

Education
Solution Series

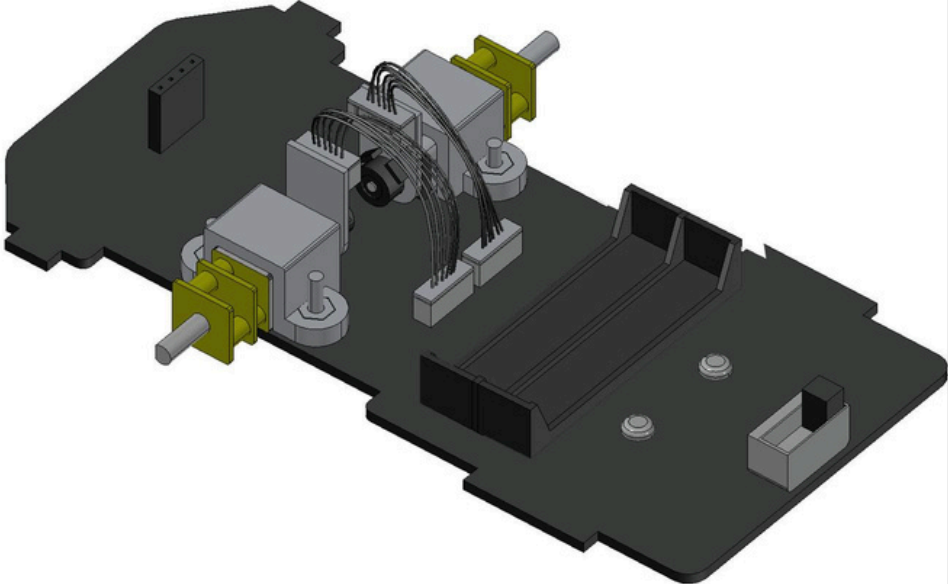
Intelligent Transportation Education Level 2 Kit



— Arduino Programming —

OBSAH

Kapitola 1 Sestavení Sharkbota – Hardware	1
Krok 1: Připojení kabelu motoru	3
Krok 2: Instalace kola SharkBotu	4
Krok 3: Instalace baterie.....	5
Krok 4: Instalace bočního panelu SharkBotu.....	6
Krok 5: Instalace těla SharkBotu.....	7
Krok 6: Instalace ultrazvukového senzoru.....	8
Návod pro instalaci Arduino IDE na Windows.....	10
Jak nainstalovat ovladač sériového portu CH340 (Windows).....	15
Výukový program pro Mac OS pro instalaci Arduino IDE	20
Jak nainstalovat sériový ovladač CH340 na MAC	23
Instalace modulu ESP32 do Arduino IDE	30
Kapitola 6: Autonomní řízení.....	35
Oddíl 1: Přehled autonomního řízení.....	36
Oddíl 2: Základní pohyb chytrého auta.....	43
Oddíl 3: Osvětlovací systémy pro chytré auto.....	53
Kapitola 7: Inteligentní řízení.....	62
Oddíl 1: Inteligentní vyhýbání se překážkám.....	63
Oddíl 2: Systém proti kolizím s alarmem.....	72
Oddíl 3: Hudba ve vozidle.....	80
Oddíl 4: Inteligentní sledování čáry (I).....	89
Oddíl 5: Inteligentní sledování čáry (II).....	97
Oddíl 6: Měření rychlosti.....	106
Oddíl 7: Jízda chytrého auta po přímce.....	116
Oddíl 8: Inteligentní hlasové ovládání.....	131
Kapitola 8: Internet věcí pro inteligentní dopravu.....	139
Oddíl 1: Technologie infračervené komunikace.....	140
Oddíl 2: Technologie Bluetooth komunikace.....	150
Oddíl 3: Technologie WiFi komunikace.....	160
Oddíl 4: Internet vozidel.....	179
Oddíl 5: Integrovaný systém inteligentní dopravy.....	195
Sledujte nás.....	201

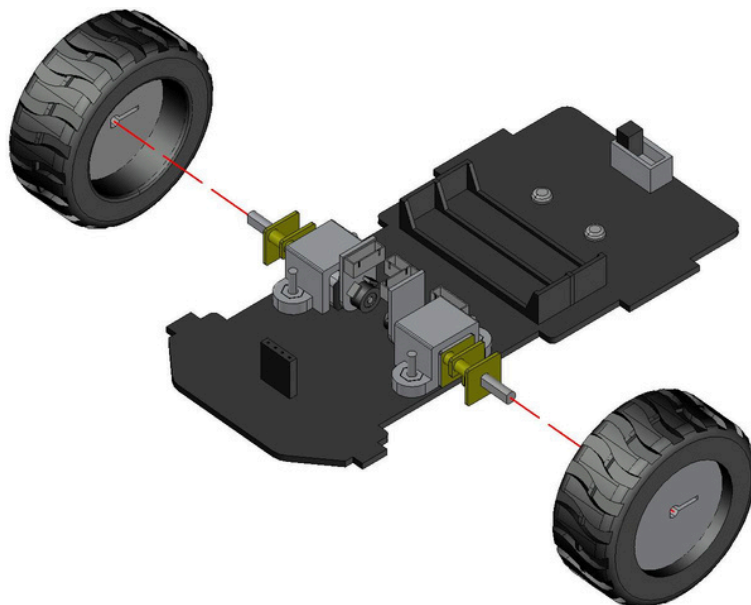
Krok 1: Připojení kabelu motoru		
Seznam součástí	Deska na podvozek auta*1	Motorový drát*2
Schéma zapojení		
Poznámka	• Použij kabel motoru k připojení motoru k odpovídajícímu konektoru. • Pokud se chceš podívat na montážní video, klikni na níže uvedený odkaz pro zobrazení videa k odpovídající sadě: https://www.youtube.com/playlistlist=PLkW5fEtHNu6KdMljKjBdHsWoRj9bcnLog • Nebo naskenuj níže uvedený QR kód. 	

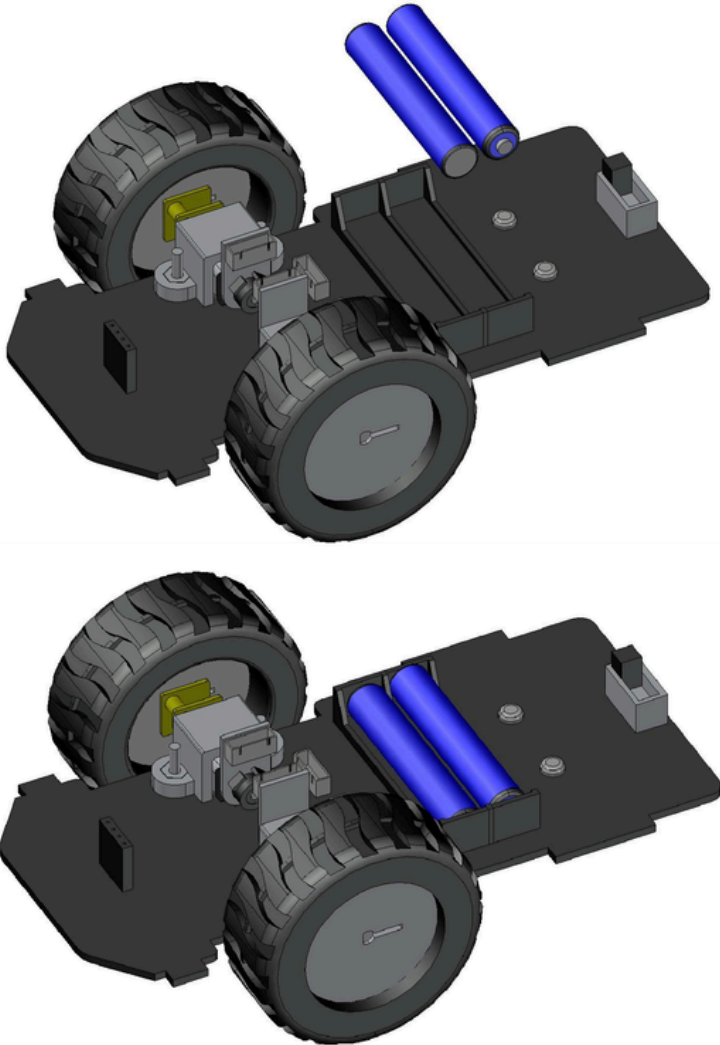
Krok 2: Instalace kola SharkBotu

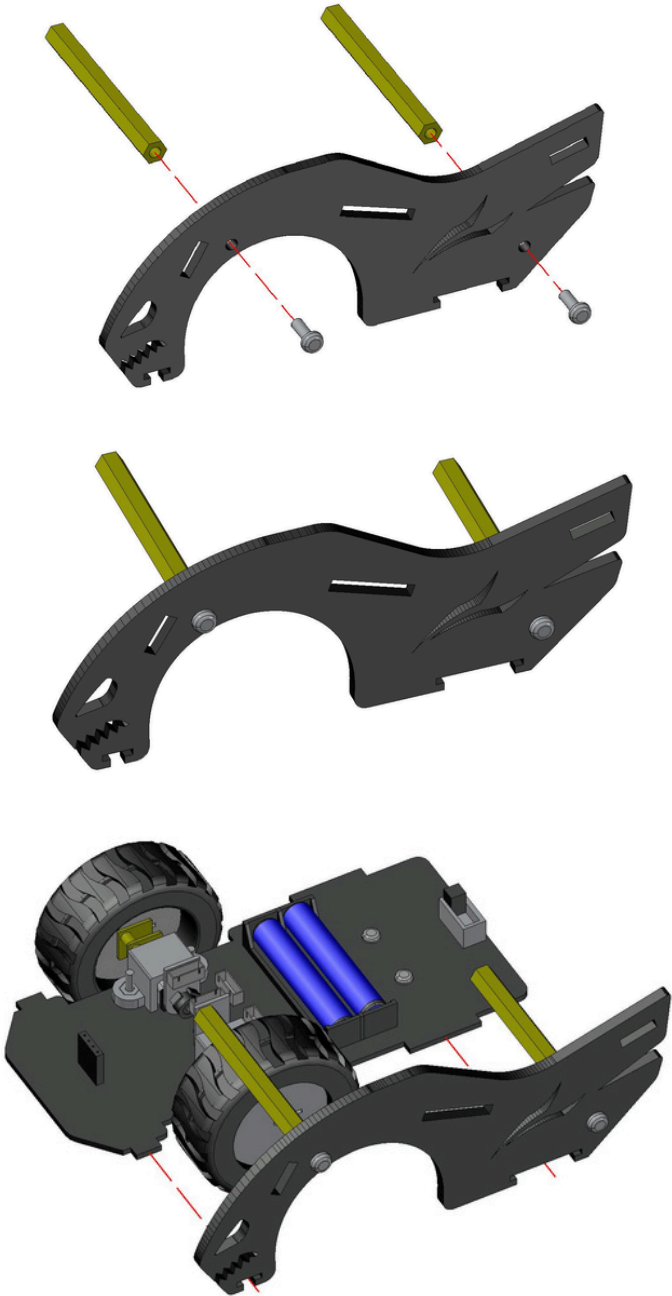
Seznam
součástí

Kolo*2

Schéma
zapojení

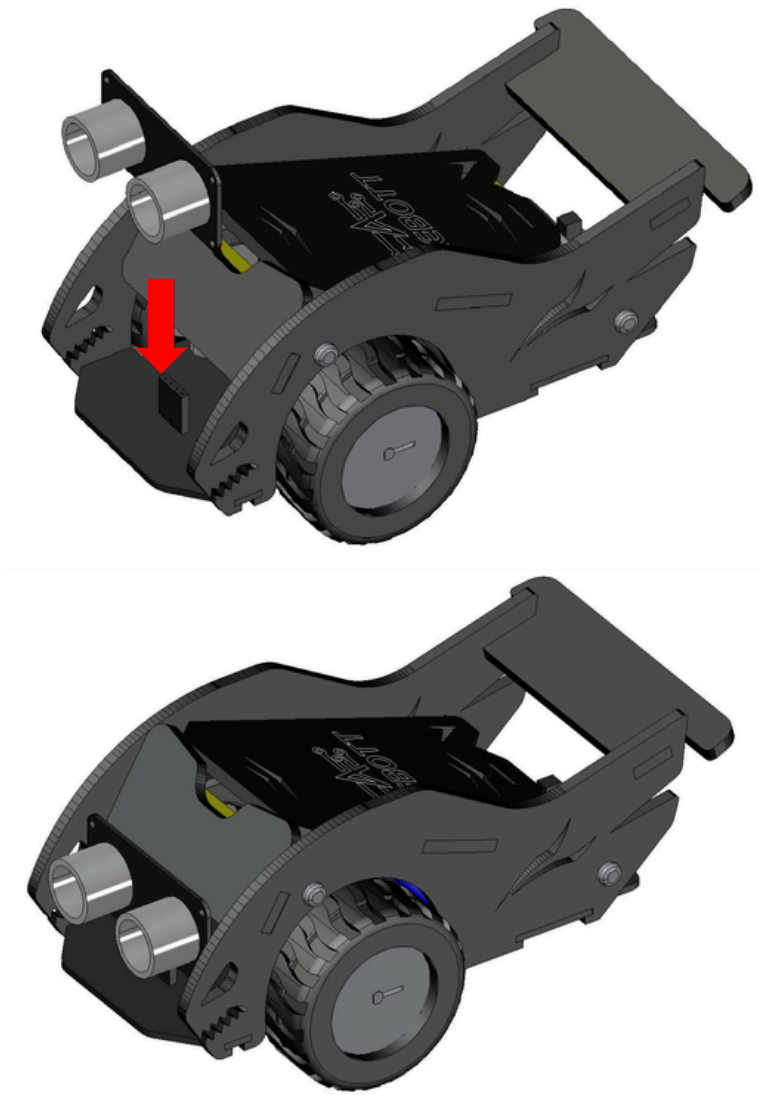


Krok 3: Instalace baterie	
Seznam součástí	10440 baterie*2
Schéma zapojení	
Poznámka	Prosím, vlož do přihrádky na baterie dvě baterie typu 10440 se správnou orientací polaritý (kladný a záporný pól).

Krok 4: Instalace bočního panelu SharkBotu		
Seznam součástí	Akrylátový boční panel*1	Šrouby M3 × 8 mm s kulatou hlavou *2
	Šroub M3 × 55 mm, dvojitý měděný sloupek *2	
Schéma zapojení		
Poznámka	Před instalací nejprve odstraň ochrannou fólii z akrylátového plechu.	

Krok 5: Instalace těla SharkBotu			
Seznam součástí	Akrylátový boční panel *1	Akrylátový zadní panel*1	Akrylátový konec *1
	Acrylic front bezel*1	Šrouby M3 × 8 mm s kulatou hlavou *2	
Schéma zapojení			
	Před instalací odstraň ochrannou fólii ze všech akrylátových panelů. Montáž začíná předním panelem SharkBotu, zadní deskou a zadní (ocelovou/zadní) částí. Poté použij šrouby M3 × 8 mm s kulatou hlavou k upevnění druhého bočního panelu k M3 × 55 mm dvojitému průchozímu sloupku.		



Krok 6: Instalace ultrazvukového senzoru	
Seznam součástek	Ultrazvukový sensor*1
Schéma zapojení	
Poznámka	Připoj ultrazvukový senzor k odpovídajícímu konektoru na základové desce.

Návod pro instalaci Arduino IDE na Windows.

Arduino IDE

Jako software s otevřeným zdrojovým kódem je Arduino IDE vyvinuto na základě Processing IDE, což je integrované vývojové prostředí oficiálně spuštěné společností Arduino.

S Arduino IDE stačí napsat kód programu do IDE a poté jej nahrát na desku Arduino a program řekne desce Arduino, co má udělat.

Stáhněte si Arduino IDE pro Windows

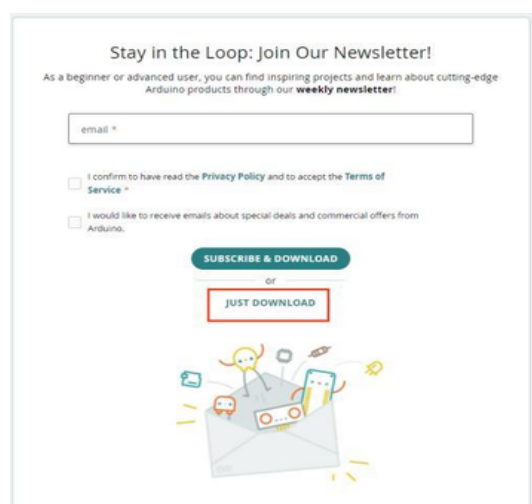
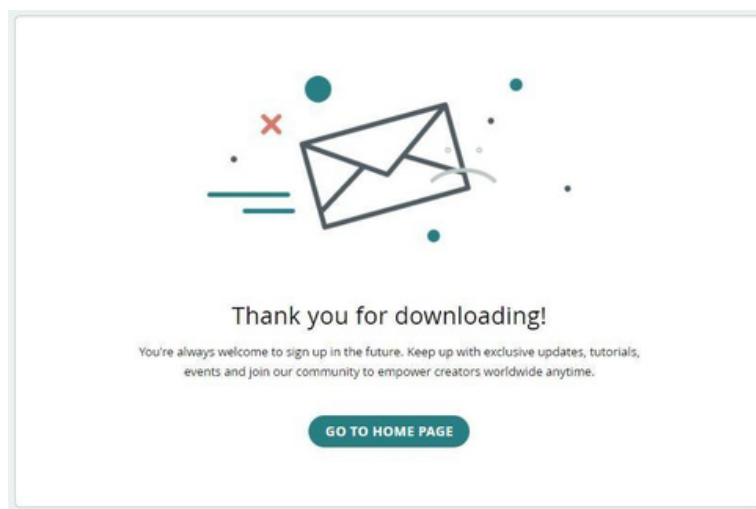
1) Stáhněte si Arduino IDE

① Stáhněte si webovou adresu Arduino IDE: <https://www.arduino.cc/en/Main/Software>, otevřete adresu URL podle následující tabulky a v budoucnu vyberte odpovídající verzi. Verze softwaru může být aktualizována, stačí nainstalovat nejnovější verzi.



② Vyberte JUST DOWNLOAD.

- ③ Když se objeví následující obrazovka, Arduino IDE se stahuje.



Nainstalujte Arduino IDE

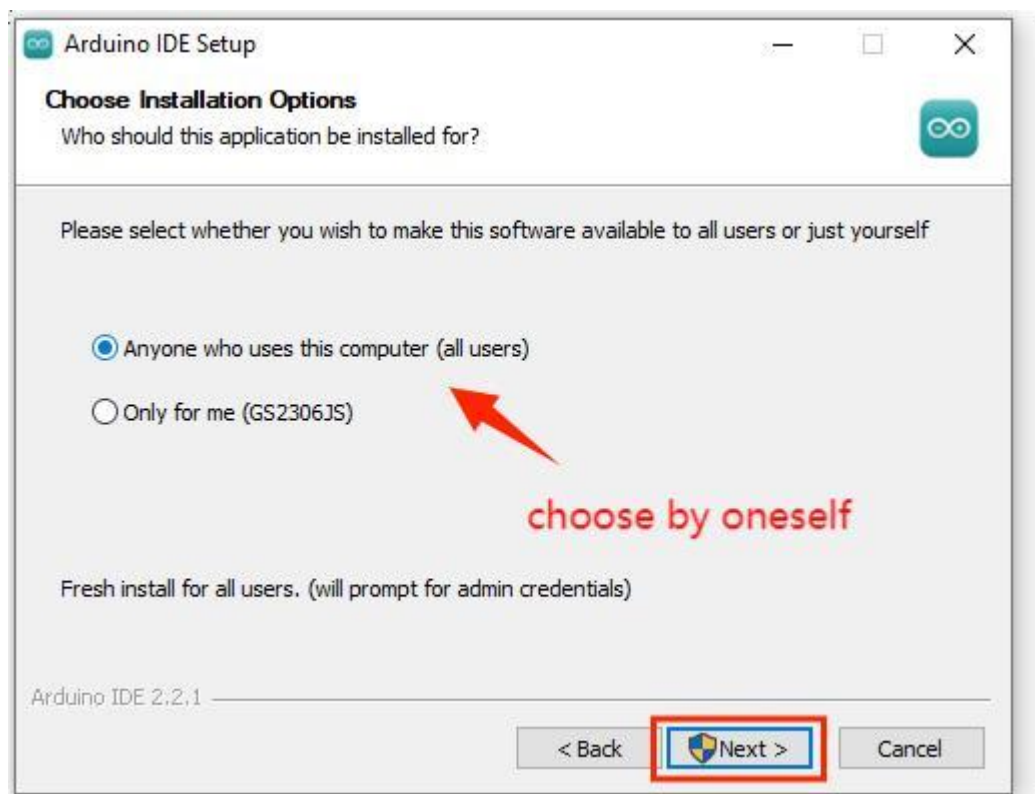
- ① Po dokončení stahování se zobrazí soubor ikony. Klepněte na Instalovat software.



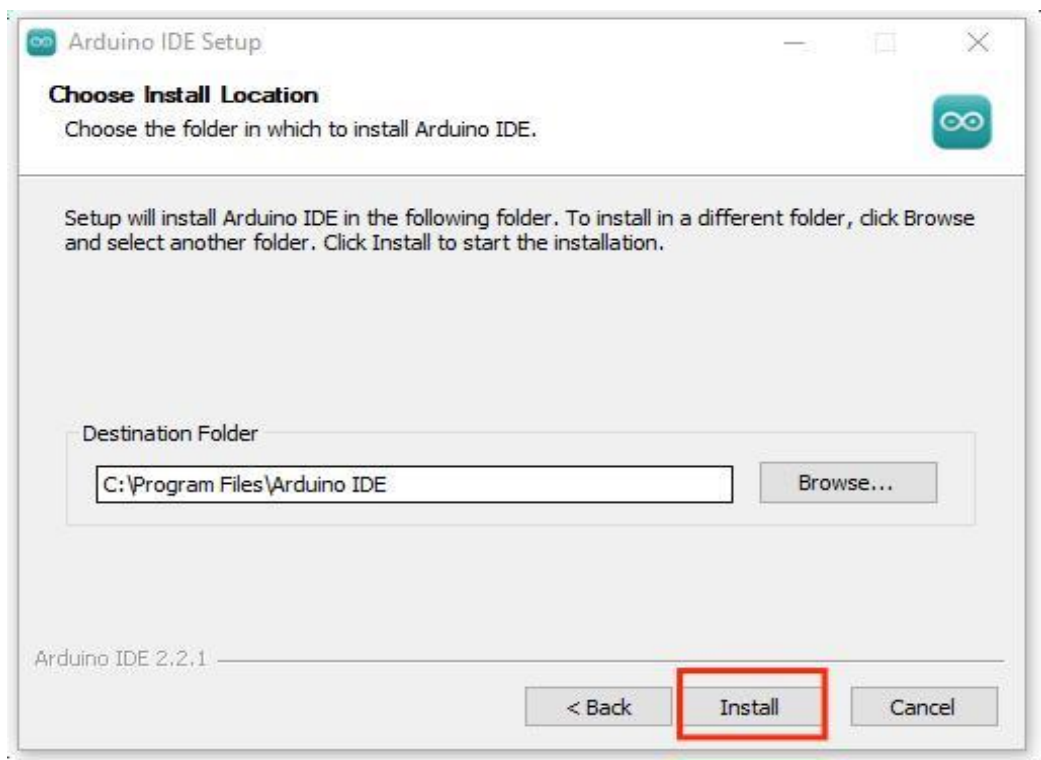
- ② Po instalaci se zobrazí následující obrazovka a vyberte „I Agree“.



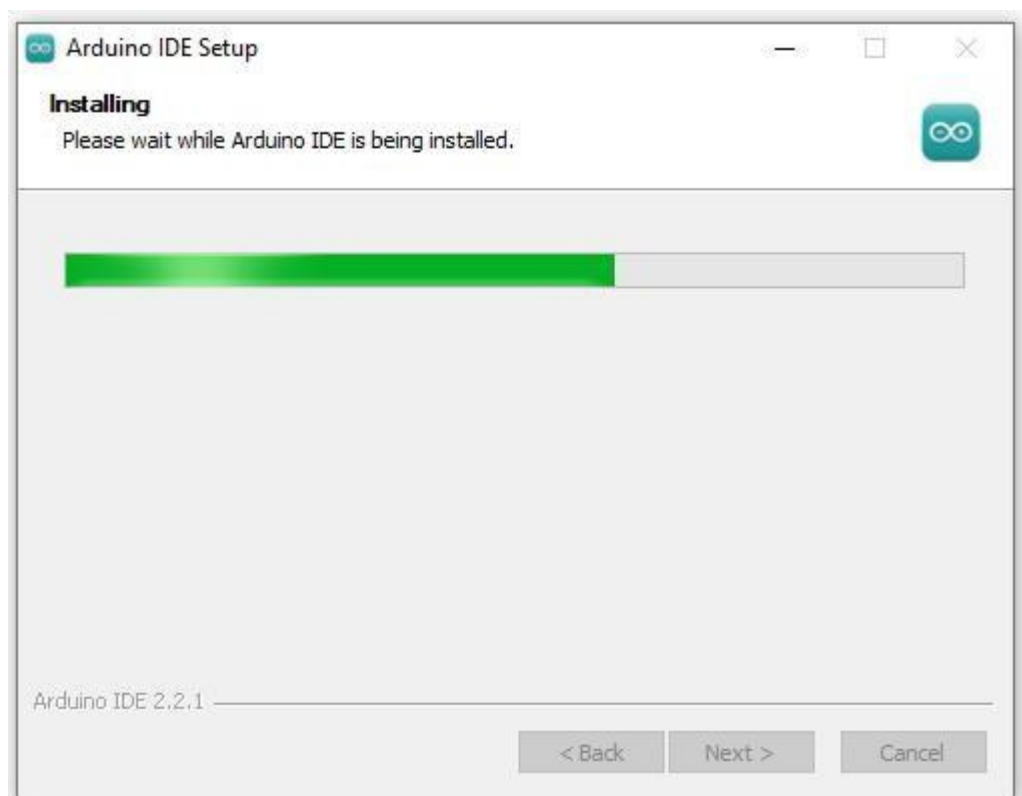
③ Po výběru „Next“ se zobrazí následující obrazovka a vyberte „Install“.



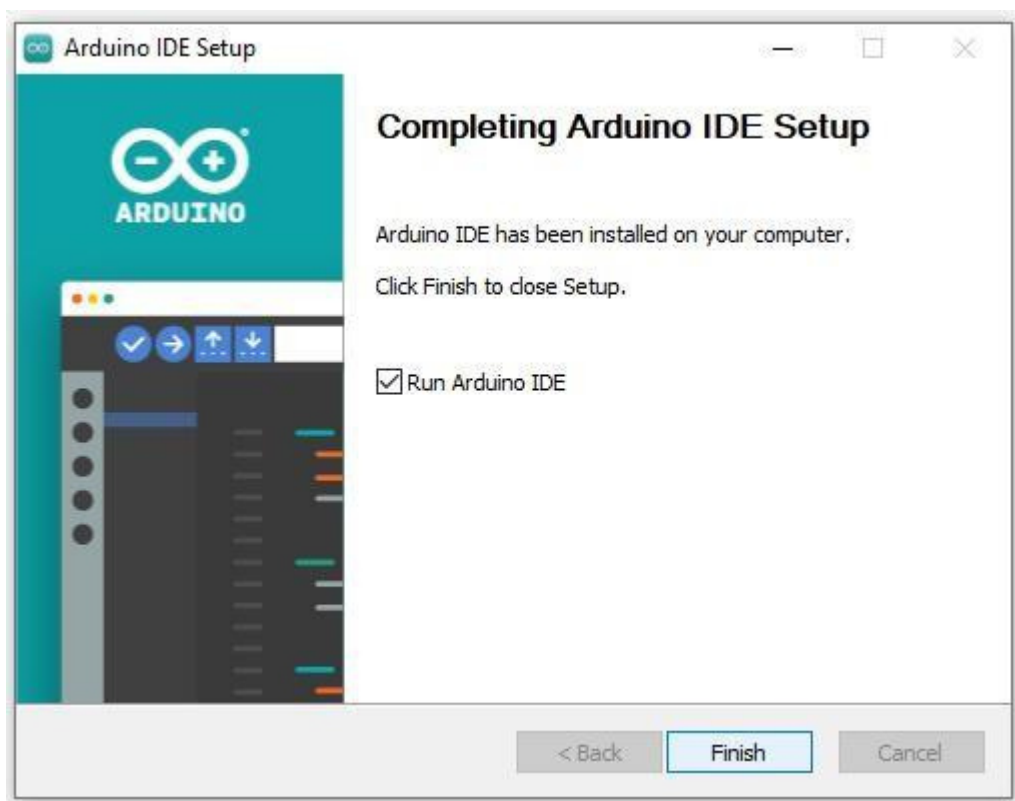
④ Vyberte „Next“ a objeví se následující obrazovka. Vyberte "Install".



⑤ Nainstalovaný software Arduino IDE.



⑥ Instalace je dokončena.



2) Instalace Arduino IDE je dokončena

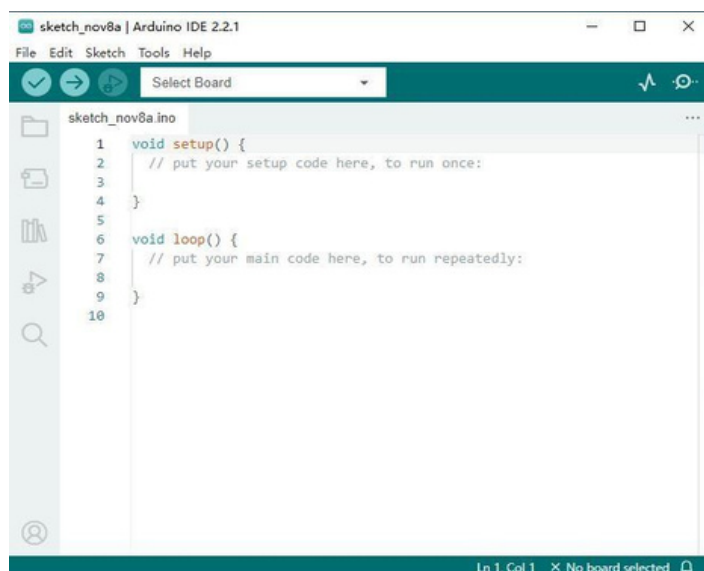


① Po instalaci se na ploše objeví ikona zástupce Arduino IDE.

- ② Po otevření se zobrazí následující obrazovka.

Jak nainstalovat ovladač sériového portu CH340

Připojte hlavní řídicí desku k počítači pomocí kabelu USB a ovladač se

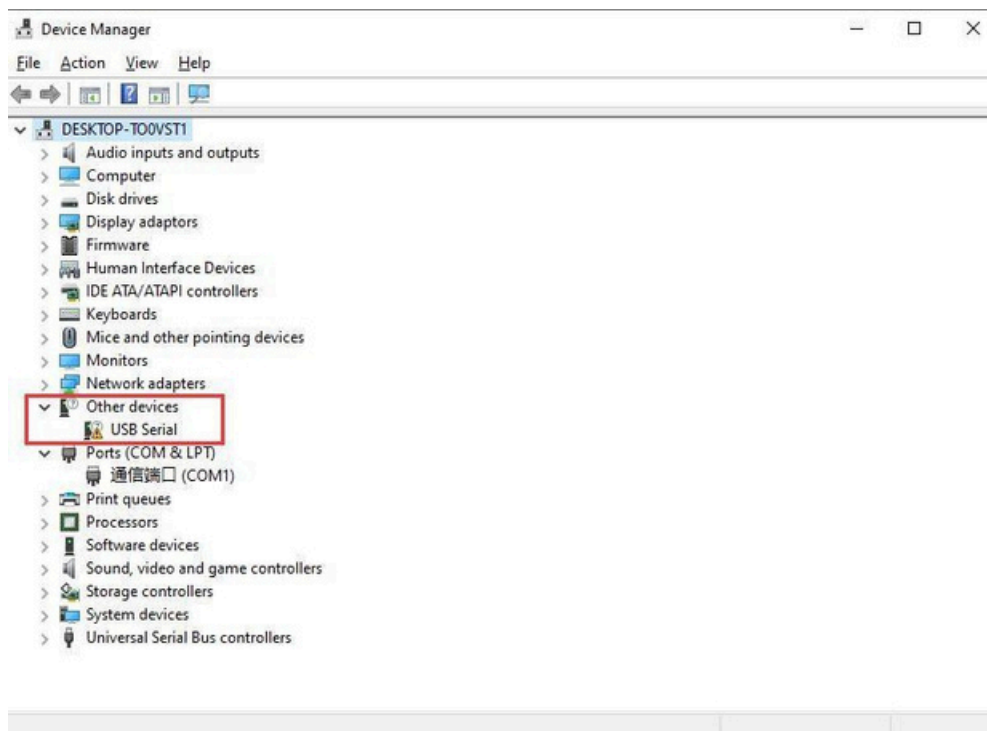


automaticky nainstaluje na systémy MacOS a Windows. Pokud se instalace ovladače nezdaří, musíte ovladač nainstalovat ručně.

Čip USB do sériového portu řídicí desky ESP32 je CH340C. Proto je potřeba nainstalovat ovladač pro čip. Proces instalace ovladače je na různých systémech v podstatě stejný. Zde si ukážeme instalaci ovladače na systému Win10. Složku "[USB Drive CH341 3 1](#)" naleznete v balíčku zdrojů, který jsme vám poskytli. Toto je soubor ovladače, který chceme nainstalovat.

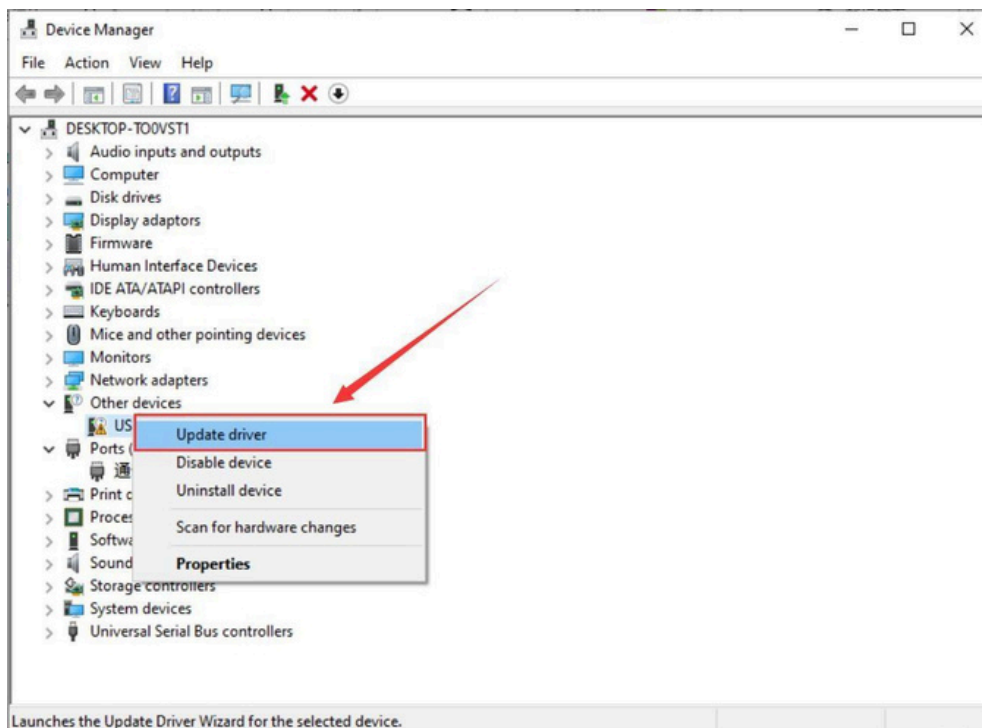
- 1) Zkontrolujte, zda je nainstalován ovladač sériového portu CH340
(pokud je nainstalován, přeskočte jej)

- ① Zapojte jeden konec USB kabelu do řídicí desky ESP32 a druhý konec do USB portu na počítači.
- ② Při prvním připojení konzole ESP32 k počítači klikněte pravým tlačítkem na "My Computer" -> "Attribute" -> Klikněte na "Device Manager" a pod "Other Devices" uvidíte buď "USB-Serial" nebo "Unknown Device".

