Model křižovatky pro výuku informatiky na střední škole

Bc. Horváth Aleš

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Aleš Horváth**
Osobní číslo:	**A21133**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Forma studia:	**Prezenční**
Téma práce:	**Model křižovatky pro výuku informatiky na střední škole**
Téma práce anglicky:	**An Intersection Model for Teaching Informatics in a Secondary School**

Zásady pro vypracování

1. Popište využití modelů při výuce informatiky na středních školách.
2. Navrhněte model křižovatky řízený zvoleným mikropočítačem.
3. Uvedený návrh hardwarově realizujte.
4. Vytvořte programové vybavení pro použitý mikropočítač, které umožní studentům vyvíjet vlastní programy pro řízení modelu křižovatky.
5. Zpracujte sadu řešených ukázkových úloh pro uvedený model.
6. Vypracujte výukovou prezentaci o vytvořeném modelu křižovatky.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. BENEŠ, Pavel. Automatizace a automatizační technika: prostředky automatizační techniky. 5., rozš. a aktualiz. vyd. Brno: Computer Press, 2014. ISBN 9788025137475.
2. CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.
3. LADMAN, Josef. Elektronické konstrukce pro začátečníky. Praha: BEN – technická literatura, 2001. ISBN 80-730-0015--6.
4. MCCONNELL, Steve. Dokonalý kód: umění programování a techniky tvorby software. Vyd. 1. Brno: Computer Press, 2005, 894 s. ISBN 802510849x.
5. PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300--110-1.
6. WHITE, Elecia. Making embedded systems. Sebastopol: O´Reilly, c2012. ISBN 9781449302146.

Vedoucí diplomové práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 26.05.2023

Aleš Horváth, v. r.
podpis studenta

ABSTRAKT

Obsahem diplomové práce je navržení modelu křižovatky v podobě výukového modelu pro žáky střední školy ve vyučovacích hodinách informatiky. Model se skládá z několika částí, jakými jsou programovatelné semafory a plastové části pro uchycení semaforu, díky kterým si žáci mohou sestavit křižovatku dle jejich představ. O celou funkčnost semaforů se stará mikrokontrolér Arduino. Pomocí řídicí jednotky je možné zprovoznit libovolné semafory s následným oživením kompletního modelu křižovatky. V práci se nachází několik řešených úloh na základě, kterých je žák po úspěšném postupu návodu schopen zrealizovat funkční model. Po absolvování všech řešených úloh a pochopení funkčnosti, má student možnost navrhnout svou vlastní koncepci modelu s úspěšným zapojením komponent včetně naprogramování řídicí jednotky pro správnou funkci světelné signalizace křižovatky.

Klíčová slova: model, křižovatka, informatika, výuka, arduino, semafor, řídicí jednotka

ABSTRACT

The goal of the thesis is to create a traffic junction model that can be used as a teaching tool in computer science classrooms for secondary school students. With the help of the model's various components, including programmable traffic lights and plastic parts for installing the lights, students can construct the intersection in accordance with their own ideas. An Arduino microcontroller manages all of the traffic light's functioning. Any traffic light can be made to function with the controller's assistance by animating the intersection's full model thereafter. The learner is expected to realize the functional model in this work after successfully completing the instructions based on a number of solved issues. The student gets the chance to develop their own model concept with successful component wiring and programming the control unit for the correct operation of the junction traffic light after finishing all the solved difficulties and comprehending functionality.

Keywords: model, crossroad, computer science, teaching, arduino, traffic light, control unit

PODĚKOVÁNÍ

Tímto písemným způsobem bych chtěl v první řadě poděkovat svému vedoucímu diplomové práce, panu Ing. Janu Dolinayovi, Ph.D. za jeho veškeré podněty pro vylepšení konceptu práce, konzultaci a v neposlední řadě také za vstřícnost. Rovněž bych chtěl poděkovat panu Ing.Mgr. Michalu Sedláčkovi, Ph.D. za prvotní nápad tématu diplomové práce. Dále patří poděkování především mým rodičům za veškerou podporu během studia a celé mé rodině.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

S postupným vývojem technologií se vyvíjí i pohled na samotnou výuku. Učitelé se snaží zaujmout žáky pomocí výukových modelů, na kterých ukazují danou problematiku. Z hlediska informatického myšlení se na trhu nachází nejrůznější programovatelné modely, díky kterým mají žáci možnost lépe pochopit probíranou látku a vyzkoušet si, zda jimi vytvořený software je funkční.

Provoz na silnicích je čím dál větší, avšak s tím je spojeno i více dopravních nehod. Tyto nehody bývají častěji zaviněny lidským faktorem, který se ovlivnit do určité míry nedá. Díky technologiím je již v dnešní době většina věcí řízena digitálně. Takovým způsobem jsou řízené i světelné křižovatky. Nehody se na nich stávají v častějším případě díky špatnému rozhodnutí člověka. Pokud mají žáci k dispozici model křižovatky, mohou si vyzkoušet takový model naprogramovat, ale také pochopit její samotný princip fungování, tak aby v budoucnu se lépe mohli vyvarovat řidičské chybě. Žák si tak rozvíjí zároveň více znalostí. První z nich je naučná část programování a druhou je samotné lepší pochopení provozu.

Cílem této diplomové práce je navrhnout a zrealizovat výukový model křižovatky pro střední školy, která bude sestavena pomocí jednotlivých komponent, ze kterých se křižovatka skládá. Křižovatku je možné sestavit na základě kreativnosti žáka.

V teoretické části je práce zaměřena na popis využití modelů ve výuce informatiky, kde je popsán i dosavadní pohled na stav výuky informatiky na základních a středních školách a požadavky na nynější výuku. V další části je práce věnována popisu používaných modelů ve výuce informatiky. Dalším obsahem jsou použité komponenty pro realizaci včetně zvolené řídicí jednotky a jejich detailnější popis. Teoretická část je zakončena obecným popisem programu, ve kterém probíhal návrh a vývoj hardwaru.

V praktické části disponuje práce návrhem konstrukce a hardwaru modelu křižovatky, samotnou realizací, vývojem křižovatky a jednotlivých komponent, jakými jsou silnice, semafory pro auta, semafory pro chodce a okolí křižovatky. Žák má možnost si libovolně tyto komponenty sestavit a následně s asistencí učitele zkontrolovat zapojení veškerých komponent. Každý semafor lze zvlášť naprogramovat dle potřeby a situace navržené křižovatky. Dále v praktické části je popsáno základní programové vybavení, díky kterému žáci mohou získat základní znalosti pro zprovoznění křižovatky. Poslední kapitolou je sada řešených

úloh pro studenty. Sada se skládá od nejjednodušších úloh až po nejsložitější, aby žák pochopil základy a princip funkce a realizace programovatelné části. Poté již je na každém žákovi, jakou křižovatku si chce dle vlastní představivosti sestavit a následně zprovoznit.

I. TEORETICKÁ ČÁST

1 VYUŽÍTÍ MODELŮ VE VÝUCE INFORMATIKY

Informatika patří v současné době mezi obory s největší rychlostí vývoje. Informační a digitální technologie jsou zásadní věcí vzdělávání nejen v osobním životě, ale i v oblasti profesního vývoje. Na tuto skutečnost reagovalo ministerstvo školství, mládeže a tělovýchovy, které provedlo revizi Rámcového vzdělávacího programu (RVP) pro základní školy. Zásadními změnami bylo zvýšení časové dotace předmětu informatika a následně byl upraven i jeho obsah. Momentálně je zaměřeno na rozvoj informatického myšlení, kde si klade za cíl naučit žáky schopnosti strukturovaného myšlení. U této schopnosti se pracuje například na kladení důrazu na důvěryhodnost informací a práce s nimi. Dále je také zaměřena na kritické myšlení, které se odráží v analýze problému, určení jeho podstaty a návrh hledaného řešení.

1.1 Pomůcky v hodinách informatiky

Informatika je z velké míry propojena s matematikou. Pro lepší vysvětlení problematiky se ve velké míře používají pomůcky vhodné k rozvoji matematických představ, algoritmizace a aktivitám s informatickým pozadím. Na trhu nalezneme i mnoho učebnic a učitel tak má více možností výběru, kterou cestou chce jít a žáky vzdělávat. Učebnice jsou doplňovány i o CD se softwarem soužící pro doplnění učebnice s názornými ukázkami. Mezi digitální učební pomůcky patří hlavně softwarové vybavení každého PC, které je nedílnou součástí výuky informatiky z jakéhokoliv pohledu. Dále se může jednat o programovatelné učební pomůcky, kterými je myšlena oblast robotiky. Na trhu je k dispozici více výrobců, kteří se neustále zabývají jejich vylepšování a přináší v novějších verzích více periferií například senzory a snímače. [1]

1.2 Virtuální realita

Hovoří se také o virtuální technice, díky ní se žák může dostat do plně virtuálního prostředí pomocí VR brýlí. Ve virtuálním světě se dokážeme pohybovat, dívat se všemi směry s dalšími periferiemi umístěných na těle jakou jsou rukavice nebo ovladače, díky kterým můžeme dokonce manipulovat s předměty. Může se také jednat o rozšířenou realitu, kdy vkládáme různé objekty do reálného světa a díky ní si můžeme představit, jak by mohla vypadat budoucí věc či objekt na určitém místě, na který bychom chtěli předmět umístit. Z virtuální

reality si můžeme odnášet nové poznatky, které by se v reálném životě mohly horším způsobem realizovat. [2]

1.3 Pohled na stav výuky informatiky a informačních a komunikačních technologií na ZŠ

Z pohledu výuky na středních školách neexistuje stejný odborný názor z pohledu učitelů na výstupní standard výkonu žáka, při dokončení základní školy. Od roku 2005 vešel v platnost dokument s názvem „Rámcový vzdělávací program“. RVP obecně tvoří závazný rámec pro tvorbu školních vzdělávacích programů škol všech oborů od předškolních až po střední vzdělání. RVP ZV pro výukovou oblast ICT (informatiky) je velmi široká a hodinová dotace je pro první stupeň ZŠ schválena na 2 hodiny týdně a pro druhý stupeň je nastavena na 4 hodiny týdně. Mnoho škol přidává hodiny z disponibilní výuky. Z hlediska oblasti informatiky (ICT) je možné říci, že některé základní školy dávají oblasti informatiky větší či menší prioritu. [3]

1.4 Pohled na stav výuky informatiky a ICT na SŠ

Žáci na střední školu přichází z různých základních škol. Tudíž je logické, že samotní žáci mají rozdílné znalosti a dovednosti, se kterými přišli ze základní školy. V oblasti informatiky si často žáci osvojují poznatky i mimo školu. Neřízeně a formou „samouka“ z důvodu toho, že v dnešní době je digitální svět přirozenou součástí života mládeže a dětí. Každá střední škola musí již od prvního ročníku svých oborů flexibilně reagovat a zařazuje do oborů předmět informatiky (ICT). Tento předmět jako každý jiný má za cíl sjednotit vstupní úroveň studentů a poté žáky posunout v problematice oblasti informatiky a dovednostech. [3]

1.5 Základní východiska a myšlenky revizí ICT kurikula

Během 19 let zůstávala jako jedna z posledních vzdělávacích oblastí, oblast informační a komunikační technologie beze změny v Rámcovém vzdělávacím programu pro základní, gymnázia i střední odborné vzdělání. V průběhu těchto 19 let se nashromáždilo od učitelů, odborných pedagogů a odborníků mnoho podnětů pro její úpravu. Úpravu a modernizaci ICT kurikula a úkoly, které plynou ze Strategie digitálního vzdělávání, řeší Národní ústav pro vzdělávání již od měsíce května roku 2016.

V roce 2021 byl vydán ministerstvem školství, mládeže a tělovýchovy revidovaný Rámcový vzdělávací program pro základní vzdělávání. Revize si dávala za cíl modernizovat dosavadní obsah vzdělávání, aby odpovídala potřebám a dynamice 21. století. Nový rámcový vzdělávací program základní vzdělávání vkládá vzdělávací oblast rozvoj digitální gramotnosti žáků, Informatiku a řadí je na úroveň klíčové kompetence.

Školy již mohou začít vyučovat dle Školního vzdělávacího programu, který je upraven v souladu s již revidovaným Rámcovým vzdělávacím programem základního vzdělávání od 1. září 2021. Je tu i hranice omezení kdy jsou školy nuceni zahájit výuku na základě úpravy ŠVP a to nejpozději od 1. září 2023. Tyto údaje se vztahují na všechny ročníky prvního stupně základní školy. Pro druhý stupeň je nejpozdější termín zahájením výuky ve všech ročnících druhého stupně s datem od 1. září 2024. [4]

1.6 Rámcové vzdělávací programy

Obecně tvoří povinný rámec pro vytváření školních vzdělávacích programů škol ve všech oborech vzdělání, od předškolního vzdělání až po střední vzdělávání, včetně základního, základního uměleckého nebo jazykového.

Rámcové vzdělávací programy stanovují hlavně délku a daný obsah vzdělávání, které může být všeobecného či odborného zaměření dle studovaného oboru. RVP stanovuje i organizační uspořádání a profesní profil. Dále zde patří formy a v neposlední řadě konkrétní cíle. Jedním z nejdůležitějších částí jsou zejména podmínky průběhu a následné ukončování vzdělávání a podmínek pro správné vytvoření ŠVP. V odvětví RVP jsou brány v potaz i podmínky pro vzdělávání žáků, které mají speciální vzdělávací potřeby. Zde se také stanovují podmínky pro důležité personální, organizační, bezpečnostní a materiální podmínky včetně ochrany zdraví.

RVP vychází z nejnovějších poznatků, ať už se jedná o vědní disciplíny, jejich úplné základy včetně praktického využití má vzdělávání umožňovat. Jedním z nejdůležitějších bodů v poznatcích vzdělávání je odvětví samotné pedagogiky, jejíž nedílnou součástí je i

psychologie. U posledních dvou odvětví můžeme hovořit o jejich účinných metodách, které se postupně stále inovují v závislosti na změnách ve školách.

Tvorbu RVP zajišťují daná ministerstva, která tvoří řadu odborníků z praxe a věd, psychologie a samotné pedagogiky. [5]

1.6.1 RVP GYMNÁZIA

Dokument je určen pro vytváření ŠVP na čtyřletých a vyšších stupních víceletých gymnázií. Dále se vymezuje základní vzdělávací úroveň pro všechny budoucí absolventy, kterou musí gymnázia dodržovat ve svém ŠVP. Škola by měla žákům do budoucna dát rozhled, který využijí při své další životní etapě, kterou mohou rozvíjet v dalším pokračování studia na vysoké škole nebo v pracovním životě. V průběhu studia by měl žák získat schopnosti a vědomosti v různých odvětvích, na základě kterých může své vědomosti a poznatky prohlubovat v jeho celoživotní etapě vzdělávání. Žák by si měl v průběhu získávání nových poznatků a vědomostí zjišťovat důvěryhodnost informace a její pravdivost. RVP gymnázia předpokládá, že student si bude informace strukturovaně a systematicky zařazovat do souvislostí s předchozími poznatky. [6]

1.6.2 RVP STŘEDNÍ ODBORNÉ ŠKOLY

Jedná se o velmi obdobné pojetí vzdělávání jako u gymnázia. Nalezneme zde však změny, které se týkají hlavně rozdílu pojetí učiva. Na Střední odborné škole je učivo zaměřeno na daný obor, dle kterého se dál rozdělují jednotlivé předměty, které v mnoha případech spolu souvisí a studenti v nich mohou vidět určité souvislosti. Hlavním rozdíl můžeme vidět v tom, že střední odborná škola má za cíl připravit žáka do určité odbornosti, zatímco u gymnázia se spíše jedná o široký rozhled z mnoha oblastí. Střední odborná škola dává žákovi již určité úzké zaměření, ze kterého může následně student vycházet již v další pracovní etapě života nebo své znalosti může dále rozvíjet studiem na vysoké škole. Žák by se měl učit tvořivě zasahovat do prostředí, kterým je obklopováno a dále umět jednat v rámci situace, která se mu naskytne. [7]

1.7 Školní vzdělávací program

Jedná se o kurikulární dokument, který je sestavován pedagogickými pracovníky každé školy v České republice. Dokument dále podléhá schválení, které má v kompetenci ředitel příslušné školy. ŠVP musí být k dispozici komukoliv, jelikož se jedná o veřejný dokument školy. Je tvořen na základě RVP, kterých se musí daná škola držet, při samotném sestavování. [8]

2 MODELY A PLATFORMY PRO VÝUKU INFORMATIKY

Ve všech školních prostředích dochází k postupnému vylepšení nejrůznějších pomůcek, které se následně aplikují v daných hodinách. Na trhu můžeme naleznout různorodé modely, které se liší použitím. Najdeme jak výukové model například pro přírodní vědy, dějepisné až po technické předměty.

Jedná se o výukové modely, které může učitel aplikovat při vysvětlování vybraného učiva, kde je vhodné doplnit samotný teoretický úsek praktickou ukázkou s možností zapojení studentů. Pomůcky mají za úkol samotnému učiteli usnadnit vysvětlování probírané oblasti, aby žáci měli lepší teoretické znalosti. Žáci si určitým způsobem také odnáší z hodiny praktické dovednosti, díky kterým lépe pochopí probíranou látku. Následující kapitoly jsou věnovány popisu modelů a platforem vhodných pro výuku informatiky na středních školách.

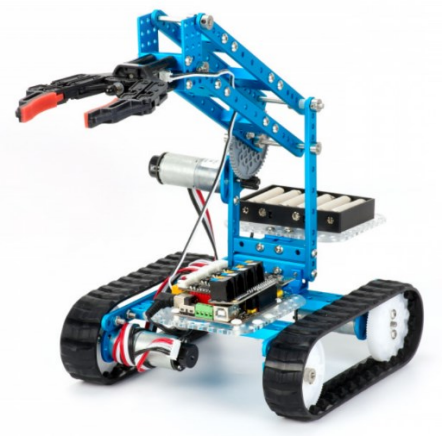
2.1 TinkerKit Braccio robot

Jedná se o plně funkční robotické rameno, která lze ovládat pomocí mikrokontroléru Arduino. K ramenu je možné připojit některé periferie jakými je kamera nebo solární panel. V balení se nachází kromě dílů pro sestrojení mechanické konstrukce také deska s konektory pro připojení servo motorů, díky kterým se robotické rameno otáčí a provádí různé úchopy. Na desce se nachází i napájecí konektor. Napájecí zdroj, který je součástí balení disponuje výstupním napětím 5V a elektrickým proudem 4A. Model je dle výrobce vhodný pro střední školy.[9]

Obrázek 1 TinkerKit Braccio robot[9]

2.2 Makeblock Ultimate Robot Kit 2.0

Oproti předchozímu robotickému rameni se jedná o vylepšeného robota s možností pohybu pomocí pásů, které umožňují robotovi jeho přesouvání na určité místo a uchopení daného předmětu pomocí chapadel. Hlavní řídicí deskou je deska MegaPi, která je založena na mikrokontroléru Arduino MEGA 2560. Ultimate robot kit lze programovat ve vývojovém prostředí přímo od výrobce mBlock nebo v prostředí Arduino IDE. Součástí balení je 10 předpřipravených projektů, kde každý z nich má jiný účel použití robota. Díky velkému množství součástek a modulů umožňuje studentům postavit si robota dle vlastních představ.[10]



Obrázek 2 Makeblock Ultimate Robot[10]

2.3 LEGO

Jedním z nejpoužívanějších výukových modelů, které najdeme na základních školách je od společnosti LEGO model Mindstorms. Koncept dané edukační sady dává možnost vlastní realizace žákovi, který si může sestavit programovatelné robotické zařízení. V sadě se nachází hlavní řídicí jednotka, která slouží k propojení ostatních komponent. K jednotce může žák připojit různé nízkonapěťové periferie, jako jsou motory, senzory a další komponenty. Lego Mindstorms nabízí rozšiřitelné části, které se dají případně dokoupit zvlášť. Veškeré elektronické periferie mají stejný konektor, který lze zapojit pouze jedním možným způsobem do řídicí jednotky. Tímto se eliminuje možnost špatného zapojení, které by mohlo vést k nesprávnému fungování připojené komponenty.

Dále se v balení programovatelné robotické stavebnice nachází samotný návod pro stavbu několika modelů. Když je zařízení sestavené, propojí se řídicí jednotka s PC a stačí jen nahrát požadovaný program do řídicí jednotky a následně má žák funkční model robota. Společnost má i svůj vlastní software pro vývoj programu. Program je volně dostupný na oficiálních stránkách výrobce, který se po úspěšném stažení nainstaluje a je připraven k použití. Od prvotní vydané verze prochází stavebnice vývojem, i samotný software. Můžeme tak naleznout více druhů softwaru pro naprogramování robotické stavebnice. [11]

Dalším z řady vzdělávací pomůcky od výrobce LEGO je model SPIKE. Od svého předchozího zmíněného modelu se jedná o verzi, která je určena pro žáky 6. až 8. ročníků základních škol. Stejně jak u modelu Mindstorms nalezneme v balení programovatelnou jednotkou, s možností zapojení periferií, jakou jsou senzory vzdálenosti, síly nebo rozpoznání barvy včetně motorů.[12]

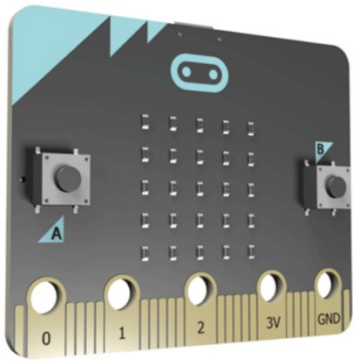
Obrázek 3 LEGO Mindstorms[13]

Stavebnice LEGO Mindstorms a Spike jsou pro žáky přínosné. Výrobce propojil samotné lego součástky s elektrotechnickými prvky včetně řídicí jednotky, která je programovatelná a studenti si díky tomu mohou naprogramovat vlastní program v závislosti na použitých komponentách. V každém balení se nachází návod včetně několika možných návodu na realizaci robota. Výrobce dodává ke stavebnicí více verzí programů pro žáky. S pozitivními názory se dají najít i negativní, které mohou u této stavebnice být například kompatibilita všech verzí s řídicími jednotkami. Ne vždy je možné napojit řídicí jednotku k dané verzi softwaru LEGO Mindstorms.

