Škola pre mimoriadne nadané deti a gymnázium v Bratislave

Mars Rover

Mapovanie prostredia pomocou technológie LIDAR

šk. rok 2019 / 2020 Samuel Dostál**Čestné vyhlásenie**

Vyhlasujem na svoju česť, že atestačnú prácu na tému *Mars Rover, Mapovanie prostredia pomocou technológie LIDAR* som vypracoval samostatne, a že som uviedol všetku použitú odbornú literatúru.

V Bratislave, dňa 30. 3. 2020 ………………………..

podpis

# Obsah

[**Obsah**](#_n29brtaylaap) **2**

[**Úvod**](#_rlocv1uqmimg) **4**

[**Hardvér**](#_ezkgip41k7ha) **5**

[Model](#_6wxosczhuqtt) 6

[Arduino](#_uyxzmc862qlr) 7

[Meranie vzdialenosti](#_9ozbupj1c7vk) 9

[Ultrazvukový senzor](#_9czpy6inocho) 9

[Infračervený senzor](#_241ixonsfj2o) 13

[LIDAR Senzor](#_9fkfig3k7pwc) 14

[Krokové motory](#_ovxelc2llowz) 17

[Motor driver A4988](#_jnfglbsxn2so) 19

[Bluetooth modul HC-05](#_vzje6ujd1e14) 20

[Gyro Senzor L3G4200](#_3fvpe3ykuz7g) 21

[3D Tlač](#_3dg3db80iodl) 22

[**Softvér**](#_3undfytnl6bf) **24**

[Arduino](#_ckvfwpd1jmkk) 24

[Radar](#_fim3zvq2tjml) 24

[Pohyb robota](#_62bhxntx6pjq) 27

[Pohyb + Radar](#_odrh383r8k8x) 30

[Server](#_ou8b8mq2st10) 30

[Komunikácia s Arduinom](#_ovojbktqt4bz) 30

[Komunikácia s klientom](#_2olw5e9pm002) 32

[Klient](#_s1ytr25s4169) 34

[**Výsledky**](#_4oyzx3ukzqvu) **35**

[**Záver**](#_6d0sglu6bmnn) **37**

[**Zhrnutie**](#_jaj711e29br) **38**

[**Resume**](#_rv7xi4oyoqpm) **38**

[**Referencie obrázkov**](#_cs50yhgk2x00) **39**

[**Zoznam použitej literatúry**](#_uf15a6ous8nc) **41**

[Knižné publikácie](#_21pfeydtgpk7) 41

[Elektronické zdroje](#_cf0y4zsih1cd) 41

# Úvod

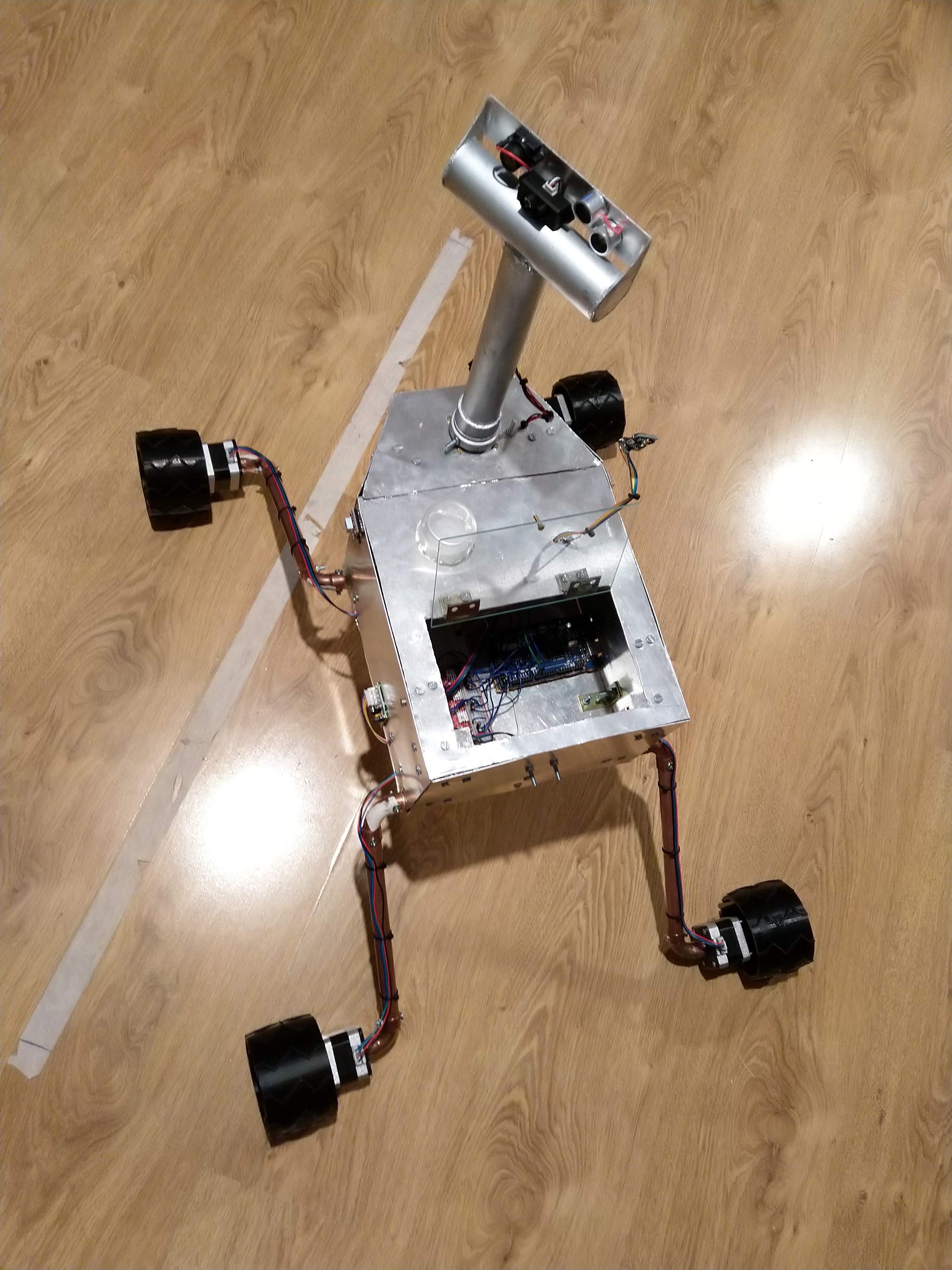
Mapovanie prostredia je problém, ktorý trápi vedcov roky. Uplatnenie môžeme vidieť napríklad pri autonómnych autách, ktoré využívajú senzory s technológiou Lidar. Táto technológia nie je žiadna novinka, nakoľko sa využívala už pri lete na mesiac v roku 1971. Široké uplatnenie, sa ale začalo objavovať iba pár rokov dozadu. Od merania výšky terénu, po presné zisťovanie vzdialeností medzi dvoma budovami.

V tejto práci sa ale pozrieme na to, ako sa dá pomocou tejto technológie mapovať prostredie. Vyrobíme si robota, ktorý má za úlohu vytvoriť mapu miestnosti v ktorej sa nachádza. Mapa bude vytvorená bez akýchkoľvek softvérov, ktoré mapu miestnosti pomocou umelej inteligencie vytvoria za nás. Použijeme iba jednoduchú matematiku, jazyk Arduino a Javascript. Cieľ práce je od základov vytvoriť systém, ktorý bude vedieť pomocou pár informácii o prostredí vytvárať mapy. Oddelene od hardvéru. Robot má podobu rovera Opportunity, ktorý sa nachádza na povrchu marsu, kam bol vyslaný v roku 2003. Keď som vymýšlal tému tejto práce, rover opportunity sa definitívne pokazil. Preto mi prišlo symbolické zvoliť taký vzhľad môjho robota.

Prácu som rozdelil na 2 veľké celky. V prvom celku sa pozrieme na to, ako som robota vyrábal, inak povedané hardvér. V druhom celku, sa naopak pozrieme ako je robot a všetko okolo naprogramované, softvér.

# Hardvér

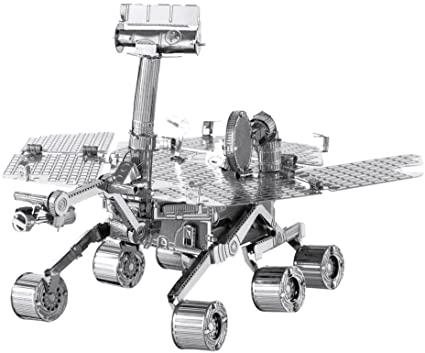
V tejto časti si ukážeme, ako som robota vyrábal. Od mikroprocesora ktorý mi počítal vzdialenosť prekážky, k bezdrôtovému pripojeniu robota k počítaču, až ku všetkým senzorom ktorými robot disponuje. Táto časť sa nachádza pred softvérovou, lebo si myslím že je lepšie najprv si pozrieť ako sa robot vyrábal, až potom ako som ho programoval. Čo je ale dôležité si uvedomiť, je, že takto robot nevznikal. Nebolo to tak, že by som najprv spravil celý hardvér, až potom som to programoval. Tieto dve časti sa vyvíjali a rozvíjali spolu. Vznikali rôzne modely, ktoré som pri finálnom robotovi ani nepoužil. Taktiež som si skúšal rôzne technológie, komponenty, ktoré sa na prvý pohľad javili ako perfektné riešenie pre môj problém, ale musel od nich nakoniec upustiť.



*Obr. č. 1 - Dokončený rover*

## Model

Prvá vec ktorá mi napadla, bola, že si musím spraviť nejaký model. Najprv v zmenšenej verzii, potom v reálnej, až potom funkčný rover. Zmenšený model som kúpil cez Amazon. Pozostával z plechových častí ktoré som musel spolu pozliepať. Vyzeralo to takto:



*Obr. č. 2 - Zmenšený model opportunity rovera*

Hneď som zistil, že to bol skvelý nápad. Pomocou pravítka, som si vedel zmerať parametre ako napríklad výška hlavy, rozmer kolies, atď. Tieto rozmery som potom aplikoval na model v reálnej veľkosti. Model som vyrábal z kartónu. Myslel som si, že dokončený robot bude presná replika toho kartónového. Až neskôr som zistil, že to nebude úplne tak. Napríklad sa mi nepodarili dať robotovi 6 kolies, ale len 4. Každopádne som si kvôli tomuto modelu uvedomil zopár vecí. Napríklad, že konečný výsledok bude pomerne veľký, na druhú stranu má rover pomerne malé kolesá.

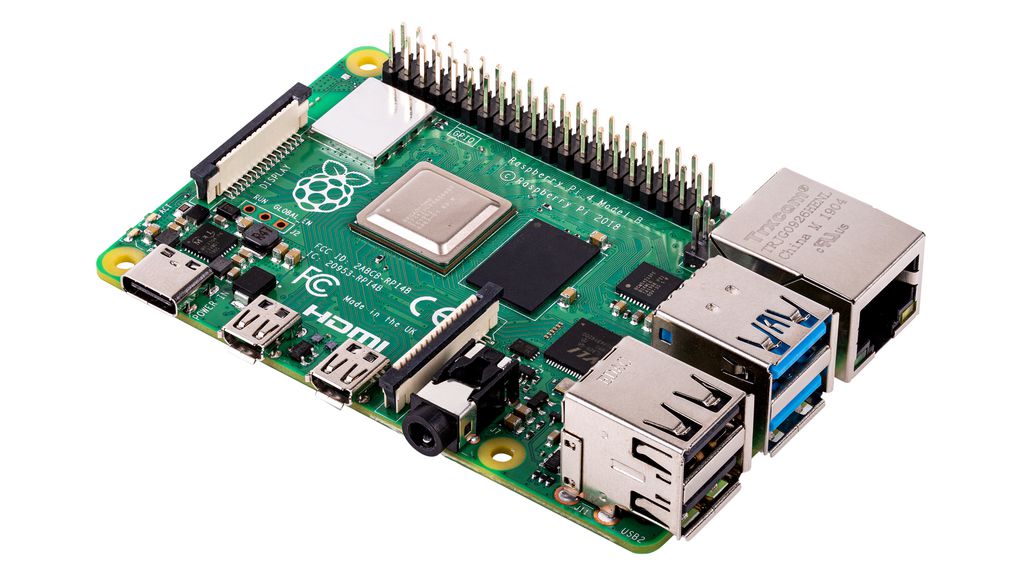


*Obr. č. 3 - Kartónový model v reálnej veľkosti*

## Arduino

Prišiel čas na rozhodovanie, aký počítač sa bude v robotovi nachádzať. Mal som dva nápady. Prvý, Raspberry Pi. Tento počítač je známy nielen svojou kompaktnosťou, ale aj výkonom. Taktiež som s ním už mal nejaké skúsenosti a jedným som disponoval. Ďaľšia obrovská výhoda je, že v sebe má zabudovanú wifi. Na druhú stranu, je to možno až trochu veľa výkonu. Na tento počítač sa dá dokonca nainštalovať aj upravená verzia Windows 10.

