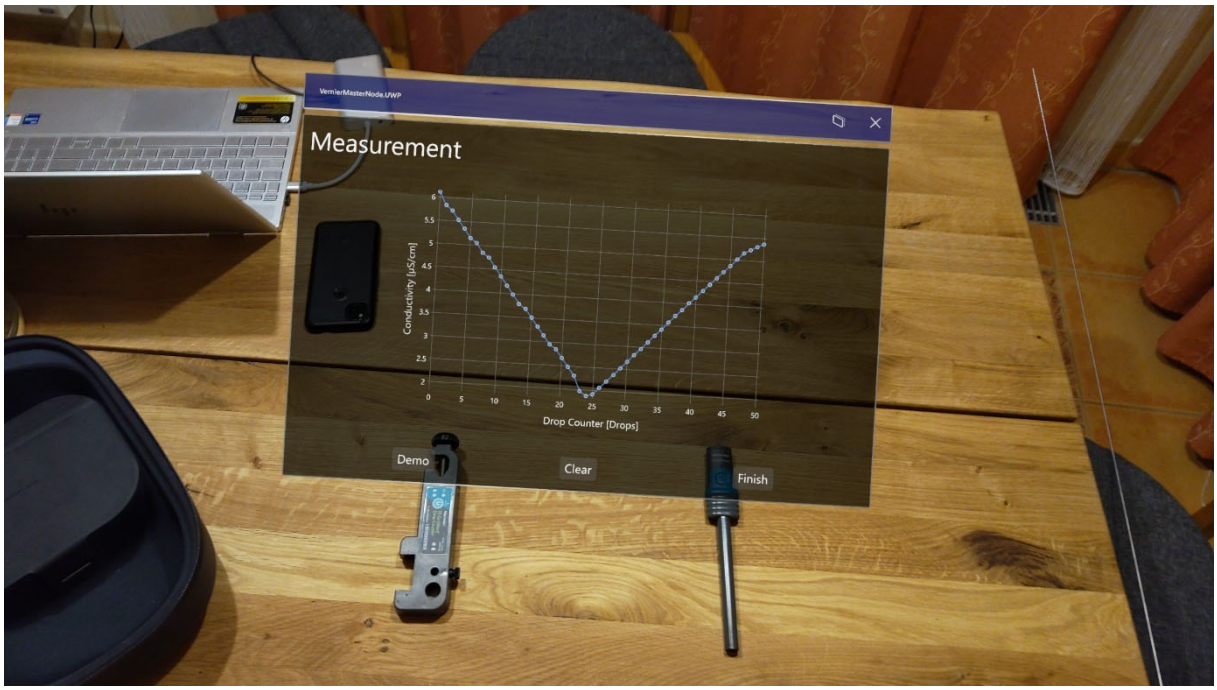


Obrázek 2: Výběr senzorů

2.6.3 Zpracování dat

Po zahájení měření jsou data živě přenášena do klientské aplikace. Finálním cílem je zpracovávat data plně pouze na serveru a odklonit tak výpočetní výkon dál od mobilních zařízení pro maximalizace výdrže baterie. V současné chvíli je ale stále potřeba alespoň minimální zapojení.

Jako příklad bych uvedl měření konduktometrické titrace. Zkráceně, při každé kapce je potřeba zanést naměřenou vodivost roztoku do grafu. Proto se po celou dobu uchovává nejaktuálnější hodnota vodivosti, při události padající kapky se tento údaj propíše do zobrazeného grafu.