2.4 Micro:bit

Jedná se o mikropočítač, který nalezne své využití hlavně ve výuce informatiky. Studenti si mohou na tomto zařízení realizovat své vlastní programy. Pro jeho jednoduchost je jeho oblíbenost mezi studenty obdobná jako u arduina. Mikropočítač je vybaven 25 LED diody a dvěma tlačítky, které jsou díky mikroprocesoru individuálně programovatelné studentem dle jeho libosti. Programovatelná deska je vybavena i bezdrátovým rozhraním bluetooth pro komunikaci mobilními zařízeními. Díky jeho malé velikosti lze zařízení přenášet i v kapse. Deska je široká 5 cm a dlouhá 4 cm.

Micro:bit je vhodný pro děti od 7 let. Kromě tlačítek a LED diody je deska vybavena kompasem a akcelometrem, který slouží pro zjišťování pohybu. Mikroprocesor lze programovat čtyřmi různými úrovněmi a to od jednoduchého programování přes microPython, Javascript nebo C++. pro zprovoznění desky je nutné mít zařízení připojené na napětí 3V. V případě jakéhokoliv problému je možné použít tlačítko RESET pro uvedení mikropočítače do výchozího stavu. Pro připojení desky k počítači a následného nahrání programu je nutné použít kabel, s konektorem micro USB, který je hojně používán u mobilních zařízení. Jeho využití se nachází ve výuce informatiky na základních školách, vyznačuje se jednoduchostí, díky čemuž je mezi žáky oblíben. [14]



Obrázek 4 Micro:bit [14]

2.5 Raspberry

Obdobně jako Arduino nabízí Raspberry vývojové desky, přesněji jednodeskové počítače Raspberry Pi. Desky byly vyvinuté na území Spojeného království. Oproti Arduinu nabízí Raspberry na vývojové desce více konektorů pro připojení periferií jako jsou HDMI a USB-A a mnoho dalších. [15]

2.5.1 Raspberry Pi

Je to jednodeskový počítač a jeho výkon se dá srovnat se slabším stolním počítačem. Obsahuje HDMI výstup pro připojení k monitoru. K dalšímu připojení periferií jako je klávesnice nebo myš se nachází na desce USB port. Po uvedení na trh se stal populárnějším, než jeho výrobce předpokládal. Původně měl sloužit jako nástroj ve výuce informatiky na základních školách. Kvůli jeho nízké pořizovací ceně je používán ve více oblastech, jako je robotika, monitorování počasí a spousta dalších. Velikost je podobná velikosti kreditní karty. Je vyráběn ve více verzích stejně tak i jeho další nástupci. Rozdíly mezi jednotlivými model nejsou pouze v názvu, ale i v počtu použitelných pinů a operační pamětí RAM. [15]

2.5.2 Raspberry Pi 2

Jedná se o následníka modelu Pi. Byl uveden na trh v roce 2015. Jeho první verze byla osazena 900 MHz procesorem s 32bitovou architekturou se čtyřmi jádry s označením ARM-

Cortex-A7. Po pozdější inovaci byl nabízen v konfiguraci o stejné frekvenci, ale s 64bitovou architekturou ARM-Cortex-A53. [15]

2.5.3 Raspberry Pi 3

Třetím modelem v pořadí je typ „Pi 3“. Na desce se nachází stejný typ procesoru jako u předchozí verze po inovaci. Rozdíl je pouze u procesoru ve frekvenci, která byla 1,2 GHz. Celkově je procesor o 50% rychlejší oproti Pi 2. Mezi zásadní změnu patří, že se na Pi 3 nachází integrované WiFi a Bluetooth moduly pro bezdrátovou komunikaci. V roce 2018 byla verze uvedena na trh se stejným procesorem o větším taktu 1,4 GHz. [15]

2.5.4 Raspberry Pi 4

Byl uveden do prodeje v roce 2019. Jak u jeho předchozího modelu obsahuje stejný typ procesoru s identickou architekturou 64 bitů. Prodává se ve třech různých variantách, ktré jsou odlišné ve velikosti operační paměti – 1, 2 a 4 GB. Od roku 2020 je k dispozici i verze s operační pamětí o velikosti 8 GB. Napájení desky je realizováno pomocí USB-C konektoru. Model podporuje po připojení pomocí dvou micro-HDMI portů 4K rozlišení. [15]

2.5.5 Raspeberry Pi Pico

Typ Pi Pico je nejmenším modelem z řad Raspberry. Disponuje 26 multifunkčními piny. Velikostí se dá přirovnat k Arduinu Nano. Dále jeho vstupní napětí se pohybuje v rozmezí od 1,8 do 5,5 V stejnosměrného napětí. Na vývojové desce nalezneme dále i senzor teploty a spousty dalších funkcí jako například režim spánku. [15]

3 PROBLEMATIKA POUŽITÍ MODELU KŘIŽOVATKY VE VÝUCE

S realizací modelu křižovatky jsou spojeny i určitá úskalí při sestavování. V následujících kapitolách je práce věnována různým problémům, které mohou vzniknout během realizace například díky nedostatečné časové dotaci. Díky těmto problematikám je důležité, aby vyučující pedagog byl informován o možných situacích, kterým je nutné věnovat pozornost a počítat s nimi.

3.1.1 Časová náročnost pro realizaci

Při sestavování modelu křižovatky se studenty ve vyučovacích hodinách se dostáváme do různých situací. Tyto situace mohou být ovlivněny časovým úsekem vyučovací hodiny či hodin informatiky. Některé školy mají vymezen časový úsek dvou vyučovacích hodin výuky informatiky v jednom dni a po sobě jdoucích vyučovacích hodin. V tomto případě je možné přizpůsobit výuku vůči dostupnému času a lépe žákům vysvětlit základní principy sestavování. Žáci si tak ze dvou vyučovacích hodin odnáší lépe pochopené učivo a při vysvětlování je zde větší prostor pro diskuzi mezi studenty a učitelem.

Najdeme však i školy, které nemají dotovaný časový prostor informatiky dvou vyučovacích hodin po sobě jdoucích. Během 45 minut je potřebné si s žáky osvojit dosavadní znalosti a následně s žáky navázat na další probíranou látku. V případě realizace křižovatky zabere žákům určitý čas samotné sestavení křižovatky a následné zapojení veškerých komponent. Poté učitel se studenty může realizovat část programování. Dostáváme se tak do situace, kdy není v takové situaci stejný prostor pro rozsáhlejší diskuzi se studenty.

V každém případě je důležité přizpůsobit rozsah a probíranou látku dané hodině, aby žáci odcházeli z hodiny informatiky s ucelenými poznatky z probírané problematiky oblasti. Nabízí se zde i pojetí skupinové výuky, kdy si může učitel rozdělit studenty do jednotlivých skupin s určitým počtem žáků. Studenti si tak mohou navzájem předávat větší množství informací a pomáhat si při nedostatečném pochopení vysvětlené látky učitelem. Žáky si touto formou rozvíjí sdílet ostatní názory a pracovat v kolektivu. Poté si žáci mohou rozvrhnout náplň práce, tak aby každý žák ze skupinky měl určitý úkol, který má v rámci sestavování modelu splnit a úspěšně se tímto způsobem dostat do finálního vzhledu křižovatky.

3.1.2 Možnosti rozvoje kreativity žáků

Jelikož se křižovatka skládá z více komponent, vytváří se zde možnost využití další schopnosti. Při sestavování může využít tvořivosti. U každého studenta je tato schopnost rozdílná a individuální. Po pochopení problematiky a naučení se ovládání základním principům chování křižovatky a jeho programování s ní spojené, může žák inovovat dosavadní řešení a navrhnout si své alternativní řešení. Nové inovované řešení může být přínosem pro ostatní spolužáky nebo dokonce i pro samotného učitele.

Při zjištění zájmu studenta ze strany učitele. Může být k takovému žákovi přistupováno individuálně na základě jeho dosavadních schopností. Vyučující po domluvě může zadat studentovi samostatnou práci na sestavení vlastní varianty křižovatky pomocí výukového modelu. Učitel může mít do budoucna další zrealizované řešené křižovatky, které může být přínosem pro další následující ročníky.

Nakonec modelové zařízení může být i ukázkou „architektonického“ a „modelářského“ vzhledu a nabýt tak na atraktivitě ku prospěchu studentů, kteří tento systém nebo model plně využívají.

3.1.3 Možnosti rozvoje žáků v oblasti 3D tisku

Po úspěšném sestavení křižovatky a naučení dovedností pro správnou realizaci komponent, se studenti mohou dostat do situace, kdy chtějí křižovatku určitým způsobem vylepšit o určité komponenty. Touto možností může být model osvětlení pro silnici, parková lavička a spoustu dalších dílů, které mohou dělat křižovatku více detailnější.

Tyto díly mohou být vyrobené pomocí 3D tiskárny. Na určitých webových stránkách, které jsou zaměřeny na 3D tisk je možné naleznout již vymodelovaný komponent, o který máme zájem. V častých případech bývají takové stránky rozvíjeny komunitou uživatelů, kteří jsou nadšenci pro 3D tisk. Žák si po úspěšné stažení určitého typu souboru a předání souboru 3D tiskárně, je schopen vytisknout model dané věci. Pokud nelze naleznout na internetu danou komponentu, daný díl se musí vymodelovat na specializovaném softwaru určeném pro návrh dílu.

Těchto softwarů je na trhu více a většina z nich nabízí studentské licence, které jsou bez poplatků, a jejich použití je zdarma. Pokud jeví student zájem o další rozvíjení znalostí v oblasti 3D tisku, může si vybraný a dostupný program s kritérii stáhnout a nainstalovat na svůj osobní počítač.

Pokud učitel disponuje znalostmi této oblasti, může být studentům nápomocen při vytváření dalších komponent.

3.1.4 Variabilita křižovatky

Modulární systém světelné křižovatky potažmo křižovatek je modulární proto, protože ze základních deskových komponentů se dá vytvořit větší množství různých typů křižovatek. Tyto křižovatky budou a jsou osazeny sloupky se semafory. Jeden sloupek může být osazen až čtyřmi semafory a to podle konkrétního typu sestavené křižovatky. Zde je to na úvaze studenta, který tuto křižovatku staví a bude ji i osazovat výše zmíněným světelným značením. Semafory jsou dvojího typu. Jeden typ je pro automobily a druhý pro chodce.

V setu všech komponent lze nalézt více typů semaforů. Semafory jsou rozdílné pro každý druh křižovatky a jejich použití záleží na zvoleném schématu křižovatky. Výukový model obsahuje semafor se světelnou signalizací pro odbočovací pruh. Pokud si však žák zvolí jednodušší schéma křižovatky, může použít semafor, který obsahuje pouze jednu světelnou signalizaci pro jízdu v přímém směru jízdy. Oba typy semaforů obsahují také světelnou signalizaci pro chodce. Co se týče desek, které demonstrují v modelu silnici nebo okolní terén. Tyto desky se mohou libovolně vedle sebe skládat a vytvořit tak další variantu modelu.

3.1.5 Použitelnost modelu

Studenti budou mít možnost se prakticky seznámit s funkčním ovládacím programovým systémem, který mohou nejen využívat, ale ti dobří nebo i nejlepší dále vylepšovat jeho základní verzi pod pedagogicko-výchovným dohledem příslušně vzdělaného učitele, instruktora nebo pedagoga či pedagogického pracovníka. Po zkušenostech s tímto modulárním systémem, modelem nebo zařízením, mohou žáci dávat návrhy i na jeho vylepšení z více aspektů.

V prvním aspektu se může jednat o desky silnic a okolních cest např. (větší počet desek, jiný účelnější tvar, jiný vzhled). Také je možnost realizace jiných semaforů, které se mohou lišit vzhledem i tvarem.

Druhým aspektem je hardwarový návrh přímo na jednotlivé elektrotechnické součástky – přepínače, vypínače. Zde je ovšem již důležitá odborná znalost elektrotechniky žáků a vyučujícího pedagoga.

3.2 Dostupnost výukových modelů křižovatky na trhu

Při procházení různých webových portálů se můžeme setkat s „kutilskými“ projekty, které se zaměřují na modelování křižovatky. Někteří nadšenci, kteří vytváří tyto modely, poskytují veškerý postup práce a realizace s potřebnými komponenty pro úspěšné dokončení modelu.

Aktuálně se na trhu nenachází komerční model, který by byl variabilní s následnou možností rozšíření. Můžeme však naleznout modul ve tvaru semaforu s vývody pro zapojení k platformě desek Arduino viz. obrázek č.5.

Obrázek 5 Modul LED semaforu [16]

Výše uvedený model neobsahuje pouzdro a pro lepší demonstraci semaforu je potřebné si okolní pouzdro navrhnout a vyrobit.

3.3 Prostředky pro výrobu modelu

V následující kapitole jsou popsány prostředky, které byly použity při samotném návrhu modelu křižovatky, pomocí kterých byl navržen hardware řídicí jednotky s dalšími částmi včetně semaforů nezbytných pro úspěšnou realizaci.

3.3.1 Deska plošných spojů

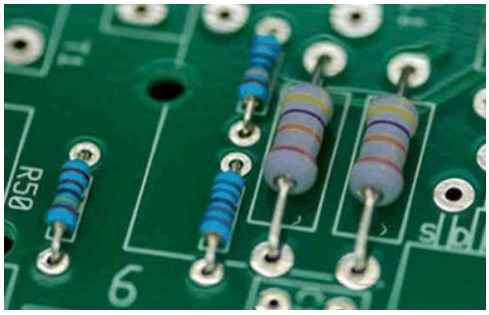
Deska plošných spojů neboli „DPS“, je deska vyrobená z izolačního materiálu a následně je z jedné nebo obou stran vybavena plochými vodiči. Izolační materiál, který tvoří základ desky, musí mít co nejlepší mechanické i elektrické vlastnosti. Co se týče mechanických vlastností, tato vlastnost zahrnuje pevnost desky a její ohnivzdornost. Nejčastějším používaným izolačním materiálem je laminát ze skelné tkaniny sycený epoxidovou pryskyřicí. Tento typ se nazývá „FR4“ a následně osazena plochými vodiči nejvíce z vodivého materiálu mědi z důvodu dobré vodivosti a jejich vlastností. V České Republice se pro základní izolační materiál můžeme setkat s názvem Cuprexit, původně se jednalo o obchodní název materiálu, který vyráběl podnik KABLO Bratislava.[17]

FR1	Papír nasycený fenolovou pryskyřicí
FR2	Papír nasycený fenolovou pryskyřicí (Pertinax)
FR3	Papír nasycený epoxidovou pryskyřicí
FR4	Tkanina ze skelných vláken sycená epoxidovou pryskyřicí
FR5	Tkanina ze skelných vláken sycená epoxidovou pryskyřicí

Tabulka 1 Označení materiálů pro DPS [17]

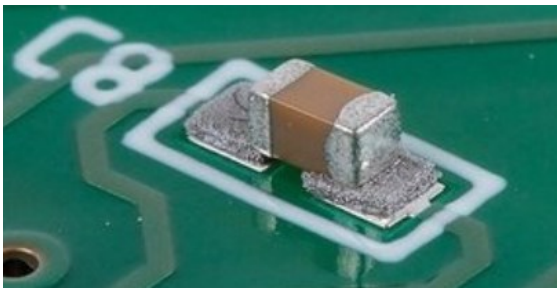
Následně se desky plošných spojů osazují elektronickými součástky dvěma technologiemi (způsoby):

První metodou je způsob THT (zkratka z angl. slov Through-hole technology) – tento způsob je v dnešní době používán primárně při výrobě v „domácích podmínkách“. Vyznačuje se tím, že drátové vývody elektronických součástek jsou skrz vyvrtané otvory v desce a z druhé strany jsou zapájeny pomocí pájecího cínu.[18]



Obrázek 6 Rezistory osazené technologií THT [18]

Druhý způsob osazení je s označením SMT (zkratka z angl. slov surface mount technology). Tento způsob se odlišuje povrchovou montáží elektronických součástek. Součástky nemají drátové vývody, ale místo nich „plošky“, díky kterým jsou připájeny přímo na plochý spoj vodiče desky. SMT součástky jsou oproti obyčejným s drátovými vývody o dost menší a jsou tak používány ve všech elektronických zařízení z důvodu menší prostorové náročnosti. [19]

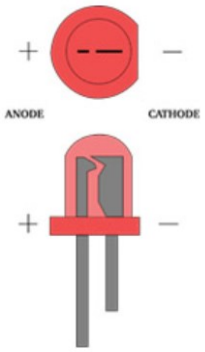


Obrázek 7 Kondenzátor osazený technologií SMT[19]

3.3.2 LED dioda

LED dioda je polovodičová polarizovaná elektronická součástka. Její název vychází z anglické zkratky (Light Emitting Diode). Její schopnost emitování světla je založena na optoelektrickém jevu diody. Tento jev nám při průchodu elektrického probudu LED diodou emituje světlo. V předešlých letech nahradila LED dioda obyčejnou žárovku, přičemž její výhody oproti klasické žárovce jsou více než výborné. Pracují již v nízkých hodnotách napětí a proudu. Z dostupných údajů se jedná o elektronickou součástku, která má zhruba 7x menší spotřebu elektrického proudu oproti jeho předchůdci žárovce. Dalšími výhodami je její účinnost, tato hodnota je 10x větší. Jsou vyráběných v různých barvách, velikostech a výkonech. Tuto diodu můžeme najít v televizích a různých zobrazovacích displejích, jakými jsou např. (mobil, tablet) a spoustu dalších zařízení. Disponuje dvěma vývody, prvním z nich je anoda,

která se připojuje na + v elektrickém obvodu (zdroje napájení). Druhý se nazývá katoda, oproti anodě bývá kratší a připojuje se na – (zem) v elektrickém obvodu. [20]



Obrázek 8 LED dioda [20]

3.3.3 Rezistor

Jedná se o pasivní elektronickou součástku, která primárně slouží k implementaci elektrického odporu v obvodu. V elektronických obvodech jsou používány pro snížení toku (velikosti) elektrického proudu. Hodnota odporu rezistoru je znázorněna pomocí barvených „proužků“ kolem rezistoru. Jejich hodnoty jsou minimálně proměnlivé v závislosti na teplotě, čase a v neposlední řadě také provozním napětím. S rezistory se běžně můžeme setkat v elektrických obvodech (zařízeních), kde dělají důležitou funkci pro správné fungování elektrického obvodu. Velikost odporu rezistoru lze pomocí barevných proužků vypočítat. Dále tato elektronická součástka se vyrábí v odporových řadách. Řady se dělí do sedmi kategorií, přičemž jejich názvy jsou následující (E-3, E-6, E-12, E-48, E-96, E-192). Číslo za písmenem „E“ určuje kolik je hodnot dané dekádě (řadě). Každá kategorie má toleranční limity. Proto jsou rezistory rozděleny do více odporových řad. Tolerance jsou následující. [21]

E-3	E-6	E-12	E-24	E-48	E-96	E-192
40% ±	20% ±	10% ±	5% ±	2% ±	1% ±	0,5% ±

Tabulka 2 Odporové řady [21]

Pokud budeme mít odpor o velikosti 100k Ω v odporové řadě E-12 bude mít odpor toleranci 10% ±, těchto zjištěných údajů si můžeme odvodit jeho reálnou hodnotu, která se může pohybovat v rozmezí 90k Ω až 110k Ω.

Obrázek 9 Rezistor [22]

3.3.4 Konektor RJ12 6P6C

Je nejrozšířenějším konektorem pro telefonní linky. Disponuje šesti sloty oproti jeho předchůdci RJ11, který jich má pouze čtyři. Konektory RJ11 a RJ12 jsou totožné a jejich rozdíl je pouze v tom, že konektor RJ11 má poslední 2 sloty volné. V dnešní době již není prakticky využíván a jeho dostupnost na trhu je čím dál více „horší“ z důvodu nepotřebnosti. Tento fakt je dán tím, že v dnešní době už telefonní linky nejsou populární a byly nahrazeny bezdrátovými technologiemi. Můžeme však naleznout případy, kdy tento konektor nalezneme i na nových zařízeních jakou jsou LEGO mindstroms ev3. Standart 6P6C nám udává údaj, že se v konektoru nachází 6 slotů pro plochý kabel se šesti žilami a všechny jsou zaplněné. Výhodou tedy je u typu konektoru RJ12 větší počet linek (o 2), pro předávání informací. Dále se můžeme setkat s označením 6P4C, 6P2C, 4P2C. Číslo před písmenem „C“ nám udává kolik kontaktů v konektoru je zapojených.