Arduino, je známe taktiež svojou kompaktnosťou, cenou a podporou na internete. Pre mňa ale najväčšia výhoda je, že sa do neho kompiluje program cez Assembler. To znamená, že Arduino pracuje na úplne najnižšom leveli čo sa týka programovania. Program sa do neho nahráva vo formáte núl a jedničiek. Z týchto dôvodov som si vybral ako počítač dosku Arduino. Presnejšie model Arduino Mega. Tento model som vybral z dôvodu, že najznámejší model Uno pre mňa nemal dostatočné množstvo pinov a nedisponoval viacerými TX-RX pinami[[1]](#footnote-0).



*Obr. č. 4 - Raspberry Pi model 4*



*Obr. č. 5 - Arduino Mega*

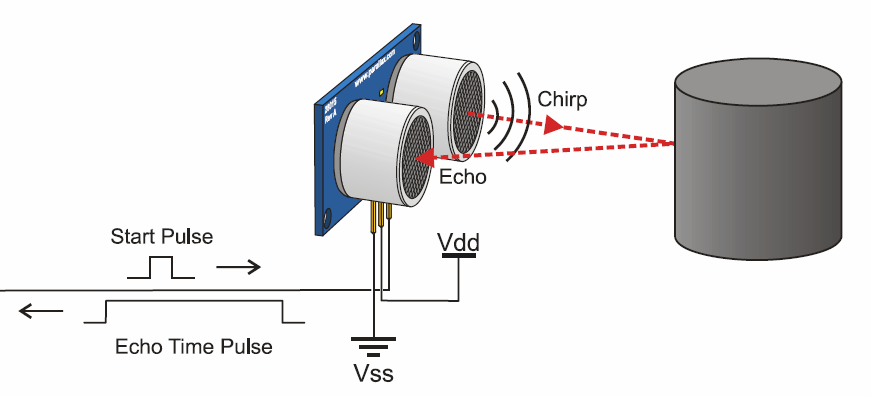
### Meranie vzdialenosti

Čo sa týka práce s Arduinom, poznáme 3 známe technológie/senzory, ktoré sa využívajú na zisťovanie vzdialenosti. Samozrejme že majú veľa rozdielov, slabín, ktoré rozhodujú ktorý senzor by sme mali pri akom projekte použiť. Podstatné ale je, že všetky majú za úlohu rovnakú vec: odmerať vzdialenosť. Ďalšia vec ktorú majú všetky senzory spoločné, je jednoduchosť ovládania. Všetky využívajú 4 piny:

1. Vcc(5V / 3.3V, väčšinov 5V)
2. Gnd (Uzemnenie)
3. Trig (Na vyvolanie signálu)[[2]](#footnote-1)
4. Echo (Na príjem signálu)

### Ultrazvukový senzor

Tento typ senzora je najpoužívanejší, najmä kvôli cene a spoľahlivosti. Funguje na princípe počítania času, za ktorý prejde zvuková vlna k najbližšej prekážke a vráti sa odrazená späť.



*Obr. č. 6 - Senzor vyšle zvukovú vlnu a meria čas, kým sa odrazí späť.*



*Obr. č. 7 - Senzor HC - SRO4*

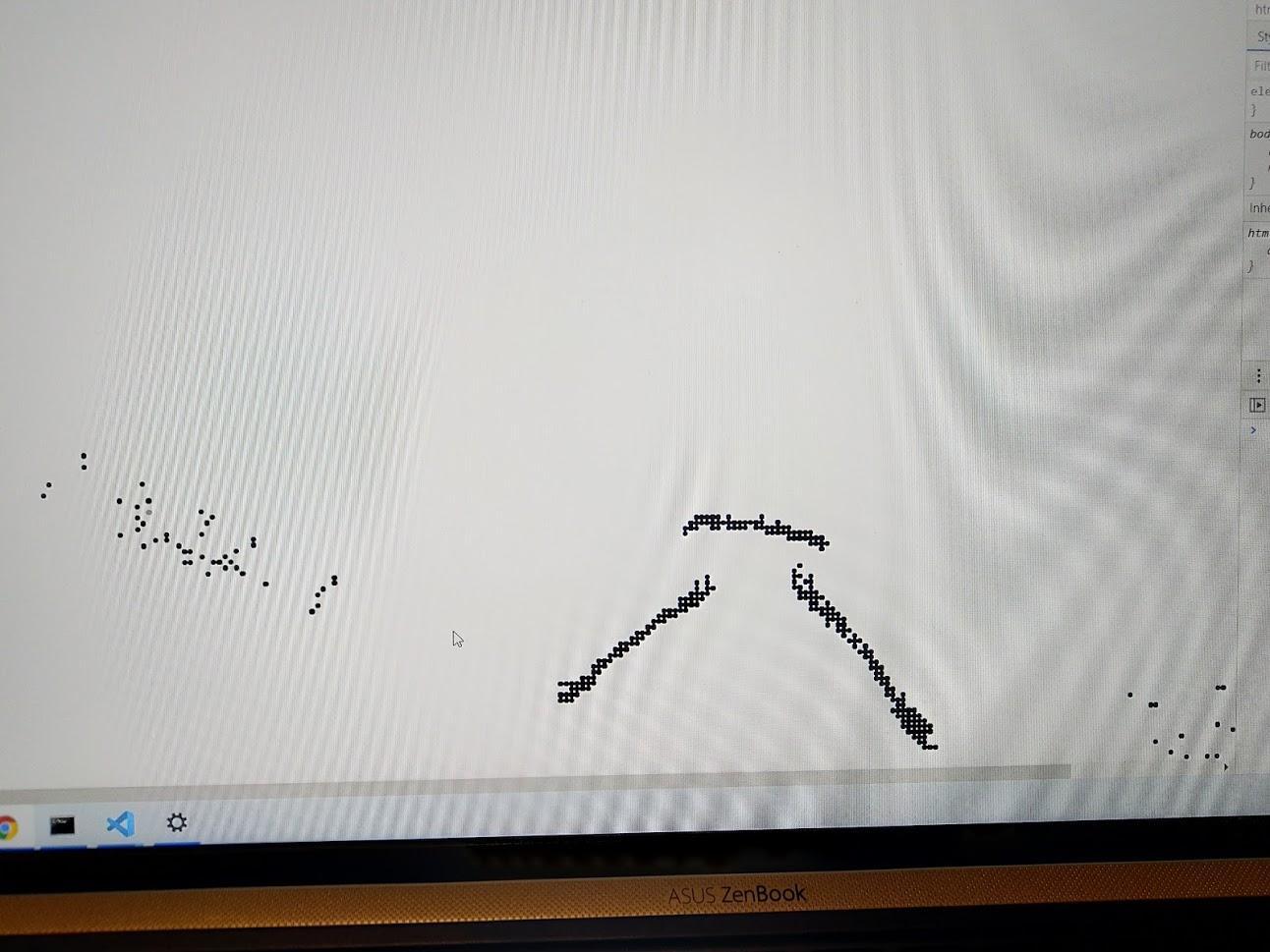
Pracuje to nasledujúcim spôsobom. Pomocou arduina zapneme Echo pin aby počúval na signál. Následne pustíme signál do Trig pinu. Hneď ako ultrazvukový senzor spracuje požiadavku, vyšle zvukovú vlnu. Tá putuje vzduchom, odrazí sa od najbližšej prekážky, vráti sa späť, kde ju zachytí prijímač. Ten dá znamenie Echo pinu, aby sa zapol. Takto sa dostane signál späť do Arduina. Jediné čo teraz musíme v Arduine spraviť, je spočítať čas ktorý uplynie medzi vyslaním signálu do Trig pinu a príjmom signálu cez Echo pin. Keďže vieme že zvuk sa vo vzduchu pohybuje rýchlosťou približne 340m/s, dokážeme vypočítať vzdialenosť prekážky.

Ukážka v Arduino jazyku:

|  |
| --- |
| void loop() {  // Zapneme trigPin na 10 microsekúnd  digitalWrite(trigPin, HIGH);  delayMicroseconds(10);  digitalWrite(trigPin, LOW);  // Signál príde späť to echoPin  duration = pulseIn(echoPin, HIGH);  // Počítanie vzdialenosti v závislosti od času  distance = duration \* 0.034/2;  } |

Veľká výhoda je, že senzor funguje takmer v každom prostredí. Keďže sa narozdiel od nasledujúcich senzorov využíva zvuková vlna, ktorej nebránia zlé svetelné ani hmlisté podmienky. Ďalšia výhoda sú relatívne presné výsledky.[[3]](#footnote-2) Tie nám zabezpečuje pomerne pomalá rýchlosť zvuku, ktorá sa dá odmerať výrazne lepšie ako napríklad rýchlosť svetla.

Prvý plán bolo použiť senzor na radar robota. To sa neskôr ukázalo ako nie najlepší nápad. Zistil som, že senzor má jednu slabinu a to pri meraní rohov miestnosti. Keď som si zo senzoru spravil radar [[4]](#footnote-3), vykreslovalo mi to takúto mapu:



*Obr. č. 8 - Problém s vykresľovaním rohov miestností*

Čo sa tu deje? Dve čiary kolmé na seba, boli dve steny krabice v ktorej som senzor testoval. Tieto steny vykreslil veľmi dobre. Problém nastáva keď sa dostaneme k rohu. Zvuková vlna ktorá sa vyšle smerom k rohu, sa odrazí od jednej steny na druhú, až potom späť k radaru. To ale Arduino nevie, má len informácie o tom, že vzdialenosť je tam väčšia (Tak to aj vykreslil).

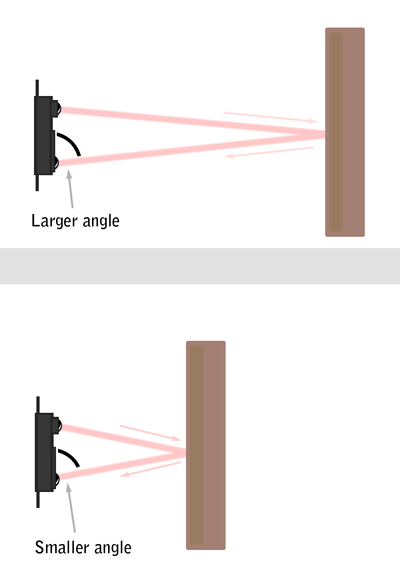
Preto som sa rozhodol senzor nepoužiť na radar. Nakoniec som ho ale na robota pridal. Nachádza sa v predu a slúži ako poistka proti nárazom do prekážky.



*Obr. č. 9 - Prvá verzia radaru - použitý ultrasonický senzor*

### Infračervený senzor

Senzor funguje na princípe merania uhla, ktorý zviera lúč svetla s senzorom:



*Obr. č. 10 - Čím väčšia vzdialenosť prekážky, tým sa vnútorný uhol blíži bližšie k 90°*

Na rozdiel od ultrazvukového senzora, vyšleme lúč svetla. Keďže sa rýchlosť svetla meria výrazne ťažšie ako rýchlosť zvuku[[5]](#footnote-4), senzor meria uhol, vyznačený na obrázku. Tento uhol sa mení v závislosti od vzdialenosti objektu.

Senzor má viacero nevýhod. Prvá je, že dobre nefunguje v zlých svetelných podmienkach. Keď priamo na senzor svieti slnko, intenzita lasera sa výrazne znižuje. Výsledok je, že senzor má problém so zachytením svetelného signálu, ktorý poslal. Najväčšia nevýhoda je maximálna vzdialenosť merania. Zatiaľ čo ultrasonický senzor má maximálnu merateľnú vzdialenosť vyše 8 metrov, Infračervený senzor ktorý som objednal mal merateľnú vzdialenosť do 1.2 metra. Z tohto dôvodu som senzor na robota ani nenamontoval.

### LIDAR Senzor

Lidar[[6]](#footnote-5) patrí medzi najdrahšie alternatívy. Na druhej strane má veľké využitie. Okrem autonómnych áut, ktoré pomocou neho vedia nakresliť celú mapu okolia, túto technológiu využíva napríklad aj najnovší iPad firmy Apple, ktorý pomocou neho zobrazuje virtuálne predmety v reálnom svete[[7]](#footnote-6). Mňa ale najviac zaujalo, že senzor využívajú rovere, ktoré boli vyslané na Mars.

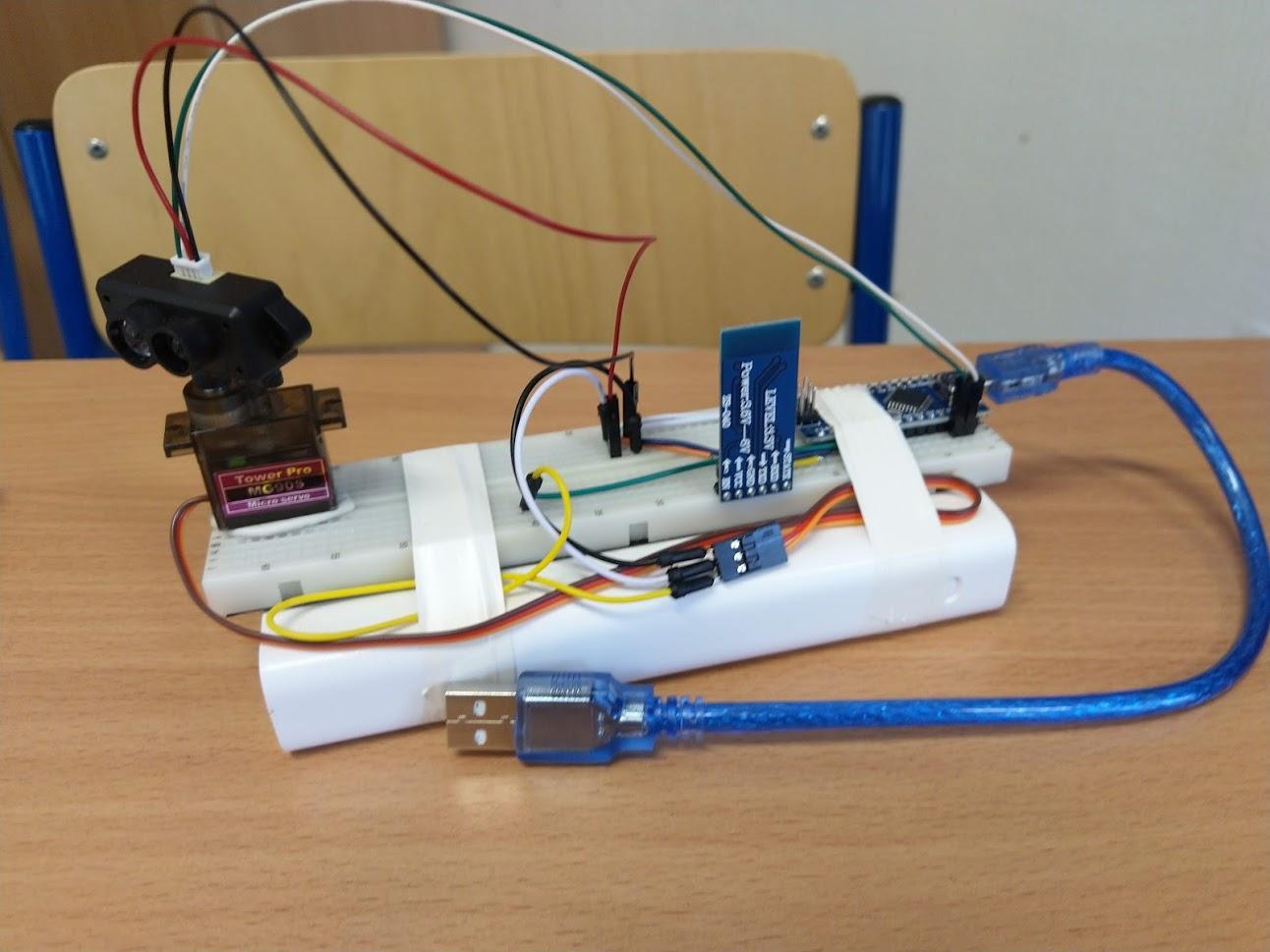
Funguje z časti na princípe ako infračervený senzor, z časti ako senzor ultrazvukový. Tiež pracuje s laserom. Narozdiel od infračerveného, kde meria uhol odrazu, LIDAR senzor meria rýchlosť, za ktorú sa svetlo dostane k prekážke a odrazí sa späť. Toto je veľmi náročná operácia, ktorú každý čip nezvládne. To spôsobuje aj cenu. Ja som objednal tieto senzory 3. Prvé dva TFMini Micro LIDAR stáli 40€ za kus, tretí TFMini Plus LIDAR 50€. Na robotovi sa nachádza TFMini Plus, ktorý ma dosah vzdialenosti až 12m a dáva presnejšie výsledky.



*Obr. č. 11 - TFMini Plus Lidar*



*Obr. č. 12 - TFMini Micro Lidar*

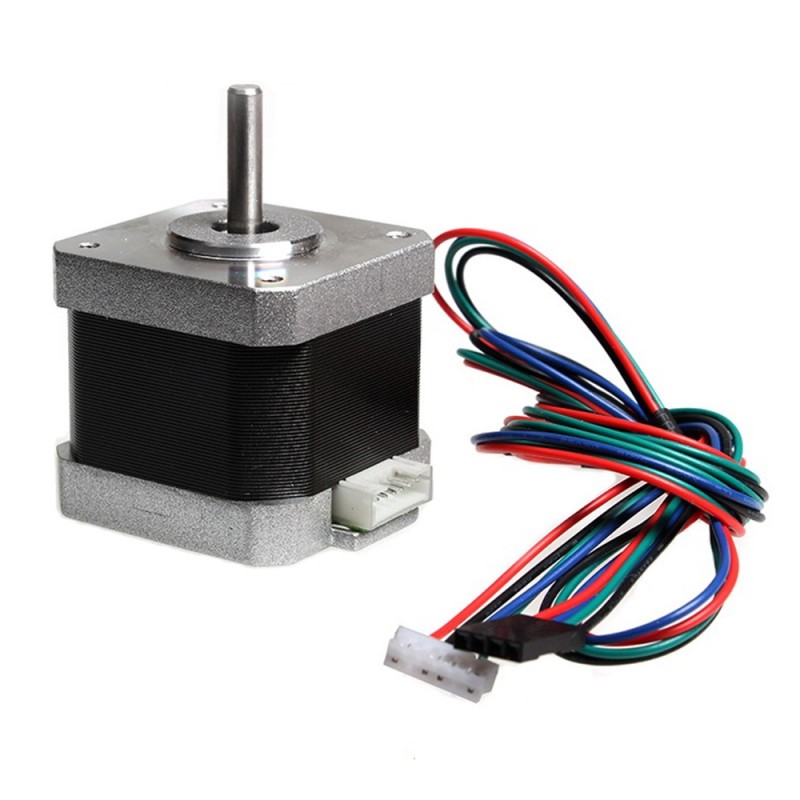


*Obr. č. 13 - Prvý radar s použitím LIDAR senzoru*

### Krokové motory

Pri voľbe motorov som nemal veľmi na výber. Keďže aj robot ktorý sa nachádza na Marse prejde za jeden deň prejde len pár metrov, nepotreboval som, aby môj rover išiel vysokou rýchlosťou. Išlo mi hlavne o to, aby bol jeho pohyb maximálne precízny. Jediný spôsob akým budem zaznamenávať o koľko sa posunul napríklad dopredu, je, že si robot bude zaznamenávať o koľko som mu povedal, aby sa pohol. Nemá žiadny senzor ktorý by mu povedal, že sa práve posunul o 4 metre.

Mal som na výber z klasických DC motorov, servo motorov a krokových motorov. DC Motorčeky majú nevýhodu, že nie sú presné, čo sa týka počtu otočiek. Nevedel by som poriadne zaznamenať, ako veľmi sa motor pohol. Servo motory fungujú trochu inak ako DC motory. DC motorom sa pustí prúd a oni sa začnú točiť. Servo motorom treba poslať signál, ktorý im povie, na aký stupeň sa majú posunúť. Často bývajú obmedzené na maximálne 180° rotáciu. Toto sa mi ale podarilo obísť. Použil som 180° servo motor, ktorému keď odstránim potenciometer[[8]](#footnote-7) ktorý mu určuje na akom stupni je, a poviem mu aby sa otočil na 180°, bude sa na požadovanú hodnotu snažiť dostať. Kvôli tomu že nemá potenciometer, sa na hodnotu nikdy nedostane. Stále mal ale tento typ motora dva problémy. Prvý, motor nebol dostatočne presný. Druhý, motor nemá dostatočný krútiaci moment. Z týchto dôvodov som zvolil krokový motor. Tento typ motora sa využíva napríklad v 3D tlačiarňach, kvôli jeho presnosti. Funguje na princípe spínania magnetických obvodov. Záleží od typu, ale motory ktoré som ja použil (Nema 17), sa vedia otočiť o najmenší uhol 1.8°. Takto dokážem robotovi povedať, aby otočil všetky motory o 1325[°](http://www.degreesymbol.co/), a viem, že sa môžem na výsledok spoľahnúť.



*Obr. č. 14 - Krokový motor Nema 17*

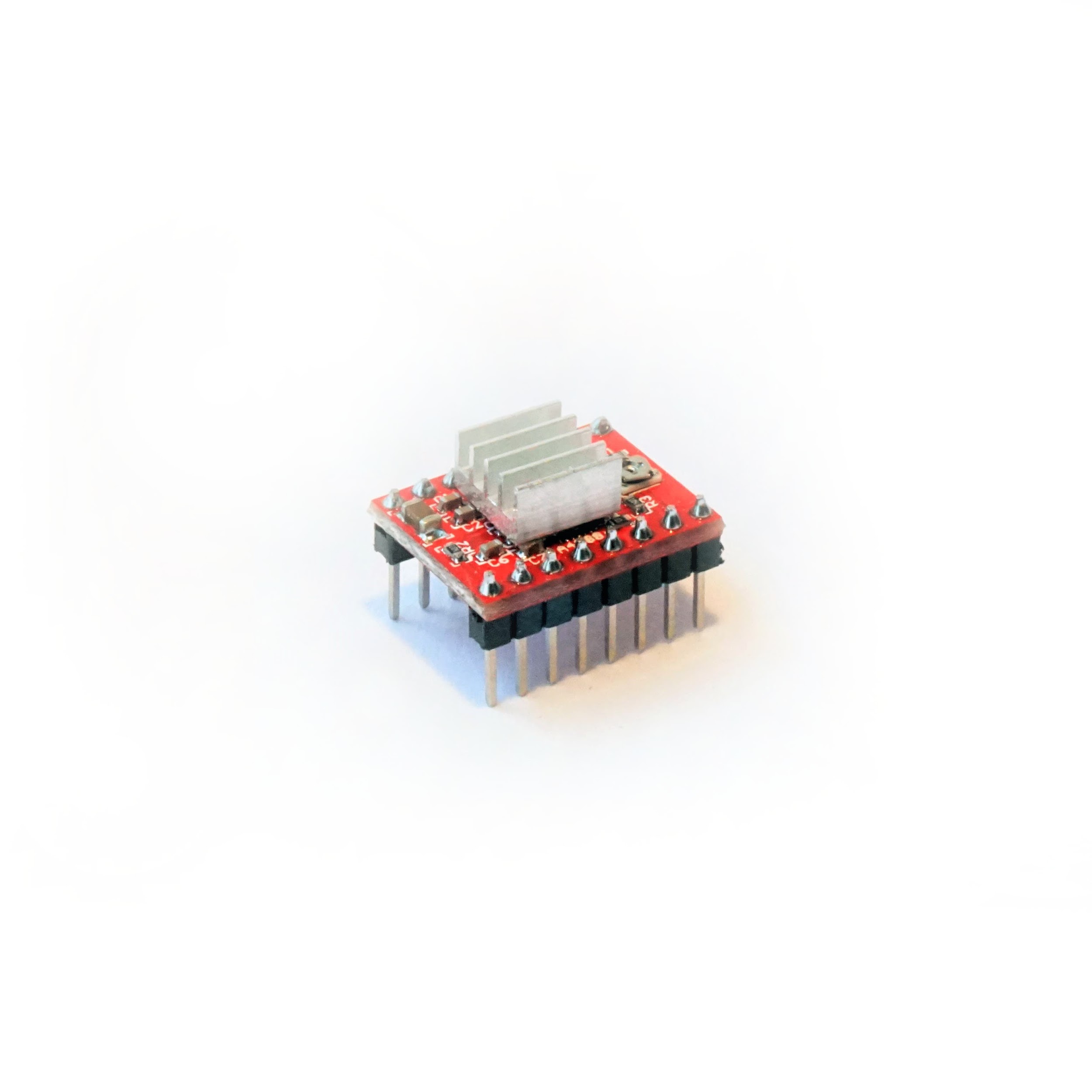


*Obr. č. 15 - Noha robota s krokovým motorom Nema 17 a vytlačeným kolesom*

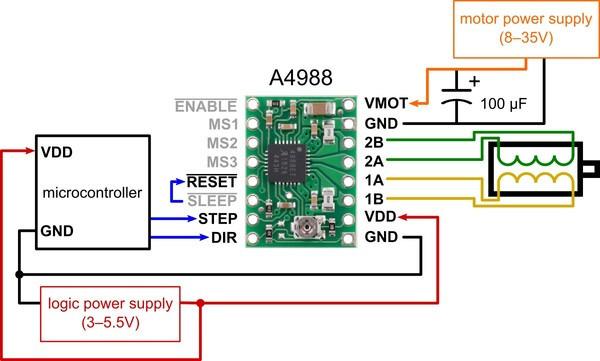
### Motor driver A4988

Krokové motory sa nedajú ovládať priamo cez dosku Arduino. Motor potrebuje napätie 8-35V, ktoré doska Arduino nedokáže vyprodukovať. Taktiež motor vyžaduje prúd 1-2A, pričom Arduino pracuje s maximálne 10mA. Z týchto dôvodov sa využívajú drivere. Driver A4988 ktorý ja používam má 16 pinov, z ktorých využívam 15. Pri tomto driveri si treba dať pozor na prehrievanie. Často sa mi stávalo, že teploty sa vyšplhali až na 80[°](http://www.degreesymbol.co/)C. Pre to som musel na driver nalepiť hliníkový plech, ktorý to spolu s ventilátorom chladili. Keď som ešte tieto opatrenia nemal, vypálil som 9 driverov, jeden mi doslova explodoval a začal horieť. Ten ale mal iný problém ako prehrievanie[[9]](#footnote-8).

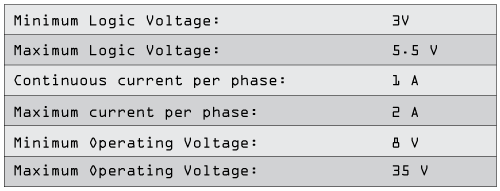
Driver spína magnetické obvody vo vnútri krokového motora. Každým zopnutím sa motor posunie o jeden krok. Keď driver tieto kroky spúšťa v pravidelńych intervaloch, motor sa plynule točí. Jediné inputy ktoré driver dostáva od arduina, sú počet krokov o ktorý sa má otočiť a smer otáčania. Driver som zapojil ako je ukázané na diagrame (Obrázok č. 17).



*Obr. č. 16 - Motor driver A4988*



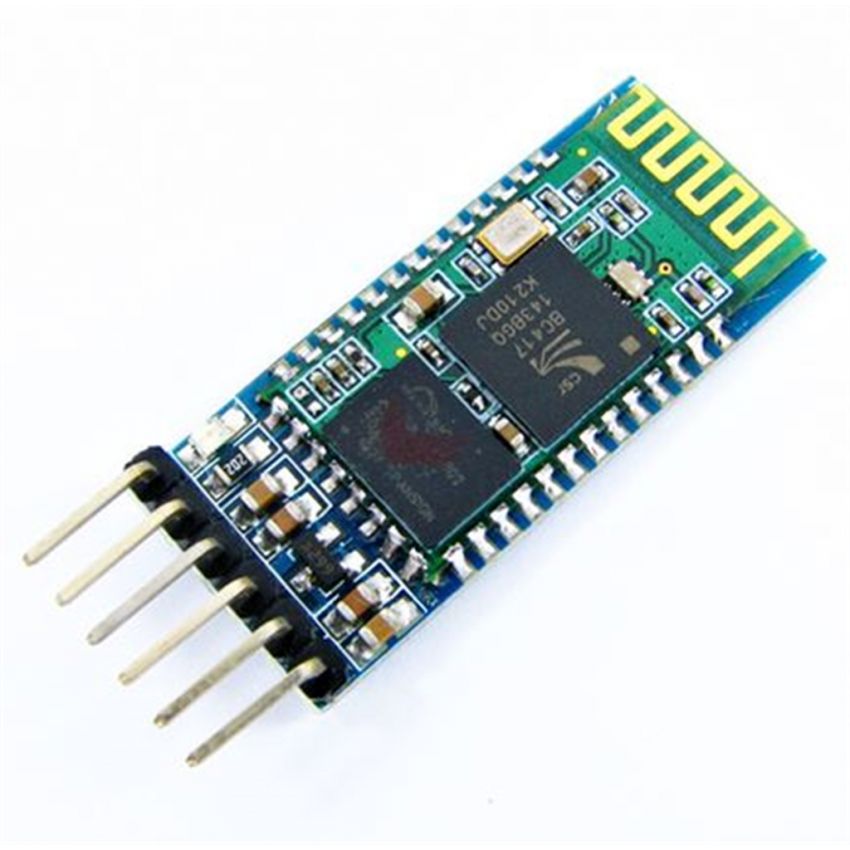
*Obr. č. 17 - Zapojenie krokového motora, drivera, Arduina*



*Obr. č. 18 - Parametre drivera*

### Bluetooth modul HC-05

Bluetooth modul HC-05 som vybral najmä kvôli jeho jednoduchosti. Ovláda sa dvomi pinami: TX a RX. Potom ako modul dostane napájanie (5V), automaticky sa zapne v párovaciom režime. Po spárovaní s mobilom alebo počítačom, je bluetooth komunikacia pripravená. Vieme poslať dáta do bluetooth modulu, ktorý ich následne vyšle cez TX pin, na ktorý je pripojené Arduino. Taktiež keď vyšleme signál z Arduina na RX pin bluetooth modulu, pošle modul signál, na pripojené zariadenie. Rýchlosť príjmu a odosielania dát sa dá nastaviť až na 1382400 bitov za sekundu.

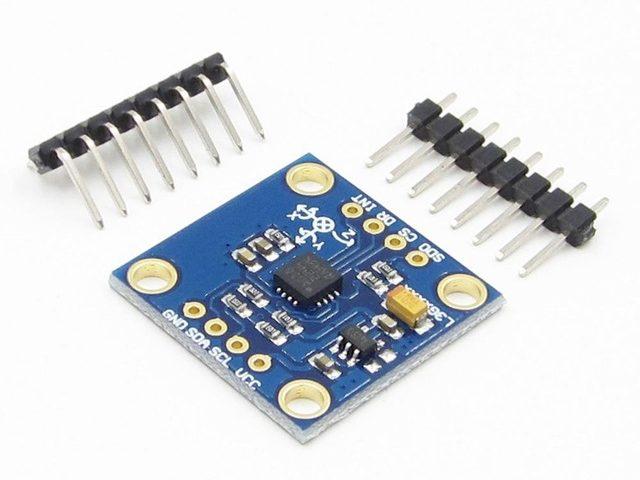


*Obr. č. 19 - Bluetooth modul HC-05*

### Gyro Senzor L3G4200

Na rôznych povrchoch sa robot otáča inak. Na tvrdých povrchoch sa kolesá pretáčajú, na mäkších sa pretáčajú menej. Z tohto dôvodu sa ťažko zisťuje o koľko sa robot otočil okolo svojej osi. Namiesto toho aby som podľa otáčiek kolies počítal uhol, rozhodol som sa využiť Gyroskopický senzor L3G4200. Nachádza sa zhora na robotovi a slúži ako kompas. Jednoducho keď sa senzor začne otáčať, vracia Arduinu údaje o rýchlosti, ktorou sa otáčal. Podľa tejto rýchlosti a počiatočnej pozície, vieme vypočítať, o koľko stupňov sa senzor otočil.

Prvý prístup ktorý som zvolil bolo použitie iného senzora, ktorý fungoval presne ako kompas. Vracial údaje v závislosti od severného póla zeme. Mal ale jeden problém, a to že bol nepresný. Musel sa kalibrovať, ale to sa mi nepodarilo. Keď som skúšal tento gyrosenzor, bol oveľa presnejší[[10]](#footnote-9) a nemusel som ho kalibrovať



*Obr. č. 20 - Gyroskopický senzor L3G4200*

## 3D Tlač

3D tlač je technika vytvárania objektov postupným ukladaním vrstiev materiálu na seba pomocou 3D tlačiarne. Keď som si robota navrhoval, zistil som, že je veľa vecí ktoré by som si potreboval na 3D tlačiarni vytlačiť. Pre to som kúpil 3D tlačiareň Ender 3. Materiál z ktorého som tlačil je PLA, ktorý je veľmi tvrdý a ľahký. Spolu som takto vyrobil vyše 40 súčiastok, od kolies ktoré sa tlačili 14 hodín kus, po milimetrové výplne. V počítači som designoval rôzne verzie súčiastok, následne som ich vytlačil, upravil som modely ktoré nefungovali a tlačil znova. Preto mám doma 30 nevyužitých súčiastok.



*Obr. č. 21 - Nepoužité alebo pokazené súčiastky*

# 

# Softvér

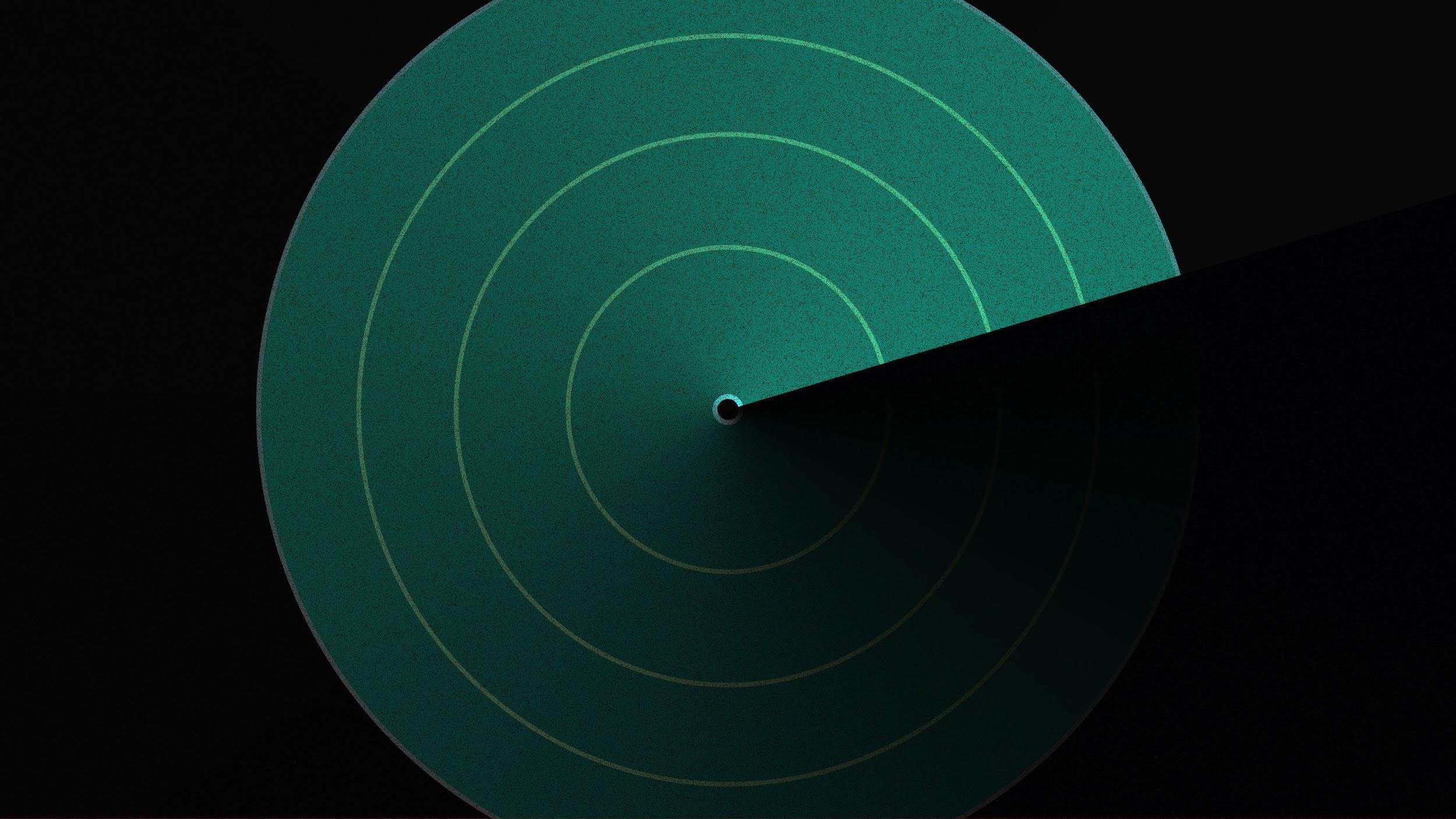
Programovanie robota je len jedna časť problému. Potrebujeme dáta získané z Arduina aj zobraziť. Pre to som sa rozhodol vytvoriť projekty: strana servera a strana klienta. Server sa pripája na Arduino a prijíma od neho dáta. Následne ich preposiela na stranu klienta, ktorá ich vyzobrazuje vo forme mapy.

## Arduino

Teraz, keď sme si vysvetlili všetky hardvérové časti ktoré robot využíva, nám už len stačí pochopiť ako to všetko spolu funguje. Vysvetlil som aké motory, senzory využívam, ale zatiaľ som nepovedal ako to spolu súvisí. Ľahšie to bude pochopiť naraz s programovaním. Táto kapitola je rozdelená na časti, ktoré by som nazval moduly. Každý modul je jedna samostatná časť robota, ktorá sa dá ľahko od zvyšku oddeliť, alebo pridať.

### Radar

Prvý modul je radar. Pozostáva z krokového motora ktorý ho otáča a LIDAR senzoru na meranie vzdialenosti. Radar sa dokáže otočiť až o 360[°](http://www.degreesymbol.co/) a funguje ako letecké radary. Pre každý jeden stupeň z 360 zistí LIDAR vzdialenosť. Súradnice môžeme matematicky zapísať ako [vzdialenosť, uhol] a nazývame ich polárne. Hneď ako Arduino získa nový bod, pošle ho cez bluetooth modul na server, ktorý mi beží na počítači. Tam sa údaj spracuje, ale o tom si povieme v kapitole o serveri.



*Obr. č. 22 - Princíp radaru - polárne súradnice*

Ako ale radar vie, kedy má hľadať nové body? Ak by bol radar aktívny celú dobu keď je robot zapnutý, mali by sme na strane servera nadmieru veľa dát, čím by sa tvorilo aj veľa nepresností. Z tohto dôvodu Arduino dostáva príkazy zo servera[[11]](#footnote-10). Príkazy ktoré súvisia s radarom sú: HEAD\_RIGHT, HEAD\_LEFT, STOP, RESET. Radar je hlava robota, pre to môžete v názvoch vidieť HEAD.

Príkaz HEAD\_RIGHT začne točiť radar v smere hodinových ručičiek, pričom pre každý stupeň zistí vzdialenosť, tým pádom polárnu súradnicu nového bodu, ktorú okamžite pošle na server. HEAD\_LEFT robí presne to isté, iba sa krúti proti smeru hodín. STOP nesúvisí len s radarom, ale s celým robotom. Zastaví akúkoľvek aktivitu, ktorú robot vykonáva. Či je to hýbanie sa, alebo zisťovanie vzdialenosti. Posledný veľmi dôležitý príkaz je RESET. Spraví, že sa robot otočí o 360[°](http://www.degreesymbol.co/) doľava. Spôsobí to, že nech je krokový motor otočený ktorýmkoľvek smerom, dostane sa do začiatočnej polohy. Začiatočná poloha je 90[°](http://www.degreesymbol.co/) voči smeru jazdy robota. To, že sa motor zastaví presne v tejto polohe, spôsobujú dve skrutky ktoré sa o seba zastavia (Vlož obrázok). Krokový motor sa bude snažiť točiť ďalej, ale bude mu v ceste brániť skrutka na robotovi.

Týmto spôsobom sa radar vždy dostane do začiatočnej polohy. Prečo sa to ale musí robiť takto komplikovane? Narozdiel od servo motorov, samotný motor nevie v akej začiatočnej polohe sa nachádza. Preto ho musíme resetovať. Tento reset sa spustí keď príde príkaz zo servera, taktiež vždy pri spustení robota. Keď má radar vyresetovanú pozíciu, je pripravený ju zaznamenávať po každom pohybe.

Tu je ukážka v Arduino jazyku, je to zjednodušená verzia, ak by sme sa ju pokúsili nahrať, tak by uplne nefungovala. Plná, komplexná verzia sa dá pozrieť na: <https://github.com/samodostal/mars-2020>. Aby to nebolo na 3 strany, tu je ukazka len pre HEAD\_RIGHT:

void loop() {

// Ak prichadza signal z bluetooth modulu

if (Serial.available() > 0) {

// Prečítaj signal

SerialState = Serial.readString();

// Ak je signal prikazHEAD\_RIGHT

if (SerialState == "HEAD\_RIGHT") {

// Nastav smer otacania radaru do prava

digitalWrite(dirHead, HIGH);

headDirection = 1;

stopHead = false;

}

}

//...

if (!stopHead) {

// Otacanie motorom

digitalWrite(stepHead, HIGH);

delayMicroseconds(3000);

digitalWrite(stepHead, LOW);

delayMicroseconds(3000);

// Mali by sme pozerat aj smer, presah 360/0 stupnov...

headAngle++;

// Zisti vzdialenost LIDAR

getTFminiData(&distance, &strength);

// Posli polarne suradnice cez bluetooth modul.

Serial.print("[" + distance + ", " + headAngle +"]");

}

}

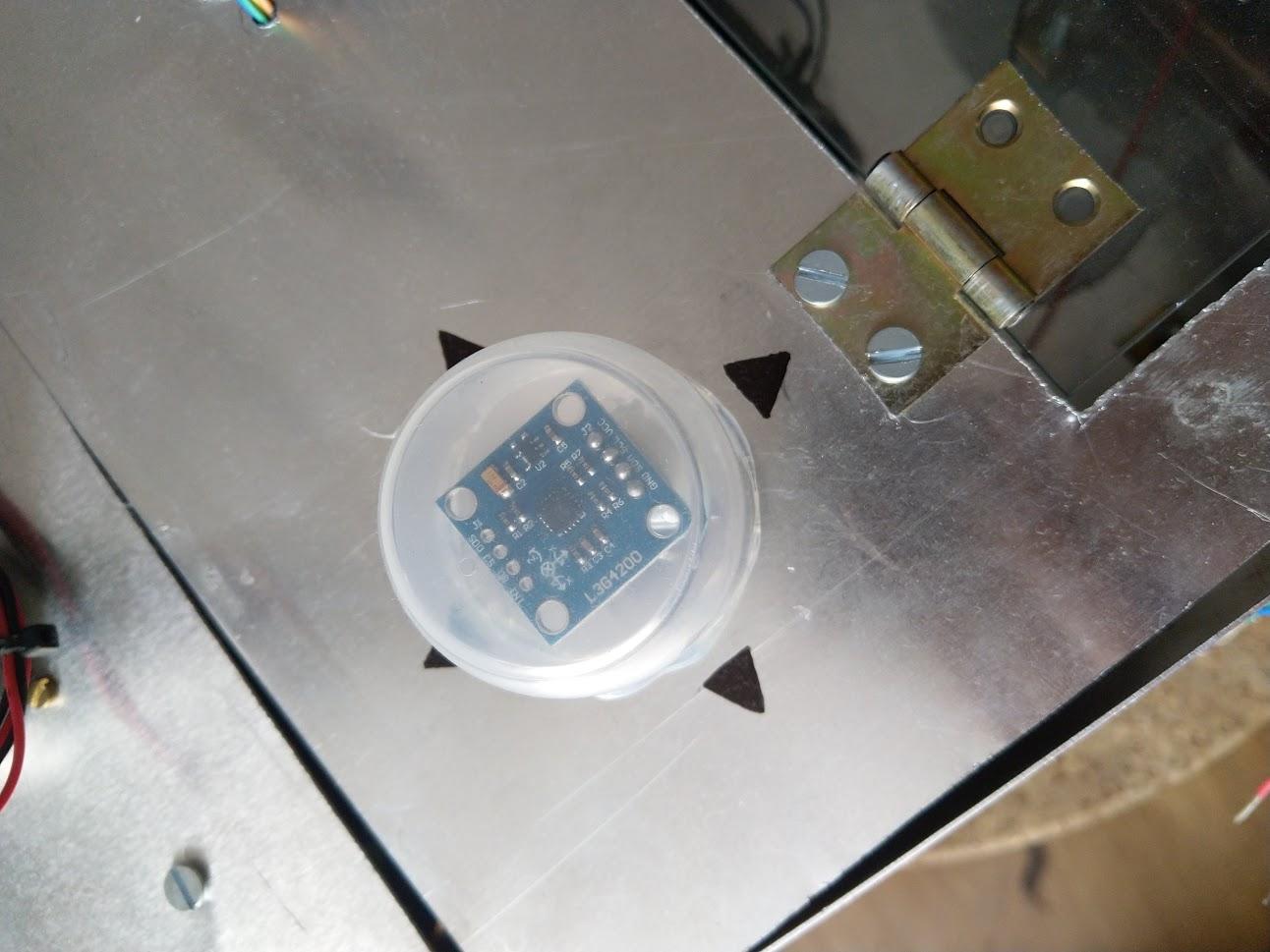
### Pohyb robota

Robot má štyri kolesá, vytlačené na 3D tlačiarni (Obrázok č. 1), ovládané krokovými motormi. Podobne ako pri radare, kolesá chceme ovládať cez príkazy, ktoré chodia zo servera. Príkazy, ktoré súvisia s pohybom robota sú: UP, DOWN, LEFT, RIGHT, STOP. UP jednoducho zapne všetky štyri kolesá, aby sa krútili rovnakým smerom dopredu. DOWN to isté dozadu. LEFT nastaví kolesa na ľavej strane robota naopak, ako kolesá na pravej strane. RIGHT to isté naopak. STOP zastaví akýkoľvek pohyb, ktorý sa práve vykonáva.

Keď sa robot pohybuje dopredu/dozadu, veľmi jednoducho sa zisťuje, o koľko sa robot pohol. Odmeral som si, že jednou otočkou sa koleso posunie o 28cm. Teraz stačí len zistiť, o koľko otočiek sa koleso otočí, aby sme vedeli kde sa robot nachádza. Napríklad keď chceme aby sa robot posunul o 1 meter dopredu, musí zapnúť všetky kolesá rovnakým smerom (dopredu), aby sa otočili o 1 / 20 \* 100 = 3.57 otočiek. Dozadu je to to isté.

Otáčanie do strán je o niečo zložitejšie. Robot sa musí otáčať okolo svojej osi. Všetky kolesá napravo točí do jednej strany, kolesá naľavo do strany druhej. Tu ale vzniká na rôznych povrchoch rôzna odchylka otočenia. Prečo? Kolesá sa sami neotáčajú. Sú vytvorené, aby vedeli s robotom hýbať dopredu a dozadu. Keď začneme robota otáčať do prava, predné kolesa sa budú trieť o zem smerom do prava, zadné kolesá do ľava. Sila trenia závisí od materiálu zeme. To u mňa spôsobilo, že keď som robota otáčal na parketách, otočil sa presne o 90[°](http://www.degreesymbol.co/), ako som predpokladal. Keď som ho skúšal otočiť na linoleu, otočil sa len o 75[°](http://www.degreesymbol.co/).

Preto som na robota pridal gyrosenzor (Obrázok č. 23). Gyroskopický senzor je zariadenie, ktoré dokáže zistiť svoju rotáciu a rýchlosť otáčania. Vždy keď sa robot točí, zapnem počítanie gyrosenzoru. Počas otáčania viem kontrolovať[[12]](#footnote-11) o koľko stupňov sa robot už otočil. Síce na niektorých povrchoch trvá točenie dlhšie, na niektorých kratšie, mám istotu, že sa robot vždy otočí tak, ako chcem.



*Obr. č. 23 - Gyroskopický senzor*

Tu je ukážka pohybu dopredu:

void loop() {

// Ak prichadza signal cez bluetooth

if (Serial.available() > 0) {

// Prečítaj signal

SerialState = Serial.readString();

// Ak je signal prikaz UP

if (SerialState == "UP") {

stopMovement = false;

digitalWrite(dirBR, HIGH);

digitalWrite(dirFR, HIGH);

digitalWrite(dirFL, LOW);

digitalWrite(dirBL, LOW);

}

}

//…

digitalWrite(stepBR, HIGH);

digitalWrite(stepFR, HIGH);

digitalWrite(stepFL, HIGH);

digitalWrite(stepBL, HIGH);

delayMicroseconds(del);

digitalWrite(stepBR, LOW);

digitalWrite(stepFR, LOW);

digitalWrite(stepFL, LOW);

digitalWrite(stepBL, LOW);

delayMicroseconds(del);

}

Celý program ktorý spúšťam na Arduine, sa dá pozrieť na stránke: <https://github.com/samodostal/mars-2020>

### Pohyb + Radar

Samotný pohyb a radar tvoria jadro robota. Jediné čo nám chýba, je posielanie dát na sever. Dáta posielame cez TX pin. Teraz sa už len musíme dohodnúť na forme dát. Keďže server bude očakávať dáta presne v nejakom tvare, musíme zabezpečiť aby mu tak aj prišli. Tvar v ktorom dáta posielam je: “[PoziciaRobotaX, PoziciaRobotaY, UholRobota, UholRadaru, VzdialenostRadaru]”. PoziciaRobotaX a PoziciaRobotaY sa zisťujú z vzdialenosti ktorú robot prešiel od jeho spustenia. UholRobota sa zisťuje z gyrosenzoru. UholRadaru a VzdielanostRadaru boli vysvetlené v kapitole radar.

Vždy keď sa zmeria nová vzdialenosť, radar sa otočí, robot sa pohne, vyšle sa na server nová sada dát.

## Server

Samotný server bol programovaný v JavaScripte, spúšťaný pomocou Node.js. Node.js je prostredie, v ktorom sa dá spustiť kód napísaný v JavaScripte, na strane servera[[13]](#footnote-12). V konečnom dôsledku to vyzerá, ako by som to napísal napríklad v Pythone a spustil cez konzolu.

Server som rozdelil na dve časti. Jedna komunikuje s Arduinom, druhá komunikuje s klientom[[14]](#footnote-13).

### Komunikácia s Arduinom

Tým že som spároval Bluetooth na Arduine s mojím počítačom, si vytvoril počítač virtuálny port. K tomuto portu viem pristupovať, posielať a prijímať dáta. Má to ale jeden problém. Node.js nevie pristupovať k systémovým nastaveniam, rovnako, ako neviete cez webovú stránku vypnúť počítač. Dá sa to ale obísť inštalovaním knižnice serialport, ktorá má prístup ku všetkým portom počítača.

Teraz dokážeme pozrieť všetky porty pripojené k zariadeniu:

const portInfoList = await SerialPort.list();

Následne dokážeme nájsť modul HC-05 podľa adresy:

const BLUETOOTH\_ADDRESS = "98:D3:91:FD:3A:4B";

Posledná vec je zapnutie počúvania na dáta, ktoré cez bluetooth na server prídu. Taktiež dokážeme poslať dáta zo servera na bluetooth. To využívame na posielanie príkazov:

const writeToBluetoothDevice = message => {

if (port) port.write(message);

}

parser.on("data", data => {

console.log("DATA RECEIVED FROM BLUETOOTH: %s", data);

});

Po tom nám z Arduina prišli dáta, ich musíme spracovať. PolohuX Y robota spracovávať netreba. To isté platí aj pre rotáciu. Čo ale treba spracovávať, je uhol a vzdialenosť ktorú nameral radar. Tieto údaje prichádzajú na server v podobe polárnych súradníc. V tomto kroku by sme ich potrebovali premeniť na normálne, karteziánske[[15]](#footnote-14). To sa dá zvládnuť pomerne jednoduchou trigoniometriou[[16]](#footnote-15):

const computeX = (distance, angle) => Math.cos(angle) \* distance;

const computeY = (distance, angle) => Math.sin(angle) \* distance;

const computeXY = (distance, angle) => ({

x: computeX(distance, angle),

y: computeY(distance, angle)

});

Stačí zavolať funkciu computeXY s parametrami: vzdialenosťou a uhlom.

### Komunikácia s klientom

Strana klienta slúži čisto na vykreslovanie mapy. Preto potrebujeme dáta dostať len jedným smerom, zo servera na klienta. Väčšinov táto komunikácia funguje tak, že klient pošle request na stranu servera a vypýta si nejaké dáta. V tomto prípade to ale nebude fungovať, lebo dáta chodia na server asynchrónne a klient nemá ako zistiť, či a kedy tam dáta prídu. Preto sme využili websockety[[17]](#footnote-16). Takto dokážeme preposlať spracované dáta z Arduina, ihneď ako na server prídu. Na použitie websocketov treba nainštalovať knižnicu ws. Potom dokážeme jednoducho vytvoriť nový websocket server:

let server = new WebSocket.Server({

port: WEBSOCKET\_PORT

});

S tým, že keď potrebujeme poslať dáta na klienta, stačí zavolať funkciu sendMessage. Má dva parametre. Prvý id, je dôležitý z dôvodu, že na websocket server dokážeme pripojiť viac klientov naraz. Id označuje klienta, na ktorý chceme správu poslať. Samotná správa sú upravené dáta, ktoré chceme poslať na vykreslenie:

const sendMessage = (id, message) => {

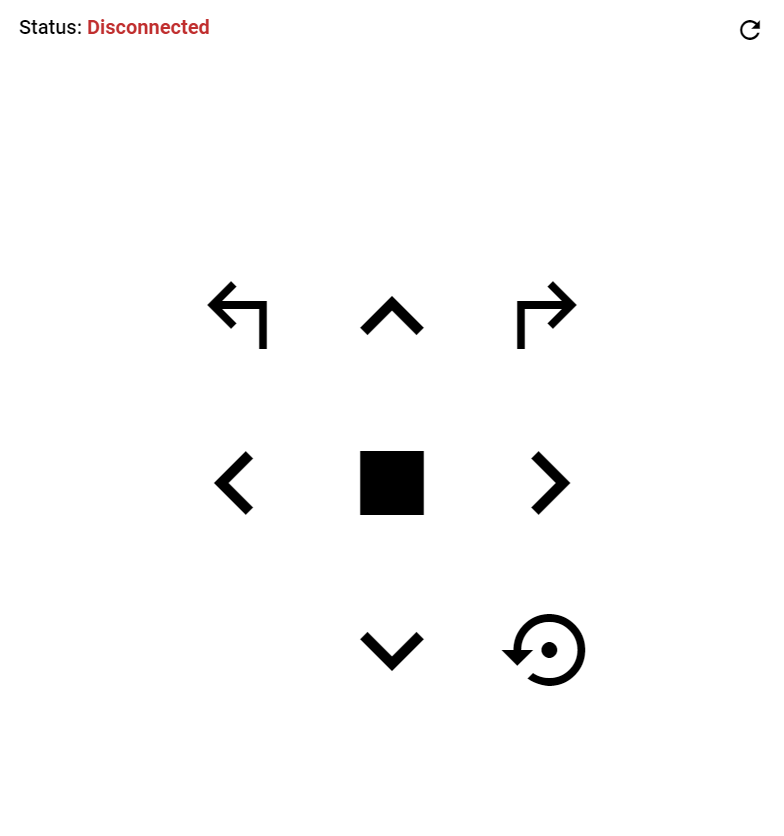
const socket = sockets[id];

if (socket)

socket.send(message);

};

Posledná vec ktorú som neadresoval, je, odkiaľ bude Arduino dostávať príkazy. Môj prvý plán bol, aby sa robot inteligentne sám pohyboval po miestnosti, čo som spraviť nestihol. Kvôli tomu som vytvoril ďalšieho klienta. Nazývam ho controller a slúži ako ovládač.



*Obr. č. 24 - Strana klienta - Controller*

Pomocou ovládača s robotom môžeme hýbať dopredu, dozadu, do strán, rotovať radarom, resetovať pozíciu radaru a zastaviť akýkoľvek pohyb robota. Pripája sa úplne rovnako ako prvý klient, pomocou websocketov. Jediný rozdiel je, že v tomto prípade dáta cestujú naopak. V prvom klientov, dáta chodia výlučne zo servera na klienta. V prípade controllera, chodia dáta iba z klienta na server. Tam na príkazy server počúva a preposiela na Arduino:

socket.on("message", message => {

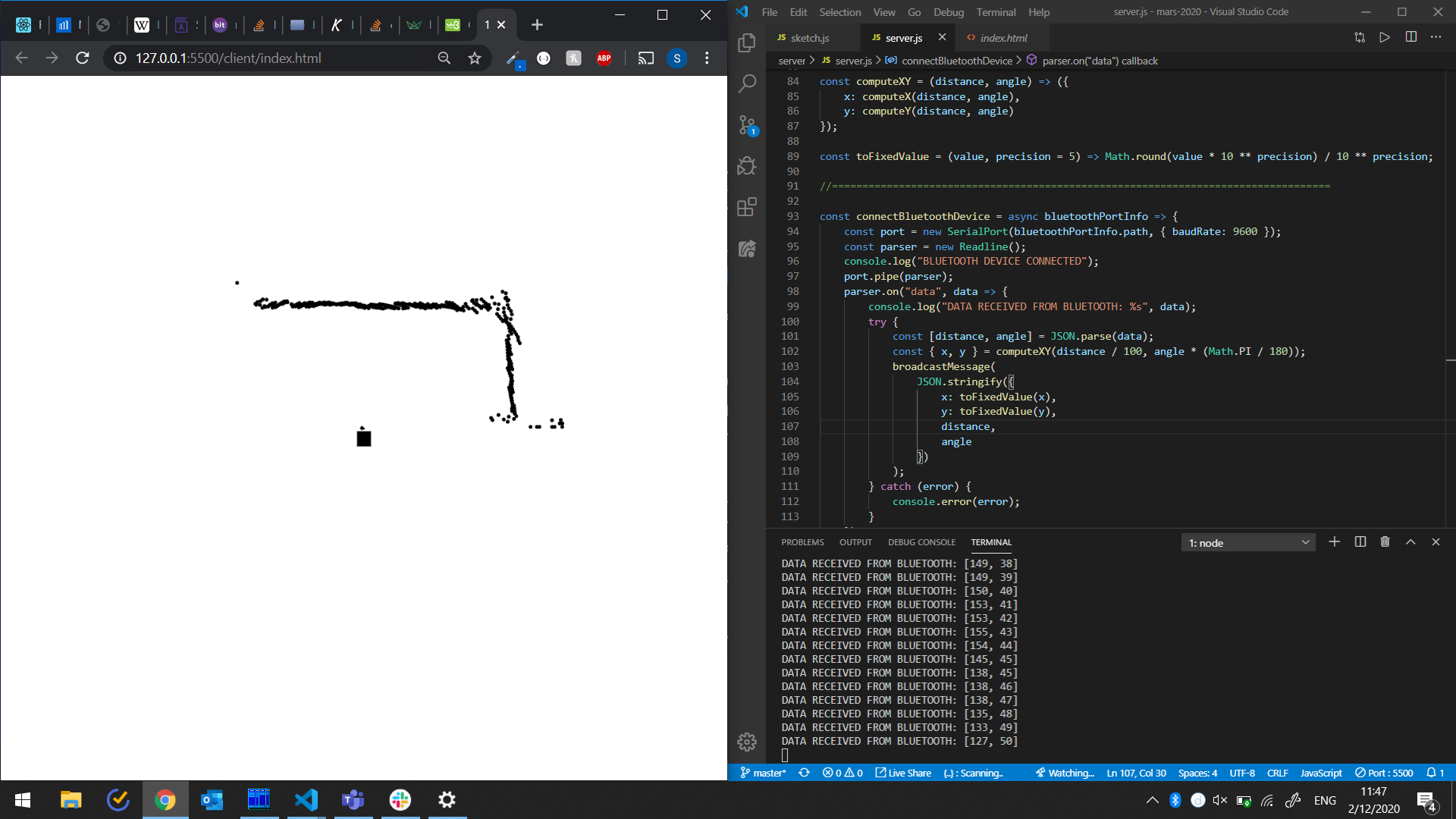
console.log("CLIENT (%d) SENT MESSAGE: %s", id, message);

writeToBluetoothDevice(message);

});

## Klient

Vykreslovanie mapy prebieha nasledujúcim spôsobom: Vždy keď prídu dáta o novom bode, alebo zmeny polohy robota, mapa sa prekreslí s novými dátami. Využívam tam knižnicu zvanú p5.js, ktorá slúži na jednoduchú prácu s canvasom[[18]](#footnote-17). Po vykreslení, mapa vyzerala takto:

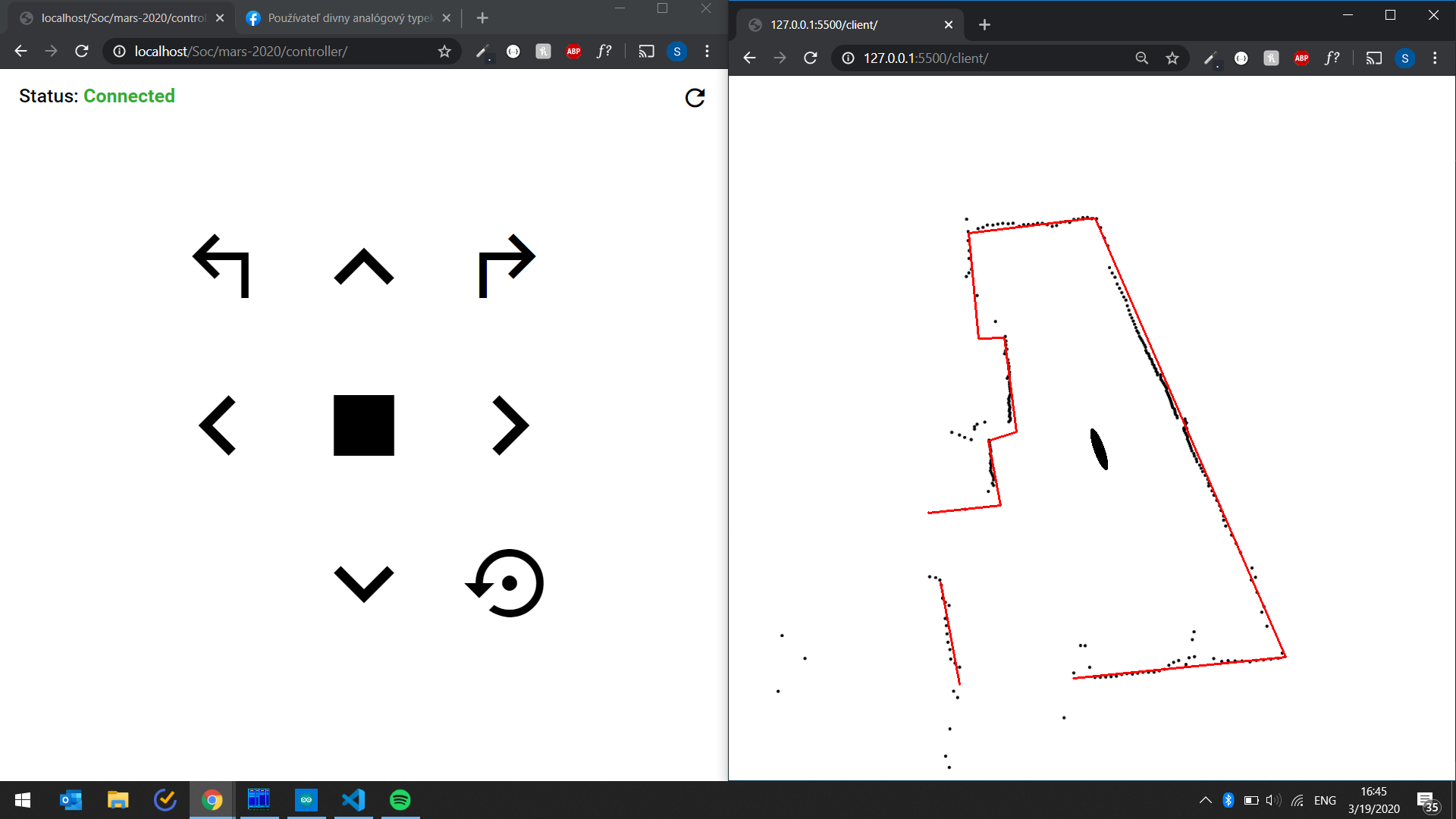
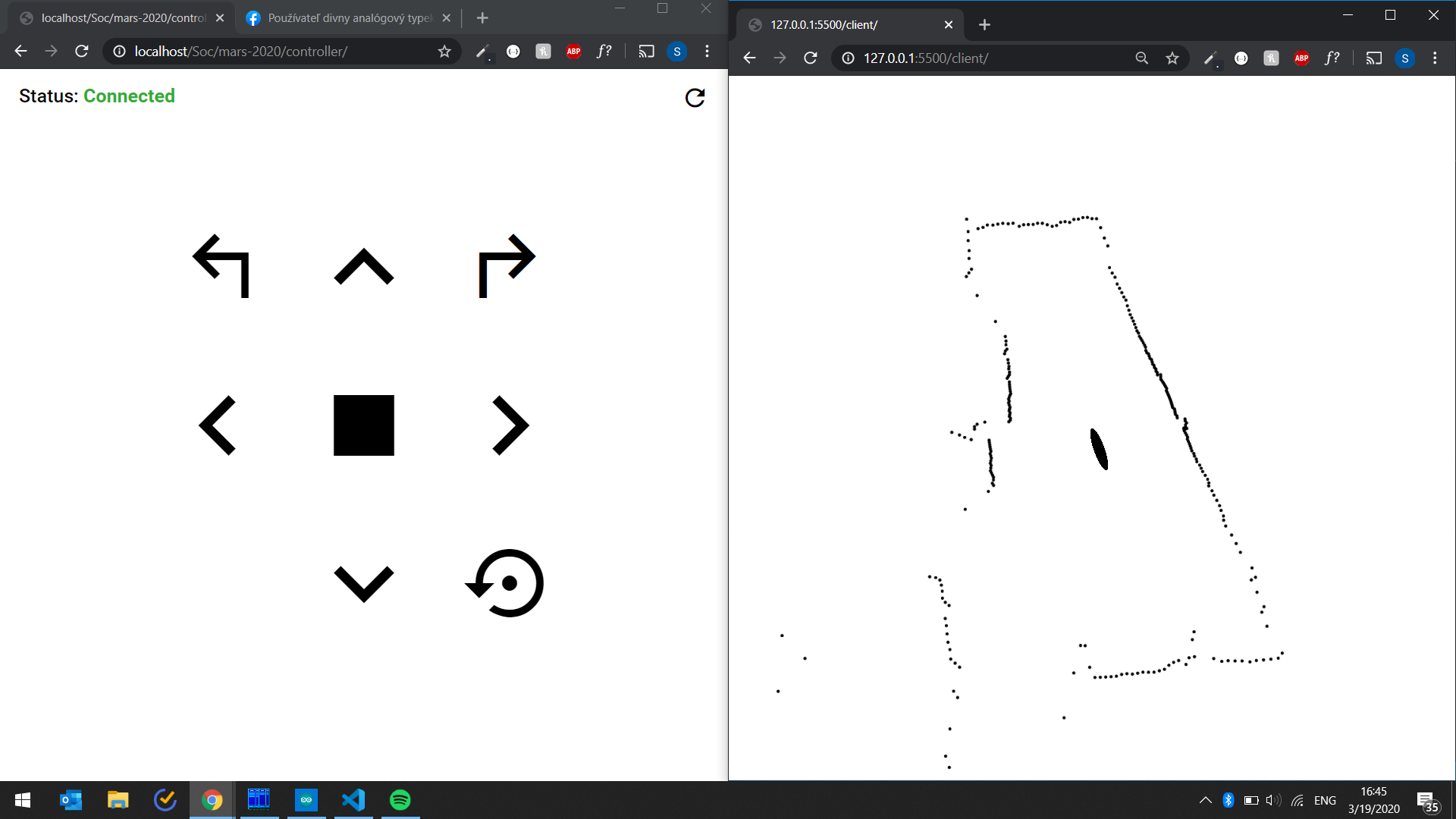


*Obr. č. 25- Vykreslenie mapy*

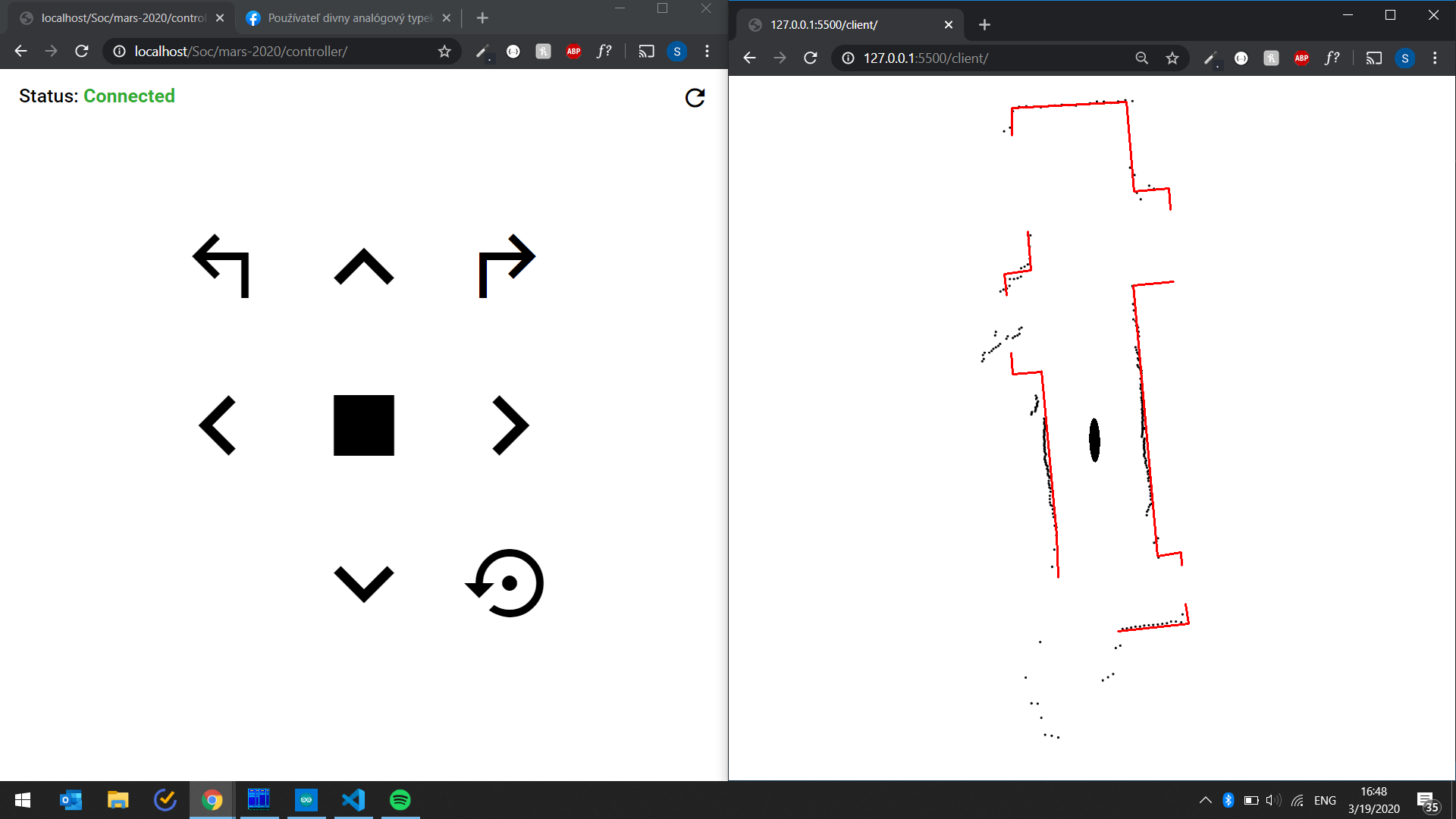
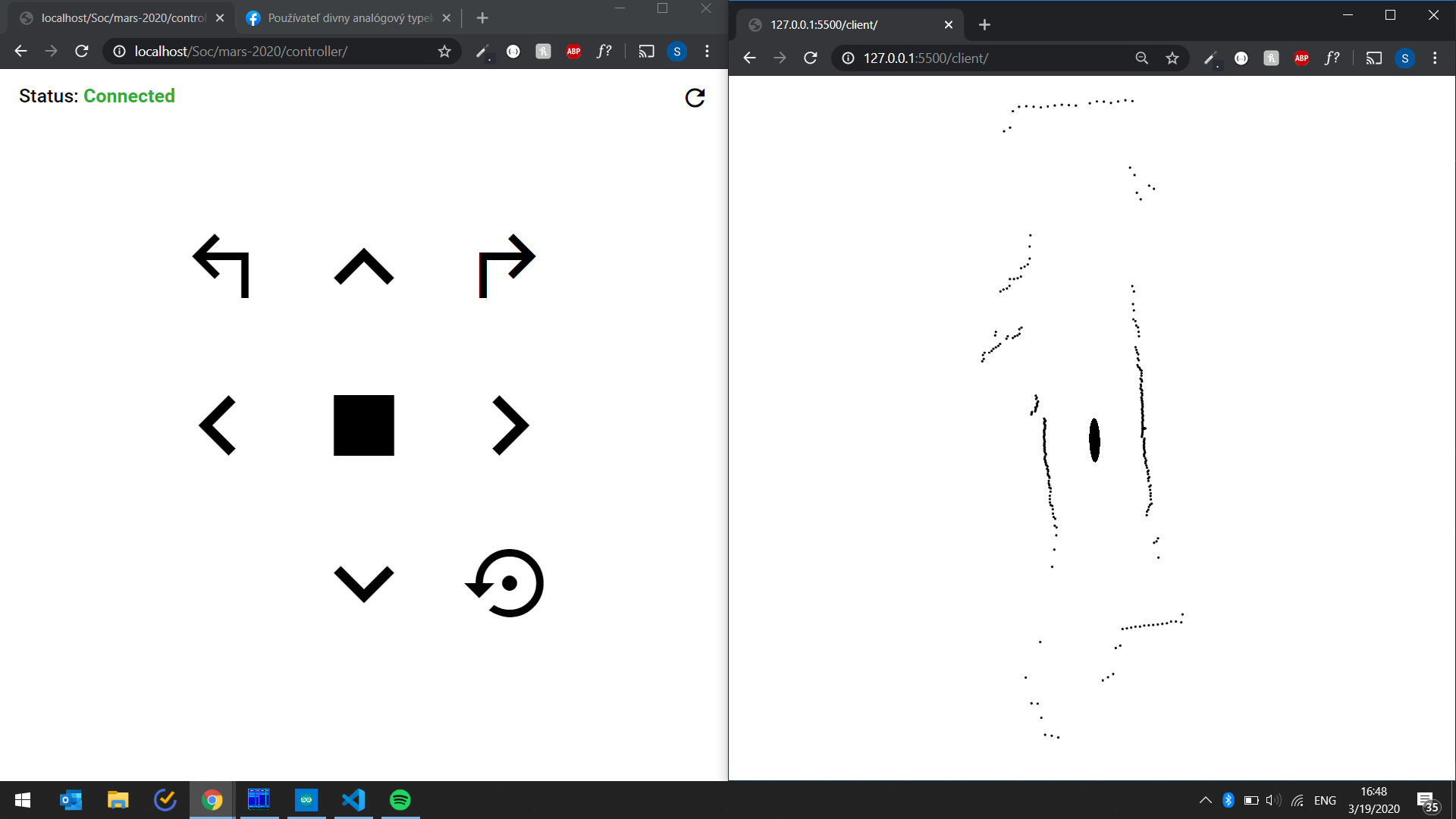
Pred robotom sa nachádza roh, dáta boli vykreslované len na intervale uhla (0, 135).

# Výsledky

Uvádzam dva príklady vykreslenia miestnosti. Na prvom obrázku je kuchyňa u nás doma, tak ako ju zmapoval robot. Na ľavo môžete vidieť controller, ktorým bol robot ovládaný. Napravo je miestnosť. Červené čiary na druhom obrázku ukazujú ako miestnosť vyzerá reálne.



To isté platí aj pre druhú miestnosť, chodbu u nás doma:



S výsledkami som veľmi spokojný, nakoľko som nečakal, že mapy budú až tak presne vykreslovať realitu. Podľa máp podľa mňa každý pochopí, ako miestnosť vyzerá a dokáže sa v nej navigovať.

# Záver

Aj keď som s prácou spokojný, vidím ešte veľa miesta na zlepšenia robota aj softvéru. Rád by som na robota pridal ďalšie 2 kolesá, tak, ako ich má aj rover Opportunity. Priestor vidím aj pri pridaní nových a vylepšených senzoroch. Čo sa týka softvéru, rád by som pridal vykreslovanie stien pomocou algoritmu Linear Regression, ktorý z viacerých bodov dokáže vytvoriť vektor. Taktiež autonómny pohyb robota.

Chcel by som sa poďakovať firme Resco, ktorá mi celý projekt zasponzorovala. Taktiež všetkým ľuďom ktorí mi pomohli s Hardvérom alebo Softvérom keď som sa zasekol, alebo mi poradili lepší spôsob ako ku danému problému pristupovať.

# Zhrnutie

Cieľom práce bolo spraviť robota, ktorý mal vedieť zmapovať svoje prostredie, následne ho poslať do môjho počítača na vykreslenie. Tento cieľ sa mi podľa mňa podaril, aj keď vidím ešte veľké miesto na zlepšenie, nielen v hardvérovej časti. V práci sa dozviete ako bol robot postavený a naprogramovaný. Okrem iného aj pochopíte, čo je to technológia LIDAR, aké má využitie a kde všade sa využíva.

# Resume

Objective of this work was to build a robot that would be able to map its environment, providing data to my computer which would draw a map of the environment. I think that this goal was accomplished, even when I still see a lot of room for improvement, not only in hardware. In this work you will be provided with information about how this robot was built and programmed. Besides that, you will understand what LIDAR stands for, what it is good for, and where are its boundaries.

# Referencie obrázkov

*Obr. č. 2:* <https://www.metalearth.com/content/images/thumbs/0001124_mars-rover_1200.png>

*Obr. č. 4:* h[ttps://www.distrelec.sk/Web/WebShopImages/landscape\_large/9-/01/Raspberry%20Pi-PI4%20MODEL%20B\_1GB-30152779-01.jpg](https://www.distrelec.sk/Web/WebShopImages/landscape_large/9-/01/Raspberry%20Pi-PI4%20MODEL%20B_1GB-30152779-01.jpg)

*Obr. č. 5:* [https://www.gme.sk/data/product/1024\_1024/pctdetail.774-026.1.jp](https://www.gme.sk/data/product/1024_1024/pctdetail.774-026.1.jpg)

*Obr. č. 6:* <https://static.wixstatic.com/media/3c6d20_7496b6e13b9a47d0ac14ce9016e68835~mv2.png/v1/fill/w_871,h_397,al_c/3c6d20_7496b6e13b9a47d0ac14ce9016e68835~mv2.png>

*Obr. č. 7:* <https://des.gbtcdn.com/uploads/pdm-desc-pic/Electronic/image/2018/10/28/15407053621090.jpg>

*Obr. č. 10:* <https://s3.amazonaws.com/robotc-wiki/wiki-images/thumb/d/d4/SHARP_IR_operation.png/400px-SHARP_IR_operation.png>

*Obr. č. 11:* <https://images-na.ssl-images-amazon.com/images/I/51252yc6UWL._AC_SY355_.jpg>

*Obr. č. 12:* <https://cdn.sparkfun.com/r/500-500/assets/parts/1/2/7/4/2/14588-TFMini_-_Micro_Infrared_Module-04.jpg>

*Obr. č. 23:* <https://www.metalearth.com/content/images/thumbs/0001124_mars-rover_1200.png>

*Obr. č. 14:* <https://aus3d.com.au/image/cache/catalog/products/stepper/40mmStepper-800x800.jpg>

*Obr. č. 16:* <https://sienci.com/wp-content/uploads/2018/05/A4988-Driver.jpg>

*Obr. č. 17:* <https://a.pololu-files.com/picture/0J10073.600.jpg?75d9ca5bb2e095e5c5f64350019e1b81>

*Obr. č. 18:* <https://howtomechatronics.com/wp-content/uploads/2015/08/A4988-Specsifications.png>

*Obr. č. 19:* <https://voidsofttech.com/wp-content/uploads/2019/06/HC-05.jpg>

*Obr. č. 20:* <https://ram-e-shop.com/wp-content/uploads/2018/10/kit_L3G4200D_gyroscope.jpg>

*Obr. č. 22:* https://s3.amazonaws.com/files.technologyreview.com/p/pub/images/quantumradar.jpg

# Zoznam použitej literatúry

## Knižné publikácie

Pinliang Dong, Qi Chen. LIDAR Remote Sensing and Applications. 221 strán. ISBN: 9781351233347.

Elizabeth Rusch, The Mighty Mars Rovers: The Incredible Adventures of Spirit and Opportunity. 80 strán. ISBN: 9780547478814

## Elektronické zdroje

https://www.electronicshub.org/raspberry-pi-vs-arduino/

https://www.blender.org/user-stories/nasas-experience-curiosity/

https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

https://create.arduino.cc/projecthub/Raushancpr/arduino-with-ir-sensor-1579b6

https://github.com/hideakitai/TFmini

https://learn.sparkfun.com/tutorials/tfmini---micro-lidar-module-hookup-guide/all

https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

https://github.com/jarzebski/Arduino-L3G4200D

https://www.npmjs.com/package/serialport

1. TX - Transmit, slúži na odosielanie dát, RX - Receive, slúži na príjem dát [↑](#footnote-ref-0)
2. Stretol som a aj s 5V, aj s 3.3V aj pri Trig pinoch, aj pri Echo [↑](#footnote-ref-1)
3. Senzor HC-SRO4 ktorý som ja používal, mal udávanú presnosť +/- 2cm. [↑](#footnote-ref-2)
4. Pomocou jedného krokového motora som meral uhol, s ultrazvukovým senzorom vzdialenosť. [↑](#footnote-ref-3)
5. Rýchlosť zvuku vo vzduchu je +/-340m/s. Rýchlosť svetla vo vzduchu je +/-299792458 m/s. [↑](#footnote-ref-4)
6. LIDAR (LIght Detection And Ranging) [↑](#footnote-ref-5)
7. Augmented Reality [↑](#footnote-ref-6)
8. Variabilný rezistor ktorý hovorí servo motoru aktuálny uhol [↑](#footnote-ref-7)
9. Pravdepodobne skrat [↑](#footnote-ref-8)
10. +/- 0.2[°](http://www.degreesymbol.co/) [↑](#footnote-ref-9)
11. Cez bluetooth modul [↑](#footnote-ref-10)
12. Počítam to pomocou zrýchlovania do smeru [↑](#footnote-ref-11)
13. V podstate ide o to, že JavaScript potrebuje JSEngine ktorý je v browseroch na to, aby sa dal spustiť. Node.js umožňuje kompilovanie JavaScript kódu u seba cez terminál. [↑](#footnote-ref-12)
14. Klient slúži na vykreslovanie mapy [↑](#footnote-ref-13)
15. Formát [x, y] [↑](#footnote-ref-14)
16. Ak ste sa niekedy učili Analytickú geometriu, tak to je presne ten vzorec na prepočet polárnych súradníc, na karteziánske [↑](#footnote-ref-15)
17. Websockety nám umožňujú posielať dáta zo servera na stranu klienta. [↑](#footnote-ref-16)
18. Plátno/Canvas slúži na vykreslovanie mapy [↑](#footnote-ref-17)