Obrázek 3: Průběh měření konduktometrické titrace

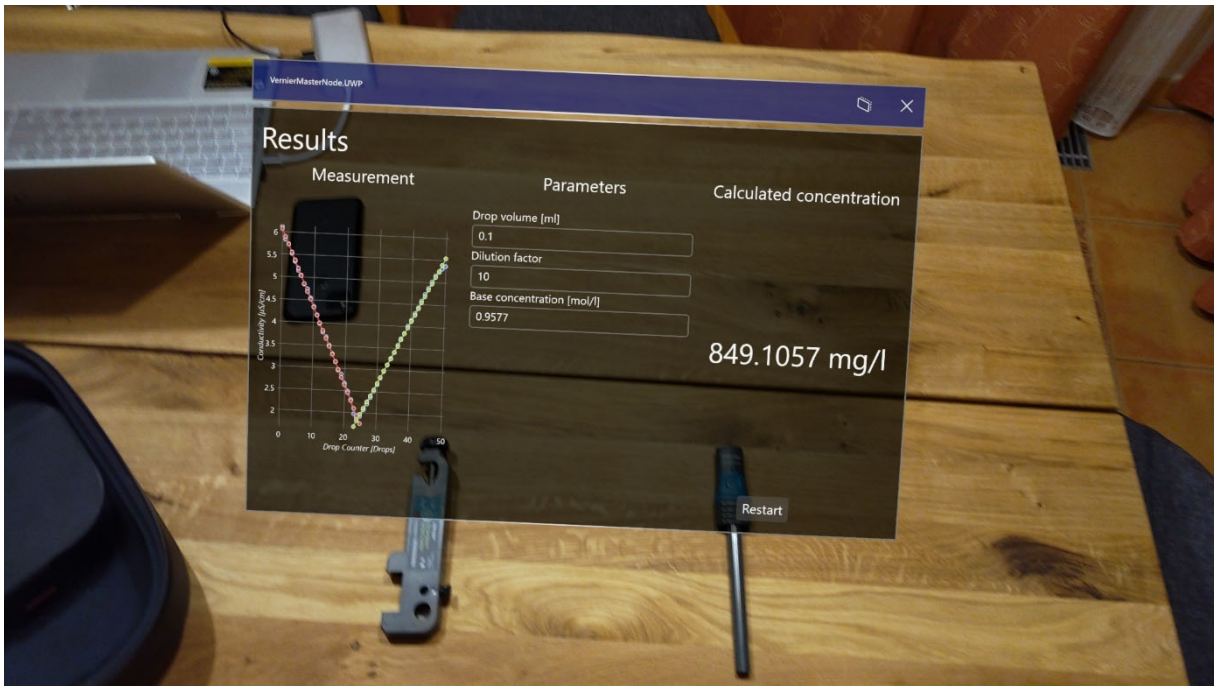
Aplikace již prošla zkouškou ohněm na reálné měřené soustavě v rámci konference Nové perspektivy v oblasti odborného vzdělávání a výcviku v měnícím se světě[11], která se odehrávala jakožto závěrečné shrnutí evropských projektů Dios[12] a FightARs[13]. Názor odborníka si můžete přečíst v sekci: Diskuse o možnostech rozšíření a vylepšení systému.

2.6.4 Vizualizace zpracovaných dat

Po ukončení měření je s daty i nadále manipulováno dle aktuálních potřeb.

V našem konkrétním případě je na datech provedena lineární regrese pro odstranění šumu a nepřesností měření. Výsledkem se pak tedy stává graf připomínající písmeno V.

[11] Závěrečná prezentace výsledků projektů Dios a FightARs
[12] DIOS – Erasmus+ KA202, projekt EU zabývající se využitím AR ve výuce
[13] FIGHTARs – Erasmus+ KA202, projekt EU zabývající se využití AR při tréninku hasičů



Obrázek 4: Výsledné shrnutí naměřených hodnot a interaktivní výpočty

2.6.5 Přidružené výpočty

Jelikož se určité proměnné a konstanty liší například podle užitých látek, a nebylo by praktické všechny tyto hodnoty též měřit, máme zde i možnost nastavit hodnoty různých proměnných ručně, přičemž ihned po editaci, se změny promítnou do výsledku. Tímto zajišťuje maximální flexibilitu a responzivitu.

3 APLIKACE VÝUKY

Potencionální využití ve výuce je enormní, jelikož praktické zkušenosti jsou jedny z nejhodnotnějších. Zároveň díky centralizaci dat může učitel kontrolovat jednotlivé práce studentů v reálném čase a rychleji reagovat v případě nesprávného postupu. Nadále uvádím příklady využití, které mi přišly zajímavé z jednotlivých oblastí.