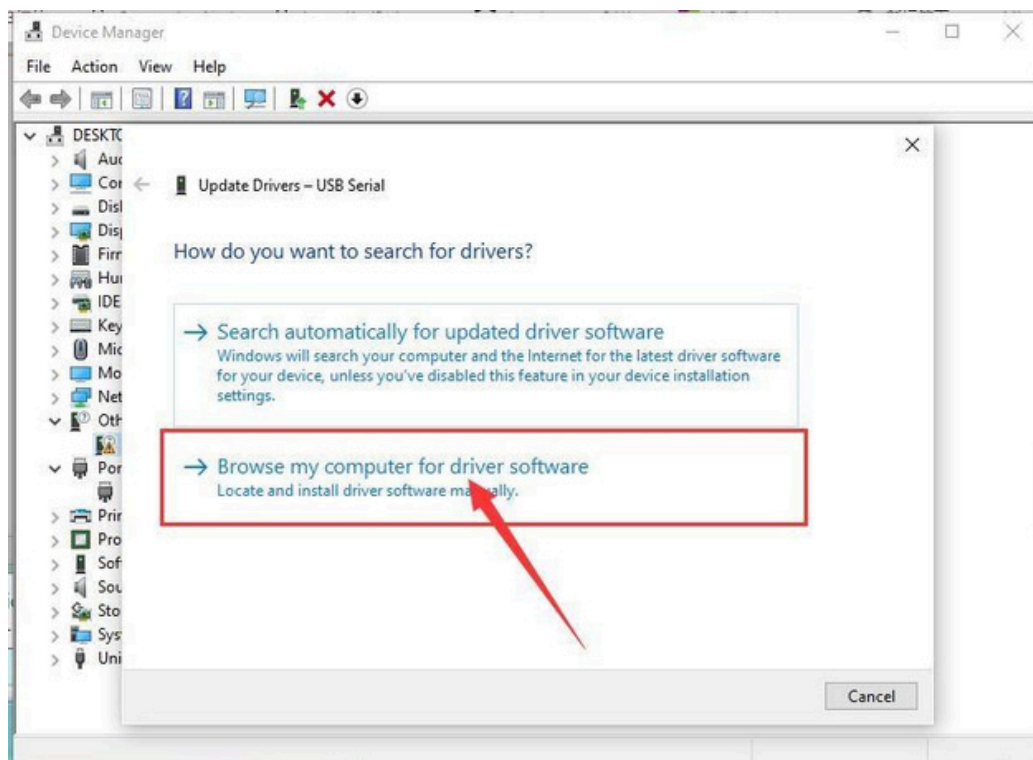


2) Nainstalujte ovladač sériového portu CH340

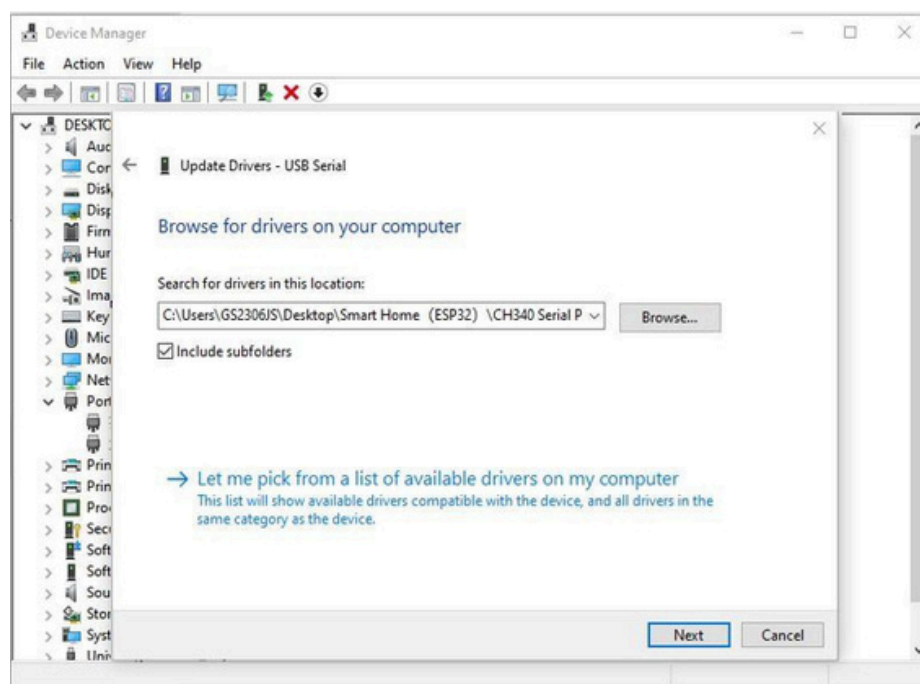
- ① Nejprve klikněte pravým tlačítkem na zařízení a vyberte možnost horní nabídky (Aktualizovat software ovladače), jak je uvedeno níže.



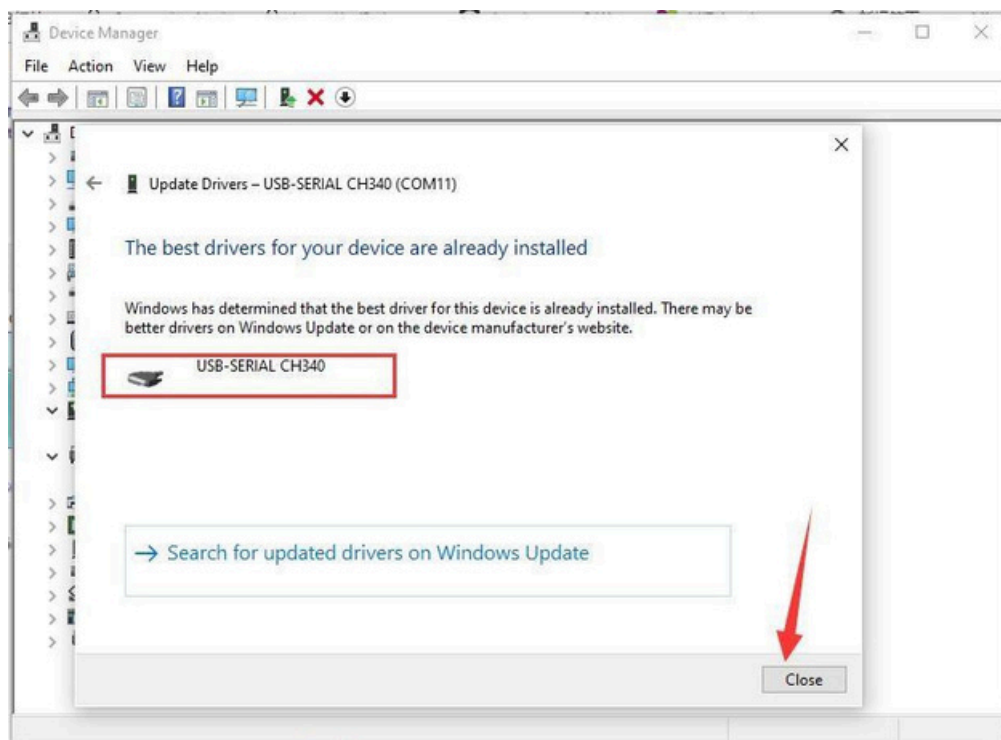
② Poté budete vyzváni k „Automatically search for updated driver software“ nebo „Browse my computer for driver software“, jak je uvedeno níže. Na této stránce vyberte „Browse my computer for driver software“.



③ Poté přidejte cestu k souboru ovladače.

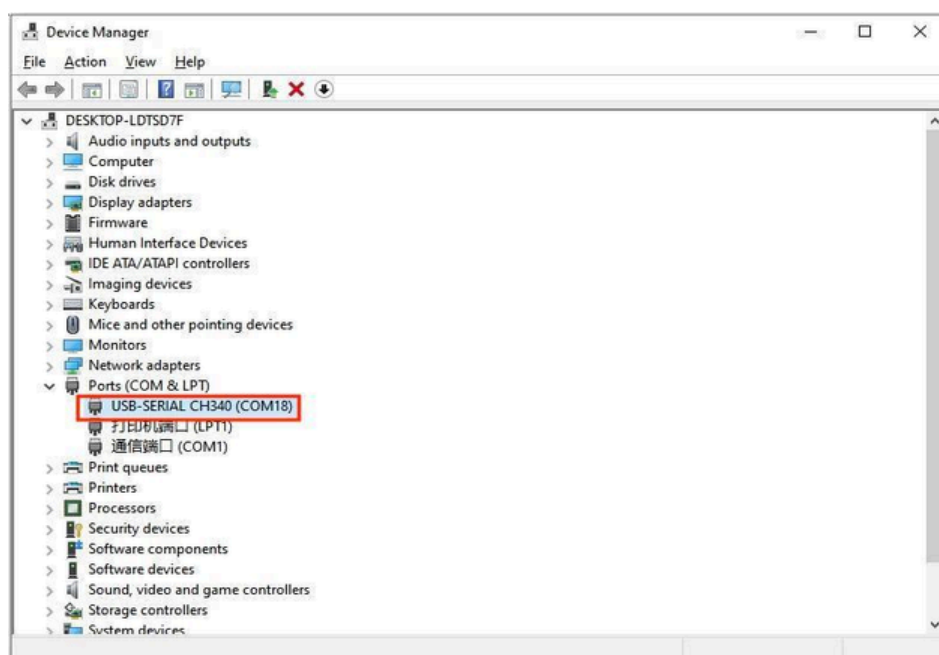


④ Po dokončení instalace softwaru obdržíte potvrzovací zprávu. Po dokončení instalace klikněte na „Close“.



3) Potvrďte, že byl ovladač sériového portu CH340 úspěšně nainstalován

Zapojte jeden konec USB kabelu do řídicí desky ESP32 a druhý konec do USB portu na počítači. Klikněte pravým tlačítkem na "My Computer" -> "Attribute" -> Klikněte na "Device Manager" a připojte se k ovládacímu panelu. Následující obrázek ukazuje, že instalace proběhla úspěšně.



Výukový program pro Mac OS pro instalaci Arduino IDE

Arduino IDE

Jako software s otevřeným zdrojovým kódem je Arduino IDE vyvinuto na základě Processing IDE, což je integrované vývojové prostředí oficiálně spuštěné společností Arduino.

S Arduino IDE stačí napsat kód programu do IDE a poté jej nahrát na desku Arduino. Program řekne Arduino, co má deska dělat.

Stáhněte si Arduino IDE pro Mac OS

1) Stáhněte si Arduino IDE

① Stáhněte si webovou adresu Arduino

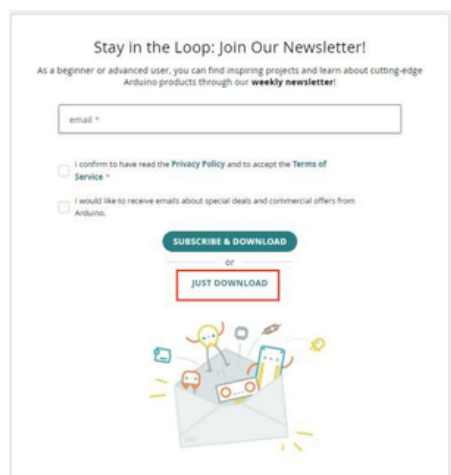
IDE: <https://www.arduino.cc/en/Main/Software>, otevřete adresu URL podle následující tabulky a v budoucnu vyberte odpovídající verzi. Verze softwaru může být aktualizována, stačí nainstalovat nejnovější verzi.



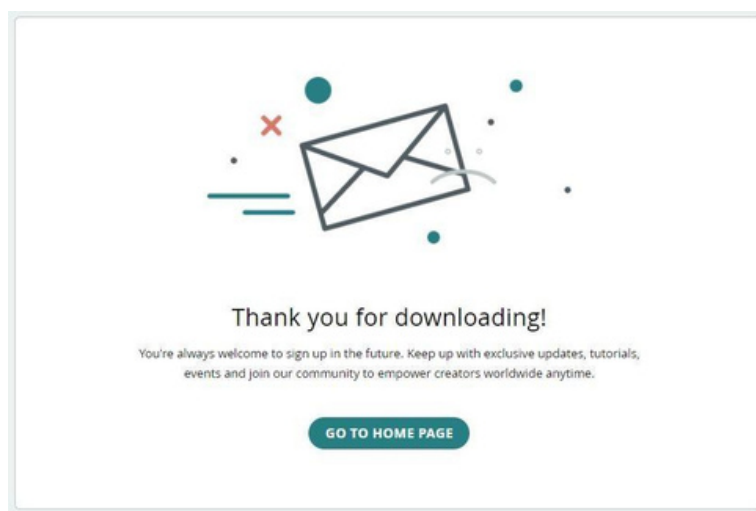
The screenshot shows the Arduino IDE 2.2.1 download page. On the left, there's a section for Arduino IDE 2.2.1 with a description of its features and a link to the documentation. On the right, there's a 'DOWNLOAD OPTIONS' section with a table of download links for Windows, Linux, and macOS. The macOS link for Intel is highlighted with a red box.

DOWNLOAD OPTIONS	
Windows	Win 10 and newer, 64 bits
Windows	MSI installer
Windows	ZIP file
Linux	ApplImage 64 bits (X86-64)
Linux	ZIP file 64 bits (X86-64)
macOS	Intel, 10.14: "Mojave" or newer, 64 bits
macOS	Apple Silicon, 11: "Big Sur" or newer, 64 bits

② Vyberte JUST DOWNLOAD.

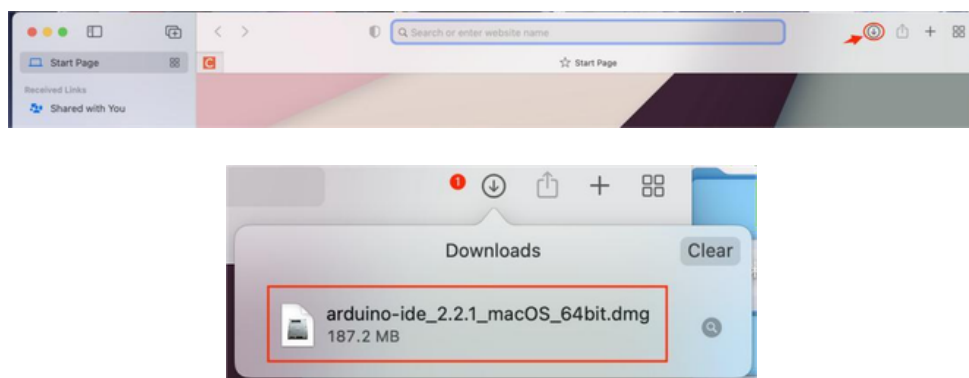


③ Když se objeví následující obrazovka, Arduino IDE se stahuje.



2) Nainstalujte Arduino IDE

① Po dokončení stahování klikněte na ikonu stahování ve vašem prohlížeči a najdete instalační program Arduino IDE.



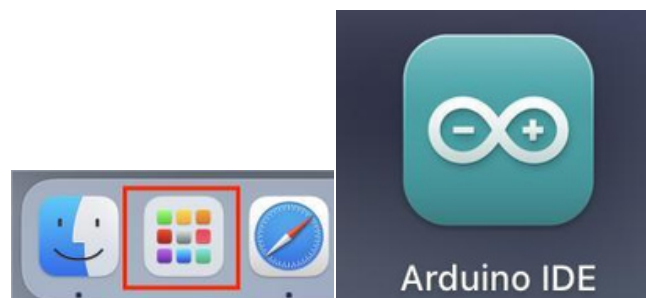
② Kliknutím na instalační balíček se zobrazí instalační obrazovka, stačí

vybrat ikonu Arduino IDE, přejít na Aplikace a program nainstalovat.

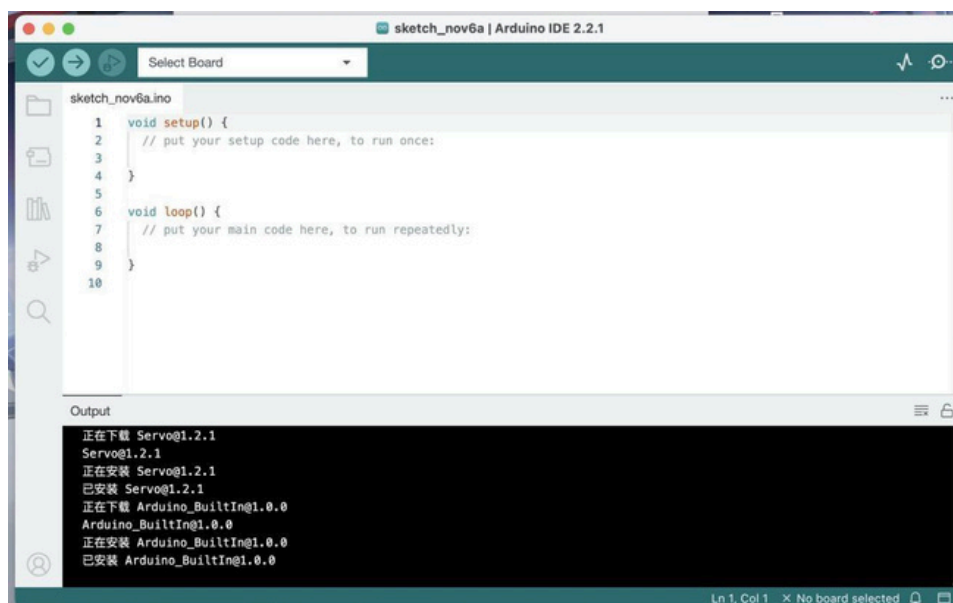


3) Instalace Arduino IDE je dokončena

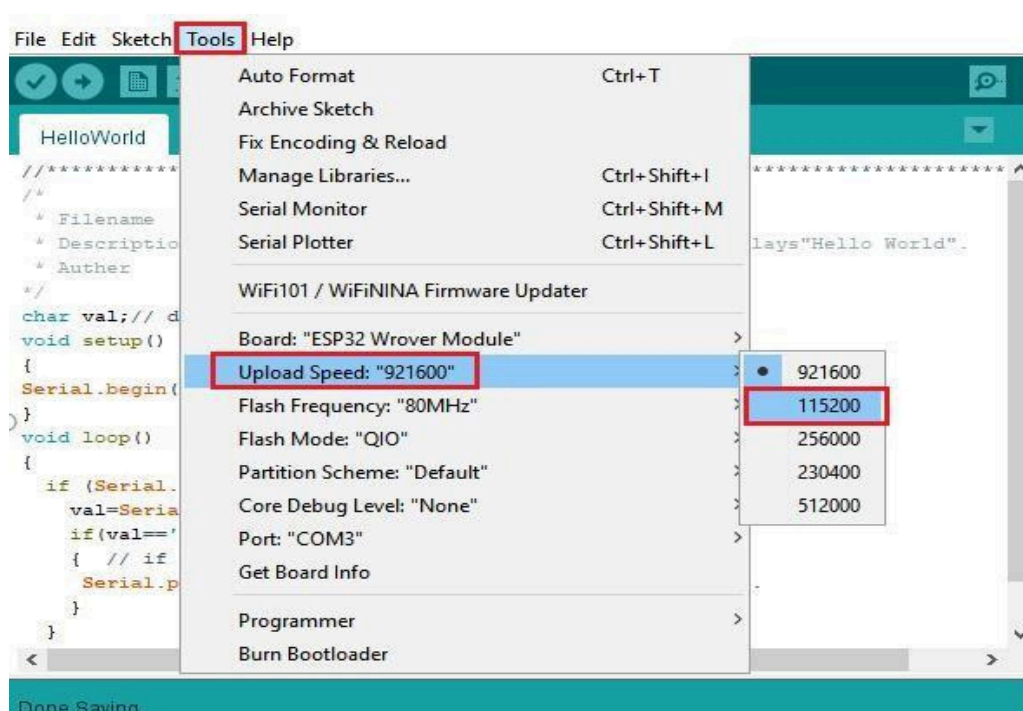
① V launchpadu vyhledejte Arduino IDE a otevřete jej.



② Otevřete program a uvidíte následující rozhraní programu.



③ Pro uživatele Mac OS nastavte baudovou rychlost na 115200 před kliknutím na tlačítko pro nahrání.



Jak nainstalovat sériový ovladač CH340 na MAC

Připojte hlavní řídicí desku k počítači pomocí kabelu USB a ovladač se automaticky nainstaluje na systémy MacOS a Windows. Pokud se instalace ovladače nezdaří, musíte ovladač nainstalovat ručně.

1) Stáhněte si ovladače

① Stáhněte si ovladač z webu a rozbalte jej do místního instalačního adresáře. Složku "[CH340 Driver file-mac](#)" naleznete v balíčku zdrojů, který jsme poskytli, toto je soubor ovladače, který chceme nainstalovat.



2) Připravte se na instalaci ovladače

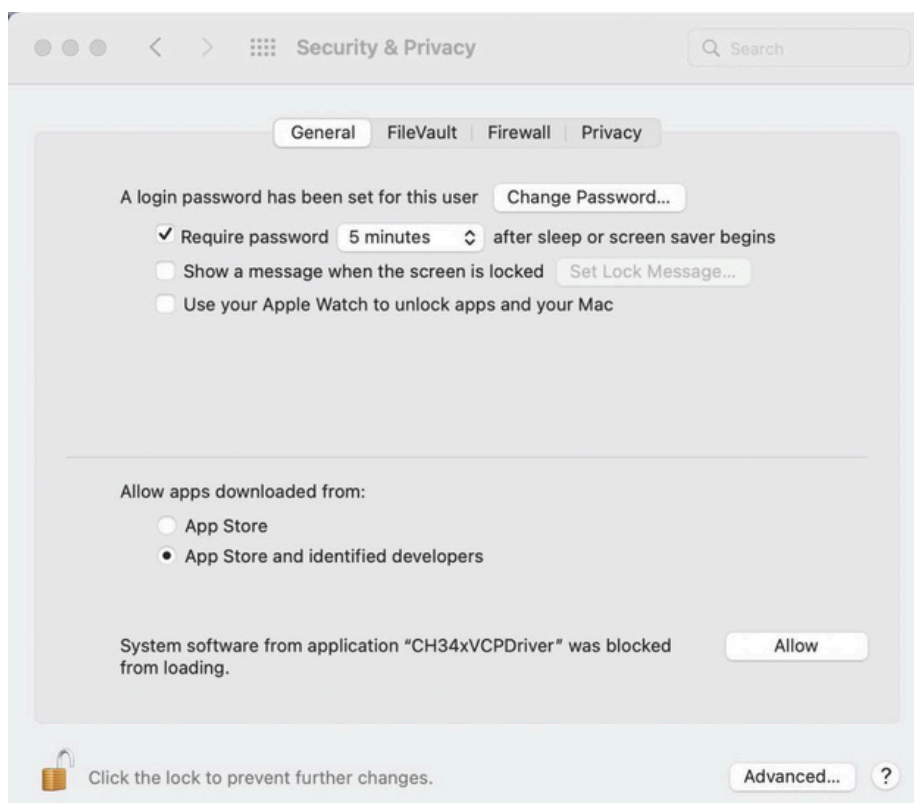
① Pokyny pro výchozí instalaci ovladače formátu pkg naleznete v dokumentu s názvem

„Installing the pkg Format Driver“.

② Pokud Rosetta není podporována v OS X 11.0 nebo novějším, viz "4. Instalace ovladače dmg."

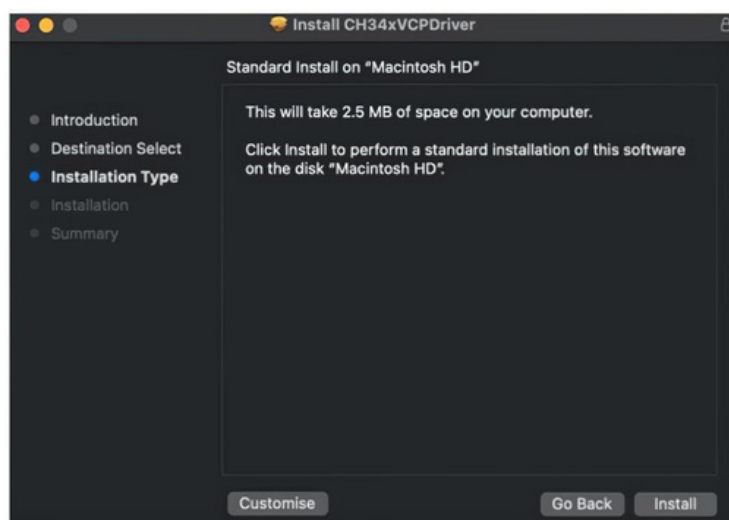
③ Před instalací přejděte na stránku "System Preferences" -> "Security and

Privacy" -> "General". V části s názvem „Allow apps downloaded from“ vyberte možnost „Mac App Store and identified developers“, abyste zajistili správnou funkčnost ovladače.

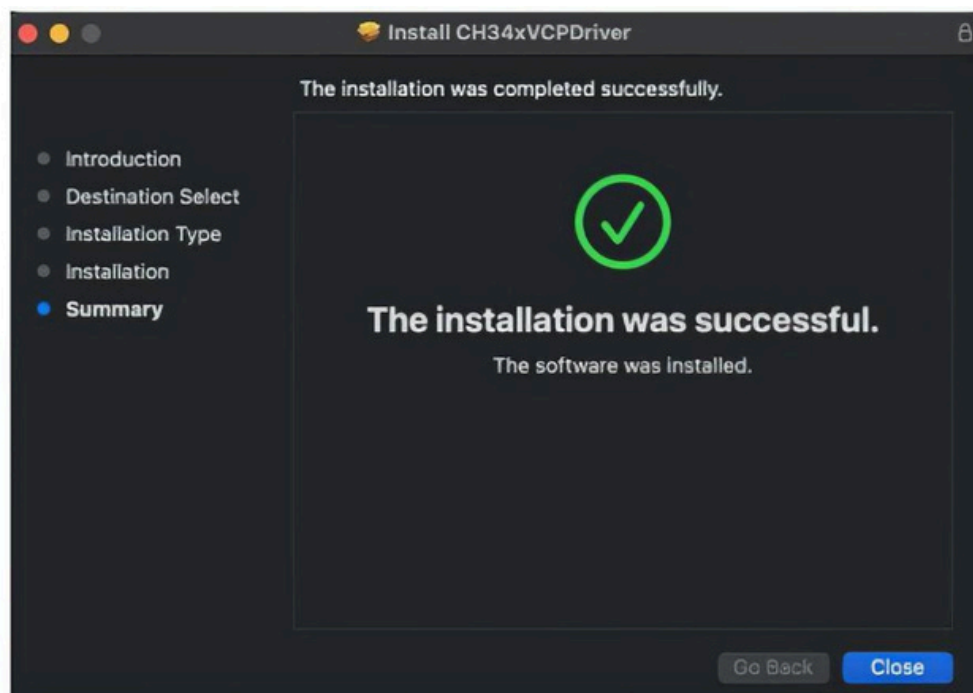
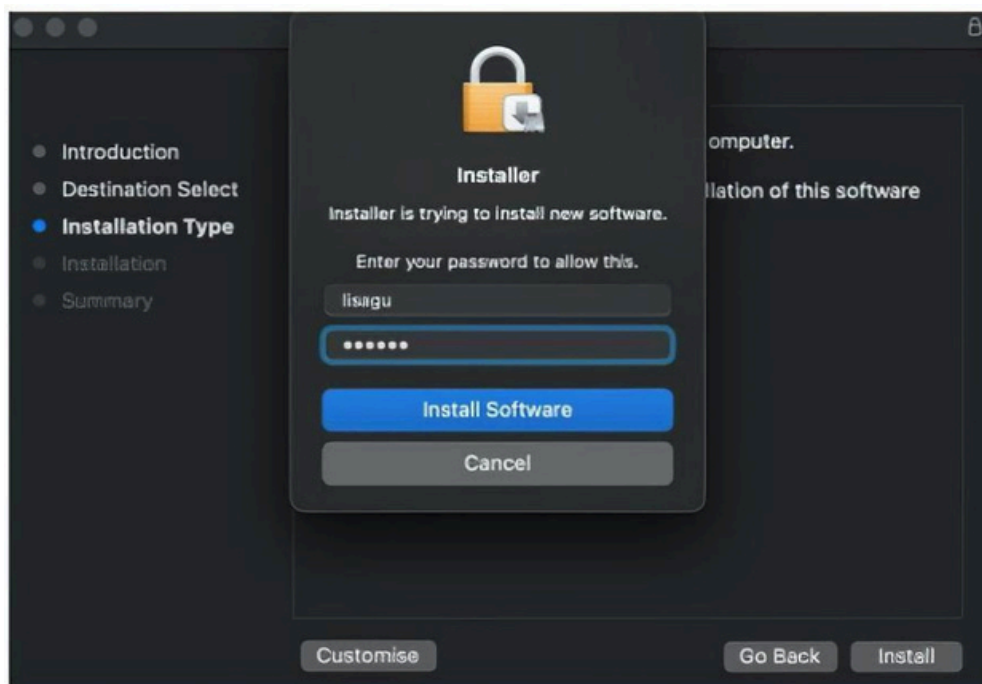


3) Nainstalujte ovladač formátu pkg

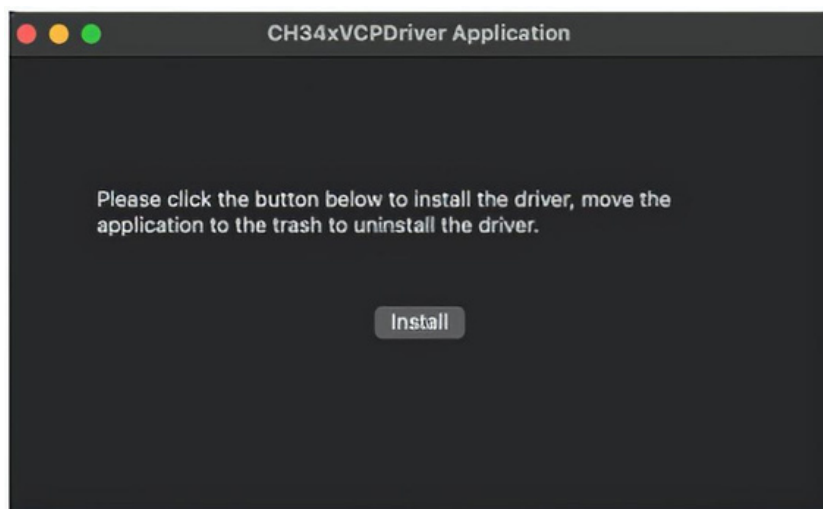
① Nainstalujte ovladač formátu pkg, klikněte na soubor ovladače -> Continue -> Install.



② Pak úspěšně nainstalujte



- ③ Nainstalujte ovladač formátu pkg na OS X 11.0 a novější: otevřete "LaunchPad" -> "CH34xVCPDriver" -> Install.

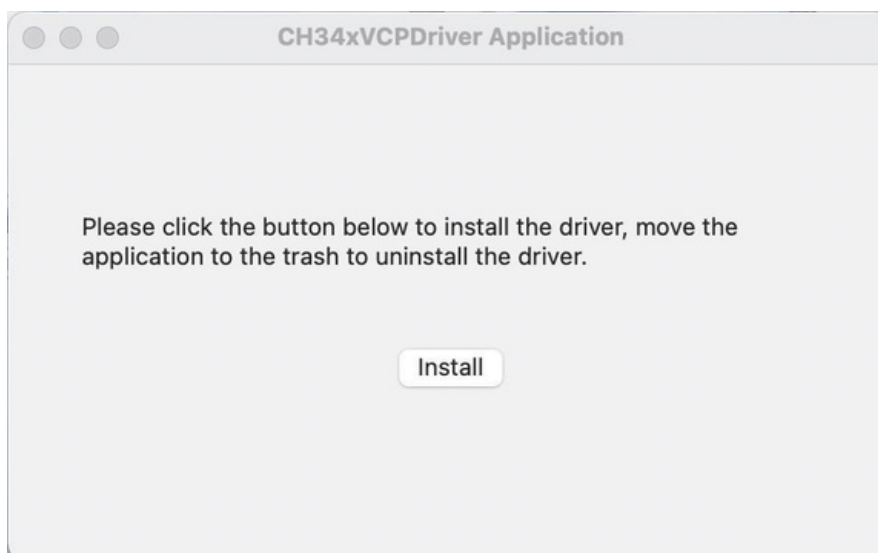


- ④ Pokud používáte OS X 10.9 až OS X 10.15, kliknutím na Restart restartujte počítač a po restartování proveďte následující kroky.

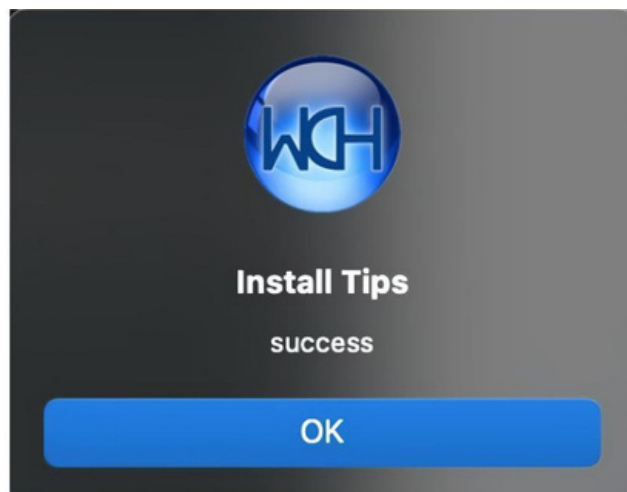


4) Nainstalujte ovladač formátu dmg

- ① Nainstalujte ovladač dmg, klikněte na soubor dmg a přetáhněte „CH34xVCPDriver“ do složky aplikace operačního systému.



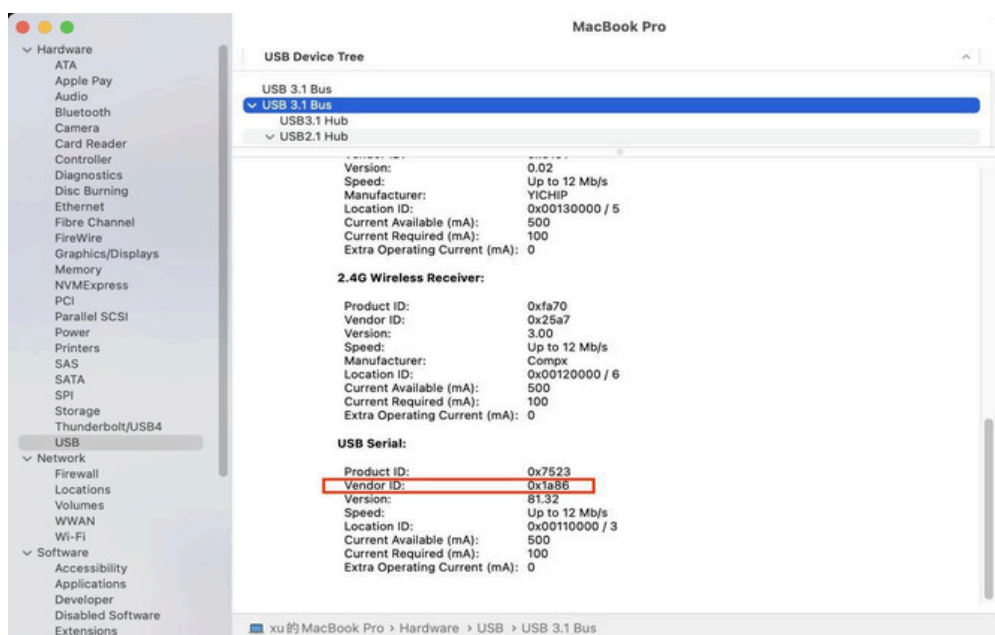
- ② Poté otevřete "LaunchPad" -> "CH34xVCPDriver" -> Install.
- ③ Nakonec se vám objeví obrazovka s nápisem „Install Tips succes“



5) Zkontrolujte, zda je nainstalován ovladač sériového portu CH340

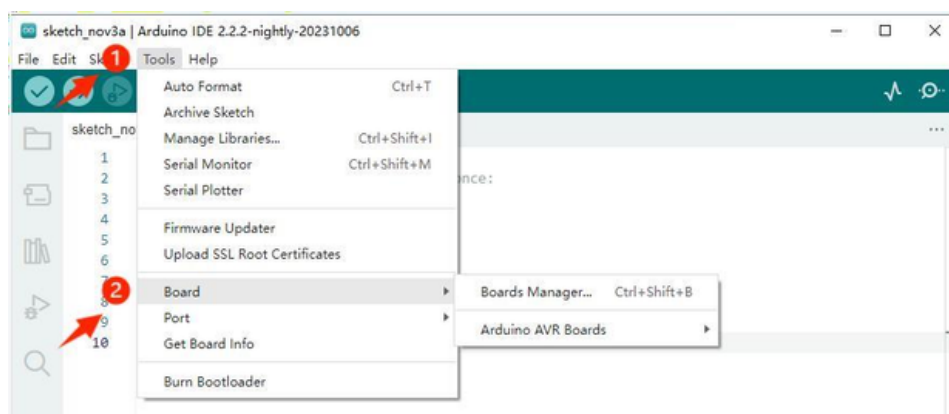
Po zasunutí řídicí desky do portu USB otevřete system report -> Hardware -> USB. Na pravé straně je USB zařízení.