Pro ukázkový příklad si zvolíme označení 4P2C. Před písmenem C vidíme číslice 2 a tím tedy se jedná, že konektor má pouze zaplněné 2 sloty ze šesti možných. Z tohoto typu vychází konektor RJ45, který je nejrozšířenějším konektorem u síťových kabelů UTP (kroucené dvoulinky). [23]

3.4 Platforma ARDUINO

Jedná se o platformu, která je založena na jednoduše použitelné hardwaru a softwaru. Arduino nabízí produkty od vývojových desek různých modelů. Tyto desky jsou schopny číst a zároveň předávat signály. Platforma je „open source“ licence. Nabízí nám software, jehož zdrojové kódy jsou volně k dispozici. V komunitě je velice oblíben a díky tomu se neustále díky programátorům vyvíjí nové knihovny, které jsou určené pro různé senzory až po nejrůznější automatizační zařízení. Desky jsou dobře dostupné na trhu i z finanční stránky. Zařízení se pohybují v jednotkách stovek korun. Záleží na zvoleném typu desky. V dnešní době je zařízení velice oblíbeno mezi studenty, ale také pro domácí potřeby. Jejich

uživatelé si rádi například programují různá zařízení pro domácí automatizaci. Všechny desky Arduino jsou kompatibilní s grafickým vývojovým prostředím platformy. [24]

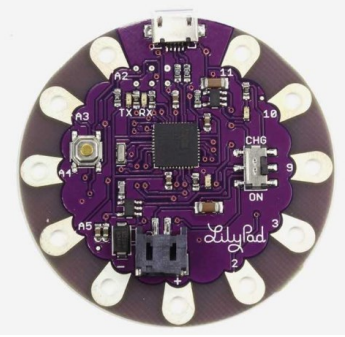
3.4.1 Historie

Projekt vznikl v Italském městě Ivrea v roce 2005, který si dával za cíl vytvořit jednoduchou prototypovací výukovou platformu pro studenty. Díky tomu umožní studentovi jednoduché používání a rychlý vývoj. [24]

3.4.2 Typy modulů desek Arduino

3.4.2.1 LilyPad Arduino

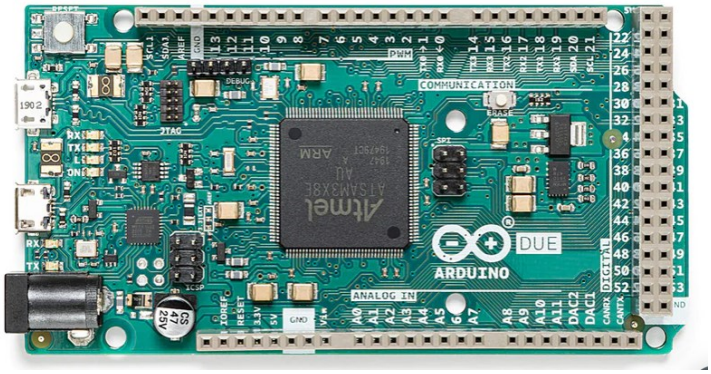
Díky jeho malé velikosti a kulatosti je určené pro projekty a přizpůsobena na nošení v e-textilu. Po celém obvodu desky se nachází otvory, které se dají využit i k uchycení mikrokontroléru k textilu. Lze jej tedy jednoduše přišít k textilní látce a používat jej například k vytvoření blikajícího svetru nebo čepice a více podobných textilních projektů. Model obsahuje 9 vstupních/výstupních pinů z nichž 4 lze použít pro pulzní šířkovou modulaci (PWM) a 4 jako analogové vstupy. Pro komunikaci mezi počítačem se na desce nachází micro USB konektor, díky kterému si můžeme naprogramovat mikrokontrolér desky, jehož název je ATmega32u4. V neposlední řadě se pro napájení celého obvodu vyskytuje konektor pro připojení baterie o hodnotě napětí 3,7V. Důležitou věcí je také resetovací tlačítko na levé straně desky. [25]



Obrázek 10 Deska LilyPad Arduino [25]

3.4.2.2 Arduino Due

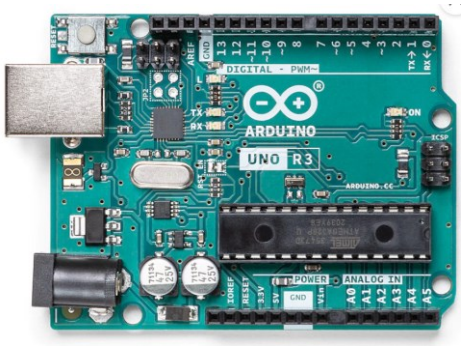
Daný model se vyznačuje jako první deskou od Arduina, která má v sobě 32bitový mikrokontrolér s jádrem ARM. Stejně jako u typu Arduino Mega se ve verzi Due nachází 54 digitálních vstupních/výstupních pinů, ze kterých je možné použít jako PWM výstupy a dalších 12 pro analogové vstupy, 4 UART porty, 2 digitálně analogové převodníky, resetovací tlačítko a tlačítko pro mazání. Napájecí konektor se nachází vedle micro USB konektoru pro propojení s počítačem. Výrobce uvádí napájecí napětí od 6V do 16V. Ideální a doporučená maximální hodnota je 12V. [26]



Obrázek 11 Deska Arduino Due [26]

3.4.2.3 Arduino Uno

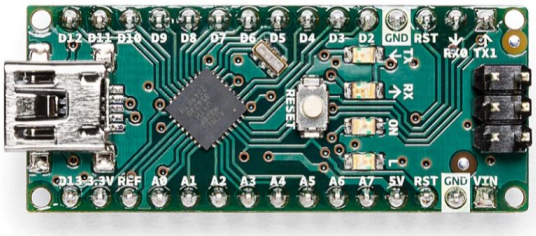
Srdcem této verze je mikrokontrolér s označením ATmega328P. Verze Uno je nejpoužívanějším modele na světě ze všech modelů Arduino. Pro programování na desce nalezneme 14 výstupních/vstupních digitálních pinů. 6 z nich je určeno pro analogové vstupy. Obdobně se nachází na desce resetovací tlačítko a USB-B konektor pro komunikaci. Provozní napětí mikrokotroléru je 5V. Napájecí napětí je v rozsahu 6-20V, přičemž 20V je maximální hraniční limit pro správnou funkčnost desky. [27]



Obrázek 12 Deska Arduino Uno [27]

3.4.2.4 Arduino Nano

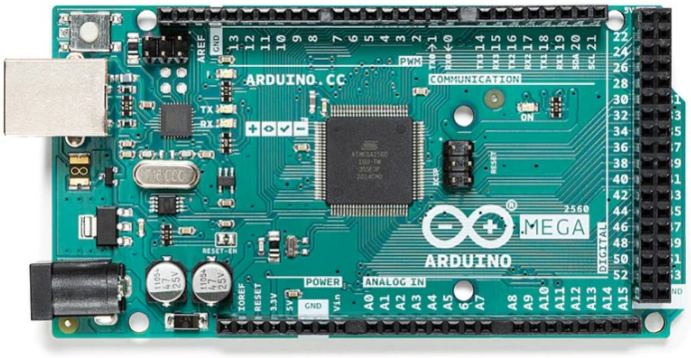
Oproti ostatním modelům se vyznačuje, tím že postrádá napájecí konektor. Napájen může tedy být buď mini-B USB kabelem, který slouží zároveň jako komunikační. Další možností je napájení pomocí zapojení pinů VIN (+) a GND (-). Hraniční maximální vstupní napětí u verze Nano je stejná jak u verze Uno tj. 6V-20V. Z celkových 14 digitálních vstupních/výstupních pinů lze 8 pinů použít pro analogové vstupní signály. Ze zbývajících 6 je možné využití jako PWM výstup. Mikrokontrolér na tomto modelu je ATmega328. [28]



Obrázek 13 Deska Arduino Nano [28]

3.4.2.5 Arduino Mega 2560

Jeho název je odvozen z názvu jeho mikrokontroléru, který nese označení ATmega2560. Z celé rodiny modelů Arduino se jedná o nejdražší desku, její cena se pohybuje ve vyšších jednotkách stovek korun. Na trhu můžeme naleznout kopie, které nám nabízejí totožné parametry i vlastnosti desky s originálním modelem, ovšem cena u klonu je třetinová. K dispozici má uživatel 54 digitálních vstupních/ výstupních pinů. 16 z nich je možné využít pro analogové vstupy a dalších 15 sloužící pro PWM výstup. Dále se nachází na desce LED dioda, která uživateli indikuje napájecí napětí. Komunikace probíhá pomocí USB-B konektoru. Pokud uživatel nahraje špatně fungující program a dostane se do nesprávně funkční části programu, může programátor použít resetovací tlačítko, které je k dispozici na desce. Napájecí napětí se opět pohybuje v rozmezí 6V až 20V. [29]



Obrázek 14 Deska Arduino Mega 2560 [29]

Pro realizaci modelu bylo nutné navrhnout desku plošných spojů, na které je uchycena řídicí jednotka Arduino Mega včetně konektorů pro připojení semaforů. Pro realizaci byl zvolen program Eagle, který byl ideální volbou na základě dřívější zkušenosti s programem.

3.5 EAGLE

Eagle je software pro automatizaci elektronického návrhu, tento program umožňuje návrhářům desek plošných spojů propojovat elektrotechnická schémata s následným umístěním součástek dle návrháře. Jeho název je zkratka z anglických slov (**E**asily **A**pplicable **G**raphical **L**ayout **E**ditor). Byl vyvinut německou společností CadSoft Computer GmbH v roce 1988.

V programu se nejdříve navrhne elektronické schéma pomocí zvolení dané komponenty. Pokud uživatel nemůže najít v knihovně danou elektronickou součástku, může si ji navrhnout. Když je schéma dle návrhu zhotoveno, přechází se do části, kdy se součástky rozmístí na desku dle uživatele a následně se mezi sebou propojují čárami (vodiči). Po kompletním návrhu desky je možné si daný vzhled vytisknout nebo převést do určitého formátu, ve kterém se data posílají výrobci pro následnou výrobu desky plošných spojů. Co se týče kompatibility s operačními systémy. Společnost nabízí software pro 3 operační systémy, kterými jsou Windows, Mac a Linux. Program Eagle nabízí pro osobní použití bezplatné stažení, avšak tato verze je dále omezena i určitými možnostmi jakými je např. počet realizovaných schémat, který je omezen na dvě. Mezi další patří počet vodivých vrstev desky, její

omezení je použití maximálně dvou vrstev. V neposlední řadě velikostní návrh desky, její maximální velikost plochy může být 80 cm^2.

Pro komerční využití je možné si zakoupit plnohodnotnou licenci. Licence není neomezená, a tak udává její společnost možnost zakoupení licence na měsíc, rok nebo 3 roky, dle volby zákazníka. Aktuální ceník licencí je dostupný na webu společnosti. Eagle je velice známý a v dnešní době hojně využívaný pro návrhy elektronických schémat a zařízení. Je zde také možnost 3D vizualizace navržené desky včetně jejich součástek. Návrhář tedy i vidí vzhled samotného produktu, před následnou výrobou. [30]

II. PRAKTICKÁ ČÁST

4 NÁVRH MODELU KŘIŽOVATKY

Kapitola se zabývá podrobným popsáním navrhnutých částí modelu křižovatky. Při vývoji se nejdříve navrhovala konstrukce modelu pomocí grafického návrhu jednotlivých komponent.

4.1 Konstrukce

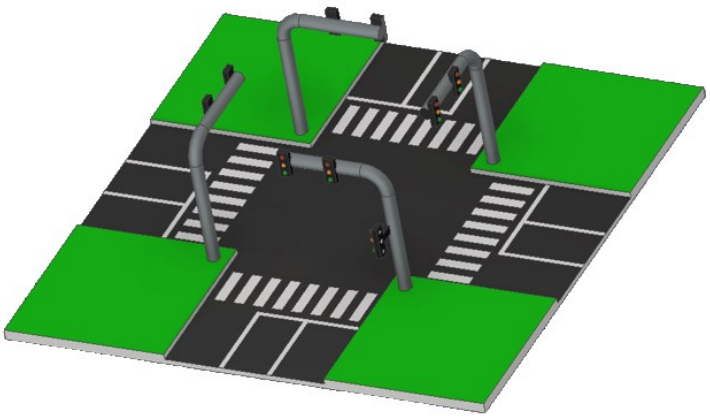
4.1.1 Tvar křižovatky

Křižovatka je tvořena z modulů, kterých je na celkové ploše desky možné využít 9. Při použití 9 modulárních desek jsou rozměry celé křižovatky 45 x 45 cm a má charakter čtvercového tvaru. [31]

4.1.2 Moduly desek

4.1.2.1 Grafický návrh

Veškeré komponenty byly navržené v programu AutoCAD sloužící pro modelaci součástek ve strojírenství až po automobilový průmysl. Oproti ostatním programům, kterých je na trhu nespočet nabízí i výstupní formát návrhu, který je možné následně předat 3D tiskárně a námi navrhnutou komponentu vytisknout. Celý koncept se skládá z mnoha součástí, které ve finální podobě tvoří celistvou verzi a na pohled realistický model dopravní křižovatky.



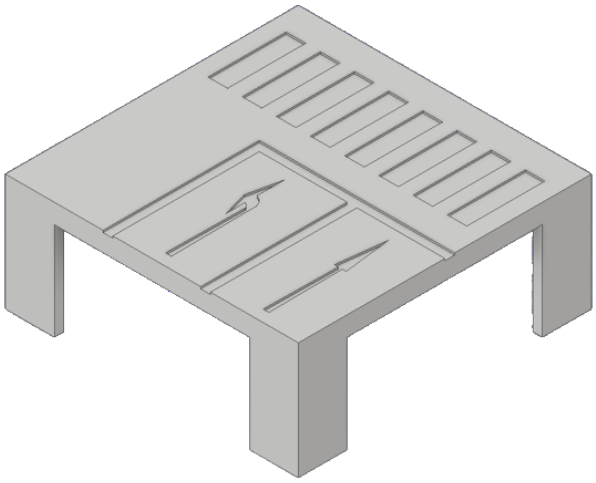
Obrázek 15 Návrh křižovatky

4.1.2.2 Podkladová deska pro model

Pro ukotvení všech komponent bylo nezbytné zvolit podkladovou desku, díky které budou moduly pevné a nepohyblivé pro lepší přehlednost a práci se zapojováním komponent. Na desce se pro systém uchycení nachází zarážky, do kterých se vloží rohy modulů desek (silnice a chodníky). Po vložení silnice nebo chodníku do zarážek již není možné pohybovat s modulem a stává se zafixovanou částí na podkladové desce. Díky podkladové desce je možné sestavenou křižovatku přenášet na libovolné místo bez nutnosti rozebrání a následného složení na novém místě.

4.1.2.3 Silnice křižovatky

Silnice je tvořena z několika částí. První částí je samotná deska silnice bez přechodu a pruhů. V místech kde se má nacházet přechod a odbočovací pruhy není plocha vyplněna materiálem. Tyto části jsou zvlášť vytisknuty a následně přilepeny na nevyplněná místa. Ve finální podobě při dolepení jízdních pruhů a přechodu je silnice hladká a zarovnána do roviny. Deska silnice je 15cm široká a 15 cm dlouhá. V rozích se nachází nohy, které jsou 6 cm vysoké. Slouží k uchycení do zarážek podkladové desky.



Obrázek 16 Silnice s odbočovacími pruhy

4.1.2.4 Chodník

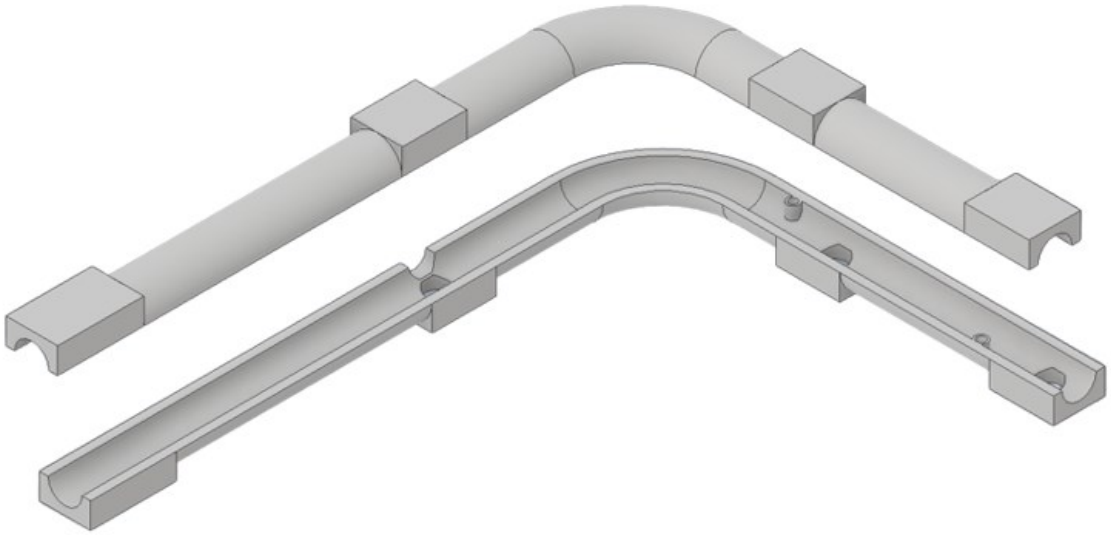
4.1.2.5 Přechod pro chodce

Další částí jsou vytisknuté panely obdélníkového tvaru značící přechod pro chodce. Pro vizuální efekt připomínající reálný přechod.

4.1.3 Návrh semaforu

4.1.3.1 Konstrukce semaforu

Semafor se skládá celkově ze dvou částí. První částí je stožár, na kterém jsou uchyceny jak semafory pro auta tak i pro chodce. Stožár je tvaru písmene „L“ o následujících rozměrech 120 x 98mm. Je rozdělen na 2 poloviny, které jsou spojeny díky dvěma kolíkům a otvorům které tvoří mechanismus zámku. Druhou částí jsou samotné semafory, které jsou ve dvou provedeních. Prvním typem je semafor pro chodce, který obsahuje pouze dvě LED diody. Druhým a zároveň posledním typem je semafor pro automobily, který je tvořen ze tří LED diod.

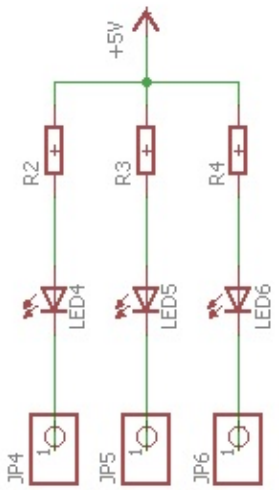


Obrázek 17 Konstrukce semaforu

Obrázek 18 Vytisknutý semafor pomocí 3D tiskárny

4.1.3.2 Vnitřní zapojení LED diod

Zapojení LED diod v samotném semaforu je realizováno pomocí společného vstupního napětí 5V. Maximální procházející proud, který může procházet LED diodou, se pohybuje u většiny běžných LED diod okolo 20 mA. Hodnota bývá proměnná a záleží na velikosti Led diody a na jeho výrobci, který tuto maximální hodnotu uvádí. Pro správné fungování LED diody a celé řídicí desky byl zvolen předřadný odpor o hodnotě 330 Ω. Procházející proud dosahuje hodnoty 9mA. Tato hodnota proudu Led diodou je dostatečná pro viditelnou signalizaci semaforu a zároveň je v bezpečných hodnotách pro správnou funkci LED diody dle hodnot výrobce. [32]



Obrázek 19 Zapojení LED diod

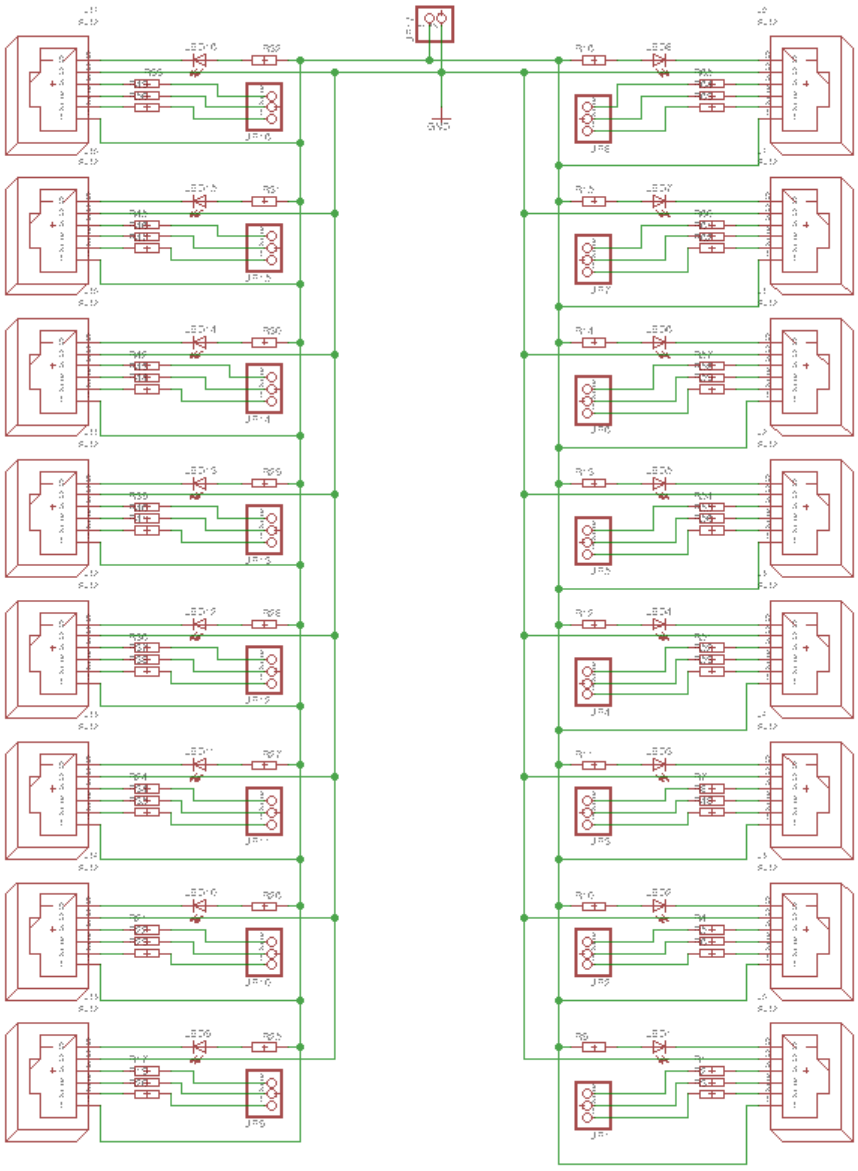
5 NÁVRH HARDWARU

Pro zapojení veškerých semaforů křižovatky bylo nutné vytvořit ucelený koncept řídicí jednotky s možností připojení konektorů od semaforů. K tomuto kroku bylo nezbytné navrhnout desku plošných spojů, která bude osazena mikrokontrolérem Arduino a konektory pro připojení až 16ti semaforů. [33]

5.1 Návrh DPS řídicí jednotky

Navrhnutá deska řídicí jednotky obsahuje otvory pro uchycení desky mikrokontroléru arduino Mega. Pro napájení celého obvodu slouží 2 piny, které jsou připojenu na napájecí piny mikrokontroléru +5V a GND. Pro lepší praktické využití a možné budoucí změny je propojení mezi jednotlivými LED diodami realizováno pomocí jumperů, díky nimž se může měnit vnitřní zapojení řídicích pinů z desky mikrokontroléru. Dále se na samotné desce nachází indikační dioda, která se při vložení protikusu konektoru rozsvítí a uživateli signalizuje, že semafor je správně zapojený. Před každým vývodem pinu se nachází rezistor pro bezpečné svícení LED diody. Pro přehlednost je každý port pojmenován písmenem od A do P. Žák při připojení daného semaforu, který je značen také názvy, má všechny potřebné informace pro správnou a funkční realizaci křižovatky.