3.1 Využití IoT ve výuce chemie

Zaměření na tuto oblast odstartovalo celý projekt. Jelikož ale nejsem chemik, musím se odkázat na pana Ing. Návesníka Ph.D. a jeho zpětnou vazbu v sekci Diskuse o možnostech rozšíření a vylepšení systému.

Příkladové úlohy tedy čítají například titraci konduktometrickou, nebo potenciometrickou. Ale v zásadě jakékoliv instrumentální měření.

3.2 Využití IoT ve výuce fyziky

Nemalou část také zastává výuka fyzikálních zákonů. Pro mě osobně byly laboratorní práce v tomto předmětu vždy potěšením. Plně si dokáži představit, o kolik zábavnější by se mohli stát pomocí této technologie.

Příkladové úlohy:

- Intenzita světla
- Intenzita magnetického pole
- Velikost sil
- Hydrostatický tlak
- Poloha
- Průtok plynů a kapalin

3.3 Využití IoT ve výuce elektroniky

Při měření elektrického obvodu lze zobrazit napětí, proud, otáčky motoru či jiné veličiny přímo v prostoru u daného zkoumaného prvku.

Příkladové úlohy:

- Paralelní a sériové zapojení: odporový dělič
- Operační zesilovač: koeficient zesílení
- Elektromotory: závislost otáček na proudu a napětí

Výhodou oproti tradiční výuce pak nastává flexibilita zobrazení měřených informací. Data nemusíme přenášet z měřáku do excelové tabulky, jelikož se nám graf tvoří automaticky přímo před očima.

4 ZÁVĚR

Za cíl jsem si před pojal vytvořit IoT síť senzorů s následnou možností vizualizace a analýzy dat.

Tento úkol se mi podařilo splnit nad moje očekávání, jelikož vzniklý produkt se vyznačuje téměř nekonečnou škálovatelností a flexibilitou.

Modelové použití by tedy vypadalo následovně. Mikrokontrolery ESP32 se umístí na každé z pracovišť studentů. Na jednu učebnu řekněme o 12 ti pracovištích bych doporučil jeden server, například v podobě Raspberry Pi[14] [12] a jeden až dva Wi-Fi přístupové body. Každý ze studentů si pak buďto pomocí osobního mobilního telefonu, nebo jiného, například školního,

[14] Raspberry Pi: mini-počítač

zařízení spáruje čidla, se kterými měří a dostává data v reálném čas přímo před svůj zrak. Učitel v průběhu celé hodiny může přehledně sledovat všechna měření žáků naráz.

Do budoucna bych chtěl pomoci zbudovat chytrou laboratoř založenou na této technologii na naší, či jiné škole, pro zlepšení intuitivní výuky. Střední průmyslová škola chemická Pardubice[15] již projevila o tuto realizaci zájem, což mě velmi potěšilo. Též bych chtěl rozšířit rámec mobilní aplikace pomocí technologie MAUI. Jedná se o plně multiplatformní technologii pro vývoj aplikací na Android, iOS, MacOS, Linux a Windows.

[15] Střední průmyslová škola chemická Pardubice – Poděbradská 94, 530 09 Pardubice

