UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

Hana Čáslavková

Bitbeam, opensourcová stavebnice, pro výuku robotiky na základních školách

Olomouc 2024

vedoucí práce: Mgr. Radim Děrda

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a uvedla jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použila.

V Olomouci dne 02. 04. 2024 …….

Hana Čáslavková

Poděkování

Ráda bych poděkovala mému vedoucímu bakalářské práce panu Mgr. Radimu Děrdovi za odborné vedení mé bakalářské práce, poskytnutí materiálů pro mou tvorbu, trpělivost a ochotu při konzultacích a množství cenných rad a doporučení při zpracovávání této práce.

Hana Čáslavková

Anotace

Jméno a příjmení:	Hana Čáslavková
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	Mgr. Radim Děrda
Rok obhajoby:	2024

Název práce:	Bitbeam, opensourcová stavebnice, pro výuku robotiky na základních školách
Název v angličtině:	Bitbeam, The Open Source Building Toy, for Teaching Robotics at Primary Schools
Anotace práce:	Hlavním cílem bakalářské práce je seznámit učitele s alternativní možností výuky robotiky prostřednictvím opensourcové stavebnice Bitbeam4. Součástí práce je návrh konstrukce robota vhodného pro disciplínu Line Follower, stavební návod, úplný souhrn STL souborů potřebných pro vytištění všech dílů robota a ukázkový program ve Scratch a C++.
Klíčová slova:	Bitbeam4, Sledovač čáry, Objíždění překážky, 3D tisk, Scratch, C++, Stavba robota, Arduino
Anotace v angličtině:	The main aim of the bachelor thesis is to introduce teachers to an alternative way of teaching robotics through the Bitbeam4 open-source kit. The thesis includes the design of a robot suitable for the Line Follower discipline, construction instructions, a complete set of STL files needed to print all parts of the robot and a sample program in Scratch and C++.
Klíčová slova v angličtině:	Bitbeam4, Line Follower, Obstacle avoidance, 3D printing, Scratch, C++, Robot construction, Arduino
Přílohy vázané v práci:	8 příloh: *Souhrn STL souborů potřebných pro vytištění všech dílů robota* *Návod na sestavení robota*

	Ukázkový program ve Scratchi – dvoustavové sledování čáry a objetí překážky *Ukázkový program v C++ – dvoustavové sledování čáry a objetí překážky* *Ukázkový program ve Scratchi – P-regulace* *Ukázkový program v C++ – P-regulace* *Ukázkový program ve Scratchi – P-regulace a objíždění překážky* *Ukázkový program v C++ – P-regulace a objíždění překážky*
Rozsah práce:	60 stran
Jazyk práce:	čeština

Obsah

Úvod

Robotické technologie hrají v našich životech stále důležitější roli. S různými roboty a naprogramovanými zařízeními se setkáváme takřka denně. Na rostoucí počet těchto technologií zareagovalo i Ministerstvo školství, mládeže a tělovýchovy. V roce 2021 totiž schválilo dlouho očekávaný zrevidovaný rámcově vzdělávací program, neboli RVP ZV, s novinkami ve vzdělávací oblasti Informatiky. Podle nové informatiky by se tedy žáci mohli zabývat již na prvním stupni například programováním, algoritmizací, modelováním či robotikou. S tím však nastává pro mnohé školy problém, kde vzít finanční prostředky na mnoho nových technologií, které k výuce nové informatiky potřebují.

Cílem této bakalářské práce je tedy seznámit učitele s alternativní možností výuky robotiky prostřednictvím opensourcové stavebnice Bitbeam4. Z této stavebnice si žáci mohou jednoduše postavit jakéhokoliv robota. Tato stavebnice by také mohla školám poskytnout levnější variantu k jiným dražším robotickým stavebnicím na trhu. Bitbeam4 je totiž open source stavebnice, což znamená že je zdarma dostupná všem. Stačí si tedy stáhnout STL soubory dílků, vytisknout si je na 3D tiskárně a můžeme začít stavět robota.

Po přečtení této bakalářské práce budeme schopni právě jednoho takového robota od základu vyrobit. Řekneme si podrobnější informace o stavebnici Bitbeam4, ze které robota budeme stavět. Poté si zmíníme, jak tuto robotickou stavebnici využít ve školách a zaměříme se podrobněji na změny v RVP ZV v oblasti Informatiky. V dalších kapitolách si představíme programovací jazyky, pomocí kterých je vhodné na základních školách roboty programovat. Začneme programovacím jazykem vhodným pro začátečníky, tedy Scratchem, a na to navážeme již pokročilejším C++.

Následně si popíšeme podrobně jednotlivé elektrosoučástky, jež je vhodné použít při stavbě jednoduchého robota. Náš robot bude mít za úkol sledovat černou čáru na bílém povrchu a objíždět překážku. Proto budeme potřebovat jeden infračervený optický senzor pro sledování čáry a jeden ultrazvukový snímač vzdálenosti pro detekci překážky, vše bude poháněno pomocí dvou stejnosměrných motorů s převodovkou.

Poté si řekneme, jak pro takového robota vyrobit kostru. Řekneme si tedy několik informací o 3D modelovacích programech, o 3D tisku, o 3D tiskárnách, ale i o vhodných filamentech. Pro zajímavost si zde také vypočítáme náklady na sestavení jednoho takového robota. Cena je opravdu velmi nízká, proto by tato stavebnice mohla být vhodnou alternativou k aktuálně používaným stavebnicím ve školách. Nyní se ve školách nachází pouze několik robotických stavebnic a žáci musí pracovat mnohdy v několika členných skupinách, protože

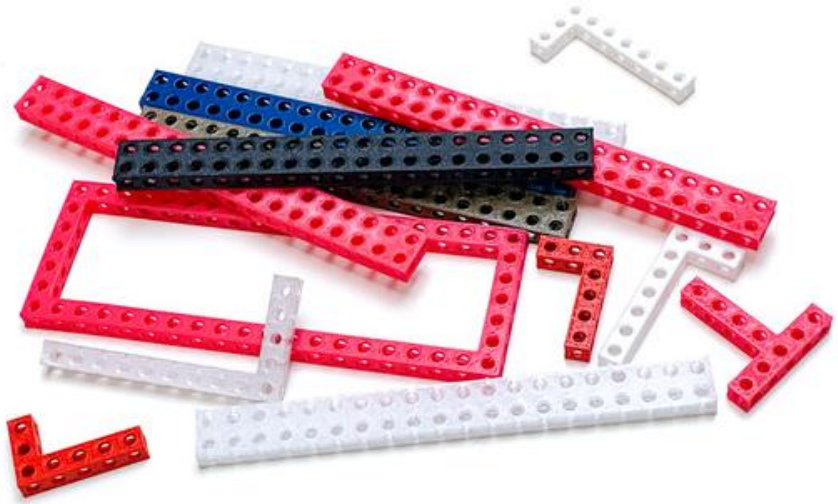
pro školu není možné kvůli vysoké ceně koupit stavebnic tolik, aby měl každý žák svou. Každý žák by však chtěl možná stavět jiného robota, s jinými funkcemi či ho jinak naprogramovat, a to by právě mohla umožnit tato open source stavebnice Bitbeam4. Její výrobní cena je tak malá, že by škola mohla těchto stavebnic mít více a každý žák si tak sestavit svého vlastního robota. Během výroby si kromě zručnosti při sestavování rozšíří své znalosti o 3D tisk či o 3D modelování, může se z toho tedy stát zábavný a komplexní projekt.

Ke konci bakalářské práce, tedy po vytisknutí těchto dílků stavebnice, se zaměříme na to, jak robota sestavit. Nalezneme zde tedy podrobný návod pro sestavení robota. Naše autíčko však musíme doplnit také o již dříve zmíněné elektrosoučástky, v další kapitole zde tedy bude podrobný popis a schéma, jak celého robota zapojit. V závěru této bakalářské práce si mimo jiné vysvětlíme, jak takového robota naprogramovat. Vysvětlíme si, jak funguje program pro sledování čáry i pro objíždění překážky. Nalezneme zde také ukázkové programy v jednodušším programovacím jazyce Scratch, ale i v pokročilejším C++.

1 Open source stavebnice Bitbeam4

Cílem této bakalářské práce je především seznámit učitele s alternativní možností výuky robotiky prostřednictvím stavebnice Bitbeam4, ze které si žáci mohou snadno postavit jakéhokoliv robota přesně podle jejich představ. V první kapitole si tedy pojďme představit tuto novou open sourcovou stavebnici Bitbeam4. Podíváme se také blíže na její využití a dozvíme se, proč tuto stavebnici zařadit do výuky programování, algoritmizace a robotiky na školách.

Stavebnice Bitbeam4 by mohla v budoucnu úspěšně sloužit jako levnější a funkční alternativa k různým dražším, a tím pro školy méně dostupným, stavebnicím na trhu. Bitbeam4 je totiž Open Source stavebnice, tedy je dostupná všem a zcela zdarma. Všechny dílky stavebnice jsou navržené tak, aby byly rozměrově normalizované s ostatními. Lze je tedy různě mezi sebou kombinovat. Součástky je možné volně stahovat a upravovat na mnoha platformách, jelikož komunita lidí, využívající Bitbeam4 a sdílející své nápady a zkušenosti, se stále rozrůstá. Všechny součástky jsou důmyslně promyšlené a navržené tak, aby si je mohl každý žák či učitel jednoduše vytisknout či vytvořit z různých materiálů, například pomocí 3D tisku nebo vyřezávání laserem do dřevěných desek. Další výhodou může být také rozměrová kompatibilita s komerčními stavebnicemi jako je například Lego Mindstorms a Technic. Výška a šířka základní jednotky kostiček je 8 mm a kulaté otvory jsou od sebe vzdáleny vždy taktéž 8 mm. Průměr díry je standardizovaný na 4,2 mm, přesně jako u Lego Technic. Vzniká také mnoho stavebních návodů, podle kterých si žáci mohou vytvořit téměř cokoliv, od auta přes jeřáb s pohyblivým ramenem až po různé vychytávky jako jsou různé držáky, krabičky a podobně (RoboTrip, 2023; TFs m-BITBEAM, 2023; Bitbeam, 2013).



Obr. 1: Bitbeam4

(Zdroj: vlastní obrázek)

1.1 Využití ve školách

Na začátku této kapitoly jsme se již dozvěděli, co je stavebnice Bitbeam4 a jaké jsou její výhody. Nyní se pojďme podívat na její uplatnění ve školách, jelikož lze předpokládat, že využití robotických stavebnic bude v hodinách informatiky stále hojnější. V roce 2021 Ministerstvo školství, mládeže a tělovýchovy totiž schválilo dlouho očekávaný nový zrevidovaný rámcově vzdělávací program neboli RVP. Tato revize RVP ZV přinesla mnoho novinek, o čemž není pochyb, jelikož RVP ZV byl poprvé vydán v roce 2005 a od té doby se pravidelně modernizuje a vyvíjí, avšak tyto zásahy se zcela vyhýbaly vzdělávacímu obsahu v oblasti Informační a komunikační technologie. Z čehož jasně plyne, že informatika se do této doby učila podle zastaralého dokumentu, který vznikl v roce 2005, pro představu si můžeme zmínit, že v této době ještě ani v České republice nebyl Gmail či YouTube a začínají se teprve objevovat první sociální sítě (Edu.cz, 2023).

Nová informatika v RVP ZV

Revize RVP přinesla zcela novou podobu informatiky. První školy začaly učit v souladu se zrevidovaným RVP již v září roku 2021. Povinnost přizpůsobit výuku nově zveřejněnému RVP ZV však mají všechny základní školy, na 1. stupni od 1. září 2023. O rok později, tedy od 1. září 2024, vstoupí v platnost tato podmínka i pro výuku na druhém stupni (Edu.cz, 2023; Ginterová, 2021).

Zrevidované RVP ZV obsahuje mnoho změn a novinek. Hlavním cílem takto zrevidovaného RVP je především rozvíjet informatické myšlení a zlepšit digitální kompetence žáků a povědomí o digitálních technologiích. Došlo mimo jiné také k pozměnění názvu celé vzdělávací oblasti, z Informačních a komunikačních technologií pouze na Informatiku (Edu.cz, 2023).

Výrazné zásahy si vyžádal především vzdělávací obsah. Do zlomového roku 2021 se měli žáci naučit pouze základy z využívání, fungování a architektury počítače a naučit se ho aktivně používat. Další náplní bylo například vyhledávání informací na internetu, které poté žáci dále zpracovávali. Na druhém stupni základního vzdělávání se opět vraceli k práci s informacemi a přidali pouze komunikaci. Abychom si to shrnuli, žáci by po dokončení základní školy podle starého RVP měli umět pouze zapnout a použít počítač, vědět z čeho se stolní počítač skládá, poslat e-mail, upravit obrázek či zpracovávat informace v textových a tabulkových editorech a následně zjištěné informace přenést do prezentace (RVP ZV, 2021).

To se však pro dnešní dobu plnou technologií zdá nedostatečné. Odborníci usoudili, že zpracovávání zjištěných informací v textových editorech již nestačí, a proto je v nově zrevidovaném RVP ZV vzdělávací obsah mnohem modernější, a dokonce rozšířený o mnoho nových činností. Podle nové informatiky by se tedy žáci mohli zabývat již na prvním stupni například programováním, algoritmizací či modelováním. Již žáci prvního stupně by tedy měli být po revizi RVP schopni pomocí zábavných aktivit a různých her určit a popsat problém, zanalyzovat ho a pokusit se nalézt vhodné řešení. Své vymyšlené algoritmy si poté budou ověřovat ve vhodných programovacích prostředích jako je například Scratch. S rostoucím počtem digitálních technologií a hodin, které již menší děti tráví na internetu a sociálních sítích, přichází také reakce Ministerstva školství, mládeže a tělovýchovy v podobě nové časové dotace věnované bezpečnosti na internetu a prevenci rizikového chování dětí na internetu a sociálních sítích. Žáci se tedy již v poměrně nízkém věku naučí bezpečně zacházet s digitálními technologiemi a pohybovat se ve virtuálním prostředí (RVP ZV, 2021).

Na druhém stupni základního vzdělávání se bude dále prohlubovat informatické myšlení, tedy žáci se naučí posuzovat různá řešení problému a poté vybrat to nejvhodnější. Dále se také naučí například plánovat činnosti, popisovat různé postupy či používat programovací jazyky. Rozšíří si opět své obzory v oblasti fungování digitálních technologií. Naučí se rozhodovat a posuzovat, kdy je vhodné daný postup zautomatizovat, které části postupu jsou stěžejní a které se naopak dají vynechat. Díky těmto činnostem na základních školách tedy nově získají žáci základní představu o programování, automatizaci, modelování či bezpečnosti na internetu. Budou tedy lépe připraveni na řešení každodenních problémů v reálném světě a na své budoucí povolání (RVP ZV, 2021; Informatické myšlení, 2018).

V rámcovém vzdělávacím programu z roku 2021 si taktéž můžeme všimnout upravené časové dotace v oblasti informatiky. Podle staré verze RVP byla povinná pouze 1 hodina informačních technologií na prvním stupni základního vzdělávání a ve stejném rozsahu také na stupni druhém. Nyní je vše ale jinak a s rozšířením vzdělávacího obsahu došlo k výraznému navýšení časových dotací. Informatika musí být nově vyučována minimálně ve čtvrtém a pátém ročníku a dále ve všech ročnících druhého stupně (RVP ZV, 2021).

V této kapitole jsme si představili Open Source stavebnici Bitbeam4, která má široké uplatnění. Dozvěděli jsme se, jak ji můžeme využít například v robotice, která se bude na základních školách po revizi RVP ZV v hodinách Informatiky vyučovat nyní častěji než kdy dřív.

2 Programovací jazyky

V předchozí kapitole jsme se dozvěděli základní informace o open-sourcové stavebnici Bitbeam4 a o rostoucím podílu hodin algoritmizace a robotizace na českých školách.

V této kapitole se podíváme blíže na programovací jazyky, které se nejčastěji používají na základních školách. Začneme od nejlehčí varianty, od Scratche, a postupně se dostaneme až k pokročilému programovacímu jazyku, k C++, který se pro programování na mikrokontroleru Arduino nejčastěji označuje jako Wiring. (Arduino.cc, 2024 a; Malý, 2017, s. 141)

2.1 Scratch

Podle vývojářského týmu Scratch a Randákové je Scratch velmi snadný vizuální programovací jazyk vhodný pro všechny začátečníky. Je navržen především pro děti ve věku 8 až 16 let. Pro děti od 5 do 7 let lze použít zjednodušenou verzi Sratche, tzv. ScratchJr (Marji, 2014, str. 1-3; Scratch).

Jedná se o programovací jazyk založený na blocích, který vyvinula skupina Lifelong Kindergarten Group při Massachusettském technologickém institutu (MIT). Žáci nemusí psát žádný text, ale pracují s obrázky a bloky. Předem připravené příkazy žáci pouze ve správném pořadí umisťují do řady za sebe. Výhodou tohoto zápisu tedy je, že žáci nemohou udělat zbytečné chyby v zápisu příkazu. Odpadávají problémy se zapomenutým středníkem, zapomenutou čárkou, vynechanými uvozovkami či špatně zapsaným příkazem či názvem proměnné. Psaní kódu ve Scratchi lze tedy přirovnat ke skládání puzzle (Marji, 2014, str. 1-3; Randáková).

Použití Scratche je velmi snadné, funguje zdarma online na většině současných webových prohlížečích či lze bezplatně stáhnout aplikaci. Může být tedy využíván nejen žáky, ale kýmkoliv, a nejen ve škole, ale klidně i doma či v knihovně. Žáci v tomto prostředí se širokou základnou online společenství můžou navrhovat a programovat vlastní hry a animace a navzájem si je sdílet s lidmi po celém světě. Je zde dostupných mnoho postav a funkcí. Dalších výhodou je také možnost volby českého jazyka. Jedná se tedy o skvělý nástroj na výuku programování a rozvoj informatického myšlení pro žáky již od útlého věku. (Randáková; Scratch).

Pro programování robota na základní škole lze využít Scratch v rámci velmi všestranné aplikace od Autodesk, v programu Tinkercad. Je zde totiž mimo jiné možné virtuálně zapojovat elektrické obvody. V nabídce komponentů nalezneme i většinu součástek potřebných pro

stavbu jednoduchého robota. Žáci si tedy mohou nejprve virtuálně zapojit robota, včetně motorů, čidel pro sledování čáry i pro snímání vzdálenosti od překážky. Takto zapojený robot může být v Tinkercadu následně i naprogramován. Máme zde možnost psát kód nejen ve Scratchi, ale také v C++. Mimo jiné lze zde kreslit i 3D modely, které si můžeme následně vytisknout na 3D tiskárně. V tomto programu lze tedy velmi snadno s žáky vytvořit celého robota, od navržení a vymodelování dílků pro sestavení robota, přes jeho zapojení, až po samotné naprogramování (Autodesk, 2024 a).

2.2 C++

Pro pokročilejší žáky můžeme na programování mikrokontroleru Arduino použít programovací jazyk C++. První nevýhodou však oproti Scratchi může být, že syntax jazyka vychází z angličtiny, takže výhodu budou mít žáci s dobrou znalostí anglického jazyka. Dalším poměrně velkým rozdílem je, že žáci musí psát kód v textových příkazech a dokonale dodržovat syntaxi. Avšak i kód v C++ můžeme i nadále psát v lehkém a intuitivním vývojovém prostředí Tinkercad. Pokročilejší variantou je poté Arduino IDE, které si blíže popíšeme dále (ITnetwork.cz, 2024 a).

C++ je stále populárnější programovací jazyk. Používá se k vytváření profesionálních počítačových programů a je jedním z nejpoužívanějších jazyků při vývoji her. C++ je multiplatformní jazyk, který lze použít k vytváření vysoce výkonných aplikací. Vyvinul ho Bjarne Stroustrup jako rozšíření jazyka C. Jazyk byl za svou dobu již čtyřikrát aktualizován, v letech 2011, 2014, 2017 a 2020 na C++11, C++14, C++17, C++20. Programovací jazyk C++ byl vyvinut jako rozšíření jazyka C , oba jazyky mají tedy téměř stejnou syntaxi, avšak je zde zásadní rozdíl, C++ podporuje třídy a objekty, zatímco C ne (W3schools, 2024).

Nyní se podíváme, proč je důležité žáky s tímto jazykem seznámit. C++ je jedním z nejpopulárnějších programovacích jazyků na světě. Jedná se o objektově orientovaný programovací jazyk, který dává programům jasnou strukturu. Je tedy velmi snadné programy přepsat, upravit a znovu využít. Tím klesají náklady na vývoj programů. C++ se podobá C, C# či Javě, je tedy poměrně snadné mezi těmito programovacími jazyky navzájem přecházet (W3schools, 2024).

V této kapitole jsme si porovnali dva oblíbené programovací jazyky vyučované na základních školách. Řekli jsme si jejich výhody a nevýhody. Představili jsme si jeden programovací jazyk vhodný pro začátečníky, tedy Scratch a jeden pokročilý programovací jazyk nesoucí název C++.

V následující kapitole si uvedeme několik elektrosoučástek potřebných pro sestavení jednoduchého robota, kterého zvládnou smontovat i úplní začátečníci. U jednotlivých součástek si také vysvětlíme princip jejich fungování, jak je zapojit a jak je vhodně napájet. Dozvíme se také, k čemu tyto elektrosoučástky můžou běžně sloužit, ale také si uvedeme příklad využití přímo v robotice.

3 Elektronika

V minulé kapitole jsme se dozvěděli, jak můžeme s žáky robota programovat. Blíže jsme si rozebrali rozdíly mezi lehkým vizuálním programovacím jazykem Scratchem a pokročilým textovým programovacím jazykem C++. V této kapitole si ukážeme, jaké elektrosoučástky je vhodné použít při stavbě našeho robota. Výčet použitých dílků na robotovi si rozšíříme o základní parametry a pravidla použití daných elektrosoučástek.

3.1 Řídící mikrokontroler

Mikrokontroler, jinými slovy jednočipový počítač, je malý integrovaný obvod, který obsahuje procesor, různé paměti a další periferie. My se podrobněji zaměříme na jednočipové počítače značky Arduino. Mozkem populárních desek od značky Arduino, tedy Arduina UNO a Arduina NANO je mikrokontroler ATMega328P. Při bližším pohledu na desku ho nemůžeme přehlédnout, mikrokontroler je ten největší čtvercový černý integrovaný obvod v samotném centru desky. Čip ATMega328P obsahuje procesor (8bitový AVR mikrokontrolér), různé paměti (SRAM, EEPROM nebo flash) a další periferie. My si nyní blíže popíšeme zajímavou paměť flash, na níž je stále uložený námi nahraný program, dokonce i po odpojení od napájení. V této paměti jsou kromě námi nahraného programu také 2 základní původní programy a to sice bootloader a program, který zajišťuje blikání LED diody na desce. Bootloader je možné vysvětlit jako program, který naběhne pokaždé automaticky ihned po zapojení desky k napájení či po stisku restartovacího tlačítka na desce. Jeho úkolem je rozpoznat, zda nechceme do paměti přes USB převodník nahrát nový program. V případě, že jsme spustili nahrávání, tak bootloader začne ukládat náš nový program na správné místo v paměti a dojde také k přehrání staršího programu. V opačném případě, tedy když se žádný program nahrávat nezačne, tak chvíli setrvá a pokud stále od USB převodníku nedostal příkaz o nahrávání, tak spustí automaticky poslední námi nahraný program, který má uložený na své flash paměti. Což nám umožňuje spustit program na Ardinu automaticky bez jakéhokoliv nastavování či manuálního spouštění pouze po připojení do powerbanky (Voda, 2019; Atmel Corporation, 2015).

Arduino

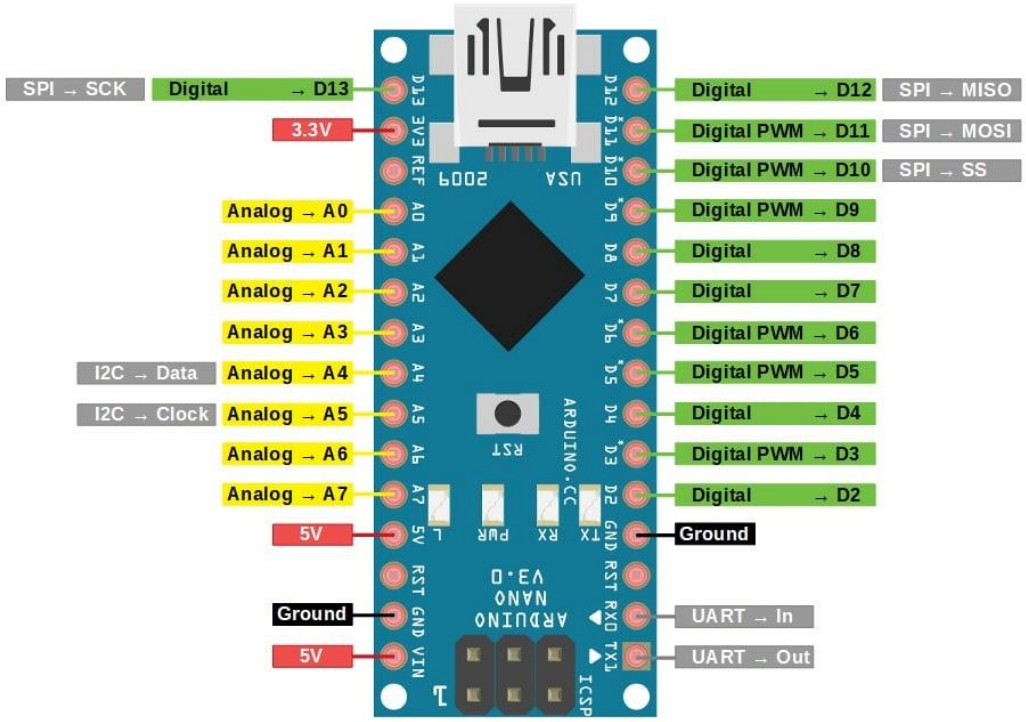
Pojďme si nyní něco říct o samotném Arduinu. Vývoj úplně prvního Arduina započal již v roce 2005, když se lidé v italském městečku Ivrea z institutu Interaction Design Institute odhodlali vyvinout cenově dostupný studentský set a ulehčit tak výuku programávní. Vše

vzniklo především z toho důvodu, že si spousta studentů nemohlo finančně dovolit jiné drahé programovatelné desky. Mezi studenty postupně rostla popularita desek Arduino a to jeho tvůrce pobídlo k tomu, aby se o svůj projekt podělili i s dalšími lidmi po celém světě. Jelikož se jedná o Open Source projekt, vzniká společně s oficiální řadou desek Arduino také spoustu neoficiálních desek, tak zvaných klonů (Voda, 2018, str. 12)

3.1.1 Arduino NANO

Arduino NANO je jedna z verzí desky Arduino. Jedná se o jednu z nejmenších desek v rodině Arduino. Pro svou malou velikost, pouhých 18 mm x 45 mm, a váhu jen 17 g je velmi oblíbená, je proto také častou volbou pro projekty, které vyžadují málo rozměrné a kompaktní řešení. Často je tedy využívána v robotech či v různých drobných zařízeních. Navzdory takto malým rozměrům se jedná o velmi kompaktně zařízenou desku, kterou lze přirovnat vybavením k populárnímu Arduinu UNO. Nabízí podobné množství pinů jako jiné Arduino desky. Obvykle má 8 analogových (A0 – A7) a 14 digitálních (D0 – D13) pinů, které mohou sloužit buď jako vstup či výstup. A 6 z nich může fungovat dokonce jako PWM výstupy, konkrétně piny s čísly D3, D5, D6, D9, D10, D11) (Arduino.cc, 2024 b; HWKITCHEN, 2024).

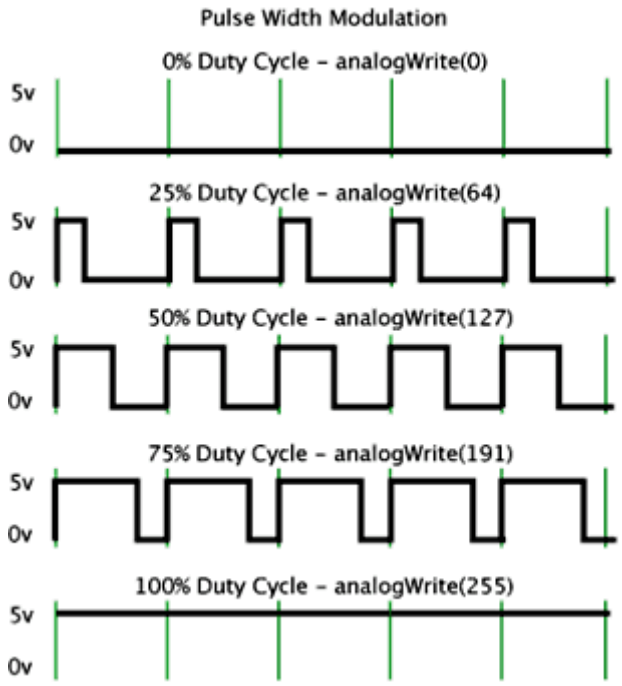
.



Obr. 2: deska Arduino NANO pinout

(Zdroj: https://robu.in/wp-content/uploads/2020/07/nano-pinout.jpg)

PWM, neboli jinými slovy pulzně šířková modulace, by se dala vysvětlit jako přechod mezi analogovým a digitálním výstupem. Pracuje v rozsahu výstupních hodnot od 0 - 255. Využití PWM pinů je poměrně široké, můžeme s ním řídit LED diody, serva či motory. Na následujícím obrázku můžeme vidět jak takové PWM výstupy fungují pro jednotlivé hodnoty (Fyzikální kabinet FyzKAB, 2024, str. 28 - 29).



Obr. 3: PWM

(Zdroj: http://mblock.fyzika.net/img/Arduino/PWM.png)

Deska Arduino Nano obvykle běží na mikrokontroleru ATMega328P. Celá deska může být napájena několika způsoby. Buď přímo pomocí Mini-B USB či USB-C kabelu, který se využívá také při samotném programování či je možné zapojení kabelů k pinům s označením Vin (v případě, že využíváme regulované externí napětí, můžeme použít také pin s označením 5V) a GND, které dále zapojíme do externích napájecích zdrojů, jako jsou například powerbanky. Jedinou nevýhodou může být absence dnes již klasického USB, budeme tedy pro programování potřebovat méně obvyklý kabel Mini USB B. Velmi často je oficiální Arduino Nano nahrazováno jeho klonem, jehož cena je třetinová. U klonů je však potřeba nainstalovat ovladače pro převodník USB CH340 (Arduino.cc, 2024 b; HWKITCHEN, 2024).

Obr. 4: Arduino NANO klon

(Zdroj: https://dratek.cz/photos/produkty/d/1/1069.jpg?m=1640679981)

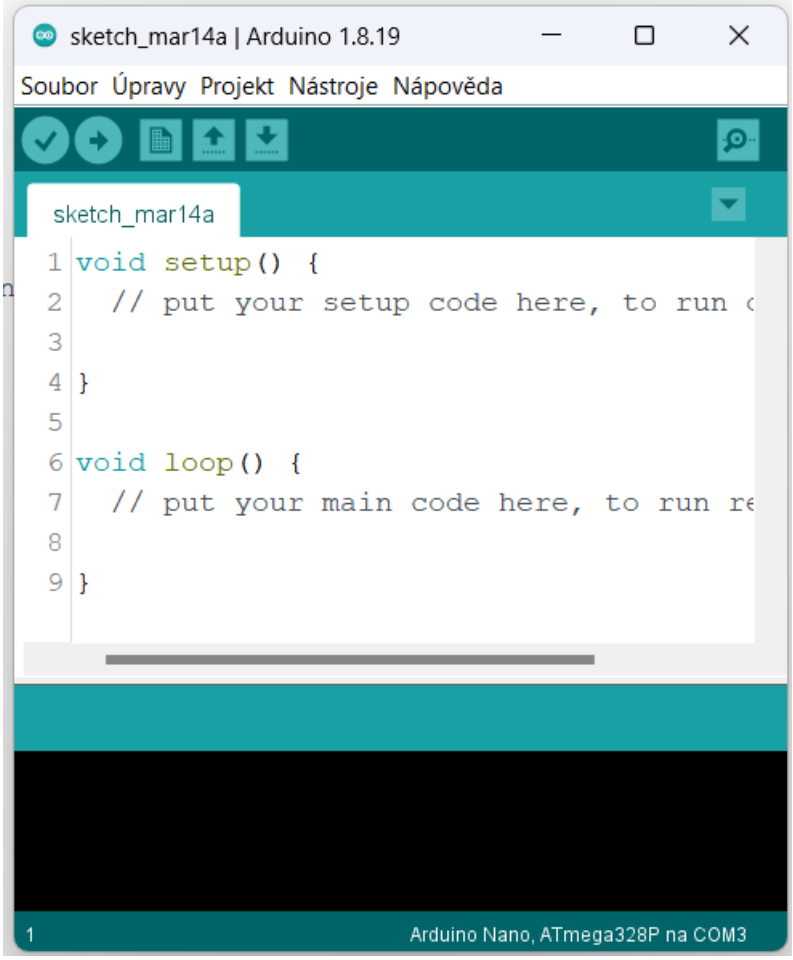
3.1.2 Arduino Software (IDE)

Software Arduino IDE je vývojové prostředí určené pro programování desek Arduino. Zkratka IDE pochází z anglického slovního spojení Integrated Development Environment, česky řečeno integrované vývojové prostředí. Arduino IDE je dostupné zdarma na oficiálních stránkách Arduina, existuje pro různé operační systémy, včetně Windows, macOS a Linux, díky čemuž je velmi populární nástroj pro programování různých projektů. Jedná se o software, který umožnuje uživatelům vytvářet, upravovat, kompilovat a nahrávat kódy do desek Arduino. Software Arduino nabízí velmi intuitivní a uživatelsky přívětivé rozhraní. Arduino IDE může fungovat tedy jako textový editor, kde uživatelé mohou psát kódy v jazyce C++. Další funkcí je kompilátor, který je možné vysvětlit jako překladač zdrojového kódu do kódu strojového, který poté nahrajeme pomocí USB portu do desky Arduino. Součástí tohoto softwaru jsou také knihovny funkcí a příklady různých jednoduchých projektů. Další nepostradatelnou funkcí při programování je sériový monitor, který nám umožňuje nechat si zde vypsat výstupní data, což nám velmi usnadní ladění programů (Voda, 2018, str. 23-24; Banzi, 2022, str. 1, 19-23).

Po spuštění softwaru Arduino IDE se nám na obrazovce v novém projektu objeví dvě funkce a to setup() a loop(). Pojďme si je nyní vysvětlit. Funkce setup() se používá především

pro inicializaci, jelikož příkazy napsané v části setup() se provedou vždy pouze jednou na začátku. Druhou zmíněnou funkcí byla loop(), která běží neustále ve smyčce (ITnetwork.cz, 2024 b).

V levém rohu tohoto prostředí na horní listě nalezneme několik záložek s názvy – Soubor, Úpravy, Projekt, Nástroje a Nápověda. Upozorníme především na záložku Nástroje, ve které můžeme nalézt sériový monitor, který využijeme u ladění téměř všech projektů. O pár řádků níže v rozbalovací nabídce objevíme taky možnost volby druhu desky, procesoru či můžeme zvolit do jakého portu v počítači jsme desku zapojili. Pod touto lištou nalezneme tlačítka s dalšími ovládacími prvky. Například znak fajfky provede kompilaci, šipka spustí nahrávání a lupa otevře sériový monitor (ITnetwork.cz, 2024 b).



Obr. 5: Arduino IDE

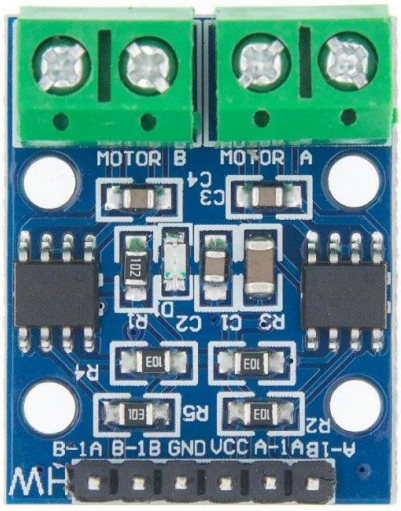
(Zdroj: vlastní obrázek)

3.2 Další používané elektrosoučástky

V první části této kapitoly jsme se věnovali velmi obsáhlému tématu, tedy Arduinu. Řekli jsme si několik informací k jeho typům desek, klonům, ale i k jeho softwaru Arduino IDE. Popsali jsme si, jak Arduino deska funguje, dozvěděli jsme se také něco o jejím čipu a pinoutu. Vše jsme si také doplnili o příběh vzniku tohoto zajímavého a nyní celosvětově rozšířeného open source projektu. Teď se pojďme podívat na několik dalších elektrosoučástek potřebných k řízení autonomního robota sledujícího čáru.

H-můstek

H-můstek modul L9110S je součástka používaná pro řízení motorů. Tento modul obsahuje dva H-můstky. Používá se k řízení dvou stejnosměrných motorů či jednoho krokového. Tato součástka se pro svou dostupnost na trhu, jednoduchému použití a nízké pořizovací ceně hojně používá v různých projektech, jako jsou například autonomní roboti či RC modely. Napájení je vhodné zvolit stejné, jako je nutné pro napájení motoru (ale pouze v rozmezí 2,5-12 V). Tento modul má následující piny: VCC: napájení modulu, GND: uzemnění, M-A: vývody motoru A, M-B: vývody motoru B, A-1A: řídicí signál pro motor A, A-1B: řídicí signál pro motor A, B-1A: řídicí signál pro motor B, B-1B: řídicí signál pro motor B. Řízení motorů si blíže ukážeme v následující pravdivostní tabulace (Tab. 1). V našem případě se bude H-můstek využívat k řízení směru otáčení a rychlosti dvou stejnosměrných motorů pomocí analogových a digitálních signálů z mikrokontroleru, tedy z Arduina (ECLIPSERA s.r.o., 2016).



Obr. 6: H-můstek L9110S

(Zdroj: https://cdn.myshoptet.com/usr/www.arduinoplc-shop.cz/user/shop/big/978-1_module-l9110-1.jpg?639c3aa1)

Tab. 1: Pravdivostní tabulka pro řízení motorů

	input A	input B	output A	output B
Jede vpřed	H	L	H	L
Jede zpět	L	H	L	H
Nejede	L	L	L	L
Aktivně brzdí	H	H	H	H

(Zdroj: vlastní tvorba)

Motory

Podle ITnetwork.cz je motor možné popsat jako elektrický točivý stroj. Funguje na principu převodu elektrické energie na energii mechanickou. V robotice se hojně využívá tzv. TT motor s převodovkou, který má plastové převody a hřídel z obou stran. Tyto motory představují levný a snadný způsob, jak rozpohybovat kola robota. Tento motor má převodovku s převodovým poměrem 1:48. TT motory lze napájet 3 až 6 V. Čím vyšší napětí, tím je větší počet otáček za minutu. Nejčastěji je však napájen z Arduina, tedy 5 V. Uvádí se, že při tomto napětí jsou výstupní otáčky 197ot/min. Za zmínku také stojí, že tyto motory se mohou otáčet v obou směrech, směrem vpřed i v vzad, což umožňuje snadnější ovládání robotů. Manipulace

s tímto motorem je poměrně snadná, jsou na něm připraveny vodivé plíšky, do jejichž otvorů lze vsunout kabely se samčími konektory. Pro stabilnější kontakty je však lepší koncovky kabelů k plíškům připájet či je alespoň elektrikářskou páskou připevnit pevně k motoru, aby se z ok na plíšku nemohly vyvléknout. V neposlední řadě lze za jeho výhodu považovat poměrně nízká hmotnost, která činí pouze 50g (ITnetwork.cz, 2024 c; LaskaKit, 2024 a).

Obr. 7: TT motor s převodovkou

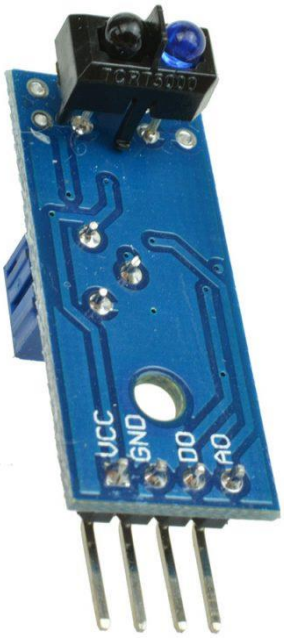
(Zdroj: https://www.kiwi-electronics.com/image/cache/catalog/product/4be8dd9c/KW-2628_3-1280x853w.jpg)

Senzory snímání čáry

Infračervený optický senzor je levný a velmi rozšířený senzor v oblasti elektroniky a robotiky. Tento senzor lze použít k detekci černé a bílé barvy, k rozpoznání pohybu či překážek. Toho se využívá v čidlech k detekci překážek či pohybu nebo v robotice k detekci černé čáry na bílém podkladu. Toto čidlo funguje na principu detekce infračerveného záření. Senzor využívá infračervenou LED diodu a fotodiodu. LED dioda vysílá infračervené záření, které se odrazí od okolních předmětů a je poté zachyceno fotodiodou. Při odražení od nového blízkého předmětu či od jiné barvy dochází ke změně intenzity odraženého světla a tím se také změní výstupní signál, který čidlo předá mikrokontroleru. Tím rozpoznáme, že jsme na bílé či černé barvě a náš line-follower, tedy jinými slovy robot sledující čáru, se může rychle vracet zpět na černou čáru. Tento senzor či podobný IR optický senzor FC-51 má uplatnění také při detekci překážek, poznáme tedy, že robotovi v cestě stojí cizí předmět (RPishop.cz, 2024; Drátek.cz).

Tento infračervený optický snímač je spolehlivý a cenově dostupný senzor, je také snadno použitelný v různých projektech díky svému jednoduchému rozhraní. Obvykle

vyžaduje jen snadné zapojení a je kompatibilní s širokou škálou mikrokontrolerů. Obsahuje pouze 4 piny: VCC, GND a signál – A0 či D0. Infračervený optický senzor TCRT5000 je tedy schopen rozlišovat mezi plochami, které odráží světlo, a plochami, které světlo pohlcují. Toto se využívá při řízení robotů sledujících čáru. Uživatel může také pomocí trimeru upravit citlivost senzoru tak, aby lépe reagoval za různých světelných podmínek (RPishop.cz, 2024; Drátek.cz).



Obr. 8: Infračervený optický senzor

(Zdroj: https://dratek.cz/photos/produkty_gal/f/49/49384.jpg?m=1646040605&_gl=1*1ox5s5g*_up*MQ..&gclid=Cj0KCQiArrCvBhCNARIsAOkAGcVCJCdty8sfB1j0nlmFs0B6A2TD1hDHLdxDDp5u82-ykahQgVqcvcAaApZ5EALw_wcB)

Ultrazvukový snímač vzdálenosti HC-SR04

Ultrazvukový snímač vzdálenosti HC-SR04 je velmi hojně používaný modul pro měření vzdálenosti pomocí ultrazvukových vln. Jeho jednou z hlavních výhod je malá velikost a nízká pořizovací cena, proto je častou volbou pro projekty s malým rozpočtem či domácí kutily. Používá se tedy u mnoha projektů s nejrůznějšími mikrokontrolery či v robotice. Snímač HC-SR04 funguje na principu vysílání ultrazvukových vln a měření času, než se tyto signály odrazí od překážky a vrátí zpět k přijímači. Na základě změřené doby pak vypočítáme vzdálenost od překážky. Zapojení tohoto modulu je velmi snadné, skládá se ze 4 pinů - VCC, GND, TRIG,

ECHO. GND je záporný pin, VCC je pin kladný, který je podle verze napájen 5 V nebo 3,3 V. Pin TRIGGER slouží jako vysílač, který vysílá ultrazvukové impulsy a poslední pin se jmenuje ECHO, jehož funkcí je přijímat impulzy, které se odrazí od překážky. Deklarovaný pracovní rozsah tohoto modulu je od 2 cm po 4 m s přesností na 3 mm. Pracovní úhel je také důležité zohlednit, ten činí 15° (HC-SR04 User Guide).

Obr. 9: Ultrazvukový snímač vzdálenosti HC-SR04
(Zdroj: https://johnny-five.io/img/components/hcsr04.jpg)

V této kapitole jsme se zaměřili na jednotlivé součástky potřebné pro sestavení jednoduchého robota s jedním infračerveným čidlem pro sledování čáry a ultrazvukovým senzorem pro objíždění překážky. Nyní už víme, jak fungují populární Arduino desky, konkrétně Arduino Nano s mikrokontrolerem ATMega328P, známe také rozložení pinů na desce i jejich využití. Umíme používat motory i senzory pro snímání čáry či detekci překážky. V následující kapitole se dozvíme, jak takového robota ze stavebnice Bitbeam4 vymodelovat a jaké 3D modelovací programy použít. Případně je možné pouze vytisknout STL soubory (viz příloha 1). Seznámíme se tedy také se samotným 3D tiskem.

4 3D modelování a 3D tisk

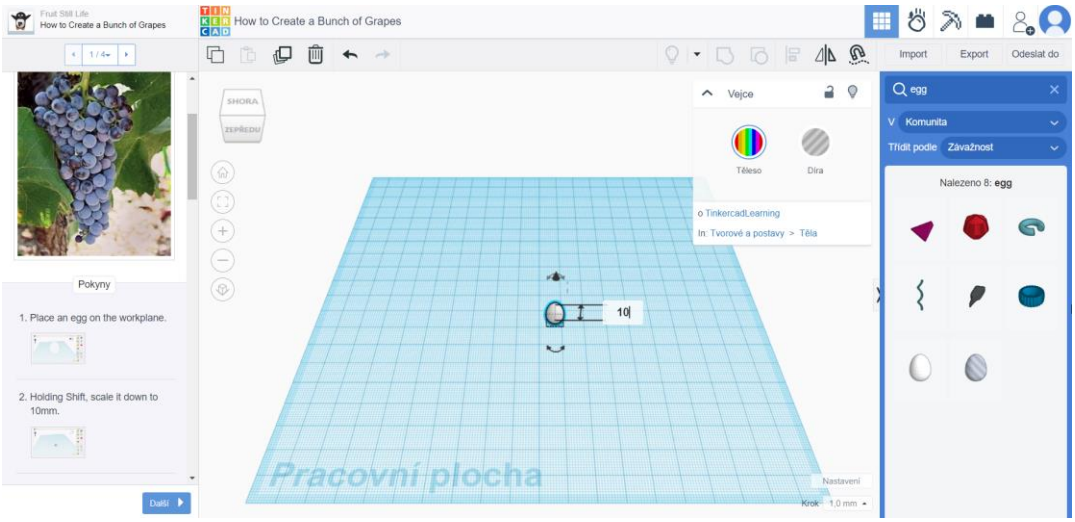
V předchozí kapitole jsme si představili elektrosoučástky, které budeme potřebovat pro jednoduchého robota, který bude schopen sledovat černou čáru na bílém povrchu a objíždět překážku. V této kapitole se dozvíme, jak pro takového robota vyrobit konstrukci. Zaměříme se na 3D modelovací programy, ve kterých si můžeme vymodelovat či upravit potřebné součástky. Řekneme si také, jak probíhá 3D tisk, něco o 3D tiskárnách ale také jaké filamenty jsou dnes k dostání.

4.1 3D modelování

Nejprve si představíme, v jakém prostředí je možné tvořit 3D modely. Seznámíme se se dvěma aplikacemi od firmy Autodesk. První z nich nese název Tinkercad a je vhodná pro úplné začátečníky, pro děti od 8 let. Dále si zmíníme již složitější software, a tím je Fusion 360, ten je vhodný již pro děti od 12 let a pokročilejší uživatele 3D modelovacích programů.

Tinkercad

Tinkercad je aplikace vyvinutá firmou Autodesk. Lze jej považovat za nejlehčí a pravděpodobně nejlépe dostupný program pro tvorbu 3D modelů. Pokud tedy hledáme přehledný, intuitivní a zábavný software vhodný pro začátečníky, pak je Tinkercad ideální varianta. Mezi velké výhody Tinkercadu se řadí také fakt, že jeho použití je úplně zdarma. Nemusíme také nic instalovat či stahovat, stačí pouze přístup k internetu a náš oblíbený internetový prohlížeč. K používání tohoto softwaru nepotřebujete vůbec žádné předchozí zkušenosti. Prostředí je velmi intuitivní. Dalším plusový bod si tento program vysloužil svými předpřipravenými projekty a výukovými lekcemi. Nalezneme zde vše od samotných počátků až po pokročilejší zábavné projekty. Na začátku nás čekají lekce, ve kterých se naučíme základní operace s tělesy, jak je umístit, jak měnit pohledy, jak těleso posunout či jak změnit jeho velikost. Po několika výukových lekcích nás dokonce čekají pokročilejší projekty, ve kterých si můžeme vymodelovat vlastní flétnu, truhličku s nápisem, různé budovy, modely zvířat i jídel (Autodesk, 2024 b).



Obr. 10: Tinkercad

(Zdroj: vlastní obrázek)

Fusion 360

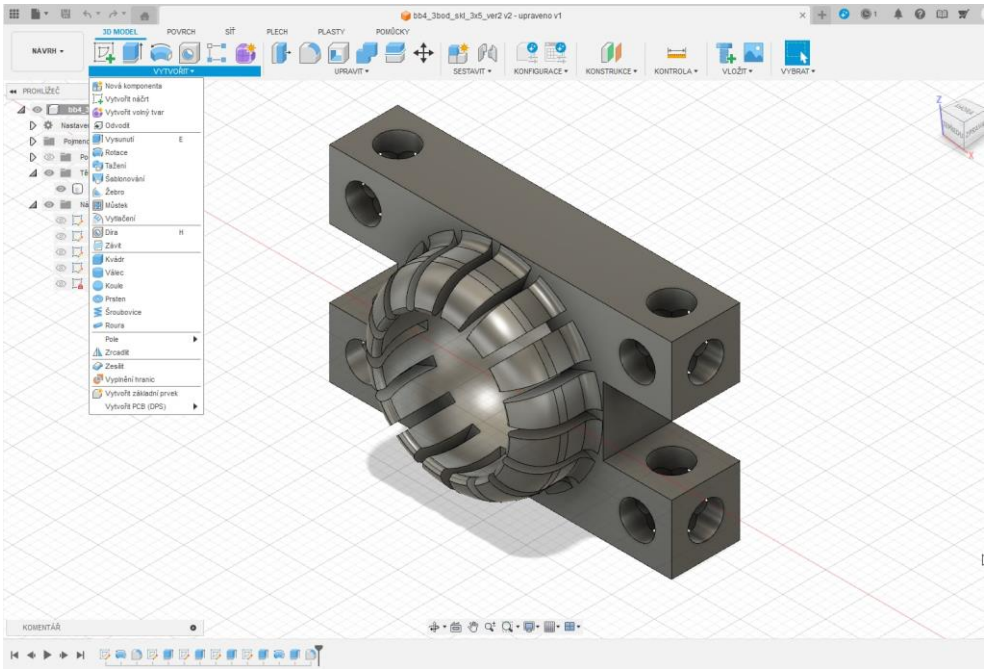
Nyní si řekneme několik informací o pokročilejším programu pro 3D modelování. Jak největší český Autodesk partner Adeon tvrdí, jedná se o revoluční cloudový software pro 3D modelování. Kromě 3D tisku nám tento 3D CAD (computer-aided design, neboli česky počítačem podporované navrhování) program umožňuje využití profesionálních prvků pro strojírenství, můžeme v něm také najít nástroje pro přípravu výroby, můžeme zde odsimulovat obrábění, ale také mnoho dalších simulačních nástrojů, jako je statické zatížení, tepelné namáhaní či simulace událostí. Odhalíme tak všechny nedostatky pohodlně ještě před spuštěním skutečné výroby. My se ale podrobněji podíváme na jeho využití pro 3D tisk (Fusion 360, 2021 a; Autodesk, 2024 c).

Fusion 360 je pro studenty, školy, ale i omezená verze pro domácí kutily zcela zdarma. Je ho možné stáhnout na oficiálních stránkách Autodesk a to hned v několika verzích. Nalezneme zde nejprve klasický placený software za 19 137 Kč za rok a variantu nesoucí název Fusion Team, ta umožňuje rozšířit licenci na další spolupracovníky a sdílet tak projekty. Níže na stránkách ale nalezneme verze, které jsou zcela zdarma, a to je omezená varianta Fusion pro osobní používání a verze Fusion pro vzdělávání, která je po ověření statutu studenta také zcela zdarma. Nyní je také možnost Fusion 360 používat pouze online, tedy bez jakéhokoliv stahování či instalace. Oficiálně je podporovaný přes webový prohlížeč Chrome. Fusion 360 tedy můžeme otevřít na jakémkoliv zařízení, na notebooku, tabletu, ale i na mobilním zařízení, což nám umožňuje pracovat neomezeně opravdu kdekoliv na dobrém internetovém připojení.

Celý tento nápad online práce v tomto 3D modelovacím programu je cílen především na studenty se studentskou licencí (Fusion 360, 2021 b; Autodesk, 2024 d).

V tomto 3D modelovacím programu můžeme vymodelovat opravdu mnoho věcí. Nalezneme zde totiž nepřeberné množství různých nástrojů. Můžeme začít pouze od vytváření náčrtů a skic, ale také můžeme provádět například parametrické modelování, volnoplošné modelování, tvořit objemová tělesa, pracovat s povrchy, upravovat importované modely, vytvářet plechové díly a jiné (Fusion 360, 2021 c; Cline, 2018, str. 1-14).

Prostředí Fusionu 360 je stále poměrně intuitivní, v horní liště máme nabídku mnoha nástrojů. V horní části můžeme přepínat mezi jednotlivými sekcemi s názvy – Solid (česky 3D model), Surface (Povrch), Mesh (Síť), Sheet metal (Plech), Plastic (Plasty) a Utilities (Pomůcky). Každá sekce dále nabízí vlastní lištu nástrojů. Každý nástroj je také doplněn o ikonu, která nám lépe pomůže pochopit, co daný nástroj umí. V levém horním rohu si také můžeme přepínat mezi jednotlivými režimy, kromě námi otevřené varianty Design (Návrh) (Obr. 11), zde nalezneme také například animaci, simulaci, vizualizaci a jiné možnosti. My si však zatím pro začátek vystačíme pouze s Návrhem a se záložkou Solid (3D model), ve které můžeme vytvářet náčrty a ty poté různými nástroji jako je vysunutí, tažení, rotace a jiné vymodelovat do 3D objektů. Za zmínku taky stojí velmi šikovný nástroj k otáčení pohledu na těleso. Vše můžeme ovládat jednoduše i bez 3D myši pomocí otáčení kostky umístěné v pravém horním rohu (Cline, 2018, str. 1-33).



Obr. 11: Fusion 360

(Zdroj: vlastní obrázek)

4.2 3D tisk

V několika následujících odstavcích se spolu podíváme na 3D tisk. Řekneme si, co to vlastně 3D tisk je, něco málo o historii, o tiskových strunách a o samotných tiskárnách.

Ondřej Stříteský ve své knize s názvem *Základy 3D tisku s Josefem Průšou* uvádí tuto definici: *„3D tisk je automatizovaný proces, při kterém se z digitální předlohy (3D modelu) vytváří fyzický model. Technologií je nyní k dispozici více – ale ta nejpoužívanější, FFF, funguje velice jednoduše. Objekt vzniká postupně, vrstvu po vrstvě, natavováním tenkého proužku plastového materiálu. Představte si, že svůj model rozkrájíte na plátky, jako bramboru na chipsy, a poté nakreslíte tavnou pistolí jeden chips za druhým, až máte celou bramboru.“* (Stříteský, 2019, str. 5)

3D tisk se původně vyvíjel pod názvem Rapid Prototyping (česky rychlá výroba prototypů) a jak už název napovídá, 3D sloužil původně především ke zrychlení a také zlevnění výroby prototypů. Díky 3D tisku odpadla nutnost vyrobit formu, testovací sérii a tak dále. Nyní si ve firmě vytisknou pouze jeden kus plánovaného výrobku na 3D tiskárně a mohou ho ihned otestovat a nemusí při zjištění každého nedostatku zadávat tvorbu nových nákladných forem a novou výrobu (Stříteský, 2019, str. 5).

Kdy se tedy 3D tisk poprvé objevil? Již v roce 1984 a za zakladatele je považován Američan Charles W. Hull, zakladatel firmy 3D Systems, který jako první vytiskl 3D objekt. V témže roce, 1984, si nechal patentovat SLA technologii, takzvanou stereolitografii. Tato metoda 3D tisku je založena na principu vytvrzování fotopolymeru pomocí UV světla. Modely vytištěné touto metodou jsou velmi detailní a nelze zde rozeznat jednotlivé vrstvy, jak je tomu u nejrozšířenější metody FFF, kterou jsme si popisovali v úvodu. Pro tuto detailnost je tato technologie využívaná v lékařství a šperkařství. Nevýhodou však je menší tisková plocha, delší tisk, ale především při tisku z pryskyřice její toxické výpary během tisku. První komerční tiskárnu založenou právě na této technologii představila již v roce 1992 firma 3D Systems (Stříteský, 2019, str. 5, 16-17).

Největší průlom v 3D tisku nastal jednoznačně v roce 2005, kdy byl Adrianem Bowyerem založen projekt RepRap. Myšlenkou tohoto projektu je vytvořit rodičovskou tiskárnu, která dokáže vytisknout mnoho součástek na stavbu dalších tiskáren. Tento projekt byl od začátku zamýšlený jako open source projekt, všechny podklady jsou tedy volně přístupné a kdokoliv je může používat, a dokonce jakkoliv upravovat. Tímto projektem se tedy 3D tisk rozšířil mezi kutily do celého světa (Stříteský, 2019, str. 5-6).

Nyní si řekneme, kde se 3D tisk využívá. Jak jsme se již dříve dozvěděli, používá se na tvorbu prototypů, dalším příkladem je třeba výroba série pouze o několika málo kusech. V dnešní době se používá také pro tvorbu na míru vyrobených předmětů. Můžeme vytisknout všelijaké věci od klíčenek se jmény, logy firem, přes různé filmové a pohádkové postavy až po tisk nedostupných náhradních dílů (Stříteský, 2019, str. 6-9).

3D tisk probíhá v několika krocích. Nejdříve ze všeho musíme vybrat 3D model, který chceme vytisknout. Tento model si můžeme buď sami vymodelovat v 3D modelovacích programech či stáhnout z internetu. V dnešní době nalezneme na internetu mnoho webových stránek, kam uživatelé nahrávají obrovské množství svých výrobků, které si můžeme zcela zdarma či za drobný poplatek stáhnout. Typický formát souboru s 3D modelem je STL. Tento soubor však tiskárna není schopná vytisknout. Musíme ho tedy v dalším kroku pomocí programu, který se nazývá slicer, převést do souboru, který je pro tiskárnu srozumitelný, do takzvaného formátu G-code. Před exportem tohoto souboru je však důležité ve sliceru nastavit parametry tisku, například zvolit typ filamentu, hustotu výplně, teplotu filamentu a podložky, výšku vrstvy, perimetry (tedy jinak řečeno počet obvodových stěn modelu) či si můžeme zapnout podpěry. Všechna tato nastavení mají vliv na kvalitu a rychlost výtisku, proto je dobré vše pečlivě zvážit. V této fázi také určujeme umístění objektu na podložce, počet kopií či požadovanou velikost objektu. Když všechny tyto parametry nastavíme, můžeme zahájit export, tedy převod STL souboru do formátu G-code. Během převodu se náš 3D model pomyslně rozkrájí na jednotlivé vrstvy a vytvoří si zde mapy linií, po kterých bude tiskárna pohybovat tryskou a tisknout. Nyní je vše připraveno a můžeme zahájit tisk, tedy poslat námi vygenerovaný G-code do tiskárny. (Stříteský, 2019, str. 24, 35-39).

Obr. 12: 3D tisk

(Zdroj: https://cdn.backend.prusa3d.com/wp-content/uploads/mini03.jpg)

Tiskové struny

Na trhu nalezneme velké množství různých druhů tiskových strun, neboli filamentů. Nejčastěji se však používají tyto 3: PLA, PETG a ASA. Materiály je vhodné vybírat vždy podle toho, co z něj zrovna plánujeme tisknout a jaké bude mít výrobek využití. Pravděpodobně nejpoužívanější filament je PLA. Z tohoto materiálu se velmi snadno tiskne. Objekty vytištěné z tohoto materiálu mají velmi krásný povrch a je tedy vhodný na detailní a malé výtisky. Další výhodou je, že při tisku příliš nezapáchá a má malou teplotní roztažnost. Výtisky z PLA se tedy příliš často od podložky neodlepují a nepraskají. Cenově se řadí mezi nejlevnější varianty. Ale vše, co má své výhody, má i nevýhody, stejně tomu je i u PLA. Tento materiál je tvrdý, ale křehký. Jedná se tady o teplotně méně odolný filament. Začíná měknout již při teplotě 60°, a proto není vhodný pro tisk součástek, které jsou vystavovány vyšším teplotám či slunečnímu záření. Podtácek pod hrneček na čaj či držák na pití do auta si tedy vytiskneme raději z teplotně odolnějších strun. Nyní si řekneme něco o materiálu PETG a ASA/ABS. Tyto materiály jsou v porovnání s PLA daleko pružnější, při drobném ohýbaní tedy nepraskají, což lze považovat za jejich velkou výhodu. Tisk je však o něco náročnější, jelikož tyto materiály se často kroutí a odlepují od podložky kvůli velké teplotní roztažnosti. Tyto materiály mají ale i další drobné nevýhody, a sice že při tisku z ABS a ASA se uvolňuje silný zápach a při tisku z PETG zase při přejezdech extruderu vznikají často nechtěné vlasce, říkáme tomu, že materiál takzvaně stringuje, ale po odstranění těchto zbytků je povrch objektu z filamentu PETG krásně lesklý. ABS je považován za vůbec první dostupný filament, ale v dnešní době se více používá jeho nástupce, již zmíněný ASA materiál. Největší výhodou těchto dvou materiálů je, že reagují s acetonem, při styku s ním začínají měknout, což nám umožňuje lehce vyhladit povrch, odstraníme tak viditelné přechody vrstev, můžeme toho však využít i při lepení jednotlivých dílků modelu (Stříteský, 2019, str. 45-47).

3D tiskárny Original Prusa

Jednou z celosvětově největších společností, zabývající se produkcí 3D tiskáren, je aktuálně jednoznačně Prusa Research. Z této české firmy se 3D tiskárny vyvážejí do celého světa. Aktuálně je podle jejich oficiálních stránek na světě 82 380 jejich evidovaných uživatelů. Na jejich stránkách si můžeme vybrat z mnoha druhů tiskáren. Liší se především ve velikosti. Nabízejí tiskárny ve verzi MINI+, MK4, velkoformátovou verzi s označením XL, ale i SLA tiskárny. Dále si můžeme zvolit, zda si chceme zakoupit plně sestavenou tiskárnu nebo si ji

sami sestavit, v tomto případě si ji můžeme objednat dokonce ve formě stavebnice. Tato firma v roce 2020 spustila zajímavý program s názvem Průša pro školy. Za cíl si klade začlenit 3D tisk do výuky, jelikož 3D tisk považují za velmi zábavnou formu rozvoje kreativity a technického myšlení. Prusa Research tedy nabízí pro všechny školy, univerzity a další vzdělávací instituce individuální zvýhodněné cenové nabídky, poradenství, školení, ale i speciální balíčky spotřebního materiálu. Další nabízenou možností je zapůjčení 3D tiskárny do školy, po splnění podmínek, jako je vytvoření vlastního projektu s 3D tiskem, je tato 3D tiskárna škole darována. Do tohoto programu se zapojilo už 2700 škol a dalších vzdělávacích institucí (Prusa pro školy, 2024).

V této kapitole jsme si představili různé 3D modelovací programy. Tinkercad, který je vhodný pro úplné začátečníky a poté jsme na to navázali pokročilejším programem Fusion 360. Dále jsme si uvedli několik informací ohledně 3D tisku. Řekli jsme si, jak 3D tisk funguje, od 3D modelu až po hotový výtisk objektu. V neposlední řadě jsme si porovnali několik druhů tiskových strun a zmínili nejpopulárnější 3D tiskárny. A vše jsme zakončili představením projektu Prusa pro školy od firmy Prusa Research. V následující kapitole se zaměříme na konstrukci robota. Náš zcela autonomně řízený robot bude postavený z open source stavebnice Bitbeam4, kterou si kompletně vytiskneme na 3D tiskárně. Nalezneme tam také stručný návod pro sestavení robota. Pro zajímavost si uvedeme aktuální ceny všech komponent robota a vypočítáme si, kolik přibližně stojí výroba jednoho takového robota.

5 Konstrukce robota

V minulé kapitole jsme si představili 3D tisk, řekli jsme si, jak probíhá a za jakých podmínek je vhodné tisknout z jakých filamentů. Zmínili jsme si také českou firmu produkující světově proslulé 3D tiskárny a jejich program Prusa pro školy, který školám nabízí výhodně 3D tiskárny pro výuku.

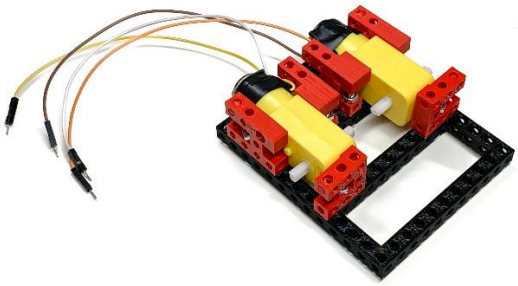
V této kapitole si popíšeme konstrukci robota. Řekneme si, kolik bude školu stát jedna stavebnice tohoto robota. Tato stavebnice bude obsahovat 3D vytištěné dílky ze stavebnice Bitbeam4, spojovací materiál a několik elektrosoučástek. Nalezneme zde také stručný návod na sestavení tohoto robota.

5.1 Kostra robota

Náš robot je celý sestavený z open source stavebnice Bitbeam4. V přílohách nalezneme kompletní souhrn STL souborů potřebných pro vytištění všech dílů robota (příloha 1). Při tisku tohoto robota, jsme spotřebovali přibližně 200 g filamentu, přepočteno na metry, přibližně 67,7 m. Robota je vhodné, z důvodu namáhaní vytištěných dílů při sestavování, ale také při jízdě a přepravě, vytisknout z materiálu PETG, který je pružnější než PLA. Při dnešních cenách za filament se cena jednoho výtisku robota bude pohybovat přibližně v rozmezí 80 až 140 Kč v závislosti na volbě firmy, která filament vyrábí. Dále jsme použili přibližně 30 šroubů a 22 matic. Za šrouby a matice jsme tedy zaplatili přibližně 50 Kč. Za použité elektrosoučástky (2 x TT motor s převodovkou, H-můstek L9110S, Infračervený senzor pro sledování čáry, Ultrazvukový měřič vzdálenosti HC-SR04, Arduino NANO R3, Propojovací kabely samec-samec, Propojovací kabely samec-samice a kabel Mini USB) jsme zaplatili přibližně 460 Kč. Celý robot nás tedy vyjde přibližně na 600 Kč (cena kalkulovaná k datu 17.3.2024) (Aurapol, 2024; Prusa Research, 2024; Laskakit, 2024 b; Spojovaci-materiál.net, 2023).

5.2 Sestavení robota

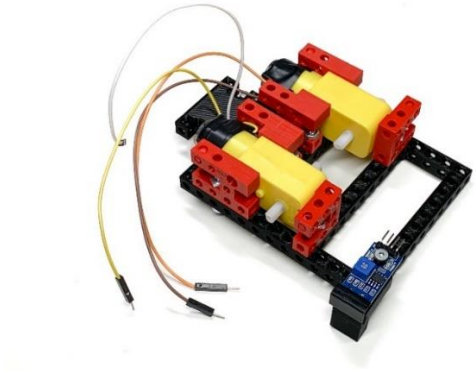
Nyní se pojďme podívat, jak takového robota sestavit. Ukážeme si nyní několik záchytných bodů během stavby robota. Podrobnější návod nalezneme v přílohách (příloha 2).



Obr. 13: Krok č. 1

(Zdroj: vlastní obrázek)

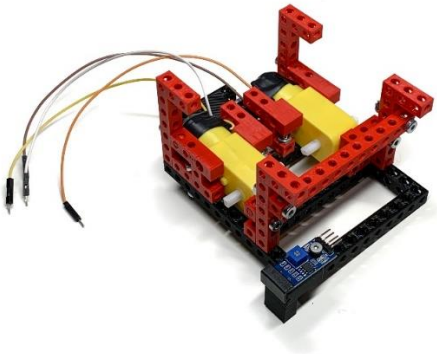
V prvním kroku si sestavíme podvozek a přišroubujeme motory doplněné o kabely pro pozdější zapojení.



Obr. 14: Krok č. 2

(Zdroj: vlastní obrázek)

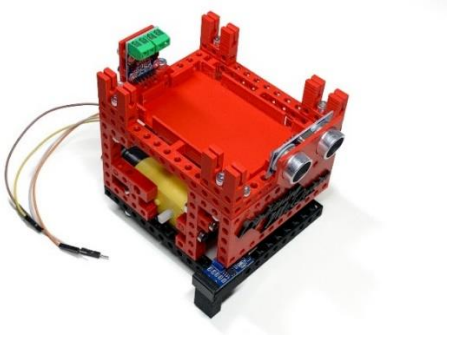
Nyní si připevníme třetí rotační bod, který jsme vyrobili pomocí skleněnky. Poté si přimontujeme do pravého rohu infračervený optický senzor pro sledování čáry.



Obr. 15: Krok č. 3

(Zdroj: vlastní obrázek)

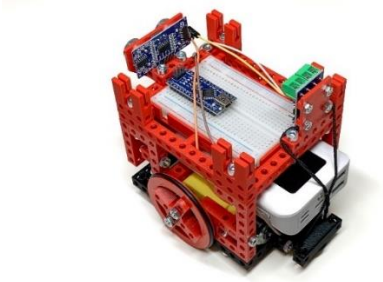
V tomto kroku si musíme vztyčit podpěrné L-dílky, abychom mohli umístit vrchní část robota. Vznikla nám tím nad motory dutina, do které později upevníme powerbanku pro napájení robota.



Obr. 16: Krok č. 4

(Zdroj: vlastní obrázek)

Nyní si připevníme vrchní část robota doplněnou o H-můstek v zadní části a v přední části o ultrazvukový senzor. Všimneme si také vytištěné SPZ v přední části robota. Každý žák si může vymodelovat a následně vytisknou SPZ se svým vlastním vzhledem či se svým jménem.



Obr. 17: Krok č. 5

(Zdroj: vlastní obrázek)

Nyní již můžeme do přihrádky ve vrchním dílu vložit nepájivé pole s Arduinem NANO. Zbývá nám také ještě přimontovat postranní kola. Do prostoru mezi první a poslední patro si vložíme powerbanku, kterou zajistíme pomocí gumičky.

Obr. 18: Krok č. 6

(Zdroj: vlastní obrázek)

Náš robot je již zcela hotový a stačí na něj nasunout střechu, kterou si můžeme jakkoliv ozdobit.

V této kapitole jsme si řekli něco o konstrukci našeho robota. Vypočítali jsme si, kolik filamentu na tisk jednoho robota budeme potřebovat a kolik nás výroba takovéhoto robota bude stát. Vysvětlili jsme si také, proč je vhodné robota vytisknout z materiálu PETG. Poté jsme se zaměřili již na jeho sestavení. Pomocí stručného návodu či úplného návodu na sestavení v přílohách jsme si ho sestavili a v následující kapitole se již podíváme, jak tohoto robota zapojit. Nalezneme zde schéma zapojení, ale také podrobný slovní popis, jak vše zapojit a na co si dát pozor.

6 Zapojení

V minulé kapitole jsme si podrobně popsali konstrukci robota. Nejprve jsme si ho vytiskli pomocí STL souborů z vhodného materiálu, tedy z pružného PETG. Poté jsme si našeho robota podle návodu také sestavili.

V této kapitole si robota zapojíme. Pro zapojení budeme potřebovat nepájivé pole, dva druhy kabelů (s koncovkami samec-samice a s koncovkami samec-samec) a několik elektrosoučástek (desku Arduino NANO, 2 TT motory s převodovkou, H-můstek, infračervený optický senzor a ultrazvukový snímač vzdálenosti HC-SR04).

Pojďme se nejprve podívat, co je vlastně nepájivé pole. Nepájivé pole bude naším základním kamenem pro stavbu robota. Nepájivé pole, jak už název napovídá, nám umožní jednotlivé součástky zapojit velmi snadno a rychle a zcela bez pájení. Výhodou je, že si můžeme obvod několikrát pozměnit či zapojit úplně od začátku, dokud nebudeme se zapojením spokojeni či dokud nebude obvod funkční (HOTAIR).

Nepájivé pole, někdy označované také anglickým názvem breadboard, lze popsat jako soustavu několika řad vzájemně propojených kontaktních bodů. Nepájivé pole se půlící rýhou opticky dělí na 2 poloviny. Na poli tedy nalezneme symetricky umístěné dva sloupce řádků, každý s 5 piny (první sloupec je nadepsaný písmeny a-e a druhý sloupec písmeny f až j), a dva postranní sloupce s vertikálně propojenými kontaktními body nesoucí označení + a -. Jednotlivé řádky v prostředních sloupcích jsou označeny čísly (1 až 30) a tyto řady jsou vzájemně odděleny. Jedna řada se skládá z 5 vzájemně propojených kontaktních bodů. Na tomto poli tedy nalezneme celkem 400 pinů, 100 postranních a 300 v centru pole (HOTAIR; Banzi, 2022, str. 207-209).

Teď pojďme začít se samotným zapojením. Nejprve si do nepájivého pole vložíme jemným tlakem v rozích programovatelnou desku Arduino NANO. Propojíme piny GND a 5V (pouze v případě, že pro napájení použijeme homologovanou powerbanku, ve všech ostatních případech připojíme napětí do Vin) s okrajovými modro-červenými lištami, tzv. napájecími kontakty, do kterých zapojíme také dva napájecí dráty vyvedené z powerbanky. Vše tedy bude napájeno přímo z powerbanky pomocí postranních napájecích kontaktů.

Náš robot bude obsahovat pouze několik elektrosoučástek. Připravíme si tedy jedno infračervené čidlo pro sledování černé čáry na bílém povrchu a jeden ultrazvukový senzor pro objíždění překážky. Vše bude poháněno pomocí dvou stejnosměrných motorů s převodovkou. K řízení motorů využijeme také H-můstek. Pojďme si tedy nyní tyto elektrosoučástky zapojit.

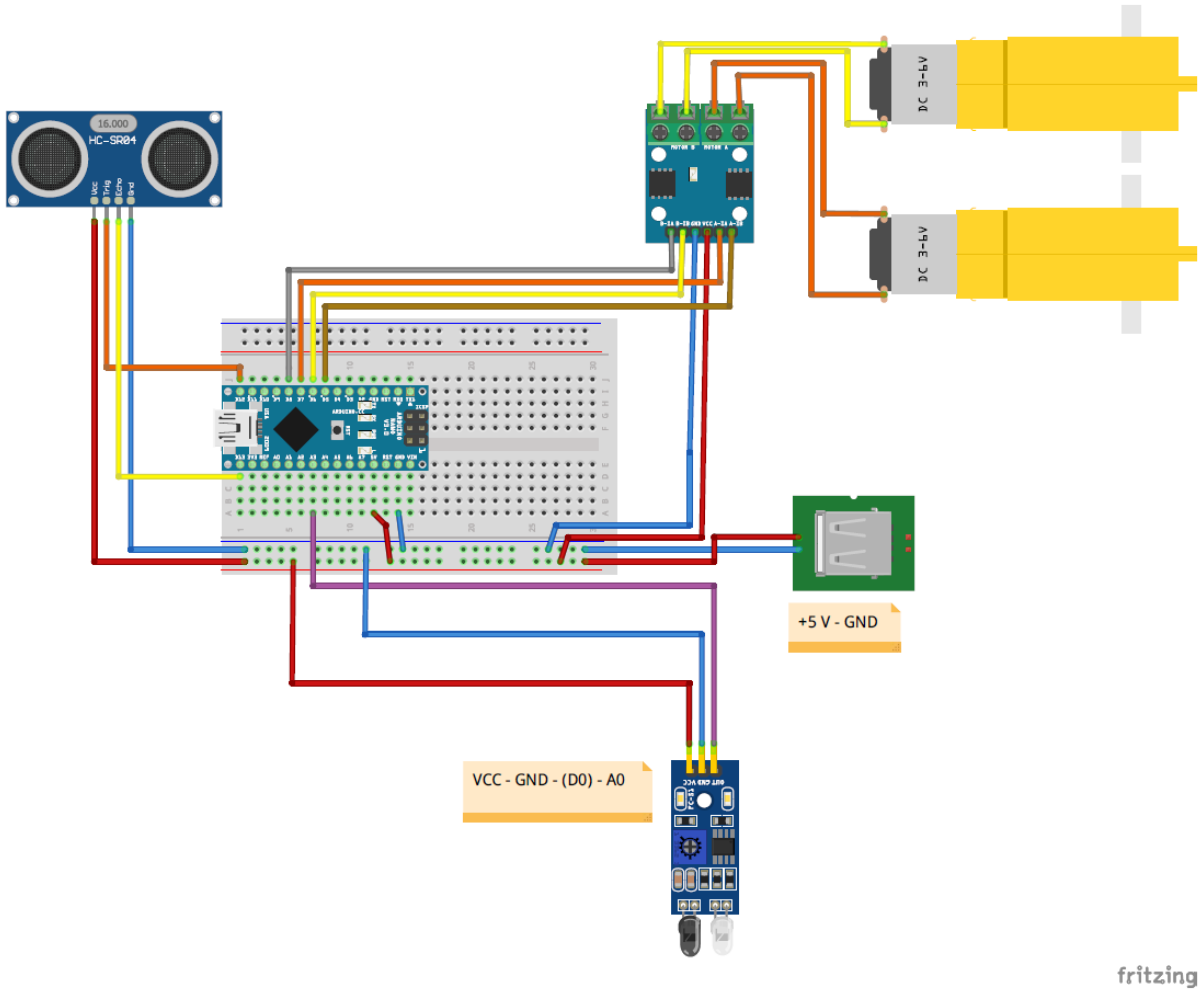
Začneme s motory. Zapojení tohoto motoru je poměrně snadné, jsou na něm připraveny vodivé plíšky, do jejichž otvorů vsuneme kabely se samčími konektory. Pro stabilnější kontakty doporučuji koncovky kabelů k plíškům raději připájet či je alespoň připevnit elektrikářskou páskou pevně k motoru, aby se z ok na plíšku koncovky nevyvlékly.

Ke každému ze dvou motorů jsme tedy připevnili 2 kabely s koncovkami samec-samec. Druhou stranu kabelů nyní připojíme k H-můstku. Nejprve ze všeho musíme na H-můstku povolit šroubky zašroubované v zelené plastové části, uvolníme tak plíšek, kterým potřebujeme později pevně přitisknout koncovky kabelů v otvoru. Do uvolněných kontaktních bodů v zeleném plastovém krytu označených nápisy Motor A a Motor B nyní vložíme koncovky kabelů. Jeden motor připojíme do části s nápisem Motor A, koncovky vložíme tedy do otvorů a připevníme je tam zašroubováním šroubků, čímž koncovku kabelu zafixujeme přitlačeným plíškem. To stejné provedeme s druhým motorem.

Nyní musíme zapojit druhou stranu H-můstku. Máme zde 6 pinů: VCC: Napájení modulu, GND: Uzemnění, A-1A: Řídicí signál pro motor A, A-1B: Řídicí signál pro motor A, B-1A: Řídicí signál pro motor B a poslední B-1B: Řídicí signál pro motor B. Pin na napájení připojíme do červené postranní lišty a pin s názvem GND do modré řady. Zbylé piny musíme přivést k Arduinu. Kabel z pinu B-1A zapojíme do pinu digitálního pinu D8; B-1B k D6, A-1A k D7 a A-1B k D5.

Teď si zapojíme čidla. Začneme infračerveným optickým senzorem pro sledování čáry. Tento snímač obsahuje pouze 4 piny: GND, signál – A0 či D0 a VCC. K zapojení si připravíme 3 kabely s koncovkami samec-samice. A zapojíme je na piny GND, VCC a V0 (D0 pin ponecháme volný). Kabel z pinu GND, tedy uzemnění zapojíme do modré postranní lišty, VCC do lišty červené a poslední pin A0 musíme přivést k Arduinu, v našem případě k pinu A3.

Dalším námi využívaným senzorem je ultrazvukový snímač vzdálenosti, kterým budeme provádět detekci překážky. Jeho zapojení je poměrně snadné. Budeme opět potřebovat pouze 4 kabely s koncovkami samec-samice. Samičí koncovky navlékneme na piny na senzoru. Konkrétně na piny s názvy VCC, GND, TRIG a ECHO. VCC zapojíme opět do napájecího kontaktu tedy do postranní červené řady označené písmenem +, GND zapojíme do řady se znakem -. Nyní přijdou na řadu zajímavější piny, a to sice ECHO a TRIG. Kabely z těchto pinů zapojíme opět k Arduinu, na digitální piny. ECHO přivedeme na pin D13 a TRIG na D12.



Obr. 19: Zapojení elektrosoučástek

(Zdroj: vlastní obrázek)

Nyní máme tedy kompletně sestavenou kostru robota, v této kapitole jsme ji doplnili i o elektrosoučástky, které jsme podle nákresu obvodu zapojili a v následující kapitole se můžeme pokusit našeho robota oživit. Nejprve si rozhýbeme motory. Poté navážeme lehkým sledováním čáry a pro pokročilejší si také vysvětlíme a vyzkoušíme naprogramovat objíždění překážky.

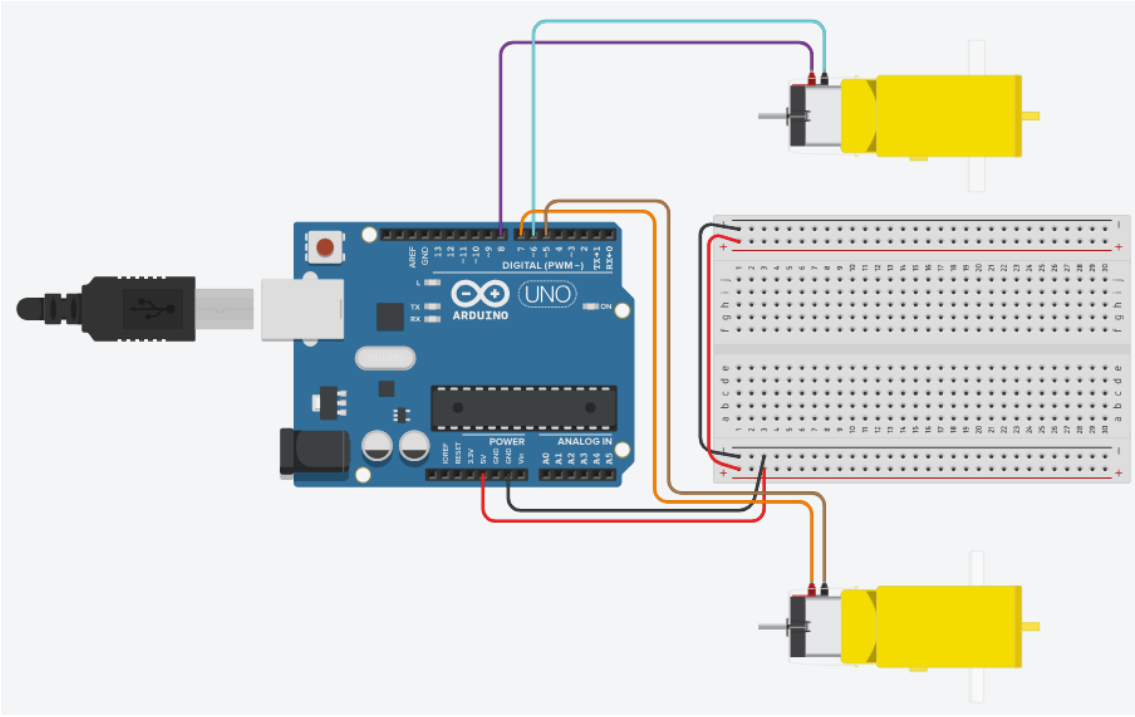
7 Architektura programu

V minulých dvou kapitolách jsme si sestavili a zapojili naše roboty. Nyní je čas je oživit. Nejprve ze všeho si vyzkoušíme ovládání motorů. Později na to také navážeme sledováním čáry a jako pokročilou rozšiřující úlohu si také ukážeme, jak můžeme robota naučit objíždět cizí předmět v dráze. Vše si ukážeme nejprve ve Scratchi a poté také v C++.

7.1 Ovládání motorů

Nejprve si vyzkoušíme napsat úplně základní kód, a sice program pro ovládání motorů. Zkusíme si napsat program, který bude sled několika jednoduchých příkazů. Nejprve si rozpohybujeme pravé kolo, poté ho zastavíme a rozpohybujeme levé a opět zastavíme. Nyní chceme, aby jely oba motory a opět po pár sekundách zastavily. Poslední příkazy budou mít za úkol roztočit obě kolečka pozpátku.

Začneme kódem ve Scratchi, tedy v programu Tinkercad. Připravíme si zde jednoduché virtuální zapojení, které nám bude simulovat našeho skutečného robota. V Tinkercadu je však toto elektronické zapojení zjednodušené, k ovládání motorů zde nepotřebujeme H-můstek, stačí připojit jednoduše motory přímo k Arduinu. Oproti tomu je v realitě nutno motory zapojit vždy přes H-můstek.



Obr. 20: Zapojení pro ovládání motorů

(Zdroj: vlastní obrázek)

Ukázkový kód ve Scratchi

Ukázkový kód v C++

```
void setup()
{
  pinMode(8, OUTPUT); //pravý motor směr
  pinMode(6, OUTPUT); //pravý motor výkon
  pinMode(7, OUTPUT); //levý motor směr
  pinMode(5, OUTPUT); //levý motor výkon
}
```

```
//bude se opakovat stále dokola ve smyčce
void loop()
{
  //točí se pravé kolo
  digitalWrite(8, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(5, HIGH);
  delay(2000); // čeká 2000 milisekund
  //stojí
  digitalWrite(8, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
  //točí se levé kolo
  digitalWrite(8, LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
  //stojí
  digitalWrite(8, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
  //točí se obě kola dopředu
  digitalWrite(8, LOW);
  digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
  digitalWrite(5, HIGH);
  delay(2000); // čeká 2000 milisekund
  //stojí
  digitalWrite(8, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
  //točí se obě kola dozadu
  digitalWrite(8, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
  //stojí
  digitalWrite(8, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(5, LOW);
  delay(2000); // čeká 2000 milisekund
}
```

7.2 Sledování čáry

Sledování čáry je poměrně snadná disciplína vhodná pro úplné začátečníky. Představíme si 2 základní metody, jak naprogramovat line-followera, tedy robota sledujícího čáru. Začneme tím nejsnadnějším způsobem a tím je dvoustavová regulace. Poté na to navážeme již pokročilejším programem a tím bude P-regulace, tedy takzvaná proporcionální regulace.

Náš robot bude černou čáru na bílém povrchu detekovat pomocí jednoho infračerveného optického snímače. Je vhodné si nejprve nechat vypisovat hodnoty z tohoto senzoru na sériový monitor, abychom zjistili, jaké výstupní hodnoty nám poskytuje robot na černé barvě a jaké na bílé, velmi nám to usnadní finální ladění programu. Bílá barva mívá z pravidla poměrně nízké hodnoty například pouhých 60, zatímco černou barvu vnímá čidlo s hodnotami až 600. Náš robot by měl sledovat okraj černé čáry. Tedy přechod mezi bílou a černou barvou. Naším úkolem tedy bude si určit nyní mezní hodnotu, kterou bychom chtěli ideálně stále sledovat. Pojďme si ji zvolit například 200. Volíme to úmyslně blíže k hodnotě bílé barvy, jelikož černá čára je velmi úzká a čidlo bude většinu času detekovat bílý povrch.

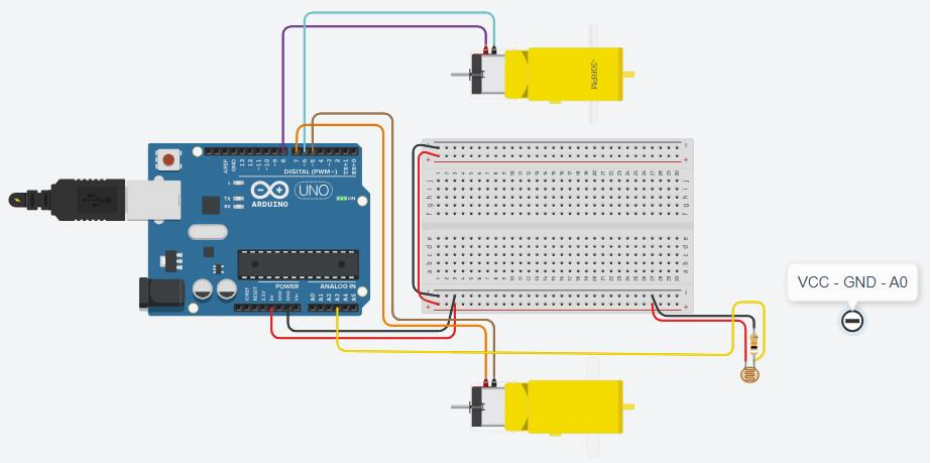
Nejprve si vysvětlíme jednodušší způsob sledování čáry. V tomto programu mohou nastat pouze dva stavy, a sice že robot vidí černou nebo bílou barvu. Mezi lidmi zajímajícími se o roboty se tento program někdy neodborně nazývá “dvoustavovka“ či kvůli kolébavým pohybům robota také “kačenka“. Naším úkolem je sledovat hranu čáry, jinými slovy, mezní hodnotu v intervalu mezi hodnotami, které čidlo vidí na bílé a černé barvě. Jelikož máme čidlo umístěné na pravé straně robota, budeme chtít sledovat pravý okraj čáry. V momentě, kdy robot najede čidlem na černou čáru, budeme chtít ihned zatočit na druhou stranu, roztočíme tedy levé kolo a robot se začne stáčet vpravo. V případě že naopak najede příliš vpravo a uvidí bílou, tak se začne točit pravé kolo, ve snaze vrátit se zpět k čáře. Robot se tedy po čáře bude pohybovat drobnými kmitavými pohyby zleva doprava.

Pokročilejší způsob sledování čáry se nazývá P-regulace, jinak řečeno, proporcionální regulace. Tato regulace spočívá v tom, že sledujeme velikost odchylky od požadované hodnoty. Robot totiž sleduje černou čáru a v momentě kdy ji ztratí, se vypočítává rozdíl pozice čidla a pozice, kde se nachází černá čára, lze to tedy vnímat jako výpočet jakési odchylky robota od čáry. Čím větší tato odchylka (tedy robotova vzdálenost) od hrany čáry je, tím rychleji by se měl robot snažit vrátit zpět k ní. V praxi to znamená, že tato chyba se započítává při výpočtu rychlosti, jakou by se měl motor točit. Pokud je tedy robot daleko od hrany čáry, rozdíl v rychlosti kol je vetší, aby se vrátil na čáru, co možná nejrychleji. Pokud je naopak blízko čáry,

pak je rozdíl rychlostí kol menší. Tímto způsobem by měl být robot schopen plynule sledovat čáru, bez zbytečného kmitání a ztrát čáry.

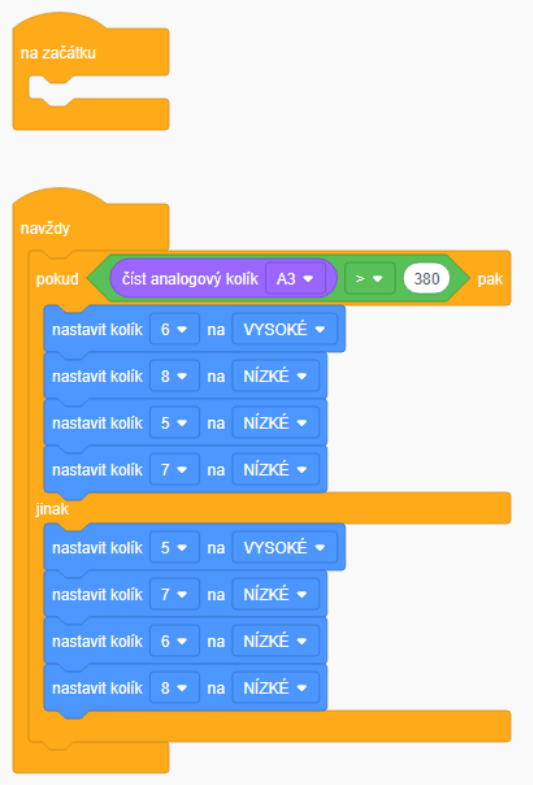
Oba tyto programy pro sledování čáry si v následujících podkapitolách představíme nejprve ve Scratchi a poté samozřejmě také v pokročilejším C++ (Arduino.cc, 2024 c).

Ukázkový kód ve Scratchi



Obr. 21: Zapojení pro sledování čáry

(Zdroj: vlastní obrázek)



Obr. 22: Scratch - Kód pro sledování čáry

(Zdroj: vlastní obrázek)

Ukázkový kód v C++

```
//deklarace proměnných
int mLv = 5; //Levý motor výkon
int mLs = 7; //Levý motor směr
int mPv = 6; //Pravý motor výkon
int mPs = 8; //Pravý motor směr

//proběhne pouze jednou na začátku
void setup()
{
  //inicializace sériové komunikace na 9600 b/s
  Serial.begin(9600);
  //nastavi piny motoru na vystupy
  pinMode(mLv, OUTPUT);
  pinMode(mLs, OUTPUT);
  pinMode(mPv, OUTPUT);
  pinMode(mPs, OUTPUT);
}

//bude se opakovat ve smyčce stále dokola
void loop()
{
    //Serial.println(analogRead(A3))
    //pro výpis hodnot z čidla - bílá 60 + černá 600 - ideálně sledovat hranici 300

    analogRead(A3); //načti hodnoty z IR čidla (zapojeno na pinu A3)
    while (analogRead(A3) < 120) //dokud je na bílé - točí pravý motor
    {
       analogWrite(mPv, 140); //pravý se točí dopředu rychlostí 140
       digitalWrite(mPs, LOW);
       analogWrite(mLv, 255); //levý stojí – aktivně brzdí
       digitalWrite(mLs, HIGH);
    }
    while ((analogRead(A3)) >= 120) //dokud je na černé - točí levý motor
    {
       analogWrite(mLv, 150); //levý se točí dopředu rychlosti 150
       digitalWrite(mLs, LOW);
       analogWrite(mPv, 255); //pravý stojí – aktivně brzdí
       digitalWrite(mPs, HIGH);
    }
}
```

7.3 Objíždění překážky

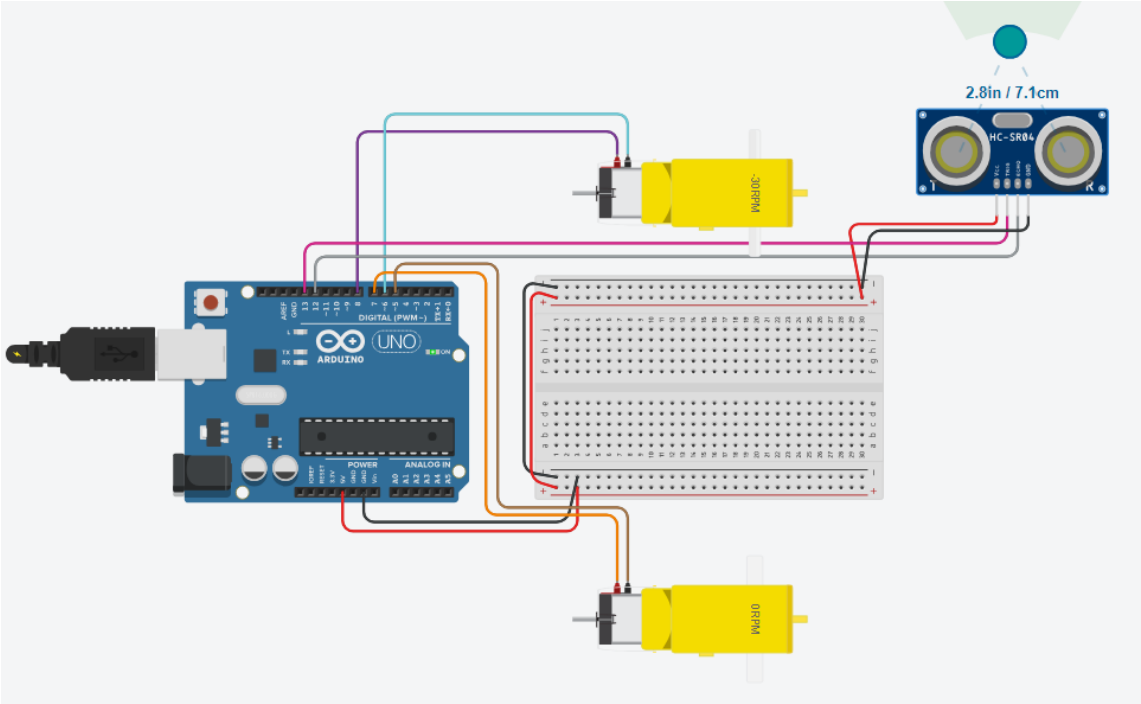
Objíždění překážky už je pokročilejší, technicky náročnější disciplína oproti samotnému line-followeru, tedy robotu, jenž sleduje pouze černou čáru na bílém povrchu. Kromě kódu pro sledování čáry potřebujeme také vyřešit chování robota v případě, že se před ním ocitne překážka. Pro detekci překážky budeme potřebovat našeho robota doplnit o další součástku, o ultrazvukový senzor vzdálenosti, jehož fungování jsme si již objasnili dříve v 3. kapitole.

Objíždění překážky se naučíme programovat poměrně jednoduchým způsobem, pomocí časování. Když se robotovi v cestě objeví překážka, dostaneme signál z ultrazvukového senzoru. Dozvíme se, že vzdálenost od okolních předmětů klesla pod námi nastavenou hranici a začneme s objížděním. Během robotických soutěžích se standartně objíždí běžně dostupná krabice od mléka či krabice o rozměrech 20x20x20 cm. My si tedy ukážeme kód pro objíždění této nejběžnější překážky, a to pro krabici od mléka.

Náš program bude obsahovat pasáž s příkazy pro robotické vozidlo, které bude v dostatečné vzdálenosti kopírovat obvod krabice a po dokončení objíždění překážky se vrátí zpět na čáru. Naším úkolem tedy bude robotovi říct, že se musí nejprve otočit vlevo, poté popojet několik centimetrů rovně a poté zatočit o 90° vpravo. Nyní je otočen zpět ve směru jízdy, avšak vzdálen od čáry, tak aby objel bez kolize krabici. V tento okamžik robotu nařídíme, aby pokračovalo rovně, čímž přejede až na pozici za překážku. V momentě, kdy mine překážku mu sdělíme, že se má opět začít vracet zpět k čáře, tedy že se má otočit vpravo a jet rovně tak dlouho, dokud nenajede zpět infračerveným čidlem na černou čáru. Poté se pootočí zpět doleva, zpátky čelem po směru jízdy a v kódu opět pokračujeme úsekem sledování čáry.

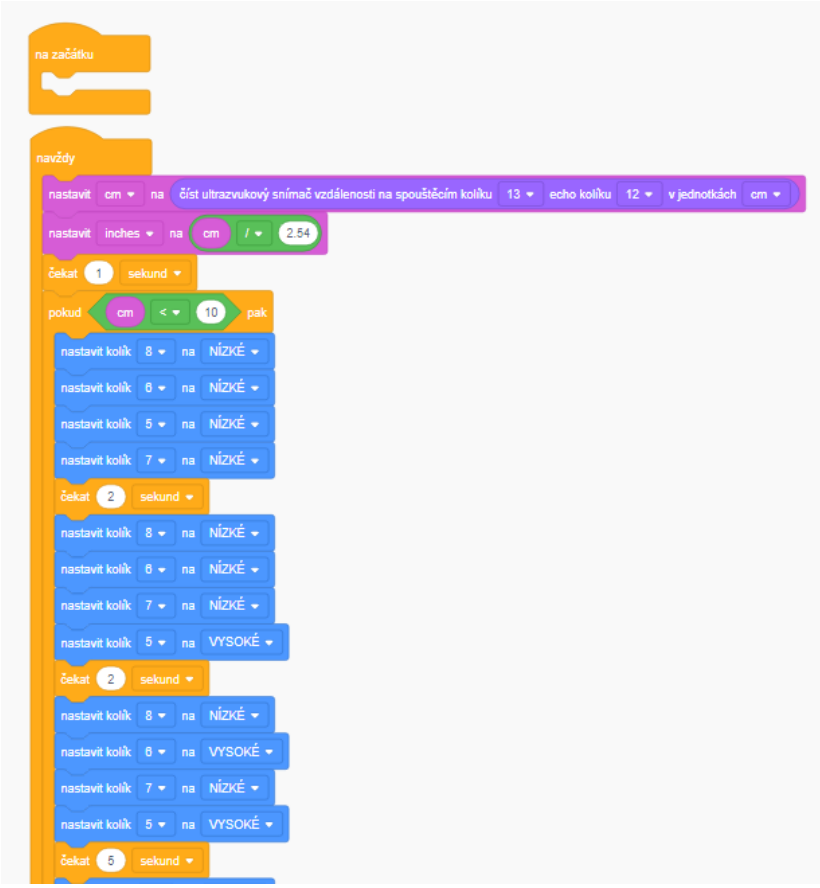
Nejprve se pokusíme naprogramovat alespoň objíždění pomocí úplného načasování, tedy bez závěrečné detekce čáry a automatického návratu na ni. Zkusíme si tedy naprogramovat časově i poslední část objíždění, tedy jet přibližně tak dlouho rovně, dokud se autonomně řízený robot nevrátí zpět na čáru, poté je nutné ho pootočit doleva a pak už zbývá jen čidlem zasáhnout čáru a začít opět s jejím sledováním. Tento styl objíždění není příliš spolehlivý, ale pro začátečníky a demonstraci objíždění překážky to je jako první krok dostačující. Jak tento úkon naprogramovat, můžeme vidět v následující ukázce z Tinkercadu, kde jsme si takovéto jednoduché objetí překážky pomocí programu ve Scratchi ukázali. V další podkapitole si ukážeme toto objetí krabice také v pokročilejším programovacím jazyce, tedy v C++.

Ukázkový kód ve Scratchi



Obr. 23: Zapojení pro objíždění překážky

(Zdroj: vlastní obrázek)



Obr. 24: Kód pro objíždění překážky

(Zdroj: vlastní obrázek)

Ukázkový kód v C++

```
//překážka - pouze časováním
int mLv = 5; //Levý motor výkon
int mLs = 7; //Levý motor směr
int mPv = 6; //Pravý motor výkon
int mPs = 8; //Pravý motor směr

int vL = 150;
int vR = 150;
int vL1 = 130; // pomalejší rychlost pro objíždění
int vR1 = 120;

int pTrig = 12;
int pEcho = 13;
int vzdalenost;

int odezva;
long ms = millis();
```

```
int vzdalenostSenzor() //vlastní funkce pro měření vzdálenosti -poté ji jen zavoláme
{
 digitalWrite(pTrig, LOW);
 delayMicroseconds(2);
 digitalWrite(pTrig, HIGH);
 delayMicroseconds(5);
 digitalWrite(pTrig, LOW);
 odezva = pulseIn(pEcho, HIGH, 8000);  // maximální délku pulzu v mikrosekundách (us)
 vzdalenost = int(odezva / 58.31);    // přepočet na cm
}

void setup() //provede se jen jednou na začátku
{
 Serial.begin(9600);

 pinMode(mLv, OUTPUT);
 pinMode(mLs, OUTPUT);
 pinMode(mPv, OUTPUT);
 pinMode(mPs, OUTPUT);


 pinMode(pTrig, OUTPUT);
 pinMode(pEcho, INPUT);
}
void loop()
{
   vzdalenostSenzor();

 if ((vzdalenost == 0) or (vzdalenost > 20))

 {

   //rovně
  digitalWrite(mPs, LOW);
  analogWrite(mPv, vR);
  digitalWrite(mLs, LOW);
  analogWrite(mLv, vL);
 }
 else
 {
  //zahájíme objíždění překážky - nyní pouze načasování do tvaru čtverce
  //doleva – točení na místě – pravé se točí vpřed a levé vzad
  digitalWrite(mPs, LOW);
  analogWrite(mPv, vR1);
  digitalWrite(mLs, HIGH);
  analogWrite(mLv, 255 - vL1);
  delay(270);
  //stop
  digitalWrite(mPs, LOW);
  analogWrite(mPv, 0);
  digitalWrite(mLs, LOW);
  analogWrite(mLv, 0);
  delay(300);
```

```
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(500);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
// Otočení doprava zhruba o 45 stupňů – pravé couvá, levé se točí vpřed
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255-vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(240);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(1250);


// Otočení doprava zhruba o 45 stupňů
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255-vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(270);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(600);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
```

```
    digitalWrite(mLs, LOW);
    analogWrite(mLv, 0);
    delay(300);
    //doleva
    digitalWrite(mPs, LOW);
    analogWrite(mPv, vR1);
    digitalWrite(mLs, HIGH);
    analogWrite(mLv, 255 - vL1);
    delay(270);
  }
}
```

V této kapitole jsme si ukázali základní principy řízení robota s jedním infračerveným čidlem pro sledování odchylky od čáry a s jedním ultrazvukovým senzorem pro detekci překážky. Nejprve jsme se naučili řídit motory a poté jsme robota naučili nejjednodušším způsobem sledovat čáru a objíždět překážku. Pokročilejší verze kódu nalezneme v přílohách, jak v jednodušším Scratchi, tak v pokročilejším C++.

Závěr

Cílem této bakalářské práce bylo seznámit učitele s alternativní možností výuky robotiky na základních školách. Smyslem této práce bylo představit levnější a snadno dostupnou stavebnici Bitbeam4, ze které si žáci mohou velmi snadno, rychle a také levně postavit robota podle svých představ. Během průzkumu cen filamentů a elektrosoučástek jsme vypočítali, že pořizovací cena robota je přibližně 600 Kč, což je v porovnání s jinými komerčními robotickými stavebnicemi na českém trhu zlomková cena. Tato stavebnice by tedy školám v budoucnu mohla umožnit pořídit každému žákovi vlastního robota, tak aby žáci nemuseli pracovat pouze s jednou stavebnicí mnohokrát i v několika členných skupinách, jak je tomu dnes. Školám totiž pro stavbu robota z této stavebnice stačí pouze 3D tiskárna, na které si mohou vytisknout tolik robotů, kolik potřebují.

V teoretické části jsme si představili, co je vlastně open source stavebnice Bitbeam4. Poté jsme si řekli, v jakých 3D modelovacích programech je vhodné s žáky modelovat, jak funguje 3D tisk a z jakých filamentů je vhodné tisknout. Navázali jsme na to kapitolou o elektrosoučástkách. V ní jsme si popsali součástky použité pro stavbu velmi jednoduchého robota sledujícího čáru a objíždějícího překážku, vhodného pro žáky základních škol. Představili jsme si také v jakých programovacích jazycích je vhodné robota programovat. Dozvěděli jsme se, že se začátečníky je vhodné začít programovat pomocí Scratche a s pokročilejšími můžeme kódy psát v C++.

K naplnění cíle mé bakalářské práce mi pomohla nejvíce praktická část, ve které jsme si popsali, jak takového robota, vhodného pro žáky základních škol, ze stavebnice Bitbeam4, sestavit. Nejprve jsme si popsali kostru robota. V přílohách také nalezneme kompletní souhrn STL souborů potřebných dílků pro sestavení tohoto robota. Tyto soubory jsou již úplně hotové a stačí je pouze vytisknout na 3D tiskárně. V další kapitole jsme si stručně popsali, jak tohoto robota sestavit. V přílohách však nalezneme podrobný návod na sestavení právě tohoto robota. Na to jsme navázali schématem pro zapojení elektrosoučástek, ale také podrobným slovním popisem tak, aby to zvládl zapojit opravdu každý, kdo se robota z této stavebnice rozhodne vyzkoušet. V závěru bakalářské práce také nalezneme vysvětlení, jak funguje sledování čáry či objíždění překážky. Uvedli jsme si zde také několik ukázkových zdrojových kódů napsaných jak v snadnějším Scratchi, tak v pokročilejším C++. Kompletní a pokročilejší programy jsou poté uvedeny opět v přílohách.

Věřím, že tato open source stavebnice Bitbeam4 si bude postupně získávat mezi učiteli i žáky stále větší oblibu, nejen pro svou pořizovací cenu, ale také kvůli jejím širokým

možnostem uplatnění. Je z ní možné velmi rychle sestavit cokoliv, co nás napadne. Využijeme ji tedy nejen při stavbě robotů, ale můžeme z ní postavit držák na telefon, jeřáb, železniční dráhu s vláčky a mnoho dalšího. Další výhodou je stále se rozšiřující základna uživatelů této stavebnice. Jelikož se jedná o open source stavebnici, tak věřím, že v budoucnu bude stále lehčí nalézt mnoho dalších návodů, STL souborů a jiných potřebných souborů a kódů pro nejrůznější stavby z této stavebnice.

Seznam použitých zdrojů

Literatura

MARJI, Majed. *Learn to Program with Scratch: A Visual Introduction to Programming with Games, Art, Science, and Math*. San Francisco, CA: No Starch Press, 2014. ISBN 978-1-59327-543-3.

MALÝ, Martin. *HRADLA, VOLTY, JEDNOČIPY: Úvod do bastlení*. Praha: CZ.NIC, 2017. ISBN 978-80-88168-26-3

VODA, Zbyšek. *Průvodce světem Arduina*. Vydání druhé. Bučovice: Martin Stříž, 2018. ISBN 978-80-87106-93-8.

BANZI, Massimo, SHILOH, Michael. *Getting Started with Arduino*. Santa Rosa: Make Community, LLC, 2022. ISBN 978-1-680-45693-6.

CLINE, Lydia Sloan. *Fusion 360 for Makers: Design Your Own Digital Models for 3D Printing and CNC Fabrication*. San Francisco: Maker Media, Inc., 2018. ISBN: 978-1-68045-355-3.

STŘÍTESKÝ, Ondřej. *Základy 3D tisku s Josefem Průšou*. Praha: Prusa Research a.s., 2019. ISBN neuvedeno.

Elektronické zdroje

ROBOTRIP. *Velká soutěž malých robotů* [online]. Olomouc: Radim Děrda, 2023 [cit. 2023-06-30]. Dostupné z: http://robotrip.cz/

TFS M-BITBEAM. *Výukové materiály* [online]. Polička: Tomáš Feltl – TFSoft, 2023 [cit. 2023-06-30]. Dostupné z: http://www.tfsoft.cz/m-bitbeam/

BITBEAM. *The Open Source Building Toy* [online]. Bitbeam, 2013 [cit. 2023-06-30]. Dostupné z: https://bitbeam.org/

EDU.CZ. *Revize RVP* [online]. Praha: MŠMT ČR & NPI ČR, 2023 [cit. 2023-06-30]. Dostupné z: https://revize.edu.cz/

GINTEROVÁ, Monika. Do škol přichází „revoluce“ v informatice. Word už stačit nebude, žáci mají umět pracovat s daty i programovat. In: *Česká televize* [online]. Praha: Česká televize, 15. února 2021 [cit. 2023-06-30]. Dostupné z: https://ct24.ceskatelevize.cz/domaci/3269122-do-skol-prichazi-revoluce-v-informatice-word-uz-stacit-nebude-zaci-maji-umet-pracovat

Rámcový vzdělávací program pro základní vzdělávání [online]. Praha: MŠMT, 2021 [cit. 2023-06-30]. Dostupné z: https://revize.edu.cz/files/rvp-zv-2021-s-vyznacenymi-zmenami.pdf

INFORMATICKÉ MYŠLENÍ. [online]. České Budějovice: Jihočeská univerzita v Českých Budějovicích, 2018 [cit. 2023-06-30]. Dostupné z: https://www.imysleni.cz/

ARDUINO.CC. [online]. 2024 [cit. 2024-03-03]. Dostupné z: https://www.arduino.cc/en/Guide/Introduction

RANDÁKOVÁ, *Pavla. PROGRAMUJU VE SCRATCHI 1: Tutoriál pro první kroky se Scratchem* [online]. Microsoft; czechitas. [cit. 2024-03-03]. Dostupné z: https://www.czechitas.cz/download/ScratchTutorial1.pdf

SCRATCH. [online]. Nadace Scratch [cit. 2024-03-03]. Dostupné z: https://scratch.mit.edu/faq

AUTODESK. *Tinkercad* [online]. Autodesk, 2024 [cit. 2024-03-03]. Dostupné z: https://www.tinkercad.com/codeblocks

ITNETWORK.CZ. *Učíme národ IT* [online]. Itnetwork.cz, 2024 [cit. 2024-03-03]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/programovaci-jazyk/seznameni-s-tinkercad-a-arduino-ide

W3SCHOOLS. *W3Schools Online Web Tutorials* [online]. W3schools, 2024 [cit. 2024-03-03]. Dostupné z: https://www.w3schools.com/cpp/cpp_intro.asp

ARDUINO.CC [online]. 2024 [cit. 2024-03-03]. Dostupné z: https://docs.arduino.cc/hardware/nano/

HWKITCHEN [online]. HWKITCHEN, 2024 [cit. 2024-03-13]. Dostupné z: https://www.hwkitchen.cz/arduino-nano/

Programování modulu Arduino v prostředí mBlock [online]. Fyzikální kabinet FyzKAB, 2024 [cit. 2024-03-13]. Dostupné z: http://mblock.fyzika.net/download/Arduino_MAXI_Starter_kit-mBlock-BETA_6-compressed.pdf

VODA, Zbyšek. ARDUINO POD POKLIČKOU: JAK FUNGUJE MIKROKONTROLÉR. In: *BASTLÍRNA HWKITCHEN* [online]. HWKITCHEN, 21. března 2019 [cit. 2024-03-14]. Dostupné z: https://bastlirna.hwkitchen.cz/arduino-pod-poklickou-jak-funguje-mikrokontroler/

ATmega328P [online]. San Jose, CA: Atmel Corporation, 2015 [cit. 2024-03-14]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

ITNETWORK.CZ. *Učíme národ IT* [online]. Itnetwork.cz, 2024 [cit. 2024-03-14]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/programovaci-jazyk/seznameni-s-tinkercad-a-arduino-ide

HC-SR04 User Guide [online]. [cit. 2024-03-08]. Dostupné z: https://dratek.cz/docs/produkty/0/773/eses1500636000.pdf

ECLIPSERA s.r.o. *H-můstek modul L9110S* [online]. ECLIPSERA s.r.o., 2016 [cit. 2024-03-08]. Dostupné z: https://dratek.cz/docs/produkty/0/68/1458419853.pdf

ITNETWORK.CZ. *Učíme národ IT* [online]. Itnetwork.cz, 2024 [cit. 2024-03-09]. Dostupné z: https://www.itnetwork.cz/hardware-pc/arduino/hardware/arduino-druhy-motoru-a-programovani-servo-motoru

LASKAKIT. *By makers for makers* [online]. LaskaKit, 2024 [cit. 2024-03-09]. Dostupné z: https://www.laskakit.cz/tt-motor-s-prevodovkou-plastove-prevody/?gad_source=1&gclid=Cj0KCQiArrCvBhCNARIsAOkAGcXzkWeJhqAr6XT3G-9VjuYngANZfaD8uVOgnYXEEUKdpp37CFAXfVgaAv38EALw_wcB

DRÁTEK.CZ [online]. ECLIPSERA s.r.o. [cit. 2024-03-09]. Dostupné z: https://dratek.cz/arduino/1142-infracerveny-opticky-senzor.html

RBISHOP.CZ. *Váš dodavatel Raspberry Pi* [online]. RPishop.cz, 2024 [cit. 2024-03-09]. Dostupné z: https://rpishop.cz/svetlo/2893-infracerveny-opticky-senzor.html?gad_source=1&gclid=Cj0KCQiArrCvBhCNARIsAOkAGcXE8D1sHIjVLGoFUKdXiVLnghRLwkKMFkMURWJZ2gBeJj8y71dldjkaApLREALw_wcB

HOTAIR [online]. HOTAIR [cit. 2024-03-15]. Dostupné z: https://www.hotair.cz/detail/plosne-spoje/nepajive-kontaktni-pole/zkusebni-nepajive-pole-400-pinu.html

AUTODESK. *Tinkercad* [online]. Autodesk, 2024 [cit. 2024-03-15]. Dostupné z: https://www.tinkercad.com/3d-design

FUSION 360. *Autodesk* [online]. Adeon CZ s.r.o, 2021 [cit. 2024-03-16]. Dostupné z: https://www.fusion360.cz/

AUTODESK [online]. Autodesk, 2024 [cit. 2024-03-15]. Dostupné z: https://www.autodesk.cz/solutions/cad-software

FUSION 360. *Autodesk* [online]. Adeon CZ s.r.o, 2021 [cit. 2024-03-16]. Dostupné z: https://www.fusion360.cz/online/

AUTODESK. *Fusion* [online]. Autodesk, 2024 [cit. 2024-03-15]. Dostupné z: https://www.autodesk.cz/products/fusion-360/overview?term=1-YEAR&tab=subscription

FUSION 360. *Autodesk* [online]. Adeon CZ s.r.o, 2021 [cit. 2024-03-16]. Dostupné z: https://www.fusion360.cz/funkce/3d-modelovani/

ARDUINO.CC. *Project hub* [online]. 2024 [cit. 2024-03-16]. Dostupné z: https://projecthub.arduino.cc/anova9347/line-follower-robot-with-pid-controller-01813f

PRUSA PRO ŠKOLY [online]. Prusa Research, a.s., 2024 [cit. 2024-03-17]. Dostupné z: https://proskoly.prusa3d.cz/

AURAPOL. *3D filament.* Aurapol s.r.o., 2024 [cit. 2024-03-17]. Dostupné z: https://www.aurapol.com/cz/pet-g-1-kg

PRUSA RESEARCH. *By Josef Prusa* [online]. Prusa Research, a.s., 2024 [cit. 2024-03-17]. Dostupné z: https://www.prusa3d.com/cs/kategorie/prusament-petg/

LASKAKIT. *By makers for makers* [online]. LaskaKit, 2024 [cit. 2024-03-17]. Dostupné z: https://www.laskakit.cz/kosik/

SPOJOVACÍ-MATERIÁL.NET [online]. Prumex s.r.o., 2023 [cit. 2024-03-17]. Dostupné z: http://www.spojovaci-material.net/sp/srouby/valcova-a-zaoblena-hlava/valcova-imbus-din-912/ocel-8-8/pozink/sroub-valcova-hlava-inbus-din-912-m4x18-8-8-pozink-3004.html

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1 – Souhrn STL souborů potřebných pro vytištění všech dílů robota

https://drive.google.com/drive/folders/1OVLCbBWWJNRVtpd2J7aPRzvuFAGmFHjw?usp=drive_link

Příloha 2 – Návod na sestavení robota

https://drive.google.com/file/d/1r3XdUCn4rj4bzgvy2NkplPs2qsF9XZCC/view?usp=drive_link

Návod na sestavení robota

ze stavebnice BitBeam4

s 1 čidlem pro sledování čáry a ultrazvukovým senzorem pro objíždění překážky

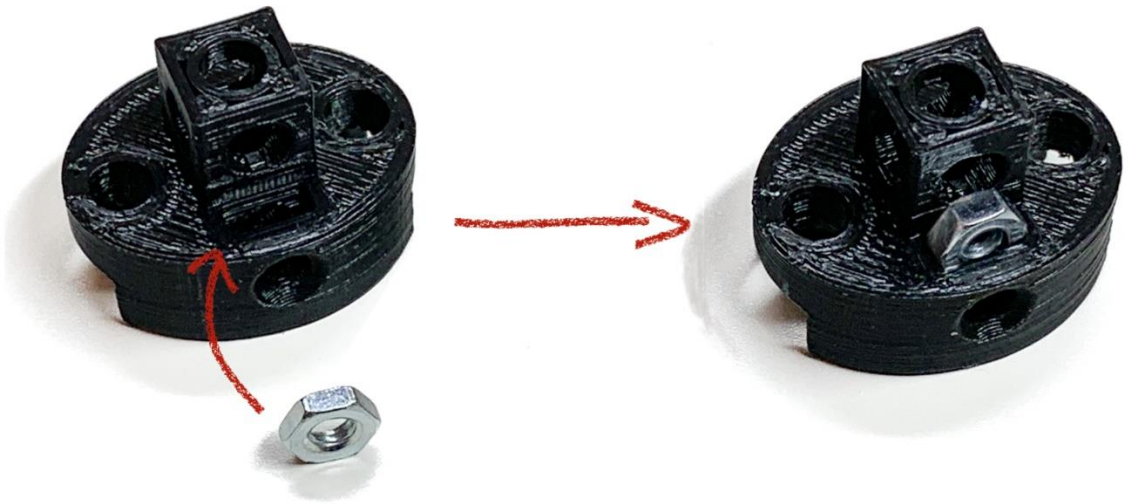
Hana Čáslavková

Potřebné nářadí:

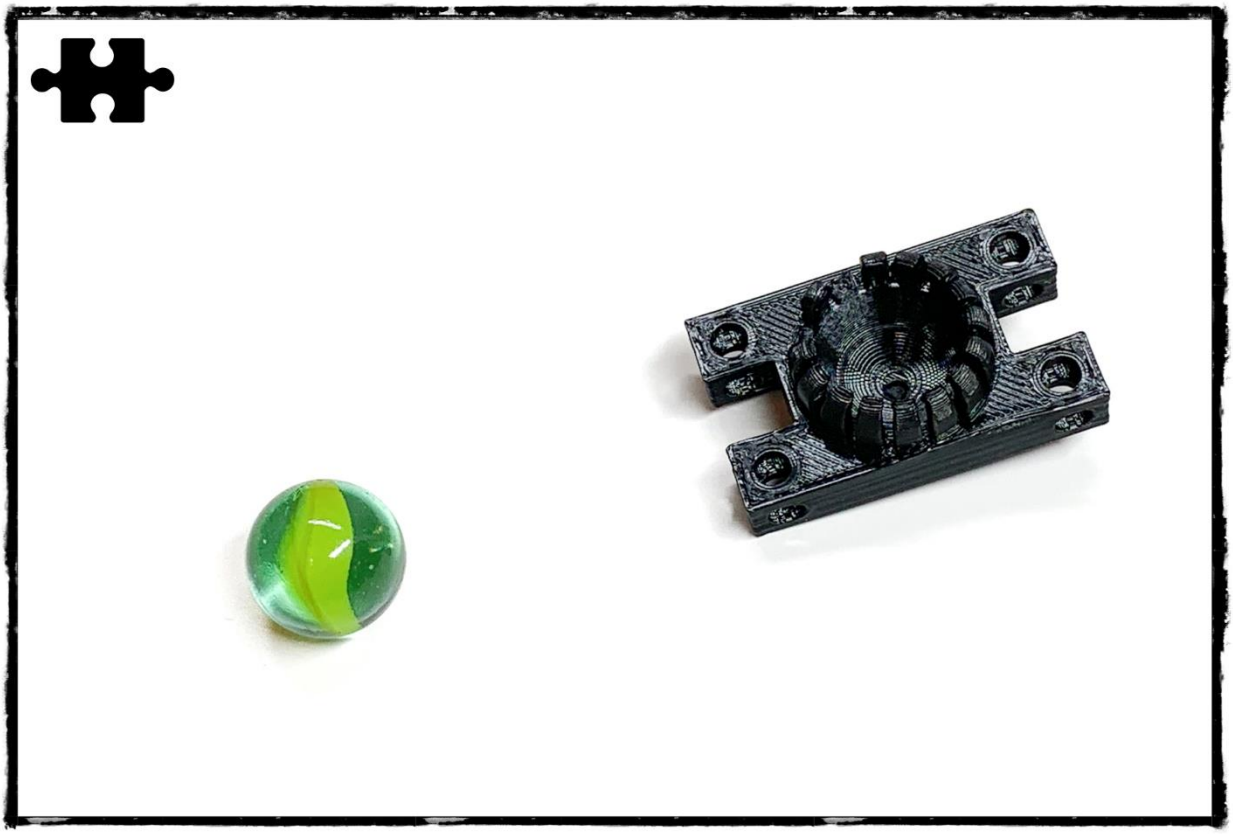
Plochý šroubovák 3 mm
Křížový šroubovák PH0
Imbus 3
Imbus 2,5
Imbus 2
Klíč 5,5
Klíč 7
Kleště špičaté

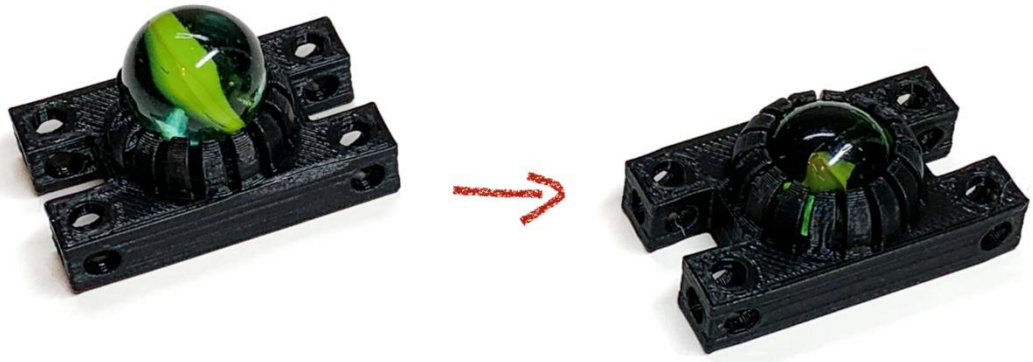
1

Šrouby: 4 ks 4x16
Matice: 4 ks M4
2 ks M3

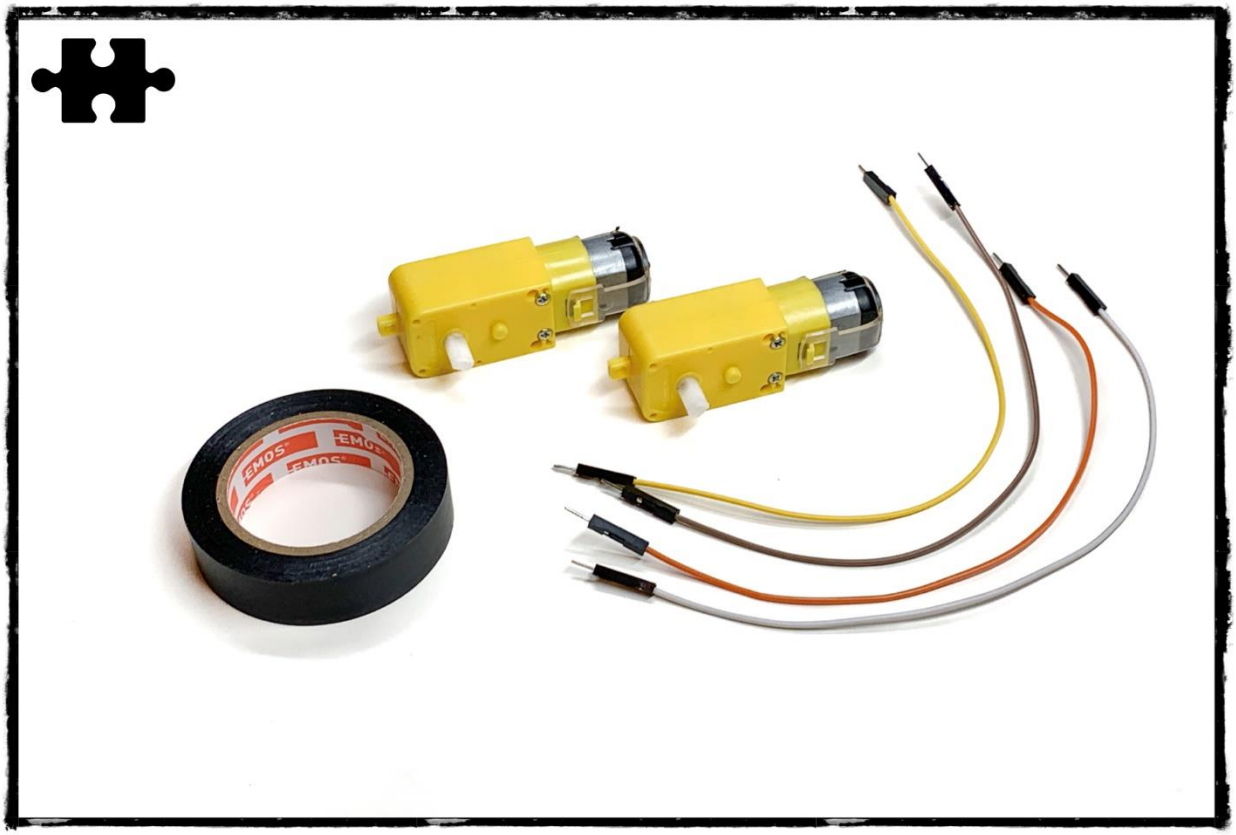


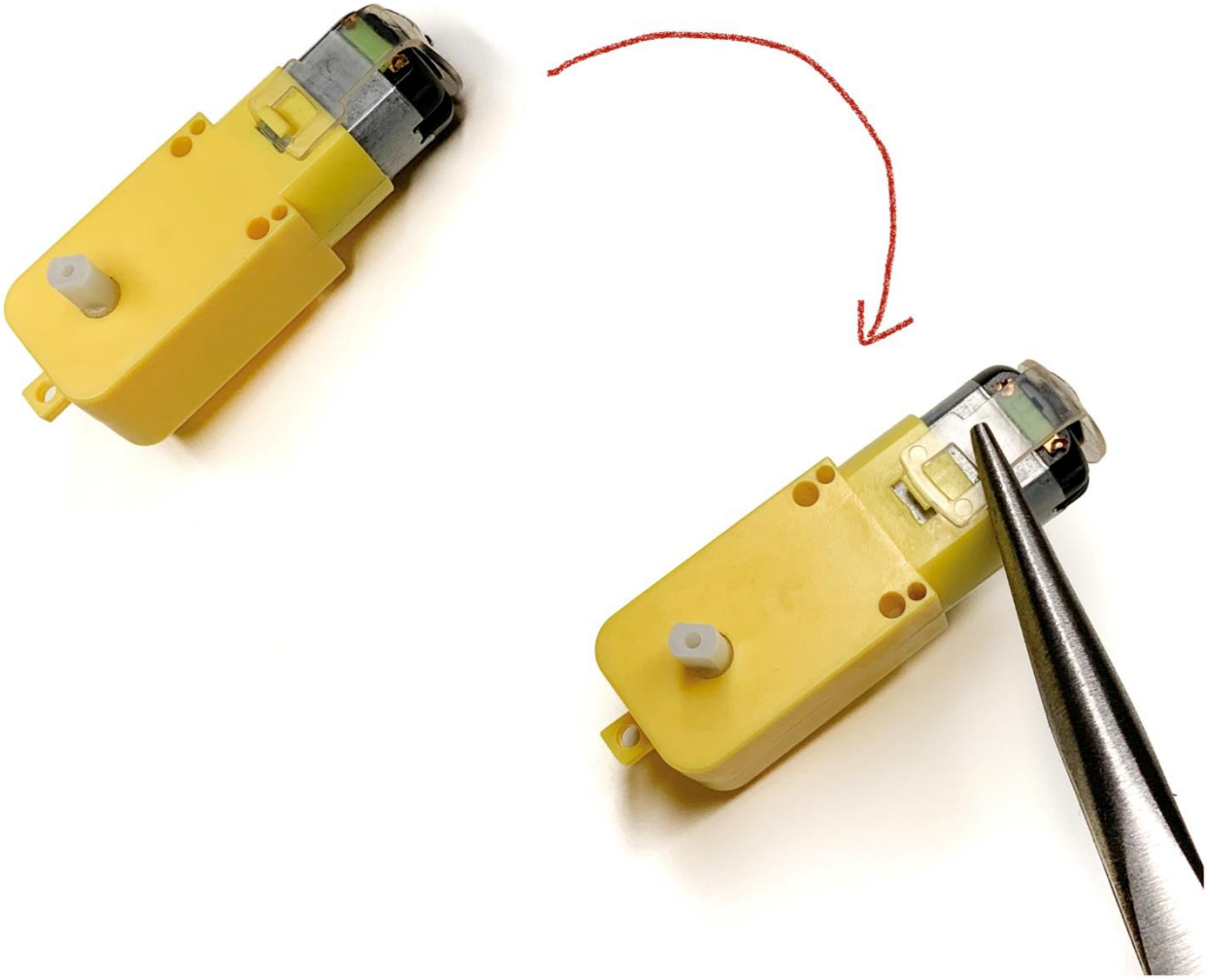
2

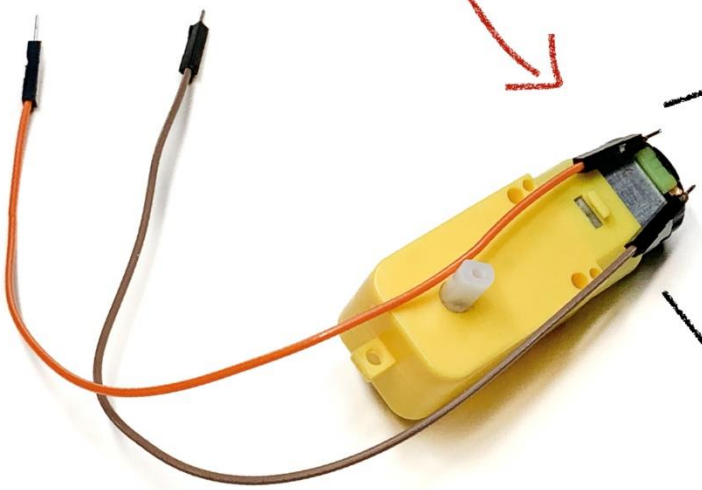


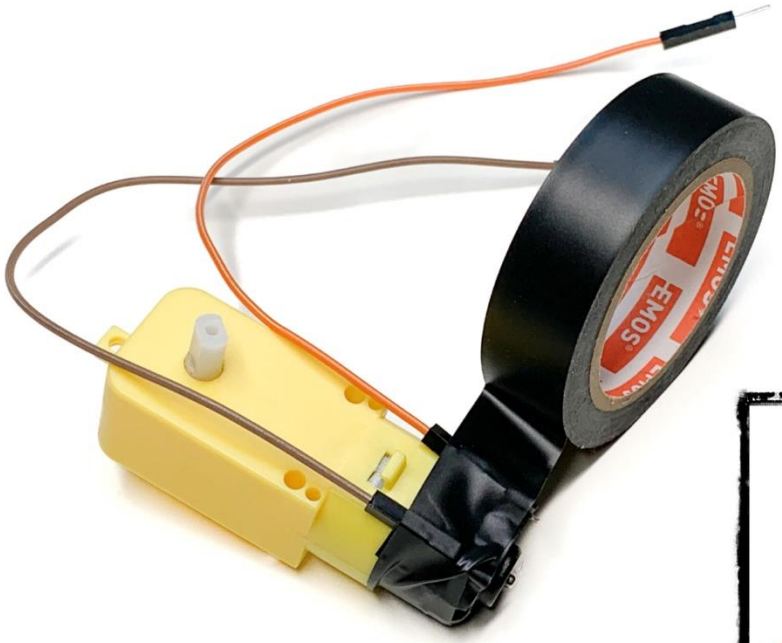


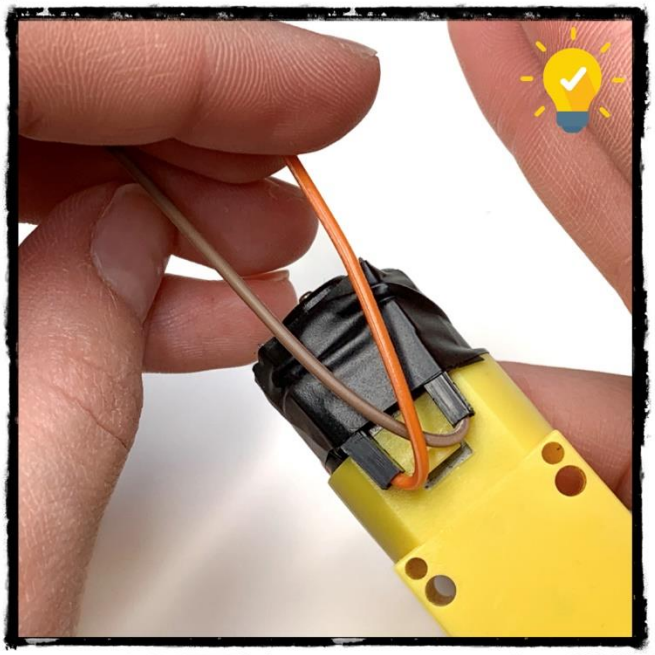
3



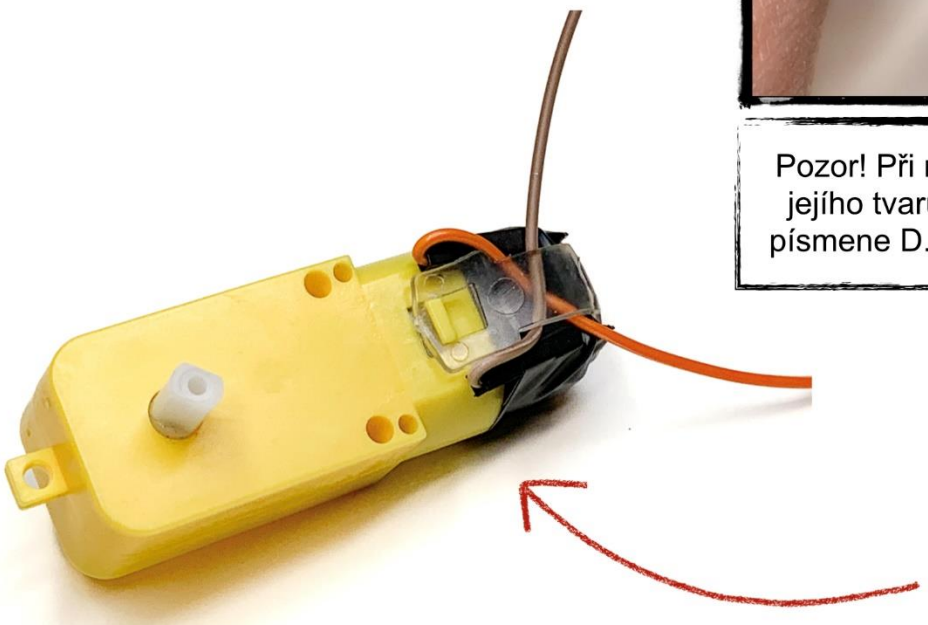






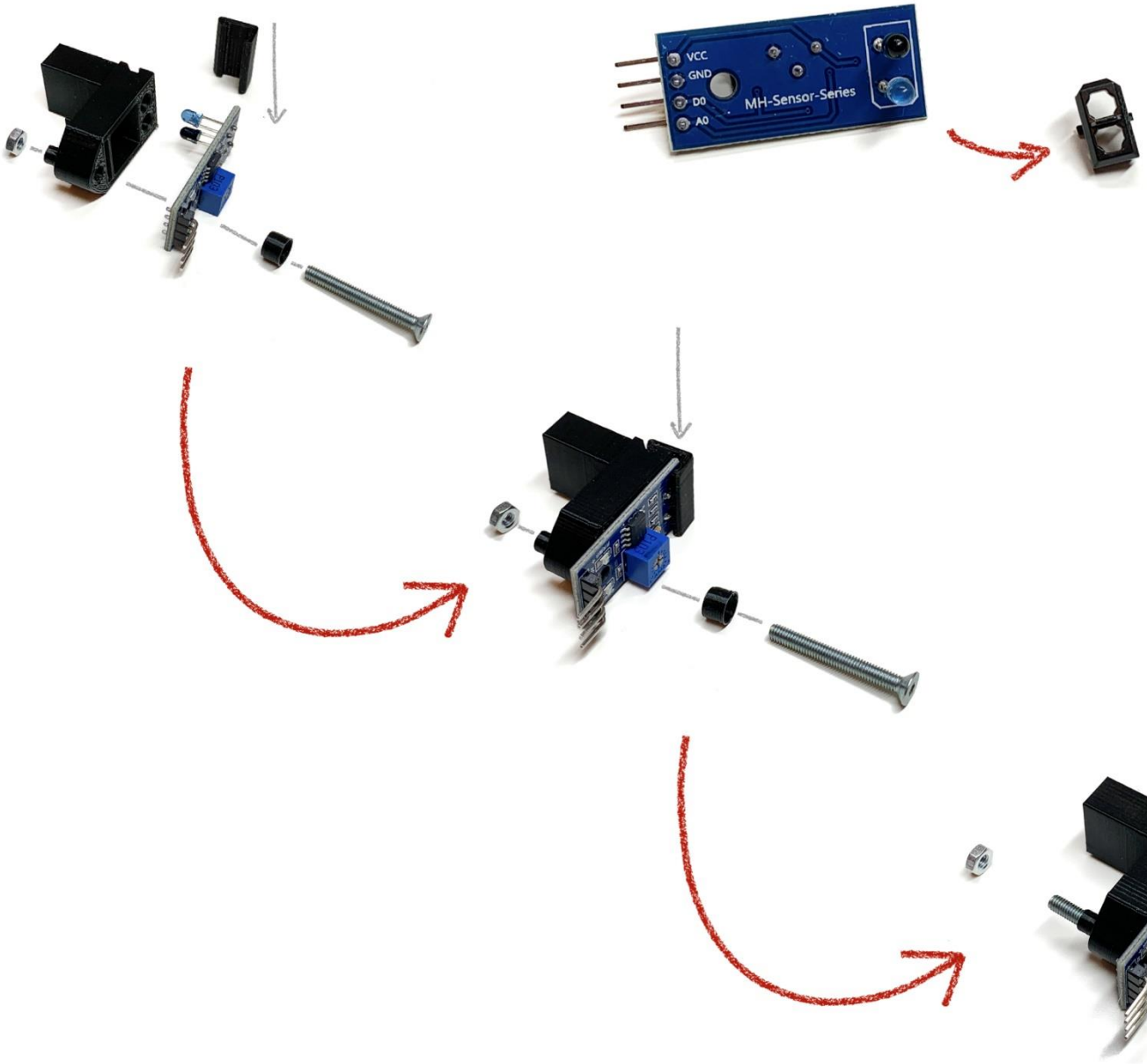


Pozor! Při nasazováni krytky si všimněte jejího tvaru. Není kruhová, ale ve tvaru písmene D. Nelze ji nasadit oboustranně!

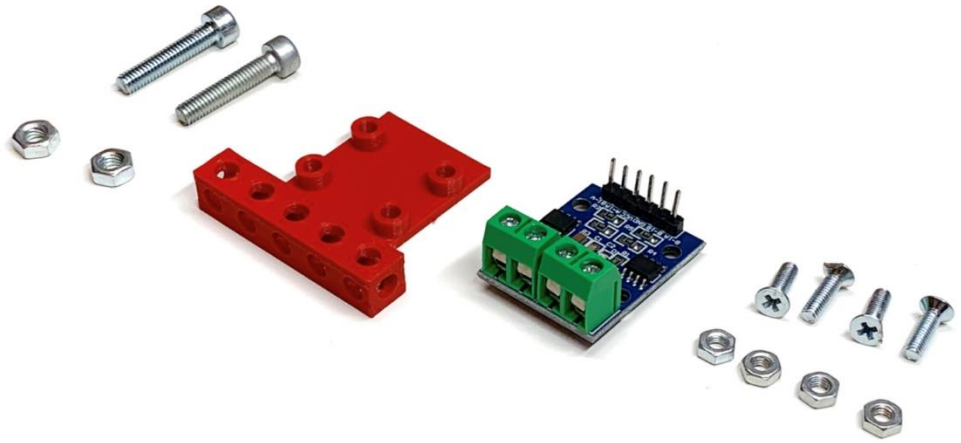


4

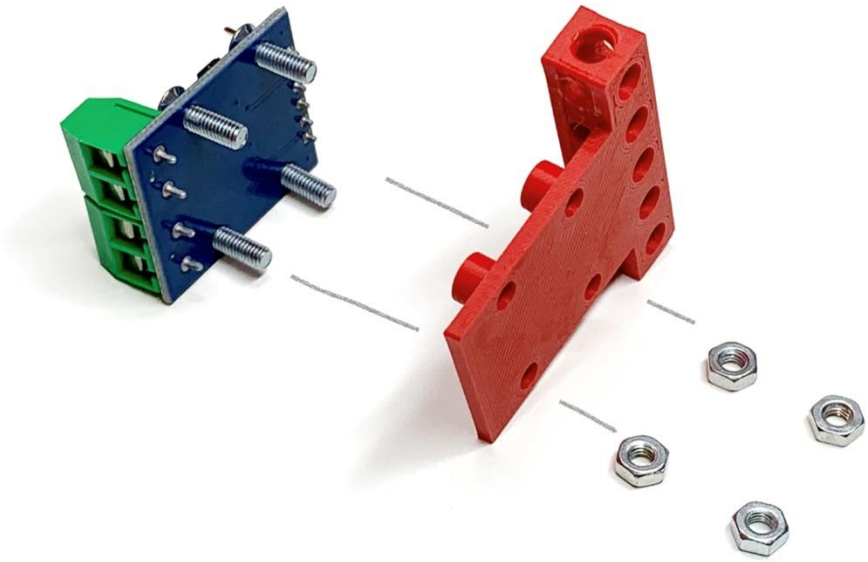
Šrouby: 1 ks 3x25
Matice: 1 ks M3

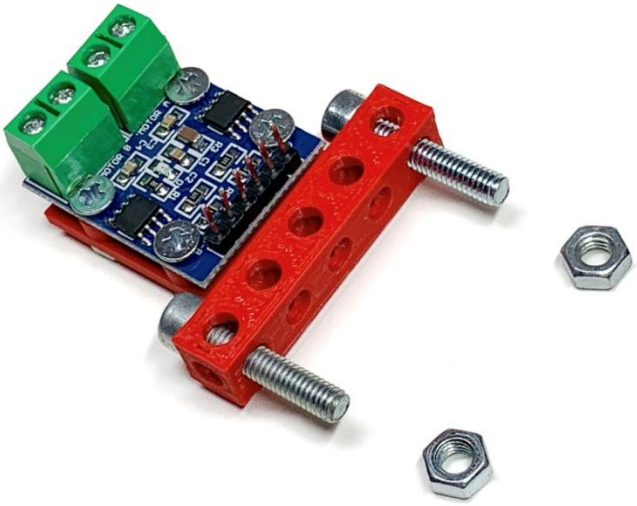


5



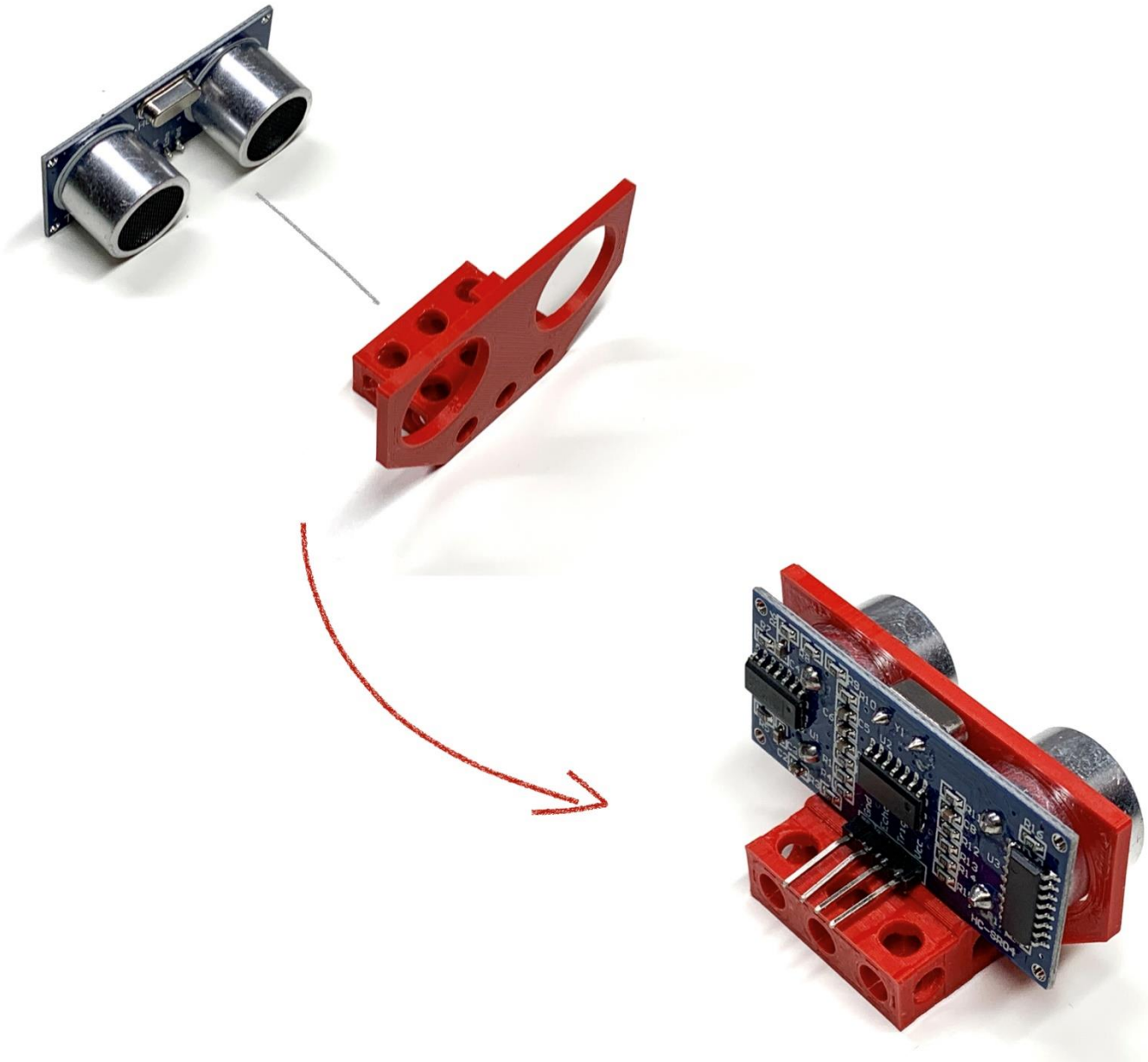
Šrouby: 4 ks 3x10
2 ks 4x20
Matice: 4 ks M3
2 ks M4



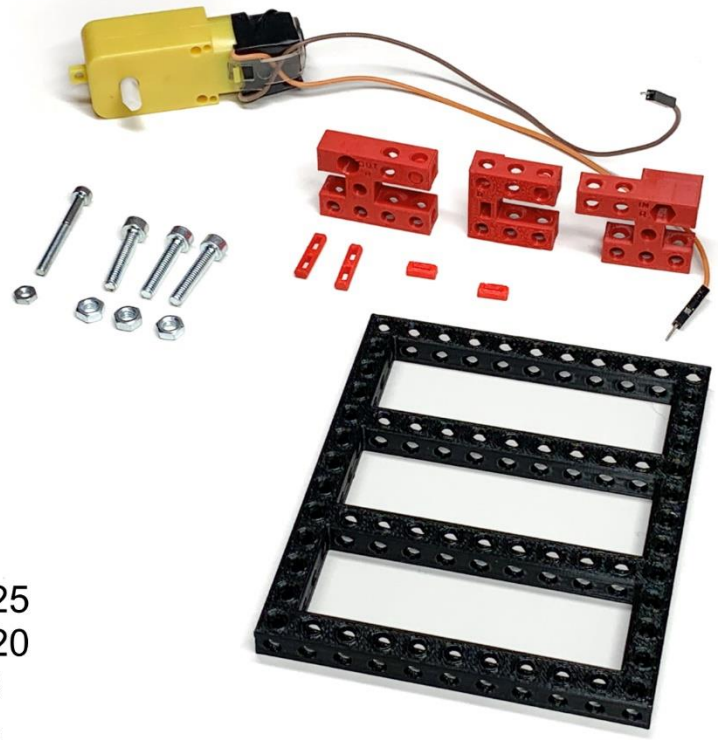


Šrouby umístěné u zelených konektorů je potřeba mírně upilovat pilníkem či uskřípnout štípacími kleštěmi.

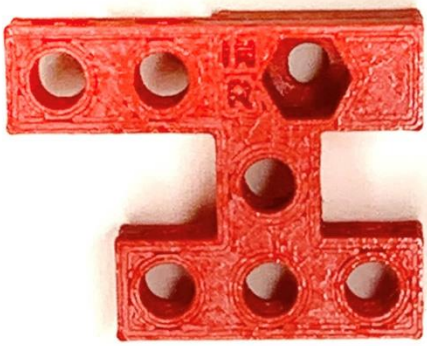
6

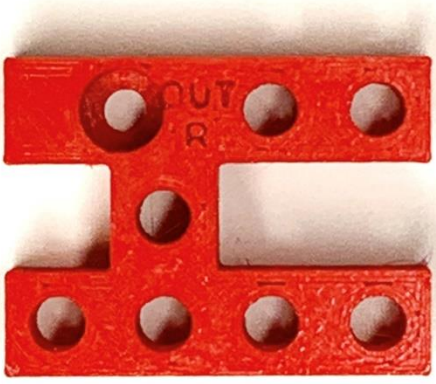


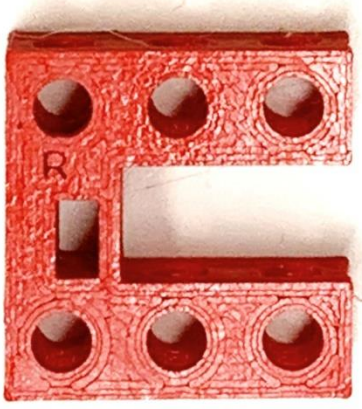
7



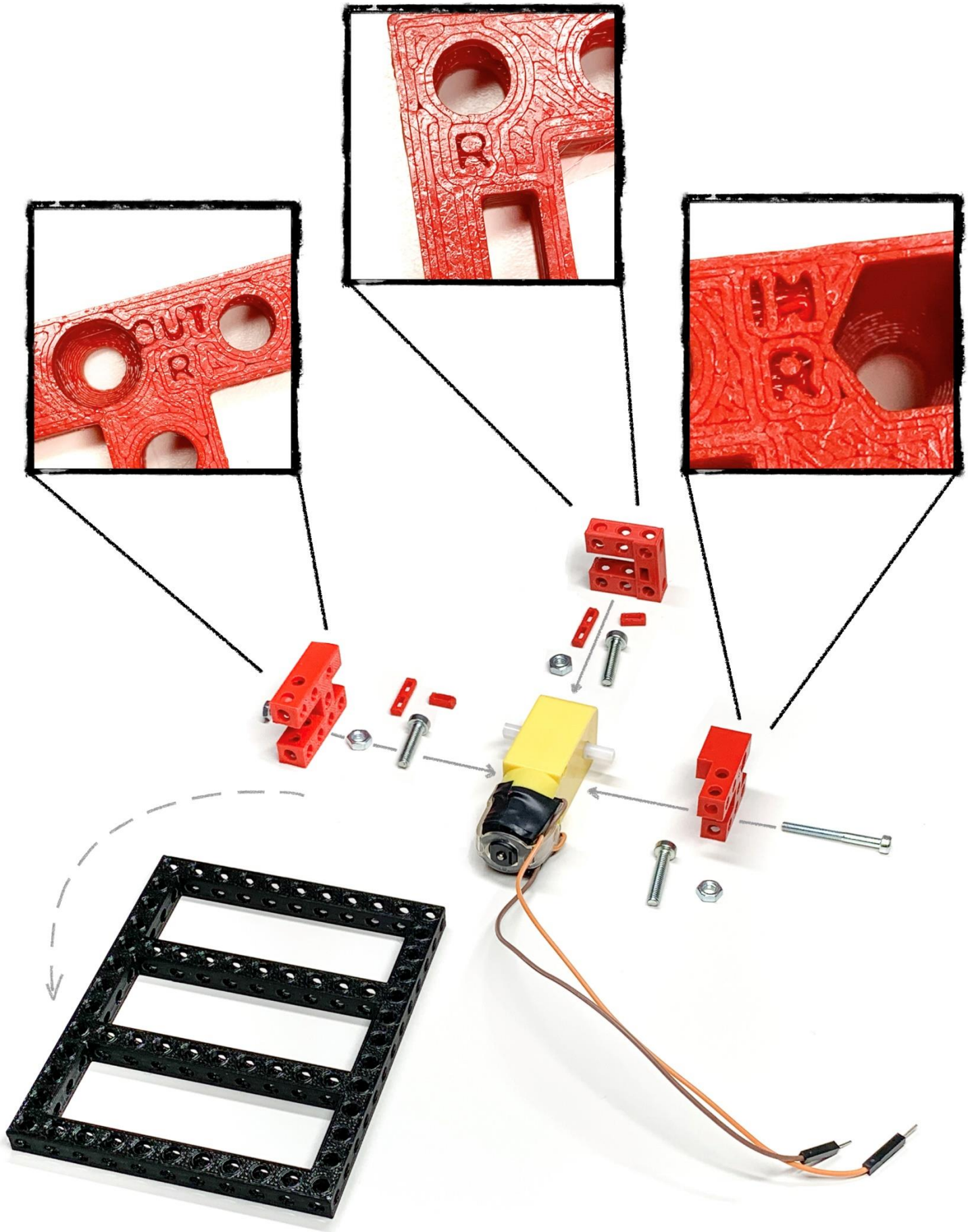
Šrouby: 1 ks 3x25
3 ks 4x20
Matice: 1 ks M3
3 ks M4



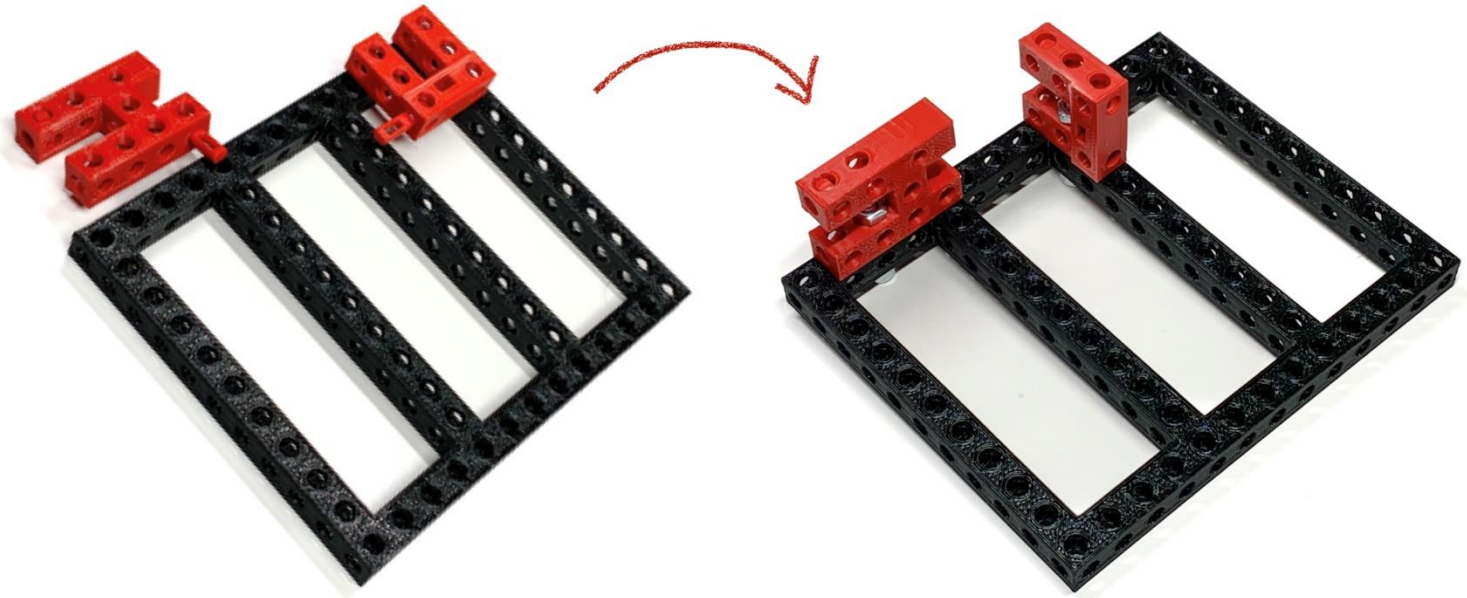


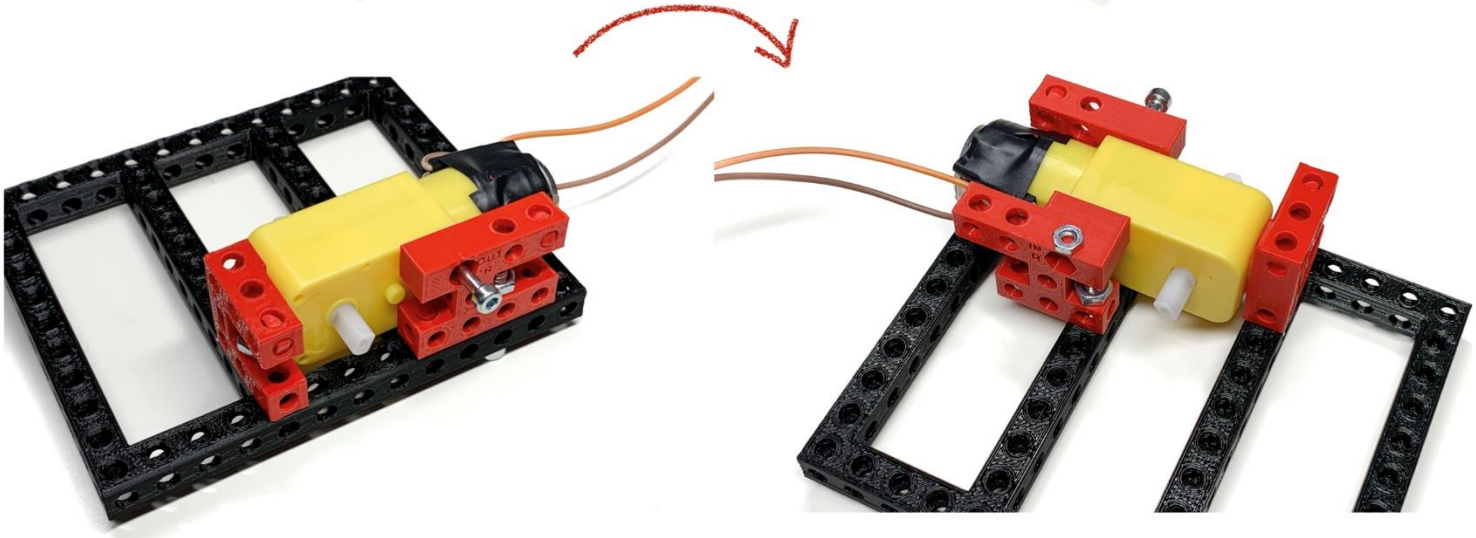


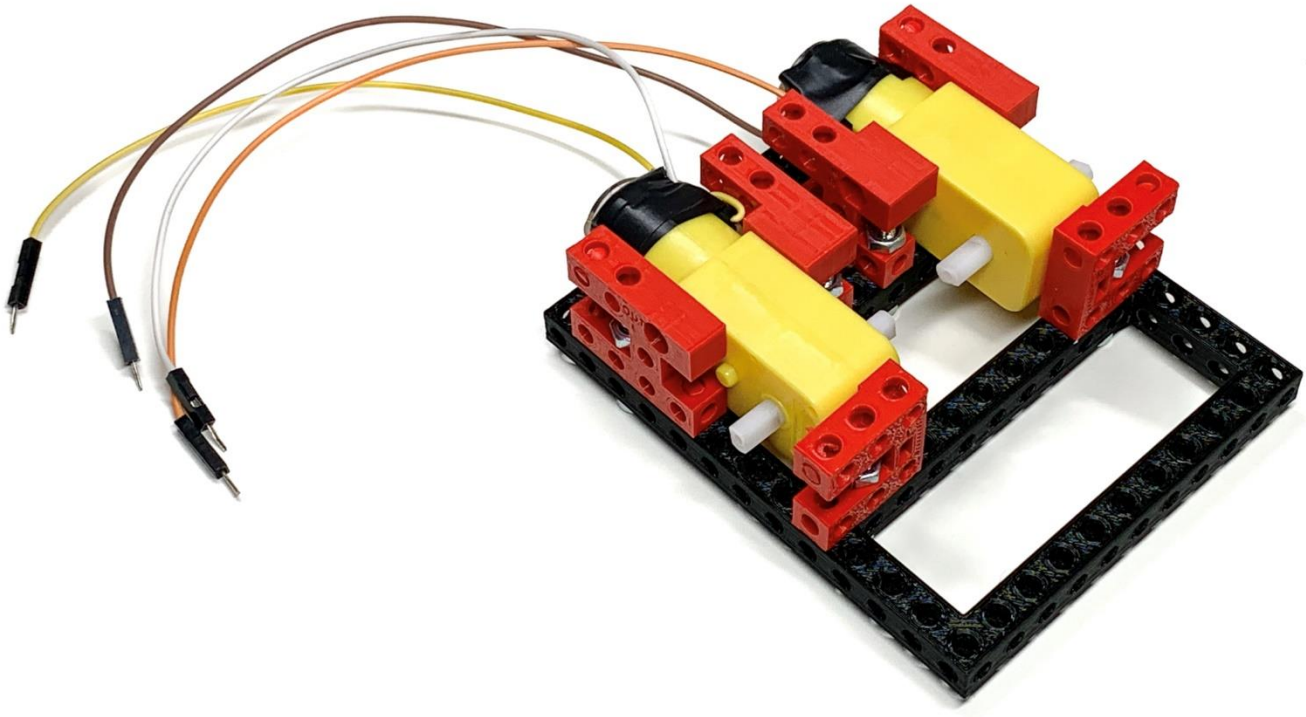
Pozor na označení dílků! Nezaměňte levé a pravé úchyty na motory!
L - levé
R - pravé


OUT
R
R
IN
R

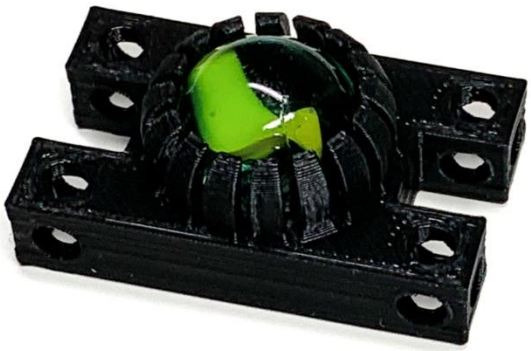
Pozor, dílky umisťujte tak, aby bylo vždy písmeno označující stranu vně!



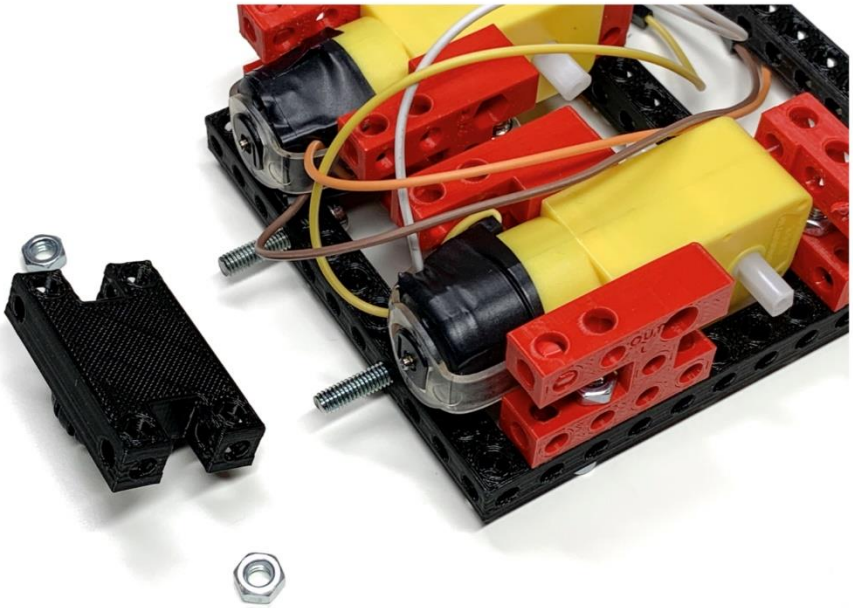


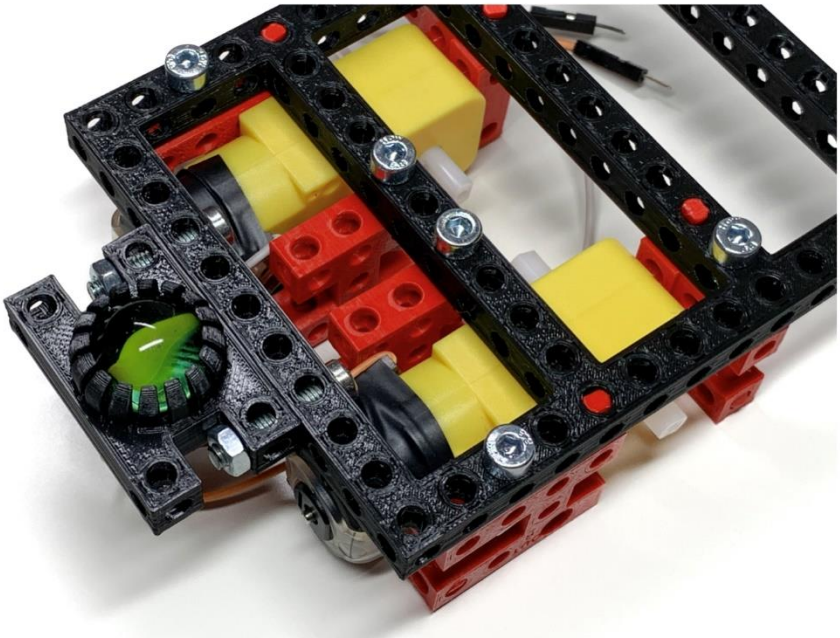


8

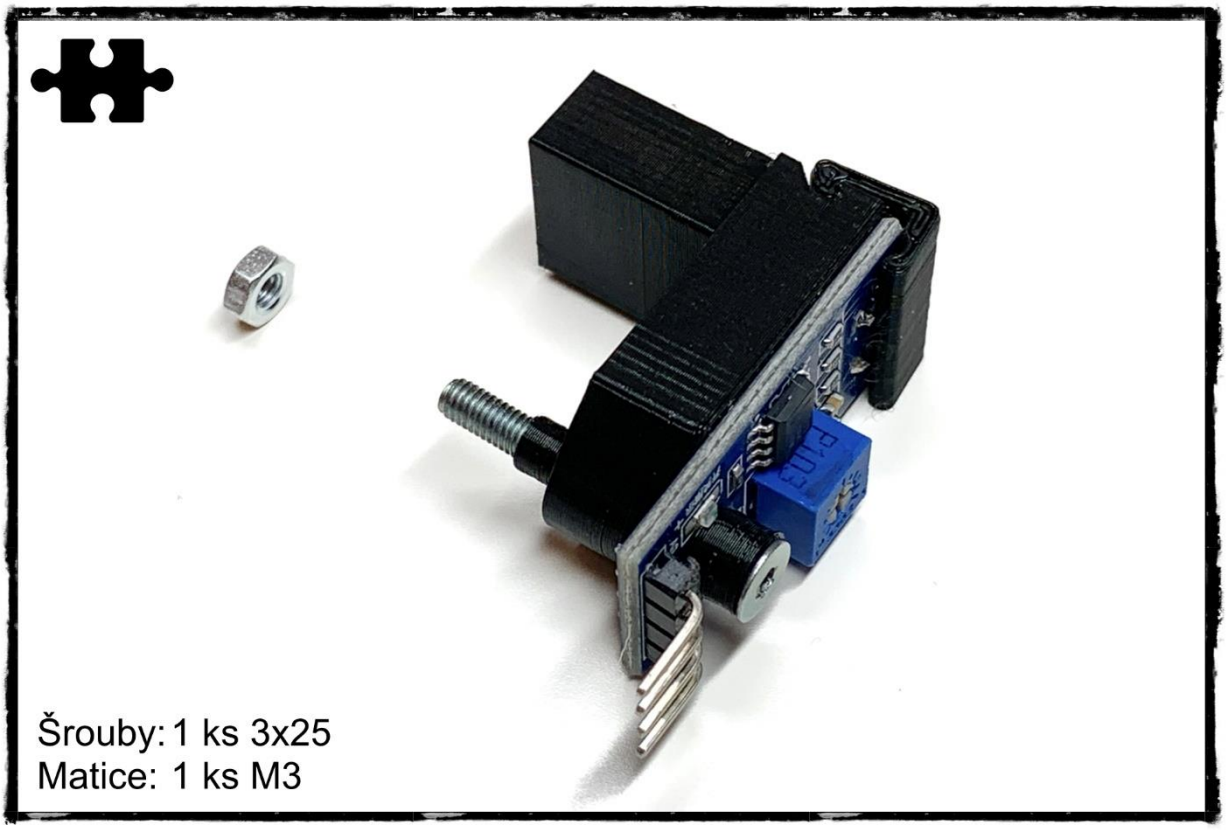


Šrouby: 2 ks 4x20
Matice: 2 ks M4

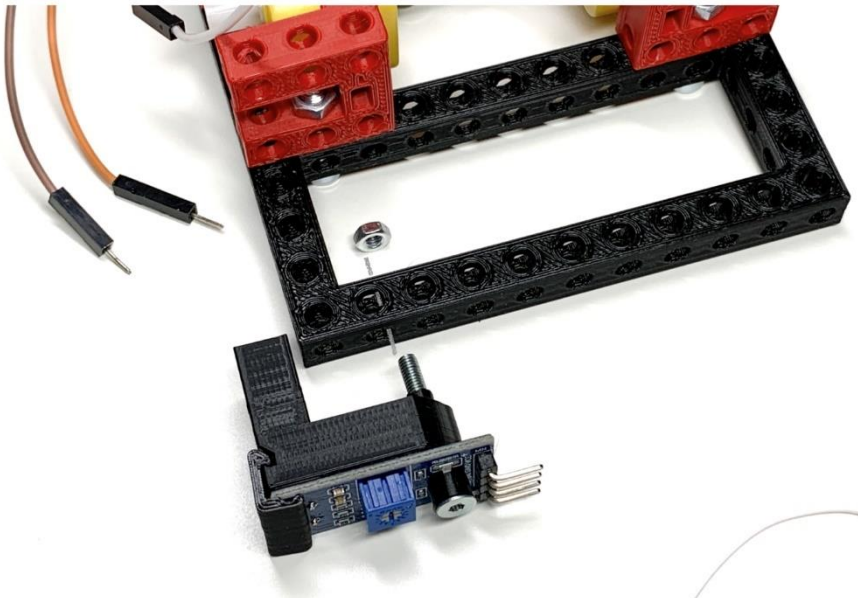


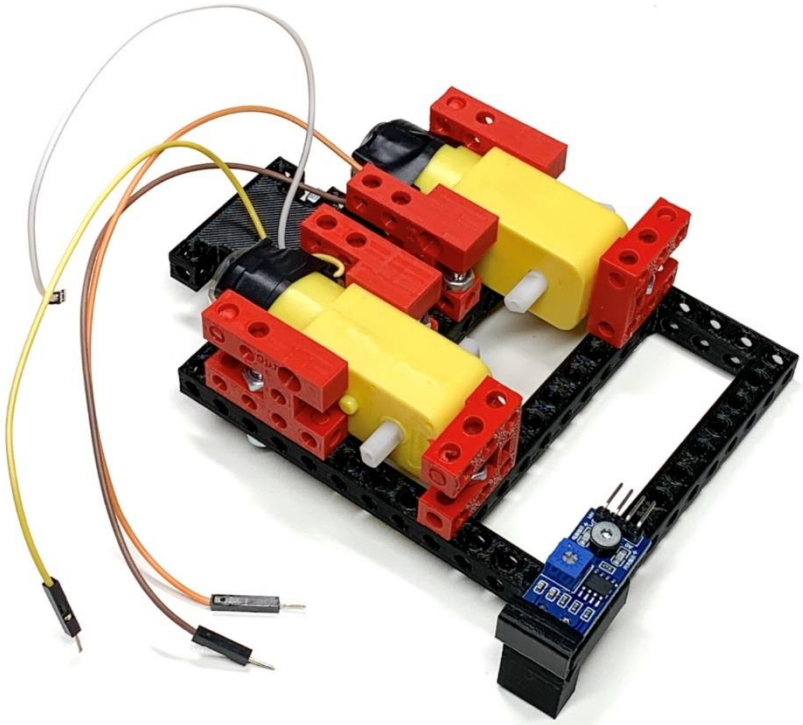


9



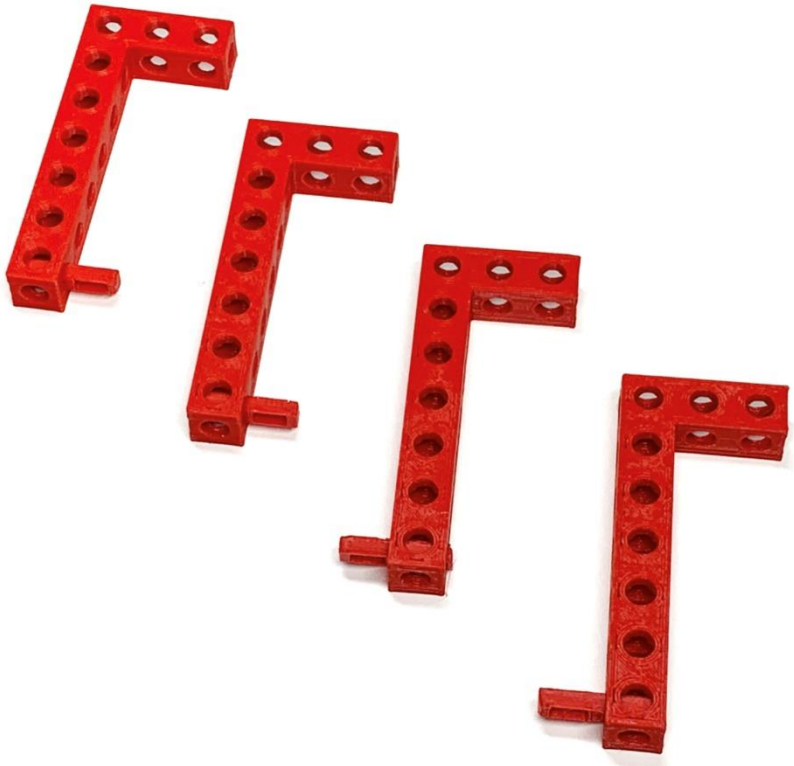
Šrouby: 1 ks 3x25
Matice: 1 ks M3

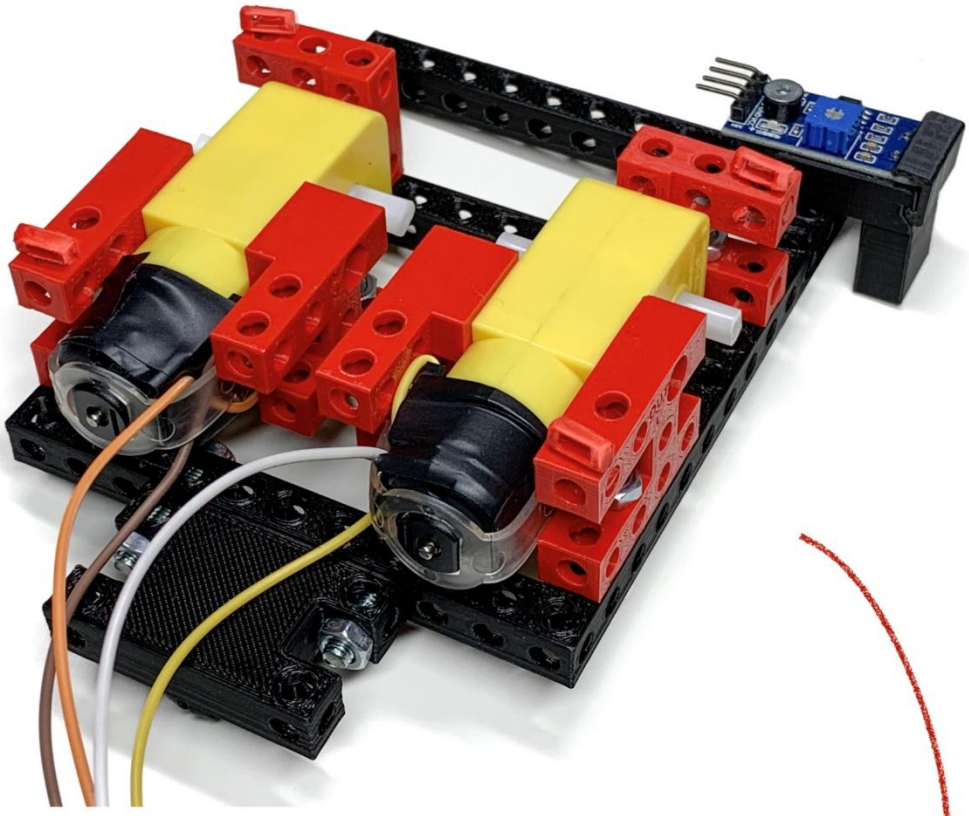




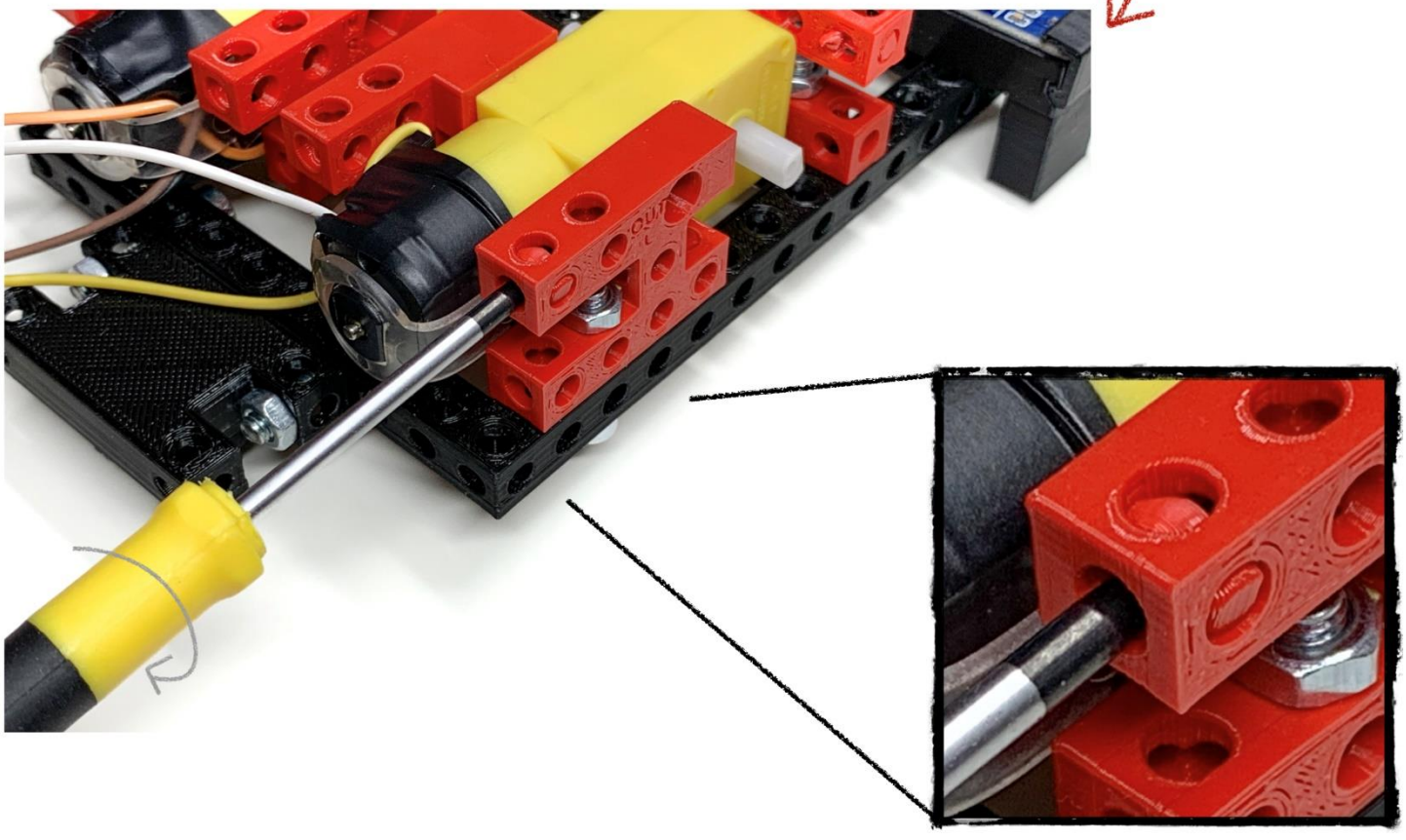
10

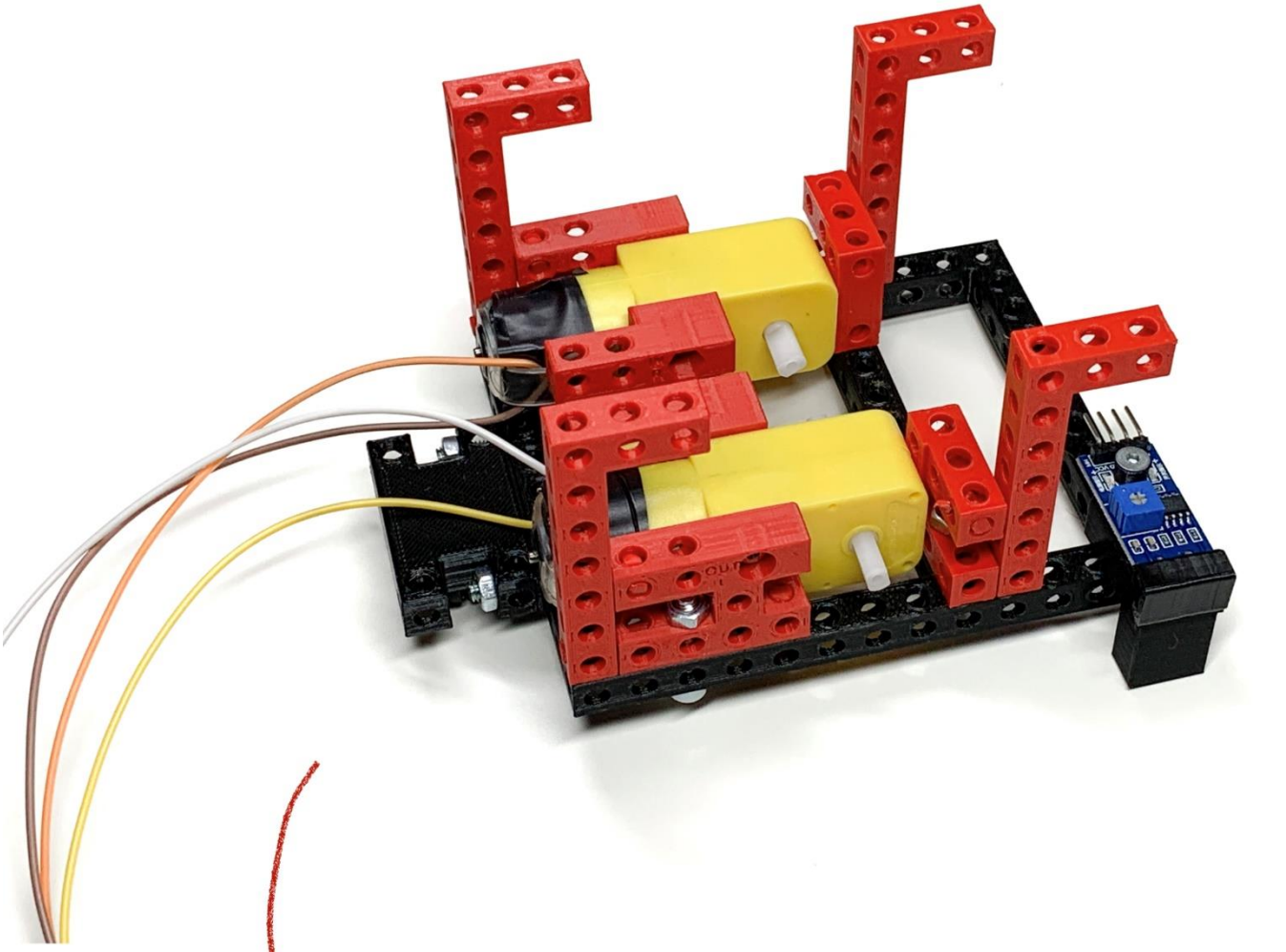
Šrouby: 4 ks 4x20

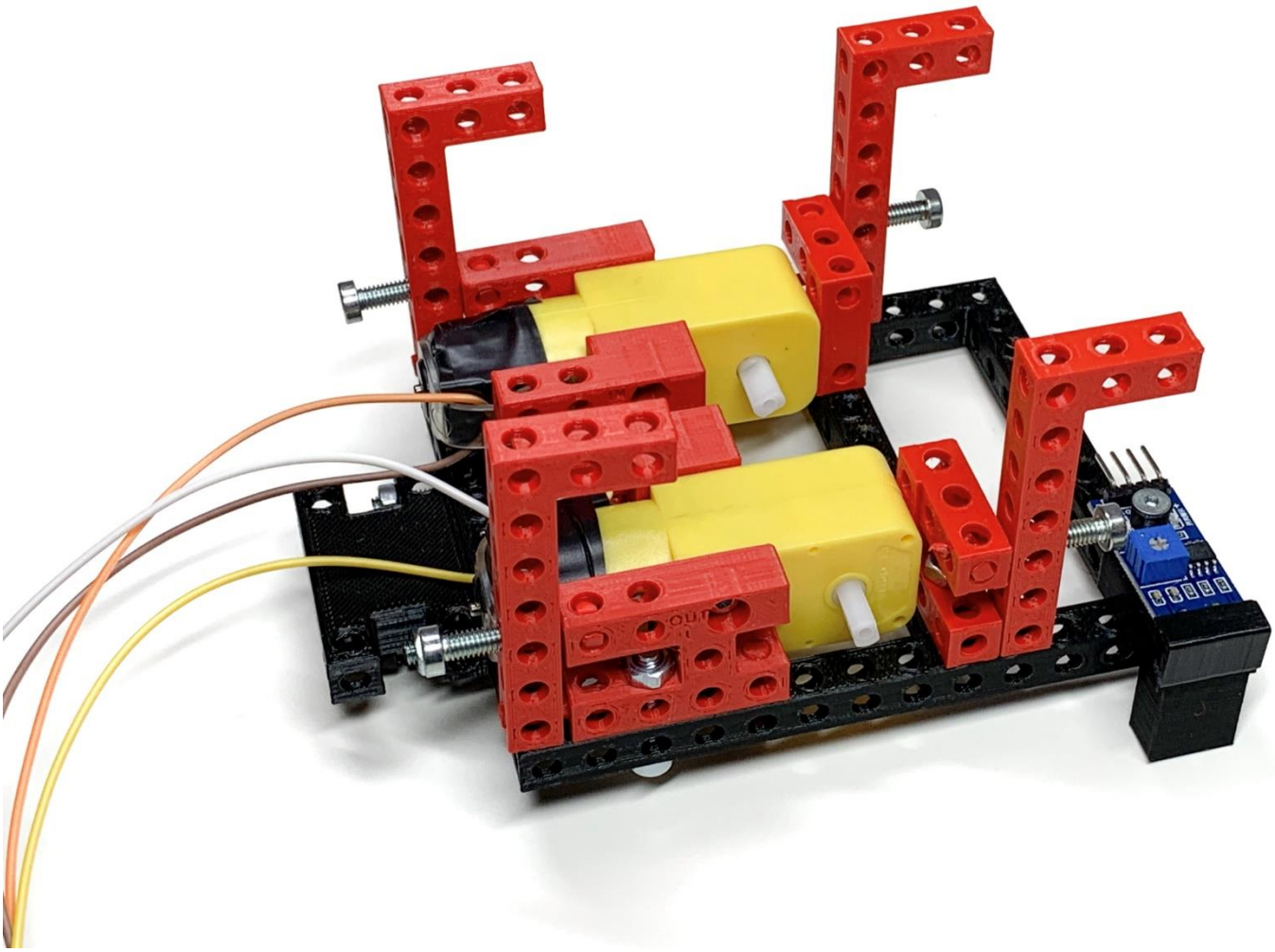




Po vsunutí hmoždinky zvětšete její otvor jemným otáčením plochým šroubovákem (3-4 mm), tak aby vznikl kulatý otvor a lépe se do něj zašrouboval šroub.

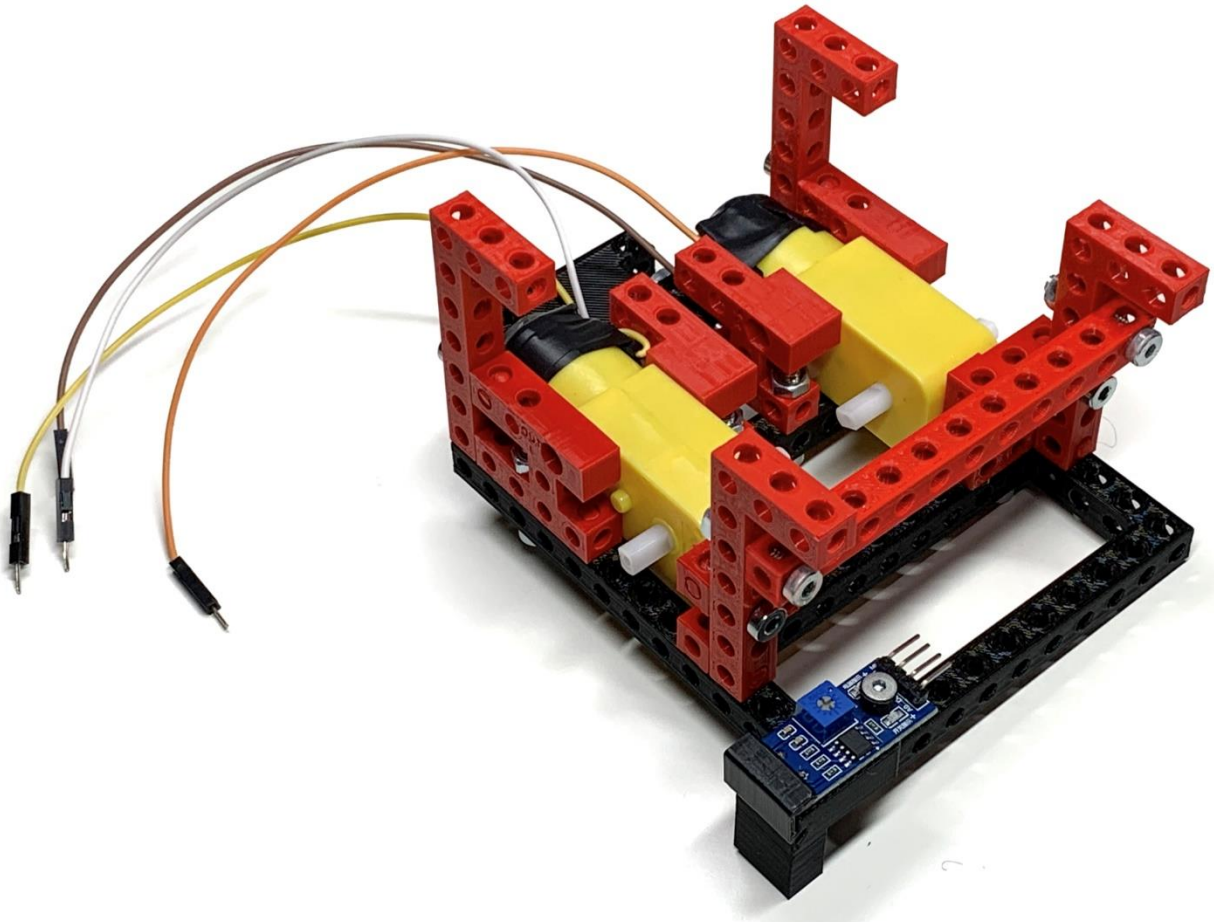




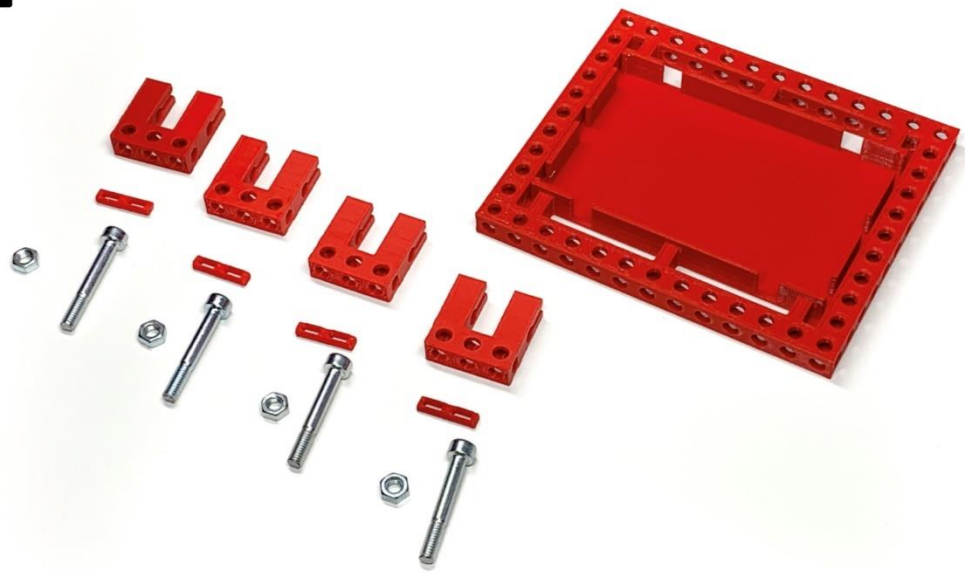


11

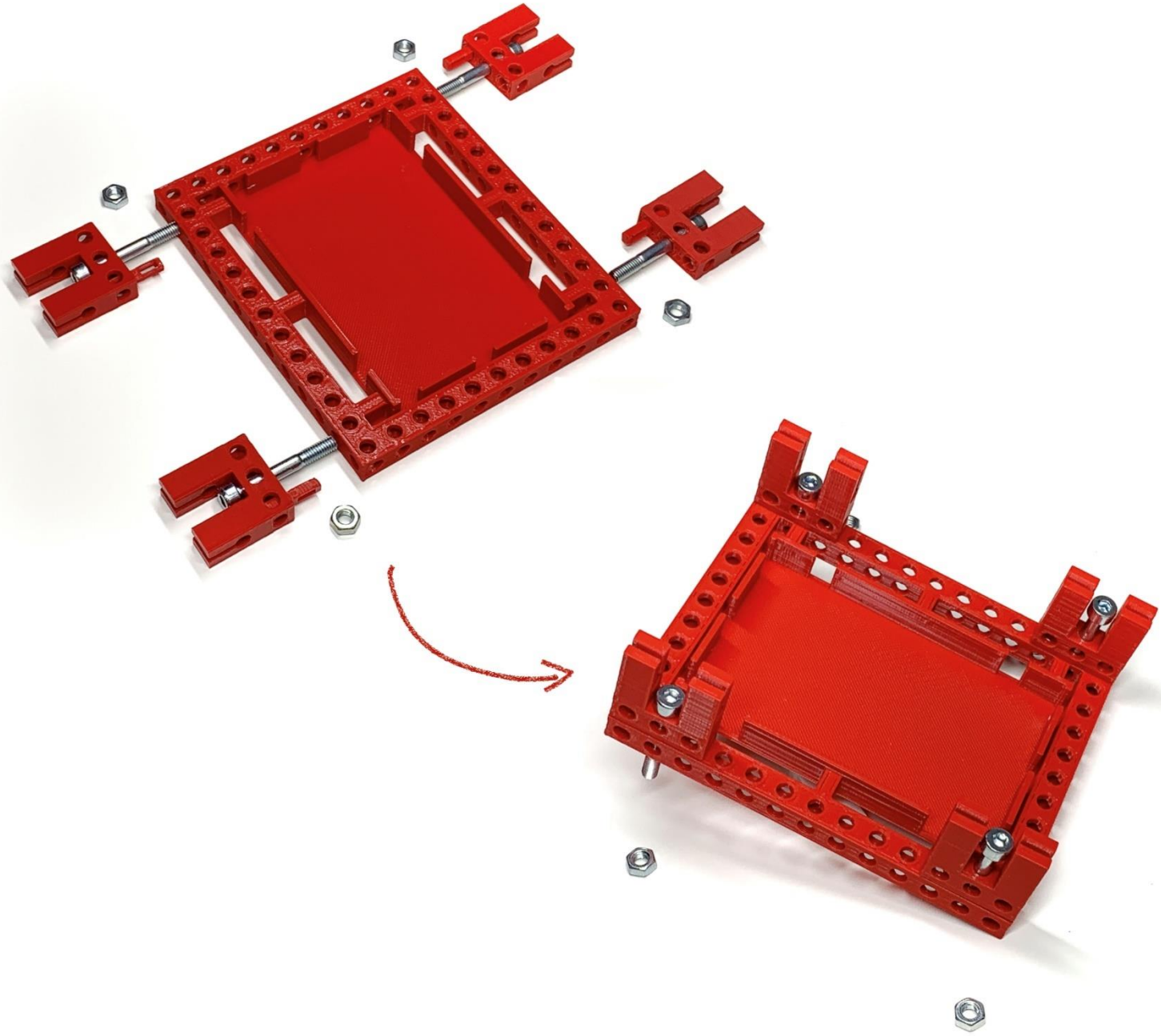
Šrouby: 2 ks 4x20
Matice: 2 ks M4



12

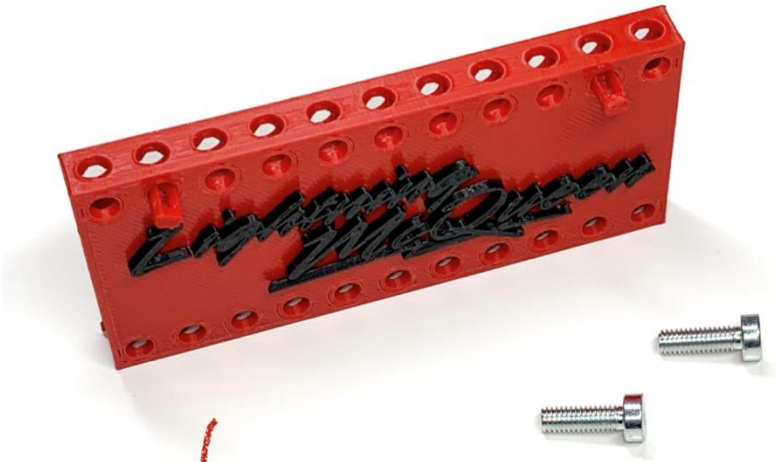


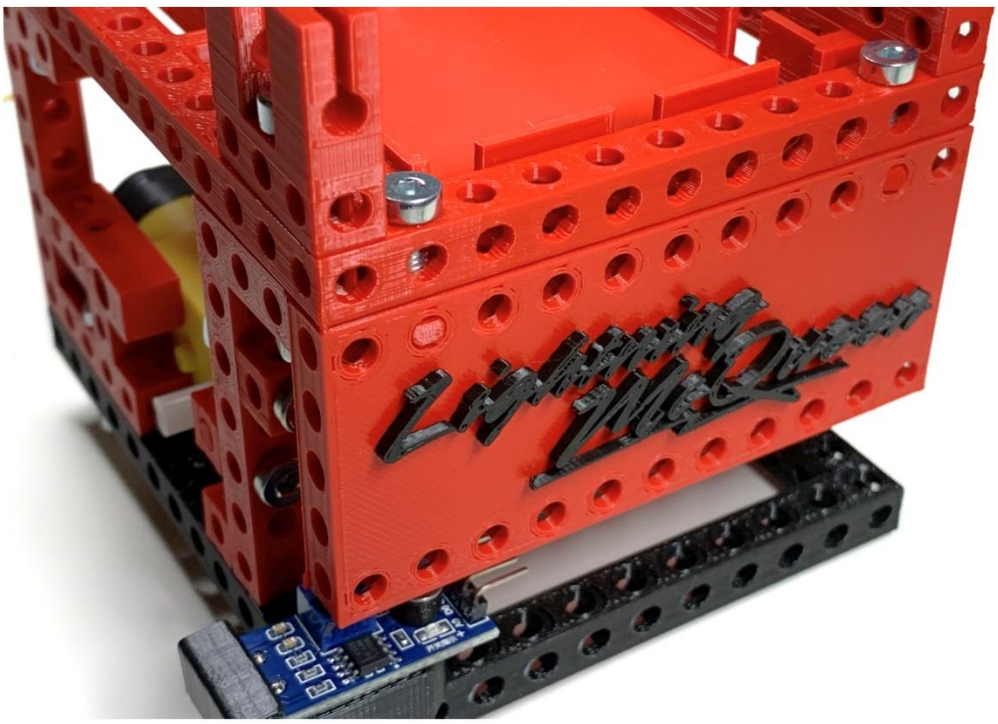
Šrouby: 4 ks 4x30
Matice: 4 ks M4



13

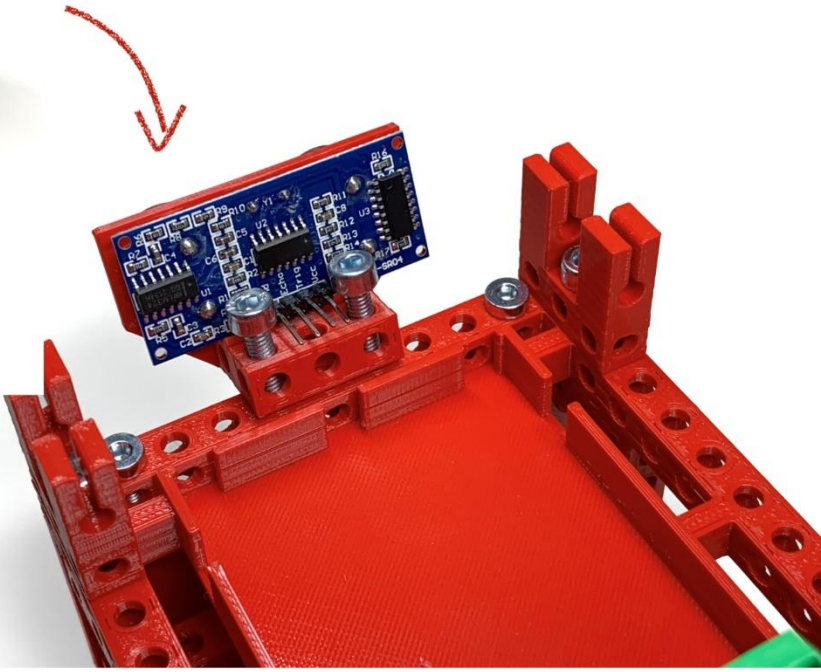
Šrouby: 2 ks 4x16

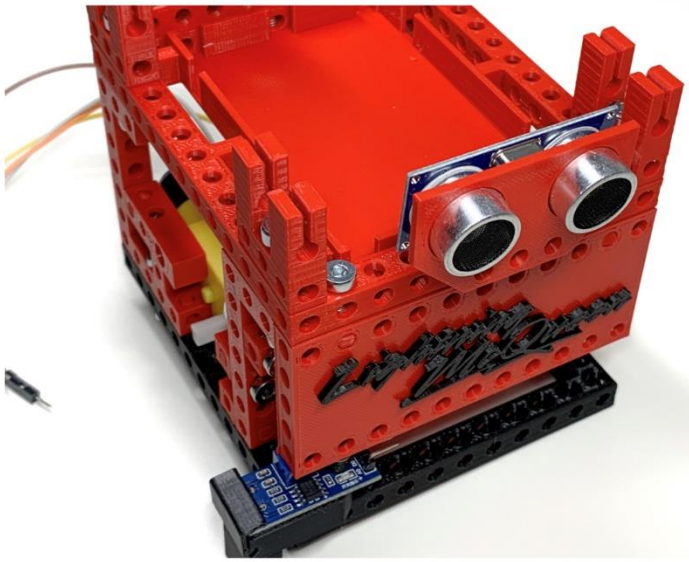




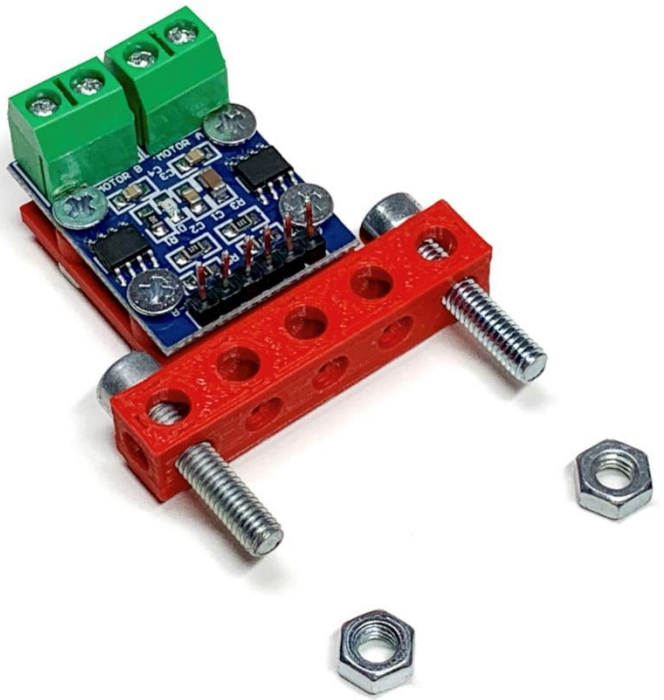
14

Šrouby: 2 ks 4x16

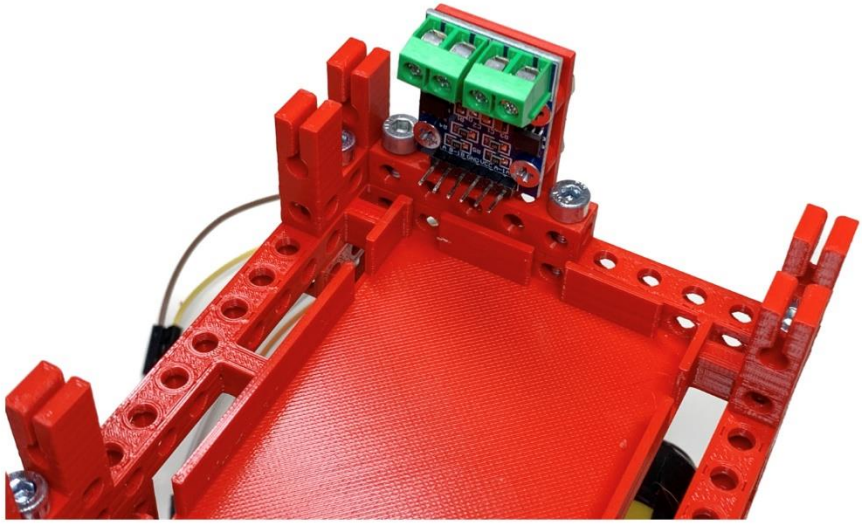


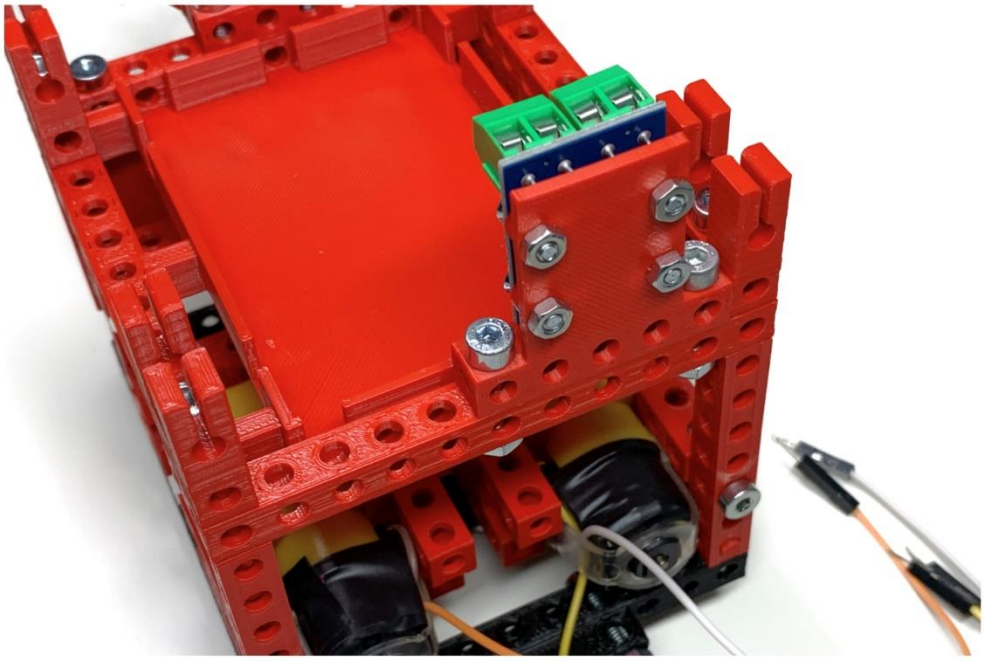


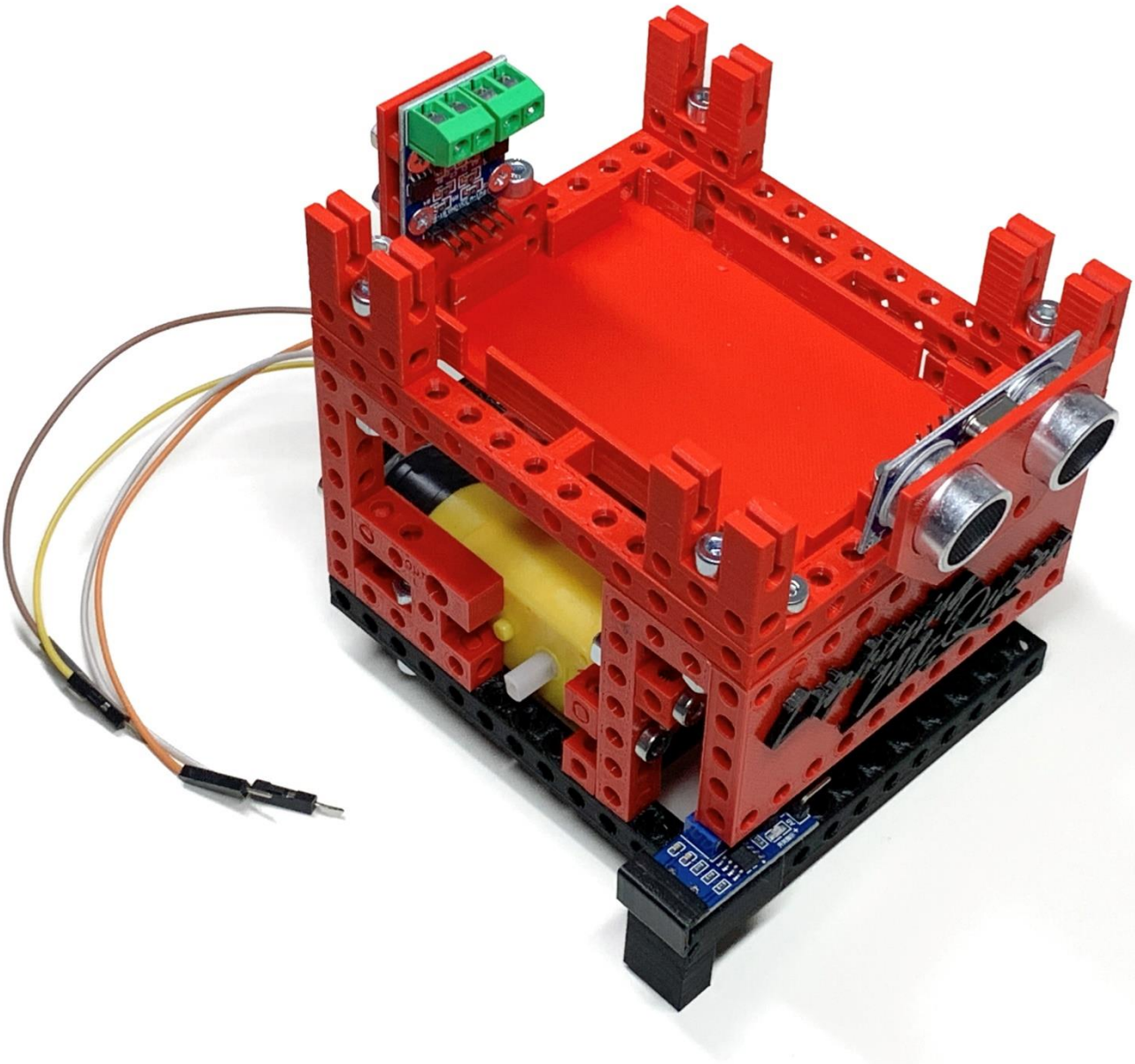
15



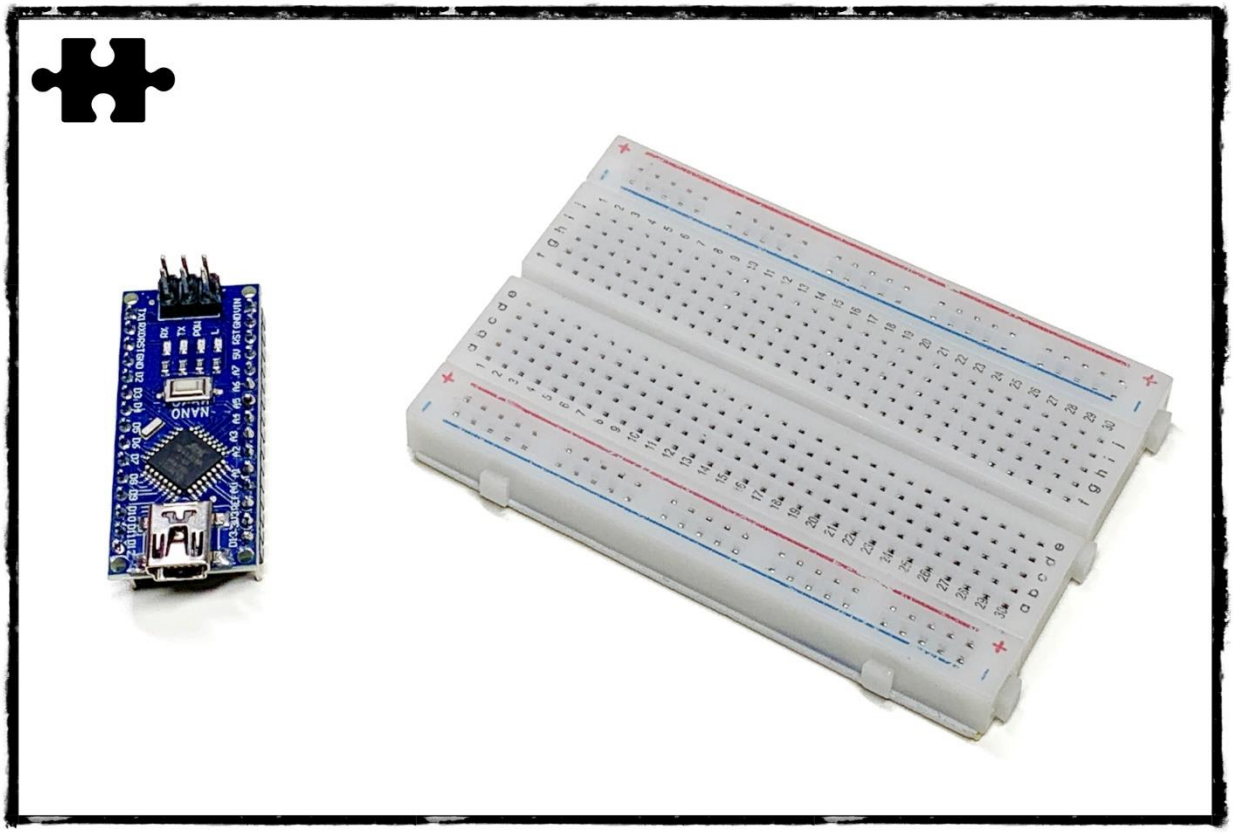
Šrouby: 2 ks 4x20
Matice: 2 ks M4

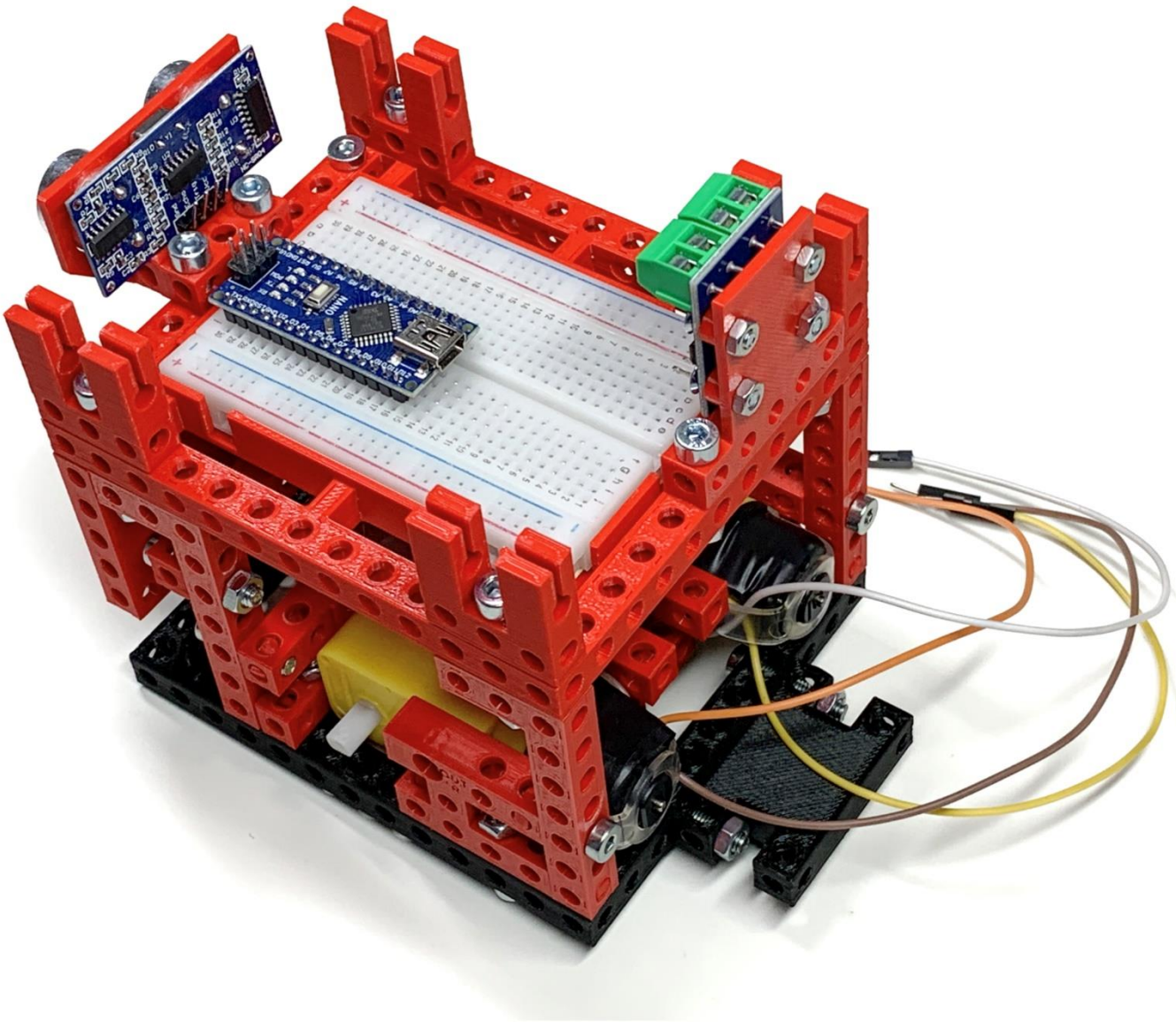






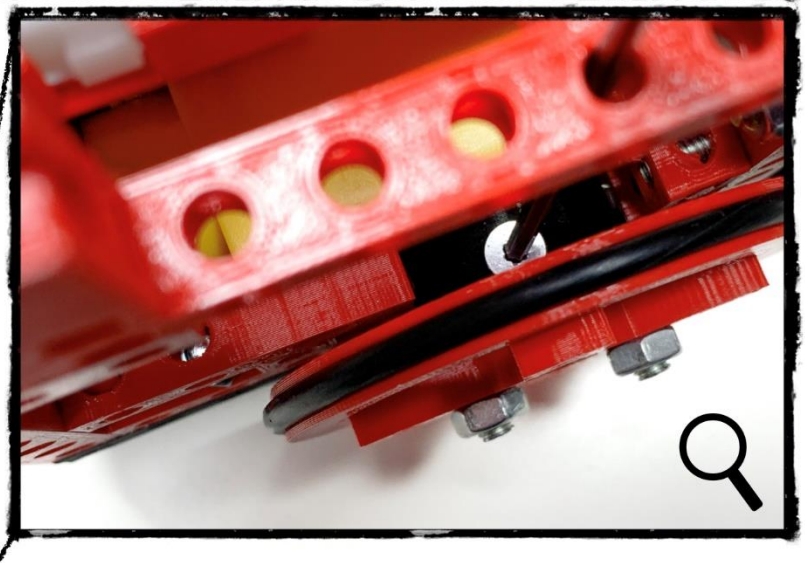
16

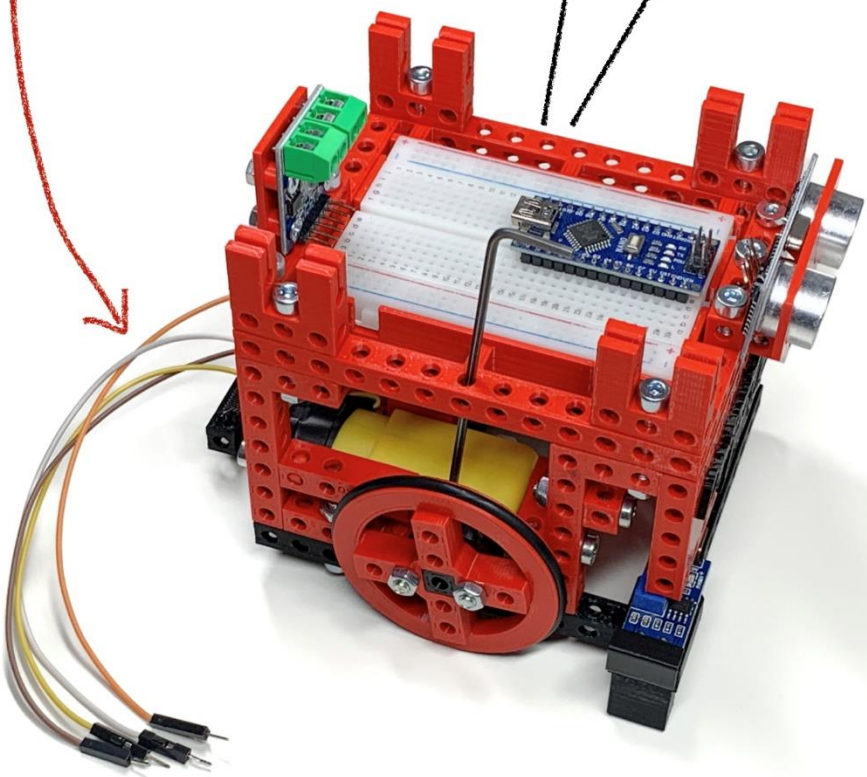


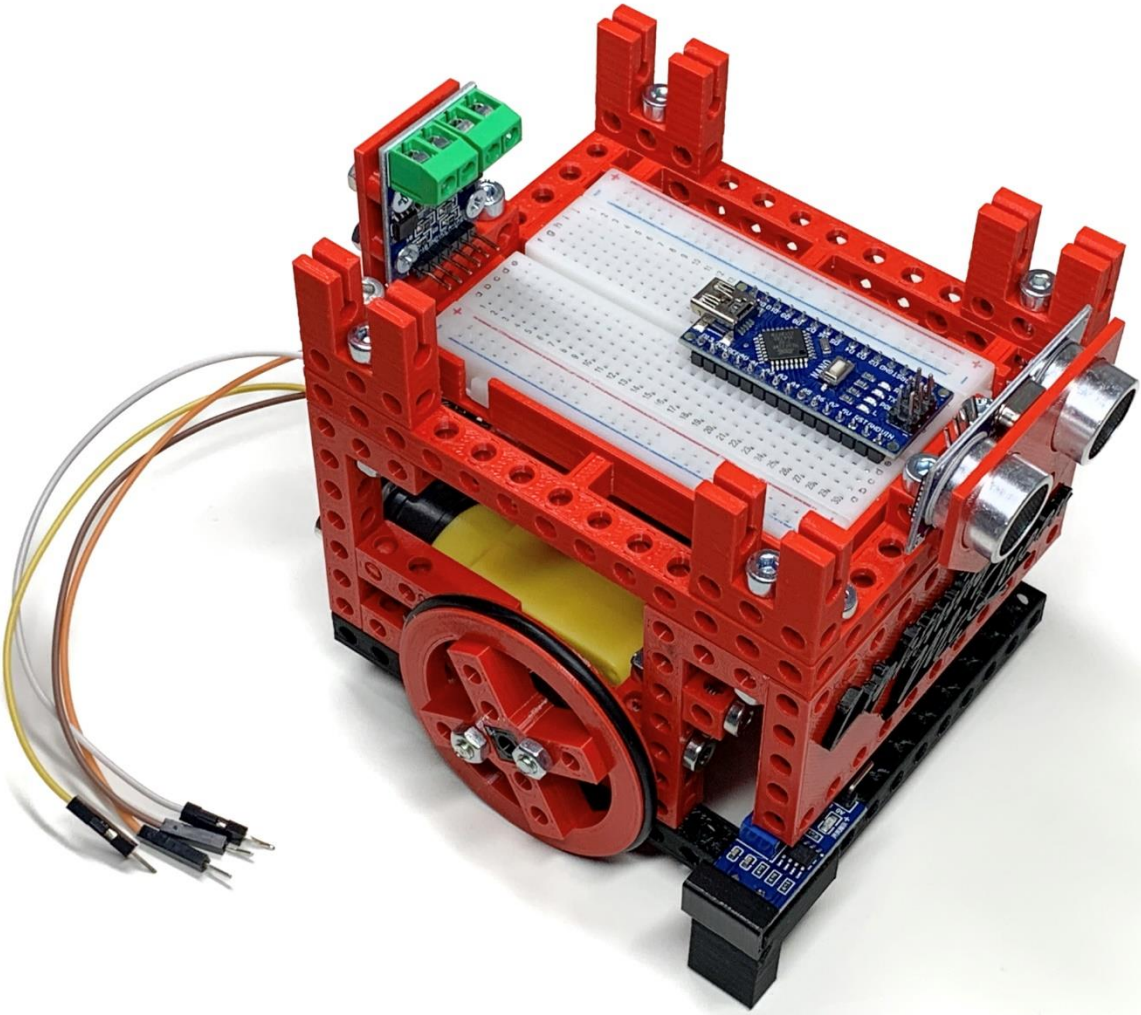


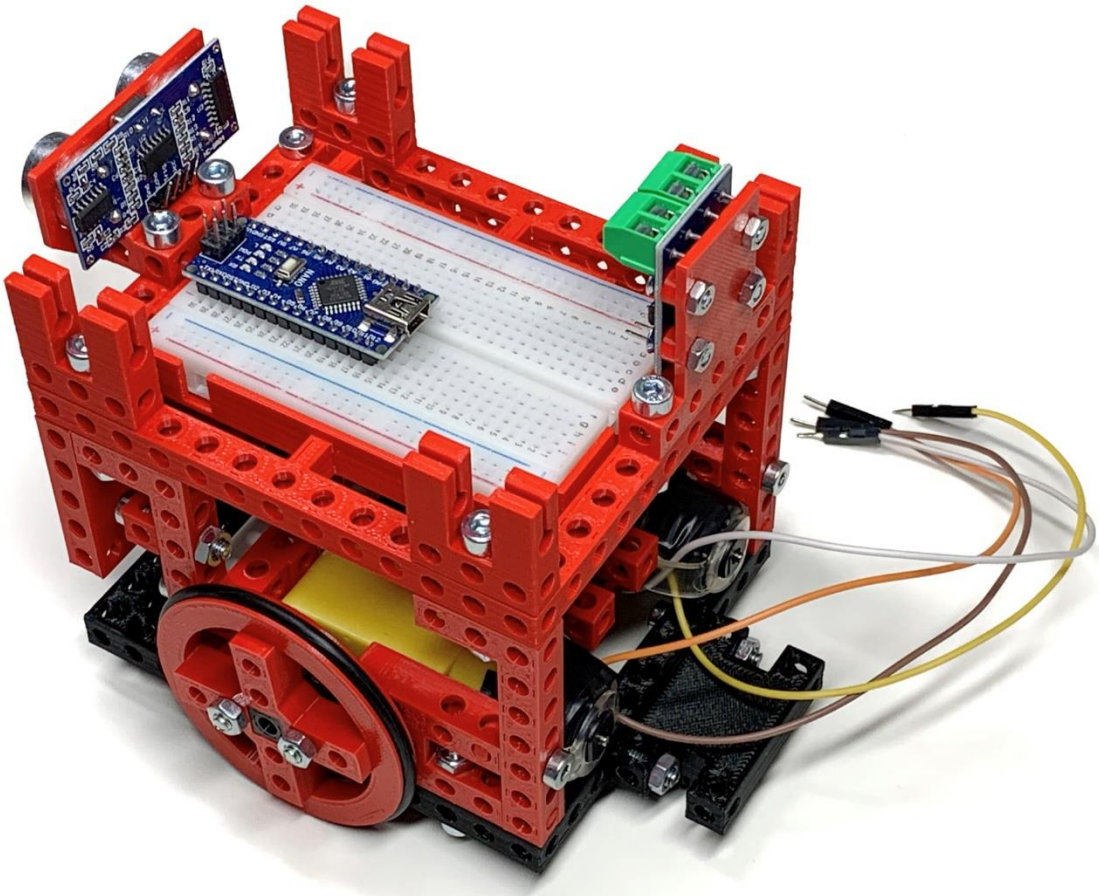
17

Šrouby: 2 ks 3x8
Matice: 2 ks M3

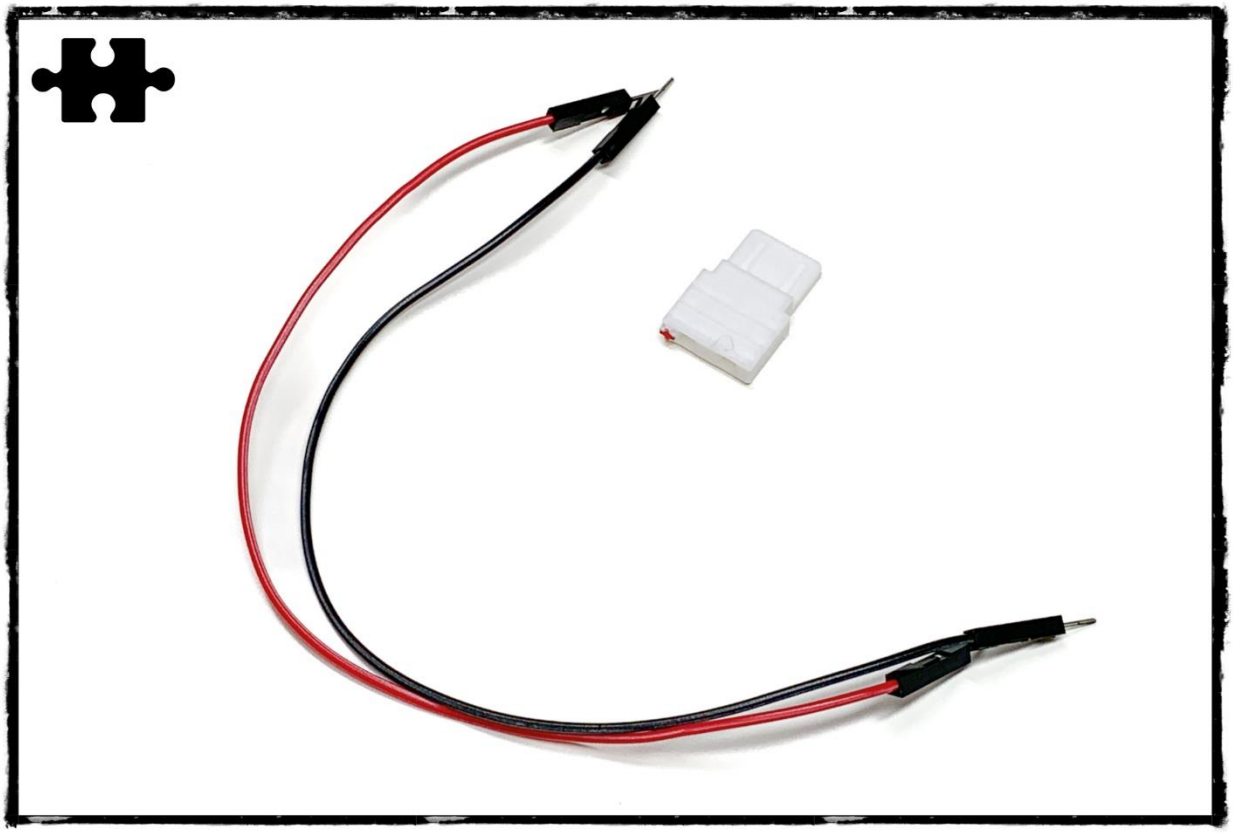


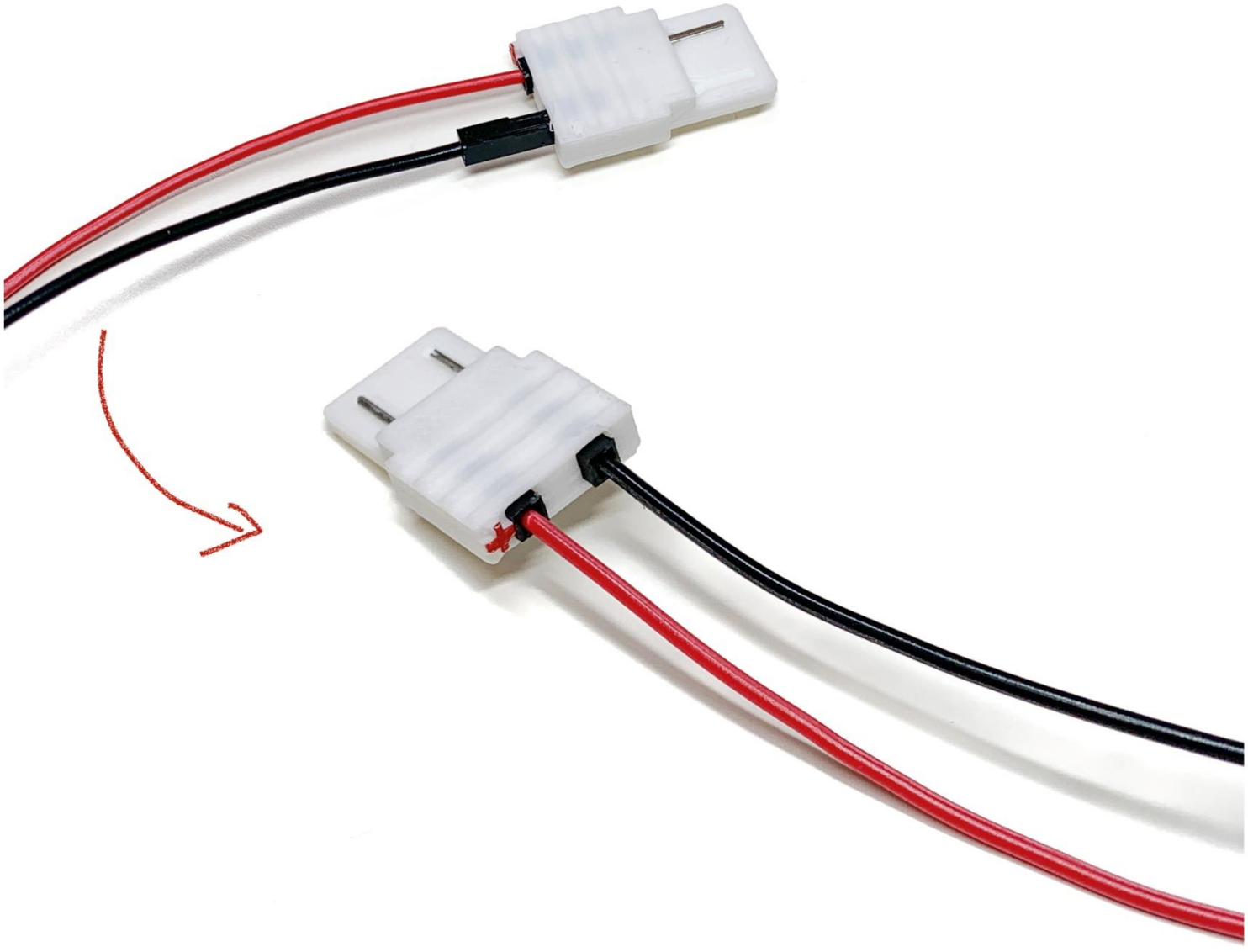




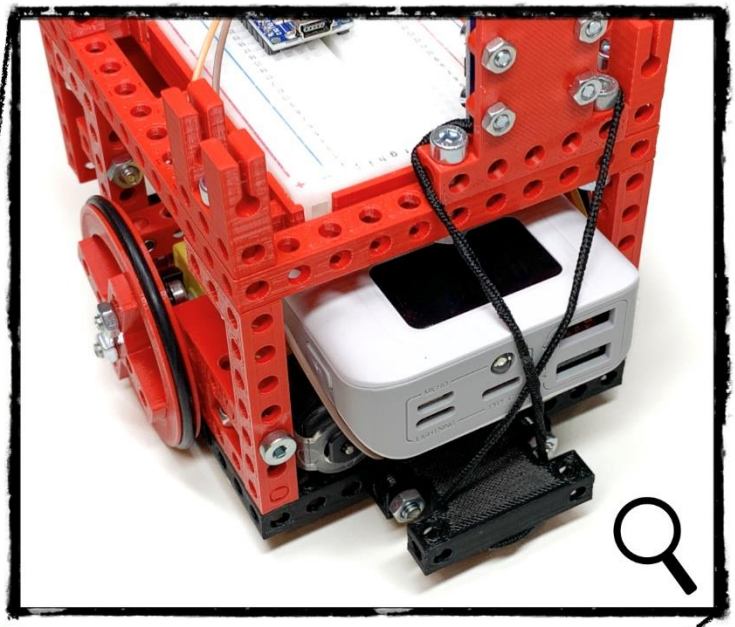


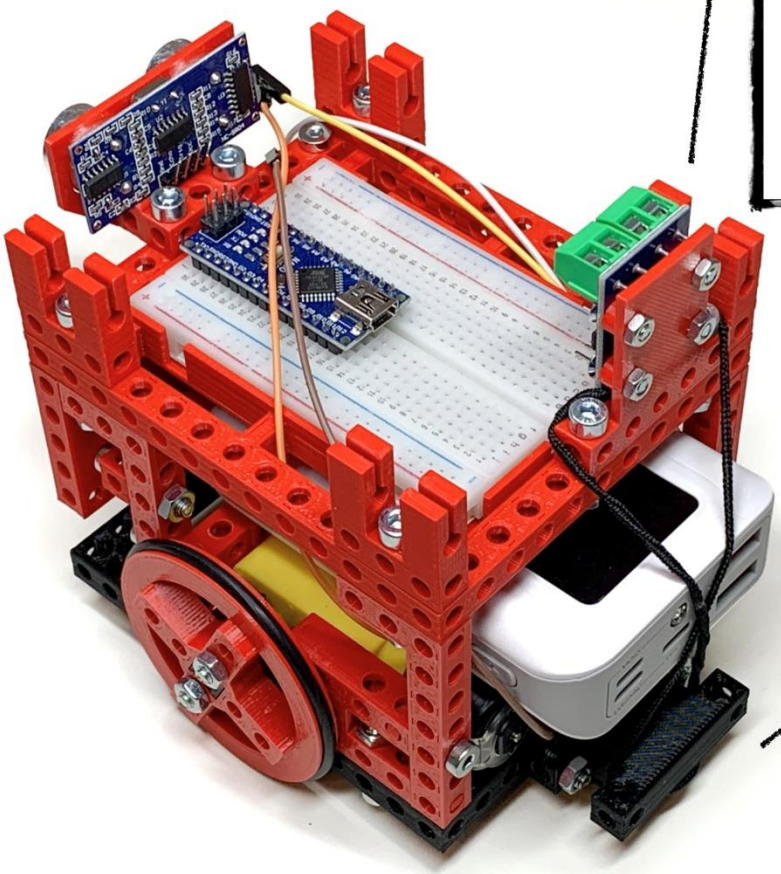
18



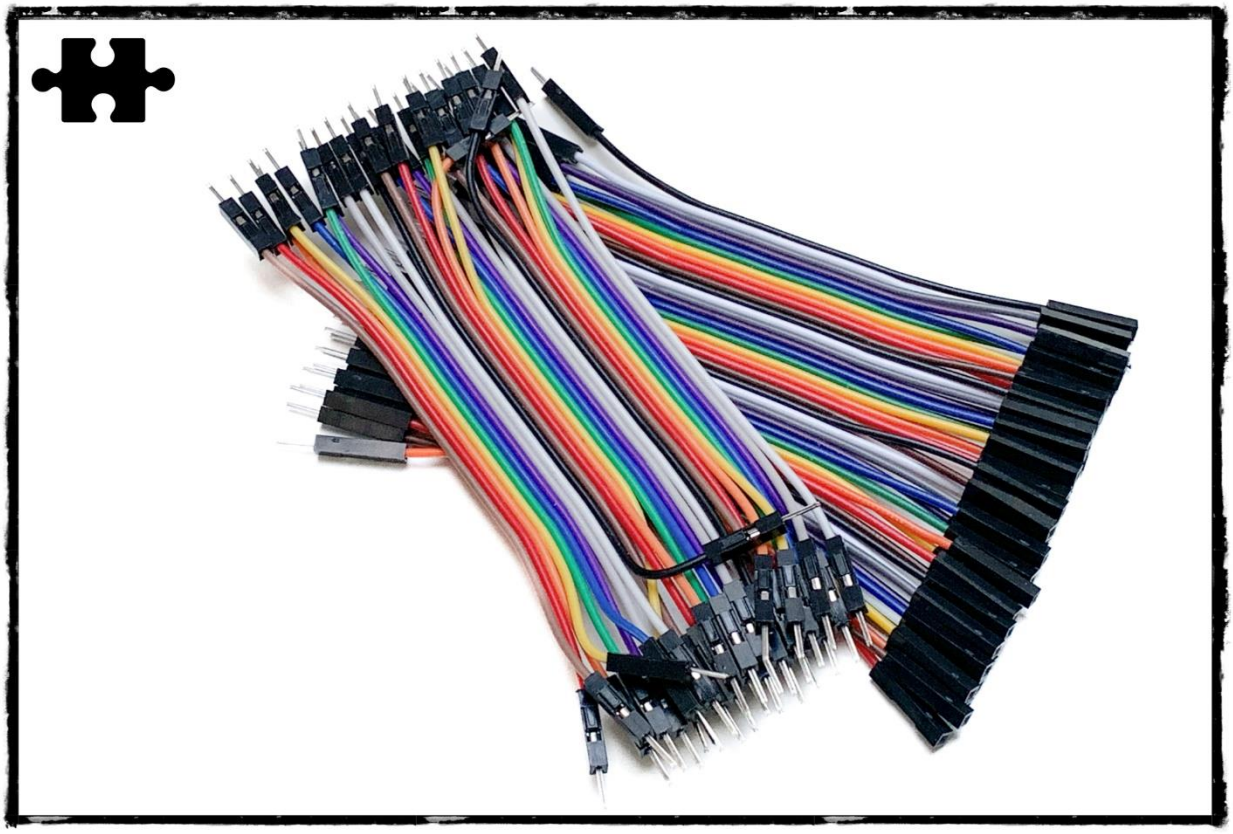


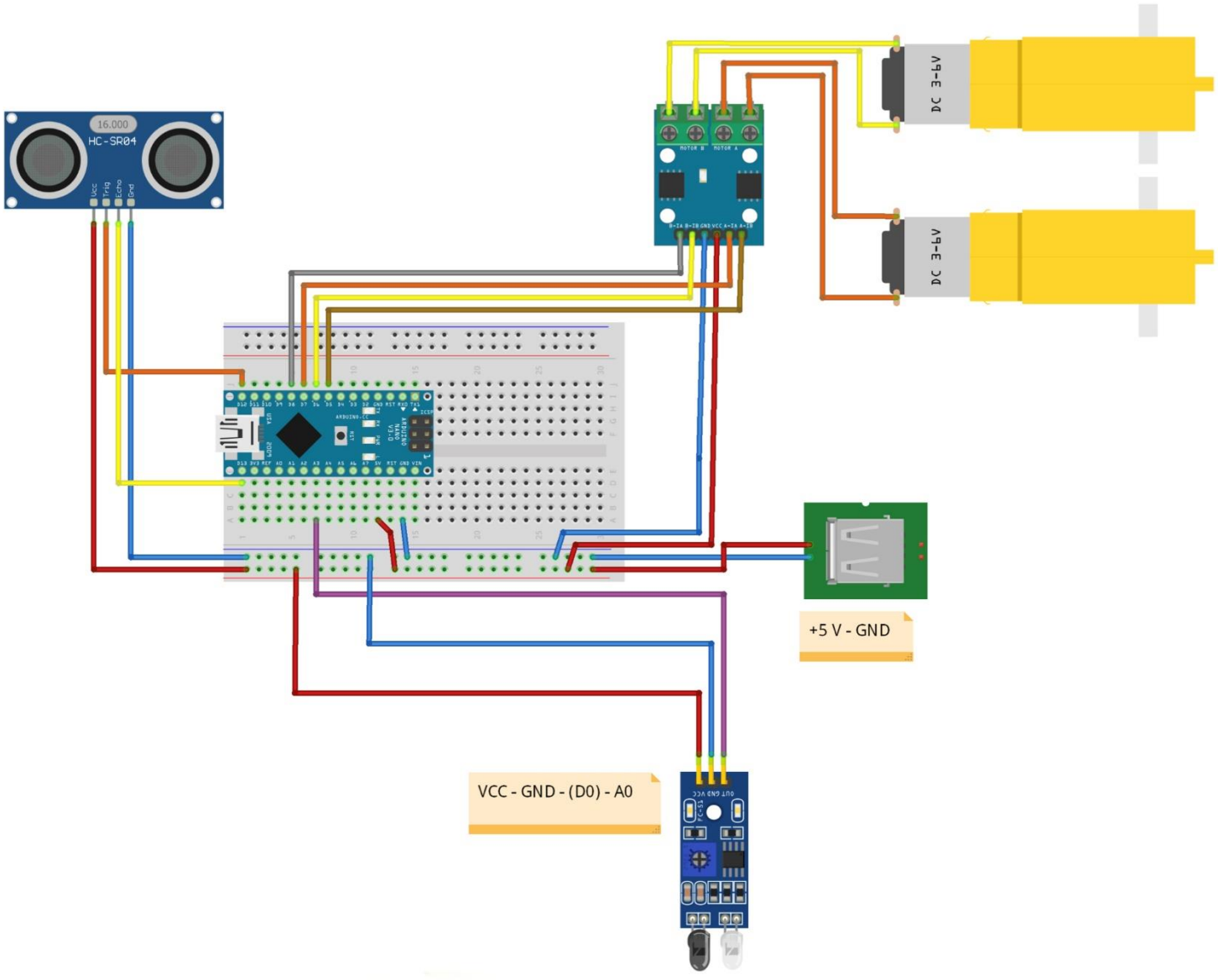
19

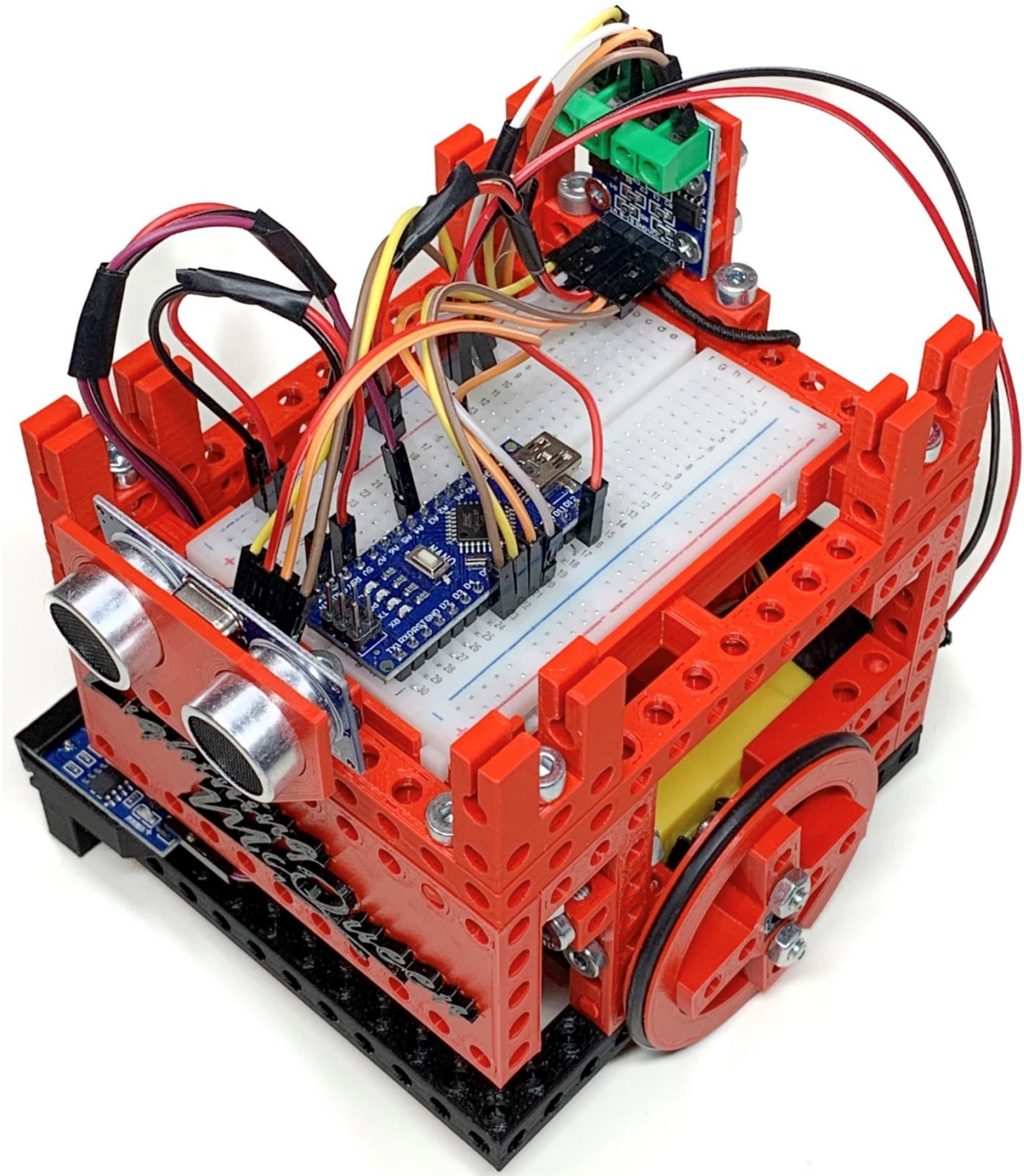




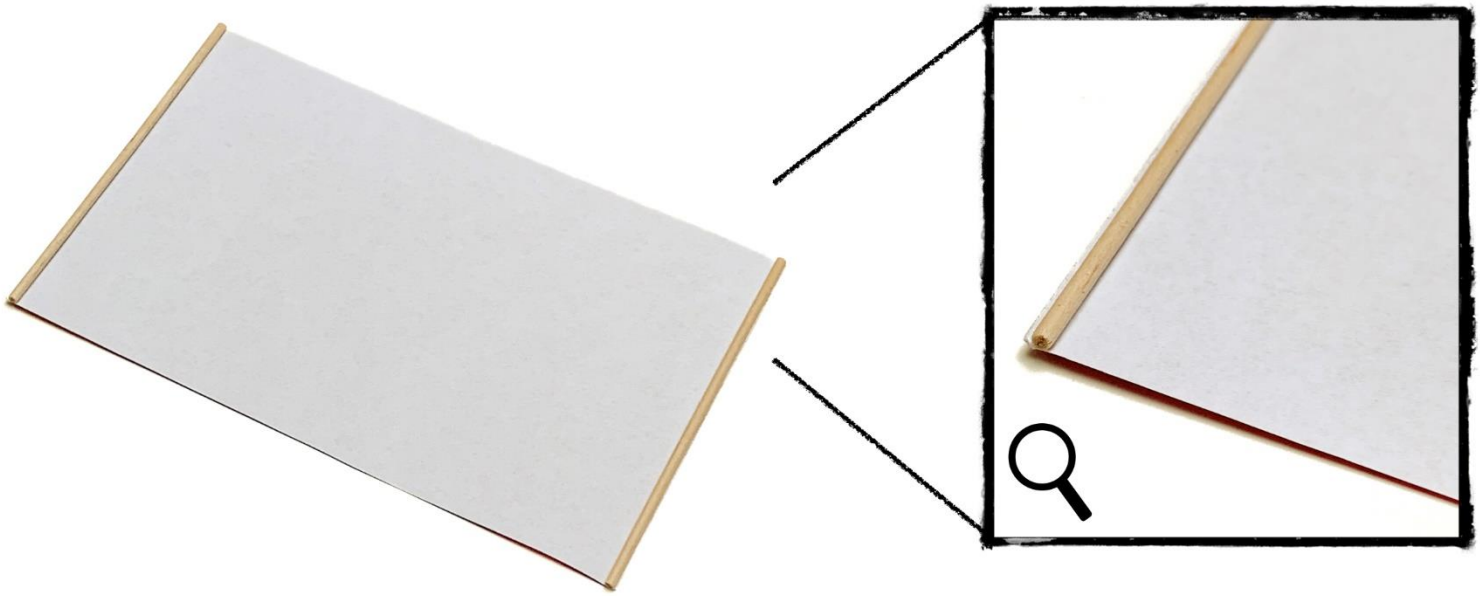
20



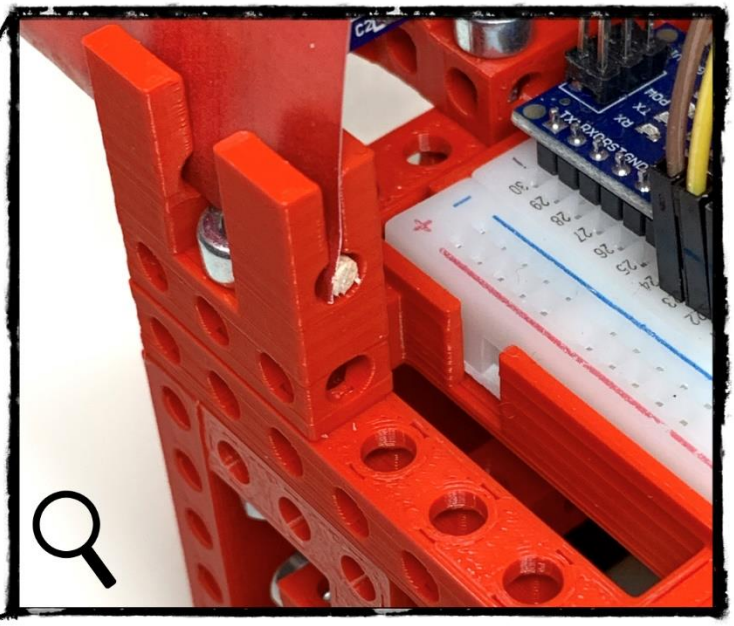

16.000
HC-SR04
Vcc
Trig
Echo
Gnd
DC 3-6V
DC 3-6V
+5 V - GND
VCC - GND - (D0) - A0

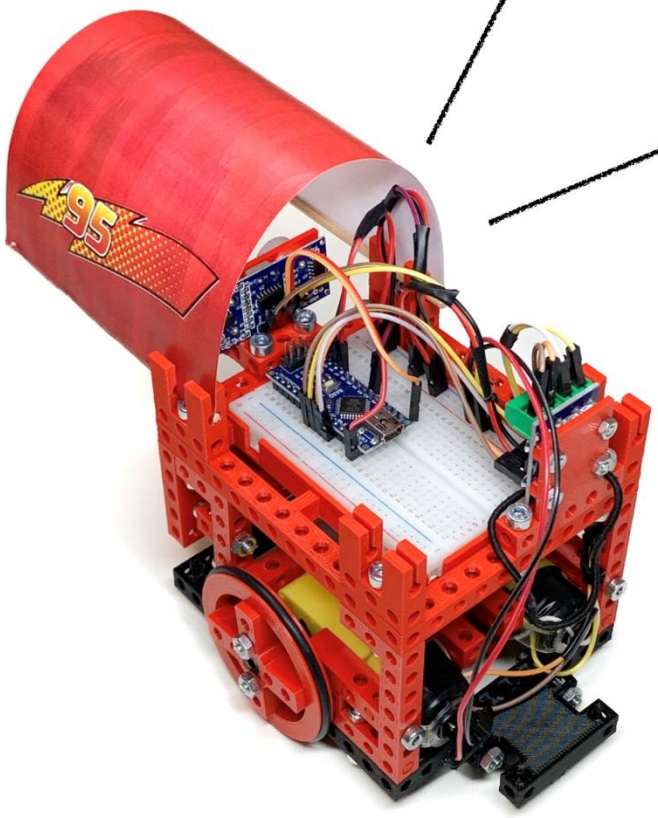


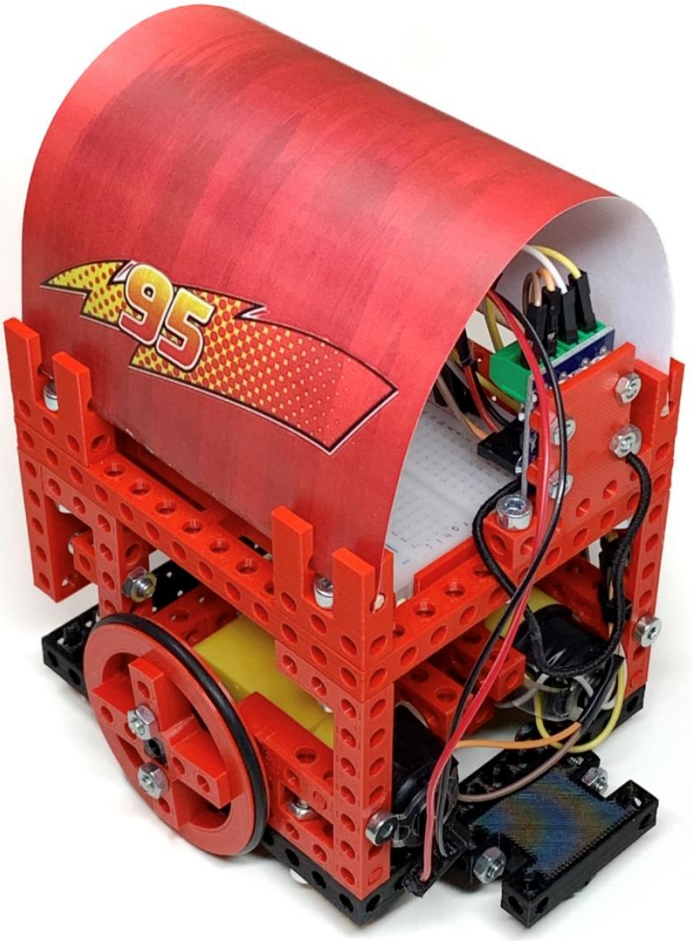
21

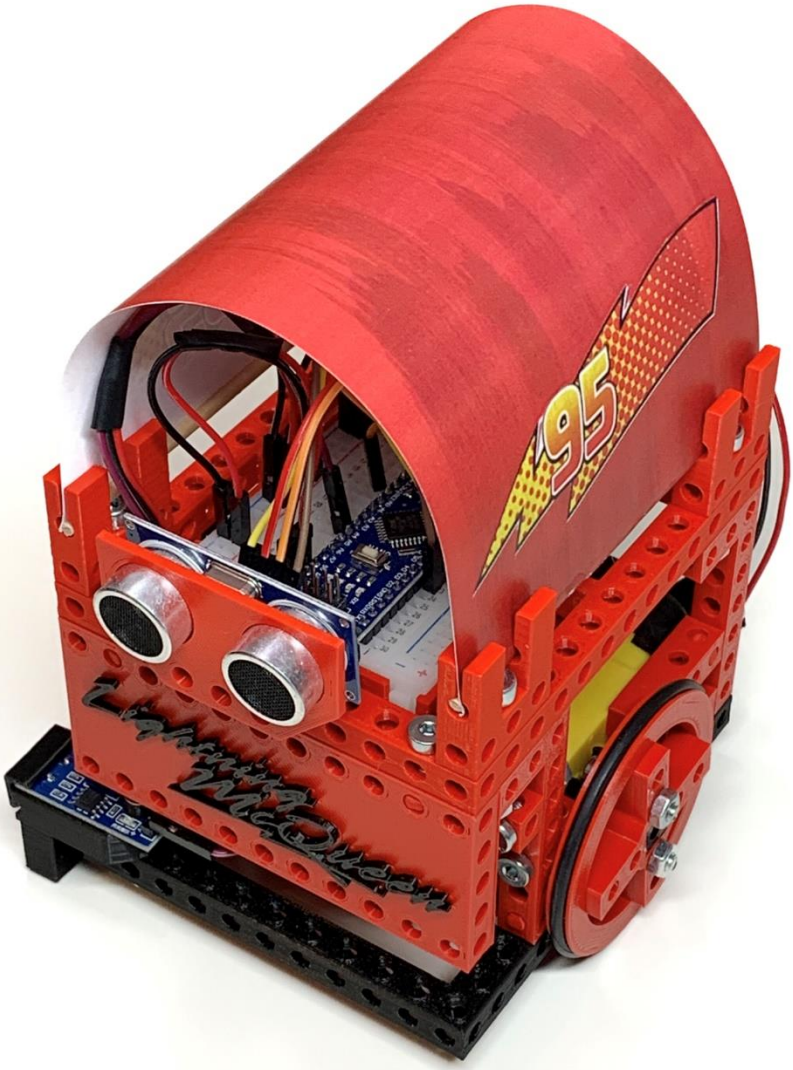


22

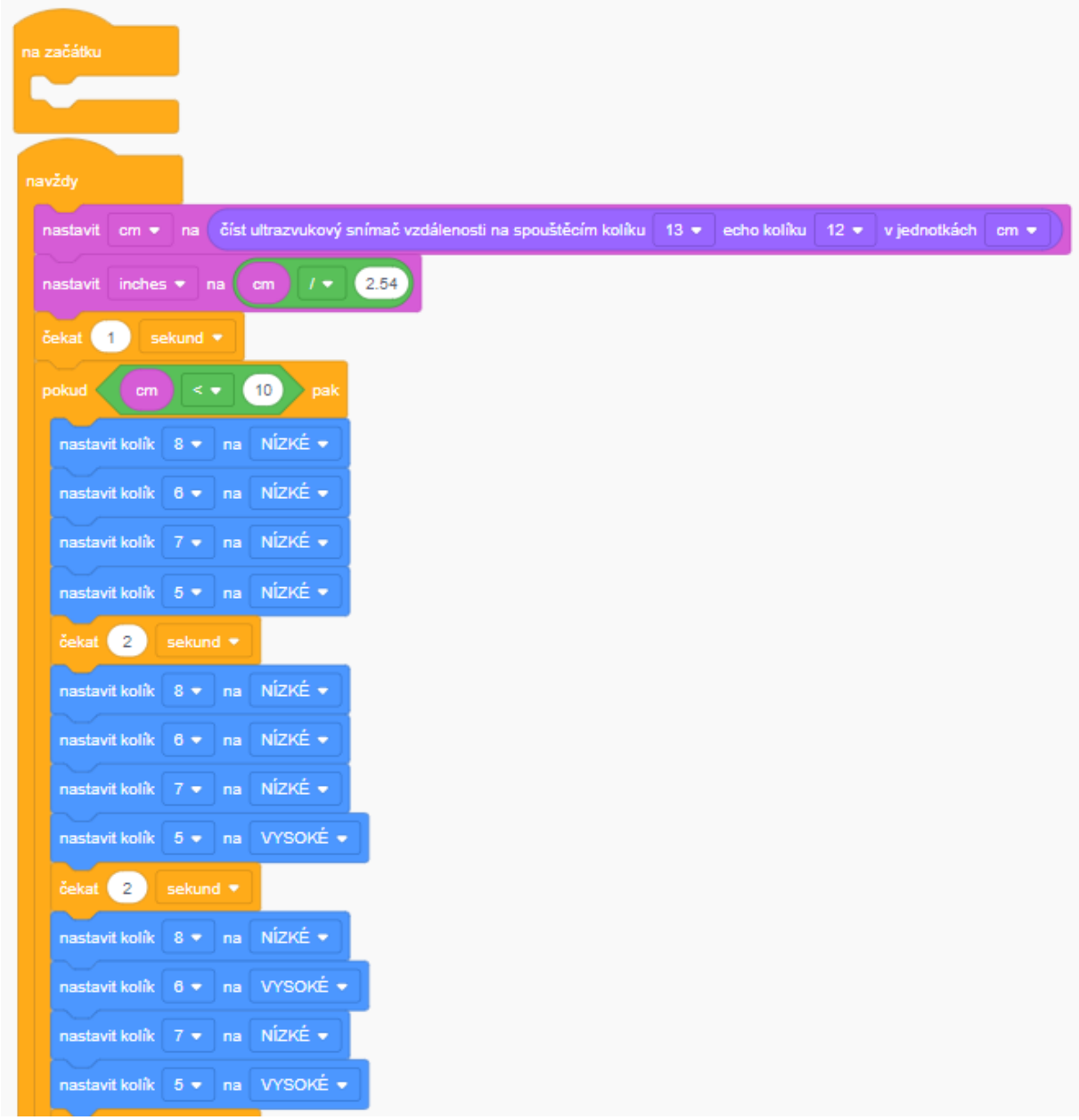


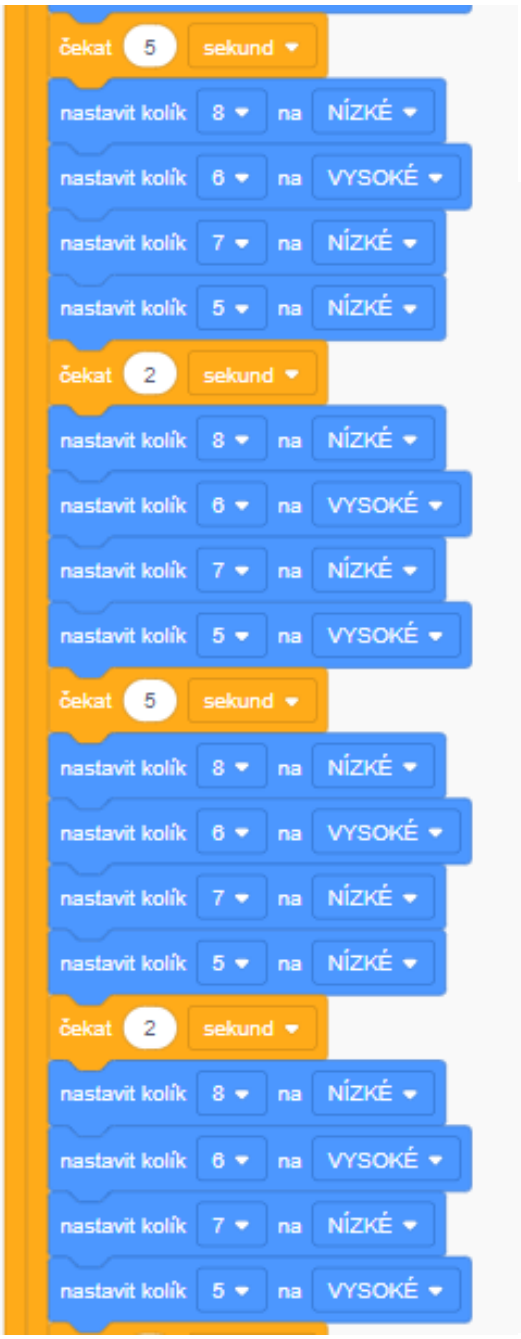


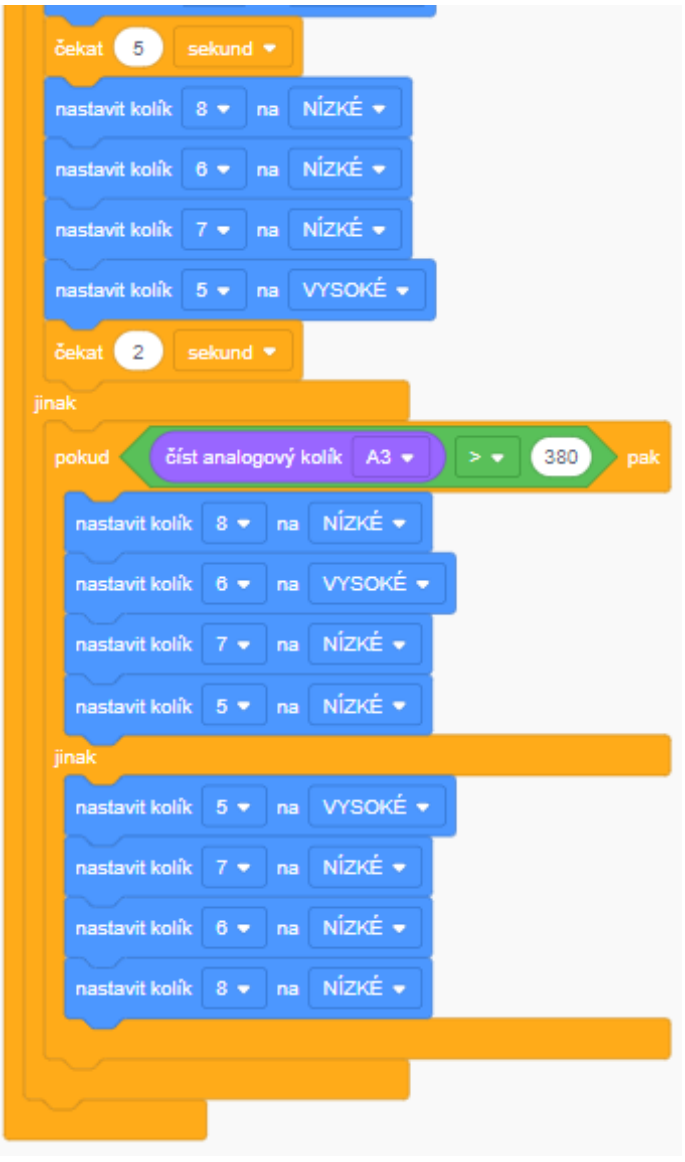

95


95

Příloha 3 – Ukázkový program ve Scratchi – dvoustavové sledování čáry a objetí překážky




čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na NÍZKÉ
čekat 2 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ
čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na NÍZKÉ
čekat 2 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ


čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na NÍZKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ
čekat 2 sekund
jinak
pokud číst analogový kolík A3 > 380 pak
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na NÍZKÉ
jinak
nastavit kolík 5 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 6 na NÍZKÉ
nastavit kolík 8 na NÍZKÉ

Příloha 4 – Ukázkový program v C++ – dvoustavové sledování čáry a objetí překážky

https://drive.google.com/file/d/1g7RRgI2TUjPtmXDaSFNNg4NF3n75qK84/view?usp=drive_link

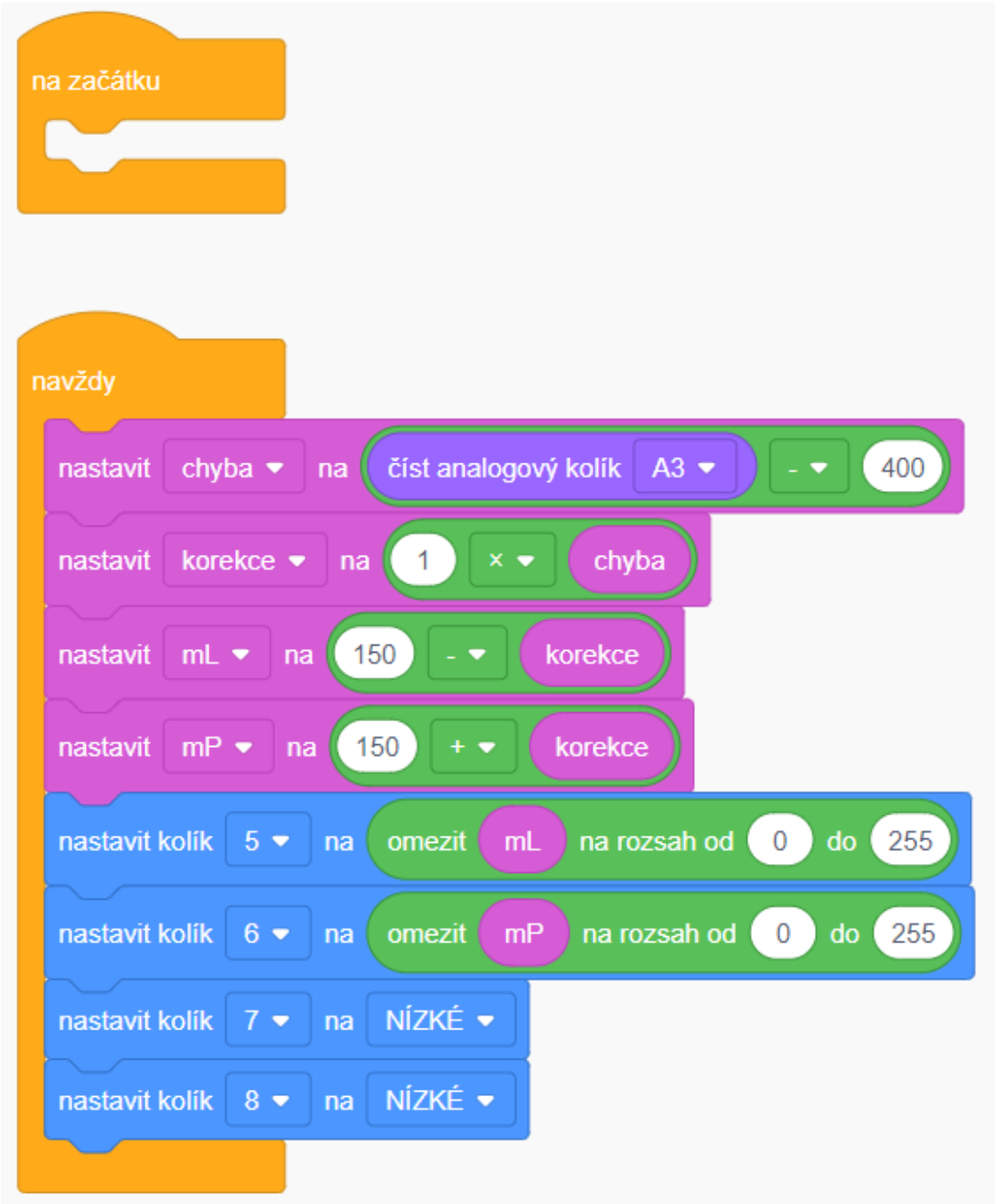
```
int mLv = 5; //Levý motor výkon
int mLs = 7; //Levý motor směr
int mPv = 6; //Pravý motor výkon
int mPs = 8; //Pravý motor směr

//bíla 60 + cerna 600 // idealne 200
//motory
int vL = 150;
int vR = 150;
int vL1 = 130;
int vR1 = 120;

//UZ senzor
int pTrig = 12;
int pEcho = 13;
int vzdalenost;

int odezva;
long ms = millis();

int vzdalenostSenzor() {
  digitalWrite(pTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(pTrig, HIGH);
  delayMicroseconds(5);
  digitalWrite(pTrig, LOW);
  odezva = pulseIn(pEcho, HIGH, 8000);  // maximální délku pulzu v mikrosekundách (us)
  vzdalenost = int(odezva / 58.31);    // přepočet na cm
}
//probehne jednou na začátku
void setup()
{
    Serial.begin(9600);

  pinMode(mLv, OUTPUT);
  pinMode(mLs, OUTPUT);
  pinMode(mPv, OUTPUT);
  pinMode(mPs, OUTPUT);

  pinMode(pTrig, OUTPUT);
  pinMode(pEcho, INPUT);
}

//bude probíhat ve smyčce
void loop()
```

```
{
   vzdalenostSenzor();


 //pokud není v dohledu překážka
 if ((vzdalenost == 0) or (vzdalenost > 20))
 {
  //sleduj čáru
  //Serial.println(analogRead(A3));
  analogRead(A3);
  while (analogRead(A3) < 120)
  {
     analogWrite(mPv, 140);
     digitalWrite(mPs, LOW);
     analogWrite(mLv, 255);
     digitalWrite(mLs, HIGH);
  }
  while ((analogRead(A3)) >= 120)
  {
     analogWrite(mLv, 150);
     digitalWrite(mLs, LOW);
     analogWrite(mPv, 255);
     digitalWrite(mPs, HIGH);
  }
 }
 //jinak - pokud vidíš překážku
 else
 {
  //OBJÍŽDĚNÍ PŘEKÁŽKY
  //doleva
  digitalWrite(mPs, LOW);
  analogWrite(mPv, vR1);
  digitalWrite(mLs, HIGH);
  analogWrite(mLv, 255 - vL1);
  delay(270);
  //stop
  digitalWrite(mPs, LOW);
  analogWrite(mPv, 0);
  digitalWrite(mLs, LOW);
  analogWrite(mLv, 0);
  delay(300);
  //rovně
  digitalWrite(mPs, LOW);
  analogWrite(mPv, vR1);
  digitalWrite(mLs, LOW);
  analogWrite(mLv, vL1);
  delay(550);
  //stop
  digitalWrite(mPs, LOW);
  analogWrite(mPv, 0);
  digitalWrite(mLs, LOW);
  analogWrite(mLv, 0);
  delay(300);
```

```
// Otočení doprava zhruba o 45 stupnu
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255-vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(220);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(1300);
// Otočení doprava zhruba o 45 stupňů
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255-vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(210);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
//jede tak dlouho rovně, dokud čidlem nenajede na černou (dokud je na bílé)
while (analogRead(A3) < 120)
{
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
}
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
```

```
    //doleva
    digitalWrite(mPs, LOW);
    analogWrite(mPv, vR1);
    digitalWrite(mLs, HIGH);
    analogWrite(mLv, 255 - vL1);
    delay(270);
  }
}
```

Příloha 5 – Ukázkový program ve Scratchi – P-regulace



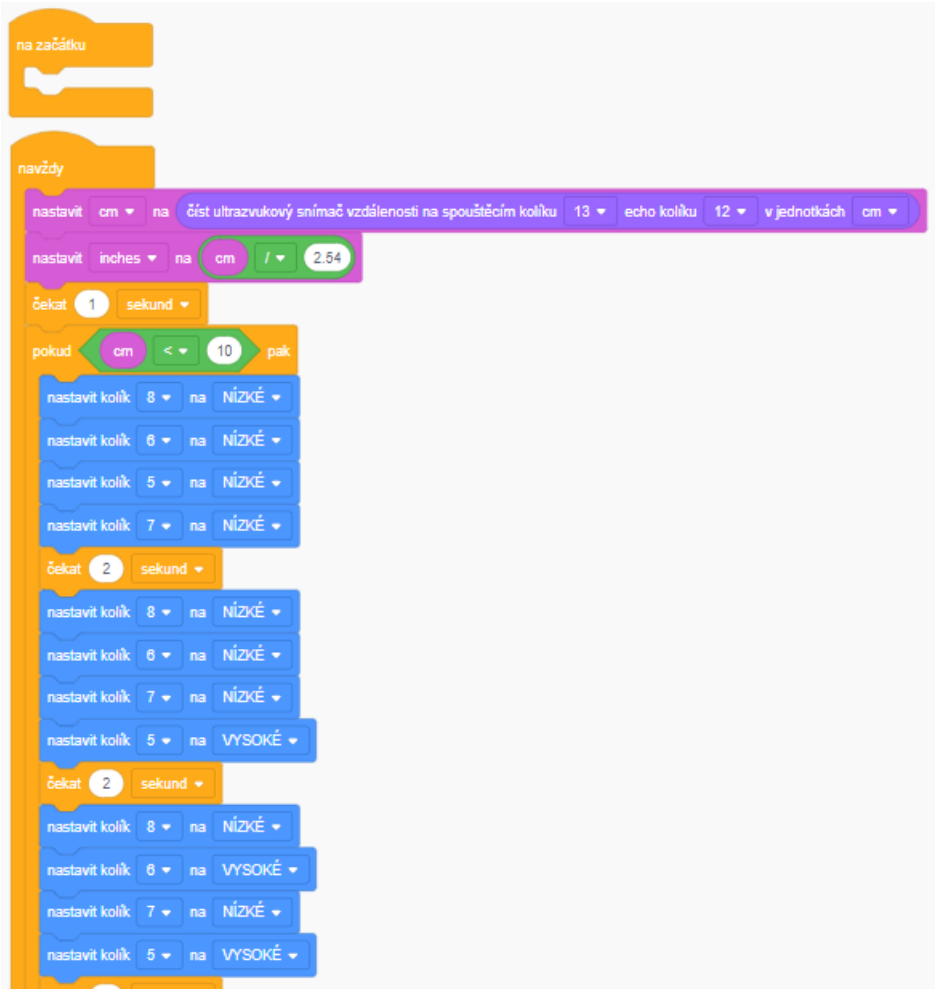
Příloha 6 – Ukázkový program v C++ – P-regulace

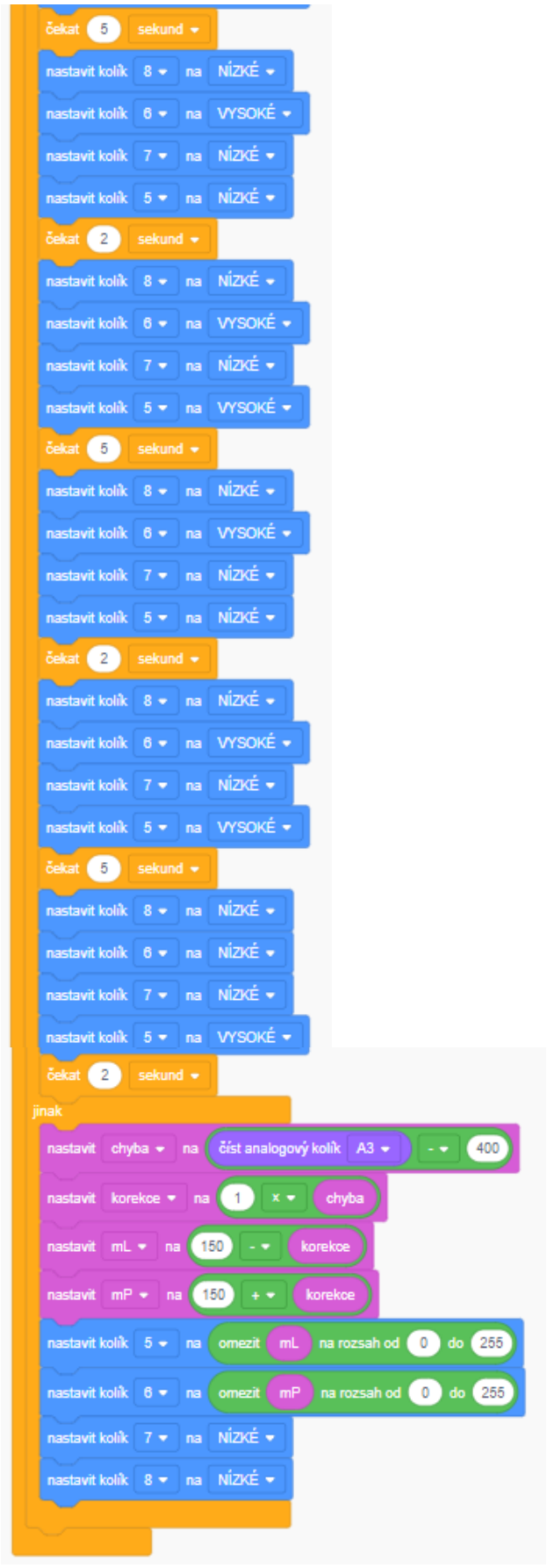
https://drive.google.com/file/d/1Wpphl8wmR-_I95VXTg0U7-n9kDg4FLEK/view?usp=drive_link

```
int mLv = 5; //Levý motor výkon
int mLs = 7; //Levý motor směr
int mPv = 6; //Pravý motor výkon
int mPs = 8; //Pravý motor směr

//deklarace proměnných pro P-regulaci - pro jízdu po čáře
float chyba;
int korekce;
int stredni = 400; //bíla 60 + černá 600 // ideálně 200
float P = 0.45;
int vL = 140;
int vR = 105;
int mL;
int mP;

void setup()
{
  Serial.begin(9600);
  pinMode(mLv, OUTPUT);
  pinMode(mLs, OUTPUT);
  pinMode(mPv, OUTPUT);
  pinMode(mPs, OUTPUT);
}
//bude probíhat stále dokola - ve smyčce
void loop()
{
    //P-regulace
    chyba = (analogRead(A3) - stredni);
    korekce = int(P * chyba);
    mL = vL + korekce;
    mP = vR - korekce;
    //ohraničení hodnot proti přetečení
    if (mL > vL)
    {
      mL = vL;
    }
    if (mL < 0)
    {
      mL = 0;
    }
    if (mP > vR)
    {
      mP = vR;
    }
    if (mP < 0)
    {
      mP = 0;
    }
```

```
        //řízení motorů
        analogWrite(mLv, mL);
        digitalWrite(mLs, LOW);
        analogWrite(mPv, mP);
        digitalWrite(mPs, LOW);
}
```

Příloha 7 – Ukázkový program ve Scratchi – P-regulace a objíždění překážky




čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na NÍZKÉ
čekat 2 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ
čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na NÍZKÉ
čekat 2 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na VYSOKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ
čekat 5 sekund
nastavit kolík 8 na NÍZKÉ
nastavit kolík 6 na NÍZKÉ
nastavit kolík 7 na NÍZKÉ
nastavit kolík 5 na VYSOKÉ
čekat 2 sekund
jinak
nastavit chyba na číst analogový kolík A3 - 400
nastavit korekce na 1 x chyba
nastavit mL na 150 - korekce
nastavit mP na 150 + korekce
nastavit kolík 5 na omezit mL na rozsah od 0 do 255
nastavit kolík 6 na omezit mP na rozsah od 0 do 255
nastavit kolík 7 na NÍZKÉ
nastavit kolík 8 na NÍZKÉ

Příloha 8 – Ukázkový program v C++ – P-regulace a objíždění překážky

https://drive.google.com/file/d/1shBppDGm8WcZCKT-YyGBGOyaInVXjXcS/view?usp=drive_link

```
int mLv = 5; //Levý motor dopředu
int mLs = 7; //Levý motor dozadu
int mPv = 6; //Pravý motor dopředu
int mPs = 8; //Pravý motor dozadu

//deklarace proměnných pro P-regulaci - pro jízdu po čáře
float chyba;
int korekce;
int stredni = 400; //bíla 60 + černá 600 // ideálně 200
float P = 0.45;
int vL = 140;
int vR = 105;
int vL1 = 130;
int vR1 = 120;
int mL;
int mP;

long t0;

//deklarace pro překážku
int pTrig = 12;
int pEcho = 13;

int vzdalenost;
int odezva;

long ms = millis();

//vzdalenostSenzor je funkce pro měření vzdálenosti kterou stačí jen zavolat a ona se změří a potom se jen v cyklu ptáme na vzdalenost
void vzdalenostSenzor()
{
  digitalWrite(pTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(pTrig, HIGH);
  delayMicroseconds(5);
  digitalWrite(pTrig, LOW);
  odezva = pulseIn(pEcho, HIGH, 8000);  // maximální délku pulzu v mikrosekundách (us)
  vzdalenost = int(odezva / 58.31);    // přepočet na cm
  //Serial.print(vzdalenost); //výpis na sériový monitor
}



//proběhne pouze jednou na začátku
```

```
void setup()
{
  Serial.begin(9600);


  pinMode(mLv, OUTPUT);
  pinMode(mLs, OUTPUT);
  pinMode(mPv, OUTPUT);
  pinMode(mPs, OUTPUT);

  pinMode(pTrig, OUTPUT);
  pinMode(pEcho, INPUT);
}
//bude probíhat stále dokola - ve smyčce
void loop()
{
  if ((vzdalenost == 0) or (vzdalenost > 20))
  {
    //P-regulace
    chyba = (analogRead(A3) - stredni);
    korekce = int(P * chyba);
    mL = vL + korekce;
    mP = vR - korekce;

    //ohraničení hodnot proti přetečení
    if (mL > vL)
    {
      mL = vL;
    }
    if (mL < 0)
    {
      mL = 0;
    }
    if (mP > vR)
    {
      mP = vR;
    }
    if (mP < 0)
    {
      mP = 0;
    }

    analogWrite(mLv, mL);
    digitalWrite(mLs, LOW);
    analogWrite(mPv, mP);
    digitalWrite(mPs, LOW);
  }

else
{
```

```
//OBJÍŽDĚNÍ PŘEKÁŽKY
//doleva
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, HIGH);
analogWrite(mLv, 255 - vL1);
delay(270);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(550);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
// Otočení doprava zhruba o 45 stupnu
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255 - vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(220);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
analogWrite(mLv, 0);
delay(300);
//rovně
digitalWrite(mPs, LOW);
analogWrite(mPv, vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(1300);
// Otočení doprava zhruba o 45 stupnu
digitalWrite(mPs, HIGH);
analogWrite(mPv, 255 - vR1);
digitalWrite(mLs, LOW);
analogWrite(mLv, vL1);
delay(210);
//stop
digitalWrite(mPs, LOW);
analogWrite(mPv, 0);
digitalWrite(mLs, LOW);
```

```
            analogWrite(mLv, 0);
            delay(300);
            //rovně
            //jede tak dlouho rovně, dokud čidelem nenajede na černou (dokud je na bílé)
            while (analogRead(A3) < 120)
            {
              digitalWrite(mPs, LOW);
              analogWrite(mPv, vR1);
              digitalWrite(mLs, LOW);
              analogWrite(mLv, vL1);
            }
            //stop
            digitalWrite(mPs, LOW);
            analogWrite(mPv, 0);
            digitalWrite(mLs, LOW);
            analogWrite(mLv, 0);
            delay(300);
            //doleva
            digitalWrite(mPs, LOW);
            analogWrite(mPv, vR1);
            digitalWrite(mLs, HIGH);
            analogWrite(mLv, 255 - vL1);
            delay(270);
          }
}
```