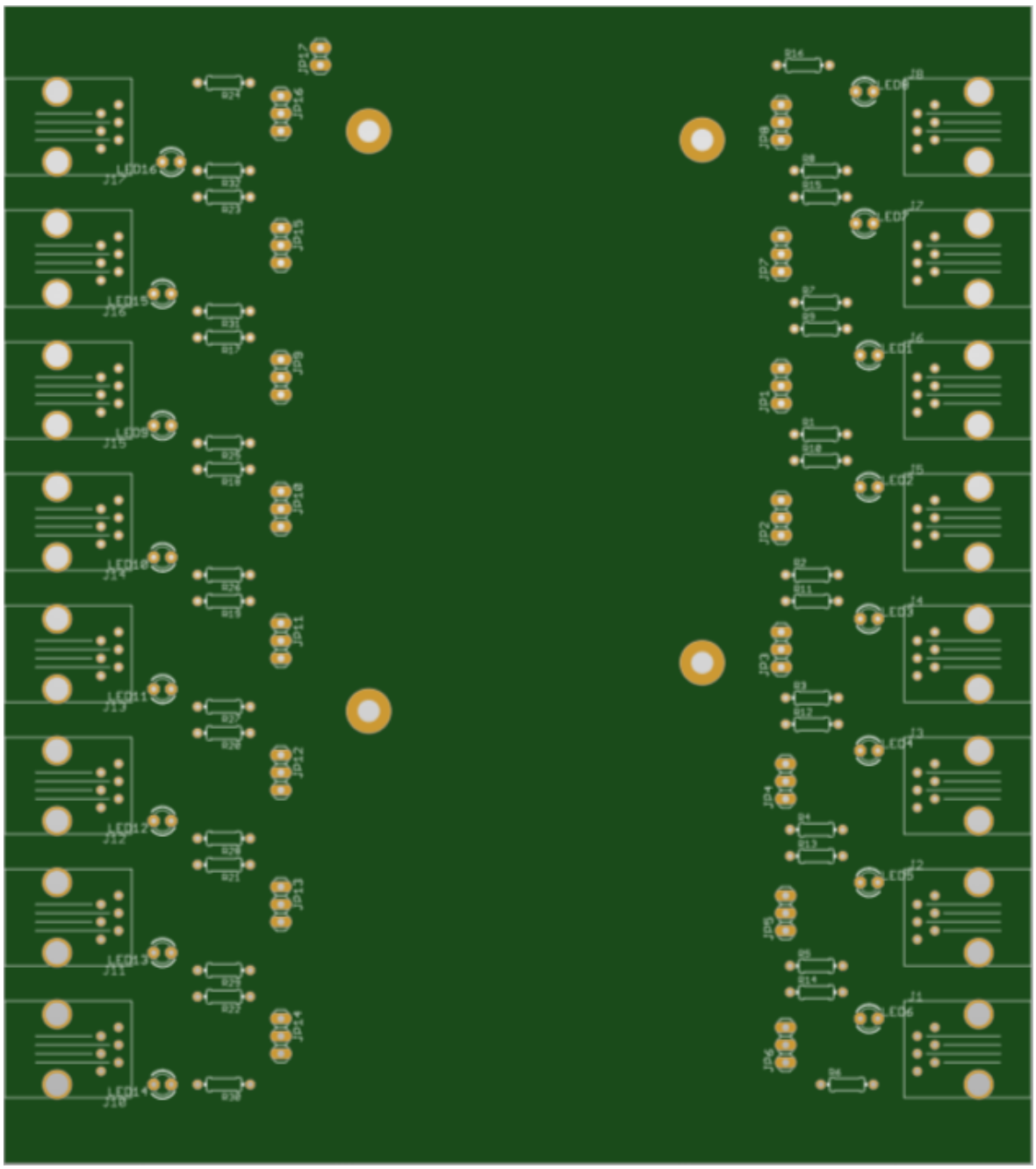
5.1.1 Schéma



Obrázek 20 Schéma řídicí jednotky

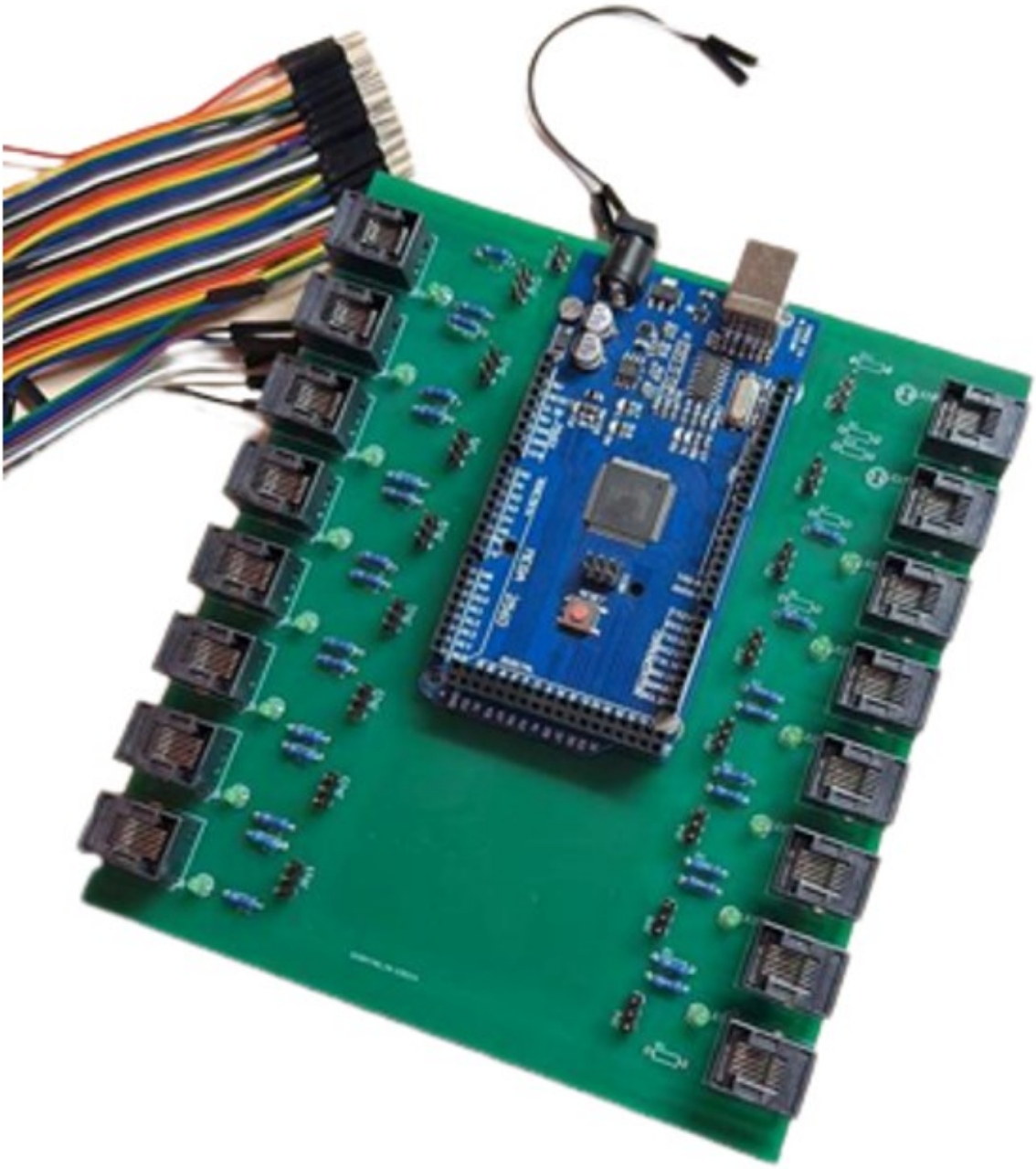
Na desce se nachází 2 konektory pro propojení napájení desky na 5V s arduinem. U každého konektoru, který slouží pro připojení semaforu, se nachází indikační dioda s předřadným rezistorem z důvodu ochrany LED diody. Při připojení konektoru dojde k uzavření elektrického obvodu a žák má díky tomu zpětnou vazbu, že semafor je zapojen správně. Dále se nachází u každého konektoru 3 vývody, které jsou propojený dále pomocí propojovacích kabelů s arduinem. Tyto vývody slouží k rozsvícení a zhasnutí LED diody. Před každým vývodem se opět nachází rezistor z důvodu správného průchodu elektrického proudu Led diodou. [34]

5.1.2 Design DPS



Obrázek 21 Design desky pro osazení

5.1.3 Vzhled DPS



Obrázek 22 Skutečný vzhled řídicí desky

5.2 Tabulka zapojení pinů

Slouží k přehlednosti a informovanosti konceptu vnitřního zapojení. Žáci si tak dle tabulky zjistí, který konektor od semaforu mají připojený do vybraného portu. Díky tomu se žák podívá do tabulky a zjistí, jaká LED dioda odpovídá řídicímu pinu pro rozsvícení a zhasnutí semaforu. Všechny semafory jsou zapojeny tak, aby odpovídaly i více typům semaforů. Jak již bylo zmíněno, na desce se nachází 2 typy. Semafor pro chodce plně koresponduje s řídicími piny pro stejné barvy semaforu nad silnicí pro vozidla.

Pokud by vyučující pedagog chtěl změnit zapojení, může tento krok realizovat poměrně snadno. Při demontáži plastového krytu lze veškeré propojovací kabely pozměnit dle vlastní libosti. Zde se nachází určitá pravidla, která je potřeba dodržovat v závislosti na technických údajů řídicí desky, aby nedošlo ke špatnému zapojení na piny mikrokontroléru, které nejsou určeny jako výstupní.

PORT A	
Signalizace	Pin zapojení
Červená	35
Oranžová	37
Zelená	36

PORT B	
Signalizace	Pin zapojení
Červená	33
Oranžová	34
Zelená	32

PORT C	
Signalizace	Pin zapojení
Červená	31
Oranžová	29
Zelená	30

PORT D	
Signalizace	Pin zapojení
Červená	26
Oranžová	28
Zelená	27

PORT E	
Signalizace	Pin zapojení
Červená	22
Oranžová	25
Zelená	24

PORT F	
Signalizace	Pin zapojení
Červená	4
Oranžová	3
Zelená	2

PORT G	
Signalizace	Pin zapojení
Červená	7
Oranžová	5
Zelená	6

PORT H	
Signalizace	Pin zapojení
Červená	9
Oranžová	8
Zelená	10

PORT I	
Signalizace	Pin zapojení
Červená	A2
Oranžová	A0
Zelená	A1

PORT J	
Signalizace	Pin zapojení
Červená	A5
Oranžová	A3
Zelená	A4

PORT K	
Signalizace	Pin zapojení
Červená	A7
Oranžová	A8
Zelená	A6

PORT L	
Signalizace	Pin zapojení
Červená	A9
Oranžová	A11
Zelená	A10

PORT M	
Signalizace	Pin zapojení
Červená	A12
Oranžová	A14
Zelená	A13

PORT N	
Signalizace	Pin zapojení
Červená	52
Oranžová	51
Zelená	53

PORT O	
Signalizace	Pin zapojení
Červená	43
Oranžová	40
Zelená	42

PORT P	
Signalizace	Pin zapojení
Červená	41
Oranžová	39
Zelená	38

Obrázek 23 Tabulka zapojení pinů desky

6 SOFTWARE PRO REALIZACI PROGRAMOVÁNÍ KŘIŽOVATKY

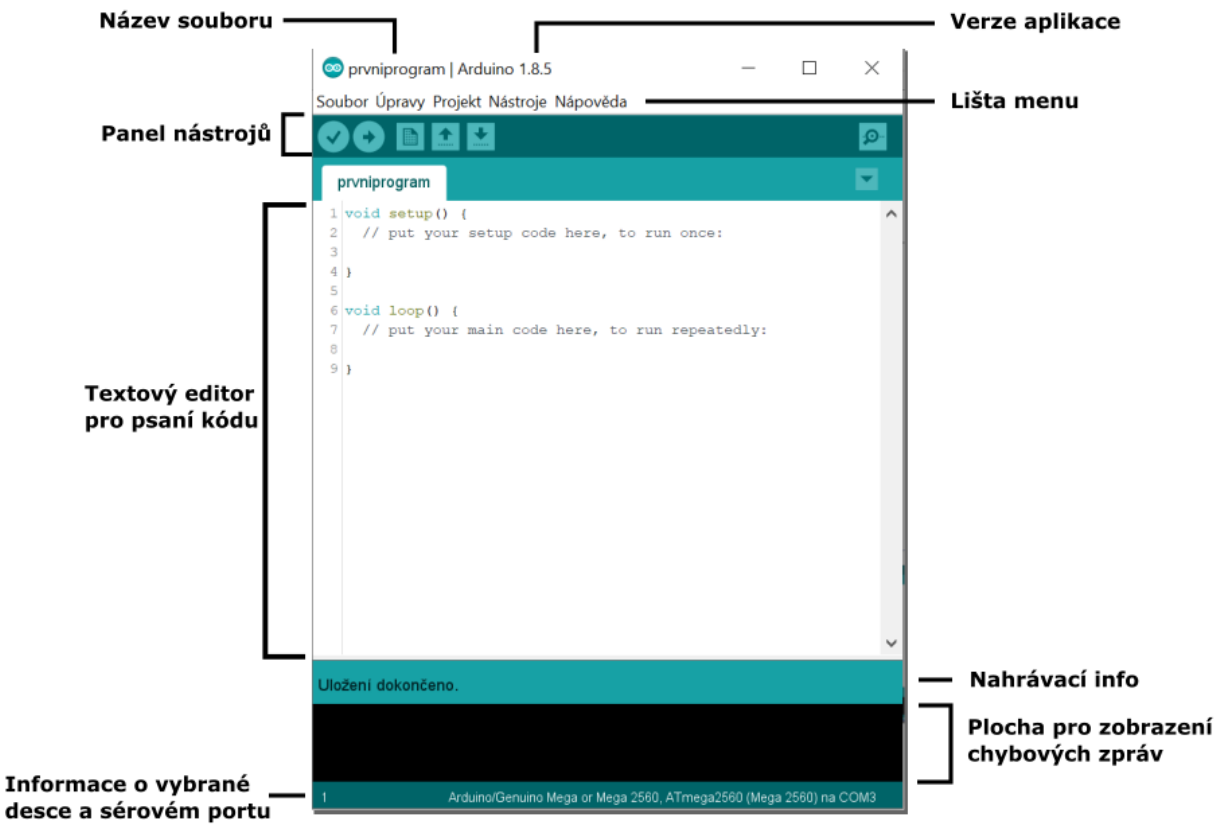
6.1 Vývojové prostředí

6.1.1 Arduino IDE

6.1.1.1 Popis programu

Vývojové prostředí Arduino IDE je „open-source“ software. Jeho licence umožňuje sdílet zdrojový kód v rámci předem definovaných podmínek. Používá se pro programování a nahrávání kódu do desky Arduino. Je vhodná pro různé typy operačních systémů, jakými jsou Windows, Linux a Mac OS. Podporuje programovací jazyky, kterými jsou C a C++. Programy neboli kódy napsané v aplikaci Arduino mají příponu souboru „**.ino**“. [35]

Po úspěšné instalaci a spuštění se uživateli zobrazí výchozí okno programu.

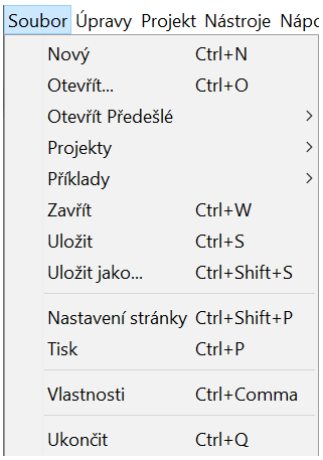


Obrázek 24 Aplikace Arduino IDE

6.1.1.2 Lišta menu

- **Soubor**

Po kliknutí na tlačítko „**Soubor**“ v nabídce menu, zobrazí se seznam viz. obrázek níže.



Obrázek 25 Záložka Soubor

6.1.1.3 Panel nástrojů

Obrázek 26 Panel nástrojů

Ověřit

Slouží a používá se ke kontrole kompilace napsaného kódu. Při neúspěšné kompilaci programu se objeví zpráva v okně chyby s jejími podrobnostmi.

Nahrát

Tlačítko „nahrát“ po jeho stisknutí zkompiluje napsaný kód. Pokud nastane chyba během kompilace programu, zobrazí se uživateli zpráva o chybě. Po úspěšném provedení kompilace kódu se nahraje kód na připojenou desku Arduino.

Před nahráním souboru bychom si měli ověřit jestli je zvolena správná vývojová deska Arduino a její komunikační port. Pro připojení desky k počítači potřebujeme USB připojení. Pokud byla provedena výše uvedená opatření, klikneme na tlačítko „nahrát“. Po nahrání souboru do desky Arduino si můžeme všimnout blikání LED na desce. Tento děj blikání netrvá dlouho dobu, avšak při pozorování během nahrávání můžeme tento jev spatřit.

Nový

Po stisknutí se otevře nové okno aplikace s možností psaní nového programu pro vývojovou desku, zvoleného typu.

Otevřít

Používá se pro otevření již založeného (vytvořeného programu). Po kliknutí na tlačítko se zobrazí okno s názvy předchozích vytvořených souborů a také další tlačítko „Otevřít“, pro vybrání souboru z libovolné složky v počítači.

Uložit

Plní funkci pro uložení dosavadně napsaného kódu v otevřeném souboru.

Sériový monitor

Nachází se na pravé straně lišty nástrojů. Po zvolení tlačítka se otevře okno, které se nazývá sériový monitor. Sériový monitor se používá pro vypisování dat z Arduina. Pro jeho správnou funkci je potřebné mít vybrán správný port.

6.1.2 Realizace programu

6.1.2.1 Vstupní a výstupní funkce Arduina

Piny Arduino desky jsou ve výchozím nastavení nakonfigurovány jakou vstupní piny. Není potřeba je znovu deklarovat jako vstupní piny pomocí příkazu pinMode(), když jsou používány jako vstupy. Piny nastavené jako pinMode(pin,INPUT), k nimž není nic připojeno případně jsou k nim připojené vodiče bez následného připojení k jiným obvodů, mohou hlásit

náhodné změny stavu pinů. Tyto náhodné stavy jsou dány tím, že zachycují elektrický šum z okolí.

Další možné nastavení pinu je pomocí vestavěného rezistoru. V čipu ATmega jsou zabudovány pull-up rezistory. Tyto vestavěné pull-up rezistory jsou dostupné nastavením pinMode() jako INPUT_PULLUP. Tenhle způsob invertuje chování režimu INPUT, kde znamená, že hodnota logické hodnoty 1 (HIGH) indikuje vypnutý senzor a logická hodnota 0 značí stav (LOW) a zapnutí senzoru. Hodnota rezistorů závisí na použitém mikrokontroléru. Většina desek Arduino má vestavěné pull-up rezistory mezi hodnotami 20 kΩ a 50 kΩ. Například na desce Arduino Due se hodnota rezistorů pohybuje v rozmezí od 50 kΩ do 150 kΩ. Přesnou hodnotu vždy nalezneme v datovém listu daného mikrokontroléru.