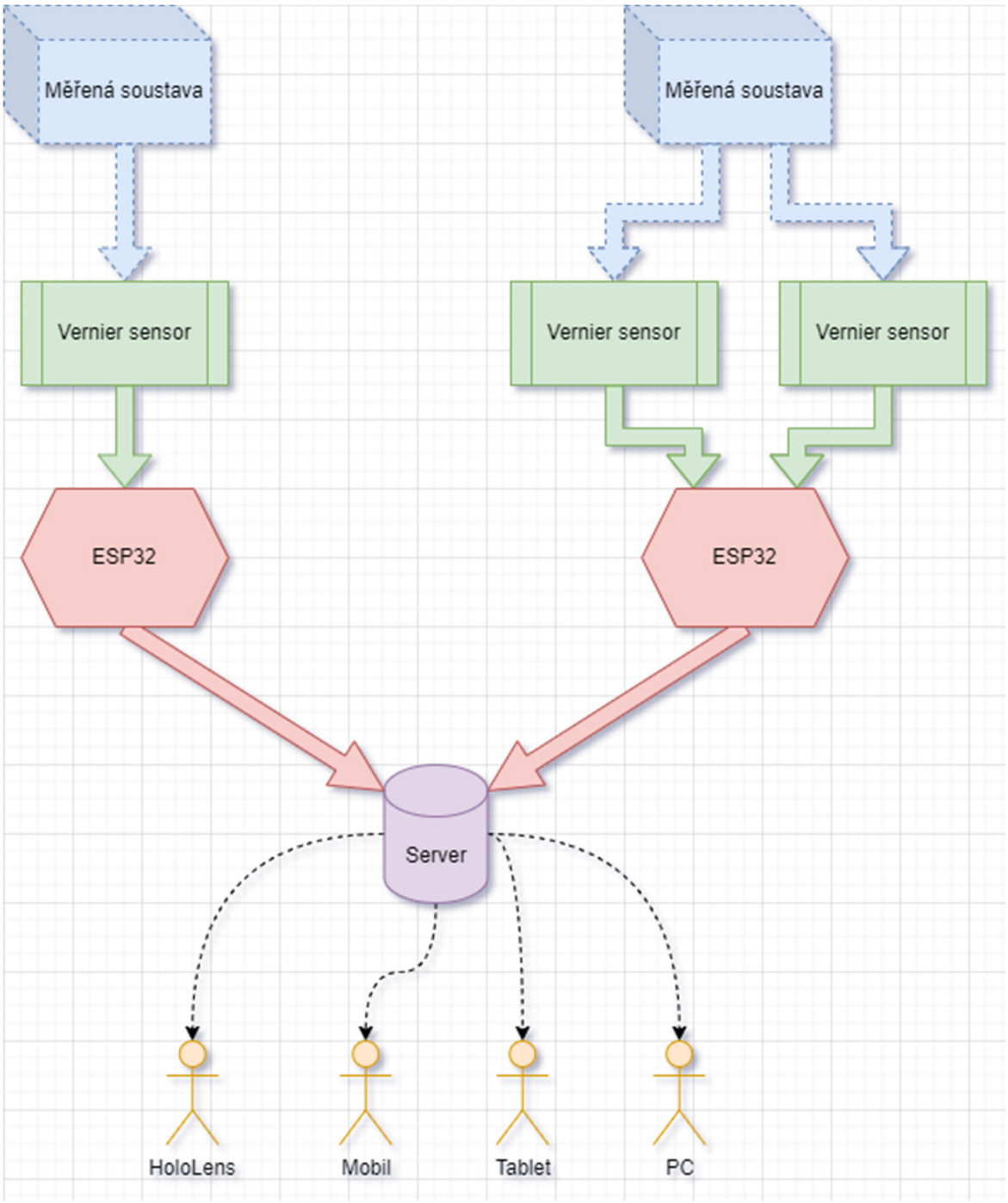


Diagram 5: Závěrečná podoba aplikace

4.1 Přínosy pro seberozvoj

Prací na tomto projektu jsem získal mnoho zkušeností s programováním v praxi. Teoretické základy mám již nějakou dobu, avšak chybělo jim pořádné zavedení do reálného světa. Jsem přesvědčen, že tento projekt obohatil nejen mě, ale doufejme, že i do budoucna náš vzdělávací systém pro dosažení ještě modernějších vzdělávacích postupů, než můžeme pozorovat do teď.

4.2 Diskuse o možnostech rozšíření a vylepšení systému

Oslovil jsem pana Ing. Jakuba Návesníka Ph.D., o zhodnocení přínosu pro oblast vyučování chemie, zde je jeho odpověď: “V rámci Erasmus + projektu DIOS (2020-1-DE02-KA202-007621) jsme na SPŠCH Pardubice řešili spojení AR/MR zařízeni Hololens II (Microsoft) s bezdrátovými senzory (Vernier) pro vytvoření Smart laboratoře využívající IoT. Vzhledem k velmi problematické komunikaci s partnerem z našeho konsorcia jsme vytvořili pouze variantu, kde jsou senzorická data sbíraná počítačem a dále streamovaná do Hololens brýlí pomocí softwaru Remote desktop. S pomocí Lukáše Šafránka jsme ale tuto řešení významně vylepšili. Lukáš vytvořil software v prostředí Hololens II, kde je možné reálná data z bezdrátových senzorů nejen přímo zobrazovat, ale zároveň je v reálném čase promítat do grafu závislosti jednoho senzorického parametru na druhém. V našem případě jsme toto softwarové řešení aplikovali pro reálnou konduktometrickou titraci silné kyseliny (HCl), silnou zásadou (NaOH). V této analytické operaci se zaznamenává závislost měrné vodivosti na přidávaném objemu titračního činidla (NaOH). Vzniklou charakteristickou křivku tvaru "V" je pak následně nutné vyhodnotit (získat objem činidla v bodě ekvivalence), tak aby bylo možné spočítat koncentraci vzorku kyseliny HCl. Lukáš tedy vytvořil ve svém navrhnutém softwaru možnost zpracování této křivky. Lineárním proložením sestupné a vzestupné větve grafu lze v softwaru vyhodnotit průsečík obou funkcí a získat tak přesný objem v bodě ekvivalence. Po dosazení objemu a dalších proměnných do matematického vyjádření lze následně softwarově vyhodnotit přesnou koncentraci neznámého vzorku HCl. Lukáš takto vytvořil unikátní softwarový/hardwarový nástroj, který významně přispěl do řešení E+ projektu a najde zároveň využití v laboratorních cvičeních 4. ročníku žáků SPŠCH Pardubice oboru aplikovaná chemie. Tento nástroj lze po následné drobné úpravě využít i pro ostatní instrumentální titrace. Celá řada elektrochemických metod (potenciometrie, konduktometrie, a dalších) tak může být takto zpracována.” (Ing. Jakub Návesník, Ph.D.)

Zpětná vazba mě velice potěšila a nemůžu se dočkat budoucí spolupráce, kdy mé řešení nasadíme do aktivní výuky.

5 POUŽITÁ LITERATURA

[1] Definice BLE (Bluetooth Low Energy). Počítačový slovník TechLib [cit. 2. 3. 2023] Dostupné z: https://tech-lib.eu/definition/ble.html

[2] Protokol TCP – Transmision Control Protocol. Ujep [cit. 2. 3. 2023] Dostupné z: http://physics.ujep.cz/~jkrejci/ZPP/Site/SPS%20Hradec/site007.html

[3] User datagram protocol (UDP). impreva [cit. 2. 3. 2023] Dostupné z: https://www.imperva.com/learn/ddos/udp-user-datagram-protocol/

[4] Modern C++ in embedded systems. embedded [cit. 2. 3. 2023] Dostupné z: https://www.embedded.com/modern-c-in-embedded-systems-part-1-myth-and-reality/

[5] C# (C-Sharp). TechTarget - WhatIs.com [cit. 2. 3. 2023] Dostupné z: https://www.techtarget.com/whatis/definition/C-Sharp

[6] Overview of ASP.NET Core SignalR. Microsoft Learn [cit. 2. 3. 2023] Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-7.0

[7] XAML platform. Microsoft Learn [cit. 2. 3. 2023] Dostupné z: https://learn.microsoft.com/en-us/windows/uwp/xaml-platform/

[8] Hololens 2. Microsoft Product Page [cit. 2. 3. 2023] Dostupné z: https://www.microsoft.com/en-us/hololens/

[9] Why HTTP is not suitable for IOT applications. Concurency [cit. 2. 3. 2023] Dostupné z: https://www.concurrency.com/blog/june-2019/why-http-is-not-suitable-for-iot-applications

[10] GODIRECT PY – Vernier Go Direct® Sensor Library [cit. 2. 3. 2023] Dostupné z: https://github.com/VernierST/godirect-py

[11] GODIRECT JS – Vernier Go Direct® Sensor Library [cit. 2. 3. 2023] Dostupné z: https://github.com/VernierST/godirect-js

[12] Raspberry Pi – Raspberry Pi Foundation [cit. 2. 3. 2023] Dostupné z: https://www.raspberrypi.org/

6 SEZNAM OBRÁZKŮ, DIAGRAMŮ, TABULEK A ZDROJOVÉHO KÓDU

7 PŘÍLOHA 1: ZDROJOVÝ KÓD

Soubor - source.zip