Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

DIPLOMOVÁ PRÁCE

Jednočipové počítače ve výuce

Single-board Computers in Education

Bc. Emil Miler

Vedoucí diplomové práce:	PhDr. Josef Procházka, Ph.D.
Studijní program:	Učitelství informačních a komunikačních technologií pro 2. stupeň základní školy a střední školy
Studijní obor:	Informační a komunikační technologie

Praha 2024

Odevzdáním této diplomové práce na téma Jednočipové počítače ve výuce potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha, 2. prosince 2024

Rád bych vyjádřil svou vděčnost všem, kteří přispěli k úspěšnému dokončení této diplomové práce.

Především děkuji svému vedoucímu, PhDr. Josefu Procházkovi, Ph.D., za jeho podporu, cenné rady během dlouhého procesu tvorby této práce. Velmi si vážím jeho vstřícnosti, trpělivosti a pochopení.

Velké poděkování patří také mému kamarádovi Tomáši Macháčkovi, který mi poskytl cenné rady při návrhu a výrobě vývojové desky a při práci se softwarem KiCad. Jeho odborná pomoc byla pro tento projekt neocenitelná.

Děkuji všem, kteří se podíleli na testování prototypů vývojové desky a poskytli zpětnou vazbu, zejména mým spolužákům a kolegům ze studentského spolku *microlab*.

Na závěr bych rád poděkoval partnerce a všem svým přátelům za jejich podporu a povzbuzení během celé tvorby této práce. Vaše pomoc a porozumění pro mě mnoho znamenaly.

Abstrakt

Diplomová práce se věnuje výuce programování s využitím jednočipových počítačů na středních školách a vyšších stupních gymnázia. Analyzuje dostupné didaktické materiály a s nimi související zařízení, která mohou být zapojena do výuky. Součástí práce je vytvoření taxonomie požadavků, návrh vlastní vývojové desky a příprava didaktických materiálů. V závěrečné části je navržený produkt otestován v praktické výuce, přičemž jsou vyhodnoceny výsledky a navrženy úpravy pro budoucí iterace vývojové desky a doprovodných materiálů.

Klíčová slova

Jednočipové počítače, výuka, programování, embdedded, vývojové desky, mikrokontroléry, Raspberry Pi, Arduino, micro:bit, ESP32, vzdělávací nástroje, algoritmizace, informatické myšlení, didaktické materiály, open-source hardware, IoT (Internet of Things), STEM, praktická výuka, senzory, GPIO, RVP G, programování v informatice, digitální technologie.

Abstract

The thesis focuses on teaching programming using microcontroller-based systems at secondary schools. It analyzes available didactic materials and related devices that can be integrated into the teaching process. The work includes the creation of a taxonomy of requirements, the design of a custom development board, and the preparation of accompanying educational materials. In the final part, the device and didactic materials are tested in practical teaching sessions, with the results evaluated and recommendations made for future iterations of the development board and its supplementary materials.

Keywords

Microcontrollers, education, programming, embedded systems, development boards, single-board computers, Raspberry Pi, Arduino, micro:bit, ESP32, educational tools, algorithm design, computational thinking, teaching materials, open-source hardware, IoT (Internet of Things), STEM, hands-on learning, sensors, GPIO, national curriculum (RVP G), programming in computer science, digital technologies.

Obsah

Úvod

Cílem této diplomové práce je navrhnout a vytvořit efektivní nástroje a související didaktické materiály pro výuku programování jednočipových počítačů.

Práce je rozdělena do několika částí, které se postupně věnují analýze současného stavu, návrhu a implementaci zařízení a jeho ověření v praxi.

První část se zaměřuje na analýzu rámcového vzdělávacího programu (RVP), dostupných didaktických materiálů a technologií, které jsou běžně využívány ve školní praxi. Identifikuje možnosti a omezení těchto nástrojů a definuje požadavky na nový systém, který by byl vhodný pro efektivní výuku programování jednočipových počítačů.

Druhá část práce zahrnuje návrh a tvorbu referenční implementace vývojové desky a souvisejících modulů. Tyto prvky společně tvoří základ pro praktickou výuku. Kapitola také popisuje proces výroby zařízení a vývoj softwaru pro jeho základní ovládání. Nedílnou součástí této části je tvorba didaktických materiálů, které jsou přímo navázány na navržené zařízení.

Závěrečná část práce se věnuje ověření navrženého systému v praxi. Ověření slouží ke zhodnocení využitelnosti zařízení a doprovodných materiálů. Na základě zpětné vazby jsou formulovány návrhy na vylepšení a rozšíření budoucích verzí vývojové desky, softwaru i podpůrných materiálů.

Tato práce tak kombinuje teoretické poznatky, praktický vývoj a zpětnovazební proces, čímž usiluje o inovaci a zkvalitnění výuky programování jednočipových počítačů na středních školách a gymnáziích.

Kapitola 1

Analýza RVP

Cílem této kapitoly je analyzovat rámcový vzdělávací program pro gymnázia (RVP G), identifikovat jeho přístup k výuce programování a navrhnout možnosti integrace navrhovaného výukového nástroje. Kapitola se také zaměřuje na kompetence, které tento nástroj může pomoci rozvíjet.

RVP G existuje v několika revizích. Mezi nejvýznamnější patří revize z roku 2007 (Balada, 2007), která byla dlouhodobě využívána, a novější revize z roku 2021 (Balada, 2021), účinná od 1. září 2022. Tento dokument strukturuje vzdělávací proces vymezením obecných klíčových kompetencí, popisu jednotlivých vzdělávacích oblastí a specifického vzdělávacího obsahu pro každou z těchto oblastí.

1.1 Zásadní změny v nové revizi RVP G

Revize RVP z roku 2021 přinesla zásadní změny, zejména zrušení původní vzdělávací oblasti *Informatika a informační a komunikační technologie*, která byla nahrazena novou vzdělávací oblastí *Informatika*. Současně byla mezi klíčové kompetence zařazena kompetence *digitální*.

Tyto změny odrážejí snahu o modernizaci vzdělávacího kurikula s cílem rozvíjet informatické myšlení žáků a prohloubit jejich pochopení principů digitálních technologií. Nová vzdělávací oblast je navíc navržena tak, aby navazovala na obdobné úpravy v RVP pro základní vzdělávání, což zajišťuje plynulost a kontinuitu ve vzdělávacím procesu.

RVP je úzce propojeno s *rámcovým učebním plánem* (MŠMT, 2021), který stanovuje minimální počty hodin pro jednotlivé předměty a jejich rozložení do ročníků. Pro oblast informatiky byl stanoven minimální počet hodin na 4. Tento rozsah je však poměrně omezený a není vhodný pro systematické zařazení povinné výuky programování jednočipových počítačů. Rámcový učební plán nicméně umožňuje využít tzv. „disponibilní časovou dotaci“, která je určena k následujícím účelům:

- realizace průřezových témat,
- zavedení dalších vyučovacích předmětů,
- vytváření profilace škol,
- posílení časové dotace jednotlivých vzdělávacích oblastí (oborů).

Disponibilní časovou dotaci může škola využít plně podle svého uvážení, přičemž o jejím konkrétním využití rozhoduje ředitel školy. Tato časová dotace umožňuje integraci navrhovaného výukového nástroje přímo do povinné výuky.

Alternativním řešením je nahrazení dosavadní výuky s jinými nástroji, jako jsou například BBC micro:bit nebo Arduino. Detailnímu srovnání a popisu těchto nástrojů je věnována kapitola 4.

1.2 Klíčové kompetence

Obecné klíčové kompetence jsou vymezeny pro různé oblasti, jako například kompetence k učení, komunikativní a také digitální. Tyto digitální kompetence jsou popsány následovně [Žák]:

- Ovládá potřebnou sadu digitálních zařízení, aplikací a služeb, využívá je při školní práci i při zapojení do veřejného života; digitální technologie a způsob jejich použití nastavuje a mění podle toho, jak se vyvíjejí dostupné možnosti a jak se mění jeho vlastní potřeby.
- Získává, posuzuje, spravuje, sdílí a sděluje data, informace a digitální obsah v různých formátech; k tomu volí efektivní postupy, strategie a způsoby, které odpovídají konkrétní situaci a účelu.
- Vytváří, vylepšuje a propojuje digitální obsah v různých formátech; vyjadřuje se za pomoci digitálních prostředků;
- Navrhuje prostřednictvím digitálních technologií taková řešení, která mu pomohou vylepšit postupy či technologie; dokáže poradit s technickými problémy;
- Vyrovnává se s proměnlivostí digitálních technologií a posuzuje, jak vývoj technologií ovlivňuje různé aspekty života jedince a společnosti a životní prostředí, zvažuje rizika a přínosy;
- Předchází situacím ohrožujícím bezpečnost zařízení i dat, situacím ohrožujícím jeho tělesné a duševní zdraví; při spolupráci, komunikaci a sdílení informací v digitálním prostředí jedná eticky, s ohleduplností a respektem k druhým.

Je patrné, že zařazením digitálních kompetencí mezi klíčové považuje RVP tuto oblast za důležitou pro každodenní život bez ohledu na zaměření. Není zde však explicitně zmíněno programování, avšak lze jej zařadit pod některou z těchto obecně formulovaných kompetencí.

1.3 Vzdělávací oblast Informatika

Zaměříme-li se však přímo na vzdělávací oblast *Informatika*, téma algoritmizace se začne objevovat více specificky, například v popisu charakteristiky vzdělávací oblasti (Balada, 2021, s. 62):

> „Studium informatiky zpřístupňuje žákům pojmy, nástroje a metody informatiky jako oboru, který se věnuje efektivnímu, tedy zejména automatizovanému zpracování infor-

mací. Tím, že žáci dokážou prostřednictvím informatických nástrojů zautomatizovat rutinní a opakující se činnosti, získají čas pro jiné činnosti či úkoly.“

Není zde sice uveden pojem „programování“, ovšem z kontextu jasně vyplývá, že výroky „automatizované zpracování informací“ a „zautomatizování rutinní a opakující se činnosti“ jasně naznačují tvorbu algoritmů, ne-li samotného programu.

Vzdělávací obsah oblasti Informatika přesněji definuje, jakým tématům se má výuka věnovat. je členěn na čtyři části, jimiž jsou:

- Data, informace a modelování
- **Algoritmizace a programování**
- Informační systémy
- Digitální technologie

Zahrnuje tedy samostatnou část věnovanou Algoritmizaci a programování, což jasně zdůrazňuje význam těchto dovedností.

1.3.1 Výuka programování v oblasti Informatiky

Programování podle RVP nemá sloužit pouze k osvojení teoretických znalostí, ale především k jejich praktickému využití. Žáci mají prostřednictvím programování získat dovednosti, které jim umožní řešit problémy a aplikovat své schopnosti na reálných příkladech. Tento přístup odpovídá moderním vzdělávacím trendům, jež kladou důraz na interaktivní učení a řešení reálných úloh. Tyto metody zároveň posilují klíčové kompetence žáků a připravují je na budoucí profesní výzvy (Pollard, 2024).

V oblasti algoritmizace a programování se očekává, že žáci budou schopni:

- Vysvětlit algoritmy a programy.
- Analyzovat problémy a navrhovat algoritmy pro jejich řešení.
- Analyzovat a porovnávat různé algoritmy podle jejich efektivity.
- Vytvářet přehledné programy, které zahrnují opakování, větvení, proměnné, seznamy, funkce atp.
- Testovat vytvořené programy a odstraňovat chyby.

Tento přístup k programování je zaměřen na systematické řešení problémů s důrazem na efektivitu a správnost algoritmu, i samotného kódu programu. Žáci se učí nejen základní programovací koncepty, ale také si osvojují jejich aplikaci na konkrétní úlohy.

1.3.2 Využití hardwarových nástrojů ve výuce

Co se týče využívání nástrojů a didaktických pomůcek, konkrétní zmínky o robotických hračkách, jednočipových počítačích nebo jiných specifických technologiích nejsou v nové revizi RVP uvedeny.

Je však pravděpodobné, že se tyto nástroje mohou objevit v rámci praktických cvičení a projektů zaměřených na informatiku a programování. Robotické hračky či jednočipové počítače mohou být využity při realizaci konkrétních projektů, v nichž žáci aplikují algoritmy, kódování a testování v reálných podmínkách. Tento přístup je v souladu s jedním z uvedených cílů vzdělávací oblasti (Balada, 2021):

> „Již od počátku formálního vzdělávání je v informatice kladen důraz na aktivní přístup žáků k řešení praktických problémů. Postupně roste jejich obtížnost, rozsah a složitost. Žáci se setkávají s čím dál větším množstvím úloh s nejasným zadáním, více možnostmi postupů řešení a otevřeným koncem."

Tento přístup dále posiluje zaměření na rozvoj schopností nacházet a ověřovat různá řešení:

> „Nacházení různých řešení, ověřování řešení na modelech či simulacích, porovnávání nalezených řešení z různých, i protichůdných hledisek a k výběru optimálního."

Z uvedeného je zřejmé, že nová revize RVP se nesnaží výuku směřovat k využití konkrétních nástrojů. Zdůrazňuje však řešení reálných problémů, které mohou být realizovány jako funkční výstupy:

> „Alespoň některé z jejich navržených a realizovaných řešení je funkční a řeší relevantní problém."

Lze tedy konstatovat, že nové RVP pro gymnázia klade důraz na programování v širším, obecném kontextu. Explicitní zmínky o specifických nástrojích, jako jsou robotické hračky nebo jednočipové počítače, zde chybí. Nicméně je zřejmé, že při výuce programování je možné využívat také jednočipové počítače a další didaktické pomůcky, které umožňují propojení teoretických znalostí s praktickými dovednostmi.

Tato metodika studentům poskytuje příležitost aplikovat algoritmické postupy v reálných situacích a zároveň porozumět principům fungování digitálních technologií prostřednictvím konkrétních zařízení. Tato zařízení mohou okamžitě reagovat světelnými signály, zvuky nebo pohybem, což významně obohacuje proces výuky a zvyšuje motivaci k učení (Tocháček, 2015).

1.4 Shrnutí

Nová revize RVP G klade důraz na rozvoj informatického myšlení žáků, což představuje významný krok směrem k modernizaci vzdělávacího obsahu. Přestože je informatika jako vzdělávací oblast pevně zakotvena, časová dotace pro její výuku je omezena. Z tohoto důvodu je navrhovaný výukový nástroj, viz kapitola 5, primárně zaměřen na využití v nepovinných aktivitách, například v rámci kroužků.

Navrhovaný nástroj lze rovněž využít jako alternativu k běžně používaným platformám, jako jsou BBC micro:bit nebo Arduino, a tím obohatit nebo zcela nahradit současnou podobu výuky programování. Další možností je začlenění tohoto nástroje do povinné výuky pomocí disponibilní časové dotace, kterou mohou školy flexibilně využívat podle svých specifických potřeb.

Nástroj nabízí široké uplatnění zejména v oblasti výuky algoritmizace a programování, kde přispívá k propojení teoretických znalostí s praktickými dovednostmi. Díky svému zaměření na reálné projekty a řešení praktických problémů podporuje komplexní rozvoj schopností žáků v oblasti informatiky.

Kapitola 2

Analýza didaktických materiálů

Tato kapitola poskytuje výběr dostupných didaktických materiálů vhodných pro výuku na gymnáziích, v kroužcích a podobných vzdělávacích prostředích v rámci výuky programování s využitím jednočipových počítačů. Z těchto učebnic jsou dále odvozeny specifické jednočipové počítače, tedy vývojové desky. Jejich vlastnostem a hodnocení v kontextu jejich využití ve výuce je dále věnována pozornost v kapitole 4.

S ohledem na specifické zaměření na vyšší stupeň gymnázia se tato práce nezabývá oblastí robotických hraček, jako jsou například *Ozobot* nebo *Sphero*. Tyto robotické hračky přinášejí do výuky nejen možnost rozvoje algoritmického myšlení, ale také zvýšení motivace žáků k učení, rozvoj kreativity a samostatného myšlení (Vaňková, 2019). Přestože tyto nástroje mají své nezastupitelné místo zejména na základních školách, patří mimo rozsah zaměření této práce, která se věnuje pokročilejším nástrojům a technologiím vhodným pro starší studenty a náročnější projekty.

2.1 Dostupné didaktické materiály

V této sekci jsou shrnuty různé didaktické materiály, které učitel může zapojit do své výuky. Výběr materiálů byl sestaven na základě konzultací s kolegy, kteří se věnují výuce programování jednočipových počítačů, a analýzy běžně dostupných tištěných učebnic, ale také materiálů v online podobě, které lze snadno nalézt při rychlém vyhledávání.

Významným zdrojem jsou učebnice a vzdělávací materiály pro školy z projektu iMyšlení, které se zaměřují na programování s využitím různých nástrojů na různých úrovních vzdělávání. Tyto materiály byly rozsáhle ověřeny v praxi, a tudíž představují spolehlivý zdroj, podle kterého mohou učitelé modelovat svou výuku. (iMyšlení, 2018)

První z nich nese název *Robotika pro základní školy: programujeme micro:bit pomocí Makecode* (Pech; Pršala et al., 2021). Jak již napovídá samotný název, tato učebnice je určena pro základní školy a zaměřuje se na výuku vizuálního blokového programování. Studentům nabízí přístupný a intuitivní způsob, jak si osvojit základy programování prostřednictvím vizuálních bloků, což značně usnadňuje pochopení principů imperativního programování. Svým obsahem však není vhodná pro výuku na

vyšším stupni gymnázia či na středních školách, a proto slouží spíše jako úvod do programování a nástroj pro rozvoj algoritmického myšlení.

Druhá učebnice s názvem *Robotika pro střední školy: programujeme micro:bit pomocí Pythonu* (Pech; Novák, 2020) se zaměřuje na imperativní programování v jazyce MicroPython, který se více blíží cílům této práce, avšak výsledný kód lze stále použít pouze pro platformu micro:bit. Nabízí také vlastní knihovny, které nejsou univerzální mezi dalšími platformami.

Třetí z učebnic, *Robotika pro střední školy: programujeme Arduino* (Novák et al., 2020), se zaměřuje na populární otevřenou platformu Arduino, která je široce využívána po celém světě pro tvorbu elektronických zařízení a systémů (Arduino, 2018). Podrobnější informace o platformě Arduino jsou uvedeny v sekci 4.1. Učebnice se věnuje základní práci s platformou, konkrétně s deskou Arduino UNO, a seznamuje čtenáře se základy jejího programovacího jazyka. Kromě teoretického základu obsahuje řadu projektů, které lze přímo začlenit do výuky. Velká část obsahu je navíc věnována základům elektrotechniky, jako je propojování jednotlivých součástek, což je nedílnou součástí práce s touto platformou.

Podobným materiálem k učebnici projektu iMyšlení je učebnice s názvem *ARDUINO PROJECTS BOOK* (Fitzgerald et al., 2013), která byla distribuována jako součást *Arduino Starter Kit* a byla dříve využívána v rámci výuky na Gymnáziu Jana Keplera. Tato učebnice obsahuje stručné a přehledné popisy elektronických součástek a jejich zapojení. Kromě toho poskytuje příklady projektů uspořádaných podle obtížnosti a pokročilosti. Každý z těchto projektů má detailně popsáno zapojení a příklad kódu, který je rozdělen do jednotlivých částí, přičemž každá část je jasně popsána v tom, co konkrétně dělá.

Společnost Arduino, která se specializuje na návrh a výrobu této platformy, nabízí vlastní sadu edukačních materiálů rozdělených do tematických bloků (Arduino, 2024a). Ty zahrnují jak základní práci s vývojovou deskou, tak i pokročilejší aplikace, například obecnou robotiku, tvorbu automatizovaného skleníku nebo práci se stavebnicí *Braccio++*, tedy robotickou paží. Všechny tyto materiály propojují programování s integrací elektronických součástek a konstrukcí větších mechanických systémů.

Jedna z dostupných učebnic, *Mastering the Arduino Uno R4* (Ibrahim, 2023), se zaměřuje na práci s vývojovou deskou *Arduino UNO R4*, jejíž podrobnější popis je uveden v sekci 4.1. Učebnice obsahuje řadu příkladů ilustrujících různé možnosti využití této desky a může posloužit jako hodnotný zdroj úkolů i inspirace pro učitele.

Česká firma Hardwario, dříve známá jako BigClown, vznikla oddělením od známé firmy Jablotron a věnuje se tvorbě vlastního hardwaru. V letech 2017 až 2019 se firma intenzivně věnovala tvorbě stavebnic určených pro výuku na školách. Stavebnice byla založena na principu připravených modulů, které bylo možné jednoduše kombinovat bez nutnosti zapojování kabelů a dalších součástek (Sedlák, 2017). Tato stavebnice byla společně s vlastními didaktickými materiály nabízena školám, včetně katedry informačních technologií pedagogické fakulty, kde byla využita ve výuce, které se účastnil

i autor této práce. Didaktické materiály byly mnohokrát revidovány až do jejich finální podoby pro STEM výuku (HARDWARIO, 2021).