6.1.2.2 Definice vstupních a výstupních pinů

Programovatelné desky Arduino disponují větším počtem vstupních a výstupních pinů. Piny lze následně nakonfigurovat tak, aby byly schopné přijímat vstupní signály nebo dané instrukce posílat do dalšího modulu nebo elektronické součástky. Definování pinů lze provést dvěma způsoby:

Použití funkce pinMode()

Používá se k definování pinů. Danou funkcí specifikuje programátor daný pin, aby fungoval výstup nebo vstup. Piny u Arduino desek jsou ve výchozím nastavení nastaveny jako vstup. Je možné povolit interní „pull-up“ rezistory v režimu INPUT_PULLUP. Konfiguraci pinů lze nastavit třemi možnostmi, které jsou následující INPUT, OUTPUT nebo INPUT_PULLUP.

```
void setup ( ) {

pinMode ( 15 , OUTPUT ) ; /* pin 13 je definován pomocí pinMode*/

}

void loop ( ) {

digitalWrite ( 15 , HIGH ) ; /* definovaný pin nastavený jako HIGH*/
delay ( 1000 ) ; /* zpoždění 1 sekundu*/
}
```

Obrázek 27 Příklad zápisu pinMode():

Použití proměnných

Proměnné v programování slouží k ukládání dat. Příkaz proměnné se skládá z jejího typu, názvu a hodnoty. Proměnné lze také využít k deklaraci pinů desky Arduino.

```
int pin = 15;
```

Obrázek 28 Příklad zápisu proměnné

Na výše uvedeném příkladu je vytvořená proměnná, jejíž typ je „**int**“, název je „**pin**“ a má hodnotu „**13**“. Tímto jsme si definovali proměnnou, kterou můžeme použít v dalším kroku definování.

```
int pin = 15;

void setup ( ) {

pinMode ( pin , OUTPUT ) ; /* pin 13 je definován pomocí pinMode*/

}

void loop ( ) {

digitalWrite ( pin , HIGH ) ; /* definovaný pin nastavený jako HIGH*/
delay ( 1000 ) ; /* zpoždění 1 sekundu*/
}
```

Obrázek 29 Příklad zápisu pinMode() použitím proměnné

Rozdíl mezi zápisy je v definování pinu desky Arduino. Výhodou definování pinu pomocí proměnné je v přehlednosti kódu a také jednoduchosti dalšího použití. Uživatel si

nemusí pamatovat, jaké číslo má daný pin a při vhodném pojmenování proměnné se lépe zorientuje při hledání čísla pinu.

6.1.2.3 Nastavení sériové komunikace

Sériová komunikace slouží k výpisu dat v sériovém portu. Příkaz pro vypsání hodnot či textu může mít více variant. Data se vypisují v podobě textu ASCII, který je pro člověka čitelný. Data na port můžeme odesílat nebo je vypisovat. Záleží na daném použití, pro který se uživatel rozhodne v rámci situace provedení kódu.

K výpisu dat na sériový port se používá příkaz „Serial.print()“. Mezi jednoduché závorky se dává parametr či obsah„ který se má vypsat na sériovém monitoru.

Například:

```
void setup() {
  Serial.begin(9600); //otevře sériový port na 9600 bps
}
void loop() {
  Serial.print("123"); //výpis textu "123" na sériový port
}
```

Obrázek 30 Ukázka výpisu na sériový port

Na výše uvedeném obrázku je názorný příklad realizace výpisu textu „123“ na sériový monitor. Pro správnou funkčnost je důležité do funkce „setup()“ zadat příkaz pro otevření portu pro sériovou komunikaci „Serial.begin()“. Jako její parametr je nezbytné uvést rychlost komunikace, která je v jednotkách „bps“ – zkratka z angl. slov„ bits per second“. Poté

již stačí do hlavní programové smyčky „loop“ zadat příkaz pro samotný výpis s textem, který chce uživatel vypsat.

Lze také použít příkaz pro výpis textu „Serial.println()“. Oproti předchozímu příkazu se liší pouze v tom, že vypisuje daný obsah vždy na nový řádek. Výpis je tím pádem přehlednější pro uživatele a jednodušší pro zorientování. Výpis na sériovou komunikaci lze použít pro vypsání hodnot ze zapojených senzorů nebo pro odladění dané části programu.

6.1.2.4 Vytvoření proměnné s datovými typy

Datové typy proměnných jsou používány a využívají se k identifikaci typu dané proměnné a také souvisejících funkcí pro práci s daty. Jejich použití slouží k deklaraci funkcí a proměnných. Datové typy, které se používají v Arduino jsou následující a uvedeny níže.

void Data typ

Datový typ void určuje sadu hodnot, která je prázdná a její použití je pouze k deklaraci funkcí. Jelikož se jedná o prázdnou sadu hodnot, její návratový typ nevrací žádnou hodnotu, při vyvolání dané smyčky.

int Data typ

Použití nalezneme na místě, na kterém víme, že bude v dané proměnné uloženo celé číslo, jako jsou například čísla -6, 5, 10, - 155. Čísla mohou být i záporného typu. Celočíselné datové typy vystupují pod názvem „int“. Tento typ se považuje za primární datový typ k ukládání čísel. Velikost celočíselného datového typu „int“ jsou 2 bajty, lépe řečeno 16 bitů. Rozsah celočíselného typu je v rozmezí čísel od záporného čísla -32768 až po číslo 32767. V deskách Arduino UNO nebo ATmega se ukládá datový typ int hodnotu pomocí 2 bajtů. U desek jako jsou Arduino Due a Zero se ukládá a zabírá v paměti 32bitů, jinak řečeno 4 bajty. Záporné číslo je uložené ve formě doplňku 2, kde nejvyšší bit nebo znaménkový je označen jako záporné číslo.

```
int c;       // deklarace proměnné
int d = 6;   // hodnota 6 přiřazená proměnné d
```

Obrázek 31 proměnná int

Char Data typ

Do datového typu char můžeme uložit libovolný počet libovolný počet znakové sady. Deklarovaný identifikátor jako znak se stává znakovou proměnnou. Typ char se v mnoha případech označuje jako celočíselný typ proměnné. Je to z toho důvodu, že symboly nebo písmena, která jsou přiřazena proměnné v paměti, reprezentují přiřazené číselné kódy a ty jsou pouze celočíselná. Velikost char datového typu znaku je minimálně 1 bajt nebo 8 bitů. Například znak „B“ má hodnotu v tabulce ASCII 66.

```
char text =  'B' ; // datová proměnná text s uoženou hodnotou B
```

Obrázek 32 proměnná char

Float Data typ

Číslo, které není celočíselné a má desetinnou nebo zlomkovou část je považováno za číslo s plovoucí desetinnou čárkou. Například číslo „1,23“ je s plovoucí desetinnou čárkou. Zatímco číslo 25 je celé číslo, „25.0“ je plovoucí číslo s desetinnou čárkou. Plovoucí čísla mohou být zapsána v exponenciálním formátu. Čísla datového typu mohou být velká v rozsahu od -3,4028235E+38 až 3,4028235E+38. Velikost typu „float“ je 32 bitů.

```
float cislo = 3.2; // deklarace proměnné cislo typu float s hodnotou 3,2
```

Obrázek 33 proměnná float

Double Data typ

Používá se také pro zpracování čísel s plovoucí desetinnou čárku nebo desetinných čísel podobně jako předchozí datový typ „float“. Oproti typu float zabírá datový typ double dvakrát větší paměťový prostor. Rozdíl při použití je ve větší přesnosti a zmíněného rozsahu. U desky arduino Due je jeho velikost v paměti 64 bitů a u desek ATmega a UNO je paměťová velikost poloviční tedy 32 bitů.

```
double body = 32.3 // deklarace proměnné body typu double s hodnotou 32.3
```

Obrázek 34 proměnná double

Unsigned int Data typ

Rozdíl oproti použití pouze celočíselného typu int a unsigned int je, že datový typ unsigned ukládá do paměti pouze kladné hodnoty proměnné. Rozsah unsigned int datového typu je od hodnoty 0 až do čísla 65 535. Ukládá hodnotu až do 16 bitů. Rozdíl mezi datovým typem je signed a unsigned je znaménkový bit. U 16ti bitového čísla je 15 bitů interpretováno s doplňkem 2. Horní bit je interpretován jako záporné nebo kladné číslo. Pokud se číslo považuje za záporné, horní bit má hodnotu „1“.

```
unsigned int pinLED = 8 ; //  deklarace proměnné pinLED typu unsigned int s hodnotou 8
```

Obrázek 35 proměnná unsigned int

Short Data typ

Opět se jedná o celočíselný datový typ, který do paměti ukládá 16bitová data. Rozsah u datového typu short je od záporného čísla -32768 až do hodnoty 32767. Například Arduino desky založené na mikrokontroléru ARM nebo ATmega obvykle ukládají datvou hodnotu 16 bitů.

```
short pin = 7;  // deklarace proměnné pin typu short s hodnotou 7
```

Obrázek 36 proměnná short

Long Data typ

Datový typ long je považován za proměnnou rozšířené velikosti, které zabírají a ukládají 32 bitů. Použití nalezneme například na takovém místě, kde již víme, že proměnná bude pracovat s velkými čísly. Při použití celých čísel by mělo aspoň za jedním z čísel následovat písmeno „L“, díky tomu se vnutí proměnné, aby bylo číslo datového typu long.

```
long rychlost = 197000L;  // deklarace proměnné rychlost typu long s hodnotou 197000
```

Obrázek 37 proměnná long

Unsigned long Data typ

Stejně jako u long typu zabírá unsigned long 32 bitů. Rozdíl je v tom, že u typu long lze ukládat číslo se zápornou hodnotou. U typu unsigned long se neukládá záporné číslo. Znamená to, že rozsah, který je v záporném rozmezí u předchozího typu, je dvojnásobný

v kladné hodnotě. Proměnná s datovým typem unsgined long může nabývat velikost od 0 do 4 294 967 295.

```
unsigned long var = 2255123698;
// deklarace proměnné var typu unsigned long s hodnotou 2255123698
```

Obrázek 38 proměnná unsigned long

Byte Data typ

Je považován za číslo bez znaménka, které ukládá do paměti s rozsahem od 0 do 255. Zabírá velikost 8 bitů.

```
byte c = 30 ; // deklarace proměnné c typu byte s hodnotou 30
```

Obrázek 39 proměnná byte

Word Data typ

Stejně jak u typu byte je datový typ word číslo bez znaménka. Může nabývat kladné hodnoty v rozmezí od čísla 0 do 65 535. Jeho velikost činí 16 bitů.

```
word f = 5000 ; // deklarace proměnné f typu word s hodnotou 5000
```

Obrázek 40 proměnná word

6.1.2.5 Podmínkové příkazy

Po zadání příkazu if se provede blok kódu, jehož zadaná podmínka je pravdivá. Pokud je podmínka nepravdivá, provede se další blok kód else. Příkazy if, else if a else jsou součástí podmíněných příkazů, které se používají k provedení určitých částí kódu na základě rozhodovací podmínky.

Příkaz if vykoná blok kódu, pokud jeho podmínka je pravdivá.

```
if (x > 5) {
  /* blok kódu, který se má provést, pokud je podmínka pravdivá,
  tedy cislo x je větší než číslo 5*/
}
```

Obrázek 41 Ukázka podmínky if

```
if (x > 5) {
  // blok kódu, který se má provést, pokud je podmínka pravdivá
} else {
  // blok kódu, který se má provést, pokud je x menší než číslo 5
}
```

Obrázek 42 Ukázka podmínky else

```
if (x > 5) {
  // blok kódu, který se má provést, pokud je x větší než 5 pravdivé
} else if (x == 3) {
  // blok kódu, který se má provést, pokud je x menší než 5 nepravdivé a x rovno číslu 3 je pravdivá
} else {
  // blok kódu, který se má provést, pokud je x menší než 5 nepravdivé a x rovno číslu 3 je nepravdivé
}
```

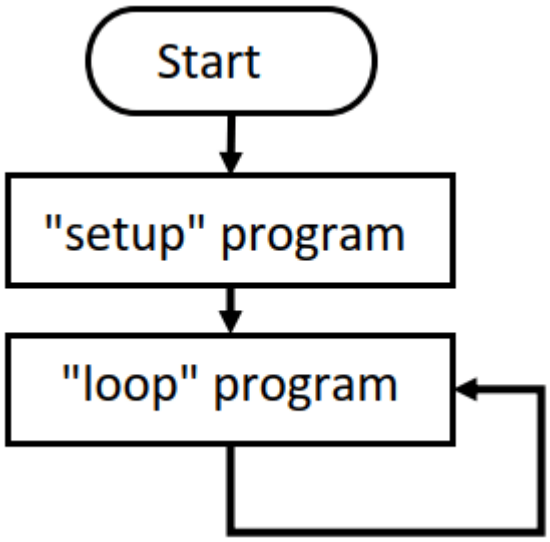
Obrázek 43 Ukázka podmínky else if

6.1.2.6 Vytvoření funkce

Funkce umožňují rozdělit konkrétní kód do různých funkcí, kde každá funkce plní konkrétní úkol v programu. Funkce jsou vytvářeny k provádění úkolu vícekrát v programu.

Jednou z výhod používání funkcí je, že se programátor vyhne nutnosti psát stejný kód znovu, což šetří paměť a čas. Pokaždé když je funkce zavolána, používá se pouze jednou napsaný kód. Funkce také slouží k rozdělení programu na jednotlivé úseky, díky nimž se stává kód „jednodušší“ lépe pochopitelný a modulárnější. Tento fakt má i dopad na samotnou čitelnost celého kódu programu.

Arduino má dvě běžné funkce. První z nich je setup() a loop(). Tyto dvě funkce jsou volány automaticky na pozadí. Kód, který se má provést ve funkci je uzavřen složenými závorkami. Funkce setup() obsahuje počáteční část kódu, která se provede pouze jednou, ihned po zapnutí mikrokontroléru. Této funkci se říká přípravný blok, kde se nastavují piny na vstupní nebo výstupní funkci. Loop() je hlavní programová smyčka, která se vykonává neustále dokola. Obsahuje příkazy, které se po provedení znovu opakují. Této části je říká prováděcí blok.



Obrázek 44 Kostra programu [36]

Deklarace funkce

Je několik pravidel, které je potřeba dodržovat pro správné fungování funkce. Jedním z pravidel je, že funkce musí mít jedinečný název. Dalším pravidlem je obsažení návratové hodnoty funkce. Po návratovém typu a názvu funkce následují závorky, do které se může ale i nemusí psát parametr, který je závislý na návratovém typu funkce. Poslední podmínkou pro správné fungování funkce je použití složitých závorek těla funkce. Kód napsaný mezi těmito závorkami je tělem dané funkce.

Pro funkci potřebujeme návratový typ. Do proměnné můžeme například uložit návratovou hodnotu funkce. Návratový typ může být použit libovolný datový typ např. int, float nebo char a další dostupné. Dále se skládá z názvu funkce. Poté zahrnuje parametr nebo více parametrů předané funkci. Parametry jsou definovány jako proměnné, které se používají k předávání dat funkci, na základě kterých funkce vyhodnotí odpověď. Tyto parametry se nazývají argumenty funkce.

```
int nasobeni(int x, int y){ //název funkce nasobeni s datovým typem int a s parametry int x a int y
  int vysledek;             //vytvoření proměnné výsledek
  vysledek = x * y;         //vynásobení proměnné x a y s následným uložením výsledku do proměnné vysledek
  return vysledek;          //návratová hodnota funkce proměnná vysledek
}
```

Obrázek 45 Ukázka vytvoření funkce

7 SADA ŘEŠENÝCH ÚLOH PRO REALIZACI KŘIŽOVATKY

Sada obsahuje několik ukázkových úloh včetně jejího řešení. Každá úloha nemá pouze jedno řešení. Žák se může ke správnému výsledku dostat i pomocí jiného vlastního řešení. Zde se nabízí zároveň více variant řešení, které dávají studentovi určitou volnost a představivost během realizace.

Vyučující pedagog má kdykoliv možnost pozměnit zapojení pinů v desce, a žák musí opět přizpůsobit programování jinému vnitřnímu zapojení modelu křižovatky. Práce s modelem se dá pojmout i formou skupinové výuky. Vyučující může rozdělit žáky do skupin po určitém počtu studentů. Další alternativou je individuální práce studenta na realizaci křižovatky a jeho zprovoznění. [37]

7.1 Úloha č. 1 – semafor pro chodce s použitím funkce

Cíle výuky:

Žáci budou schopni rozpoznat elektronické propojovací části modelu od mechanických.

Žáci dokáží rozlišit semafory a konektory, pro správnou identifikaci a propojení.

Žáci rozvíjí vlastní představivost, na základě které upravují zapojení křižovatky.

Žáci dokáží propojit semafor pro chodce s řídicí jednotkou.

Žáci dokáží propojit řídicí modul s počítačem.

Žáci zvládnou napsat program pro zprovoznění semaforu.

Žáci budou schopni nahrát program přes počítač do řídicí jednotky Arduino.

Pomůcky:

Model křižovatky s veškerými komponenty

Počítač s nainstalovaným grafickým prostředím Arduino IDE

Rozsah:

2x45 min.

Zadání

1. Vyberte z dostupných komponent semafor a desku a následně semafor do desky připevněte.

2. Vyberte si port **B** na řídicí jednotce Arduino pro zapojení semaforu.

3. Zapojte konektor od semaforu pro chodce označeným číslem **3** do řídicí jednotky.

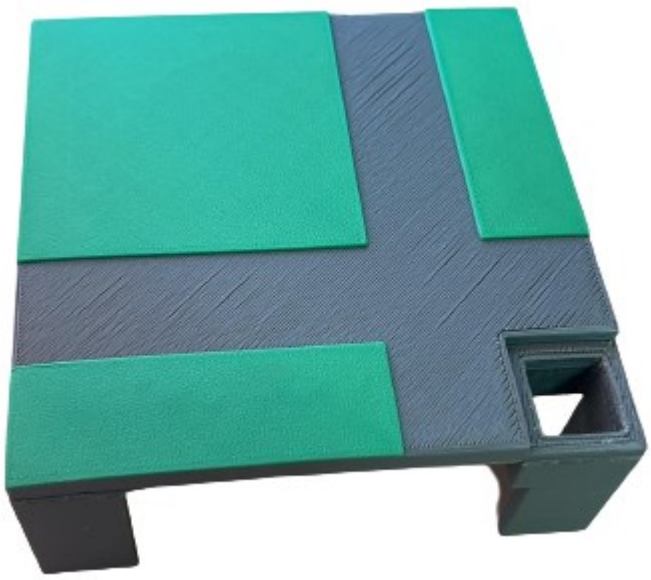
4. Vytvořte funkční program dle ukázkového programu nebo na základě vlastní představivosti pro funkci semaforu pro chodce.

5. Nahrajte program do řídicí jednotky.

6. Otestujte a zkontrolujte vizuální funkci modelu semaforu chodce.

1. Uchycení semaforu do desky

Ze všech komponentů, které jsou k dispozici v krabici, nalezneme díl na níže uvedeném obrázku.



Obrázek 46 Deska pro semafor

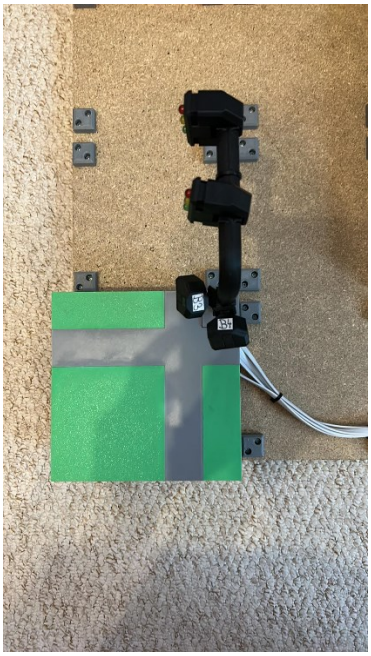
Deska na semafor má v jednom rohu otvor. Tento otvor slouží pro uchycení námi vybraného semaforu. Druhou potřebnou komponentou je samotný semafor.

Obrázek 47 Samostatný semafor

Po nalezení semaforu a desky, spojíme tyto dvě komponenty. Otvorem desky nejprve provlečeme 4 kabely s konektory na konci. Poté stačí dotlačit semafor do desky a výsledné uchycení by mělo vypadat stejně jak na obrázku pod tímto textem.

Obrázek 48 Spojená deska se semaforem

Když máme díly spojené do sebe, položíme semafor s deskou do podkladové desky na příslušné místo, kde chceme mít umístěn semafor. Na podkladové desce se nacházejí plastové kotvy. Každá kotva slouží pro ukotvení rohu desky se semaforem. Do našeho zvoleného místa se čtyřmi kotvami umístíme desku se semaforem a následně zatlačíme desku mezi kotvy, dokud se nebude deska dotýkat podkladové desky viz. obrázek.



Obrázek 49 Uchycený semafor do podkladové desky

2. Zvolení portu semaforu

Na řídicí jednotce se nachází 16 portů, do kterých můžeme zapojit semafor. Například si můžeme zvolit „**port B**“. Podíváme se do tabulky s porty a zjistíme si čísla pinů zapojení, které budeme potřebovat při programování řídicí jednotky.

PORT B	
Signalizace	**Pin zapojení**
Červená	33
Oranžová	34
Zelená	32

Obrázek 50 Zvolený port B

3. Zapojení semaforu do řídicí jednotky

Na každém semaforu se nachází štítek s názvem, který slouží pro identifikaci správného konektoru. Semafory pro chodce mají vždy označení písmenem od A do D s čísly 3 a 4. Vyberte si libovolný semafor a zapojte jeho konektor s číslem 3 do portu B.