Pokud zařízení USB funguje správně, můžete najít zařízení s "Vendor ID" [0x1a86].



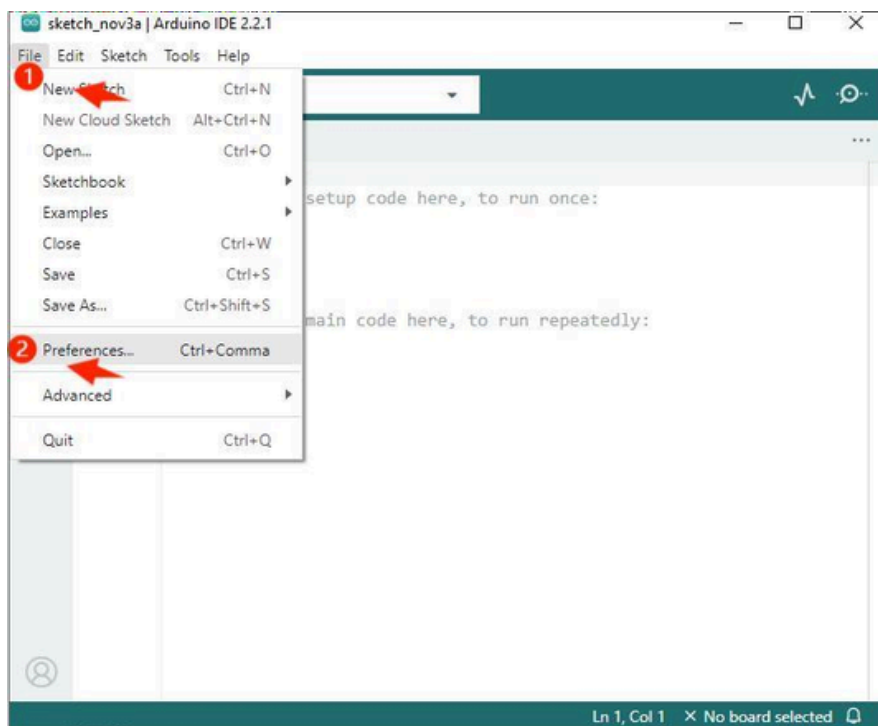
Instalace modulu ESP32 do Arduino IDE

Když otevřete Arduino IDE a vyberete Tools > Board, zjistíte, že v Arduino IDE je pouze Arduino AVR Board a žádný ESP32.

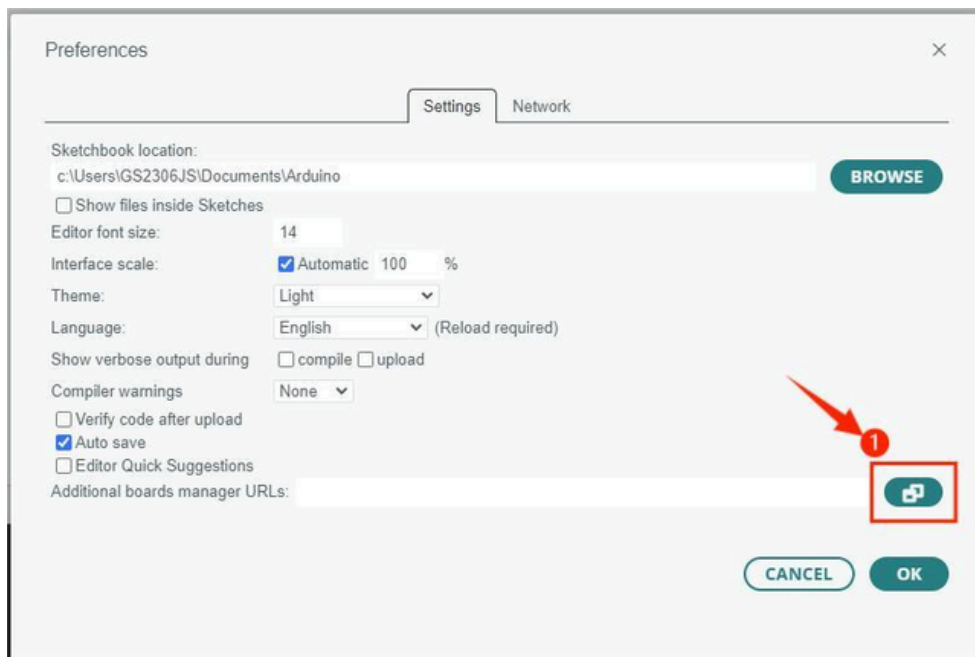


Ovládací deska, kterou tentokrát používáme, je ESP32 ovládací deska, takže musíme nainstalovat ESP32 desku do Arduino IDE. Postupujte podle následujících kroků:

1. Otevřít File > Preferences

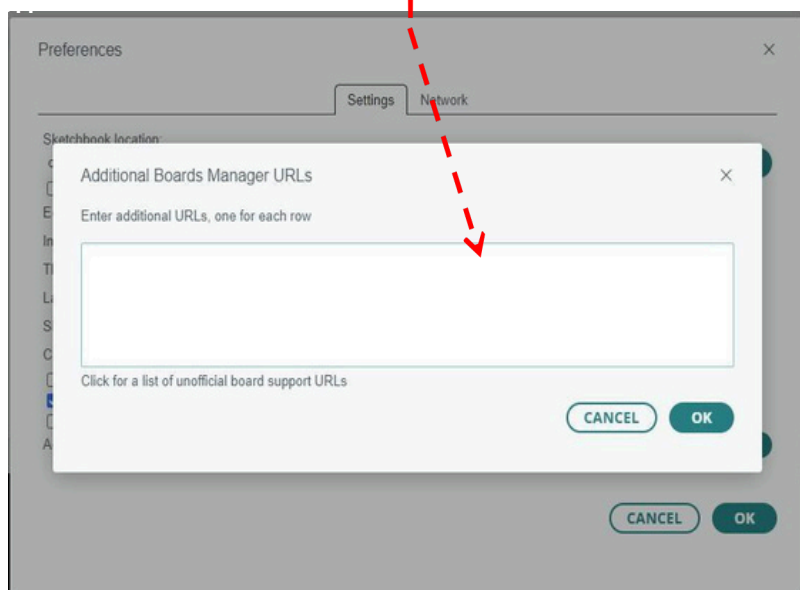


2. Přidat adresu správy vývojové rady URL

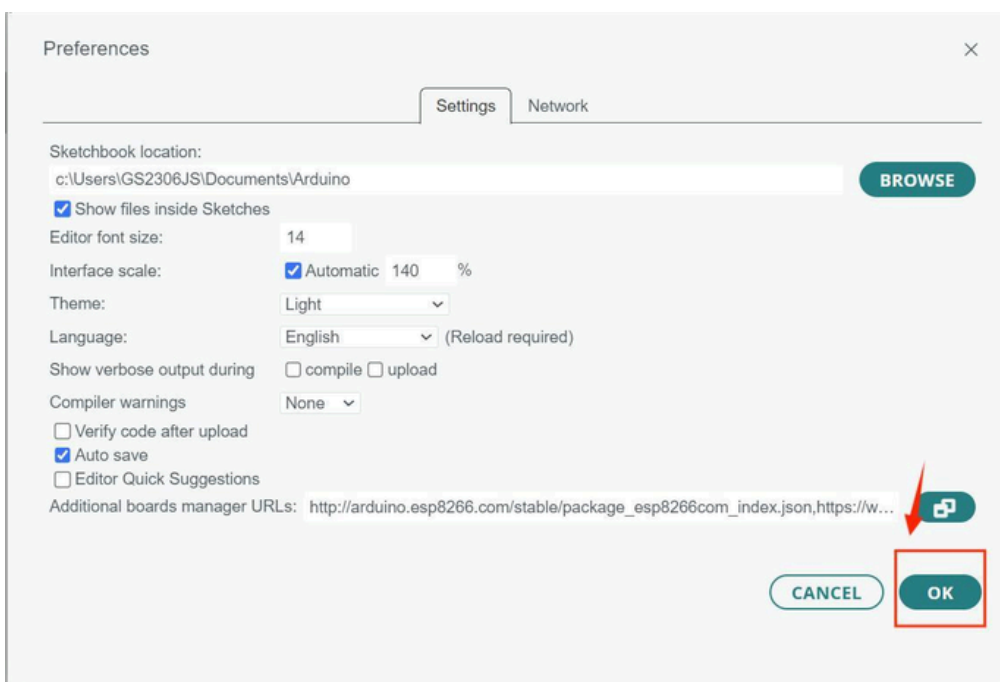
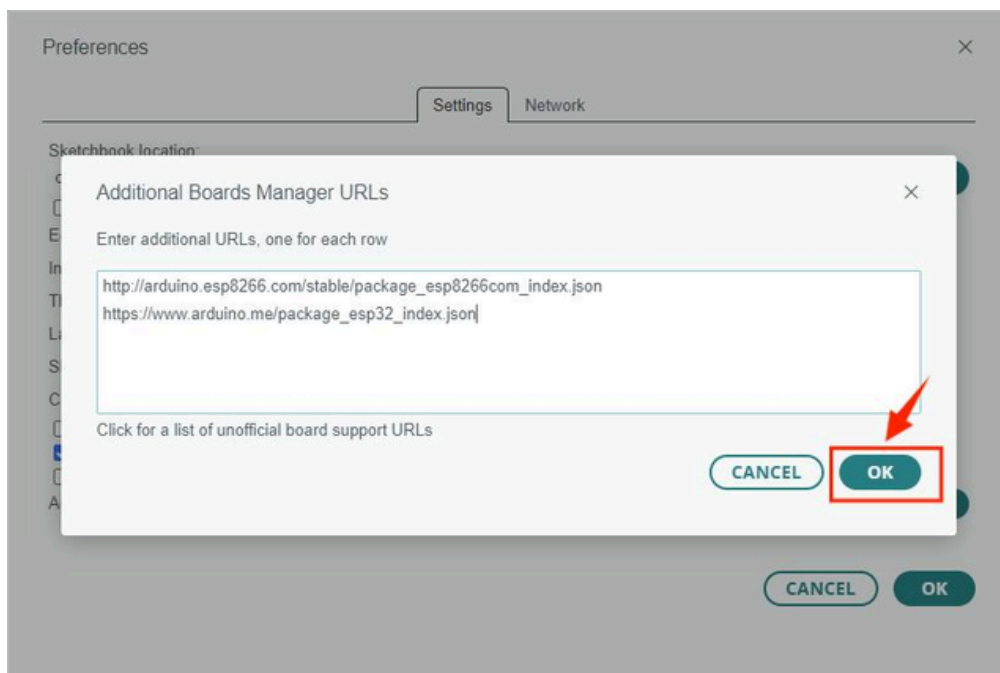


3. Zkopírujte adresu URL z textového pole níže a přidejte ji do "Additional Boards Manager URLs".

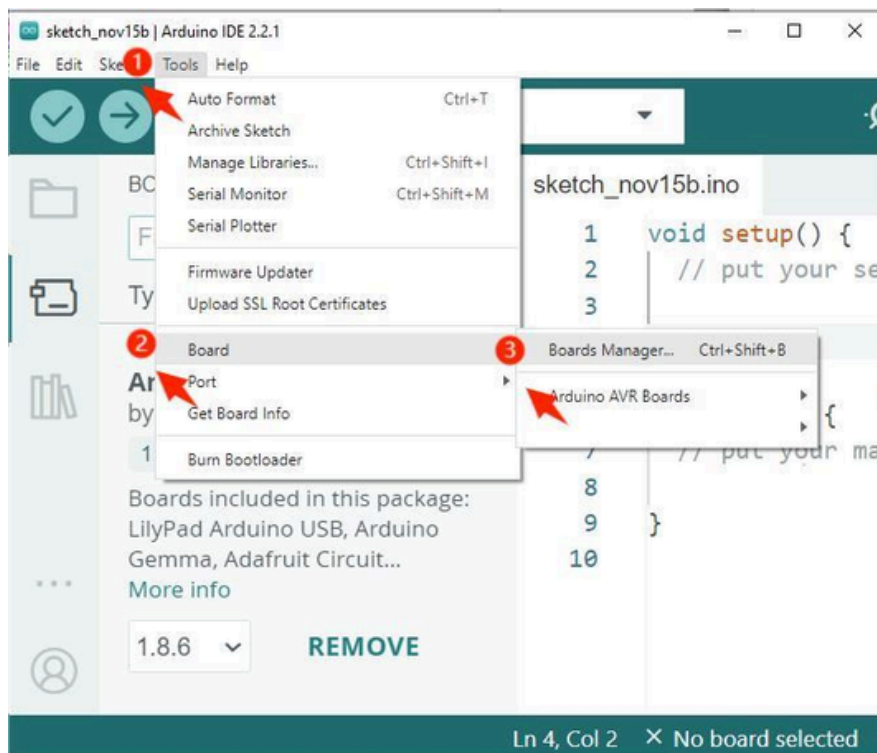
http://arduino.esp8266.com/stable/package_esp8266com_index.json
https://www.arduino.me/package_esp32_index.json



4. Po přidání adresy URL klikněte na "OK".

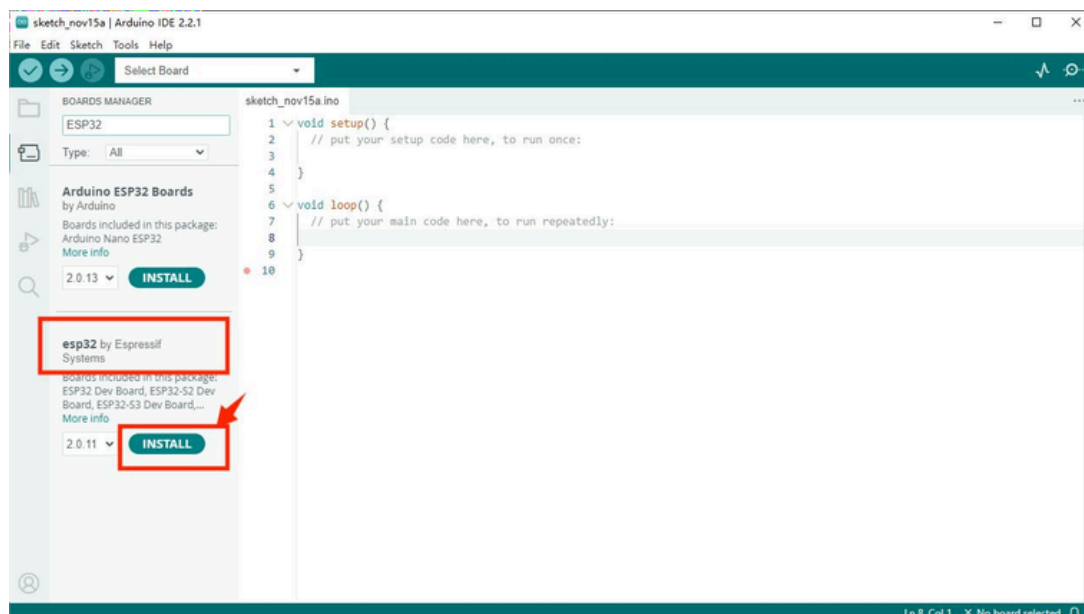


5. Klikněte Tools > Board > Boards Manager...

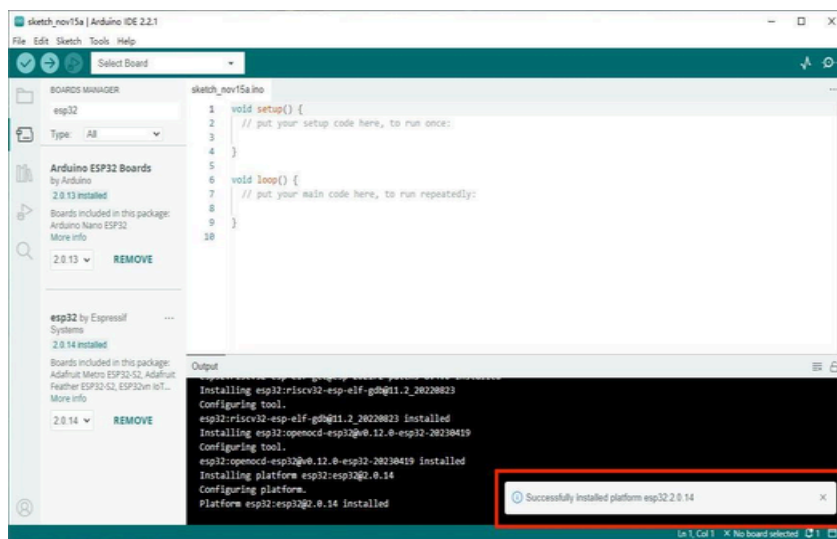


6. Vyhledejte "ESP32" ve vyhledávacím panelu BOARDS MANAGER a nainstalujte ho.

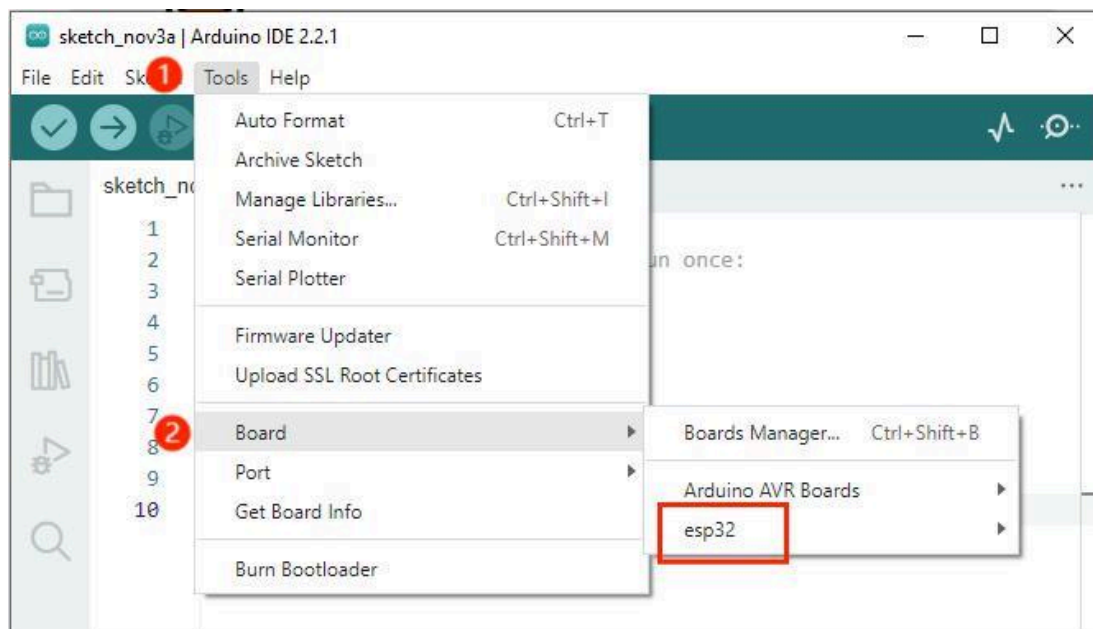
Upozornění: Nainstalujte verzi 2.0.12 esp32, protože nová verze není kompatibilní s knihovnou návodů a může způsobit chyby programu! Pokud byla verze 3.0 již nainstalována, odinstalujte a znovu nainstalujte verzi 2.0 esp32.



7. Počkejte, až bude instalace dokončena a zavřete Arduino IDE, když se zobrazí následující rozhraní.



8. Otevřete Arduino IDE znovu, vyberte Tools > Board a zjistíte, že se objeví deska esp32.

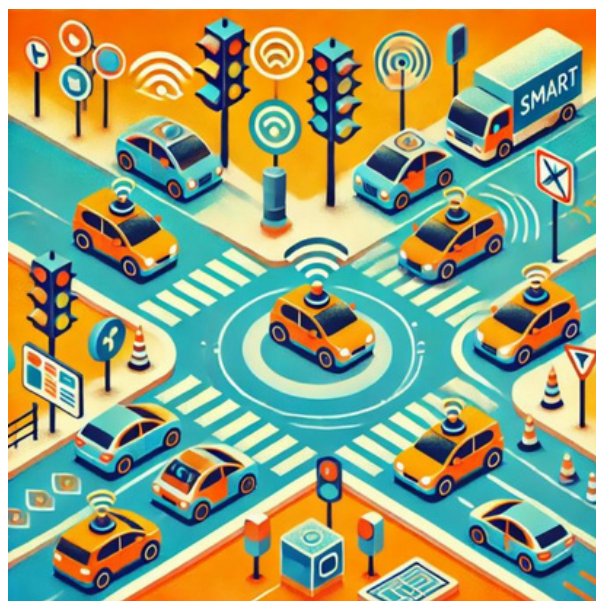




Kapitola 6: Autonomní řízení

Představ si den v budoucnosti, kdy nastoupíš do auta, řekneš mu, kam chceš jet, a pak si můžeš v klidu odpočívat, číst knihu, poslouchat hudbu nebo si dokonce zdřímnout, zatímco tě auto samo doveze do cíle. To už není scéna ze sci-fi filmu, ale technologie autonomního řízení, která se postupně stává realitou! Samořídící technologie mění způsob, jakým cestujeme, a činí dopravu bezpečnější, efektivnější a chytřejší.

V této kapitole společně prozkoumáme tajemství autonomního řízení, pochopíme jeho princip fungování, aplikační scénáře a budoucí vývojový trend. Pomocí praktických úloh si také vyzkoušíš zábavu autonomního řízení při ovládání inteligentního auta Sharkbot! Jsi připraven? Pojdme společně začít tuto úžasnou technologickou cestu!





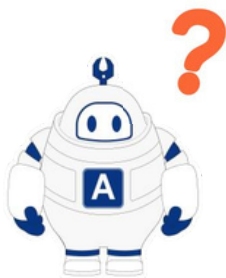
Oddíl 1: Přehled autonomního řízení



Cíle kurzu



-  Pochopit koncept autonomního řízení
-  Porozumět klíčovým technologiím autonomního řízení
-  Porozumět aplikacím autonomního řízení
-  Sestavit vozítko SharkBot



Viděl jsi někdy auto, které jezdí samo bez řidiče? Jak to vlastně dokáže?

Auto, které dokáže jezdit samo bez řidiče, je ve skutečnosti autonomní (samořídící) vozidlo, které využívá technologii autonomního řízení

Vznik samořídících aut učiní naše cestování pohodlnější a bezpečnější. Mohou snížit dopravní nehody způsobené chybami řidičů, zmírnit dopravní zácpy a dokonce usnadnit cestování starším lidem a osobám s omezenou pohyblivostí.





Tak co je vlastně technologie autonomního řízení? Jak umožňuje autům jezdit samostatně?

Technologie autonomního řízení



Technologie autonomního řízení označuje technologii, která umožňuje vozidlům vykonávat jízdní úkoly samostatně bez zásahu lidského řidiče. Pomocí senzorů, kamer, radarů a dalšího vybavení vnímá okolní prostředí a prostřednictvím počítačových systémů rozhoduje a řídí vozidlo, čímž umožňuje automatickou jízdu. Technologie autonomního řízení se obvykle dělí do šesti úrovní (L0–L5):

L0 (žádná automatizace): zcela závislé na lidském řidiči.

L1 (asistované řízení): vozidlo může pomáhat s některými jednoduchými úkoly, například autobraking systémem – ABS

L2 (částečně autonomní řízení): Vozidlo může současně ovládat volant i plynový pedál, ale řidič musí být neustále připraven převzít řízení.

L3 (podmíněný autopilot): Za určitých podmínek může vozidlo jezdit zcela automaticky, ale v případě potřeby vyzve řidiče k převzetí řízení.

L4 (vysoce autonomní řízení): Ve většině případů může vozidlo jezdit zcela automaticky bez zásahu člověka.





L5 (plně autonomní řízení): Vozidlo může jezdit zcela automaticky za jakýchkoli podmínek bez zásahu člověka.

Key Technologies for Autonomous Driving



Realizace technologie autonomního řízení závisí na společném působení několika klíčových technologií:

LiDAR (laserový radar): Vytváří vysoce přesné 3D mapy okolí pomocí vysílání laserových paprsků, které měří vzdálenosti.



Ultrazvukové senzory: Používají se pro detekci překážek na krátkou vzdálenost, často například v systémech automatického parkování.

Počítačové vidění: Pomocí technologie rozpoznávání obrazu identifikuje silnice, vozidla, chodce a další objekty. Například algoritmy hlubokého učení dokážou rozpoznávat dopravní značky a chování chodců.

Umělá inteligence: Využívá metody strojového učení k rozhodování a řízení. Například posilované učení pomáhá vozidlům dělat optimální rozhodnutí v složitých situacích.

Vysoce přesné mapy: Poskytují detailní informace o silnicích a pomáhají vozidlům s lokalizací a navigací. Přesnost těchto map může dosahovat až centimetrové úrovně.





Vehicle-to-Everything (V2X): Umožňuje komunikaci v reálném čase mezi vozidly navzájem (V2V) a mezi vozidly a infrastrukturou (V2I) pomocí 5G komunikační technologie, čímž zvyšuje efektivitu a bezpečnost dopravy.

Aplikace autonomního řízení



Technologie autonomního řízení má široké spektrum aplikačních scénářů a mezi typické příklady patří:

Městská doprava: Samořídící auta mohou jezdit autonomně v městském provozu, čímž snižují dopravní zácpy a počet nehod. Například samořídící taxi by mohla být dostupná 24 hodin denně a omezit zbytečné stání motorů.



Logistika a přeprava: Samořídící kamiony lze využít pro dálkovou nákladní dopravu a zvýšení efektivity přepravy. Například společnost Amazon testuje autonomní doručovací vozidla.

Sdílená mobilita: Autonomní sdílená vozidla mohou poskytovat pohodlnější dopravní služby a snížit potřebu vlastnit soukromé auto.

Speciální scénáře: V uzavřených oblastech, jako jsou doly, přístavy nebo zemědělská pole, mohou autonomní vozidla zvýšit efektivitu práce. Například autonomní traktory už jsou v některých farmách v provozu.





SharkBot



SharkBot je samořídící chytré auto navržené pro inteligentní dopravní systém. Je vybaveno čipem ESP32 S3 jako řídicí jednotkou a ultrazvukovými senzory, sledovacími (line tracking) senzory, modulem pro rozpoznávání hlasu, kódovacími motory, RGB LED osvětlením a dalšími moduly. Díky tomu dokáže realizovat více funkcí autonomního řízení.



ITS Laboratoř



Sestavování SharkBotu



Popis projektu



Sestav SharkBot auto podle postupového montážního schématu krok za krokem.

Note: The step-by-step construction files are in the ["English\Arduino\(Experienced Learner\)\8.Assemble documentation"](#) folder included with this tutorial.



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	Car PCB	1
	Ultrazvukový Senzor	1
	Kola	2
	Set akrylových plátů	1





2. Schéma hardwarové struktury





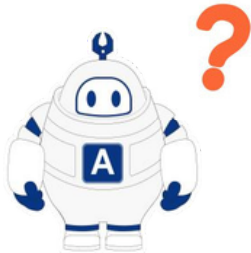
Oddíl 2: Základní pohyb chytrého auta



Cíle kurzu



-  Pochopit princip fungování motoru N20.
-  Zvládnout metody řízení motoru N20.
-  Zvládnout základní metody řízení pohybu auta SharkBot.
-  Naprogramovat SharkBot tak, aby se pohyboval ve tvaru čtverce.



Jak se vlastně chytré auto pohybuje? Odkud bere svou „sílu“?

Pohyb chytrého auta se neobejde bez jeho „stehen“ – motoru. Motor je zdrojem energie chytrého auta, který dokáže přeměnit elektrickou energii na mechanickou energii a roztáčet kola. Řízením směru a rychlosti otáčení motoru můžeme realizovat základní pohyby chytrého auta, jako je jízda vpřed, vzad, zatáčení vlevo, zatáčení vpravo a podobně. Aby se chytré auto pohybovalo podle našich pokynů, je potřeba pochopit princip fungování motoru a naučit se řídit jeho chování pomocí programování.





SharkBot používá motor N20. Dále společně prozkoumáme, jak řídit pohyb SharkBotu.

N20 motor



1.N20 motor seznámení

Motor N20 je druh miniaturního stejnosměrného motoru, jehož princip fungování je založen na elektromagnetické indukci. Když proud prochází cívkou motoru, vzniká magnetické pole, které interaguje s permanentními magnety uvnitř motoru a vytváří rotační sílu, která pohání motor k otáčení. Motor N20 je široce používán v malých zařízeních, jako jsou chytrá auta a roboti atd. Vyznačuje se malými rozměry, nízkou hmotností a nízkou spotřebou energie, což je velmi vhodné pro napájecí systém vzdělávacích chytrých aut. Motor N20 použitý v této lekci má navíc přidanou redukční převodovku, aby bylo možné zvýšit výstupní točivý moment motoru, a tím zvýšit nosnost auta. Jeho hlavní parametry jsou následující:



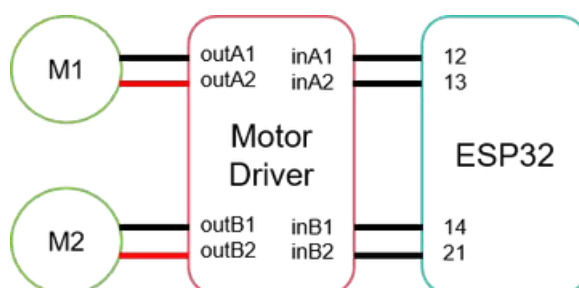
Parametry	Hodnoty
Jmenovité napětí	12V



Otáčky naprázdno	150 rpm
Jmenovitý točivý moment	3.0 kg.cm
Převodový poměr redukce	1:150

2. Řízení N20 Motoru

Motor N20 patří do kategorie stejnosměrných motorů a jeho rychlost a směr pohybu lze upravovat změnou proudu. Řídicí jednotka ESP32 však sama o sobě nemá schopnost přímo řídit stejnosměrné motory, a proto je nutné použít řídicí (driver) čip k nepřímému ovládání motoru a využít technologii PWM (pulzně šířková modulace) pro přesné řízení jeho chodu. Použitý ovladač motoru N20 na SharkBotu je DRV8833, který dokáže řídit dva stejnosměrné motory.



Zapojení pinů desky ovladače DRV8833 k ESP32 je uvedeno v následující tabulce:

N20 Motorový řadič	Piny řadiče	ESP32 Pins
Pravý motor	RightMotor_A1	Pin12
	RightMotor_A2	Pin13
Levý motor	LeftMotor_A1	Pin14
	LeftMotor_A2	Pin21





Změna rychlosti: Změnou velikosti proudu motoru lze měnit jeho rychlost.

Referenční program je následující:

```
analogWrite(RightMotor_A1, 0);  
analogWrite(RightMotor_A2, pwm); //pwm range 0-255
```

Změna směru: Směr motoru lze změnit změnou směru proudu v motoru.

Referenční program je následující:

```
analogWrite(RightMotor_A1, 0); //positive rotation  
analogWrite(RightMotor_A2, pwm);  
  
analogWrite(RightMotor_A1, pwm); //invert rotation  
analogWrite(RightMotor_A2, 0);
```

Základní řízení pohybu chytrých aut



Chytrá auta obvykle používají diferenciální řízení k realizaci řízení pohybu. Diferenciální řízení znamená ovládání rychlosti a směru levých a pravých kol, čímž se realizuje jízda vpřed, vzad, zatáčení vlevo, zatáčení vpravo a další pohyby chytrého auta.

Základní princip řízení pohybu:

Jízda vpřed: levá i pravá kola se otáčejí dopředu současně a stejnou rychlostí.

Referenční program je následující:





```
analogWrite(RightMotor_A1, 0); //Right motor turns forward
analogWrite(RightMotor_A2, pwm);

analogWrite(LeftMotor_A1, 0); //left motor turns forward
analogWrite(LeftMotor_A2, pwm);
```

Zpětný chod: levé i pravé kolo se současně otáčejí dozadu a stejnou rychlostí.

Referenční program je následující:

```
analogWrite(RightMotor_A1, pwm); //Right motor turns backward
analogWrite(RightMotor_A2, 0);

analogWrite(LeftMotor_A1, pwm); //left motor turns backward
analogWrite(LeftMotor_A2, 0);
```

Zatáčení vlevo: levé kolo se zastaví nebo se otáčí dozadu, zatímco pravé kolo se otáčí dopředu. Referenční program je následující:

```
analogWrite(RightMotor_A1, 0); //Right motor turns forward
analogWrite(RightMotor_A2, pwm);

analogWrite(LeftMotor_A1, pwm); //left motor turns backward
analogWrite(LeftMotor_A2, 0);
```

Zatáčení vpravo: pravé kolo se zastaví nebo se otáčí dozadu, zatímco levé kolo se otáčí dopředu. Referenční program je následující:





```
analogWrite(RightMotor_A1, pwm); //Right motor turns backward
analogWrite(RightMotor_A2, 0);

analogWrite(LeftMotor_A1, 0); //left motor turns forward
analogWrite(LeftMotor_A2, pwm);
```

Zastavení: pravé i levé kolo se zastaví. Referenční program je následující:

```
analogWrite(RightMotor_A1, 0); //right motor stops
analogWrite(RightMotor_A2, 0);

analogWrite(LeftMotor_A1, 0); //left motor stops
analogWrite(LeftMotor_A2, 0);
```



Tips

Při realizaci úlohy řízení jízdy auta v přímém směru může kvůli rozdílům ve výkonu motorů, odlišnostem tření kol o podklad a dalším faktorům docházet k tomu, že i když nastavíme stejnou rychlost pro oba motory, může se auto během jízdy odchylovat od předem určené dráhy. V takovém případě je nutné upravovat rychlost podle aktuální situace: pokud auto uhýbá doprava, zvýšíme rychlost pravého kola; pokud auto uhýbá doleva, zvýšíme rychlost levého kola.





ITS Laboratoř



Chytré auto se pohybuje ve tvaru čtverce



Popis projektu



Programování chytrého auta pro pohyb po čtvercové dráze ve směru hodinových ručiček.



Hardwarové schéma

1. Hardware List

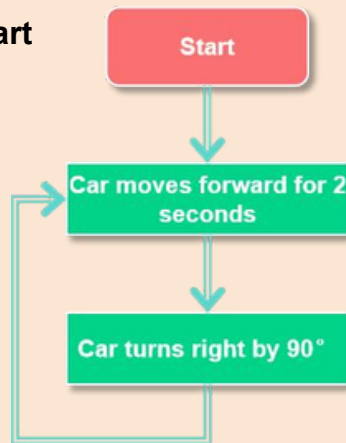
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programming

1. Program Flowchart



2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\01GoSquare](#)

```
#define RightMotor_A1 12 // Define the pin number of the motor driver
pin #define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21

void setup() {
  pinMode (RightMotor _A1,OUTPUT
T);    pinMode    (RightMotor
_A2,OUTPUT    T);    pinMode
(LeftMotor_    A1,OUTPUT    );
} pinMode (LeftMotor_ A2,OUTPUT

//Control motor function, pwm greater than 0, motor forward, pwm less than
0, motor backward, pwm equal to 0, motor stop
void motorControl(int left_pwm, int right_pwm)
{
  if(left_pwm > 0){
    analogWrite (LeftMotor_ A1, 0) ;
    analogWrite (LeftMotor_ A2,left_p wm );
```





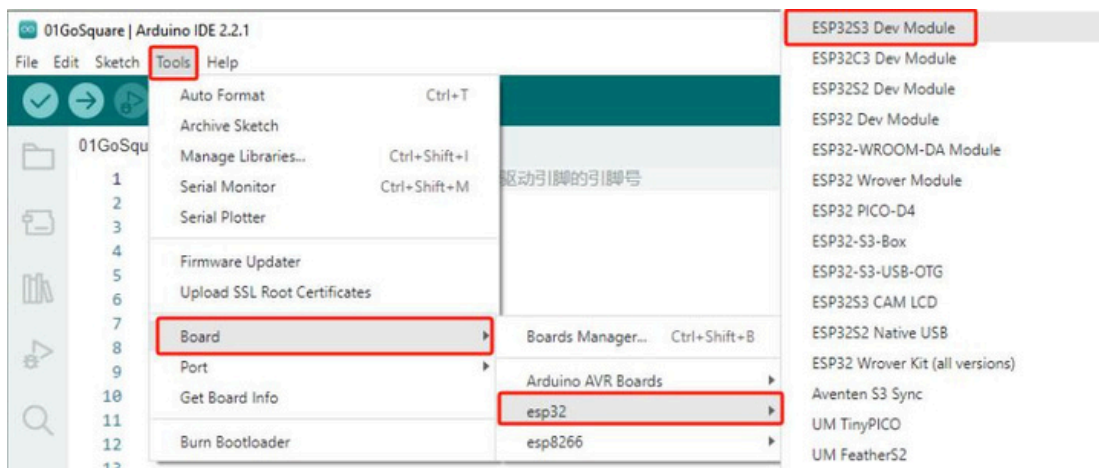
```
}  
else{  
    analogWrite (LeftMotor_A 1,-left_pwm);  
    analogWrite (LeftMotor_A 2,0) ;  
}  
if(right_pwm > 0){  
    analogWrite (RightMotor_ A1,0 );  
    analogWrite (RightMotor_ A2,right_pwm);  
}  
else{  
    analogWrite (RightMotor_ A1,-right_pwm);  
    analogWrite (RightMotor_ A2,0 );  
}  
delay(5) ;  
}  
  
void loop() {  
    motorControl(100,100);//The car moves forward for 2 seconds.  
    delay(2000 );  
    motorControl(100,-100);//The car turns right by 90 degrees.  
    delay(350 );  
}
```





Tipy

1. Před nahráním programu prosím vyber správný model základní desky „ESP32S3 Dev Module“.



2. Pokud potřebuješ upravit vzdálenost, kterou auto ujede vpřed, nebo upravit úhel, pod kterým auto zatáčí, můžeš změnit dobu, po kterou auto vykonává danou akci, jak je znázorněno na obrázku níže.

```
void loop() {  
  motorControl(100,100);  
  delay(2000);  
  motorControl(100,-100);  
  delay(350);  
}
```





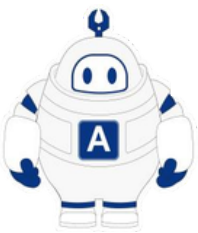
Oddíl 3: Osvětlovací systémy pro chytré auto



Cíle kurzu

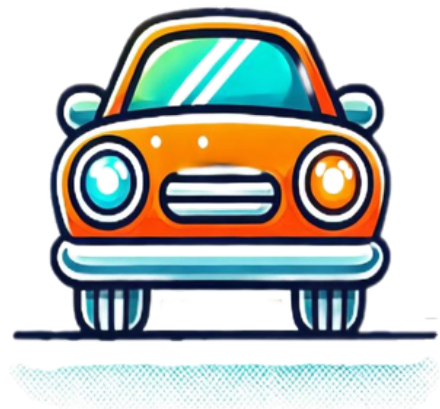


-  Pochopit princip fungování RGB světel
-  Pochopit princip tří základních barev světla
-  Zvládnout metody řízení RGB světel
-  Navrhnout osvětlovací systém pro chytré auto



Všiml sis někdy světel na autě? K čemu slouží?

Světla na autě neslouží jen k osvětlení, ale také hrají důležitou roli při signalizaci. Například červená brzdová světla upozorňují auto za vámi, aby zpomalilo, a oranžové směrovky dávají ostatním vozidlům najevo záměr odbočení. U chytrých aut hraje osvětlovací systém také důležitou roli. Pomocí změn světel můžeme vizualizovat stav pohybu chytrého auta, například jízdu vpřed, vzad nebo zatačení a podobně. Kromě toho lze světla využít také ke zlepšení vzhledu chytrého auta a dodat mu více technologický dojem.,





Osvětlovací systém na SharkBotu je tvořen světelným pásem WS2812 obsahujícím dvě RGB LED diody. Dále se naučíme princip fungování WS2812 a navrhne osvětlovací systém pro SharkBot.

Princip fungování RGB světla



LED pásek WS2812 se skládá z více RGB LED diod.

Každá RGB LED dioda je typ LED světla, které dokáže zobrazit více barev a obsahuje tři malé LED diody, které mohou vyzařovat červené, zelené a modré světlo.

Umožňuje míchat tyto tři barvy světla v různých jasech a poměrech, a tím vytvářet různé barvy.



commoncathode



commonanode

V počítačích se jas červené, zelené a modré barvy obvykle kvantifikuje v rozsahu 0–255, kde 0 znamená nejnižší jas a 255 nejvyšší jas. Když jsou hodnoty všech tří barev 0, světla jsou vypnutá, když jsou hodnoty všech tří barev 255, světlo svítí bíle.

U SharkBotu je pásek WS2812 připojen k pinu 48 na ESP32.



Ovládání RGB LED pásku



Každou RGB LED diodu na WS2812 lze ovládat nezávisle. V Arduinou ovládáme jejich činnost pomocí knihovny „Adafruit_NeoPixel.h“. Ovládací příkazy jsou následující:

```
//Create a light strip object and set the pin number of the light strip and the number of  
lamp beads:  
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN);  
//Lamp Bead Initialization:  
strip.begin();  
//Set the brightness of the light strip:  
strip.setBrightness(BRIGHTNESS);  
//Set the RGB value, each value ranges from 0-255:  
strip.Color(R,G,B)  
//Set the color of the light bulb at the specified position, i is the position of the light  
bulb:  
strip.setPixelColor(i,strip.Color(R,G,B));  
//Open lamp beads:  
strip.show()
```

Tři základní barvy světla



Princip fungování RGB tříbarevné lampy úzce souvisí s principem tří základních barev. Základní barvy jsou červená, zelená a modrá, což jsou tři základní barvy světla.





Pomocí jejich různých poměrů míchání lze vytvořit mnoho různých barev.

Například:

Červená + zelená = žlutá

Červená + modrá = fialová

Zelená + modrá = tyrkysová

Červená + zelená + modrá = bílá

Vztah mezi osvětlením a pohybem



U chytrých aut mohou světla sloužit nejen k dekoraci, ale také k přenosu informací. Pomocí změn světel můžeme vizualizovat stav pohybu chytrého auta. Například:

Vpřed: světlo svítí zeleně, což označuje, že vozidlo jede vpřed.

Zpátečka: světlo bliká červeně, což označuje, že vozidlo couvá.

Zatáčení vlevo: levé světlo svítí žlutě, což označuje, že vozidlo se chystá odbočit vlevo.

Zatáčení vpravo: pravé světlo svítí žlutě, což označuje, že vozidlo se chystá odbočit vpravo.

Zastavení: světlo svítí červeně, což označuje, že vozidlo zastavilo.

Kombinací světel a pohybových stavů mohou chytrá auta lépe komunikovat s ostatními vozidly a chodci a tím zvýšit bezpečnost jízdy.





ITS Laboratoř



Návrh osvětlovacího systému pro chytré auto



Popis projektu



Prostřednictvím programovaného řízení RGB tříbarevných světel se realizuje propojení světel a pohybového stavu chytrého auta.



Hardwarové schéma

1. Hardware List

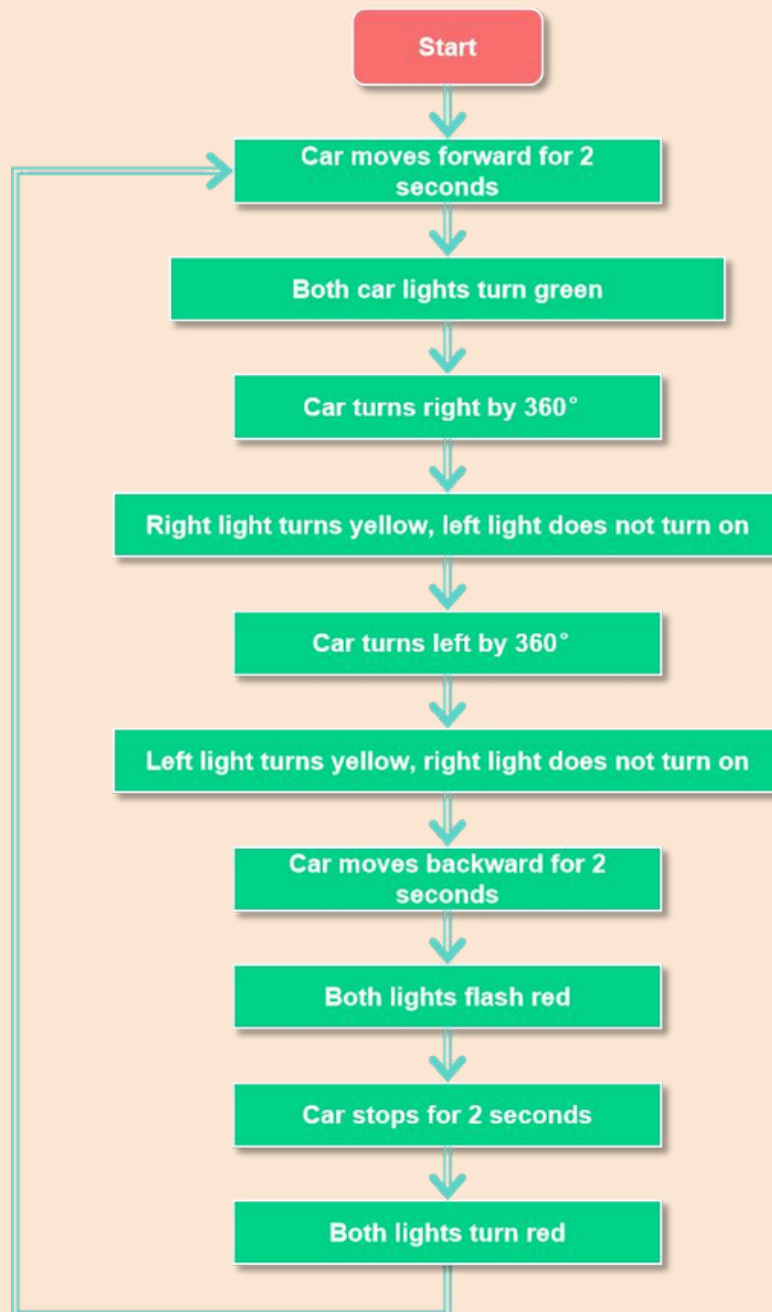
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

1. Program Flowchart





2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\02CarLight](#)

```
#include <Adafruit_NeoPixel.h> //Call the lamp strip library #define
RightMotor_A1 12 //Define pin numbers for motor drive pins #define
RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define LED_PIN 48 //Claim RGB lamp strip pins
#define LED_COUNT 2 //Declare the number of LEDs
#define BRIGHTNESS 50
#define RightLight0 //Declare the position of the lamp on the left
#define LeftLight1 //Declare the position of the lamp on the right

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN);
void setup() {
    pinMode (RightMotor_ A1,OUTPUT
    );    pinMode (RightMotor_
    A2,OUTPUT );    pinMode
    (LeftMotor_A 1,OUTPUT ); pinMode
    (LeftMotor_A 2,OUTPUT ); //Initialize the light strip
    strip.begin();    strip.
} setBrightness (BRIGHTNESS );

void loop() {
    motorControl(100,100);//car forward
    strip.setPixelColor(RightLight, strip.Color(0, 255, 0)); //green
    light strip.setPixelColor(LeftLight, strip.Color(0, 255, 0));
    //green light strip. show();
    delay( 2000);
    motorControl(100,-100);//Car, turn right.
    strip.setPixelColor(RightLight, strip.Color(255, 255, 0));
    strip.setPixelColor(LeftLight, strip.Color(0, 0, 0));
    strip. show();
    delay( 2000);
    motorControl(-100,100);//Car turn left.
```





```
strip.setPixelColor(RightLight, strip.Color(0, 0, 0));
strip.setPixelColor(LeftLight, strip.Color(255, 255, 0));
strip.show ();
delay(2000 );
motorControl(-100,-100);//car, back up.
for(int i = 0; i < 2; i++){//blinking red lamp
    strip.setPixelColor(RightLight, strip.Color(255, 0, 0));
    strip.setPixelColor(LeftLight, strip.Color(255, 0, 0));
    strip .show ();
    delay (500 );
    strip.setPixelColor(RightLight, strip.Color(0, 0, 0));
    strip.setPixelColor(LeftLight, strip.Color(0, 0, 0));
    strip .show ();
    delay (500 );
}
motorControl(0,0);//The small car stops
strip.setPixelColor(RightLight, strip.Color(255, 0, 0));//red light
strip.setPixelColor(LeftLight, strip.Color(255, 0, 0)); //red light
strip.show ();
delay(2000 );
}
/*
Motor control function: when the PWM value is greater than 0,
the motor moves forward; when the PWM value is less than 0,
the motor moves backward; when the PWM value is equal to 0, the motor stops.
*/
void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite( LeftMotor_A 1,0);
        analogWrite( LeftMotor_A 2,left_pwm );
    }
    else{
        analogWrite( LeftMotor_A 1,-left_pwm);
        analogWrite( LeftMotor_A 2,0);
    }
}
```





```
if(right_pwm > 0){  
    analogWrite (RightMotor _A1,0 );  
    analogWrite (RightMotor _A2,right_pwm) ;  
}  
else{  
    analogWrite (RightMotor _A1,-right_pwm );  
    analogWrite (RightMotor _A2,0 );  
}  
delay( 5);  
}
```



Kapitola 7: Inteligentní řízení

Kromě realizace automatického řízení se intelligence chytrých aut projevuje také ve schopnosti flexibilně reagovat na změny prostředí a realizovat inteligentní řízení. Inteligentní řízení je klíčovou schopností chytrých aut, která umožňuje vozidlu vnímat prostředí, analyzovat data a činit rozhodnutí, a tím dosáhnout bezpečnější, efektivnější a inteligentnější jízdy. Ať už jde o vyhýbání se překážkám, sledování čáry, řízení rychlosti nebo hlasovou interakci, to vše jsou konkrétní aplikace technologie inteligentního řízení. Tyto technologie nejen činí chytrá auta chytřejšími, ale také vytvářejí základ pro budoucí inteligentní dopravní systémy. Dále společně prozkoumáme technologii inteligentního řízení chytrých aut!





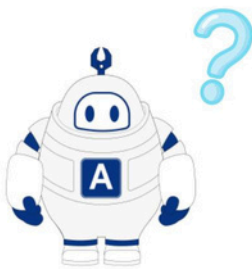
Oddíl 1: Inteligentní vyhýbání se překážkám



Cíle kurzu



-  Pochopit princip fungování ultrazvukového senzoru
-  Porozumět principu vyhýbání se překážkám pomocí ultrazvukového senzoru
-  Program to achieve automatic obstacle avoidance for SharkBot



How does a smart car avoid obstacles? How does it know when something is in the way?



IBěhem jízdy mohou chytrá auta narazit na různé překážky, jako jsou zdi, nábytek, chodci atd. Aby mohla jezdit bezpečně, musí mít schopnost automatického vyhýbání se překážkám. Automatické vyhýbání se překážkám znamená, že chytré auto pomocí senzorů vnímá okolní prostředí a při detekci překážky automaticky upraví trasu jízdy nebo zastaví.





Realizace této funkce vyžaduje použití senzorů pro vyhýbání se překážkám, které pomáhají chytrému autu „vidět“ překážky před sebou. Existuje mnoho způsobů, jak realizovat automatické vyhýbání se překážkám, mezi běžné metody patří: Ultrazvukové vyhýbání se překážkám: využívá ultrazvukové senzory k detekci vzdálenosti překážek před vozidlem.

Infračervené vyhýbání se překážkám: využívá infračervené senzory k detekci přítomnosti překážek.

Vizuální vyhýbání se překážkám: využívá kameru a technologii počítačového vidění k identifikaci překážek.

LiDAR vyhýbání se překážkám: využívá LiDAR k vytváření vysoce přesných map prostředí. V této lekci použijeme ultrazvukové senzory, které pomohou SharkBotu vyhýbat se překážkám. V této lekci prozkoumáme, jak mohou ultrazvukové senzory pomoci SharkBotu vyhýbat se překážkám.

Ultrazvukový senzor



Ultrazvukový senzor je zařízení, které používá ultrazvuk k měření vzdálenosti. Určuje vzdálenost k překážce tak, že vysílá ultrazvukové vlny, přijímá odražené vlny a vypočítává dobu jejich šíření. Funguje následovně:



- 1. Vysílání ultrazvukových vln:** Poté, co pin Trig přijme vysokou úroveň signálu trvající alespoň 10 μ s, senzor vyšle sérii ultrazvukových signálů s frekvencí přibližně 40 kHz.



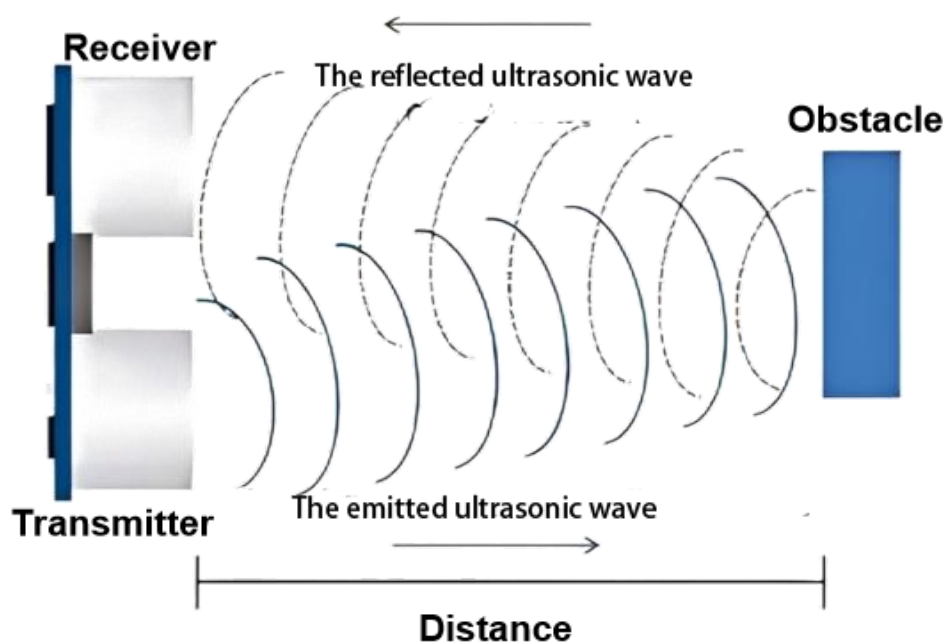


2. Přijímání odražených vln: Po vyslání ultrazvukových vln výstupní pin Echo ultrazvukového senzoru generuje signál vysoké úrovně. Pokud je před senzorem překážka, ultrazvukový signál se odrazí zpět. Když senzor přijme odražený ultrazvukový signál, pin Echo přejde na nízkou úroveň. Výpočtem délky trvání vysoké úrovně na pinu Echo můžeme určit dobu šíření ultrazvukových vln ve vzduchu.

3. Výpočet vzdálenosti: Na základě rychlosti šíření a doby šíření ultrazvukových vln se vypočítá vzdálenost k překážce. Vzorec je:

$$\text{Vzdálenost} = (\text{rychlost zvuku} \times \text{čas}) / 2$$

kde rychlost zvuku je ve vzduchu přibližně 340 m/s, děleno dvěma, protože ultrazvuk vykonává zpáteční (tam a zpět) cestu.





U SharkBotu je pin Trig ultrazvukového senzoru připojen k pinu 16 ESP32 a pin Echo je připojen k pinu 15 ESP32. Referenční program pro měření vzdálenosti pomocí ultrazvukového senzoru je následující:

```
// emitting ultrasonic waves
digitalWrite(Trig_PIN, LOW);
delayMicroseconds(2);
digitalWrite(Trig_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(Trig_PIN, LOW);

//Recover ultrasound and calculate the distance
float distance = pulseIn(Echo_PIN, HIGH) * 0.034 / 2.0;
```

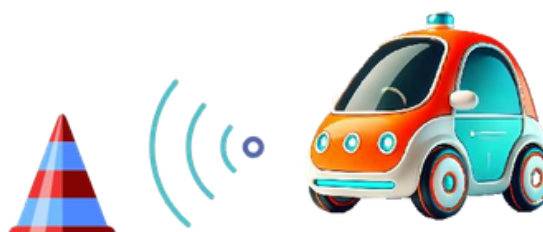
Princip ultrazvukového vyhýbání se překážkám



Princip ultrazvukového vyhýbání se překážkám spočívá v použití ultrazvukových senzorů k detekci

vzdálenosti překážek před vozidlem

a následné úpravě pohybového stavu



chytrého auta na základě výsledků měření. Konkrétní kroky jsou následující:

Detekce vzdálenosti: ultrazvukový senzor nepřetržitě monitoruje vzdálenost k překážce před vozidlem.

Vyhodnocení vzdálenosti: jakmile je zjištěná vzdálenost menší než přednastavená bezpečnostní hranice, je vyhodnoceno, že se před vozidlem nachází překážka.





Provedení vyhýbání: podle strategie vyhýbání může chytré auto zvolit zastavení, couvání nebo změnu směru jízdy.



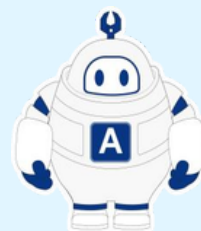
ITS Laboratoř



Automatické vyhýbání se překážkám u chytrých aut



Popis projektu



Funkce automatického vyhýbání se překážkám u chytrého auta je realizována pomocí programování.



Hardwarové schéma

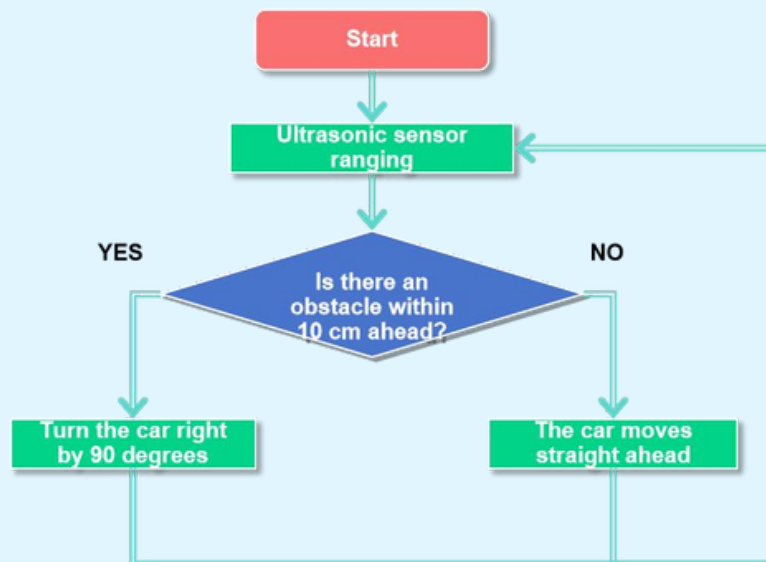
1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1



Programování

1. Program Flowchart



2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduino program\03AvoidanceObstacle](#)

```

#define RightMotor_A1 12
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define Trig_PIN 16 //Define the pins of the ultrasonic sensor
#define Echo_PIN 15
void setup() {
    pinMode(Trig_PIN, OUTPUT); // Set the Trig pin to output mode.
    pinMode(Echo_PIN, INPUT); // Set the Echo pin to input mode.
    pinMode(RightMotor_A1, OUTPUT); //
    pinMode(RightMotor_A2, OUTPUT);
    pinMode(LeftMotor_A1, OUTPUT);
    pinMode(LeftMotor_A2, OUTPUT);
  }

```





```
Serial .begin (115200 );
}

void loop() {
  // put your main code here, to run repeatedly:
  float distance = getDistance();
  if (distance < 10){
    motorControl(100,-100);//Turn the car right.
    delay (500);
  }
  else {
    motorControl (100,100 );
  }
}

float getDistance() { //Get the distance detected by the ultrasonic
sensor.
  digitalWrite(Trig_PIN, LOW); //Emit ultrasonic waves
  delayMicro seconds (2 );
  digitalWrite(Trig_PIN, HIGH);
  delayMicro seconds (10 );
  digitalWrite(Trig_PIN, LOW);
  float distance = pulseIn(Echo_PIN, HIGH) * 0.034 / 2.0;
  Serial .println (distance );
  return distance;
}

//Motor control function: when PWM > 0, the motor moves forward; when
PWM < 0, the motor moves backward; when PWM = 0, the motor stops.
void motorControl(int left_pwm, int right_pwm)
{
  if(left_pwm > 0){
    analogWrite( LeftMotor_A1, 0);
    analogWrite( LeftMotor_A2, left_pwm );
  }
  else {
    analogWrite( LeftMotor_A1, -left_pwm );
    analogWrite( LeftMotor_A2, 0);
  }
  if(right_pwm > 0){
    analogWrite( RightMotor_A1 ,0);
```





```
    analogWrite( RightMotor_A2, right_pwm );  
  }  
  else {  
    analogWrite( RightMotor_A1, -right_pwm );  
    analogWrite( RightMotor_A2, 0 );  
  }  
  delay (5);  
}
```

3. Popis instrukcí

<code>delayMicroseconds()</code>	Umožňuje programu běžet po stanovenou dobu zpoždění v mikrosekundách.
<code>pulseIn(pin,value)</code>	Čte délku pulzu na pinu (HIGH nebo LOW). Kde „pin“ je číslo pinu, na kterém se měří pulz, a „value“ je typ pulzu, který se má číst, přičemž může být HIGH nebo LOW.



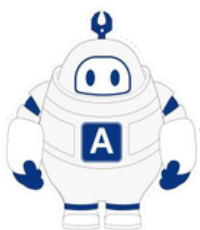
Oddíl 2: Systém proti kolizím s alarmem



Cíle kurzu



-  Pochopit princip fungování bzučáku
-  Pochopit aplikace bzučáku
-  Porozumět principu systému proti kolizím s alarmem
-  Navrhnout systém proti kolizím s alarmem pro chytré auto



Slyšel jsi někdy auto, které při couvání vydává „tikající“ zvuk? Jak to funguje?

V běžném životě, když auto couvá nebo se přibližuje k překážce, ozývá se „tik-tak“ alarm, který upozorňuje řidiče. Tento zvuk pochází ze systému varování před kolizí. Ve světě chytrých aut hrají systémy varování před kolizí také zásadní roli.



Pomáhají řidičům včas rozpoznat potenciální překážky před vozidlem, aby se zabránilo srážce, a zároveň zajišťují, že chodci nebo jiná vozidla mohou včas rozpoznat a vyhnout se chytrému autu.



Systémy proti kolizím s alarmem



Systém varování před kolizí je bezpečnostní systém, který pomocí senzorů detekuje překážky a při zjištění nebezpečí spustí alarm. Jeho hlavním cílem je zabránit tomu, aby zařízení narazilo do překážek, a tím předejít nehodám a poškození.

Pracovní postup systému varování před kolizí:

Fáze detekce: Systém používá senzory (např. ultrazvukové, infračervené, laserové atd.) k detekci okolního prostředí a získání informací o vzdálenosti překážek.

Fáze vyhodnocení: Řídicí jednotka (např. mikrokontrolér) analyzuje data ze senzorů, aby určila, zda existuje riziko kolize.

Fáze alarmu: Pokud existuje riziko, systém upozorní uživatele na potřebu akce pomocí zvuku (např. bzučák), světla (např. LED) nebo jiných prostředků.

V této lekci zkombinujeme ultrazvukové senzory, RGB světla a bzučák a navrhne systém varování před kolizí pro chytré auto.

Bzučák



Bzučák je druh elektronického zvukového zařízení, které dokáže přeměnit elektrické signály na zvukové signály. Bzučáky se dělí na aktivní a pasivní bzučáky.

Tato lekce používá pasivní bzučák. Pasivní bzučáky nemají vlastní





vestavěný oscilátor a vyžaduje externí signál k zajištění provozní frekvence. Obvykle obsahuje piezoelektrický keramický prvek, který se při přivedení střídavého napětí deformuje a vytváří zvuk. Pasivní bzučáky jsou široce používány v elektronických zařízeních, která vyžadují zvuková upozornění nebo alarmy, jako jsou zabezpečovací systémy, domácí spotřebiče, automobily, časovače a vestavěné systémy.

Metody řízení bzučáku



V Arduinu lze pasivní bzučák ovládat pomocí funkce `tone()`, která posílá bzučáku signál obdélníkového průběhu s pevnou frekvencí, aby vydával zvuk. Čím vyšší je frekvence signálu, tím vyšší je zvuk; čím nižší je frekvence, tím nižší je zvuk.

U SharkBotu je bzučák připojen k pinu 39 ESP32.

Referenční program pro ovládání bzučáku je následující:

```
//Controls the buzzer to send a square wave signal of 1000 Hz
tone(Buzzer_PIN, 1000);

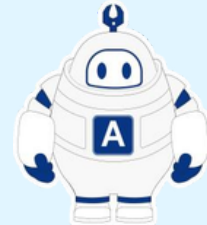
//Off Buzzer
noTone(Buzzer_PIN);
```



ITS Laboratoř



Systém varování před kolizí



Popis projektu

Prostřednictvím programování realizujte funkci varování před kolizí u chytrého auta. Při setkání s překážkou chytré auto vydá zvuk alarmu, rozsvítí červené světlo a poté se automaticky vyhne překážce.



Hardwarové schéma

1. Hardware List

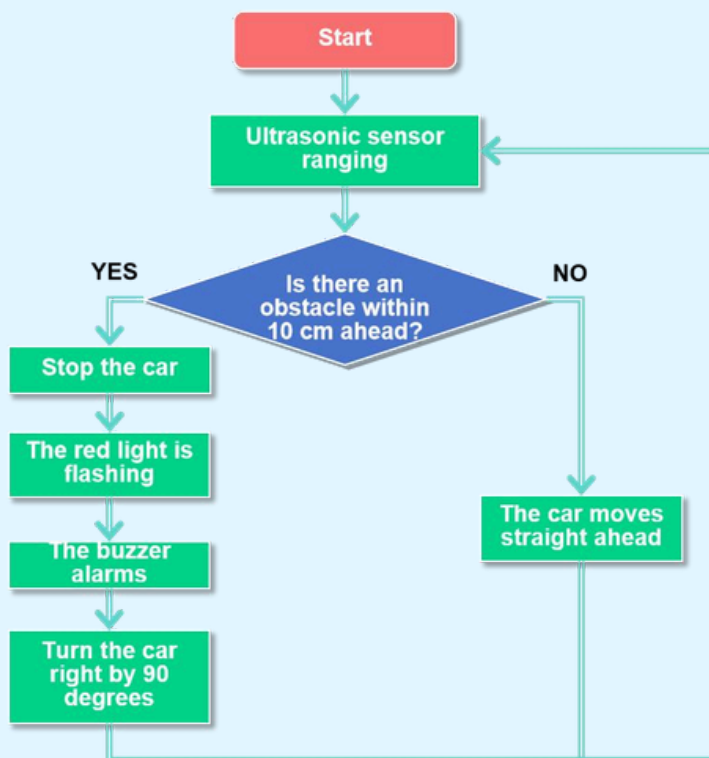
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

1. Program Flowchart



2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduino program\04ObstacleAlarmSystem](#)

```
#include <Adafruit_NeoPixel.h> //call the light strip program library

#define RightMotor_A1 12 // Define the pin number of the motor driver pin
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define Trig_PIN 16 //define ultrasonic sensor pin
#define Echo_PIN 15
#define Buzzer 39 //define Buzzer pin
```





```

#define LED_PIN 48 //Declare the pins of the RGB LED strip.
#define LED_COUNT 2
#define BRIGHTNESS 50

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN); //Create a light strip
object.
void setup() {
    pinMode(RightMotor_A1,OUTPUT); //Set the working mode of the input pins
of the driver board.
    pinMode (RightMotor _A2,OUTPUT );
    pinMode (LeftMotor_ A1,OUTPUT );
    pinMode (LeftMotor_ A2,OUTPUT );
    pinMode(Trig_PIN, OUTPUT); //Set the Trig pin as an output pin.
    pinMode(Echo_PIN, INPUT); // set echoPin to input mode
    pinMode (Buzzer,OUTPUT );
    strip.begin(); //Initialize the light strip.
    strip .setBrightness(BRIGHTNESS );
    delay (500);
}

void loop() {
    float distance = getDistance();
    if (distance < 10){
        motorControl(0 ,0);
        Alarm ();
        motorControl(100,-100); //Turn the car right.
        delay (500);
    }
    else {
        motorControl(100 ,100);
    }
}

void Alarm(){
    strip.setPixelColor(0, strip.Color(255, 0, 0));
    strip.setPixelColor(1, strip.Color(255, 0, 0));
    strip .show();
    for (int i = 1000; i <= 2000; i += 100) { // Increase the frequency
from 1000 Hz to 2000 Hz.

```





```
tone(Buzzer, i);
delay(100); // Last for 100 ms.
}
strip.setPixelColor(0, strip.Color(0, 0, 0));
strip.setPixelColor(1, strip.Color(0, 0, 0));
strip.show();
for (int i = 2000; i >= 1000; i -= 100) { // Decrease the frequency
from 2000 Hz to 1000 Hz.
    tone(Buzzer, i);
    delay(100);
}
noTone (Buzzer);
}

float getDistance() { //Obtain the distance measured by the ultrasonic
sensor.
    float distance;
    digitalWrite(Trig_PIN, LOW);
    delayMicro seconds (2);
    digitalWrite(Trig_PIN, HIGH);
    delayMicro seconds (10);
    digitalWrite(Trig_PIN, LOW);
    distance = pulseIn(Echo_PIN, HIGH)* 0.034 / 2.0;
    return distance;
}

//Motor control function: when PWM > 0, the motor moves forward; when
PWM < 0, the motor moves backward; when PWM = 0, the motor stops.
void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite (LeftMotor_ A1, 0);
        analogWrite (LeftMotor_ A2, left_p wm );
    }
    else {
        analogWrite (LeftMotor_ A1, -left_ pwm );
        analogWrite (LeftMotor_ A2, 0);
    }
}
```





```
if(right_pwm > 0){  
    analogWrite( RightMotor_A1 ,0);  
    analogWrite( RightMotor_A2 ,right_pwm);  
}  
else {  
    analogWrite( RightMotor_A1 ,-right_pwm);  
    analogWrite( RightMotor_A2 ,0);  
}  
delay (5);  
}
```





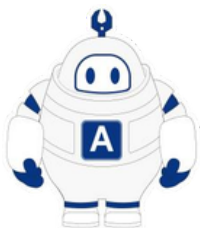
Oddíl 3: Hudba ve vozidle



Cíle kurzu



-  Pochopit princip vzniku zvuku
-  Pochopit složení hudby
-  Pochopit princip polí (arrays)
-  Naprogramovat auto SharkBot tak, aby přehrávalo hudbu



Chceš mít hudbu ve svém autě? Jak přimět SharkBot, aby přehrával hudbu

Hudba ve vozidle je důležitou součástí automobilu, která může pomocí uklidňujících melodií zmírnit únavu cestujících a vytvořit relaxační a příjemnou atmosféru během jízdy.

Bzučák na chytrém autě SharkBot můžeme využít k přehrávání hudby, takže SharkBot může při jízdě také hrát hudbu. V minulé lekci

jsme se již naučili, jak přimět bzučák vydávat alarm, ale jak docílit toho, aby bzučák přehrával hudbu?





Jak vzniká zvuk?



Základem hudby je zvuk. Abychom pochopili původ hudby, musíme nejprve pochopit, jak vzniká zvuk. Zvuk je vyvolán vibracemi objektu. Když objekt vibruje, vzduch, který je s ním v těsném kontaktu, také vibruje a vznikají zvukové vlny. Tyto zvukové vlny se šíří vzduchem a nakonec se dostanou do našich uší, což nám umožňuje vnímat zvuk.

Objekt, který vytváří zvuk, se nazývá zdroj zvuku, například hudební nástroj. Různé zdroje zvuku vytvářejí zvuky s různými frekvencemi a barvami tónu (timbrem) díky svým jedinečným vibračním



vzorům. Například strunné nástroje vytvářejí zvuk vibrací strun, dechové nástroje vytvářejí zvuk vibrací vzduchu v trubici a bicí nástroje vytvářejí zvuk úderem nebo nárazem na povrch.

Složení hudby



Hudba se skládá ze série tónů a rytmů. Tóny představují výšku zvuku a různé tóny odpovídají různým frekvencím zvuku; tóny v tónině C odpovídají následujícím frekvencím:





Note	C3(1)	D3(2)	E3(3)	F3(4)	G3(5)	A3(6)	B3(7)
Freq	131	147	165	175	196	221	248
Note	C4(1)	D4(2)	E4(3)	F4(4)	G4(5)	A4(6)	B4(7)
Freq	262	294	330	350	393	441	495
Note	C5(1)	D5(2)	E5(3)	F5(4)	G5(5)	A5(6)	B5(7)
Freq	525	589	661	700	786	882	990

Rytmus řídí časování tónů. Každá skladba v notovém zápisu má svůj vlastní rytmus, který udává, jak dlouho tón trvá. To je znázorněno následovně:

$\text{♩} = 100$
 $1 = \text{C } \frac{4}{4}$

TWINKLE TWINKLE LITTLE STAR

1	1	5	5	6	6	5	-	4	4	3	3	2	2	1	-
5	5	4	4	3	3	2	-	5	5	4	4	3	3	2	-
1	1	5	5	6	6	5	-	4	4	3	3	2	2	1	-

1. Běžná nota, například „1“, zabírá jeden takt.
2. Pokud je nota podtržená, např. „1“, znamená to, že nota zabírá půl taktu.
3. Pokud je za notou tečka, například „1.“, znamená to, že nota zabírá 1,5 taktu.
4. Pokud je za notou vodorovná čára, například „1 -“, znamená to, že nota zabírá 2 takty.

V závislosti na stylu skladby se délka jednoho taktu liší. Pokud v notovém zápisu vidíme $\text{♩} = 100$, což znamená 100 taktů za minutu, víme, že délka jednoho taktu je $60\,000 / 100 = 600$ ms.

Kombinace těchto tónů a rytmů vytváří melodii a rytmus, které jsou základními stavebními prvky hudby.





Pole (Array)



Pole (array) je datový typ používaný k ukládání více hodnot. Každý prvek pole má jedinečný index, který lze použít pro přístup k jednotlivým prvkům pole. Syntaxe pro definování pole je následující:

```
type názevPole[velikostPole];
```

kde „type“ je datový typ prvků v poli, „arrayName“ je název pole a „arraySize“ je velikost pole. Následuje příklad.

```
int num[5];
```

Výše uvedený kód definuje pole pěti celých čísel, ke kterým lze přistupovat pomocí num[0] až num[4].

Pole lze inicializovat již při jeho definování. Příklad:

```
int num[5] = {1, 2, 3, 4, 5};
```

Výše uvedený kód definuje pole pěti celých čísel a inicializuje ho hodnotami 1, 2, 3, 4, 5.

Pole lze také inicializovat následujícím způsobem:

```
int num[] = {1, 2, 3, 4, 5};
```

K prvkům pole lze přistupovat pomocí indexů. Indexy začínají od 0, např.:

```
int num[5] = {1, 2, 3, 4, 5};
```

```
int x = num[2];
```

Pole se často používají ve spojení s příkazem for, který pomocí smyčky prochází všechny prvky pole. Například:

```
int num[5] = {1, 2, 3, 4, 5};
```





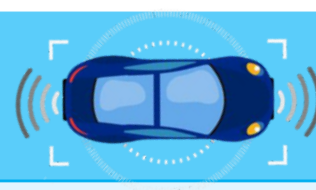
```
for (int i = 0; i < 5; i++) {  
    Serial.println(num[i]);  
}
```

Výše uvedený kód postupně přistupuje ke všem prvkům pole pomocí příkazu for.





ITS Laboratoř



Hudba ve vozidle



Popis projektu



Naprogramuj SharkBot tak, aby přehrál „Twinkle Twinkle Little Star“.

Twinkle Twinkle Little Star

J = 100

1=C 4/4

||: 1 1 5 5 | 6 6 5 — | 4 4 3 3 | 2 2 1 — |

Twin-kle, twin-kle, li - ttle star, How I won-der what you are ?



Hardwarové schéma

1.Hardware List

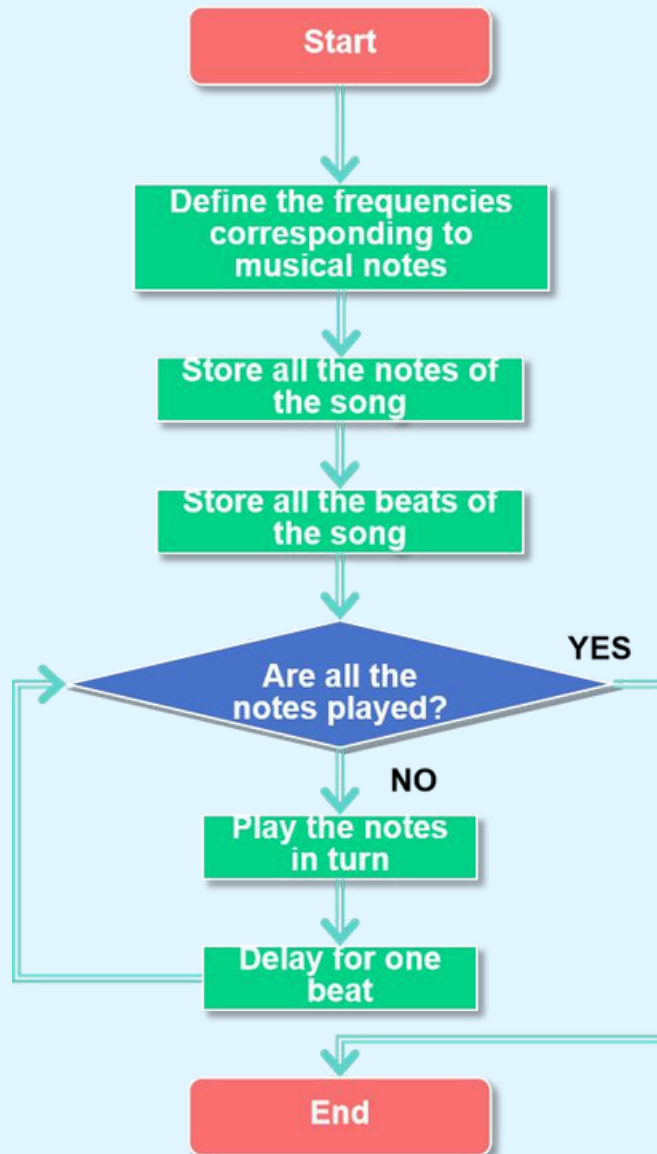
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

1. Program Flowchart





2.Progamování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických

materiálů přiložené k této učebnici::

English\Arduino(Experienced Learner)\3.Arduinoprogram\05LittleStar

```
#define Buzzer 39 //define Buzzer pin
//define the frequency of the note
#define C3 131
#define D3 147
#define E3 165
#define F3 175
#define G3 196
#define A3 221
#define B3 248

#define C4 262
#define D4 294
#define E4 330
#define F4 350
#define G4 393
#define A4 441
#define B4 495

#define C5 525
#define D5 589
#define E5 661
#define F5 700
#define G5 786
#define A5 882
#define B5 990
#define N 0

int length;
// littel star
int tune[] = { C4, N, C4, G4, N, G4, A4, N, A4, G4, N, F4, N, F4, E4,
N, E4, D4, N, D4, C4 };
float durt[] = { 0.99, 0.01, 1, 0.99, 0.01, 1, 0.99, 0.01, 1, 1.95, 0.05,
0.99, 0.01, 1, 0.99, 0.01, 1, 0.99, 0.01, 1, 2 };

void setup() {
```





```
length = sizeof(tune) / sizeof(tune[0]);  
}  
  
void loop() {  
  for(int x=0;x<length;x++) {  
    tone(Buzzer, tune[x]);  
    delay(500 * durt[x]);  
    noTone( Buzzer);  
  }  
  while( 1);  
}
```





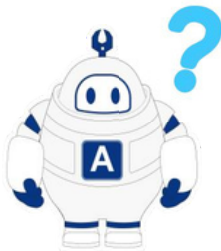
Oddíl 4: Inteligentní sledování čáry (I)



Cíle kurzu



-  Pochopit princip fungování snímače šedé (grayscale)
-  Porozumět algoritmu sledování čáry pomocí dvoukanálového snímače šedé
-  Naprogramovat realizaci jednoduché funkce sledování čáry pro chytré auto.



Jak přimět chytré auto, aby sledovalo pevně danou trasu

IV některých automatizovaných scénářích, jako jsou logistická vozidla nebo chytré autobusy ve výrobních halách, je často nutné jezdit po pevně dané trase. Tato funkce se nazývá sledování čáry. Systém sledování čáry u chytrého auta pomáhá vozidlu „vidět“ značení na zemi (např. černé čáry) a pomocí řídicího algoritmu upravovat směr jízdy tak, aby auto vždy jelo po vyznačené trase.





Chytrá auta realizují funkci sledování čáry především pomocí různých senzorů a řídicích algoritmů, které umožňují rozpoznání a sledování předem dané trasy. Mezi běžné senzory pro tuto funkci patří: snímače šedé (grayscale), infračervené senzory, magnetické senzory, LiDAR, kamerové (vizuální) senzory atd. V této lekci použijeme snímače šedé k realizaci sledování čáry u chytrého auta.

Snímač šedé



Snímač šedé (grayscale sensor) určuje úroveň šedi povrchu tak, že na něj vysílá světlo a měří intenzitu odraženého světla. Oblasti s vyšší úrovní šedi odrážejí světlo silněji, zatímco oblasti s nižší úrovní šedi odrážejí světlo méně. Intenzita světelného signálu přijatého senzorem se převádí na elektrický signál, který umožňuje určit změny jasu povrchu. Snímače šedé se široce používají v automatizačních řídicích systémech, sledování robotů, navigaci AGV, rozpoznávání barev a monitorování prostředí. Při sledování čáry u chytrého auta mohou snímače šedé detekovat rozdíl v intenzitě odraženého světla mezi černým a světlým povrchem, a tím navádět chytré auto po trase.

Algoritmus sledování čáry dvoukanálovým snímačem



SharkBot má čtyři snímače šedé (grayscale senzory), v této lekci budeme používat prostřední dva snímače pro sledování čáry a návrh jeho algoritmu je založen na následující logice.





1. Layout senzoru

Nainstalujte dva snímače šedé symetricky uprostřed přední části SmartBotu, přímo nad černou čarou. U SharkBotu je z těchto dvou centrálních senzorů levý připojen k pinu 5 ESP32 a pravý k pinu 6 ESP32.



1. Nastavení prahové hodnoty snímače šedé

Prahová hodnota snímače šedé je klíčový parametr pro rozlišení černé čáry a bílého podkladu. Prahová hodnota se nastavuje následovně:

(1) Změřte hodnotu šedi černé čáry: Umístěte senzor na černou čáru a zaznamenejte hodnotu šedi jeho výstupu (např. hodnota šedi černé čáry je 3000).

(2) Změřte hodnotu šedi bílého podkladu: Umístěte senzor na bílý povrch a zaznamenejte hodnotu šedi jeho výstupu (např. hodnota šedi bílého podkladu je 50).

(3) Výpočet prahové hodnoty: Obvykle se vezme střední hodnota mezi hodnotou černé čáry a bílého podkladu jako prahová hodnota (například: práh = $(3000 + 50) / 2 = 1525$).

(4) Aplikace prahové hodnoty: Nastavte prahovou hodnotu v programu, která se používá k určení



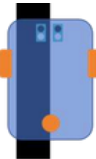


zda senzor detekuje černou čáru. Pokud je hodnota šedi ze senzoru nižší než prahová hodnota, považuje se to za detekci černé čáry; v opačném případě se považuje za detekci bílého podkladu.

V Arduino je možné měřit hodnotu snímače šedé pomocí sériového monitoru, referenční program je následující:

```
//read the value of the SharkBot's second grayscale sensor  
int sensor_value = analogRead(TrackSensor2);  
  
//Serial output of the sensor's value  
Serial.println( sensor_value);
```

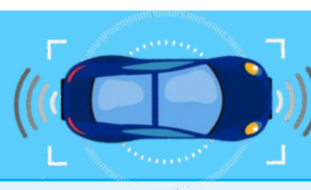
3. Algoritmus sledování čáry u chytrého auta:

Sensorstatus	Description	Smart Car Movement
	Levý senzor detekuje bílou oblast a pravý senzor detekuje černou čáru..	Zatočit vpravo
	Pravý senzor detekuje bílou oblast a levý senzor detekuje černou čáru.	Zatočit vlevo
	Oba senzory detekují černou čáru.	Jet rovně
	Žádný ze senzorů nedetekuje černou čáru.	Na základě jednotlivých případů / podle konkrétní situace





ITS Laboratoř



Intelligentní sledování čáry (I)



Popis projektu

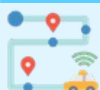


Programováním se realizuje, aby chytré auto sledovalo černou trasu na mapě. Mapa je zobrazena níže:



Poznámka: mapu je potřeba vytisknout samostatně, soubor k tisku je přiložen ve složce s tutoriálem.

["English\Arduino\(Experienced Learner\)\5.M ap\Map1\(60x90cm\).pdf"](#).



Hardwarové schéma

1.Hardware List

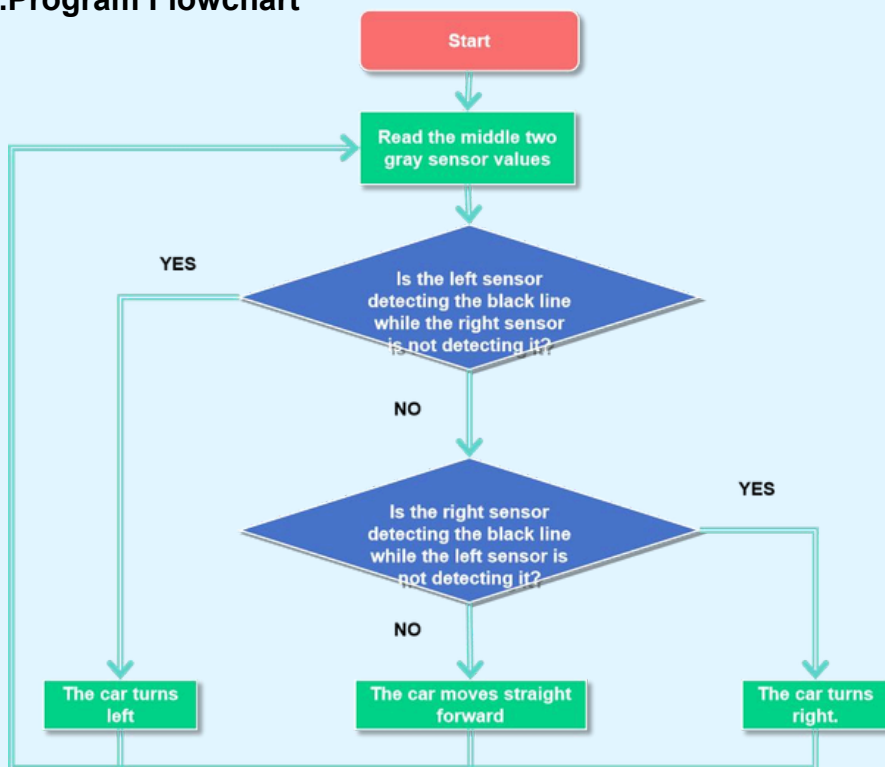
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1
	Mapa	1





Programování

1. Program Flowchart



2 Coding

The source code for this lesson can be found and opened in the electronic materials folder included with this textbook:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\06LineTracing1](#)

```
#define RightMotor_A1 12
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define TrackSensor2 5 // Define the pins for the grayscale sensor
#define TrackSensor3 6
int left_sensor_value;
int right_sensor_value;
void setup() {
    pinMode( RightMotor_A1 ,OUTPUT );
    pinMode( RightMotor_A2 ,OUTPUT );
```





```

pinMode (LeftMotor_ A1,OUTPUT );
pinMode (LeftMotor_ A2,OUTPUT );
pinMode(TrackSensor2, INPUT);//Set the pins of the grayscale sensor
to input mode
pinMode(TrackSensor3, INPUT);
}
void loop() {
    left_sensor_value = analogRead(TrackSensor2);//Read gray sensor
values
    right_sensor_value = analogRead(TrackSensor3);
    if(left_sensor_value > 1500 && right_sensor_value < 1500){
        motorControl(-100,100);//Black line detected on the left, no black
line detected on the right, left.
    }
    else if(left_sensor_value < 1500 && right_sensor_value > 1500){
        motorControl(100,-100);//Black line detected on the left, no black
line detected on the right, turn right.
    }
    else {
        motorControl (100 ,100);
    }
}
//Control motor function, pwm greater than 0, motor forward, pwm less
than 0, motor backward, pwm equal to 0, motor stop
void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite (LeftMotor_ A1, 0);
        analogWrite (LeftMotor_ A2,left_p wm );
    }
    else {
        analogWrite (LeftMotor_ A1,-left_ pwm );
        analogWrite (LeftMotor_ A2, 0);
    }
    if(right_pwm > 0){
        analogWrite (RightMotor _A1, 0);
        analogWrite (RightMotor _A2,right _pwm) ;
    }
}

```





```
else {  
    analogWrite (RightMotor _A1,-right_pwm );  
    analogWrite (RightMotor _A2, 0);  
}  
delay (5);  
}
```



Upozornění

Pokud auto během sledování čáry nefunguje správně, může to být způsobeno následujícími důvody:

- 1. Auto je příliš pomalé.**
- 2. Pro vyřešení problému můžete v programu zvýšit hodnotu PWM motoru, aby vozidlo mělo dostatečnou rychlost.**
- 3. ESP32 může kvůli rušení okolního světla nesprávně vyhodnotit bílou oblast jako černou čáru.**
- 4. Pro vyřešení tohoto problému nejprve otestujte konkrétní hodnoty snímače šedé na bílé ploše a černé čáře a podle toho upravte prahovou hodnotu v programu.**

prahová hodnota = (hodnota černé čáry + hodnota bílé plochy) / 2

```
if(left_sensor_value > 1500 && right_sensor_value < 1500){
```





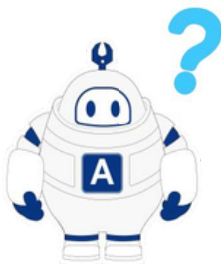
Oddíl 5: Inteligentní sledování čáry (II)



Cíle kurzu

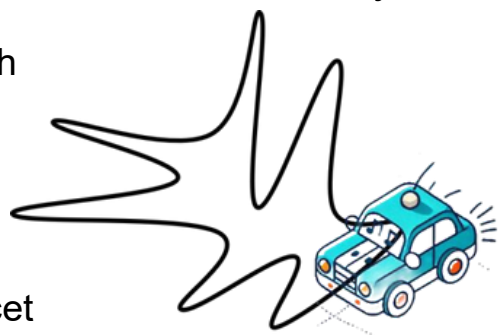


-  Pochopit algoritmus sledování čáry pomocí čtyřkanálového snímače šedé (grayscale senzoru)
-  Naprogramovat a implementovat pokročilou funkci sledování čáry pro chytré auto.



If the route becomes complicated (e.g., curves, intersections), do you think the 2-way sensor will still be able to handle it?

V minulé lekci jsme použili dvoukanálový snímač šedé k realizaci jednoduché funkce sledování čáry, ale v reálných situacích mohou trasy obsahovat ostré zatáčky, křižovatky a další složité situace. Abychom těmto výzvám lépe čelili, je potřeba zvýšit počet senzorů pro zlepšení přesnosti detekce a řídicí schopnosti systému sledování čáry.



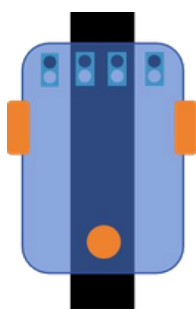


Algoritmus sledování čáry čtyřkanálovým snímačem



1. Layout sensoru

Čtyřkanálový snímač šedé se skládá ze čtyř nezávislých snímačů šedé, které jsou obvykle umístěny v přední spodní části chytrého auta v jedné řadě. Prostřední dva senzory směřují na černou čáru a první a čtvrtý senzor jsou umístěny po stranách černé čáry. Každý senzor dokáže nezávisle detekovat polohu černé čáry, a tím poskytuje komplexnější informace o prostředí. U SharkBotu jsou tyto čtyři senzory připojeny k pinům ESP32 v pořadí zleva doprava: Pin4, Pin5, Pin6 a Pin7.



1. Algoritmus sledování čáry pomocí 4kanálového snímače

Jádro algoritmu 4kanálového sledování čáry spočívá v určení polohy a míry odchylky černé čáry a následné úpravě směru jízdy chytrého auta pomocí hodnot šedi ze čtyř senzorů. Následuje konkrétní algoritmus:





Sensor status	smart car adjusting its movement
	Rovně
	Zatočit vlevo
	Zatočit vpravo
	Zatočit vlevo
	Zatočit vlevo
	Zatočit vpravo
	Zatočit vpravo
	Zatočit vpravo
	Zatočit vlevo





	Rovně
	Zatočit vpravo



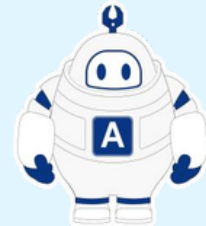
ITS Laboratoř



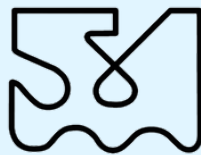
Inteligentní sledování čáry (II)



Popis projektu



Programováním se realizuje, aby chytré auto sledovalo černou trajektorii na mapě. Mapa je zobrazena níže:



Poznámka: Mapu je potřeba vytisknout samostatně, tiskový soubor se nachází ve složce "[English\Arduino\(Experienced Learner\)\5.Map\Map2\(60x90cm\).pdf](#)" přiložené k tomuto tutoriálu



Hardwarové schéma

1.Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1
	Mapa	1





Programování

2. Programování:

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce

elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduino program\07LineTracing2](#)

```
#define RightMotor_A1 12 //Define the pin numbers for the motor driver pins.
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define TrackSensor1 4 //Define the pins of the grayscale sensor
#define TrackSensor2 5
#define TrackSensor3 6
#define TrackSensor4 7
uint8_t out_flag;

int sensor1_value, sensor2_value, sensor3_value, sensor4_value;
void setup() {
    pinMode (RightMotor_A1, OUTPUT);
    pinMode (RightMotor_A2, OUTPUT);
    pinMode (LeftMotor_A1, OUTPUT);
    pinMode (LeftMotor_A2, OUTPUT);
    pinMode (TrackSensor1, INPUT);
    pinMode (TrackSensor2, INPUT);
    pinMode (TrackSensor3, INPUT);
    pinMode (TrackSensor4, INPUT);
}
void loop() {
    sensor1_value = analogRead(TrackSensor1);
    sensor2_value = analogRead(TrackSensor2);
    sensor3_value = analogRead(TrackSensor3);
    sensor4_value = analogRead(TrackSensor4);
    //The number 1 in the following comments indicates that a black line is
    //detected, and 0 indicates that no black line is detected.
    if (sensor1_value > 1700 && sensor2_value > 1700 && sensor3_value > 1700 &&
        sensor4_value > 1700) // 1111
    { // forward
        motorContr ol (200, 200 );
    }
}
```





```
else if (sensor1_value<1700 && sensor2_value>1700 && sensor3_value>1700 &&
sensor4_value<1700)    // 0110
{ // forward
    motorContr ol (200,200 );
}
else if (sensor1_value<1700 && sensor2_value>1700 && sensor3_value<1700
&& sensor4_value<1700)    // 0100
{ // left
    motorContr ol (-200, 200);
}
else if (sensor1_value<1700 &&sensor2_value<1700 && sensor3_value>1700
&& sensor4_value<1700)    // 0010
{ // right
    motorContr ol (150,- 100);
}
else if (sensor1_value>1700) // 1xxx
{ // left
    out_flag = 1;
}
else if (sensor4_value>1700) // xxx1
{ // right
    out_flag = 2;
}
else if(out_flag==1)
{ // left
    motorContr ol (-100, 150);
}
else if(out_flag==2)
{ // right
    motorContr ol (150,- 100);
}
}
```





```
/*Control the motor function: when the speed is greater than 0,  
the motor moves forward; when the speed is less than 0,  
the motor reverses; when the speed is equal to 0, the motor stops.*/  
//The PWM range is 0-255.  
void motorControl(int left_pwm, int right_pwm)  
{  
    if(left_pwm > 0){  
        analogWrite( LeftMotor_A1, 0);  
        analogWrite( LeftMotor_A2, left_pwm );  
    }  
    else {  
        analogWrite( LeftMotor_A1, -left_pwm );  
        analogWrite( LeftMotor_A2, 0);  
    }  
    if(right_pwm > 0){  
        analogWrite( RightMotor_A1 ,0);  
        analogWrite( RightMotor_A2 ,right_pwm);  
    }  
    else {  
        analogWrite( RightMotor_A1 ,-right_pwm);  
        analogWrite( RightMotor_A2 ,0);  
    }  
    delay (5);  
}
```





Upozornění

Pokud auto během sledování čáry nejezdí správně, může to být způsobeno následujícími důvody:

- 1. Vozidlo je příliš pomalé.**
- 2. Pro vyřešení problému můžete v programu zvýšit hodnotu PWM motoru, aby vozidlo mělo dostatečnou rychlost.**
- 3. ESP32 může kvůli rušení okolním světlem nesprávně vyhodnotit bílou plochu jako černou čáru.**
- 4. Pro vyřešení tohoto problému nejprve změřte konkrétní hodnoty snímače šedé na bílé ploše a na černé čáře a podle toho upravte prahovou hodnotu v programu.**

prahová hodnota = (hodnota černé čáry + hodnota bílé plochy) / 2

```
if (sensor1_value>1700 && sensor2_value>1700 && sensor3_value>1700 && sensor4_value>1700,
```



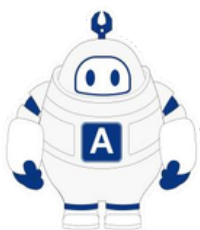
Oddíl 6: Měření rychlosti



Cíle kurzu



-  Pochopit princip fungování enkodéru
-  Pochopit metodu měření rychlosti pomocí enkodéru
-  Pochopit roli přerušovacích (interrupt) funkcí
-  Napsat program pro měření rychlosti motoru SharkBotu



Jak přesně změřit rychlost chytrého auta během jízdy?

Sledování rychlosti v reálném čase je velmi důležitá funkce při provozu chytrého auta. Například při úloze sledování čáry přímo ovlivňuje řízení rychlosti auta stabilitu a efektivitu jízdy.



Pro měření rychlosti auta potřebujeme zařízení zvané enkodér a motor typu N20 s vestavěným enkodérem. Dále si ukážeme, jak enkodér měří rychlost vozidla.

Enkodér





Enkodér je druh senzoru používaného k detekci rotace nebo lineárního pohybu. Dokáže převést mechanický pohyb na elektrický signál a následně generovat pulzní nebo digitální signál, který poskytuje informace o rychlosti, poloze a směru.



Enkodéry se podle principu fungování dělí na optické (fotoelektrické) enkodéry a Hallovy enkodéry.

V této lekci používáme inkrementální Hallův enkodér. Hallův enkodér je rotační enkodér založený na Hallově jevu, který měří rychlost a směr motoru pomocí detekce změn magnetického pole. N20 enkodérový motor SharkBotu používá AB dvoufázový enkodér a každý fázový pulzní signál enkodéru vytvoří 7 pulzů při každé jedné otáčce motoru.

V projektu SharkBot je návod pro připojení enkodéru k ESP32 následující:

Encoder	Encoder pin	ESP32 pinout
Levý enkodér	LeftEncoder_A	Pin2
	LeftEncoder_B	Pin1
Pravý enkodér	RightEncoder_A	Pin10
	RightEncoder_B	Pin47

Měření rychlosti vozidla pomocí enkodéru





Měření rychlosti chytrého auta ve skutečnosti znamená měření rychlosti motoru. Inkrementální Hallův enkodér generuje dvoufázové pulzní signály A a B, pomocí kterých lze vypočítat rychlost motoru podle počtu pulzů a časového intervalu mezi nimi. Směr otáčení motoru lze určit detekcí fázového vztahu mezi signály A a B. Konkrétní postup je následující:

1. Zaznamenat počet pulzů

Zaznamenejte počet pulzů, které enkodér vygeneruje během určitého časového intervalu.

2. Výpočet rychlosti

Na základě počtu pulzů, rozlišení enkodéru (počet pulzů na otáčku / PPR), převodového poměru a časového intervalu výpočtu se vypočítá otáčková rychlost motoru.

Například pokud se rychlost počítá každých 100 ms a jednotkou je RPM (počet otáček za minutu), vzorec pro výpočet rychlosti motoru je následující:

$$\text{Rychlost motoru} = (\text{počet pulzů} \times 60 \times 1000) / (\text{PPR} \times \text{převodový poměr} \times \text{časový interval})$$

3. Výpočet směru otáčení motoru

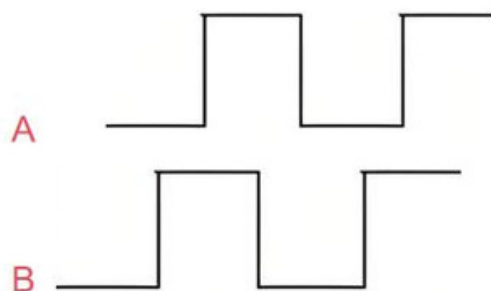
Když jsou úrovně signálů fáze A a fáze B stejné, motor se otáčí ve směru hodinových ručiček; když se úrovně fáze A a fáze B liší, motor se otáčí proti směru hodinových ručiček.

Když má pulzní signál fáze A náběžnou hranu, pulzní signál fáze B je v logické úrovni HIGH,

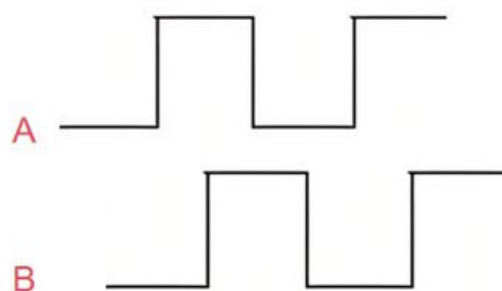




což znamená, že se motor otáčí ve směru hodinových ručiček.



Když má pulzní signál fáze A náběžnou hranu a signál fáze B je v logické úrovni LOW, znamená to, že se motor otáčí proti směru hodinových ručiček.



Funkce přerušení



Abychom mohli v reálném čase měřit pulzní signál z enkodéru, je potřeba v programu použít funkci přerušení. Přerušení (interrupt) je část kódu nezávislá na hlavním programu. Když nastane podmínka pro přerušení, CPU dočasně přeruší vykonávání hlavního programu a místo toho provede kód přerušovací funkce.

Na řídicí desce ESP32 jsou k dispozici hardwarová přerušení, která lze použít k detekci externích událostí (např. stisk tlačítka, detekce pulzů enkodéru apod.). Všechny piny ESP32 podporují přerušení a různé režimy spouštění.





V Arduino se funkce `attachInterrupt()` používá k nastavení režimu přerušení a obslužné funkce přerušení na desce ESP32. Referenční kód je následující:

```
//Set the interrupt mode, which is triggered when the encoder A-phase pin detects a rising edge pulse
void setup(){
    attachInterrupt(LeftEncoder_A,EncoderISR,Rising);
}

// Interrupt function to count the number of encoder pulses
void EncoderISR(){
    // Read the level of the encoder phase B
    int stateB = digitalRead(LeftEncoder_A);

    if(stateB == HIGH){//AB phase are both high
        pulseCount++;//motor rotates clockwise
    }
    else{// Phase A is all high, phase B is low
        pulseCount++;//motor rotates counterclockwise
    }
}
```





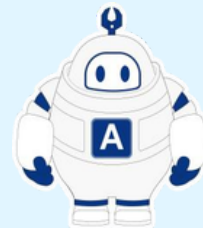
ITS Laboratoř



Měření rychlosti



Popis projektu



Pomocí enkodérového motoru N20 a Arduina realizujte měření rychlosti motoru a zobrazte ji v sériovém monitoru. Jak je znázorněno na obrázku níže:

```

Output  Serial Monitor x
Message (Enter to send message to 'ESP32S3 Dev Modu... No Line Ending 115200 baud
14:05:08.596 -> Left_rpm:131.43
14:05:08.596 -> Right_rpm:132.57
14:05:08.689 -> Left_rpm:130.29
14:05:08.689 -> Right_rpm:132.57
14:05:08.813 -> Left_rpm:132.00
14:05:08.813 -> Right_rpm:132.57
14:05:08.905 -> Left_rpm:132.00
14:05:08.905 -> Right_rpm:132.57
14:05:08.984 -> Left_rpm:130.86
14:05:08.984 -> Right_rpm:132.57
14:05:09.121 -> Left_rpm:133.71
14:05:09.121 -> Right_rpm:132.57
  
```



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

2.V Arduinu se funkce `attachInterrupt()` používá k nastavení režimu přerušení a obslužné funkce přerušení na desce ESP32. Referenční kód je následující:

[English\Arduino\(Experienced Learner\)\3.Arduino program\08SpeedMeasure](#)

```
#define RightMotor_A1 12 // Define the pin number of the motor driver pin
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define LeftEncoder_A 2 // Define the pins for the AB phase of the encoder
#define LeftEncoder_B 1
#define RightEncoder_A 10
#define RightEncoder_B 47

long left_pulseCount = 0; // the number of pulses of the left encoder
long right_pulseCount = 0; // the number of pulses of the right encoder
long interval = 100; // time interval for reading the number of pulses
long previousTime = 0; // current point in time
long lastTime = 0; // the last time that
int PPR = 7; // number of pulses per revolution of the encoder
int gearRatio = 150; // reduction ratio
float right_rpm = 0; // right motor rpm
float left_rpm = 0; // left motor rpm

void setup() {
    pinMode(RightMotor_A1, OUTPUT); // Set the working mode of the input pins of
    the driver board
    pinMode(RightMotor_A2, OUTPUT);
    pinMode(LeftMotor_A1, OUTPUT);
    pinMode(LeftMotor_A2, OUTPUT);
    pinMode(RightEncoder_A, INPUT); // set the encoder pin to input mode
    pinMode(RightEncoder_B, INPUT);
    pinMode(LeftEncoder_A, INPUT);
    pinMode(LeftEncoder_B, INPUT);
    // Set the interrupt to be triggered when the A-phase signal changes
    attachInterrupt(digitalPinToInterrupt(RightEncoder_A), rightEncoderISR,
    RISING);
```





```

    attachInterrupt(digitalPinToInterrupt(LeftEncoder_A), leftEncoderISR,
    RISING);
    Serial .begin (115200 );
}
void loop() {
    previousTime = millis();
    motorContr ol (255,255 );
    if(previousTime - lastTime >= interval){
// Calculate the speed of the left motor
        left_rpm = (left_pulseCount * 60.0 * 1000) / (PPR * gearRatio * interval);
// Calculate the speed of the right motor
        right_rpm = (right_pulseCount * 60.0 * 1000) / (PPR * gearRatio *
interval) ;
        Serial .print ("Left_rpm:" );
        Serial .println (left_rpm );
        Serial .print ("Right_rpm:" );
        Serial .println (right_rpm );
        left_pulseCount = 0;
        right_pulseCount = 0;
        lastTime = millis();
    }
}
void leftEncoderISR() { //Calculate the number of pulses from the left
encoder.
    static unsigned long lastInterruptTime = 0;
    unsigned long interruptTime = micros();
    if (interruptTime - lastInterruptTime > 100) { //Stabilize voltage.
        if (digitalRead(LeftEncoder_B) == LOW) {
            left_pulseCount++; //WhentheAandBphaselevelsarenotequal,
the left motor rotates forward.
        } else {
            left_pulse Count--;
        }
    }
    lastInterruptTime = interruptTime;
}

```





```
void rightEncoderISR() { //Calculate the number of pulses from the right
encoder.
    static unsigned long lastInterruptTime = 0;
    unsigned long interruptTime = micros();
    if (interruptTime - lastInterruptTime > 100) { // Voltage stabilization
        if (digitalRead(RightEncoder_B) == HIGH) {
            right_pulseCount++; //When the A and B phase levels are equal, the
right motor rotates forward.
        } else {
            right_pulseCount--;
        }
    }
    lastInterruptTime = interruptTime;
}

void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite( LeftMotor_A1, 0);
        analogWrite( LeftMotor_A2, left_pwm );
    }
    else {
        analogWrite( LeftMotor_A1, -left_pwm );
        analogWrite( LeftMotor_A2, 0);
    }
    if(right_pwm > 0){
        analogWrite( RightMotor_A1 ,0);
        analogWrite( RightMotor_A2 ,right_pwm);
    }
    else {
        analogWrite( RightMotor_A1 ,-right_pwm);
        analogWrite( RightMotor_A2 ,0);
    }
    delay (5);
}
```





Tipy

Protože jmenovité napětí motoru je 12 V a napětí poskytované SharkBotem je přibližně 8 V, motor nemusí dosáhnout maximální rychlosti 150 ot/min.



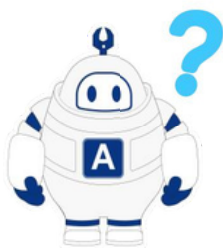
Oddíl 7: Jízda chytrého auta po přímce



Cíle kurzu



-  Pochopit koncept uzavřené zpětnovazební smyčky
-  Pochopit koncept PID regulace
-  Aplikovat PID algoritmus pro řízení chytrého auta tak, aby jelo rovně po přímce



Když chytré auto jede po přímce, někdy se stáčí na jednu stranu a nejede rovně. Jaký je důvod?

Udržení jízdy v přímém směru je základní, ale velmi důležitá funkce chytrého auta. Kvůli rozdílům ve výkonu motorů, tření kol a dalším faktorům však nemusí mít obě kola stejnou rychlost, což způsobuje vybočování z přímé dráhy.

Aby byla rychlost obou kol vyrovnaná, je potřeba v reálném čase měřit rychlost kol a podle toho ji průběžně upravovat





i reálném čase podle detekované rychlosti tak, aby rychlost obou kol dosáhla stejné hodnoty. Toto řízení rychlosti kol v reálném čase se nazývá uzavřená zpětnovazební regulace (closed-loop feedback control)

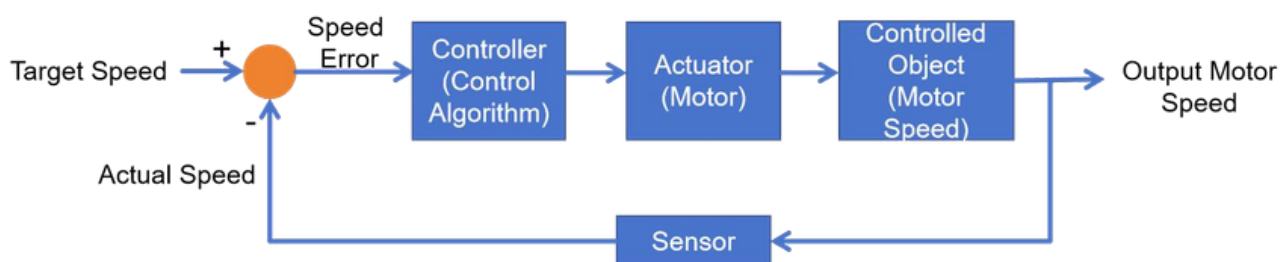
Uzavřená zpětnovazební regulace



Uzavřená zpětnovazební regulace (Closed-Loop Feedback Control) je metoda automatického řízení, při které systém v reálném čase sleduje svůj výstup a na základě zpětné vazby provádí úpravy tak, aby byl systém stabilní a pracoval podle očekávání. Tato metoda se široce používá v průmyslové automatizaci, řízení robotů, chytré domácnosti, autonomním řízení vozidel a dalších oblastech. Systém uzavřené zpětnovazební regulace se obvykle skládá z následujících klíčových komponent:

- 1. Nastavená (požadovaná) hodnota:** určuje cílový výstup systému.
- 2. Řídicí jednotka:** je zodpovědná za výpočet řídicího signálu, obvykle pomocí některého řídicího algoritmu, který provádí potřebné výpočty
- 3. Akční člen:** na základě výstupních řídicích signálů z regulátoru provádí odpovídající činnost, například řízení pohybu motoru.
- 4. Senzory:** detekují skutečný výstup systému, například měří rychlost motoru, a poskytují tato data zpětnou vazbu řídicí jednotce.
- 5. Výpočet chyby:** Řídicí jednotka vypočítá rozdíl (chybu) mezi požadovanou hodnotou a skutečnou hodnotou a upravuje výstup tak, aby tuto chybu minimalizovala. Na příkladu řízení rychlosti motoru je pracovní postup uzavřené smyčky následující:





V uzavřené zpětnovazební regulaci je klíčovým prvkem řídicí algoritmus. Mezi běžné algoritmy uzavřeného řízení patří PID algoritmus, fuzzy řízení, neuronové síťové řídicí algoritmy atd. V této lekci se zaměříme na PID algoritmus.



1. Pointa algoritmu

PID algoritmus je klasický algoritmus v uzavřené zpětnovazební regulaci, který se široce používá v průmyslovém řízení i v oblasti chytrých aut. PID je zkratka pro Proporcionální (Proportional), Integrační (Integral) a Derivační (Derivative) složku řízení. Jeho úkolem je upravovat výstupní hodnotu na základě chyby mezi skutečnou a požadovanou hodnotou tak, aby se skutečná hodnota co nejrychleji přiblížila cílové hodnotě.

Řídicí funkce jednotlivých složek jsou následující:

Proporcionální (P) řízení: upravuje výstup úměrně velikosti aktuální chyby. Čím větší je chyba, tím větší je výstup, což zrychluje odezvu systému a umožňuje rychle přiblížit rychlost motoru k cílové hodnotě.





Integrační (I) řízení: upravuje výstup na základě kumulované (nasčítané) hodnoty chyby. Používá se k odstranění dlouhodobé ustálené chyby systému.

Derivační (D) řízení: upravuje výstup podle rychlosti změny chyby. Slouží k predikci budoucího trendu chyby a pomáhá potlačit kolísání a přestřelování systému.

2. Vzorec PID algoritmu

Výstup = $K_p \times \text{chyba} + K_i \times \text{integrál chyby} + K_d \times \text{rychlost změny chyby}$

kde K_p , K_i a K_d jsou proporcionální, integrační a derivační koeficienty.

Chyba = cílová hodnota – skutečná hodnota

Integrál chyby = kumulace všech předchozích chyb

Derivační složka (změna chyby) = aktuální chyba – předchozí chyba

V této lekci, aby se zjednodušil proces programování, použijeme knihovnu „PID.h“, která umožňuje přímo volat PID algoritmus. Základní instrukce jsou následující:





```
//Create a PID object, need to set the PID input, output, target value
// and the values of the three coefficients Kp, Ki, Kd, and the direction of the PID
calculation, the default value of which is DIRECT
PID myPID(&Input,&Output,&Target,Kp,Ki,Kd,DIRECT);
//Set the PID operating mode
myPID.SetMode(AUTOMATIC);
//Set the time interval for PID calculation
myPID.SetSampleTime(interval);
//Set the output range of PID
myPID.SetOutputLimits(0,255);
```

Nastavení PID koeficientů



V PID regulaci je nastavení PID koeficientů nejdůležitějším krokem. Správně zvolené PID koeficienty umožňují, aby se skutečný výstup rychleji a stabilněji přibližoval požadované hodnotě.

Ne každý řídicí systém musí využívat všechny tři složky PID. Někdy stačí pouze P regulace, jindy PD regulace a v některých případech je potřeba plné PID řízení. Vše závisí na konkrétních požadavcích a vlastnostech daného systému.

A jak se nastavují PID koeficienty?

V běžných aplikacích PID regulátoru se obvykle používá metoda pokus–omyl pro nastavení parametrů PID. Konkrétní postup je následující:

Nastavte hodnoty K_p , K_i a K_d na 0.

Nejprve sledujeme porovnání mezi skutečnou rychlostí vozíku a





cílovou rychlostí, když jsou všechny tři hodnoty K_p , K_i a K_d nastaveny na nulu.

V Arduino můžeme pomocí sériového portu vypisovat skutečnou rychlost auta a cílovou rychlost a poté sledovat jejich průběh v grafu pomocí „Serial Plotter“.

Nahrajte PID program SharkBotu a v Arduino IDE klikněte na:



Následuje příklad nastavení PID parametrů pro PID regulaci rychlosti levého motoru. Modrá křivka představuje cílovou rychlost, oranžová křivka představuje skutečnou rychlost motoru.



2. Nastavení K_p

Z výše uvedeného obrázku zjistíme, že mezi skutečnou rychlostí levého motoru a cílovou rychlostí je velká chyba. Nejprve postupně zvyšujeme hodnotu K_p , aby se rychlost motoru přiblížila cílové hodnotě.

Metoda nastavení:

(1) Postupně zvyšujte K_p (např. 1.0, 1.5, 2.0).





(2) Sledujte, zda se auto dokáže rychleji přiblížit cílové rychlosti, ale zároveň nesmí docházet k výraznému kmitání. Následující graf ukazuje průběh rychlosti motoru při $K_p = 4.0$, který je blíže cílové hodnotě než předtím.



(3) Pokud se na obrázku objeví velké kmitání, je nutné snížit hodnotu K_p , dokud kmitání nezmizí.





1. Nastavení Ki

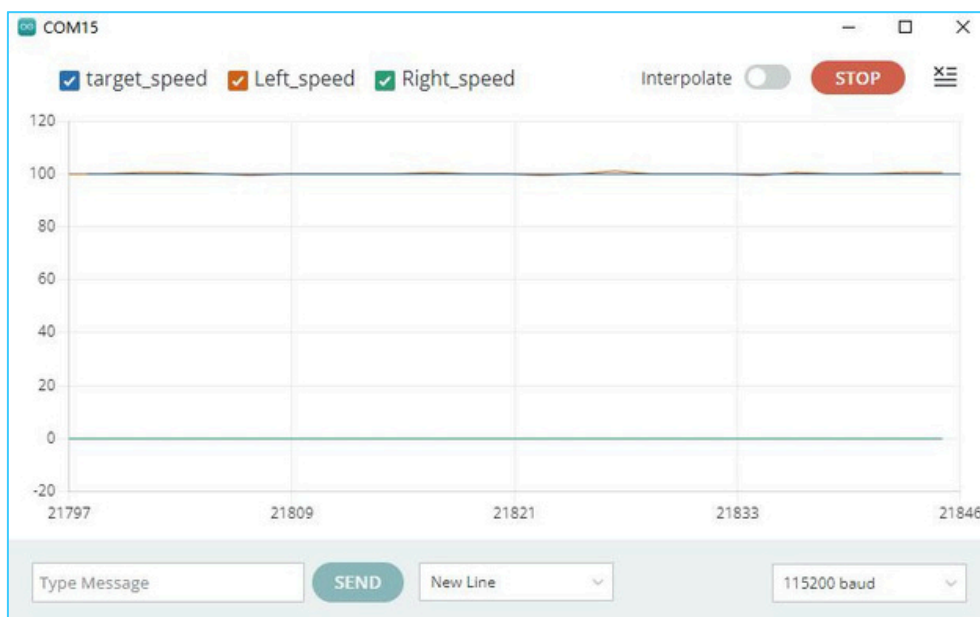
Pokud po zvýšení K_p stále existuje ustálená chyba, postupně zvyšujeme hodnotu K_i , dokud nedosáhneme vhodného nastavení.

Metoda nastavení:

1. **Začněte s $K_i = 0.5$.**
2. **Postupně zvyšujte K_i (např. 1.0, 1.5, 2.0).**
3. **Sledujte, zda rychlost dokáže stabilně dosáhnout cílové hodnoty.**
4. **Pokud dojde k přestřelení (rychlost překročí cílovou hodnotu a poté klesne zpět), znamená to, že K_i je příliš velké a je potřeba ho snížit.**

Následující graf ukazuje skutečnou rychlost motoru při $K_i = 2.0$, která se téměř shoduje s cílovou hodnotou.





1. Nastavení Kd

Pokud jsou K_p a K_i nastaveny, ale stále dochází ke kmitání, znamená to, že systém reaguje na chybu příliš agresivně. V takovém případě je potřeba zvýšit hodnotu K_d . Pokud však k žádnému kmitání nedochází, není nutné K_d upravovat.

Postup nastavení:

1. Postupně zvyšujte K_d (např. 0.0001, 0.0003, 0.0005).
2. Sledujte, zda se sníží kmitání (otřesy) auta.





ITS Laboratoř



Jízda chytrého auta po přímce



Popis projektu



PID algoritmus umožňuje chytrému autu jet po přímce a sleduje jeho výkon v různých podmínkách na silnici.



Hardwarové schéma

1. Hardware list

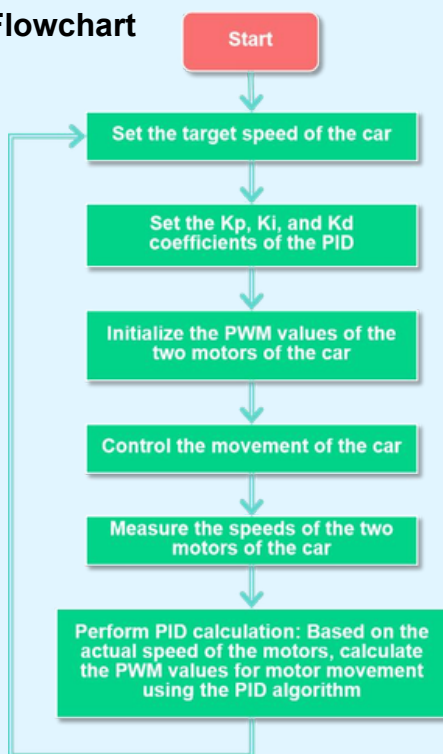
Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programming

1. Program Flowchart



Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduino program\09PID GoStraight](#)

```
#include<PID.h>Import the PID library

#define RightMotor_A1 12 //Define the pin numbers of the motor driver pins
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define LeftEncoder_A 2 //Define the pins for the encoder's A and B phases
#define LeftEncoder_B 1
#define RightEncoder_A 10
#define RightEncoder_B 47
```





```

long left_pulseCount = 0; //Pulse count of the left encoder
long right_pulseCount = 0; //Pulse count of the right encoder
long interval = 100; //Read the time interval for PID calculation
long previousTime = 0; //Current time point
long lastTime = 0; //Last time
int PPR = 7; //Pulses per revolution of the encoder
int gearRatio = 150; //Reduction ratio
double right_speed = 0; //Rotational speed of the right motor
double left_speed = 0; //Rotational speed of the left motor
double left_pwm = 100; //Initialize the power of the left motor
double right_pwm = 100; //Initialize the power of the right motor
double target_speed = 100; //Target Speed
//Set the coefficients of the left PID:
double left_Kp = 2.0, left_Ki = 3.0, left_Kd = 0.007;
//Set the coefficients of the right PID:
double right_Kp = 2.0, right_Ki = 3.0, right_Kd = 0.07;

/*
Generate a PID regulator object, set the PID input to the actual motor
speed, set the PID output to the motor power, set the desired value to
the target speed, and set the Kp, Ki, Kd, and PID control direction in
sequence.* /
PID leftPID(&left_speed, &left_pwm, &target_speed, left_Kp,
            left_Ki, left_Kd, DIRECT);
PID rightPID(&right_speed, &right_pwm, &target_speed, right_Kp,
            right_Ki, right_Kd, DIRECT );

void setup() {
    pinMode( RightMotor_ A1, OUTPUT ); //
    pinMode( RightMotor_ A2, OUTPUT );
    pinMode( LeftMotor_ A 1, OUTPUT );
    pinMode( LeftMotor_ A 2, OUTPUT );
    pinMode( RightEncode r_A, INPUT ); //
    pinMode( RightEncode r_B, INPUT );
    pinMode( LeftEncoder _A, INPUT );
    pinMode( LeftEncoder _B, INPUT );
    //Set an interrupt to be triggered when the A-phase signal changes.

```





```
attachInterrupt (digitalPinToInterrupt (RightEncoder_A ),
                rightEncoderISR, RISING);
attachInterrupt (digitalPinToInterrupt (LeftEncoder_A ),
                leftEncoderISR, RISING);
Serial .begin( 115200);
leftPID.SetMode(AUTOMATIC);//Set the PID working mode
leftPID.SetSampleTime(interval);//Set the time interval for PID
calculation.
leftPID.SetOutputLimits(0,255);//Set the output range of the PID.
rightPID.SetMode(AUTOMATIC);//Set the PID operating mode.
rightPID.SetSampleTime(interval);//Set the time interval for PID
calculation.
rightPID.SetOutputLimits(0,255);//Set the output range of the PID.
}

void loop() {
    previousTime = millis();
    motorControl(left_pwm,right_pwm);//set the power of the cart
    if(previousTime - lastTime >= interval){
        left_speed = (left_pulseCount * 60.0 * 1000) / (PPR * gearRatio *
interval);
        right_speed = (right_pulseCount * 60.0 * 1000) / (PPR * gearRatio
* interval);// Calculate the speed of the right motor
leftPID.Compute();//PID calculation
rightPID.Compute();//PID calculation
        Serial .print("Left_speed:");
        Serial .print(left_speed );
        Serial.print("      Right_speed:");
        Serial .println (right_speed );
        Serial .print("Left_pwm: ") ;
        Serial .print(left_pwm ) ;
        Serial.print("      Right_pwm:");
        Serial .println (right_pwm) ;
        left_pulseCount = 0;
        right_pulseCount = 0;
        lastTime = millis();
    }
}
```





```

void leftEncoderISR(){//Calculate the pulse count of the left encoder.
    int stateB = digitalRead(LeftEncoder_B);
    if (stateB == LOW){
        left_pulseCount++;//When AB phase level is not equal, the left motor
turns forward
    }else{
        left_pulseCount--;
    }
}

void rightEncoderISR(){//Calculate the number of pulses in the right
encoder
    int stateB = digitalRead(RightEncoder_B);
    if (stateB == HIGH){
        right_pulseCount++;//When AB phase level is equal, the right motor
turns forward
    }else{
        right_pulseCount--;
    }
}

//Control motor function, PWM is greater than 0, motor forward, PWM is less
than 0, motor backward, PWM is equal to 0, motor stop
void motorControl(int left_PWM, int right_PWM)
{
    if(left_PWM > 0){
        analogWrite (LeftMotor_ A1,0) ;
        analogWrite (LeftMotor_ A2,left_P WM);
    }
    else{
        analogWrite (LeftMotor_ A1,-left_ PWM);
        analogWrite (LeftMotor_ A2,0) ;
    }
    if(right_PWM > 0){
        analogWrite (RightMotor _A1,0 );
        analogWrite (RightMotor _A2,right _PWM) ;
    }
    else{

```





```
    analogWrite( RightMotor_A1, -right_PWM);  
    analogWrite( RightMotor_A2, 0);  
}  
delay (5);  
}
```





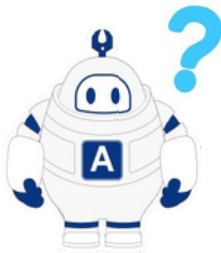
Oddíl 8: Inteligentní hlasové ovládání



Cíle kurzu



-  Pochopit koncept rozpoznávání řeči.
-  Pochopit aplikace rozpoznávání řeči.
-  Zvládnout použití hlasových příkazů SharkBot pro rozpoznávání řeči.



Have you ever used a voice assistant on your cell phone? How can you make your smart car also have voice assistant function and control it by voice?

S neustálým pokrokem v technologii umělé inteligence se technologie hlasové interakce široce používá v mnoha oblastech.

Mnoho chytrých zařízení a systémů je vybaveno funkcí hlasového ovládání, například pomocí hlasového asistenta pro řízení chytré domácnosti nebo pro ovládání mobilního telefonu a podobně. Stejně tak se hlasové ovládání používá i v chytrých autech. K realizaci této funkce musíme využít technologii rozpoznávání hlasu.





Nyní se podívejme hlouběji do tajů rozpoznávání hlasu, aby chytrá auta dokázala porozumět našim hlasovým příkazům a reagovat na ně.

Rozpoznávání řeči



Rozpoznávání řeči je technologie, která převádí lidské hlasové signály na text nebo příkazy. Jejím hlavním cílem je umožnit strojům „rozumět“ lidské řeči a na základě hlasových instrukcí provádět odpovídající operace. Široce se používá v inteligentních asistentech, automatizovaném zákaznickém servisu, hlasovém zadávání, chytré domácnosti a dalších oblastech.

Rozpoznávání řeči se podle aplikačního scénáře dělí na online rozpoznávání řeči a offline rozpoznávání řeči.

Online rozpoznávání řeči: vyžaduje připojení k internetu, spoléhá na cloudové servery pro zpracování a rozpoznání řeči, má velký slovník a je vhodné pro složité scénáře hlasové interakce.

Offline rozpoznávání řeči: nevyžaduje připojení k internetu, zpracování a rozpoznání řeči probíhá přímo na lokálním zařízení, reakce je rychlá.

Proces rozpoznávání řeči



Základní kroky rozpoznávání řeči lze zjednodušit následovně:

Sběr zvuku: mikrofon přijímá hlasové signály.



Potlačení šumu: filtrování okolního hluku a extrakce užitečných zvukových signálů.

Extrakce příznaků: převod zvukových vln na spektrogramy nebo MFCC (Mel Frequency Cepstral Coefficient) vlastnosti.

Porovnání klíčových slov: porovnání s přednastavenou knihovnou příkazů a výstup odpovídajícího výsledku.

Knihovna rozpoznávání řeči



SharkBot používá offline modul rozpoznávání řeči k rozpoznání hlasu. Ten rozpoznává pevně daný hlasový obsah a převádí jej na odpovídající hexadecimální data, která jsou následně přenesena do řídicí jednotky prostřednictvím sériové komunikace. Následuje knihovna hlasových příkazů modulu rozpoznávání řeči:

Hlasový příkaz	Signalinginstruction	Caraction
Wakeword="HiSharkBot"	/	/
Forward GoForward	0x01	Forward
Backward GoBackward	0x02	Backup
TurnLeft TurnRight	0x03	Turnleft
	0x04	Turnright
LowSpeed HlghSpeed	0x05	low-speedoperation
	0x06	High-speedoperation





Turnontheredlight	0x07	Turnontheredlight
Turnofftheredlight	0x08	Turnofftheredlight
Turnonthegreenlight	0x09	Givethegreenlight
Turnoffthegreenlight	0x0A	Turnoffthegreenlight
Turnonthebluelight	0x0B	Turnonthebluelight
Turnoffthebluelight	0x0C	Turnoffthebluelight
LineTracing	0x0D	Cartracking
ObstacleAvoidance	0x0E	Carobstacleavoidance
Following	0x0F	Thecarfollowstheobject
Stop	0x10	Stop
HappyBirthday	0x11	Playthe"HappyBirthday"
MerryChristmas	0x12	song
LittleStar	0x13	Playthesong"Merry Christmas"
		Playthesong"LittleStar"

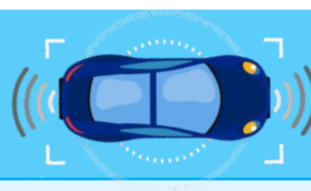
Hlasový rozpoznávací modul SharkBotu vyžaduje probouzecí příkaz „Hi SharkBot“ pro aktivaci funkce rozpoznávání hlasu.

Hlasový modul podporuje přizpůsobené hlasové příkazy. Podrobné kroky obsluhy naleznete v souboru „English\Arduino(Experienced Learner)\7.VoiceRecognitionModule description“ v přiložených materiálech.





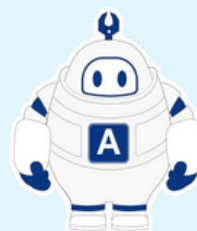
ITS Laboratoř



Inteligentní hlasové ovládání



Popis projektu



Ovládejte chytré auto hlasem – jízda vpřed, vzad, zatáčení vlevo, vpravo a přehrávání hudby.



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

2.Coding

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\10SpeechRecognition](#)

```
#define RightMotor_A1 12 // Define the pin number of the motor driver pin
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define Buzzer 39 //define Buzzer pins
#define VOICE_TX_PIN 42
#define VOICE_RX_PIN 42
#define VoiceSerial Serial2
#define C4 262// Define the frequency of the note
#define D4 294
#define E4 330
#define F4 350
#define G4 393
#define A4 441
#define B4 495
#define N 0

int length;
int tune[] = { C4, N, C4, G4, N, G4, A4, N, A4, G4, N, F4, N, F4, E4, N, E4, D4, N, D4, C4 }; // littel star
float durt[] = { 0.99, 0.01, 1, 0.99, 0.01, 1, 0.99, 0.01, 1, 1.95, 0.05, 0.99, 0.01, 1, 0.99, 0.01, 1, 0.99, 0.01, 1, 2 };
void setup() {
  pinMode(RightMotor_A1,OUTPUT); //Set the working mode of the input pins of the driver board

  pinMode (RightMotor _A2,OUTPU T );
  pinMode (LeftMotor_ A1,OUTPUT ) ;
  pinMode (LeftMotor_ A2,OUTPUT ) ;
  length = sizeof(tune) / sizeof(tune[0]);
  VoiceSerial 1 .begin(9600 ,SERIAL_8N1 ,VOICE_RX_ PIN,VOICE _TX_PIN ); delay (1000 );
}
```





```

void loop() {
  if (VoiceSerial.available() > 0)
  {
    int data = VoiceSerial.read();
    if (data == 0x01){//forward
      motorContr ol(255 ,255);
    }
    else if (data == 0x02){//backward
      motorContr ol(-255 ,-255);
    }
    else if (data == 0x03){// turn left
      motorContr ol(-255 ,255);
    }
    else if (data == 0x04){ // turn right
      motorContr ol(255 ,-255);
    }
    else if (data == 0x10){ // stop
      motorContr ol(0, 0);
    }
    else if (data == 0x13){ // play music
      motorContr ol(0, 0);
      music() ;
    }
  }
}

void music(){
  for(int x=0;x<length;x++) {
    tone(Buzzer, tune[x]);
    delay(500 * durt[x]);
    noTone (Buzzer) ;
  }
}

void motorControl(int left_pwm, int right_pwm)
{
  if(left_pwm > 0){

```





```
    analogWrite (LeftMotor_ A1, 0);  
    analogWrite (LeftMotor_ A2, left_P WM );  
}  
else {  
    analogWrite (LeftMotor_ A1, -left_ PWM );  
    analogWrite (LeftMotor_ A2, 0);  
}  
if(right_PWM > 0){  
    analogWrite (RightMotor _A1, 0);  
    analogWrite (RightMotor _A2, right _PWM );  
}  
else {  
    analogWrite (RightMotor _A1, -righ t_PWM );  
    analogWrite (RightMotor _A2, 0);  
}  
delay (5);  
}
```





Kapitola 8: Internet věcí pro inteligentní dopravu

Přemýšlel jsi někdy nad tím, jak se zobrazuje poloha autobusů ve městě v reálném čase? Jak jsou semaforey dynamicky upravovány podle hustoty dopravy? Jak silniční elektronické tabule zobrazují nejnovější dopravní informace? Za všemi těmito prvky „internetu všeho“ je nepostradatelná podpora bezdrátových komunikačních technologií.

V inteligentním dopravním systému hraje technologie internetu věcí (IoT) roli „nervové sítě“, která propojuje auta, silnice a cloud do jednoho celku prostřednictvím infračervené komunikace, Bluetooth, WiFi a dalších komunikačních metod. V této kapitole se podrobně seznámíme se základními technologiemi bezdrátové komunikace: infračervená komunikace nám umožní prozkoumat princip krátkodosahového dálkového ovládání; Bluetooth komunikace odhalí pohodlí propojení chytrých zařízení; a WiFi komunikace nám umožní zvládnout cloudovou interakci a přenos dat na velké vzdálenosti.





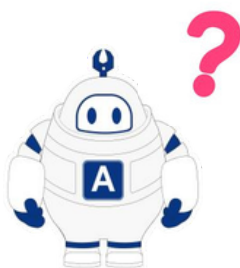
Oddíl 1: Technologie infračervené komunikace



Cíle kurzu



-  Pochopit princip infračervené komunikace.
-  Zvládnout používání infračerveného přijímače a infračerveného dálkového ovladače.
-  Naprogramovat ovládání pohybu auta pomocí dálkového ovladače.



Kromě ovládání chytrého auta hlasem, jaké další způsoby můžeš použít pro jeho dálkové ovládání?

Kromě hlasového ovládání je běžnou metodou dálkového ovládání také infračervené ovládání, například u televizních ovladačů nebo ovladačů klimatizace. Tato funkce dálkového ovládání je realizována pomocí technologie infračervené komunikace. Jaká je to technologie?





Technologie infračervené komunikace



Infračervená komunikace je bezdrátová komunikační technologie, která využívá infračervené světelné vlny pro přenos dat. Infračervené světlo se v elektromagnetickém spektru nachází mezi mikrovlnami a viditelným světlem, s rozsahem vlnových délek přibližně od 700 nanometrů do 1 milimetru.

Realizace infračervené komunikace závisí hlavně na dvou částech: infračerveném vysílači a infračerveném přijímači. Infračervený vysílač je zodpovědný za vysílání infračervených signálů a infračervený přijímač za jejich příjem. Konkrétní proces zahrnuje následující klíčové kroky:

- **Kódování dat**
- Na začátku komunikace zdrojové zařízení (například dálkový ovladač) zakóduje data určená k přenosu (například příkazy jako zapnutí/vypnutí televize, nastavení hlasitosti apod.) do elektrických signálů. Tento krok obvykle zahrnuje převod dat do specifického kódovaného formátu pro usnadnění přenosu a příjmu.
- **Modulace signálu**
- Zakódované elektrické signály jsou modulátorem převedeny na infračervené signály. Během procesu modulace se obvykle používá pulzně šířková modulace (PWM) k reprezentaci dat. Modulátor řídí spínání infračervených LED diod (svítivých diod), čímž vytváří odpovídající infračervené světelné pulzy.





- **Přenos signálu**

- Infračervené LED diody vysílají pulzy infračerveného světla, které se šíří vzduchem k přijímacímu zařízení. Infračervená komunikace je obvykle krátkodosahová a vyžaduje přímou viditelnost, protože infračervené světlo nemůže procházet stěnami ani jinými neprůhlednými objekty.

- **Příjem a demodulace signálu**

- Infračervený přijímač (obvykle fotodioda) na přijímacím zařízení detekuje infračervené světelné pulzy a převádí je zpět na elektrické signály. Tyto elektrické signály jsou následně demodulovány demodulátorem, aby se obnovil původní datový formát.

- **Dekódování dat**

- Nakonec přijímací zařízení provede dekodování demodulovaného signálu, obnoví původní data nebo příkazy a následně provede odpovídající operace, například změnu kanálu televizoru nebo úpravu hlasitosti. Výhody infračervené komunikace zahrnují nízké náklady, snadné použití a malou citlivost na rušení, proto se široce používá v dálkovém ovládání, krátkodobém přenosu dat a podobně. Její omezení však zahrnují krátký komunikační dosah, nemožnost průchodu překážkami a citlivost na silné světelné zdroje. Přesto je infračervená komunikace stále velmi efektivní v mnoha konkrétních aplikacích.

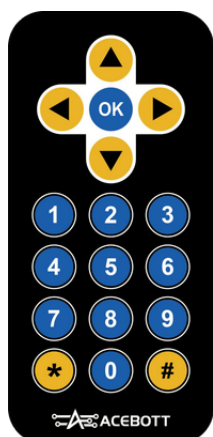




IR dálkový ovladač

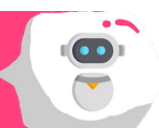


Běžně používaným infračerveným vysílačem je infračervený dálkový ovladač, který se skládá z několika tlačítek a infračervených LED diod. Každému tlačítku odpovídá jiná hodnota infračerveného kódu. Po stisknutí tlačítka infračervená LED dioda vysílá infračervené světlo s odpovídajícím kódem. Infračervené dálkové ovladače používají různé způsoby kódování, přičemž nejběžnějším je kódování NEC.



Button	Value	Button	Value	Button	Value
	B946FF00		EA15FF00		BF40FF00
	BB44FF00		BC43FF00		E916FF00
	E619FF00		F20DFF00		F30CFF00
	E718FF00		A15EFF00		F708FF00
	E31CFF00		A55AFF00		AD52FF00
	BD42FF00		B54AFF00		

Infračervený přijímací modul



V tomto kurzu používáme infračervený přijímací modul, který se skládá z infračerveného přijímače, napájení a dalších elektronických součástí. Funkcí tohoto modulu je přijímat infračervené světlo vysílané infračerveným dálkovým ovladačem a převádět infračervené světlo na odpovídající úroňový (logický) signál, který je vstupem do řídicí jednotky. Řídicí jednotka pak pomocí programového algoritmu tento signál převede na hodnotu odpovídající kódu tlačítka.





Poznámka: Infračervený dálkový ovladač musí být natočen směrem k infračervenému přijímači, aby mohl vyslat signál, jinak infračervený přijímač signál obtížně zachytí.



V Arduinu používáme knihovnu „IRremote.h“ pro realizaci infračervené komunikace, základní příkazy jsou následující:



```
// Create an infrared receiver object and specify the pins of the infrared receiver
IRrecv myIRrecv(IRpin);
// turn on the infrared receiving function
myIRrecv.enableIRIn();
//whether the infrared signal is received and decoded
myIRrecv.decode()
//get the received infrared code
MyIRrecv.decodedIRData.decodedRawDate
//whether to send duplicate codes
myIRrecv.decodedIRData.flags
//Continue to receive infrared signals
myIRrecv.resume();
```

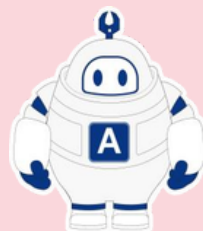




ITS Laboratoř

Auto ovládané infračerveným
dálkovým ovladačem

Popis projektu



Použij infračervený dálkový ovladač k dálkovému ovládání chytrého auta SharkBot, aby se pohybovalo vpřed, vzad, vlevo a vpravo.



Hardwarové schéma

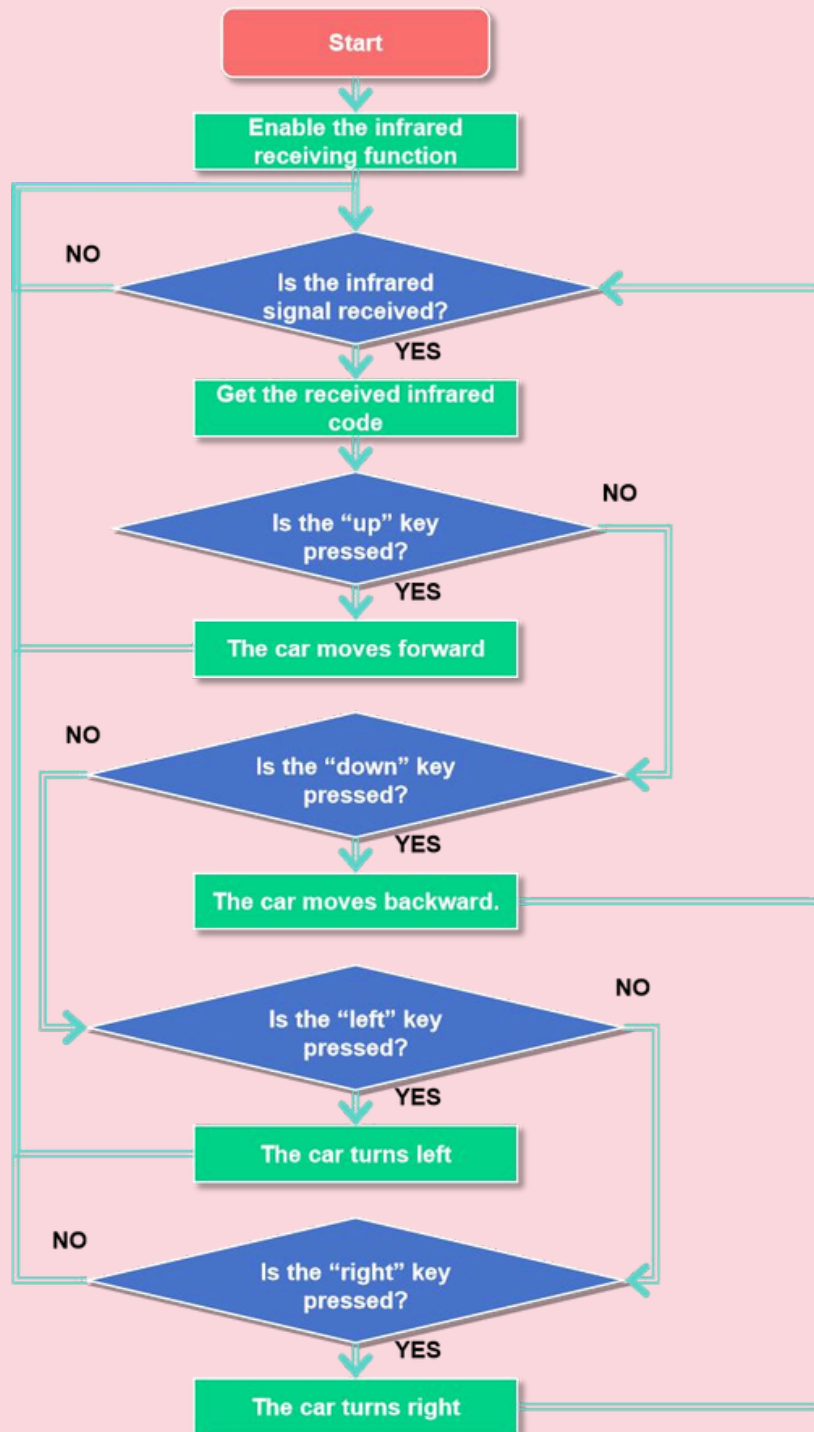
1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	IR dálkový ovladač	1
	Type-C Data kabel	1



Programování

1. Program Flowchart





Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\11IRremote](#)

```
#include <IRremote.h>
#define RightMotor_A1 12 // Define the pin number of the motor driver
pin #define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define IRpin 11 // Define the pins of the infrared receiver module

unsigned long lastCommandTime; // the time of the last received code
const unsigned long commandTimeout = 100; // timeout
uint32_t last_decode = 0; // the last encoding
uint32_t current_decode = 0; // the current decode
IRrecv myIRrecv(IRpin);
void setup() {
    pinMode(RightMotor_A1,OUTPUT); //Set the working mode of the input pins
of the driver board
    pinMode( RightMotor_A2,OUTPUT );
    pinMode( LeftMotor_A1 ,OUTPUT) ;
    pinMode( LeftMotor_A2 ,OUTPUT) ;
    myIRrecv.enableIRIn(); // enable infrared reception
    Serial.begin (115200 );
}
void loop() {
    if (myIRrecv.decode()) { //Has the infrared code been received?
        lastCommandTime = millis();
        current_decode = myIRrecv.decodedIRData.decodedRawData;
        if (myIRrecv.decodedIRData.flags) { //Is it a repeated code?
            current_decode = last_decode; // Update the current code.
        }
        Serial.print(current_decode, HEX);
        Serial.println ("") ;
        switch (current_decode) {
            case 0xB946FF00: // If it is the code value of the "up" key.
                motorControl(240,255); //The car moves forward.
                break ;
            case 0xEA15FF00: // If it is the code value of the "down" key.
```





```

        motorControl(-240,-255); //The car reverses
        break ;
    case 0xBB44FF00: // If it is the code value of the "left" key.
        motorControl(240,-255); //The car turns left.
        break ;
    case 0xBC43FF00: // If it is the code value of the "right" key.
        motorControl(-240,255); //The car turns right.
        break ;
}

last_decode = current_decode; // Update the previous code value.
myIRrecv.resume(); // Continue to receive infrared codes.
}

if (millis() - lastCommandTime > commandTimeout) {
    motorControl(0,0); //If thereceptiontimetimesout,thecarstops.
    delay (5);
}
}

void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite (LeftMotor_A 1,left_pwm);
        analogWrite (LeftMotor_A 2,0) ;
    }
    else{
        analogWrite (LeftMotor_A 1,0) ;
        analogWrite (LeftMotor_A 2,-left_pwm);
    }
    if(right_pwm > 0){
        analogWrite (RightMotor_ A1,right_pwm);
        analogWrite (RightMotor_ A2,0 );
    }
    else{
        analogWrite (RightMotor_ A1,0 );
        analogWrite (RightMotor_ A2,-right_pwm);
    }
    delay(5) ;
}

```





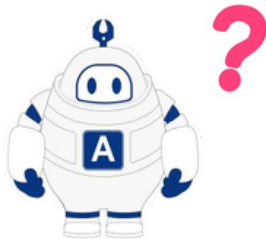
Oddíl 2: Technologie Bluetooth komunikace



Cíle kurzu



-  Pochopit princip Bluetooth komunikace.
-  Zvládnout používání Bluetooth modulu ESP32.
-  Navrhnout funkci Bluetooth ovládání pro auto SharkBot.



Když je před infračerveným dálkovým ovladačem překážka, nemůže dálkové ovládání fungovat? Jak lze realizovat bezbariérovou dálkovou komunikaci?

V předchozí lekci jsme použili technologii infračervené komunikace k realizaci dálkového ovládání chytrých aut. Infračervená komunikace má však špatnou schopnost průchodu překážkami a jakmile překážka zablokuje infračervený vysílač, infračervený přijímač

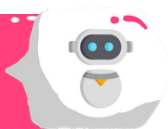
nemůže signál přijmout. Při použití dálkového ovladače k řízení chytrého auta je nutné mířit přímo na infračervený přijímač, což ztěžuje obsluhu. Aby to pro nás bylo pohodlnější





pro ovládání auta můžeme použít technologii Bluetooth komunikace. Dále se seznámíme s možnostmi a principy Bluetooth technologie.

Bluetooth komunikace



Bluetooth komunikace je bezdrátový technologický standard pro výměnu dat na krátké vzdálenosti. Umožňuje komunikaci mezi různými zařízeními bez potřeby fyzického propojení. Tato komunikace dokáže pronikat stěnami a dalšími překážkami, takže zařízení mohou komunikovat i v různých místnostech.

K dnešnímu dni existuje několik verzí Bluetooth protokolu: Bluetooth 2.0, Bluetooth 2.1, Bluetooth 3.0, Bluetooth 4.0, Bluetooth 4.2, Bluetooth 5.0 atd.

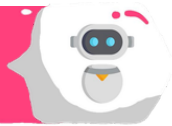
Bluetooth komunikace se podle způsobu provozu dělí na BT Bluetooth (klasický Bluetooth) a BLE Bluetooth (Bluetooth s nízkou spotřebou). BT Bluetooth podporuje vyšší přenosovou rychlost a je vhodný pro aplikace vyžadující nepřetržitý datový tok, například přenos zvuku a videa.

BLE Bluetooth funguje od verze Bluetooth 4.0 a novější. Oproti BT Bluetooth má velmi nízkou spotřebu energie, ale také nižší přenosovou rychlost. Je vhodný pro přenos malého množství dat, například pro monitorování senzorů.





Kroky Bluetooth komunikace



V této lekci budeme používat BLE Bluetooth komunikaci. Zařízení s nízkou spotřebou energie komunikují pomocí broadcast mechanismu, který má následující kroky:

Vyhledání zařízení a připojení

Vysílání: podřízené zařízení (slave) aktivně vysílá broadcast paket, aby oznámilo svou přítomnost. Skenování: hlavní zařízení (master) skenuje okolní vysílající zařízení.

Připojení: hlavní zařízení odešle požadavek na připojení a naváže spojení s podřízeným zařízením.

Přenos dat

Po navázání spojení může hlavní zařízení číst nebo zapisovat data z/do podřízeného zařízení. Podřízené zařízení může také aktivně odesílat data hlavnímu zařízení pomocí notifikací nebo příkazů.

Odpojení

Po dokončení komunikace se může hlavní nebo podřízené zařízení aktivně odpojit. V Arduinou použijeme knihovnu „ACB_BLEDevice.h“ pro realizaci BLE komunikace.





Bluetooth funkce ESP32. Základní instrukce jsou následující:

```
// Create a BLE object and set the service UUID and feature UUID
ACB_BLEDevice BLE(SERVICE_UUID,CHARACTERISTIC_UUID);

//Turn on BLE broadcasting
BLE.Bluetooth_Name("SharkBot");
//Listening for Bluetooth device access
BLE.deviceConnected
//Receiving data
BLE.ServerReceived();
//get the received data
BLE.ServerData
```





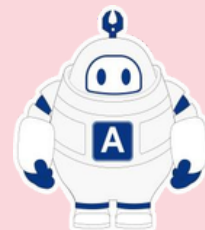
ITS Laboratoř



Bluetooth Komunikace



Popis projektu



Programování pro realizaci ovládání pohybu chytrého auta pomocí Bluetooth komunikace z mobilního telefonu.



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1



Programování

2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici

[English\Arduino\(Experienced Learner\)\3.Arduino program\12Bluetooth](#)

```
#include <ACB_BLEDevice.h>

#define RightMotor_A1 12
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
//Define the service UUID and characteristic UUID.
#define SERVICE_UUID "6e400001-b5a3-f393-e0a9-e50e24dcca9f"
#define CHARACTERISTIC_UUID "6e400002-b5a3-f393-e0a9-e50e24dcca9f"
ACB_BLEDevice BLE(SERVICE_UUID,CHARACTERISTIC_UUID);

void setup() {
    Serial .begin( 115200);
    BLE. Bluetooth_Nam e("SharkBot" );
    pinMode(RightMotor_A1, OUTPUT); //Set the working mode of the input
pins of the driver board.
    pinMode(RightMotor_A2, OUTPUT);
    pinMode(LeftMotor_A1, OUTPUT);
    pinMode(LeftMotor_A2, OUTPUT);
}

void loop() {
    if (BLE.deviceConnected){//Listen for any Bluetooth connections.
        BLE.ServerReceived();//Receive Bluetooth data.
        if (BLE.ServerData == "forward") {//Has "forward" been received?
            motorControl(255, 255);
        } else if (BLE.ServerData == "backward") {//Has "backward" been
received?
            motorControl(-255, -255);
        } else if (BLE.ServerData == "left"){//Has "left" been received?
```





```
        motorControl(-120, 120);
    } else if (BLE.ServerData == "right") { //Has "right" been received?
        motorControl(120, -120);
    } else if (BLE.ServerData == "stop") { //Has "stop" been received?
        motorControl(0, 0);
    }
}
}

//Motor control function: when PWM > 0, the motor moves forward; when
//PWM < 0, the motor moves backward; when PWM = 0, the motor stops.
void motorControl(int left_pwm, int right_pwm) {
    if (left_pwm > 0) {
        analogWrite(LeftMotor_A1, left_pwm);
        analogWrite(LeftMotor_A2, 0);
    } else {
        analogWrite(LeftMotor_A1, 0);
        analogWrite(LeftMotor_A2, -left_pwm);
    }
    if (right_pwm > 0) {
        analogWrite(RightMotor_A1, right_pwm);
        analogWrite(RightMotor_A2, 0);
    } else {
        analogWrite(RightMotor_A1, 0);
        analogWrite(RightMotor_A2, -right_pwm);
    }
    delay( 5);
}
```





APP control

After uploading the Bluetooth program of the cart, we need to control it through APP. The steps are as follows:

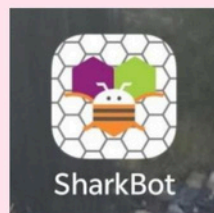
1.APP Installation

Find "SharkBot.apk" in the attached file "English\Arduino(Experienced Learner)\6.APP", upload the package to your phone and double click it to install.

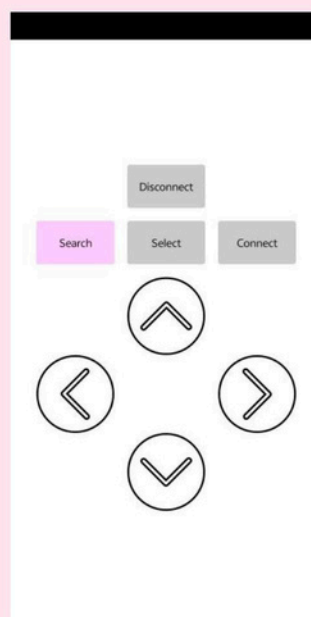
Note: Since this APP is written by APP INVENTOR, it only supports Android system cell phone, not IOS system cell phone.

2.APP control

(1)After APP installation, the icon is as follows:

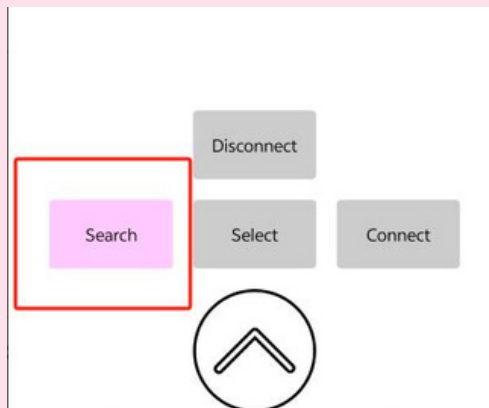


(2) Click on the icon to enter the APP page.

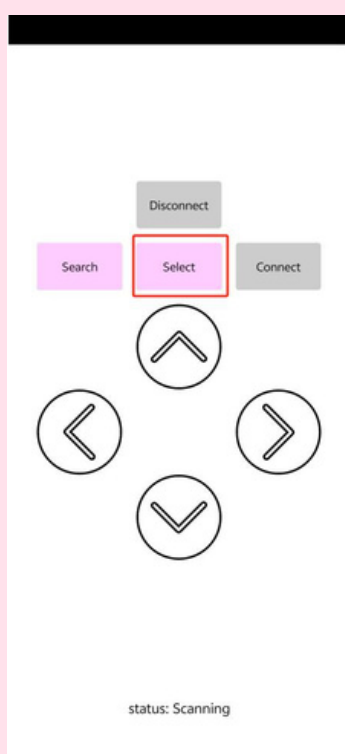




(3) Klikněte na tlačítko „Hledat“ pro vyhledání Bluetooth.



(4) Po dokončení vyhledávání Bluetooth se tlačítko „Vybrat“ zbarví červeně. Poté klikněte na tlačítko „Vybrat“ a vstupte na stránku seznamu Bluetooth zařízení.



SharkBot
52:05:48:2D:DB:84 null -45
2C:58:B9:7A:90:29 null -62
51:6C:4C:A6:13:D9 null -63
75:67:4A:4B:F5:D2 null -63
DA:16:9E:4E:86:89 null -66
3C:84:27:27:85:61 SharkBot -70
6B:12:96:F0:22:C9 null -71
E0:81:28:03:3A:24 null -73
F4:AF:39:F4:C6:E3 null -73
EF:0B:71:98:A0:EE null -78
E7:BC:1A:84:2B:66 null -80
77:5F:07:7B:CF:40 null -80

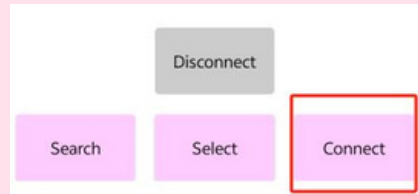
(5) Vyberte Bluetooth zařízení, které obsahuje text „SharkBot“.

3C:84:27:27:85:61 SharkBot -70

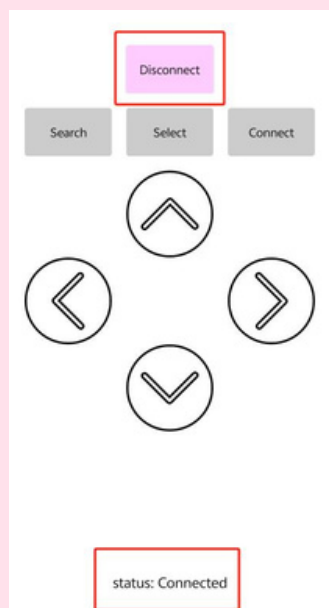




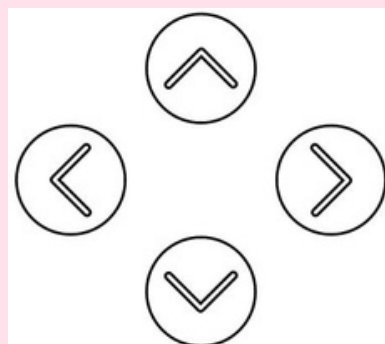
(6) Klikněte na tlačítko „Připojit“ pro připojení k Bluetooth.



(7) Po úspěšném připojení Bluetooth se tlačítko „Odpojit“ zbarví červeně a stav „Status:“ ve spodní části se změní na „Připojeno“.



(8) Po úspěšném připojení Bluetooth můžete ovládat SharkBot pomocí směrových šipek.



(9) Klikněte na tlačítko „Odpojit“ pro ukončení Bluetooth připojení.





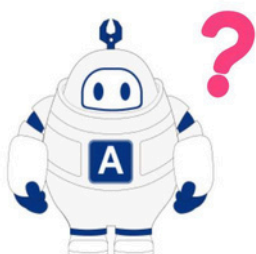
Oddíl 3: Technologie WiFi komunikace



Cíle kurzu



-  Pochopit koncept WiFi komunikace.
-  Pochopit WiFi funkce ESP32.
-  Pochopit koncept protokolu TCP.
-  Navrhnout WiFi řídicí funkci pro chytré auto.
-  Používat aplikaci ACEBOTT APP pro ovládání chytrého auta.



Have you ever used a product with WiFi capability? How are they WIFI enabled?

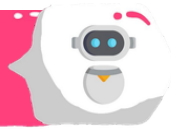
WiFi je všudypřítomná v našem životě, nejběžnější zařízení s WiFi funkcí jsou mobilní telefony a počítače. Doma obvykle používáme router s WiFi funkcí, který vytváří bezdrátovou síť, a poté připojujeme mobilní telefon a počítač ke stejné WiFi síti. Tímto způsobem mobilní telefon a počítač nepotřebují připojení k internetovému kabelu, aby mohly mít funkci





Internetu. Ve skutečnosti dnes stále více aut disponuje funkcí WiFi, pomocí které lze prostřednictvím WiFi komunikační technologie realizovat dálkové ovládání chytrého auta pomocí mobilního telefonu. Jak to funguje? Pojďme to společně rozluštit.

WiFi Komunikace



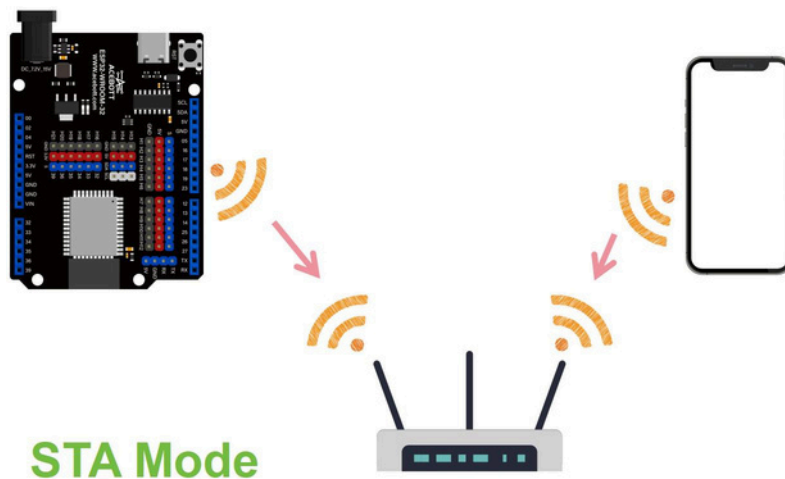
Technologie WiFi komunikace je bezdrátová technologie lokálních sítí (WLAN), která umožňuje elektronickým zařízením, jako jsou chytré telefony, tablety, notebooky apod., bezdrátově se připojit k internetu nebo lokální síti (LAN). WiFi je zkratka pro „Wireless Fidelity“ a je založena na standardu IEEE 802.11, který poskytuje bezpečné, spolehlivé a rychlé bezdrátové připojení.

WiFi komunikační technologie propojuje zařízení do stejné sítě prostřednictvím bezdrátového routeru nebo přístupového bodu (AP), podobně jako vytvoření „silnice“ mezi jednotlivými zařízeními, která umožňuje síťovou komunikaci. V rámci WiFi sítě mohou zařízení mezi sebou přijímat a odesílat data.

WiFi funkce ESP32 pracuje ve třech režimech:

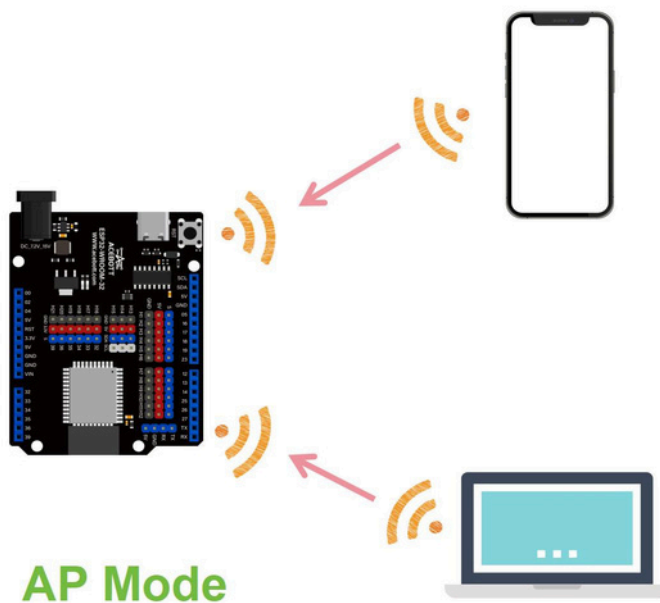
V tomto režimu funguje ESP32 jako koncové zařízení připojené k WiFi síti vytvářené jinými zařízeními.





2. Režim přístupového bodu (AP režim)

V tomto režimu funguje ESP32 jako WiFi vysílač, který vytváří vlastní WiFi síťový signál, ke kterému se mohou připojit ostatní zařízení.



3. Režim stanice + přístupového bodu

V tomto režimu může ESP32 fungovat buď jako koncové zařízení připojené k jiné WiFi síti, nebo jako přístupový bod, který vytváří WiFi síť pro ostatní zařízení.



Komunikace WiFi pouze na hardwarové úrovni otevírá komunikační kanál mezi zařízeními, ale zatím neumožňuje samotný přenos dat. Protože v síti není jednoznačná identifikace zařízení, zařízení neví, komu má data odeslat ani odkud je má přijímat.

Tento proces je velmi podobný psaní dopisů přátelům. Když píšeme dopis, musíme znát adresu příjemce, aby dopis dorazil správné osobě, a zároveň musíme uvést svou vlastní adresu, aby nám mohl být dopis odpovězen. Stejně tak i v počítačové síti má každé zařízení svou jedinečnou adresu a přenos dat probíhá mezi těmito adresami. Tato síťová adresa se nazývá IP adresa.

V této lekci nastavíme základní desku ESP32 u SharkBotu do režimu AP (přístupového bodu) a budeme čekat, až se k síti SharkBotu připojí mobilní telefon.

IP adresa



IP adresa každého síťového zařízení je přidělována IP protokolem (Internet Protocol), který je základem internetové komunikace. IP protokol se dělí na dvě verze, IPv4 a IPv6. IPv6 je vylepšená verze IPv4, která může poskytovat větší počet IP adres. V tomto kurzu používáme protokol IPv4, proto se zaměříme na IPv4.

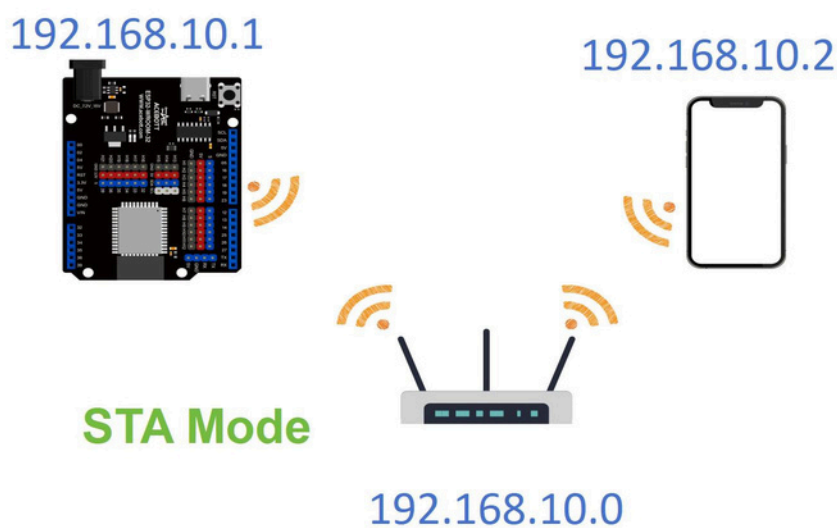
IPv4 adresa se skládá ze 4 čísel, například:





192.168.10.136 – tato adresa se skládá ze dvou částí. První tři čísla (192.168.10) představují adresu WiFi sítě, ke které je zařízení připojeno, a poslední číslo (136) je číslo zařízení v rámci této WiFi sítě.

Například nyní máme základní desku ESP32 a mobilní telefon, které jsou připojeny k WiFi síti routeru. Vestavěný DHCP server routeru přiřadí každému ze tří zařízení samostatnou IP adresu. Předpokládejme, že IP adresa routeru je 192.168.10.0, IP adresa desky ESP32 je 192.168.10.1 a IP adresa mobilního telefonu je 192.168.10.3.



Jakmile mají zařízení IP adresy, mohou pomocí těchto adres přenášet data. Aby byla zajištěna přesnost dat během přenosu, můžeme použít různé přenosové protokoly, jako jsou HTTP, TCP nebo UDP. V této části se zaměříme na protokol TCP.



TCP Protokol



TCP je spojově orientovaný a spolehlivý protokol, který zajišťuje, že pakety dorazí na přijímací stranu ve správném pořadí. Pokud jsou ztraceny nebo poškozeny, TCP je znovu odešle. TCP se široce používá v aplikacích vyžadujících vysokou spolehlivost, jako je prohlížení webu, e-mail nebo přenos souborů.

TCP přenos vyžaduje vytvoření serveru a klienta pro realizaci komunikace mezi nimi. V tomto kurzu používáme základní desku ESP32 v SharkBotu jako server a mobilní telefon jako klient, aby bylo možné realizovat datovou komunikaci mezi telefonem a ESP32 pomocí TCP protokolu.

Přenos dat přes TCP se dělí na tři kroky:

1. Navázání TCP spojení

Navázání TCP spojení je prvním krokem pro realizaci TCP komunikace ESP32. ESP32 jako server musí čekat na požadavek na připojení od klienta na určeném portu, zatímco mobilní telefon jako klient musí aktivně navázat spojení se serverem.

2. Odesílání a příjem dat

Jakmile je TCP spojení navázáno, může mezi klientem a serverem probíhat přenos dat.





3. Odpojení

Když již bezdrátovou komunikaci nepotřebujeme, můžeme TCP spojení odpojit, aby se šetřily systémové prostředky.

V Arduinu používáme knihovnu „WiFi.h“ pro realizaci WiFi komunikace, hlavní instrukce jsou následující:

```
// Create a server object and set the server port number
WiFiServer server(100);

//create a client object
WiFiClient client;

//Set the bandwidth of the WiFi signal
WiFi.setTxPower(WIFI_POWER_19_5dBm);

//Set WiFi to AP mode
WiFi.mode(WIFI_AP);

//Set the WiFi name and WiFi password
WiFi.softAP("SharkBot", "12345678", 5)

// Turn on the web server
server.begin();

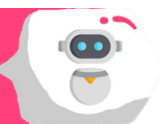
//Wait for client connection, return true if there is client connection, otherwise return
false
server.available()

//whether the client is connected
client.connected()

//whether the client sends data
client.available()

// read the data from the client
client.read()
```

ACEBOTT APLIKACE



Pro ovládání SharkBotu pomocí mobilního telefonu je také potřeba nainstalovat aplikaci ACEBOTT APP do mobilního telefonu a použít ji





ACEBOTT APP k ovládání pohybu chytrého auta. Konkrétní kroky jsou následující:

Pokud používáte telefon se systémem iOS, je potřeba vyhledat klíčové slovo „ACEBOTT“ v App Store a následně aplikaci stáhnout. Pokud používáte telefon se systémem Android, je potřeba vyhledat klíčové slovo „ACEBOTT“ v Google Play Store a následně aplikaci stáhnout. Ikona je zobrazena vpravo.

Formát paketu



Když aplikace ACEBOTT ovládá pohyb SharkBotu prostřednictvím WiFi komunikace, neodesílá se pouze jeden údaj, ale řetězec dat ve fixním formátu.

Konkrétní formát je následující:

Data	255	85	10	0	0	0	0	0	0	1	14	0	1
Index	0-1	2	3-8				9	10	11	12			

Jak je uvedeno v tabulce výše, aplikace ACEBOTT APP ovládá SharkBot odesíláním celkem 13 dat, která mají různé významy.

Bit 0–1: hlavička paketu, která oznamuje komunikujícímu zařízení začátek přenosu dat.

Bit 2: udává délku následně odesílaných dat.

Bity 3–8: ID příkazu pro ovládání vozítka SharkBot.

Bit 9: udává typ funkce SharkBotu, například základní pohyb, sledování čáry, vyhýbání se překážkám atd.





10. pozice: ovládání modulů na autě, např. řízení motoru, ovládání bzučáku.

Bit 11: číslo komunikačního portu.

Bit 12: udává konkrétní akci, kterou má SharkBot provést, např. jízda vpřed nebo vzad.

Pokud chceme ovládat pohyb SharkBotu pomocí ACEBOTT APP, odpovídající datový paket je následující:

PacketssentbytheAPP	SharkBotaction
255851000000011401	Forward
255851000000011402	Backup
255851000000011403	Totheleft
255851000000011404	Totheright
255851000000011400	Stop

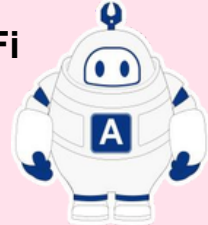




ITS laboratoř



Ovládání chytrého auta pomocí WiFi signálu



Popis projektu

Ovládejte pohyb chytrého auta pomocí mobilního telefonu prostřednictvím technologie WiFi komunikace.



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1





Programování

2. Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

[English\Arduino\(Experienced Learner\)\3.Arduinoprogram\13APPControl1](#)

```
#include <WiFi.h>
#include <Arduino.h>

const char *ssid = "Sharkbot";          // Set the WiFi name.
const char *password = "12345678";     // Set the WiFi password.
WiFiServer server(100);                 // Create a server object and set the
port.
WiFiClient client;                      // Create a client object.
char buffer[52];                        // An array for storing data.
byte dataLen, index_a = 0;

bool isStart = false;                  // A flag for receiving data.
unsigned char prevc = 0;                // The previous data.
byte action, device, val;              // The command data of the car.
bool st = false;                       // Communication status.
const unsigned long timeout = 3000;    // Communication timeout period.

#define RightMotor_A1 12
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
unsigned char readBuffer(int index_r) { // Read data.
    return buffer[index_r];
}
void writeBuffer(int index_w, unsigned char c) { // Write data.
    buffer[index_w] = c;
}

void setup() {
    Serial.begin(115200);
}
```





```

WiFi.setTxPower(WIFI_POWER_19_5dBm); //Set the WiFi bandwidth.
WiFi.mode(WIFI_AP); //Set WiFi to AP mode.
WiFi.softAP(ssid, password, 5); //Create a WiFi network and set the WiFi
name and password.
delay( 100);
server.begin(); //Start the server.
pinMode(RightMotor_A1,      OUTPUT);
pinMode(RightMotor_A2,      OUTPUT);
pinMode(LeftMotor_A1,       OUTPUT);
pinMode(LeftMotor_A2, OUTPUT);
}

void loop() {
  client = server.available(); // Wait for client connection.
  if (client) //If there is a client connection.
  {
    Serial.println("[Client connected]");
    unsigned long previousMillis = millis(); // Record the connection
time point.
    while (client.connected()) // If the client remains connected.
    {
      if((millis() -previousMill is) >timeout&&client . available() ==0){
        break; //If the communication time times out, exit the process.
      }
      if (client.available()) // If the client sends data.
      {
        previousMillis = millis(); //Record the current time point.
        unsigned char c = client.read() & 0xff; // Read the data sent by
the client.
        if (c == 0x55 && !isStart && prevc == 0xff) { //Has 85 been received?

          index_a = 1;
          isStart = true; //Set the flag for starting to receive data to
True.
        } else {
          prevc = c;
          if (isStart) { //If starting to receive data.
            if (index_a == 2) { //If it is the third piece of data.
              dataLen = c; //Set the length of the subsequent data.
            } else if (index_a > 2) {

```





```
        dataLen--; //Starting from the third piece of data, decrease
the data length by one for each received data item.
    }
    writeBuffer(index_a, c);
    if (dataLen == 0 && index_a > 3) { // Data reception completed.
        isStart = false;
        parseData(); //Analyze the data.
        index_a = 0;
    }
}
}
index_a++;
}
}
client.stop(); //Disconnect the client connection.
motorControl(0, 0); // Stop the car.
Serial.println("[Client disconnected]");
}
}

void parseData() {
    action = readBuffer(9); //Get the 9th piece of data.
    device = readBuffer(10); //Get the 10th piece of data.
    val = readBuffer(12); //Get the 12th piece of data.
    if (action == 0x01 && device == 0x0e) {
        switch (val) {
            case 0x01:
                motorControl(190, 190);
                break ;
            case 0x02:
                motorControl(-190, -190);
                break ;
            case 0x03:
                motorControl(-90, 90);
                break ;
            case 0x04:
                motorControl(90, -90);
                break ;
            case 0x00:

```





```
        motorControl(0, 0);  
        break ;  
    default :  
        break ;  
    }  
}  
}
```

//control motor function, pwm is greater than 0, motor forward, pwm is less than 0, motor backward, pwm is equal to 0, motor stops, //control motor function, pwm is greater than 0, motor forward, pwm is less than 0, motor backward, pwm is equal to 0, motor stops

```
void motorControl(int left_pwm, int right_pwm) {  
    if (left_pwm > 0) {  
        analogWrite(LeftMotor_A1, left_pwm);  
        analogWrite(LeftMotor_A2, 0);  
    } else {  
        analogWrite(LeftMotor_A1, 0);  
        analogWrite(LeftMotor_A2, -left_pwm);  
    }  
    if (right_pwm > 0) {  
        analogWrite(RightMotor_A1, right_pwm);  
        analogWrite(RightMotor_A2, 0);  
    } else {  
        analogWrite(RightMotor_A1, 0);  
        analogWrite(RightMotor_A2, -right_pwm);  
    }  
    delay(5) ;  
}
```





APP control

After uploading the WiFi program of the cart, we need to connect our cell phone to the WiFi network of the cart and control it via ACEBOTT APP.

1. WiFi Connection

Power on the SharkBot, turn on your cell phone's WiFi search function and find the WiFi named "SharkBot" with the password "12345678".

2. APP Connection

(1) Open the installed ACEBOTT APP to enter the home page.



(2) Select SharkBot.



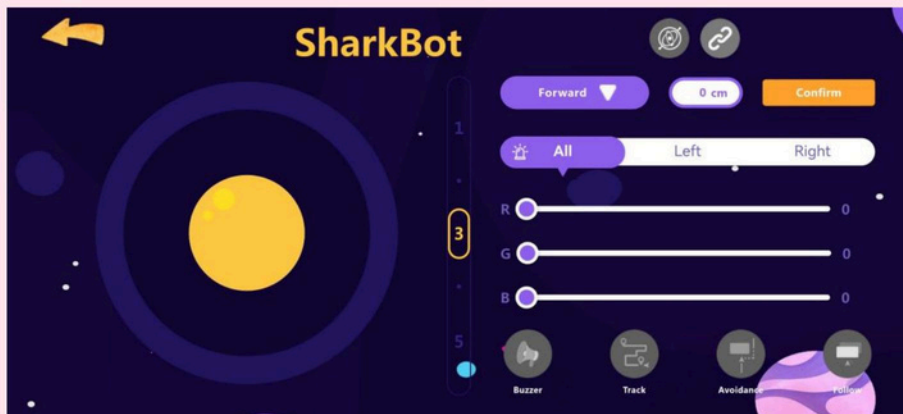


APP control

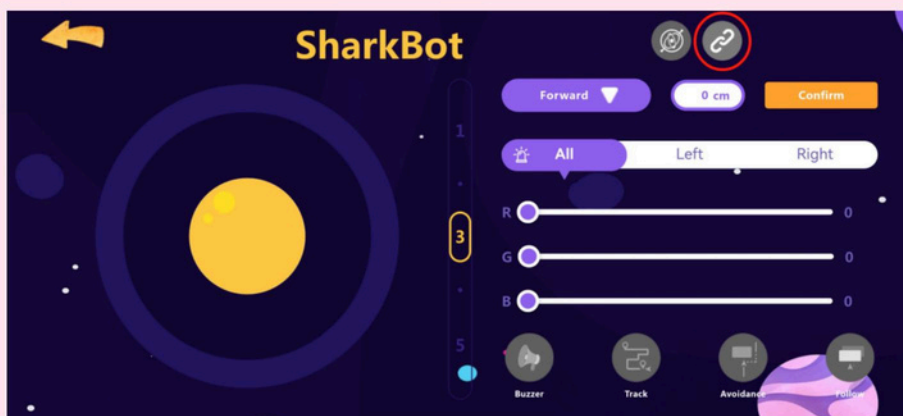
(3) Click the SharkBot icon, then select the Control button.



(4) Click the Control icon to enter the control interface of SharkBot.



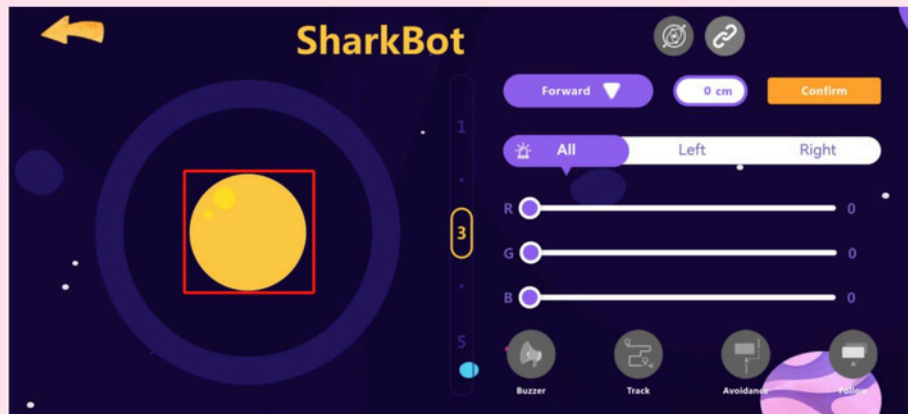
(5) Click the connection icon in the upper right corner to connect the SharkBot.





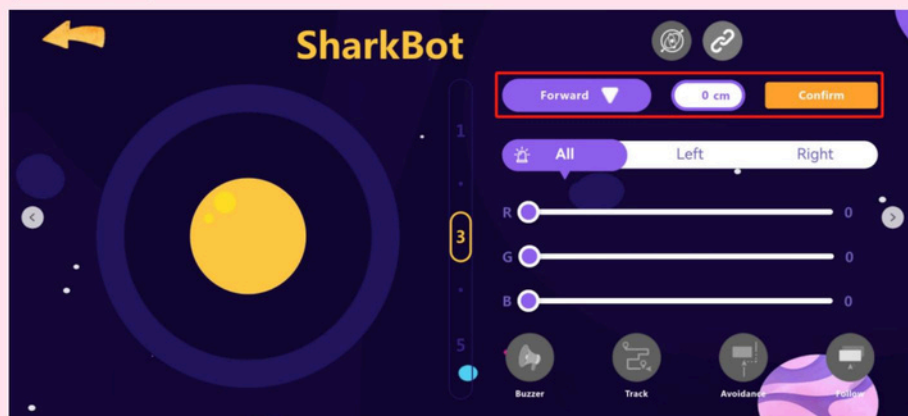
APP control

(6) Manipulate the motion of the smart car through the joystick of the control interface.



Note:The programs in this course are only configured for joystick control of SharkBot. If you want to use all the functions on the app, upload the program in "[English\Arduino\(Experienced Learner\)\3.Arduino_program\14APPControl2\14APPControl2.ino](#)" to SharkBot. Then, other control methods will be unlocked.

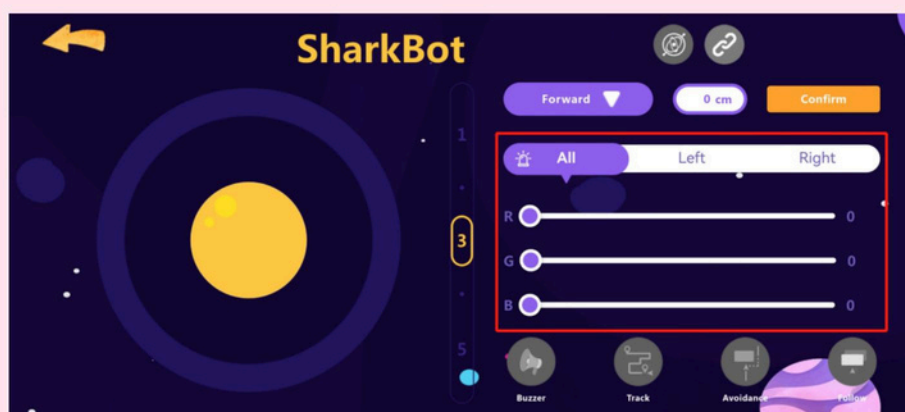
(7) Move the smart car by Specified Distance or Rotate by Specified Angle via Keyboard Input .



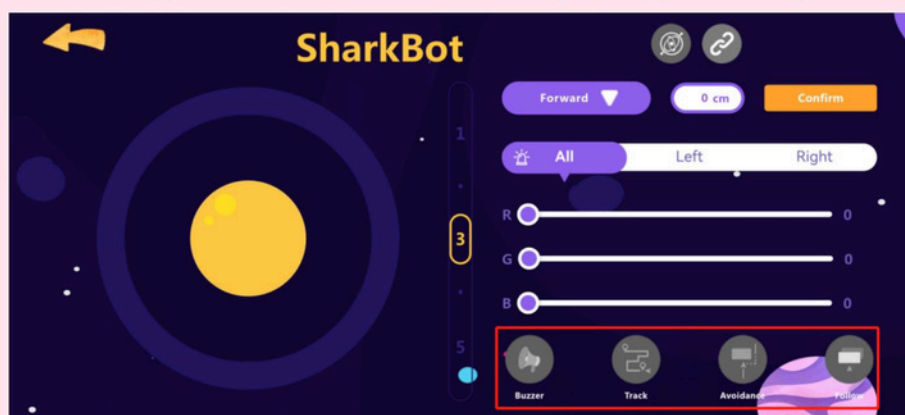


APP control

(8) smart car Headlight Settings .Configure the headlights separately for "All," "Left," and "Right." Adjust the headlight colors using the three sliders below.



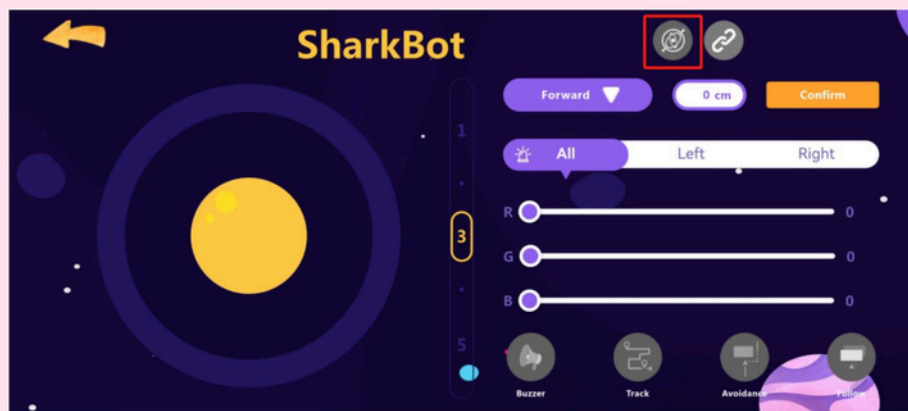
(9) Four buttons are arranged at the bottom, which can be used to activate the following functions of the smart car respectively: music playback, line - following, obstacle avoidance, and following. It is important to note that once the smart car enters the "Track", "Avoidance", or "Follow" mode, it will keep performing the corresponding function in a continuous loop. If you want to exit the function, you need to click the corresponding button again.





APP control

(10) In the upper - right corner of the smart car's operation interface, a gyroscope control option is provided. After clicking this button, you can control the movement of the vehicle through the gyroscope on your mobile phone. If your mobile phone does not have a built - in gyroscope, you can disregard this function.





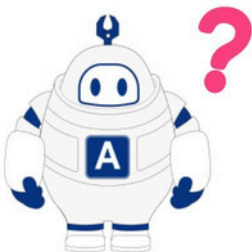
Oddíl 4: Internet vozidel



Cíle kurzu



-  Pochopit koncept Internetu vozidel (IoV).
-  Pochopit klíčové technologie Internetu vozidel.
-  Pochopit princip protokolu ESP-NOW.
-  Programově realizovat interakci mezi chytrými auty a semaforey.



How can we improve the efficiency of transportation during rush hour?

Ve velkých městech s velkým počtem obyvatel často dochází ve špičce k dopravním zácpám. Pokud by auto mohlo předem znát stav silnic a získat doporučenou optimalizovanou trasu, a semaforey by se navíc automaticky přizpůsobovaly délce dopravního proudu, mohlo by to výrazně snížit dopravní zácpy. K realizaci tohoto je potřeba využít technologii propojených vozidel. Co je tedy technologie propojených vozidel? Pojdme to nyní prozkoumat.





The concept of Internet of Vehicles



Internet vozidel (IoV) označuje propojení vozidel, silniční infrastruktury, systémů řízení dopravy a dalších prvků prostřednictvím bezdrátových technologií do inteligentní dopravní komunikační sítě.

V rámci IoV mohou autonomní vozidla nejen



vnímat okolní prostředí, ale také komunikovat s inteligentními semaforey, ostatními vozidly a cloudovými platformami. Například když se autonomní vozidlo přiblíží ke křižovatce, může interagovat s inteligentními semaforey a získat informace o stavu světelné signalizace, a tím upravit rychlost jízdy, aby se vyhnulo projetí na červenou nebo dlouhému čekání.

Mezi hlavní funkce Internetu vozidel patří:

Interakce informací v reálném čase: sdílení informací mezi vozidly (V2V), mezi vozidly a infrastrukturou (V2I) a mezi vozidly a cloudem (V2C).

Inteligentní řízení dopravy: optimalizace dopravního toku prostřednictvím analýzy dat za účelem snížení dopravních zácp a nehod.

Podpora autonomního řízení: poskytování podpory pro vnímání prostředí, plánování trasy a rozhodování pro autonomní vozidla.





ESP-NOW Protokol



V aplikačních scénářích Internetu vozidel je často potřeba zajistit komunikaci mezi více zařízeními. TCP protokol, který jsme probírali v předchozí lekci, však podporuje pouze komunikaci typu bod–bod, což nevyhovuje našim požadavkům. Proto v následujících lekcích použijeme protokol ESP-NOW k realizaci komunikace mezi vozem SharkBot a dalšími zařízeními.

ESP-NOW je bezdrátová technologie, která umožňuje komunikaci mezi dvěma čipy ESP32 bez nutnosti WiFi připojení. Vyvinula ji společnost Espressif Systems a je vhodná pro scénáře vyžadující krátkou vzdálenost, nízkou spotřebu energie a rychlou komunikaci.

Můžeme si ji představit jako dvě vysílačky, které se „znají“ podle své identity (MAC adresy) a mohou spolu přímo komunikovat bez připojení k internetu.

Protokol ESP-NOW také podporuje mechanismus vysílání (broadcast), který umožňuje komunikaci „jeden k mnoha“, což je velmi vhodné pro potřeby Internetu věcí i Internetu vozidel.

Pracovní postup vysílače při broadcast komunikaci ESP-NOW je následující:

1. Nastavení WiFi režimu na STA režim
2. Režim STA umožňuje zařízení odesílat nebo přijímat data, ale nepřipojuje se k žádné síti Wi-Fi.





```
WiFi.mode(WIFI_STA);
```

2. Inicializujte ESP-NOW

Inicializace komunikačního protokolu ESP-NOW a aktivace souvisejících interních funkcí.

```
if (esp_now_init() != ESP_OK) {  
    Serial.println("Error initializing ESP-NOW");  
    return;}  
}
```

3. Registrace funkce zpětného volání pro odesílání (volitelné)

Po odeslání dat ESP32 automaticky zavolá tuto funkci, která poskytne zpětnou vazbu o tom, zda byl přenos úspěšný.

```
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)  
{  
    Serial.print("\r\nLast Packet Send Status:\t");  
    Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success" : "Fail");  
}  
esp_now_register_send_cb(OnDataSent);
```

4: Konfigurace nastavení zařízení

Před vysíláním dat je potřeba nakonfigurovat zařízení, včetně jeho adresy a dalších nezbytných parametrů. Při realizaci broadcast komunikace pomocí ESP-NOW pro odesílání dat není nutné zadávat MAC adresu konkrétního zařízení. Místo toho se používá broadcast adresa FF:FF:FF:FF:FF:FF, která označuje, že data jsou odesílána všem zařízením.





```
uint8_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

esp_now_peer_info_t peerInfo;
memcpy(peerInfo.peer_addr, broadcastAddress, 6);
peerInfo.channel = 0;
peerInfo.encrypt = false;
if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    ESP.restart();
}
```

Odesílání dat

Přeneste data na cílové zařízení (nebo zařízení).

```
esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData))
```

Pracovní postup přijímače pro broadcast komunikaci ESP-NOW:

Nastavení WiFi režimu na STA režim

Režim STA umožňuje zařízení odesílat nebo přijímat data bez připojení k WiFi síti.

```
WiFi.mode(WIFI_STA);
```

1. Inicializace ESP-NOW

Inicializujte komunikační protokol ESP-NOW pro přípravu na příjem dat.

```
if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;}

```

3.Registrace funkce zpětného volání pro příjem

Zaregistrujte callback funkci, která se automaticky spustí při přijetí dat. V této funkci lze následně přijatá data zpracovat.

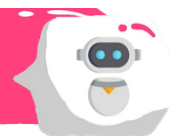




```
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {  
    memcpy(&receivedData, incomingData, sizeof(receivedData));  
}  
esp_now_register_recv_cb(OnDataRecv);
```

V Arduino se pro realizaci komunikace pomocí protokolu ESP-NOW používají knihovny „WiFi.h“ a „esp_now.h“.

Interakce mezi chytrými auty a semaforey



V této lekci budeme zkoumat interakci mezi vozem

SharkBot a inteligentním semaforem v úrovni

Smart Traffic Level 1. Když se vozidlo

SharkBot přiblíží ke křižovatce, inteligentní

semafor mu prostřednictvím bezdrátové komunikace

odešle informace o stavu světelné signalizace. Vozidlo

poté na základě těchto informací rozhodne o svém pohybu. Konkrétní kroky jsou následující:



Nastavení WiFi režimu

Nastavte WiFi režim jak u vozidla SharkBot, tak u inteligentního semaforu na STA režim.

Inicializace ESP-NOW

Na obou zařízeních (SharkBot i inteligentní semafor) inicializujte komunikační protokol ESP-NOW.



Intelligentní semafor vysílá data

Intelligentní semafor vysílá data do vozidla SharkBot.

SharkBot naslouchá datům

Vozidlo SharkBot naslouchá datům z inteligentního semaforu a načítá přijatá data.

Rozhodování o pohybu SharkBotu

Když vozidlo SharkBot dorazí ke křižovatce při sledování čáry, rozhodne o svém pohybu na základě přijatých informací o semaforu. Pokud informace ukazují červenou, vozidlo zastaví; jinak bude pokračovat v jízdě vpřed.

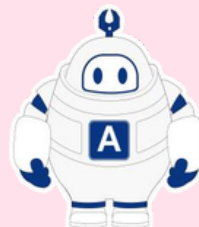




ITS Laboratoř



Interakce mezi SharkBotem a semaforem



Popis projektu

Programování pro realizaci interakce mezi SharkBotem a inteligentním semaforem. Když SharkBot dorazí k semaforu, pokud svítí červená, zastaví se; pokud svítí zelená nebo žlutá, pokračuje v jízdě vpřed.

Poznámka: K experimentu je potřeba použít mapu. Soubor pro tisk mapy v tomto kurzu je součástí informační složky [„English\Arduino\(Experienced Learner\)\5.Map\Map3\(100x80cm\).pdf“](#).



Hardwarové schéma

1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Inteligentní semafor	1
	Type-C Data kabel	1
	Mapa	1



2. Popis mapy

SharkBot se pohybuje po černé čáře na mapě a inteligentní semafor jsou umístěny následovně:



Poznámka: Umístěte auto do oblasti Start pro zahájení provozu.



Programování

1. Kód SharkBotu

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici: [English\Arduino\(Experienced Learner\)\3.Arduinoprogram\15CarInter net\sharkbot](#)

```
#include <Arduino.h>
#include <WiFi.h>
#include <esp_now.h> //include esp-now library
#include <Adafruit_NeoPixel.h> //include RGB strip library

#define RightMotor_A1 12 //define pins of motor driver
#define RightMotor_A2 13
#define LeftMotor_A1 14
#define LeftMotor_A2 21
#define Buzzer_Pin 39 // define pin of
#define LED_PIN 48 //Declare the pin of the LED light strip.
#define LED_COUNT 2 //Declare the number of LED beads on the
LED light strip.
#define BRIGHTNESS 50 //Define the brightness of the LED light
strip.
#define RightLight 0
```





```
#define LeftLight 1           //Declare the serial numbers of the LED beads
                                on the right side.

int LineTrack1 = 4, LineTrack2 = 5, LineTrack3 = 6, LineTrack4 = 7;
int Track1, Track2, Track3, Track4;

int cross = 0; //Mark the intersection.
String actualData; //Receive data.

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN); //Create a light strip
object.

void setup() {
  Serial .begin( 115200);
  strip.begin();                //Initialize the light strip.
  delay (100) ;
  strip.setBrightness(BRIGHTNESS); //Set the brightness of the light
strip.
  delay (100) ;
  pinMode(RightMotor_A1,OUTPUT); //Set the working mode of the input pins
on the driver board.
  pinMode (RightMotor _A2,OUTPUT);
  pinMode (LeftMotor_ A1,OUTPUT );
  pinMode (LeftMotor_ A2,OUTPUT );
  pinMode(LineTrack1, INPUT);
  pinMode(LineTrack2, INPUT);
  pinMode(LineTrack3, INPUT);
  pinMode(LineTrack4, INPUT);
  pinMode(Buzzer_Pin, OUTPUT);
  strip.setPixelColor(RightLight, strip.Color(255, 0, 0)); //
  strip.setPixelColor(LeftLight,  strip.Color(255, 0, 0));
  strip .show ();

  tone(Buzzer_Pin, 262);
  delay (1000 );
  noTone (Buzzer_Pin );
  delay (100) ;
  init_espnow();
  delay (100) ;
}
```





```

void init_espnow()
{
    WiFi.mode(WIFI_STA); // Set to STA mode.
    // Initialize ESP-NOW.
    if (esp_now_init() != ESP_OK) {
        Serial.println("ESP-NOW Initialization Failed");
        return;
    }

    // Register a receive callback.
    esp_now_register_recv_cb([](const uint8_t *mac, const uint8_t *data,
int len) {
        // Display the sender's MAC address.
        char macStr[18];
        snprintf(macStr, sizeof(macStr), "%02X:%02X:%02X:%02X:%02X:%02X",
            mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
        Serial.print("FROM: ");
        Serial. println(macStr
        );
        // Display the message content.
        Serial.print("Content: ");
        actualData = String((const char*)data);
        Serial. print(actualData );
        Serial. println() ;
    });
}

void get_LineTrackValue()//Read the value of the line-following sensor.
{
    Track1 = analogRead(LineTrack1);
    Track2 = analogRead(LineTrack2);
    Track3 = analogRead(LineTrack3);
    Track4 = analogRead(LineTrack4);
}

```





```
// Line-following algorithm.
void line_tracking() {
    get_LineTrackValue ();
    if(Track2 > 1200 && Track3 > 1200){
        motorControl(200, 200);
    }else if(Track2 > 1200 && Track3 < 1200){
        motorControl(-200, 200);
    }else if(Track2 < 1200 && Track3 > 1200){
        motorControl(200, -200);
    }
    else{
        motorControl(200, -200);
    }
}

/*Function to control the motor:
- When PWM > 0, the motor moves forward.
- When PWM < 0, the motor moves backward.
- When PWM = 0, the motor stops.*/
void motorControl(int left_pwm, int right_pwm)
{
    if(left_pwm > 0){
        analogWrite (LeftMotor_ A1, 0) ;
        analogWrite (LeftMotor_ A2,left_pwm );
    }
    else{
        analogWrite (LeftMotor_ A1,-left_pwm);
        analogWrite (LeftMotor_ A2, 0) ;
    }
    if(right_pwm > 0){
        analogWrite (RightMotor _A1,0 );
        analogWrite (RightMotor _A2,right_pwm) ;
    }
    else{
        analogWrite (RightMotor _A1,-right_pwm );
        analogWrite (RightMotor _A2,0 );
    }
    delay( 5);
}
```





```
void loop() {
    get_LineTrackValue(); //Get the value of the line-following sensor.
    while(!(Track1 > 1200 && Track2 > 1200 && Track3 > 1200 && Track4 >
    { 1200})) //Keep following the line until reaching the intersection.
        line_track ing() ;
    }
    cross++;
    if(cross == 1){
        strip.setPixelColor(RightLight, strip.Color(0, 255, 0)); //Green
light.
        strip.setPixelColor(LeftLight, strip.Color(0, 255, 0)); //Green
light.
        strip .show();
        while(actualData == "Red LED1"){ // Determine whether it is a red
light. If it is a red light, wait until the red light ends.
            motorControl(0, 0); // Stop.
        }
        actualData = "";
    }
    if(cross == 2){
        cross = 0;
    }
    motorControl(100, 100);
    delay (300) ;
}
```

2. Kód inteligentního semaforu

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici: English\Arduino(Experienced Learner)\3.Arduino program\15CarInternet\SmartTrafficLight

[English\Arduino\(Experienced Learner\)\3.Arduino program\15CarInternet\SmartTrafficLight](#)

```
#include <WiFi.h>
#include <Wire.h> //Import the relevant libraries of I2C
#include <TM1650.h> //Import the library of 4-digit digital tubes
#include <esp_now.h>
#define R 14 // Define the pins of the traffic light module #define
Y 2
#define G 0
```





```
// Broadcast address
uint8_t broadcastMac[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

//The content broadcast
String dataToSend = "Green LED1";
String dataToSend1 = "Red LED1";
String dataToSend2 = "Yellow LED1";

char dis[4]; // Create an array to store the text displayed by the digital
tube
TM1650 d;    //Create a digital tube object
int state;

void setup() {
  Serial .begin( 115200);
  Wire.begin(); // Enable I2C communication
  d.init(); //Initialize the digital tube
  d.setBrightness(TM1650_MIN_BRIGHT); // Set the brightness of the
digital tube
  d.displayOn(); //Enable the display function of the digital tube
  pinMode(R, OUTPUT); //Set the working mode of the traffic light pins
  pinMode(Y, OUTPUT);
  pinMode(G, OUTPUT);
  init_espnw();//Initialize the ESP-NOW communication protocol
  Traffic_Light(1, 1, 1);//After the pairing is successful, the
light will be on for 2 seconds.
  delay (2000 );
}

void loop() {

  Traffic_Light(0, 0, 1); //The green light is on.
  state = 1;
  displayTime(5);          // Five-second countdown.

  Traffic_Light(0, 1, 0); //The yellow light is on.
  state = 2;
  displayTime(2);          //Two-second countdown.
```





```

Traffic_Light(1, 0, 0); //The red light is on.
state = 3;
displayTime(6);          // 6-second countdown.
state = 0;
}
void init_espnw()
{
    WiFi.mode(WIFI_STA); // It must be set to STA mode
    // Initialize ESP-NOW
    if (esp_now_init() != ESP_OK) {
        Serial.println("The initialization of ESP-NOW failed");
        ESP.restart();
    }

    // Configure the broadcast peer device
    esp_now_peer_info_t peerInfo;
    memset(&peerInfo, 0, sizeof(peerInfo)); // Empty the structure
    memcpy(peerInfo.peer_addr, broadcastMac, 6);
    peerInfo.channel = 0;          // Use the current channel
    peerInfo.encrypt = false; // Unencrypted

    // Add broadcast peer devices
    if (esp_now_add_peer(&peerInfo) != ESP_OK) {
        Serial.println("Failed to add the broadcasting equipment");
        ESP.restart();
    }

    // Register to send a callback
    esp_now_register_send_cb([](const uint8_t *mac,
    esp_now_send_status_t status) {
        Serial.print("[Send status] ");
        Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success" :
"Failure" );
    });
}

```





```
// Traffic light control function.
void Traffic_Light(int r, int y, int g) {
    digitalWrite(R, r);
    digitalWrite(Y, y);
    digitalWrite(G, g);
}
// Countdown display function
void displayTime(int seconds) {
    //Execute the loop until the time is less than 1.
    while (seconds >= 1) {
        if (state == 1){
            // Broadcast green light message.
            esp_now_send(broadcastMac, (uint8_t*)dataToSend.c_str(),
dataToSend. length()) ;
        } else if (state == 2){
            // Broadcast the yellow light message.
            esp_now_send(broadcastMac, (uint8_t*)dataToSend2.c_str(),
dataToSend2 .length()) ;
        } else if (state == 3){
            // Broadcast the red light message.
            esp_now_send(broadcastMac, (uint8_t*)dataToSend1.c_str(),
dataToSend2 .length()) ;
        }
        itoa(seconds, dis, 10);
        d.displayStr ing (dis);
        delay (1000);
        seconds--;
        d.clear ();
    }
}
```





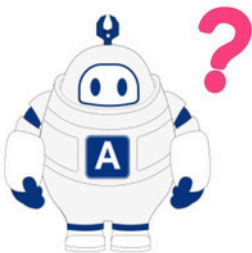
Oddíl 5: Integrovaný systém inteligentní dopravy



Cíle kurzu



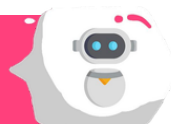
-  Princip integrovaného systému inteligentní dopravy
-  Programové řešení pro realizaci síťové komunikace mezi chytrými auty a inteligentními semaforey, inteligentními přechody a inteligentními parkovišti.



Can smart cars also communicate with other transportation facilities?

V předchozí lekci jsme dokončili síťovou komunikaci mezi SharkBotem a inteligentními semaforey v úrovni 1 inteligentní dopravy. Kromě semaforů může SharkBot také interagovat a komunikovat s inteligentními přechody a inteligentními parkovišti, čímž vzniká kompletní integrovaný systém inteligentní dopravy. V návaznosti na předchozí lekci nyní přidáme interaktivní funkce inteligentního přechodu a inteligentního parkoviště, abychom dokončili návrh integrovaného systému inteligentní dopravy.

Princip fungování systému ITS





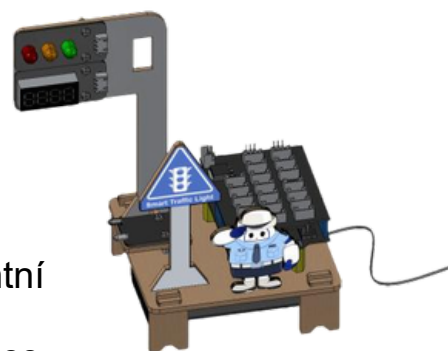
Na rozdíl od předchozí lekce musí SharkBot v integrovaném systému inteligentní dopravy komunikovat se třemi dalšími zařízeními. Aby mohl SharkBot rozpoznat, ze kterého zařízení data pocházejí, je nutné zajistit obousměrnou komunikaci mezi SharkBotem a každým zařízením. Konkrétní komunikační procesy jsou následující:

1. Interakce mezi SharkBotem a inteligentním semaforem

(1) Umístěte SharkBot do oblasti Start, aby začal úlohu sledování čáry, a odešlete signál inteligentnímu semaforu, že se blíží ke křižovatce.

(2) Po přijetí signálu od SharkBotu odešle inteligentní semafor zprávu o stavu červené světelné signalizace zpět SharkBotu.

(3) Když SharkBot dorazí ke křižovatce semaforu, na základě přijatých informací o světlech zvolí další akci. Pokud svítí červená, zastaví se; jinak pokračuje v jízdě vpřed.



2. Interakce mezi SharkBotem a inteligentním přechodem (Smart Crossing)

(1) Po projetí křižovatky inteligentního semaforu odešle SharkBot zprávu inteligentnímu přechodu, že se blíží do jeho oblasti.

(2) Po přijetí zprávy od SharkBotu odešle inteligentní přechod stav semaforu pro chodce do SharkBotu.



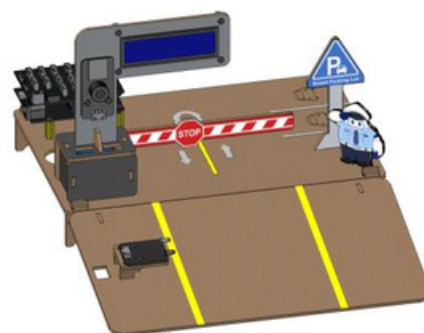


(3) Když SharkBot dorazí ke křižovatce inteligentního přechodu, na základě přijatých informací o světelné signalizaci zvolí další akci. Pokud svítí červená, zastaví se; jinak pokračuje v jízdě vpřed.

3. Interakce mezi SharkBotem a inteligentním parkovištěm

(1) Po projetí inteligentním přechodem SharkBot pokračuje po čáře ke křižovatce inteligentního parkoviště a odbočí doprava směrem na parkoviště.

(2) Když infračervený senzor inteligentního parkoviště detekuje SharkBot, odešle mu zprávu, aby zastavil a čekal na přiložení karty.



(3) Po přijetí zprávy o parkování SharkBot zastaví a čeká na další pokyn pro zaparkování.

(4) Po přiložení karty uživatelem se zvedne závora parkoviště a odešle se zpráva SharkBotu, aby vjel na parkoviště. Po vjezdu SharkBotu se závora opět spustí.

(5) Když uživatel znovu přiloží kartu, parkoviště zvedne závoru a odešle zprávu SharkBotu, aby vyjel z parkoviště. Po výjezdu SharkBotu se závora opět spustí.

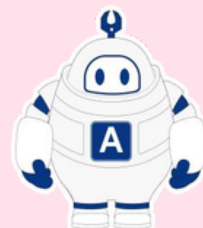




ITS Laboratoř



Integrovaný systém inteligentní dopravy



Popis projektu

Programování pro realizaci komplexní interakce mezi SharkBotem, inteligentním semaforem, inteligentním přechodem a inteligentním parkovištěm a dokončení návrhu systému inteligentní dopravy.



Hardwarové schéma

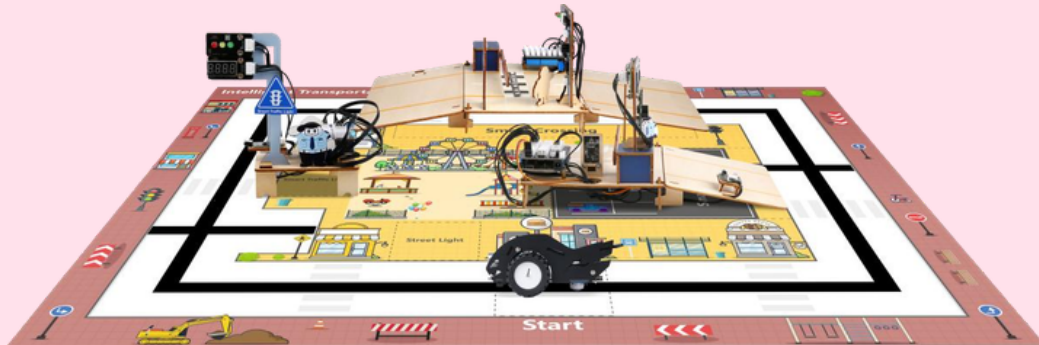
1. Hardware List

Picture	Name	Quantity
	SharkBot	1
	Type-C Data kabel	1
	Inteligentní semafor	1
	Inteligentní parkoviště	1
	Inteligentní přechod	1
	Mapa	1



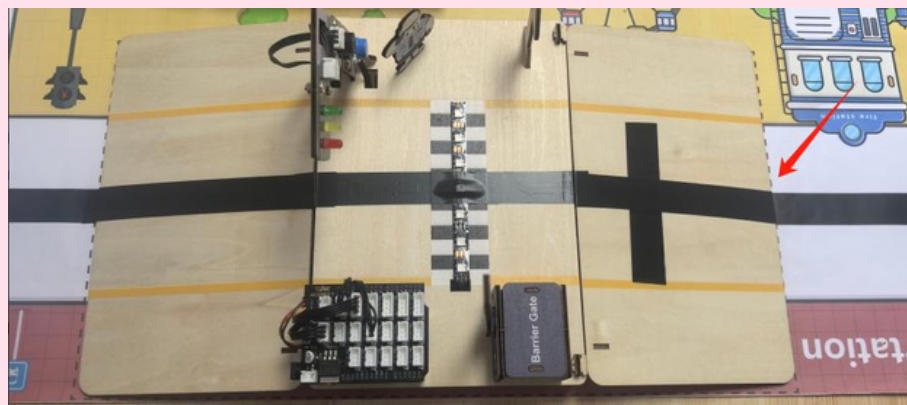
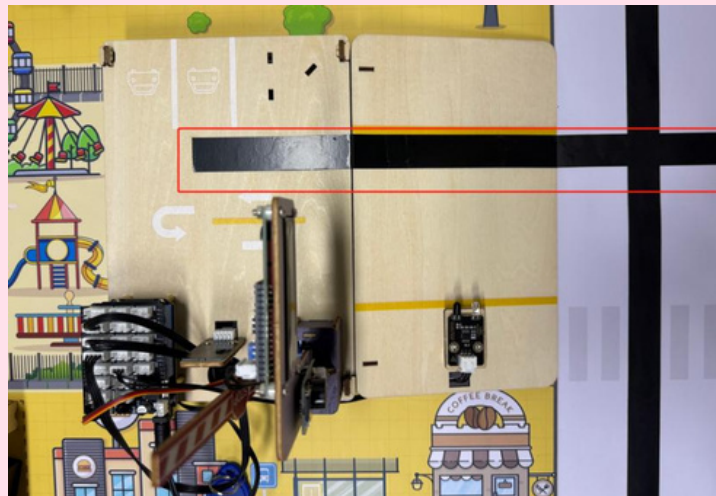
2. Popis mapy

Vozidlo SharkBot se pohybuje po černé čáře na mapě. Pozice inteligentního semaforu, inteligentního přechodu a inteligentního parkoviště jsou znázorněny níže:



Poznámka:

1. Po rozmístění parkoviště a přechodu nalepte černou pásku podle tohoto schématu.





2. Při lepení pásky ponechte určitou mezeru v místě spojení mezi mapou a parkovištěm/chodníkem, což může vozidlu účinně pomoci při vyjíždění po rampě.



Programování

Zdrojový kód pro tuto lekci lze najít a otevřít ve složce elektronických materiálů přiložené k této učebnici:

Engl is h\A rd uin o([Exp er ien cedLearner](#))\3.Arduino program\16Comprehe nsiveSystem

Poznámka: Pokud projekt nefunguje správně, nejpravděpodobnější příčinou je problém se sledováním čáry. Můžete sami zkontrolovat část kódu pro sledování čáry a upravit prahovou hodnotu sledování čáry.

```
int LineTrack1 = 4, LineTrack2 = 5, LineTrack3 = 6, LineTrack4 = 7;  
int Track1, Track2, Track3, Track4;  
int threshold = 1200;
```

Sledujte nás

Naskenujte QR kódy a sledujte nás pro řešení problémů a nejnovější novinky.

Máme velmi rozsáhlou komunitu, která je velmi nápomocná při řešení problémů, a také tým podpory připravený odpovědět na jakékoli dotazy.



ACEBOTT FB QR Code



YouTube QR Code



ACEBOTT STEM Education Tech Co.,Ltd

0755-23956813

www.cebott.com

Room 306, Building 212, Tairan Science Park,Terra 4th Rd,Futian District, ShenZhen, China, 518042