Zdá se však, že společnost Hardwario se nyní zaměřuje spíše na industriální a profesionální technologická řešení než na edukační materiály. Tento posun je patrný ze stagnujících úprav didaktických materiálů, nefunkčních hypertextových odkazů a neaktivního vývoje nástrojů pro práci se stavebnicí TOWER. Tato skutečnost naznačuje, že společnost pravděpodobně upustila od aktivní podpory vzdělávacích projektů ve prospěch vývoje technologií určených pro průmyslové aplikace.

Další často používanou učebnicí je *Hradla, volty, jednočipy: Úvod do bastlení* (Malý, 2017), která se zaměřuje na základy elektroniky. Začíná vysvětlením pojmů jako napětí, proud a odpor, včetně Ohmova zákona, a dále se zabývá elektronickými součástkami, digitálními obvody řady TTL (74xx) a pokročilejšími prvky, jako jsou senzory, paměti či různé jednočipové počítače. Nepředpokládá rozsáhlé technické znalosti, vyžaduje pouze základní orientaci v práci s počítačem, a znalost programování není nutná. Učebnice není založena na konkrétní vývojové desce či jednotné pomůcce, což ji činí univerzální, avšak svým zaměřením se hodí spíše pro výuku na středních odborných školách.

Podobná je také relativně nová učebnice *ESP32 prakticky* (Malý, 2024), která se věnuje vývojovým deskám založeným na platformě ESP32 od společnosti Espressif. Pokrývá témata od základního nastavení a programování v Arduino IDE, přes připojení k Wi-Fi a Bluetooth, až po pokročilé techniky, jako je použití operačního systému FreeRTOS, příjem satelitních signálů nebo programování v ESP-IDF. Obsahuje praktické návody a příklady, které z ní tvoří užitečnou příručku jak pro začátečníky, tak pro pokročilé vývojáře.

V praxi se lze setkat také s různými edicemi učebnice *The Official Raspberry Pi Beginners Guide* (Halfacree, 2024), přičemž nejnovější je pátá edice. Tato učebnice se zaměřuje na práci s populárním mikropočítačem Raspberry Pi, včetně základního zapojení, instalace operačního systému, připojení dalších hardwarových komponent a také samotnému programování. Programovací část je pokryta jak vizuálním jazykem Scratch, tak i imperativním jazykem Python, což umožňuje přizpůsobit obsah uživatelům s různou úrovní zkušeností. Díky tomu je učebnice využitelná ve výuce na různých úrovních vzdělávání a je přizpůsobitelná schopnostem a potřebám žáků.

V návaznosti na didaktické materiály pro Raspberry Pi lze zmínit i podobnou učebnici s názvem *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux* (Molloy, 2019), která se věnuje obdobné problematice, ale s využitím vývojové desky *BeagleBoard*. Tato deska představuje hybridní řešení mezi jednoduchým Arduinem a komplexním Raspberry Pi. Učebnice pokrývá základy práce s touto deskou a programování v prostředí Embedded Linux.

2.2 Shrnutí dostupných technologií

Z výše zmíněných didaktických materiálů lze tedy vyvodit následující technické pomůcky, tedy stavebnice či vývojové desky, které nacházejí uplatnění při výuce programování jednočipových počítačů na úrovni vyššího gymnázia:

- BBC micro:bit
- Arduino
- ESP32
- Raspberry Pi
- BeagleBone
- Hardwario TOWER

Každá z těchto technologií nabízí specifické vlastnosti a možnosti využití, což z nich činí užitečné nástroje pro rozvoj dovedností v oblasti elektroniky, programování a informatického myšlení. Podrobné analýze a vyhodnocení jednotlivých nástrojů se věnuje kapitola 4, která zohledňuje jejich technické parametry, cenovou dostupnost a další kritéria, která jsou definována v kapitole 3.

Kapitola 3

Stanovení hodnotících kritérií

Proces výběru konkrétních zařízení je realizován vzhledem k definovaným kritériím, jež jsou zásadní pro zajištění spolehlivého nasazení těchto prostředků do výuky. Tato kritéria byla systematicky stanovena autorem této práce po konzultaci s pedagogy, kteří se specializují na výuku předmětů souvisejících s programováním s pomocí takovýchto zařízení na gymnáziích a středních školách.

Zajímavé shrnutí poskytla kolegyně Hana Šandová, která se dlouhodobě zabývá tématem výuky programování a prací s uvedenými nástroji. Ve své e-mailové komunikaci (Šandová, 2024) vymezila důležitá kritéria, která jsou pro ni rozhodující při výběru a implementaci vzdělávacích pomůcek do výuky. Tato kritéria zahrnují:

- Cena
- Cílová skupina
- Uživatelská přívětivost, dostupnost materiálů atp.
- Kompatibilita a přesah/rozšiřitelnost

Na základě tohoto shrnutí byla definována výsledná kritéria, která zároveň reflektují technické požadavky na univerzální a moderní přístup k práci s vybranými zařízeními.

Cílová skupina

Zásadním faktorem je volba vhodného zařízení v závislosti na cílové skupině, ve které má být zařízení využito. Je nezbytné zohlednit jak náročnost provozu, tak i uživatelskou přívětivost softwarového vybavení. Pro mladší žáky bude samozřejmě vhodnější použití vizuálního blokového programování, zatímco pro starší žáky na střední škole a vyšším gymnáziu je vhodné zvolit klasický imperativní, objektový nebo jiný typ programování skrze textový programovací jazyk.

Pořizovací cena

V kontextu, kdy je výuka programování fyzických zařízení zařazena spíše jako volitelný seminář než povinný předmět pro všechny žáky, hraje otázka finanční dostupnosti důležitou roli při rozhodování o výběru technologických prostředků a jejich zapojení do výuky. Cenová náročnost zařízení totiž přímo ovlivňuje, zda bude možné konkrétní pomůcku vůbec do výuky zařadit.

Hana Šandová ve svém e-mailu zdůrazňuje význam pořizovací ceny a popisuje obtíže spojené s financováním těchto pomůcek. Uvádí, že právě pořizovací cena je zásadní při rozhodování o vhodnosti konkrétní technologie pro výuku:

> „Cena je pro mě osobně důležitý faktor ... Často si nejprve koupím robota (např. Ozobot, mBot, BBC micro:bit) pro vlastní potřebu a když mi dává smysl usilovat o zařazení do výuky, tak se snažím i pomoci hledat cestu k financování (např. granty Meet&Code nebo od zřizovatele na vzdělávání).“

Zároveň kritizuje neefektivní využívání dotací, které vede ke zbytečnému pořizování drahých zařízení bez jasného účelu, což podle ní často končí jejich nevyužitím:

> „Evropské peníze, které nutí školy plošně nakupovat drahé věci aniž by věděli k čemu a pak se na ně ve školách práší, se mi nelíbí, ale to je jiná kapitola.“

Tyto názory upozorňují na důležitost modularity a všestrannosti zařízení, což souvisí s kritérii výběru technologie, která jsou podrobněji rozpracována v následujících částech. Modularita umožňuje přizpůsobení zařízení konkrétním potřebám výuky, zatímco široká škála využití přináší lepší návratnost investice a zabraňuje tomu, aby technologie zůstala nevyužita. Je tedy důležité, aby zvolený nástroj byl cenově dostupný.

Dostupnost dokumentace

Jedním z kritérií při výběru zařízení je dostupnost kvalitní dokumentace a literatury, které jsou nezbytné pro efektivní práci s daným zařízením. Pokud je dokumentace nedostatečná nebo obtížně přístupná, představuje to výraznou překážku při zavádění nástroje do výuky, protože učitel je v takovém případě nucen vytvářet dokumentaci a didaktické materiály vlastními silami. Tato problematika úzce souvisí také s dostupností softwarových nástrojů potřebných pro správnou obsluhu zařízení, viz následující kritérium.

Dostupnost softwarového vybavení

Není pochyb o tom, že dostupnost softwarové podpory je zásadní pro zajištění efektivity a dlouhodobého využívání zařízení. Významnou roli mají softwarové knihovny, které jsou nezbytné pro správné fungování a plné využití možností daného zařízení. Tato problematika však přesahuje pouhé zajištění základní funkčnosti; souvisí také s bezpečností a stabilitou zařízení, což jsou důležité faktory pro hladký průběh výuky.

Kvalitní softwarová podpora zahrnuje pravidelnou aktualizaci a údržbu knihoven, které odstraňují případné chyby a zajišťují stabilní provoz. Dalším důležitým aspektem je přístup ke zdrojovému kódu. Otevřený zdrojový kód umožňuje nejen oficiálním vývojářům, ale i širší komunitě, aby platformu přizpůsobovali, vylepšovali a rozšiřovali její možnosti. Tento přístup podporuje rychlejší odhalování a řešení chyb, inovace a širší přijetí dané technologie. Komunitní spolupráce na otevřeném kódu také napomáhá vytvoření prosperujícího ekosystému a přispívá k udržitelnému rozvoji softwarové platformy. (Park, 2022)

Podpora PlatformIO

Toto kritérium souvisí s výše zmíněným, a to softwarová podpora PlatformIO. Představuje zásadní prvek pro efektivní práci s daným zařízením, což je úzce spojeno s dostupností softwarového vybavení. PlatformIO je integrované vývojové prostředí a platforma pro správu projektů, která se zaměřuje na usnadnění vývoje.

Integrace s PlatformIO nabízí jednotné prostředí pro správu kódu, kompilaci, nahrávání firmware a monitorování vývoje projektů. Tato podpora významně zlepšuje efektivitu vývojového procesu a přispívá k lepší organizaci práce. PlatformIO navíc umožňuje rychlý přístup k široké škále softwarových knihoven, což urychluje vývoj a sjednocuje pracovní postupy napříč různými operačními systémy (PlatformIO, 2023).

Jednou z největších výhod je nahrazení rozhraní Arduino IDE intuitivnějším prostředím PlatformIO IDE, které poskytuje přehledné nástroje zjednodušující celý proces od psaní kódu po jeho nasazení.

Podpora PlatformIO navíc přispívá k větší stabilitě a bezpečnosti projektů, viz předchozí kritérium. Umožňuje snadnou údržbu a aktualizaci knihoven, což zajišťuje, že projekty mohou pružně reagovat na nové technologické požadavky. Díky podpoře široké škály mikrokontrolérů a platforem poskytuje PlatformIO značnou flexibilitu a rozšiřitelnost.

Shrnuto, podpora PlatformIO zjednodušuje technické aspekty práce s mikrokontroléry a vestavěnými systémy, což nejen zvyšuje efektivitu vývoje, ale také usnadňuje výuku programování. Díky tomu se stává důležitým nástrojem v oblasti vzdělávání.

Rozšiřitelnost

Rozšiřitelností se zde rozumí modularita zařízení, která umožňuje jeho přizpůsobení a použití v různých kontextech. Některá zařízení se snaží zjednodušit použití tím, že nabízejí pouze omezený počet senzorů a výstupů. Tento přístup však může mít negativní důsledky, protože taková zařízení často omezují možnosti aplikace pouze na jednoduché úkoly, které jsou uzpůsobeny spíše schopnostem zařízení než schopnostem žáka. To může vést k omezení flexibility a rozmanitosti úloh.

Navíc některá zařízení nenabízejí adekvátní možnosti pro rozšíření, jako jsou například GPIO (General Purpose Input/Output) piny, které umožňují připojení dalších senzorů, modulů a periferií. Bez těchto možností rozšíření se snižuje potenciál zařízení pro komplexnější projekty a pokročilé úlohy. Studenti jsou pak omezeni při experimentování a rozvoji technických dovedností, což brání hlubšímu porozumění a aplikaci technologií.

Ideální zařízení umožňují snadnou integraci dalších komponent, poskytují základnu pro výuku začátečníků a zároveň nabízejí možnost rozšíření pro pokročilé projekty. Tímto způsobem podporují dlouhodobý rozvoj dovedností studentů, čímž se stávají univerzálním nástrojem pro různé úrovně výuky a rozvoje. Modularita a rozšiřitelnost zařízení jsou tedy důležitými vlastnostmi, které umožňují přizpůsobit výukové prostředky specifickým potřebám a cílům vzdělávacího procesu.

Přínos pro výuku programování

Jedním z hlavních cílů zavedení hardwarových technických pomůcek do výuky je zlepšení a rozšíření dovedností žáků v oblasti programování a technického myšlení. Kvalitně zvolené zařízení by mělo podporovat praktické učení, kde žáci mohou aplikovat teoretické znalosti v reálných situacích. Důležitým přínosem takových zařízení je možnost okamžité zpětné vazby, kdy žáci vidí výsledky svého programování v reálném čase, například blikáním světelných diod, nebo zvukem. To může posílit jejich motivaci a zapojení do výuky. (Vaňková, 2019)

Je však důležité zohlednit i praktické aspekty práce s těmito zařízeními. Některá zařízení, jako například Arduino, vyžadují zapojování kabelů a dalších komponent, což může být časově náročné a složité pro začátečníky. Tento proces může zabrat značnou část vyučovací hodiny, čímž omezuje čas, který lze věnovat samotnému programování. Při výuce zaměřené na programování a nikoli na elektrotechniku je tedy třeba zvážit výběr zařízení, která minimalizují potřebu složitých hardwarových úprav a umožní studentům soustředit se více na programování a logiku.

Kapitola 4

Hodnocení technologií a nástrojů

V návaznosti na kritéria stanovená v kapitole 3 a analýzu dostupných didaktických materiálů v sekci 2.1 jsou v této kapitole vyhodnoceny vybrané hardwarové platformy. Kapitola je zakončena celkovým shrnutím poznatků a úvah o vhodnosti těchto nástrojů pro účely výuky programování.

4.1 Hodnocení jednotlivých nástrojů

BBC micro:bit

BBC micro:bit je široce využívanou platformou ve vzdělávacím prostředí, zejména na úrovni základních škol, pro svou schopnost efektivně podporovat výuku. Tato platforma disponuje rozsáhlým spektrem senzorů, LED a dalších výstupů, což umožňuje vytváření komplexních vzdělávacích úloh, které aktivně zapojují studenty do procesu učení. O procesu zapojení do výuky pojednává například článek „BBC micro:bit ve škole“ (Havířová, 2019). Hlavní výhody tohoto nástroje popisuje autorka článku následovně:

> „Velkou výhodou je kompaktnost celého zařízení. Díky displeji, rádiové komunikaci a mnoha senzorům si dlouho vystačíte se samotnou destičkou, u které se nemusíte starat o správné zapojení vodičů a součástek. Micro:bit ale není omezen jen na vestavěné senzory, pomocí rozšiřujících pinů je možné připojovat další moduly a součástky, takže se otevírá obrovský prostor pro projekty.“

Zajímavé učebnice pro využití BBC micro:bit ve výuce poskytuje dříve zmíněný portál *Informatické myšlení*, viz sekce 2.1

Zařízení micro:bit je vybaveno širokou škálou senzorů, LED maticí a dalšími prvky, což z něj činí atraktivní nástroj pro výuku základů programování. Nicméně, jeho využití je omezeno na interpretované jazyky a vlastní knihovny. To představuje určitou nevýhodu, jelikož takové prostředí a nástroje neumožňují vytváření univerzálního kódu, který by mohl být aplikován v reálných projektech mimo specifické prostředí micro:bitu. V důsledku toho se studenti naučí pouze základní koncepty programo-

vání, které jsou specifické právě pro micro:bit, což omezuje jejich schopnost přenést tyto dovednosti na jiné platformy a do praxe při řešení reálných problémů či projektů. Můžeme tedy říci, že nástroj je vhodný spíše pro jednoduché konstrukty a nehodí se pro tvorbu komplexních projektů a výrobků.

Arduino

Arduino je open-source platforma, která kombinuje jednoduchý hardware a software pro práci s elektronikou. Arduino dokáže číst různé vstupy, například ze senzorů světla, teploty, nebo stisk tlačítka a převádět je na výstupy, jako je spuštění motoru, nebo rozsvícení LED.

K programování se využívá jazyk *Arduino* založený na jazyce *Wiring* a vývojové prostředí *Arduino IDE*. Wiring, potažmo jazyk Arduino, je open-source programovací jazyk navržený pro mikrokontroléry, který zjednodušuje proces psaní vestavěných programů. Je postaven na jazyce *C++* a používá se v rámci vytváření softwaru pro ovládání elektronických zařízení. (Ramon, 2014)

Tato platforma našla uplatnění v tisících projektů, od vzdělávacích nástrojů až po vědecké přístroje. Díky své jednoduchosti ji využívají nejen studenti a učitelé, ale také designéři, umělci, hudebníci a amatérští tvůrci, kteří s její pomocí realizují interaktivní prototypy, instalace nebo experimenty s robotikou.

Jednou z hlavních výhod Arduina je jeho dostupnost. Desky jsou cenově přijatelné, software je otevřený a flexibilní a funguje na různých operačních systémech. Prostředí Arduino IDE je snadné na ovládání pro začátečníky, ale zároveň nabízí pokročilé možnosti pro zkušené uživatele. Plány hardware i software jsou zveřejněny jako open-source, což umožňuje rozšiřování funkcí a přizpůsobení podle potřeb. (Arduino, 2018)

Problém při využití Arduina ve výuce programování spočívá v nutnosti fyzického zapojování různých komponent, jako jsou senzory, výstupy a napájení. To vede k tomu, že významná část výuky se musí věnovat praktickým aspektům a základům elektroniky či fyziky, což ubírá čas, který by mohl být využit k samotnému programování.

Částečným řešením je derivátní deska *Arduino UNO R4*, která kromě funkcí původního *Arduino UNO* obsahuje i rozšiřující komponenty, například adresovatelnou LED matici podobnou té u BBC micro:bit, a některé další senzory (Arduino, 2024b). Přestože nabízí určitá vylepšení, pro složitější projekty není dostatečně vybavená a stále vyžaduje rozšiřování prostřednictvím zapojování dalších komponent, což původní problém odstraňuje pouze částečně.

ESP32

ESP32 je mikrokontrolér vyvinutý společností *Espressif Systems*, který nabízí široké spektrum využití díky integrované podpoře Wi-Fi a Bluetooth, přičemž některé varianty podporují i technologii Zigbee. Mikrokontrolér poskytuje obdobné funkce a podporu softwarových platforem jako Arduino, avšak

vyniká vyšším výkonem a rozšířenými možnostmi konektivity. (Espressif Systems, 2023)

Tato hardwarová platforma se jeví jako vhodná pro potřeby výuky, a to zejména díky svému výkonu, flexibilitě a nativní podpoře nástroje PlatformIO. Zásadní nevýhodou však zůstává skutečnost, že obdobně jako u Arduina jsou dostupné vývojové desky navrženy pro manuální zapojování jednotlivých komponent. Příkladem může být oficiální vývojová deska *ESP32-DevKitC V4* (Espressif Systems, 2024b). Tento přístup ve vzdělávacím prostředí znamená, že značná část hodiny musí být věnována právě manuálnímu zapojování komponent, což ubírá čas samotnému programování.

Raspberry Pi

Rodina počítačů Raspberry Pi, podobně jako Arduino a ESP, umožňuje připojení externích senzorů a modulů prostřednictvím GPIO pinů a svou funkčnost rozšiřuje díky široké škále dostupných modulů. Na rozdíl od těchto zařízení však Raspberry Pi představuje plnohodnotný počítač s operačním systémem, což jej přibližuje klasickému programování na školních počítačích, s možností využití různých jazyků a paradigmat (Butts, 2021). Alternativy jako Odroid nebo Banana Pi mohou nabídnout vyšší výkon nebo nižší cenu, avšak i tyto mikropočítače sdílejí podobné vlastnosti (Bauduin, 2023).

Navzdory své všestrannosti není Raspberry Pi ideální volbou pro samostatnou výuku programování. Vyšší pořizovací cena a nutnost pravidelné údržby operačního systému mohou být značnou překážkou. Navíc se programování na této platformě více vzdaluje od přímé práce s hardwarem, která je charakteristická pro vývojové desky, jako je Arduino. Na Raspberry Pi je programování více abstrahované, což může omezovat pocit přímé interakce s hardwarem. Tento rozdíl v přístupu může mít vliv na celkové vnímání a pochopení principů embedded systémů. Z těchto důvodů nelze tento typ zařízení považovat za vhodný pro výuku programování jednočipových počítačů.

BeagleBone

Pozornost si zaslouží také vývojové desky BeagleBone, které patří do stejné kategorie zařízení jako Raspberry Pi a další zmíněné alternativy. BeagleBone se vyznačuje vysokou rozšiřitelností díky dostupným modulům a podpůrným materiálům, a je tedy skvělou platformu pro tvorbu složitějších robotických systémů a automatizací. Vývojáři navíc nabízejí speciální výukové materiály a sady zaměřené na školní prostředí (Petazzoni, 2023), včetně rozšiřující desky *TechLab* určené přímo pro vzdělávací účely (BeagleBone, 2024).

Nicméně, tyto desky závisí na Embedded Linuxu, což může přinášet problémy spojené s větší složitostí systému, stejně jako v případě Raspberry Pi. Aktuálně také nejsou plně podporovány vývojovým prostředím PlatformIO, což dokládá probíhající diskuse na GitHubu (Kravets, 2018). Podobně jako u Raspberry Pi může být práce s BeagleBone vnímána jako příliš abstrahovaná, což ji vzdaluje od přímého přístupu k hardwaru, typického pro jednočipové počítače.

Hardwario TOWER

Český Hardwario TOWER je komplexní platforma pro vývoj a správu chytrých zařízení a aplikací v prostředí IoT. Skládá se z různých hardwarových modulů, jako jsou senzory a komunikační rozhraní, které lze snadno propojit a integrovat do široké škály projektů. Princip této platformy spočívá v tom, že jednotlivé moduly lze jednoduše připojit k hlavní vývojové desce s procesorem, čímž se snadno rozšiřuje funkcionalita celého systému. (Sedlák, 2017)

Zmíněná stavebnice však trpí velmi komplexním softwarovým vybavením a je víceméně nutné vytvářet programy v připraveném prostředí *BigClown Playground*, které nabízelo pouze nástroj pro blokové programování s názvem *Node-RED*, viz článek „BigClown Playground“ od samotných vývojářů stavebnice (BigClown, 2018). Toto softwarové řešení značně omezuje možnosti práce se stavebnicí. Druhou možností je programovat vlastní firmware, avšak tento proces je velmi složitý v porovnání s jinými platformami a svým rozsahem je vhodný spíše pro vysoké školy. Tato skutečnost nenaplňuje cíle a kritéria zvolená v kapitole 3.

Zdá se, že i přes dobře navržený hardware stavebnice nezískala přízeň uživatelů ani škol. Přispěla k tomu i vysoká pořizovací cena stavebnice a jednotlivých modulů, viz komentáře uživatelů u dříve zmíněného článku na portálu Lupa.cz[1] (Sedlák, 2017).

4.2 Shrnutí hodnocení

Navzdory široké nabídce dostupných hardwarových platforem, které se zdají vhodné pro výuku programování jednočipových počítačů na středních školách a vyšších gymnáziích, se žádná z nich nejeví jako zcela ideální. Arduino i univerzálnější ESP32 představují výkonné a flexibilní nástroje, avšak jejich využití je značně omezeno nutností manuálního zapojování součástek. Tento proces často zbytečně zatěžuje výuku a odklání pozornost od samotného programování.

Na druhou stranu, Hardwario TOWER přináší inovativní koncept snadno propojitelných modulů, které minimalizují potřebu manuální práce, ale jeho využití je výrazně limitováno složitostí softwarového prostředí a vysokými pořizovacími náklady. Podobný problém má i BBC micro:bit, který sice nabízí dobře integrované senzory a funkce, avšak jeho univerzálnost a přenositelnost vytvořených programů na jiné platformy jsou velmi omezené.

Raspberry Pi a BeagleBone sice poskytují větší výpočetní výkon a možnosti operačního systému, ale jejich využití je orientováno spíše na obecné programování než na výuku s využitím jednočipových počítačů, a navíc vyžadují značnou údržbu a jsou finančně náročnější.

S ohledem na analyzované nástroje se zdá, že žádné z dostupných řešení neodpovídají kritériím stano-

[1] `https://www.lupa.cz/clanky/liberecky-bigclown-se-odtrhl-od-jablotronu-a-s-hardwarovou-skladackou-jde-do-sveta/`

vených v kapitole 3. Alternativní cestou se proto jeví vytvoření vlastního systému, který by kombinoval přednosti jednotlivých platforem – jednoduchost a univerzálnost Arduina a ESP32, modularitu a snadnou integraci komponent z Hardwario TOWER, a zároveň by zachoval nízkou pořizovací cenu a snadné ovládání vhodné pro vzdělávací prostředí. Takový systém by mohl poskytnout ideální rovnováhu mezi praktičností, výkonem a uživatelskou přívětivostí.

Kapitola 5

Implementace vlastního zařízení

Tato kapitola se zaměřuje na učitele, kteří by se chtěli vydat cestou implementace vlastního zařízení splňujícího výše stanovená kritéria, viz kapitola 3. Vzhledem k tomu, že zařízení není v tuto chvíli komerčně dostupné, je nezbytné nechat jej vyrobit u libovolného výrobce PCB. Je zde rozebrán postup návrhu takového zařízení, včetně výběru typů součástek, výběr vhodného software a samotný návrh tištěných spojů.

První část kapitoly je věnována výběru vhodných nástrojů pro tvorbu všech částí zařízení. Následuje popis návrhu základní vývojové desky, která určuje základní rozměry, a dále jsou uvedeny detaily týkající se rozšiřujících modulů.

Poslední část kapitoly se zaměřuje na přípravu softwarového vybavení potřebného pro práci s navrženou vývojovou deskou a jejími moduly.

V návaznosti kapitolu 3 jsou zde shrnuty základní parametry, dle kterých je zařízení navrhováno a které ovlivňují výběr vhodných komponent. Cílem výsledného produktu je splnit tyto podmínky:

- Vhodné pro cílovou skupinu žáků vyššího gymnázia
- Nízká pořizovací cena
- Přehledná dokumentace
- Podpora běžně dostupných softwarových knihoven
- Podpora PlatformIO
- Široké spektrum využití a snadná rozšiřitelnost
- Pozitivní přínos pro výuku programování

5.1 Výběr nástroje pro návrh modulů

Výběr optimálního nástroje probíhal na základě analýzy dostupného softwaru schopného pracovat s komplexními návrhy plošných spojů. Agregaci takovýchto nástrojů poskytuje článek *Best PCB Design Software*, který jednotlivé programy nejen uvádí, ale také porovnává a hodnotí. (3DSourced, 2023)

Některé z uvedených nástrojů nejsou svou jednoduchostí vhodné pro vytváření složitých návrhů, jako je právě vývojová deska z této diplomové práce. Níže je shrnutí nástrojů, které byly vyhodnoceny jako vhodné pro tvorbu složitého návrhu:

5.1.1 Altium Designer

Altium Designer je špičkový software pro návrh plošných spojů (PCB), který vyniká zejména svými pokročilými funkcemi pro 3D vizualizaci, tvorbu schémat a návrh desek. Je zaměřen na náročné projekty, což se projevuje podporou složitých vícevrstvých desek a designů s flexibilními částmi. Umožňuje také optimalizaci pracovních postupů, čímž usnadňuje celý proces návrhu od počáteční fáze až po výrobu. Díky integraci se simulačními nástroji a knihovnami komponent mohou inženýři snadno spravovat své návrhy a sledovat dostupnost součástek i náklady v reálném čase.

Pokud jde o dostupnost a licencování, Altium Designer je placený software, což jej činí dostupným především pro komerční firmy a zkušené profesionály. Licence je nákladnější, ale zároveň poskytuje přístup k pravidelným aktualizacím a technické podpoře. Tento nákladný model je zaměřen na profesionály, kteří vyžadují špičkovou technologii a pokročilé nástroje pro návrh složitých desek.

Na druhou stranu, software není dostupný pro různé operační systémy, nýbrž jen pro systém Windows, což může omezovat některé uživatele. Uživatelé systémů GNU/Linux nebo macOS musí hledat alternativní řešení nebo využívat virtualizační nástroje. Tento omezený přístup může být nevýhodou pro značnou část uživatelů.

5.1.2 Autodesk Eagle

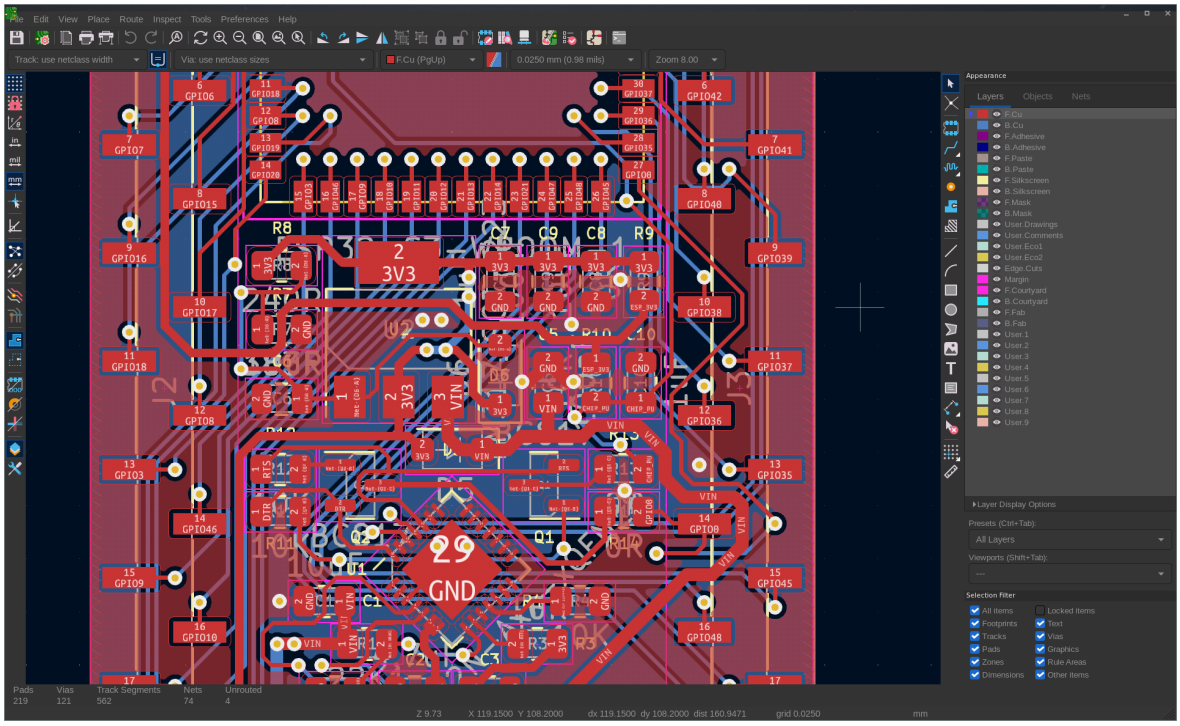
Autodesk Eagle je oblíbený nástroj pro návrh PCB, který kombinuje editaci schémat, tvorbu návrhů desek a přístup k rozsáhlé knihovně komponent v uživatelsky přívětivé platformě. Nabízí robustní funkce, jako jsou modulární návrhové bloky, které umožňují znovu použít části návrhů, a nástroje pro online spolupráci, což je ideální pro týmy pracující na dálku. K programu je také dostupná široká škála dokumentace a edukačních materiálů.

Co se týče licencování a podpory operačních systémů, Eagle je dostupný na více platformách, včetně Windows, macOS a Linuxu, což rozšiřuje jeho dostupnost pro různé typy uživatelů. Program je k dispozici v placené i bezplatné verzi, avšak verze zdarma je omezena v počtu vrstev a velikosti desky, což může limitovat možnosti pokročilých návrhů. Verze zdarma je proto vhodnější pro menší projekty a hobby využití, zatímco komerční licence přináší plný přístup k profesionálním funkcím, jako je například práce s vícevrstvými deskami, pokročilé knihovny a podpora cloudového ukládání.

5.1.3 KiCad

KiCad nabízí širokou škálu funkcí pro profesionální i amatérské návrháře. Mezi jeho hlavní přednosti patří rozsáhlé knihovny komponent, které usnadňují návrh a výběr potřebných součástek, nebo také podpora 3D vizualizací, což umožňuje uživatelům lépe si představit výsledný produkt. Kromě toho je KiCad známý pro svou flexibilitu a otevřenost, což ho činí ideálním nástrojem pro různé typy projektů, od jednoduchých až po složité elektronické návrhy. Díky aktivní komunitě a pravidelným aktualizacím je nástroj neustále vylepšován.

KiCad je svobodný software, což znamená, že je zdarma a jeho zdrojový kód je veřejně dostupný. Tato vlastnost je obzvlášť ceněná mezi vývojáři a studenty, kteří mohou software využívat bez nutnosti platit licenční poplatky. Podporuje širokou škálu operačních systémů, včetně Windows, GNU/Linux a macOS, což zajišťuje dostupnost pro uživatele různých platforem.



Obrázek 5.1: Vývojové prostředí softwaru KiCad; Trasování desky.

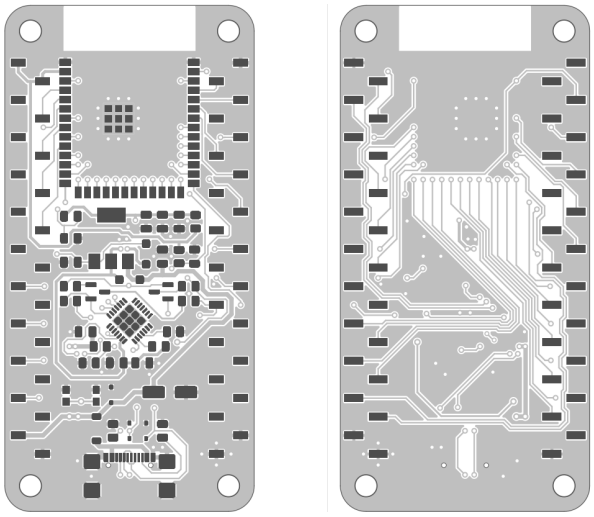
Pro návrh vývojové desky a modulů byl zvolen nástroj KiCad. Na rozdíl od Altium Designeru a Autodesk Eagle je zcela zdarma a nevyžaduje placenou licenci, což z něj činí ekonomicky dostupnou volbu pro široké spektrum uživatelů. Jako svobodný software poskytuje také volný přístup ke zdrojovému kódu, což umožňuje přizpůsobení a rozšiřování funkcí dle potřeb uživatelů. Další předností KiCadu je jeho podpora různých operačních systémů, což zvyšuje jeho flexibilitu a dostupnost. Tyto faktory, společně s aktivní komunitou uživatelů, činí KiCad atraktivním nástrojem, který umožňuje se snadno zapojit do úprav vytvořených návrhů v rámci této práce.

5.2 Návrh modulů

Tato sekce se zabývá procesem návrhu prototypu vývojové desky a jednotlivých modulů. Výsledné zdrojové soubory pro jejich výrobu jsou volně dostupné ve veřejném Git repozitáři na adrese `https://github.com/realcharmer/capyboard`.

5.2.1 Základní vývojová deska

Deska je osazena výhradně povrchově montovanými součástkami (SMD), což zajišťuje, že na zadní straně desky není žádná vystupující ostrá hrana. Toto uspořádání minimalizuje riziko nechtěného poškrábání pracovního stolu a umožňuje bezpečné používání zařízení v různých prostředích, včetně učeben bez pracovních podložek. Avšak nevýhodou je, že v domácích podmínkách není možné desku osadit bez použití specializovaných nástrojů. Navíc je třeba zdůraznit, že cena některých SMD součástek může být nepatrně vyšší, stejně tak i náklady za výrobu. Při odběru většího množství desek však tento cenový rozdíl nehraje příliš velkou roli.

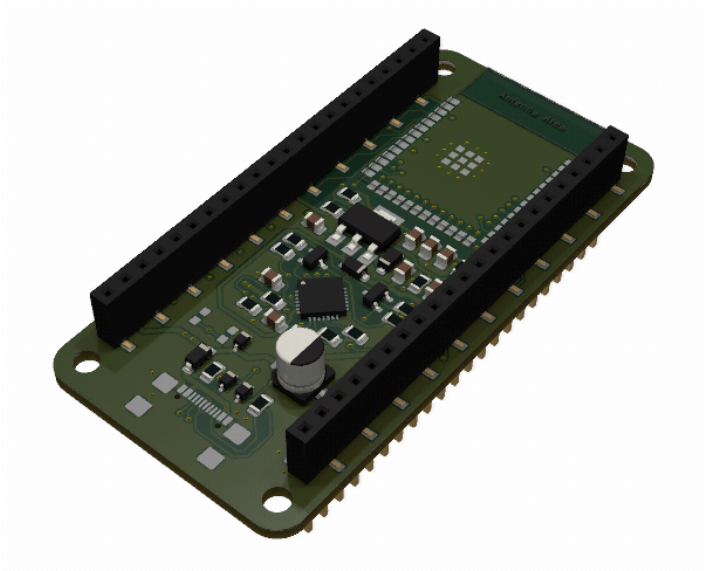


Obrázek 5.2: Schéma vývojové desky

Na obrázku 5.2 je zobrazeno schéma prototypu vývojové desky, které zahrnuje pohled na vrchní stranu (vlevo) a spodní stranu (vpravo). Tmavě jsou zvýrazněny pájecí plochy pro jednotlivé SMD komponenty a trasy propojující tyto komponenty. Návrh desky byl proveden s důrazem na minimalizaci rušení, což zahrnuje pečlivé trasování cest, implementaci výřezu pro anténu a optimalizaci přenosových charakteristik. K zajištění stabilního napájení přispívá široká napájecí cesta a „prokovy“ spojující vrchní a spodní stranu desky, které pomáhají při distribuci napájení a zemnění. Doplňující funkci v eliminaci šumu plní také správně navržené filtry (The Sierra Circuits Team, 2021). Tyto prvky společně přispívají k dosažení spolehlivého a stabilního provozu desky.

Deska postrádá jakékoli fyzické ovládací prvky, senzory či indikátory výstupů, jako například světelné diody, opomeneme-li však integrované senzory přímo v ESP32, například senzor teploty, magnetismu, nebo kapacitní senzory (Asih, 2023). Tato funkcionalita je výhradně implementována prostřednictvím modulů, které jsou připojeny k základní desce pomocí postranních konektorů běžně známých pod názvem „Pin Header“.

Na každé straně je osazeno dvanáct konektorů v jedné řadě s roztečí *2.54 mm*, což představuje standardní formát pro takovéto konektory, využívaný například v platformě Arduino či jiných vývojových deskách s platformou ESP32. Druhým a zároveň posledním dostupným konektorem je USB-C, který slouží jak pro napájení desky a modulů, tak i pro programování ESP32. Rozměry samotné desky jsou velmi kompaktní, konkrétně *34 mm* na šířku a *69 mm* na výšku.



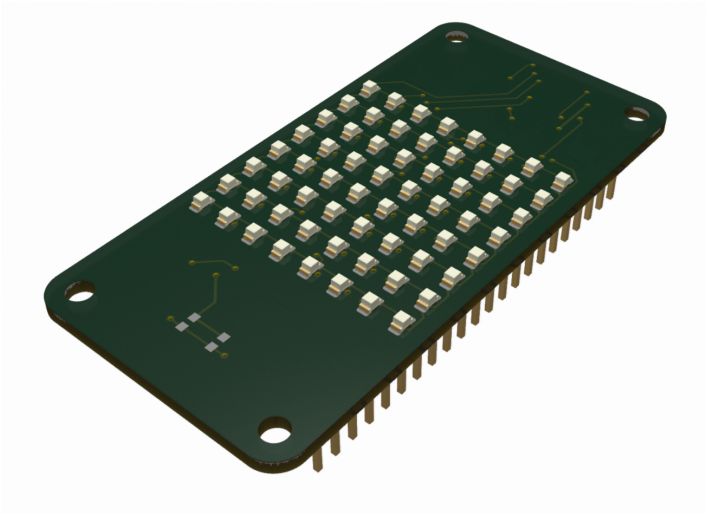
Obrázek 5.3: 3D náhled prototypu vývojové desky

Za zmínku rovněž stojí fakt, že využitím postranních konektorů typu SMD, ačkoliv zvětšují plochu celé desky, je možné desku v budoucnu rozšířit o stejnou sadu konektorů i ze spodní strany, například pro modul s baterií. Tento přístup byl inspirován právě zmíněnou stavebnicí Hardwario TOWER, viz sekce 4.1, která disponuje napájecím bateriovým modulem chytře připojitelným ze spodní strany vývojové desky.[1]

[1] V nové verzi vývojové desky jsou již tyto konektory ze spodní strany implementovány.

5.2.2 Světelný modul

Tento modul poskytuje matici diod ve vzoru 8×8. Každá ze 64 světelných diod v matici může být samostatně zapnuta nebo vypnuta, což umožňuje zobrazování různých vzorů, písmen nebo čísel. Inspirací pro tento modul byl známý vzdělávací mikroprocesorový systém BBC micro:bit, který také využívá matici LED pro zobrazování informací a interaktivních vzorů. Stejně jako micro:bit, i tento modul umožňuje uživatelům snadno programovat a ovládat jednotlivé LED, což je ideální pro výuku programování. Tímto způsobem se modul stává vynikajícím nástrojem pro vzdělávací účely, experimentování a vytváření různých projektů s vizuální zpětnou vazbou.



Obrázek 5.4: 3D náhled světeleného modulu

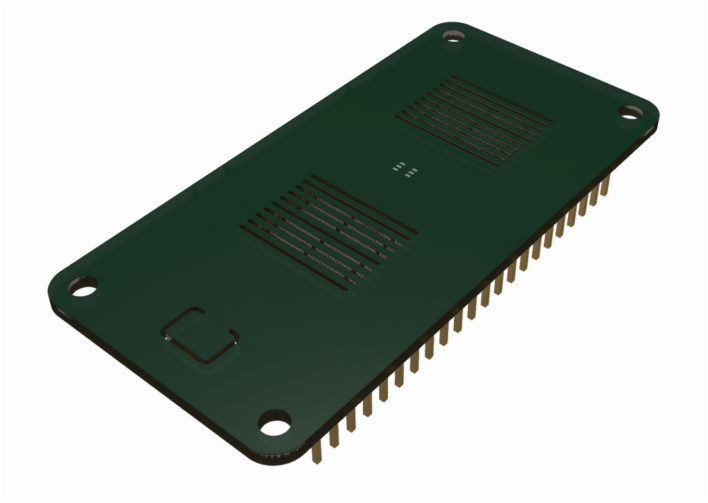
Funkcionalita tohoto modulu čerpá inspiraci z dříve zmíněného nástroje BBC micro:bit, viz kapitola 4. Jednou z jeho předních a široce využívaných funkcí je světelná matice diod, umožňující individuální adresování jednotlivých bodů. Díky tomu lze snadno vytvářet zajímavé tvary, vzory nebo animace. Na rozdíl od BBC micro:bit, který disponuje maticí o velikosti 5×5 diod, je tento modul vybaven maticí 8×8 diod, což poskytuje širší možnosti pro tvorbu komplexnějších tvarů, vzorů a animací.

Modul využívá čip MAX7221, který je specializován na adresování a ovládání jednotlivých diod, čímž usnadňuje jejich řízení. Čip MAX7221 je přímo řízen základní deskou, což znamená, že procesor ESP32 poskytuje všechny potřebné signály a napájení pro provoz modulu. Komunikace mezi hlavní deskou a modulem probíhá prostřednictvím rozhraní SPI (Serial Peripheral Interface). Tento čip je zároveň podporován běžně dostupnými softwarovými knihovnami.

SPI je sériové komunikační rozhraní, které umožňuje rychlý a efektivní přenos dat mezi hlavní deskou a čipem MAX7221. Toto rozhraní využívá několik pinů pro přenos dat, hodinový signál a další řídicí signály, což zajišťuje spolehlivou komunikaci a ovládání diod v matici. (Dhaker, 2018)

5.2.3 Klimatický modul

Tento modul představuje komplexní sadu senzorů určených k přesnému měření klimatických a dalších environmentálních veličin. Díky této sadě lze snadno vytvořit jednoduchou, ale plně funkční meteostanici pro vnitřní použití, která poskytuje přesné údaje o podmínkách v místnosti či jiném uzavřeném prostoru. Kromě toho může modul sloužit také k individuálnímu monitorování specifických parametrů okolního prostředí, což jej činí vhodným nástrojem pro řadu aplikací, od domácí automatizace až po monitorovací systémy v laboratořích či průmyslu.



Obrázek 5.5: 3D náhled klimatického modulu

Modul je navržen s ohledem na maximální kompatibilitu a využívá výhradně běžně dostupné senzory, které lze ovládat standardními softwarovými knihovnami. Toto řešení zajišťuje širokou přenositelnost výsledného kódu i na jiné vývojové platformy, pokud jsou kompatibilní s uvedenými knihovnami. Obsahuje čtyři základní senzory pro sběr okolních veličin:

- **Senzor světla** – Tento senzor umožňuje měření intenzity okolního osvětlení, což umožňuje monitorování světelných podmínek, například pro zajištění vhodného osvětlení rostlin nebo pracovních prostor.
- **Teplotní senzor** – Senzor teploty umožňuje přesné sledování okolní teploty, což je užitečné pro různé aplikace, od základního monitorování teploty v místnosti až po detekci teplotních změn v reálném čase, například pro optimalizaci topných nebo klimatizačních systémů.
- **Senzor tlaku a vlhkosti** – Tento kombinovaný senzor měří atmosférický tlak a relativní vlhkost vzduchu, čímž poskytuje užitečné údaje pro předpovědi a analýzu prostředí.
- **Senzor CO_2** – Tento senzor měří koncentraci oxidu uhličitého v okolním prostředí, což umožňuje sledování kvality vzduchu v uzavřených prostorách. Tato data mohou být využita k optimalizaci systémů větrání a k zajištění zdravějšího a produktivnějšího prostředí.

Je však nutné zdůraznit, že tento modul nebyl pro testování v praxi vyroben, a to z důvodu nedostatku finančních prostředků. Projekt zůstal ve fázi návrhu a nebyl realizován do fyzické podoby, což znamená, že nebylo možné provést testování ani validaci funkcí na reálném zařízení. Tento stav omezuje možnosti praktického použití návrhu, ale poskytuje základ pro případnou budoucí výrobu a testování, pokud budou k dispozici potřebné finanční zdroje.

5.3 Pořizovací cena

V rámci práce bylo vyrobeno pět prototypů vývojové desky a pět světelných modulů. Výsledné ceny byly přepočítány z amerických dolarů podle aktuálního kurzu k listopadu 2024, který činí 23,96 českých korun za 1 americký dolar. Ceny jsou tedy následující:

Modul	Vývojová deska	Světelný modul
Výroba	96 Kč	96 Kč
Osazení	2295 Kč	443 Kč
Cena za kus	478 Kč	108 Kč

K výsledné pořizovací ceně je nutné připočítat také dovozní daň ve výši 815 Kč a náklady na dopravu, které v tomto případě činily 765 Kč. Celková pořizovací cena tak dosáhla přibližně 4318 Kč za pět kusů, což odpovídá 903 Kč za jeden kus prototypu, zahrnujícího základní desku a světelný modul.

Je však důležité mít na paměti, že ceny prototypů bývají obvykle vyšší. Při výrobě ve větším objemu dochází k výraznému snížení nákladů na jednotlivé kusy. Cena jednotlivých součástek se navíc může lišit v závislosti na jejich aktuální dostupnosti.

5.4 Software

Tato sekce se věnuje softwarovým aspektům práce s vývojovou deskou. Zabývá se výběrem vhodné platformy pro vývoj a konfigurací potřebných nástrojů. Důraz je kladen na přípravu projektu, nastavení základních parametrů a usnadnění práce uživatelům prostřednictvím optimalizovaných konfiguračních souborů. Sekce se rovněž zabývá správou softwarového prostředí a zdrojového kódu, které zajišťují snadné využití vývojové desky v praxi.

Veškerá zde zmíněná a zároveň průběžně aktualizovaná konfigurace je dostupná ve veřejném Git repozitáři na adrese `https://github.com/realcharmer/capyboard-starter`.

5.4.1 Výběr softwarové platformy

Vzhledem k tomu, že vývojová deska využívá implementaci ESP32, podporuje různé platformy. V současné době jsou hlavními platformami *Arduino* (Nese stejný název jako zařízení Arduino) a „ESP-IDF“ (Espressif IoT Development Framework). Volba mezi těmito platformami závisí na požadavcích projektu, osobních preferencích a úrovni kontroly, kterou chce uživatel nad hardwarem udržovat.

Pro demonstraci lze použít jednoduchý program pro rozblikání jedné diody, také známý pod názvem *blink*, který je často implementován jako ukázkový program v řadě učebnic, například ve zmíněném *Arduino Projects Book* (Fitzgerald et al., 2013), či v učebnici *Robotika pro střední školy: programujeme Arduino* (Novák et al., 2020) z portálu *Informatické myšlení*.

```
void setup() {
    pinMode(2, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
}
```

Kód 1: Příklad implementace programu Blink pro platformu Arduino

```
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>

extern "C" void app_main() {
    gpio_pad_select_gpio(GPIO_NUM_2);
    gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT);

    while (1) {
        gpio_set_level(GPIO_NUM_2, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        gpio_set_level(GPIO_NUM_2, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Kód 2: Příklad implementace programu Blink pro platformu ESP-IDF

Při vývoji na platformě Arduino může být dosaženo jednodušší abstrakce, což je vhodné pro začátečníky. Arduino poskytuje vysokoúrovňové funkce a knihovny, což zjednodušuje celý proces vývoje programu. Pro tuto platformu je také dostupná rozsáhlá škála knihoven a dokumentace. Lze dokonce využít dokumentaci existujících projektů pro zařízení postavená na čipech z rodiny ATMEL (Arduino)

a libovolné další projekty pro ESP. Naopak vývoj s platformou ESP-IDF nabízí nižší úroveň abstrakce s přímou kontrolu hardwaru.

Čtenáři, který má zkušenosti s programováním podobných zařízení, je ukázka 1 pravděpodobně známější, než příklad 2, který využívá ESP-IDF. Pro účely referenční implementace byla proto zvolena platforma Arduino, jelikož pro ni existuje mnoho dostupných projektů, knihoven a učebnic. Navíc jsou učitelé pravděpodobněji obeznámeni s touto platformou spíše než s ESP-IDF. Nic však nebrání učiteli kurzu zakomponovat ESP-IDF do výuky v případě, že se chce této oblasti věnovat.

5.4.2 Konfigurace projektu

Pro efektivní využití vývojové desky je nutné připravit základní konfiguraci projektu pro PlatformIO. Je nutné podotknout, že tyto konfigurační soubory se mohou s dalšími verzemi vývojové desky měnit. Nejnovější verze konfiguračních souborů jsou vždy dostupné ve zmíněném Git repozitáři. Tato konfigurace se skládá z několika částí:

- Soubor *platformio.ini*: Tento soubor obsahuje základní konfiguraci desky, včetně specifikace čipu a dalších nezbytných parametrů pro kompilaci a nahrávání firmwaru.
- Složka *src*: Tato složka obsahuje samotný zdrojový kód projektu.
- Soubor *makefile*: Tento soubor slouží pro jednoduché ovládání projektu, umožňuje automatizovat běžné úkoly, jako je kompilace, nahrávání firmwaru na desku nebo spuštění sériového monitoru bez nutnosti znát příkazy pro PlatformIO.

Protože prototyp vývojové desky byl vyroben s využitím ESP32-S3-WROOM-1 s pamětí o velikosti pouze 4 MB, na rozdíl od běžných verzí s 8 MB, je nutné vytvořit vlastní paměťovou strukturu, bez které nelze výsledný firmware spouštět. Tato paměťová struktura je definována ve formátu hodnot oddělených čárkami (CSV, Comma-Separated Values) v souboru *partitions.csv* umístěném v kořenové složce projektu. Tento přístup umožňuje přesnou specifikaci a správu paměťových oblastí, což je nutné pro správné fungování firmwaru na prototypu s omezenou pamětí. O tomto omezení se blíže pojednává v kapitole 9.

```
1  # Name,   Type, SubType, Offset,   Size, Flags
2  nvs,      data, nvs,     0x9000,   0x5000,
3  otadata,  data, ota,     0xe000,   0x2000,
4  app0,     app,  ota_0,   0x10000,  0x1E0000,
5  app1,     app,  ota_1,   0x1F0000, 0x1E0000,
6  spiffs,   data, spiffs,  0x3D0000, 0x30000,
```

Kód 3: Soubor partitions.csv s definicí paměťové struktury

Jak bylo zjištěno při testování vývojové desky v praxi, viz kapitola 8, při vyšší paměťové náročnosti výsledného firmware dochází k chybě během jeho spouštění. To je způsobeno nedostatečně velkou

pamětí pro firmware a bylo tedy nutné vytvořit novou paměťovou strukturu. Tato změna spočívá v odstranění oddílů pro funkci OTA (Over-the-air update), která umožňuje nahrávání nové verze firmwaru bezdrátově. Jelikož tato funkcionalita není součástí didaktických materiálů, je možné ji odstranit. Pokročilý uživatel je však schopen tuto funkcionalitu obnovit v případě potřeby.

```
# Name,   Type, SubType, Offset,   Size, Flags
nvs,      data, nvs,     0x9000,   0x5000,
app,      app,  factory, 0x10000,  0x3C0000,
spiffs,   data, spiffs,  0x3D0000, 0x30000,
```

Kód 4: Definice paměťové struktury po odstranění funkce OTA a navýšení paměti pro firmware

Druhá verze prototypu, v reakci na uvedený problém, byla osazena čipem *ESP32-S3-WROOM-1-N8*, který disponuje 8 MB paměti Flash. Toto rozšíření paměti řeší nedostatek paměťových zdrojů, který vzniká při použití programu s velkým počtem knihoven a tím pádem i výrazným paměťovým zatížením.

Další důležitou součástí konfigurace je soubor *platformio.ini*, který zajišťuje správné nastavení projektu pro konkrétní vývojovou desku. Tento soubor obsahuje základní nastavení prostředí, definuje použitou platformu (viz sekce 5.4.1) a importuje také zmíněnou konfiguraci paměti. V tomto souboru se rovněž nastavuje rychlost sériové komunikace, tzv. *baud rate*.

```
[env:esp32-s3-devkitc-1]
platform = espressif32
board = esp32-s3-devkitc-1
framework = arduino
board_build.partitions = partitions.csv
board_upload.flash_size = 4MB
monitor_speed = 115200
```

Kód 5: Základní konfigurace PlatformIO pro vývojovou desku

Poslední důležitou částí je *Makefile*, tedy konfigurační soubor pro systém GNU Make, který výrazně usnadňuje práci s příkazy. Uživatel nemusí ručně vypisovat jednotlivé příkazy pro PlatformIO; místo toho stačí použít příkaz `make` spolu s jednou z dostupných akcí. Níže je uvedena ukázka aktuální podoby souboru *Makefile*, viz ukázka kódu 6.

Makefile poskytuje uživateli několik výchozích funkcionalit, včetně *build* pro kompilaci zdrojového kódu, *upload* pro nahrání programu do paměti vývojové desky a *clean* pro odstranění dočasných souborů vytvořených během kompilace. Dále obsahuje funkci *monitor*, která umožňuje uživateli otevřít sériovou linku přímo k zařízení za účelem sledování živého výstupu nebo ovládání zařízení pomocí příkazů. Po spuštění programu *make* jsou výchozím nastavením automaticky vyvolány funkce *build* a *upload*. Po úspěšném nahrání firmwaru do paměti je program na vývojové desce automaticky spuštěn.

```
PROJECT_DIR = $(CURDIR)
PLATFORMIO_BIN = /bin/pio

all: build upload

build:
    $(PLATFORMIO_BIN) run -d $(PROJECT_DIR)

upload:
    $(PLATFORMIO_BIN) run -d $(PROJECT_DIR) -t upload

clean:
    $(PLATFORMIO_BIN) run -d $(PROJECT_DIR) -t clean

monitor:
    $(PLATFORMIO_BIN) device monitor

.PHONY: all build upload clean monitor
```

Kód 6: Makefile pro práci s projektem

5.4.3 Práce s Git repozitářem

Konfigurační kroky není nutné provádět ručně; stačí naklonovat připravený Git repozitář ze zmíněného repozitáře na adrese `https://github.com/realcharmer/capyboard-starter`. Tento postup lze provést následujícím příkazem:

```
git clone https://github.com/realcharmer/capyboard-starter
```

Repozitář obsahuje popsané konfigurační soubory *platformio.ini*, *makefile* pro jednoduché ovládání, *partitions.csv* pro nastavení paměti [2] a složku *src* se zdrojovým kódem, což usnadňuje zahájení práce na projektu.

Je důležité mít na paměti, že soubory referenční konfigurace z příkladů 3, 5 a 6 se mohou v budoucnu lišit od poslední revize v Git repozitáři, a proto je doporučeno pravidelně kontrolovat jejich potenciální aktualizace a úpravy. Informace o práci se zařízením jsou vždy aktualizované v dokumentaci, o které se píše v kapitole 6.

Jedna z takových změn je použití čipu ESP32-S3-WROOM-1 s větší pamětí při implementaci nové verze prototypu, jak bylo rozebráno v předchozí sekci a v kapitole 9, která eliminuje potřebu redefinice paměťové struktury v souboru *partitions.csv*.

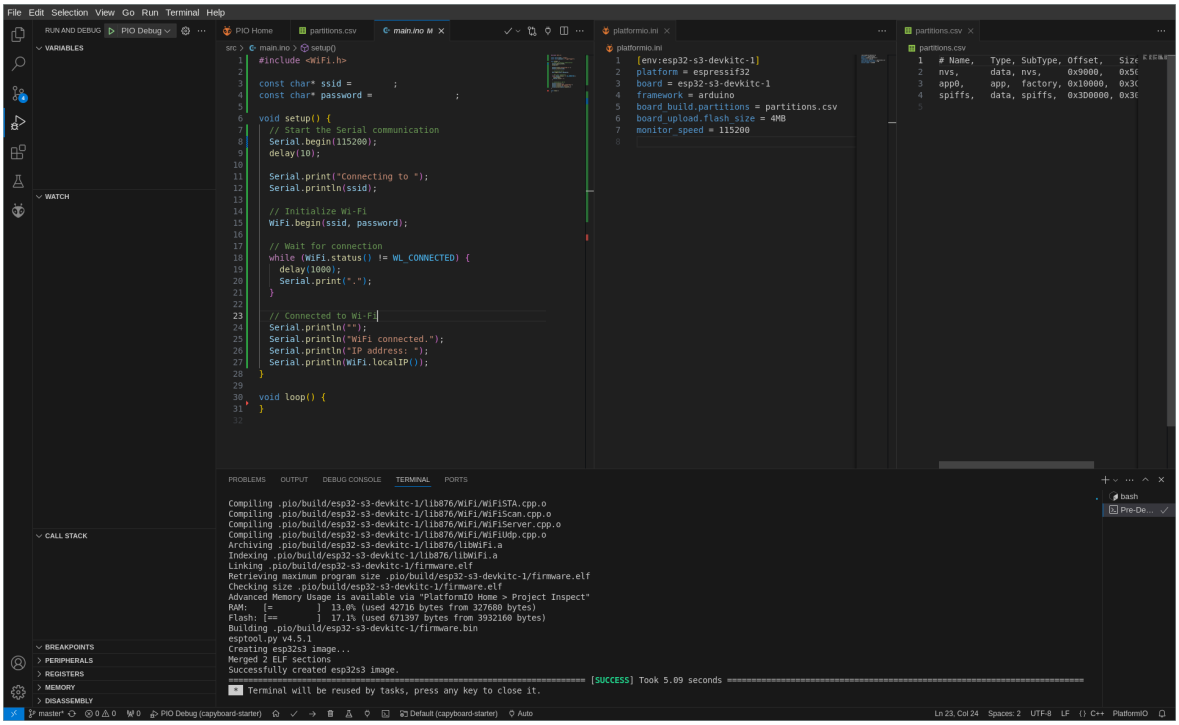
[2] Soubor *parititons.csv* není pro druhou iteraci vývojové desky nutný a není tedy obsažen v novějších verzích repozitáře.

5.4.4 PlatformIO IDE

PlatformIO IDE je moderní integrované vývojové prostředí, které rozšiřuje možnosti samotného systému PlatformIO, popsaného v kapitole 3. Primárně je určeno pro vývoj embedded systémů a zaměřuje se na odstranění problémů způsobených fragmentací nástrojů mezi různými platformami. Je postaveno na populárním editoru Visual Studio Code, což mu zajišťuje pokročilé funkce, jako je našeptávání, automatická detekce a oprava chyb v kódu či podpora rozšíření. Díky své otevřené povaze a nulovým nákladům na licencování poskytuje vývojářům plnou kontrolu nad prostředím. (Kravets, 2020)

PlatformIO IDE je zároveň výbornou volbou pro vzdělávání, neboť nabízí jednotné prostředí přístupné na všech běžných operačních systémech, což umožňuje snadnou integraci do výuky bez ohledu na používaná zařízení studentů. Integrované funkce, jako je správa knihoven nebo verzovací systém Git, přispívají k efektivnímu učení a moderním přístupům k výuce programování. Minimalizací času potřebného na nastavení a konfiguraci nástrojů umožňuje PlatformIO IDE zaměřit se přímo na samotné programování, což jej činí ideální volbou pro školy a další vzdělávací instituce.

Vývojová deska Capyboard je plně kompatibilní s PlatformIO IDE, což umožňuje snadnou integraci do moderního vývojového prostředí. Zároveň je podporována i skrze zmíněné konfigurační soubory. Uživatel si tedy může zvolit svůj preferovaný způsob práce se vývojovou deskou.



Obrázek 5.6: Vývojové prostředí PlatformIO IDE

Kapitola 6

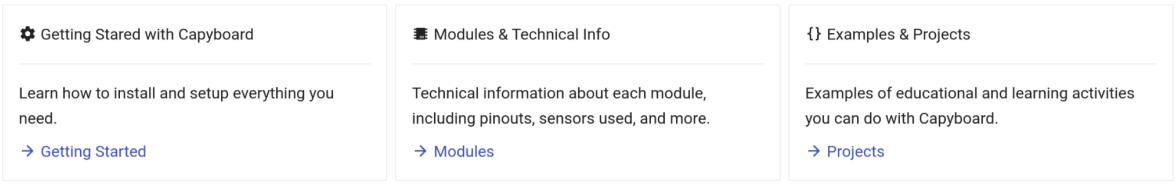
Sada didaktických materiálů

Tato kapitola se věnuje návrhu didaktických materiálů, které jsou následně implementovány v kapitole 7 a publikovány jako interaktivní webová stránka na adrese `https://docs.capyboard.dev`. Zmíněná kapitola 7 se zabývá převážně technickými aspekty implementace zde navržených didaktických materiálů do formy webových stránek, zatímco tato kapitola popisuje obsah, jeho strukturu a některé ukázkové příklady, které mohou sloužit jako inspirace pro výuku.

Didaktické materiály jsou navrženy tak, aby sloužily jako průvodce pro uživatele, ať už se jedná o učitele, studenty, nebo další zájemce, kteří chtějí pracovat s vývojovou deskou. Obsah dokumentace je přizpůsoben specifikům navržené vývojové desky a jednotlivým modulům. Poskytuje přehledně strukturované a srozumitelné informace, které usnadňují pochopení jednotlivých částí systému a jejich praktické využití.

6.1 Struktura

Didaktické materiály obsahují veškeré informace potřebné k provozování vývojové desky a modulů. Struktura didaktických materiálů je členěna do tří základních kategorií:



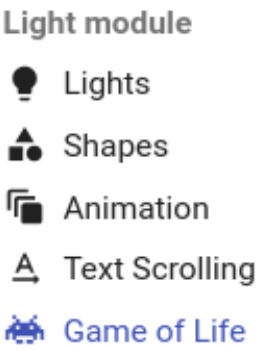
Obrázek 6.1: Popis jednotlivých sekcí na úvodní stránce webové dokumentace, viz kapitola 7.

První část se zaměřuje na základní konfiguraci vývojového prostředí a přípravu k práci s vývojovou deskou. Zahrnuje proces instalace PlatformIO a dalších potřebných nástrojů a obsahuje připravenou šablonu, která slouží jako základ pro vytvoření jednoduchého projektu.

Druhá část se věnuje popisu vývojové desky a jednotlivých modulů, včetně hardwarových specifikací, používaných komponent, aktivně využívaných pinů u modulů a doporučených knihoven pro práci s nimi, spolu s ukázkovým kódem. Tím je zajištěno, že uživatel nemusí vyhledávat informace v externí dokumentaci a může okamžitě začít pracovat.

Třetí, nejobsáhlejší část, se zaměřuje na ukázkové příklady, které jsou rozděleny podle jednotlivých hardwarových modulů a podle náročnosti realizace. Každý příklad obsahuje úvodní zadání a ukázkový základní kód. Důležitým prvkem ukázkových příkladů jsou tzv. *výzvy*, které postupně navazují na základní zadání a rozšiřují ho o další kroky. Tyto výzvy představují jednotlivé kroky, kterými by měl uživatel projít a tím si postupně rozšiřovat své dovednosti. Výhodou tohoto přístupu je možnost individuálního tempa. V kontextu výuky v kroužku by měl vyučující zastávat roli mediátora a pomáhat žákům s individuálními problémy, zatímco žáci pracují samostatně nebo ve skupinách vlastním tempem.

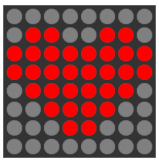
Jednotlivé příklady jsou zároveň rozděleny do kategorií podle konkrétních modulů, přičemž jedna sada příkladů je věnována samotné vývojové desce, další světelnému modulu, a podobně. V rámci těchto kategorií jsou příklady dále uspořádány podle obtížnosti, od základních úloh až po složitější zadání. Lze tedy říci, že pokud uživatel postupně projde všechny příklady včetně připravených výzev, jednotlivé úkoly na sebe plynule navazují jak tematicky, tak i z hlediska obtížnosti.



Obrázek 6.2: Jednotlivé úkoly pro práci se světelným modulem ve webové verzi dokumentace.

Ukázkou této návaznosti mohou být příklady pro světelný modul, viz obrázek 6.2. Zde je první část věnována přípravě modulu a základní práci s knihovnou *MD_MAX72XX*. Jednotlivé výzvy vedou uživatele k hlubšímu osvojení práce s knihovnou i samotným modulem a připomínají základní prvky imperativního programování, jako jsou například cykly. Prvním úkolem je pro uživatele rozblikání světelného modulu a vytvoření jednoduché animace. V navazujícím zadání je uživatel veden k vykreslení jednoduchého obrázku a následně, v dalším kroku, k jeho složitější animaci. Tento proces je průběžně doplňován zmíněnými výzvami, které nabízejí další kroky a rozšíření úkolu. Nakonec se uživatel propracuje až k vytvoření simulace tzv. „Game of Life“ neboli hry života.

6.2 Ukázkový příklad – Vykreslení tvaru



Nakreslete vlastní tvar, například kruh, čtverec nebo složitější tvar. Toho dosáhnete vytvořením dvojrozměrného pole bajtů s jednoduchou logikou: 1 znamená zapnutou diodu a 0 vypnutou diodu.

Výzvy

1. Invertujte svůj tvar, aniž byste upravovali samotné pole. Pixely, které jsou aktuálně zapnuté, by měly být vypnuté, a naopak.
2. Definujte svůj tvar v šestnáctkové soustavě (hexadecimální) namísto dvojkové (binární).

Ukázka základního kódu

```
#include <MD_MAX72xx.h>

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define CS_PIN 10
#define SEGMENTS 1

MD_MAX72XX display = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, SEGMENTS);

const int frame[8][8] = {
    {0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 1, 0, 0, 1, 1, 0},
    {1, 1, 1, 1, 1, 1, 1, 1},
    {1, 1, 1, 1, 1, 1, 1, 1},
    {0, 1, 1, 1, 1, 1, 1, 0},
    {0, 0, 1, 1, 1, 1, 0, 0},
    {0, 0, 0, 1, 1, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0}
};

void displayShape(const int frame[8][8]) {
    for (int row = 0; row < 8; row++) {
        for (int col = 0; col < 8; col++) {
            display.setPoint(col, row, frame[row][col]);
        }
    }
}

void setup() {
    display.begin();
    display.clear();
    displayShape(frame);
}

void loop() {
}
```

6.3 Ukázkový příklad – Animace textu

Vytvořte animovaný posunující se text pomocí knihovny *Parola*. Toho lze samozřejmě dosáhnout i pomocí animace po jednotlivých snímcích, ale využitím knihoven není třeba „znovu objevovat kolo“. Knihovna *Parola* vychází z knihovny *MD_MAX72xx* a rozšiřuje ji o řadu funkcí, jako je zarovnání, posun, animace a mnoho dalšího.

Nezapomeňte knihovnu importovat v souboru `platformio.ini`:

```
lib_deps =
  MD_MAX72XX
  MD_Parola
```

Výzvy

1. Upravte program tak, aby se text změnil po každém dokončení animace.
2. Nastavte, aby se směr a rychlost posouvání změnily s každým novým textem.

Ukázka základního kódu

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define CS_PIN 10
#define SEGMENTS 1

MD_Parola display = MD_Parola(HARDWARE_TYPE, CS_PIN, SEGMENTS);

void setup() {
    display.begin();
    display.displayClear();

    ledMatrix.displayScroll("Capyboard", PA_CENTER, PA_SCROLL_LEFT, 100);
}

void loop() {
    if (ledMatrix.displayAnimate()) {
        ledMatrix.displayReset();
    }
}
```

Kapitola 7

Webová dokumentace

Jedním z výstupů této práce je dokumentace, která poskytuje nejen návod k přípravě a provozu zařízení, ale také obsahuje ukázky projektů, na nichž lze zařízení využít. Tyto příklady, uvedené v kapitole 6, mohou učitelé začlenit do výuky jako inspiraci nebo přímo jako úkoly pro studenty.

Tato sekce se věnuje procesu tvorby, úpravy a publikování webové verze didaktických materiálů. Poskytuje uživatelům informace potřebné k vytvoření vlastních verzí těchto materiálů, přizpůsobených specifickým potřebám výuky, včetně přidání vlastního obsahu a úkolů. Zároveň slouží jako návod, jak provádět změny v oficiální webové dokumentaci, dostupné na adrese `https://docs.capyboard.dev`.

Oficiální dokumentace je otevřena k úpravám veřejnosti, která může své změny navrhovat prostřednictvím standardních „pull requestů“ v repozitáři `https://github.com/realcharmer/capyboard-docs`. Tento přístup umožňuje širší komunitě přispívat k obohacování obsahu a zajišťuje pravidelnou aktualizaci materiálů podle aktuálních potřeb a podnětů z praxe.

Fyzické učebnice, jako je například zmíněná učebnice *ARDUINO PROJECTS BOOK*, mají své nezbytné výhody, ale také několik nevýhod, které omezují jejich efektivitu v moderním vzdělávání. Jednou z hlavních nevýhod je jejich statický charakter. Obsah fyzických učebnic není aktualizován, což může znamenat zastaralé informace a nedostatečné pokrytí nových technologických vývojů, například softwarových knihoven. Fyzické učebnice jsou zároveň vázány na fyzický nosič (papír), což ztěžuje sdílení obsahu mezi studenty, a mohou být i finančně náročné.

Naproti tomu, vytvoření interaktivní webové stránky jako alternativy k fyzické učebnici přináší řadu výhod. Webové stránky umožňují snadnou aktualizaci obsahu a jeho okamžité zpřístupnění všem uživatelům. Interaktivní prvky mohou usnadnit orientaci v obsahu, vyhledávání, či práci s ukázkami kódu, který je možné jednoduše kopírovat.

Digitální vzdělávací materiály mohou pozitivně ovlivnit motivaci studentů a zlepšit jejich výsledky ve srovnání s tradičními metodami výuky. Vytváření digitálních studijních materiálů s otevřenou licencí a možností jejich úprav podle potřeb konkrétních skupin žáků je důležitý prvek efektivního vzdělávacího procesu. Tato flexibilita umožňuje pedagogům přizpůsobit obsah tak, aby odpovídal rozdílným úrovním znalostí a individuálním potřebám studentů. (Černý, 2022)

7.1 Výběr vhodné technologie

Pro tvorbu dokumentace existuje několik možných přístupů. Jedním z nich je využití dynamické webové aplikace ve stylu „wiki“ s databází. Nicméně takové systémy jsou často zbytečně složité pro většinu použití, a je proto výhodnější zvolit některý z funkčních generátorů statických webových stránek. Tím nejen získáme rychlejší výslednou webovou stránku, ale zároveň se vyhneme potřebě provozování databázového systému a případně také webového serveru, nýbrž statickou stránku lze také používat lokálně bez nutnosti připojení k internetu. (Buckler, 2021)

Nabízí se celá řada takových generátorů. Ty nejpoužívanější z nich mohou být například:

Název	**Jazyk**	**Jazyk šablon**	**Jazyk vstupu**	**Vícejazyčná podpora**
mdBook	Rust	Tera	Markdown	✗[1]
MkDocs	Python	Jinja2	Markdown	✓
Docusaurus	JavaScript	React	Markdown	✓
Zola	Rust	Tera	Markdown	✓

Do tohoto seznamu lze zahrnout i spoustu dalších generátorů, které nejsou přímo určené pro tvorbu dokumentace, avšak nabízejí pro tento účel připravené šablony. Je jich však tolik a tak často se jejich nabídka mění, že není praktické je zde vypisovat všechny. Při výběru vhodného generátoru záleží hlavně na preferencích uživatele, tedy autora dokumentace. Důležitá je i softwarová podpora, protože některé systémy mohou být složitější na instalaci či zakomponování do automatizovaných CI[2] systémů než jiné.

Použití statického generátoru může rovněž výrazně snížit bezpečnostní rizika, která jsou charakteristická pro dynamické webové aplikace. Jak uvádí dokument *OWASP Top Ten* (OWASP, 2021), který se zaměřuje na analýzu nejběžněji využívaných typů útoků na webové aplikace, statické webové stránky eliminují mnoho potenciálních bezpečnostních hrozeb tím, že minimalizují možnosti pro útoky, jako jsou *SQL injection* nebo *cross-site scripting (XSS)*. Využitím statické webové stránky se tedy aktivně předchází podobným typům zranitelností.

Ze zmíněných systémů byl autorem práce vybrán systém MkDocs. Hlavním důvodem pro jeho zvolení je jeho jednoduchost, uživatelská přívětivost a vícejazyčná podpora. Kromě toho existuje rozšíření, resp. alternativní verze s názvem *Material for MkDocs*[3], která nabízí oproti původnímu systému MkDocs moderní a atraktivní uživatelské rozhraní.

---

[1] Podpora více jazyků není v tuto chvíli implementována, viz `https://github.com/rust-lang/mdBook/issues/5`

[2] Continuous Integration – Automatická správa a integrace zdrojového kódu. V kontextu webových stránek často slouží ke kompilaci a publikaci výsledného webu.

[3] `https://squidfunk.github.io/mkdocs-material/`

Důležitým aspektem při výběru MkDocs byla také jeho podpora a dostupnost rozsáhlé dokumentace a sada rozšíření. Použití MkDocs a Material for MkDocs tak představuje výhodnou kombinaci jednoduchosti, funkčnosti a vizuální atraktivity, což přispívá k efektivní tvorbě a správě technické dokumentace, ale také k jejímu snadnému užívání z pohledu učitele i žáků.

7.2 Instalace a vývoj

Instalace systému MkDocs, resp. Material for MkDocs, je relativně snadná a intuitivní, což umožňuje její rychlé zprovoznění. Stačí se řídit oficiální dokumentací (Donath et al., 2024).

Existuje několik možných způsobů instalace. Každý z těchto způsobů instalace má své výhody a nevýhody, a volba konkrétní metody závisí na specifických potřebách a preferencích uživatele. Je tedy možné instalaci těmito způsoby:

1. Přímo do systému s využitím systémových Python knihoven
2. S využitím virtuálního prostředí Python Virtual Environment
3. Pomocí oficiálního kontejneru v systému Docker

Systémová instalace

První přístup, tedy instalace přímo do systému, je jednoduchý, ale může způsobit konflikty mezi verzemi knihoven, pokud je na systému nainstalováno více Python aplikací. Je možné jej provést snadno, a to pomocí nástroje *pip*.

Pip je nástroj, který umožňuje instalaci a správu softwarových balíčků napsaných právě v jazyce Python. Umožňuje snadné stahování a instalaci knihoven a dalších závislostí z PyPI (Python Package Index) a zajišťuje, že všechny potřebné komponenty jsou správně nainstalovány. (Kollár, 2020)

Instalace MkDocs a Material for MkDocs pomocí *pip* je možná jednoduchým příkazem:

```
pip install mkdocs-material mkdocs-awesome-pages-plugin
```

Příkaz stáhne a nainstaluje MkDocs spolu s potřebnými závislostmi. Zároveň je nainstalováno rozšíření *MkDocs Awesome Pages Plugin*[4], které je v dokumentaci využíváno pro systematické řazení jednotlivých stránek v navigaci. Tento typ instalace je vhodný pro uživatele, kteří chtějí rychle začít pracovat bez potřeby nastavení virtuálního prostředí nebo kontejnerizace.

[4] `https://github.com/lukasgeiter/mkdocs-awesome-pages-plugin`

Python Virtualenv

Druhou možností je využití virtuálního prostředí pomocí *Python Virtual Environment*, známého pod názvem *Virtualenv* nebo *Venv*. Ten umožňuje vytvořit oddělené prostředí, kde jsou všechny závislosti a balíčky nainstalovány lokálně, aniž by ovlivňovaly globální systémové nastavení.

Po aktivaci virtuálního prostředí se všechny příkazy *pip* a *python* vztahují pouze k tomuto izolovanému prostředí, což usnadňuje správu závislostí a verzí knihoven. Tento přístup je obzvláště vhodný pro vývojáře, kteří potřebují konzistentní a kontrolované prostředí pro každý svůj projekt, čímž se minimalizuje riziko konfliktů mezi různými verzemi balíčků. (Breuss, 2016)

Postup instalace MkDocs pomocí *Venv* zahrnuje následující kroky:

```
python -m venv venv
source venv/bin/activate
pip install mkdocs-material mkdocs-awesome-pages-plugin
```

V obou případech je následně možné spouštět systém MkDocs, a to následujícími příkazy s různými možnostmi a funkcionalitou:

- `mkdocs new [dir-name]` - Vytvoření nového projektu.
- `mkdocs serve` - Spuštění lokálního serveru s náhledem webových stránek.
- `mkdocs build` - Vygenerování výsledných souborů pro publikaci na webu.

Docker

Další možností je instalace pomocí oficiálního kontejneru v systému Docker. Tento způsob je vhodný pro uživatele, kteří chtějí izolovat instalaci od zbytku systému pomocí kontejnerizace. Docker umožňuje snadné nasazení a spuštění MkDocs bez ohledu na hostitelské prostředí. Na druhou stranu však způsobuje náročnější postup při instalaci možných rozšíření systému. Hodí se tedy hlavně do automatizace, která může výsledné webové stránky generovat a publikovat, viz sekce 7.4.

Docker je platforma pro kontejnerizaci, která umožňuje balení aplikací a jejich závislostí do kontejnerů. Kontejnery jsou lehké, přenosné a konzistentní prostředí, která zajišťují, že aplikace poběží stejným způsobem bez ohledu na to, kde jsou nasazeny. Docker zjednodušuje vývoj, testování a nasazení aplikací tím, že eliminuje problémy způsobené rozdíly mezi vývojovým a produkčním prostředím.

Použití Dockeru zajišťuje, že aplikace běží v izolovaném prostředí, což minimalizuje problémy s kompatibilitou a závislostmi. Tento přístup je ideální pro vývojáře a týmy, které potřebují konzistentní a reprodukovatelné prostředí pro své aplikace. Docker navíc usnadňuje škálování a nasazení aplikací na různé platformy a infrastruktury, což zvyšuje flexibilitu a efektivitu práce. (Ratan, 2017)

Pro instalaci Material for MkDocs pomocí Dockeru stačí použít oficiální kontejner. Postup zahrnuje stažení obrazu kontejneru z knihovny *Docker Hub* a následovné vytvoření a spuštění funkčního kontejneru. Příkaz pro stažení kontejneru vypadá následovně:

```
docker pull squidfunk/mkdocs-material
```

Protože výchozím zdrojem pro kontejnery v systému Docker je právně knihovna *Docker Hub*, není nutné zde specifikovat adresu a stačí pouze název samotného kontejneru. Spuštění lze pak provést například tímto příkazem:

```
docker run --rm -it -p 127.0.0.1:8000:8000 -v ${PWD}:/docs -u $(id -u):$(id -g)
↪  squidfunk/mkdocs-material serve
```

- `--rm` zajistí, že kontejner bude automaticky odstraněn po ukončení.
- `-it` umožňuje interaktivní mód.
- `-p 127.0.0.1:8000:8000` zajistí správné namapování síťového portu 8000 do kontejneru, což je nutné pro správné fungování příkazu `serve`.
- `-v ${PWD}:/docs` připojuje aktuální pracovní adresář na hostitelském systému do adresáře */docs* uvnitř kontejneru.
- `-u $(id -u):$(id -g)` mapují UID a GID uživatele do kontejneru, čímž se zajistí, že vytvořené soubory bude vlastnit stávající uživatel, nikoliv uživatel *root*.
- `squidfunk/mkdocs-material` specifikuje název kontejneru pro jeho stažení z *Docker Hub*.
- `serve` specifikuje akci, kterou má systém MkDocs vykonat (v tomto případě vytváření živého náhledu skrze lokální webový server).

Je nutné podotknout, že stažení kontejneru se provede automaticky, pokud se uživatel snaží kontejner spustit bez jeho předchozího manuálního stažení. Docker v tomto případě nejprve stáhne požadovaný obraz z Docker Hubu a následně spustí kontejner s tímto obrazem. Je tedy možné první krok stažení vynechat.

Pokud se uživateli zdá zmíněný příklad pro spuštění kontejneru složitý, a to oprávněně, je možné si pro něj vytvořit alias v systémovém Shellu, například:

```
alias mkdocs="docker run --rm -it -p 127.0.0.1:8000:8000 -v ${PWD}:/docs -u $(id -u):$(id -g)
↪  squidfunk/mkdocs-material"
```

Tímto způsobem je možné se vyhnout nutnosti zadávat dlouhý a komplikovaný příkaz při každém spuštění kontejneru. V takovém případě pak z pohledu běžného uživatele nelze rozeznat, v jakém prostředí je aplikace nainstalována.

7.3 Struktura a obsah

Obsah celé dokumentace se podobá běžné technické dokumentaci, která je často využívána v mnoha softwarových i hardwarových projektech. Obsah je rozdělen do jednotlivých hlavních sekcí, které jsou logicky členěny dle povahy obsahu. Tato struktura je nejen přehledná, ale také představuje standardní přístup ke strukturování dokumentace. Tento přístup usnadňuje orientaci a používání dokumentace díky své konzistenci a srozumitelnosti. Poukazují na to články, které se touto problematikou zabývají, například články *How to write technical documentation* (Ashby, 2023), nebo *5 Steps to Create Technical Documentation That's (Actually) Helpful* (MacKay, 2018).

Struktura dokumentace je navržena tak, aby pokrývala všechny aspekty projektu, včetně technických specifikací, návodů k instalaci, konfigurace a použití. Díky tomuto uspořádání mohou uživatelé snadno nalézt potřebné informace a rychle se zorientovat v jejím obsahu. Hlavní sekce zahrnují:

- Úvod
- Instalace a konfigurace softwaru
- Příprava zařízení a jeho ovládání
- Přehled jednotlivých modulů
- Ukázkové programy pro práci s vývojovou deskou a moduly

Sekce s ukázkovými příklady obsahuje příklady pro každý modul, které lze aplikovat ve výuce. Uživatelům, respektive učitelům, je tedy poskytován ucelený pohled na zařízení a jeho možnosti, ale také praktické návody a příklady, které pomáhají efektivně využívat dané zařízení ve výuce a usnadňují její přípravu. Viz předchozí kapitola 6.

Z technického hlediska je struktura dokumentace následující:

- Složka `docs` obsahuje samotný obsah, ze kterého se generují výsledné webové stránky.
- Soubor `mkdocs.yml`, který obsahuje základní konfiguraci projektu.
- Soubor `.gitignore`, který zajišťuje, že některé soubory a složky nebudou ukládány do Git repozitáře (například složka pro Python Venv).

MkDocs tedy generuje výsledné statické webové stránky ze zdrojového textu nacházejícího se ve složce *docs*, které je možné následně publikovat na jakýkoliv webový server.

Zdrojové soubory dokumentace lze najít v Git repozitáři dostupném na adrese `https://github.com/realcharmer/capyboard-docs`. Pro získání obsahu repozitáře je nejprve nutné provést klonování pomocí následujícího příkazu:

```
git clone https://github.com/realcharmer/capyboard-docs
```

Tímto způsobem je možné získat kompletní dokumentaci ve formátu vhodném pro další úpravy a publikaci. Aktuální verze dokumentace je však také dostupná na adrese `https://docs.capyboard.dev` bez nutnosti klonování repozitáře a instalace programu MkDocs. Její generování a publikace jsou popsány v následující kapitole.

7.4 Automatické publikování

Aby nebylo třeba při každé změně obsahu dokumentace instalovat systém MkDocs, ručně generovat soubory webových stránek a kopírovat je na webserver, je možné využít automatizovaných procesů, často nazývaných *Actions*. Tyto procesy mohou být spouštěny v různých situacích, například při nahrání nových změn do Git repozitáře, a mohou provádět rozmanité a složité úkony.

```yaml
name: Build

on:
  push:
    branches:
      - master

env:
  HOST: ${{ secrets.SSH_HOSTNAME }}
  HOST_DIR: ${{ secrets.SSH_TARGET_DIR }}
  SSH_USERNAME: ${{ secrets.SSH_USERNAME }}
  SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        run: |
          git clone https://github.com/realcharmer/capyboard-docs.git .

      - name: Install MkDocs
        run: |
          pip install mkdocs-material mkdocs-awesome-pages-plugin

      - name: Build
        run: |
          mkdocs build

      - name: Deploy
        run: |
          eval "$(ssh-agent -s)"
          echo "${SSH_PRIVATE_KEY}" | ssh-add -
          mkdir -p ~/.ssh/
          ssh-keyscan -H "${HOST}" >> ~/.ssh/known_hosts
          rsync -ra --delete-after site/ "${SSH_USERNAME}@${HOST}:${HOST_DIR}"
```

Kód 7: YAML konfigurace pro automatické generování a publikování pomocí *GitHub Actions*

Tyto kroky se obvykle provádějí uvnitř izolovaného kontejneru, známého jako *runner*, který je pro každý průběh procesu nově vytvořen. Tento postup zaručuje konzistenci prostředí při každém běhu, což minimalizuje riziko problémů s tzv. „znečištěním“ prostředí předchozími běhy.

Konfigurace uvedená v ukázce 7 slouží jako praktický příklad nastavení procesu generování a publikace navržené webové aplikace. Tato konfigurace je kompatibilní nejen se systémem GitHub Actions, ale lze ji přizpůsobit i jiným nástrojům, jako jsou *Gitea* nebo *Forgejo*. Konfigurační proces se skládá z několika kroků, přičemž první část určuje podmínky, za kterých má být proces spuštěn.

```
on:
  push:
    branches:
      - master
```

Tajné proměnné v systému *GitHub Actions*, v angličtině známé pod názvem „secrets“, jsou citlivá data, jako například přístupové klíče, hesla nebo API tokeny, které mohou být použity v rámci automatizovaných procesů (původním názvem *workflow*) pro přístup k externím službám nebo k nastavení při běhu automatizovaných procesů.

```
env:
  HOST: ${{ secrets.SSH_HOSTNAME }}
  HOST_DIR: ${{ secrets.SSH_TARGET_DIR }}
  SSH_USERNAME: ${{ secrets.SSH_USERNAME }}
  SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}
```

Tyto tajné informace jsou uloženy zašifrovaně a jsou předány do běhů těchto procesů prostřednictvím proměnných, což umožňuje ochranu citlivých informací a zabezpečení automatizovaných operací. Ukázkový workflow používá následující tajné proměnné:

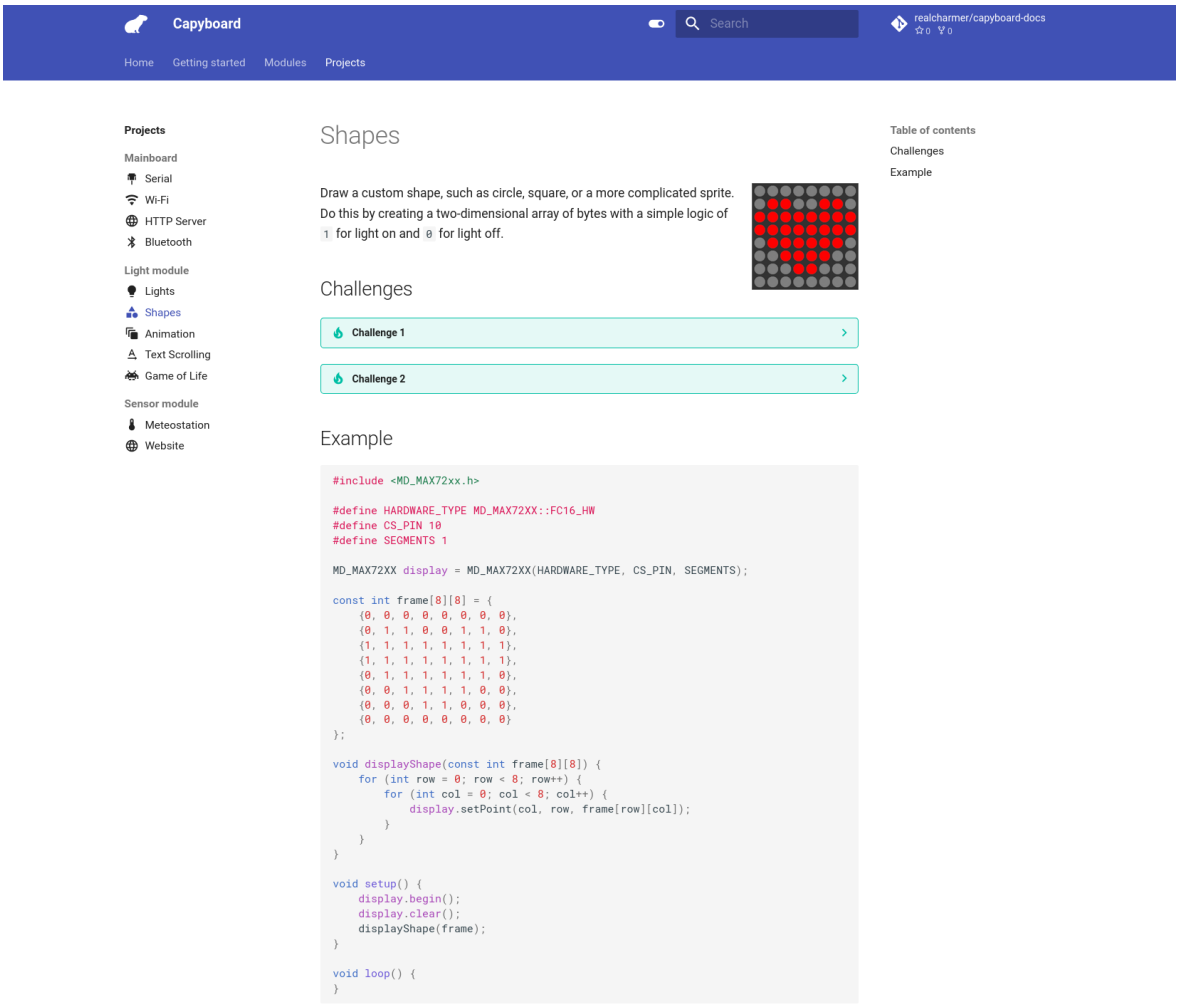
Název	**Popis**
`SSH_HOSTNAME`	Adresa webového serveru
`SSH_TARGET_DIR`	Cílová složka (kořenová složka webového serveru)
`SSH_USERNAME`	Uživatelské jméno pro SSH
`SSH_PRIVATE_KEY`	Soukromý klíč pro SSH

Poslední část konfigurace již definuje jednotlivé kroky, které budou vykonávány v rámci procesu:

1. Naklonování obsahu repozitáře
2. Instalace systému MkDocs, viz kapitola 7.2
3. Vygenerování výsledných souborů webových stránek
4. Konfigurace SSH a kopírování souborů na webserver

7.5 Barevná schémata

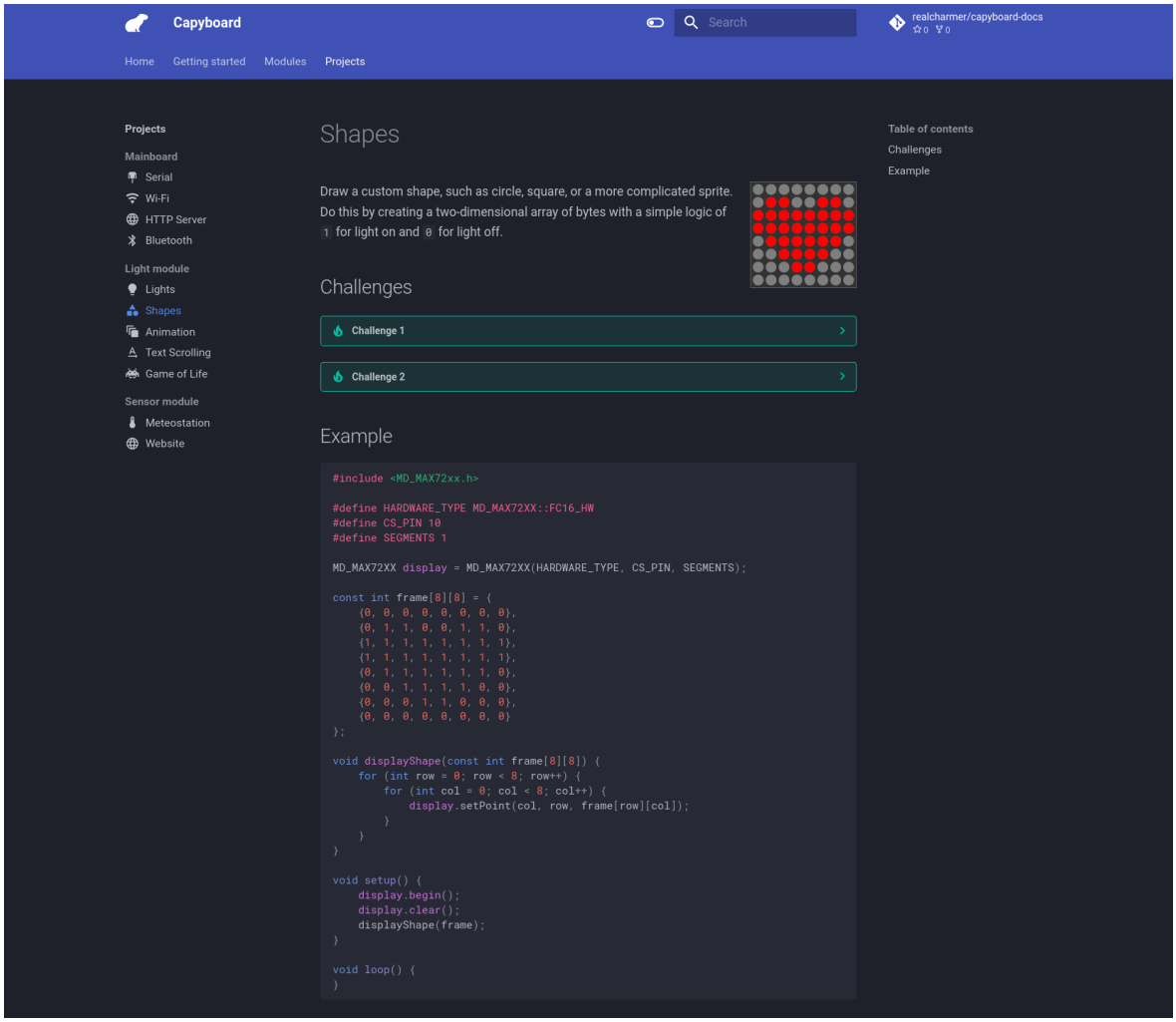
Studie s názvem *How Terra improved user engagement thanks to Dark Mode* (Bandarra et al., 2021) uvádí, že značná část uživatelů dává přednost tmavému pozadí. Výzkum naznačuje, že tmavé pozadí může významně zlepšit uživatelskou zkušenost a zvýšit zapojení uživatelů na webových stránkách. Preferenci tmavého režimu lze přičíst jeho estetickým vlastnostem a jeho schopnosti zlepšit čitelnost obsahu na obrazovkách v prostředí s nízkým osvětlením.



Obrázek 7.1: Náhled na příklad ze sekce 6.2 ve světlé verzi.

Dalším důležitým aspektem, který stojí za popularitou tmavého režimu, je jeho pozitivní vliv na snížení únavy očí. Světlé pozadí při delším používání může vést k větší zátěži pro oči, zatímco tmavé pozadí působí méně intenzivně a bývá vnímáno jako příjemnější. Studie dokonce uvádí, že více než polovina uživatelů zažila podráždění očí nebo poškození zraku spojené s dlouhodobým sledováním obrazovek (Smith, 2016). Tmavý režim tak může přispět ke zmírnění těchto potíží.

Kromě toho tmavé pozadí také přispívá k menšímu opotřebení displejů s technologií OLED. Displeje typu OLED také spotřebovávají méně energie při zobrazování tmavých barev, což může vést k prodloužení jejich životnosti. (Braun, 2019).



Obrázek 7.2: Náhled na příklad ze sekce 6.2 ve tmavé verzi.

Z těchto důvodů byl tmavý vzhled webové dokumentace zvolen jako výchozí, aby se maximalizoval uživatelský komfort a efektivita při práci s dokumentací. Uživatel má však možnost zvolit barevné schéma dle osobních preferencí přepínačem umístěným vlevo od pole pro vyhledávání.

Kapitola 8

Ověření v praxi

Vytvořená vývojová deska byla spolu se dvěma moduly testována ve dvou odlišných scénářích. Prvním bylo ověření v rámci kroužku programování jednočipových počítačů na Gymnáziu Jana Keplera. Druhé proběhlo formou workshopu organizovaného studentským spolkem *microlab* na Pedagogické fakultě Univerzity Karlovy.

Místo	Gymnázium Jana Keplera	Pedagogická fakulta, UK
Počet účastníků	7	5
Věkový rozsah	14–18	20–25
Počet setkání	2	1
Celkový časový rozsah	6 hodin	3 hodiny

Každá ze dvou následujících sekcí se zabývá detaily průběhu ověřování a hodnocení jednotlivých prvků práce, zejména samotné vývojové desky, didaktických materiálů a jejich formy, vhodnosti připravených úkolů a celkové použitelnosti těchto komponent ve výuce. Sekce se zaměřují na podrobné posouzení efektivity těchto prvků v praktickém prostředí a poskytují zpětnou vazbu pro další optimalizaci a zlepšení výukového procesu, která je podrobně rozebrána v kapitole 9.

8.1 Kroužek, Gymnázium Jana Keplera

Na Gymnáziu byl uspořádán krátký kroužek cílený právě na programování jednočipových počítačů. Přihlásilo se celkem 7 žáků, většina z nich znala nebo měla zkušenosti s platformou Arduino, někteří dokonce z předchozích kroužků, avšak objevili se i žáci, kteří se s podobnými technologiemi nesetkali. Přihlásili se převážně starší žáci vyššího gymnázia, avšak kroužku se účastnilo i dva mladší žáci z nižšího stupně. Jednalo se tedy o celkově rozmanitou skupinu.

Kroužek sestával ze dvou setkání, každé po třech výukových hodinách. Byly zde probrány základy fungování a využití jednočipových počítačů, následovala příprava prostředí k programování vývojové desky, seznámení s webovou dokumentací z kapitoly 7 a účastníci kroužku si postupně vyzkoušeli všechny úkoly blíže popsané v kapitole 6.

Pro výuku byla zvolena forma, při které žáci pracovali ve skupinách o počtu dva nebo tři, bez přímého vedení učitele, který jen poskytoval rady a odpovědi na případné dotazy a řídil průběh práce na jednotlivých úkolech. K tomuto rozdělení vedla skutečnost, že nebyl k dispozici dostatečný počet vývojových desek pro každého žáka. Tento přístup také podporoval spolupráci mezi studenty, kteří si mohli vzájemně pomáhat a sdílet své znalosti, což přispělo k pozitivnímu a produktivnímu učebnímu prostředí. Zpravidla starší žáci aktivně pomáhali těm mladším a vysvětlovali jim jednotlivé koncepty.

Vzhledem k tomu, že čas byl rezervován pouze na dvě setkání, byla výuka zaměřena výhradně na práci se světelným modulem a připravené úkoly pro tento modul. Někteří žáci však dokončili své úkoly rychleji a ke konci kroužku si nad rámec tematického plánu vyzkoušeli i další možnosti vývojové desky, například práci s technologií Wi-Fi.

Výsledky ověření

S pozitivním ohlasem se setkala forma didaktických materiálů v podobě webové stránky. Ze své podstaty poskytuje jednoduchý přehled jednotlivých modulů a slouží jako dobrý základ pro plnění jednotlivých úkolů. Učitel mohl webovou stránku nejen promítat, ale každý žák si poté mohl dle potřeby prohlížet dokumentaci sám. Tato forma umožnila žákům snadný a okamžitý přístup k potřebným informacím a podpořila jejich samostatnost při řešení úkolů.

Struktura jednotlivých úkolů s využitím několika úrovní obtížnosti se ukázala být velice úspěšnou, zejména v prostředí, kde pracovali studenti s rozmanitou úrovní zkušeností. Úkoly byly navrženy tak, aby zahrnovaly jak základní, tak pokročilé části, což umožnilo každému studentovi najít úroveň, která odpovídala jeho znalostem a schopnostem. V praxi se nestalo, že by někdo nezvládal základní část, nebo naopak neměl co dělat a zůstal nečinně čekat. Díky této struktuře úkolů měl každý student vždy výzvu odpovídající jeho schopnostem, což vedlo k maximálnímu zapojení všech žáků.

Z technického hlediska se během těchto setkání ukázalo, že původní navrhované změny struktury paměti v sekci 5.4.2 nestačí v případě, že je výsledný firmware příliš veliký, například při integraci mnoha externích knihoven. Problém byl na místě vyřešen změnou paměťových oddílů, viz 5.4.2.

Praktická zkušenost z kroužku rovněž zdůraznila důležitost efektivního vývojového prostředí. PlatformIO a s ním spojený editor PlatformIO IDE se ukázaly jako užitečné nástroje, protože poskytují jednotné prostředí pro správu kódu, kompilaci, nahrávání firmware a jeho jednoduchý vývoj. Integrace s PlatformIO výrazně zjednodušuje průběh výuky a přispívá k jejímu hladkému průběhu.

Ukázalo se, že eliminováním potřeby zapojování jednotlivých senzorů a tím, že byla vývojová deska s moduly připravena na okamžité programování, bylo možné se se studenty dostat mnohem dál než během výuky kroužků právě s platformou Arduino, která se odehrávala v předchozích letech. Studenti tak měli více času věnovat se samotnému programování a řešení úkolů, místo aby se zabývali složitým zapojováním hardwaru. Tato zjednodušená příprava výrazně přispěla k efektivnějšímu využití času během kroužku a umožnila hlubší porozumění programovacím konceptům a využívaným knihovnám.

8.2 Workshop, Pedagogická fakulta, Univerzita Karlova

Workshop s rozsahem tří vyučovacích hodin byl uspořádán na Pedagogické fakultě Univerzity Karlovy u příležitosti setkání neformálního spolku *microlab*, který je zaměřen právě na informační technologie, zejména na programování, elektrotechniku a kybernetiku. Workshopu se aktivně zúčastnilo celkem pět vysokoškolských studentů, nejen z prostředí Pedagogické fakulty, ale také z Fakulty elektrotechnické ČVUT a Matematicko-fyzikální fakulty Univerzity Karlovy. Věk účastníků pokrýval dvacet až dvacet pět let a všichni účastníci byli mírně pokročilí až pokročilí v obecné oblasti programování.

Čtyři z účastníků měli zkušenost s programováním jednočipových počítačů, dva z nich se považovali za pokročilé, a jeden účastník se již dokonce věnoval návrhu vlastních jednočipových počítačů. Lze tedy stanovit, že účastnická základna, byť malá, byla rozmanitá s ohledem na znalosti a dovednosti v této oblasti.

Na začátku workshopu byl vysvětlen princip fungování vývojové desky a softwarového vybavení, kterým se deska ovládá. Rozsah pouze tří vyučovacích hodin se opět soustředil zejména na práci se světelným modulem. Účastníci si osvojili práci s vývojovou deskou výrazně rychleji a pracovali na připravených úkolech. Někteří účastníci si také vyzkoušeli úkoly, které jsou cíleny na práci se samotnou vývojovou deskou.

Výsledky ověření

Praktická zkušenost z workshopu zdůraznila důležitost efektivního vývojového prostředí a volba systému PlatformIO se potvrdila jako správná. V tomto případě pracoval každý ve vlastním prostředí, nikoli pouze v PlatformIO IDE. Nikdo ze zúčastněných se nemusel zabývat složitou konfigurací a stačilo jen nainstalovat PlatformIO a u některých také Git pro naklonování repozitáře se šablonou základního projektu. Někteří však využili možnost stažení této šablony formou archivu *zip*, která je uvedena v dokumentaci.

Obrázek 8.1: Tlačítko pro stažení šablony projektu v archivu zip.

Struktura jednotlivých úkolů s využitím několika úrovní obtížnosti se opět ukázala jako velmi úspěšná. Úkoly byly navrženy tak, aby zahrnovaly jak základní, tak pokročilé části, což umožnilo každému účastníkovi postupovat vlastním tempem. Díky této struktuře úkolů měl každý účastník vždy výzvu odpovídající jeho schopnostem, což vedlo k maximálnímu zapojení všech zúčastněných.

Při práci na úkolech určených pro vývojovou desku však bylo zjištěno, že původní návaznost těchto

úkolů není dostatečně dobře seřazena dle obtížnosti, kdy mezi úkoly zaměřenými na sériovou linku a poté na WiFi je obrovský skok v náročnosti. Proto byly implementovány úpravy ve struktuře těchto úkolů, viz následující kapitola 9.

Podobně jako u kroužku na gymnáziu, i zde se pozitivně osvědčila forma didaktických materiálů v podobě webové stránky. Účastníci cítili potřebu splnit všechny připravené výzvy a považovali tento přístup za zábavnou formu vzdělávání.

Vývojová deska i didaktické materiály se ukázaly být vhodnými i pro studenty vysoké školy. S ohledem na jejich pokročilejší znalosti a dovednosti oproti žákům gymnázia však zabere připravená sada úkolů podstatně méně času. Tento fakt je důležitý pro další úpravy a přizpůsobení materiálů, aby byly dostatečně náročné a atraktivní i pro vysokoškolské studenty. Workshop tedy ukázal, že je možné využít stávající strukturu úkolů a rozšířit ji o další pokročilé úkoly a projekty, které by lépe odpovídaly schopnostem vysokoškolských studentů. Není tedy nutné vývojovou desku omezovat pouze na prostředí střední školy, potažmo gymnázia.

Celkově lze říci, že workshop, po úpravách vycházejících z první reflexe na gymnáziu, potvrdil efektivitu navrženého přístupu a metodiky. Z ověření tedy vyplývá, že zvolené materiály a struktura úkolů jsou vhodné pro široké spektrum studentů s různou úrovní znalostí a dovedností.

8.3 Shrnutí výsledků

Didaktické materiály jsou velmi flexibilní a snadno přizpůsobitelné potřebám výuky, protože nejsou příliš specifické. Nabízejí různé úrovně obtížnosti prostřednictvím připravených výzev, což umožňuje jejich využití pro studenty s různými úrovněmi znalostí. Forma dokumentace ve formě webové stránky je velmi dobře hodnocena, zejména díky její přehlednosti a snadnému přístupu.

Softwarová platforma je spolehlivá a uživatelsky přívětivá. Funguje bez problémů na různých operačních systémech, aniž by vyžadovala složitou ruční konfiguraci, což značně usnadňuje práci.

Hardware měl v předchozí verzi některé nedostatky, které však byly odstraněny v nové revizi vývojové desky a modulů, což výrazně zlepšilo jeho použitelnost.

Tento systém je ideální zejména pro výuku v rámci zájmových kroužků, ale své místo by mohl najít i jako součást povinné výuky. Jeho využití je však nejvhodnější pro studenty vyšších ročníků gymnázií, kteří již disponují pokročilejšími znalostmi v oblasti programování. Díky své modularitě a přehlednosti přirozeně navazuje na výuku s nástrojem micro:bit a zároveň představuje most k pokročilejší práci s Arduinem, které obohacuje výuku o praktické zapojování periferií.

Kapitola 9

Reflexe

Tato kapitola se zabývá vyhodnocením didaktických materiálů a možnými změnami pro další iteraci vývojové desky, které vyplývají z výsledků ověřování během praktické výuky a testování, jak je podrobně popsáno v kapitole 8. Výsledky těchto ověřování zahrnují zpětnou vazbu ohledně funkčnosti, spolehlivosti a uživatelské přívětivosti vývojové desky, ale také strukturu a náročnost vytvořených úkolů. Zohledňují se také názory a doporučení kolegů, kteří se podíleli na konzultaci výsledného produktu. Tyto konzultace přinesly cenné postřehy týkající se optimalizace návrhu, snížení výrobních nákladů a zlepšení celkového pohodlí a přehlednosti pro uživatele. Na základě těchto poznatků je možné identifikovat možné oblasti pro zlepšení, jako je výběr alternativních komponent, úpravy v návrhu desky pro usnadnění montáže, změny didaktických materiálů a širší možnosti využití vývojové desky ve výuce, nebo v běžné praxi.

9.1 Didaktické materiály

Jak bylo popsáno v kapitole 8, ověření probíhalo ve dvou scénářích, konkrétně v prostředí střední školy a následně na vysoké škole. Účastníky tvořili žáci a studenti různých věkových a dovednostních kategorií. Z důvodu časové tísně byly oba kurzy zaměřeny na práci se základní deskou a světelným modulem. Praxe ukázala, že začínat s bezdrátovým spojením, jak je navrženo v původní struktuře didaktických materiálů, je značně náročné. Je proto vhodné nejprve posílit dovednosti v práci s vývojovou deskou a se samotným programovacím prostředím a jazykem pomocí jednodušších úkolů.

Z původní struktury byla tedy rozdělena kategorie *Základní deska* na kategorie *Základy* a *Sítě*. Toto uspořádání se v návaznosti na výsledky ověření jeví jako nejvhodnější pro systematické a efektivní osvojení potřebných dovedností. Zde je srovnání původní a upravené struktury úkolů:

Původní struktura

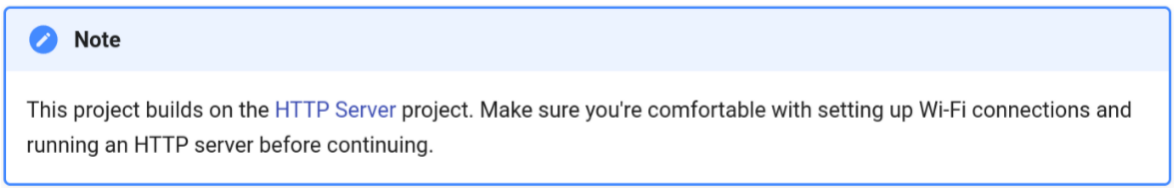
- Základní deska
  - Sériová linka
  - Wi-Fi
  - HTTP Server
  - Bluetooth
- Světelný modul
  - Světla
  - Tvary
  - Animace
  - Rolovací text
  - Hra života
- Klimatický modul
  - Meteostanice
  - Webová stránka

Nová struktura

- Základy
  - Sériová linka
- Světelný modul
  - Světla
  - Tvary
  - Animace
  - Rolovací text
  - Hra života
- Klimatický modul
  - Meteostanice
  - Webová stránka
- Sítě
  - Wi-Fi
  - HTTP Server
  - Bluetooth

Kategorie *Základy* nyní obsahuje úkoly zaměřené na základní práci s vývojovou deskou, primárně tedy se sériovou linkou, která je potřebná pro většinu dalších navazujících úkolů z ostatních kategorií. Uživatel si zde procvičí základní práci s vývojovou deskou, sériovou linkou a syntaxi jazyka.

Druhá nová kategorie *Sítě* se již věnuje zmíněným náročnějším úkolům, tedy zprovoznění Wi-Fi, HTTP serveru a technologii Bluetooth a je zařazena za všechny ostatní kategorie. Protože je například práce s Wi-Fi sítí obsažena v některých dalších úkolech, je na ni občas odkazováno jako na prerekvizitu formou viditelného upozornění v hlavičce stránky, viz obrázek 9.1, což narušuje linearitu postupu v jednotlivých úkolech. I přesto je tento nelineární přístup u některých úkolů v praxi výhodnější.



Obrázek 9.1: Upozornění na prerekvizitu pro úkol.

Lze však očekávat, že struktura úkolů bude v budoucnu podrobena změnám vzhledem k přibývajícímu obsahu, dalším úpravám a testování nových funkcí zařízení. Aktuální verze sady úkolů bude vždy k dispozici na webové verzi dokumentace.

S ohledem na časové rozvržení materiálů se zdá být původní návrh správný, tj. rozsah šesti setkání, který vychází z praxe kroužku programování platformy Arduino na Gymnáziu Jana Keplera. Tento rozsah se osvědčil a je také cílem pro navržené didaktické materiály, tedy jednotlivé úkoly. Lze tedy předpokládat, že stejný rozsah bude adekvátní i pro sadu připravených úkolů a není třeba provádět mnoho úprav v návaznosti na časové rozvržení. Záleží však na učiteli, jaké úkoly se rozhodne se žáky řešit a kolik času jim bude věnovat. To ovšem také závisí i na dovednostech skupiny žáků a výsledná časová náročnost se může u různých skupin lišit.

Forma dokumentace a didaktických materiálů v podobě webových stránek se ukázala jako velmi vhodná. Uživatelé během testování vyzdvihovali jejich přehlednost a jednoduchost. Zároveň je velmi snadné provádět úpravy a rozšiřovat obsah těchto webových stránek nejen ze strany autorů, ale i uživatelů a veřejnosti jednoduchou úpravou ve verzovacím systému Git.

9.2 Softwarové vybavení

Zvolení PlatformIO jakožto vývojové platformy se v praxi ukázalo jako velmi vhodné řešení. PlatformIO poskytuje jednotnou platformu pro správu zdrojového kódu a hardware, což umožňuje efektivní práci s vývojovou deskou napříč různými operačními systémy. Díky této multiplatformní podpoře mohou uživatelé minimalizovat technické obtíže související s kompatibilitou.

Jedním z důležitých aspektů je zavedení jednotného prostředí, tedy editoru a s ním spjatých nástrojů. Jednotné prostředí se ukázalo jako důležité pro hladký průběh práce s celým systémem, zejména v kontextu vzdělávacích kurzů s mnoha účastníky. I když PlatformIO není úzce spjato s konkrétním textovým editorem či vývojovým prostředím, sjednocení takového prostředí mezi všemi účastníky kurzu přináší výrazné výhody. Vedoucí výuky může zvolit vhodné prostředí v návaznosti na skupinu účastníků a jejich úroveň znalostí a dovedností.

Toto sjednocení umožňuje učiteli efektivně prezentovat základní principy systému prostřednictvím jednoho sdíleného nástroje. Minimalizuje se tím riziko fragmentace individuálních problémů, které by mohly vzniknout při využití různých vývojových prostředí s odlišnými funkcemi a nastaveními.

Flexibilita PlatformIO však zároveň umožňuje pokročilým uživatelům zvolit si vlastní, pro ně vhodnější vývojové prostředí. Tento přístup podporuje individuální přizpůsobení na základě osobních preferencí a zkušeností, což je výhodné zejména při práci na složitějších projektech.

Autor práce proto doporučuje využití vývojového prostředí PlatformIO IDE (viz sekce 5.4.4), které kombinuje textový editor, správu knihoven a multiplatformní podporu. Tato kombinace nástrojů vytváří ucelený a intuitivní rámec, který je ideální jak pro začínající, tak pokročilé uživatele, a představuje vhodnou volbu zejména pro vzdělávací kurzy a projekty zaměřené na práci s vývojovými deskami.

9.3 Hardware

Následující poznatky jsou založeny na testování první verze vývojové desky. Na základě těchto poznatků vyplývajících z testování v praxi, viz kapitola 8, byla vytvořena druhá verze, která řeší značnou část problémů, jež se během testování objevily. Tato nová verze je nyní dostupná jako oficiální funkční iterace v dříve zmíněném Git repozitáři na adrese `https://github.com/realcharmer/capyboard`.

Pořizovací cena

Zásadním problémem, kterému čelí prototyp vývojové desky, i přes původní cíl nízkých výrobních nákladů, je vysoká pořizovací cena. Tato částka je ovlivněna několika faktory, zejména složitým procesem osazování některých komponent a jejich vysokou cenou, jako je například SMD USB-C konektor, nebo samotný ESP32 modul, který vyžaduje složitý proces pájení, který není snadné reprodukovat v domácích podmínkách.

Dalším nevýhodným aspektem je výroba v malých sériích, která je nákladná, viz sekce 5.3. Při větší sériové výrobě by cena výrazně poklesla. Nicméně, nelze soupeřit s rozsáhlou produkcí konkurenčních nástrojů, jako je například micro:bit, které jsou vyráběny ve velkých množstvích a za nižší cenu. Tyto konkurenční nástroje však mají i mnoho nevýhod, jak je rozebráno konkrétně v kapitole 4.

Když vezmeme v potaz nákupní cenu některých stavebnic, například desky Arduino a potřebných rozšíření a součástek, zjistíme, že abychom touto sadou a jejími součástkami pokryli možnosti implementovaného prototypu, vyšplhá se výsledná cena také poměrně vysoko. Samotné Arduino UNO lze v tuto chvíli sehnat za přibližně 700 korun[1], ale jakmile přidáme senzory, moduly pro komunikaci, napájecí komponenty a další příslušenství, celkové náklady rychle rostou. Naopak jiný typ vývojových desek, například zmíněný micro:bit, svou podstatou ani neumožňuje tak rozsáhlá rozšíření funkcionality.

Jako příklad může sloužit situace, kdy chceme z Arduina vytvořit zařízení s podobnými funkcemi jako má prototyp vývojové desky. Potřebovali bychom zakoupit základní desku Arduino, několik senzorů (např. teplotní senzor, světelný senzor), moduly pro komunikaci (např. Wi-Fi nebo Bluetooth), různé konektory, nepájivé pole (breadboard) a případně i další rozšiřující komponenty. Kromě vysokých nákladů na komponenty bychom také museli investovat čas a úsilí do jejich integrace.

V kontrastu s tím, Capyboard nabízí hotové řešení, které eliminuje potřebu mnoha externího příslušenství, což může z dlouhodobého hlediska představovat nejen úsporu nákladů, ale i času a úsilí. Pro vzdělávací instituce, které se snaží efektivně spravovat své zdroje, může být tato integrovaná přístupná platforma výrazně výhodnější, navzdory počátečním investicím.

[1] Aktuální cena v Internetovém obchodě `rpishop.cz`

Velikost paměti

První verze prototypu desky Capyboard využívá jako své jádro modul *ESP32-S3-WROOM-1-N4* s pamětí o velikosti pouze 4 MB, což vyžaduje speciální konfiguraci pro strukturu paměti, viz sekce 5.4.2. To nijak nebrání funkcionalitě, avšak představuje další konfigurační soubor, který může zbytečně navyšovat komplexnost při iniciálním pochopení celého systému u nových uživatelů.

Tato potřeba speciální konfigurace může být pro nováčky odrazující, zejména pokud nejsou obeznámeni s detaily hardwarové architektury a správou paměti. Na druhou stranu, pro pokročilejší uživatele tato konfigurace nabízí možnost optimalizace a efektivního využití dostupné paměti.

V praxi se však ukázalo, že původní navrhované řešení změny struktury paměti v sekci 5.4.2 nestačí v případě, že je výsledný firmware příliš veliký, například při integraci mnoha externích knihoven. Proto bylo potřeba strukturu paměti optimalizovat. Optimalizace zahrnovala přerozdělení dostupné paměti tak, aby byl maximalizován prostor pro firmware a minimalizovány nepoužívané oblasti. To má za následek znehodnocení funkce OTA (Over-the-air update), tedy bezdrátové aktualizace firmwaru. Podrobnosti o těchto změnách jsou popsány v příkladu 4 v sekci 5.4.2.

Konečné a optimální řešení však vzniklo v podobě druhé verze prototypu, která nově využívá modul *ESP32-S3-WROOM-1-N8* s větší dostupnou pamětí. Tato verze vývojové desky je nyní dostupná ve výše zmíněném Git repozitáři.

Tlačítko Reset

Původní prototyp byl navržen bez jakýchkoliv přídavných LED či tlačítek. Při prvním testování se však ukázalo, že toto omezení představuje problém, protože některé operace vyžadují resetování programu bez přerušení spojení se sériovým monitorem. Reset je přitom možné provést stažením pinu EN k zemi. Softwarově lze ESP32 resetovat pouze nahráním nového firmwaru nebo pomocí nástroje *esptool.py*, který však vyžaduje, aby nebyla otevřena sériová linka. To lze provést například následujícím příkazem:

```
esptool.py --port /dev/ttyUSB0 run
```

Tento postup je však velice krkolomný a bylo by vhodné na vývojovou desku implementovat tlačítko pro reset, jelikož jeho využitím se ulehčí mnoho čekání na znovu-nahrání firmware, či eliminuje potřebu pro nástroj *esptool.py*. Nové revize vývojové desky již tuto změnu zahrnují a lze provést reset pomocí tlačítka umístěného přímo na desce.

Změny v usazení USB-C konektoru

Během testování vznikl podnět, který poukazoval na fakt, že umístění USB-C konektoru na úrovni kraje desky může představovat významný problém při integraci desky do ochranného krytu nebo krabičky. V takové konfiguraci je připojení USB-C kabelu obtížné a nepraktické, protože konektor není snadno přístupný. Vhodnější by bylo, kdyby USB konektor vyčníval, což by umožnilo snadné a pohodlné připojení kabelu bez nutnosti vyjmutí vývojové desky ven z krabičky.

Na druhou stranu, umístění zařízení do krabičky je relevantní v případě, že je již naprogramované. Vytvoření otvoru pro USB-C konektor by jej mohlo vystavit prachu a dalším nečistotám, což by mohlo ovlivnit jeho funkčnost. V praxi se pravděpodobně ukáže jako lepší řešení zařízení z krabičky při každé potřebě přeprogramování vyjmout. V tomto případě by současné umístění konektoru na kraji desky bylo vyhovující.

Modul s baterií

Po naprogramování desky a vytvoření plnohodnotně funkčního výrobku, například meteostanice s použitím klimatického modulu, nelze výsledné zařízení snadno umístit do místnosti a provozovat například ke sledování stavu ovzduší. To je způsobeno tím, že základní desku lze napájet pouze skrze USB-C konektor nebo připojením baterie na správné GPIO piny. Ukázalo se tedy, že bateriový modul je velmi vítaným rozšířením pro další iterace vývojové desky, jak bylo zmíněno v kapitole 5.2.1 následovně:

> „Za zmínku rovněž stojí fakt, že využitím postranních konektorů typu SMD, ačkoliv zvětšují plochu celé desky, je možné desku v budoucnu rozšířit o stejnou sadu konektorů i ze spodní strany, například pro modul s baterií.“

Tato úprava zjednodušuje použití zařízení v reálném světě a také zvyšuje jeho atraktivitu pro různé praktické aplikace, od monitorování životního prostředí až po autonomní senzory a zařízení nasazená v terénu. Připojení baterie by tedy umožnilo nepřetržitý provoz bez nutnosti modul připojovat a napájet jej skrze USB-C. Poslední verze prototypu je již pro tento modul připravena a nabízí konektory z obou stran.

Zajištění správné orientace připojení modulů

V tuto chvíli je prototyp desky osazený dvěma identickými bočními konektory, jak je vidět na obrázku 5.3. To může způsobit situaci, kdy uživatel zapojí libovolný modul opačně, tedy otočený o 180 stupňů se špatnou polaritou. To by nemělo poškodit vývojovou desku ani samotný modul, avšak nebude jej možné ovládat.

Tento problém může mít významné důsledky pro funkčnost celého systému a negativně ovlivňuje

pohodlnost práce se stavebnicí. Nesprávné zapojení může vést k frustraci uživatele, zejména pokud není okamžitě zřejmé, že problém spočívá v nesprávné polaritě konektoru.

Řešením tohoto problému by mohlo být osazení vývojové desky jiným typem konektorů, nebo využití konektorů s různou délkou. To ovšem nutně neeliminuje problém s opačným propojením.

Další možností je zahrnutí ochranných mechanismů přímo na úrovni desky nebo modulu. To může zahrnovat například diody pro ochranu proti obrácené polaritě, které zamezí průchodu proudu v případě nesprávného zapojení. Takový přístup však může zvyšovat náklady a složitost návrhu desky.

Nejvhodnějším řešením by mohlo být použití klíčovaných konektorů, které fyzicky zabrání nesprá-nmu zapojení. Klíčované konektory mají asymetrický tvar nebo speciální zářezy, které umožňují správné spojení pouze jedním směrem. Nabízí se tedy využít boční konektory, které na desce nejsou propojeny k žádné součástce nebo čipu (často označované jako NC – Not Connected). V tomto případě by bylo možné jeden z těchto konektorů zaslepit, čímž by se fyzicky zabránilo možnosti opačného zapojení modulu. Toto řešení je relativně jednoduché a levné na implementaci a zároveň poskytuje vysokou míru spolehlivosti při zamezení nesprávného zapojení.

Vrstvení jednotlivých modulů

Stávající rozšiřující moduly jsou navrženy s konektory směřujícími pouze dolů, což umožňuje jejich připojení k vývojové desce. Tento přístup vytváří velmi jednoduchý systém, avšak neumožňuje vrstvení jednotlivých modulů. V této souvislosti se nabízí otázka, zda by nebylo žádoucí rozšířit funkcionalitu systému o možnost zapojení více modulů současně, a zda je výhodné obětovat jednoduchost ve prospěch modularity.

Jednou z možností je navrhnout nové moduly, které budou připraveny pro sériové zapojení tím, že budou mít konektory i na své druhé straně. Tento přístup by neměl způsobit žádné problémy, protože pro komunikaci s moduly se využívají protokoly SPI nebo I^2C.

Jako druhá možnost se nabízí návrh modulu, který by poskytoval více výstupů pro připojení dalších modulů a agregoval by je do jedné sběrnice.

Zigbee a Thread

Jak bylo dříve zmíněno, prototyp vývojové desky využívá modul ESP32-S3-WROOM-1, který disponuje několika bezdrátovými technologiemi pro přenos dat, konkrétně Bluetooth a Wi-Fi. Firma ESPRESSIF však nabízí i další verze tohoto čipu, které rozšiřují možnosti bezdrátové komunikace. Podrobnosti o různých verzích čipů jsou dostupné v nástroji ESPRESSIF Product Selector[2], z nichž konkrétně ESP32-C6 a ESP32-H2 podporují technologie *Zigbee* a *Thread*. Využitím zmíněných alter-

[2] `https://products.espressif.com`

nativních čipů lze dosáhnout podpory i těchto bezdrátových technologií. (Espressif Systems, 2024a)

Zigbee i Thread jsou protokoly pro bezdrátové sítě založené na standardu IEEE 802.15.4, které jsou oblíbené zejména v aplikacích zaměřených na nízkou spotřebu energie a robustní komunikaci. Tyto technologie, zejména Zigbee, jsou hojně využívány v oblasti IoT, jak je patrné z aktuální nabídky chytrých zařízení pro domácnosti. Typicky se používá v chytrých domácnostech, průmyslové automatizaci a dalších aplikacích, kde je požadována nízká latence a spolehlivost. (ZigBee Alliance, 2013)

IEEE 802.15.4 je standard navržený pro nízkou spotřebu energie a nízkou přenosovou rychlost pro bezdrátové síťové technologie. Tento standard definuje fyzickou a linkovou vrstvu pro bezdrátové sítě s malým dosahem a nízkými nároky na energetickou spotřebu, jak popisuje oficiální standard (IEEE, 2020). Hlavní rysy standardu IEEE 802.15.4 zahrnují:

- **Nízká spotřeba energie:** Optimalizován pro zařízení s omezeným napájením.
- **Nízká datová rychlost:** Nabízí přenosové rychlosti typicky od 20 kb/s do 250 kb/s.
- **Síťová topologie:** Podporuje hvězdicové, clusterové a meshové topologie.
- **Bezpečnost:** Zahrnuje základní mechanismy pro zabezpečení dat a přístupu k síti.
- **Frekvence a dosah:** Pracuje v bezlicenčním pásmu rádiových frekvencí (např. 2,4 GHz nebo 868/915 MHz) s dosahem obvykle několika desítek metrů.

Popisky GPIO

Pokud se uživatel rozhodne využívat vývojovou desku bez připojených modulů, je orientace v GPIO konektorech bez relevantních popisků značně složitá. Proto by bylo vhodné označit tyto konektory podle zavedených konvencí, což výrazně usnadní jejich použití a minimalizuje chyby při zapojování vlastních komponent v případě, že uživatel nechce používat pouze navržené rozšiřující moduly.

První možností je vytvořit sítotiskovou masku, známou v angličtině jako „silkscreen mask“, která zahrnuje bílé nebo černé popisky jednotlivých součástek a konektorů. Tento přístup je běžně používaný v průmyslu pro jasné a trvalé označení komponent. Nicméně, vývojová deska je velice kompaktní a pro umístění těchto popisků by pravděpodobně musela být rozšířena, což nemusí být vždy možné nebo žádoucí. Alternativně lze popisky natisknout na spodní stranu desky, což sice zachovává kompaktní rozměry, ale může ztížit čitelnost a orientaci při práci s deskou.

Další možností je umístění popisků přímo na plastový obal jednotlivých bočních konektorů. Tento přístup zajišťuje, že popisky jsou snadno viditelné a přímo připojené k relevantním konektorům, což usnadňuje jejich identifikaci. Nicméně, výroba konektorů s takto specifickým označením může být finančně náročná, což by mohlo zvýšit celkové náklady na výrobu vývojové desky.

Závěr

Analýza rámcového vzdělávacího programu pro gymnázia potvrdila význam rozvoje informatického myšlení, algoritmizace a programování jako zásadních součástí moderního vzdělávání. Přezkoumání dostupných didaktických materiálů pak poskytlo důležité podklady pro vytvoření vlastního výukového nástroje zaměřeného na výuku programování s využitím jednočipových počítačů.

Zásadním přínosem práce je návrh a realizace vývojové desky s přidruženými moduly, doplněná o detailní webovou dokumentaci. Tím bylo vytvořeno alternativní řešení pro uživatele, kterým nevyhovují běžně dostupné platformy, a které poskytuje značnou flexibilitu pro různé vzdělávací potřeby.

Návrh sady didaktických materiálů slouží jako ucelený vzor, který podporuje ověření vývojové desky v praxi a přípravu další výuky. Tato strukturovaná sada materiálů umožňuje efektivní začlenění programování a práce s navrženou deskou do školních osnov, čímž usnadňuje rozvoj digitálních kompetencí v oblasti algoritmizace a programování.

Praktické ověření vytvořených nástrojů a materiálů v reálném vzdělávacím prostředí potvrdilo jejich přínosnost a použitelnost. Zároveň odhalilo určité nedostatky, které poskytly cennou zpětnou vazbu pro jejich další vývoj. Na základě těchto poznatků byly navrženy úpravy pro budoucí iterace nástroje i doprovodných didaktických materiálů. Celkové výsledky potvrzují, že navržený výukový nástroj efektivně přispívá k rozvoji informatického myšlení a kompetencí v oblasti algoritmizace a programování, a to prostřednictvím propojení teoretických znalostí s praktickými zkušenostmi.

Bibliografie

3DSOURCED, 2023. *10 Best PCB Design Software in 2023* [online]. [cit. 2024-01-22]. Dostupné z: https://www.3dsourced.com/3d-software/best-pcb-design-software/.

ARDUINO, 2018. *What is Arduino?* [online]. [cit. 2024-03-20]. Dostupné z: https://www.arduino.cc/en/Guide/Introduction.

ARDUINO, 2024a. *Arduino Courses* [online]. [cit. 2024-03-20]. Dostupné z: https://www.arduino.cc/education/courses.

ARDUINO, 2024b. *Arduino UNO R4 Product Reference Manual*. Tech. zpr. Arduino.

ASHBY, Steve, 2023. *How to write technical documentation — with examples* [online]. [cit. 2024-06-05]. Dostupné z: https://www.gitbook.com/blog/how-to-write-technical-documentation-with-examples.

ASIH, Kinanti Wening, 2023. *All About Sensors! ESP32's internal components* [online]. [cit. 2024-03-12]. Dostupné z: https://medium.com/@conantiwening/all-about-sensors-esp32s-internal-components-aca3aed0b892.

BALADA, Jan, 2007. *Rámcový vzdělávací program pro gymnázia*. Praha: Výzkumný ústav pedagogický v Praze. ISBN 978-80-87000-11-3.

BALADA, Jan, 2021. *Rámcový vzdělávací program pro gymnázia* [online]. [cit. 2024-02-27]. Dostupné z: https://www.edu.cz/wp-content/uploads/2021/09/001_RVP_GYM_uplne_zneni.pdf.

BANDARRA, André Cipriani; RENZULLI, Demián; SOUZA, Guilherme Moser de, 2021. *How Terra improved user engagement thanks to Dark Mode* [online]. [cit. 2024-06-07]. Dostupné z: https://web.dev/case-studies/terra-dark-mode.

BAUDUIN, Sven, 2023. *The best Raspberry Pi alternatives* [online]. [cit. 2024-03-20]. Dostupné z: https://www.pcworld.com/article/2137681/the-best-alternatives-to-the-raspberry-pi.html.

BEAGLEBONE, 2024. *PocketBeagle TechLab* [online]. [cit. 2024-01-22]. Dostupné z: https://www.beagleboard.org/boards/techlab.

BIGCLOWN, 2018. *BigClown Playground* [online]. [cit. 2024-10-26]. Dostupné z: https://medium.com/@bigclown/bigclown-playground-99e4917eb53b.

BRAUN, Andrew, 2019. *Are Dark Themes Really Better for Your Eyes and Battery?* [online]. [cit. 2024-06-07]. Dostupné z: https://www.maketecheasier.com/are-dark-themes-better-for-eyes-battery/.

BREUSS, Martin, 2016. *Python Virtual Environments: A Primer* [online]. [cit. 2024-06-05]. Dostupné z: https://realpython.com/python-virtual-environments-a-primer/.

BUCKLER, Craig, 2021. *7 Reasons to Use a Static Site Generator* [online]. [cit. 2024-03-20]. Dostupné z: https://www.sitepoint.com/7-reasons-use-static-site-generator/.

BUTTS, Jeff, 2021. *What is the Raspberry Pi?* [online]. [cit. 2024-03-20]. Dostupné z: https://www.howtogeek.com/754492/what-is-raspberry-pi/.

ČERNÝ, Michal, 2022. *DigCompEDu 2.2: Tvorba a úprava digitálních zdrojů* [online]. [cit. 2024-11-10]. Dostupné z: https://clanky.rvp.cz/clanek/23168/DIGCOMPEDU-2-2-TVORBA-A-UPRAVA-DIGITALNICH-ZDROJU.html.

DHAKER, Piyu, 2018. *Introduction to SPI Interface* [online]. [cit. 2024-11-06]. Dostupné z: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html.

DONATH, Martin et al., 2024. *Material for MkDocs Documentation* [online]. [cit. 2024-02-25]. Dostupné z: https://squidfunk.github.io/mkdocs-material.

ESPRESSIF SYSTEMS, 2023. *ESP32 Series Datasheet v4.4*. Tech. zpr. Espressif Systems.

ESPRESSIF SYSTEMS, 2024a. *ESP Product Selector* [online]. [cit. 2024-01-22]. Dostupné z: https://products.espressif.com/.

ESPRESSIF SYSTEMS, 2024b. *ESP32-DevKitC V4* [online]. [cit. 2024-01-22]. Dostupné z: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp32-devkitc/user_guide.html.

FITZGERALD, Scott; SHILOH, Michael, 2013. *ARDUINO PROJECTS BOOK*. Second reprint. Arduino LLC.

HALFACREE, Gareth, 2024. *The Official Raspberry Pi Beginners Guide 5th Edition*. 194 Science Park, Cambridge: Raspberry Pi, Ltd. ISBN 978-1912047260.

HARDWARIO, 2021. *HARDWARIO aktivní STEM výuka* [online]. [cit. 2024-03-08]. Dostupné z: https://stem.hardwario.com/v/cs.

HAVÍŘOVÁ, Barbora, 2019. *BBC micro:bit ve škole* [online]. [cit. 2024-03-20]. Dostupné z: https://www.e-mole.cz/clanek/bbc-microbit-ve-skole.

IBRAHIM, Dogan, 2023. *Mastering the Arduino Uno R4*. Elektor International Media B.V. ISBN ISBN 978-3-89576-578-0.

IEEE, 2020. IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, s. 1–800. Dostupné z DOI: 10.1109/IEEESTD.2020.9144691.

IMYŠLENÍ, 2018. *Učebnice a vzdělávací materiály pro školy* [online]. [cit. 2024-03-20]. Dostupné z: https://imysleni.cz/ucebnice.

KOLLÁR, László Kiss, 2020. *Managing Python packages the right way* [online]. [cit. 2024-06-05]. Dostupné z: https://opensource.com/article/19/4/managing-python-packages.

KRAVETS, Ivan, 2018. *Add support for BeagleBone Black* [online]. [cit. 2024-01-22]. Dostupné z: https://github.com/platformio/platform-linux_arm/issues/10.

KRAVETS, Ivan, 2020. *Next-generation IDE for decades, not years* [online]. [cit. 2024-11-20]. Dostupné z: https://piolabs.com/blog/insights/next-generation-ide-for-decades.html.

MACKAY, Jory, 2018. *5 Steps to Create Technical Documentation That's (Actually) Helpful* [online]. [cit. 2024-06-05]. Dostupné z: https://plan.io/blog/technical-documentation/.

MALÝ, Martin, 2017. *Hradla, volty, jednočipy: Úvod do bastlení*. Praha: CZ.NIC, z. s. p. o. ISBN 978-80-88168-24-9.

MALÝ, Martin, 2024. *ESP32 Prakticky*. Praha: CZ.NIC, z. s. p. o. ISBN 978-80-88168-79-9.

MOLLOY, Derek, 2019. *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux*. Wiley. ISBN 978-1119533160.

MŠMT, 2021. *Rámcový učební plán* [online]. [cit. 2024-02-27]. Dostupné z: https://digitalizace.rvp.cz/files/rvp-g-7-ramcovy-ucebni-plan.pdf.

NOVÁK, Milan; PECH, Jiří, 2020. *Robotika pro střední školy: programujeme Arduino*. Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta. ISBN 978-80-7394-786-6.

OWASP, 2021. *OWASP Top Ten* [online]. [cit. 2024-03-20]. Dostupné z: https://owasp.org/Top10/.

PARK, Andrew, 2022. *Advantages of Open Source Software for Devs and Companies* [online]. [cit. 2024-11-25]. Dostupné z: https://www.heavybit.com/library/article/open-source-software-benefits-advantages/.

PECH, Jiří; NOVÁK, Milan, 2020. *Robotika pro střední školy: programujeme Micro:bit pomocí Pythonu*. Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta. ISBN 978-80-7394-787-3.

PECH, Jiří; PRŠALA, Jan; VANÍČEK, Jiří; NOVÁK, Milan, 2021. *Robotika pro základní školy: programujeme micro:bit pomocí Makecode*. Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta. ISBN 978-80-7394-851-1.

PETAZZONI, Thomas, 2023. *Bootlin Embedded Linux course now available on BeaglePlay* [online]. [cit. 2024-01-22]. Dostupné z: https://bootlin.com/blog/bootlin-embedded-linux-course-now-available-on-beagleplay/.

PLATFORMIO, 2023. *What is PlatformIO?* [online]. [cit. 2024-03-08]. Dostupné z: https://docs.platformio.org/en/latest/what-is-platformio.html.

POLLARD, Barry, 2024. *Hardware And Software Used In Education* [online]. [cit. 2024-11-27]. Dostupné z: https://adivi.com/blog/hardware-and-software-used-in-education/.

RAMON, Manoel Carlos, 2014. Arduino IDE and Wiring Language. In: *Intel® Galileo and Intel® Galileo Gen 2: API Features and Arduino Projects for Linux Programmers*. Berkeley, CA: Apress, s. 93–143. ISBN 978-1-4302-6838-3. Dostupné z DOI: 10.1007/978-1-4302-6838-3_3.

RATAN, Vivek, 2017. *Docker: A Favourite in the DevOps World* [online]. [cit. 2024-06-05]. Dostupné z: https://www.opensourceforu.com/2017/02/docker-favourite-devops-world/.

SEDLÁK, Jan, 2017. *Liberecký BigClown se odtrhl od Jablotronu a s hardwarovou skládačkou jde do světa* [online]. [cit. 2024-10-25]. Dostupné z: https://www.lupa.cz/clanky/liberecky-bigclown-se-odtrhl-od-jablotronu-a-s-hardwarovou-skladackou-jde-do-sveta/.

SMITH, Sandy, 2016. *Most Americans Experience Digital Eye Strain from Overexposure to Computers* [online]. [cit. 2024-06-07]. Dostupné z: https://www.ehstoday.com/ppe/eye-face-head/article/21917665/most-americans-experience-digital-eye-strain-from-overexposure-to-computers.

ŠANDOVÁ, Hanka, 2024. *Vyjádření do diplomky* [online]. [cit. 2024-02-25]. Osobní komunikace [E-mailová zpráva].

THE SIERRA CIRCUITS TEAM, 2021. *What is the Use of a Decoupling Capacitor* [online]. [cit. 2024-02-27]. Dostupné z: https://www.protoexpress.com/blog/decoupling-capacitor-use/.

TOCHÁČEK, Daniel, 2015. *Využití edukačně robotických sad ve vzdělávacím procesu na základních a středních školách*. Praha. ISBN 978-80-7603-264-4. Dis. pr. Univerzita Karlova, Pedagogická fakulta.

VAŇKOVÁ, Petra, 2019. *Robotické programovatelné hračky ve výuce*. Praha: Univerzita Karlova, Pedagogická fakulta. ISBN 978-80-7603-264-4.

ZIGBEE ALLIANCE, 2013. *ZigBee Specification FAQ* [online]. [cit. 2024-05-20]. Dostupné z: https://web.archive.org/web/20130627172453/http://www.zigbee.org/Specifications/ZigBee/FAQ.aspx.

,

Prohlašuji, že při tvorbě této práce byly nástroje umělé inteligence využity výhradně jako podpora pro stylistické úpravy, opravy pravopisu a návrhy textových formulací. Tyto nástroje byly používány způsobem respektujícím všechny platné předpisy a pravidla, včetně Etického kodexu Univerzity Karlovy. Práce přitom zachovává originalitu a integritu mé vlastní tvorby.