4. Vytvoření funkčního programu

Nejprve je potřeba definovat piny portu B, do kterého jsme připojili semafor. Dle tabulky zapojení si zjistíme čísla pinů portu B. V případě semaforu pro chodce potřebujeme pro realizaci pouze 2 piny, které nám signalizují barvy zelenou a červenou. Čísla pinů jsou 33 pro červenou LED diodu a 32 pro zelenou.

Ve funkci „**setup()**“ musíme nastavit piny pro zelenou a červenou LED diodu jako výstupní pomocí příkazu **pinMode**.

V hlavní programové smyčce „**loop()**“, kde se volají ostatní námi napsané funkce, voláme funkci „**semaforChodec()**“ s parametry 4000 a 2000. Tyto parametry nám udávají čas, jak dlouho bude svítit červená a zelená LED dioda na semaforu.

V poslední části programu je samotná funkce „**semaforChodec()**“, která nám zajišťuje funkci semaforu.

```
int cervenaA3 = 33;               //definice pinu pro cervenou LED diodu na PORTU B
int zelenaA3 = 32;                //definice pinu pro zelenou LED diodu na PORTU B


void setup() {

  pinMode(cervenaA3, OUTPUT);     //nastavení pinu cervenaA3 na výstup
  pinMode(zelenaA3, OUTPUT);      //nastavení pinu zelenaA3 na výstup

}

void loop() {

  semaforChodec(4000,2000);               //volání funkce semaforChodec()

}

void semaforChodec(int cervena,int zelena) {
  digitalWrite(cervenaA3, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervena);                   //zpoždění 10s - červená LED dioda svítí
  digitalWrite(cervenaA3, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(zelenaA3, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelena);                   //zpoždění 10s - zelená LED dioda svítí
  digitalWrite(zelenaA3, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
}
```

Obrázek 51 Ukázkový program pro semafor chodce

5. Nahrání programu do řídicí jednotky Arduino

Pokud je kód programu semaforu pro chodce hotový, propojíme řídicí jednotku s počítačem pomocí kabelu.

Obrázek 52 Propojovací kabel mezi PC a řídicí jednotkou [38]

Po propojení počítače a řídicí jednotky zkontrolujeme propojení na portu USB a zvolenou desku mikrokontroléru Arduino Mega.

Kontrola funkčnosti naprogramovaného semaforu pro chodce

Pokud jste úspěšně nahráli program do řídicí jednotky. Zkontrolujte, jestli zapojený semafor svítí a funguje správně dle Vašeho programu. V případě ukázkového programu uvedeného výše, se na semaforu pro chodce rozsvítí červená LED dioda na 4 sekundy a následně zelená na 2 sekundy.

Bonusový úkol

♠ Upravte parametry funkce, aby na semaforu pro chodce svítila zelená LED dioda 6 sekund a červená 5,5 sekundy.

7.2 Úloha č. 2 – semafory pro auta s použitím sériové komunikace a příkazu switch

Cíle výuky:

Žáci rozvíjí získané předchozí znalosti.

Žáci budou schopni změnit program, aby fungoval jiný semafor.

Žáci budou schopni komunikace s řídicí jednotkou po sériové lince.

Žáci zvládnout upravit příkaz switch.

Pomůcky:

Model křižovatky s veškerými komponenty

Počítač s nainstalovaným grafickým prostředím Arduino IDE

Rozsah:

2x45 min.

Zadání:

1. Vyberte z dostupných komponent semafor a desku a následně semafor do desky připevněte.

2. Vyberte si porty **A, B, E, F** na řídicí jednotce Arduino pro zapojení semaforů pro auta

3. Zapojte konektory od semaforů pro auta s čísly (1,2) do řídicí jednotky

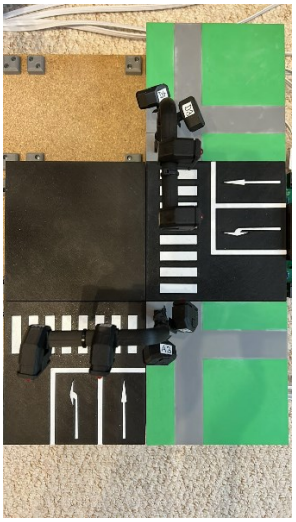
4. Vytvořte funkční program dle ukázkového programu nebo na základě vlastní představivosti pro funkci semaforů pro auta s použitím sériové komunikace.

5. Nahrajte program do řídicí jednotky.

6. Otestujte a zkontrolujte vizuální funkci modelu semaforů pro auta.

2. Zvolení komponent a uchycení

Z dostupných komponent vyberte **2 semafory**. Poté oba semafory upevněte libovolně do desek určených pro semafory. Následně kabely s konektory prostrčte pod ostatními deskami k řídicí jednotce.



Obrázek 53 Zvolené a uchycené komponenty

3. Zvolení portů semaforů

Po úspěšném uchycení semaforů najděte na desce porty s označením písmen „**A**“, „**B**“, „**E**“ a „**F**“.

PORT A	
Signalizace	**Pin zapojení**
Červená	35
Oranžová	37
Zelená	36

PORT B	
Signalizace	**Pin zapojení**
Červená	33
Oranžová	34
Zelená	32

Obrázek 54 Připojení prvního semaforu do portů A a B

PORT E	
Signalizace	**Pin zapojení**
Červená	22
Oranžová	25
Zelená	24

PORT F	
Signalizace	**Pin zapojení**
Červená	4
Oranžová	3
Zelená	2

Obrázek 55 Připojení druhého semaforu do portů E a F

Díky tabulce, vidíme čísla pinů, která budeme potřebovat během psaní programu pro správnou funkci semaforu.

Konektory od kabelů semaforů s čísly (1, 2) zapojte do příslušných portů (A, B, E, F) řídicí jednotky.

4. Vytvoření funkčního programu

Nejprve je potřebné si definovat piny portů pomocí předchozího obrázku.

```
int semaforA1cervena = 22;    //definice pinu pro červenou LED na portu E
int semaforA1oranzova = 25;   //definice pinu pro oranžovou LED na portu E
int semaforA1zelena =  24;    //definice pinu pro zelenou LED na portu E

int semaforA2cervena = 4;     //definice pinu pro červenou LED na portu F
int semaforA2oranzova = 3;    //definice pinu pro oranžovou LED na portu F
int semaforA2zelena = 2;      //definice pinu pro zelenou LED na portu F

int semaforB1cervena = 35;    //definice pinu pro červenou LED na portu A
int semaforB1oranzova = 37;   //definice pinu pro oranžovou LED na portu A
int semaforB1zelena =  36;    //definice pinu pro zelenou LED na portu A

int semaforB2cervena = 33;    //definice pinu pro červenou LED na portu B
int semaforB2oranzova = 34;   //definice pinu pro oranžovou LED na portu B
int semaforB2zelena = 32;     //definice pinu pro zelenou LED na portu B
```

Obrázek 56 Definice pinů zapojení

V druhém kroku je důležité nastavit ve smyčce „**setup()**“ příkaz pro zapnutí komunikace se sériovým monitorem a následně definovat všechny předchozí piny jako výstupní.

```
void setup() {
  Serial.begin(9600);                    //Nastavení funkce sériového monitoru

  //nastavení proměnných jako výstup
  pinMode(semaforA1cervena, OUTPUT);
  pinMode(semaforA1oranzova, OUTPUT);
  pinMode(semaforA1zelena, OUTPUT);

  pinMode(semaforA2cervena, OUTPUT);
  pinMode(semaforA2oranzova, OUTPUT);
  pinMode(semaforA2zelena, OUTPUT);

  pinMode(semaforB1cervena, OUTPUT);
  pinMode(semaforB1oranzova, OUTPUT);
  pinMode(semaforB1zelena, OUTPUT);

  pinMode(semaforB2cervena, OUTPUT);
  pinMode(semaforB2oranzova, OUTPUT);
  pinMode(semaforB2zelena, OUTPUT);


  //nastavení hodnoty pinu pro červené LED diody na hodnotu LOW - LED svítí
  digitalWrite(semaforA1cervena, LOW);
  digitalWrite(semaforA2cervena, LOW);

  //nastavení hodnoty pinu pro oranžové a zelené LED diody na hodnotu HIGH - LED nesvítí
  digitalWrite(semaforA1oranzova, HIGH);
  digitalWrite(semaforA2oranzova, HIGH);

  digitalWrite(semaforB1oranzova, HIGH);
  digitalWrite(semaforB2oranzova, HIGH);
}
```

Obrázek 57 Smyčka setup()

Nejdříve jsme si nastavili komunikaci se sériovým monitorem pomocí příkazu **Serial.begin**(9600). Číslo v závorkách nám udává rychlost v jednotkách baud. V další části kódu jsou všechny definované piny nastaveny jako výstupní. V poslední části jsou na piny červených LED zapsány hodnoty LOW, aby nám při spuštění programu svítily červené LED diody. U ostatních LED diod (oranžová a zelená) jsou hodnoty nastavené na HIGH, díky kterým nám ostatní LED diody nesvítí.

```
void loop() {
  Serial.println("Zadejte písmeno semaforu:");           //výpis na textu na sériový monitor
  while (Serial.available() == 0) {}                     //smyčka na čekání na data
  int data = Serial.read();  //read until timeout        //uložení hodnoty ze sériového monitoru do proměnné data

  switch (data)                                          //vytvoření příkazu switch na proměnnou data
                                                         //- dle hodnoty proměnné data se provede určitý case
  {
    case 'A':                                            //pokud je proměnná data A provede se funkce semaforA1A2
      semaforA1A2(2000, 2000, 4000);                     //provedení funkce
      break;                                             //příkaz break - tímto příkazem musí končit každý "case"
    case 'B':                                            //pokud je proměnná data B provede se funkce semaforB1B2
      semaforB1B2(2000, 2000, 4000);                     //provedení funkce
      break;                                             //příkaz break - tímto příkazem musí končit každý "case"
    default:                                             //výchozí hodnota příkazu switch
      Serial.println("Zadejte správné písmeno semaforu"); //výpis na textu na sériový monitor
      break;                                             //příkaz break - tímto příkazem musí končit každý "case"
  }
}
```

Obrázek 58 Smyčka loop()

Ve smyčce „**loop()**“ jsme si jako první příkaz vypsali text „Zadejte písmeno semaforu“, aby uživatel věděl, že má zadat písmeno daného semaforu pro jeho spuštění. Poté se v programu nachází další příkaz a tím je cyklus „while“, který čeká na data zadané ze sériového monitoru. Dalším příkazem je proměnná pro uložení hodnoty zadané do sérového monitoru. V dalším kroku je provedení příkazu switch, který má v sobě příkazy které má provést v závislosti na zadaném písmenu v sériovém monitoru. V poslední části programu jsou definovány funkce semaforů s názvy semaforA1A2 a semaforB1B2.

```
void semaforA1A2(int cervenaA1, int oranzovaA1, int zelenaA1) {
  Serial.println("Zadali jste semafor A. Prosím Čekejte na provedení funkce"); //výpis textu na sério.
  digitalWrite(semaforA1cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforA2cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA1);                       //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforA1oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforA2oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA1);                      //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA1cervena, HIGH);   //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA2cervena, HIGH);   //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA1oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA1zelena, LOW);     //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforA2zelena, LOW);     //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaA1);                        //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforA1zelena, HIGH);    //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA2zelena, HIGH);    //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA1oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforA2oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA1);                      //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA1oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA1cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforA2cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 59 funkce semaforA1A2

```
void semaforB1B2(int cervenaB1, int oranzovaB1, int zelenaB1) {
  Serial.println("Zadali jste semafor B. Prosím Čekejte na provedení funkce");
  digitalWrite(semaforB1cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaB1);                       //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforB1oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforB2oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                      //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforB1cervena, HIGH);   //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB2cervena, HIGH);   //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB1oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB2oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB1zelena, LOW);     //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforB2zelena, LOW);     //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaB1);                        //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforB1zelena, HIGH);    //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB2zelena, HIGH);    //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB1oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforB2oranzova, LOW);   //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                      //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforB1oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB2oranzova, HIGH);  //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB1cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);    //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 60 funkce semaforB1B2

5. Nahrání programu do řídicí jednotky Arduino

Pokud je kód programu semaforu pro auta hotový, propojíme řídicí jednotku s počítačem pomocí kabelu.

Obrázek 61 Propojovací kabel mezi PC a řídicí jednotkou [38]

Po propojení počítače a řídicí jednotky zkontrolujeme propojení na portu USB a zvolenou desku mikrokontroléru Arduino Mega.

Kontrola funkčnosti naprogramovaných semaforů pro auta

Pokud jste úspěšně nahráli program do řídicí jednotky. Zkontrolujte, jestli zapojený semafor svítí a funguje správně dle Vašeho programu. Dle ukázkového programu, po zadání písmene A se provede funkce **semaforA1A2**. V případě zadání písmene B se provede funkce **semaforB1B2**.

Bonusový úkol

♠ Upravte příkaz switch v hlavní programové smyčce „loop()“, aby při napsání písmena C v sériovém monitoru, se provedla funkce semaforA1A2 s parametry (3000, 3500, 5000).

7.3 Úloha č. 3 – silnice se světelnou signalizací ve tvaru I pro chodce a auta

Cíle výuky:

Žáci rozvíjí získané předchozí znalosti.

Pomůcky:

Model křižovatky s veškerými komponenty

Počítač s nainstalovaným grafickým prostředím Arduino IDE

Rozsah:

3x45 min.

Zadání:

1. Vyberte z dostupných komponent semafory a desky a následně semafory do desky připevněte.

2. Vyberte si porty **D, F, G, J, K, M** na řídicí jednotce Arduino pro zapojení semaforů

3. Zapojte konektory od semaforů pro auta (čísla 1, 2) a semaforů pro chodce (čísla 3, 4) do řídicí jednotky podle **tabulky zapojení konektorů.**

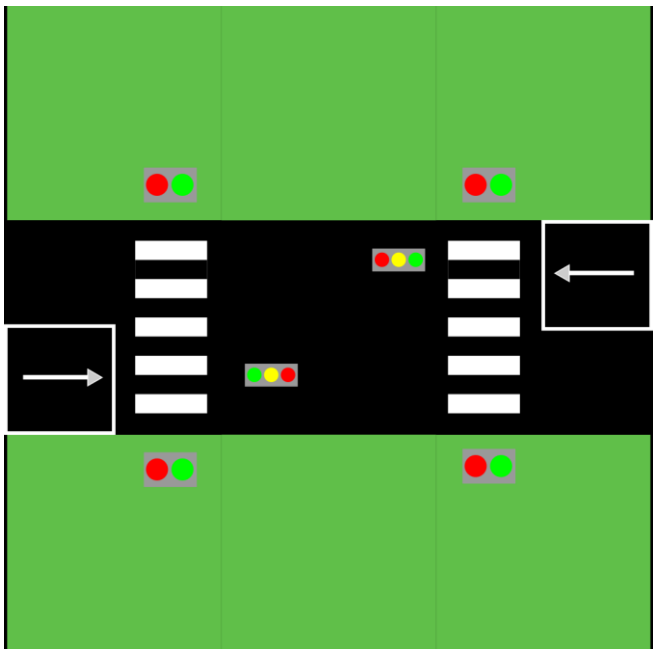
4. Vytvořte funkční program dle ukázkového programu nebo na základě vlastní představivosti pro funkce semaforů pro auta a chodce.

5. Nahrajte program do řídicí jednotky.

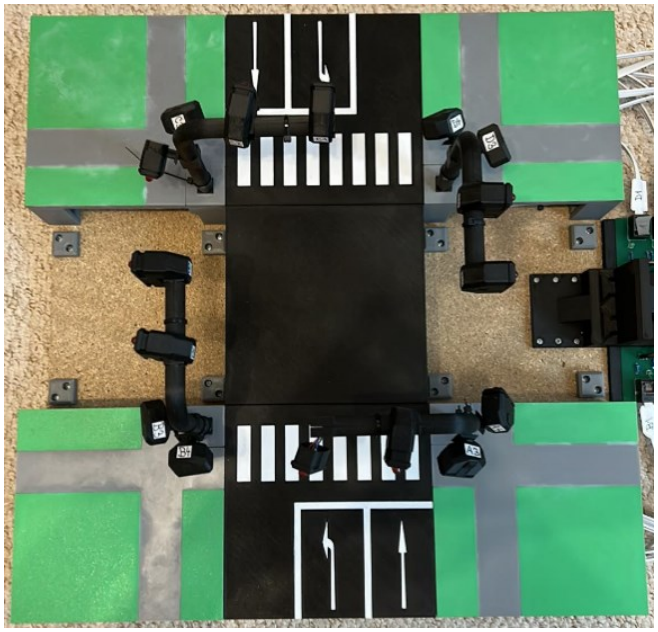
6. Otestujte a zkontrolujte vizuální funkci modelu semaforů pro auta a semaforů pro chodce.

2. Zvolení komponent a uchycení

Z dostupných komponent vyberte **2 semafory**. Poté oba semafory upevněte naproti sobě do desek určených pro semafory. Následně kabely s konektory prostrčte pod ostatními deskami k řídicí jednotce.



Obrázek 62 Situační nákres



Obrázek 63 Vzhled silnice ve tvaru I

3. Zvolení portů semaforů

Po úspěšném uchycení semaforů najděte na desce porty s označením písmen **D**, **F**, **G**, **J**, **K**, **M**.

PORT D	
Signalizace	Pin zapojení
Červená	26
Oranžová	28
Zelená	27

PORT F	
Signalizace	Pin zapojení
Červená	4
Oranžová	3
Zelená	2

PORT G	
Signalizace	Pin zapojení
Červená	7
Oranžová	5
Zelená	6

PORT J	
Signalizace	Pin zapojení
Červená	A5
Oranžová	A3
Zelená	A4

PORT K	
Signalizace	Pin zapojení
Červená	A7
Oranžová	A8
Zelená	A6

PORT M	
Signalizace	Pin zapojení
Červená	A12
Oranžová	A14
Zelená	A13

Obrázek 64 Porty s čísly pinů

Díky obrázku, vidíme čísla pinů, která budeme potřebovat během psaní programu pro správnou funkci semaforů.

Zapojení konektorů od semaforů do portů řídicí jednotky provedeme pomocí následující tabulky.

Číslo Portu	Číslo konektoru
D	B4
F	A2
G	A3
J	C3
K	C2

M	D4

Tabulka 3 Zapojení konektorů do řídicí jednotky

4. Vytvoření funkčního programu

Nejprve je potřebné si definovat piny portů pomocí předchozího obrázku.

```
//definice pinů od semaforu pro chodce na portu F
int semaforA2cervena = 4;
int semaforA2oranzova = 3;
int semaforA2zelena = 2;

//definice pinů od semaforu pro chodce na portu G
int semaforA3cervena = 7;
int semaforA3zelena = 6;

//definice pinů od semaforu pro chodce na portu D
int semaforB4cervena = 26;
int semaforB4zelena = 27;

//definice pinů od semaforu pro auta na portu K
int semaforC2cervena = A7;
int semaforC2oranzova = A8;
int semaforC2zelena = A6;

//definice pinů od semaforu pro chodce na portu J
int semaforC3cervena = A5;
int semaforC3zelena = A4;

//definice pinů od semaforu pro chodce na portu M
int semaforD4cervena = A12;
int semaforD4zelena = A13;
```

Obrázek 65 Definice pinů zapojení

V druhém kroku je důležité definovat ve smyčce „**setup()**“ všechny předchozí piny jako výstupní a zapsat počáteční hodnoty semaforům pro auta.

```
void setup() {

  //nastavení proměnných jako výstup
  pinMode(semaforA2cervena, OUTPUT);
  pinMode(semaforA2oranzova, OUTPUT);
  pinMode(semaforA2zelena, OUTPUT);

  pinMode(semaforA3cervena, OUTPUT);
  pinMode(semaforA3zelena, OUTPUT);

  pinMode(semaforB4cervena, OUTPUT);
  pinMode(semaforB4zelena, OUTPUT);

  pinMode(semaforC2cervena, OUTPUT);
  pinMode(semaforC2oranzova, OUTPUT);
  pinMode(semaforC2zelena, OUTPUT);

  pinMode(semaforC3cervena, OUTPUT);
  pinMode(semaforC3zelena, OUTPUT);

  pinMode(semaforD4cervena, OUTPUT);
  pinMode(semaforD4zelena, OUTPUT);

  //zápis počátečních hodnot semaforů A2 a C2
  digitalWrite(semaforA2cervena, LOW);
  digitalWrite(semaforA2oranzova, HIGH);
  digitalWrite(semaforA2zelena, HIGH);

  digitalWrite(semaforC2cervena, LOW);
  digitalWrite(semaforC2oranzova, HIGH);
  digitalWrite(semaforC2zelena, HIGH);


}
```

Obrázek 66 Smyčka setup()

V první části kódu jsou všechny definované piny nastaveny jako výstupní. V poslední části jsou na piny červených LED zapsány hodnoty LOW, aby nám při spuštění programu svítily červené LED diody. U ostatních LED diod (oranžová a zelená) jsou hodnoty nastavené na HIGH, díky kterým nám ostatní LED diody nesvítí.

```
void loop() {

  //volání dvou funkcí
  semaforChodecA3B4C3D4(2000, 4000); //volání funkce semaforů pro chodce

  semaforA2C2(2000, 2000, 4000); //volání funkce semaforů pro auta

}
```

Obrázek 67 Smyčka loop()

Ve smyčce „**loop()**“ jsou volány dvě funkce, prvním z nich je funkce pro semafory chodců a jako druhá je volána funkce pro semafory aut. Tyto dvě funkce jsou vloženy pod poslední složenou závorku od smyčky „**loop()**“.

```
void semaforChodecA3B4C3D4(int cervenaA3, int zelenaA3) {
  digitalWrite(semaforA3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA3);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforA3cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB4cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC3cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforD4cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA3zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforB4zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforC3zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforD4zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaA3);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforA3zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB4zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC3zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforD4zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 68 funkce semaforChodecA3B4C3D4

```
void semaforA2C2(int cervenaA2, int oranzovaA2, int zelenaA2)
{
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA2);
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);
  digitalWrite(semaforA2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(zelenaA2);
  digitalWrite(semaforA2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 69 funkce semaforA2C2

5. Nahrání programu do řídicí jednotky Arduino

Pokud je kód programu semaforu pro auta hotový, propojíme řídicí jednotku s počítačem pomocí kabelu.

Obrázek 70 Propojovací kabel mezi PC a řídicí jednotkou [38]

Po propojení počítače a řídicí jednotky zkontrolujeme propojení na portu USB a zvolenou desku mikrokontroléru Arduino Mega.

Kontrola funkčnosti naprogramované silnice se světelnou signalizací ve tvaru I pro chodce a auta

Pokud jste úspěšně nahráli program do řídicí jednotky. Zkontrolujte, jestli zapojené semafory svítí a fungují dle Vašeho programu. Dle ukázkového programu nejprve svítí u všech semaforů červená LED dioda, poté se rozsvítí na 4 sekundy zelené LED diody od semaforů pro chodce a následně se rozsvítí od semaforů pro auta červené LED na 2 sekundy, oranžové LED také na 2 sekundy a jako poslední zelené na 4 sekundy.

Bonusový úkol

♠ Vymyslete vlastní funkci semaforů a realizujte ji pomocí vytvoření funkce s libovolným názvem a následným zavoláním funkce v hlavní programové smyčce loop().

7.4 Úloha č. 4 - křižovatka ve tvaru T

Cíle výuky:

Žáci rozvíjí získané předchozí znalosti.

Pomůcky:

Model křižovatky s veškerými komponenty

Počítač s nainstalovaným grafickým prostředím Arduino IDE

Rozsah:

4x45 min.

Zadání:

1. Vyberte z dostupných komponent semafory a desky a následně semafory do desky připevněte.

2. Vyberte si porty **A, B, C, D, E, F, G, I, J, K, M** na řídicí jednotce Arduino pro zapojení semaforů

3. Zapojte konektory od semaforů pro auta (čísla 1, 2) a semaforů pro chodce (čísla 3, 4) do řídicí jednotky podle tabulky zapojení.

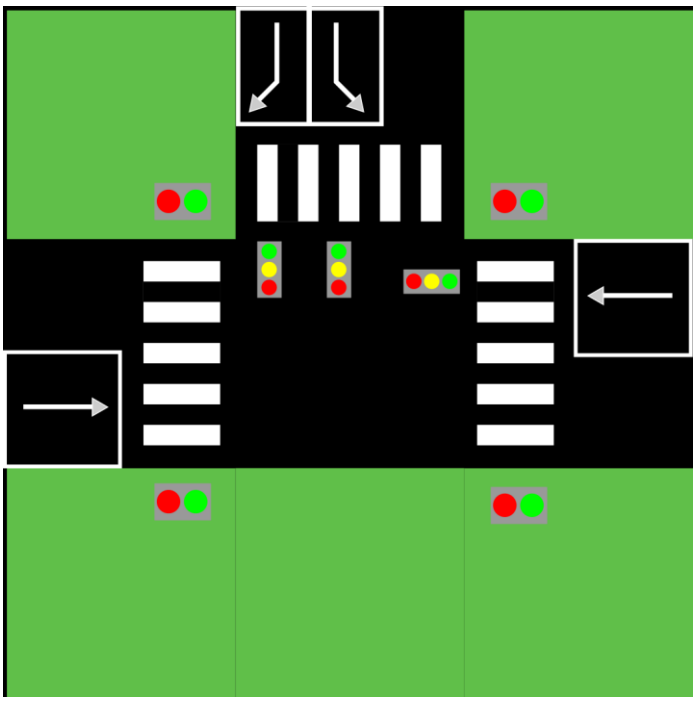
4. Vytvořte funkční program dle ukázkového programu nebo na základě vlastní představivosti pro funkce semaforů pro auta a chodce ve tvaru křižovatky písmena T.

5. Nahrajte program do řídicí jednotky.

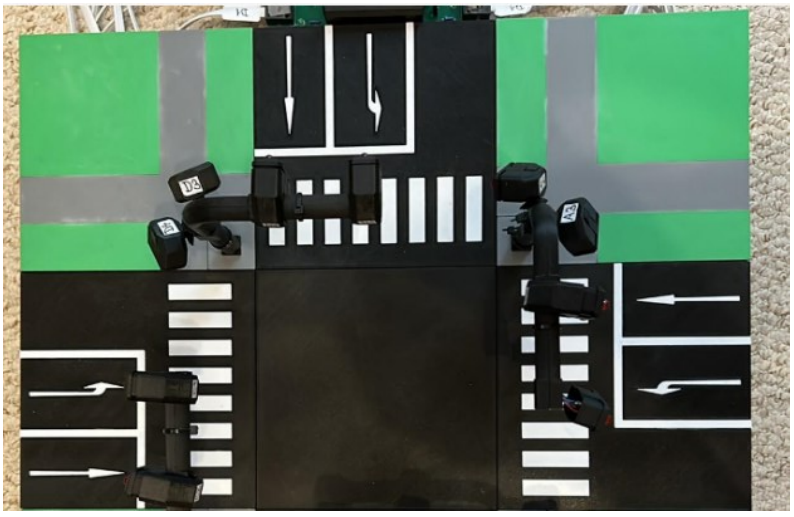
6. Otestujte a zkontrolujte vizuální funkci modelu semaforů pro auta a semaforů pro chodce.

2. Zvolení komponent a uchycení

Z dostupných komponent vyberte **4 semafory**. Poté všechny semafory upevněte do desek určených pro semafory. Následně kabely s konektory prostrčte pod ostatními deskami k řídicí jednotce.



Obrázek 71 Situační nákres



Obrázek 72 Vzhled křižovatky ve tvaru T

3. Zvolení portů semaforů

Po úspěšném uchycení semaforů najděte na desce porty s označením písmen **A, B, C, D, E, F, G, I, J, K, M**.

PORT A

Signalizace	Pin zapojení
Červená	35
Oranžová	37
Zelená	36

PORT B

Signalizace	Pin zapojení
Červená	33
Oranžová	34
Zelená	32

PORT C

Signalizace	Pin zapojení
Červená	31
Oranžová	29
Zelená	30

PORT D

Signalizace	Pin zapojení
Červená	26
Oranžová	28
Zelená	27

PORT E

Signalizace	Pin zapojení
Červená	22
Oranžová	25
Zelená	24

PORT F

Signalizace	Pin zapojení
Červená	4
Oranžová	3
Zelená	2

PORT G

Signalizace	Pin zapojení
Červená	7
Oranžová	5
Zelená	6

PORT I

Signalizace	Pin zapojení
Červená	A2
Oranžová	A0
Zelená	A1

PORT J

Signalizace	Pin zapojení
Červená	A5
Oranžová	A3
Zelená	A4

PORT K

Signalizace	Pin zapojení
Červená	A7
Oranžová	A8
Zelená	A6

PORT M

Signalizace	Pin zapojení
Červená	A12
Oranžová	A14
Zelená	A13

Obrázek 73 Porty s čísly pinů

Díky obrázku, vidíme čísla pinů, která budeme potřebovat během psaní programu pro správnou funkci semaforů.

Zapojení konektorů od semaforů do portů řídicí jednotky provedeme pomocí následující tabulky.

Číslo Portu	Číslo konektoru
A	B1
B	B2
C	B3
D	B4
E	A1
F	A2
G	A3
I	C4
J	C3
K	C2
M	D4

Tabulka 4 Zapojení konektorů do řídicí jednotky

4. Vytvoření funkčního programu

Nejprve je potřebné si definovat piny portů pomocí předchozího obrázku.

```
// definice pinů pro semafor A1 na port E
int semaforA1cervena = 22;
int semaforA1oranzova = 25;
int semaforA1zelena =  24;

// definice pinů pro semafor A2 na port F
int semaforA2cervena = 4;
int semaforA2oranzova = 3;
int semaforA2zelena = 2;

// definice pinů pro semafor A3 na port G
int semaforA3cervena = 7;
int semaforA3zelena = 6;

// definice pinů pro semafor B1 na port A
int semaforB1cervena = 35;
int semaforB1oranzova = 37;
int semaforB1zelena =  36;

// definice pinů pro semafor B2 na port B
int semaforB2cervena = 33;
int semaforB2oranzova = 34;
int semaforB2zelena = 32;

// definice pinů pro semafor B3 na port C
int semaforB3cervena = 31;
int semaforB3zelena = 30;

// definice pinů pro semafor B4 na port D
int semaforB4cervena = 26;
int semaforB4zelena = 27;

// definice pinů pro semafor C2 na port K
int semaforC2cervena = A7;
int semaforC2oranzova = A8;
int semaforC2zelena = A6;


// definice pinů pro semafor C3 na port J
int semaforC3cervena = A5;
int semaforC3zelena = A4;

// definice pinů pro semafor C4 na port I
int semaforC4cervena = A2;
int semaforC4zelena = A1;

// definice pinů pro semafor D4 na port M
int semaforD4cervena = A12;
int semaforD4zelena = A13;
```

Obrázek 74 Definice pinů zapojení

V druhém kroku je důležité definovat ve smyčce „**setup()**“ všechny předchozí piny jako výstupní a zapsat počáteční hodnoty semaforům pro auta.

```
void setup() {

  //nastavení pinů jako výstupy
  pinMode(semaforA1cervena, OUTPUT);
  pinMode(semaforA1oranzova, OUTPUT);
  pinMode(semaforA1zelena, OUTPUT);

  pinMode(semaforA2cervena, OUTPUT);
  pinMode(semaforA2oranzova, OUTPUT);
  pinMode(semaforA2zelena, OUTPUT);

  pinMode(semaforA3cervena, OUTPUT);
  pinMode(semaforA3zelena, OUTPUT);

  pinMode(semaforB1cervena, OUTPUT);
  pinMode(semaforB1oranzova, OUTPUT);
  pinMode(semaforB1zelena, OUTPUT);

  pinMode(semaforB2cervena, OUTPUT);
  pinMode(semaforB2oranzova, OUTPUT);
  pinMode(semaforB2zelena, OUTPUT);

  pinMode(semaforB3cervena, OUTPUT);
  pinMode(semaforB3zelena, OUTPUT);

  pinMode(semaforB4cervena, OUTPUT);
  pinMode(semaforB4zelena, OUTPUT);

  pinMode(semaforC2cervena, OUTPUT);
  pinMode(semaforC2oranzova, OUTPUT);
  pinMode(semaforC2zelena, OUTPUT);

  pinMode(semaforC3cervena, OUTPUT);
  pinMode(semaforC3zelena, OUTPUT);


  pinMode(semaforC4cervena, OUTPUT);
  pinMode(semaforC4zelena, OUTPUT);

  pinMode(semaforD4cervena, OUTPUT);
  pinMode(semaforD4zelena, OUTPUT);

  //zápis výchozích hodnot LED semaforů
  //pro správnou výchozí funkčnost
  digitalWrite(semaforA1cervena, LOW);
  digitalWrite(semaforA1oranzova, HIGH);
  digitalWrite(semaforA1zelena, HIGH);

  digitalWrite(semaforA2cervena, LOW);
  digitalWrite(semaforA2oranzova, HIGH);
  digitalWrite(semaforA2zelena, HIGH);

  digitalWrite(semaforC2cervena, HIGH);
  digitalWrite(semaforC2oranzova, HIGH);
  digitalWrite(semaforC2zelena, HIGH);

  digitalWrite(semaforB1cervena, LOW);
  digitalWrite(semaforB1oranzova, HIGH);
  digitalWrite(semaforB1zelena, HIGH);

  digitalWrite(semaforB2cervena, LOW);
  digitalWrite(semaforB2oranzova, HIGH);
  digitalWrite(semaforB2zelena, HIGH);


}
```

Obrázek 75 Smyčka setup()

V první části kódu jsou všechny definované piny nastaveny jako výstupní. V poslední části jsou na piny červených LED zapsány hodnoty LOW, aby nám při spuštění programu svítily červené LED diody. U ostatních LED diod (oranžová a zelená) jsou hodnoty nastavené na HIGH, díky kterým nám ostatní LED diody nesvítí.

```
void loop() {

  //volání funkcí
  semaforB1B2(2000, 2000, 4000); //volání funkce semaforu B pro auta

  semaforChodecA3B4C3D4(2000, 4000); //volání funkce semaforů pro chodce A3 B4 C3 a D4

  semaforA2C2C1(2000, 2000, 4000);   //volání funkce semaforů pro auta A2 C2 a C1

  semaforChodecB3C4(2000, 4000);     //volání funkce semaforů pro chodce B3 a C4

}
```

Obrázek 76 Smyčka loop()

Ve smyčce „**loop()**“ jsou volány čtyři funkce, první z nich je funkce **semaforB1B2** pro auta, jako druhá je volána funkce **semaforChodecA3B4C3D4** pro semafory chodců. Následuje funkce **semaforA2C2C1** pro semafory aut a jako poslední funkce je volána semaforChodecB3B4 pro funkci semaforů pro chodce. Veškeré funkce jsou napsány pod hlavní programovou smyčkou „**loop()**“.

```
void semaforB1B2(int cervenaB1, int oranzovaB1, int zelenaB1) {
  digitalWrite(semaforB1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaB1);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforB1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforB2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforB1cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB1zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforB2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaB1);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforB1zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 77 funkce semaforB1B2

```
void semaforChodecA3B4C3D4(int cervenaA3, int zelenaA3) {
  digitalWrite(semaforA3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA3);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforA3cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB4cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC3cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforD4cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA3zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforB4zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforC3zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforD4zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaA3);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforA3zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB4zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC3zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforD4zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 78 funkce semaforChodecA3B4C3D4

```
void semaforA2C2C1(int cervenaA2, int oranzovaA2, int zelenaA2)
{
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA2);
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);
  digitalWrite(semaforA2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(zelenaA2);
  digitalWrite(semaforA2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforAloranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforAlcervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforAloranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforAlzelena, LOW);    //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(zelenaA2);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforAlzelena, HIGH);   //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforAloranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA2);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforAloranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforAlcervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 79 funkce semaforA2C2C1

```
void semaforChodecB3C4(int cervenaB3, int zelenaB3) {
  digitalWrite(semaforB3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaB3);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforB3cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC4cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB3zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforC4zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaB3);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforB3zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC4zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB3cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC4cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
}
```

Obrázek 80 funkce semaforChodecB3C4

5. Nahrání programu do řídicí jednotky Arduino

Pokud je kód programu křižovatky ve tvaru T hotový, propojíme řídicí jednotku s počítačem pomocí kabelu.

Obrázek 81 Propojovací kabel mezi PC a řídicí jednotkou [38]

Po propojení počítače a řídicí jednotky zkontrolujeme propojení na portu USB a zvolenou desku mikrokontroléru Arduino Mega.

Kontrola funkčnosti naprogramovaného křižovatky ve tvaru T

Pokud jste úspěšně nahráli program do řídicí jednotky. Zkontrolujte, jestli zapojené semafory svítí a fungují dle Vašeho programu. Dle ukázkového programu nejprve svítí u všech semaforů červená LED dioda, poté se jako první zprovozní semafor B pro auta, Následují semafory pro chodce A3, B4, C3 a D4, tak aby nedošlo ke kolizi mezi semafory pro auta a chodce. Poté jsou spuštěny semafory pro auta s označením A2,C2 a C1. Jako poslední následuje funkce pro zbylé semafory pro chodce B3 a C4.

Bonusový úkol

♠ Změňte celkovou funkci křižovatky vlastním programem dle Vaší představivosti.

7.5 Úloha č. 5 - křižovatka ve tvaru X bez semaforů pro chodce

Cíle výuky:

Žáci rozvíjí získané předchozí znalosti.

Žáci budou schopni změnit program, aby fungoval na jiné hodnoty dat.

Žáci zvládnout upravit podmíněné příkazy if a else.

Pomůcky:

Model křižovatky s veškerými komponenty

Počítač s nainstalovaným grafickým prostředím Arduino IDE

Rozsah:

4x45 min.

Zadání:

1. Vyberte z dostupných komponent semafory a desky a následně semafory do desky připevněte.

2. Vyberte si porty **A, B, C, D, E, F, G, I, J, K, M** na řídicí jednotce Arduino pro zapojení semaforů

3. Zapojte konektory od semaforů pro auta (čísla 1, 2) a semaforů pro chodce (čísla 3, 4) do řídicí jednotky podle tabulky zapojení.

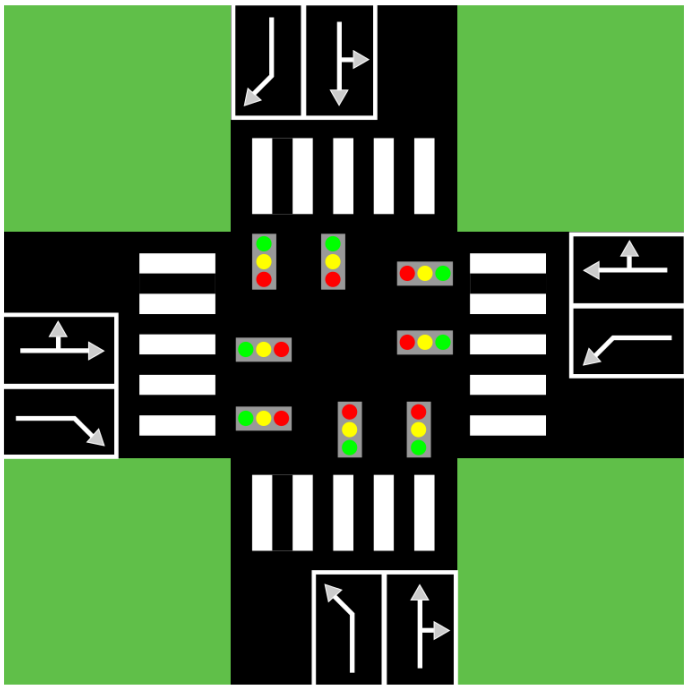
4. Vytvořte funkční program dle ukázkového programu nebo na základě vlastní představivosti pro funkce semaforů pro chodce a auta s použitím sériové komunikace.

5. Nahrajte program do řídicí jednotky.

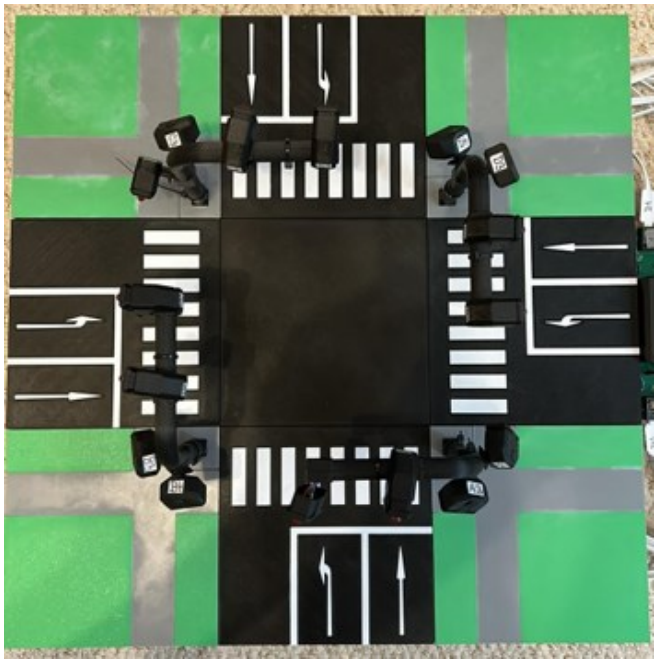
6. Otestujte a zkontrolujte vizuální funkci modelu semaforů pro auta a semaforů pro chodce.

2. Zvolení komponent a uchycení

Z dostupných komponent vyberte **4 semafory**. Poté všechny semafory upevněte do desek určených pro semafory. Následně kabely s konektory prostrčte pod ostatními deskami k řídicí jednotce.



Obrázek 82 Situační nákres



Obrázek 83 vzhled křižovatky ve tvaru X

3. Zvolení portů semaforů

Po úspěšném uchycení semaforů najděte na desce porty s označením písmen **A, B, C, D, E, F, G, H**.

PORT A	
Signalizace	Pin zapojení
Červená	35
Oranžová	37
Zelená	36

PORT B	
Signalizace	Pin zapojení
Červená	33
Oranžová	34
Zelená	32

PORT C	
Signalizace	Pin zapojení
Červená	31
Oranžová	29
Zelená	30

PORT D	
Signalizace	Pin zapojení
Červená	26
Oranžová	28
Zelená	27

PORT E	
Signalizace	Pin zapojení
Červená	22
Oranžová	25
Zelená	24

PORT F	
Signalizace	Pin zapojení
Červená	4
Oranžová	3
Zelená	2

PORT G	
Signalizace	Pin zapojení
Červená	7
Oranžová	5
Zelená	6

PORT H	
Signalizace	Pin zapojení
Červená	9
Oranžová	8
Zelená	10

Obrázek 84 Porty s čísly pinů

Díky obrázku, vidíme čísla pinů, která budeme potřebovat během psaní programu pro správnou funkci semaforů.

Zapojení konektorů od semaforů do portů řídicí jednotky provedeme pomocí následující tabulky.

Číslo Portu	Číslo konektoru
A	A1
B	A2
C	B1
D	B2
E	C1
F	C2
G	D1
H	D2

Tabulka 5 Zapojení konektorů do řídicí jednotky

4. Vytvoření funkčního programu

Nejprve definujeme piny portů pomocí předchozího obrázku.

```
//definice pinů semaforu A1 pro auta na portu A
int semaforA1cervena = 35;
int semaforA1oranzova = 37;
int semaforA1zelena =  36;

//definice pinů semaforu A2 pro auta na portu B
int semaforA2cervena = 33;
int semaforA2oranzova = 34;
int semaforA2zelena = 32;

//definice pinů semaforu B1 pro auta na portu C
int semaforB1cervena = 31;
int semaforB1oranzova = 29;
int semaforB1zelena =  30;

//definice pinů semaforu B2 pro auta na portu D
int semaforB2cervena = 26;
int semaforB2oranzova = 28;
int semaforB2zelena = 27;

//definice pinů semaforu C1 pro auta na portu E
int semaforC1cervena = 22;
int semaforC1oranzova = 25;
int semaforC1zelena =  24;

//definice pinů semaforu C2 pro auta na portu F
int semaforC2cervena = 4;
int semaforC2oranzova = 3;
int semaforC2zelena = 2;

//definice pinů semaforu D1 pro auta na portu G
int semaforD1cervena = 7;
int semaforD1oranzova = 5;
int semaforD1zelena =  6;

//definice pinů semaforu D2 pro auta na portu H
int semaforD2cervena = 9;
int semaforD2oranzova = 8;
int semaforD2zelena = 10;
```

Obrázek 85 Definice pinů zapojení

V druhém kroku je důležité definovat ve smyčce „**setup()**“ všechny předchozí piny jako výstupní a zapsat počáteční hodnoty semaforům pro auta, aby svítili pouze červené LED diody.

```
void setup() {
  //nastavení sériové komunikace (monitoru)
  Serial.begin(9600);

  //nastavení všech pinů jako výstup
  pinMode(semaforA1cervena, OUTPUT);
  pinMode(semaforA1oranzova, OUTPUT);
  pinMode(semaforA1zelena, OUTPUT);

  pinMode(semaforA2cervena, OUTPUT);
  pinMode(semaforA2oranzova, OUTPUT);
  pinMode(semaforA2zelena, OUTPUT);

  pinMode(semaforB1cervena, OUTPUT);
  pinMode(semaforB1oranzova, OUTPUT);
  pinMode(semaforB1zelena, OUTPUT);

  pinMode(semaforB2cervena, OUTPUT);
  pinMode(semaforB2oranzova, OUTPUT);
  pinMode(semaforB2zelena, OUTPUT);

  pinMode(semaforC1cervena, OUTPUT);
  pinMode(semaforC1oranzova, OUTPUT);
  pinMode(semaforC1zelena, OUTPUT);

  pinMode(semaforC2cervena, OUTPUT);
  pinMode(semaforC2oranzova, OUTPUT);
  pinMode(semaforC2zelena, OUTPUT);

  pinMode(semaforD1cervena, OUTPUT);
  pinMode(semaforD1oranzova, OUTPUT);
  pinMode(semaforD1zelena, OUTPUT);

  pinMode(semaforD2cervena, OUTPUT);
  pinMode(semaforD2oranzova, OUTPUT);
  pinMode(semaforD2zelena, OUTPUT);
```

```
  //počáteční nastavení hodnot semaforů pro červené LED diody - svítí
  digitalWrite(semaforA1cervena, LOW);
  digitalWrite(semaforA2cervena, LOW);
  digitalWrite(semaforB1cervena, LOW);
  digitalWrite(semaforB2cervena, LOW);
  digitalWrite(semaforC1cervena, LOW);
  digitalWrite(semaforC2cervena, LOW);
  digitalWrite(semaforD1cervena, LOW);
  digitalWrite(semaforD2cervena, LOW);

  //počáteční nastavení hodnot semaforů pro oranžové LED diody - nesvítí
  digitalWrite(semaforA1oranzova, HIGH);
  digitalWrite(semaforA2oranzova, HIGH);
  digitalWrite(semaforB1oranzova, HIGH);
  digitalWrite(semaforB2oranzova, HIGH);
  digitalWrite(semaforC1oranzova, HIGH);
  digitalWrite(semaforC2oranzova, HIGH);
  digitalWrite(semaforD1oranzova, HIGH);
  digitalWrite(semaforD2oranzova, HIGH);

  //počáteční nastavení hodnot semaforů pro zelené LED diody - nesvítí
  digitalWrite(semaforA1zelena, HIGH);
  digitalWrite(semaforA2zelena, HIGH);
  digitalWrite(semaforB1zelena, HIGH);
  digitalWrite(semaforB2zelena, HIGH);
  digitalWrite(semaforC1zelena, HIGH);
  digitalWrite(semaforC2zelena, HIGH);
  digitalWrite(semaforD1zelena, HIGH);
  digitalWrite(semaforD2zelena, HIGH);
}
```

Obrázek 86 Smyčka setup()

Nejdříve je nastavena komunikace se sériovým monitor pro zadávání příkazů mikrokontroléru, poté jsou všechny definované piny nastaveny jako výstupní. V poslední části jsou na piny červených LED zapsány hodnoty LOW, aby nám při spuštění programu svítily červené LED diody. U ostatních LED diod (oranžová a zelená) jsou hodnoty nastavené na hodnotu HIGH, díky kterým nám ostatní LED diody nesvítí.

```
void loop() {
  Serial.println("Zadejte písmeno semaforu:");
  while (Serial.available() == 0) {}     //cyklus while pro čekání na data
  String data = Serial.readString();
  if (data == "A") {                    //pokud je zadané písmeno A provede se funkce semaforA1A2
    semaforA1A2(2000, 2000, 4000);
  }
  else if (data == "B") {               //pokud je zadané písmeno B provede se funkce semaforB1B2
    semaforB1B2(2000, 2000, 4000);
  }
  else if (data == "C") {
    semaforC1C2(2000, 2000, 4000);      //pokud je zadané písmeno C provede se funkce semaforC1C2
  }
  else if (data == "D") {               //pokud je zadané písmeno B provede se funkce semaforD1D2
    semaforD1D2(2000, 2000, 4000);
  }else {
    Serial.println("Zadejte správné písmeno semaforu");
  }
}
```

Obrázek 87 Smyčka loop()

Ve smyčce „**loop()**“ jsou volány čtyři funkce. Tyto funkce jsou provedeny na základě rozhodovací podmínky if a else. Pokud zadáme do seriového monitoru jedno z písmen A,B,C nebo D, provede určitá funkce. V ukázkovém programu dojde ke spuštění jednoho ze semaforů, pokud uživatel zadá jedno z písmen A,B,C nebo D.

```
void semaforA1A2(int cervenaA1, int oranzovaA1, int zelenaA1) {
  Serial.println("Zadali jste semafor A. Prosím Čekejte na provedení funkce");
  digitalWrite(semaforA1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaA1);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforA1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA1cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforA1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA1zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforA2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaA1);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforA1zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforA1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforA2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaA1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforA1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforA1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforA2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 88 funkce semaforA1A2

```
void semaforB1B2(int cervenaB1, int oranzovaB1, int zelenaB1) {
  Serial.println("Zadali jste semafor B. Prosím Čekejte na provedení funkce");
  digitalWrite(semaforB1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaB1);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforB1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforB2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforB1cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforB1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB1zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforB2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaB1);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforB1zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforB1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforB2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforB1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforB1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforB2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 89 funkce semaforB1B2

```
void semaforC1C2(int cervenaC1, int oranzovaC1, int zelenaC1) {
  Serial.println("Zadali jste semafor C. Prosím Čekejte na provedení funkce");
  digitalWrite(semaforC1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaC1);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforC1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaB1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforC1cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforC1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC1zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforC2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaC1);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforC1zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforC1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforC2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaC1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforC1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforC1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforC2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 90 funkce semaforC1C2

```
void semaforD1D2(int cervenaD1, int oranzovaD1, int zelenaD1) {
  Serial.println("Zadali jste semafor D. Prosím Čekejte na provedení funkce");
  digitalWrite(semaforD1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  delay(cervenaD1);                      //zpoždění 2s - červená LED dioda svítí
  digitalWrite(semaforD1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforD2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaD1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforD1cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforD2cervena, HIGH);  //nastavení pinu na hodnotu 1 - LED červená nesvítí
  digitalWrite(semaforD1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforD2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforD1zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  digitalWrite(semaforD2zelena, LOW);    //nastavení pinu na hodnotu 0 - LED zelená svítí
  delay(zelenaD1);                       //zpoždění 4s - zelená LED dioda svítí
  digitalWrite(semaforD1zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforD2zelena, HIGH);   //nastavení pinu na hodnotu 1 - LED zelená nesvítí
  digitalWrite(semaforD1oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  digitalWrite(semaforD2oranzova, LOW);  //nastavení pinu na hodnotu 0 - LED oranžová svítí
  delay(oranzovaD1);                     //zpoždění 2s - oranžová LED dioda svítí
  digitalWrite(semaforD1oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforD2oranzova, HIGH); //nastavení pinu na hodnotu 1 - LED oranžová nesvítí
  digitalWrite(semaforD1cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  digitalWrite(semaforD2cervena, LOW);   //nastavení pinu na hodnotu 0 - LED červená svítí
  Serial.println("Dokončeno");
}
```

Obrázek 91 funkce semaforD1D2

5. Nahrání programu do řídicí jednotky Arduino

Pokud je kód programu křižovatky ve tvaru X hotový, propojíme řídicí jednotku s počítačem pomocí kabelu.

Obrázek 92 Propojovací kabel mezi PC a řídicí jednotkou [38]

Po propojení počítače a řídicí jednotky zkontrolujeme propojení na portu USB a zvolenou desku mikrokontroléru Arduino Mega.

Kontrola funkčnosti naprogramované křižovatky ve tvaru X bez semaforů pro chodce

Pokud jste úspěšně nahráli program do řídicí jednotky. Zkontrolujte, jestli zapojené semafory svítí a fungují dle Vašeho programu. Dle ukázkového programu nejprve svítí u všech semaforů červená LED dioda. Po zadání jednoho z písmen do sériového monitoru se dle ukázkového programu spustí příslušný semafor se stejným označením písmene jako v kódu programu.

Bonusový úkol

♠ Do ukázkového programu přidejte semafory chodců a následně zapojte všechny potřebné konektory od semaforů pro chodce do řídicí jednotky dle Vašeho programu. Program nahrajte do zařízení a vizuálně otestujte, jestli semafory pro chodce fungují dle Vašeho přidaného kódu programu.

ZÁVĚR

Cílem práce bylo navrhnout a sestavit funkční výukový model křižovatky v rámci vyučovacích hodin informatiky na střední škole. Navržení všech komponentů probíhalo v programu Autodesk invertor. Následně navržené komponenty byly vytisknuty na 3D tiskárně za použití filamentu typu PLA, který je plně biologicky odbouratelný. Model křižovatky je možné díky její variabilitě různých komponentů měnit. Vyučují má tedy více možností ke změně typu křižovatky.

Výukový model křižovatky je navržen tak, aby splňoval požadavky na novou informatiku, která se zavádí do výuky předmětu. Pomocí výukového modelu mohou žáci rozvíjet své dosavadní vědomosti, vylepšovat si kreativitu pomocí vlastního rozvržení křižovatky a učit se řešit problémy, se kterými se mohou potýkat během realizace křižovatky. Stejně tak má i možnost vyučující pozměnit vnitřní zapojení pinů pro změnění více variant programu. Model spíše patří do výuky informatiky na střední škole, avšak je možné použít jej ve výuce informatiky na základní škole. Jeho použití je důležité rozlišit dle zájmu žáků a také výkonnosti. Model klade důraz na znalost vyučujícího pedagoga v dané oblasti. Zde je tedy důležité, aby vyučující učitel měl určitý přehled v oblasti elektrotechniky a algoritmizace. Důležitou věcí je bezpečnost. Jelikož je řídicí jednotka připojena pomocí USB kabelu do počítače, žáci se nedostanou do kontaktu s vyšším napětím než 5V. V tomto případě se jedná o nízké a bezpečné napětí. Jako většina výukových modelů, lze i model křižovatky dále rozšiřovat například o tlačítka pro chodce, která nejsou součástí výukového modelu. Další možností je doplnění vzhledu křižovatky o modely aut nebo lavičky a okolní stromy, které lze vytisknout také na 3D tiskárně.

SEZNAM POUŽITÉ LITERATURY

[1] 301 Moved Permanently. 301 Moved Permanently [online]. Dostupné z: https://www.insgraf.cz/1021/Didakticke-pomucky-do-hodin-informatiky

[2] 301 Moved Permanently. 301 Moved Permanently [online]. Dostupné z: https://www.sj.news/virtualni-realita-jako-oficialni-soucast-vyuky-program-vr-schools-zaky-vtahne-do-taju-vesmiru-i-lidskeho-tela/

[3] Odborný článek: Výuka informatiky a ICT na SŠ v ČR – rok 2011. Metodický portál / Odborné články [online]. Dostupné z: https://clanky.rvp.cz/clanek/k/o/14359/VYUKA-INFORMATIKY-A-ICT-NA-SS-V-CR---ROK-2011.html

[4] Základní východiska a teze revizí ICT kurikula, Národní pedagogický institut České republiky (dříve Národní ústav pro vzdělávání). Národní pedagogický institut České republiky (dříve Národní ústav pro vzdělávání) [online]. Copyright © [cit. 02.05.2023]. Dostupné z: http://archiv-nuv.npi.cz/t/1-zakladni-vychodiska-a-teze-revizi-ict-kurikula.html

[5] RVP – Rámcové vzdělávací programy - edu.cz. edu.cz - Jednotný metodický portál MŠMT [online]. Copyright © 2022 [cit. 02.05.2023]. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/

[6] RVP G* - Rámcové vzdělávací programy pro gymnázia - edu.cz. edu.cz - Jednotný metodický portál MŠMT [online]. Copyright © 2022 [cit. 02.05.2023]. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcove-vzdelavaci-programy-pro-gymnazia-rvp-g/

[7] RVP SOV - Rámcové vzdělávací programy středního odborného vzdělávání - edu.cz. edu.cz - Jednotný metodický portál MŠMT [online]. Copyright © 2022 [cit. 02.05.2023]. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcove-vzdelavaci-programy-stredniho-odborneho-vzdelavani-rvp-sov/

[8] Odborný článek: ŠVP. Metodický portál / Odborné články [online]. Dostupné z: https://clanky.rvp.cz/clanek/c/ZI/412/SVP.html

[9] Arduino Tinkerkit Braccio robotická ruka - HWKITCHEN. Váš parťák ve světě tvoření | HWKitchen.cz [online]. Copyright © HWKITCHEN, všechna práva vyhrazena [cit. 13.05.2023]. Dostupné z: https://www.hwkitchen.cz/arduino-tinkerkit-braccio-roboticka-ruka/?gclid=EAIaIQobChMIz5DBo93v_gIVzO93Ch2QsAN8EAkYAiABEgIJa_D_BwE

[10] Makeblock Ultimate Robot Kit 2.0. RPishop.cz [online]. Copyright © Copyright 2023 RPishop.cz. [cit. 13.05.2023]. Dostupné z: https://rpishop.cz/mbot/1170-make-block-ultimate-robot-kit-20-6928819504424.html

[11] Lego Mindstorms – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Lego_Mindstorms

[12] 45678 LEGO® Education SPIKE™ Prime Základní souprava. EDUXE, distributor učebních pomůcek Brick Soft, DUPLO, LEGO, WeDo a MINDSTORMS [online]. Dostupné z: https://www.eduxe.cz/p/353/45678-spike-prime-zakladni-souprava

[13] Just a moment.... Just a moment... [online]. Dostupné z: https://www.mall.cz/lego-9/lego-mindstorms-51515-roboti-vynalezce?gclid=EAIaIQobChMIpO6Srbjw_gIVUgkGAB2OCgiAEAQYASABEgIzoPD_BwE

[14] BBC micro:bit V1.5 - mikropočítač pro výuku programování - HWKITCHEN. Váš parťák ve světě tvoření | HWKitchen.cz [online]. Copyright © HWKITCHEN, všechna práva vyhrazena [cit. 02.05.2023]. Dostupné z: https://www.hwkitchen.cz/bbc-microbit-mikropocitac-pro-vyuku-programovani/

[15] Raspberry Pi – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Raspberry_Pi

[16] Modul LED semafor | dratek.cz. dratek.cz: VELKOOBCHOD, MALOOBCHOD S ARDUINEM [online]. Copyright © Copyright ECLIPSERA s.r.o. [cit. 02.05.2023]. Dostupné z: https://dratek.cz/arduino/7719-modul-led-semafor.html?gclid=EAIaIQobChMIltvs5I3W_gIVWLnVCh3BfAiREAQYASABEgKg6fD_BwE

[17] Co je deska plošných spojů (DPS). KiCad [online]. Dostupné z: https://kicad.spseol.cz/cojedps.html

[18] THT vs SMT PCB Technology | Advantages & Disadvantages. Circuits DIY — Electronics Projects, Tutorials, Circuits & Datasheets [online]. Copyright © 2023 Circuits DIY [cit. 13.05.2023]. Dostupné z: https://www.circuits-diy.com/tht-vs-smt-pcb-technology-advantages-disadvantages/

[19] SURFACE MOUNT PROCESS - Surface Mount Process. SURFACE MOUNT PROCESS - Surface Mount Process [online]. Dostupné z: https://www.surfacemountprocess.com/

[20] Co je to LED dioda? - LEDme s. r. o.. LED osvětlení - Velkoobchod, maloobchod [online]. Copyright © All rights reserved. [cit. 13.05.2023]. Dostupné z: https://ledme.cz/textove-novinky/clanky/co-je-LED-dioda

[21] Rezistory, ich hodnotové rady, tolerancie a označenie. Elektrolab.eu - Predaj a nakúp elektronické komponenty, spotrebnú... [online]. Copyright © 2023 Elektrolab.eu. Všetky práva vyhradené. Vytvoril [cit. 13.05.2023]. Dostupné z: https://www.elektrolab.eu/blog/rezistory-ich-hodnotove-rady-tolerancie-a-oznacenie

[22] Just a moment.... Just a moment... [online]. Dostupné z: https://eepower.com/resistor-guide/resistor-standards-and-codes/resistor-color-code/#

[23] Difference Between RJ11 and RJ12 | Difference Between. Difference Between Similar Terms and Objects [online]. Dostupné z: http://www.differencebetween.net/technology/difference-between-rj11-and-rj12/

[24] What is Arduino? | Arduino Documentation. Arduino Docs | Arduino Documentation [online]. Dostupné z: https://docs.arduino.cc/learn/starting-guide/whats-arduino

[25] LilyPad Arduino Main Board | Arduino Documentation. Arduino Docs | Arduino Documentation [online]. Dostupné z: https://docs.arduino.cc/retired/boards/lilypad-arduino-main-board

[26] Access denied. Access denied [online]. Dostupné z: https://store-usa.arduino.cc/products/arduino-due

[27] Access denied. Access denied [online]. Dostupné z: https://store.arduino.cc/products/arduino-uno-rev3

[28] Access denied. Access denied [online]. Dostupné z: https://store.arduino.cc/products/arduino-nano

[29] Access denied. Access denied [online]. Dostupné z: https://store.arduino.cc/products/arduino-mega-2560-rev3

[30] EAGLE | PCB Design And Electrical Schematic Software | Autodesk. Autodesk India | 3D Design, Engineering & Entertainment Software [online]. Copyright © 2023 Autodesk Inc. All rights reserved [cit. 13.05.2023]. Dostupné z: https://www.autodesk.in/products/eagle/overview?term=1-YEAR&tab=subscription

[31] LADMAN, Josef. Elektronické konstrukce pro začátečníky. Praha: BEN - technická literatura, 2001. ISBN 80-730-0015-6.

[32] PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.

[33] CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.

[34] BENEŠ, Pavel. Automatizace a automatizační technika: prostředky automatizační techniky. 5., rozš. a aktualiz. vyd. Brno: Computer Press, 2014. ISBN

[35] 9788025137475

[36] Úvod do programování mikrokontroléru – Zonepi blog. Zonepi blog – Zprávy, recenze, návody a projekty ze světa mini počítačů. [online]. Copyright © 2023 [cit. 13.05.2023]. Dostupné z: https://blog.zonepi.cz/maker-uno-ve-vyuce-2-5-uvod-do-programovani-vystupu/

[37] MCCONNELL, Steve. Dokonalý kód: umění programování a techniky tvorby software. Vyd. 1. Brno: Computer Press, 2005, 894 s. ISBN 802510849x.

[38] PremiumCord stíněný kabel USB 2.0 A-B 5m | Smarty.cz. Stačí pípnout | Smarty.cz [online]. Copyright © Smarty.cz [cit. 13.05.2023]. Dostupné z: https://www.smarty.cz/PremiumCord-stineny-kabel-USB-2-0-A-B-5m-p76211

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ICT	Informační a komunikační technologie.
RVP	Rámcový vzdělávací program
ŠVP	Školní vzdělávací program
LED	Elektroluminiscenční dioda
DPS	Deska plošných spojů
SMT	Způsob osazování desky plošných spojů
THT	Způsob osazování desky plošných spojů
ARM	Architektura procesoru
V	Jednotka elektrického napětí
A	Jednotka elektrického proudu
IDE	Vývojové prostředí
USB	Univerzální sériová sběrnice
HDMI	Označení nekomprimovaného obrazového a zvukového signálu v digitálním formátu

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH