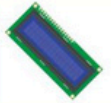




Tutoriál pro ESP32 Sada prozačátečníky

Programování Arduina

Seznam sad

 ACEBOTT ESP32 Max V1.0 1PC	 Ultrasonic Sensor 1PC	 1602 LCD Module 1PC	 PIR Motion 1PC	 Servo SG90 9G 1PC
 DHT11 1PC	 IR Remote Controller 1PC	 400 Tie-points Breadboard 2PCS	 Passive Buzzer 1PC	 Active Buzzer 1PC
 7-Segment Display 1PC	 4 Digit Digital Tube 1PC	 Potentiometer 1PC	 DC Motor Driver 1PC	 M-M BreadBoard Jumper Wire 1PC
 Fan 1PC	 DC Motor 1PC	 USB Cable 1M 1PC	 Button 7PCS	 6xAA Battery Holder 1PC
 F-M Dupont Wire 20PCS	 Red LED 5PCS	 Blue LED 5PCS	 Green LED 5PCS	 RGB LED 1PC
 S8050 Transistor 1PC	 Photoresistor 1PC	 IR Receiver 1PC	 Resistor (1k/10k/220Ω) 30PCS	

Obsah

Kapitola 1 Příprava.....	1
Úvod k řídicí desce ESP32.....	2
Nastavení programovacího prostředí	5
Kapitola 2 Modul Učení.....	31
Lekce 1 LED dioda.....	32
Lekce 2 Dýchání světla	43
Lekce 3 RGB LED.....	53
Tlačítko Lekce 4.....	59
Lekce 5 Infračervený senzor pro člověka.....	65
Lekce 6 Fotorezistor	71
Lekce 7 Potenciometr.....	78
Lekce 8 Aktivní bzučák	84
Lekce 9 Pasivní bzučák	91
Lekce 10 Stejnoseměrný motor	100
Lekce 11 Servo.....	108
Lekce 12 Snímač teploty a vlhkosti.....	113
Lekce 13 Ultrazvukový senzor	120
Lekce 14 8segmentový displej.....	128
Lekce 15 Čtyřmístná digitální elektronka.....	134
Lekce 16 Infračervené dálkové ovládání	141
Lekce 17 Displej LCD1602.....	148
Kapitola 3 Komplexní projekty.....	156
Lekce 18 Plynoucí světlo	157
Lekce 19 Podsvícení tlačítka	162
Lekce 20 Lidský senzor Světlo	170
Lekce 21 Světlo řízené světlem	175
Lekce 22 Úprava barev světla	180
Lekce 23 Elektronické kostky.....	185
Lekce 24 Počítadlo	192
Lekce 25 Bzučák kvízu	200
Lekce 26 Ovocné piano.....	207
Lekce 27 Ventilátor s proměnnou rychlostí.....	215

Lekce 28 Ventilátor s regulací teploty.....	221
Lekce 29 Ventilátor s infračerveným dálkovým ovládáním.....	226
Lekce 30 Monitorování environmentálních dat.....	232
Lekce 31 Systém radarového poplašného systému.....	238
Lekce 32 Světlo Bluetooth	244
Lekce 33 Ovládání webu	252
 Kapitola 4 Projekty rozšíření.....	266
Lekce 34 Chytrý bionický motýl.....	267
Lekce 35 Chytrý odpadkový koš.....	291
Lekce 36 Chytrá brána.....	303
Lekce 37 Oscilační ventilátor.....	319
Lekce 38 Elektronické piano.....	347

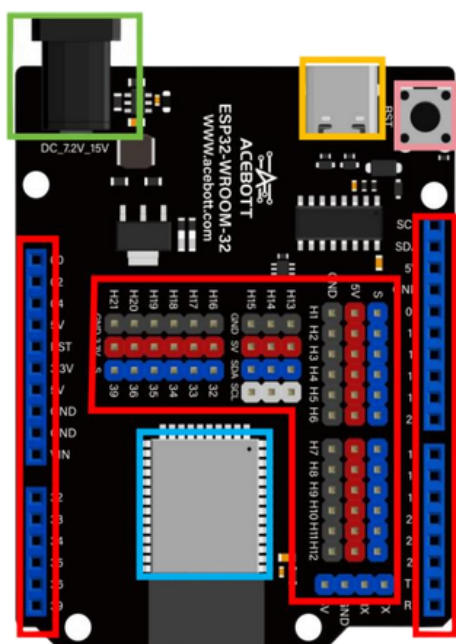
Kapitola 1 Příprava

V této kapitole získáme přehled základních funkcí řídicí desky ESP32, dokončíme nastavení vývojového prostředí, nainstalujeme ovladač sériového portu a provedeme počáteční konfiguraci desky. Také se naučíme, jak úspěšně nahrát program na řídicí desku ESP32, a položit tak pevný základ pro následnou práci v kurzu a implementaci projektů.

Úvod do řídicí desky ESP32

ESP32 je nízkopříkonový, vysoce integrovaný duální mikrokontrolér s podporou Wi-Fi a Bluetooth, vyvinutý společností Espressif Systems. Řadič ESP32 je velmi oblíbený díky svým výkonným funkcím a bohatým komunikačním rozhraním, díky čemuž je vhodný pro řadu aplikací internetu věcí (IoT), vývoj vestavěných systémů a implementaci projektů.

ACEBOTT—Blokové schéma desky řídicí jednotky ESP32:



Příkon: Lze připojit k externímu napájecímu zdroji 7,2 V - 15 V pro napájení vývojové desky.

Port typu C: Používá se pro stahování programů a může také poskytovat 5V napájení vývojové desce.

Resetovací klíč: Používá se k restartu čipu ESP32 pro opětovné spuštění programu.

GPIO: Univerzální vstupně/výstupní rozhraní používané pro vstup a výstup dat.

Modul ESP32: obsahuje ESP32 mikroprocesor, Bluetooth a Wi-Fi.

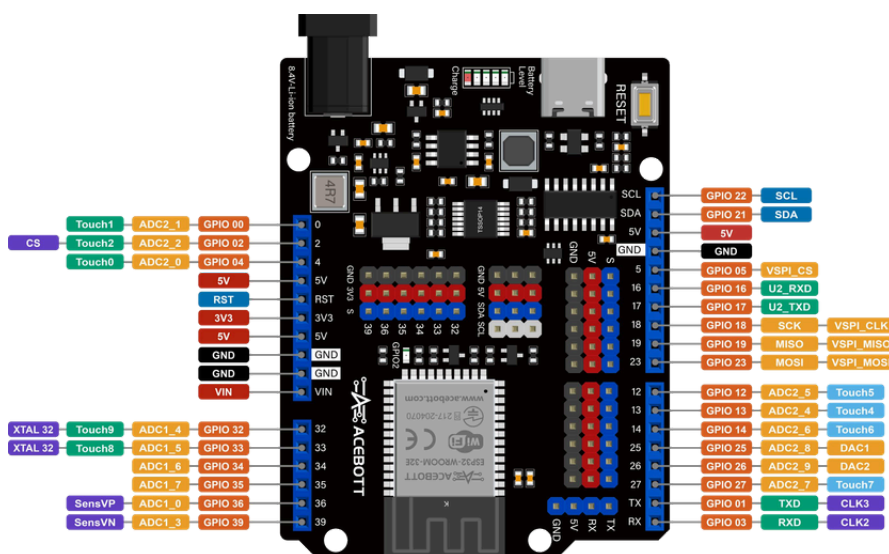
ACEBOTT—Vlastnosti desky řídicí jednotky ESP32

Vysoký výkon: Čip ESP32 je vybaven dvoujádrovým procesorem s taktovací frekvencí až 240 MHz, který nabízí výkonné výpočetní a procesorové schopnosti.

Funkce Wi-Fi a Bluetooth: Může se připojit k internetu a komunikovat s dalšími zařízeními prostřednictvím Wi-Fi. Funkce Bluetooth umožňuje bezdrátové připojení k dalším zařízením s technologií Bluetooth.

Paměť a úložiště: Deska má 4 MB integrované flash paměti pro ukládání programů a dat. Má také velké množství SRAM pro spouštění aplikací.

Zapojení pinů desky řídicí jednotky ACEBOTT—ESP32:



Rozložení pinů ADC

ADC převodník ESP32 je 12bitový analogově-digitální převodník, což znamená, že převádí vstupní napětí mezi 0 a 3,3 V na celočíselnou hodnotu mezi 0 a 4095 (2^{12}). Jeho rozlišení je tedy přibližně $3,3 \text{ V} \div 4096 \approx 0,0008 \text{ V}$, což znamená, že každá jednotka představuje změnu napětí přibližně 0,8 milivoltů. ESP32 má dva ADC moduly (ADC1 a ADC2), z nichž každý spravuje více analogových kanálů. (Na obrázku jsou piny označené ADC1 spravovány modulem ADC1 a piny označené ADC2 jsou spravovány modulem ADC2.)

Poznámka: Je-li povoleno Wi-Fi, modul ADC2 s ním sdílí hardwarové prostředky, čímž se piny spravované ADC2 stanou nepoužitelnými. Čtení se nezdaří nebo vrátí nesprávné hodnoty. Je-li povoleno Wi-Fi, je

pro sběr analogového signálu se doporučuje používat pouze piny spravované ADC1.

Kapacitní dotykové snímací piny

ESP32 má devět kapacitních dotykových GPIO (piny označené v diagramu jako Touch). Když je kapacitní zátěž (například lidský prst) umístěna v blízkosti GPIO, ESP32 detekuje změnu kapacity. Dotykové plošky lze vytvořit připojením jakéhokoli vodivého předmětu k těmto pinům, například hliníkové fólie, vodivé látky nebo vodivé barvy.

I2C pin

Řídicí deska ESP32 má dva nezávislé řadiče sběrnice I2C (I2C0 a I2C1). Každá sběrnice může paralelně připojit až přibližně 112 unikátně adresovaných senzorů a periférií.

SPI piny

Řadič ESP32 má tři rozhraní SPI (SPI, HSPI a VSPI).

Poznámka: Aktivní SPI rozhraní jsou pouze VSPI a HSPI; třetí SPI sběrnici používá integrovaná flash paměť.

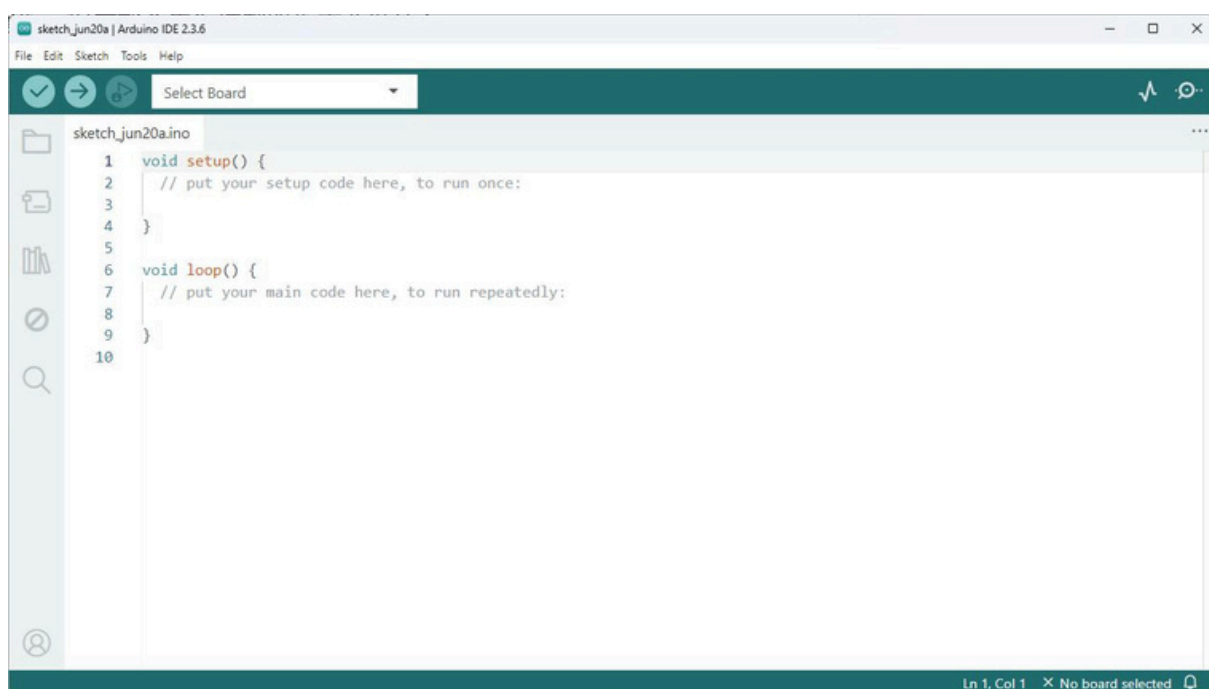
Piny sériového portu

Řídicí deska ESP32 má tři rozhraní UART: UART0, UART1 a UART2. UART0 se připojuje k převodníku USB-sériové rozhraní pro přenos a ladění programu. UART1 je vyhrazen pro integrovaný čip flash paměti. Při připojování k zařízení UART použijte UART2 (GPIO16 a GPIO17).

Nastavení programovacího prostředí

V projektu ESP32 budeme jako programovací software primárně používat Arduino IDE. Jedná se o programovací nástroj s otevřeným zdrojovým kódem, který je kompatibilní s různými řídicími deskami, jako jsou Arduino, ESP32, ESP8266, STM32 a další.

Pomocí Arduino IDE jednoduše napíšete programový kód do IDE a nahrajete ho do řídicí desky, načež program řídicí desce dá instrukce, co má dělat.



I. Nainstalujte si vývojové prostředí Arduino.

Nejprve si otevřete oficiální webové stránky a stáhněte si Arduino IDE: <https://www.arduino.cc/en/Main/Software>

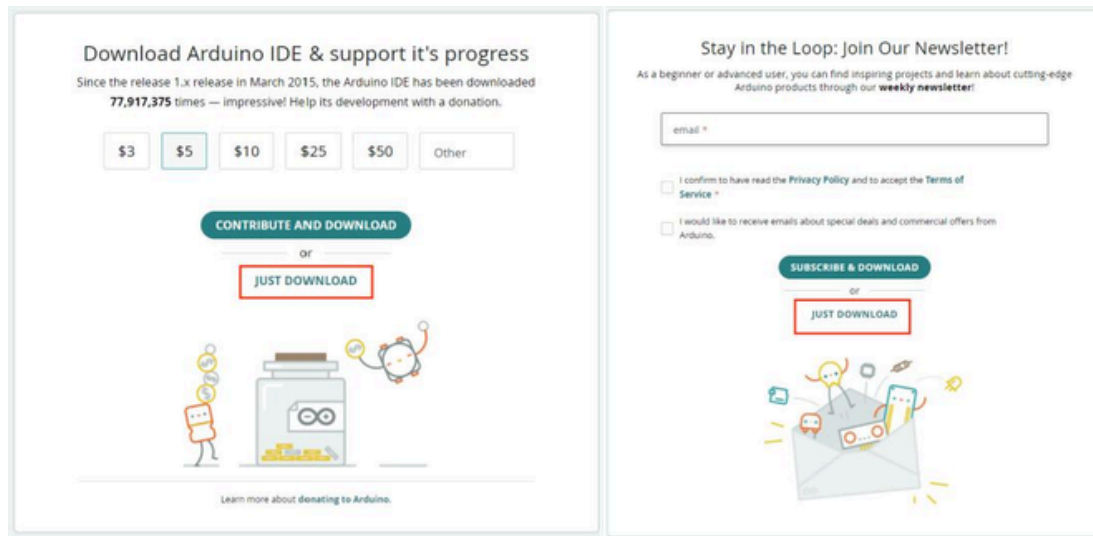
1 Instalace na systém Windows

- ① Klikněte na „Windows Win 10 a novější, 64 bitů“.



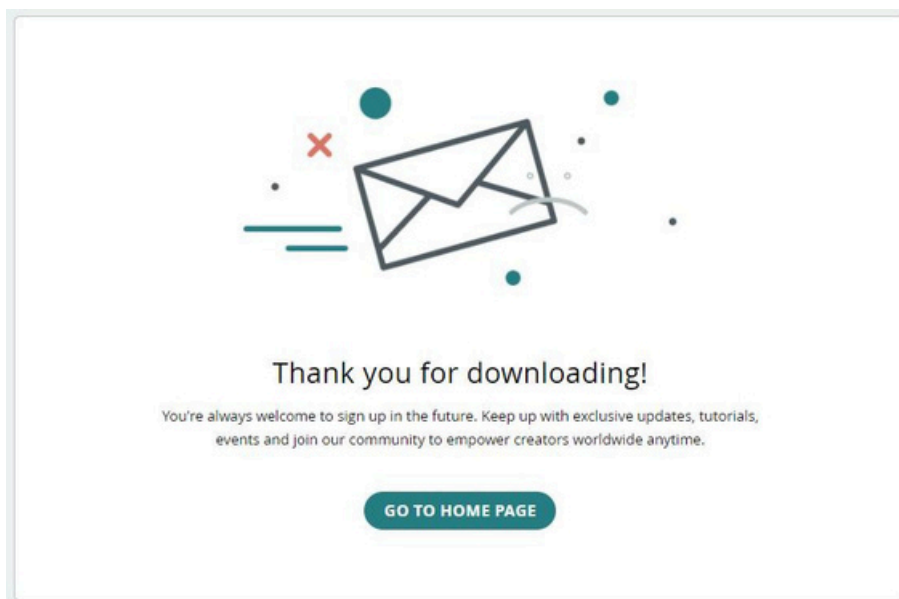
The screenshot shows the Arduino IDE 2.3.6 download page. On the left, there's a section with the Arduino logo and the title "Arduino IDE 2.3.6". Below the title, it describes the new major release as faster and more powerful, featuring a modern editor, autocompletion, code navigation, and a live debugger. It also mentions that for more details, users should refer to the "Arduino IDE 2.0 documentation". A section for "SOURCE CODE" states that the IDE is open source and hosted on GitHub. On the right, under "DOWNLOAD OPTIONS", there are links for Windows (Win 10 and newer, 64 bits), Linux (AppImage 64 bits and ZIP file 64 bits), and macOS (Intel and Apple Silicon). The "Windows Win 10 and newer, 64 bits" link is highlighted with a red box.

2 Vyberte JEN STÁHNOUT.

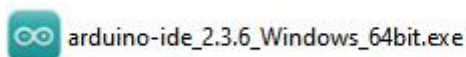


The screenshot shows the Arduino IDE download and donation page. On the left, there's a section titled "Download Arduino IDE & support it's progress" with a donation form. The form has buttons for \$3, \$5, \$10, \$25, \$50, and "Other". Below the form, there's a "CONTRIBUTE AND DOWNLOAD" button and a "JUST DOWNLOAD" button. On the right, there's a section titled "Stay in the Loop: Join Our Newsletter!" with a form to enter an email address. Below the form, there are checkboxes for "I confirm to have read the Privacy Policy and to accept the Terms of Service" and "I would like to receive emails about special deals and commercial offers from Arduino". Below the checkboxes, there's a "SUBSCRIBE & DOWNLOAD" button and a "JUST DOWNLOAD" button. The "JUST DOWNLOAD" button is highlighted with a red box.

- ③ Když se objeví následující obrazovka, znamená to, že je nainstalováno Arduino IDE.



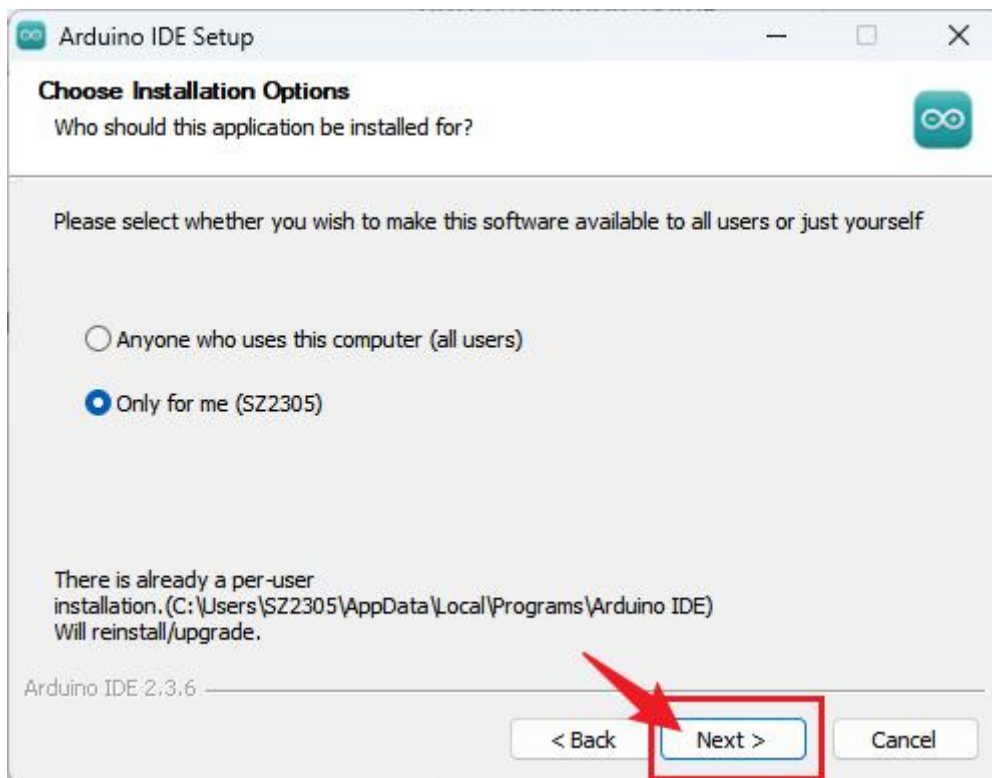
4 Po dokončení stahování se zobrazí soubor s ikonou. Klikněte na tlačítko Instalovat software.



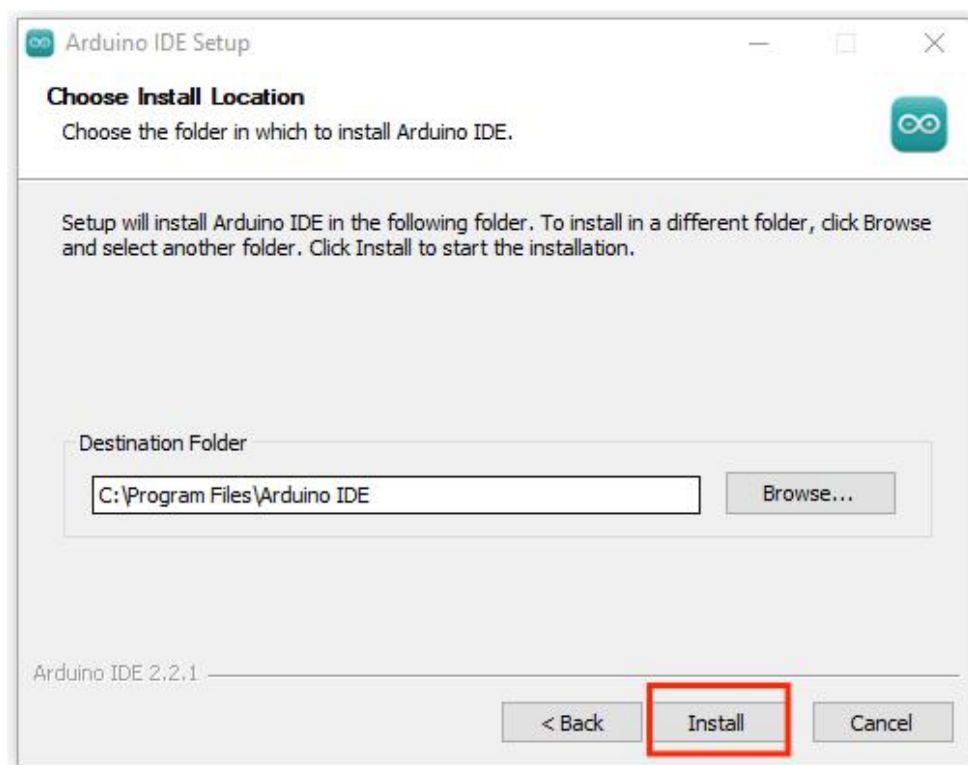
⑤ Po instalaci se zobrazí následující obrazovka, kde vyberte možnost „Souhlasím“.



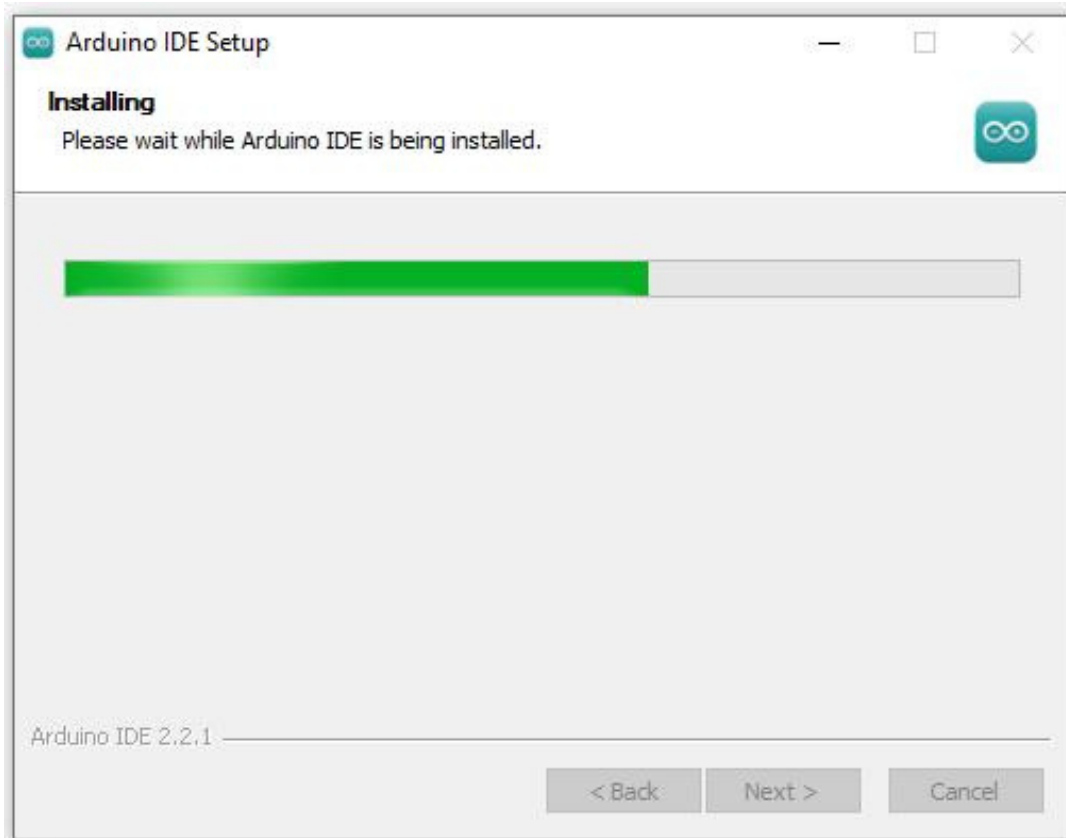
⑥ Po výběru možnosti „Souhlasím“ se zobrazí následující obrazovka, kde vyberte "Další".



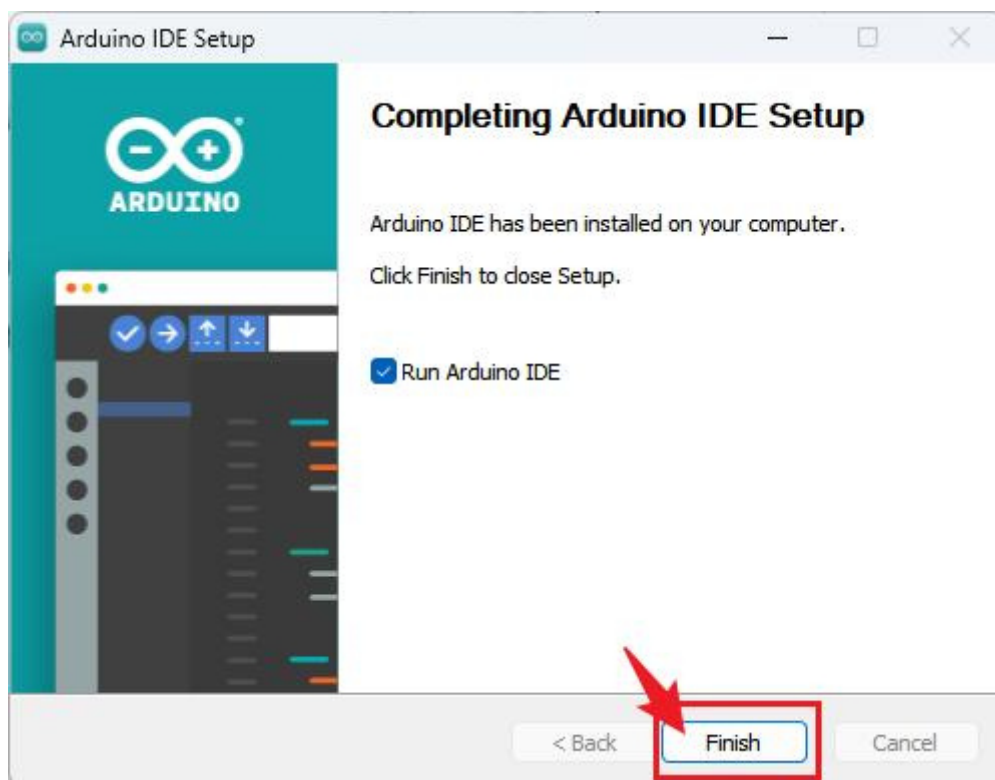
⑦ Vyberte „Další“ a zobrazí se následující obrazovka. Vyberte „Instalovat“.



⑧ Nainstalovaný software pro Arduino IDE.



- ⑨ Po dokončení instalace klikněte na tlačítko „Dokončit“.



- ⑩ Po instalaci se na vašem počítači zobrazí ikona zástupce Arduino IDE.



2 Instalace v systému Mac OS

① Protože počítače Mac používají různé architektury čipů, je třeba Při instalaci Arduino IDE vyberte odpovídající verzi (ověřte si, který čip váš Mac používá, v sekci „O tomto Macu“):

Pokud má váš Mac čip Intel, klikněte na „Intel, 10.14: 'Mojave' nebo novější, 64 bitů“ pro stažení.

Pokud váš Mac má čip Apple Silicon (například M1/M2/M3), klikněte na „macOS Apple Silicon, 11: 'Big Sur' nebo novější, 64 bitů“ pro stažení.

A screenshot of the Arduino IDE 2.3.6 download page. The page has a light blue header with the Arduino logo and the text "Arduino IDE 2.3.6". Below the header, there is a description of the new release, a link to the documentation, and a section for source code. On the right side, there is a dark teal sidebar titled "DOWNLOAD OPTIONS" which lists download links for Windows, Linux, and macOS. The macOS options are highlighted with a red box.

Arduino IDE 2.3.6

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

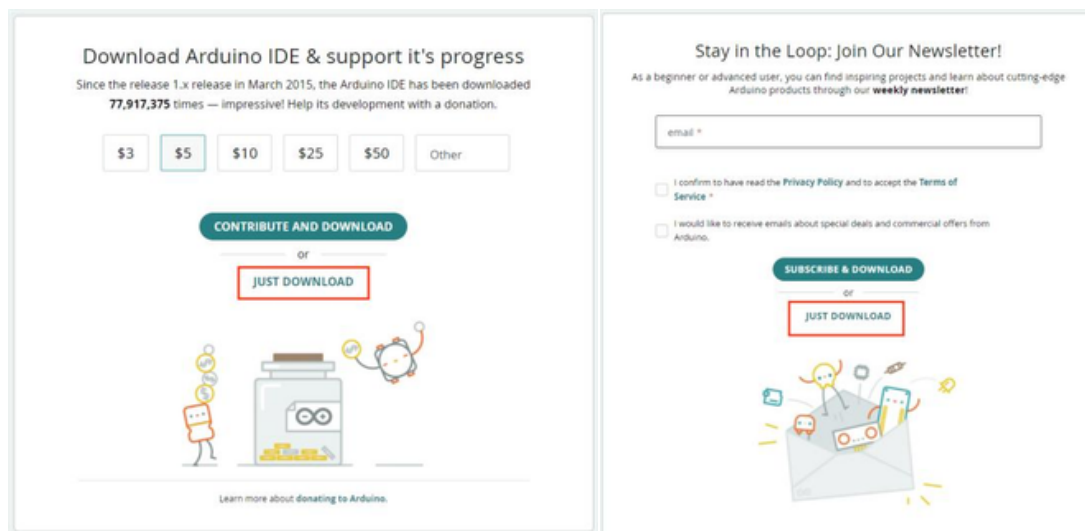
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits

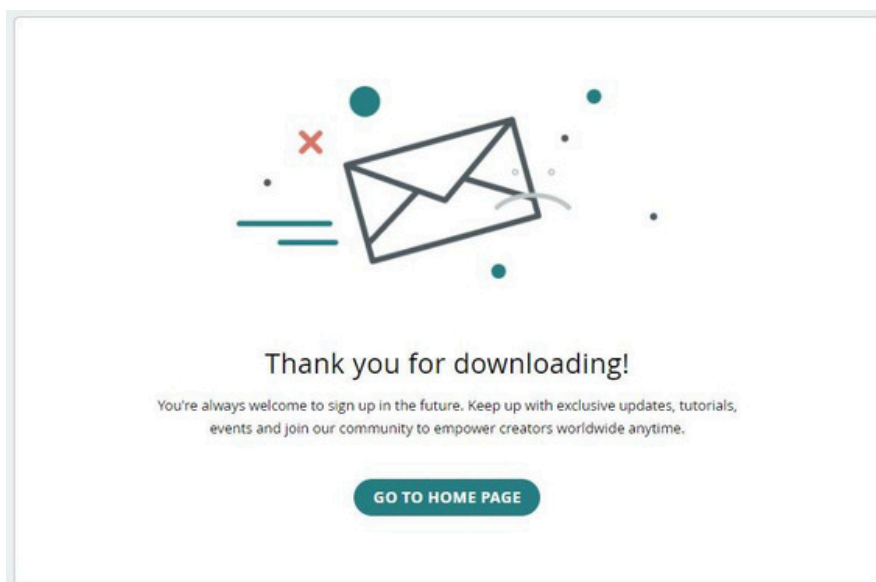
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

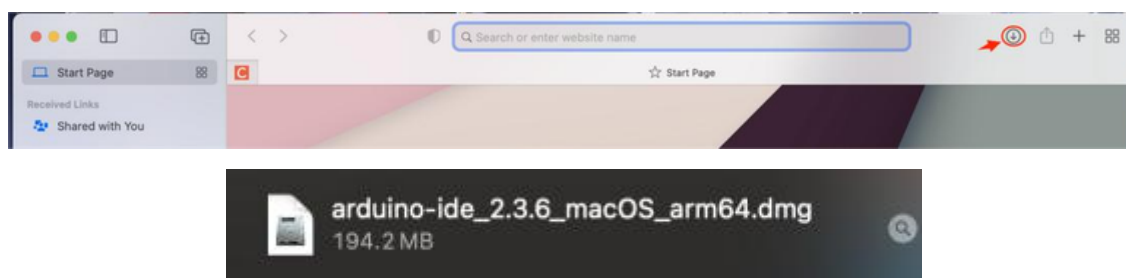
2 Vyberte JEN STÁHNOUT.



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



4 Po dokončení stahování klikněte na ikonu stahování ve vašem prohlížeč a najděte instalační program Arduino IDE.



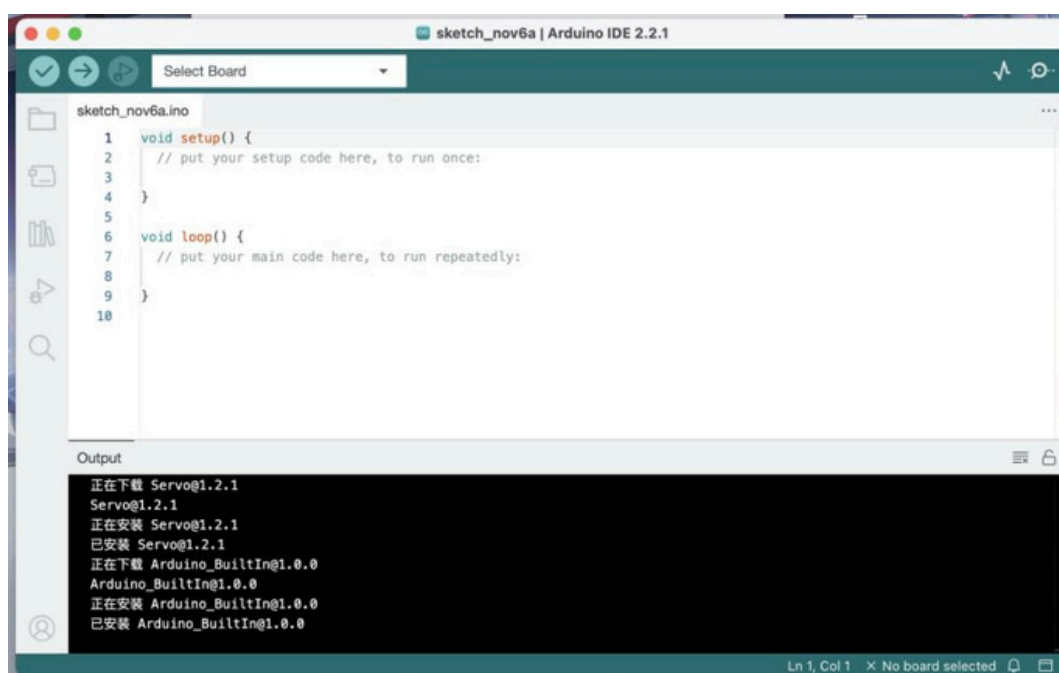
⑤ Kliknutím na instalační balíček se zobrazí instalační obrazovka, stačí vybrat. Klikněte na ikonu Arduino IDE a přejděte do Aplikací, kde program nainstalujete.



⑥ V pracovní desce najdete Arduino IDE a otevřete ho.



⑦ Po otevření se vám zobrazí následující softwarové rozhraní.



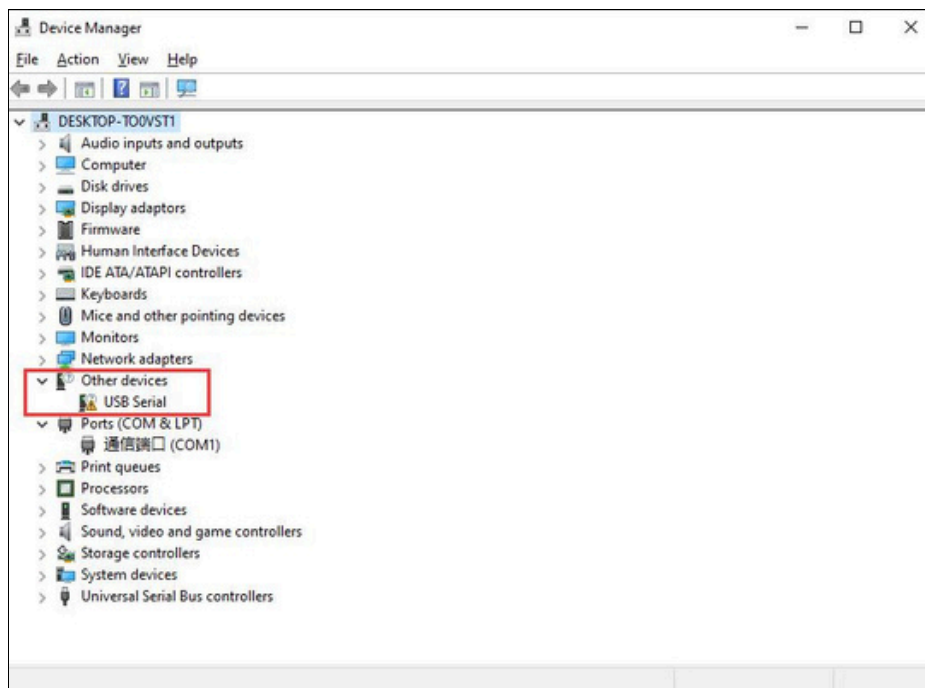
II. Instalace ovladače sériového portu

Připojte řídicí desku ESP32 k počítači pomocí kabelu USB. V systémech macOS a Windows se ovladače nainstalují automaticky. Pokud se instalace ovladače nezdaří, bude pravděpodobně nutné je nainstalovat ručně.

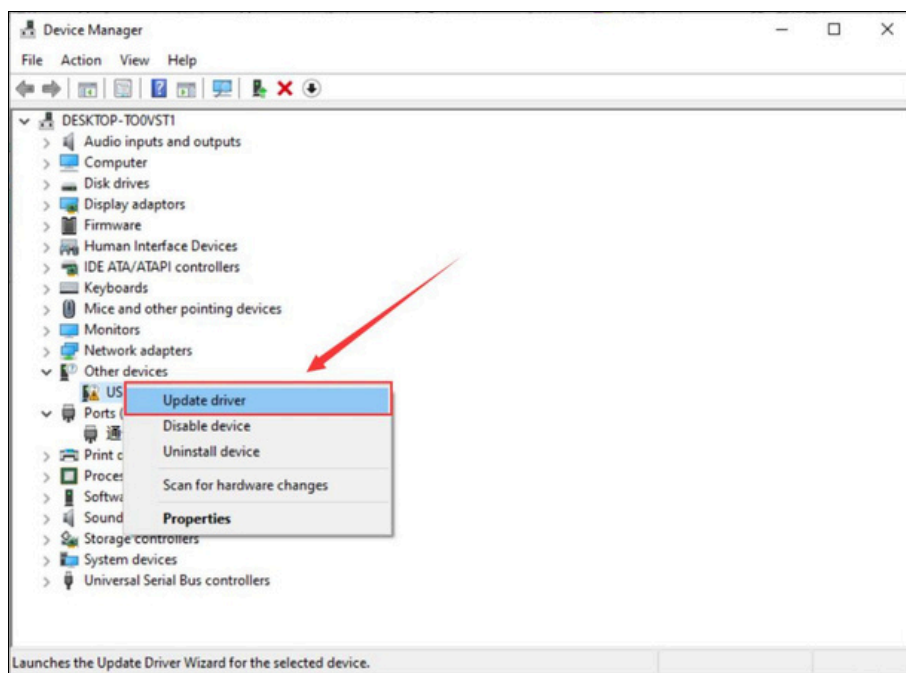
1. Režim instalace ovladače systému Windows

① Po připojení desky ESP32 k počítači přes USB klikněte pravým tlačítkem myši

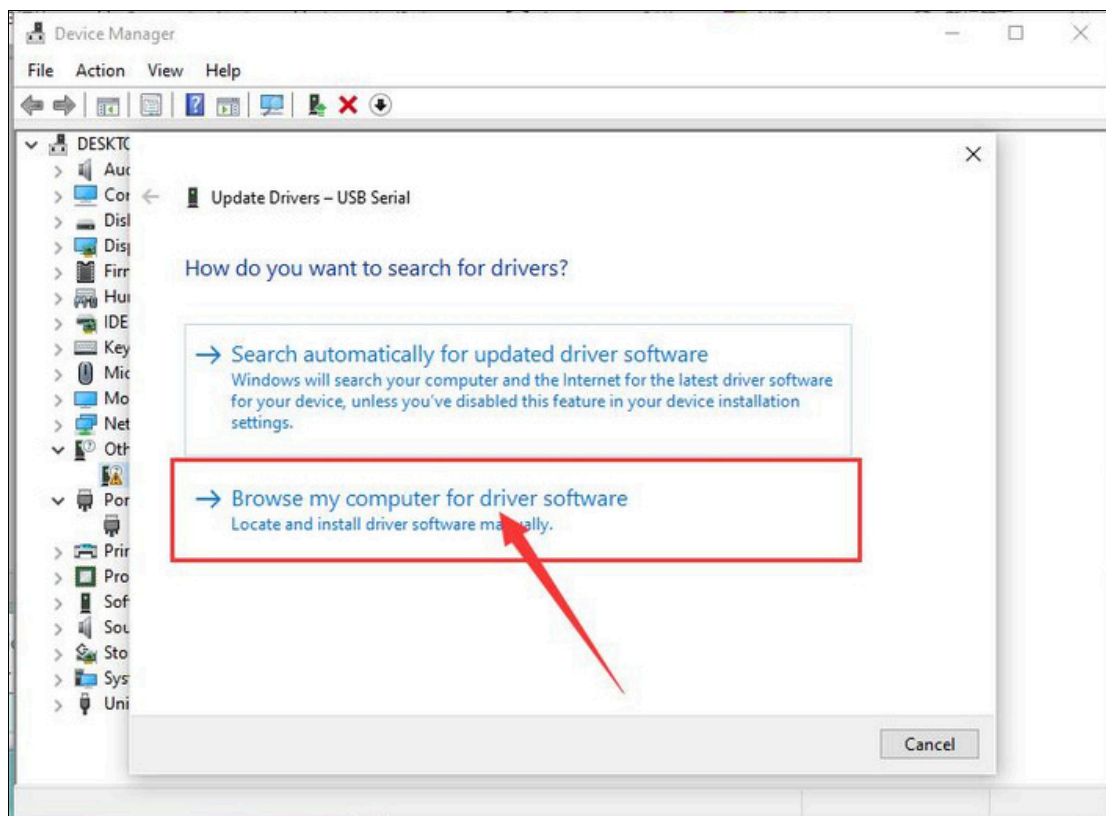
V nabídce „Tento počítač“ -> „Vlastnosti“ klikněte na „Správce zařízení“. V části „Další zařízení“ uvidíte „USB-Serial“ nebo „Neznámé zařízení“.



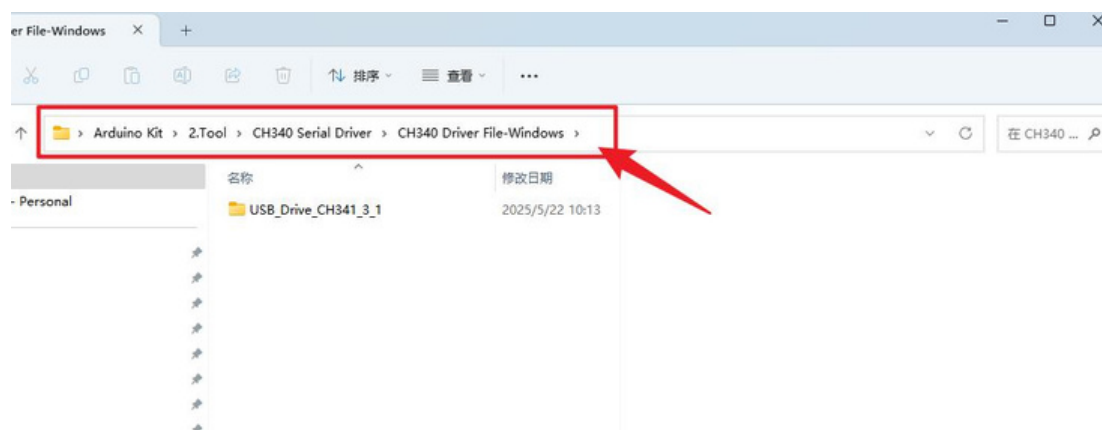
2 Klikněte pravým tlačítkem myši na zařízení a v horní nabídce vyberte možnost „Aktualizovat ovladač Software“, jak je znázorněno na obrázku níže.



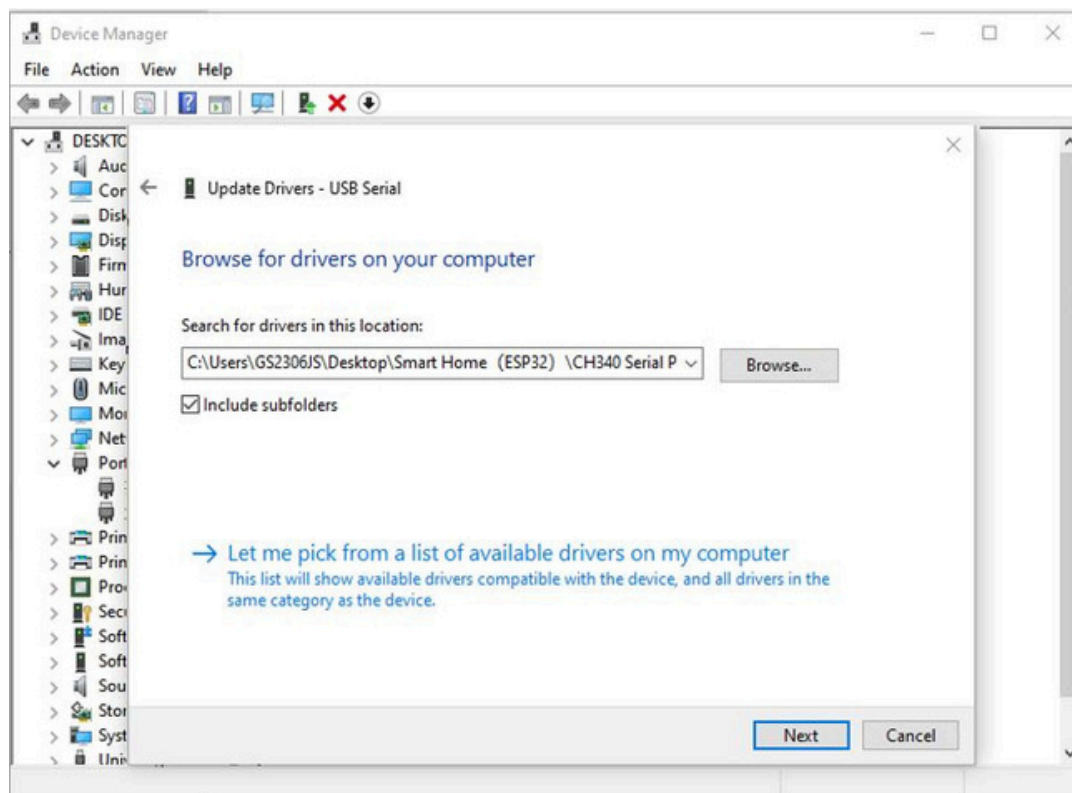
③ Poté se zobrazí výzva „Automaticky vyhledat aktualizovaný ovladač software“ nebo „Vyhledat v počítači software ovladače“, jak je znázorněno na obrázku níže. Na této stránce vyberte možnost „Vyhledat v počítači software ovladače“.



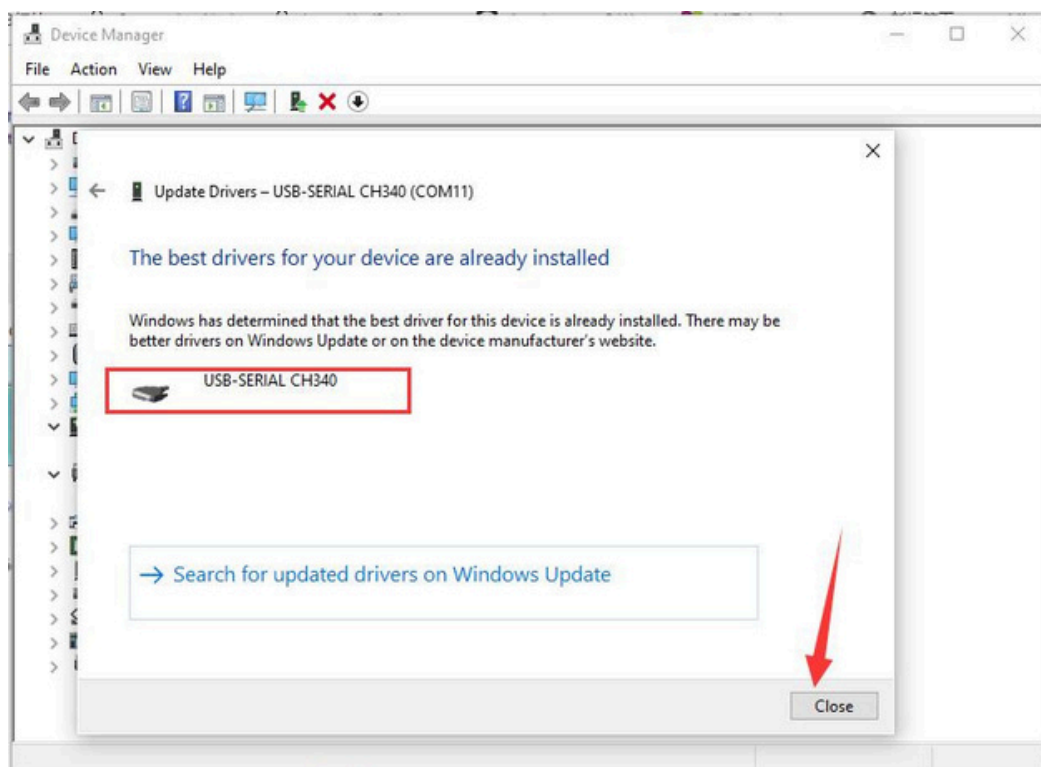
4 Otevřete složku "[Angličtina\Arduino \(zkušený student\)\3.Řidič Soubor ovladače install\CH340 - Windows](#)" a vyhledejte požadovaný soubor ovladače „USB_Drive_CH341_3_1“. Zkopírujte cestu k ovladači zobrazenou na obrázku.



- ⑤ Přidejte zkopírovanou cestu ovladače a klikněte na tlačítko „Další“.

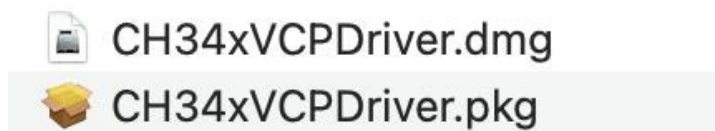


- ⑥ Po dokončení instalace ovladače obdržíte potvrzení zpráva. Po instalaci klikněte na tlačítko „Zavřít“.

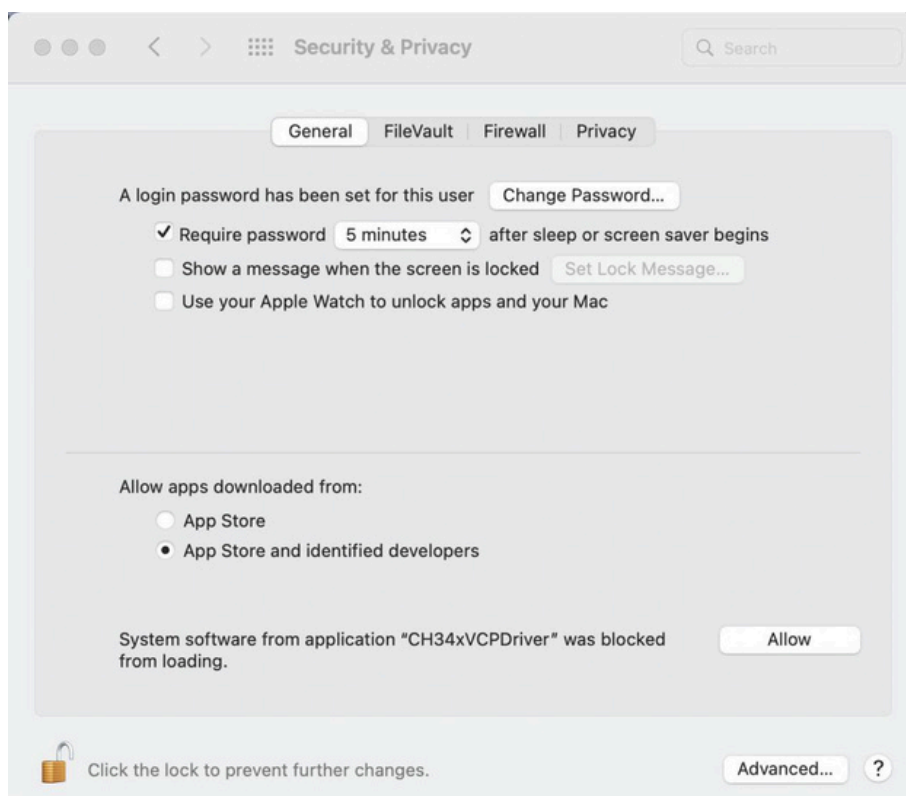


2. Režim instalace ovladače systému MAC OS

① Otevřete složku "[Angličtina\Arduino \(zkušený student\)\3.Řidič](#)
[Instalační soubor ovladače CH340 – MAC](#)" a vyhledejte požadované soubory ovladače „CH34xVCPDriver“ a „CH34xVCPDriver.pkg“.

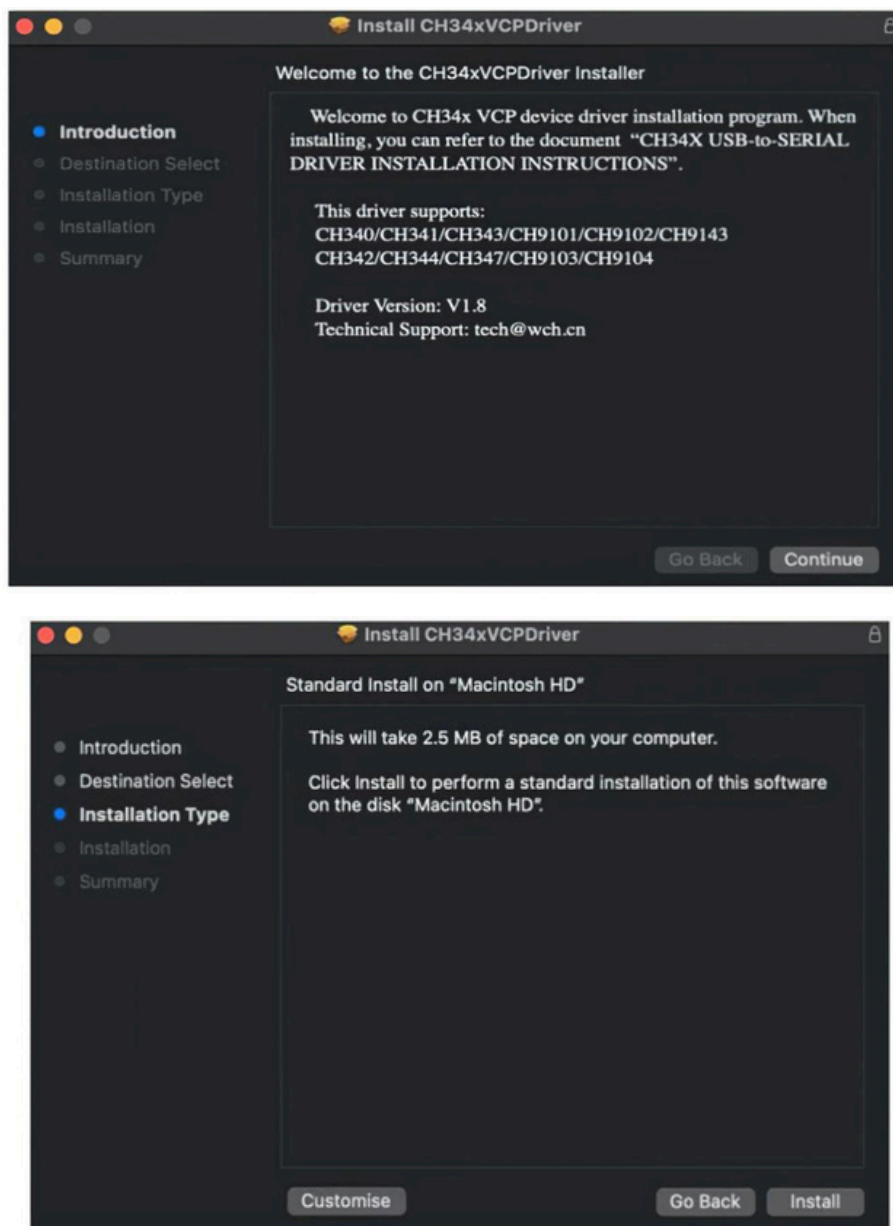


2 Před instalací přejděte do sekce „Předvolby systému“ -> „Zabezpečení a soukromí“ „->“ Obecné“ a v části „Povolit stahování aplikací z následujících adres“ vyberte „Mac App Store a identifikovaní vývojáři“, aby ovladač fungoval správně.

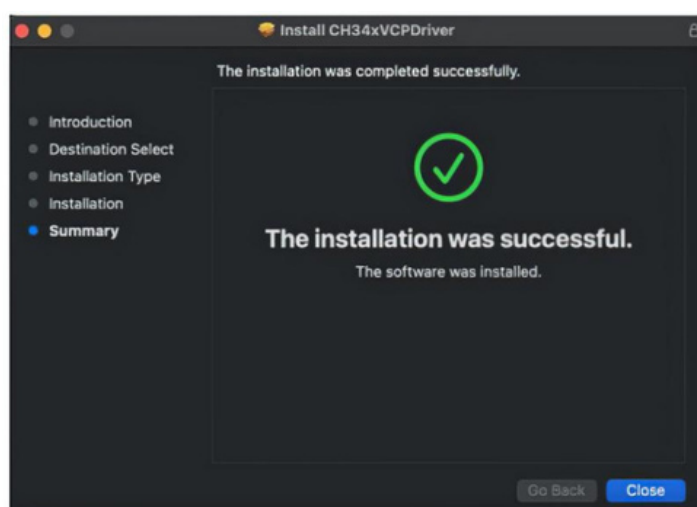
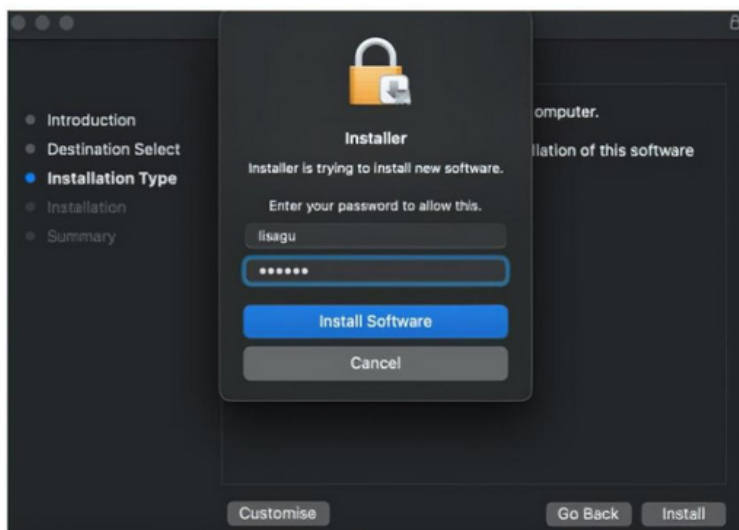


Poznámka: Výchozí způsob instalace ovladače ve formátu .pkg naleznete v kroku③" Pokud je vaše verze OS X 11.0 nebo vyšší a nefunguje podpora Rosetta, viz prosím krok"4"

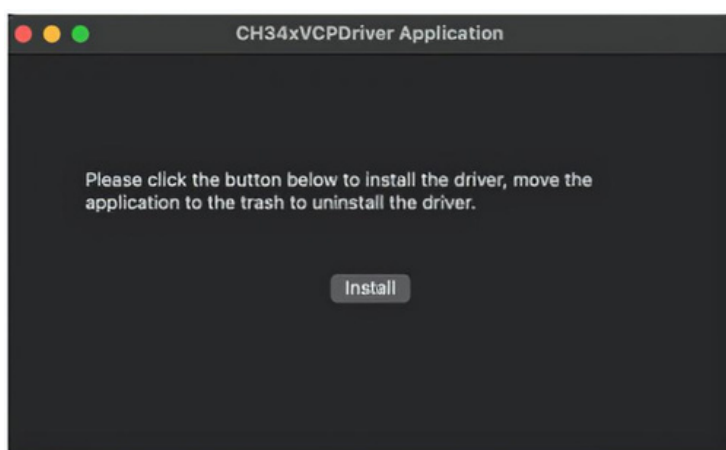
- ③ Chcete-li nainstalovat ovladač ve formátu .pkg, klikněte na soubor ovladače -> „Pokračovat“ -> "Instalovat".



Instalace proběhla úspěšně.



Nainstalujte ovladač formátu pkg v OS X 11.0 a novějším: otevřete "LaunchPad"->"CH34xVCPDriver"-> Instalovat.



Pokud používáte OS X 10.9 až OS X 10.15, klikněte na Restartovat pro restartování počítače a po restartu proveďte následující kroky.

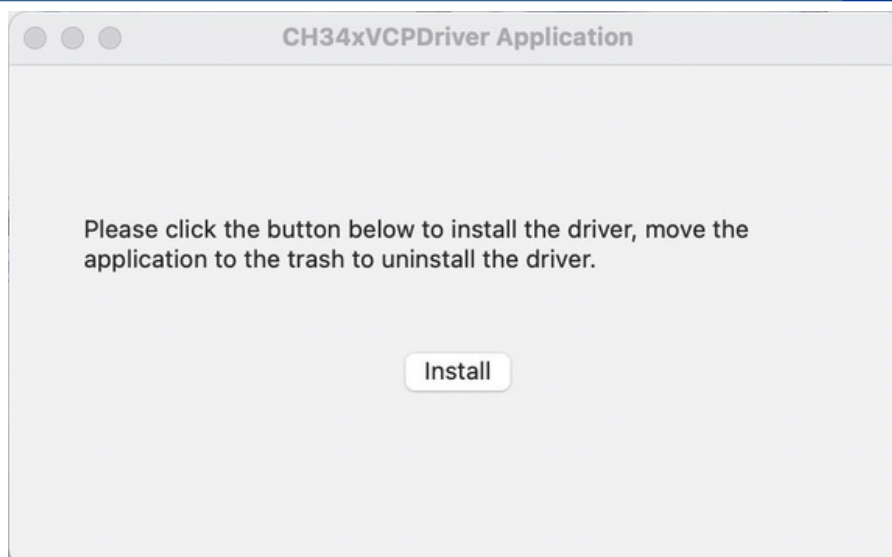


- 4 Nainstalujte ovladač dmg, klikněte na soubor dmg a přetáhněte jej do souboru „CH34xVCPDriver“, do složky aplikací operačního systému.



Pak otevřete "LaunchPad" -> "CH34xVCPDriver" -> Instalovat.



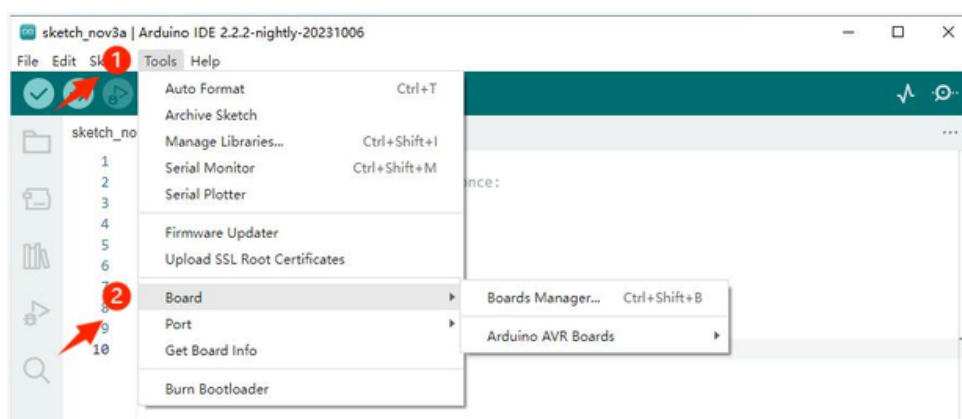


Instalace proběhla úspěšně.

III. Instalace knihovny ESP32

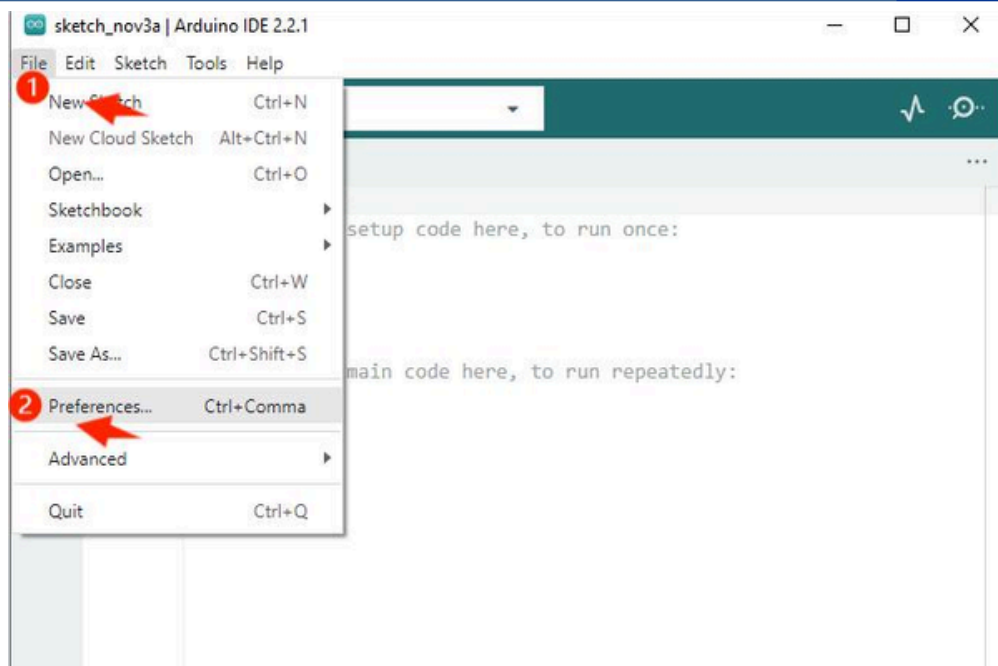
Protože robotické rameno používá řídicí desku ESP32, je nutné přidat také knihovnu ESP32, aby bylo možné desku ESP32 naprogramovat v Arduino IDE.

Když otevřete Arduino IDE, vyberte Nástroje>Deska, zjistíte, že Arduino IDE obsahuje pouze desky Arduino AVR a žádný ESP32.

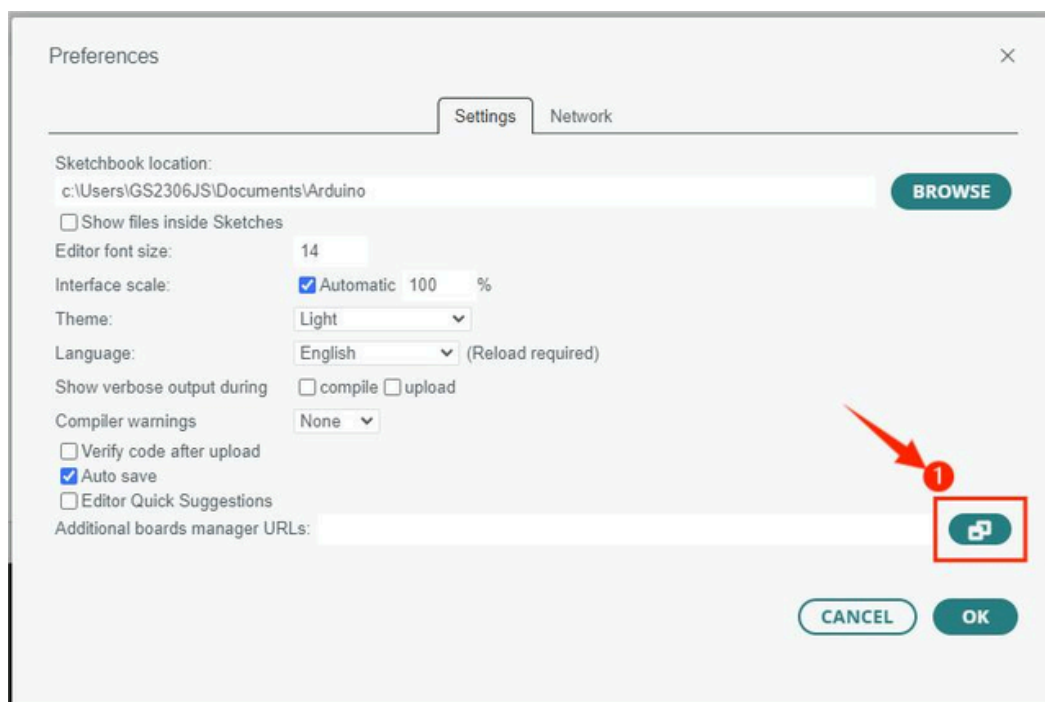


Dále musíme nainstalovat knihovnu ESP32, postupujte takto:

- ① Otevřít Soubor > Předvolby

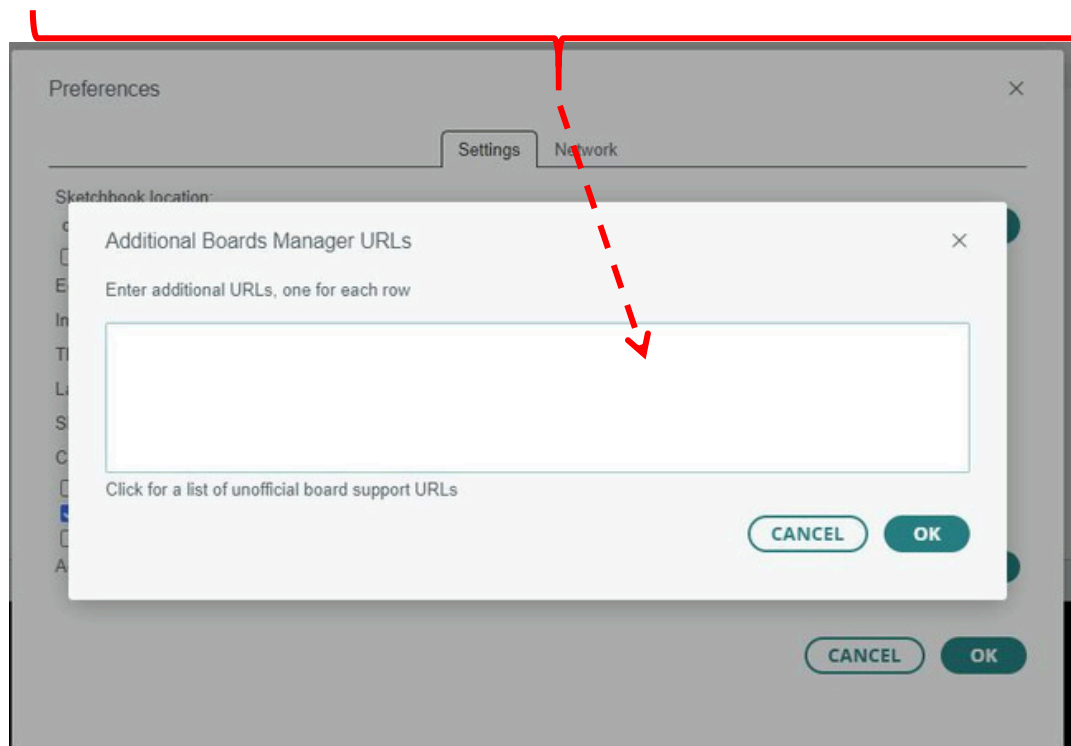


2 Přidejte URL adresu pro správu řídicí desky.

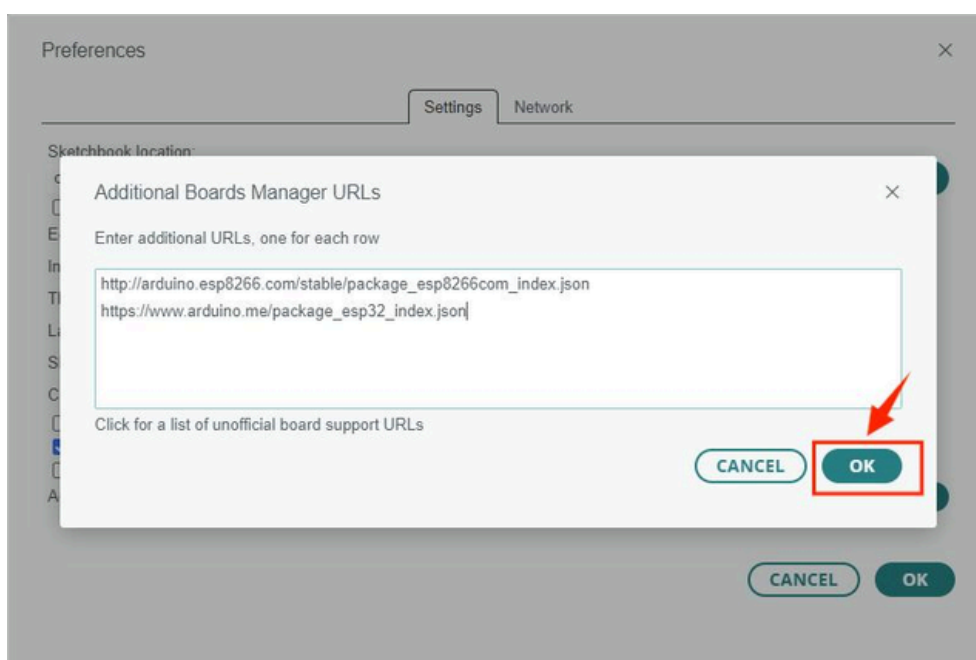


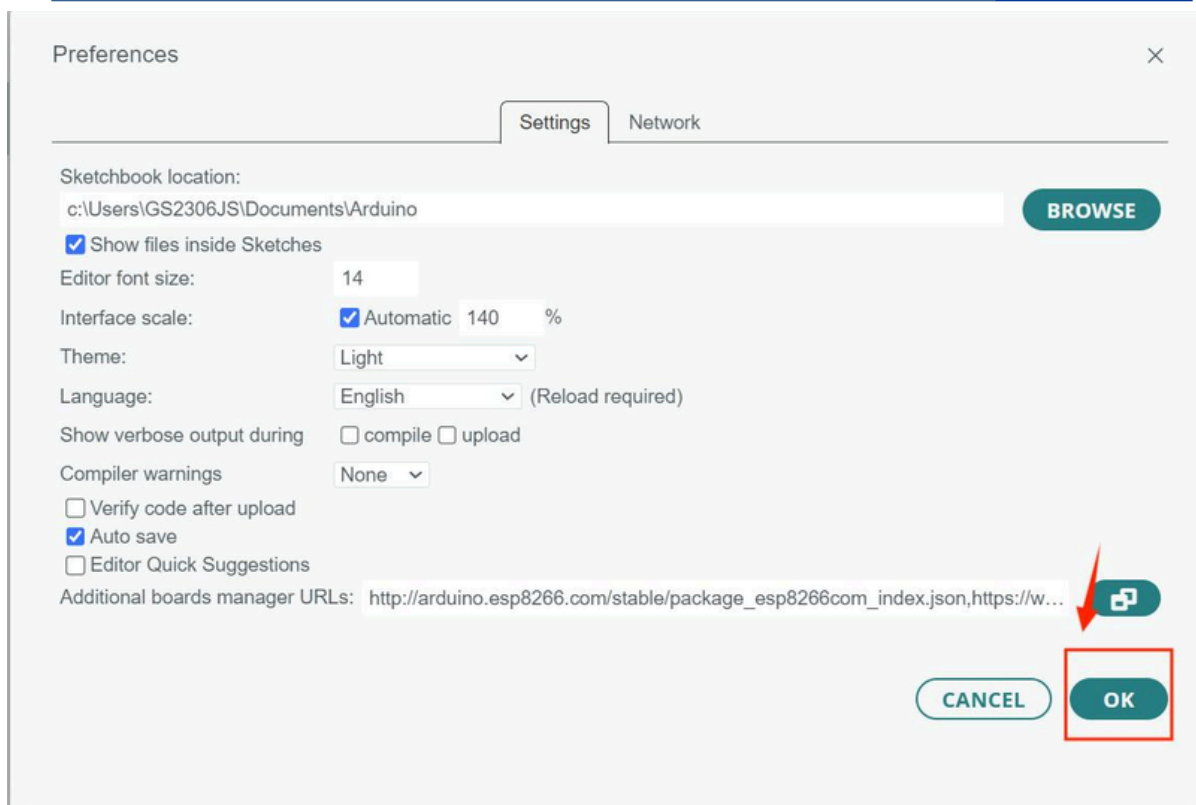
- ③ Zkopírujte URL adresu z dolního textového pole a přidejte ji do složky „Další nástěnky“. URL adresy správce

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



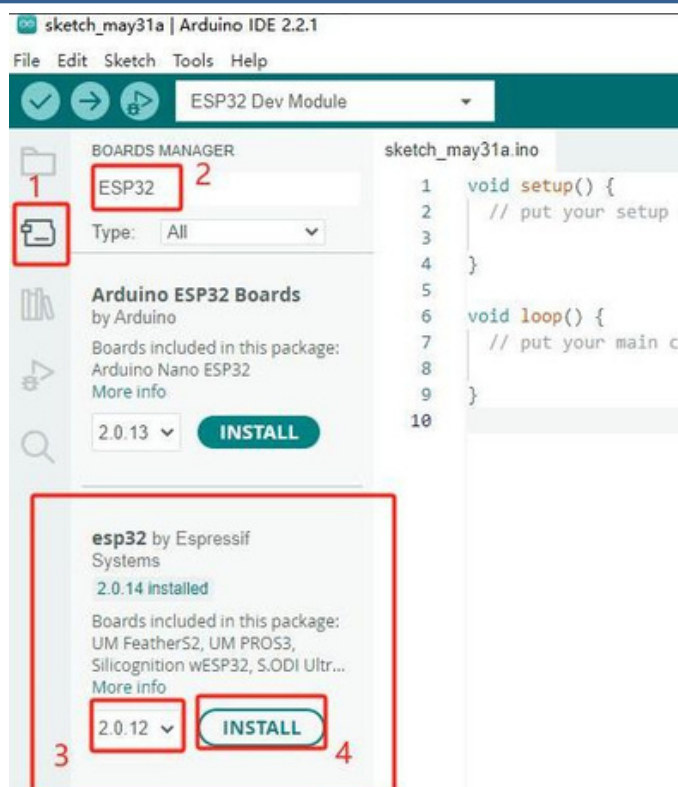
- 4 Po přidání URL adresy klikněte na tlačítko „OK“.



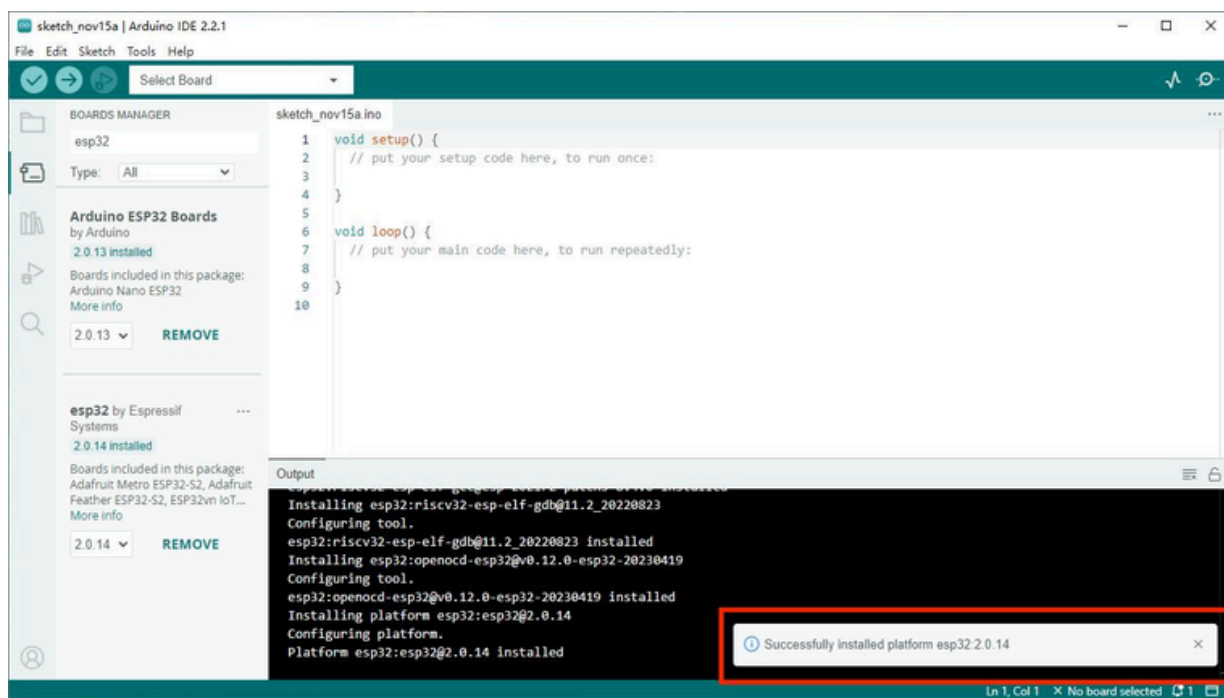


⑤ Klikněte na ikonu Desky v levém panelu nástrojů Arduina a poté na Zobrazí se rozhraní „SPRÁVCE DESEK“. Do vyhledávacího pole zadejte „ESP32“ pro spuštění vyhledávání. Z výsledků vyhledávání vyberte balíček desek ESP32 (verze 2.0.12) a klikněte na „INSTALOVAT“ pro jeho instalaci.

Poznámka: Nainstalujte si prosím verzi 2.0.12 (neinstalujte verzi 3.0 nebo vyšší, protože pozdější nahrávání programů může způsobit chyby).

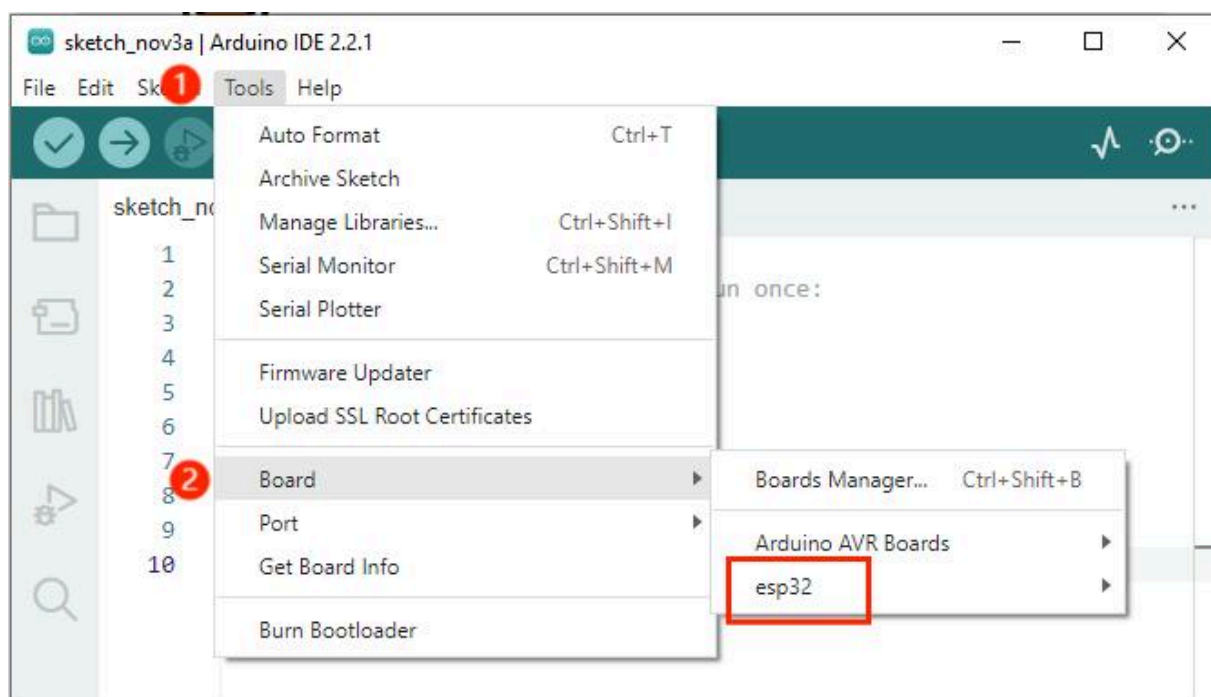


⑥ Jakmile se zobrazí následující rozhraní, počkejte prosím na dokončení instalace. Je hotovo, zavřete Arduino IDE.



⑦ Znovu otevřete Arduino IDE, vyberte Nástroje > Deska a uvidíte soubor esp32.

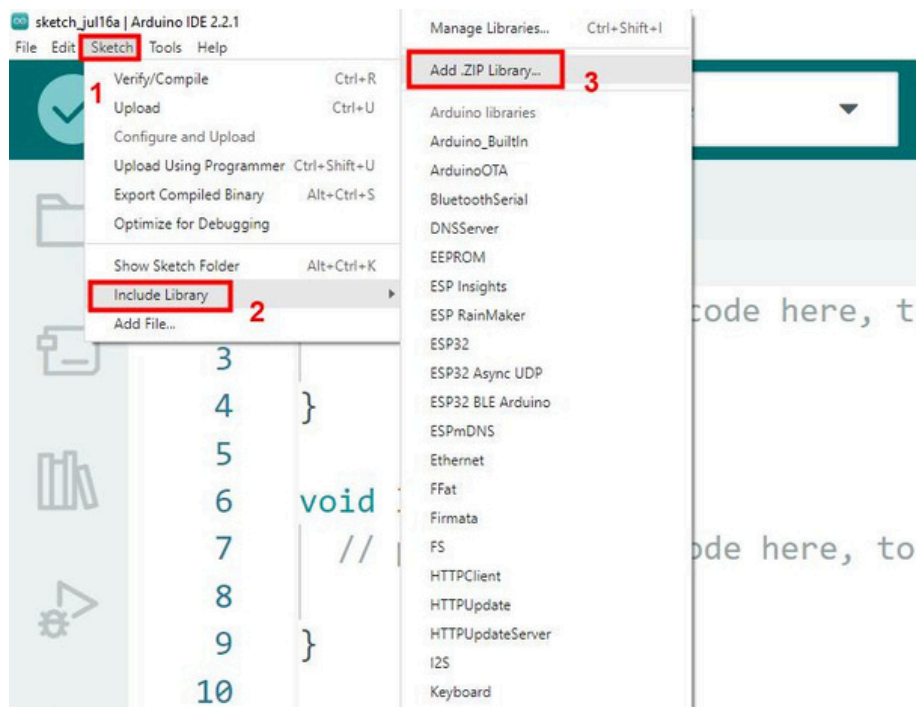
objeví se deska.



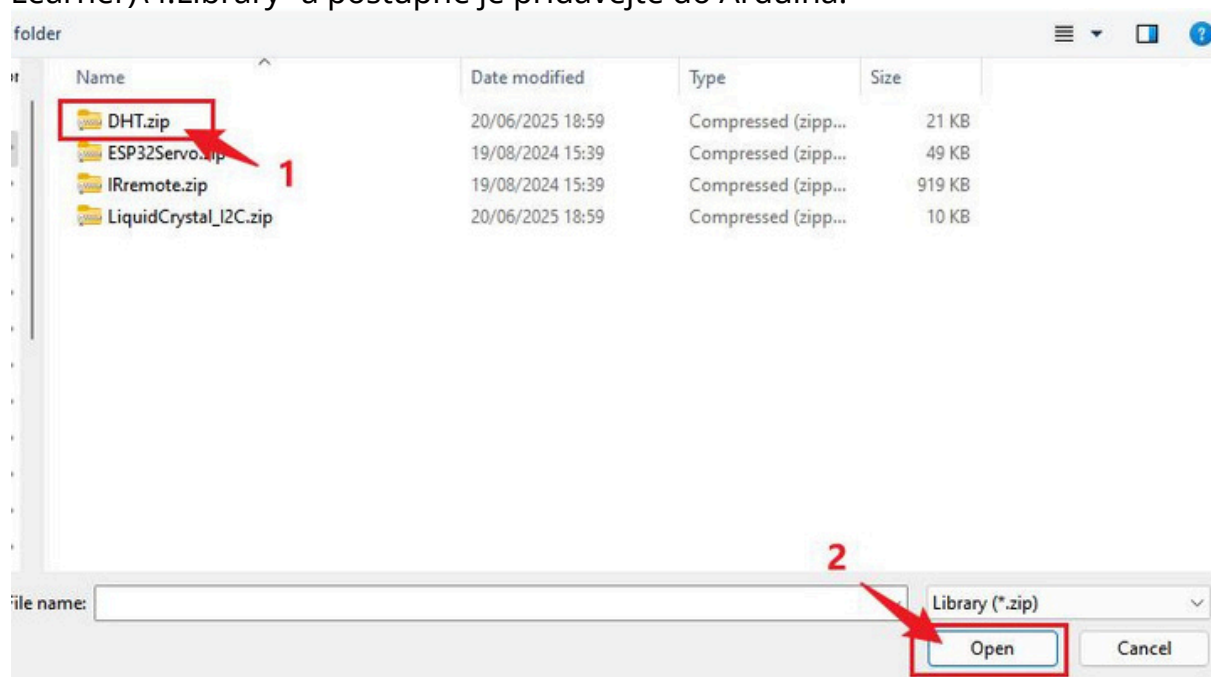
IV. Přidání souborů knihovny

Chcete-li do Arduino IDE nainstalovat novou knihovnu, můžete ji přidat importem knihovny .ZIP.

- ① Otevřete Arduino IDE, klikněte na Sketch klikněte na Include Library Přidat knihovnu .ZIP.



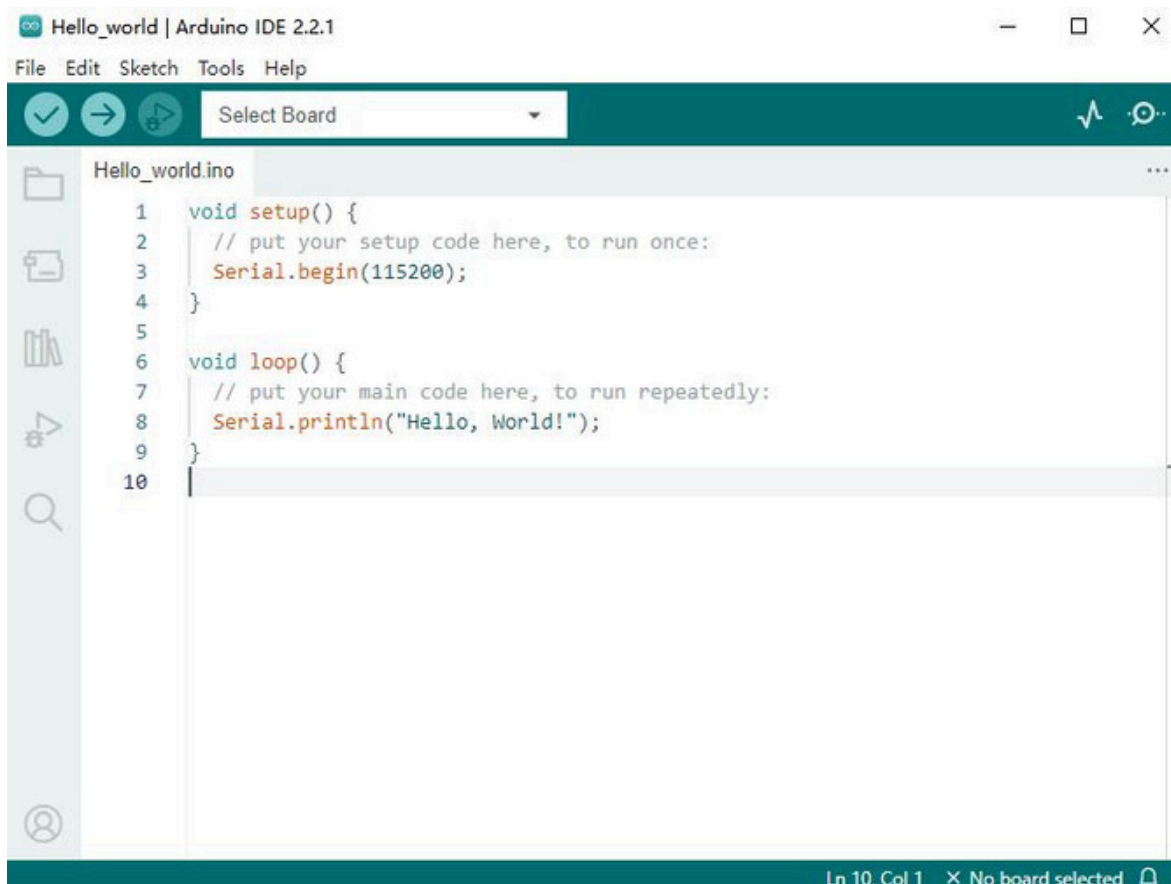
2 Vyberte komprimované soubory v knihovně „English\Arduino(Experienced Learner)\4.Library“ a postupně je přidávejte do Arduina.



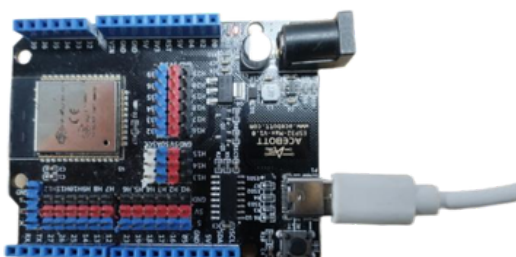
V. Nahrávání programu

Aby bylo možné ovládat desku ESP32 a provádět úkoly podle našich instrukcí, je třeba na desku nahrát příkazový program. Dále se naučíme, jak nahrát program „Hello World“ do ESP32, díky kterému nás deska bude neustále vítat.

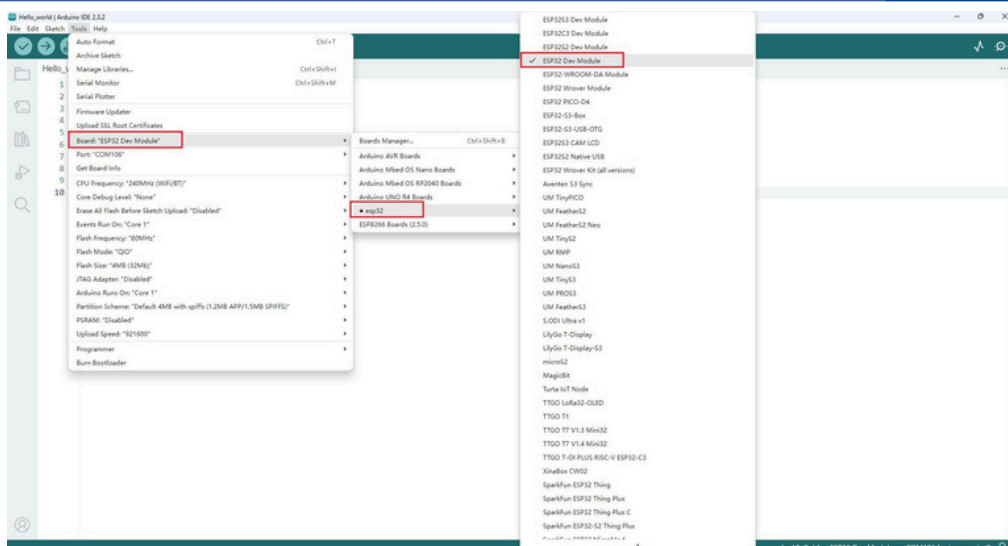
- ① OTEVŘENO "[Hello_world.ino](#)" v "Angličtina\Arduino (zkušený Žák)\2.Kód\Hello_world".



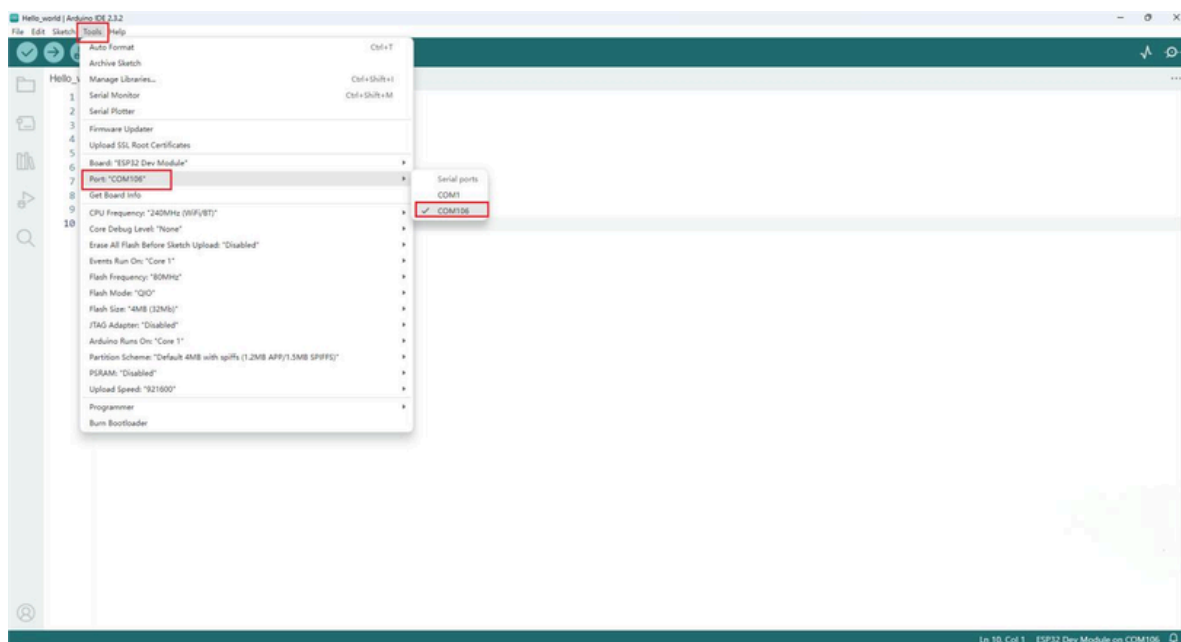
- 2 Připojte desku ESP32 k počítači pomocí kabelu typu C.



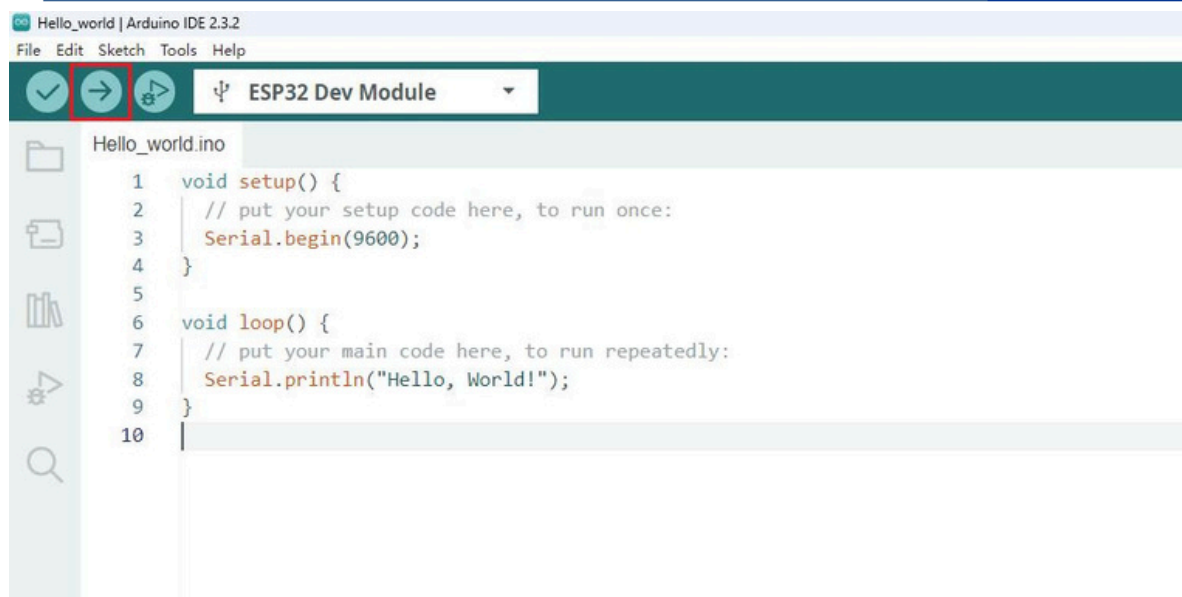
- ③ Vyberte si desku: Nástroje > Deska > esp32 > Vývojový modul ESP32.



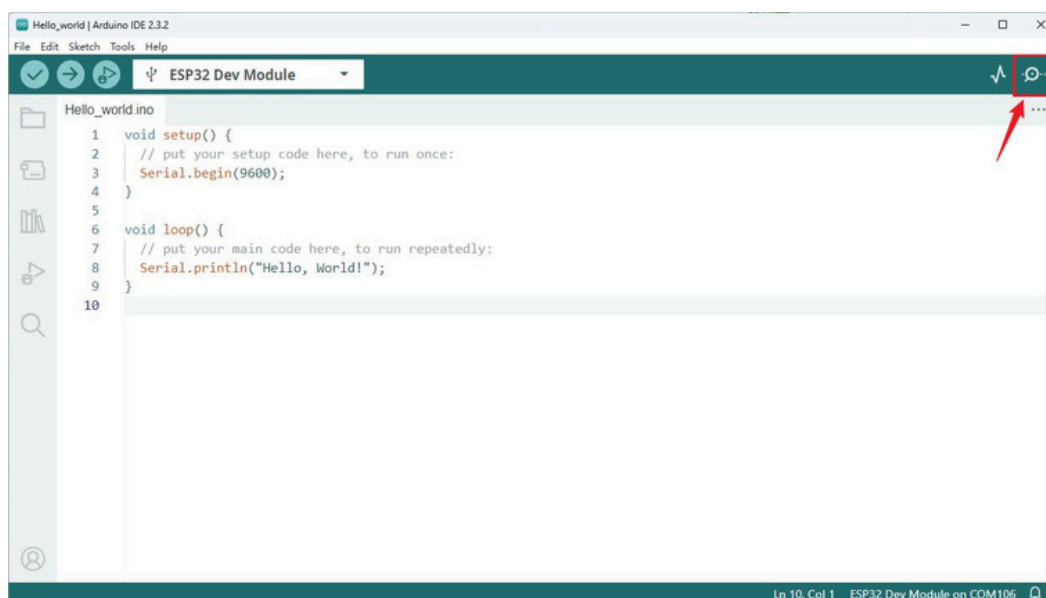
4 Vyberte port: Nástroje > Port > COMx (číslo COM portu může liší se pro každý počítač; číslo portu, které zmizí po odpojení a znovu se objeví po připojení, je číslo vašeho portu).



⑤ Klikněte na  tlačítko pro nahrání programu na desku ESP32.



⑥ Po úspěšném nahrání kódu klikněte na tlačítko  tlačítko pro otevření sériový monitor.



Dále uvidíte, že sériový monitor bude nepřetržitě tisknout „Hello World!“. (Poznámka: Ujistěte se, že jste jako přenosovou rychlost vybrali 115200.)



Popis instrukce

setup() : Když se program spustí, instrukce uvnitř setup() jsou spuštěny jako první. Obvykle se používá k inicializaci proměnných, I/O portů a některých funkcí. Funkce setup() se spustí pouze jednou, když je vývojová deska zapnuta nebo resetována.

loop() : Funkce loop() se provede po funkci setup() a její instrukce lze spouštět nepřetržitě.

Serial.begin(val): Otevře sériový komunikační port a nastaví

Přenosová rychlost (baudová rychlost) desky ESP32. val představuje hodnotu baudové rychlosti (běžné baudové rychlosti jsou 9600, 115200).

Poznámka: Při komunikaci mezi zařízeními musí být přenosové rychlosti zařízení stejné, jinak přijímající strana obdrží zkreslená data místo správných dat.

Serial.println(hodnota): Výstup dat do sériového portu počítače.

Přenášená data mohou být libovolného typu. val představuje data, která mají být vytištěna.

Kapitola 2 Modul Učení

Tato kapitola provede studenty principy fungování a používáním běžných elektronických modulů. Prováděním samostatných experimentů s každým modulem si studenti postupně osvojí dovednosti v oblasti řízení hardwaru a programování, čímž položí pevný technický základ pro samostatný vývoj a komplexní projekty.

Lekce 1 LED



Cíle kurzu

- Pochopení struktury a principu fungování LED diod
- Pochopíte funkci rezistorů a jak určit jejich hodnoty odporu
- Zvládněte základní strukturu nepájivého pole
- Zvládnutí používání digitálních výstupních příkazů ESP32 IO pomocí
- programovacích instrukcí k aktivaci LED diody v ESP32



Úvod do kurzu

Světlo hraje v našich životech klíčovou roli.

Ať už je to světla, ulice nebo zadní světla a semaforey, světlo nejen přináší osvětlení,



ale také sděluje informace a bezpečnost. Zapínání a vypínání světel nám může pomoci identifikovat směr, rozpoznávat signály a dokonce nás upozornit na bezpečnost. Například zadní světla auta se rozsvítí, aby signalizovala autu jedoucímu za námi „Brzdím“, a semafor se rozsvítí červeně, aby nám řekl „Prosím, zastavte“. Zejména LED diody jsou klíčovým vynálezem moderních technologií. Díky své úspornosti energie, šetrnosti k životnímu prostředí, dlouhé životnosti a kompaktním rozměrům se široce používají v každém aspektu našeho života.

V této lekci se naučíme, jak používat řídicí desku ESP32 k ovládání zapínání a vypínání LED diod.

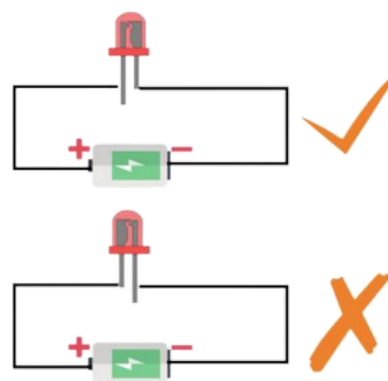


Body znalostí

1.LED LED diody jsou elektronické součástky, které přeměňují elektrickou energii na světlo. Když proud protéká z kladného pólu na záporný pól LED diody, polovodičový materiál uvnitř LED diody uvolní elektrickou energii ve formě světla a vyzařuje světlo. Protože LED dioda má Charakteristika diody - jednosměrná vodivost, proud může protékat pouze z kladného pólu do záporného pólu a nemůže protékat v opačném směru. Proto je při jejím použití nutné rozlišovat kladný a záporný pól. Pokud jsou kladný a záporný pól zapojeny obráceně, LED dioda se normálně nerozsvítí.



Jak určit kladný a záporný pól LED lampy? Sledujte délku dvou pinů LED lampy. Konec delšího pinu se nazývá anoda (+) a konec kratšího pinu se nazývá katoda (-).



V řídicí desce ESP32 odpovídá vysoká úroveň 5 V nebo 3,3 V, zatímco nízká úroveň 0 V. Když je jeden konec LED diody připojen k pinu I/O a druhý konec k GND (tj. nízká úroveň), nastavení pinu I/O na vysokou úroveň vytvoří v uzavřeném obvodu napěťový rozdíl, čímž se generuje proud (tekoucí z pinu I/O do GND) a LED dioda se rozsvítí. Když je pin I/O nastaven na nízkou úroveň, oba konce LED diody mají nízké napětí, což má za následek žádný napěťový rozdíl a žádný proud, což způsobí, že LED dioda zhasne.

V Arduino můžeme použít "**digitalWrite(pin, hodnota)**" příkaz pro nastavení zadaného pinu na výstup vysoké nebo nízké úrovně. „pin“ se používá k označení ovládaného pinu a „val“ se používá k nastavení, zda je výstup na vysoké (HIGH nebo 1) nebo nízké úrovni (LOW nebo 0).

Například: Ovládání 5. pinu ESP32 pro výstup vysoké nebo nízké úrovně

`digitalWrite(5, HIGH)`: Pátý pin má vysokou úroveň výstupu, čímž se rozsvítí LED. `digitalWrite(5, LOW)`: Pátý pin má nízkou úroveň výstupu, čímž se LED zhasne.

2. Základní součástky obvodu

Aby LED dioda rozsvítila, musí být správně zapojena do obvodu. Základní obvod se obvykle skládá ze čtyř částí: **zdroj napájení, elektrické spotřebiče, dráty a vypínač**. Společně tvoří uzavřený obvod, který umožňuje normální tok proudu.



Napájení Zařízení, které dodává elektrickou energii do obvodu.

Mezi běžné zdroje energie patří suché články, akumulátory, lithiové baterie a generátory. K měření výstupní energie ze zdroje napájení obvykle používáme napětí. Jednotkou napětí je volt (V).



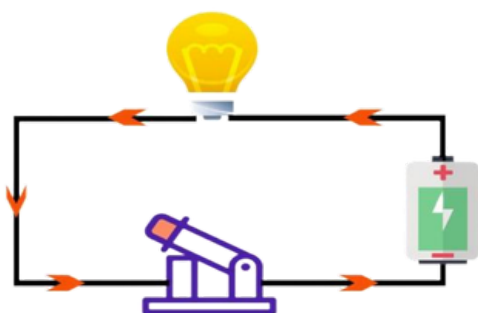
Elektrické spotřebiče Zařízení, která spotřebovávají elektrickou energii a přeměňují ji na jiné formy energie, jako jsou žárovky, motory a reproduktory.



hněv Vodiče, které spojují zdroj energie s elektrickými spotřebiči a jsou zodpovědné za přenos elektrické energie k nim. Mezi běžné materiály patří měď a hliník.



čarodějnice Zařízení, která spojují a odpojují obvody, a tím řídí provoz elektrických spotřebičů.



Když obvod vytvoří uzavřenou smyčku, protéká jím proud. Když proud protéká LED diodou, LED dioda přijímá energii a začne vyzařovat světlo.

3. Proud a napětí

Proud Vodiče obsahují volně se pohybující elektrické náboje. Po připojení ke zdroji energie napětí způsobí, že se náboje ve vodiči pohybují určitým směrem a vytvářejí elektrický proud. Pohyb nábojů v obvodu je podobný toku molekul vody v potrubí.

Napětí Představuje „hnací sílu“ elektrického proudu, stejně jako tlak vody v proudící vodě.

Směr elektrického proudu Proudí od kladného pólu (konec s vysokým napětím) zdroje napájení k zápornému pólu (konec s nízkým napětím) zdroje napájení, stejně jako voda proudí od vysokého k nízkému.

Když žárovkou protéká proud, žárovka se rozsvítí. Ve stejném obvodu, když se napětí (čím větší je tlak) a proud (čím

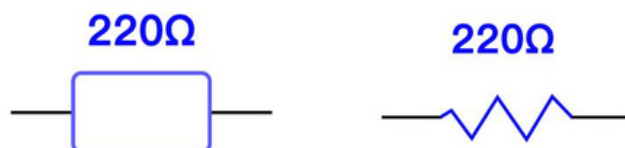
(rychlejší), žárovka bude jasnější. Pokud je však proud procházející LED diodou příliš velký, snadno ji spálí, proto ji musíme použít s rezistorem, který sníží proud procházející rezistorem a ochrání LED diodu.

4. Rezistory

Rezistor je běžná elektronická součástka. Jeho funkcí je omezit tok proudu v obvodu a chránit normální provoz ostatních součástí.

Rezistory se široce používají v různých obvodech.

Dvě běžné schématické značky pro rezistory.



Jednotkou odporu je ohm (Ω) a větší jednotky zahrnují $k\Omega$, $M\Omega$ atd. Jejich vztah lze vyjádřit jako: $1M\Omega=1000k\Omega$, $1k\Omega=1000\Omega$. Při použití rezistoru musíme nejprve znát jeho odpor. Existují dva způsoby:

(1) Sledujte logo barevného kroužkuVětšina rezistorů je na povrchu označena čtyřmi nebo pěti barevnými kroužky a hodnotu odporu lze rychle posoudit podle barvy.

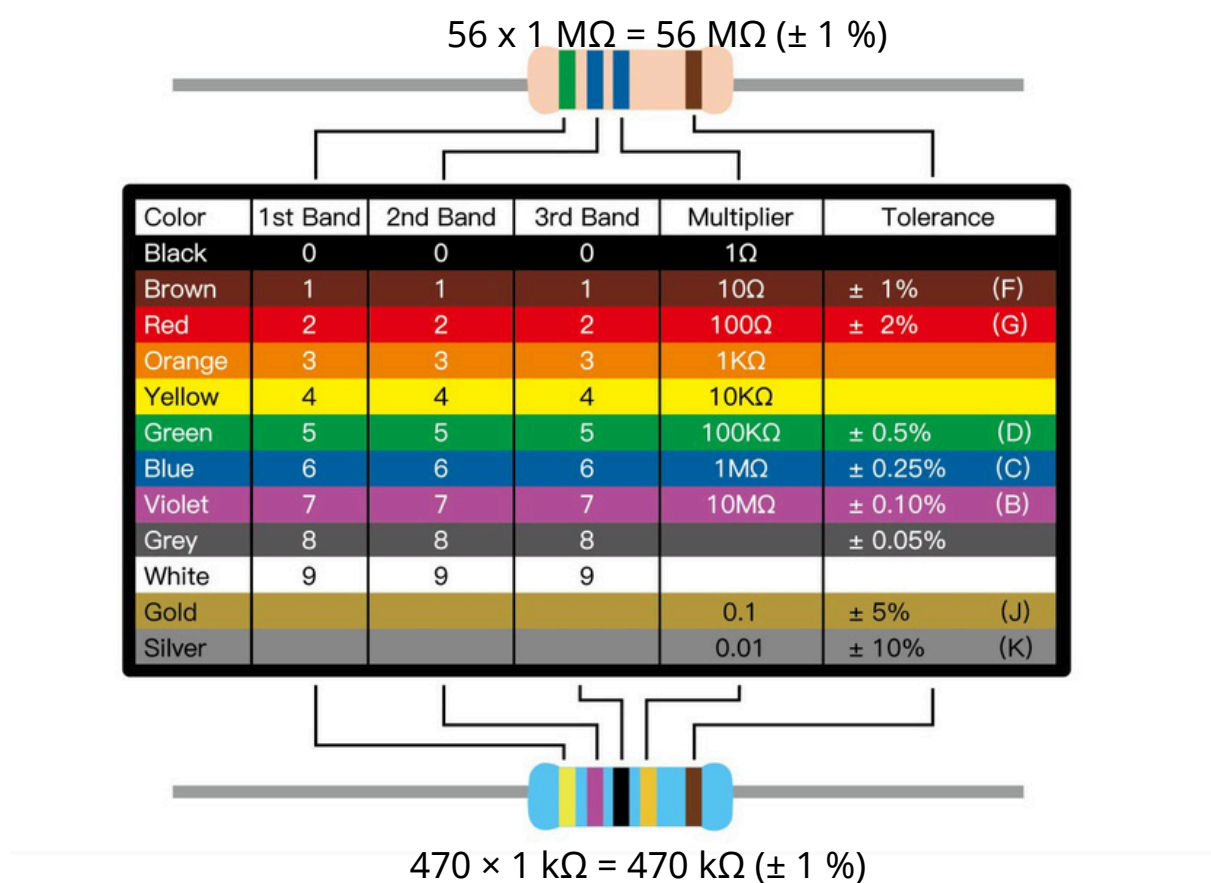
(2) K měření použijte multimetrPro přesné měření je vhodnější přímé měření hodnoty odporu rezistoru.

Doporučuje se použít první metodu, protože je pohodlnější a rychlejší.

Jak identifikovat rezistory pomocí barevných proužků?

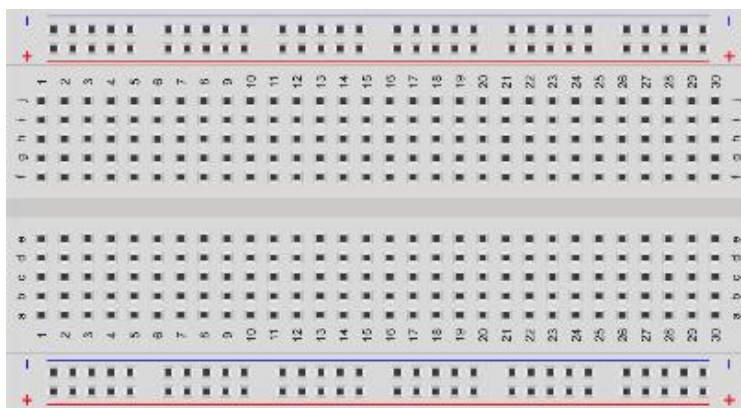
První dva pruhy čtyřpásmového barevného pruhu představují jednotky a desítky; třetí pruh představuje násobitel a čtvrtý pruh představuje toleranci.

První tři pruhy pětipásmového barevného pruhu představují jednotky, desítky a stovky; čtvrtý pruh představuje násobitel a pátý pruh představuje toleranci.



5. Prkénko na výrobu – nepájivé pole

Nepájivé pole je obdélníková plastová deska s mnoha malými otvory. Tyto otvory usnadňují vkládání elektronických součástek pro sestavení obvodů. Jedná se o nepájivý experimentální nástroj pro obvody, který se používá především k rychlému připojení a testování elektronických součástek. Uvnitř má kovové konektorové proužky, které automaticky spojují vložené součástky bez pájení.



Za otvory v nepájivé desce se nacházejí kovové proužky, které tvoří elektrická spojení. Když vložíme piny elektronické součástky do otvorů na nepájivé desce a pokud jsou piny vloženy do stejné sady otvorů spojených kovovými proužky, vytvoří se mezi piny elektrické spojení.



6. Konstanty

Konstanta označuje proměnnou, jejíž hodnota se během provádění programu nemění. Je vhodná pro reprezentaci věcí, jako jsou čísla pinů, pevné fyzikální veličiny nebo řídicí parametry. Definování konstant může váš kód učinit přehlednějším, srozumitelnějším a usnadnit jeho úpravu a údržbu.

V programu, který jsme se v této lekci naučili, definujeme konstantu s názvem LED s hodnotou 12, jak je znázorněno na následujícím obrázku:

```
#define LED 12
```

Tato instrukce znamená, že pinu číslo 12 přidělujeme název LED. V následujících částech programu můžete jednoduše použít název LED k ovládání daného pinu, aniž byste museli pokaždé psát konkrétní číslo 12.

Například pokud se později rozhodnete připojit LED diodu k pinu 13, stačí v tomto jednom řádku definice konstanty změnit číslo 12 na 13. Nemusíte procházet celým programem a hledat a nahrazovat každé místo, kde bylo číslo 12 použito. To výrazně zlepšuje udržitelnost vašeho kódu.

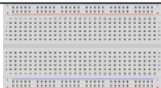


Experiment

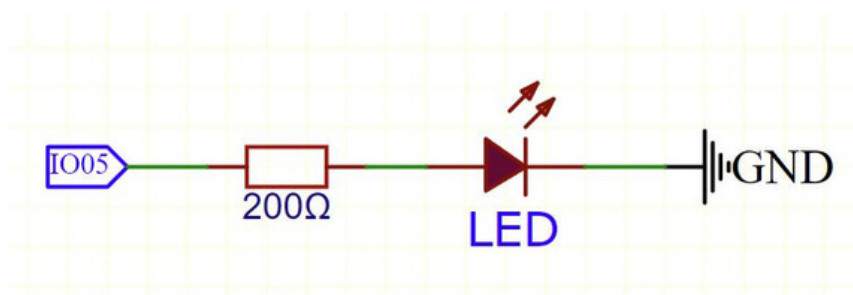
1. Popis projektu

Naprogramujte LED tak, aby se na jednu sekundu rozsvítila a poté na jednu sekundu zhasla, čímž se cyklus opakuje.

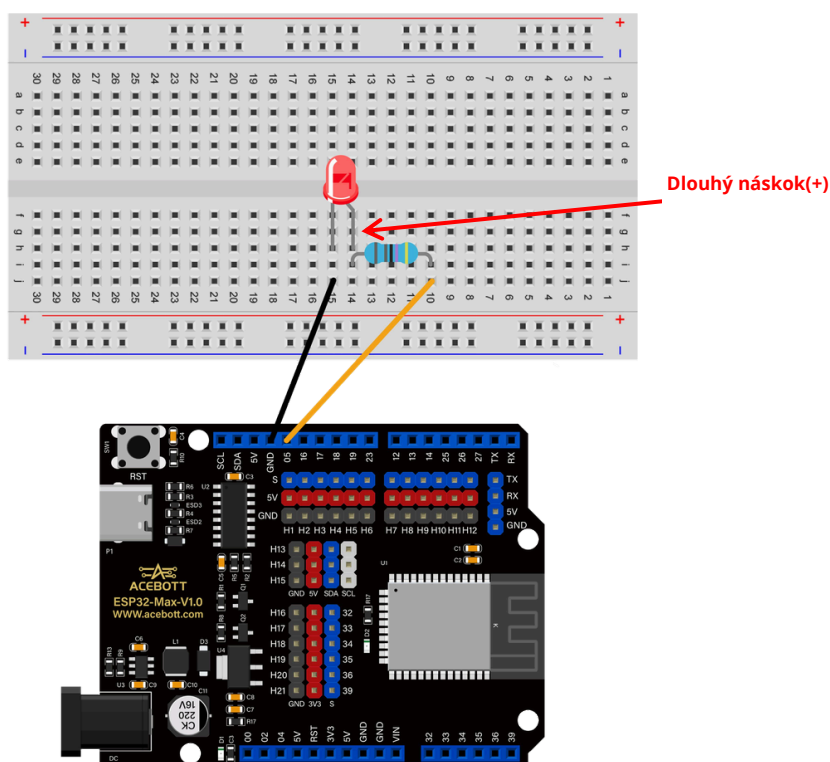
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Rezistor 220Ω	1

3. Schéma zapojení



4. Připojení hardwaru

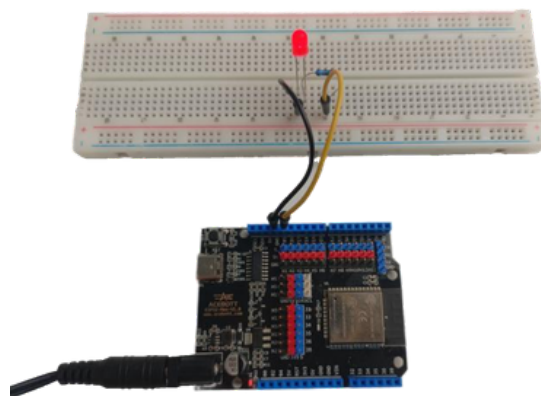


5. Nahrát program

Otevřete "[LED.ino](#)" soubor umístěný v adresáři „English\Arduino(Experienced Learner)\2.Code\lesson 1 LED\LED“ nebo zkopírujte následující kód do Arduino IDE. Poté postupujte podle dříve naučených kroků a vyberte správnou vývojovou desku (ESP32 Dev Module) a příslušný port, a nakonec klikněte  nahrát, nahrát program.

```
// Definice GPIO pinu připojeného k LED
# definovatLEDPin5
prázdnostanastavení() {
    pinMode(LED Pin, VÝSTUP); // Nakonfigurujte pin jako výstup
}
prázdnostasmyčka() {
    digitálníPsaní(LED Pin, VYSOKÝ); // Zapněte LED diodu
    zpoždění(1000); // Počkejte 1 sekundu
    digitálníPsaní(LED Pin, NÍZKÝ); // Vypněte LED diodu
    zpoždění(1000); // Počkejte 1 sekundu
}
```

Po dokončení nahrávání programu začne LED dioda blikat, jak je znázorněno níže:



6. Popis instrukcí

pinMode(pin, režim): Nastaví režim pinu na vstup nebo výstup. „pin“ představuje zadané číslo I/O pinu a „mode“ označuje pracovní režim pinu, který může být buď OUTPUT (výstup), nebo INPUT (vstup).

Poznámka: Pokud chcete přenášet datové informace do řídicí desky prostřednictvím pinu, nastavte pin do vstupního režimu (INPUT). Pokud řídicí deska odesílá řídicí příkazy na pin, nastavte pin do výstupního režimu (OUTPUT).

digitalWrite(pin, hodnota): Nastavuje digitální signálový výstup pinem. „pin“

představuje číslo pinu a „val“ představuje hodnotu výstupního digitálního signálu. Digitální signál má pouze dvě hodnoty: HIGH (vysoká úroveň) a LOW (nízká úroveň).

:

① Před použitím `digitalWrite(pin, val)` musíte nejprve použít `pinMode(pin, režim)` pro nastavení pinu do výstupního režimu.

2Na ESP32 nemají všechny I/O piny výstupní funkci. Konkrétně

Piny 34, 35, 36 a 39 jsou pouze vstupní a nepodporují výstup. **zpoždění**

(čas) Pozastaví program na zadanou dobu. „čas“ představuje dobu trvání pauzy v milisekundách (ms).

Lekce 2 Pulzující světlo



Cíle kurzu

- Pochopení principu fungování dýchacího světla.
- Pochopení definic digitálních a analogových signálů.
- Pochopení technologie PWM.
- Pochopení konceptu proměnných
- Zvládnutí používání proměnných
- Použití technologie PWM k řízení jasu LED



Úvod do kurzu

V přírodě vzestup a pokles dechu, třístění vln a střídání světla a stínu obsahují jedinečné rytmy a melodie. Tyto pohyby nejen dodávají energii životu, ale také propůjčují přírodní krajině bohatou krásu. Dýchající světlo čerpá inspiraci z tohoto přirozeného rytmu a simuluje dýchání.



rytmus postupným stmíváním světla. Nejenže prolamuje chlad a monotónnost elektronických zařízení, ale také jim vdechuje jemný, živý rytmus a dokonale propojuje technologii s krásou přírody.

V této lekci použijeme ovladač ESP32 k ovládání LED světla a dosáhneme tak efektu „dýchajícího světla“.



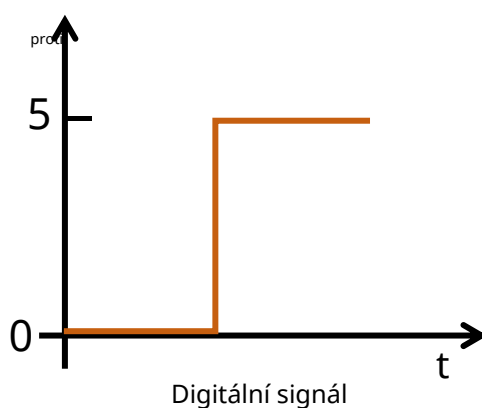
Body znalostí

1. Digitální a analogové signály

V elektronickém světě se signály používají k přenosu informací. Běžné signály se vyskytují ve dvou formách: digitální a analogové.

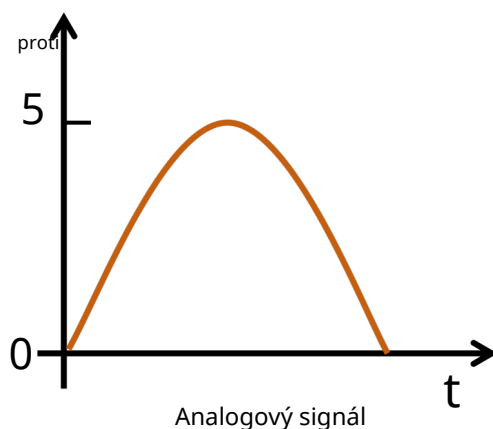
V předchozí lekci jsme použili příkazy `digitalWrite()`, které způsobily, že I/O port ESP32 bude vydávat napětí buď 5 V, nebo 0 V, čímž se LED dioda rozsvítí a zhasne. Tento typ signálu, který má pouze dva stavy, se nazývá digitální signál, podobně jako odpověď na otázku typu pravda/nepravda – buď pravda, nebo nepravda. Mnoho každodenních scénářů se na tuto jednoduchou a spolehlivou metodu ovládání spoléhá, například zapínání a vypínání inteligentních zámků dveří, spouštění detektorů kouře a reakce na tlačítka dálkového ovladače. V podstatě tyto signály přenášejí příkazy pomocí „0“ a „1“.

Jak je znázorněno na obrázku níže, v daném okamžiku může být napětí buď 0V nebo 5V.



Analogové signály jsou spojitě se měnící signály, které mohou nabývat jakékoli hodnoty napětí v určitém rozsahu. Mnoho fyzikálních veličin v reálném životě, jako je teplota, zvuk a světlo, se spojitě mění, a proto se často vyjadřují a zachycují pomocí analogových signálů.

Jak je znázorněno na obrázku níže, při odečítání napětí v daném okamžiku může mít libovolnou hodnotu mezi 0 a 5 V.

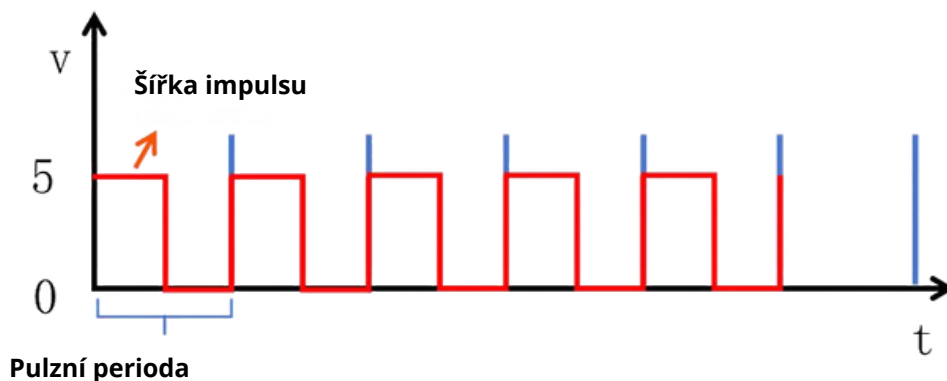


Poznámka: ESP32 nemůže přímo generovat spojitý napěťový signál. Místo toho musíme k simulaci analogového napěťového signálu použít technologii PWM.

2. Principy PWM technologie

PWM (Pulse Width Modulation) je technologie, která využívá digitální signály k simulaci analogových signálů za účelem získání různých hodnot napětí.

Digitální signály mohou přímo řídit připojení (vysoká úroveň) a odpojení (nízká úroveň) obvodu. Častým připojováním a odpojováním obvodu po delší dobu lze generovat pulzní signál, jak je znázorněno na obrázku níže.



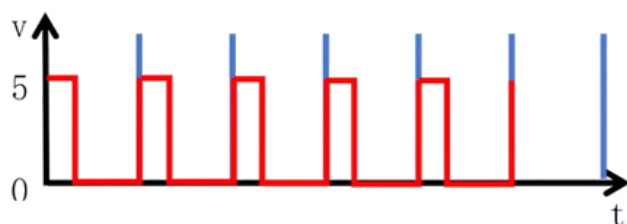
Ve výše uvedeném grafu červená čára představuje pulzní signál. Vzdálenost mezi modrými čarami představuje periodu pulzního signálu. Doba trvání vysokého napětí v každém cyklu se nazývá šířka pulzu a poměr šířky pulzu k periodě se nazývá pracovní cyklus. Čím déle vysoké napětí v cyklu přetrvává, tím větší je šířka pulzu, tím větší je pracovní cyklus a tím vyšší je výstupní napětí. Technologie pulzně šířkové modulace (PWM) mění konečné výstupní napětí změnou šířky pulzu.

V Arduino můžeme použít "**analogWrite(pin, hodnota)**" pro výstup signálu pulzně šířkové modulace (PWM) na zadaný pin. V této funkci je první parametr „pin“ číslo pinu, které se používá k určení, který pin bude vydávat signál PWM, a druhý parametr „value“ je výstupní hodnota PWM, která se používá k nastavení pracovního cyklu signálu PWM. Vstupní rozsah je od 0 do 255, což odpovídá rozsahu pracovního cyklu od 0 % do 100 %.

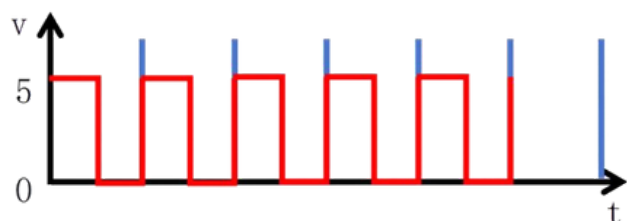
Vezměme si jako příklad pin 5 ESP32. Pokud použijeme "**analogWrite(5, 0)**", bude mít výstupní PWM signál pracovní cyklus 0 %, což odpovídá výstupnímu napětí 0 V. Pokud použijeme "**analogWrite(5, 127)**", bude mít výstupní PWM signál pracovní cyklus 50 %, což odpovídá výstupnímu napětí 2,5 V. Při použití "**analogWrite(5, 255)**" bude mít výstupní PWM signál pracovní cyklus 100 %, což odpovídá výstupnímu napětí 5 V.



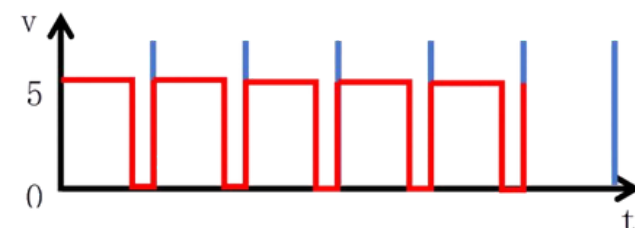
0% pracovní cyklus -- 0 V



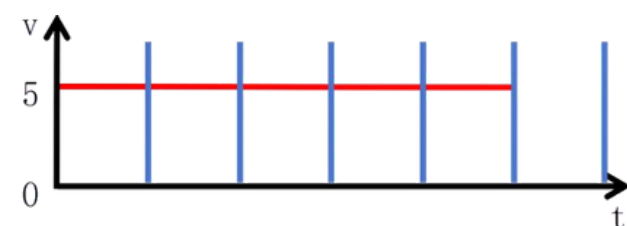
25% pracovní cyklus -- 1,25 V



50% pracovní cyklus -- 2,5 V



75% pracovní cyklus -- 3,75 V



100% pracovní cyklus -- 5 V

3. Pulzující světlo

Dýchací světlo je LED světelný efekt, ve kterém se jas postupně rozjasňuje a ztlumuje, stejně jako lidské dýchání, a vytváří tak rytmický, plynule se měnící vizuální efekt.

Jak fungují pulzování světla?

Klíčem k implementaci „dýchajícího“ světla je ovládání jasu LED diod tak, aby se postupně zvyšoval a poté snižoval v rámci cyklu, a to opakovaně.

Jas LED diody je určen proudem, který jí protéká. Podle Ohmova zákona platí $U=IR$, takže při konstantním odporu platí, že čím větší je napětí, tím větší je generovaný proud. V tomto případě používáme technologii PWM (pulzně šířková modulace) k úpravě průměrného protékajícího napětí.

přes LED, čímž se řídí proud a tím i jas LED.

PWM Technologie přímo nemění napětí, ale řídí „ekvivalentní jas“ LED diody rychlým přepínáním mezi „vysokou úrovní“ a „nízkou úrovní“ a úpravou poměru času, který vysoká úroveň zabírá (tj. pracovní cyklus). Čím vyšší je pracovní cyklus, tím jasnější je LED dioda; čím nižší je pracovní cyklus, tím slabší je jas. Plynulou změnou pracovního cyklu lze dosáhnout jemného, postupného „dýchání“ LED světla.

4. Proměnné

V programu se proměnné používají k ukládání a zpracování dat a fungují jako „krabice“, která může uchovávat hodnoty, což nám usnadňuje čtení, úpravu nebo opětovné použití dat kdykoli v programu.

V programu pro „dýchající světlo“ obvykle používáme proměnnou (například jas) k zaznamenání aktuální hodnoty jasu LED diody.

```
int brightness;
```

Postupným zvyšováním nebo snižováním hodnoty této proměnné podle logiky programu můžeme ovládat LED diodu tak, aby cyklicky svítila mezi jasným a tlumeným světlem, čímž vytváříme efekt jemného „dýchání“.

Formát definice proměnné:

NázevProměnnéDataType = PočátečníHodnota;

V počítačích se mezi běžně používané proměnné řadí hlavně tyto typy:

Typ	Popis	Příklad
celé číslo	Celé číslo (-32 768 až 32 767)	úhel int = 90;

plovák	Číslo s plovoucí desetinnou čárkou (desetinná čísla)	teplota plováku = 23,5;
booleovská hodnota	Booleovská hodnota (pravda nebo nepravda)	bool_ledState = true;
char	Znak (jeden znak)	char písmeno = 'A';
řetězec	Text (posloupnost postav)	Název řetězce = "ESP32";

5."pro"Příkaz smyčky

Smyčka „for“ je struktura příkazů, která opakovaně spouští segment kódu, obzvláště vhodná pro scénáře, kde je znám počet iterací, jako je rozsvícení více LED diod, generování efektu dýchajícího světla nebo opakované čtení a zápis dat. Určuje začátek, konec a způsob provedení smyčky nastavením počáteční hodnoty, podmínky posouzení a operace kroku.

Syntaktický formát smyčky for je následující:

```
pro(Počáteční výraz; Podmínka cyklu; Výraz pro zvýšení hodnoty)
{ Blok příkazů;
}
```

Počáteční výraz Používá se pro inicializaci proměnné smyčky, provede se pouze jednou před spuštěním smyčky.

Podmínka smyčky Detekováno před každou smyčkou. Pokud je podmínka pravdivá, provedou se blok příkazů a inkrementační výraz uvnitř složených závorek. Pokud je podmínka nepravdivá, smyčka končí.

Výraz pro zvýšení Používá se ke zvýšení nebo snížení hodnoty proměnné smyčky, provádí se po instrukcích v bloku příkazů smyčky.

Například:

```

pro(celý textjas =0jas <=255jas+=5) { analogWrite(LED, jas); zpoždění(5);//
Ovládání rychlosti dýchání }

```

pro (int jas = 0; jas <= 255; jas + = 5) Hodnota int brightness = 0 v závorkách označuje vytvoření proměnné s názvem brightness a její inicializaci na 0. Brightness <= 255 je podmínkou pro pokračování smyčky. Pokud není splněna, smyčka končí. Brightness += 5 označuje zvyšování jasu o 5 pokaždé, dokud nepřekročí 255.

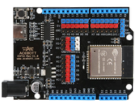
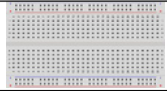


IV. Experiment

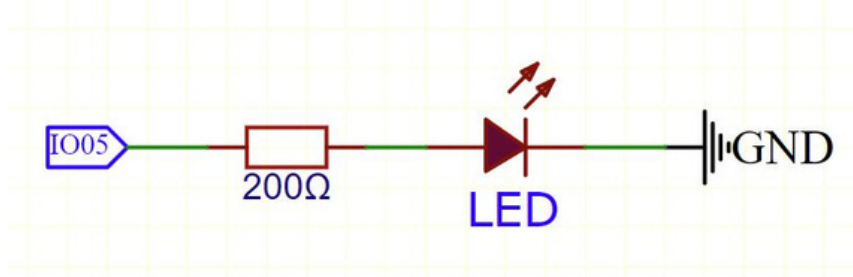
1. Popis projektu

Nechte LED diodu postupně zjasňovat a poté postupně slábnout.

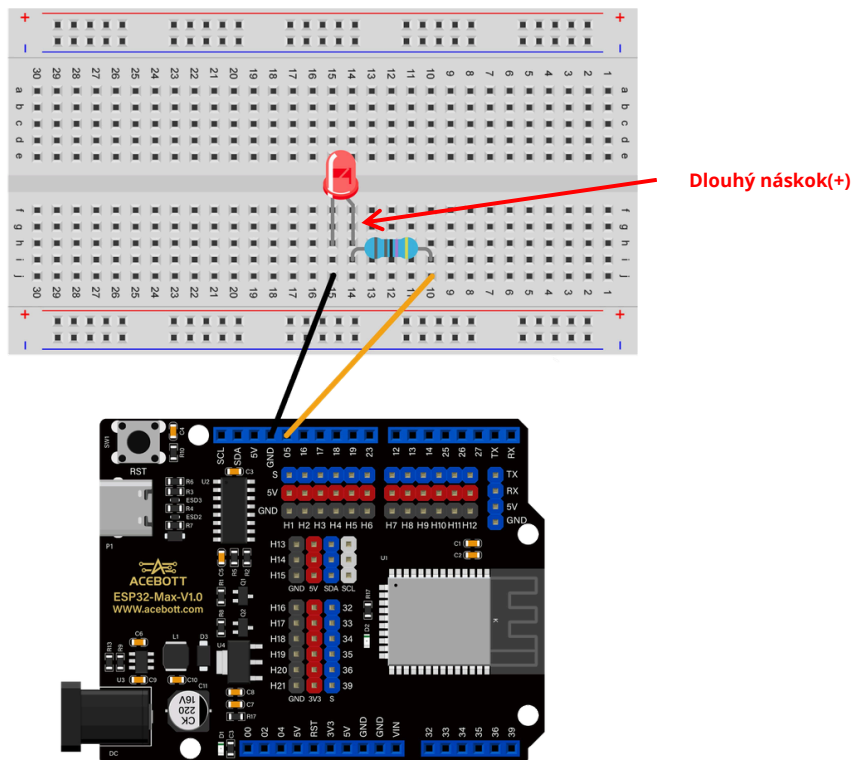
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Rezistor 220Ω	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Breathing_light.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 2 Breathing light\Breathing_light" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

dříve naučené kroky, klikněte  nahrát program.

```
celé čísloLED Pin =5;
prázdnotanastavení() {
  pinMode(LED Pin, VÝSTUP); }

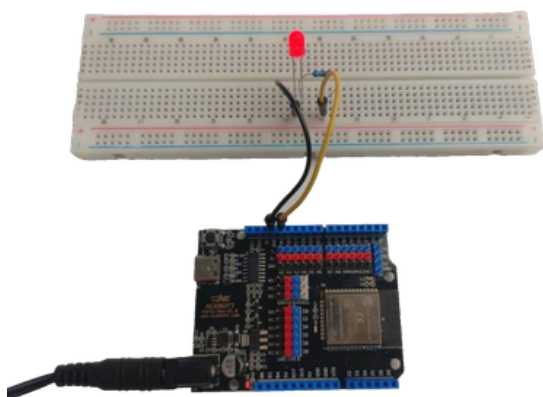
prázdnotasmýčka() {
  // Postupně zvyšujte jas LED diody
```



```
pro(celé číslo jas = 0 jas <= 255 jas += 5) { analogWrite(LED Pin, jas); zpoždění(5); // Ovládání rychlosti dýchání }

// Postupné snižování jasů LED diody
pro(celé číslo jas = 255 jas >= 0 jas -= 5) { analogWrite(LED Pin, jas); zpoždění(5);
;
}
}
```

Po nahrání programu se LED dioda postupně rozsvěcuje a poté postupně zhasíná, jak je znázorněno níže:



6. Popis instrukcí

analogWrite(pin, hodnota): Vysílá analogovou „hodnotu napětí“ (ve skutečnosti pulzní signál PWM) na zadaný pin, která se používá k řízení „kontinuálních efektů“, jako je jas zařízení nebo rychlost motoru. pin je číslo pinu a hodnota je výstupní hodnota v rozsahu od 0 do 255.

Lekce 3 RGB LED



Cíle kurzu

- Pochopení struktury a principů fungování RGB lamp
- Zvládnutí základních principů míchání barev v RGB lampách (princip aditivních barev)
- Použití programovacích instrukcí k ovládní barev RGB lamp



Úvod do kurzu



Mnoho z těchto barevných změn je ve skutečnosti neoddělitelných od magické složky – RGB světla. RGB znamená červenou, zelenou a modrou. Jsou jako základní pigmenty na paletě, které nám umožňují vytvářet miliony barev jejich smícháním v různých poměrech. Programováním můžeme nejen zobrazovat pevné barvy, ale také ovládat jejich rytmické změny, simulovat dýchající světla, vytvářet duhové přechody a další skvělé efekty.

V této lekci prozkoumáme principy fungování RGB lamp a naučíme se, jak pomocí ESP32 ovládat RGB lampy a vytvářet tak různé barevné efekty.

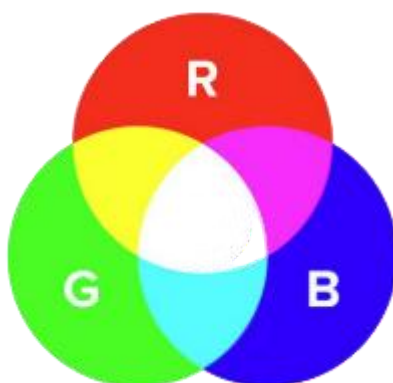


Body znalostí

1. Tři základní barvy světla

Téměř všechny barvy v přírodě lze vytvořit smícháním červené, zelené a modré v různých poměrech. Naopak většinu barev v přírodě lze rozložit na červenou, zelenou a modrou, takže červená (R), zelená (G) a modrá (B) se nazývají tři základní barvy.

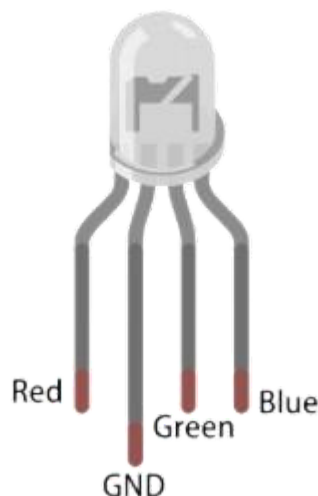
Když se dvě nebo tři základní barvy překrývají v určitých poměrech, vzniknou nové barvy. Například tři velké kruhy na míchacím kole tří základních barev představují červenou, zelenou a modrou. Překrývající se kruhy představují míchání barev. Například smícháním stejného množství červené a zelené vznikne žlutá, smícháním stejného množství červené a modré vznikne fialová a smícháním stejného množství červené, zelené a modré vznikne bílá. Při smíchání tří základních barev v různých poměrech se výsledná barva bude lišit, což umožní vytvořit širokou škálu barev.



2. RGB LED

RGB LED dioda je integrovaná optoelektronická součástka, která integruje LED diody tří základních barev (červené, zelené a modré) do jednoho pouzdra. Tyto tři LED diody sdílejí společný katodový pin, zatímco každý anodový pin...

ovládá intenzitu odpovídající barvy.



Změnou intenzity elektrického signálu aplikovaného na každou anodu pomocí technologie PWM můžeme vytvářet různé barvy. Například smícháním vysoce intenzivního červeného a zeleného světla vzniká žlutá, zatímco smícháním modrého a zeleného světla vzniká azurová.

Běžná tabulka barev RGB

Barvy	R	G	B.
Bílý	255	255	255
Červený	255	0	0
Zelený	0	255	0
Modrý	0	0	255
Žlutá	255	255	0
Fialová	80	0	80

V Arduino můžeme použít **analogWrite(pin, hodnota)** nezávisle ovládat jas kanálů R, G a B v RGB LED (s rozsahem 0 až 255). Úpravou kombinace úrovní jasu ze tří barevných kanálů můžeme dosáhnout toho, aby LED dioda zobrazovala různé barvy.

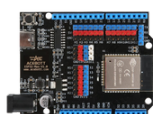
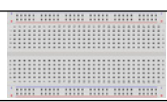


Experiment

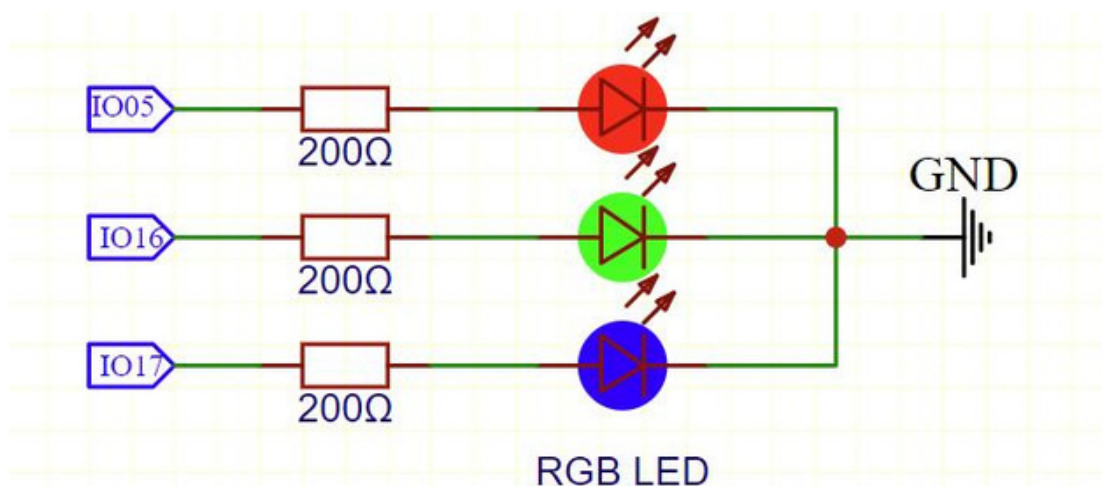
1. Popis projektu

Ovládejte RGB LED diodu tak, aby cyklicky přepínala mezi červenou, zelenou, modrou, žlutou a fialovou.

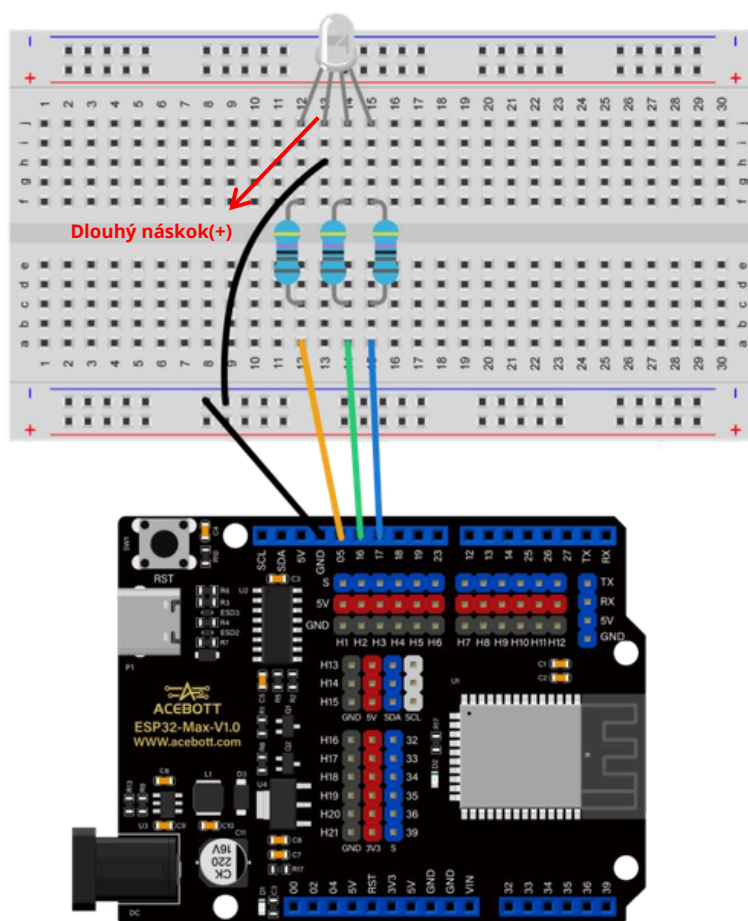
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	RGB LED	1
	Rezistor 220Ω	3

3. Schéma zapojení



4. Hardwarové připojení



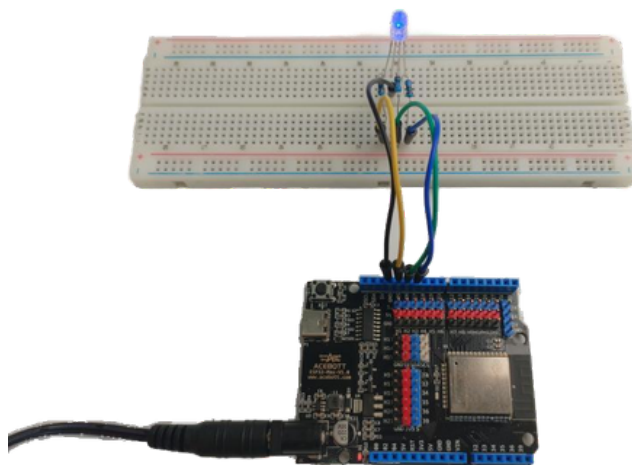
5. Referenční program

Otevřete "[RGB.ino](#)" soubor v "Angličtina\Arduino (zkušený Žák)\2.Kód\lekce 3 RGB\RGB" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušný port podle dříve naučených kroků klikněte na  nahrát program.

```
// Define RGB LED pins
const int redPin = 5;
const int greenPin = 16;
const int bluePin = 17;
void setup() {
  pinMode(redPin,OUTPUT);
  pinMode(greenPin,OUTPUT);
}
```

```
pinMode(bluePin,OUTPUT);
}
void loop() {
  // Red
  analogWrite(redPin, 255);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 0);
  delay(1000);
  // Green
  analogWrite(redPin, 0);
  analogWrite(greenPin, 255);
  analogWrite(bluePin, 0);
  delay(1000);
  // Blue
  analogWrite(redPin, 0);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 255);
  delay(1000);
  // Yellow
  analogWrite(redPin, 255);
  analogWrite(greenPin, 255);
  analogWrite(bluePin, 0);
  delay(1000);
  // Purple
  analogWrite(redPin, 80);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 80);
  delay(1000);
}
```

Po úspěšném nahrání programu začne RGB LED měnit barvy, jak je znázorněno níže:



Tlačítko Lekce 4



Cíle kurzu

- Pochopení struktury a principu fungování tlačítek. Zvládnutí
- metod detekce tlačítek.
- Použití programovacích instrukcí ke čtení vstupních hodnot tlačítek



Úvod do kurzu

Ať už v elektronických systémech nebo v našem každodenním životě, „vstup“ přemostňuje komunikaci mezi lidmi a zařízeními. Nejběžnějším a klasickým vstupním prvkem je tlačítko. Může...



Vypadají nenápadně, ale jsou všudypřítomné v široké škále zařízení: od průmyslových ovládacích konzolí po chytrá nositelná zařízení, od herních ovladačů po palubní desky automobilů, a dokonce i ve vypínačích, zvoncích a dálkových ovladačích, které používáte každý den.

V této lekci se ponoříme do struktury a principů fungování tlačítka a implementujeme programování, aby deska řídicí jednotky ESP32 reagovala na vstup z tlačítka.

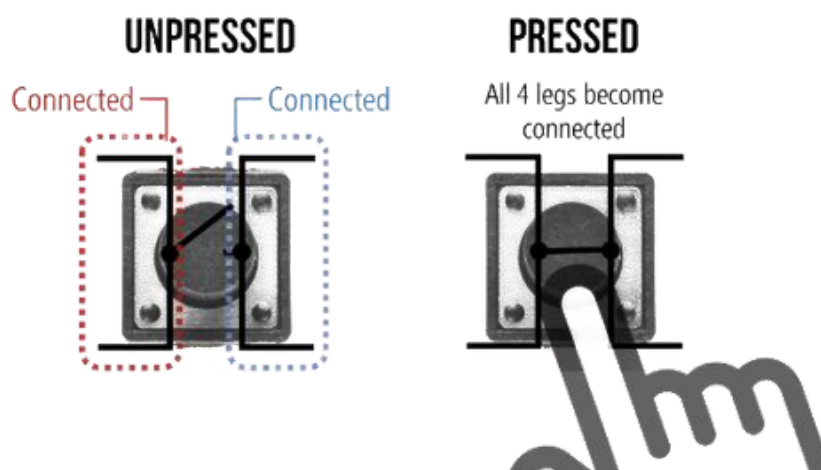


Body znalostí

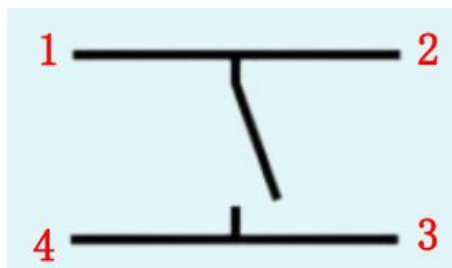
1. Tlačítko

Tlačítko je běžný mechanický spínač používaný k zapojení nebo odpojení elektrického obvodu. Obvykle se skládá z klávesy, pružiny a

vodivé kontakty. Když stiskneme krytku tlačítka, vnitřní pružina se stlačí, kontakty se sepnou a obvod se zapne; po uvolnění tlačítka se pružina resetuje, kontakty se rozpojí a obvod se vrátí do rozpojeného stavu. Tlačítka se široce používají v různých elektronických zařízeních, jako jsou mobilní telefony, dálkové ovladače televizorů, počítačové klávesnice atd., a jsou jednou z nejzákladnějších vstupních metod v interakci člověka s počítačem.



Po stisknutí tlačítka obvod tlačítka vygeneruje nízkou úroveň a po uvolnění tlačítka vygeneruje vysokou úroveň.



Otočte tlačítko, uvidíte vodorovnou čáru a dva piny rovnoběžné s vodorovnou čarou jsou vodivé. Proto by mělo být připojení obvodu tlačítka diagonální nebo dva koncové piny kolmé k vodorovné čáře.

V Arduino můžeme číst stav vysoké nebo nízké úrovně I/O pinu pomocí funkce "**digitalRead(pin)**Příkaz ;". Protože zde používáme pull-up obvod,

Data přečtená při nestisknutí tlačítka jsou 1 a data přečtená při stisknutí tlačítka jsou 0.

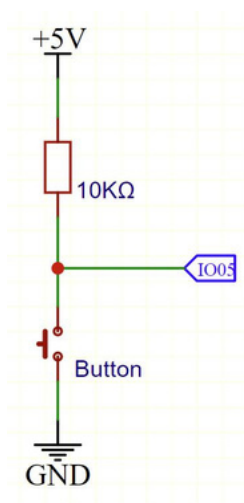
2. Externí pull-up obvod

Externí pull-up obvod je běžná metoda zapojení obvodů. Jeho funkcí je udržovat pin na řídicí desce ve stavu „high“ (výchozí hodnota je 1), když není stisknuto tlačítko nebo není připojen žádný signál, čímž se zabrání nestabilitě.

Jak funguje externí pull-up obvod?

Pokud tlačítko není stisknuto, je pin GPIO připojen k VCC přes pull-up rezistor. Protože obvod není uzavřený, neprotéká téměř žádný proud, na rezistoru nedochází k úbytku napětí, GPIO zůstává na vysoké úrovni a regulátor zobrazuje hodnotu 1.

Po stisknutí tlačítka se pin GPIO přitáhne k GND, čímž vznikne proudová cesta: VCC → pull-up rezistor → GPIO → tlačítko → GND. Protože je odpor tlačítka blízký 0, napětí na pull-up rezistoru zcela klesne, napětí na GPIO je nízké a regulátor ukazuje 0.



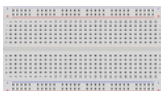


Experiment

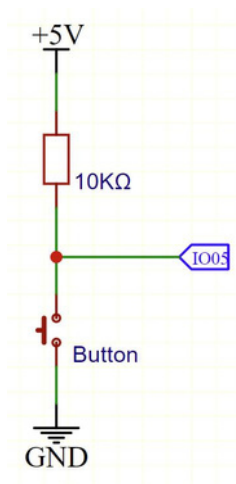
1. Popis projektu

Zobrazí vstupní hodnotu tlačítka v monitoru sériového portu.

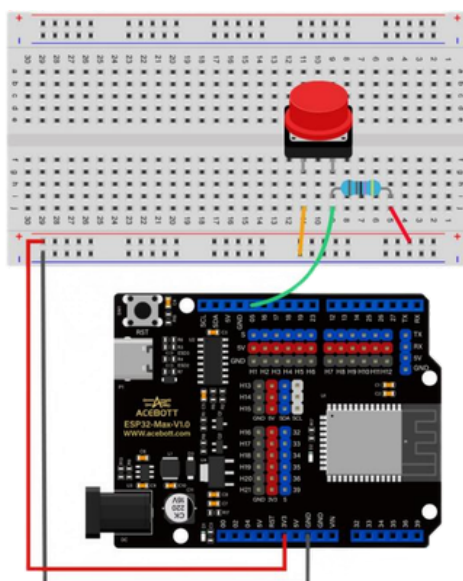
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Tlačítko	1
	Rezistor 10 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Button.ino](#)" soubor v "Angličtina\Arduino (Advanced Learner)\2.Code\lesson 4 Button\Button" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutí  nahrát program.

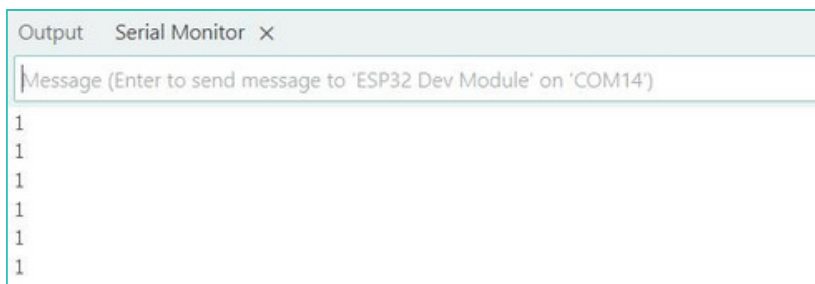
```
#define buttonPin 5

void setup() {
  Serial.begin(115200);
  pinMode(buttonPin, INPUT);
}

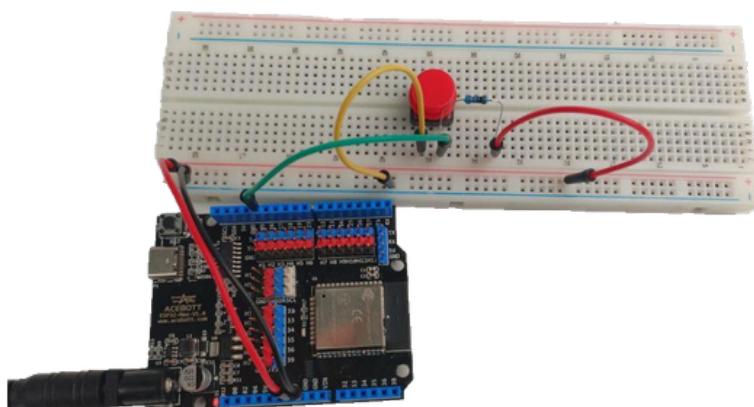
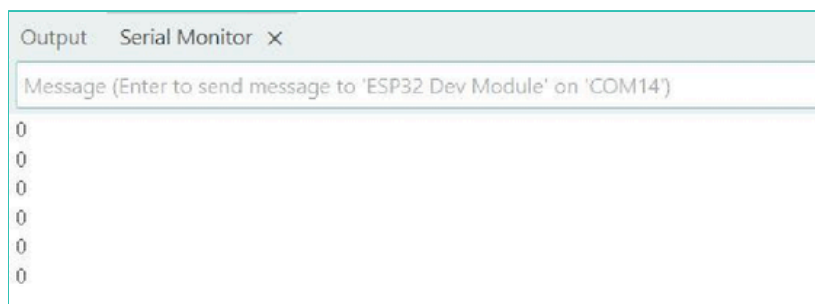
void loop() {
  int buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
}
```

Po úspěšném nahrání programu otevřete Monitor sériového portu a zobrazte stav tlačítka v reálném čase.

Tlačítko uvolněno:



Stisknuté tlačítko:



6. Popis instrukcí

digitalRead(pin): Pokud je zadaný pin nastaven na vstupní režim, lze tuto funkci použít ke čtení vstupní hodnoty zadaného pinu. Vstupní hodnota se pohybuje od 0 do 1. pin je číslo zadaného pinu.

Lekce 5 Infračervený senzor pro lidskou bytost



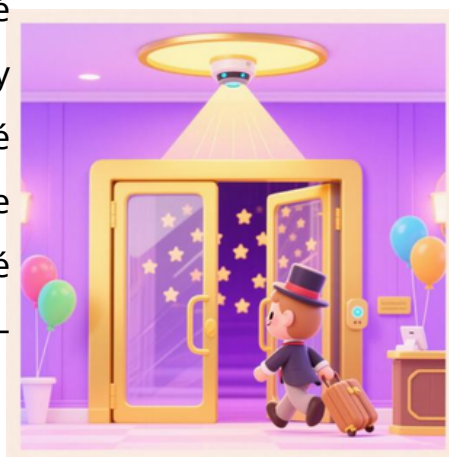
Cíle kurzu

- Pochopíte strukturu a princip fungování infračerveného senzoru pro lidský zrak
- Zvládněte metodu detekce stavu infračerveného senzoru u člověka
- Použijte programovací instrukce k načtení vstupní hodnoty infračerveného senzoru pro lidské oko



Úvod do kurzu

Všimli jste si někdy, že když vejdete na veřejné toalety, chodbu nebo výtah, světla se automaticky rozsvítí? Nebo se mnoho systémů chytré domácnosti dokáže automaticky aktivovat, když se k vám někdo přiblíží. Tyto funkce „snímání lidské přítomnosti“ se spoléhají na magický senzor – lidský infračervený senzor.



V této lekci se hlouběji ponoříme do struktury a principů fungování senzoru lidského těla a naprogramujeme ESP32 tak, aby četl stav infračerveného senzoru lidského těla.



Body znalostí

1. Infračervené záření

Infračervené záření (IR) je typ elektromagnetického záření nacházejícího se pod viditelným spektrem, s



Vlnové délky se obvykle pohybují od 0,7 mikronu do 1000 mikronů.

Infračervené záření je generováno teplem objektu; čím vyšší je teplota, tím silnější je vyzařované infračervené záření. Infračervené záření je pro lidské oko neviditelné a běžně se používá v aplikacích, jako je komunikace na dlouhé vzdálenosti, termovize, detekce objektů a měření teploty.

2. Lidský infračervený senzor

Infračervený senzor pro lidskou aktivitu (PIR senzor) je senzor speciálně navržený pro detekci lidské aktivity. Jeho princip fungování spočívá v tom, že když se osoba nebo zvíře pohybuje, vyzařuje infračervené záření. PIR senzor dokáže přesně vnímat změny v tomto infračerveném záření a určit, zda se osoba nebo zvíře pohybuje.



vstoupil do svého snímacího rozsahu. Tento senzor má vysokou praktickou hodnotu a lze jej použít v široké škále oblastí, včetně bezpečnostního monitorování a systémů inteligentní domácnosti.

Modul lidského infračerveného senzoru má dva potenciometry: jeden pro nastavení citlivosti a druhý pro nastavení detekční vzdálenosti. Aby modul lidského infračerveného senzoru fungoval lépe, je třeba jej otočit zcela proti směru hodinových ručiček.

Otáčením regulátoru zpoždění ve směru hodinových ručiček se zpoždění snímání zvyšuje, otáčením proti směru hodinových ručiček se snižuje.

Otáčením regulátoru vzdálenosti ve směru hodinových ručiček se dosah snímání zvětšuje, otáčením proti směru hodinových ručiček se snižuje.

**Zpoždění
Nastavení****Vzdálenost
Nastavení**

Poznámka: Infračervený senzor pro lidskou přítomnost detekuje změny infračerveného záření v prostředí a detekuje tak přítomnost osoby. Pokud osoba stojí, senzor nedokáže detekovat změny záření a může mylně detekovat nepřítomnost osoby.

Protože infračervený senzor lidského záření čte také digitální signály, použijte **digitalRead(pin)** instrukce pro čtení vstupní hodnoty infračerveného senzoru lidského záření, stejně jako čtení stisků kláves.

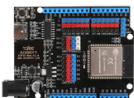


Experiment

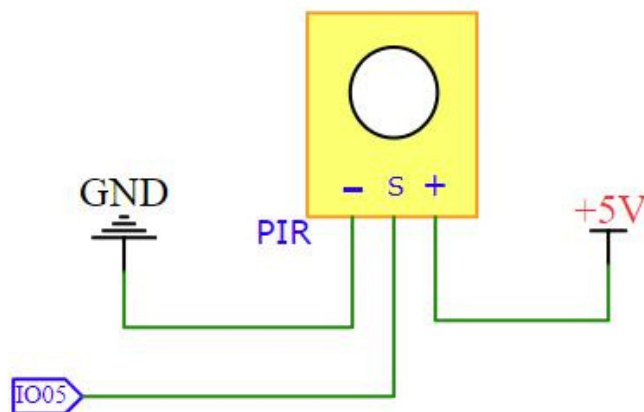
1. Popis projektu

Zobrazte vstupní hodnotu infračerveného senzoru lidského záření v reálném čase na konzoli.

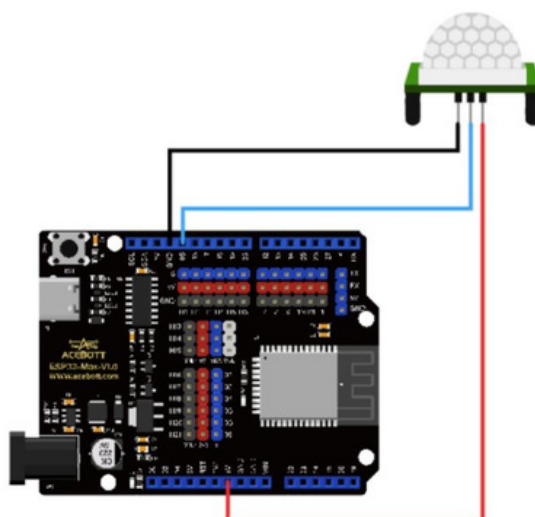
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	PIR senzor	1

3. Schéma zapojení hardwaru



4. Hardwarové připojení



5. Referenční program

Otevřete "[PIR_sensor.ino](#)" v souboru "English\Arduino(Experienced Learner)\2.Code\lesson 5 PIR sensor\PIR_sensor" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených pokynů

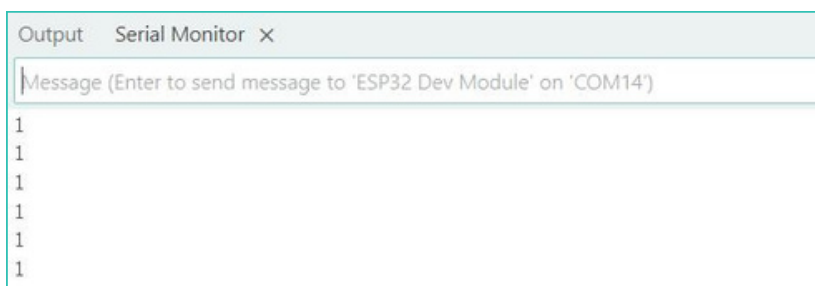
kroky, klikněte  nahrát program.

```
#define pirPin 5 // the number of the pir pin
int pirState = 0;
void setup() {
  Serial.begin(115200);
  // initialize the pir pin as an input
  pinMode(pirPin, INPUT);
}
```

```
}  
  
void loop() {  
  // read the state of the pir value  
  pirState = digitalRead(pirPin);  
  Serial.println(pirState);  
  delay(100);  
}
```

Po úspěšném nahrání programu bude sériový monitor zobrazovat hodnoty detekce PIR (pasivní infračervený) pohybového senzoru v reálném čase:

Detekce osoby:

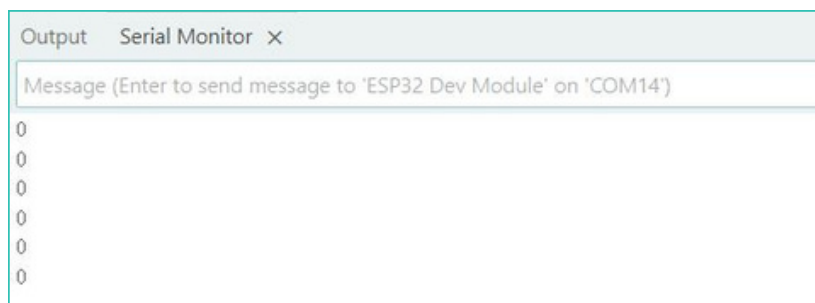


Output Serial Monitor X

Message (Enter to send message to 'ESP32 Dev Module' on 'COM14')

1
1
1
1
1
1
1
1

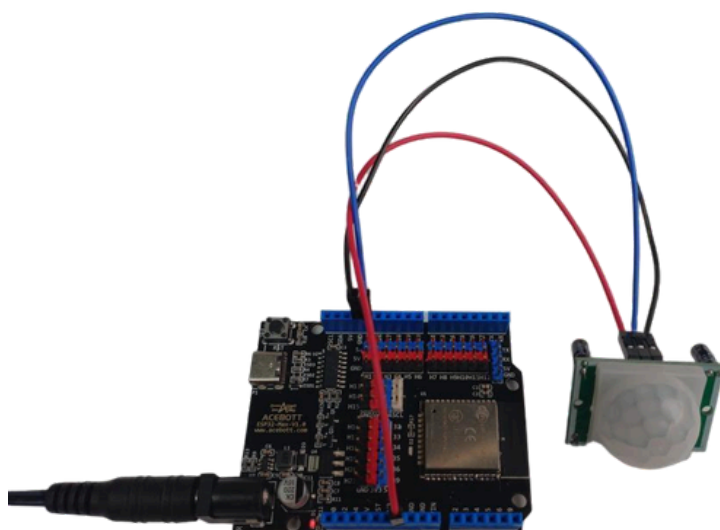
Nikoho neodhalit:



Output Serial Monitor X

Message (Enter to send message to 'ESP32 Dev Module' on 'COM14')

0
0
0
0
0
0
0



Lekce 6 Fotorezistor



Cíle kurzu

- Pochopení struktury a principu fungování fotorezistoru.
- Zvládnutí použití fotorezistoru.
- Použití programovacích instrukcí k odečtení úrovně okolního osvětlení



Úvod do kurzu

Všimli jste si někdy, že se pouliční osvětlení automaticky rozsvěcí, když se setmí, nebo že váš telefon automaticky upravuje jas obrazovky na základě okolního světla? Za těmito „chytrými“ funkcemi se skrývá drobná elektronická součástka – fotorezistor.



V této lekci získáme hlubší pochopení principu fungování fotorezistoru a pomocí programování implementujeme detekci intenzity světla v ESP32.



Body znalostí

1. Fotorezistor

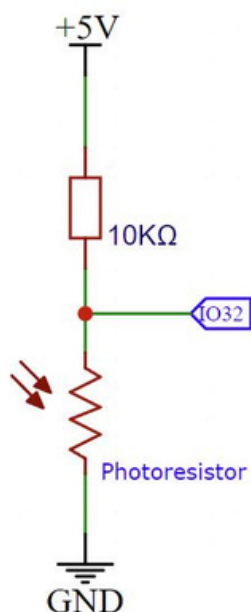
Světelný senzor, známý také jako fotorezistor, je senzor, který detekuje intenzitu okolního světla. Jeho princip fungování je založen na elektrickém odporu světlocitlivých materiálů, obvykle vyrobených z polovodičových materiálů. Odpor světelného senzoru je nízký v jasném světle a vysoký v tlumeném světle.



V této lekci použijeme fotorezistor a rezistor k vytvoření světelného senzoru. Protože intenzita světla je analogový signál, měl by být tento pin připojen k analogovému vstupnímu pinu na desce řídicí jednotky a hodnota napětí by měla být odečtena pomocí ADC (analogově-digitálního převodníku).

Princip činnosti fotocitlivého obvodu

Fotorezistor a pevný rezistor jsou zapojeny sériově pomocí pull-up obvodu a tvoří tak dělič napětí. Čím slabší světlo, tím větší hodnotu čte řídicí deska ESP32. Čím jasnější světlo, tím menší hodnotu čte řídicí deska ESP32.



Obvod fotosenzoru čte analogovou hodnotu napětí, která je

převedeno na digitální hodnotu mezi 0 a 4095 pomocí ADC. V obvodu se tato hodnota napětí mění s intenzitou světla:

Když se světlo zesílí, odpor fotorezistoru se snižuje. Protože fotorezistor a pevný rezistor jsou zapojeny sériově a tvoří obvod děliče napětí, napětí v dělicím bodě se vypočítá následovně:

$$V_{out} = V_{CC} \times \frac{R_{LDR}}{R_{fixed} + R_{LDR}}$$

Když se odpor fotorezistoru sníží, zlomek se zmenší, napětí v bodě děliče napětí klesne, a proto se napětí odečítané ADC sníží.

Podobně, když intenzita světla slábně, odpor fotorezistoru se zvyšuje, podíl se zvětšuje, napětí v bodě děliče napětí se zvyšuje a hodnota napětí naměřená ADC se zvyšuje.

Piny 2, 4, 12, 13, 14, 15, 25, 26, 27, 32, 33, 34, 35, 36 a 39 lze použít jako analogové vstupní piny. ADC desky řídicí jednotky ESP32 má 12bitové rozlišení a rozsah jeho analogových vstupních hodnot je 0 až 4095.

Poznámka: Pokud je funkce Wi-Fi povolena, modul ADC2 s ní bude sdílet hardwarové prostředky. Piny 2, 4, 12, 13, 14, 15, 25, 26 a 27 se nemusí číst nebo mohou vracet nesprávné hodnoty.

V Arduino můžeme použít **AnalogRead (pin)** příkaz pro získání hodnoty napětí zadaného analogového pinu, která je poté převedena na

digitální hodnota pomocí ADC (analogově-digitálního převodníku) s rozsahem 0 až 4095.

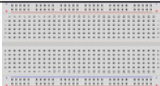


Experiment

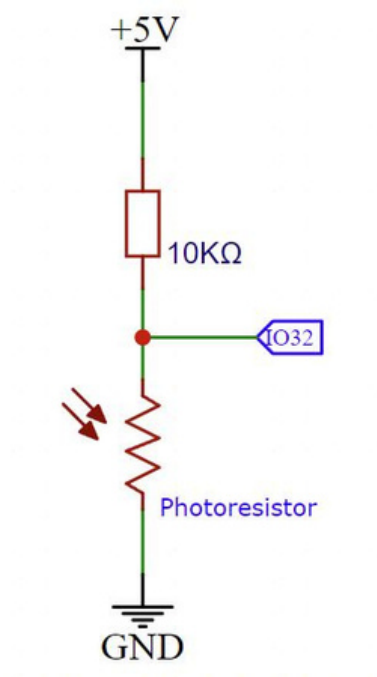
1. Popis projektu

Odečtete analogovou hodnotu fotorezistoru a zobrazte ji v reálném čase na sériovém monitoru pro detekci změn intenzity okolního světla.

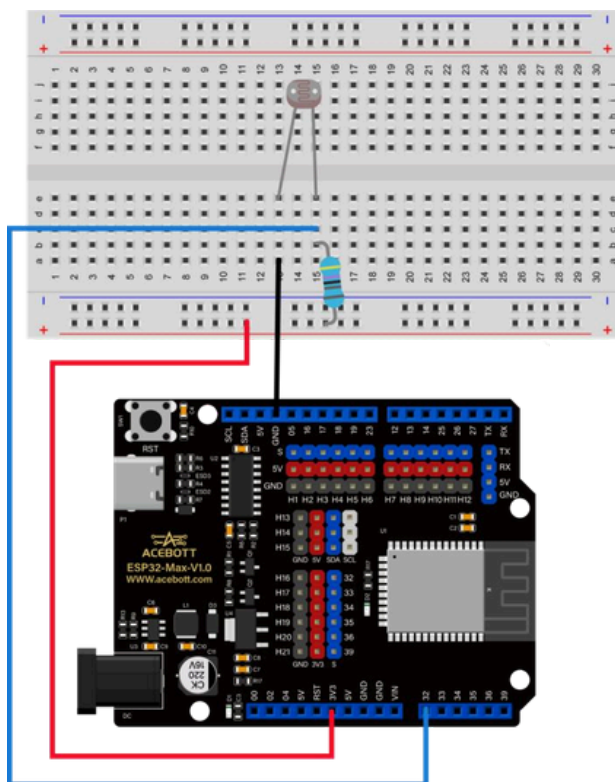
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Fotorezistor	1
	Rezistor 10 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Photoresistor.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 6 Photoresistor\Photoresistor" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

dříve naučené kroky, klikněte  nahrát program.

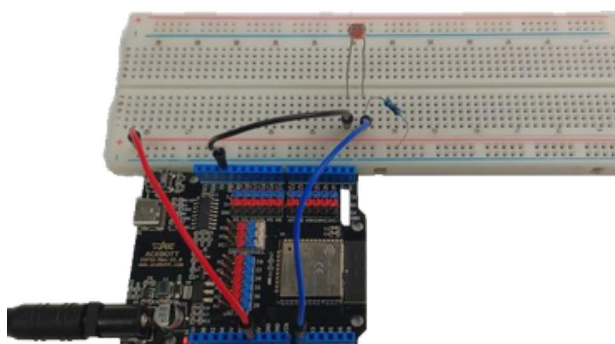
```
#define Photoresistor 32
int light_value = 0;
void setup() {
  Serial.begin(115200);
  pinMode(Photoresistor, INPUT);
}
void loop() {
  light_value = analogRead(Photoresistor);
  Serial.printf("Light sensor value = %d\n", light_value);
  delay(100);
}
```


Po úspěšném nahrání programu sériový monitor zobrazuje hodnoty fotorezistoru v reálném čase.

```
Output  Serial Monitor X
Message (Enter to send message to 'ESP32 Dev Module' on 'COM50')
Light sensor value = 640
Light sensor value = 647
Light sensor value = 647
Light sensor value = 638
Light sensor value = 645
Light sensor value = 651
```

Když fotorezistor zakryjete rukou, hodnota se zvýší.

```
Output  Serial Monitor X
Message (Enter to send message to 'ESP32 Dev Module' on 'COM53')
Light sensor value = 4095
Light sensor value = 4095
Light sensor value = 4095
Light sensor value = 4095
Light sensor value = 4095
Light sensor value = 4095
```



6. Popis instrukcí

AnalogRead (pin): Čte analogovou hodnotu ze zadaného analogového pinu. ESP32 má několik měřicích kanálů, z nichž každý obsahuje 12bitový analogově-digitální převodník, který mapuje hodnoty napětí mezi 0-5V na numerické vstupy mezi 0-4095.

Lekce 7 Potenciometr



Cíle kurzu

- Pochopení struktury a principu fungování potenciometru Zvládnutí
- používání potenciometru
- Použití programovacích instrukcí k načtení vstupní hodnoty potenciometru



Úvod do kurzu

Všimli jste si někdy knoflíků na reproduktorech nebo lampách? Ve skutečnosti používají potenciometry. Potenciometr je součástka, která upravuje napětí otáčením. Široce se používá v situacích, jako je ovládání hlasitosti a nastavení jasu.



V této lekci se naučíme, jak používat ESP32 k odečtení hodnoty potenciometru.

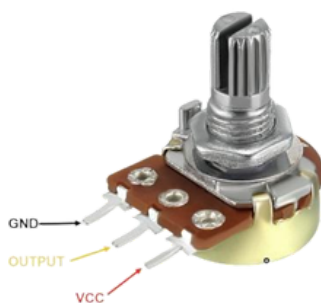


Body znalostí

1. Potenciometr

Potenciometr je třípólová nastavitelná elektronická součástka, která upravuje napětí a proud v obvodu otáčením nebo posunutím kontaktu za účelem změny hodnoty odporu.

Zapojení pinů potenciometru je následující:



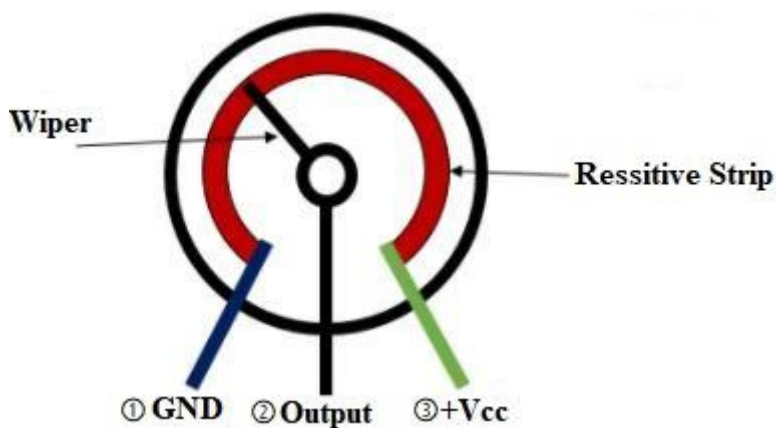
Jak funguje potenciometr?

Když otáčíme knoflíkem potenciometru, posuvný kontakt se pohybuje po rezistoru a mění odpor mezi jedním koncem a posuvným kontaktem.

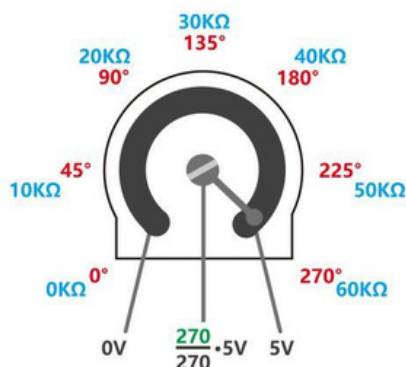
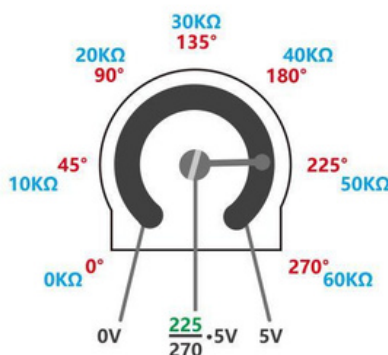
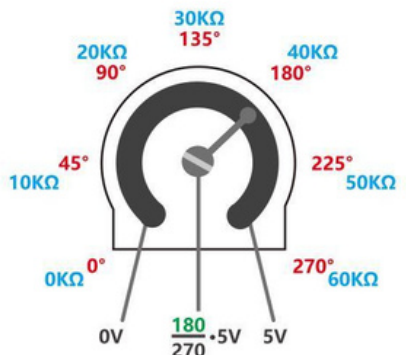
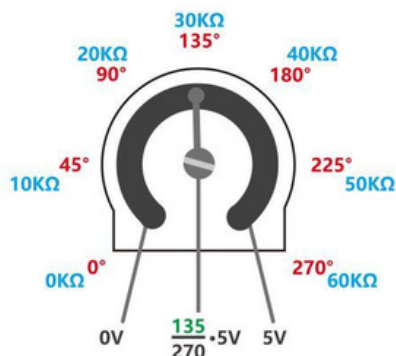
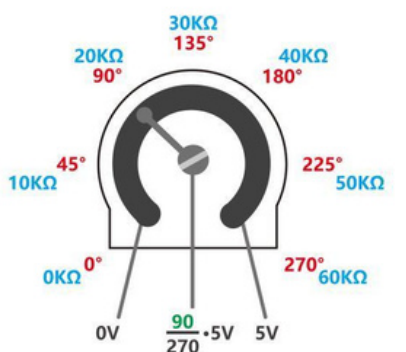
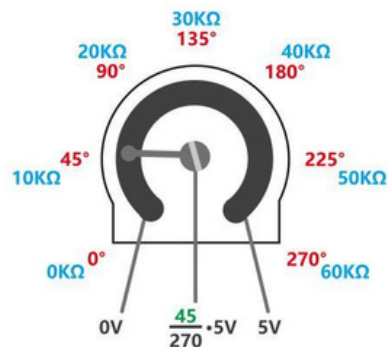
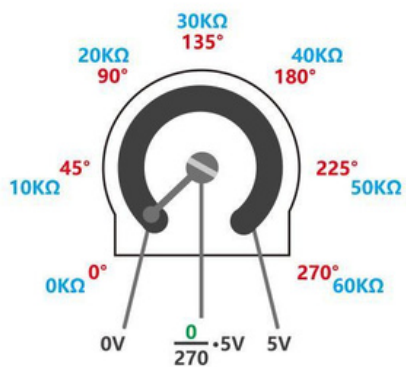
Pokud je levý konec potenciometru připojen k zápornému pólu napájení a pravý konec ke kladnému pólu napájení, napětí na prostředním pinu se bude měnit v závislosti na poloze knoflíku:

Otočit doleva Odpor klesá, napětí klesá, blíží se 0V **Otočit doprava**

Odpor se zvyšuje, napětí se zvyšuje, blíží se VCC (například 3,3 V nebo 5 V)



Toto měnící se napětí lze odečítat přes analogový vstupní port a je široce používáno v situacích, jako je ovládání hlasitosti, nastavení jasu a ovládání úhlu serva.



Potenciometr vydává analogový signál. V Arduino můžeme číst

vstupní hodnotu potenciometru pomocí "**AnalogRead (pin)**"příkaz ";", stejně jako jsme to udělali v předchozí lekci s fotorezistorem.

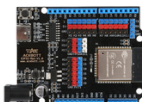
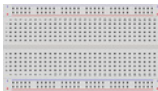


Experiment

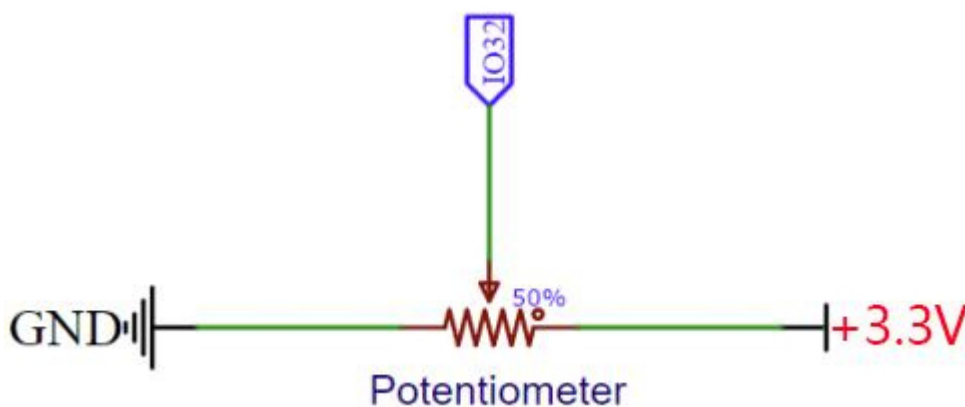
1. Popis projektu

Zobrazte vstupní hodnotu potenciometru v reálném čase pomocí sériového monitoru a sledujte, jak se mění při otáčení potenciometru.

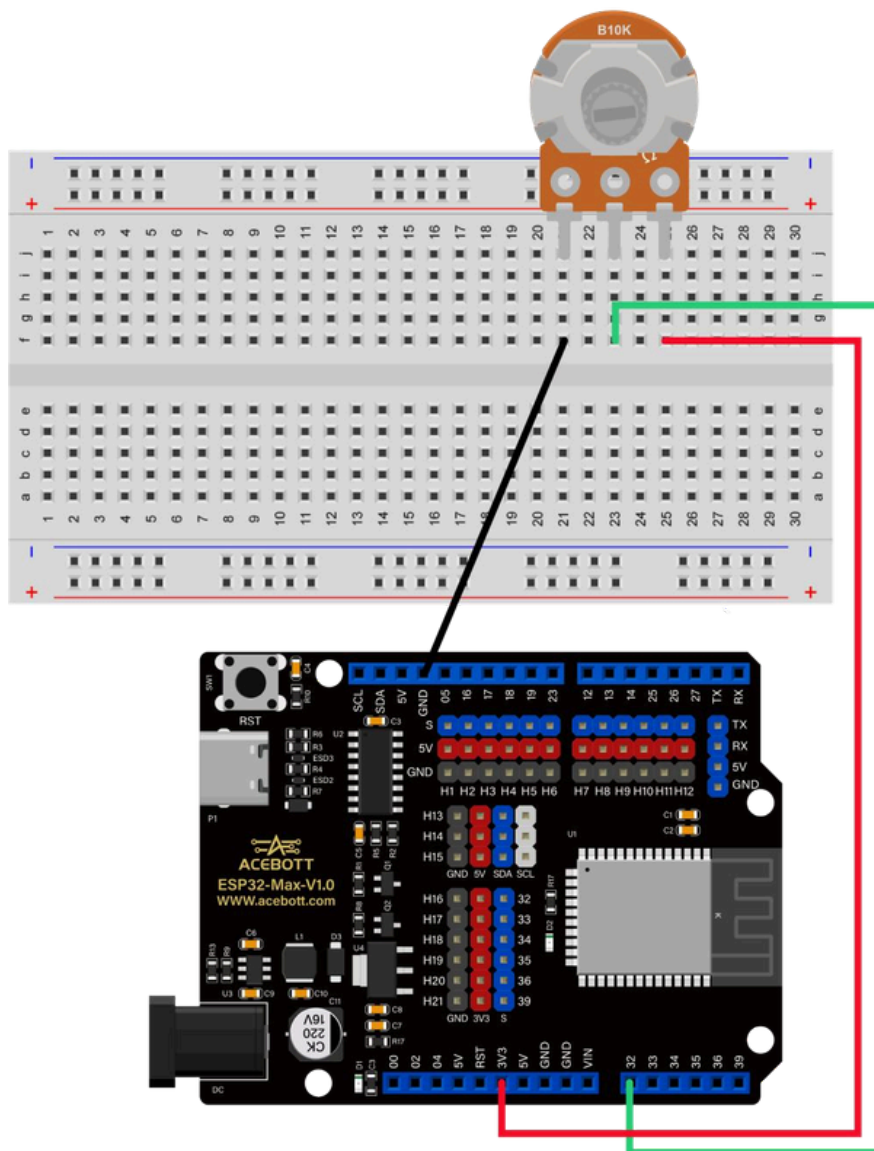
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Potenciometr	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Potenciometr.ino](#)" v souboru "English\Arduino(Experienced Learner)\2.Code\lesson7 Potenciometer\Potentiometer" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

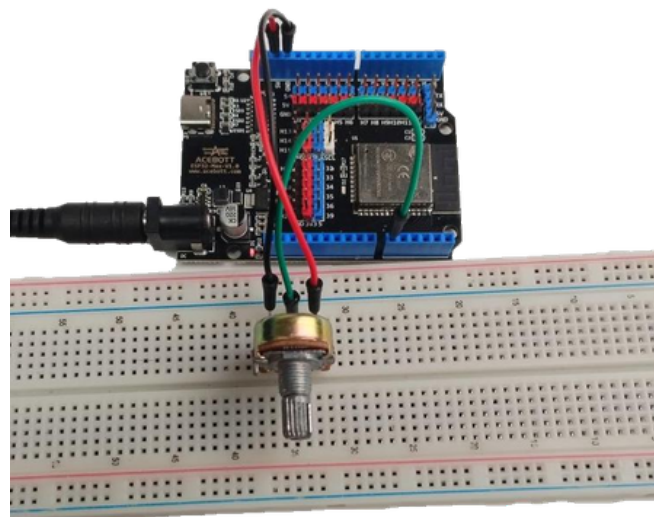
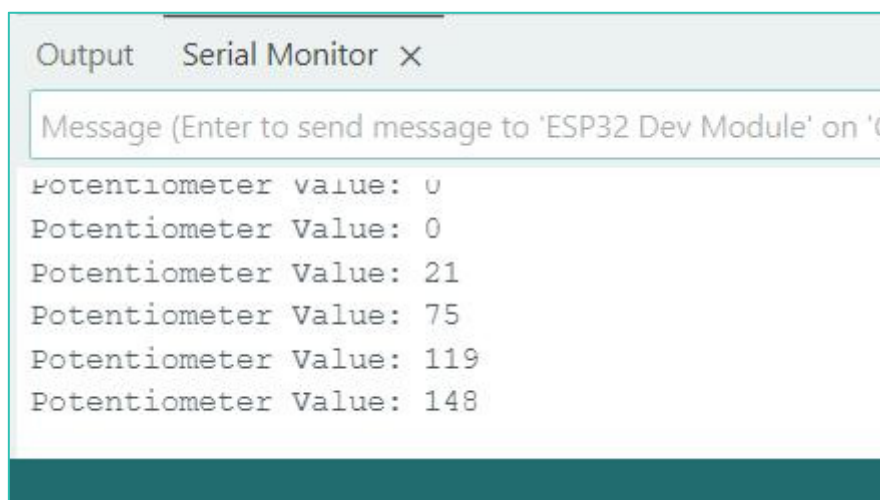
dříve naučené kroky, klikněte  nahrát program.

```
#define potPin 32
int potValue;
void setup() {
  Serial.begin(115200);
}

void loop() {
```

```
potValue = analogRead(potPin); // Read potentiometer value
Serial.print("Potentiometer Value: ");
Serial.println(potValue);
delay(100);
}
```

Po úspěšném nahrání programu bude sériový monitor zobrazovat vstupní hodnotu potenciometru v reálném čase (s 0 vlevo, hodnota se postupně zvyšuje otáčením doprava a při otočení úplně doprava dosáhne 4095).



Lekce 8 Aktivní bzučák



Cíle kurzu

- Pochopit strukturu a princip fungování aktivního bzučáku.
- Pochopit základní funkce a zesilování tranzistorů. Zvládnout použití tranzistorů k řízení aktivního bzučáku.
- Naučte se ovládat zvuk aktivního bzučáku
- Použijte programovací instrukce k ovládání bzučáku a vytvořte pípnutí



Úvod do kurzu

Bzučáky se široce používají jako zvuková výstupní zařízení v elektronických zařízeních. Od budíků a praček až po poplašné zařízení proti vloupání a zvonky u dveří se nacházejí v celé řadě běžných elektronických produktů. Dnes se dozvíme o této „mluvící elektronické součástce“ – aktivním bzučáku.



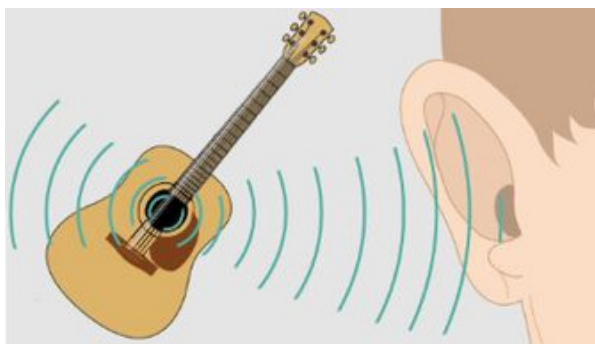
V této lekci se naučíte, jak vytvořit aktivní bzučák a jak pomocí tranzistorů zesílit jeho zvuk.



Body znalostí

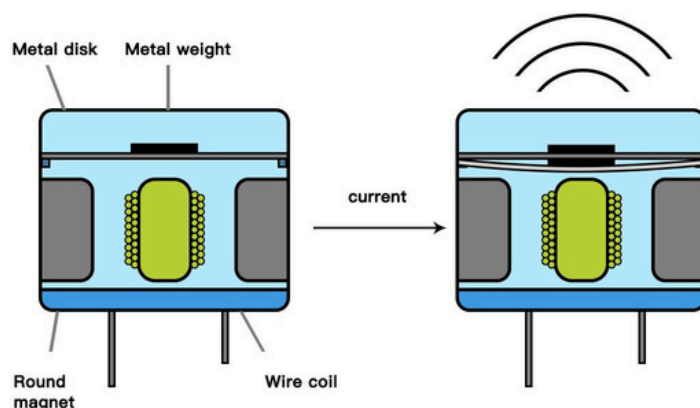
1. Jak vzniká zvuk

Zvuk je způsoben vibracemi objektu. Když objekt vibruje, vibruje i vzduch, který je s ním v těsném kontaktu, a vytváří tak zvukové vlny. Tyto zvukové vlny se šíří vzduchem a nakonec dosáhnou našich uší, což nám umožňuje vnímat zvuk.



2. Aktivní bzučák

Bzučák je elektronická součástka, která generuje zvukové signály. Bzučáky fungují na základě piezoelektrického jevu nebo elektromagnetické indukce. Piezoelektrický bzučák obsahuje piezoelektrický keramický prvek. Když je na něj aplikováno napětí, keramika vibruje a vytváří zvuk. Elektromagnetický bzučák se skládá z elektromagnetické cívky a vibrující membrány. Když cívkou prochází proud, membrána vibruje a také vytváří zvuk.



Bzučáky se dělí na aktivní a pasivní bzučáky.

Aktivní bzučák obsahuje vnitřní oscilační obvod. Když je na něj přivedeno stejnosměrné napětí, vnitřní oscilační obvod automaticky generuje obdélníkový signál, což způsobuje periodické vibrace piezoelektrické keramické nebo elektromagnetické membrány, čímž vzniká nepřetržitý zvuk. K produkci nepřetržitého zvuku vyžaduje pouze stejnosměrné napájení (například 3 V nebo 5 V) bez nutnosti externího řídicího signálu. Díky tomu se velmi snadno ovládá a je vhodný pro aplikace, jako jsou alarmy a výstrahy.

Sada obsahuje dva typy bzučáků. V této lekci použijeme aktivní bzučák. Otočte oba bzučáky. Ten s utěsněnou zadní stranou je aktivní bzučák, který budeme používat.

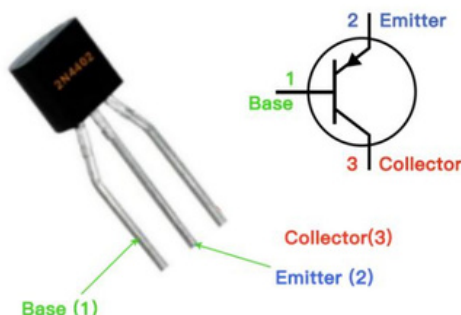


V Arduino stačí použít příkaz "**digitalWrite(BuzzerPin, HIGH);**" pro nastavení řídicího pinu na HIGH a připojení druhého pinu k GND, aby aktivní bzučák vydával zvuk.

3. Tranzistor

Tranzistor je polovodičová součástka se třemi vývody, běžně používaná pro zesilování signálu nebo řízení spínačů. Má tři piny: bázi (B), kolektor (C) a emitor (E). Zesiluje proud regulací malého množství proudu protékajícího vstupem báze, čímž reguluje větší proud protékající mezi kolektorem a emitorem. V praktických aplikacích se

tranzistory se často používají k zesílení slabých zvukových signálů a řídicích signálů, což umožňuje, aby výstupní signál byl dostatečně silný pro napájení zátěžových zařízení, jako jsou bzučáky a reproduktory. Jsou základními součástkami elektronických obvodů.

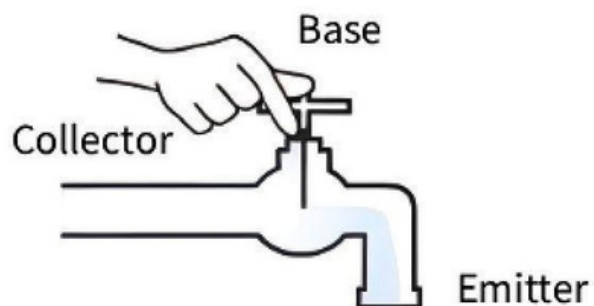


Jak funguje tranzistor

Princip fungování tranzistoru je podobný kohoutku s funkcí zapnutí/vypnutí. Tranzistor můžeme přirovnat k systému kohoutků, kde:

- Základna (B) je jako rukojeť, která otevírá a zavírá kohoutek a ovládá tak tok proudu.
- Kolektor (C) je jako vodovodní potrubí připojené k kohoutku a slouží jako vstup pro proud.
- Emitor (E) je výstup pro průtok vody a výstupní svorka pro proud.

Když bází protéká malý proud (jako při jemném otočení kohoutkem), vnitřní struktura tranzistoru se „zapne“, což umožňuje tok většího proudu z kolektoru do emitoru (stejně jako voda tekoucí z potrubí, když je otevřen kohoutek). Tím se dosáhne funkce řízení velkého proudu malým proudem, čímž se proud zesiluje.



V této lekci se tranzistor používá hlavně k zesílení proudu aktivního bzučáku, čímž se zvýší jeho hlasitost.

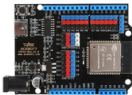
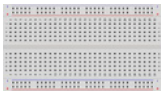


Experiment

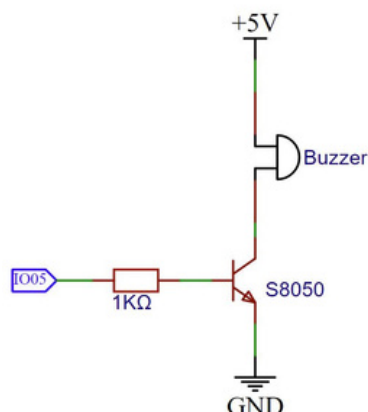
1. Popis projektu

Bzučák pípá každých 0,5 sekundy a každý tón trvá 0,5 sekundy.

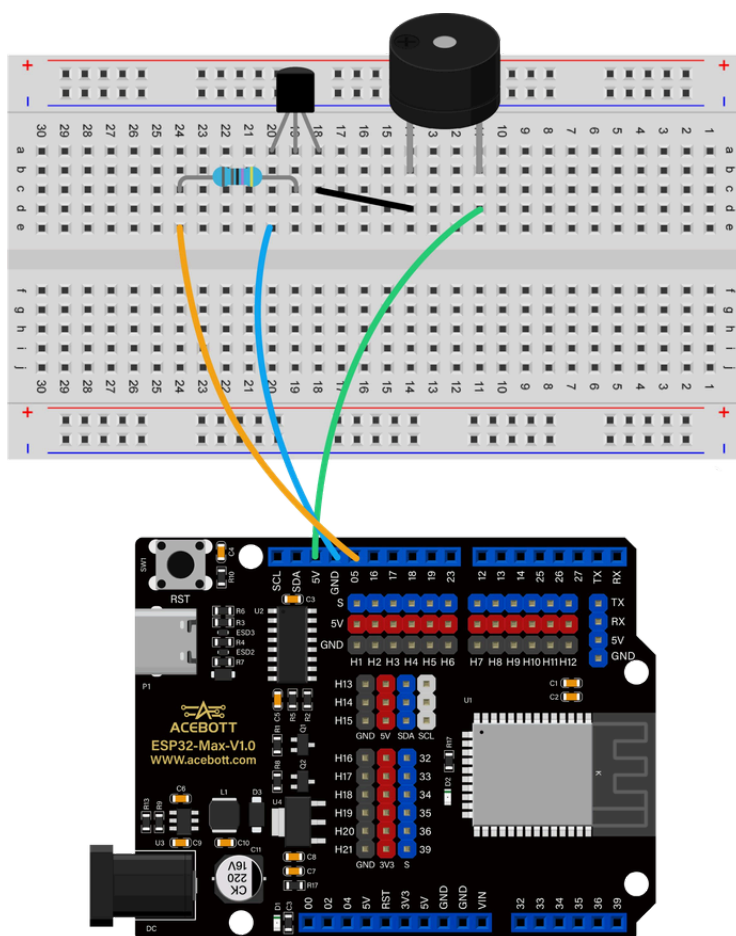
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 body	1
	Aktivní bzučák	1
	Tranzistor	1
	Rezistor 1 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Active_buzzer.ino](#)" v souboru "English\Arduino(Experienced Learner)\2.Code\lesson 8 Active buzzer\Active_buzzer\Active_buzzer.ino" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu

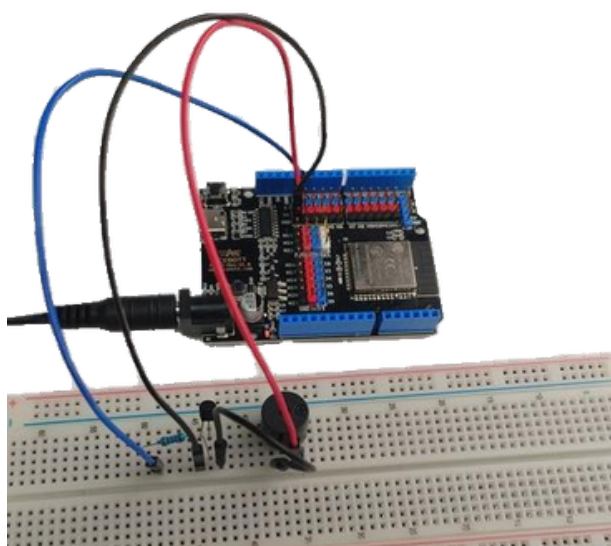
podle dříve naučených kroků klikněte



nahrát program.

```
#define buzzerPin 5 // the buzzer pin
void setup()
{
  pinMode(buzzerPin, OUTPUT); // Set as output
}
void loop()
{
  digitalWrite(buzzerPin, HIGH);
  delay(500);
  digitalWrite(buzzerPin, LOW);
  delay(500);
}
```

Po úspěšném nahrání programu pípá bzučák každých 0,5 sekundy.



Lekce 9 Pasivní bzučák



Cíle kurzu

- Pochopení fungování pasivního bzučáku
- Ovládejte pasivní bzučák pomocí PWM a vytvářejte různé tóny.
- Hraní „Twinkle Twinkle Little Star“ s pasivním bzučákem



Úvod do kurzu

Hudba je důležitou součástí našeho každodenního života. Je to víc než jen forma zábavy; je to zásadní nástroj pro emocionální propojení lidí, zprostředkování kultury, podporu sociálních vazeb a zlepšení kognitivních schopností. Její rozmanité role dávají hudbě jedinečné a nenahraditelné místo v lidské společnosti.



V této lekci se naučíte, jak pomocí kódu ovládat pasivní bzučák, který bude produkovat různé tóny a přehrávat hudbu.



Body znalostí

1. Pasivní bzučák

V předchozí lekci jsme se naučili, jak ovládat aktivní bzučák. Aktivní bzučáky však mohou vydávat pouze jeden pevný zvuk. Pokud chcete vydávat různé zvuky nebo melodie, musíte použít pasivní bzučák. Na rozdíl od aktivních bzučáků nemají pasivní bzučáky...

vnitřní oscilační obvod, a proto nemohou samy generovat zvuk. Místo toho vyžadují externí obdélníkový signál s proměnnou frekvencí, obvykle programově generovaný signál PWM (pulzně šířková modulace). Když se změní frekvence vstupního signálu, vibrační membrána uvnitř bzučáku odpovídajícím způsobem změní svou vibrační frekvenci a vytvoří různé tóny.

Sada obsahuje dva typy bzučáků. V této lekci použijeme pasivní bzučák. Otočte oba bzučáky; ten bez těsnění na zadní straně je pasivní.



Aktivní bzučák

Pasivní bzučák

2. Pole

Pole je datový typ používaný k ukládání více hodnot. Každý prvek v poli má jedinečný index, který lze použít pro přístup k prvkům v poli.

Poznámka: Všechny hodnoty uložené v poli musí být stejného datového typu.

Syntaxe pro definování pole je následující:

```
type arrayName[arraySize];
```


Kde type je datový typ prvků v poli, arrayName je název pole a arraySize je velikost pole. Níže je uveden příklad:

```
int cislo[5];
```

Výše uvedený kód definuje pole obsahující 5 celých čísel a k těmto prvkům můžete přistupovat pomocí num[0] až num[4].

Pole lze inicializovat při jejich definování. Například:

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

K prvkům pole můžete přistupovat pomocí indexu (poznámka: indexování začíná od 0). Například:

```
int cislo[5] = {1, 2, 3, 4, 5};  
Serial.println(cislo[2]);
```

Pole se často používají společně s příkazy for k iteraci všech prvků v poli. Například:

```
int cislo[5] = {1, 2, 3, 4, 5};  
for(int i = 0; i < 5; i++) {  
    Serial.println(num[i]);  
}
```

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

Níže je uveden kompletní program, který inicializuje pole a vypíše každý prvek pole přes sériový port.

```
int cislo[5] = {1, 2, 3, 4, 5};  
void setup() {  
    for(int i = 0; i < 5; i++) {  
        Serial.println(num[i]);  
    }  
}  
void loop() {  
}
```

V programu pro tuto lekci budeme používat dvě pole: notes a beats, která slouží k ukládání hudebních not melodie a jim odpovídajících dob. Pole notes obsahuje notové kombinace, které tvoří melodii v pořadí; pole beats odpovídá jedna ku jedné poli notes a udává počet dob pro každou notu, tj. její trvání.

Tímto způsobem můžeme jasně oddělit a spravovat melodii a rytmus, což usnadňuje a zpřehledňuje strukturu kódu. Zároveň nám to usnadňuje úpravu melodie nebo úpravu rytmu, a tím vytváření hudebních děl různých stylů. Později v programu zkombinujeme funkce `tone()` a `delay()`, abychom na základě dat z těchto dvou polí přehráli celou melodii v pořadí.

```
//Store each note of "Twinkle Twinkle Little Star"  
int notes[]={C4,C4,G4,G4,A4,A4,G4,  
| | | | | F4,F4,E4,E4,D4,D4,C4};  
//Store the beats of "Twinkle Twinkle Little Star"  
float beats[]={1,1,1,1,1,1,2,  
| | | | | 1,1,1,1,1,1,2};
```

3. Přehrávání hudby

Při použití pasivního bzučáku k přehrávání melodie „Twinkle Twinkle Little Star“ potřebujeme znát hudební zápis a odpovídající frekvenci každé noty.

Běžné hudební noty a jejich odpovídající frekvence (jednotka: Hz):

■ Basový rozsah ■ Střední rozsah ■ Výškový rozsah

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

Noty pro „Twinkle, Twinkle, Little Star“:

Twinkle Twinkle Little Star

♩ = 100
=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?

Noty a rytmy v hudbě:

(1) Normální nota, jako například první nota 1, zabírá jednu dobu.

(2) Podtržená nota představuje 0,5 doby.

(3) Nota s tečkou za sebou představuje 0,5 doby navíc, tj. jedna nota má 1 + 0,5 = 1,5 doby.

(4) Pomlčka za notou označuje další dobu, což znamená, že nota trvá 1 + 1 = 2 doby.

Typ poznámky	Počet úderů	Symbol
Normální nota		
Podtržená poznámka	1 doba	1
	0,5 taktu	<u>1</u>
	1,5 taktu	1.
Tečkovaná poznámka		
Poznámka s -	2 doby	1-

Doba je doba mezi notami. Prodleva mezi dobami se liší píseň od písňě. V hudebním zápisu vidíme ♩ = 100, což značí 100 úderů za minutu. Víme tedy, že délka doby je 60 000 (100 = 600 ms).

V Arduino můžeme použít **delay(beats[i] * 600)**; pro ovládání délky přehrávání každé noty. Zde beats[i] představuje hodnotu doby aktuální noty a 600 je koeficient multiplikátoru doby trvání doby, což znamená, že každá doba trvá 600 milisekund. Úpravou tohoto koeficientu můžete flexibilně nastavit délku každé doby, a tím ovládat celkové tempo hudební skladby.

Při hraní noty můžete použít **tone(pin, frekvence)**; příkaz pro výstup PWM obdélníkového signálu odpovídající frekvence na zadaném pinu, čímž se vytvoří požadovaná výška tónu. Po dohrání každé noty použijte **noTone(pin)**; příkaz pro zastavení zvukového výstupu, zajištění jasné mezery mezi notami a zesílení rytmického pocitu z melodie.

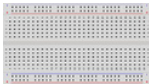


Experiment

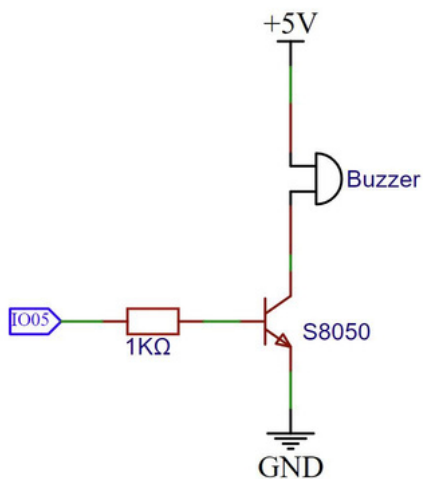
1. Popis projektu

Ovládejte tón pasivního bzučáku a nechte ho hrát „Twinkle Twinkle Little Star“.

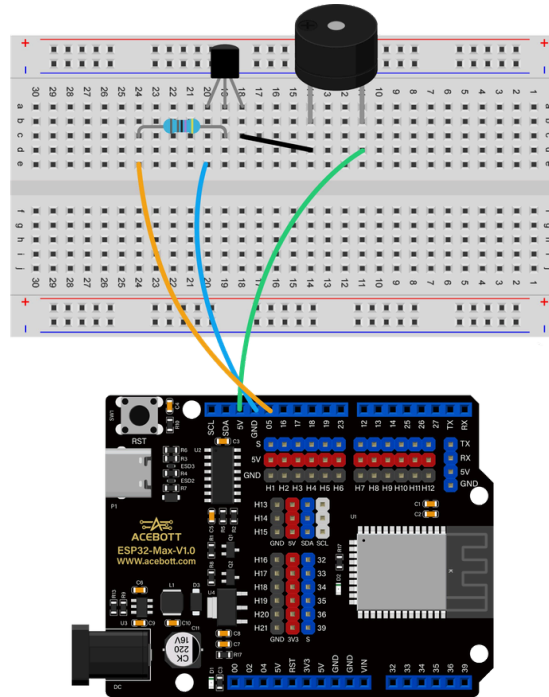
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Pasivní bzučák	1
	Tranzistor	1
	Rezistor 1 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Passive_buzzer.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 9 Passive buzzer\Passive_buzzer" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

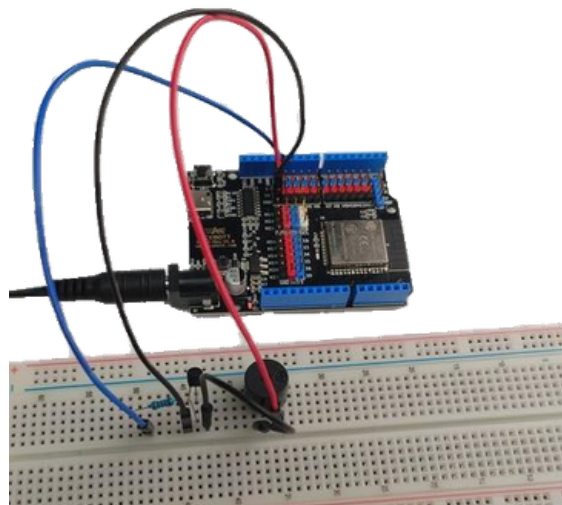
dříve naučené kroky, klikněte  nahrát program.

```
//Define the frequency of each note
#define C4 523
#define D4 587
#define E4 659
#define F4 698
#define G4 784
#define A4 880
#define B4 988
#define buzzer 5 //Declare the buzzer pin

//Store each note of "Twinkle Twinkle Little Star"
int notes[]={C4,C4,G4,G4,A4,A4,G4,
             F4,F4,E4,E4,D4,D4,C4};
//Store the beats of "Twinkle Twinkle Little Star"
float beats[]={1,1,1,1,1,1,2,
               1,1,1,1,1,1,2};
int length = sizeof(notes)/sizeof(notes[0]); //Count the number of notes
```

```
void setup() {  
  // put your setup code here, to run once:  
  pinMode(buzzer,OUTPUT); //Set buzzer as output mode  
  Serial.begin(115200);  
}  
void loop() {  
  // put your main code here, to run repeatedly:  
  for(int i=0;i<length;i++){  
    tone(buzzer,notes[i]); //Play the note  
    //Each beat is 600 ms, and each note delays for a duration corresponding  
    //to its beat count.  
    delay(beats[i]*600);  
    noTone(buzzer); //Stop playing the note  
  }  
  delay(2000); //Loop playback with a 2-second delay  
}
```

Po úspěšném nahrání programu se ozve bzučák s názvem „Twinkle Twinkle Little Star“.



6. Popis instrukcí

tone (pin, frekvence): Řídí zadaný pin pro výstup obdélníkového signálu s určenou frekvencí a spravuje dobu trvání výstupu obdélníkového signálu. pin je zadané číslo pinu a frekvence je frekvence obdélníkového signálu PWM.

noTone(pin): Vypne obdélníkový výstup na zadaném pinu. Používá se v programech pro ovládání bzučáku k zastavení zvuku bzučáku.

Lekce 10 Stejnosměrný motor



Cíle kurzu

- Pochopte strukturu a princip činnosti stejnosměrného motoru
- Zvládněte metody řízení stejnosměrného motoru pomocí modulu ovladače motoru
- Použijte programovací instrukce k řízení směru otáčení a rychlosti stejnosměrného motoru



Úvod do kurzu

Stejnosměrné motory lze nalézt v mnoha chytrých zařízeních, jako jsou automobily, ventilátory a elektrické dveře. Stejnosměrný motor řídí směr a rychlost otáčení podle směru a velikosti proudu.



V této lekci budeme simulovat provoz těchto zařízení a naučíme se, jak pomocí programových instrukcí ovládat stejnosměrný motor, což způsobí jeho otáčení vpřed, vzad, zrychlení nebo zpomalení.



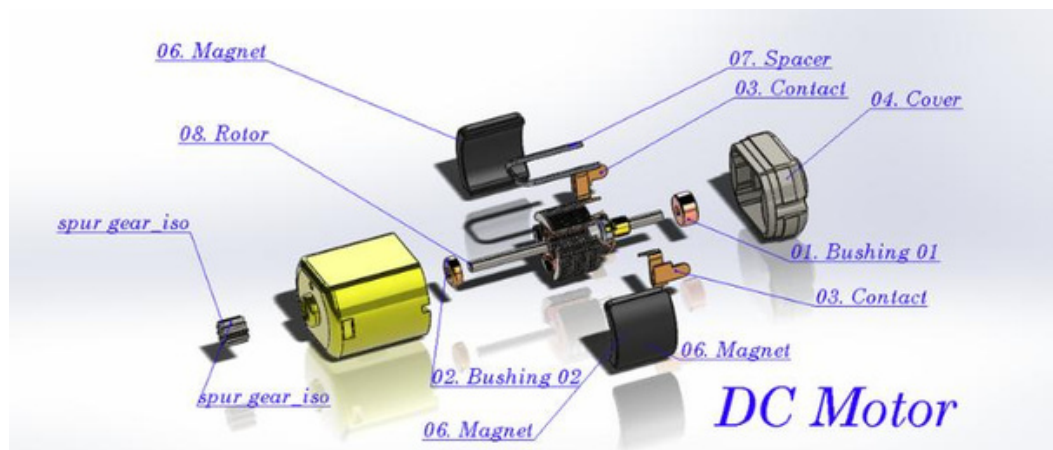
Body znalostí

1. Stejnosměrný motor

Stejnosměrný motor je zařízení, které přeměňuje elektrickou energii na mechanickou a je široce používáno v různých aplikacích vyžadujících řízený rotační pohyb. Dokáže přesně řídit svou rychlost a směr nastavením vstupního proudu nebo napětí. Je vhodný pro aplikace, jako jsou ventilátory, elektrické nářadí, elektrická vozidla, roboty, malé spotřebiče a automatizační zařízení.



Stejnosměrný motor se skládá ze dvou částí: statoru a rotoru. Stator je stacionární část motoru, obvykle sestávající z vinutí (cívek) a železného jádra. Když vinutími protéká proud, generované magnetické pole se nazývá pevné magnetické pole. Rotor je pohyblivá část motoru, obvykle vyrobená z vodičů. Když rotorem protéká proud, je vystaven síle v pevném magnetickém poli, která generuje mechanický pohyb, který způsobuje otáčení rotoru.



Stejnoseměrný motor má dva piny, jeden připojený ke kladnému pólu zdroje napájení a druhý k zápornému pólu zdroje napájení. Tím se motor rozběhne. Pokud jsou kladný a záporný pól zapojeny opačně, motor se bude pohybovat v opačném směru. Řízením proudu stejnosměrného motoru lze řídit jeho otáčky.

Poznámka: Motor vyžaduje k provozu určité napětí a proud. Protože výstupní proud z IO portu řídicí desky je extrémně malý a nemůže pohánět stejnosměrný motor, nelze piny stejnosměrného motoru přímo připojit k IO portu řídicí desky. Pro řízení provozu stejnosměrného motoru prostřednictvím IO portu řídicí desky je vyžadován modul ovladače motoru.

2. Modul ovladače motoru

Modul ovladače motoru je elektronický modul používaný k řízení provozu motoru. Jeho hlavní funkcí je převod řídicího signálu na proud a napětí potřebné k pohonu motoru. Funguje jako můstek mezi řídicí deskou ESP32 a stejnosměrným motorem, což umožňuje mikrokontroléru bezpečně a efektivně řídit směr a rychlost motoru.

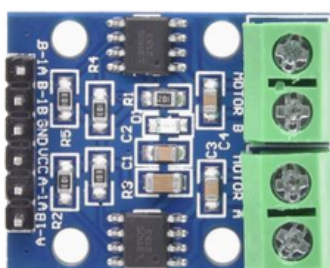
Funkce modulu ovladače motoru:

(1) Zajistěte dostatečný proud a napětíVýstupní proud z IO portu řídicí desky je omezený a nemůže přímo řídit motor. Modul ovladače může poskytnout požadovaný proud a napětí pro motor.

(2) Realizace otáčení motoru vpřed a vzadŘízením vstupního signálu modulu pohonu motoru lze snadno dosáhnout otáčení motoru vpřed a vzad.

(3) Funkce regulace rychlosti: Modul ovladače podporuje vstupní signál PWM a řídí otáčky motoru úpravou pracovního cyklu PWM.

(4) Ochranný obvodVětšina modulů budičů má vestavěnou ochranu proti nadproudu a přehřátí, která zajišťuje bezpečnost motoru a řídicího obvodu.



Směr a rychlost motoru jsou určeny směrem a velikostí elektrického proudu. V Arduino lze k řízení směru a rychlosti otáčení motoru použít příkaz `analogWrite(pin, value);`. Zde `pin` označuje řídicí pin motoru a `value` je výstupní hodnota PWM (pulzně šířková modulace) – čím větší je hodnota, tím vyšší je rychlost motoru.

Příklad kódu pro otáčení motoru vpřed:

```
analogWrite(motor1A,hodnota);  
analogWrite(motor2A,0);
```

Když se motor otáčí v opačném směru, příklad kódu vypadá takto:

```
analogWrite(motor1A,0);
analogWrite(motor2A,hodnota);
```



Experimentovat

Experiment 1: Řízení otáčení stejnosměrného motoru plnou rychlostí vpřed a vzad

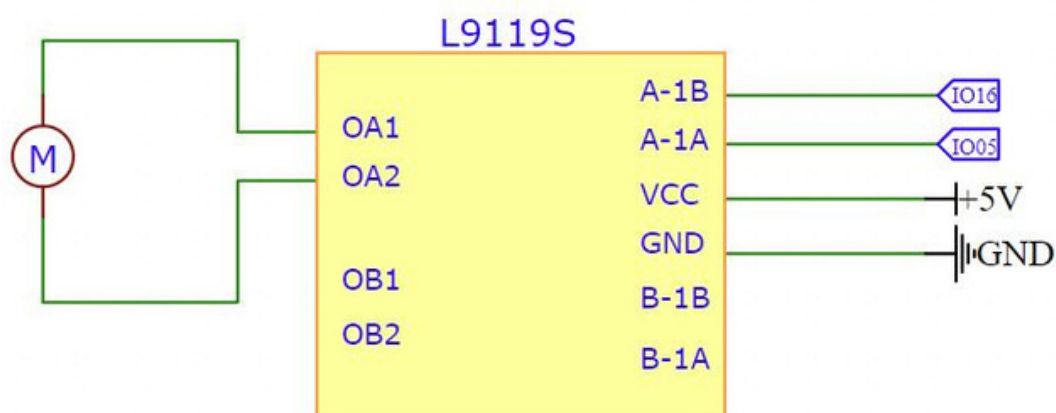
1. Popis projektu

Ovládejte stejnosměrný motor tak, aby se otáčel dopředu po dobu 2 sekund a poté dozadu po dobu 2 sekund.

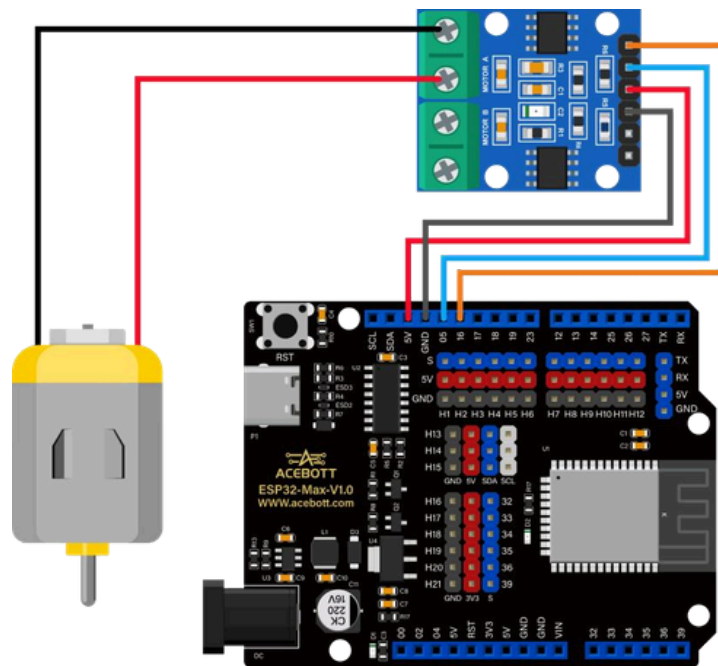
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Ovladač stejnosměrného motoru	1
	Stejnoseměrný motor	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[10_1_motor.ino](#)" soubor v "English\Arduino(Experienced Learner) \2.Code\lesson 10 motor\10_1_motor" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků klikněte na



nahrát program.

```
#define motor1A 5
#define motor2A 16

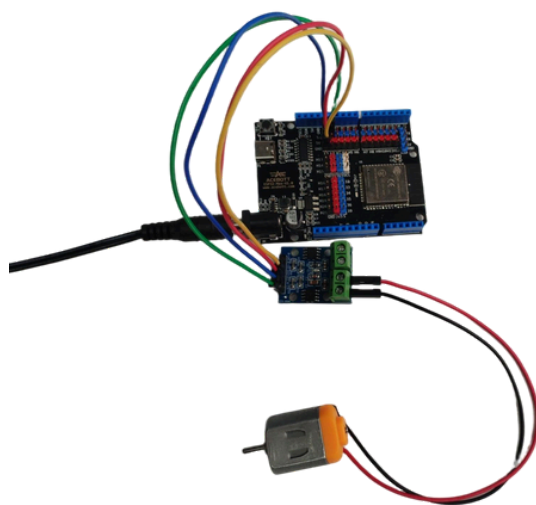
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.
  pinMode(motor1A, OUTPUT);
  pinMode(motor2A, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  // Rotate
  analogWrite(motor1A, 255);
  analogWrite(motor2A, 0);
  delay(2000);

  // Rotate in the opposite direction
  analogWrite(motor1A, 0);
  analogWrite(motor2A, 255);
}
```

```
delay(2000);  
  
// Stop  
analogWrite(motor1A, 0);  
analogWrite(motor2A, 0);  
delay(2000);  
}
```

Po úspěšném nahrání programu se motor opakovaně otáčí dopředu po dobu 2 sekund, dozadu po dobu 2 sekund a zastaví se na 2 sekundy.



Experiment 2: Řízení rychlosti stejnosměrného motoru

1. Popis projektu

Ovládejte stejnosměrný motor pomocí PWM, abyste postupně zvyšovali a snižovali jeho otáčky a opakovali cyklus.

2. Referenční program

Otevřete "[10_2_motor.ino](#)" soubor v adresáři "English\Arduino(Experienced Learner)\2.Code\lesson 10 motor\10_2_motor" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutím  nahrát program.

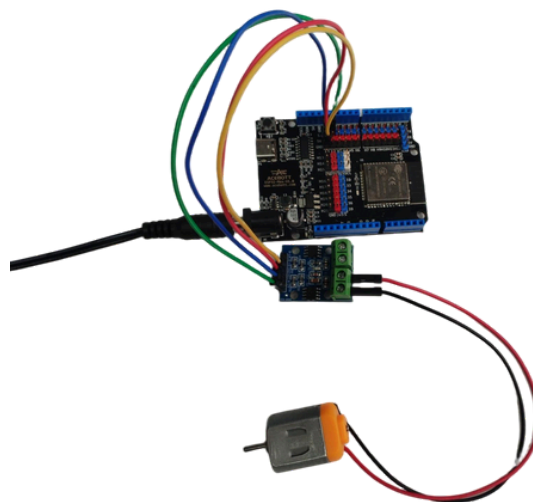
```
#define motor1A 5
#define motor2A 16
int speed;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.
  Serial.begin(115200);
  pinMode(motor1A, OUTPUT);
  pinMode(motor2A, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  for(speed=0;speed<=255;speed++){
    Serial.println(speed);
    analogWrite(motor1A,speed);
    analogWrite(motor2A,0);
    delay(50);
  }

  for(speed=255;speed>=0;speed--){
    Serial.println(speed);
    analogWrite(motor1A,speed);
    analogWrite(motor2A,0);
    delay(50);
  }
}
```

Po úspěšném nahrání programu se otáčky motoru pomalu zvyšují a poté pomalu snižují.



Lekce 11 Servo



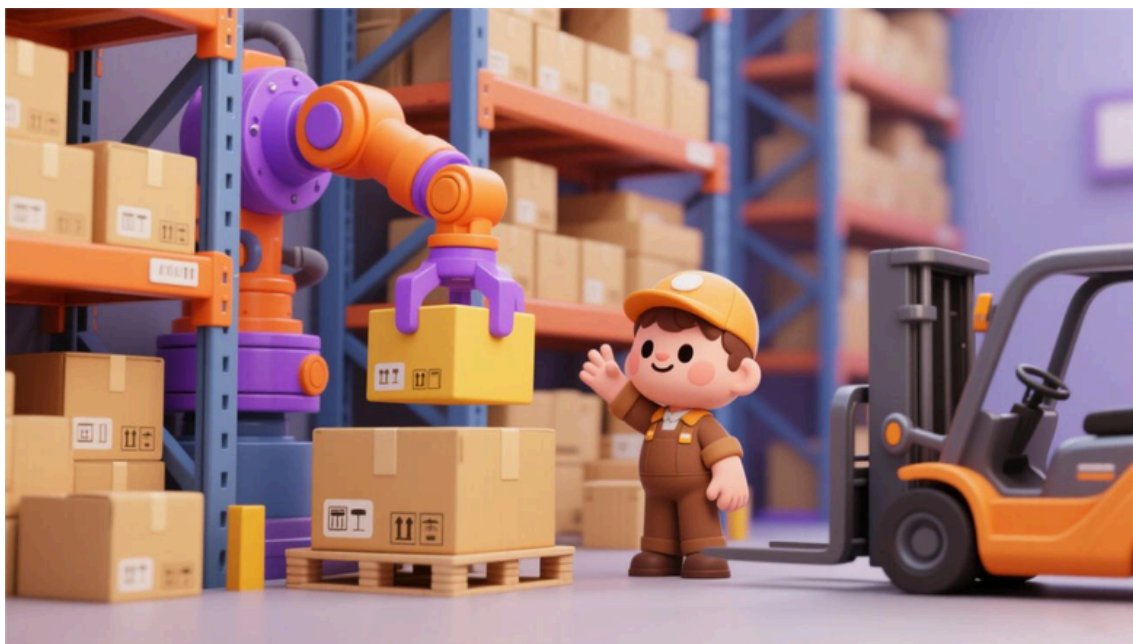
Cíle kurzu

- Pochopení struktury a základních principů servomotorů. Zvládnutí
- používání knihovny servomotorů.
- Použití programovacích instrukcí k řízení rotace serva



Úvod do kurzu

V každodenním životě se setkáváme s mnoha zdánlivě jednoduchými, ale zároveň technologicky vyspělými zařízeními, jako jsou automatické dveře, kamery, robotická ramena a dokonce i chytré zametací roboty. Jejich přesné „otáčení“, „vlny“ a „nastavení“ se spoléhají na serva.



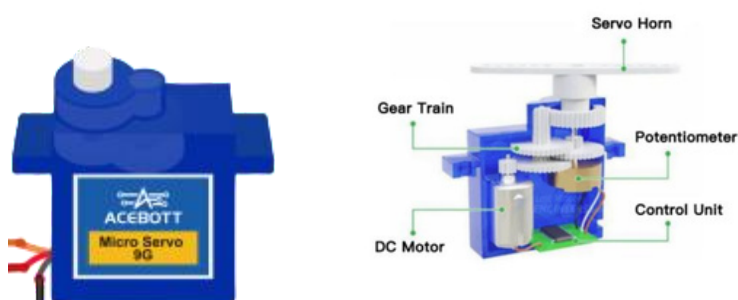
V této lekci se naučíme, jak používat řídicí desku ESP32 k řízení přesné rotace serva.



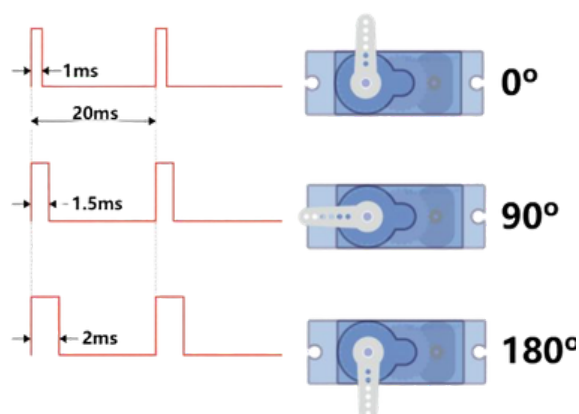
Body znalostí

1. Princip činnosti serva

Servo je zařízení, které přeměňuje elektrickou energii na mechanickou energii, což umožňuje přesné ovládání úhlu. Obvykle se skládá z pouzdra, soukolí s proměnnou rychlostí, motoru, nastavitelného potenciometru, elektronické řídicí desky a volantu.



Jeho princip fungování spočívá v tom, že elektronická řídicí deska přijímá řídicí signál ze zdroje signálu a pohání motor k otáčení. Na základě signálu PWM (modulace šířky impulsů) se šířka impulsu upravuje tak, aby se servo natočilo o zadaný úhel. Servo konkrétně nepřetržitě přijímá řídicí signály s určitou frekvencí (obvykle 50 Hz, nebo jeden cyklus každých 20 milisekund). V každém cyklu délka trvání vysokofrekvenčního impulsu (šířka impulsu) určuje polohu natočení serva: čím delší je šířka impulsu, tím větší je úhel natočení serva; čím kratší je šířka impulsu, tím menší je úhel natočení.



Scénáře aplikací Roboti, modely letadel, automatizovaná výroba

linky a další aplikace vyžadující přesné řízení polohy nebo pohybu.

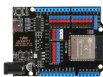


Experiment

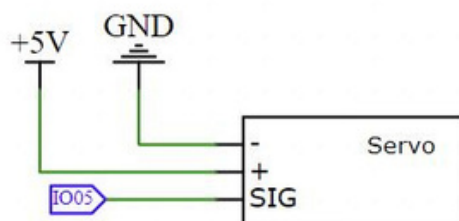
1. Popis projektu

Řídí vratný pohyb serva od 0° do 180°.

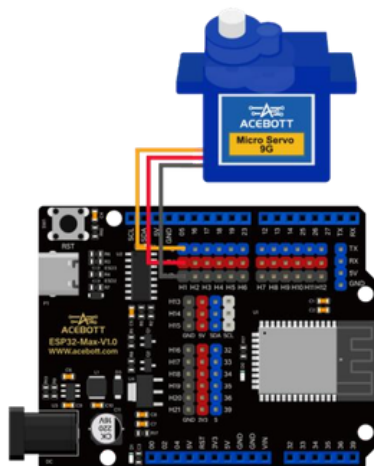
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Servo	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Servo.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 11 Servo\Servo" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků klikněte na



nahrát program.

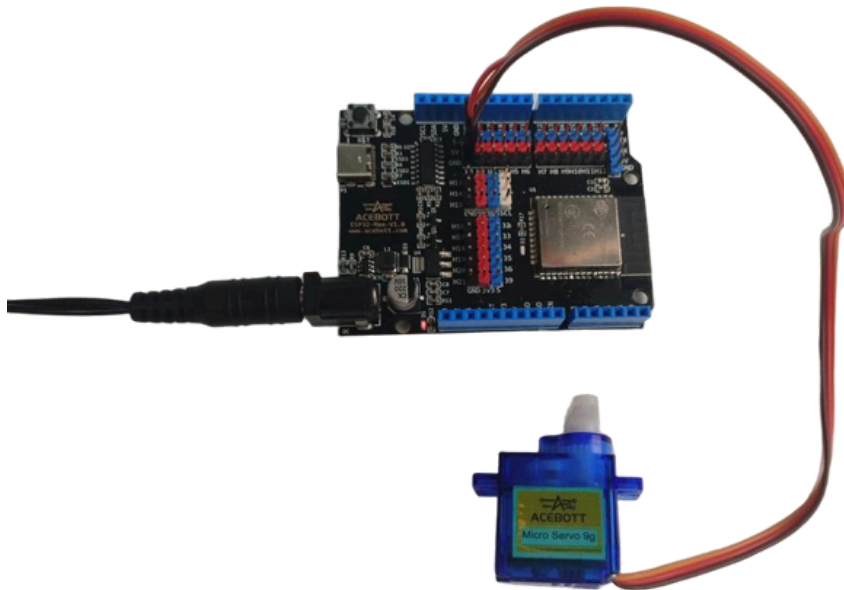
```
#include <ESP32Servo.h>
Servo myServo;
#define servoPin 5 // Servo signal connected to GPIO 5

void setup() {
  // Attach the servo to the pin with pulse width range (500–2400 µs)
  myServo.attach(servoPin, 500, 2400);
  myServo.write(0); // Initialize servo to 0 degrees
  delay(500); // Small delay for stabilization
}

void loop() {
  // Move servo from 0° to 180°
  for (int angle = 0; angle <= 180; angle++) {
    myServo.write(angle); // Set servo angle
    delay(15); // Wait for smooth movement
  }

  // Move servo from 180° back to 0°
  for (int angle = 180; angle >= 0; angle--) {
    myServo.write(angle); // Set servo angle
    delay(15); // Wait for smooth movement
  }
}
```

Po úspěšném nahrání programu se servo otáčí tam a zpět mezi 0 a 180 stupni.



6. Popis instrukcí

include<ESP32Servo.h> Importujte knihovnu ESP32Servo pro ovládání serv na ESP32, což umožňuje nastavení úhlu a řízení polohy.

Servo myServo Vytvořte objekt serva s názvem myServo pro ovládání serva. Tento objekt umožňuje volání souvisejících funkcí (jako například attach(), write() atd.) pro nastavení pinu serva a úhlu natočení.

attach(pin) Funkce používaná k navázání serva na určený GPIO pin.

write(úhel): Otočení serva do zadané úhlové polohy, kde se úhel obvykle pohybuje od 0 do 180 stupňů. Například myServo.write(90) otočí servo do střední polohy (90 stupňů).

Lekce 12 Snímač teploty a vlhkosti



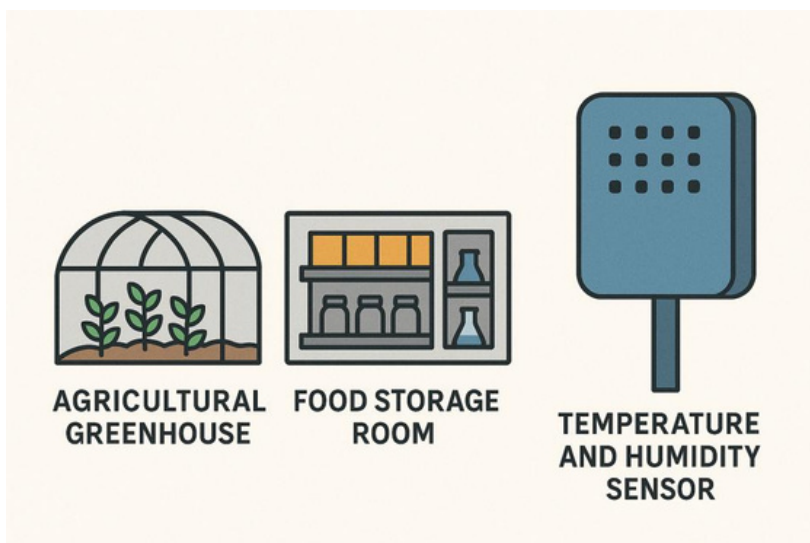
Cíle kurzu

- Pochopíte základní principy teplotních a vlhkostních senzorů.
- Zvládněte odečítání hodnot teploty a vlhkosti.
- Napište program, který bude zobrazovat informace o teplotě a vlhkosti v reálném čase na sériovém portu.



Úvod do kurzu

Teplota a vlhkost všudypřítomně ovlivňují naše zdraví a pohodlí: například nadměrná vlhkost v létě může vést k plísním, zatímco nadměrné sucho v zimě může způsobit suchou a popraskanou pokožku. V zemědělských sklenících, skladech potravin a laboratořích jsou teplota a vlhkost ještě důležitějšími faktory ovlivňujícími růst plodin, konzervaci potravin a přesnost experimentů. Aby inteligentní systémy mohly tyto změny prostředí „vnímat“, jsou potřeba senzory teploty a vlhkosti, které fungují jako „oči a nosy“ a shromažďují data o prostředí v reálném čase.



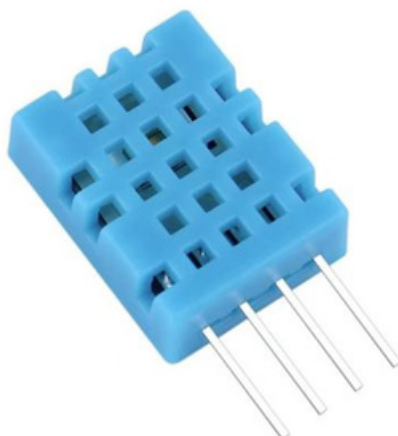
V této lekci se naučíme, jak fungují senzory teploty a vlhkosti a jak číst a zobrazovat okolní teplotu a vlhkost v reálném čase.



Body znalostí

1. Snímač teploty a vlhkosti

DHT11 je digitální senzor, který integruje funkce detekce teploty a vlhkosti. Jeho výhodou jsou malé rozměry, nízká spotřeba energie a stabilní přenos signálu. Je široce používán v oblastech monitorování životního prostředí, jako jsou domácnosti, zemědělství a laboratoře.



Princip detekce vlhkosti:

DHT11 používá kapacitní senzor vlhkosti k měření obsahu vodní páry ve vzduchu. Se změnou vlhkosti vzduchu se mění i hodnota kapacity hygroskopického materiálu uvnitř senzoru. Vnitřní čip převádí tuto změnu kapacity na digitální signál představující aktuální vlhkost (jednotka: %RH, relativní vlhkost).

Princip detekce teploty:

Měření teploty využívá NTC termistor. Když okolní teplota stoupá, odpor termistoru se snižuje; když

S klesající teplotou se odpor zvyšuje. DHT11 používá interní analogově-digitální převod k převodu změny odporu na hodnotu teploty (jednotka: °C).

Parametry snímače teploty a vlhkosti DHT11:

Parametry	Popis
Napětí zdroje napájení	Stejnoseměrné napětí: 3,3–5,5 V
Rozsah měření teploty	-20 až +60 °C
Rozsah měření vlhkosti	5–95 % relativní vlhkosti
Přesnost měření teploty	±2 °C
Přesnost měření vlhkosti	±5 % relativní vlhkosti
Stabilita	Teplota: ±1 °C Vlhkost: ±1 % relativní vlhkosti

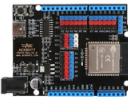
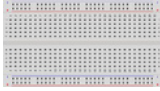


Experiment

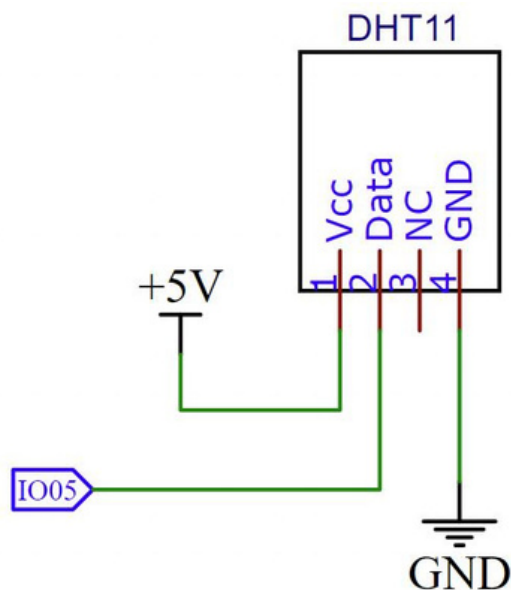
1. Popis projektu

Sériový monitor zobrazuje teplotu a vlhkost v reálném čase.

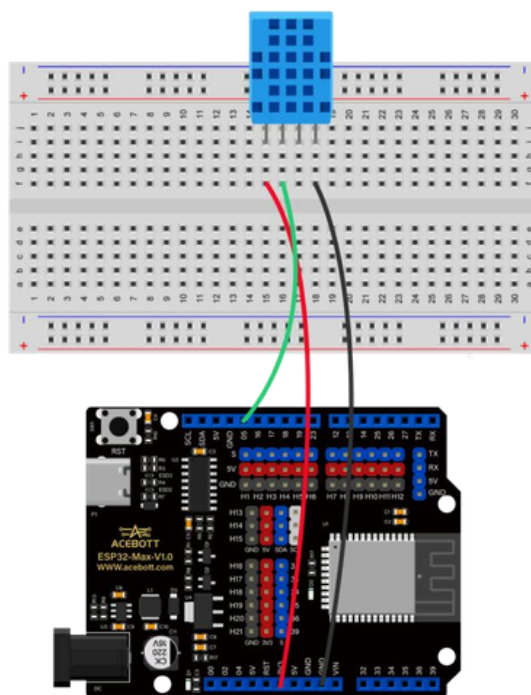
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Teplota a vlhkost Senzor	1
	Nepájivé pole s 400 spojovacími body	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Senzor teploty a vlhkosti.ino](#)" soubor v jazyce "English \Arduino(Experienced Learner)\2.Code\lesson 12 Temperature and Humidity Sensor\Temperature_and_Humidity_Sensor" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojového souboru

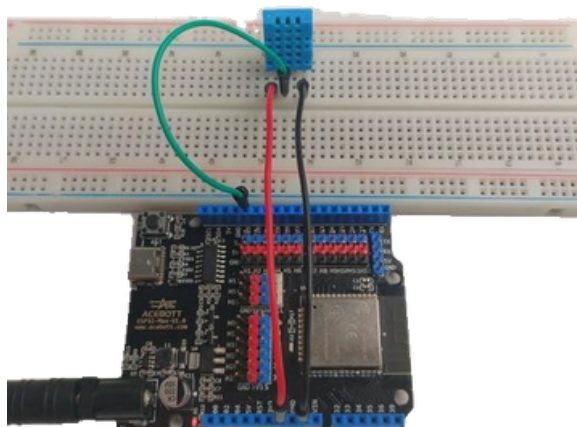
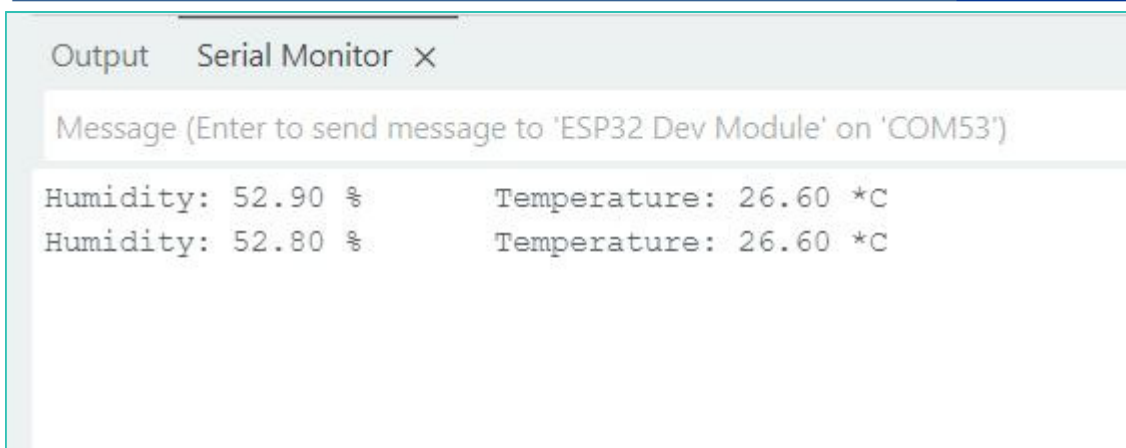
deska (vývojový modul ESP32) a příslušný port podle

dříve naučené kroky, klikněte  nahrát program.

```
#include <DHT.h>
#define DHTPIN 5 // Set the pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // DHT11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(115200);
  Serial.println("DHT11 test!");
  dht.begin();
}

void loop() {
  // Wait two seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (it's a very slow sensor)
  float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temperature = dht.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Print the humidity and temperature
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
}
```

Po dokončení nahrávání programu monitor sériového portu zobrazuje hodnoty okolní teploty a vlhkosti v reálném čase.



6. Popis instrukcí

`include<DHT.h>` Importujte knihovnu DHT pro čtení dat teploty a vlhkosti ze senzoru.

`DHT dht (pin, typ)` Vytvořte objekt senzoru DHT s názvem `dht`, který umožňuje volání funkcí pro inicializaci senzoru a načtení hodnot teploty a vlhkosti. Pin představuje zadané číslo I/O pinu a `typ` představuje model senzoru (např. DHT11 nebo DHT22).

`begin()` Inicializace senzoru teploty a vlhkosti.

`readHumidity()` Odečtete aktuální relativní vlhkost okolí, naměřenou

v %RH (relativní vlhkosti).

readTemperature(isFahrenheit): Přečte aktuální okolní teplotu. Parametr isFahrenheit určuje jednotku teploty: pokud je nastaven na true, vrácená hodnota je ve Fahrenheitech (°F); při nastavení na hodnotu false (výchozí), vrácená hodnota je ve stupních Celsia (°C).

Lekce 13 Ultrazvukový senzor



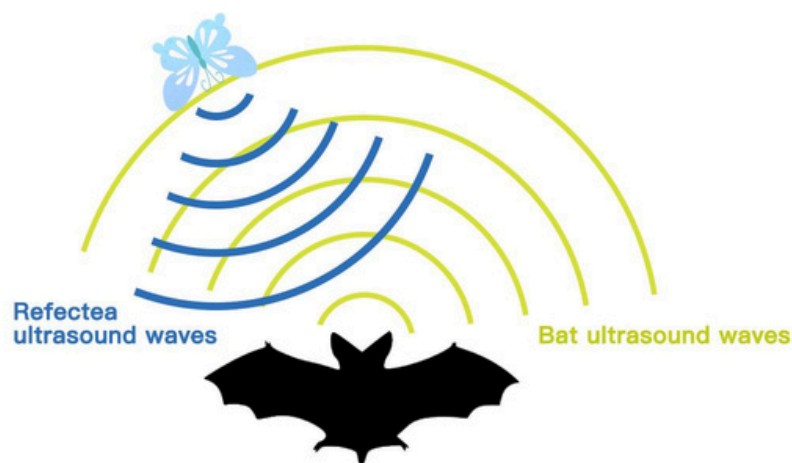
Cíle kurzu

- Pochopíte základní principy ultrazvukového modulu. Zvládněte, jak
- používat ultrazvukový modul k měření vzdálenosti.
- Napište program pro ovládání sériového portu a zobrazení dat měření vzdálenosti ultrazvukového modulu v reálném čase.



Úvod do kurzu

Víte, jak netopýři létají ve tmě a přesně chytají kořist? Používají ultrazvuk k vysílání vysokofrekvenčních zvukových vln. Když se zvukové vlny odrážejí od překážek, netopýři vypočítají časový rozdíl mezi ozvěnami, aby určili vzdálenost a směr. Ultrazvukové senzory napodobují tento přírodní jev a pomocí ultrazvuku měří vzdálenost objektů.



V této lekci se naučíme, jak fungují ultrazvukové senzory a jak číst a zobrazovat vzdálenost objektů v prostředí v reálném čase. Měřením časového rozdílu mezi ultrazvukovou emisí a ozvěnou

Na základě příjmu signálu můžeme vypočítat vzdálenost mezi cílovým objektem a senzorem. Tato data o vzdálenosti lze využít v různých aplikacích, jako je vyhýbání se překážkám roboty, monitorování hladiny kapalin a inteligentní parkování, což umožňuje vnímání a reakci na prostředí v reálném čase.



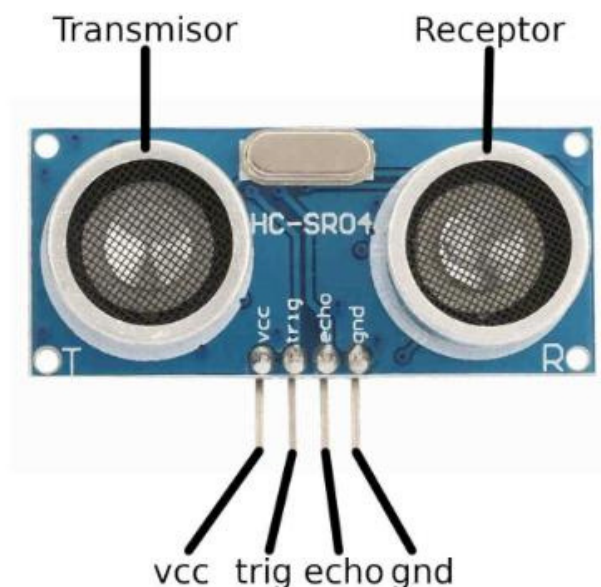
Body znalostí

1. Ultrazvukové vlny

Ultrazvukové vlny jsou zvukové vlny s frekvencemi nad horní hranicí lidského sluchu (přibližně 20 000 Hz nebo 20 kHz). Přestože jsou pro lidské ucho neslyšitelné, ultrazvukové vlny se mohou šířit vzduchem, vodou a pevnými látkami. Ultrazvukové vlny se díky své krátké vlnové délce, silnému průniku a vynikající směrovosti široce používají v lékařské diagnostice, průmyslové kontrole, měření vzdálenosti a čištění.

2. Ultrazvukový modul

Ultrazvukový modul je běžný modul pro měření vzdálenosti. Neustále detekuje ozvěny odražené ultrazvukovými vlnami po jejich vyzařování. Měří časový rozdíl mezi vyslanými a přijatými ozvěnami a tento časový rozdíl používá k výpočtu vzdálenosti. Skládá se hlavně ze dvou piezoelektrických keramických ultrazvukových senzorů a periferního obvodu pro zpracování signálu. Jeden senzor slouží k vysílání ultrazvukových signálů a druhý k přijímání odražených ultrazvukových signálů.



(1) Parametry modulu:

Parametry	Popis
Provozní napětí	DC 5V
Provozní proud	15mA
Pracovní frekvence	40kHz
Úhel detekce	15 stupňů
Detekční vzdálenost	2cm - 450cm

(2) Popis pinu:

VCC:

Napájecí pin, připojený k pinu 5V na desce řídicí jednotky ESP32.

Trig (spouštěcí pin):

Používá se ke spuštění modulu k vysílání ultrazvukového signálu.

Echo (Echo Pin):

Používá se k příjmu odražených ultrazvukových signálů a k výstupu vysokofrekvenčního impulsu. Šířka impulsu představuje čas potřebný k šíření ultrazvukové vlny tam a zpět, který se používá k výpočtu vzdálenosti.

Zem: Zemní pin, připojený k pinu GND na desce řídicí jednotky ESP32.

(3) Princip fungování ultrazvukového modulu

① Spouštění rozsahu

Připojte pin TRIG k IO portu na řídicí jednotce a spusťte proces změny rozsahu odesláním vysokého signálu po dobu alespoň 10 mikrosekund.

2 Vyslání ultrazvukového impulsu

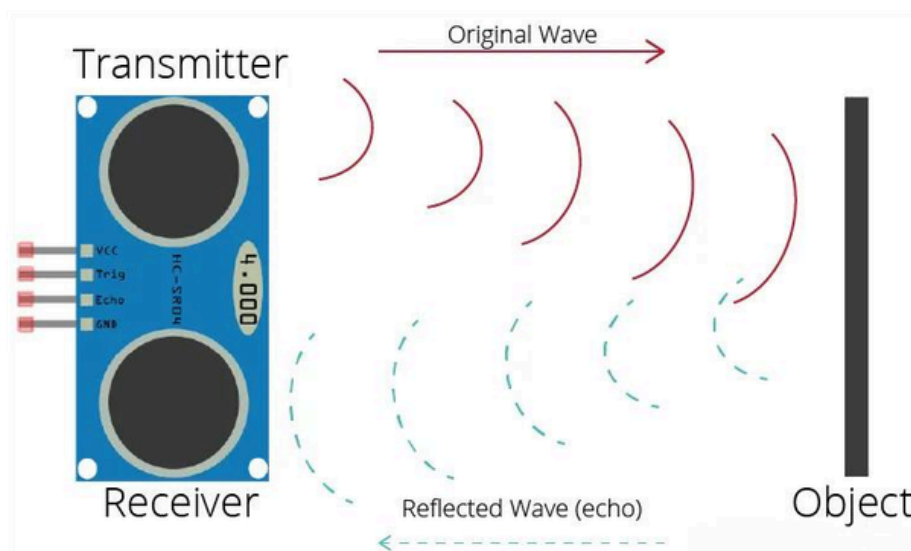
Modul automaticky vysílá osm ultrazvukových obdélníkových impulzů o frekvenci 40 kHz a monitoruje jakékoli odrazové signály ozvěny.

③ Příjem signálu

Když je detekována ozvěna, modul vyšle na pin ECHO impuls vysoké úrovně. Délka tohoto impulsu se rovná době potřebné k vyslání a přijetí ultrazvukové vlny. Změřením šířky tohoto impulsu může modul vypočítat vzdálenost k cíli.

Vzorec pro výpočet vzdálenosti:

$$\text{vzdálenost} = \frac{\text{čas} \times \text{frekvence}}{2}$$





Experiment

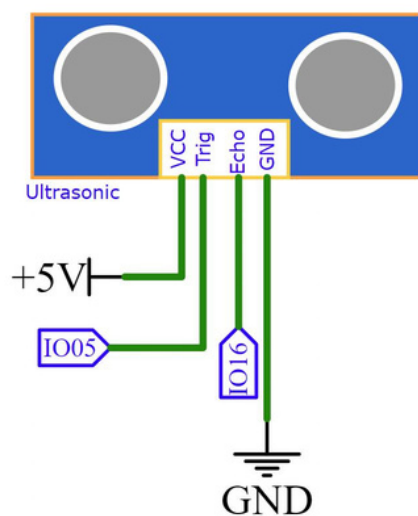
1. Popis projektu

Zobrazujte data měření vzdálenosti v reálném čase prostřednictvím sériového monitoru.

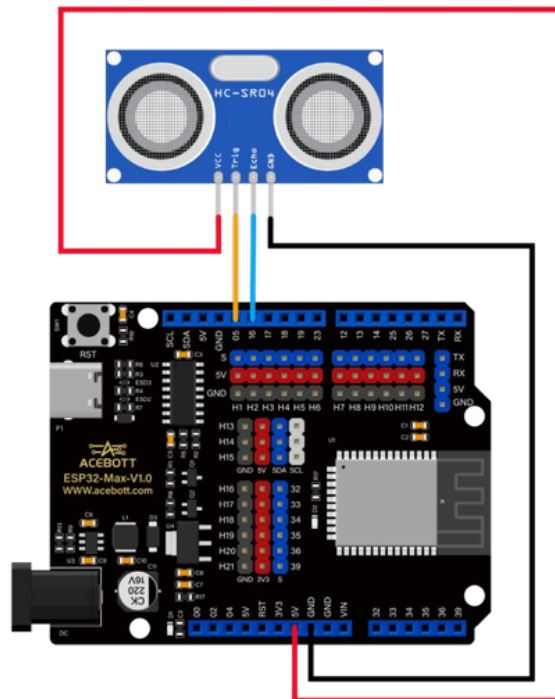
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Ultrazvukový senzor	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Ultrazvuk.ino](#)" v souboru "English\Arduino(Experienced Learner) \2.Code\lesson 13 Ultrasonic\Ultrasonic" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutí  nahrát program.

```
// Define ultrasonic sensor pins
const int trigPin = 5;
const int echoPin = 16;

void setup() {
  Serial.begin(115200); // Initialize serial communication.
  pinMode(trigPin, OUTPUT); // Set the trigger pin as output.
  pinMode(echoPin, INPUT); // Set the echo pin as input.
}

void loop() {
  long duration;
  float distance;
```

```
// Send ultrasonic pulse.
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

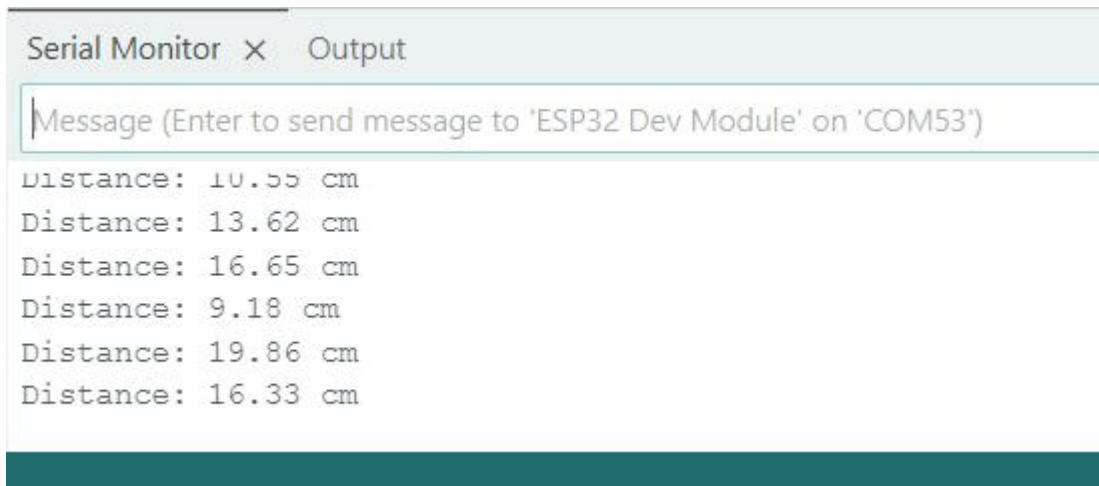
// Read the duration of the echo pulse.
duration = pulseIn(echoPin, HIGH);

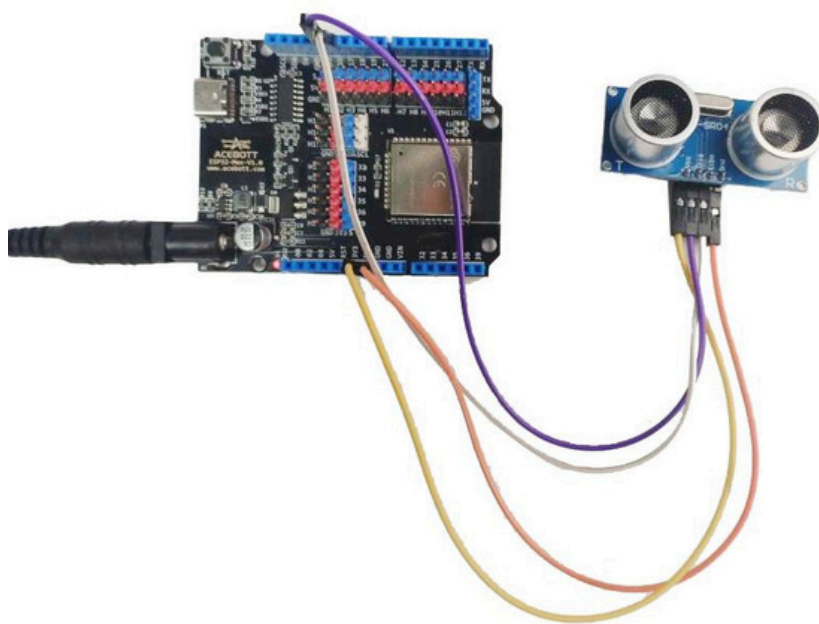
// Calculate distance in centimeters
// Speed of sound is approximately 343 m/s, which equals 0.0343 cm/μs
// Divide the total time by 2 to get the one-way travel time
distance = (duration * 0.0343) / 2;

// Display the distance value.
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");

delay(500); //Measure once every 0.5 seconds.
}
```

Po úspěšném nahrání programu se na displeji sériového portu zobrazí data měření vzdálenosti v reálném čase.





6. Popis instrukcí

pulseIn(pin, hodnota): Měří trvání pulzního signálu na zadaném pinu v mikrosekundách (μs). Pin představuje zadané číslo I/O pinu a hodnota udává stav úrovně pulzu, buď HIGH (vysoká), nebo LOW (nízká), a určuje, zda se má měřit pulz vysoké nebo nízké úrovně.

delayMicroseconds(čas): Pozastaví provádění programu na zadaný počet mikrosekund (μs).

Lekce 14 8segmentový displej



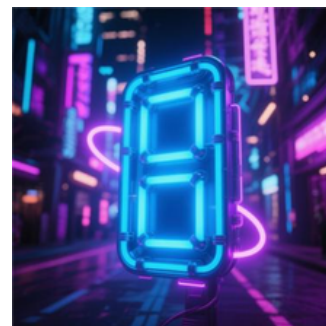
Cíle kurzu

- Pochopíte základní strukturu a princip fungování 8segmentového displeje.
- Zvládněte ovládání displeje pro zobrazení čísel.
- Použijte programovací instrukce k ovládání 8segmentového displeje pro cyklování na displeji od 0 do 9.



Úvod do kurzu

Všimli jste si někdy blikajících čísel na elektronických hodinách, ukazatelích pater výtahů, elektronických vahách a dalších zařízeních? Tato čísla se zobrazují pomocí elektronické součástky zvané 8segmentový displej. Skládá se z osmi nezávislých světelných diod.



segmenty a rozsvícením v různých kombinacích dokáže zobrazovat čísla a části písmen.

V této lekci se seznámíme se strukturou a základním principem fungování 8segmentového displeje. Také se naučíme, jak ovládat 8segmentový displej pro zobrazení čísel nebo písmen.



Body znalostí

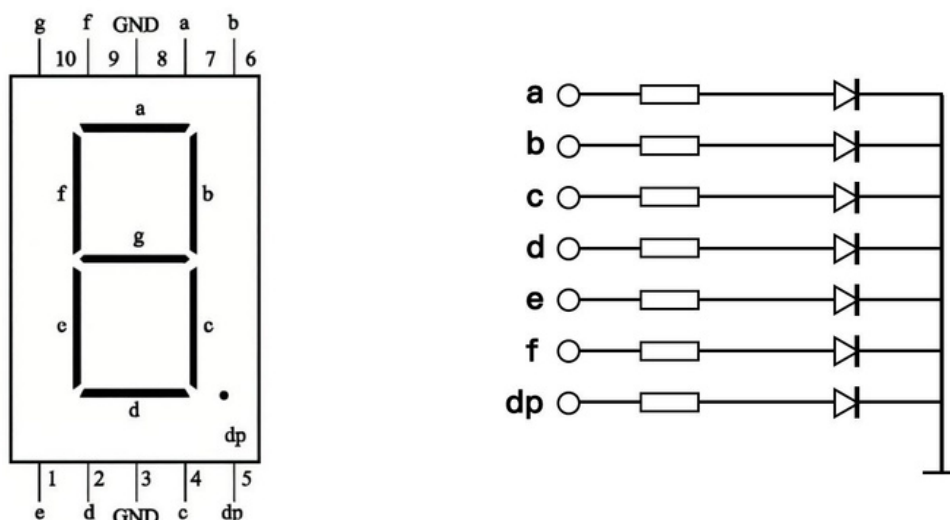
1. 8segmentový displej

Osmisegmentový displej se skládá z osmi nezávislých světelných diod.

segmenty uspořádané do tvaru čísla „8“. Rozsvícením různých kombinací může zobrazovat čísla 0 až 9 a některá písmena. Každý segment se obvykle skládá z LED diody, která se rozsvítí, když je připojen proud. Ovládání různých segmentů umožňuje zobrazení různých znaků. 8segmentové displeje se široce používají v elektronických hodinách, čítačích a měřicích přístrojích.



Zapojení pinů 8segmentového displeje (společná katoda):



Pro ovládání digitální trubice tedy stačí ovládat LED diody pouze jednotlivě. Například pro zobrazení čísla 5 stačí rozsvítit segmenty LED diod a, f, g, c a d a ztlumit LED diody b, e a dp. Tím dosáhneme požadovaného efektu.

Tabulka kódů digitálních segmentů 8segmentového displeje (0-9):

Číslo	Piny displeje							
	A	b	C	d	E	F	G	dp
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0

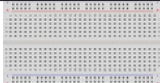


Experiment

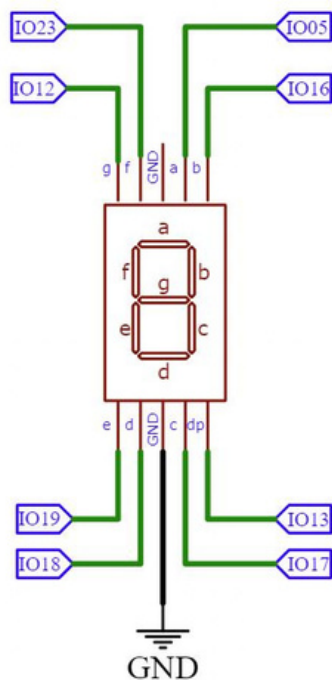
1. Popis projektu

Ovládejte 8segmentový displej tak, aby zobrazoval čísla 0 až 9, každé po dobu 0,5 sekundy.

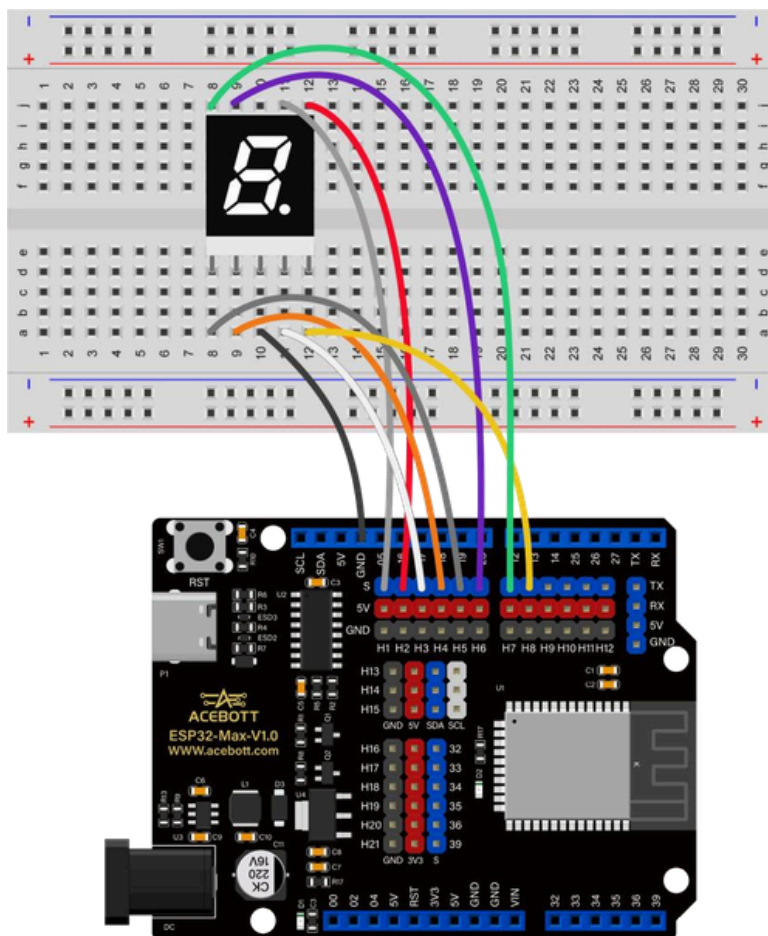
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	8segmentový displej	1
	Nepájivé pole s 400 spojovacími body	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Eight Segment Display.ino](#)" soubor v "English \Arduino (Experienced Learner) \2.Code\lesson 14 Eight_Segment_Display \Eight_Segment_Display" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušný port podle dříve naučených kroků klikněte na  nahrát program.

```
// Define the output pins and store all pin numbers in an array.
int pin_a = 5;
int pin_b = 16;
int pin_c = 17;
int pin_d = 18;
int pin_e = 19;
int pin_f = 23;
int pin_g = 12;
int pin_dp = 13;

int pin_array[8] = { pin_a, pin_b, pin_c, pin_d, pin_e, pin_f, pin_g,
pin_dp };

// Define a two-dimensional array for numeric display logic.
int number_array[][8] = {
{ 1, 1, 1, 1, 1, 1, 0, 0 },// 0
{ 0, 1, 1, 0, 0, 0, 0, 0 },// 1
{ 1, 1, 0, 1, 1, 0, 1, 0 },// 2
{ 1, 1, 1, 1, 0, 0, 1, 0 },// 3
{ 0, 1, 1, 0, 0, 1, 1, 0 },// 4
{ 1, 0, 1, 1, 0, 1, 1, 0 },// 5
{ 1, 0, 1, 1, 1, 1, 1, 0 },// 6
{ 1, 1, 1, 0, 0, 0, 0, 0 },// 7
{ 1, 1, 1, 1, 1, 1, 1, 0 },// 8
{ 1, 1, 1, 1, 0, 1, 1, 0 },// 9
};

void display_number(int num) {
for (int i = 0; i < 8; i++) {
digitalWrite(pin_array[i], HIGH);
}

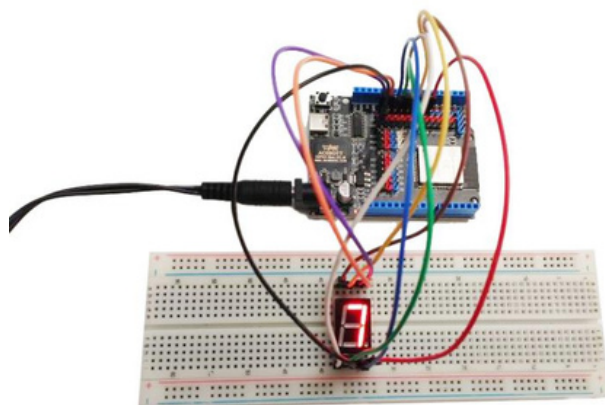
for (int i = 0; i < 8; i++) {
digitalWrite(pin_array[i], number_array[num][i]);
}
}

void setup() {
```



```
for (int i = 0; i < 8; i++) {  
  pinMode(pin_array[i], OUTPUT);  
  digitalWrite(pin_array[i], HIGH);  
}  
}  
  
void loop() {  
  for (int i = 0; i < 10; i++) {  
    display_number(i);  
    delay(500);  
  }  
}
```

Po úspěšném nahrání programu se na 8segmentovém displeji zobrazí čísla 0 až 9, každé po dobu 0,5 sekundy.



Lekce 15 4místná digitální elektronka



Cíle kurzu

- Pochopíte základní principy fungování 4místné digitální elektronky. Ovládněte metodu multiplexování k ovládání digitální elektronky.
- Použijte programovací instrukce k ovládání 4místné digitální trubice pro zobrazení čísel



Úvod do kurzu

V předchozí lekci jsme se naučili, jak používat 8segmentový displej k zobrazení čísel a zvládli jsme ovládání segmentového kódu. Může se však stát, že dokáže zobrazit pouze jedno číslo, což je velmi omezující, pokud potřebujete zobrazit více čísel (například čas, teplotu nebo hodnoty počtu).



V této lekci se naučíme, jak používat čtyřmístný 8segmentový displej a zvládneme technologii dynamického skenování k dosažení nepřetržitého zobrazení více číslic.

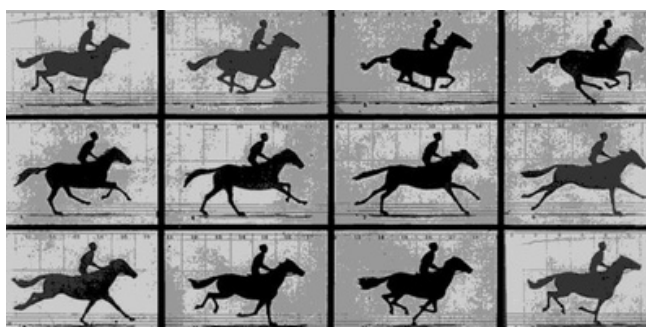


Body znalostí

1. Perzistence vidění

Perzistence vidění označuje „zbytkové vnímání“ vnějších obrazů lidským okem po krátkou dobu. Když světlo nebo obraz zmizí ze zorného pole, lidské oko okamžitě neztratí vnímání; místo toho nadále „vidí“ původní obraz po dobu přibližně 0,1 sekundy. Tento jev se nazývá perzistence vidění.

Perzistence vidění je v každodenním životě běžná. Například filmy a animace přehrávají statické obrazy vysokou rychlostí (například 24 snímků za sekundu), což způsobuje, že je lidské oko vnímá jako nepřetržitý pohyb; lopatky ventilátoru se vysokorychlostními pohyby zdají oku rozmazané nebo dokonce „mizí“; a použití baterky k kreslení vzorů nebo textu v noci a následné pořízení fotografie vytváří efekt „malby světlem“. Je to proto, že pohyb světla zanechává v lidském oku krátký dozvuk. Tyto jevy jsou založeny na principu perzistence vidění, který nám umožňuje vnímat nepřetržité dynamické obrazy nebo trajektorie.



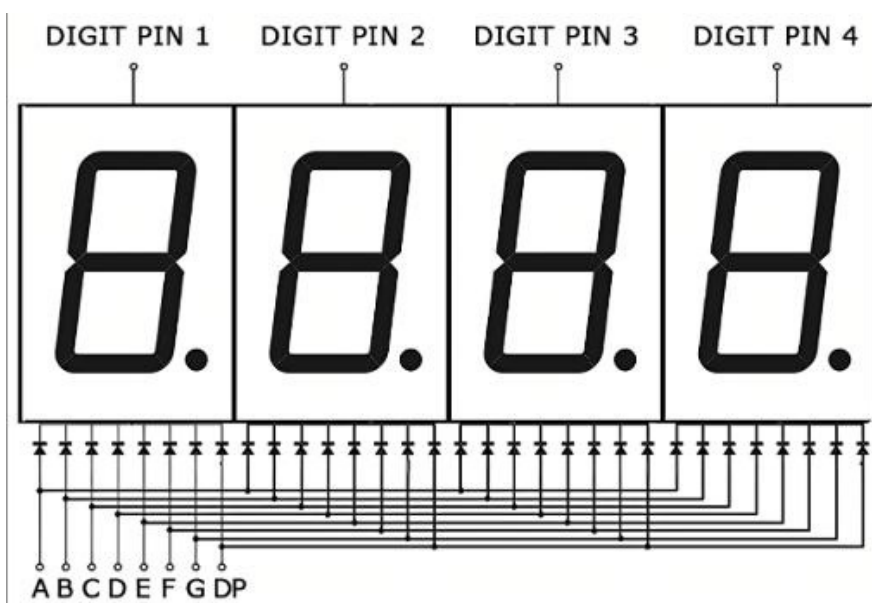
2. Čtyřmístná osmisegmentová digitální trubice Čtyřmístná

osmisegmentová digitální elektronka se skládá ze 4 nezávislých osmisegmentových digitálních elektronek.



Typická jednomístná osmisegmentová digitální elektronka obvykle vyžaduje připojení osmi segmentových výběrových linek (a až g a dp) a společného pinu (GND nebo VCC, v závislosti na typu). Pro úsporu pinů používá čtyřmístná osmisegmentová digitální elektronka sdílenou osmisegmentovou výběrovou linku a další čtyřbitovou výběrovou linku (D1 až D4) pro výběr aktuálně aktivní číslice.

- Řádky pro výběr segmentů (a až g, dp) Řídí digitální obsah, který se má zobrazit pro každou číslici;
- Linka pro výběr bitů (D1~D4) řídí, která digitální elektronka svítí.



V daném okamžiku může svítit pouze jedna číslice; jinak všechny číslice sdílející stejný segment budou zobrazovat stejnou informaci. Abychom dosáhli efektu „zobrazení“ čtyř číslic současně, používáme metodu zvanou dynamické skenování (multiplexování):

Rychle rozsvítíte každý displej postupně a vypíšete odpovídající číslo.

Každá číslice se zobrazí po velmi krátkou dobu (například 1~5 ms) a poté se okamžitě přepne na další číslici;

Vzhledem k perzistenci vidění lidského oka se zdá, že čtyři číslice jsou zobrazeny současně.

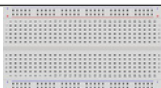


Experiment

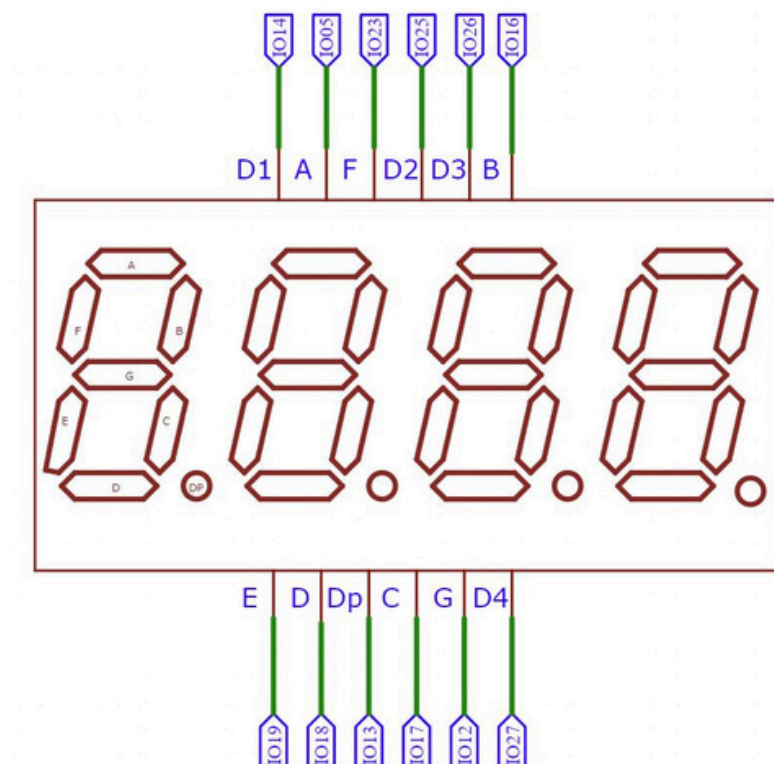
1. Popis projektu

Ovládejte čtyřmístnou osmisegmentovou digitální trubici tak, aby se zobrazila hodnota „1234“.

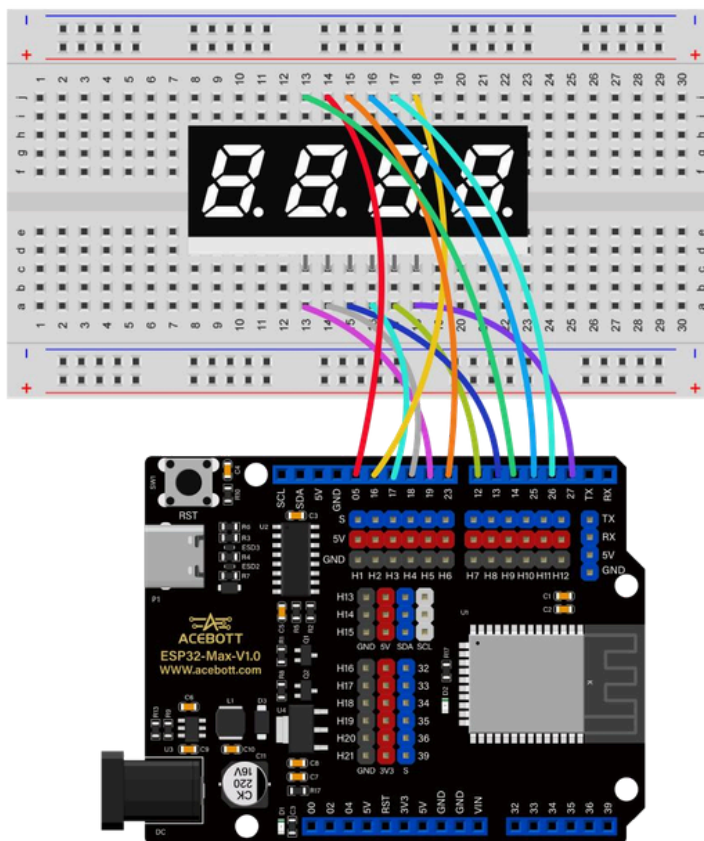
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	4místná digitální trubice	1
	Nepájivé pole s 400 spojovacími body	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Čtyřmístný osmisegmentový display.ino](#)" soubor v "Angličtina\Arduino (zkušený) Žák)\2.Kód\lekce 15 4místný 8segmentový displej\Four_digit_Eight_Segment_Display" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

```
// Define digit enable pins and store them in an array
int seg_1 = 14;
int seg_2 = 25;
int seg_3 = 26;
int seg_4 = 27;

int seg_array[4] = { seg_1, seg_2, seg_3, seg_4 };

// Define segment control pins and store them in an array
int a = 5;
```

```

int b = 16;
int c = 17;
int d = 18;
int e = 19;
int f = 23;
int g = 12;
int dp = 13;

int led_array[8] = { a, b, c, d, e, f, g, dp };

// Define a 2D array for segment logic corresponding to digits 0-9
int logic_array[][8] = {
{ 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
{ 0, 1, 1, 0, 0, 0, 0, 0 }, // 1
{ 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
{ 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
{ 0, 1, 1, 0, 0, 1, 1, 0 }, // 4
{ 1, 0, 1, 1, 0, 1, 1, 0 }, // 5
{ 1, 0, 1, 1, 1, 1, 1, 0 }, // 6
{ 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
{ 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
{ 1, 1, 1, 1, 0, 1, 1, 0 }, // 9
};

// Turn off all segments and digits
void clear() {
// Turn off all segments
for (int i = 0; i < 8; i++) {
digitalWrite(led_array[i], LOW);
}
// Disable all digits (active LOW)
for (int i = 0; i < 4; i++) {
digitalWrite(seg_array[i], HIGH); // Disable digit (assuming active LOW)
}
}

// Display a digit (0-9) at a given position (0-3)
void display_number(int order, int number) {
clear(); // Clear all digits and segments before displaying new digit
digitalWrite(seg_array[order], LOW); // Enable selected digit (active LOW)
for (int i = 0; i < 8; i++) {
digitalWrite(led_array[i], logic_array[number][i]); // Set segments for
the number
}
}

void setup() {
// Initialize digit enable pins as outputs and disable all digits
for (int i = 0; i < 4; i++) {

```

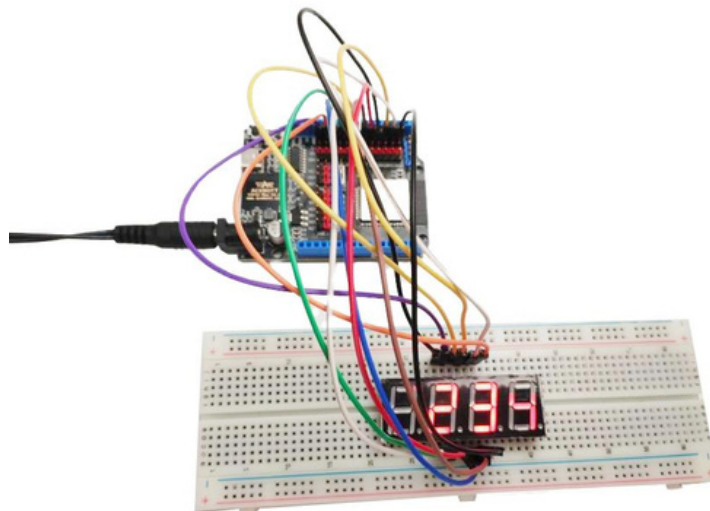


```
pinMode(seg_array[i], OUTPUT);
digitalWrite(seg_array[i], HIGH); // Disable all digits initially (active
LOW)
}
// Initialize segment pins as outputs and turn all segments off
for (int i = 0; i < 8; i++) {
pinMode(led_array[i], OUTPUT);
digitalWrite(led_array[i], LOW); // Turn off all segments initially
}
}

void loop() {
int digits[4] = {1, 2, 3, 4}; // The numbers to display on the 4 digits

// Scan through each digit and display the corresponding number
for (int i = 0; i < 4; i++) {
display_number(i, digits[i]); // Display digit at position i
delay(5); // Short delay for multiplexing
}
}
```

Po úspěšném nahrání programu se na čtyřmístném osmissegmentovém digitálním displeji zobrazí číslo 1234.



Lekce 16 Infračervené dálkové ovládání



Cíle kurzu

- Pochopíte základní principy a mechanismy fungování infračervených dálkových ovladačů.
- Pochopíte mechanismus přenosu signálu infračervených dálkových ovladačů a analýzu signálu infračervenými přijímači.
- Zvládněte schopnost psát programy pro analýzu a zpracování přijatých signálů infračerveného dálkového ovládání.



Úvod do kurzu

Dálkové ovladače se staly nepostradatelným nástrojem v našem každodenním životě a hojně se používají v různých domácích spotřebičích, jako jsou televizory, klimatizace a ventilátory. I když se ovládání tlačítka může zdát jednoduché, ve skutečnosti využívají úžasnou technologii: infračervené dálkové ovládání. Infračervené dálkové ovladače vysílají infračervené světlo, které je pro lidské oko neviditelné, k přenosu povelových signálů, což umožňuje zařízením přesně a rychle reagovat na operace uživatele.



V této lekci se ponoříme do základních principů a mechanismů fungování infračervených dálkových ovladačů a prozkoumáme, jak jsou infračervené signály kódovány, odesílány a přijímány. Také se naučíme, jak povolit ústředně přijímat a interpretovat infračervené signály z dálkového ovladače, což umožní bezdrátové ovládání zařízení. Prostřednictvím tohoto procesu si nejen osvojíte základní technologii infračervených dálkových ovladačů, ale také zažijete pohodlí a zábavu moderního inteligentního ovládání.



Body znalostí

1. Základní principy infračerveného dálkového ovládání

Infračervené dálkové ovládání je bezdrátová technologie ovládání, která využívá infračervené světlo pro komunikaci. V předchozí lekci o lidských infračervených senzorech jsme se dozvěděli, že infračervené světlo jsou v podstatě elektromagnetické vlny s vlnovou délkou mezi 0,7 a 1000 mikrony, které jsou pro lidské oko neviditelné (ačkoli kamery dokáží infračervené světlo pozorovat a zachytit). Dálkové ovládání vysílá modulované infračervené signály prostřednictvím infračervené emitující diody. Řídící jednotka (například televizor, ventilátor nebo ovládací panel automobilu) tyto signály přijímá a interpretuje prostřednictvím infračerveného přijímacího modulu, čímž dokončuje řídicí operaci.

Infračervené dálkové ovládání funguje následovně

(1) Kódovaný přenos: Každé tlačítko dálkového ovladače má odpovídající kód (obvykle hexadecimální číslo). Po stisknutí tlačítka dálkový ovladač převede kód na sérii pulzních signálů pomocí specifického komunikačního protokolu (toho se dosahuje rychlým řízením infračervené emitující diody (IR LED), která bliká (zapíná a vypíná) s frekvencí 38 000krát za sekundu, čímž vzniká infračervený nosný signál). Tento signál je poté přenášen prostřednictvím IR LED diody s...

modulovaná nosná frekvence 38 kHz.

(2) Infračervený přenos: Modulované infračervené světlo se šíří v pulzech vzduchem k cílovému zařízení.

(3) Příjem a dekódování: Přijímací strana používá infračervený přijímač k příjmu infračerveného pulzního signálu a jeho převodu na změny hladiny. Program na řídicí desce jej poté dekóduje a obnoví jeho původní hodnotu.

(4) Provádění příkazů: Řídicí deska na základě výsledku dekódování určí, kterou klávesu uživatel stiskl, a provede odpovídající operaci (například: rozsvícení LED, ovládání motoru, přepnutí režimu spouštění atd.).



IR dálkové ovládání



IR přijímač

Poznámka: Při použití infračerveného dálkového ovladače je nutné nainstalovat knihovnu IRremote podle pokynů v části „4. Přidání souborů knihovny“ v části „Příprava“ (Poznámka: Knihovna IRremote vyžaduje verzi 3.0 nebo vyšší).

Tabulka kódů infračerveného dálkového ovládání

Klíč Lisy	Hodnoty kódu	Klíč Lisy	Hodnoty kódu
	0xB946FF00		0xE718FF00
	0xBB44FF00		0xA15EFF00

	0xBC43FF00		0xF708FF00
	0xEA15FF00		0xE31CFF00
	0xBF40FF00		0xA55AFF00
	0xE916FF00		0xBD42FF00
	0xE619FF00		0xAD52FF00
	0xF20DFF00		0xB54AFF00
	0xF30CFF00		

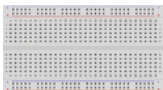


Experiment

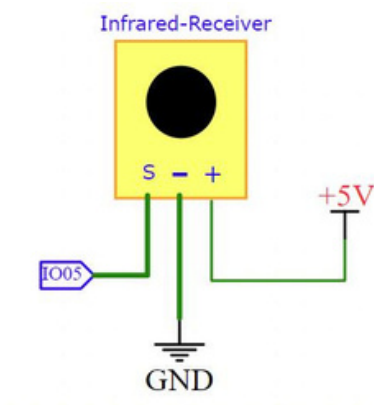
1. Popis projektu

Když je přijat infračervený signál, program jej analyzuje a vytiskne odpovídající kód klávesy (zobrazený v hexadecimálním formátu) a poté se připraví na příjem dalšího infračerveného signálu, což umožňuje čtení a zobrazení signálu infračerveného dálkového ovládání v reálném čase.

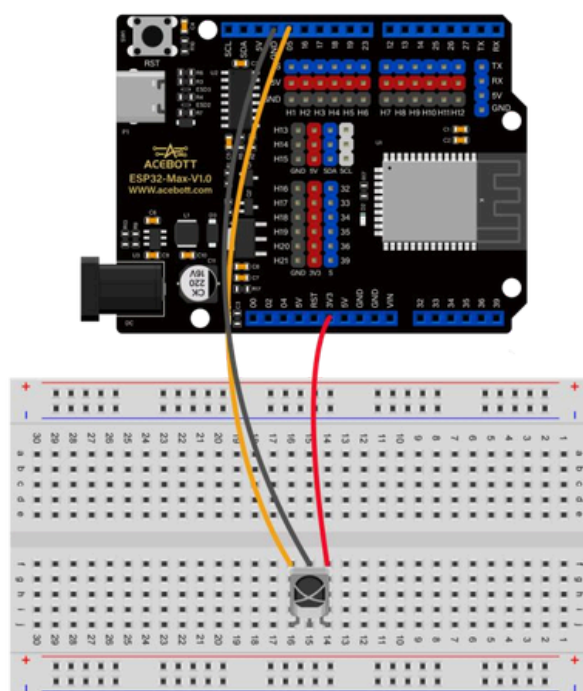
2. Seznam hardwaru

Obráz	Jméno	Množství
	Nepájivé pole s 400 spojovacími body	1
	Deska řadiče ESP32	1
	IR přijímač	1
	IR dálkový ovladač	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Infrared_Remote_Control.ino](#)" soubor v "Angličtina\Arduino (zkušený Learner)\2.Kód\lekce 16 Infračervené Remote Control\Infrared_Remote_Control" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutí  nahrát program.

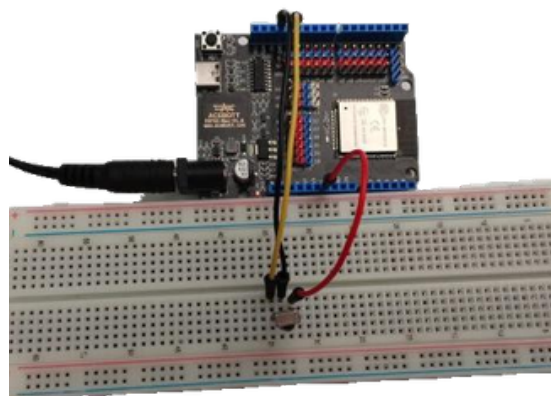
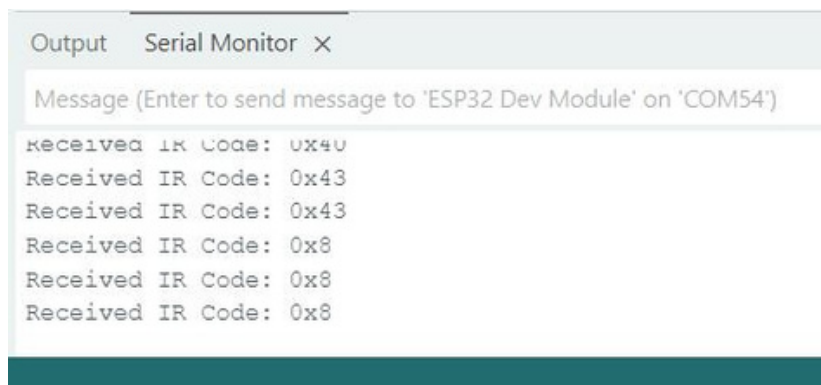
```
# zahrnout<IRremote.h>
```

```
#define IR_RECEIVE_PIN 5

void setup() {
  Serial.begin(115200);
  // Start the infrared receiver
  IrReceiver.begin(IR_RECEIVE_PIN);
  Serial.println("IR Receiver ready. Please press a button on the remote.");
}

void loop() {
  if (IrReceiver.decode()) {
    Serial.print("Received IR Code: 0x");
    //Print the key code
    Serial.println(IrReceiver.decodedIRData.command, HEX);
    IrReceiver.resume();// Receive the next IR signal
  }
}
```

Po úspěšném nahrání programu stiskněte tlačítko na infračerveném dálkovém ovladači a sériový monitor vytiskne odpovídající hodnotu kódu klávesy.



6. Popis instrukcí

include<IRremote.h> Importujte knihovnu IRremote pro příjem a odesílání infračervených signálů, což umožní funkci infračerveného dálkového ovládání.

IrReceiver.begin(IR_RECEIVE_PIN) Inicializujte modul infračerveného přijímače, abyste jej připravili na příjem infračervených signálů na zadaném pinu.

IrReceiver.decode(): Používá se k ověření, zda byl přijat infračervený signál. Tato funkce vrací booleovskou hodnotu: vrácení hodnoty True znamená, že byl úspěšně přijat kompletní rámec infračerveného signálu; vrácení hodnoty False znamená, že dosud nebyl přijat žádný signál nebo že data signálu ještě nejsou kompletní.

IrReceiver.decodedIRData.command: Používá se k načtení kódu příkazu z dekodovaného infračerveného signálu. Představuje specifickou hodnotu instrukce odpovídající stisknutému tlačítku na aktuálním infračerveném dálkovém ovladači a obvykle se používá k určení, které tlačítko uživatel stiskl.

IrReceiver.resume(): Používá se k obnovení stavu příjmu infračerveného přijímače a jeho přípravě k příjmu dalšího infračerveného signálu. Po volání této funkce přijímač vymaže předchozí datovou vyrovnávací paměť a pokračuje v naslouchání novým infračerveným signálům.

Lekce 17 Displej LCD1602



Cíle kurzu

- Pochopte základní strukturu a princip zobrazení displeje LCD1602.
- Zvládněte používání knihovny LiquidCrystal_I2C pro inicializaci zobrazovaných a výstupních informací.



Úvod do kurzu

V každodenním životě často vidíme text zobrazený na obrazovce, například výtahy zobrazující patra, autobusové zastávky oznamující informace o příjezdu a dokonce i uvítací zprávy na prodejních automatech. Tyto funkce zobrazení jsou neoddělitelné od důležité součásti – displeje z tekutých krystalů (LCD).



V této lekci se naučíme, jak pomocí řídicí desky ESP32 ovládat displej z tekutých krystalů LCD1602 a zobrazovat text.



Body znalostí

1. LCD1602 displej

LCD1602 je běžný znakově orientovaný LCD modul schopný zobrazovat dva řádky po 16 znacích současně, přičemž každý znak se skládá z bodové matice 5×8. Využívá technologii displeje z tekutých krystalů (LCD) a nabízí výhody, jako je nízká spotřeba energie, kompaktní rozměry a jasný displej. Tento modul se obvykle připojuje k řídicí desce ESP32 přes paralelní rozhraní nebo rozhraní I2C. Lze jej použít k zobrazení anglických znaků (AZ, az), čísel (0-9), symbolů (., : ; ? ! + - * = @ # % & '()) a vlastních znaků. Je široce používán ve výrobě elektroniky, informačních displejích a chytrých zařízeních.



Princip zobrazení LCD1602 spočívá v tom, že řídicí jednotka odesílá odpovídající data bodové matice na displej a LCD modul rozsvítí odpovídající pixely, čímž zobrazí požadované znaky a grafiku.

Například: Pro zobrazení znaku A v bodové matici 5x7 jednoduše nastavte odpovídající pozici požadovaného znaku na 1 (0 pro vypnuto, 1 pro zapnuto).

01110	□ ■ ■ ■ □
10001	■ □ □ □ ■
10001	■ □ □ □ ■
11111	■ ■ ■ ■ ■
10001	■ □ □ □ ■
10001	■ □ □ □ ■
00000	□ □ □ □ □

Při použití LCD1602 displeje z tekutých krystalů je třeba otočit potenciometrem na zadní straně pro nastavení jasu podsvícení tak, aby se text zobrazil normálně.



Poznámka: Pro ovládání displeje LCD1602 a zobrazování informací je třeba nainstalovat knihovnu LiquidCrystal_I2C dle pokynů v části „IV.Add Library Files“ v části „Preparation“.

2. Komunikace I2C

Komunikace I2C je sériový komunikační protokol, který využívá dva vodiče (datovou linku SDA a hodinovou linku SCL) k propojení více zařízení. Hlavní zařízení řídí čtení a zápis dat pomocí podřízeného zařízení prostřednictvím jedinečné adresy zařízení (adresa zařízení je pevně zakódována uvnitř čipu a nelze ji změnit), čímž se realizuje synchronní přenos mezi zařízeními a jednoduchá a efektivní výměna dat.

SCL (sériová hodinová linka)Hodinový signál generovaný hlavním zařízením.

SDA (sériová datová linka)Obousměrná datová přenosová linka používaná pro odesílání a příjem dat.

Kroky komunikace I2C (1)

Počáteční podmínka

Hlavní zařízení připojí linku SDA k nízké úrovni z vysoké, zatímco SCL udržuje vysokou, čímž signalizuje sběrnici zahájení komunikace.

(2) Odeslání adresy zařízení (Přenos adresy)

Hlavní zařízení odesílá 7bitovou adresu a bit pro čtení/zápis cíle.

podřízené zařízení, které upozorní podřízené zařízení, aby se připravilo na příjem nebo odesílání dat.

(3) Potvrzovací signál (ACK/NACK)

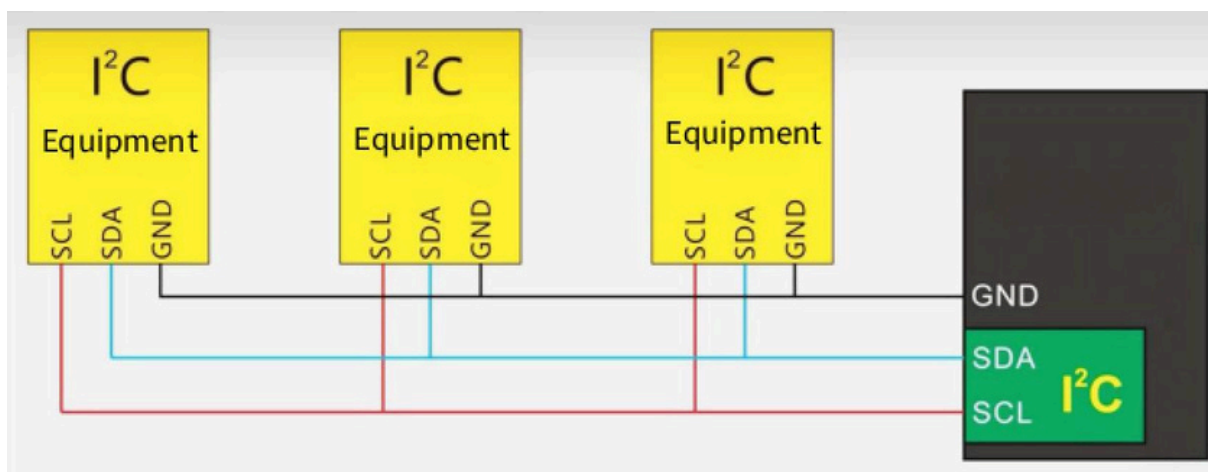
Po přijetí adresy od zařízení odešle signál ACK, který indikuje odpověď, jinak odešle signál NACK, který indikuje, že nedošlo k žádné odpovědi.

(4) Přenos dat

V závislosti na směru čtení/zápisu začne hlavní nebo podřízené zařízení odesílat datové bajty. Po odeslání každého bajtu musí přijímač vrátit signál ACK.

(5) Podmínka zastavení

Na konci komunikace master přepne úroveň SDA na vysokou, zatímco SCL zůstává na vysoké úrovni, čímž uvolní sběrnici.



Formát komunikace I2C (odesílání informací do dvou zařízení I2C):

```
#include <Wire.h>
void setup() {
  Wire.begin(21, 22);
  Serial.begin(115200);
}

void loop() {
  Wire.beginTransmission(0x3C); // Device 1 address (assumed to be 0x3C)

  // Send data

  Wire.write(message);
  Wire.endTransmission();
}
```

```
Wire.beginTransmission(0x68); // Device 2 address (assumed to be 0x68)

// Send data
Wire.write(message);
Wire.endTransmission();
delay(1000);
}
```



Experiment

1. Popis projektu

Ovládejte LCD1602 tak, aby na prvním řádku zobrazoval „ACEBOTT“ a na druhém řádku „Hello world!“.

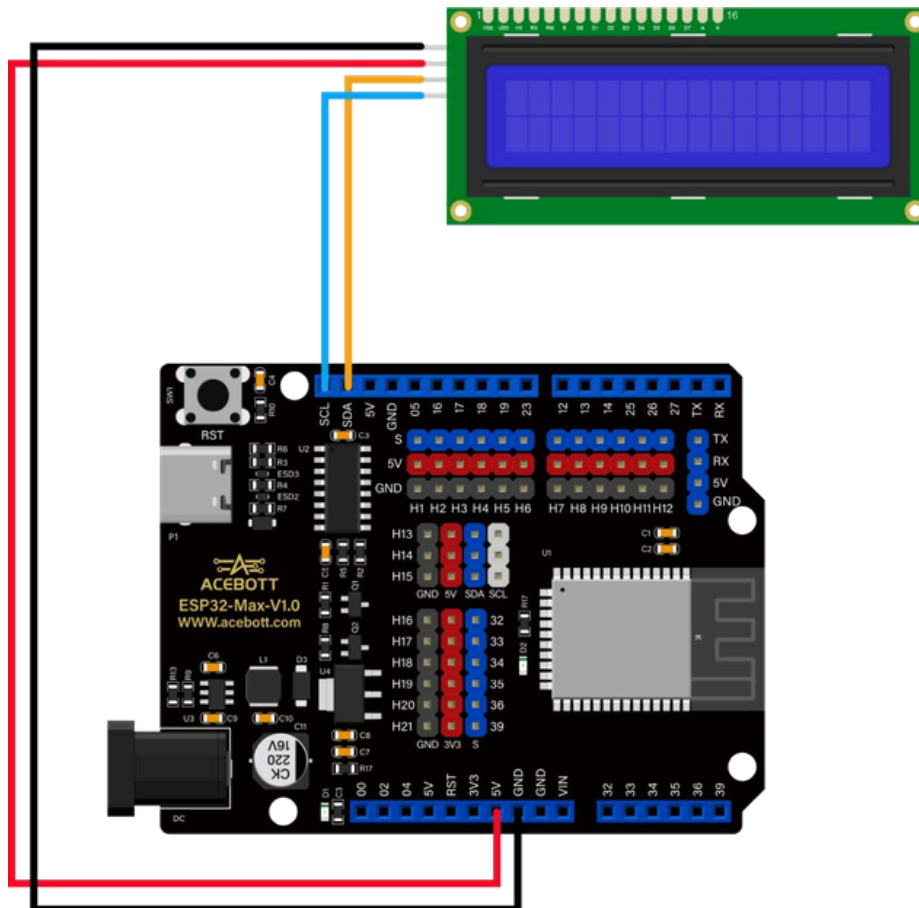
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	LCD1602	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[LCD1602.ino](#)" soubor v adresáři "English\Arduino(Experienced Learner)\2.Code\lesson 17 LCD1602\LCD1602" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutí  nahrát program.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

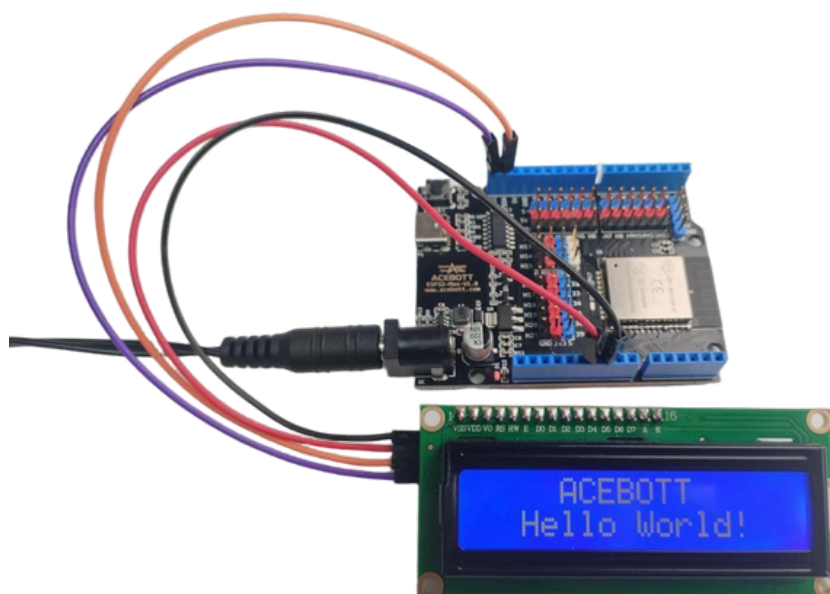
// Creates an LCD object. Parameters: I2C address of the LCD, number of
// columns, and number of rows
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  lcd.setCursor(0, 0); // Set the cursor to column 0, row 0
  lcd.print("  ACEBOTT  ");
}
```

```
lcd.setCursor(0, 1);    // Set the cursor to column 0, row 1
lcd.print(" Hello World! ");
}

void loop() {
}
```

Po úspěšném nahrání programu se na LCD1602 zobrazí „ACEBOTT“ a „Hello World!“.



6. Popis instrukcí `LiquidCrystal_I2C` `lcd(adresa_lcd, sloupce_lcd, řádky_lcd)`: Používá se k vytvoření objektu LCD displeje s názvem `lcd` zadáním adresy I2C (například 0x27 nebo 0x3F), počtu sloupců a počtu řádků za účelem inicializace modulu displeje LCD1602. Po vytvoření můžete objekt `lcd` použít k volání různých funkcí, jako je nastavení pozice kurzoru nebo tisk textu, a ovládat tak LCD obrazovku.

`init()` Inicializace LCD displeje, nastavení parametrů zobrazení a příprava obrazovky pro normální provoz.

backlight(): Zapnutí podsvícení LCD displeje zajistí jasnější a lépe viditelný obsah obrazovky.

setCursor(sloupec, řádek): Nastaví pozici kurzoru a připraví obsah k zobrazení v zadaném sloupci (col) a řádku (row). Parametr col začíná od 0 a označuje číslo sloupce, a parametr row začíná od 0 a označuje číslo řádku.

print(): Zobrazí text nebo čísla na aktuální pozici kurzoru. Můžete předat řetězec, znaky nebo číselné hodnoty, které se zobrazí na LCD obrazovce.

Kapitola 3 Komplexní projekty

Tato kapitola studenty provede kombinací více elektronických modulů k dokončení projektu inteligentního řízení s praktickou aplikační hodnotou. Prostřednictvím systematických praktických operací si studenti nejen upevní své znalosti o každém modulu, ale také si osvojí základní dovednosti, jako je spolupráce modulů, návrh strukturálního uspořádání a multifunkční integrace, čímž si zlepší své komplexní aplikační dovednosti při řešení praktických problémů.

Lekce 18 Plynoucí světlo



Cíle kurzu

- Pochopte základní principy proudícího světla
- Pochopte „fenomén perzistence vidění“ a jeho aplikaci v proudícím světle
- Zvládněte používání programových instrukcí k dosažení efektu proudících světél



Úvod do kurzu

V předchozí lekci jsme se naučili, jak ovládat zapínání a vypínání LED diody. Pokud tedy máme více LED diod, můžeme je rozsvěcet „rytmicky“ v pořadí, jako jevištní světla?

V této lekci budeme ovládat několik LED diod se postupně rozsvěcujících a poté zhasínajících, čímž vytvářejí efekt běžícího světla.



Body znalostí

1. Plynulé světlo

Plynulé světlo je stav, kdy se více LED diod rozsvěcuje a zhasíná postupně podle nastavené sekvence a časového intervalu, čímž vytváří vizuální efekt podobný pohybu vody.

Principy fungování proudícího světla

Princip běžícího světla spočívá v postupném ovládání stavu zapnutí/vypnutí více LED diod, jejich rozsvěcování zleva doprava nebo zprava doleva, čímž se vytváří „plynulý“ vizuální efekt. V podstatě se jedná o časové ovládání více LED diod v kombinaci s fenoménem perzistence vidění, které vytváří iluzi nepřetržitého pohybu.

(1) Paralelní ovládání více LED diod

Více LED diod je připojeno k různým GPIO pinům a každou LED diodu lze ovládat nezávisle.

(2) Postupné zapínání a vypínání LED diod

Pomocí příkazů rozsvítíte jednotlivé LED diody postupně a po chvíli je vypnete, než rozsvítíte další LED diodu, čímž vytvoříte efekt „plynulosti“.

(3) Použití delaye k vytvoření rytmu

Po rozsvícení každé LED diody přidejte delay() pro řízení časového intervalu před rozsvícením další LED diody, čímž vytvoříte trojrozměrný efekt „plynulosti“.

(4) Zachování vidění pro posílení smyslu pro koherenci Kvůli

perzistenci vidění lidské oko mylně vnímá „tok“ jasných skvrn, když se LED diody rozsvěcují jedna po druhé a vytvářejí iluzi proudících světél.

Plynulá světla se široce používají v reklamních cedulích, osvětlení jevištních ploch, vánočních dekoracích, zadních světlech automobilů a indikátorech stavu elektronických výrobků. Nejenže mají výrazný dynamický estetický vzhled, ale také dokáží efektivně sdělovat informace a přitahovat pozornost.

S rozvojem interaktivních technologií se v interaktivních zařízeních široce používají i plynulá světla. Světelný efekt je například spouštěn tlačítky, světelnými senzory nebo zvukovými senzory, takže světlo a lidské akce jsou propojeny v reálném čase. Tato forma integrace vizuálního vyjádření a interaktivního zážitku se stala důležitým prvkem v tvůrčích projektech, uměleckých instalacích a návrhu inteligentních systémů.

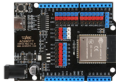
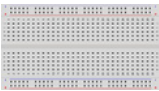


Experiment

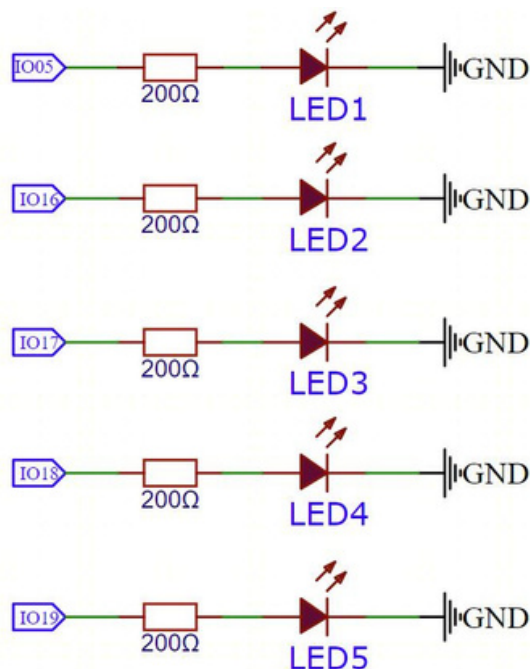
1. Popis projektu

Postupně rozsvěcujte a poté zhasínejte LED diody, abyste vytvořili efekt plynulé barvy.

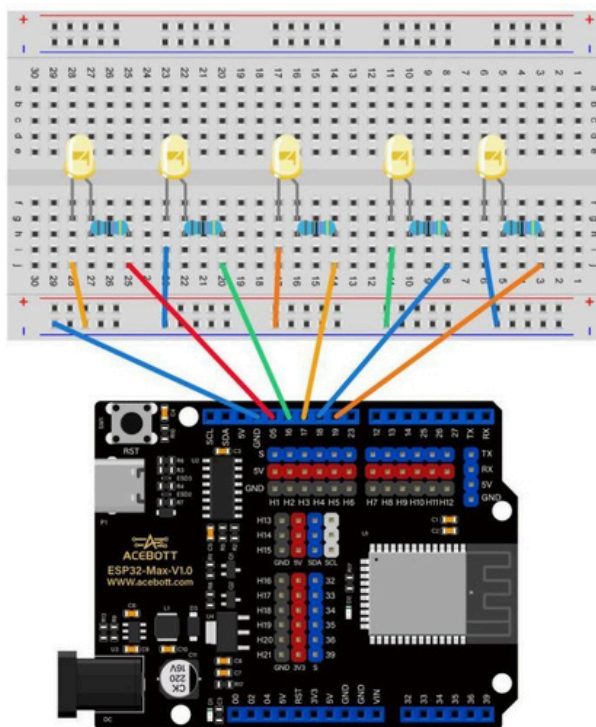
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	5
	Rezistor 220Ω	5

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Running_light.ino](#)" v souboru "English\Arduino(Experienced Learner)\2.Code\lesson 18 Running light\Running_light" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

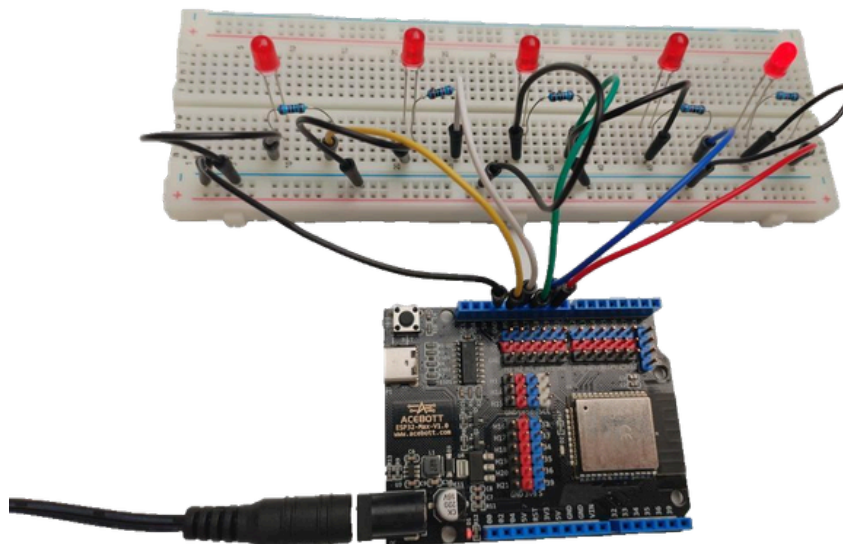
dříve naučené kroky, klikněte  nahrát program.

```
int ledPins[] = {5, 16, 17, 18, 19}; // LED pin numbers

void setup() {
  // Set all LED pins as output
  for (int i = 0; i < 5; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  // Loop through each LED
  for (int i = 0; i < 5; i++) {
    digitalWrite(ledPins[i], HIGH); // Turn on LED
    delay(200); // Wait
    digitalWrite(ledPins[i], LOW); // Turn off LED
  }
}
```

Po úspěšném nahrání programu se LED diody postupně rozsvítí a poté zhasnou, jako plynoucí světlo.



Lekce 19 Kontrolka tlačítka



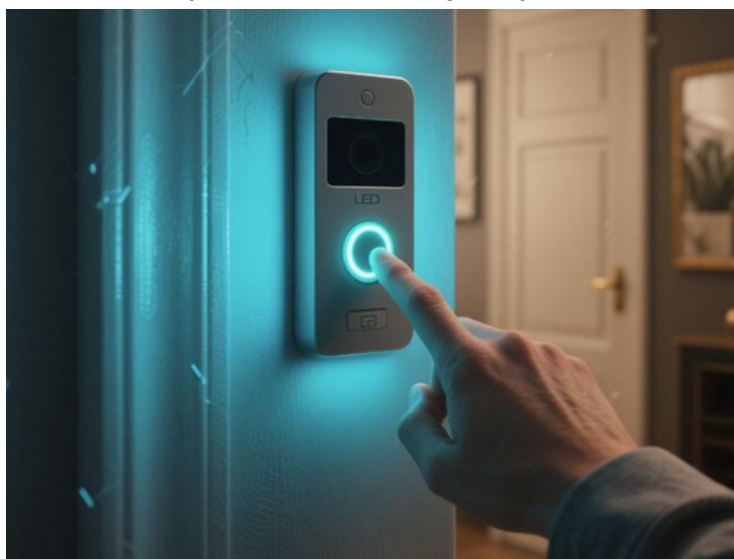
Cíle kurzu

- Pochopíte dva způsoby ovládání tlačítek: okamžité ovládání a ovládání přepínáním stavů
- Zvládněte základní používání příkazů if.
- Zvládněte metodu odskoku tlačítek.
- Zvládněte používání programových instrukcí pro ovládání LED diod pomocí tlačítek



Úvod do kurzu

V každodenním životě často používáme tlačítka k ovládání různých elektrických zařízení, například stisknutím spínače zapneme stolní lampu nebo stisknutím tlačítka na dálkovém ovladači ovládáme televizor. Tyto zdánlivě jednoduché operace jsou ve skutečnosti neoddělitelné od logického ovládání „vstupu tlačítek“ a „výstupu zařízení“.



V této lekci se prostřednictvím dvou experimentů naučíme dva způsoby ovládání LED diod pomocí tlačítek: okamžité ovládání „stisknutím rozsvítíte“,

„uvolněním vypnete“ a ovládání přepínání stavů „stiskněte jednou pro zapnutí, stiskněte znovu pro vypnutí“.



Body znalostí

1. příkaz if

Příkaz if je podmíněný příkaz používaný k určení, zda je výraz pravdivý (true). Pokud je podmínka pravdivá, provede se zadaný kód. Je základem pro implementaci logického větvení v programu.

(1) Výpis z jedné pobočky (pokud)

Jednovětвовá struktura if znamená: pokud je podmínka pravdivá, provést určitý blok kódu; jinak jej přeskočit.

Syntaktická struktura:

```
if(podmínka) {  
    // Kód, který se provede, když je podmínka pravdivá }
```

(2) Příkaz s dvojitou větví (if...else)

Struktura if...else znamená, že když je podmínka pravdivá, provede se jeden blok kódu; jinak se provede další blok kódu, čímž se vytvoří dvě větve.

Syntaktická struktura:

```
if (podmínka) {  
    // Kód, který se provede, když je podmínka pravdivá  
}  
  
else{  
    // Kód, který se provede, když je podmínka nepravdivá  
}
```

(3) Vícevětвовá struktura (pokud...jinak pokud...jinak)

Používá se k vyhodnocení více podmínek. Vyhodnocuje m podmínek postupně a jakmile je splněna první podmínka, provede se odpovídající kód a vyhodnocování se zastaví.

Syntaktická struktura:

```
if (condition1) {  
    // Executed when condition1 is true  
} else if (condition2) {  
    // Executed when condition2 is true  
} else if (condition3) {  
    // Executed when condition3 is true  
} else {  
    // Executed when none of the above conditions are true  
}
```

2 Podmínky úsudku

Podmínky úsudku se vztahují k výrazům používaným k vyhodnocení pravdivosti nebo nepravdy a vrácení booleovské hodnoty (pravda nebo nepravda). Mezi běžné podmínky úsudku patří:

Porovnávací operace: == (rovná se), != (nerovná se), > (větší než), < (menší než), >= (větší nebo rovno), <= (menší nebo rovno)

Například: $a > 10$, $x == y$, $teplota \leq 30$

Logické operace: && (a), || (nebo), ! (ne)

Například: $x > 0 \ \&\& \ x < 100$, `!buttonPressed`

3. Kontrolka tlačítka

Tlačítkové světlo je obvodový systém, který zapíná nebo vypíná LED diodu stisknutím tlačítka.

Princip fungování tlačítka (1)

Logika okamžitého řízení

Logika okamžitého ovládání je metoda ovládání typu „stiskněte pro spuštění, uvolnění pro zastavení“. Po stisknutí tlačítka se LED dioda okamžitě rozsvítí a po uvolnění tlačítka LED dioda okamžitě zhasne. Reaguje rychle a je vhodná pro aplikace, které vyžadují okamžitou zpětnou vazbu, jako jsou například zvonky u dveří.

(2) Logika přepínání stavů

Ovládání přepínáním stavů je metoda ovládání „přepínáním jedním kliknutím“, vhodná pro aplikace, jako jsou stolní lampy a vypínače. Pokaždé, když je stisknuto tlačítko, LED dioda přepíná mezi „zapnuto“ a „vypnuto“. Protože mechanická tlačítka při stisknutí nebo uvolnění chvějí, může být jedno stisknutí klávesy mylně rozpoznáno jako více stisknutí kláves. Proto je v programu vyžadováno zpracování pro potlačení odskoků (například detekce zpoždění), aby se zajistilo, že každé stisknutí klávesy spustí pouze jedno přepnutí, což zlepšuje spolehlivost a stabilitu ovládání.

Princip a metoda odskoku:

Abychom se vyhnuli chybnému odhadu, musíme po detekci změny stisknutí klávesy „krátkou dobu počkat“, než ověříme, zda byla klávesa skutečně stisknuta. Tato metoda dokáže efektivně filtrovat krátké chvění mechanické klávesy během fyzického kontaktu a zajistit, aby program reagoval na platnou operaci klávesy pouze jednou.

Formát detekce tlačítek:

```
if(digitalRead(buttonPin)==LOW){//First press detected
delay(200);//Wait for the debounce to end.
while(digitalRead(buttonPin)==LOW);//Wait for the button to be released
(pressed up).
}
```



Experiment

Experiment 1: Okamžité ovládání osvětlení tlačítka

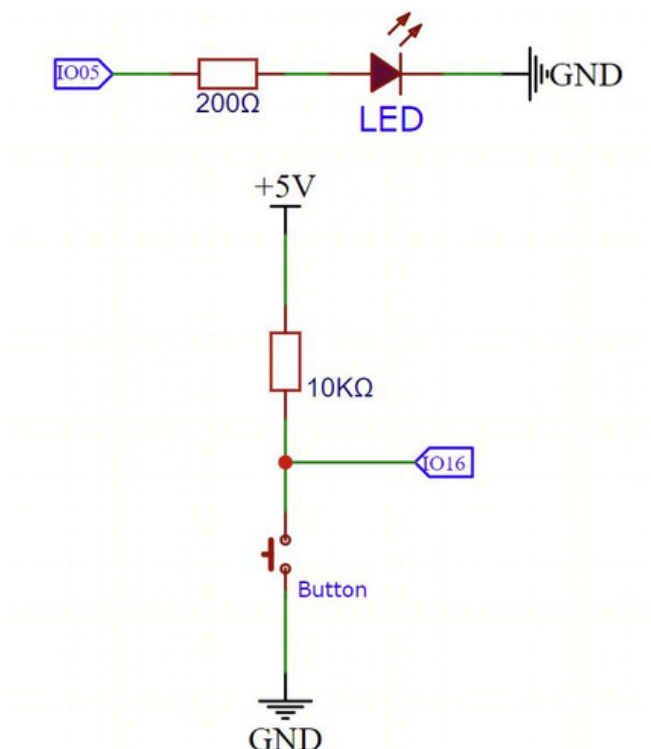
1. Popis projektu

Stisknutím tlačítka se LED dioda zapíná a vypíná, uvolněním se vypíná.

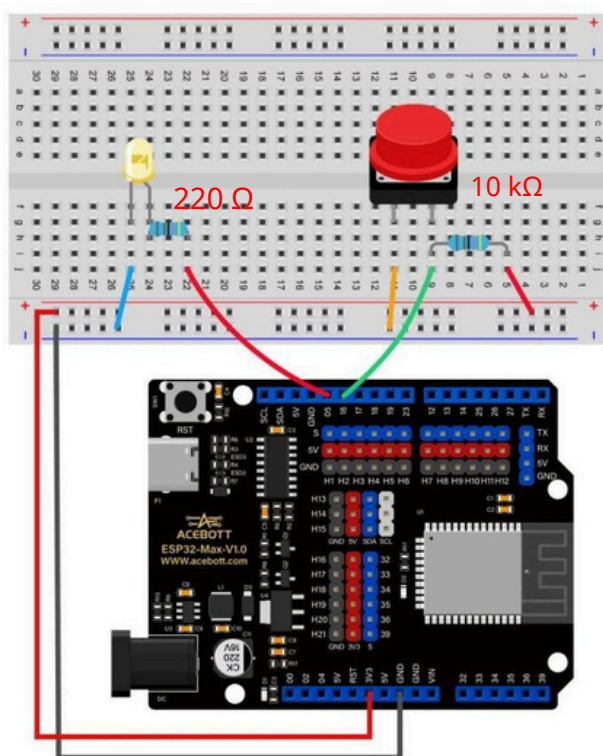
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Tlačítko	1
	Rezistor 220Ω	1
	Rezistor 10 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Button_light_1.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 19 Button light\Button_light_1" nebo zkopírujte

následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

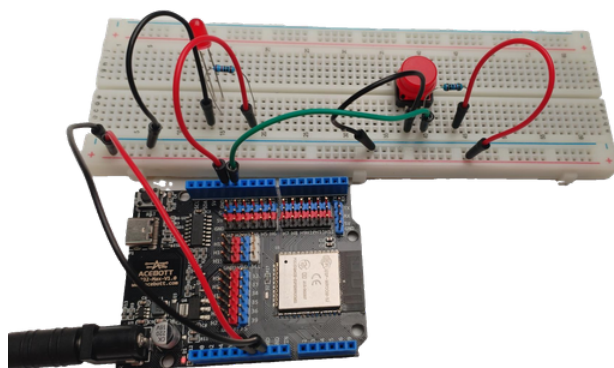
```
#define ledPin 5
#define buttonPin 16

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);
  } else if (buttonState == HIGH){
    digitalWrite(ledPin, LOW);
  }
}
```

Po úspěšném nahrání programu stisknutím tlačítka LED diodu zapnete a vypnete, uvolněním ji vypnete.



Experiment 2: Ovládání světla tlačítka

přepínačem 1. Popis projektu

Stisknutím tlačítka se LED dioda přepíná mezi zapnutím a vypnutím, přičemž se její stav jednou změní.

2. Referenční program

Otevřete "[Button_light_2.ino](#)" soubor v "English\Arduino(Experienced

Learner) \2.Code\lesson 19 Podsvícení tlačítka\Button_light_2" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

```
#define ledPin 5
#define buttonPin 16

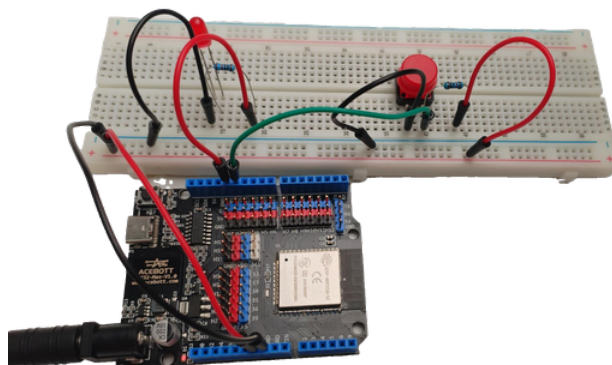
// Initialize the LED state: false = OFF, true = ON
bool light_state = false;

void setup() {
  pinMode(ledPin, OUTPUT); // Set pin 5 as output mode
  pinMode(buttonPin, INPUT); // Set pin 32 as input mode
}

void loop() {
  if (digitalRead(buttonPin) == 0) { // Check if the button is pressed
    delay(200); // Delay 0.2s to improve detection accuracy (debounce)
    while (digitalRead(buttonPin) == 0); // Wait until the button is released
    light_state = !light_state; // Toggle the LED state
  }

  if (light_state) {
    digitalWrite(ledPin, HIGH); // Turn the LED ON
  } else {
    digitalWrite(ledPin, LOW); // Turn the LED OFF
  }
}
```

Po úspěšném nahrání programu se každým stisknutím tlačítka LED zapne nebo vypne.



Lekce 20 Lidský senzor Světlo



Cíle kurzu

- Pochopíte princip fungování světla senzoru lidského těla
- Ovládněte používání příkazů if pro automatické ovládání světel se senzory pohybu.
- Zvládněte metodu programování pro implementaci funkce osvětlení senzoru lidského těla



Úvod do kurzu

V našich životech se automatická senzorová světla hojně používají, například senzorová světla na chodbách, automatická světla na veřejných toaletách a bezpečnostní osvětlení v chytrých domech. Dokážou se automaticky rozsvítit, když někdo projde kolem, a automaticky zhasnout, když nikdo není v okolí, což je energeticky úsporné a pohodlné.



V této lekci se naučíme, jak detekovat lidský pohyb pomocí pohybového senzoru a implementovat jednoduchý systém ovládání osvětlení s detekcí pohybu pomocí programu.



Body znalostí

1. Světlo s detekcí pohybu

Světlo s pohybovým senzorem je inteligentní osvětlovací zařízení, které automaticky snímá lidskou aktivitu a zapíná a vypíná světlo. Obvykle používá infračervený (PIR) senzor nebo mikrovlnný radarový senzor k detekci pohybu lidí nebo zvířat. Když se někdo přiblíží nebo se pohybuje v dosahu snímání, světlo se automaticky rozsvítí; pokud se nikdo po určitou dobu nepohne, světlo se automaticky vypne.

Jak fungují světla s detekcí pohybu

Světlo s pohybovým senzorem využívá technologii infračerveného snímání, která detekuje, zda osoba vstoupí do snímané oblasti, pomocí vestavěného PIR (pasivní infračervený) senzoru. Když se osoba přiblíží, povrch těla nepřetržitě vyzařuje infračervené světlo o vlnové délce přibližně 10 μm . Dva pyroelektrické prvky v PIR senzoru detekují změnu infračerveného záření. Pokud osoba vstoupí do snímací oblasti, intenzita infračerveného záření detekovaná těmito dvěma prvky se stane nevyváženou, což má za následek rozdíl elektrických signálů. Tento rozdílový signál je zesílen vnitřními obvody a vydáván jako digitální signál (vysoká úroveň indikuje přítomnost, nízká úroveň indikuje nepřítomnost). Když řídicí jednotka ESP32 načte tuto vysokou úroveň, okamžitě rozsvítí LED diodu nebo aktivuje relé pro rozsvícení světla. Pokud po určitou dobu není detekována žádná lidská aktivita, světlo se automaticky vypne.

Světla s pohybovými senzory se široce používají v místech, jako jsou chodby, chodby, schodiště, kanceláře, koupelny a vchody do domů. Automaticky se rozsvítí, když někdo projde kolem, a poskytují tak pohodlné osvětlení a zároveň zabraňují delšímu rozsvícení světla.

zlepšení energetické účinnosti.

Poznámka: Infračervený senzor pro lidskou přítomnost detekuje změny infračerveného záření v prostředí a detekuje tak přítomnost osoby.

Pokud osoba stojí, senzor nedokáže detekovat změny záření a může mylně detekovat nepřítomnost osoby.

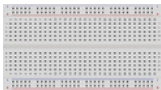


Experiment

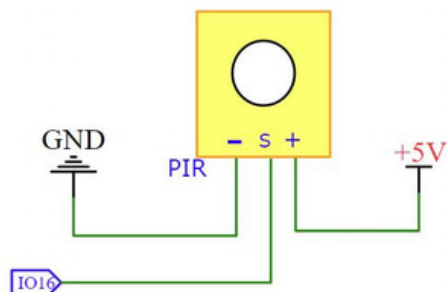
1. Popis projektu

Když je detekována lidská aktivita, LED se automaticky rozsvítí; když osoba odejde, LED se po nastavené prodlevě automaticky vypne.

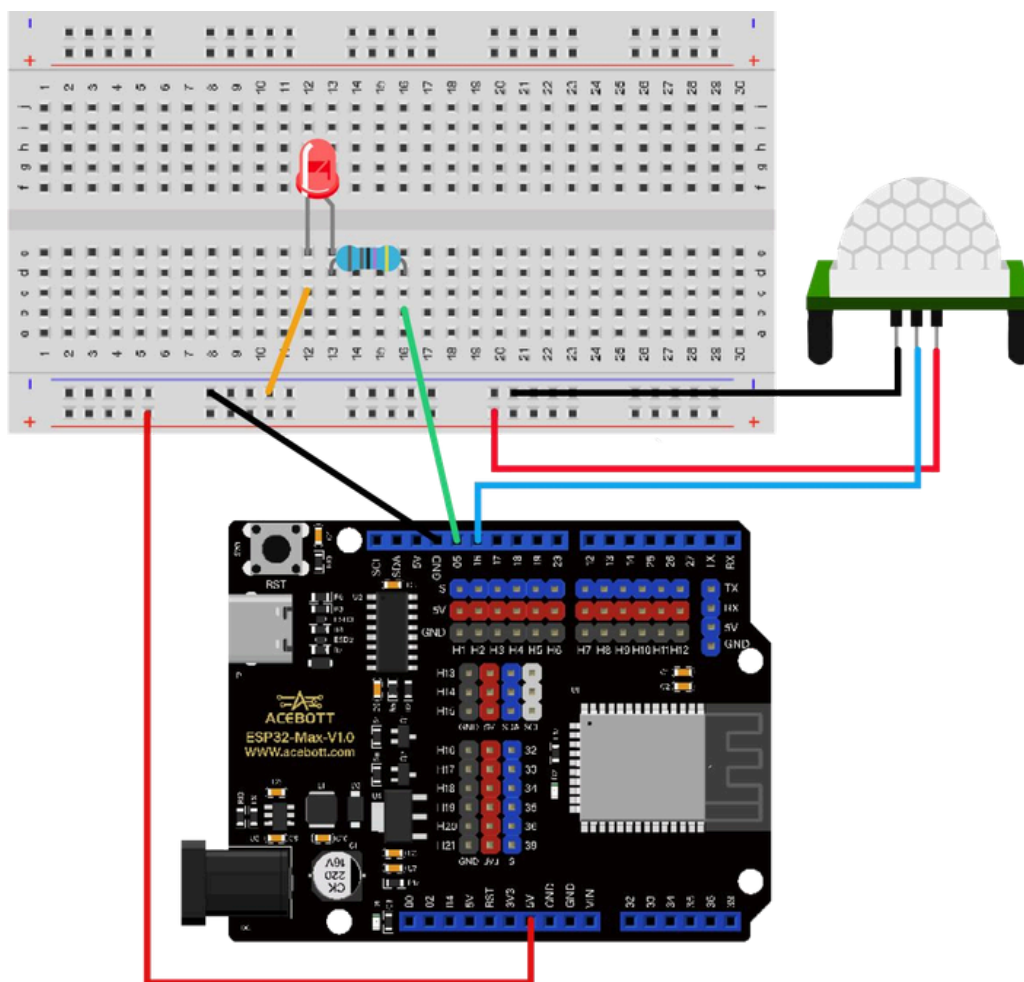
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	PIR senzor	1
	LED	1
	Rezistor 220Ω	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[PIR_Sensor_Light.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 20 PIR Sensor Light\PIR_Sensor_Light" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojového souboru

deska (vývojový modul ESP32) a příslušný port podle

dříve naučené kroky, klikněte  nahrát program.

```
#define LED_PIN 5 // LED control pin
#define PIR_PIN 16 // PIR sensor output pin

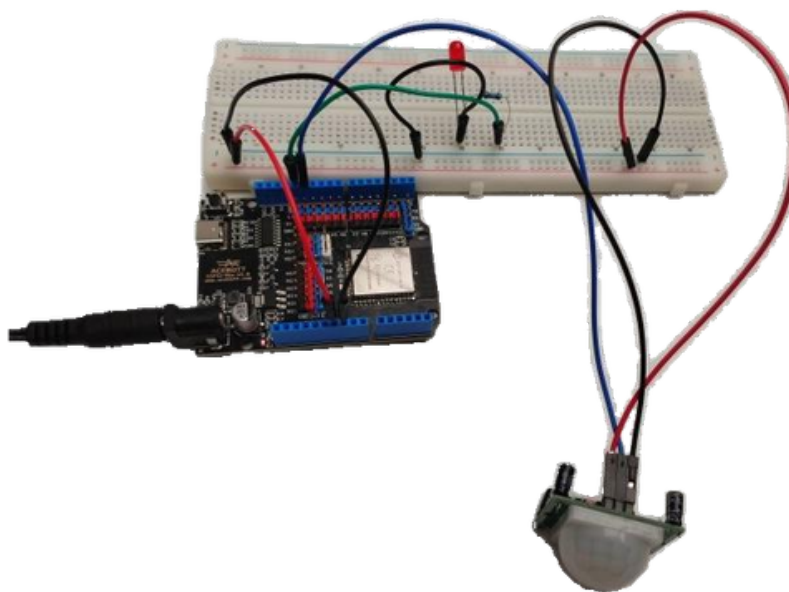
void setup() {
  pinMode(PIR_PIN, INPUT); // Set PIR pin as input
  pinMode(LED_PIN, OUTPUT); // Set LED pin as output
  digitalWrite(LED_PIN, LOW); // LED starts OFF
}

void loop() {
  int pirState = digitalRead(PIR_PIN); // Read PIR sensor

  if (pirState == HIGH) {
    digitalWrite(LED_PIN, HIGH); // Turn ON LED when motion detected
  } else {
    digitalWrite(LED_PIN, LOW); // Turn OFF LED when no motion
  }

  delay(50); // Optional small delay to avoid rapid flickering
}
```

Po úspěšném nahrání programu se LED dioda rozsvítí, když je detekován pohyb osoby. Pokud není detekován žádný pohyb osoby, LED dioda zhasne.



Lekce 21 Světlo řízené světlem



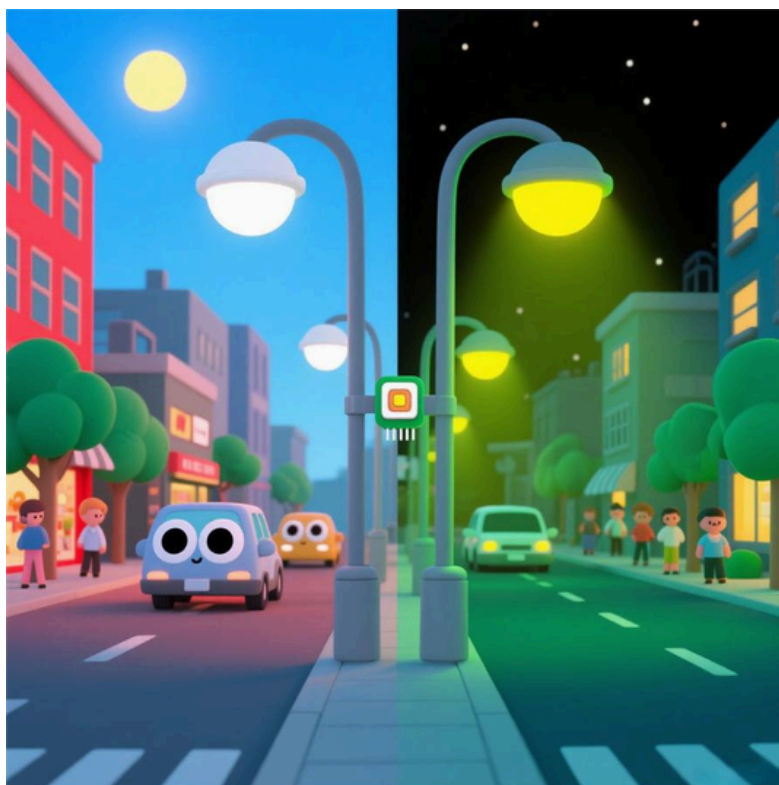
I. Cíle kurzu

- Pochopení fungování světel řízených světlem
- Zvládnutí programování pro implementaci světelně řízených světelných funkcí



II. Úvod do kurzu

Světla ovládaná světlem se v našem každodenním životě široce používají. Například pouliční lampy se mohou automaticky zapínat a vypínat na základě intenzity okolního světla: zhasínají, když je během dne dostatek světla, a rozsvěcují se, když je v noci světla nedostatek.



V této lekci se naučíme, jak pomocí světelného senzoru detekovat okolní jas a programově ovládat LED světlo, a implementovat tak automatizované osvětlení řízené světlem.



III. Klíčové body

1. Světla řízená světlem

Světla ovládaná světlem jsou osvětlovací zařízení, která se automaticky zapínají a vypínají na základě intenzity okolního světla, rozsvěcují se, když je tma, a zhasínají, když je světlo.

Jak fungují světla řízená světlem

Pro detekci intenzity okolního světla použijte fotorezistor (prvek citlivý na světlo). Když se prostředí ztmaví, odpor fotorezistoru se zvýší, což způsobí změnu napětí v obvodu. Řídicí jednotka ESP32 čte tento napěťový signál a převádí jej na odpovídající hodnotu jasu. Když je intenzita světla pod přednastavenou prahovou hodnotou doby zapnutí, světlo se automaticky zapne; naopak, když je intenzita světla nad prahovou hodnotou doby vypnutí, světlo se automaticky vypne. Tímto způsobem dosahují lampy řízené světlem inteligentního řízení, automatického zapínání a vypínání podle okolních světelných podmínek, čímž se dosahuje úspory energie a pohodlí.

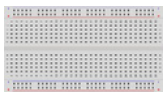
Světla řízená světlem se široce používají ve venkovním zahradním osvětlení, pouličním osvětlení, osvětlení chodeb a veřejných prostorách, kde je vyžadováno automatické nastavení osvětlení na základě přirozeného světla. To efektivně snižuje ruční obsluhu, zlepšuje energetickou účinnost a zvyšuje pohodlí uživatele.



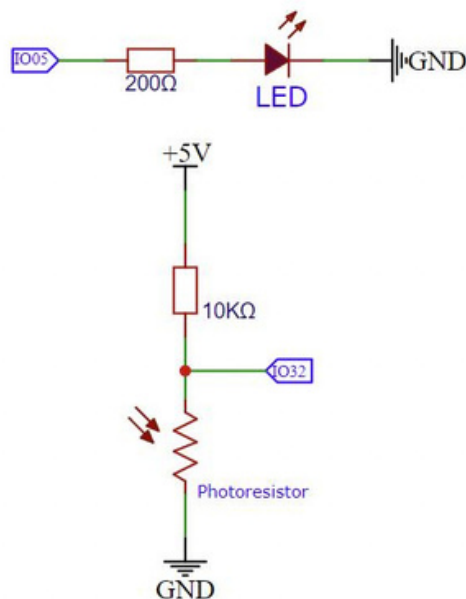
IV. Experiment

1. Popis projektu Pro měření intenzity okolního osvětlení použijte fotorezistor s rozsahem čtení od 0 do 4095. Pokud je intenzita světla nižší než 2000, LED světlo se automaticky rozsvítí; pokud je intenzita světla vyšší než 2000, LED světlo se automaticky vypne, čímž se zavede automatické zapínání a vypínání světla podle okolních světelných podmínek.

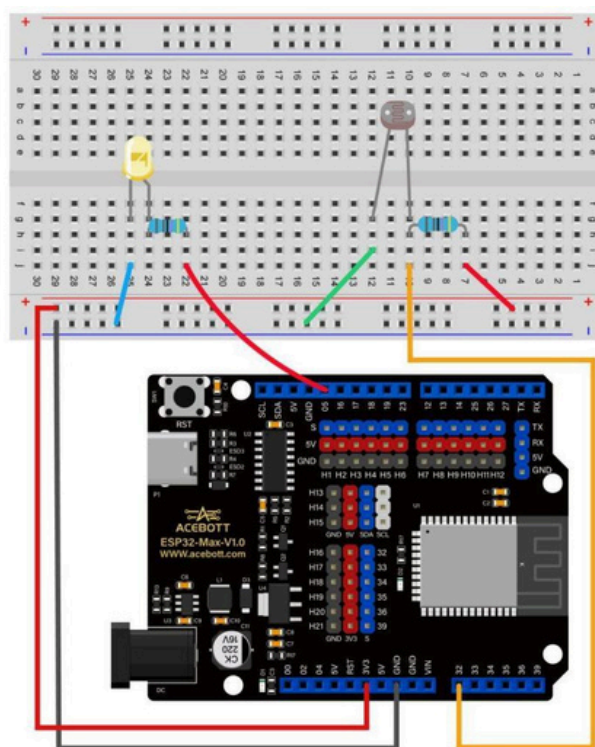
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Fotorezistor	1
	Rezistor 220Ω	1
	Rezistor 10 kΩ	1
		1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Light controlled lamp.ino](#)"

soubor

V

"Angličtina

\\Arduino (zkušený student) \\2.Kód\\lekce 21 Lampa ovládaná světlem\\Light_controlled_lamp" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a

příslušný port podle dříve naučených kroků klikněte na program.



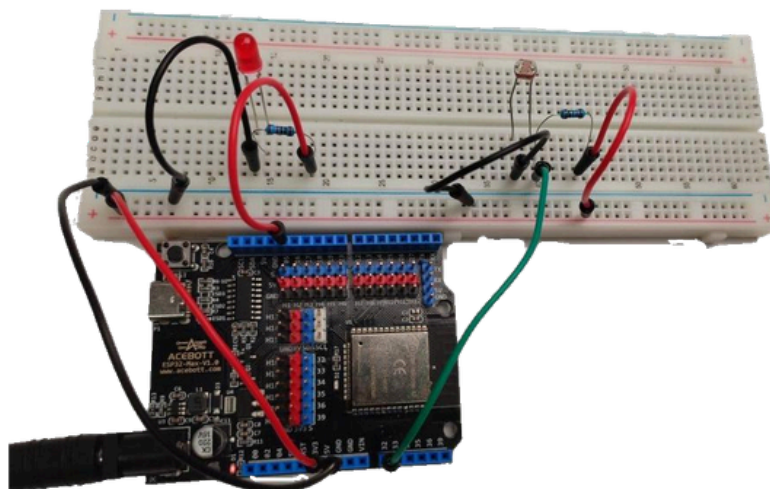
nahrát

```
#define LDR_PIN 32 // ADC pin for LDR sensor
#define LED_PIN 5 // Digital output pin for LED

void setup() {
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW); // Start with LED OFF
  Serial.begin(115200); // Initialize serial monitor
}

void loop() {
  //The darker it gets, the higher the value.
  int ldrValue = analogRead(LDR_PIN); // Read LDR sensor value
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);
  if (ldrValue >= 2000) {
    // It's dark, turn ON the LED
    digitalWrite(LED_PIN, HIGH);
    Serial.println("It's dark - LED ON");
  } else {
    // It's bright, turn OFF the LED
    digitalWrite(LED_PIN, LOW);
    Serial.println("It's bright - LED OFF");
  }
  delay(100); // Short delay
}
```

Po úspěšném nahrání programu se LED světlo vypne při jasném světle a rozsvítí se při tmě.



Lekce 22 Úprava barev světla



Cíle kurzu

- Pochopení principů barevně upravovaného světla.
- Napsání programu pro implementaci barevně upravovaného světla.



Úvod do kurzu

V každodenním životě často vidíme osvětlovací zařízení, která automaticky mění barvu na základě různých atmosfér nebo potřeb, jako je například ambientní osvětlení ložnice a barevná světla na audio systémech. Tato barevná světla jsou nejen krásná, ale také zlepšují interaktivitu a uživatelský zážitek.



V této lekci se naučíme, jak vyrobit jednoduchou a zábavnou „lampu s možností nastavení barev“. Jednoduše otočte potenciometrem a ovládejte RGB světlo, přepínejte mezi několika přednastavenými barvami a vytvářejte personalizované světelné efekty.



Body znalostí

1. Jak fungují světla pro úpravu barev

Lampa s regulací barev využívá princip míchání světla RGB k vytváření různých světelných efektů úpravou poměru jasu červených, zelených a modrých LED diod. Na základě tohoto principu má systém několik přednastavených barevných kombinací a jako vstupní zařízení používá potenciometr. Po otočení potenciometru regulátor odečítá výstupní napětí a přepíná barvu světla podle rozsahu napětí, což umožňuje jednoduché a intuitivní personalizované ovládání osvětlení.

Světla s regulací barev se široce používají v inteligentním domácím ambientním osvětlení, osvětlení pódíí, dekorativním osvětlení stolů, kreativních elektronických dílech a dalších aplikacích. Prostřednictvím jednoduchých interaktivních operací mohou uživatelé přepínat mezi různými barvami, čímž vylepšují okolní atmosféru a zlepšují vizuální zážitek.

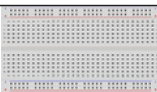


Experiment

1. Popis projektu

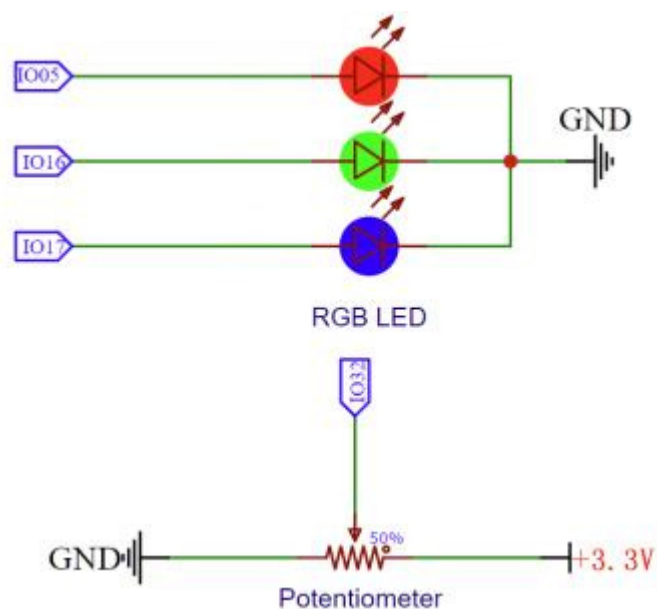
Otáčením potenciometru ovládáte RGB světlo a zobrazujete sedm přednastavených barev, čímž dosáhnete přepínání a zobrazování barev.

2. Seznam hardwaru

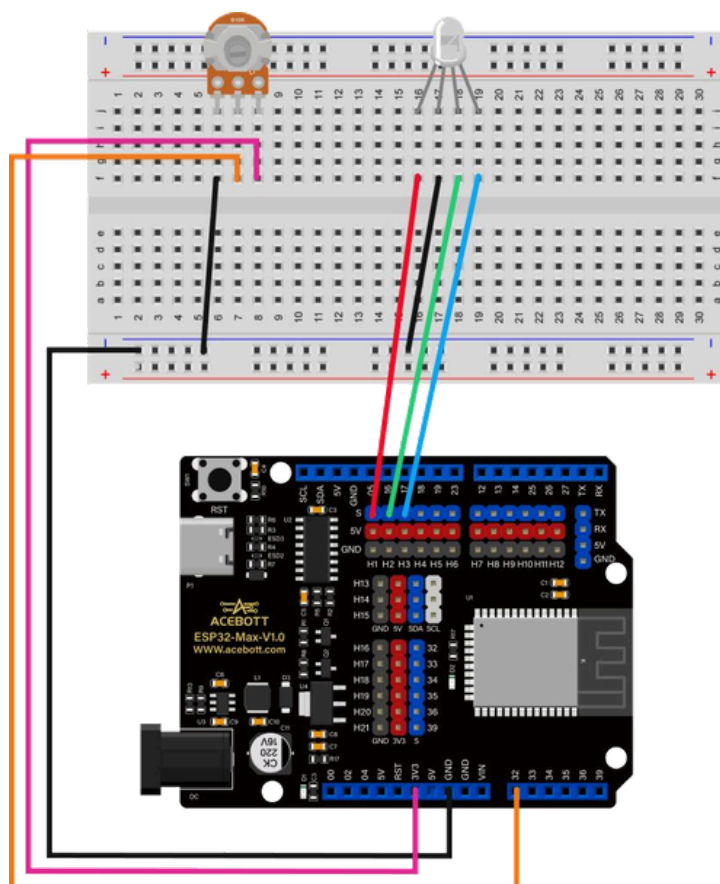
Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Potenciometr	1

	RGB LED	1
	Rezistor 220Ω	3

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Color_adjustable_light.ino](#)" „Arduino(Zkusoušbeorný student)

\\2.Code\\lesson 22 Barevně nastavitelné světlo\\Color_adjustable_light"

nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a

příslušný port podle dříve naučených kroků klikněte na



nahrát

```
#define POT_PIN 32 // Potentiometer connected to analog pin GPIO32
#define RED_PIN 16 // Red LED PWM pin
#define GREEN_PIN 17 // Green LED PWM pin
#define BLUE_PIN 18 // Blue LED PWM pin

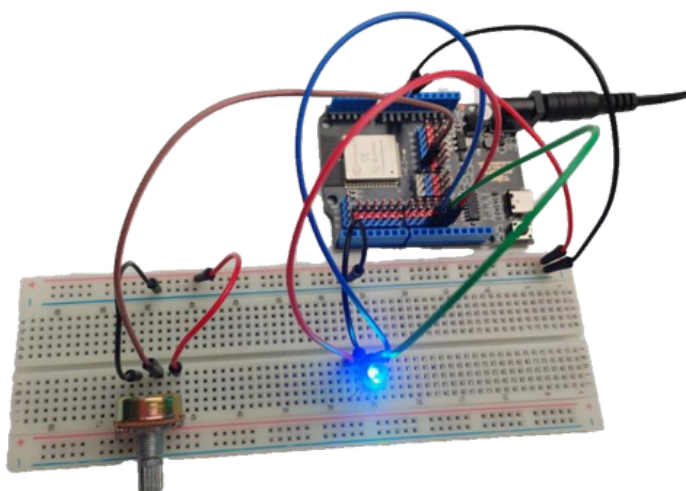
int r = 0, g = 0, b = 0;
void setup() {
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop() {
  int potValue = analogRead(POT_PIN); // Read analog value from potentiometer (0~4095)
  int level = map(potValue, 0, 4095, 0, 6); // Map to 7 levels: 0 to 6
  // Set RGB values based on the current level
  if (level == 0) { // Black (off)
    r = 0; g = 0; b = 0;
  } else if (level == 1) { // Red
    r = 255; g = 0; b = 0;
  } else if (level == 2) { // Green
    r = 0; g = 255; b = 0;
  } else if (level == 3) { // Blue
    r = 0; g = 0; b = 255;
  } else if (level == 4) { // Purple (Red + Blue)
    r = 128; g = 0; b = 128;
  } else if (level == 5) { // Yellow (Red + Green)
    r = 255; g = 255; b = 0;
  } else if (level == 6) { // White (Red + Green + Blue)
    r = 255; g = 255; b = 255;
  }

  // Write PWM values to RGB pins
  analogWrite(RED_PIN, r);
  analogWrite(GREEN_PIN, g);
}
```

```
analogWrite(BLUE_PIN, b);  
}  
  
delay(100); // Short delay to prevent flicker  
}
```

Po úspěšném nahrání programu otočte potenciometrem a barva RGB LED se začne měnit.



Lekce 23 Elektronické kostky



Cíle kurzu

- Pochopíte princip fungování elektronických kostek.
- Zvládněte generování a používání náhodných čísel.
- Napište program, který implementuje funkce elektronických kostek



Úvod do kurzu

V našem každodenním životě jsou kostky velmi běžnými herními rekvizitami, které se používají ke generování náhodných bodů, jež určují výsledek hry nebo další tah. V této lekci se naučíme, jak vyrobit jednoduché a zajímavé elektronické kostky. Stačí stisknout tlačítko a čísla 1 až 6 se náhodně objeví na 8segmentové digitální trubici.



Prostřednictvím tohoto projektu si nejen užijete zábavu s elektronickými technologiemi, ale také se naučíte, jak implementovat generování náhodných čísel a ovládání tlačítky. Pojdte si společně zahrát s elektronickými kostkami!



Body znalostí

1. Elektronické kostky

Elektronické kostky jsou inteligentní zařízení, která využívají elektronické obvody k simulaci tradičních kostek. Po stisknutí tlačítka LED dioda náhodně zobrazí tečkovaný vzor čísel 1 až 6, čímž simuluje proces hodu kostkou.

**Princip fungování elektronických kostek:**

(1) SpoušťHráč stiskne tlačítko, aby systém spustil akci „házení kostkou“.

(2) Generování náhodných číselŘídící deska ESP32 používá program k vygenerování náhodného celého čísla mezi 1 a 6, které simuluje počet bodů na kostkách.

(3) Zobrazení bodůPoužijte 8segmentovou digitální trubici k zobrazení náhodných bodů.

2. Náhodná čísla

Náhodná čísla se používají ke generování nepředvídatelných hodnot v určitém rozsahu, simulujíc náhodné události v přírodě. Ve výrobě a programování elektroniky se náhodná čísla často používají k implementaci různých funkcí, jako je simulace hodu kostkou, losování, spouštění náhodných událostí ve hrách a rozhodování robotů o jejich akcích. Pomocí náhodných čísel mohou zařízení vykazovat rozmanité a nepředvídatelné chování, což zvyšuje zajímavost a praktičnost.

Náhodné semeno

Generátor náhodných čísel ve skutečnosti vytváří pomocí matematických výpočtů posloupnost čísel, která se „zdají být náhodná“ – to jsou

nazývají se pseudonáhodná čísla. Jejich počáteční bod je určen číslem známým jako „náhodná hodnota“. Pokud se při každém spuštění programu použije stejná hodnota, vygenerovaná sekvence náhodných čísel bude přesně stejná. Abychom zajistili, že čísla generovaná pokaždé budou odlišná, obvykle používáme jako náhodnou hodnotu elektrické signály (drobné rušivé šумы z prostředí) čtené z nezapojeného analogového pinu. Díky tomu jsou náhodná čísla realističtější a obtížněji předvídatelná, čímž se vyhneme opakujícím se sekvencím. Šablona pro Arduino pro generování náhodných čísel:

```
void setup() {  
  // Initialize the random seed using noise from an unconnected analog pin  
  randomSeed(analogRead(pin));  
}  
  
void loop() {  
  int randNum = random(min, max); // Generate a random number between min  
  and max-1  
}
```

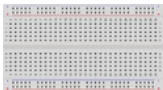


Experiment

1. Popis projektu

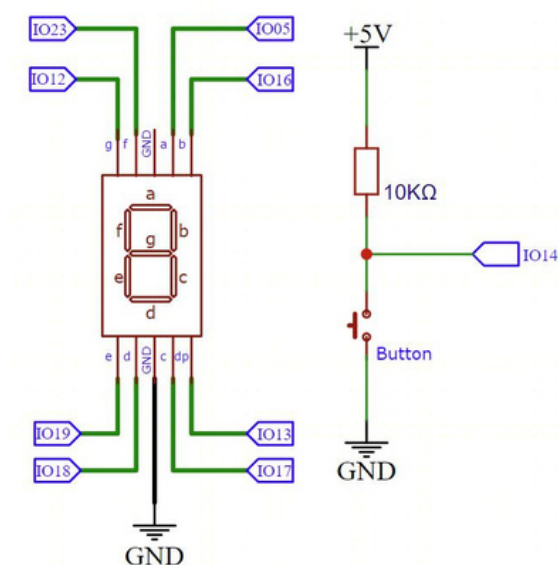
Po stisknutí tlačítka řídicí deska ESP32 náhodně vygeneruje číslo mezi 1 a 6 a zobrazí ho na 8segmentové digitální trubici.

2. Seznam hardwaru

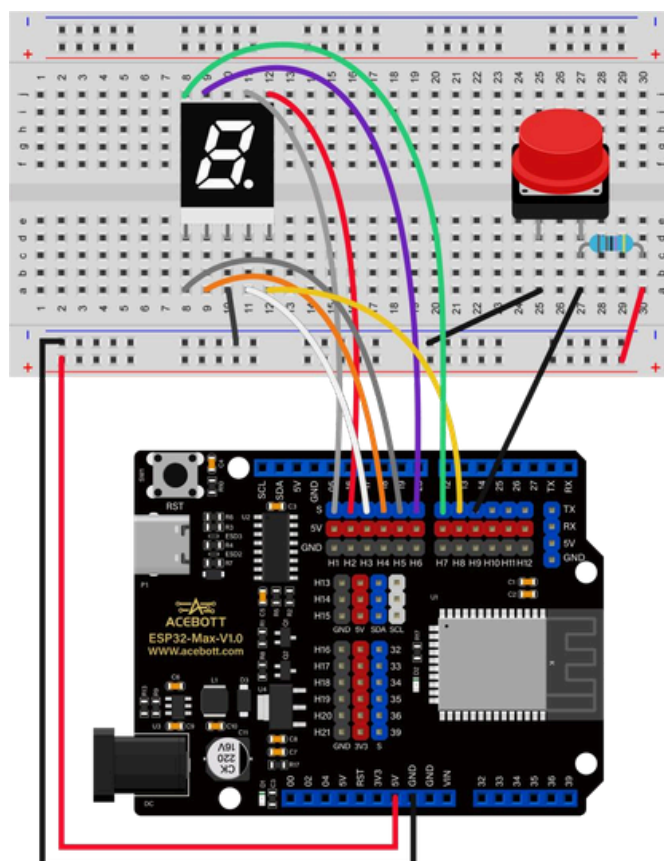
Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1

	8segmentový displej	1
	Tlačítko	1
	Rezistor 10 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Electronic Dice.ino](#)" soubor v souboru „English\Arduino(Experienced Learner)\2.Code\lesson 23 Electronic Dice\Electronic_Dice“ nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů dříve naučené kroky, klikněte  nahrát program.

```
// Define the pins connected to the 7-segment display segments
int pin_a = 5;
int pin_b = 16;
int pin_c = 17;
int pin_d = 18;
int pin_e = 19;
int pin_f = 23;
int pin_g = 12;
int pin_dp = 13;

// Store the pin numbers in an array for easier control
int pin_array[8] = {pin_a, pin_b, pin_c, pin_d, pin_e, pin_f, pin_g, pin_dp};

// Define the segment patterns for digits 0-9 (1 = ON, 0 = OFF)
// Order: a, b, c, d, e, f, g, dp
int number_array[][8] = {
{1, 1, 1, 1, 1, 1, 0, 0}, // 0
{0, 1, 1, 0, 0, 0, 0, 0}, // 1
{1, 1, 0, 1, 1, 0, 1, 0}, // 2
{1, 1, 1, 1, 0, 0, 1, 0}, // 3
{0, 1, 1, 0, 0, 1, 1, 0}, // 4
{1, 0, 1, 1, 0, 1, 1, 0}, // 5
{1, 0, 1, 1, 1, 1, 1, 0}, // 6
{1, 1, 1, 0, 0, 0, 0, 0}, // 7
{1, 1, 1, 1, 1, 1, 1, 0}, // 8
{1, 1, 1, 1, 0, 1, 1, 0} // 9
};

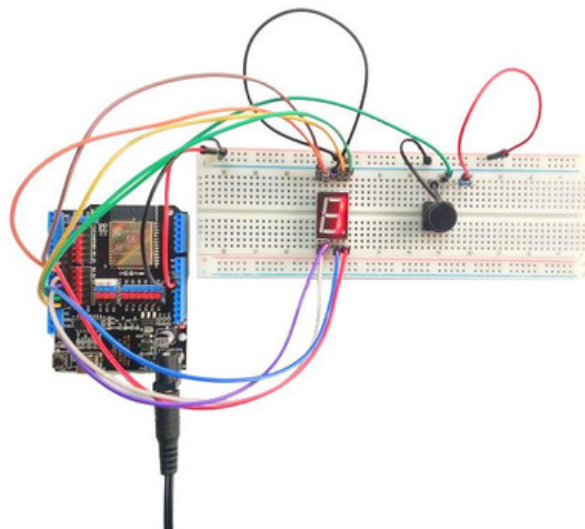
// Define the button pin
#define BUTTON_PIN 14

int currentNumber = 0; // Store the currently displayed number

// Function to display a digit (0-9) on the 7-segment display
void display_number(int num) {
if (num < 0 || num > 9) return; // Check for valid digit
for (int i = 0; i < 8; i++) {
digitalWrite(pin_array[i], number_array[num][i]); // Set segment states
```

```
}  
}  
  
void setup() {  
  // Initialize all segment pins as OUTPUT and turn them OFF  
  for (int i = 0; i < 8; i++) {  
    pinMode(pin_array[i], OUTPUT);  
    digitalWrite(pin_array[i], 0); // All segments OFF  
  }  
  pinMode(BUTTON_PIN, INPUT); // Set button pin as input  
  // Seed the random number generator using an unconnected analog pin  
  randomSeed(analogRead(34));  
}  
  
void loop() {  
  // Check if the button is pressed (active LOW)  
  if (digitalRead(BUTTON_PIN) == LOW) {  
    delay(200); // Debounce delay  
    while(digitalRead(BUTTON_PIN) == LOW);  
    currentNumber = random(1, 7); // Generate random number 1-6  
    displayNumber(currentNumber); // Show it on 7-segment display  
  }  
}
```

Po úspěšném nahrání programu stiskněte tlačítko a 8segmentový displej náhodně zobrazí číslo (1-6), jak je znázorněno níže:



6. Popis instrukcí

randomSeed(seed): Používá se k nastavení počátečního bodu pro náhodné číslo generování náhodných čísel, což zajišťuje, že při každém spuštění programu jsou generována různá náhodná čísla. Seed je počáteční hodnota pro náhodné číslo

generátor a obvykle se odebírá z nepřipojeného analogového vstupního pinu, aby se získala náhodnější počáteční hodnota.

random(min, max): Generuje náhodné celé číslo mezi min (včetně) a max-1 (včetně).

Lekce 24 Počítadlo



Cíle kurzu

- Pochopení fungování čítačů
- Napsání programu pro implementaci čítače



Úvod do kurzu

Počítadla hrají zásadní roli v našem každodenním životě a elektronických zařízeních. Ať už počítáme časy, měříme čas nebo řídíme provoz zařízení, čítače nám pomáhají přesně zaznamenávat a spravovat různé události.



V této lekci se naučíme, jak navrhnout jednoduchý čítač. Tři tlačítka se používají k navyšování, snižování a vynulování čísel a k zobrazení výsledku se používá čtyřmístný 8segmentový displej. Zvládnete základní principy ovládání klávesami a digitálního zobrazení a zažijete zábavu s elektronickým počítáním. Pojdme se společně vydat na tuto zajímavou cestu počítáním!



Body znalostí

1. Počítadlo

Čítač je elektronické nebo programové logické zařízení používané k zaznamenávání počtu výskytů události. Dokáže automaticky zvýšit, snížit nebo vynulovat hodnotu pokaždé, když je spuštěn externí signál (například tlačítko).

Jak funguje počítadlo

(1) Ovládání tlačítkem Tyto tři klávesy implementují funkce „přičíst jednu“, „odčíst jednu“ a „vymazat“. Když uživatel stiskne odpovídající klávesu, systém provede odpovídající operaci počítání.

(2) Zpracování počítání Řídící deska ESP32 upravuje hodnotu podle signálu z klávesy, provádí +1, -1 nebo nulování a zajišťuje, aby rozsah počítání nepřekročil přednastavený limit (například 0 až 9999).

(3) Digitální displej: K zobrazení aktuální hodnoty počítadla v reálném čase se používá čtyřmístná osmisegmentová digitální trubice.

Počítadla se široce používají v různých elektronických zařízeních, jako jsou krokoměry, zapisovače rychlosti a herní výsledkové tabule.

2. Operace se zbytkem (%)

Operace modulo je matematická operace používaná k nalezení zbytku po dělení dvou čísel. V programovacích jazycích se k vyjádření operace modulo obvykle používá symbol %.

Například: $1234 \div 10 = 123$ se zbytkem 4

Tuto operaci se zbytkem můžeme pochopit dělením jablka.

Předpokládejme, že máme čtyři jablka pro čtyři osoby a je jich tak akorát

aby každá osoba dostala jedno, takže nezbyvá žádné a zbytek je 0.

Předpokládejme, že čtyři lidé si rozdělí pět jablek, každý s jedním, zbývá jedno jablko a zbytek je 1. Předpokládejme, že čtyři lidé si rozdělí dvě jablka a pro každou osobu není dostatek jablek, takže se nerozdělí žádná jablka, zbývají dvě jablka a zbytek je 2.

3. Čtyřmístné oddělení

Čtyřmístné dělení je proces, kterým se čtyřmístné celé číslo (například 1234) rozdělí na tisíce, stovky, desítky a jednočíslice. Používá se pro digitální zobrazování, takže každá digitální zobrazovací trubice zobrazuje odpovídající číslo.

Vezměme si jako příklad číslo 1234 a metoda oddělení jednotlivých číslic je následující:

Jednotky: Vydělte číslo 10 a zbytek, výsledkem je 4, tj. $1234 \% 10 = 4$

Desítky: Vydělte číslo 10 a poté z něj vypočítejte zbytek, výsledkem je 3, tj. $(1234/10)\%10=3$

Stovky: Vydělte číslo 100 a poté z něj vypočítejte zbytek, výsledkem je 2, tj. $(1234/100)\%10=2$

Tisíce: Vydělte číslo číslem 1000, výsledkem je 1, tj. $1234/1000=1$

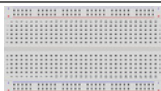


Experiment

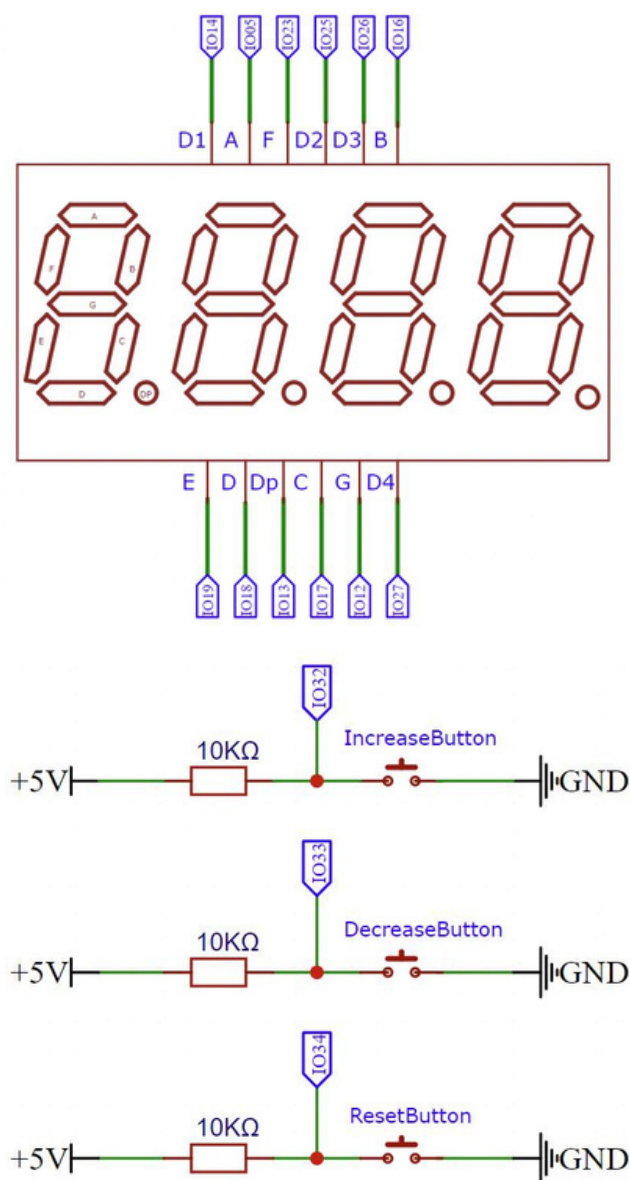
1. Popis projektu

Pomocí tří tlačítek můžete provést operace přičítání jedné, odčítání jedné a mazání čísla a výsledek počítání se zobrazí v reálném čase na čtyřmístném 8segmentovém digitálním displeji.

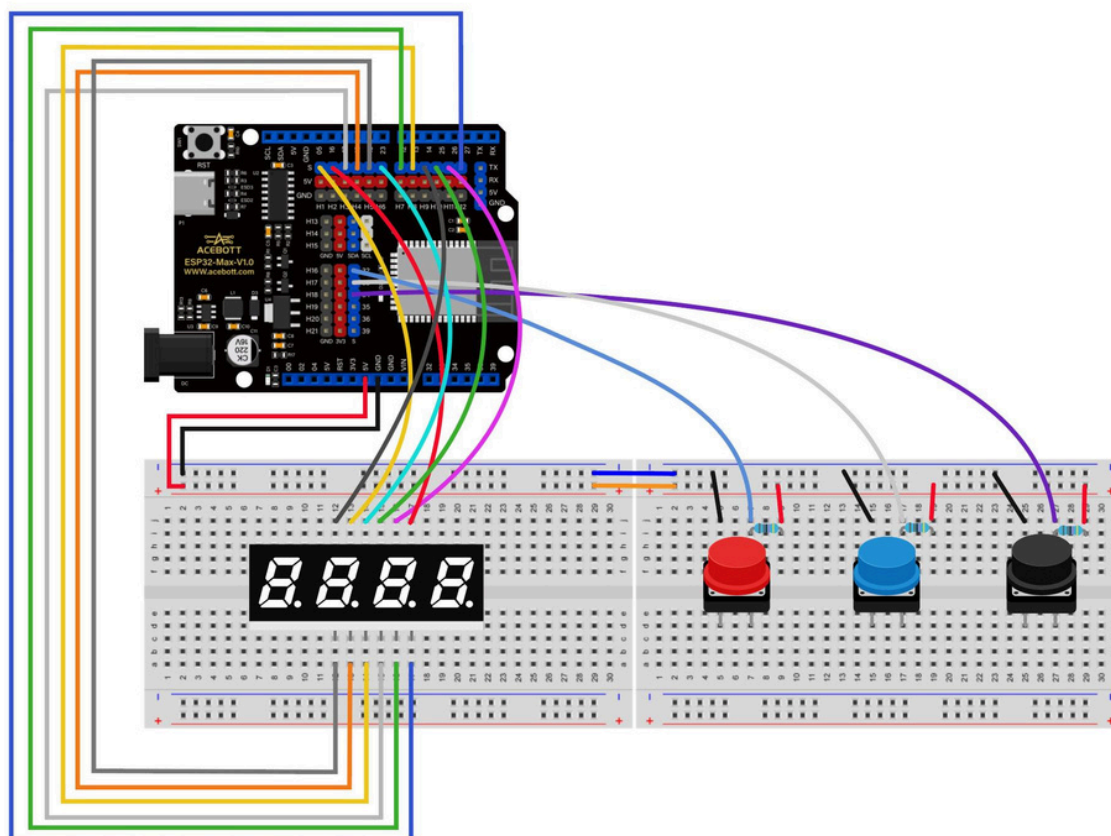
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	2
	Tlačítko	3
	Rezistor 10 kΩ	3
	4místná digitální trubice	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Counter.ino](#)" soubor v "English\Arduino(Experienced Student)\2.Code\lesson 24 Counter\Counter" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků klikněte na



nahrát program.

```
// Define digit enable pins and store them in an array
int seg_1 = 14;
int seg_2 = 25;
int seg_3 = 26;
int seg_4 = 27;

int seg_array[4] = { seg_1, seg_2, seg_3, seg_4 };
// Define segment control pins and store them in an array
int a = 5;
int b = 16;
```



```

int c = 17;
int d = 18;
int e = 19;
int f = 23;
int g = 12;
int dp = 13;

int led_array[8] = { a, b, c, d, e, f, g, dp };

// Define button pins
int increaseButtonPin = 32;
int decreaseButtonPin = 33;
int resetButtonPin = 34;
int number=0;

// Define a 2D array for segment logic corresponding to digits 0-9
// Each row represents segment states for one digit (a-g, dp), 1=ON, 0=OFF
int logic_array[][8] = {
//a, b, c, d, e, f, g, dp
{ 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
{ 0, 1, 1, 0, 0, 0, 0, 0 }, // 1
{ 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
{ 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
{ 0, 1, 1, 0, 0, 1, 1, 0 }, // 4
{ 1, 0, 1, 1, 0, 1, 1, 0 }, // 5
{ 1, 0, 1, 1, 1, 1, 1, 0 }, // 6
{ 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
{ 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
{ 1, 1, 1, 1, 0, 1, 1, 0 }, // 9
};

// Turn off all digits and segments
void clear() {
for (int i = 0; i < 4; i++) {
digitalWrite(seg_array[i], HIGH); // Disable digit (common cathode)
}
for (int i = 0; i < 8; i++) {
digitalWrite(led_array[i], LOW); // Turn off all segments
}
}

// Display a digit (0-9) at a given position (0-3)
void display_number(int order, int number) {
digitalWrite(seg_array[order], LOW); // Enable this digit
for (int i = 0; i < 8; i++) {
digitalWrite(led_array[i], logic_array[number][i]);
}
}

```

```

// Display a 4-digit number without leading zeros
void display_4_number(int number) {
  if (number < 10000) {
    int number_array[4];
    for (int i = 3; i >= 0; i--) {
      number_array[i] = number % 10;
      number /= 10;
    }

    bool leading_zero = true;
    for (int i = 0; i < 4; i++) {
      if (number_array[i] != 0 || i == 3) {
        leading_zero = false;
      }
      if (!leading_zero) {
        display_number(i, number_array[i]);
      }
      delay(5); // Brief display to simulate multiplexing
      clear(); // Prevent ghosting
    }
  }
}

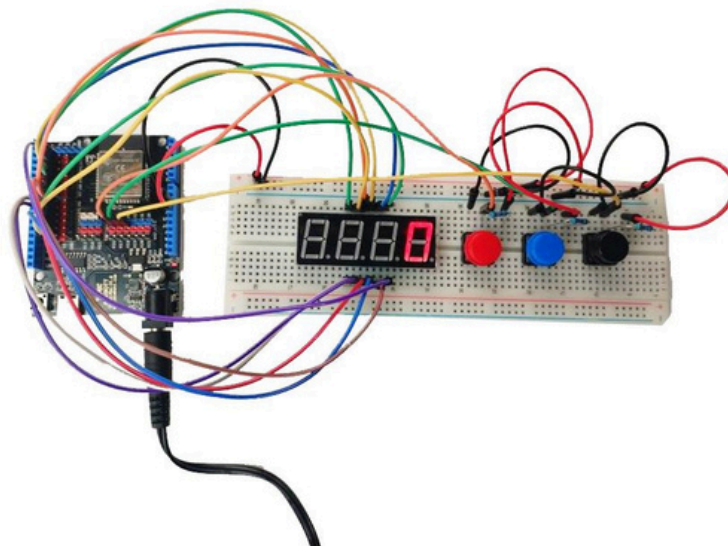
void setup() {
  for (int i = 0; i < 4; i++) {
    pinMode(seg_array[i], OUTPUT);
    digitalWrite(seg_array[i], HIGH);
  }
  for (int i = 0; i < 8; i++) {
    pinMode(led_array[i], OUTPUT);
    digitalWrite(led_array[i], LOW);
  }
  pinMode(increaseButtonPin, INPUT);
  pinMode(decreaseButtonPin, INPUT);
  pinMode(resetButtonPin, INPUT);
  display_4_number(number);
}

void loop() {
  // Non-blocking button detection
  if (digitalRead(increaseButtonPin) == LOW) {
    delay(200);
    if(digitalRead(increaseButtonPin) == LOW){
      number++;
    }
  }
  if (number >= 10000) {
    number = 0;
  }
}

```

```
}  
if (digitalRead(decreaseButtonPin) == LOW){  
  delay(200);  
  if(digitalRead(increaseButtonPin) == LOW){  
  
    if (number > 0) {  
      number--;  
    } else {  
      number = 0;  
    }  
  }  
}  
if (digitalRead(resetButtonPin) == LOW) {  
  number = 0;  
}  
// Display the current count value  
display_4_number(number);  
}
```

Po úspěšném nahrání programu se na čtyřmístném osmisegmentovém displeji zobrazí číslo 0. Stisknutím červeného tlačítka se číslo zvýší o 1, stisknutím modrého tlačítka se číslo o 1 sníží a stisknutím černého tlačítka se číslo vynuluje na 0.



Lekce 25 Bzučák kvízu



Cíle kurzu

- Pochopíte princip fungování bzučáku kvízu
- Zvládněte logiku detekce více vstupů a určování priorit
- Napište program pro implementaci bzučáku kvízu



Úvod do kurzu

V různých vědomostních soutěžích a aktivitách je bzučák kvízu důležitým pomocným nástrojem. Dokáže rychle určit, kdo odpověděl první, a zajišťuje tak férovost a zábavu.



V této lekci se naučíme, jak navrhnout jednoduchý a praktický bzučák pro kvíz. Prostřednictvím tohoto projektu si nejen osvojíte základní logiku detekce více vstupů a určování priorit, ale také si zlepšíte programátorské dovednosti a schopnosti hardwarových aplikací. Pojdme se do toho pustit a zažít zábavu z odpovídání bzučákem!



Body znalostí

1. Bzučák kvízu Bzučák v kvízu je elektronické zařízení používané v soutěžích k určení, kdo stiskl tlačítko jako první. Dokáže rychle identifikovat prvního hráče, který stiskl tlačítko, a uzamknout výsledek, přičemž zabrání ostatním v pokračování v zadávání, čímž zajistí spravedlivý a spravedlivý proces odpovídání.

Jak funguje bzučák kvízu

(1) Začněte s detekcí odpovědi Po spuštění programu je záznamník ve výchozím nastavení v neaktivním stavu a klávesové zkratky hráče jsou neplatné. Teprve když hostitel stiskne „tlačítko start“, systém se přepne do aktivního režimu a začne monitorovat záznamníky všech hráčů.

(2) Detekce odpovědi Po vstupu do aktivního stavu program průběžně dotazuje klávesy každého hráče. Jakmile je detekováno stisknutí klávesy, je hráč okamžitě určen jako první odpovídající a jeho číslo je zaznamenáno.

(3) Zámek přijímače Po úspěšné odpovědi systém okamžitě přejde do uzamčeného stavu a ignoruje vstupy ostatních hráčů, čímž zajistí, že bude identifikován pouze první hráč, který stiskne tlačítko.

(4) Zpětná vazba k výsledkům Systém používá bzučák a LED světla k indikaci výsledku rychlé odpovědi. Bzučák signalizuje konec rychlé odpovědi a kontrolka odpovídající číslu soutěžícího se rozsvítí, což znamená, že soutěžící v tomto kole úspěšně odpověděl na otázku.

Kvízový bzučák se široce používá ve výukových soutěžích, interaktivních hrách a jiných příležitostech, což může efektivně zlepšit smysl

účast a interaktivitu a zlepšit zábavu a efektivitu aktivit.

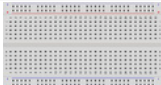


Experiment

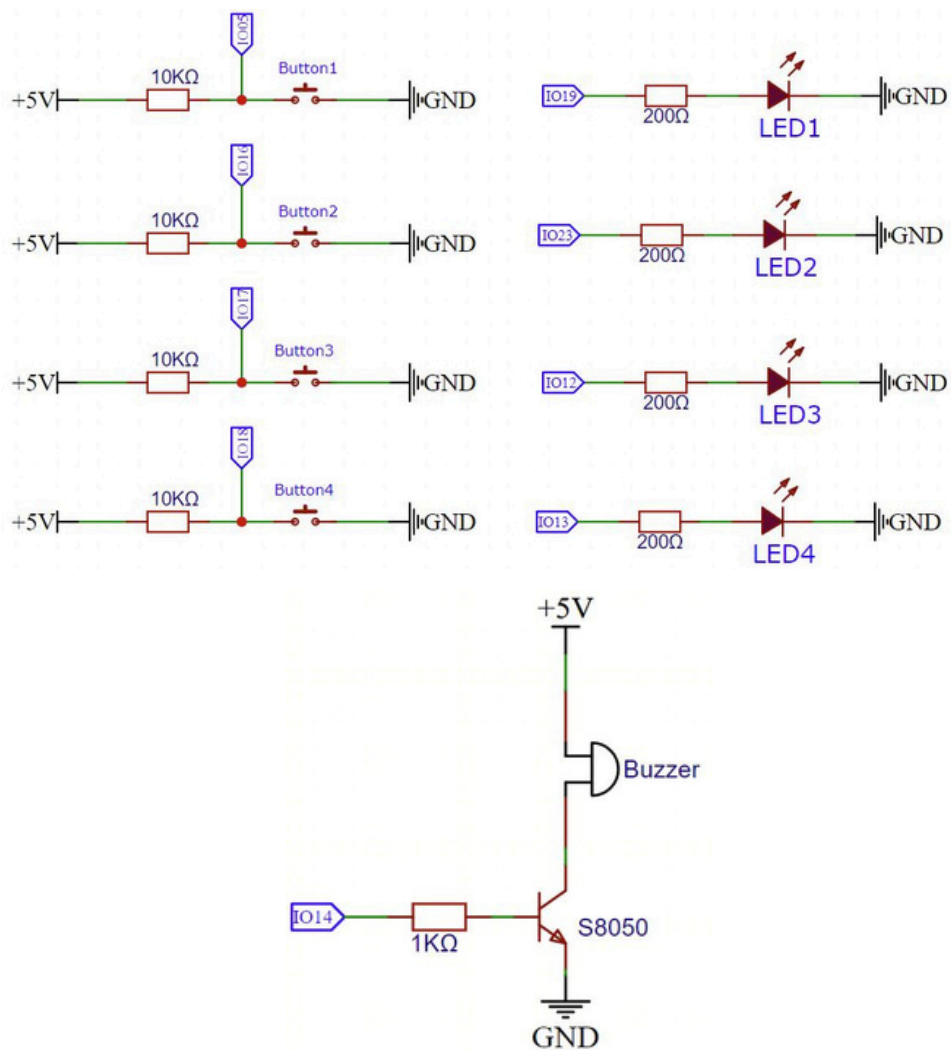
1. Popis projektu

Program implementuje funkci bzučáku pro čtyři hráče. Detekcí vstupu ze čtyř tlačítek systém určí, kdo stiskl tlačítko jako první, zaznamená úspěšnou odpověď hráče, spustí bzučák a rozsvítí kontrolku hráče, čímž zobrazí výsledek.

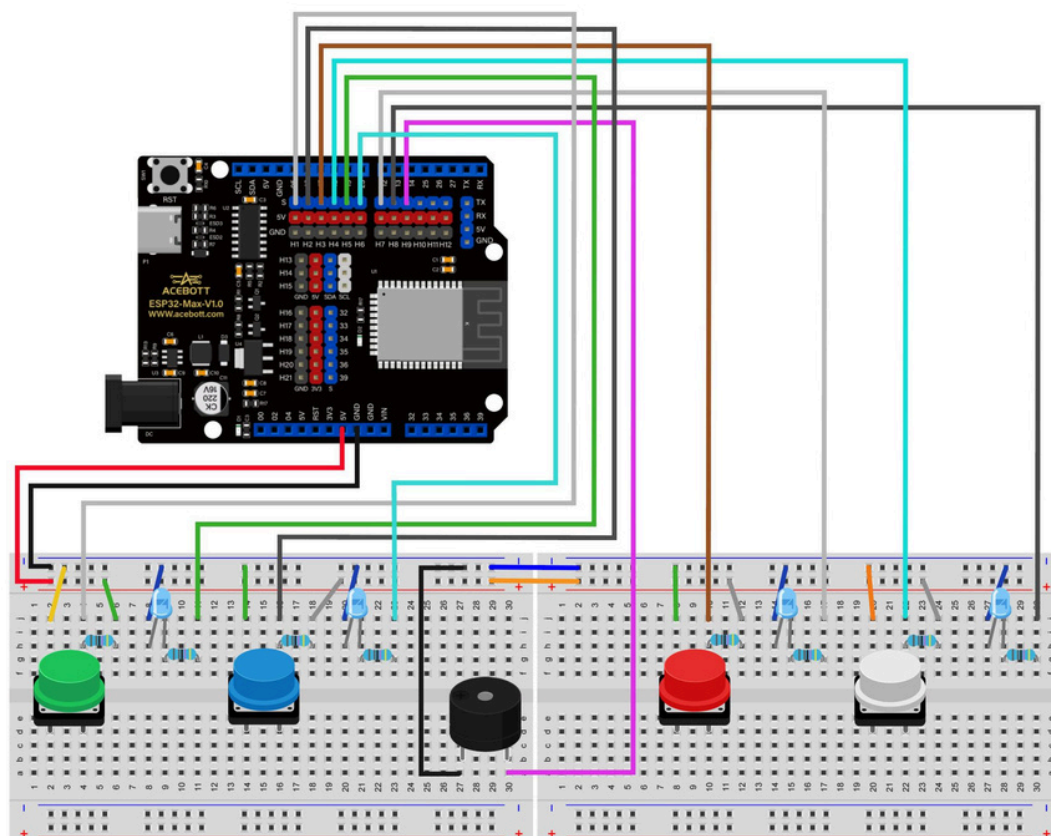
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	2
	Tlačítko	4
	LED	4
	Aktivní bzučák	1
	Rezistor 10 kΩ	1
	Rezistor 220Ω	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Quiz_Buzzer.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 25 Quiz Buzzer\Quiz_Buzzer" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

dříve naučené kroky, klikněte  nahrát program.

```
// Define button pins
const int buttonPins[] = {5, 16, 17, 18};
// Define LED pins
const int ledPins[] = {19, 23, 12, 13};
// Define buzzer pin
const int buzzerPin = 14;

int winner = -1; // Identifier of the player who pressed the button first,
-1 means no one has buzzed in

void setup() {
  // Initialize button pins as input with internal pull-up resistors
  for (int i = 0; i < 4; i++) {
    pinMode(buttonPins[i], INPUT);
  }
}
```

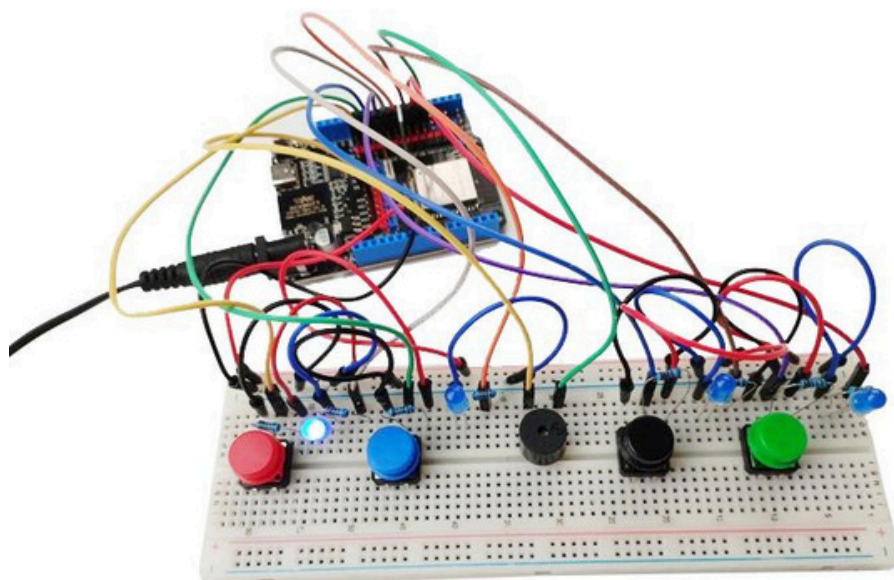


```
}

// Initialize LED pins as output
for (int i = 0; i < 4; i++) {
    pinMode(ledPins[i], OUTPUT);
    digitalWrite(ledPins[i], LOW);
}
pinMode(buzzerPin, OUTPUT);
Serial.begin(115200);
Serial.println("Quiz Buzzer Ready!");
}

void loop() {
    // Detect who buzzed in first
    if (winner == -1) {
        for (int i = 0; i < 4; i++) {
            if (digitalRead(buttonPins[i]) == LOW) { // Button is pressed
                delay(200); // Simple debounce delay
                while(digitalRead(buttonPins[i]) == LOW);
                winner = i; // Record the winner
                Serial.print("Player ");
                Serial.print(winner + 1);
                Serial.println(" wins!");
                // Light up the LED of the winner
                for (int j = 0; j < 4; j++) {
                    if (j == winner) {
                        digitalWrite(ledPins[j], HIGH); // Winner's LED ON
                    } else {
                        digitalWrite(ledPins[j], LOW); // Other LEDs OFF
                    }
                }
                // Buzzer sound to indicate successful buzz-in
                digitalWrite(buzzerPin, HIGH);
                delay(200);
                digitalWrite(buzzerPin, LOW);
                break;
            }
        }
    }
}
```

Po úspěšném nahrání programu stiskněte první tlačítko, ozve se bzučák a rozsvítí se LED dioda prvního hráče, což znamená, že hráč 1 úspěšně odpověděl na otázku.



Lekce 26 Ovocné piano



Cíle kurzu

- Pochopíte principy fungování ovocného piana.
- Zvládněte principy kapacitní detekce dotyku. Napište
- program pro implementaci funkce ovocného piana.



Úvod do kurzu

Napadlo vás někdy, že ovoce, které jíme, se může skutečně proměnit v klávesy klavíru? Je to proto, že ovoce je bohaté na vodu a elektrolyty, které mají určitý stupeň vodivosti a mohou sloužit jako dotykové elektrody. Když se ovoce dotkneme prsty, regulátor dokáže zaznamenat změnu kapacity a spustit odpovídající tón.



Pomocí této jednoduché a inovativní metody můžeme proměnit ovoce v zábavné „ovocné piano“.

V této lekci si postavíme ovocné piano, přičemž ovoce použijeme jako náhradu za běžné klávesy. Dozvíme se o kapacitní technologii dotykového snímání, zažijeme úžasnou kombinaci elektroniky a každodenního života a oceníme kouzlo kreativních elektronických projektů.

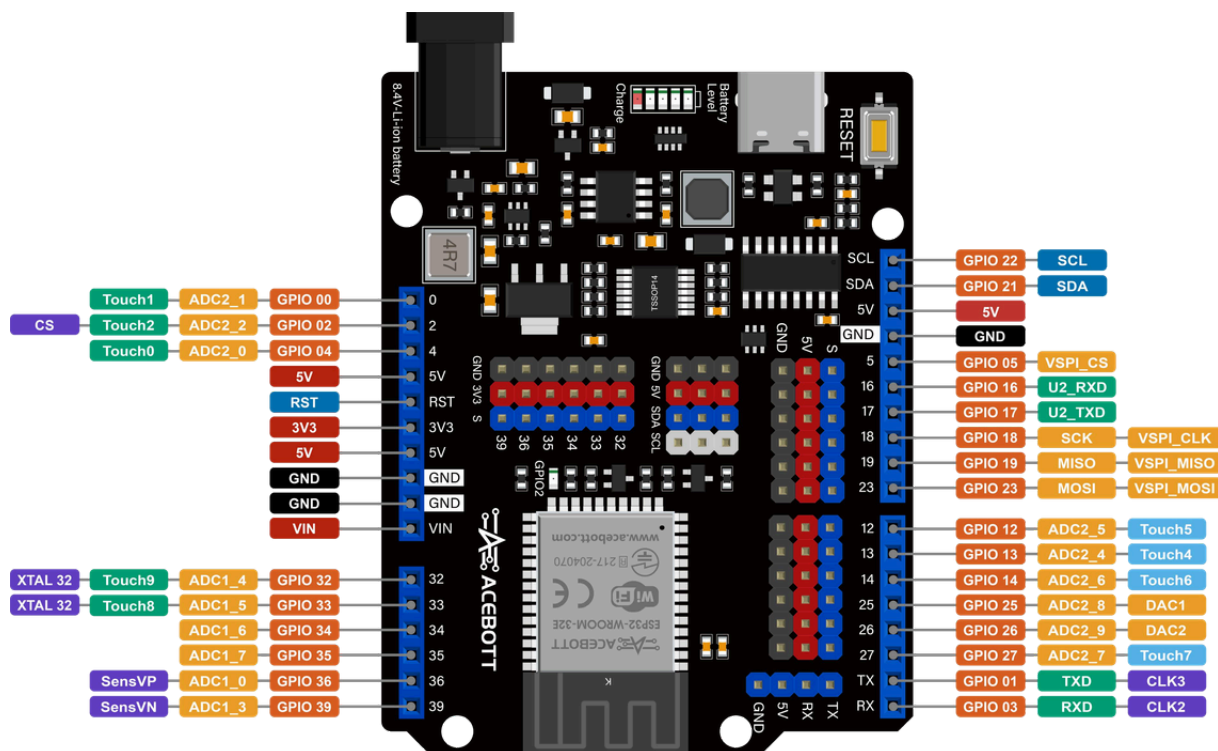


Body znalostí

1. Princip kapacitního dotykového snímání

Technologie kapacitního dotykového snímání je technologie, která detekuje stav dotyku monitorováním změn kapacity. Za normálních okolností se mezi dotykovým pinem ovladače ESP32 a okolním světem vytváří stabilní kapacitní systém. Když se vodivý objekt, jako je lidské tělo, přiblíží nebo dotkne připojené elektrody (například ovoce), lidské tělo vytvoří se systémem vazebnou kapacitu, čímž se změní původní rozložení elektrického pole a způsobí změnu celkové kapacity. Ovladač ESP32 má interní modul pro snímání kapacity, který dokáže nepřetržitě monitorovat hodnotu kapacity na každém dotykovém pinu. Pokud se zjistí, že se kapacita významně odchyluje od normální hodnoty, lze to posoudit jako „dotykovou událost“.

Piny s označením Touch v diagramu zapojení pinů desky řadiče ESP32 mají kapacitní dotykové snímání a umožňují detekci dotyku.



Poznámka: Protože GPIO0 a GPIO2 mají speciální spouštěcí funkce, nemusí funkce dotykového snímání fungovat správně. Pro dosažení spolehlivého kapacitního dotykového snímání se doporučuje použít následující piny: GPIO4, 12, 13, 14, 27, 32 a 33.

Technologie kapacitního dotyku se široce používá v různých moderních elektronických produktech díky svým výhodám, jako je absence mechanického kontaktu, vysoká citlivost a odolnost proti opotřebení. Například dotykové obrazovky chytrých telefonů a tabletů využívají kapacitní snímání pro přesné ovládání. V kreativní elektronice, jako jsou ovocné piana a interaktivní umělecké instalace, kapacitní dotyk také vykazuje velký inovativní potenciál a zábavu.

2. Ovocné piano

Ovocné piano využívá kapacitní dotykovou funkci ESP32 k použití běžného ovoce (jako jsou jablka, pomeranče, banány atd.) jako „dotykových tlačítek“. Dotykem na různé druhy ovoce lze zahrát různé tóny a vytvořit tak zajímavý elektronický hudební nástroj.

Jak funguje ovocné piano?

(1) Spusťte detekci dotyku Po spuštění programu ESP32 inicializuje dotykové piny a nastaví prahovou hodnotu snímání každého dotykového kanálu. Systém začne monitorovat dotykový vstup.

(2) Kapacitní snímací mechanismus Každý plod je připojen k dotykovému pinu ESP32 pomocí drátu, čímž vzniká jednoduchý kapacitní obvod. Voda a buněčná tkáňová struktura v plodu fungují jako kapacitní médium. Když se prst přiblíží k plodu nebo se ho dotkne, elektrické pole lidského těla ovlivní hodnotu kapacity mezi elektrodou a zemí, což způsobí její významnou změnu (hodnota kapacity se sníží).

(3) Úsudek a reakce na dotek Dotykový senzor ESP32 detekuje

že hodnota kapacity kanálu se změní nad přednastavenou prahovou hodnotu, což je považováno za platný dotek. Program zahraje odpovídající notu.



Experimentovat

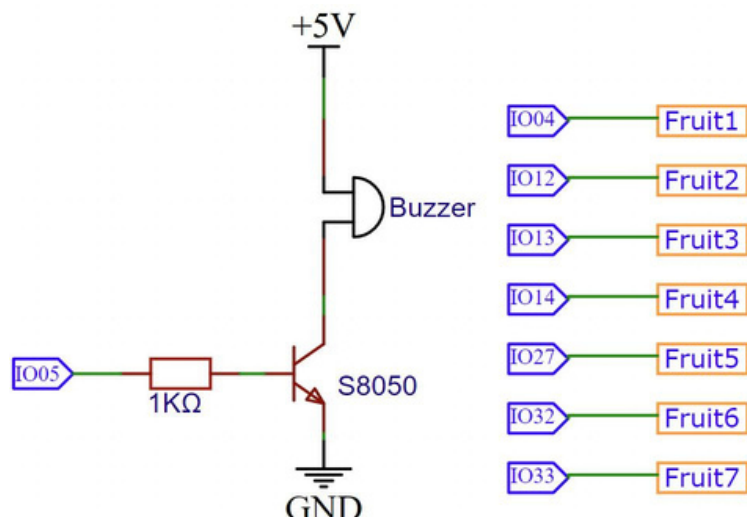
1. Popis projektu

Každý dotykový pin ESP32 je připojen k jinému ovoci. Když se uživatel dotkne ovoce, řídicí deska ESP32 detekuje dotykový signál a ovládá bzučák tak, aby přehrál odpovídající tón podle odpovídající noty (DO, RE, MI, FA, SO, LA, TI).

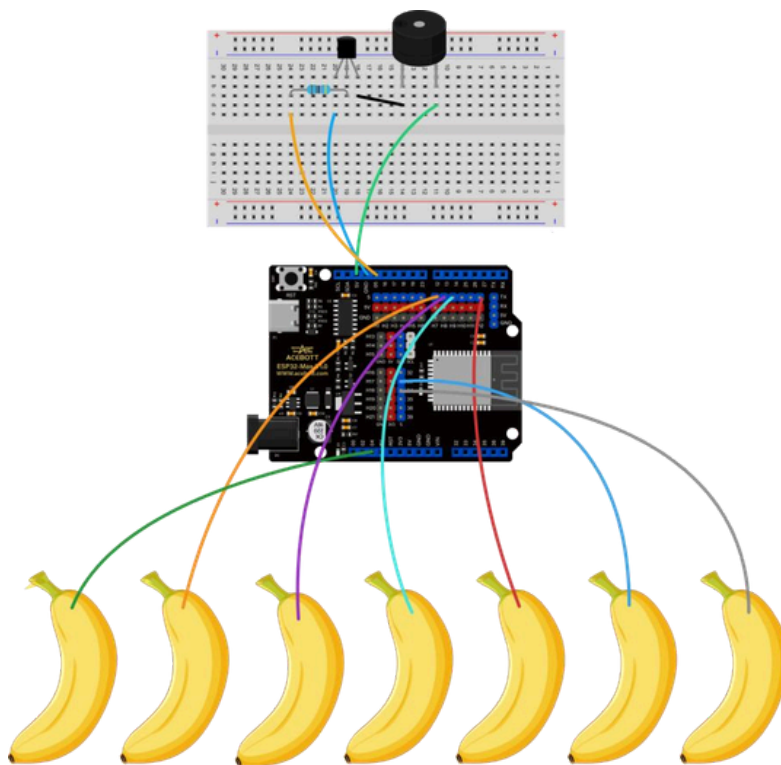
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Banán	7
	Pasivní bzučák	1
	Tranzistor	1
	Rezistor 1 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Fruit_Piano.ino](#)" soubor v "English\Arduino(Experienced Learner) \2.Code\lesson 26 Fruit Piano\Fruit_Piano" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků, kliknutí ➡ nahrát program.


```
#define BUZZER_PIN 5 // Buzzer output pin

// Define ESP32 touch input pins
#define TOUCH_PIN_0 4 // T0 - DO
#define TOUCH_PIN_1 12 // T1 - RE
#define TOUCH_PIN_2 13 // T2 - MI
#define TOUCH_PIN_3 14 // T3 - FA
#define TOUCH_PIN_4 27 // T4 - SO
#define TOUCH_PIN_5 32 // T5 - LA
#define TOUCH_PIN_6 33 // T6 - TI

// Corresponding note names
const char* noteNames[] = {"DO", "RE", "MI", "FA", "SO", "LA", "TI"};

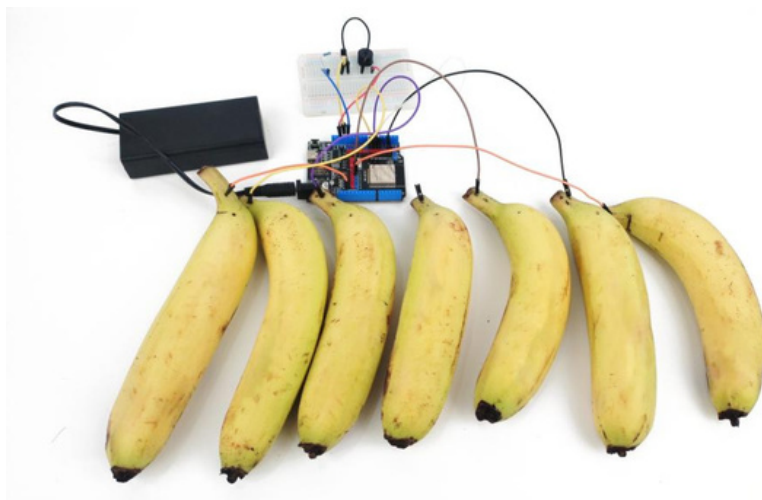
// Note frequencies in Hz
const int tones[] = {523, 587, 659, 698, 784, 880, 988};

// Touch threshold value: lower means touched
int touchThreshold = 20;
void setup() {
  Serial.begin(115200); // Start serial monitor
  pinMode(BUZZER_PIN, OUTPUT); // Set buzzer pin as output
  Serial.println("7-tone Fruit Piano started. Reading capacitance values...");
}
void loop() {
  // Read capacitance (touch) values from all pins
  int t0 = touchRead(TOUCH_PIN_0);
  int t1 = touchRead(TOUCH_PIN_1);
  int t2 = touchRead(TOUCH_PIN_2);
  int t3 = touchRead(TOUCH_PIN_3);
  int t4 = touchRead(TOUCH_PIN_4);
  int t5 = touchRead(TOUCH_PIN_5);
  int t6 = touchRead(TOUCH_PIN_6);
  // Print the touch values for debugging
  Serial.print("T0=");
  Serial.print(t0);
  Serial.print(" T1=");
  Serial.print(t1);
  Serial.print(" T2=");
  Serial.print(t2);
  Serial.print(" T3=");
  Serial.print(t3);
  Serial.print(" T4=");
  Serial.print(t4);
  Serial.print(" T5=");
  Serial.print(t5);
  Serial.print(" T6=");
```



```
Serial.println(t6);  
// Check if any pin is being touched, play corresponding tone  
if (t0 < touchThreshold) {  
  playNote(tones[0], noteNames[0]);  
} else if (t1 < touchThreshold) {  
  playNote(tones[1], noteNames[1]);  
} else if (t2 < touchThreshold) {  
  playNote(tones[2], noteNames[2]);  
} else if (t3 < touchThreshold) {  
  playNote(tones[3], noteNames[3]);  
} else if (t4 < touchThreshold) {  
  playNote(tones[4], noteNames[4]);  
} else if (t5 < touchThreshold) {  
  playNote(tones[5], noteNames[5]);  
} else if (t6 < touchThreshold) {  
  playNote(tones[6], noteNames[6]);  
} else {  
  noTone(BUZZER_PIN); // No touch detected, stop buzzer  
}  
delay(100); // Small delay to reduce noise  
}  
// Play a tone and print the note name  
void playNote(int frequency, const char* note) {  
  tone(BUZZER_PIN, frequency);  
  Serial.print("Playing: ");  
  Serial.println(note);  
}
```

Po úspěšném nahrání programu se dotkněte odpovídajícího ovoce a zahrajte odpovídající notu.



Output	Serial Monitor	×
Message (Enter to send message to 'ESP32 Dev Module' on 'COM14')		
T0: 34	T1: 23	T2: 21
T3: 26	T4: 30	T5: 12
T6: 12		
T0: 33	T1: 22	T2: 21
T3: 27	T4: 30	T5: 12
T6: 12		
T0: 33	T1: 22	T2: 21
T3: 27	T4: 30	T5: 12
T6: 12		
T0: 33	T1: 22	T2: 21
T3: 27	T4: 29	T5: 14
T6: 14		
T0: 34	T1: 21	T2: 17
T3: 27	T4: 29	T5: 14
T6: 14		
T0: 34	T1: 21	T2: 17
T3: 27	T4: 29	T5: 14
T6: 14		
T0: 34	T1: 21	T2: 24
T3: 28	T4: 32	T5: 13
T6: 13		

6. Popis instrukcí touchRead(pin): Používá se ke čtení hodnoty kapacity zadaného pinu. Vrací celé číslo, kde menší hodnota značí vyšší kapacitu. Tato funkce se běžně používá k detekci dotyku nebo přiblížení. Parametr pin označuje číslo dotykového pinu, který má být čten. **Poznámka:** Podporovány jsou pouze piny 4, 12, 13, 14, 27, 32 a 33.

Lekce 27 Ventilátor s proměnnou rychlostí



Cíle kurzu

- Pochopení principu fungování elektrického ventilátoru s proměnnou rychlostí
- Napsání programu pro implementaci elektrického ventilátoru s proměnnou rychlostí



Úvod do kurzu

V každodenním životě, zejména na cestách, jsou ruční ventilátory s proměnnou rychlostí oblíbené pro své pohodlné přepínání rychlostí, které umožňuje flexibilní nastavení rychlosti větru podle potřeb a poskytuje tak co nejpohodlnější chlazení.



V této lekci se naučíme, jak navrhnout elektrický ventilátor s proměnnou rychlostí. Zvládnete pohodlné nastavení rychlosti ventilátoru pomocí tlačítek, vyzkoušíte si běžné operace řazení v každodenním životě a zlepšíte praktičnost a uživatelský zážitek z chytrých zařízení.



Body znalostí

1. Ventilátor s proměnnou rychlostí

Ventilátor s proměnnou rychlostí je ventilátorové zařízení, které podporuje víceúrovňovou regulaci rychlosti. Rychlost se upravuje změnou napájení motoru, čímž se dosahuje různé intenzity větrného výkonu, aby se splnily potřeby uživatelů na chlazení v různých prostředích.

Jak funguje ventilátor s proměnnou rychlostí?

Ventilátor má tři nastavení rychlosti: 0, 1 a 2. Při hodnotě 0 se ventilátor úplně zastaví, při hodnotě 1 běží nízkou rychlostí a při hodnotě 2 dosáhne vysoké rychlosti. Tři rychlosti ventilátoru se ovládají třemi nezávislými tlačítky, z nichž každé odpovídá určité rychlosti. Stisknutím odpovídajícího tlačítka se ventilátor přepne na danou rychlost a spustí se provoz.



Experimentovat

1. Popis projektu

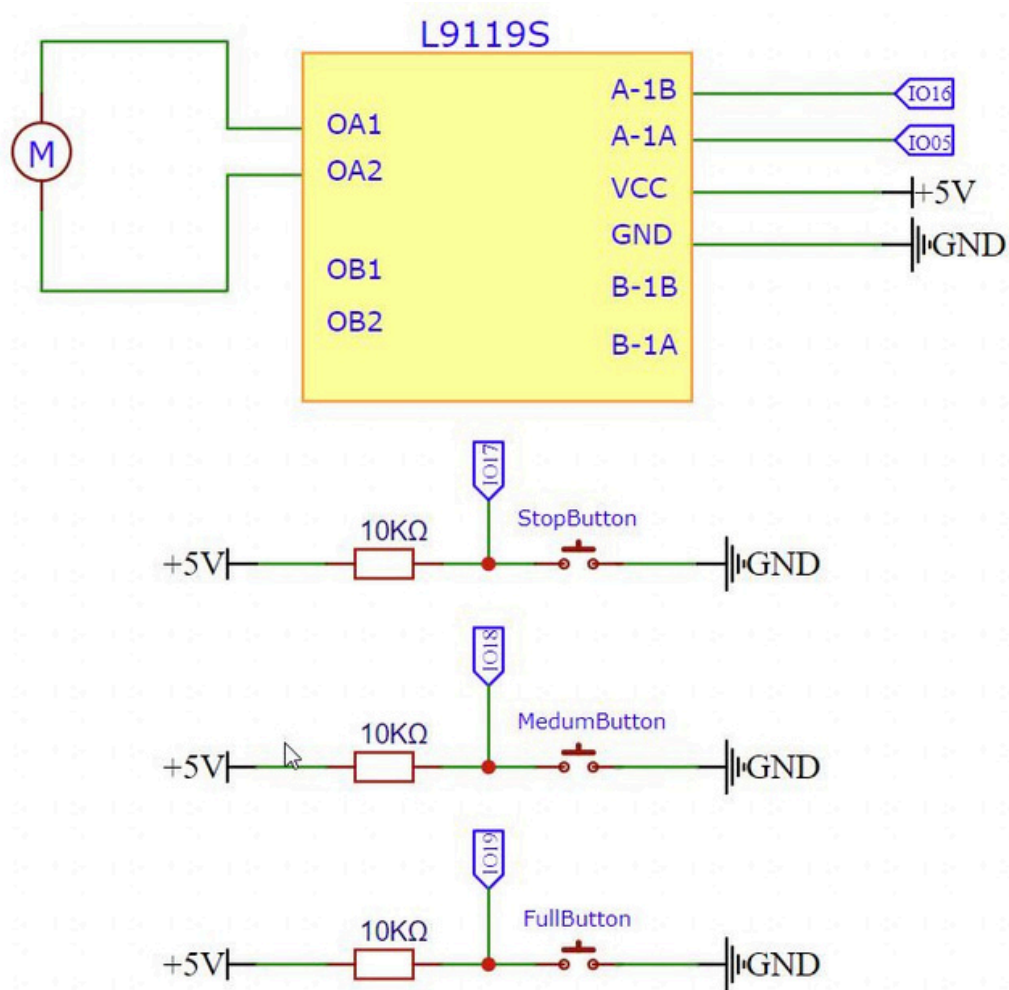
Program ovládá rychlost ventilátoru pomocí tří tlačítek a používá PWM řízení k dosažení třístupňové regulace rychlosti: stop, nízká rychlost a plná rychlost.

2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Ovladač stejnosměrného motoru	1
	Stejnoseměrný motor	1

	Větrák	1
	Nepájivé pole s 400 spojovacími body	1
	Tlačítko	3
	Rezistor 10 kΩ	3

3. Schéma zapojení



4. Hardwarové připojení


```
int fanSpeed = 0; // Current fan speed (0-255)
int lastSpeed = -1; // Store previous speed to prevent redundant writes

void setup() {
  Serial.begin(115200);
  pinMode(motor1A, OUTPUT);
  pinMode(motor2A, OUTPUT);

  pinMode(buttonStop, INPUT); // Button: use external pull-up
  pinMode(buttonMedium, INPUT);
  pinMode(buttonFull, INPUT);

  analogWrite(motor1A, 0); // Fan initially off
  analogWrite(motor2A, 0); // Reverse direction pin off
}

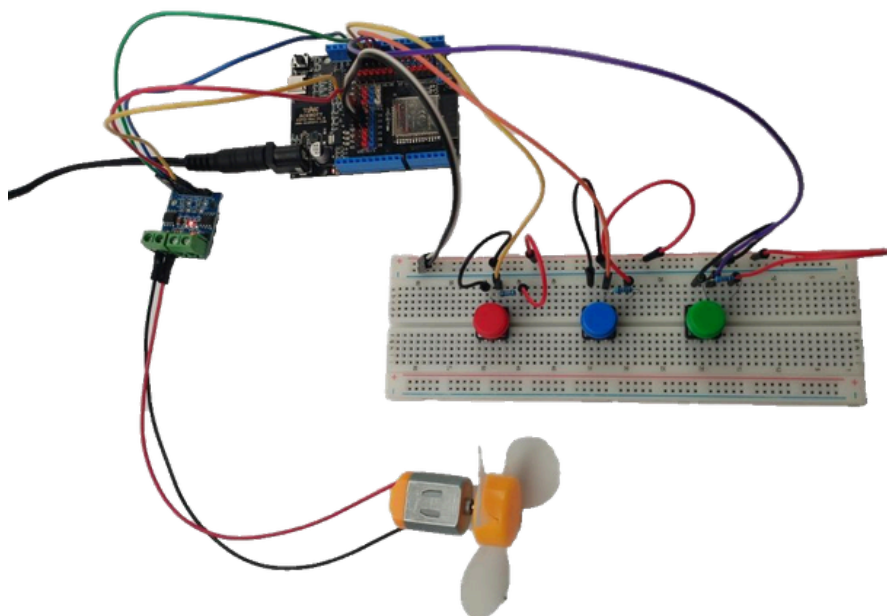
void loop() {
  // Check if STOP button is pressed
  if (digitalRead(buttonStop) == LOW) {
    delay(200); // Debounce
    while(digitalRead(buttonStop) == LOW);
    fanSpeed = 0;
  }
  // Else check if MEDIUM button is pressed
  else if (digitalRead(buttonMedium) == LOW) {
    delay(200); // Debounce
    while(digitalRead(buttonMedium) == LOW);
    fanSpeed = 179; // ~70%
  }
  // Else check if FULL button is pressed
  else if (digitalRead(buttonFull) == LOW) {
    delay(50); // Debounce
    while(digitalRead(buttonFull) == LOW);
    fanSpeed = 255; // 100%
  }

  // Only update motor if speed changed
  if (fanSpeed != lastSpeed) {
    analogWrite(motor1A, fanSpeed);
    analogWrite(motor2A, 0);
    Serial.print("Fan speed set to: ");
    Serial.print(fanSpeed);
    Serial.println("/255");
    lastSpeed = fanSpeed;
  }

  delay(100); // Short loop delay
}
```


}

Po úspěšném nahrání programu stiskněte modré tlačítko pro spuštění ventilátoru na nízkou rychlost, zelené tlačítko pro spuštění na vysokou rychlost a tlačítko pro jeho zastavení.



Lekce 28 Ventilátor s regulací teploty



Cíle kurzu

- Pochopte principy fungování elektrických ventilátorů s regulací teploty
- Napište program pro implementaci elektrických ventilátorů s regulací teploty



Úvod do kurzu

V horkých letních dnech nebo v uzavřených prostorách teplotní výkyvy přímo ovlivňují naše pohodlí a správné fungování zařízení. Představte si, že by ventilátor mohl automaticky upravovat svou rychlost na základě okolní teploty, a dokonce se automaticky zapnout, když je teplota příliš vysoká, a zajistit tak optimální cirkulaci vzduchu. Takový



Chytré zařízení by nejen zlepšilo kvalitu života, ale také ušetřilo energii. V této lekci se naučíme, jak navrhnout elektrický ventilátor s regulací teploty a vyzkoušíme si pohodlí a praktičnost inteligentního ovládání.



Body znalostí

1. Elektrický ventilátor s regulací teploty

Elektrický ventilátor s regulací teploty je inteligentní ventilátor, který automaticky upravuje svou rychlost na základě okolní teploty. Když detekuje

Při zvýšení okolní teploty se automaticky zvýší rychlost, aby se zvýšil chladicí účinek; při poklesu teploty se automaticky zpomalí nebo zastaví, čímž se dosáhne regulace rychlosti větru, která vyvažuje úsporu energie a komfort.

Jak funguje elektrický ventilátor s regulací teploty?

Používá teplotní senzor k monitorování okolní teploty v reálném čase. Když teplota dosáhne nastavené prahové hodnoty, senzor vyšle signál do řídicí jednotky, která automaticky upraví otáčky ventilátoru na základě kolísání teploty. Například ventilátor se zastaví, když je teplota nízká, a zrychlí se s rostoucí teplotou, čímž se zvýší cirkulace vzduchu a sníží teplota. Tato uzavřená regulace umožňuje inteligentní provoz ventilátoru, což zajišťuje jak komfort, tak energetickou účinnost.

Elektrické ventilátory s regulací teploty se široce používají v domácnostech, kancelářích, továrních halách, pro chlazení počítačových místností, skleníkového zemědělství a chlazení elektronických zařízení. Automaticky upravují rychlost větru na základě skutečné teploty, čímž zajišťují stabilní provoz zařízení nebo růst rostlin, zlepšují celkovou účinnost a bezpečnost a šetří energii.

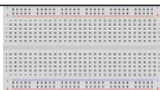


Experiment

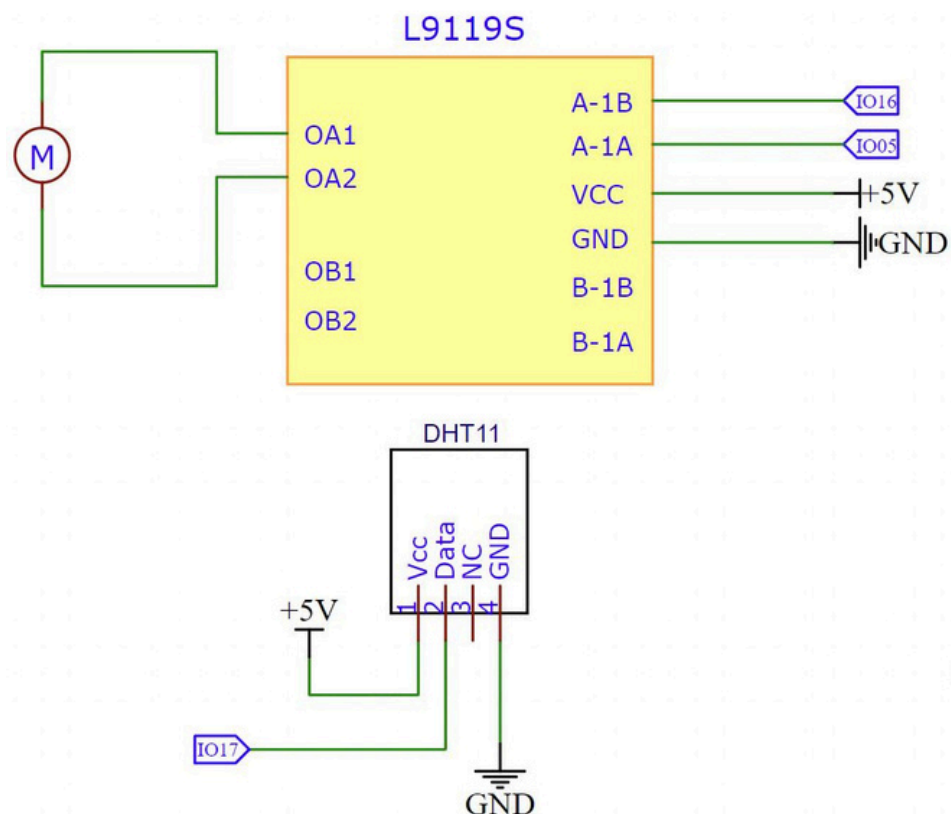
1. Popis projektu

Pomocí senzoru teploty a vlhkosti můžete v reálném čase odečítat okolní teplotu a řídit otáčky motoru na základě změn teploty. Pokud je teplota pod 30 °C, motor se zastaví; mezi 30 °C a 32 °C běží motor nízkou rychlostí; a nad 32 °C běží motor vysokou rychlostí.

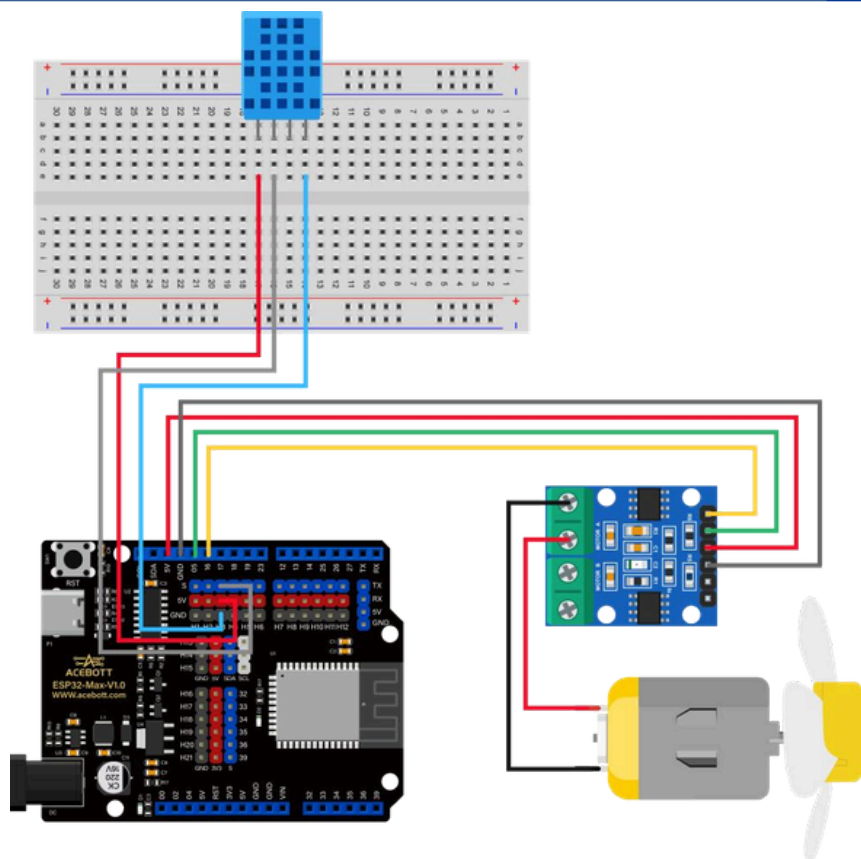
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Ovladač stejnosměrného motoru	1
	Stejnoseměrný motor	1
	Větrák	1
	DHT11	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Smart_Temperature_Controlled_Fan.ino](#)" „soubor v jazyce „English \Arduino(Experienced Learner) \2.Code\lesson 28 Inteligentní ventilátor s regulací teploty\Smart_Temperature_Controlled_Fan“ nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu

podle dříve naučených kroků klikněte



nahrát program.

```
#include <DHT.h>
#define motor1A 5
#define motor2A 16
#define DHTPIN 17 // Set the pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  // initialize digital pin as an output.
  pinMode(motor1A, OUTPUT);
  pinMode(motor2A, OUTPUT);
  dht.begin();
}
```

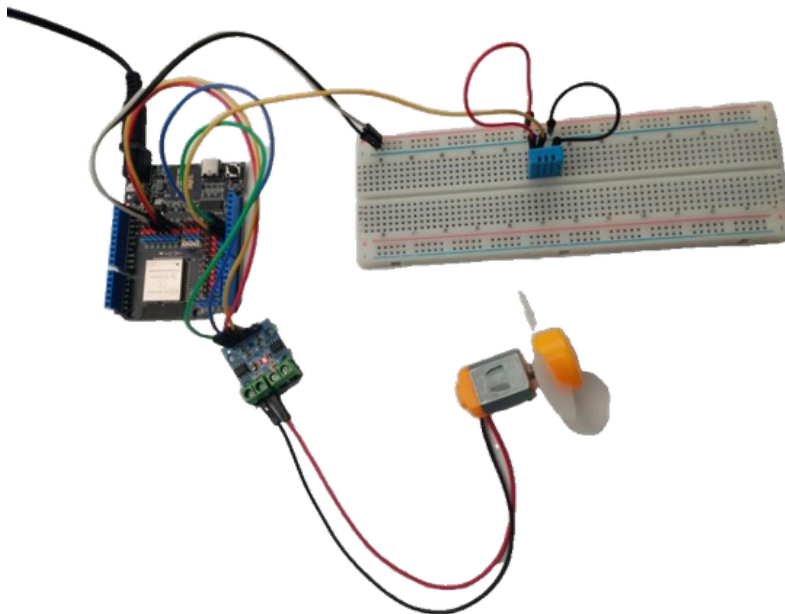
```
}  
void loop() {  
  delay(1000);  
  float temperature = dht.readTemperature();  
  if (isnan(temperature)) {  
    Serial.println("Failed to read from DHT sensor!");  
    return;  
  }else{  
    Serial.print("Temperature: ");  
    Serial.println(temperature);  
    if(temperature<30){  
      analogWrite(motor1A, 0);  
      analogWrite(motor2A, 0);  
    }else if(temperature>30 && temperature<32){  
      analogWrite(motor1A, 136);  
      analogWrite(motor2A, 0);  
    }else if(temperature>32){  
      analogWrite(motor1A, 255);  
      analogWrite(motor2A, 0);  
    }  
  }  
}
```

Po nahrání programu, pokud je teplota

$\leq 25\text{ }^{\circ}\text{C}$, pak $25\text{ }^{\circ}\text{C} <$

$30\text{ }^{\circ}\text{C}$, ventilátor běží nízkou rychlostí; při $32\text{ }^{\circ}\text{C}$ běží vysokou

rychlostí.



Lekce 29 Ventilátor s infračerveným dálkovým ovládáním



Cíle kurzu

- Pochopení principu fungování elektrických ventilátorů s infračerveným dálkovým ovládáním
- Napište program pro implementaci infračerveného dálkového ovládání elektrických ventilátorů.



Úvod do kurzu

Přáli byste si doma během horkého odpoledne na dálku ovládat výkon a rychlost ventilátoru, aniž byste museli vstávat? Ve skutečnosti je tato „vychytávka pro lenochy“ umožněna technologií infračerveného dálkového ovládání.



V této lekci se naučíme, jak dálkově ovládat elektrický ventilátor pomocí infračerveného dálkového ovladače, pochopíme principy této široce používané technologie v domácích spotřebičích a postavíme si vlastní chytrý ventilátor.



Body znalostí

1. Elektrický ventilátor s infračerveným dálkovým ovládáním

Elektrický ventilátor s infračerveným dálkovým ovládáním je ventilátorové zařízení ovládané

Protokol infračervené komunikace. Infračervený signál vysílaný dálkovým ovládáním je modulován a přijímán přijímacím modulem ventilátoru. Řídicí systém po analýze provádí odpovídající operace k realizaci funkcí, jako je nastavení rychlosti větru, ovládání motoru a otáčení motoru.

Princip fungování elektrického ventilátoru s infračerveným dálkovým ovládáním

Řídicí příkaz je přenášen infračervenými signály pro dálkové ovládání a regulaci rychlosti větru ventilátoru. Elektrický ventilátor s infračerveným dálkovým ovládáním se skládá hlavně z následujících tří částí:

(1) Infračervený vysílač (dálkové ovládání)

Dálkový ovladač má vestavěnou infračervenou diodu. Když uživatel stiskne tlačítko, bliká na specifické frekvenci (obvykle 38 kHz) a vysílá kódovaný infračervený signál. Ačkoli jsou tyto signály neviditelné, každé tlačítko má jedinečnou kódovou hodnotu, která představuje jiný příkaz.

(2) Modul pro příjem infračerveného signálu

Terminál pro ovládání ventilátoru je vybaven modulem infračerveného přijímače, který dokáže detekovat infračervený světelný signál vyzařovaný dálkovým ovladačem a převést jej na digitální signál (změny vysoké a nízké úrovně) a poté jej předat mikrokontroléru ke zpracování. Tento modul je obvykle nejcitlivější na nosnou frekvenci 38 kHz, aby byl zajištěn přesný příjem.

(3) Dekódování a ovládání desky řídicí jednotky ESP32

Řídicí deska ESP32 používá knihovnu IRremote k dekodování přijatých infračervených signálů, čímž identifikuje konkrétní tlačítko stisknuté uživatelem. Na základě dekodovaného kódu klávesy provede odpovídající řídicí operaci. Například:

- Stiskněte klávesu „0“ (kód: 0xAD52FF00): Vypněte ventilátor.
- Stiskněte klávesu „1“ (kód: 0xE916FF00): Nastavte rychlost ventilátoru na 70 %.

-Stiskněte klávesu „2“ (kód: 0xE619FF00): Nastavte rychlost ventilátoru na 100 %.

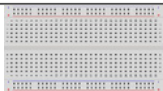


Experiment

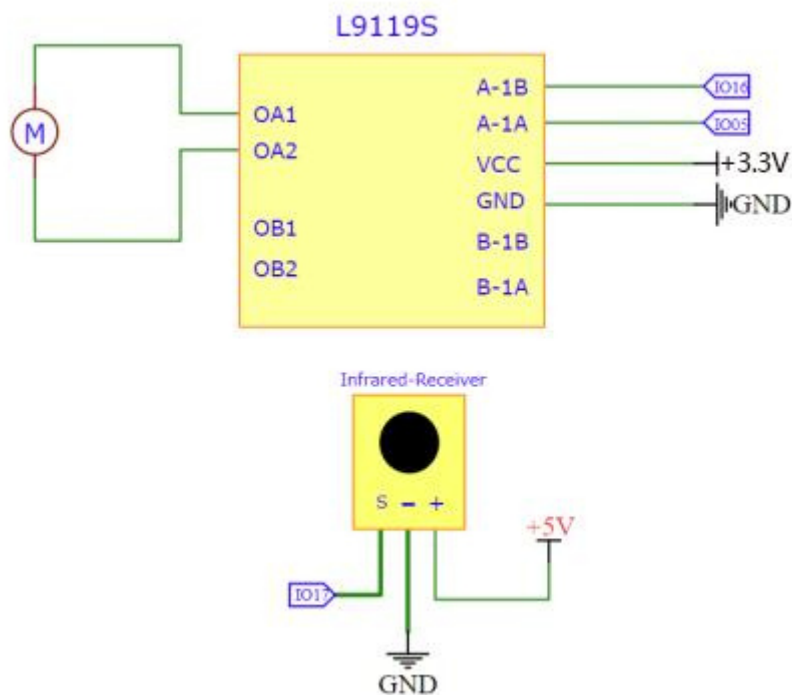
1. Popis projektu

Pro ovládání ventilátoru pomocí infračerveného dálkového ovladače po stisknutí tlačítka na dálkovém ovladači řídicí deska ESP32 přijme a dekóduje infračervený signál a provede odpovídající operaci na základě kódu klávesy: stisknutím tlačítka „0“ se ventilátor zastaví, stisknutím tlačítka „1“ se ventilátor zpomalí a stisknutím tlačítka „2“ se ventilátor zrychlí.

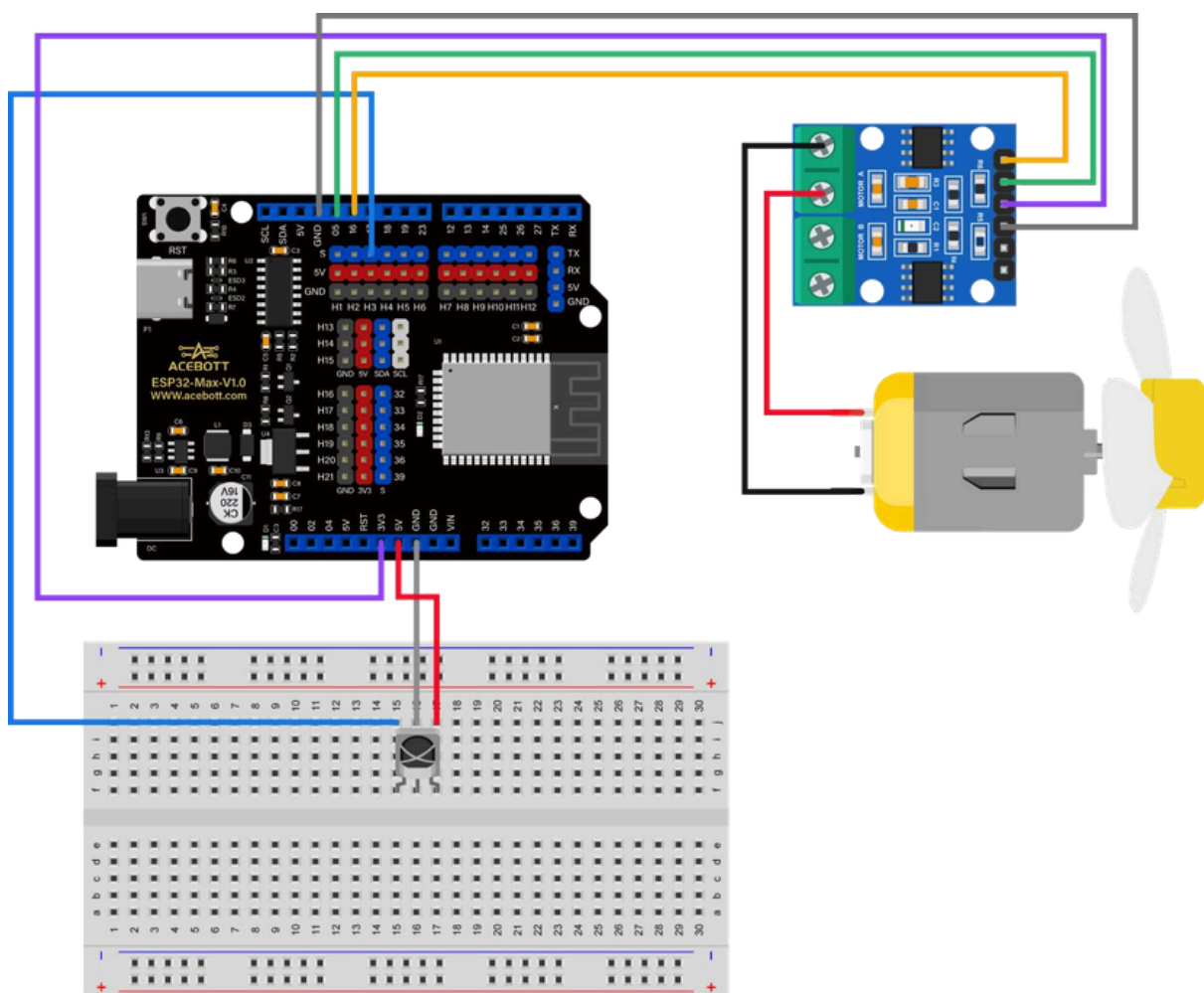
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Ovladač stejnosměrného motoru	1
	Stejnoseměrný motor	1
	Větrák	1
	IR přijímač	1
	IR dálkový ovladač	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[IRRemoteFan.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 29 IRRemoteFan\IRRemoteFan" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

```
#include <IRremote.h>

#define IR_RECEIVE_PIN 17 // IR receiver pin
#define motor1A 5 // Fan forward control pin (PWM)
#define motor2A 16 // Fan reverse control pin (keep at 0 for one-directional spin)

int fanSpeed = 0; // Current fan speed (range: 0-255)
int lastSpeed = -1; // Last recorded speed (used to avoid redundant writes)

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

  // Initialize motor control pins
  pinMode(motor1A, OUTPUT);
  pinMode(motor2A, OUTPUT);
  analogWrite(motor1A, 0); // Set initial fan speed to 0
  analogWrite(motor2A, 0); // Ensure reverse pin is 0

  // Initialize the IR receiver
  IrReceiver.begin(IR_RECEIVE_PIN);
  Serial.println("IR Receiver ready. Please press a button on the remote.");
}

void loop() {
  // Check if an IR signal is received
  if (IrReceiver.decode()) {
    uint8_t command = IrReceiver.decodedIRData.command;
    Serial.print("Received IR Code: 0x");
    Serial.println(command, HEX);

    // Map specific IR codes to fan speed levels
    switch (command) {
      case 0x52: // "Stop" button: turn off the fan
        fanSpeed = 0;
        break;
      case 0x16: // "Speed 1" button: 70% speed
        fanSpeed = 179;
```

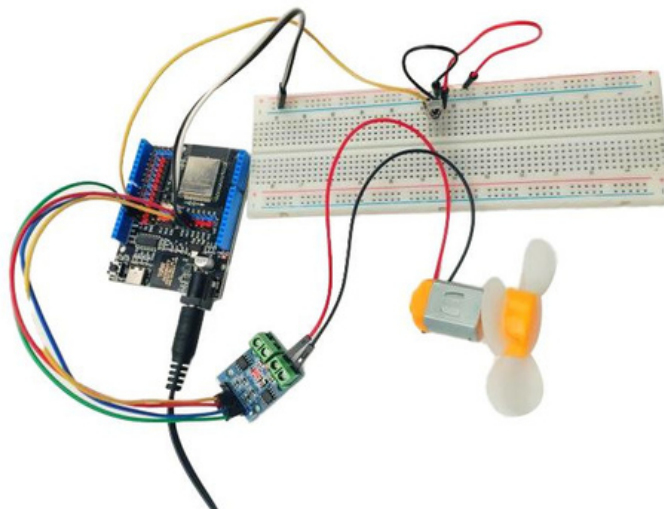
```
break;
case 0x19: // "Speed 2" button: 100% speed
fanSpeed = 255;
break;
default:
// Ignore unsupported IR codes
break;
}

// If the fan speed has changed, update PWM output
if (fanSpeed != lastSpeed) {
analogWrite(motor1A, fanSpeed);
analogWrite(motor2A, 0); // Keep reverse pin off
Serial.print("Fan speed set to: ");
Serial.print(fanSpeed);
Serial.println("/255");
lastSpeed = fanSpeed;
}

IrReceiver.resume(); // Prepare to receive the next IR signal
}

delay(50); // Small delay for basic debounce
}
```

Po úspěšném nahrání programu stiskněte tlačítko „0“ pro zastavení ventilátoru, stiskněte tlačítko „1“ pro spuštění ventilátoru na nízkou rychlost a stiskněte tlačítko „2“ pro spuštění ventilátoru na vysokou rychlost.



Lekce 30 Monitorování environmentálních dat



Cíle kurzu

- Pochopte principy fungování systému monitorování dat
- Napište program pro implementaci funkcí systému pro monitorování životního prostředí



Úvod do kurzu

V situacích, jako je chytré zemědělství, chytré domy a skleníky, je monitorování environmentálních dat v reálném čase klíčové pro udržení stabilních životních a produkčních podmínek. Například růst rostlin vyžaduje vhodné světlo, teplotu a vlhkost; nadměrně vysoké nebo nízké teploty mohou ovlivnit výnosy plodin. Například v chytré domácnosti, když je vnitřní teplota příliš vysoká, systém monitorování prostředí automaticky zapne klimatizaci, aby ji ochladil; když je světlo nedostatečné, inteligentní osvětlovací systém automaticky dodá další světlo, čímž vytvoří pohodlné a energeticky úsporné životní prostředí.



V této lekci se naučíme, jak pomocí senzorů shromažďovat informace o intenzitě světla, teplotě a vlhkosti a zobrazovat je v reálném čase na LCD obrazovce, abychom vytvořili systém pro monitorování a zobrazování stavu prostředí.



Body znalostí

1. Systém pro monitorování a zobrazování dat

Systém pro monitorování a zobrazování dat shromažďuje data pomocí senzorů a zobrazuje výsledky monitorování v reálném čase na zobrazovacím zařízení, což umožňuje pozorování a správu různých změn dat.

Principy fungování systému pro monitorování a zobrazování dat

Systém pro monitorování a zobrazování dat využívá senzory ke shromažďování informací o fyzikálních veličinách v prostředí v reálném čase, jako je intenzita světla, teplota a vlhkost. Konkrétní proces zahrnuje následující kroky:

(1) Fáze snímání dat ze senzoru

Fotorezistor mění svůj odpor v závislosti na intenzitě okolního světla a vydává analogové napětí; DHT11 periodicky měří teplotu a vlhkost a vydává digitální signál.

(2) Převod a čtení signálu

Analogový signál je převeden na digitální signál pomocí ADC a mikrokontrolér jej čte a následně zpracovává.

(3) Fáze zpracování dat

Mikrokontrolér filtruje a ověřuje shromážděná data, odstraňuje abnormální hodnoty a převádí je na zobrazitelné hodnoty.

(4) Fáze zobrazení dat

Data se zobrazují v reálném čase na LCD displeji v digitální i textové podobě, což uživatelům usnadňuje sledování parametrů prostředí.

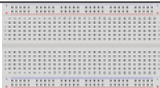
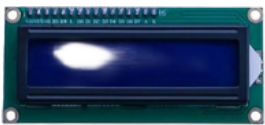


Experiment

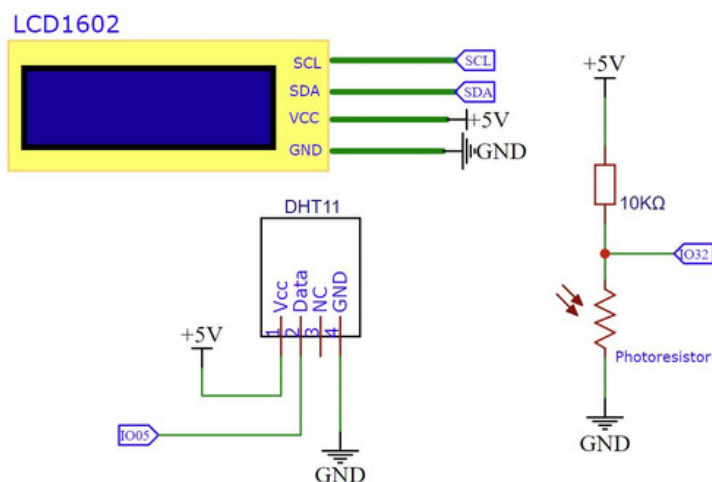
1. Popis projektu

Systém shromažďuje v reálném čase aktuální teplotu, vlhkost a intenzitu osvětlení prostředí a tyto informace dynamicky a v reálném čase zobrazuje na obrazovce LCD1602.

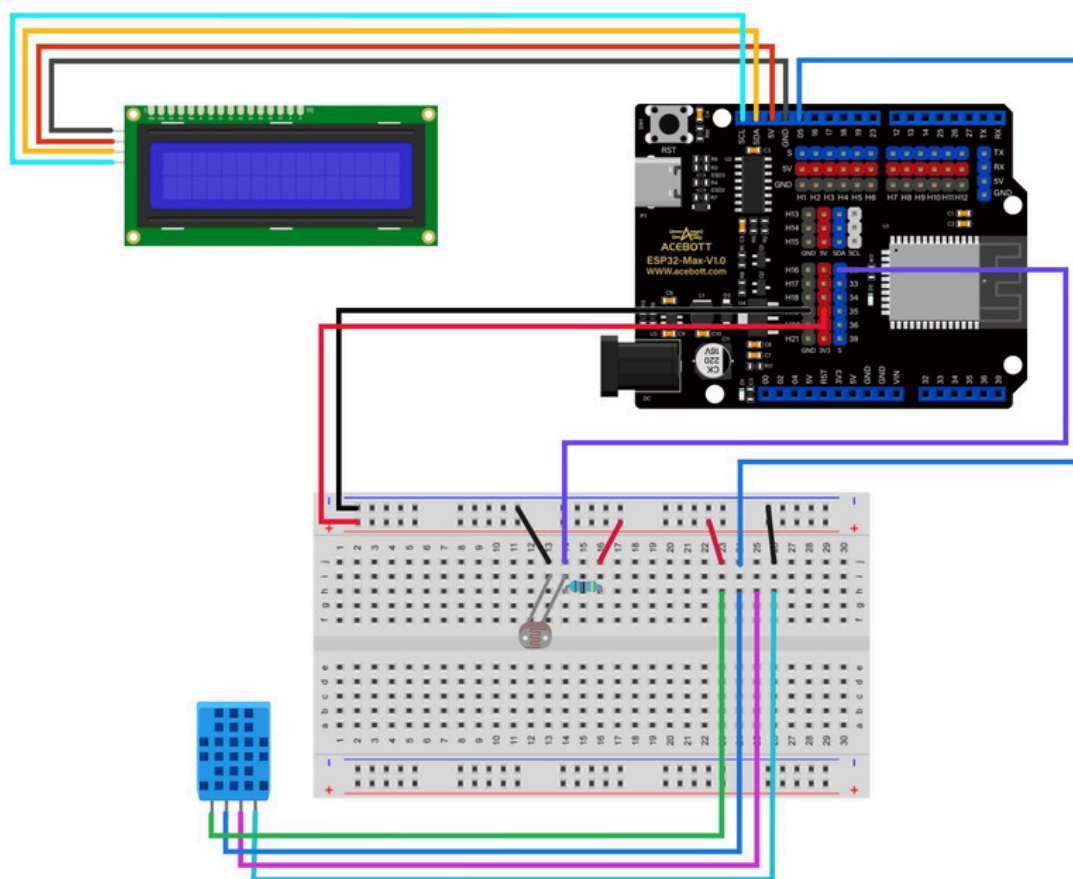
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	Snímač teploty a vlhkosti	1
	Fotorezistor	1
	LCD1602	1
	Rezistor 10 kΩ	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Systém monitorování životního prostředí.ino](#)" soubor v "angličtině

\\Arduino(zkušený Žák)\\2.Kód\\lekce 30 Životní prostředí

Monitorovací systém\\Systém_monitorování_životního_prostředí\\" nebo zkopírujte

následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

// Define LCD I2C address and size (16 columns, 2 rows)
// Adjust address if needed (0x27 or 0x3F)
LiquidCrystal_I2C lcd(0x27, 16, 2);
// DHT11 sensor configuration
#define DHTPIN 5 // Pin connected to the DHT11 data pin
#define DHTTYPE DHT11 // Define sensor type
DHT dht(DHTPIN, DHTTYPE); // Create DHT sensor object
// Photoresistor (LDR) analog pin
#define Photoresistor 32

void setup() {
  Serial.begin(115200); // Initialize serial communication for debugging
  // Initialize LCD
  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  // Initialize DHT sensor
  dht.begin();
  // Set the photoresistor pin as input
  pinMode(Photoresistor, INPUT);
}

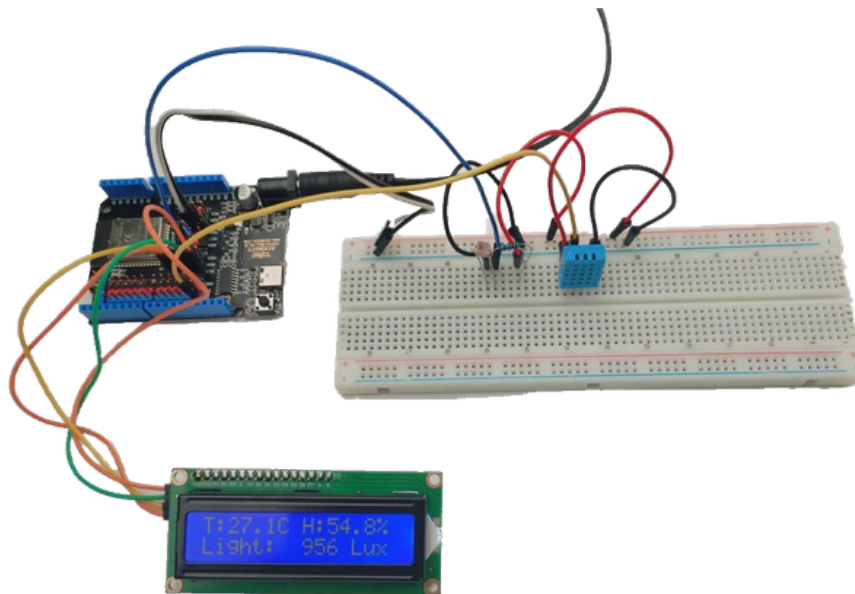
void loop() {
  // Read humidity and temperature from DHT11
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius
  // Read light intensity from photoresistor (map 0-4095 to 1000-0 for Lux approximation)
  int light_value = map(analogRead(Photoresistor), 0, 4095, 1000, 0);
  // Check if DHT sensor reading failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.setCursor(0, 0);
    lcd.print("Sensor Error ");
    delay(2000);
    return;
  }
  // Debug output to Serial Monitor
  Serial.printf("Temp: %.1f°C Humi: %.1f%% Light: %d Lux\n", temperature, humidity, light_value);
```



```
// Update LCD display with temperature and humidity
lcd.setCursor(0, 0);
lcd.printf("T:%.1fC H:%.1f%%", temperature, humidity); // Line 1:
Temperature and Humidity
// Display light intensity in Lux
lcd.setCursor(0, 1);
lcd.printf("Light: %4d Lux", light_value); // Line 2: Light level
delay(1000); // Wait 1 second before next update
}
```

Po úspěšném nahrání programu LCD1602 zobrazuje teplotu, vlhkost a intenzitu světla v reálném čase.

Jak je uvedeno níže:



6. Popis instrukcí

mapa(x, odNízké, odVysoké, doNízké, doVysoké) Používá se k mapování hodnoty x z původního rozsahu [fromLow, fromHigh] do cílového rozsahu [toLow, toHigh].

Lekce 31 Systém radarového poplašného systému



Cíle kurzu

- Pochopíte princip fungování radarového alarmu
- Napište program pro implementaci funkcí radarového alarmu



Úvod do kurzu

V reálném životě se alarmy se snímáním vzdálenosti široce používají v oblasti inteligentní kontroly přístupu, parkovacích asistenčních systémů a bezpečnostního monitorování.



V této lekci se naučíme, jak pomocí ultrazvukového senzoru detekovat vzdálenost k překážce vpředu a zobrazit výsledky měření v reálném čase na LCD1602. Když překážka vstoupí do přednastaveného varovného rozsahu, ozve se bzučák a rozsvítí se LED dioda, čímž se aktivuje systém varování před radarovým signálem.



Body znalostí

1. Radarový poplašný systém

Radarový alarmový systém využívá ultrazvukové nebo milimetrové radarové senzory k detekci, zda se před vámi blíží objekt. Jakmile cíl vstoupí do přednastaveného dosahu, systém spustí alarm.

Jak funguje radarový alarm

Systém radarového alarmu vysílá a přijímá ultrazvukové signály, měří dobu letu překážky a vypočítává vzdálenost mezi překážkou a senzorem. Když detekovaná vzdálenost klesne pod přednastavený varovný rozsah, systém spustí alarm a rozsvítí kontrolku. Displej také zobrazuje informace o vzdálenosti v reálném čase, efektivně monitoruje okolní prostředí a poskytuje bezpečnostní připomenutí.



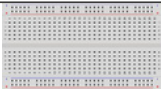
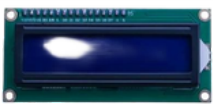
Experiment

1. Popis projektu

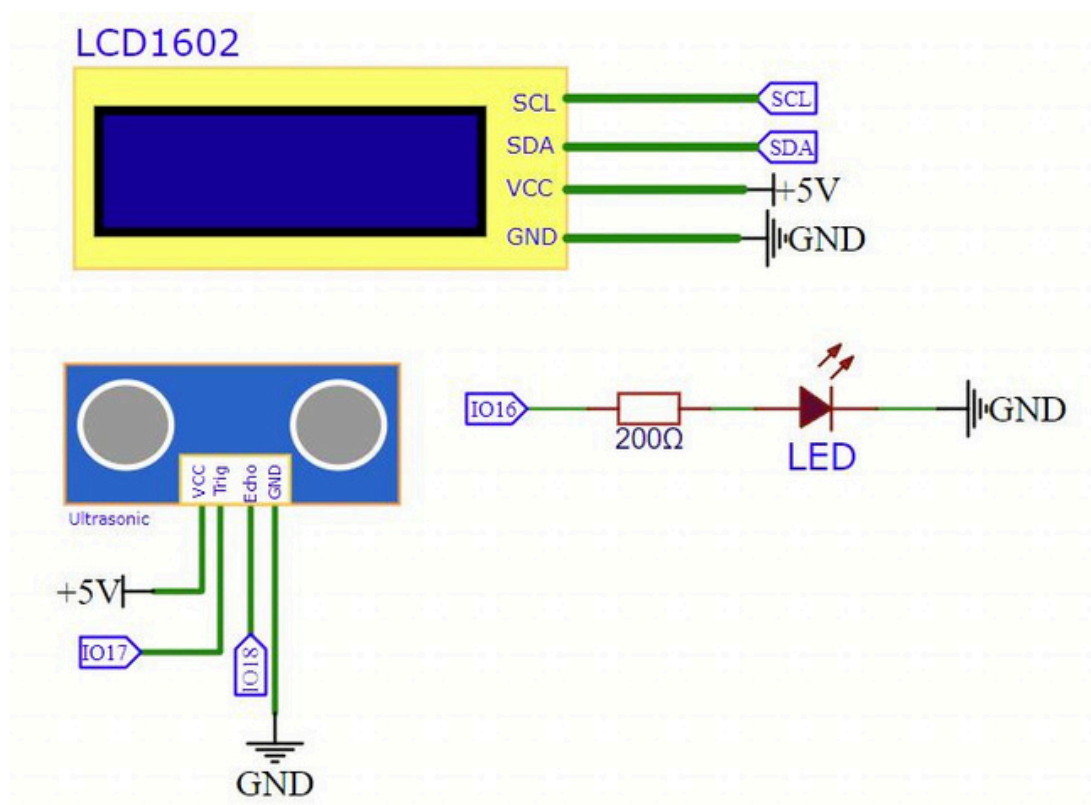
Ultrazvukový senzor měří v reálném čase vzdálenost k překážce vpředu a zobrazuje vzdálenost k objektu na LCD1602 displeji. Pokud je vzdálenost menší než 15 cm, systém automaticky rozsvítí LED diodu a spustí bzučák, který spustí alarm. Zároveň se na druhém řádku obrazovky zobrazí nápis „VAROVÁNÍ“, čímž se aktivuje radarový alarm.

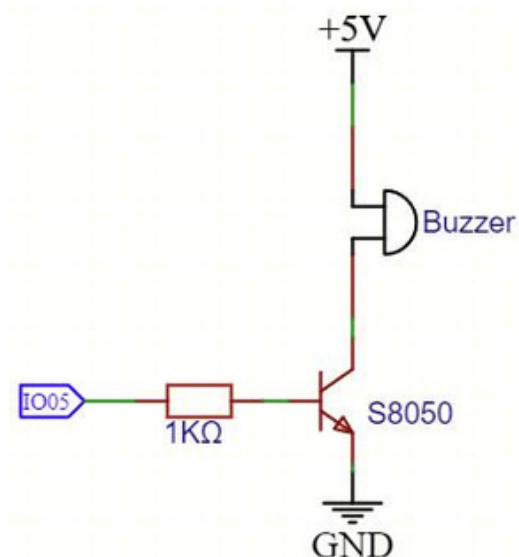
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1

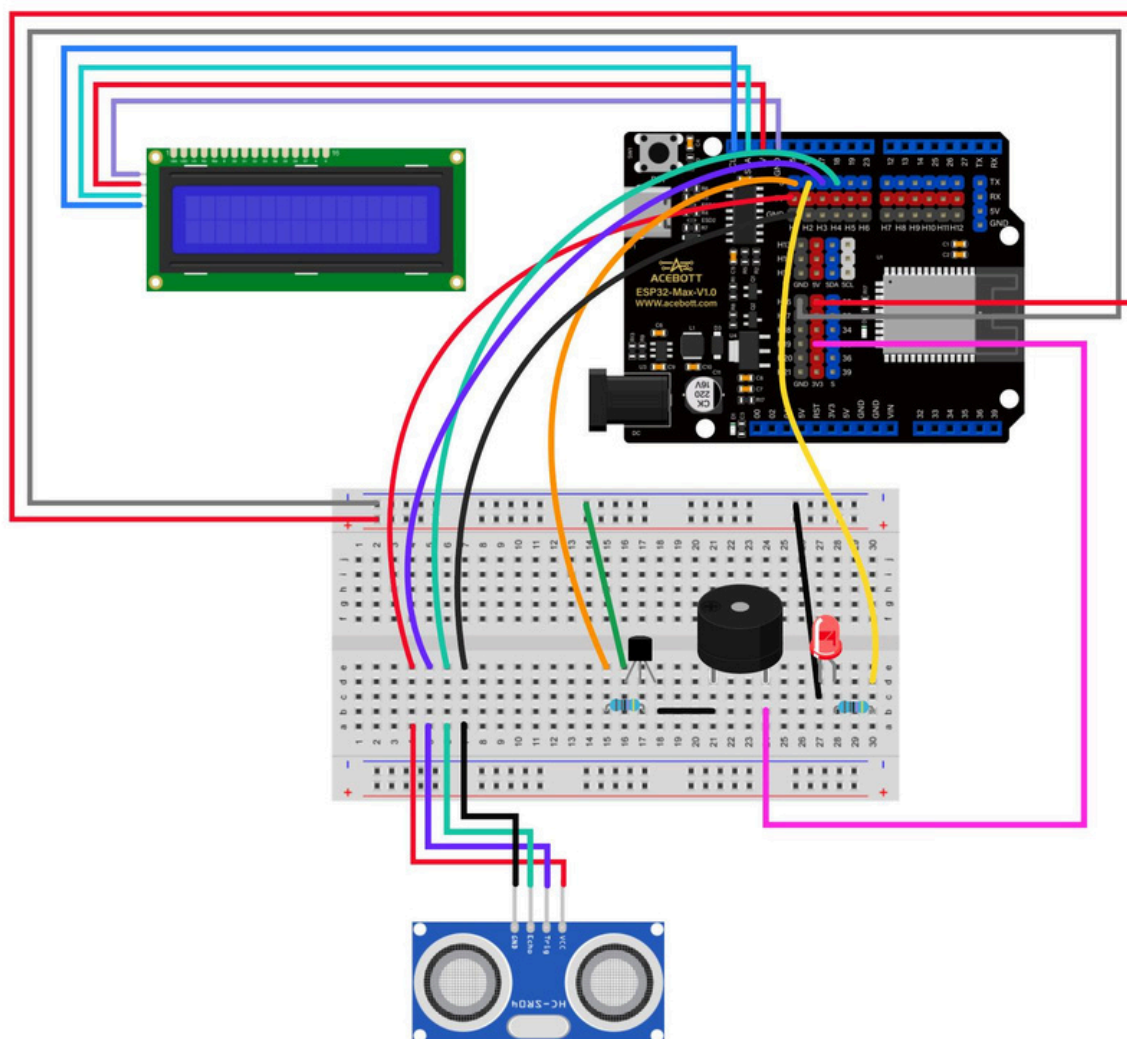
	Nepájivé pole s 400 spojovacími body	1
	LCD1602	1
	Ultrazvukový senzor	1
	Aktivní bzučák	1
	LED	1
	Rezistor 220Ω	1

3. Schéma zapojení





4. Hardwarové připojení



5. Referenční program

Otevřete "Radar_Alarm_System.ino" v "Angličtina \Arduino(Zkušený Žák)\2.Kód\lekce 31 RadarPoplach System\Radar_Alarm_System" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků klikněte na  nahrát program.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin definitions
const int trigPin = 17;
const int echoPin = 18;
const int buzzerPin = 5;
const int ledPin = 16;
// LCD1602 object (I2C address 0x27, 16 columns and 2 rows)
LiquidCrystal_I2C lcd(0x27, 16, 2);

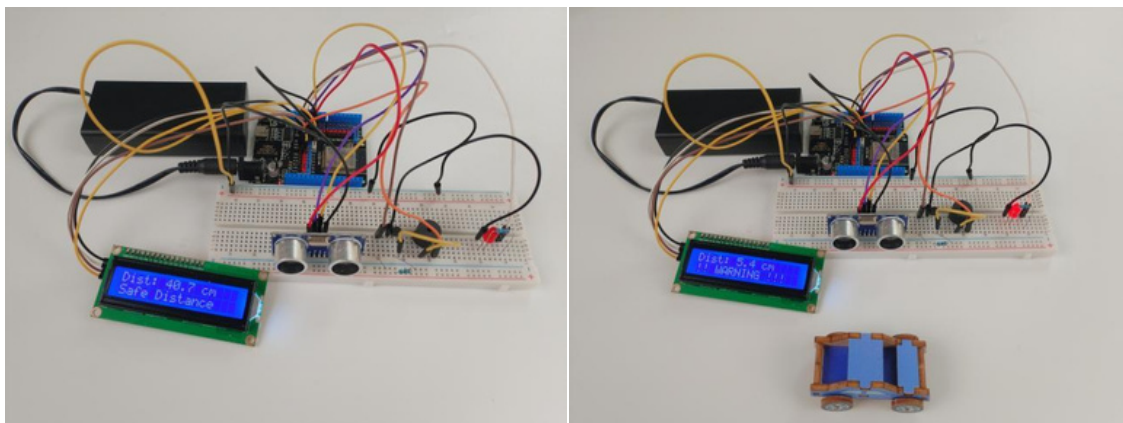
void setup() {
  // Initialize serial communication
  Serial.begin(115200);
  // Configure pin modes
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  // Initialize the LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Radar System Init");
  delay(1000);
  lcd.clear();
}

void loop() {
  long duration;
  float distance;
  // Send ultrasonic pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Read the duration of the echo pulse
```



```
duration = pulseIn(echoPin, HIGH);  
// Calculate distance in centimeters  
distance = (duration * 0.0343) / 2;  
// Print distance to serial monitor  
Serial.print("Distance: ");  
Serial.print(distance);  
Serial.println(" cm");  
// Display distance on the LCD  
lcd.setCursor(0, 0);  
lcd.print("Dist: ");  
lcd.print(distance, 1); // 1 decimal place  
lcd.print(" cm ");  
// Trigger alarm if object is too close  
if (distance < 15.0) {  
  digitalWrite(buzzerPin, HIGH);  
  digitalWrite(ledPin, HIGH);  
  lcd.setCursor(0, 1);  
  lcd.print("!! WARNING !!!  ");  
} else {  
  digitalWrite(buzzerPin, LOW);  
  digitalWrite(ledPin, LOW);  
  lcd.setCursor(0, 1);  
  lcd.print("Safe Distance  ");  
}  
delay(500); // Wait 0.5 seconds before next measurement  
}
```

Po úspěšném nahrání programu se na LCD1602 v reálném čase zobrazí vzdálenost k překážce před vámi. Když se překážka objeví v dosahu 15 cm od ultrazvukového senzoru, rozsvítí se LED dioda a bzučák vydá nepřetržitý alarm.



Lekce 32 Bluetooth světlo



Cíle kurzu

- Pochopení základních principů komunikace Bluetooth. Zvládnutí
- používání funkce Bluetooth řídicí desky ESP32. Dokončení návrhu
- „Bluetooth Light“.



Úvod do kurzu

Tradiční světla lze zapínat a vypínat pouze ručně, zatímco chytrá světla lze ovládat bezdrátově na dálku. Mezi těmito inteligentními řešeními ovládání je komunikační technologie Bluetooth velmi běžnou technologií bezdrátové komunikace s krátkým dosahem. Tato technologie se široce používá v každodenním životě, například v Bluetooth headsetech, Bluetooth reproduktorech a Bluetooth v autě.



V této lekci se naučíme, jak používat vestavěnou funkci Bluetooth v ESP32 pro komunikaci s mobilním telefonem.

analyzovat signály Bluetooth a ovládat zapínání a vypínání LED diody pomocí kódu, čímž položit základy pro komplexnější inteligentní řízení.



Body znalostí

1. Komunikace přes Bluetooth

Bluetooth je bezdrátová komunikační technologie krátkého dosahu určená pro nízkonákladovou bezdrátovou výměnu dat mezi zařízeními s nízkou spotřebou energie. Běžně se používá v mobilních telefonech, počítačích, sluchátkách, zařízeních chytré domácnosti atd.

Typy Bluetooth:

Klasický Bluetooth: Vhodný pro vysokorychlostní přenos dat, jako je přehrávání zvuku a přenos souborů.

Bluetooth Low Energy (BLE): Vhodné pro periodickou datovou komunikaci s nízkou šířkou pásma mezi zařízeními s nízkou spotřebou energie, jako je sběr dat ze senzorů a chytrá nositelná zařízení.

Jak funguje Bluetooth:

Bluetooth využívá rádiové vlny ve frekvenčním pásmu 2,4 GHz pro bezdrátovou komunikaci mezi zařízeními na krátkou vzdálenost. Používá strukturu master-slave. Hlavní zařízení iniciuje spojení a řídí komunikaci. Podřízené zařízení naváže spojení poté, co je objeveno prostřednictvím pravidelného vysílání. Po navázání spojení si obě strany vyměňují data prostřednictvím definovaných služeb a charakteristik, které podporují čtení, zápis a upozorňování dat, čímž se zajistí přenos informací mezi zařízeními v reálném čase.

Proces komunikace Bluetooth:

(1) Reklama Podřízené zařízení pravidelně odesílá reklamní pakety a hlavní zařízení tyto pakety prohledává, aby našlo cílové zařízení.

(2) Párování a vytváření vazby Při navazování spojení mohou být pro zajištění bezpečnosti komunikace použity procesy ověřování a šifrování.

(3) Připojení: Poté, co se hlavní zařízení připojí k podřízenému zařízení, si obě strany mohou vyměňovat data.

(4) Odpojení: Spojení je ukončeno po dokončení komunikace nebo po uplynutí časového limitu.

2. Kontrolka Bluetooth

Bluetooth světlo je lampa, kterou lze bezdrátově ovládat pomocí mobilního telefonu nebo jiného zařízení Bluetooth. Využívá komunikaci Bluetooth k umožnění funkcí, jako je zapínání a vypínání světla, úprava jasů a změna barvy. Kombinuje bezdrátové ovládání s technologií inteligentního osvětlení a je běžnou metodou ovládání v chytrých domácnostech.

Jak fungují světla Bluetooth

Bluetooth lampa využívá vestavěný Bluetooth modul řídicí desky ESP32 k dosažení bezdrátové komunikace s mobilními telefony nebo jinými Bluetooth zařízeními. Pracovní proces zahrnuje hlavně následující kroky:

1. Párování Bluetooth Řídicí deska (například ESP32) v kontrolce Bluetooth zapne funkci Bluetooth a čeká na připojení externího zařízení (například mobilního telefonu). Mobilní telefon vyhledá a spáruje se přes Bluetooth, aby navázal komunikaci.

kanál.

2. Odesílání a přijímání příkazů Uživatel odešle příkaz pro ovládání světla prostřednictvím mobilní aplikace a řídicí deska ESP32 tato data příkazu přijme prostřednictvím modulu Bluetooth.

3. Analýza příkazů Řídicí program analyzuje přijatá data Bluetooth a identifikuje odpovídající operace pro různé příkazy.

4. Provedení řízení osvětlení: Podle příkazu se LED světlo zapne nebo vypne.

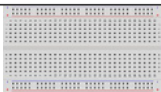


Experiment

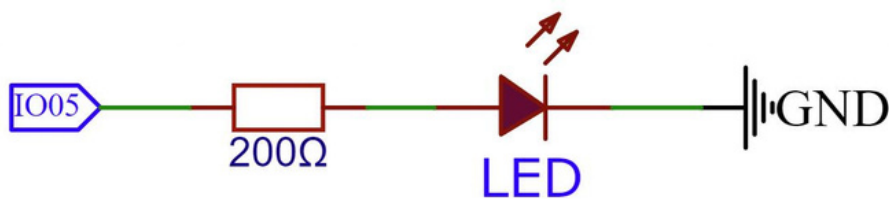
1. Popis projektu

Ovládejte LED světlo na dálku pomocí funkce Bluetooth zařízení ESP32. Když řídicí deska ESP32 přijme přes Bluetooth signál 0, LED se rozsvítí; když přijme signál 1, LED se vypne.

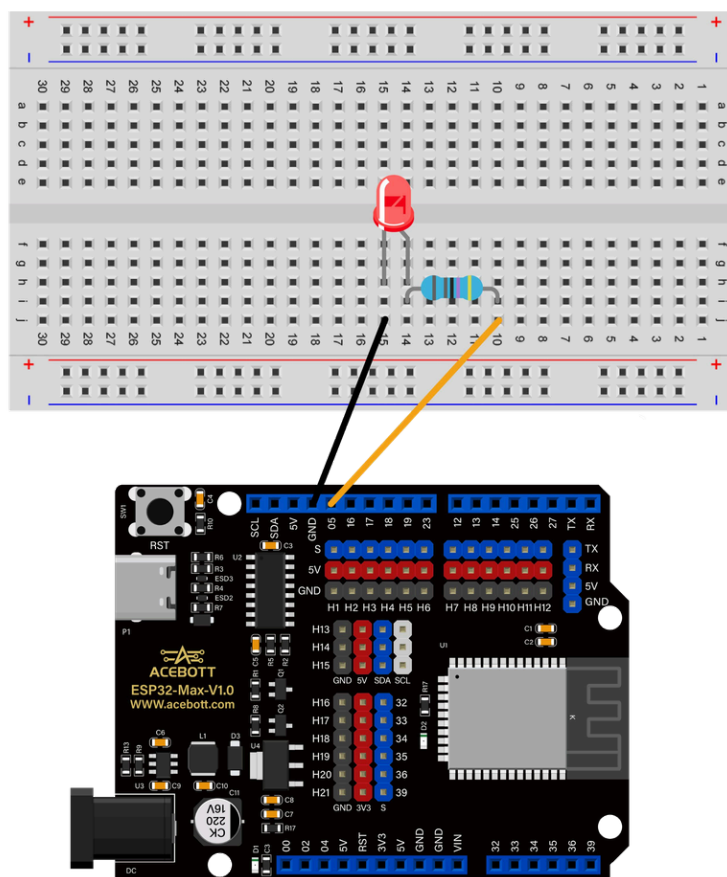
2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Rezistor 220Ω	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[BluetoothLED.ino](#)" soubor v "English\Arduino(Experienced Learner)\2.Code\lesson 32 BluetoothLED\BluetoothLED" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle

dříve naučené kroky, klikněte  nahrát program.

```
#include "BluetoothSerial.h"
#define LED 5 // Declare the LED pin
BluetoothSerial SerialBT; // Create a Bluetooth object
```

```
void setup() {  
  pinMode(LED, OUTPUT); // Set the LED pin to output mode  
  Serial.begin(115200);  
  SerialBT.begin("Bluetooth LED"); // Enable Bluetooth and set the Bluetooth  
  name  
  delay(1000);  
  Serial.println("The device started, now you can pair it with bluetooth!");  
}  
  
void loop() {  
  int message;  
  if (SerialBT.available()) { // Check if data is received from other devices:  
    message = SerialBT.read(); // Read the data  
    Serial.println(message);  
  }  
  if (message == 0) { // If the read data is equal to 0  
    digitalWrite(LED, HIGH); // Turn on the LED  
  }  
  if (message == 1) { // If the read data is equal to 1  
    digitalWrite(LED, LOW); // Turn off the LED  
  }  
  delay(20);  
}
```

ESP32入门学习套件 > 5.APP

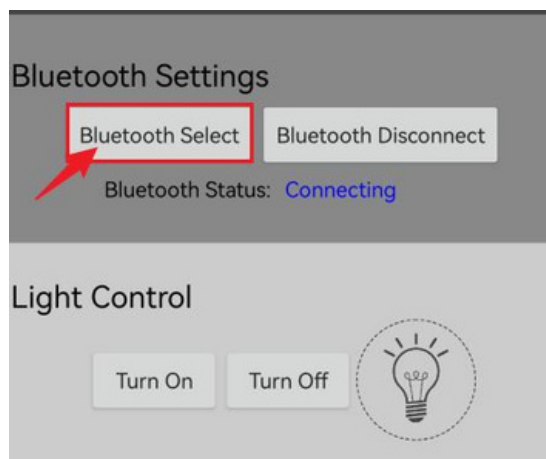
Name	Date modified	Type	Size
 Bluetooth_light.apk	18/06/2025 17:43	APK File	3,987 KB

Po úspěšné instalaci se na ploše telefonu zobrazí ikona aplikace Bluetooth Light. Kliknutím na ikonu otevřete aplikaci Bluetooth Light a můžete ovládat LED diodu.

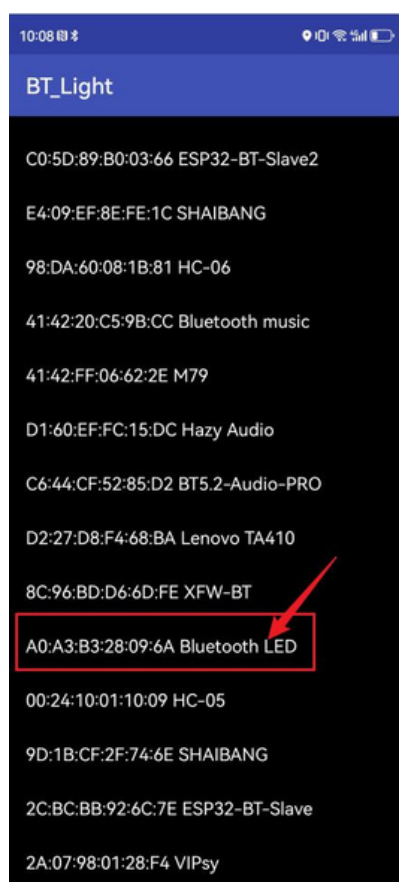


BT_Light

Po otevření aplikace klikněte na „Vybrat Bluetooth“.

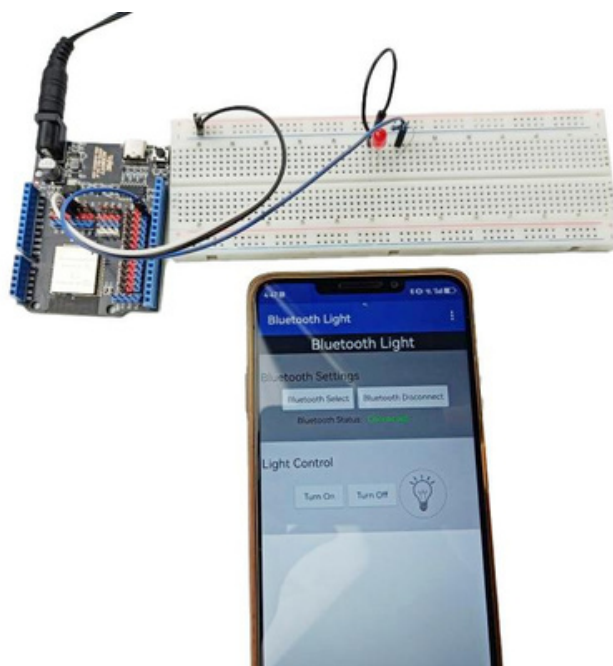
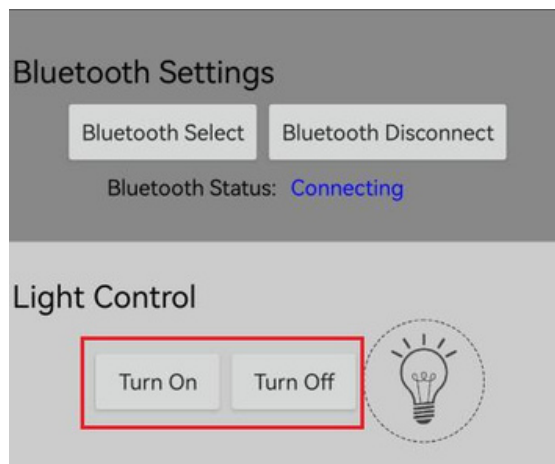


V zobrazeném seznamu Bluetooth vyhledejte název Bluetooth kontrolky „Bluetooth LED“.



Po úspěšném připojení Bluetooth klikněte na „Zapnout“ a „Zapnout“

Vypnuto" v ladicím rozhraní pro ovládání LED světla a sledování, zda je LED světlo ovládáno podle očekávání.



6. Popis instrukcí

SerialBT.begin("Bluetooth LED"); Povolte Bluetooth a nastavte název Bluetooth.

SerialBT.available(); Zkontrolujte, zda jsou data přijímána z jiných zařízení.

SerialBT.read(); Čtení dat odeslaných zařízením Bluetooth.

Lekce 33 Ovládání webu



Cíle kurzu

- Pochopíte základní principy Wi-Fi komunikace
- Zvládněte používání Wi-Fi funkcí řídicí desky ESP32
- Implementujte funkci vzdáleného ovládání přes web

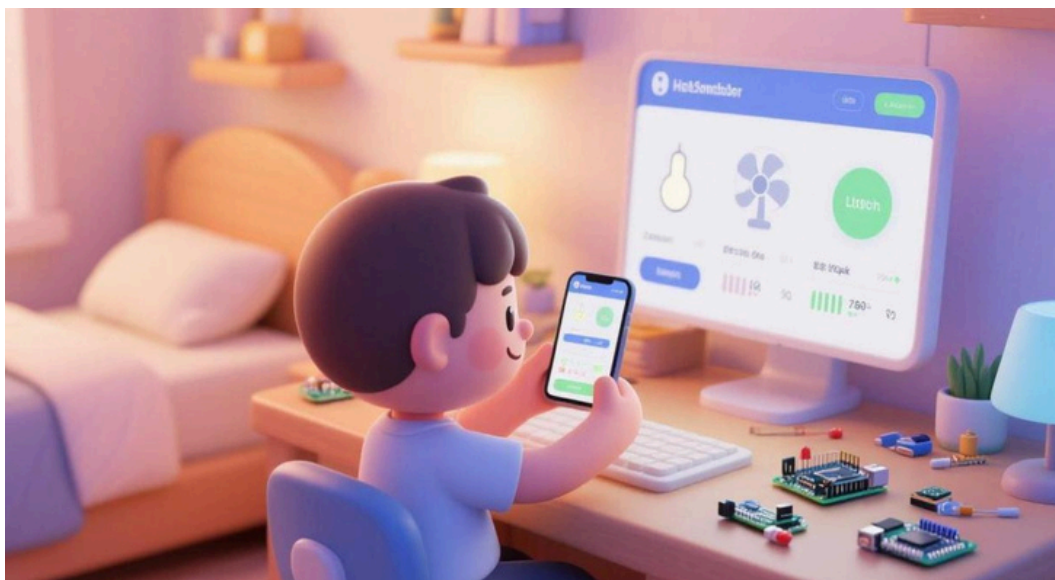


Úvod do kurzu

Představte si, že byste mohli ovládat světla, ventilátory nebo dokonce směr malého vozidla ve svém domě pouze pomocí telefonu nebo počítače.

webovou stránku – bez složitých kabelů, vše prostřednictvím prohlížeče. V tomto

V této lekci se naučíme, jak připojit elektronickou stavebnici ESP32 k síti a jak ji dálkově ovládat prostřednictvím webové stránky.



Tato lekce se bude zabývat tím, jak používat Wi-Fi funkci ESP32 k umožnění obousměrné komunikace mezi webovou stránkou a zařízením:

Můžete odesílat příkazy do ESP32 prostřednictvím webové stránky a také z ESP32 můžete načítat data ze senzorů.



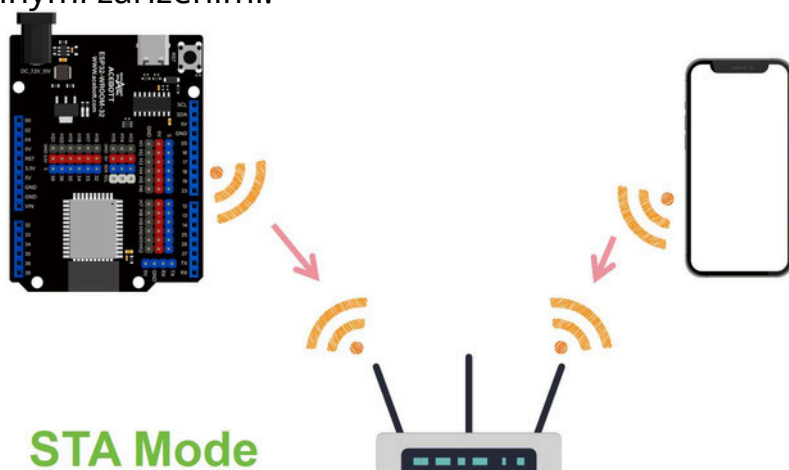
Body znalostí

1. Technologie komunikace Wi-Fi

Komunikační technologie Wi-Fi je typ bezdrátové lokální sítě (WLAN) vyvinuté na základě standardu IEEE 802.11, která poskytuje bezpečné, spolehlivé a rychlé bezdrátové připojení. Funkce Wi-Fi routeru ESP32 má tři provozní režimy:

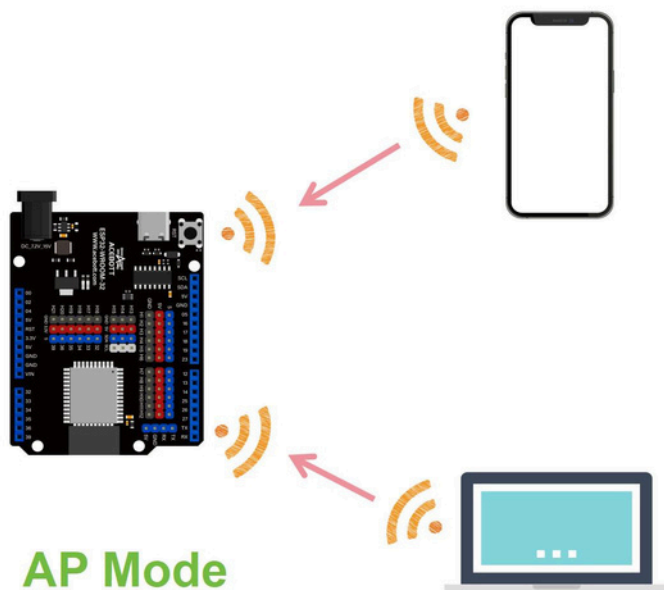
① Režim bezdrátové stanice (také nazývaný režim STA)

V tomto režimu se ESP32 připojuje jako koncové zařízení k síti Wi-Fi generované jinými zařízeními.



2 Režim přístupového bodu (také nazývaný režim AP)

V tomto režimu ESP32 funguje jako generátor Wi-Fi a vytváří si vlastní signál Wi-Fi sítě, ke kterému se mohou připojit další zařízení.



③ Režim bezdrátové stanice + režim přístupového bodu

V tomto režimu se ESP32 může připojit jako terminál k jiným Wi-Fi sítím a zároveň fungovat jako přístupový bod pro generování Wi-Fi sítě, ke které se mohou připojit další zařízení.

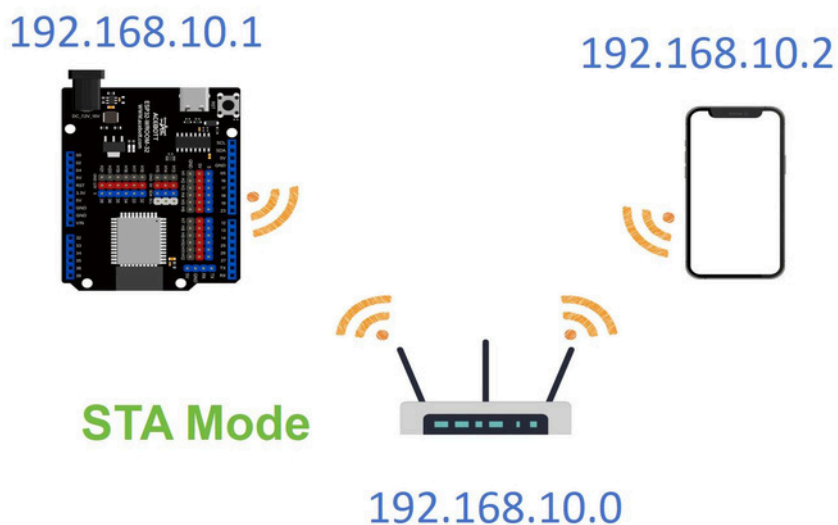
2. IP adresa

Síť Wi-Fi poskytuje komunikační kanál pro zařízení, ale přenos dat mezi zařízeními vyžaduje jasný identifikátor adresy, kterým je IP adresa. IP protokol má dvě verze: IPv4 a IPv6, přičemž IPv6 je upgradem IPv4, který může poskytovat více IP adres.

Tento tutoriál používá protokol IPv4, takže se zaměříme na IPv4. Adresa IPv4 se skládá ze čtyř čísel v rozsahu od 0 do 255 (např. 192.168.10.136), kde první tři čísla (např. 192.168.10) představují adresu segmentu sítě a poslední číslo (např. 136) představuje jedinečný identifikátor zařízení v dané síti.

Například poté, co router přiřadí IP adresu, ESP32 a

Mobilní telefony mohou mezi sebou komunikovat v rámci lokální sítě prostřednictvím svých příslušných IP adres.



3. Pracovní postup webové kontroly

Princip webové kontroly je založen na režimu AP ESP32 a protokolu HTTP, implementovaném prostřednictvím následujících základních procesů:

ESP32 vytváří Wi-Fi hotspot jako přístupový bod (režim AP);

Uživatelské zařízení se připojí k hotspotu a získá IP adresu; v

V režimu AP je ESP32 pevně nastaven na 192.168.4.1;

Prohlížeč přistupuje k IP adrese ESP32, aby načtl přednastavená ovládací stránka HTML;

Uživatel ovládá ovládací prvky stránky (např. posuvný přepínač) pro odeslání

Požadavek na kontrolu HTTP;

ESP32 analyzuje parametry požadavku a řídí hardware.

(LED/ventilátor) přes GPIO/PWM;

Data ze senzorů (např. každé 2 sekundy) se shromažďují na pozadí a dynamicky aktualizuje zobrazení webové stránky.

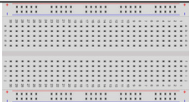


Experiment

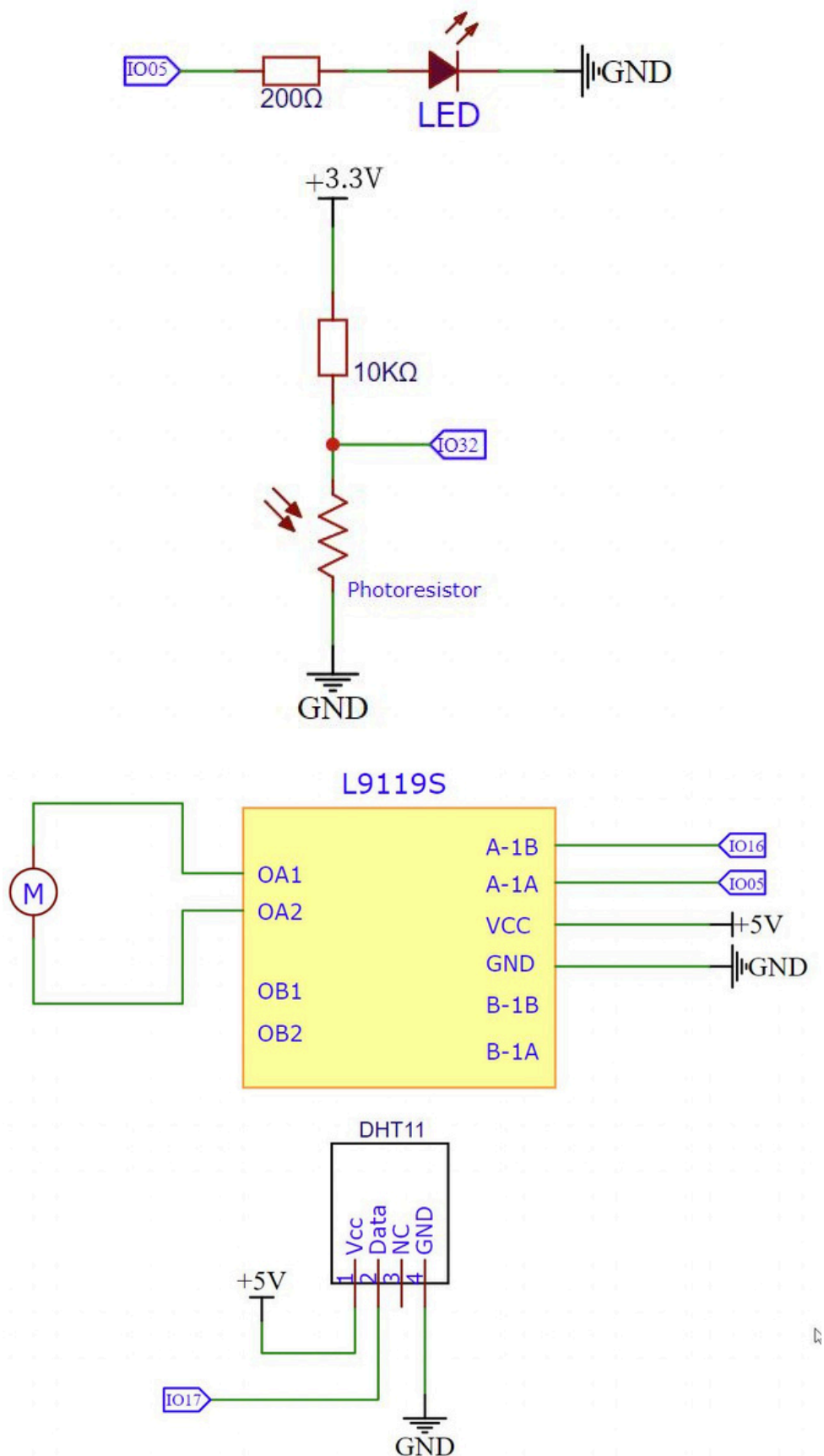
1. Popis projektu

Pomocí Wi-Fi funkce ESP32 můžete dálkově ovládat LED a ventilátor a zároveň načítat data ze světelného senzoru a senzoru teploty a vlhkosti na webové stránce a zobrazovat je na webové stránce.

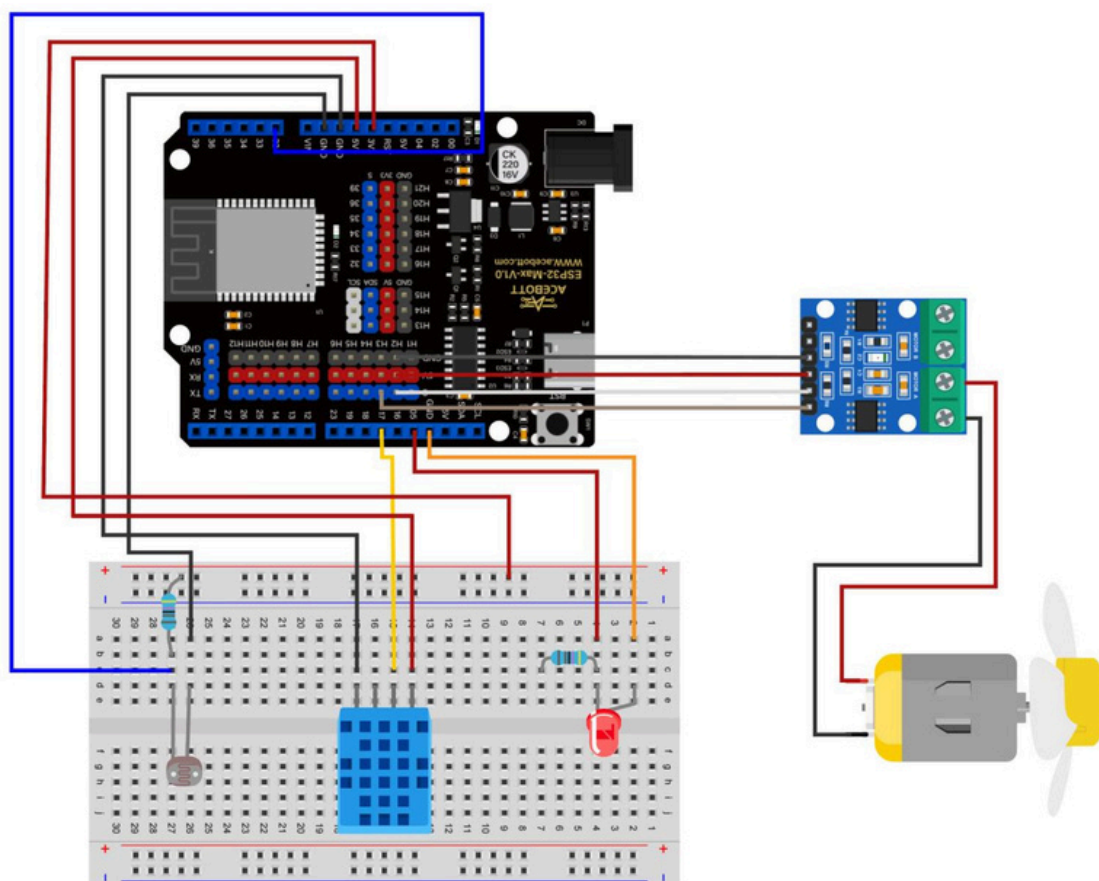
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Rezistor 220Ω	1
	Rezistor 10 kΩ	1
	Ovladač stejnosměrného motoru	1
	Stejnoseměrný motor	1
	Větrák	1
	Fotorezistor	1
	DHT11	1

3. Schéma zapojení



4. Hardwarové připojení



5. Referenční program

Otevřete "[Web_Control.ino](#)" v souboru "English\Arduino(Experienced Learner)\2.Code\lesson 33 Web Control" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků,

kliknutí  nahrát program.

```
#include <WiFi.h>
#include <WebServer.h>
#include <DHT.h>

// Define pins
#define motor1A 16
#define motor2A 17
#define LED_PIN 5
#define DHT_PIN 18
#define LIGHT_SENSOR_PIN 32
```

```
// Define DHT sensor type
#define DHT_TYPE DHT11
// Create objects
DHT dht(DHT_PIN, DHT_TYPE);
WebServer server(80);

// Store current states
bool fanState = false;
bool ledState = false;
float temperature = 0;
float humidity = 0;
int lightLevel = 0;

// PWM configuration
const int freq = 5000;
const int resolution = 8;
const int fanChannel1 = 0;
const int fanChannel2 = 1;

// WiFi hotspot configuration
const char* ssid = "Web_control";
const char* password = "12345678";

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

  // Configure PWM
  ledcSetup(fanChannel1, freq, resolution);
  ledcSetup(fanChannel2, freq, resolution);
  ledcAttachPin(motor1A, fanChannel1);
  ledcAttachPin(motor2A, fanChannel2);

  // Initialize pins
  pinMode(LED_PIN, OUTPUT);
  pinMode(LIGHT_SENSOR_PIN, INPUT);

  // Initialize sensors
  dht.begin();

  // Initialize device states
  ledcWrite(fanChannel1, 0);
  ledcWrite(fanChannel2, 0);
  digitalWrite(LED_PIN, LOW);

  // Create WiFi hotspot
  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
}
```

```
// Set server routes
server.on("/", handleRoot);
server.on("/control", handleControl);
server.on("/data", handleData);

// Start server
server.begin();
Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();

  // Update sensor data every 2 seconds
  static unsigned long lastUpdate = 0;
  if (millis() - lastUpdate > 2000) {
    updateSensorData();
    lastUpdate = millis();
  }
}

// Update sensor data
void updateSensorData() {
  // Read temperature and humidity
  float newTemp = dht.readTemperature();
  float newHumidity = dht.readHumidity();
  int newLightLevel = analogRead(LIGHT_SENSOR_PIN);

  // Only update values if read successfully
  if (!isnan(newTemp)) temperature = newTemp;
  if (!isnan(newHumidity)) humidity = newHumidity;
  lightLevel = newLightLevel;

  // Debug output
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print("°C, Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Light: ");
  Serial.println(lightLevel);
}

// Handle root request
void handleRoot() {
  // Set correct character encoding and content type
  String html = R"rawliteral(
<!DOCTYPE html>
<html>
```



```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>ESP32 Controller</title>
  <style>
    body { font-family: Arial, "PingFang SC", "Hiragino Sans GB",
"Microsoft YaHei", sans-serif; margin: 0 auto; padding: 20px; max-width:
600px; }
    .card { background: #f9f9f9; padding: 20px; margin: 10px 0;
border-radius: 10px; box-shadow: 0 2px 5px rgba(0,0,0,0.1); }
    .switch { position: relative; display: inline-block; width: 60px;
height: 34px; }
    .switch input { opacity: 0; width: 0; height: 0; }
    .slider { position: absolute; cursor: pointer; top: 0; left: 0; right:
0; bottom: 0; background-color: #ccc; transition: .4s; border-radius:
34px; }
    .slider:before { position: absolute; content: ""; height: 26px; width:
26px; left: 4px; bottom: 4px; background-color: white; transition: .4s;
border-radius: 50%; }
    input:checked + .slider { background-color: #2196F3; }
    input:checked + .slider:before { transform: translateX(26px); }
    .sensor-data { font-size: 24px; font-weight: bold; color: #333; }
    .auto-refresh { margin-top: 20px; }
    .error { color: red; }
  </style>
</head>
<body>
  <h1>ESP32 Device Controller</h1>

  <div class="card">
    <h2>Fan Control</h2>
    <label class="switch">
      <input type="checkbox" id="fanSwitch"
onchange="controlDevice('fan', this.checked)">
      <span class="slider"></span>
    </label>
    <span id="fanStatus">Off</span>
  </div>

  <div class="card">
    <h2>LED Control</h2>
    <label class="switch">
      <input type="checkbox" id="ledSwitch"
onchange="controlDevice('led', this.checked)">
      <span class="slider"></span>
    </label>
    <span id="ledStatus">Off</span>
  </div>

```

```

    <div class="card">
      <h2>Sensor Data</h2>
      <p>Temperature: <span id="temperature"
class="sensor-data">0</span> °C</p>
      <p>Humidity: <span id="humidity" class="sensor-data">0</span> %</p>
      <p>Light Level: <span id="light" class="sensor-data">0</span></p>
      <p id="dataError" class="error" style="display:none;">Failed to get
data</p>
    </div>

    <div class="auto-refresh">
      <p>Data updates automatically every 2 seconds</p>
    </div>

    <script>
      // Control device
      function controlDevice(device, state) {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "/control?device=" + device + "&state=" + (state ?
"on" : "off"), true);
        xhr.send();

        // Update status display
        document.getElementById(device + "Status").textContent = state ?
"On" : "Off";
      }

      // Get sensor data
      function updateSensorData() {
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function() {
          if (this.readyState == 4) {
            if (this.status == 200) {
              try {
                var data = JSON.parse(this.responseText);
                document.getElementById("temperature").textContent =
data.temperature.toFixed(1);
                document.getElementById("humidity").textContent =
data.humidity.toFixed(1);
                document.getElementById("light").textContent =
data.light;
                document.getElementById("dataError").style.display =
"none";
              } catch (e) {
                console.error("JSON parsing error:", e);
                document.getElementById("dataError").style.display =
"block";

```

```

        }
    } else {
        document.getElementById("dataError").style.display =
"block";
    }
}
};
xhr.onerror = function() {
    document.getElementById("dataError").style.display = "block";
};
xhr.open("GET", "/data", true);
xhr.send();
}

// Update sensor data every 2 seconds
setInterval(updateSensorData, 2000);

// Get initial data when page loads
window.onload = function() {
    updateSensorData();
};
</script>
</body>
</html>
)rawliteral";

// Send response with correct character set
server.setHeader("Content-Type", "text/html; charset=UTF-8");
server.send(200, "text/html", html);
}

// Handle control request
void handleControl() {
    String device = server.arg("device");
    String state = server.arg("state");

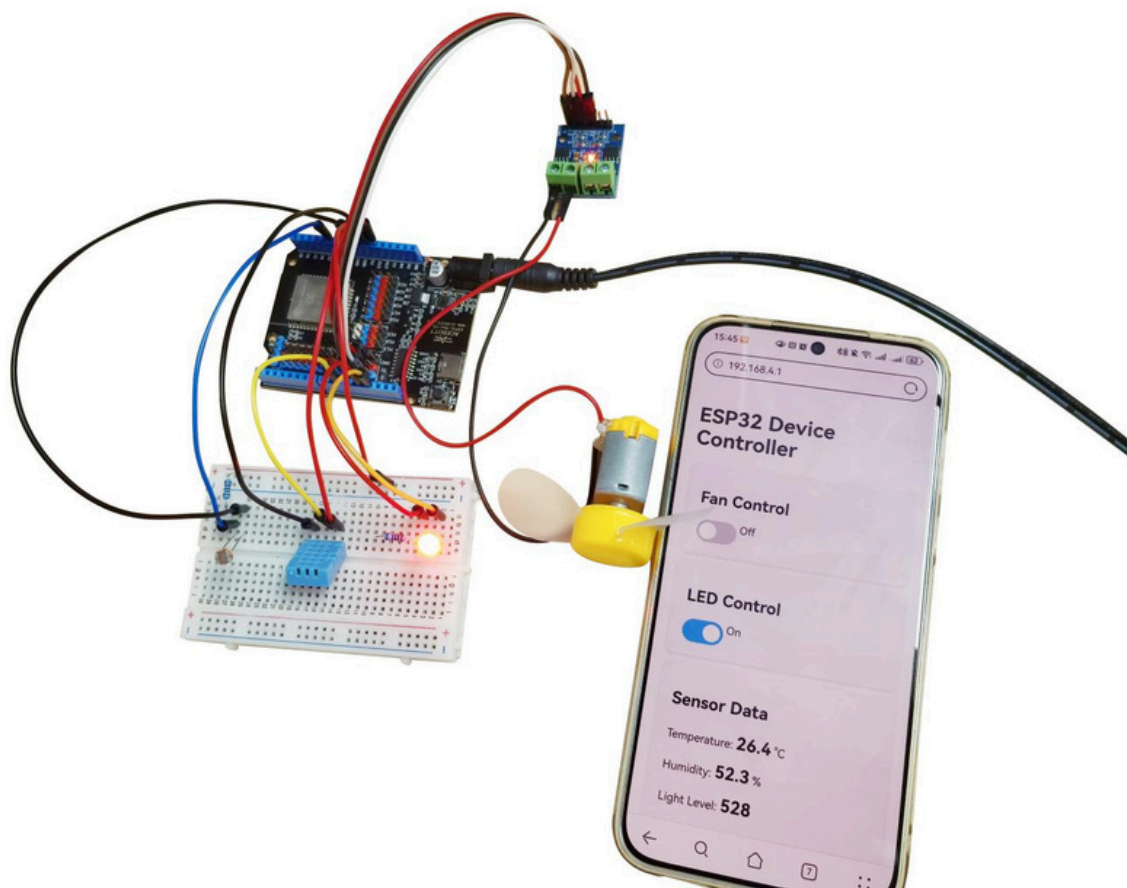
    if (device == "fan") {
        fanState = (state == "on");
        if(fanState){
            ledcWrite(fanChannel1, 255); // Full speed forward
            ledcWrite(fanChannel2, 0);
            Serial.println("Fan: On");
        } else {
            ledcWrite(fanChannel1, 0);
            ledcWrite(fanChannel2, 0);
            Serial.println("Fan: Off");
        }
    }
} else if (device == "led") {

```

```
    ledState = (state == "on");  
    digitalWrite(LED_PIN, ledState ? HIGH : LOW);  
    Serial.println("LED: " + state);  
}  
  
server.sendHeader("Content-Type", "text/plain; charset=UTF-8");  
server.send(200, "text/plain", "OK");  
}  
  
// Handle data request  
void handleData() {  
    String json = "{\"temperature\":\"" + String(temperature) +  
                  "\",\"humidity\":\"" + String(humidity) +  
                  "\",\"light\":\"" + String(lightLevel) + "\"}";  
  
    Serial.println("Sending JSON data: " + json); // Debug output  
  
    server.sendHeader("Content-Type", "application/json; charset=UTF-8");  
    server.send(200, "application/json", json);  
}
```

Po úspěšném nahrání programu otevřete nastavení Wi-Fi v mobilním telefonu, připojte se k síti Wi-Fi s názvem „WebControl“ s heslem „12345678“. Poté otevřete prohlížeč a do vyhledávacího řádku prohlížeče zadejte IP adresu 192.168.4.1 (v režimu AP je IP adresa ESP32 pevně nastavena na 192.168.4.1). Prohlížeč poté načte následující webovou stránku.

Tlačítka můžete ovládat LED diodu (zapnutí/vypnutí) a ventilátor (spuštění/zastavení). Zároveň se na webové stránce zobrazují údaje o teplotě a vlhkosti prostředí získané ze senzoru teploty a vlhkosti a také údaje o okolním osvětlení detekované světelným senzorem.



Kapitola 4 Projekty rozšíření

Tato kapitola je pokročilá a zaměřuje se na hlubokou integraci konstrukčních a řídicích systémů. Jejím cílem je rozvíjet inženýrské konstrukční myšlení a inovační dovednosti studentů. Stavbou inteligentních zařízení s mechanickými strukturami a interaktivní logikou si studenti prostřednictvím praktických cvičení zlepší dovednosti v oblasti systémové integrace a implementace projektů.

Poznámka: Obsah této kapitoly vyžaduje pro učení použití dřevěné tabule QE201. Sadu QE201 si prosím zakupte samostatně.

Lekce 34 Chytrý bionický motýl



Cíle kurzu

- Pochopíte koncept „bioniky“
- Naučte se sestavit chytrý bionický model motýla
- Napište program, který simuluje mávání křídel bionického motýla.



Úvod do kurzu

V přírodě motýli tančí nejen s elegancí, ale také s úžasnou energetickou účinností. Spoléhají se na svá lehká křídla k provádění složitých manévřů, jako je let, vznášení se a řízení. Vědci se těmito tvory inspirovali k návrhu různých bionických robotů.



V této lekci budeme simulovat třepotavé pohyby motýla a postavíme si vlastního „chytrého bionického motýla“.



Body znalostí

1. Biomimikry

Biomimikry jsou interdisciplinární disciplíny, které studují strukturu, funkci a pohyb živých organismů v přírodě a aplikují své principy na inženýrství, design a vývoj technologií. Pomáhají lidem řešit složité technické problémy napodobováním moudrosti přírody a představují dokonalou kombinaci „biologie“ a „inženýrství“.

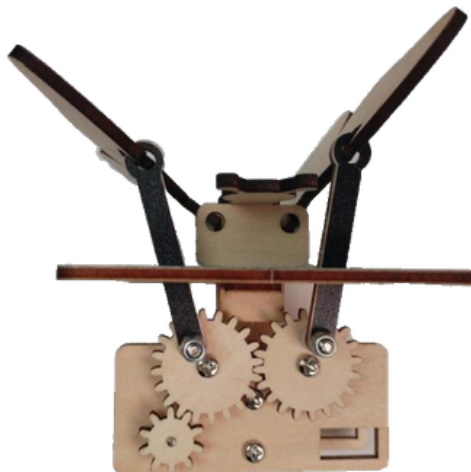
Ve světě biomimikry vědci často pozorují tvory v přírodě, aby se inspirovali pro technologické inovace. Například vážka má při letu vynikající stabilitu a flexibilitu, což inspirovalo design kvadrokoptérových dronů; dlouhý tvar zobáku ledňáčka může snížit odpor při vstupu do vody a používá se v aerodynamickém designu hlavy rychlovlaku. Pevnost pavoučího hedvábí daleko převyšuje pevnost oceli, což vedlo vědce k vývoji vysoce pevných syntetických vláken; chodidla gekonů se mohou volně pohybovat po hladkých stěnách a jejich struktura je napodobována pro výrobu nanoměřítkových biomimetických adhezivních materiálů; lehký a rytmický pohyb křídel motýlů byl použit k vývoji bionických létajících robotů a dekorativního umění s estetickými funkcemi.

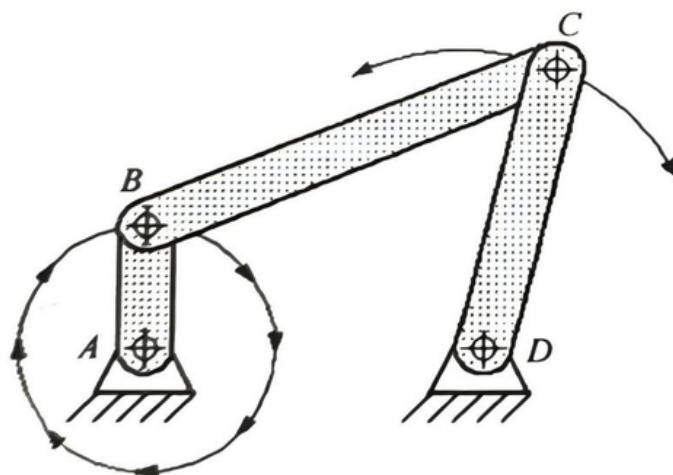
2. Inteligentní bionický motýl

Inteligentní bionický motýl kombinuje převážně strukturu převodového mechanismu s klikovým mechanismem, čímž dosahuje bionického efektu mávání křídel a simuluje letové pohyby skutečných motýlů.

Strukturální princip inteligentního bionického motýla

Inteligentní bionický motýl kombinuje převodovou konstrukci s vačkovým mechanismem, čímž dosahuje bionického efektu mávání křídel a simuluje letové pohyby skutečných motýlů.





V ten inteligentní bionický motýl struktura, když ten motor začíná, ten motorotáčí senavysílat moc naten ostatní dva velký ozubená kola podle výstroj zasnoubení. Když tenspojovací tyč je přinesl níže podle A velký výstroj (klika), ten motýlí křídla klapka dolů. Když ten spojovací tyč je přinesl nahorupodle ten velký výstroj, ten motýlí křídla klapka nahoru.



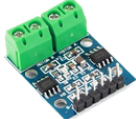
Experiment

1. Popis projektu

Ultrazvukový modul provádí detekci vzdálenosti v reálném čase. Pokud je vzdálenost objektu před motýlem menší než 20 cm, motýl začne mávat křídly, a pokud je detekční vzdálenost větší než 20 cm, křídla přestanou mávat.

2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Ultrazvukový senzor	1

	Ovladač stejnosměrného motoru	1
	Stejnosměrný motor	1

3. Sestavte bionického motýla (pokud jste si nezakoupili stavební materiál, můžete krok montáže přeskočit a přejít přímo k připojení hardwaru a učení programu)

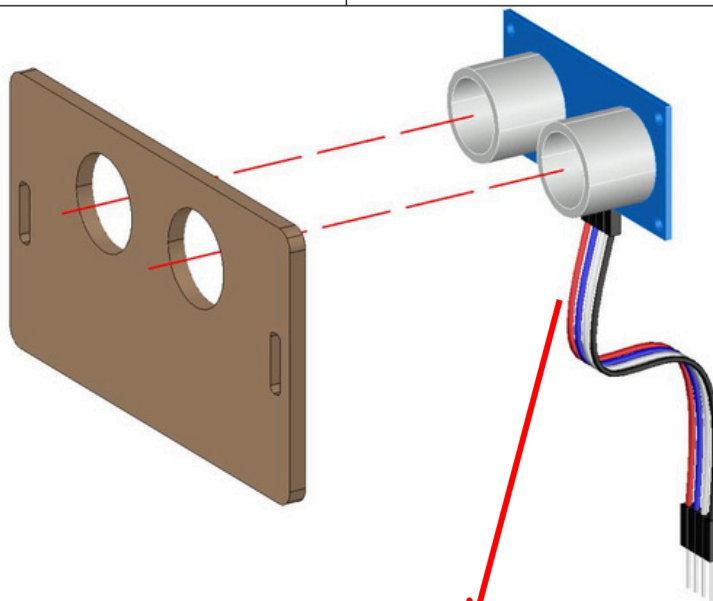
Krok 1 Instalace ultrazvukového modulu

Seznam dílů

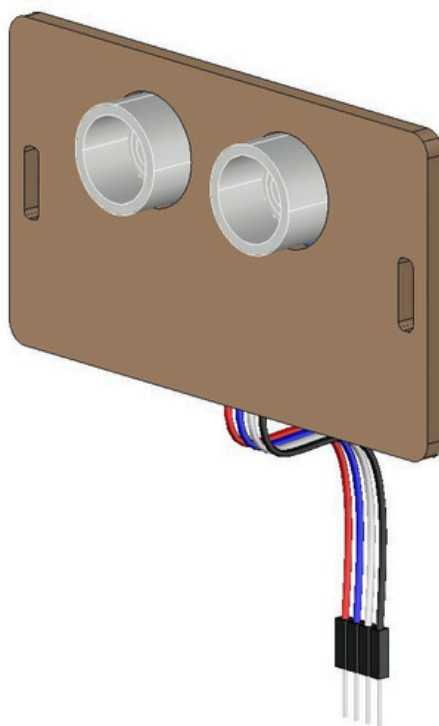
Ultrazvukový modul*1

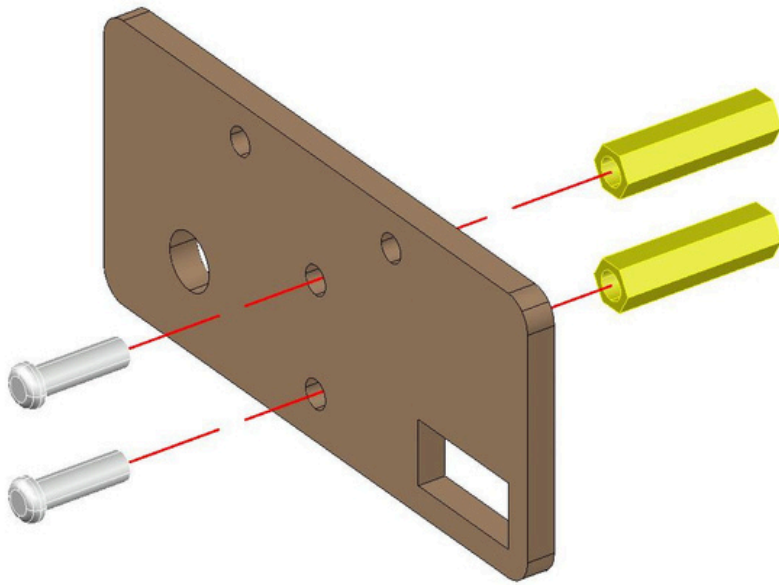
Řada Dupont samec-samice*4

Ilustrace

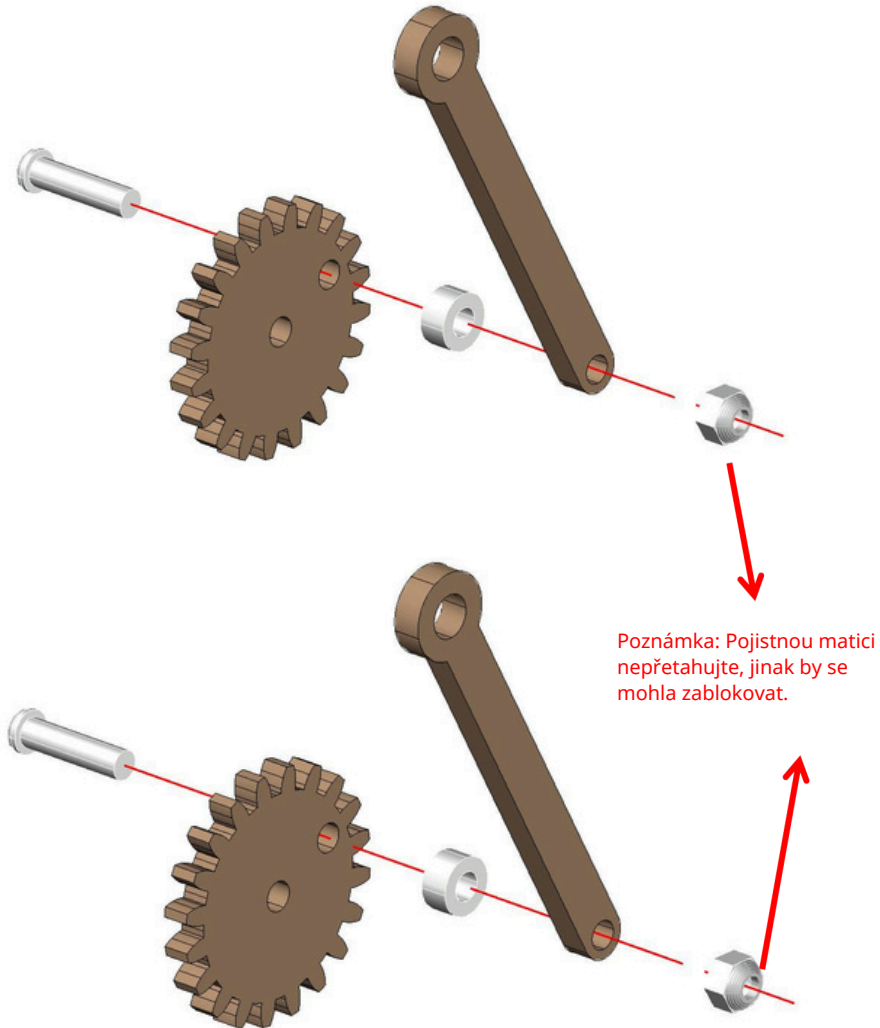


Připojte vodiče DuPont k pinům ultrazvukového modulu: červený vodič k VCC, modrý vodič k Trig, bílý vodič k Echo a černý vodič k GND.

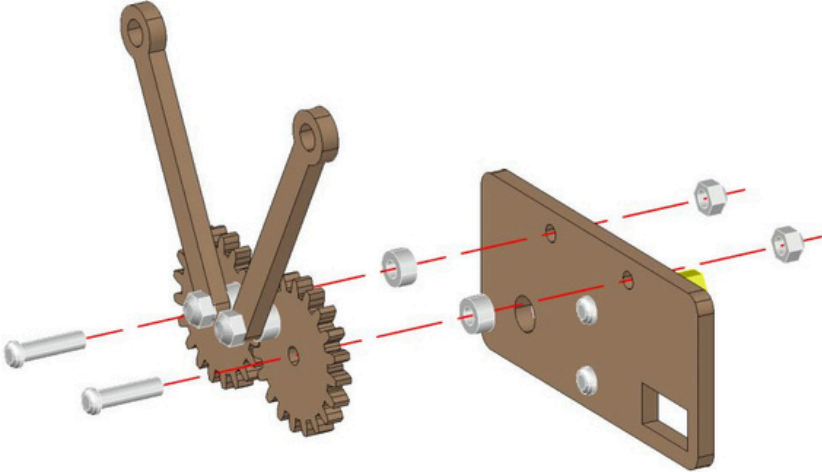
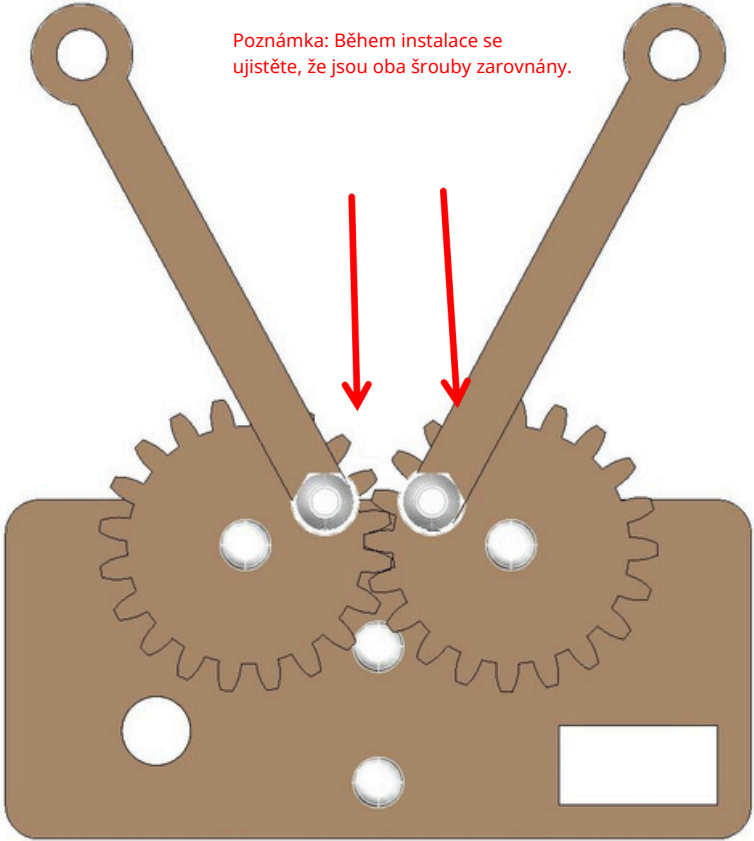


Krok 2 Instalace dvouprůchodového měděného sloupku		
Seznam dílů	M3*10MM kulatá hlava Šrouby*2	M3*12MM Dvouprůchodová měď Pilíř*2
Ilustrace		

Krok 3 Sestavte konstrukci klikové hřídele a ojnice

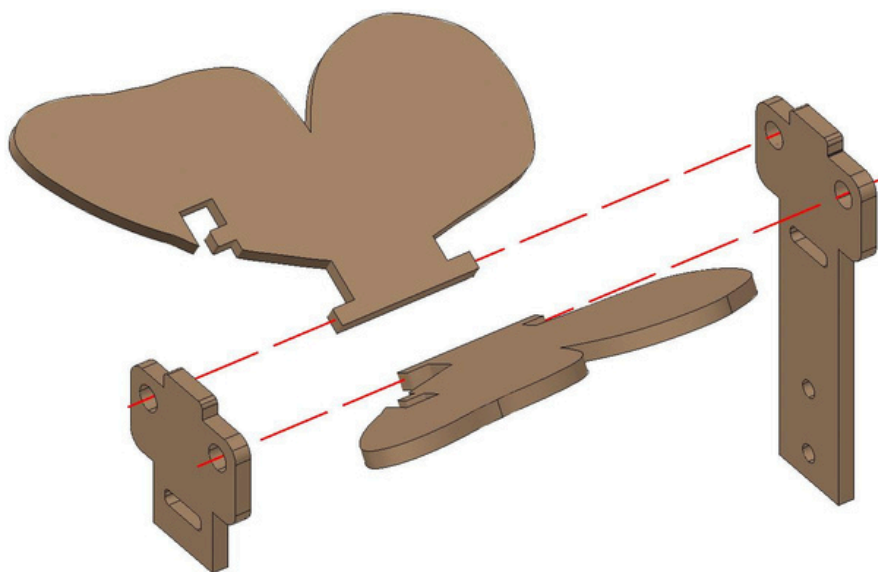
Seznam dílů	M3*14MM Plochá hlava Šrouby*2	M3 Poniklované Pojistné matice*2	3*7*3MM nylon Těsnění*2
Ilustrace	 Poznámka: Pojistnou matici nepřetahujte, jinak by se mohla zablokovat.		

Krok 4 Nainstalujte konstrukci klikové hřídele a ojnice

Seznam dílů	M3*14MM Plochá hlava Šrouby*2	M3 Poniklované Pojistné matice*2	3*7*3MM nylon Těsnění*2
Ilustrace	 Poznámka: Během instalace se ujistěte, že jsou oba šrouby zarovnaný. 		

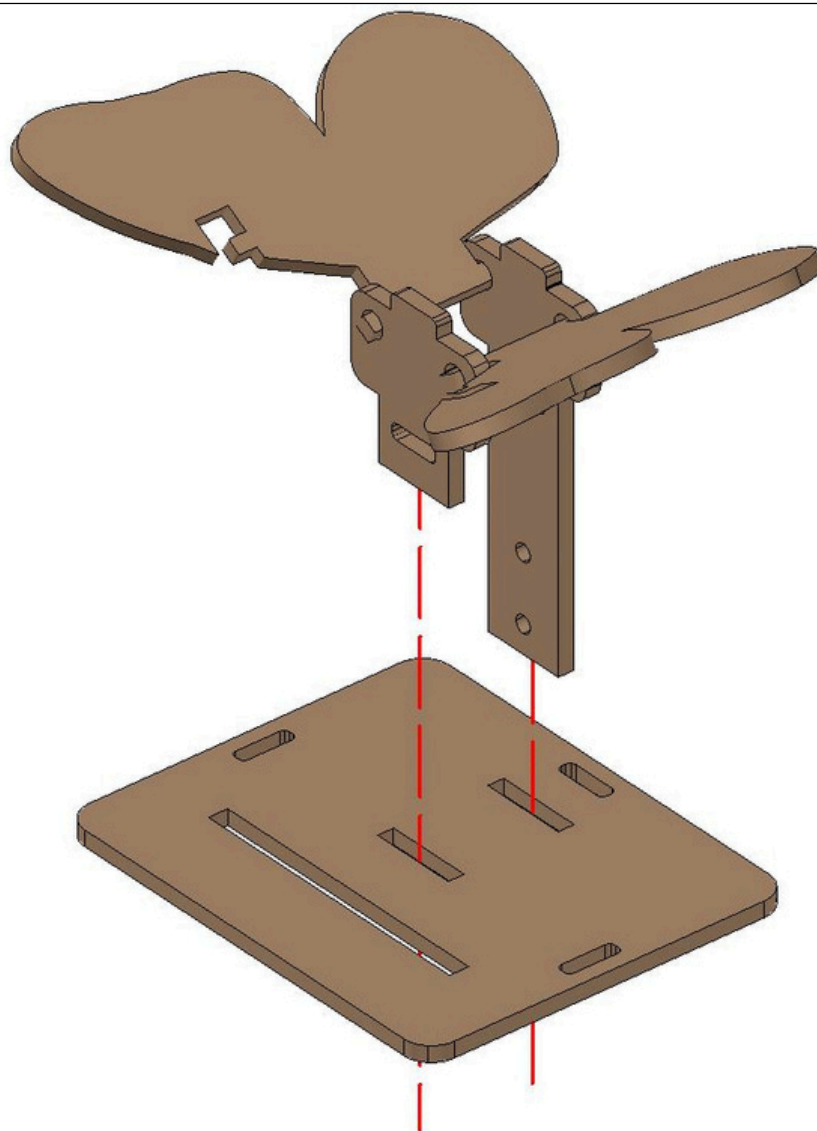
Krok 5. Instalace motýlích křídel

Ilustrace



Krok 6. Zajistěte rám těla motýla

Ilustrace

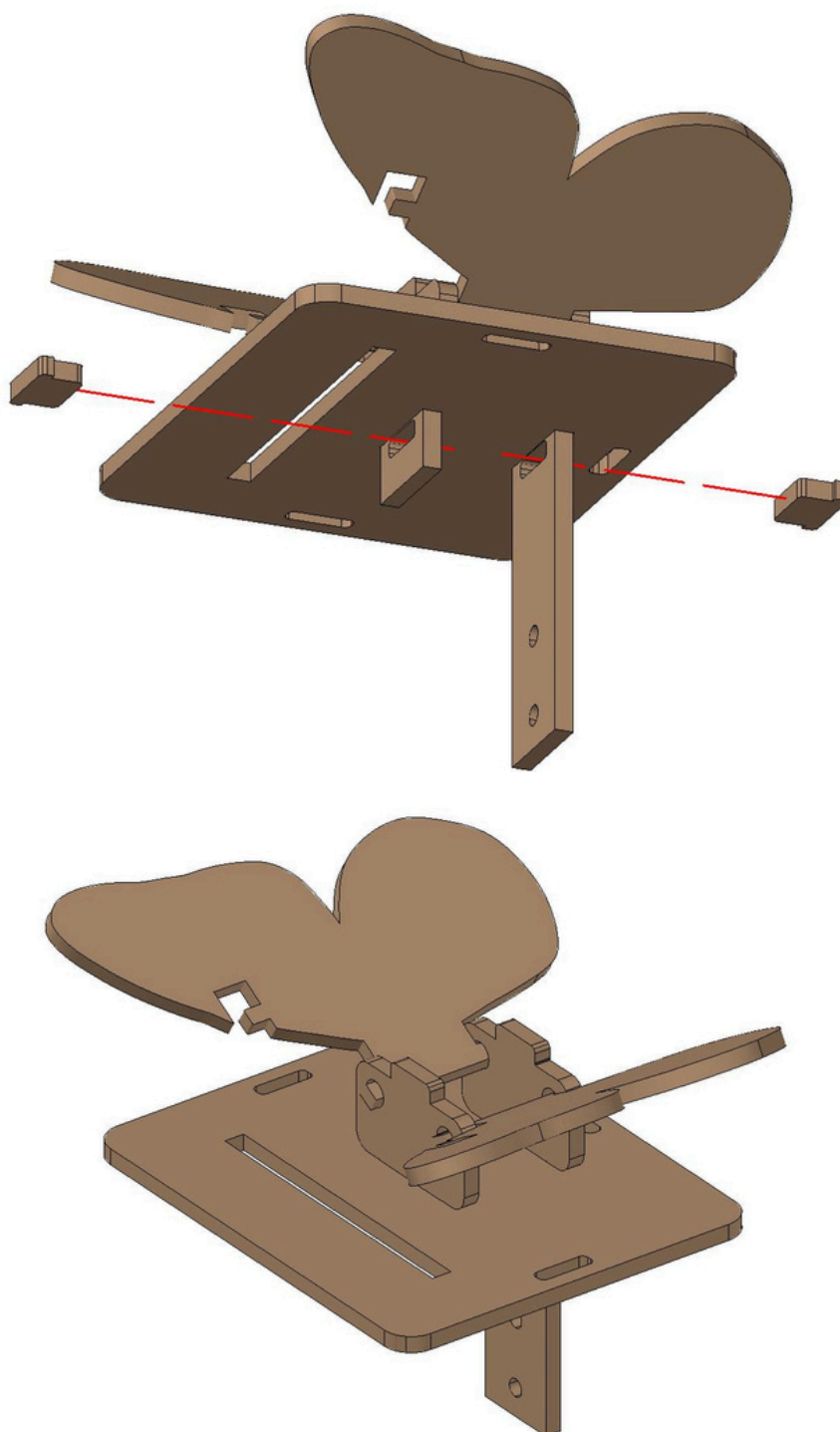


Krok 7. Zajistěte rám těla motýla

Seznam dílů

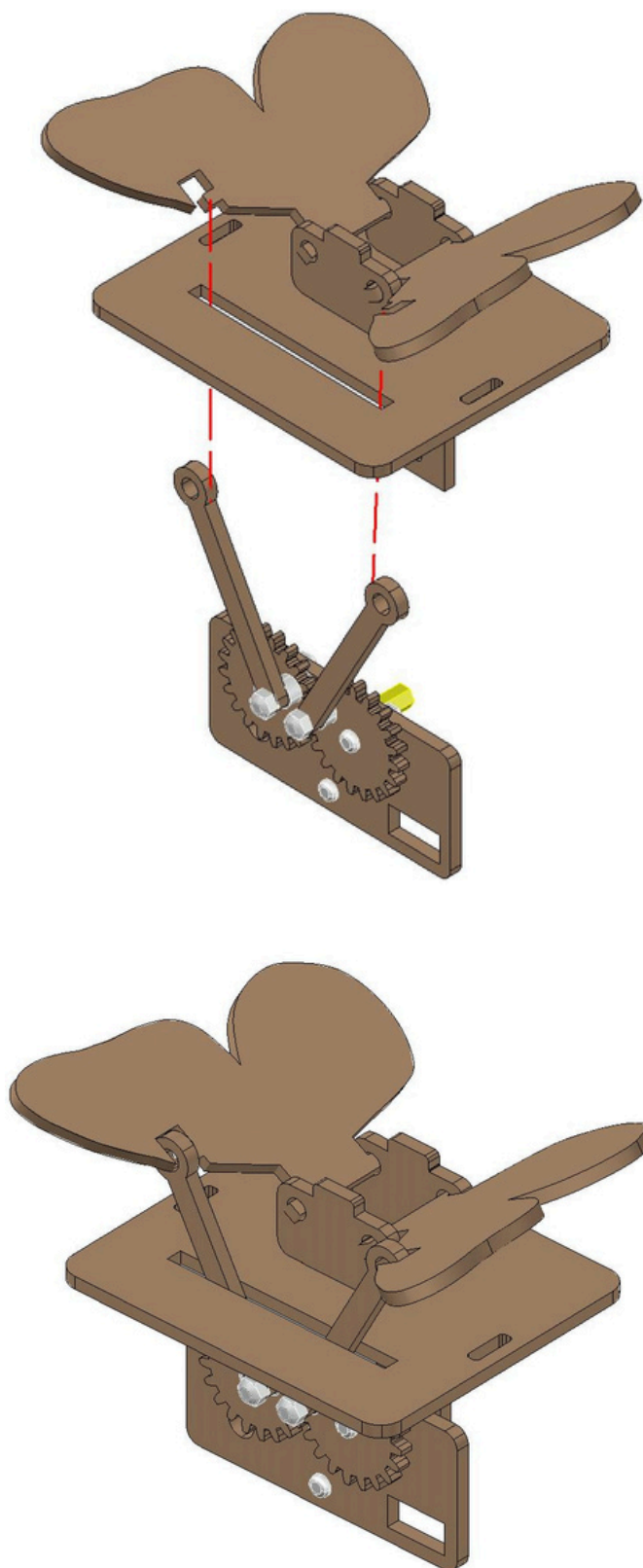
Západka*2

Ilustrace



Krok 8. Zajistěte motýlí křídla

Ilustrace

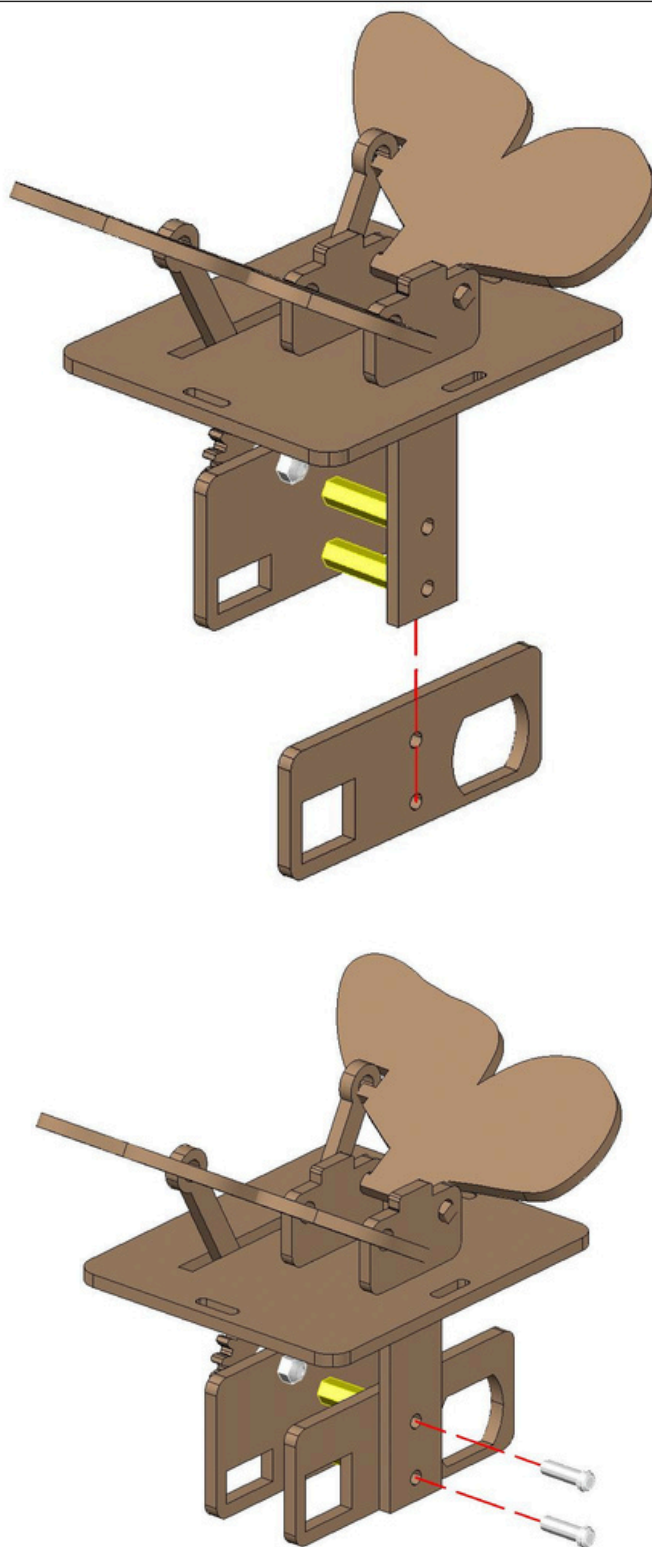


Krok 9 Nainstalujte upevňovací desku motoru

Seznam dílů

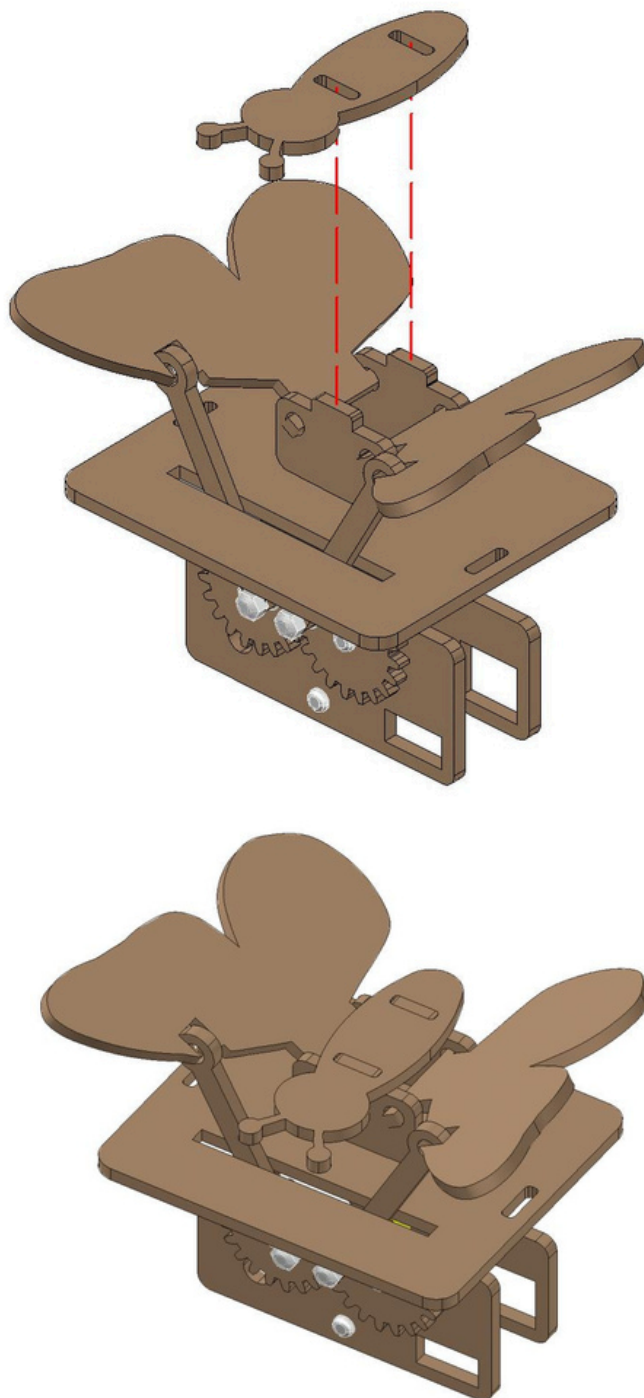
Šrouby s kulatou hlavou M3*10MM*2

Ilustrace



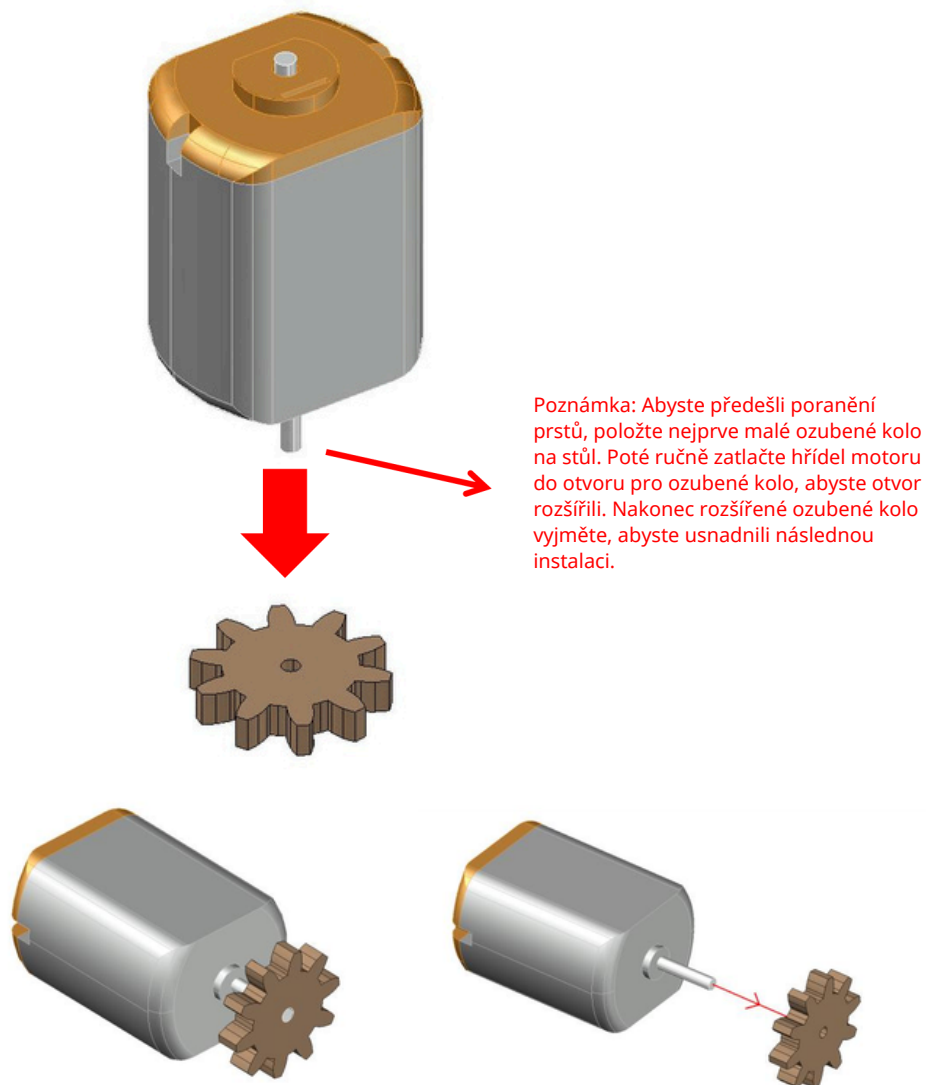
Krok 10. Instalace těla motýla

Ilustrace



Krok 11. Rozšíření otvoru pastorku

Ilustrace

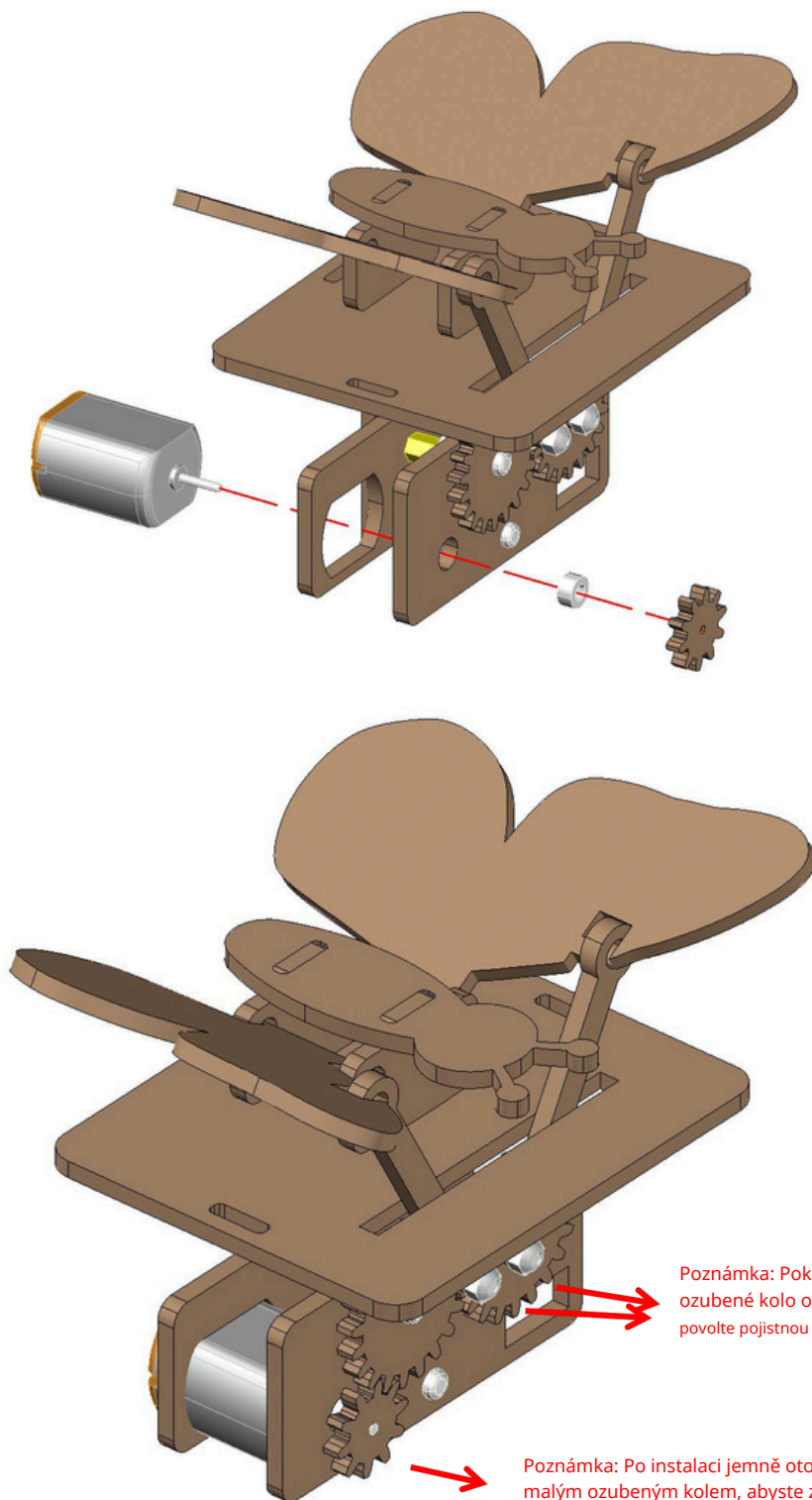


Krok 12 Instalace motoru

Seznam dílů

Motor*1

Ilustrace

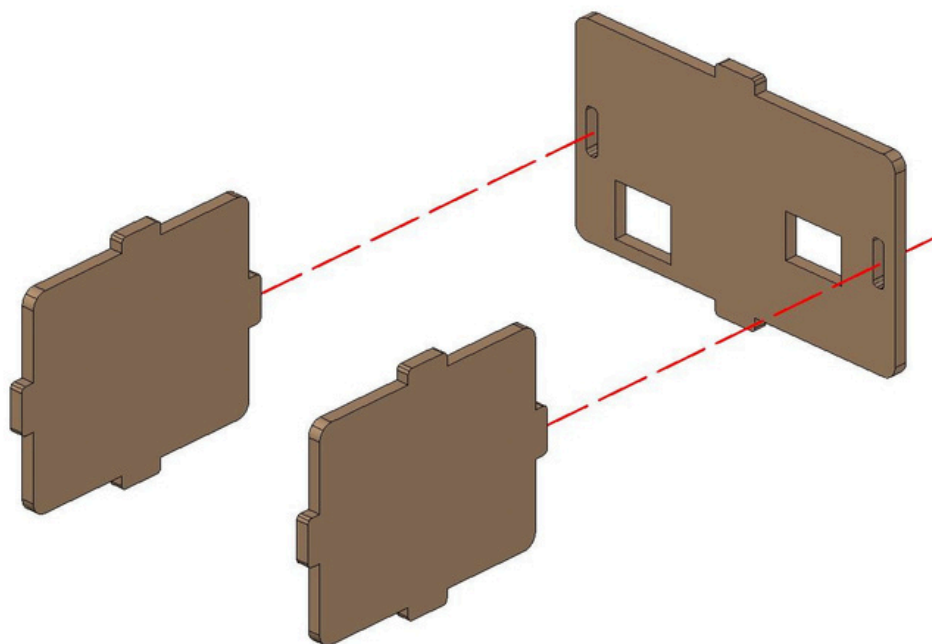


Poznámka: Pokud se ozubené kolo obtížně otáčí, povolte pojistnou matici.

Poznámka: Po instalaci jemně otočte malým ozubeným kolem, abyste zajistili, že může plynule pohánět další dvě velká ozubená kola a synchronně se otáčet.

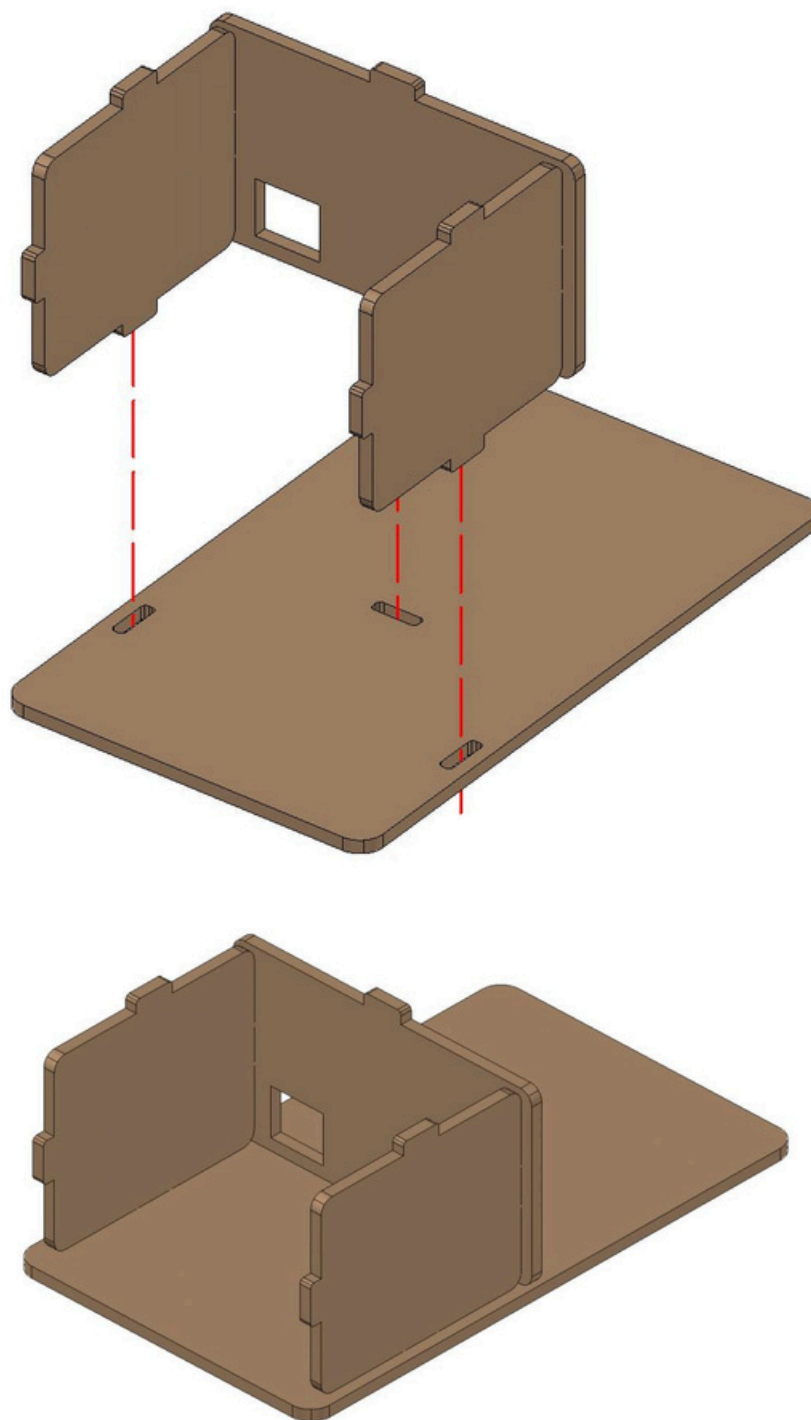
Krok 13 Instalace bočních panelů

Ilustrace



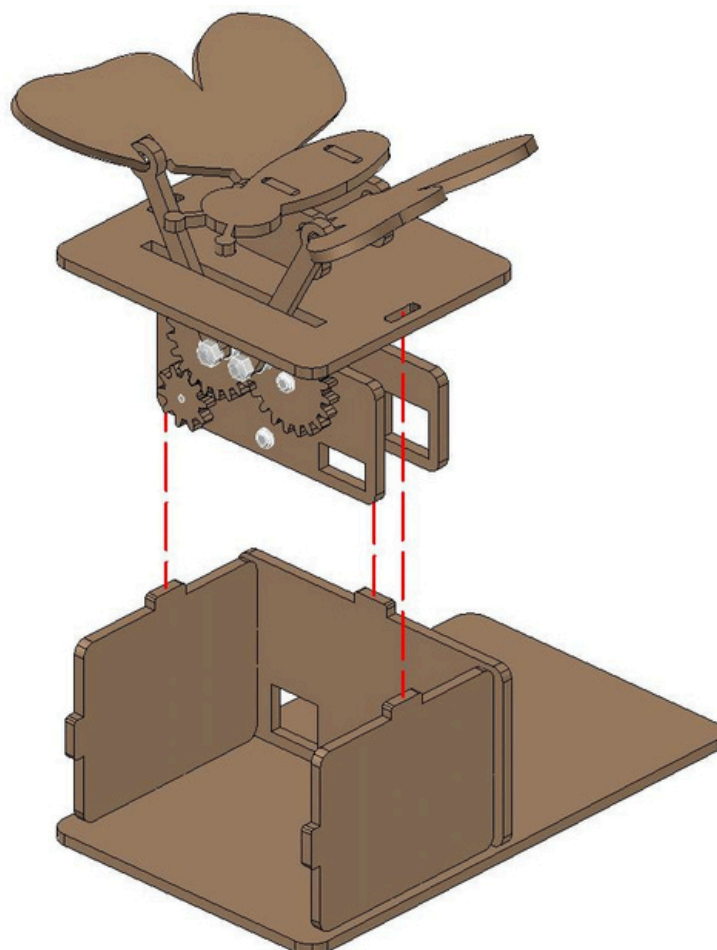
Krok 14 Instalace spodního panelu

Ilustrace



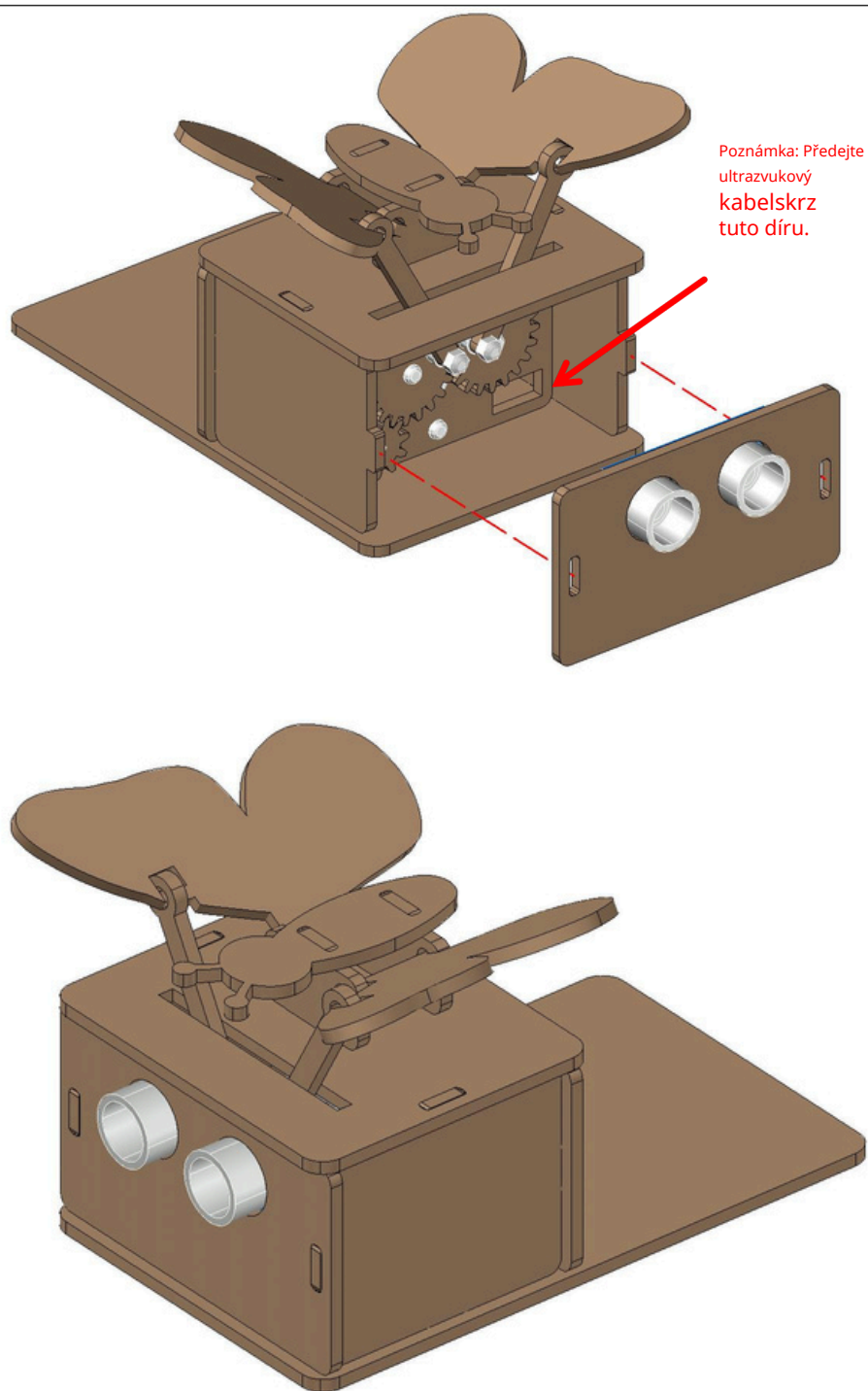
Krok 15. Zajištění těla motýla

Ilustrace

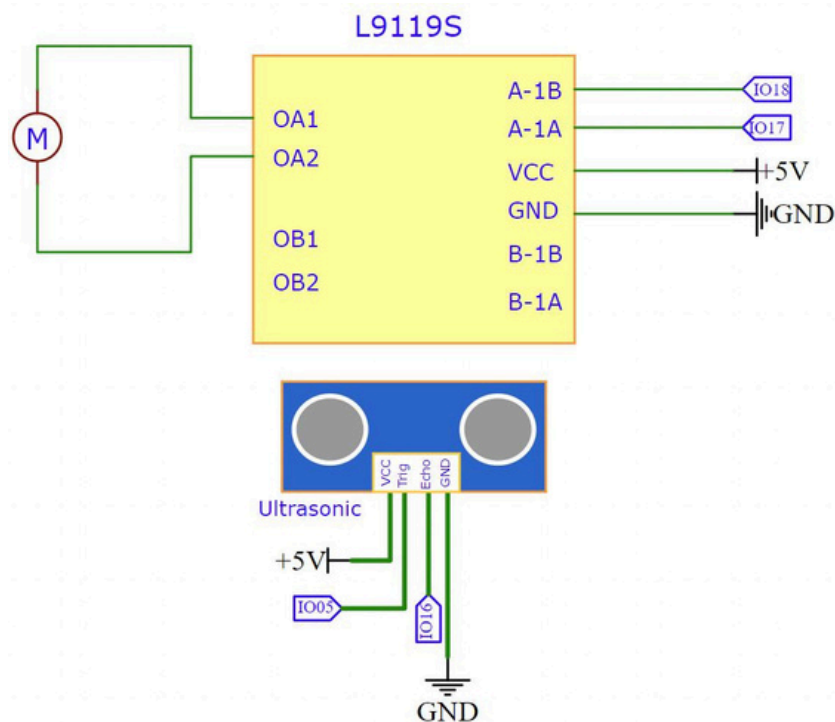


Krok 16 Instalace předního panelu

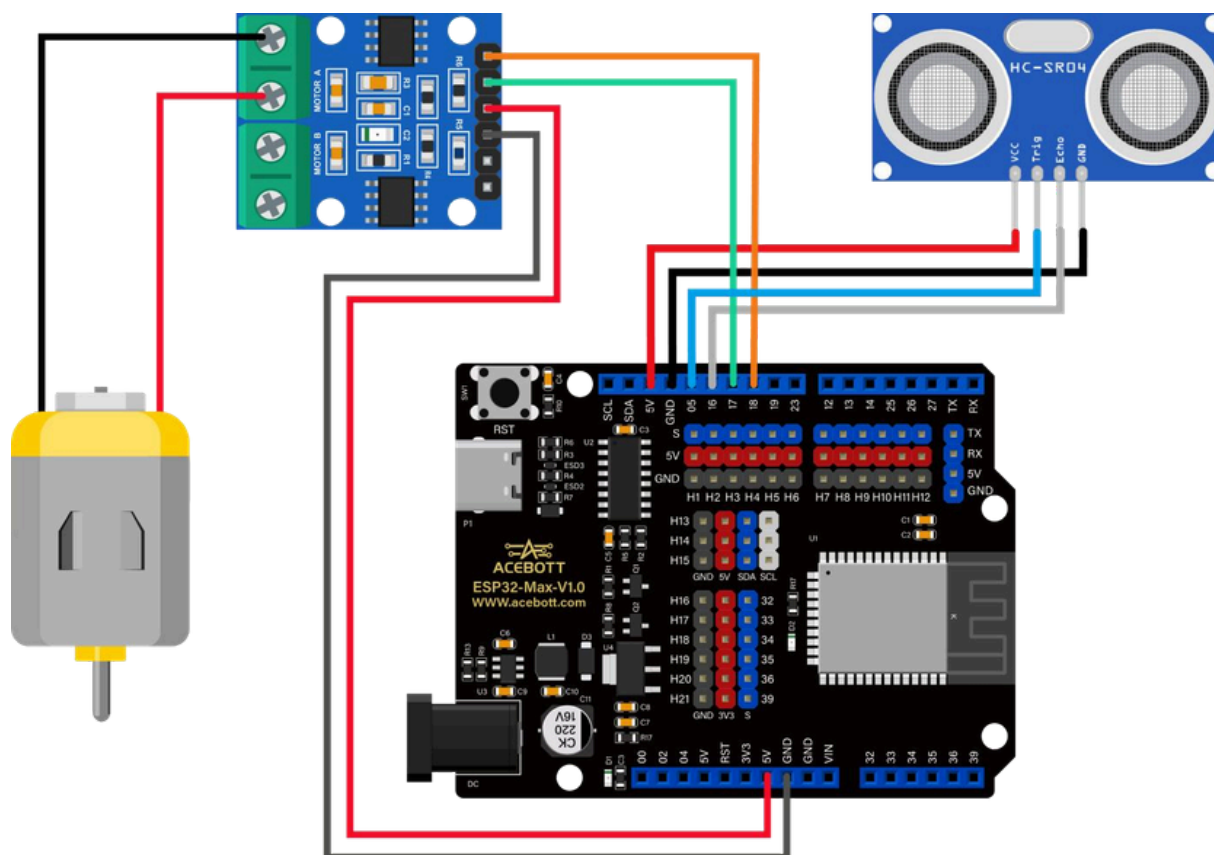
Ilustrace



4. Schéma zapojení



5. Připojení hardwaru



6. Referenční postupy Otevřete "[Smart_Bionic_Butterfly.ino](#)"

„\Arduino(Zkušný student)\2.Code\lesson 34 Smasorutbo rBionivc
Butterfly\Smart_Bionic_Butterfly“ nebo zkopírujte následující kód do
Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a
příslušného portu podle dříve naučených kroků klikněte na



nahrát program.

```
const int trigPin = 5; // Trigger pin of ultrasonic sensor
const int echoPin = 16; // Echo pin of ultrasonic sensor
const int motor1A = 17; // Motor direction control pin
const int motor1B = 18; // PWM speed control pin
const float thresholdDistance = 20.0; // Distance threshold in centimeters

void setup() {
  Serial.begin(115200); // Initialize serial communication
  pinMode(trigPin, OUTPUT); // Set trigger pin as output
  pinMode(echoPin, INPUT); // Set echo pin as input
  pinMode(motor1A, OUTPUT); // Set motor direction pin as output
  pinMode(motor1B, OUTPUT); // Set motor speed pin as output
}

void loop() {
  float distance = getDistance(); // Measure distance
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  // If object is detected within the threshold distance, run motor
  if (distance != -1 && distance < thresholdDistance) {
    analogWrite(motor1B, 255); // Set direction
    analogWrite(motor1A, 0); // Set PWM speed
  } else if (distance != -1 && distance >= thresholdDistance) {
    analogWrite(motor1B, 0); // Set direction
    analogWrite(motor1A, 0); // Set PWM speed
  }
  delay(100); // Short delay for better response
}

// Measure distance using ultrasonic sensor
float getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
```

```
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);  
  
unsigned long pulseTime = pulseIn(echoPin, HIGH, 5000); // Timeout after  
5000  $\mu$ s  
  
if (pulseTime > 0) {  
    return pulseTime * 0.0343 / 2; // Convert pulse duration to distance (cm)  
} else {  
    return -1; // Return -1 if timeout (no object detected)  
}  
}
```

Po úspěšném nahrání programu začněte mávat křídly, když je někdo detekován do vzdálenosti 20 cm přímo před vámi.

Lekce 35 Chytrý odpadkový koš



Cíle kurzu

- Naučte se, jak fungují chytré odpadkové koše a jaké jsou scénáře použití.
- Naučte se sestavit model chytrého odpadkového koše.
 - Napište program, který implementuje funkci automatického otevírání víka inteligentních odpadkových košů.



Úvod do kurzu

V každodenním životě se hygiena odpadkových košů často opomíjí. Chytrý odpadkový koš pomocí senzorů snímá blízkost lidských rukou nebo předmětů, automaticky otevírá víko, zabraňuje přímému kontaktu a účinně zlepšuje hygienu a snadnost používání. S rozvojem chytrých domácností se navrhuje stále více chytrých zařízení a chytré odpadkové koše jsou jedním z nich.



V této lekci se naučíme, jak si postavit praktický chytrý odpadkový koš.



Body znalostí

1. Jak funguje chytrý odpadkový koš?

Chytré odpadkové koše obvykle používají infračervené nebo ultrazvukové senzory k detekci přiblížení osoby nebo předmětu. Pokud je detekovaná vzdálenost menší než nastavená prahová hodnota, servopohon ovládá víko tak, aby se otevřelo a automaticky zavřelo, když se předmět vzdálí. Tím se dosáhne „bezkontaktního“ provozu, což zvyšuje pohodlí a hygienu.



Experiment

1. Popis projektu

Tento program využívá ultrazvukový senzor k detekci blízkosti člověka. Když je někdo detekován v dosahu 25 centimetrů, servomotor automaticky otevře víko koše. Když se objekt vzdálí, servomotor zpozdí zavření víka, čímž vzniká bezkontaktní inteligentní systém koše.

2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Ultrazvukový senzor	1
	Servo	1

3. Sestavte chytrý odpadkový koš (pokud jste si nezakoupili stavební materiál, můžete krok montáže přeskočit a přejít přímo k připojení hardwaru a učení programu)

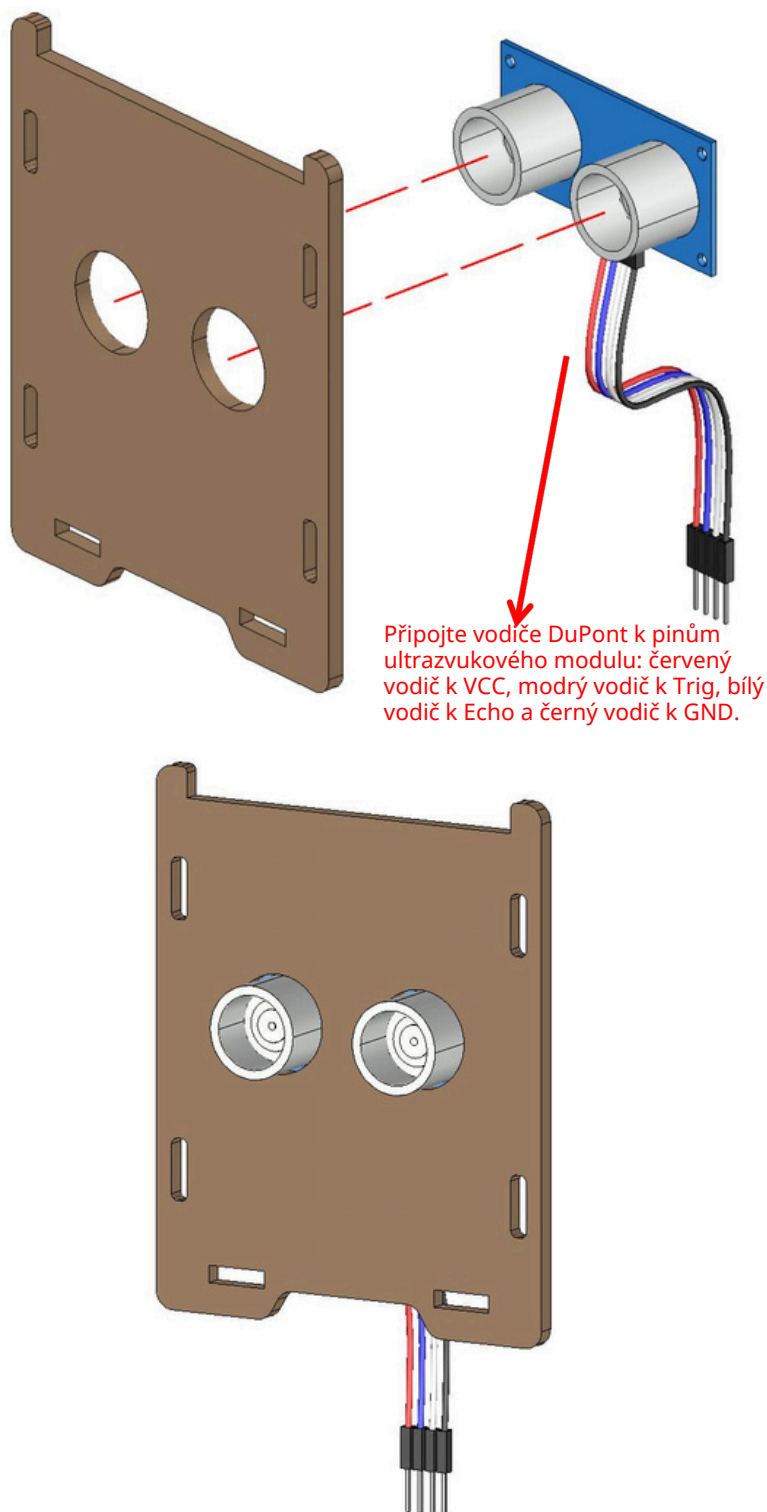
Krok 1 Instalace ultrazvukového modulu

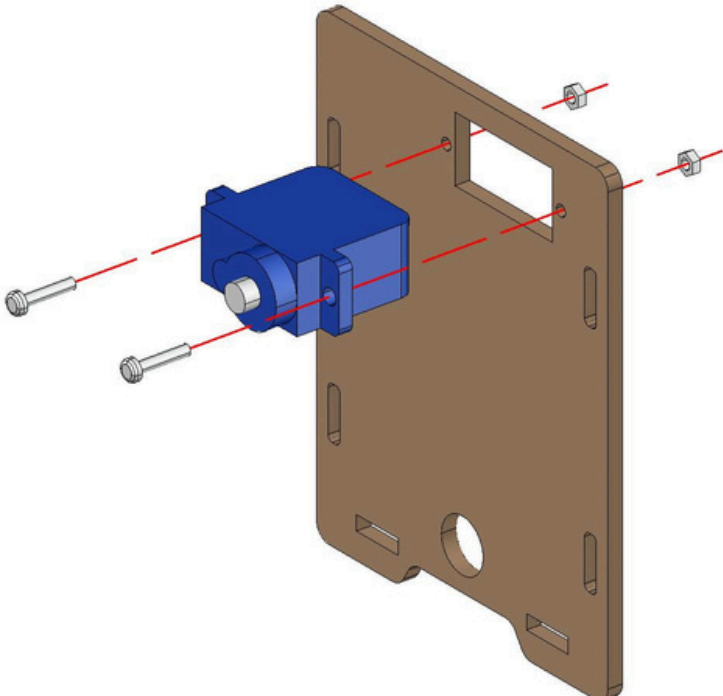
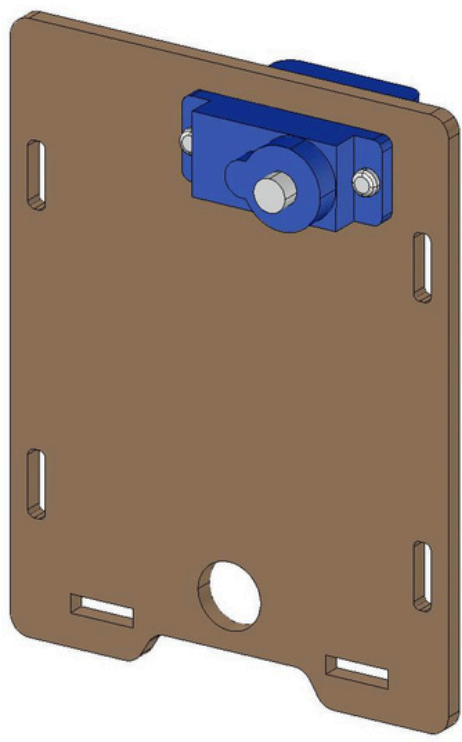
Seznam dílů

Ultrazvukový modul*1

Řada Dupont samec-samice*4

Ilustrace



Krok 2 Instalace serva			
Seznam dílů	Servo *1	M2*14MM kulatá hlava Šroub*2	Matice M2*2
Ilustrace			
			

Krok 3. Nainstalujte volant a spodní desku

Seznam dílů

Servo klakson*1

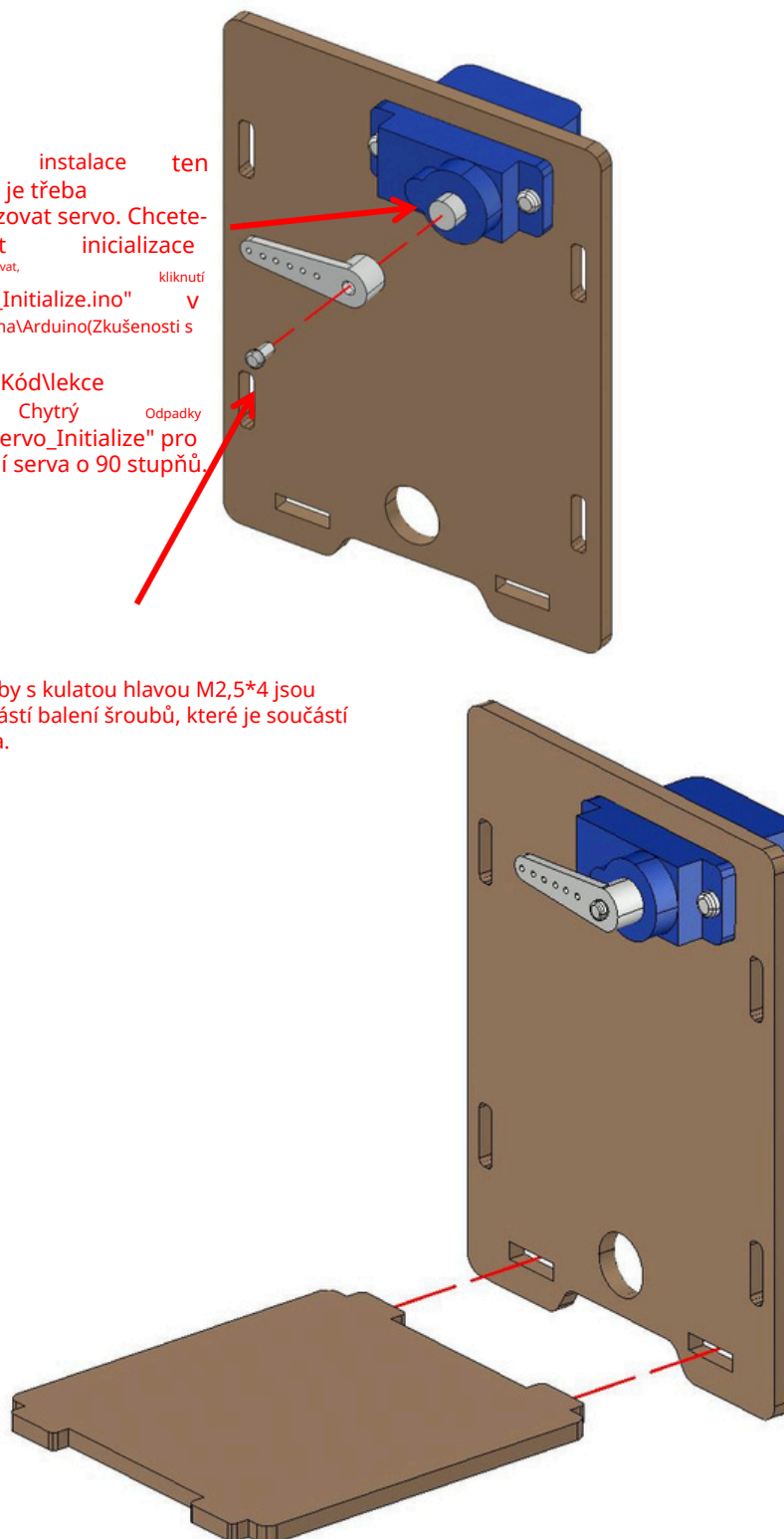
Šrouby s kulatou hlavou M2,5*4*1

Před instalací tohoto
volant, je třeba
inicializovat servo. Chcete-
li získat inicializace
naprogramovat,
kliknutí
"Servo_Initialize.ino" v
"Angličtina\Arduino(Zkušenosti s

Žák)\2.Kód\lekce
34 Chytrý Odpadky
"Can\Servo_Initialize" pro
otočení serva o 90 stupňů.

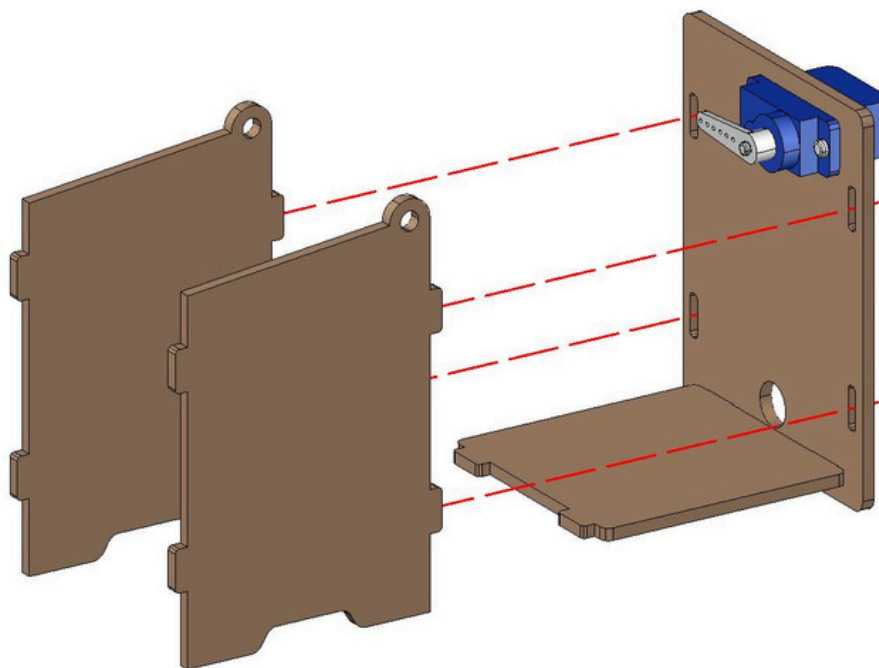
Šrouby s kulatou hlavou M2,5*4 jsou
součástí balení šroubů, které je součástí
serva.

ilustrace



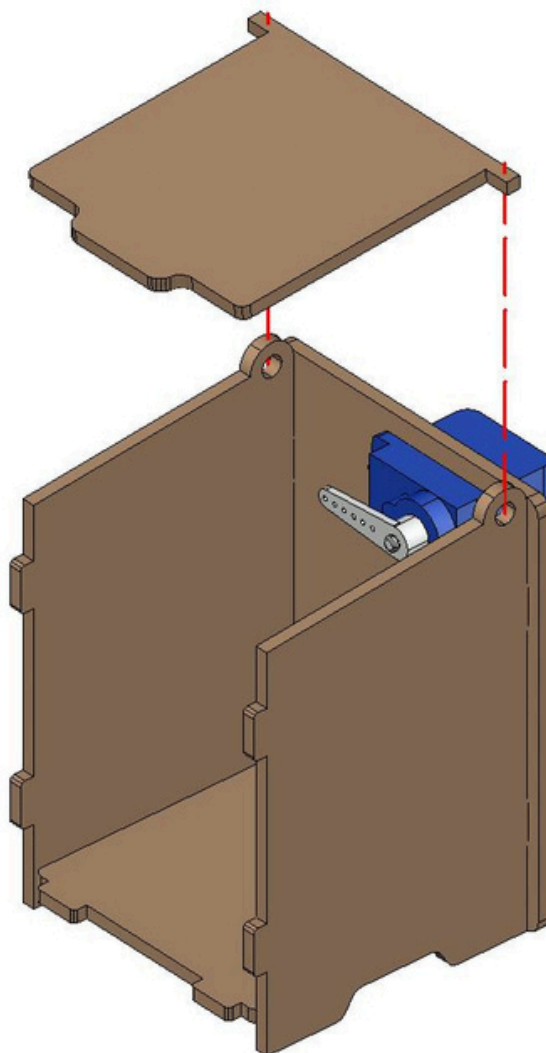
Krok 4. Instalace bočních panelů odpadkového koše

Ilustrace



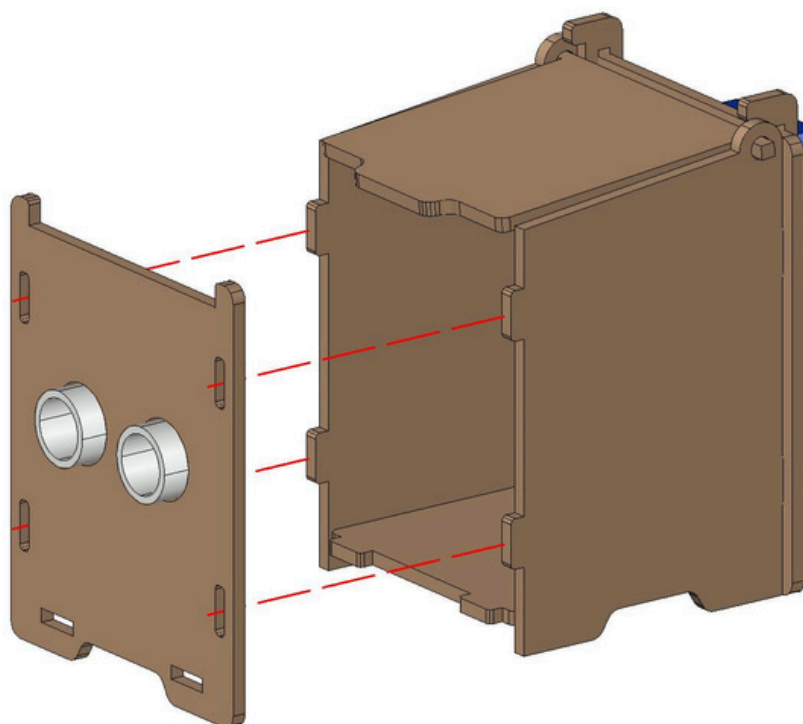
Krok 5. Instalace krytu odpadkového koše

Ilustrace



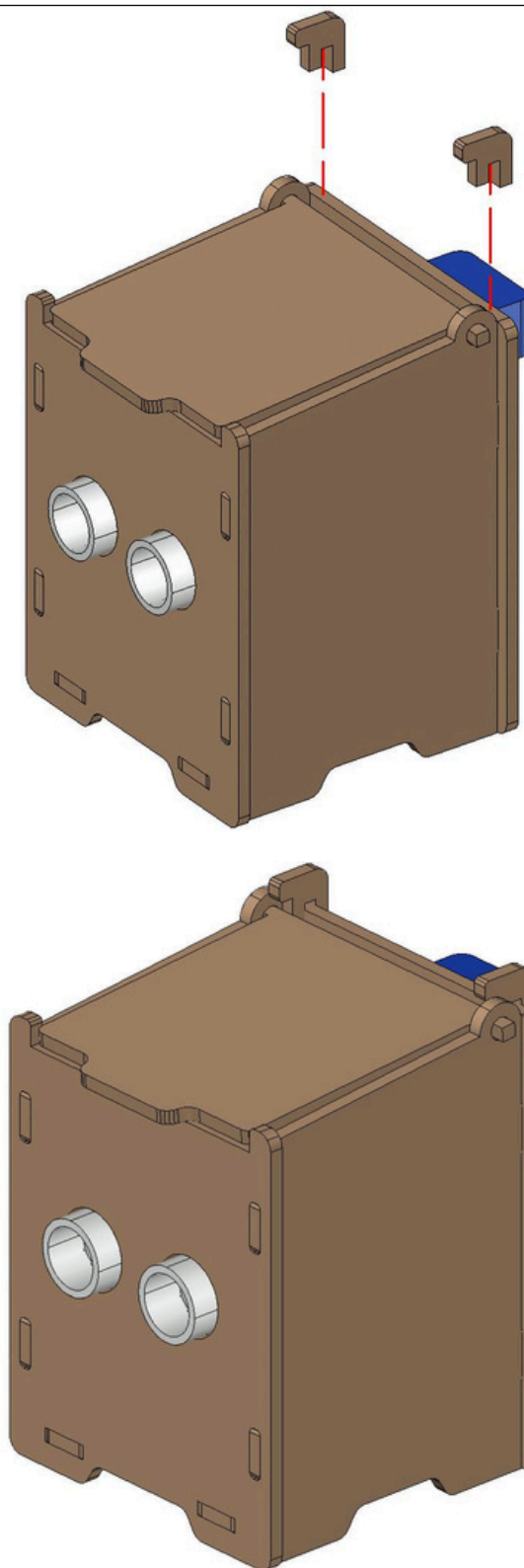
Krok 6. Instalace předního panelu odpadkového koše

Ilustrace

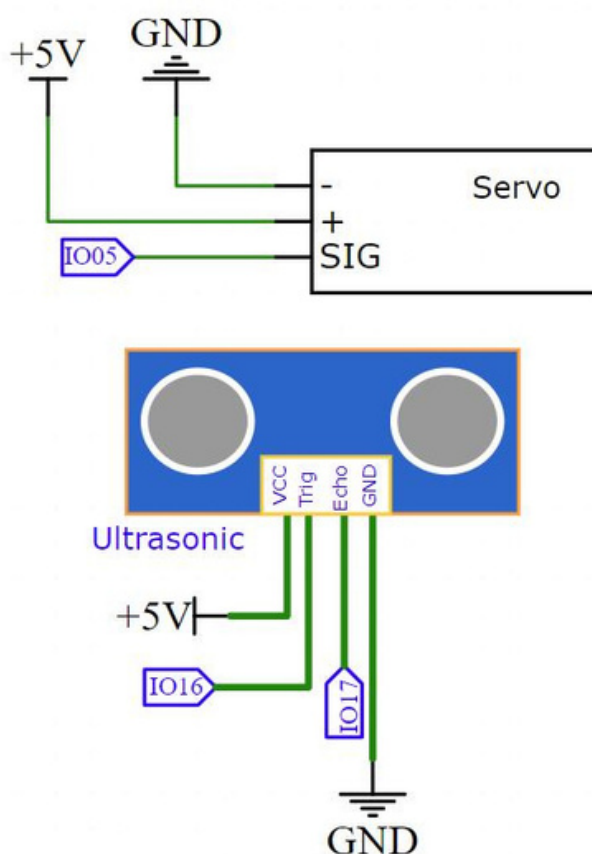


Krok 7. Instalace omezovače krytu odpadkového koše

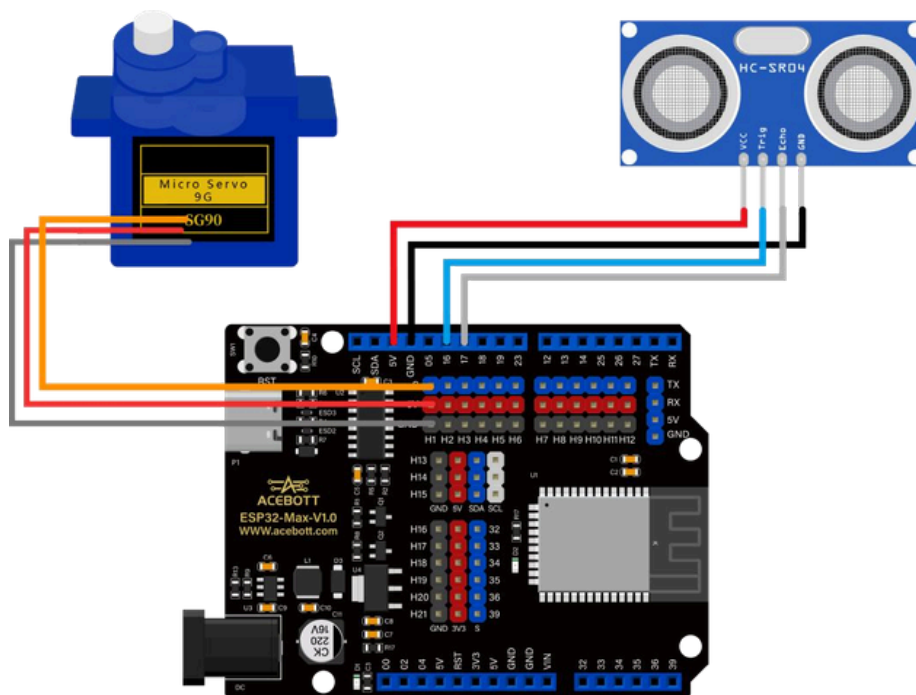
Ilustrace



4. Schéma zapojení



5. Připojení hardwaru



6. Referenční postupy

Otevřete "[Smart_Trash_Can.ino](#)" soubor v "English\Arduino(Experienced

Learner)\2.Code\lesson 35 Chytrý odpadkový koš\Smart_Trash_Can" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů

dříve naučené kroky, klikněte  nahrát program.

```
#include <ESP32Servo.h>

#define TRIG_PIN 16          // Ultrasonic sensor trigger pin
#define ECHO_PIN 17          // Ultrasonic sensor echo pin
#define SERVO_PIN 5          // Servo control pin

int angle = 90;              // Initial servo angle (closed position)
bool isOpened = false;      // Flag to indicate if the lid is currently open

Servo myServo;

// Function to read distance from ultrasonic sensor (in centimeters)
float readDistance() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    long duration = pulseIn(ECHO_PIN, HIGH); // Timeout after 30ms
    long distance = duration * 0.034 / 2;     // Convert time to distance
    delay(100);                             // Small delay to
    stabilize reading
    return distance;
}

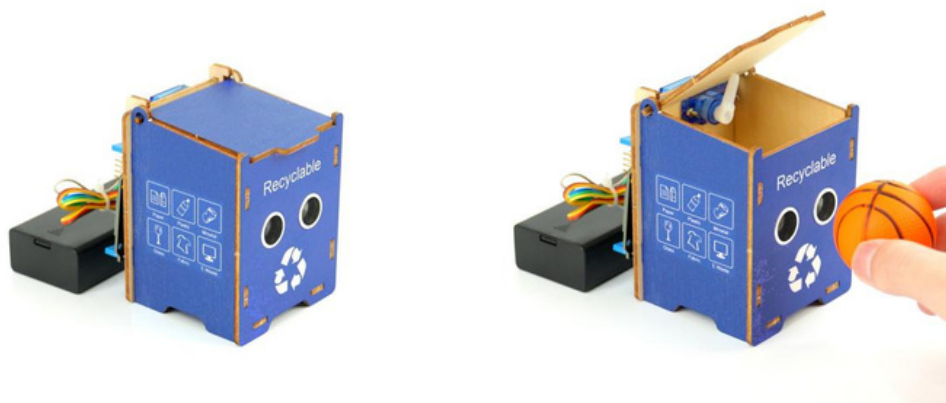
void setup() {
    Serial.begin(115200);          // Initialize serial monitor
    pinMode(TRIG_PIN, OUTPUT);     // Set trigger pin as output
    pinMode(ECHO_PIN, INPUT);      // Set echo pin as input
    myServo.setPeriodHertz(50);    // Standard servo frequency: 50Hz
    myServo.attach(SERVO_PIN, 500, 2500); // Set servo pulse width range
    (500-2500µs)
    myServo.write(angle);          // Move servo to initial (closed)
    position
}

void loop() {
    float distance = readDistance(); // Get distance measurement
    Serial.print("Distance: ");
    Serial.print(distance);
}
```

```
Serial.println(" cm");

// If an object is detected within 25 cm and lid is closed, open it
if (distance > 0 && distance < 25 && isOpened == false) {
  for (angle = 90; angle < 170; angle++) {
    myServo.write(angle);    // Gradually open the lid
    delay(5);                // Small delay for smooth motion
  }
  isOpened = true;          // Update state flag
}
// If object is far and lid is open, close it
else if (distance >= 25 && isOpened == true) {
  delay(500);                // Wait 0.5 seconds before closing
  for (angle = 170; angle > 90; angle--) {
    myServo.write(angle);    // Gradually close the lid
    delay(5);                // Small delay for smooth motion
  }
  isOpened = false;         // Update state flag
}
delay(200);                  // Short delay before next reading
}
```

Po úspěšném nahrání programu se chytrý odpadkový koš otevře, když detekuje přicházející osobu, a zavře se, když nikdo detekován není.



Lekce 36 Chytrá brána



Cíle kurzu

- Zjistěte, jak fungují chytré turnikety
- Sestavte a postavte inteligentní model turniketu
- Napište program pro implementaci funkce inteligentního turniketu



Úvod do kurzu

Už jste někdy viděli u vjezdu do nákupního centra, kancelářské budovy nebo veřejného parkoviště chytrou bránu, která automaticky zvedá zábradlí, když se blíží auto? Toto zařízení nevyžaduje ruční ovládání a dokáže automaticky detekovat a otevřít zábradlí podle blízkosti vozidla, což výrazně zlepšuje efektivitu dopravy a úroveň inteligence. Jak tedy „ví“, že se blíží auto? Jak ovládat zvedání nebo spouštění zábradlí?



V této lekci si postavíme model inteligentní automobilové brány. Použijeme ultrazvukové senzory pro snímání vozidel a servopohon pro řízení pohybu.

bariéry, zažijeme praktické využití inteligentních systémů v reálném provozu.



Body znalostí

1. Inteligentní brány

Inteligentní brána je inteligentní zařízení používané pro správu přístupu vozidel a automatické ovládání. Běžně se používá v místech, jako jsou obytné komplexy, parkoviště, kancelářské brány a vjezdy na dálnice. Integruje snímání, řídicí logiku a akční členy pro identifikaci, určení a uvolnění nebo zachycení vjíždějících a vyjíždějících vozidel.

Jak fungují chytré brány?

Základním principem inteligentní automobilové brány je automatizovaný proces „snímání + řízení + provedení“. Nejprve ultrazvukový senzor instalovaný v přední části brány nepřetržitě vysílá zvukové vlny a měří dobu návratu, aby určil, zda se blíží vozidlo. Jakmile je naměřená vzdálenost menší než nastavená hodnota (například 20 cm), hlavní řídicí čip (například ESP32) okamžitě reaguje a odesílá řídicí signál do serva. Po přijetí signálu se servo začne otáčet, čímž se závora zvedne a dokončí se akce „otevření brány“. Když vozidlo projede a opustí snímanou oblast, systém znovu potvrdí, že v okolí nejsou žádné překážky, a poté dá servu pokyn k pomalému spuštění závory a jejímu návratu do původního stavu. Celý proces nevyžaduje žádnou ruční obsluhu a spoléhá se na data ze senzorů a programové logické řízení pro dosažení efektivního a bezpečného automatického řízení dopravy.

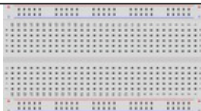


Experiment

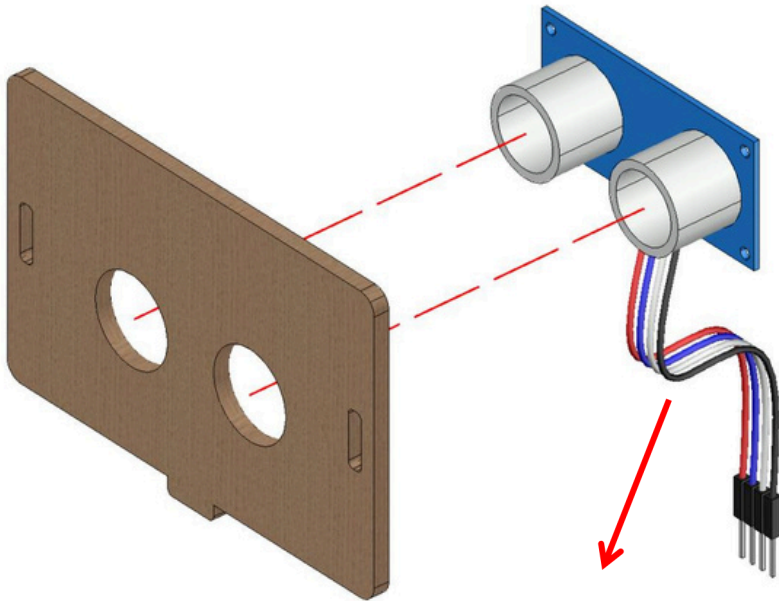
1. Popis projektu

Tento program implementuje inteligentní systém brány. Když se vozidlo nebo objekt přiblíží k bráně (s detekční vzdáleností menší než 14 cm), systém pomocí serva automaticky zvedne závoru brány a rozsvítí kontrolku, která signalizuje průjezd. Když vozidlo odjede (s detekční vzdáleností větší než 20 cm), servo pomalu spustí závoru brány a zhasne kontrolku, čímž dokončí inteligentní operaci otevírání a zavírání.

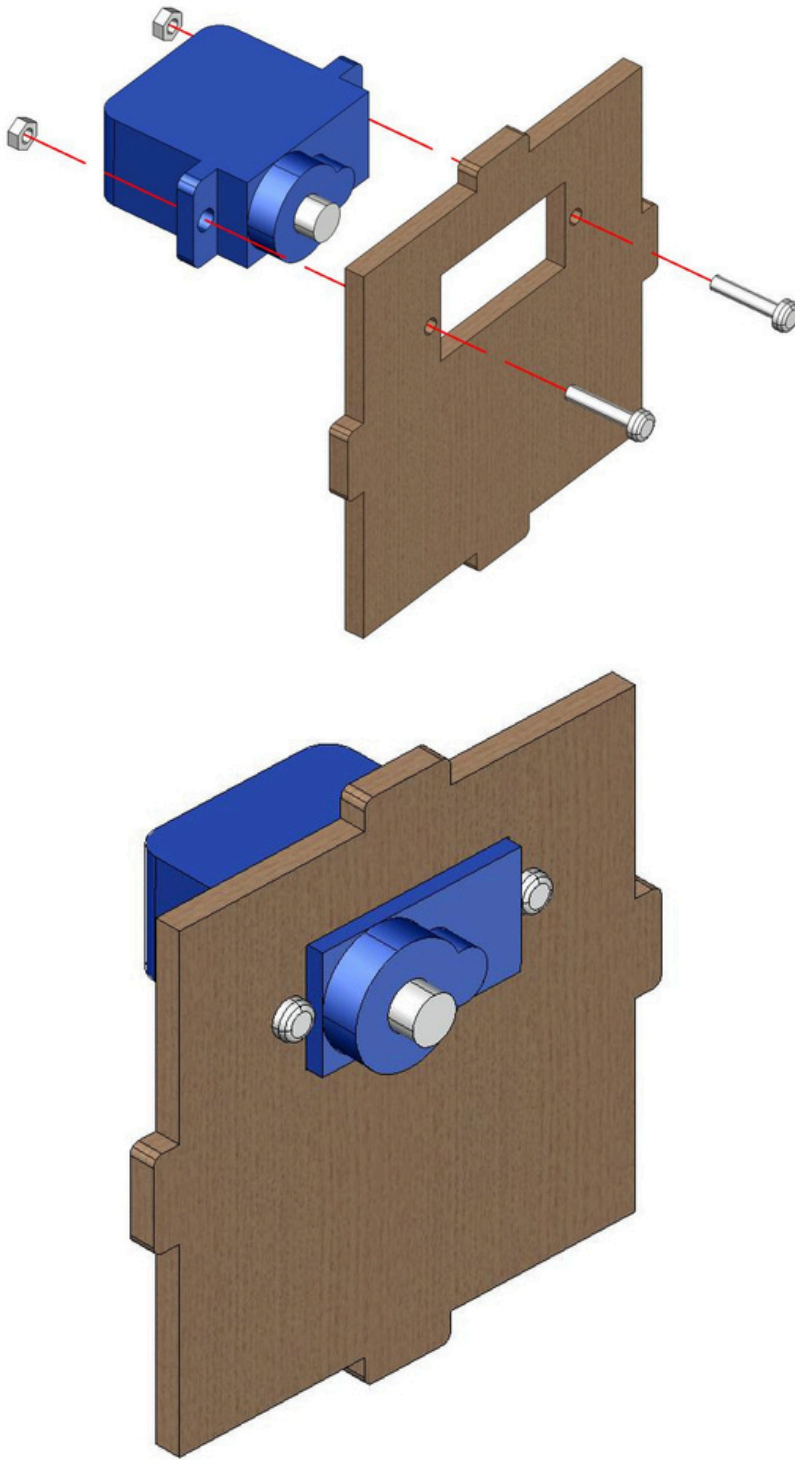
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Ultrazvukový senzor	1
	Servo	1
	Nepájivé pole s 400 spojovacími body	1
	LED	1
	Rezistor 220Ω	1

3. Sestavte model inteligentní brány (Pokud jste si nezakoupili stavební materiál, přeskočte kroky montáže a pokračujte přímo k připojení hardwaru a programování)

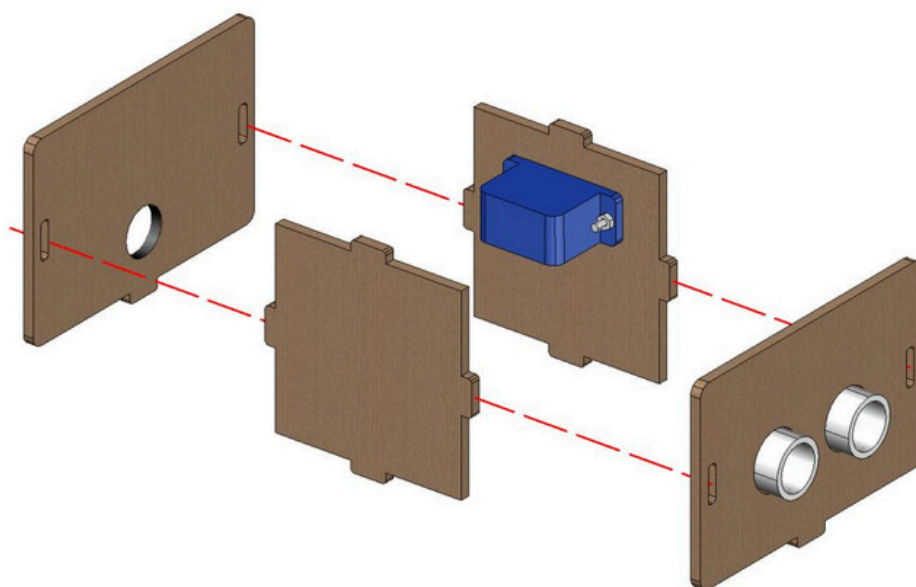
Krok 1 Instalace ultrazvukového modulu		
Seznam dílů	Ultrazvukový modul*1	Řada Dupont samec-samice*4
Ilustrace		
	Připojte vodiče DuPont k pinům ultrazvukového modulu: červený vodič k VCC, modrý vodič k Trig, bílý vodič k Echo a černý vodič k GND.	

Krok 2 Instalace serva

Seznam dílů	Servo *1	M2*14mm kulatá hlava Šrouby*2	Matice M2*2
Ilustrace			

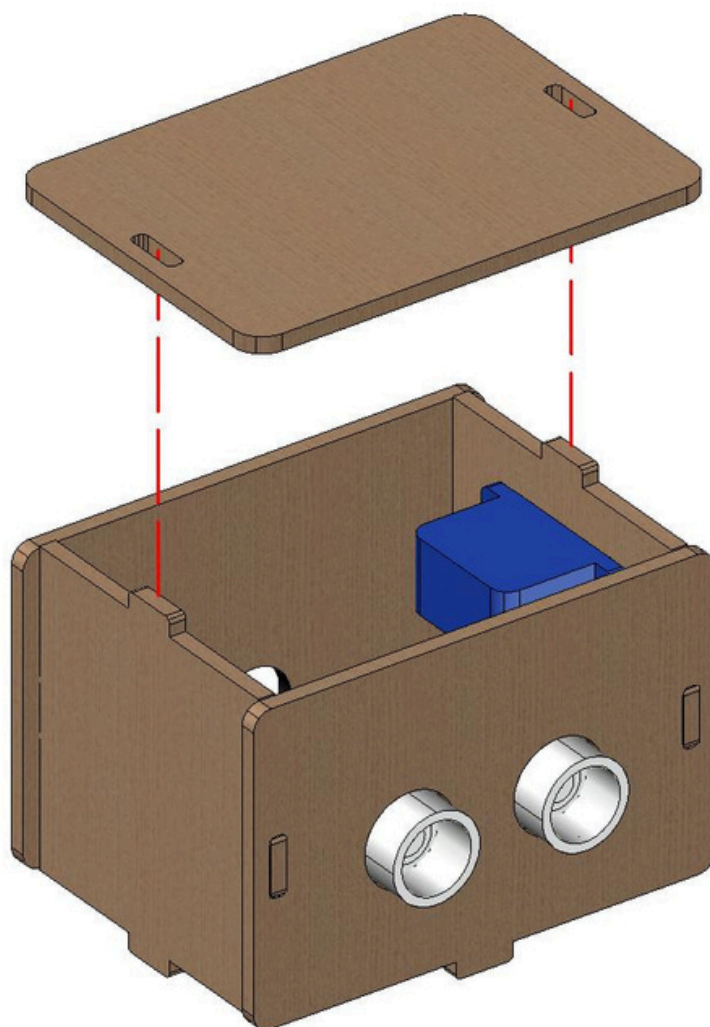
Krok 3 Instalace brány

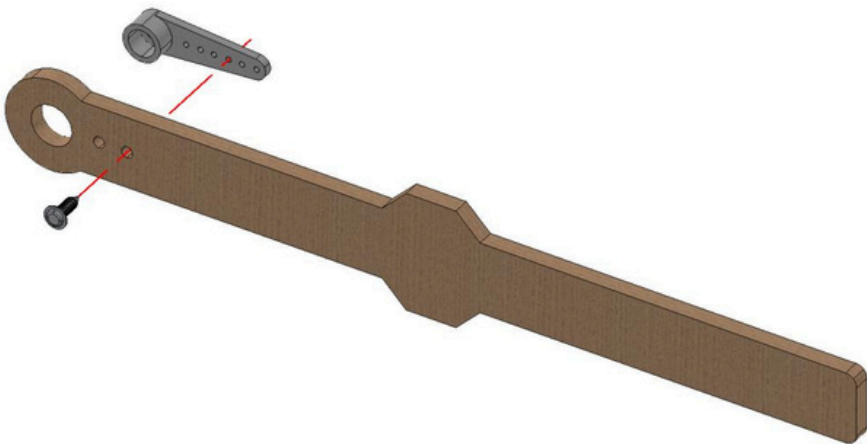
Ilustrace

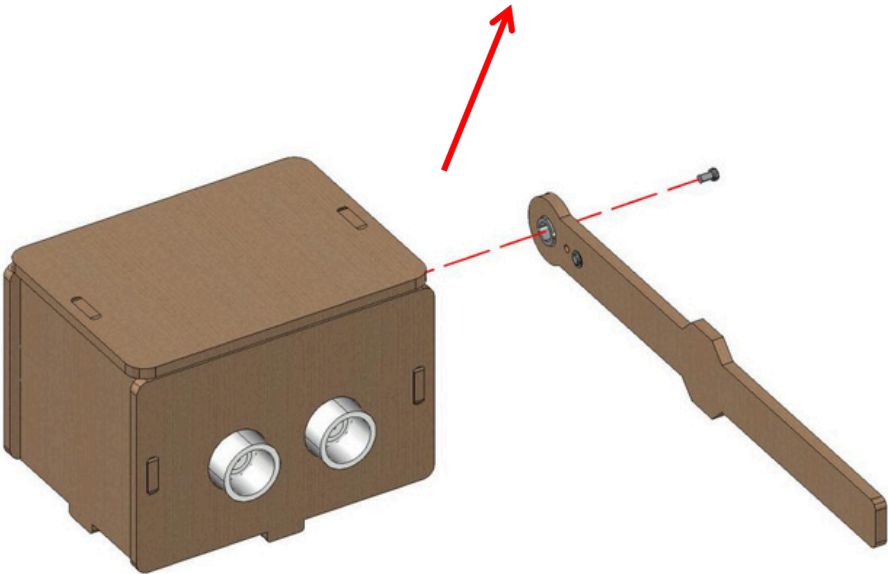


Krok 4 Instalace krytu brány

Ilustrace

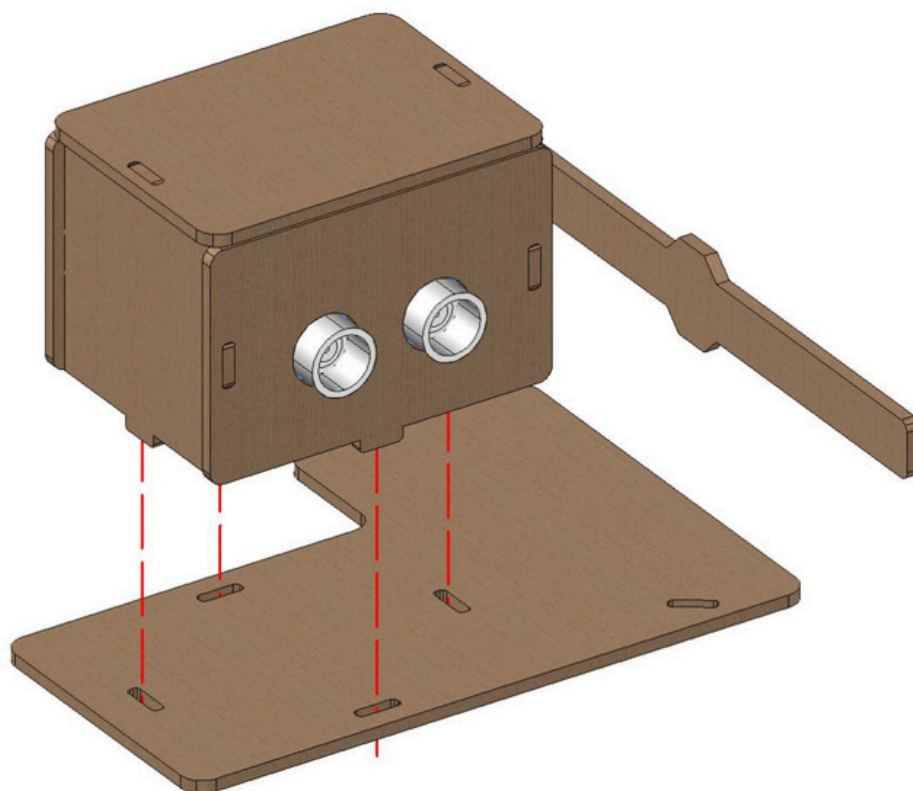


Krok 5 Nainstalujte volant		
Seznam dílů	M1,7*6 Samořezné Šroub*1	Řídicíkotouč s polovičními rameny*1
Ilustrace		

Krok 6. Instalace zvedací tyče	
Seznam dílů	Šrouby s kulatou hlavou M2,5*4*1
Ilustrace	Před zajištěním ramene výtahu nahrajte soubor „Servo_Initialize.ino“ do souboru „English\Arduino(Experienced Learner)\2.Code\lesson 35 Smart Barrier Gate\Servo_Initialize“, čímž otočíte servo o 90 stupňů.
	

Krok 7 Instalace základny brány

Ilustrace

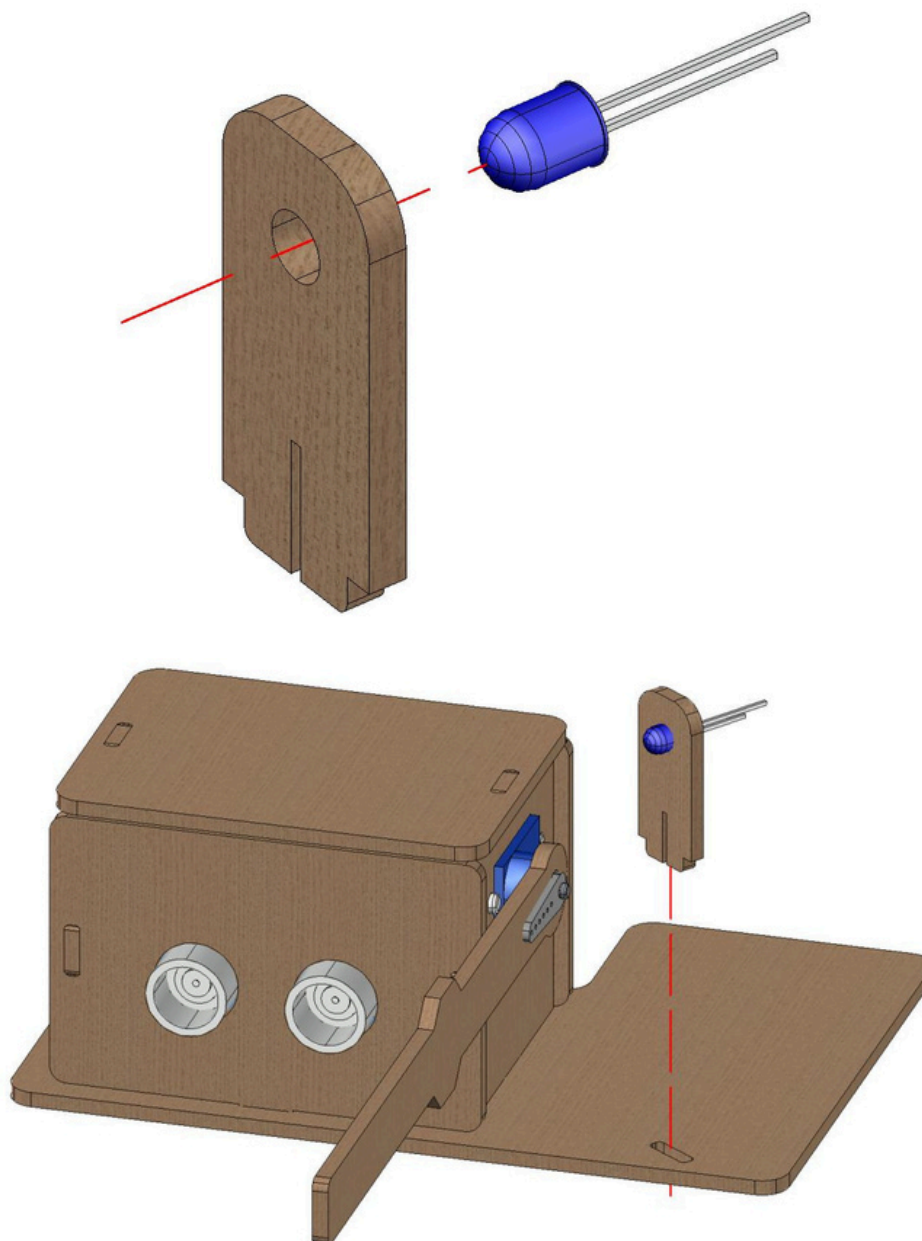


Krok 8 Instalace LED diody

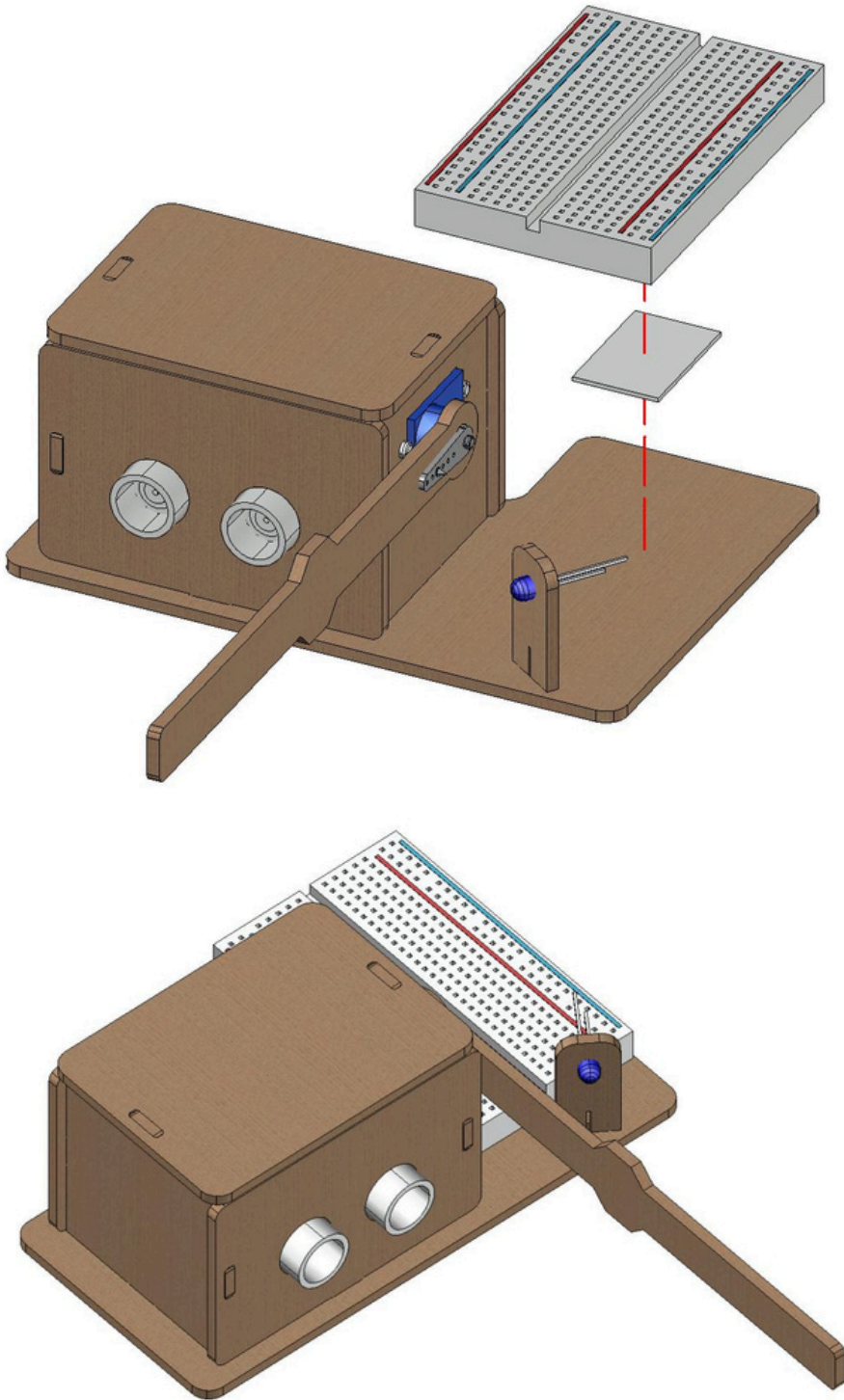
Seznam dílů

LED*1

Ilustrace



Krok 9. Instalace nepájivého pole

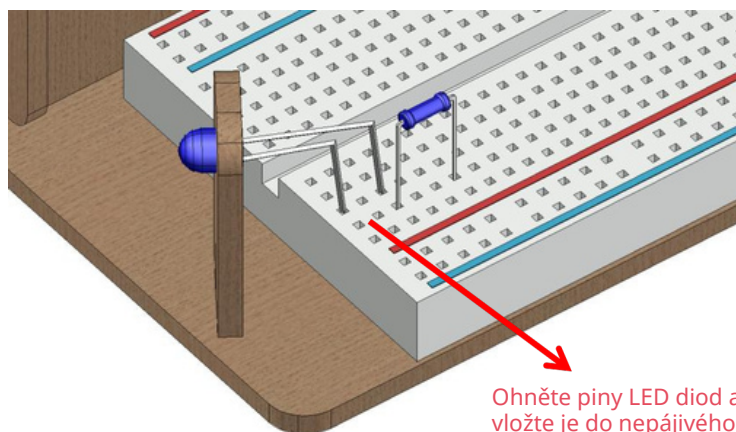
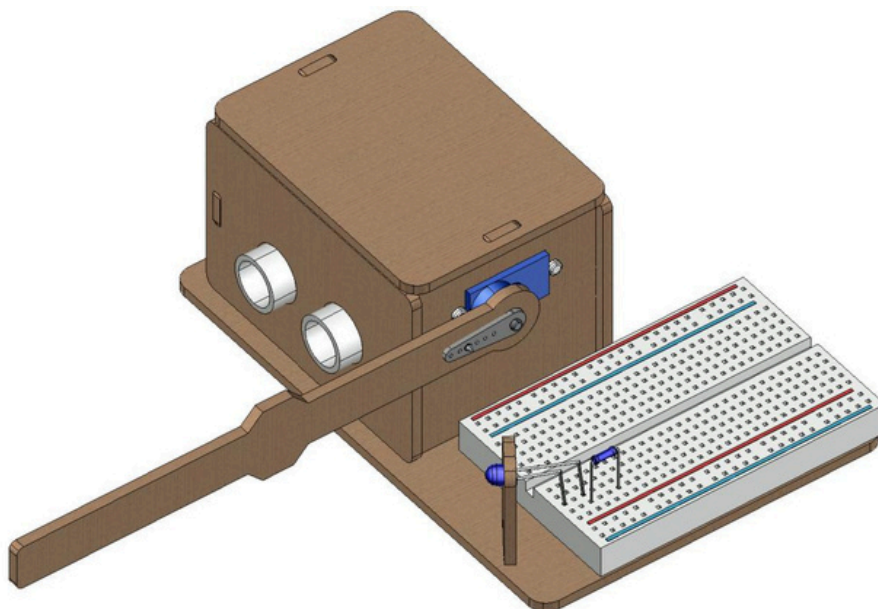
Seznam dílů	Prkénko*1	Oboustranná páska*1
Ilustrace		

Krok 10 Instalace LED diody

Seznam dílů

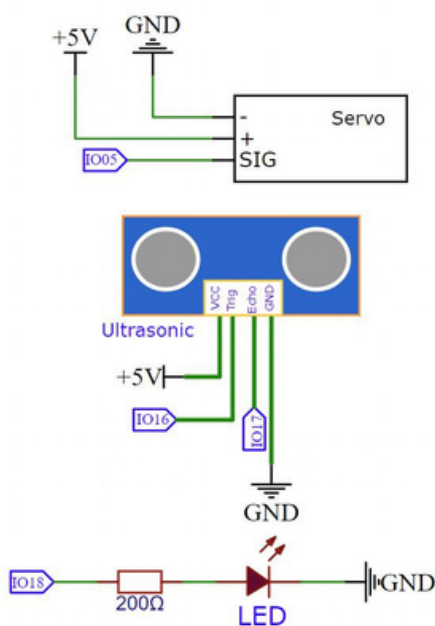
rezistor 220Ω

Ilustrace

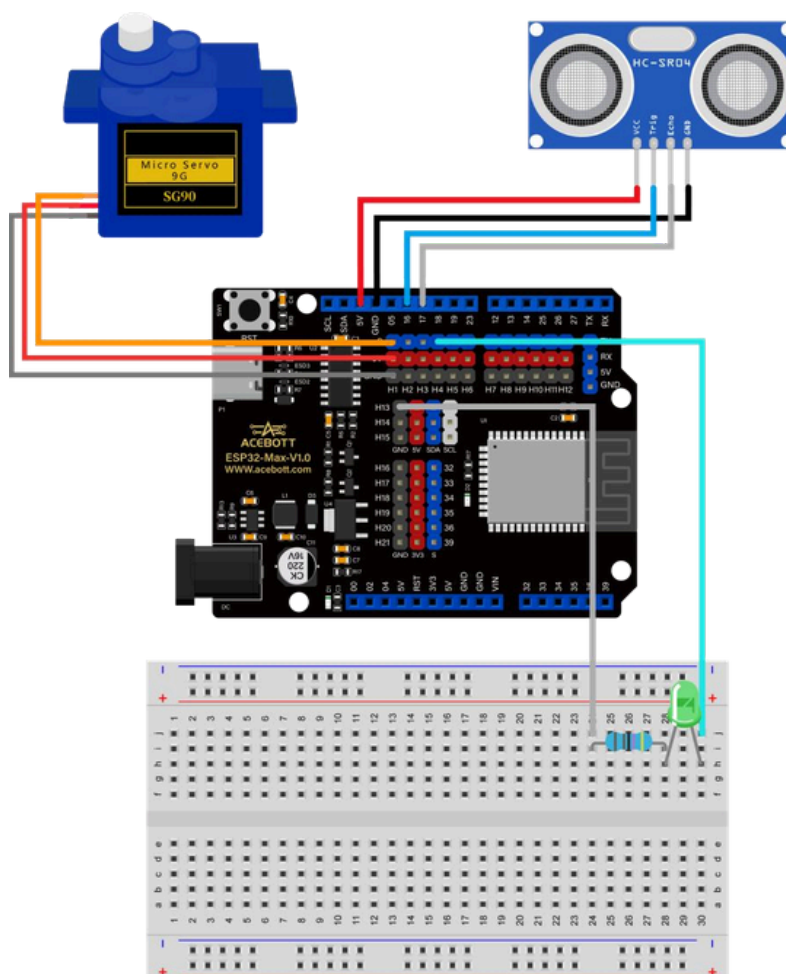


Ohněte piny LED diod a
vloďte je do nepájivého pole.

4. Schéma zapojení



5. Připojení hardwaru



6. Referenční postupy

Otevřete "[Smart Barrier Gate.ino](#)" soubor v adresáři „English\Arduino(Experienced Learner)\2.Code\lesson 36 Smart Barrier Gate\Smart_Barrier_Gate“ nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle dříve naučených kroků klikněte  nahrát program.

```
#include <ESP32Servo.h>

#define TRIG_PIN 16 // Ultrasonic sensor trigger pin
#define ECHO_PIN 17 // Ultrasonic sensor echo pin
#define LED_PIN 18 // Indicator LED pin
#define SERVO_PIN 5 // Servo motor control pin

Servo servo;
int angle = 90;
int flag = 0; // State flag: 0 = closed, 1 = opened

void setup() {
  Serial.begin(115200); // Initialize serial monitor
  pinMode(TRIG_PIN, OUTPUT); // Set trigger pin as output
  pinMode(ECHO_PIN, INPUT); // Set echo pin as input
  pinMode(LED_PIN, OUTPUT); // Set LED pin as output
  servo.attach(SERVO_PIN); // Attach servo to pin
  servo.write(angle); // Default angle: 90° (closed position)
}

void loop() {
  long distance = readDistance(); // Read distance from ultrasonic sensor
  Serial.println(distance); // Print distance to serial monitor
  // If object (e.g., car) is detected within 14 cm and gate is closed
  if (distance > 0 && distance <= 14 && flag == 0) {
    digitalWrite(LED_PIN, HIGH); // Turn on LED (indicates presence)
    for (angle = 90; angle > 0; angle--) {
      servo.write(angle); // Gradually close the lid
      delay(5); // Small delay for smooth motion
    }
    flag = 1; // Update state to opened
  }
  // If object is gone (distance > 20 cm) and gate is opened
  else if (distance > 20 && flag == 1) {
    digitalWrite(LED_PIN, LOW); // Turn off LED
    for (angle = 0; angle < 90; angle++) {
      servo.write(angle); // Gradually open the lid
      delay(5); // Small delay for smooth motion
    }
  }
}
```

```
    flag = 0;                                // Update state to closed
  }
  delay(500); // Small delay for stability
}

// Function to read distance in centimeters from ultrasonic sensor
float readDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH); // Timeout after 30ms
  long distance = duration * 0.034 / 2;     // Convert time to distance
  delay(100);                             // Small delay to
  stabilize reading
  return distance;
}
```

Po úspěšném nahrání programu se rameno zvedne, když je před branou auto, a spustí se, když před ní žádné auto není.



Lekce 37 Oscilační ventilátor



Cíle kurzu

- Pochopíte princip fungování oscilačního ventilátoru.
- Naučíte se sestavit model oscilačního ventilátoru.
- Napište program pro implementaci funkcí řazení převodů a oscilace oscilačního ventilátoru.



Úvod do kurzu

Během horkého léta často používáme elektrické ventilátory k ochlazování. Na rozdíl od běžných ventilátorů se oscilační ventilátory mohou pohybovat doleva a doprava a rovnoměrně rozvádět chladný vzduch po celé místnosti. V moderních domácích spotřebičích je této schopnosti „automatického řízení“ dosaženo především motory nebo servopohony.



V této lekci se naučíme, jak pomocí servopohonů, motorů a programového řízení postavit jednoduchý „oscilační ventilátor“ a simulovat jeho přívod vzduchu a oscilační funkce!



Body znalostí

1. Jak funguje oscilační ventilátor?

Oscilační ventilátor se skládá ze dvou hlavních součástí: motoru, který pohání lopatky, aby se otáčely vysokou rychlostí a vytvářely tak nepřetržitý vánek; a oscilačního mechanismu, který způsobuje, že se hlava ventilátoru kývá doleva a doprava, čímž se vzduch rovnoměrně rozděluje po větší ploše. Tradiční oscilační ventilátory používají malý motor a převodovky k přeměně rotačního pohybu motoru na vratný pohyb hlavy ventilátoru. Moderní inteligentní oscilační ventilátory naopak často používají servopohony, které přijímají řídicí signály pro přesné nastavení úhlu kmitání, čímž se dosahuje flexibilní a programovatelné levé a pravé oscilace.

V reálném provozu motor otáčí lopatkami ventilátoru pro generování větru a oscilační mechanismus řídí hlavu ventilátoru tak, aby opakovaně oscilovala v přednastaveném rozsahu úhlů (například 45 stupňů doleva a doprava). Rychlost a amplitudu oscilace lze nastavit mechanickým provedením nebo programovým kódem. ESP32 přesně řídí servopohon nebo motor pomocí řídicích signálů, čímž dokončuje automatickou oscilační funkci oscilačního ventilátoru.



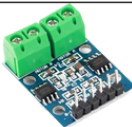
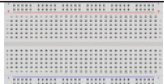
Experiment

1. Popis projektu

Tento program implementuje inteligentní systém oscilačního ventilátoru. Uživatelé mohou ventilátor zapínat a vypínat a ovládat rychlost větru (tři nastavení) jediným stisknutím tlačítka. Mohou také ovládat servo, které osciluje doleva a doprava a simuluje tak různé úhly proudění vzduchu. Systém kombinuje ovládání tlačítka, PWM regulaci rychlosti a řízení serva pro dosažení rychlosti větru.

nastavení a oscilace, simulující uživatelský zážitek skutečného elektrického ventilátoru.

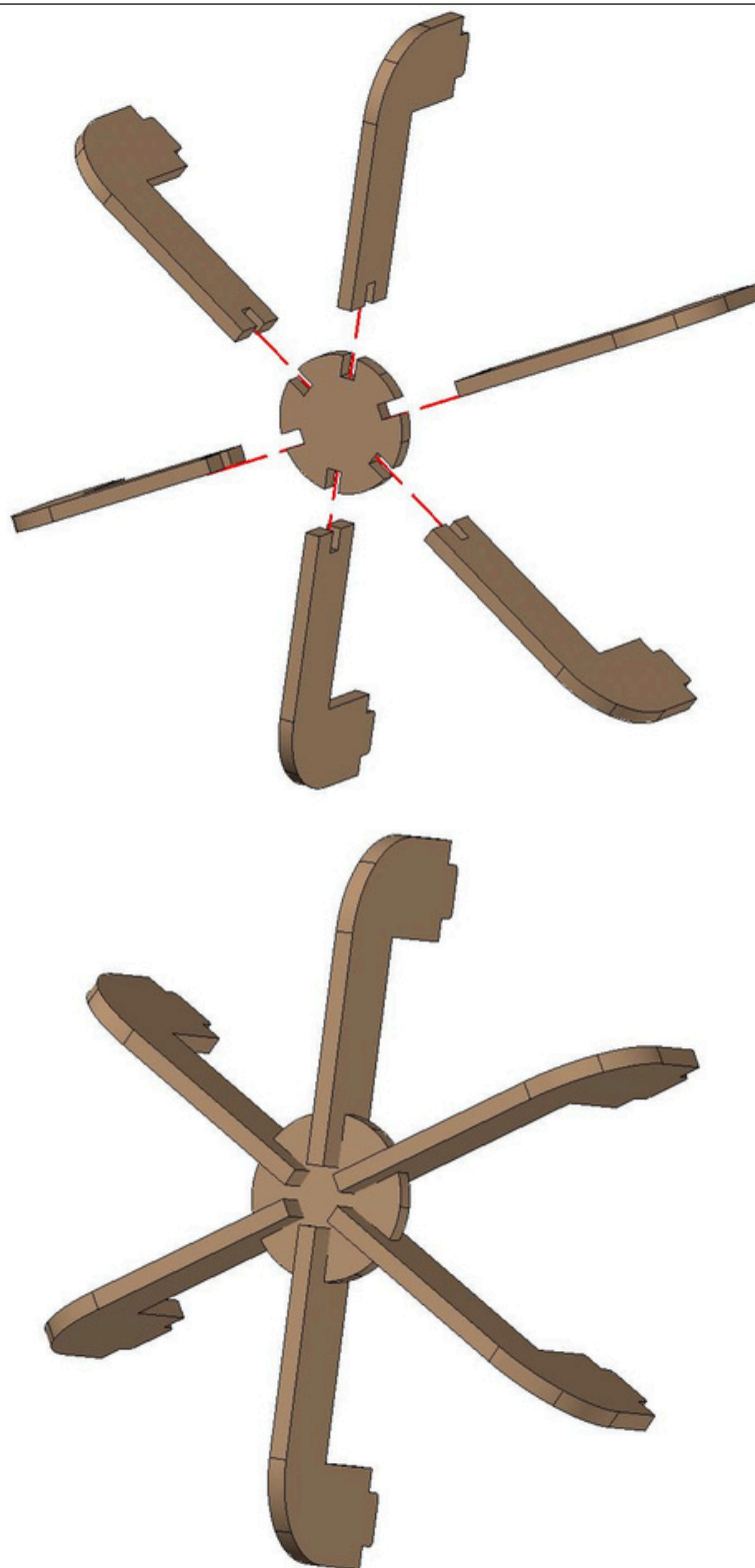
2. Seznam hardwaru

Obraz	Jméno	Množství
	Deska řadiče ESP32	1
	Ovladač stejnosměrného motoru	1
	Stejnosměrný motor	1
	Větrák	1
	Nepájivé pole s 400 spojovacími body	1
	Servo	1
	Tlačítko	3
	Rezistor 10 kΩ	3

3. Sestavení oscilačního ventilátoru (Pokud jste si nezakoupili stavební materiál, přeskočte kroky montáže a pokračujte přímo k připojení hardwaru a učení programu)

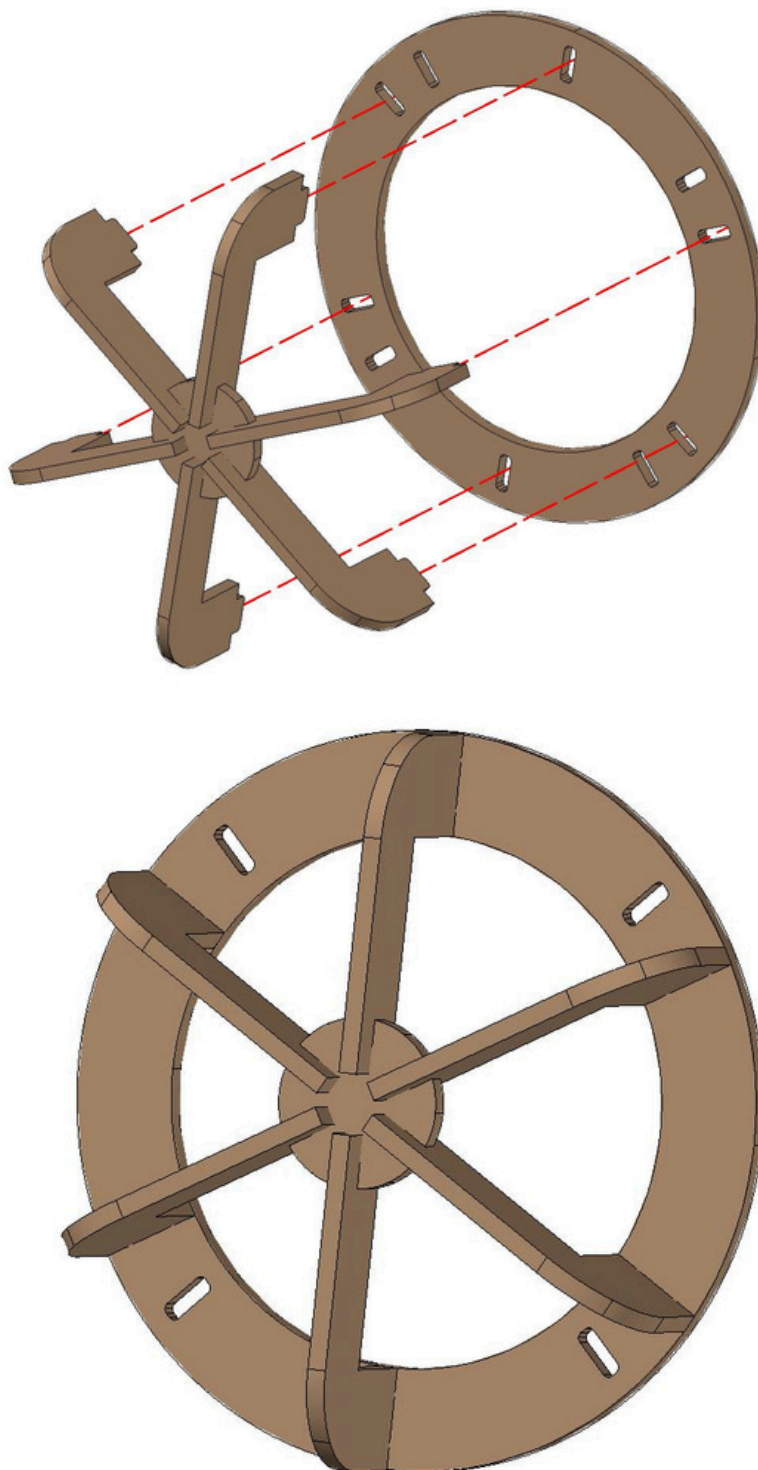
Krok 1 Sestavte kryt předního ventilátoru

Ilustrace



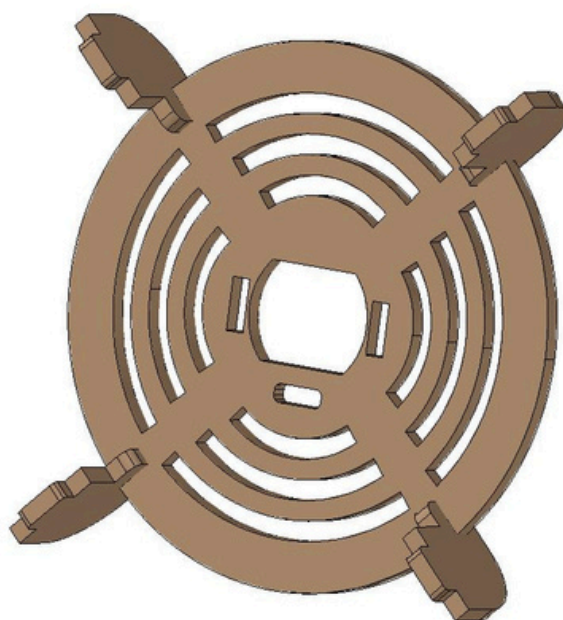
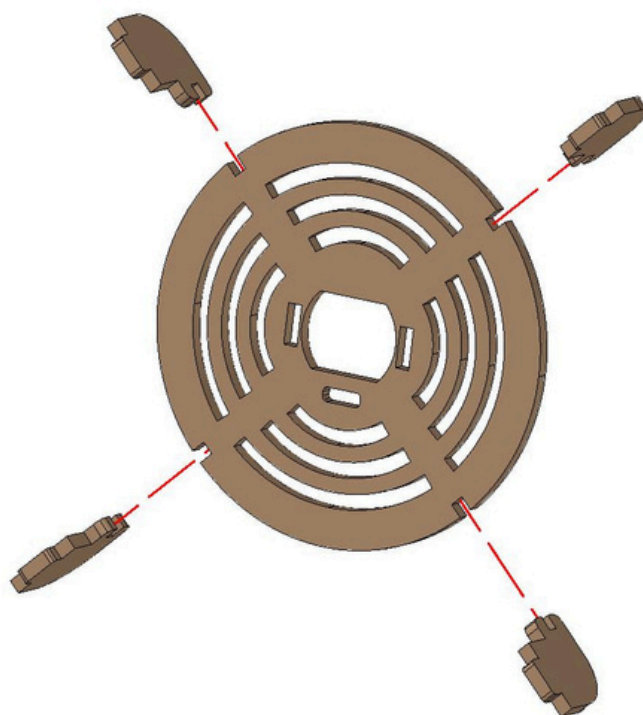
Krok 2. Instalace krytu předního ventilátoru

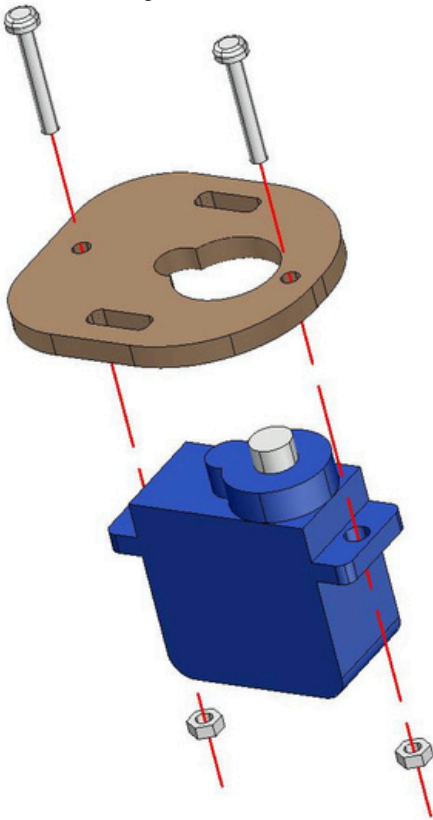
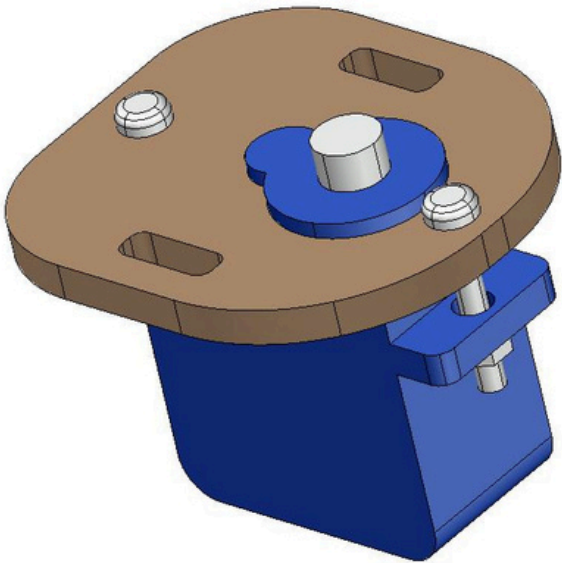
Ilustrace



Krok 3. Sestavte kryt zadního ventilátoru

Ilustrace



Krok 4 Instalace serva			
Seznam dílů	Servo *1	M2*14 kulatá hlava Šrouby*2	Matice M2*2
Ilustrace			
			

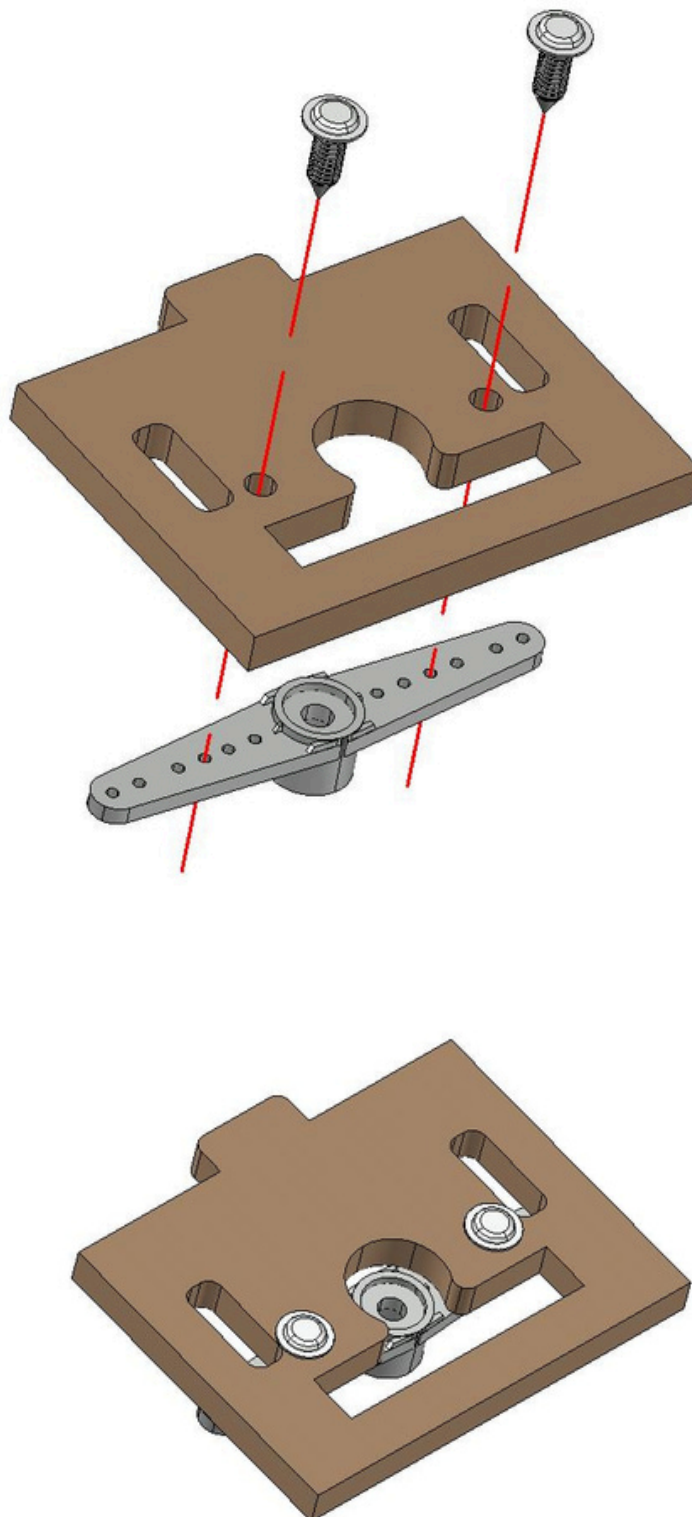
Krok 5. Nainstalujte volant

Seznam dílů

Volant*1

2x velké kulaté samořezné
šrouby s plochou hlavou M1,7*6

Ilustrace



Krok 6. Zajištění volantu

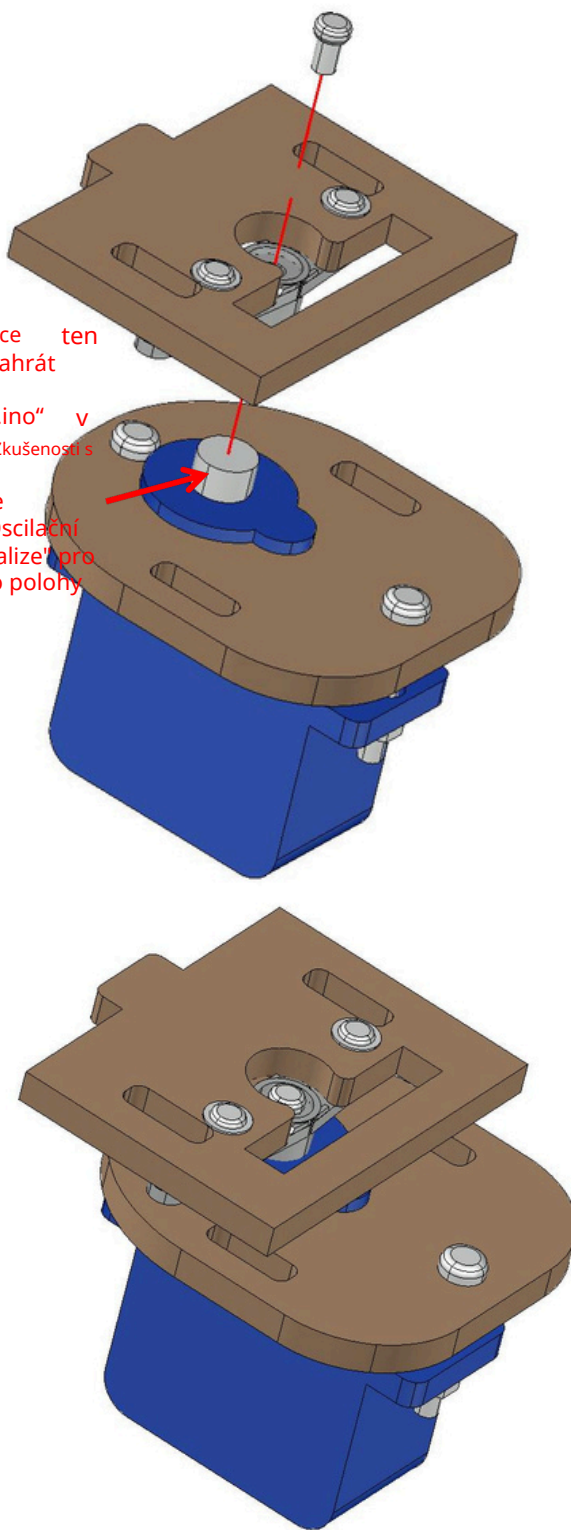
Seznam dílů

Šroub s kulatou hlavou M2,5*4*1

Ilustrace

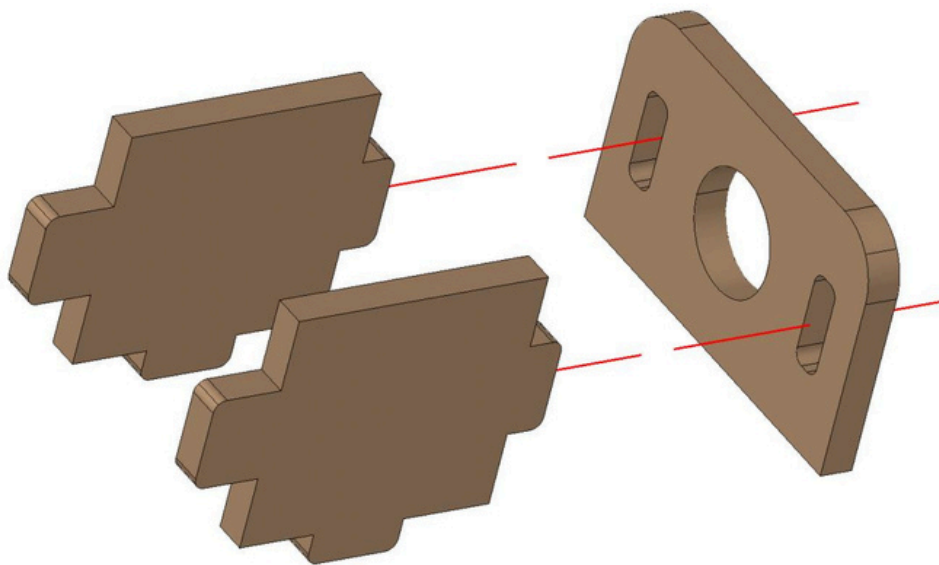
Před instalací ten řízení kolo, nahrát program „Servo_Initialize.ino“ v „Angličtina\Arduino(Zkušenosti s

Žák)\2.Kód\lekce 36 Oscilační "Fan\Servo_Initialize" pro otočení serva do polohy 90 stupňů.



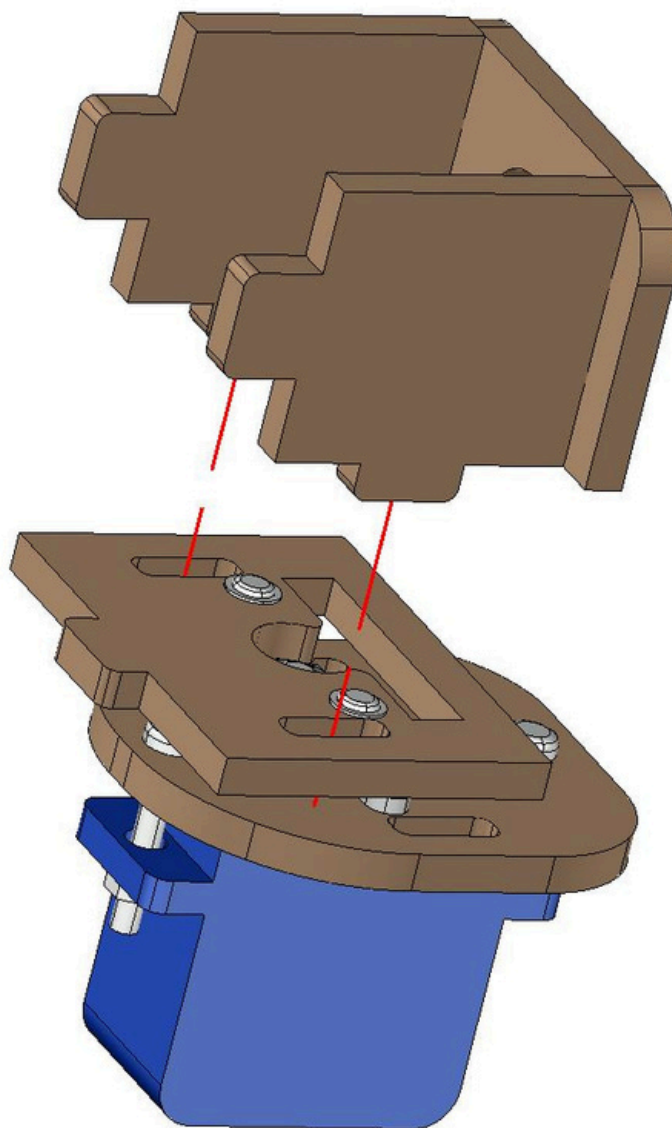
Krok 7. Sestavte držák motoru

Ilustrace



Krok 8. Instalace držáku motoru

Ilustrace

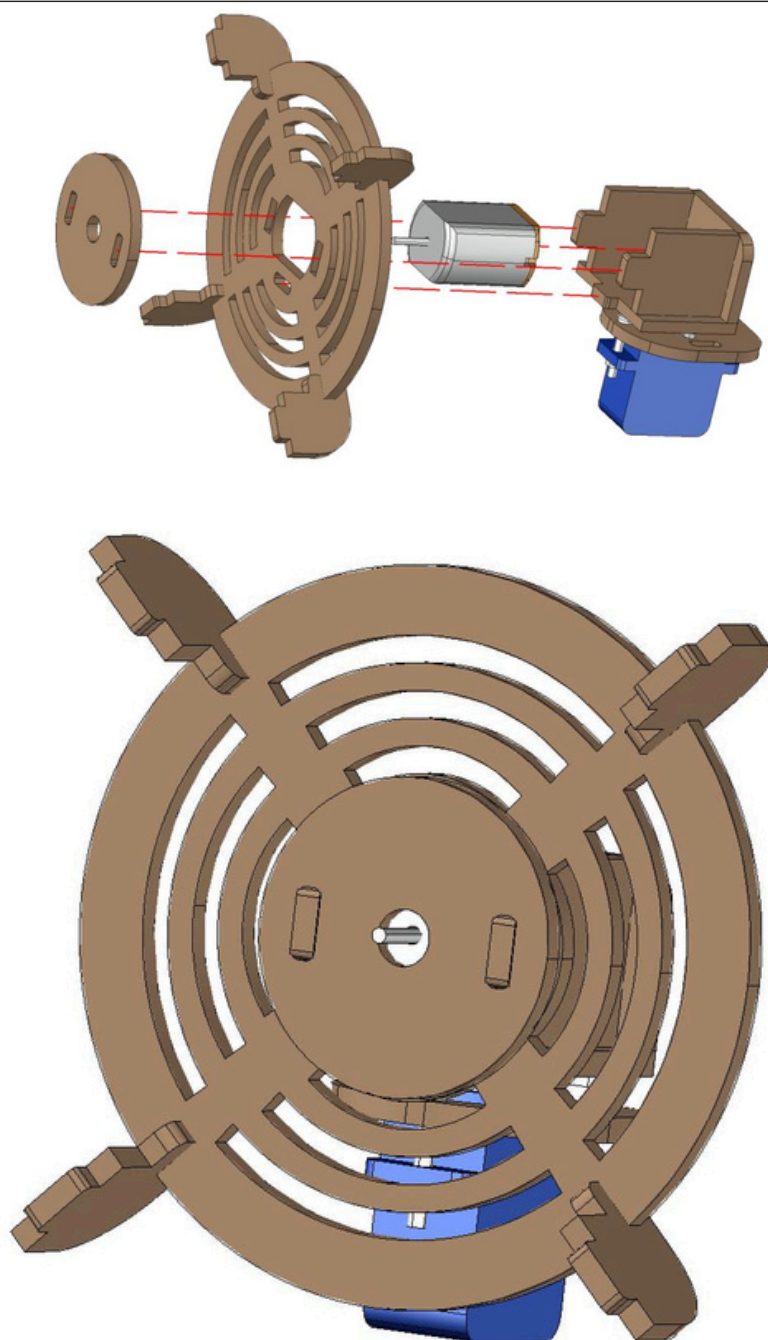


Krok 9 Instalace motoru a zadního krytu

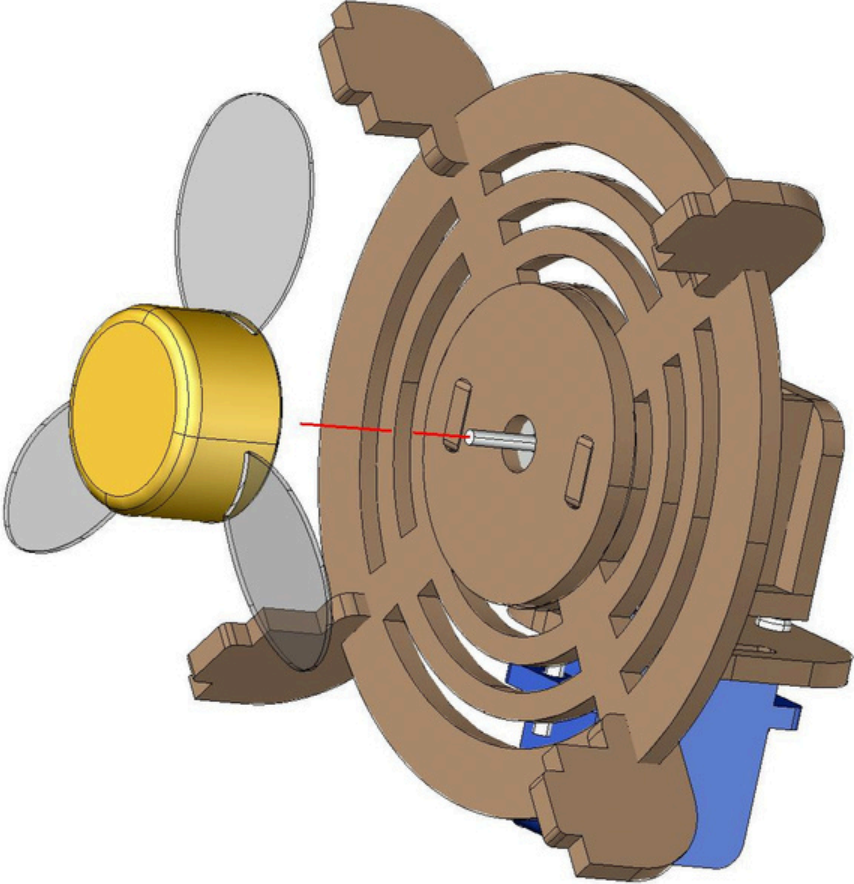
Seznam dílů

Motor*1

Ilustrace

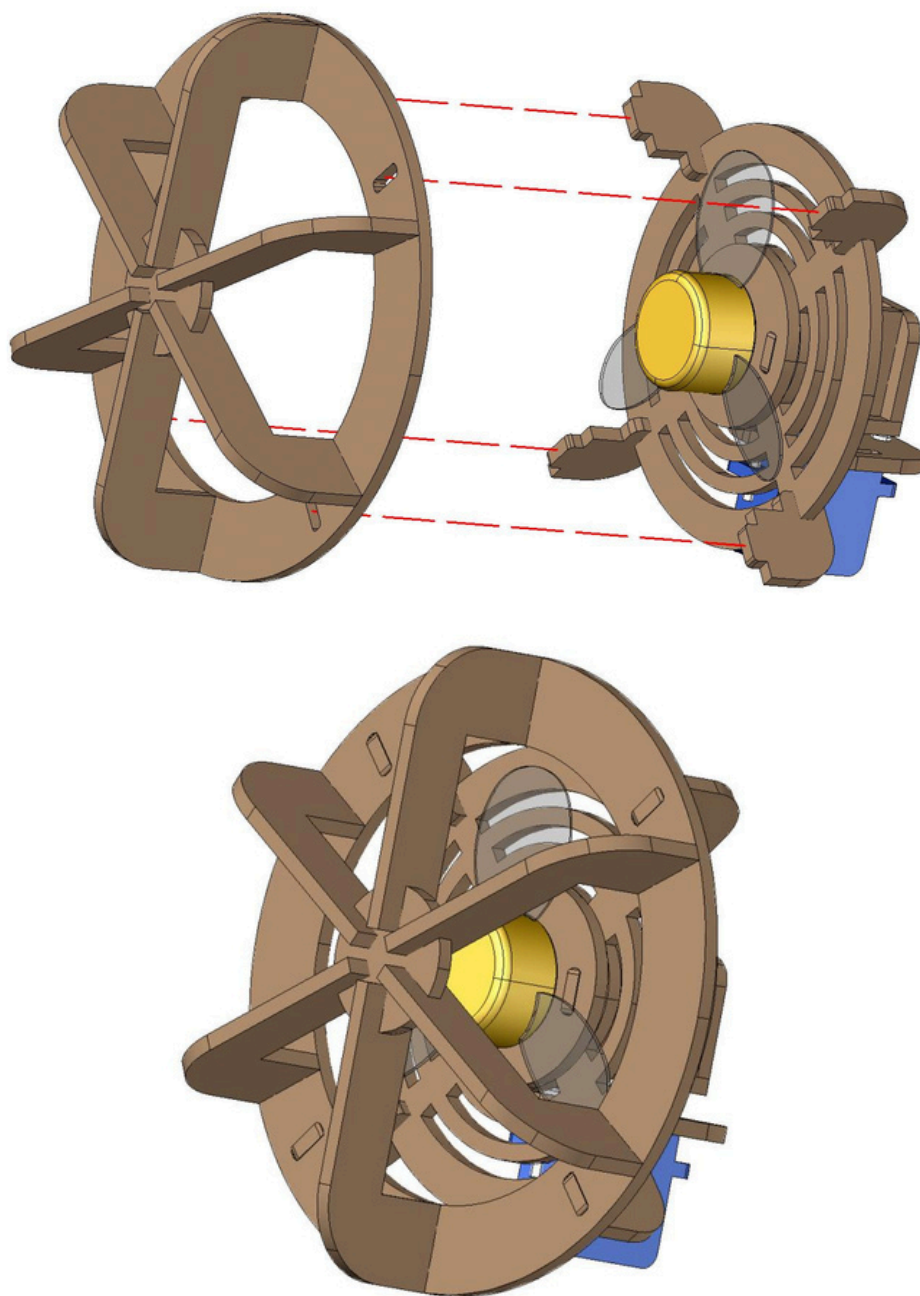


Krok 10 Nainstalujte lopatky ventilátoru

Seznam dílů	Lopatka ventilátoru*1
Ilustrace	

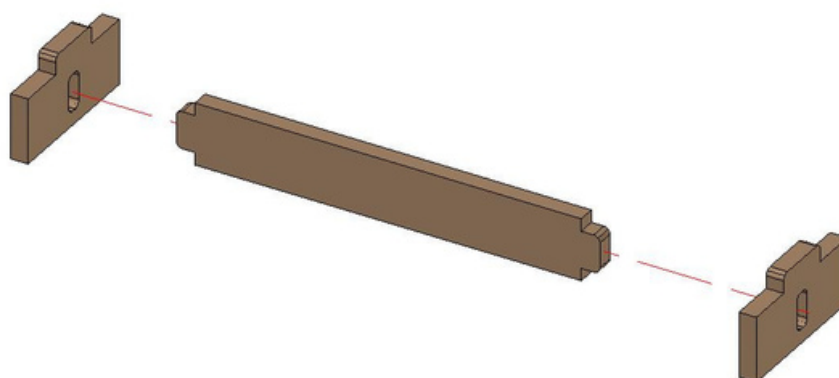
Krok 11 Nainstalujte přední kryt

Ilustrace



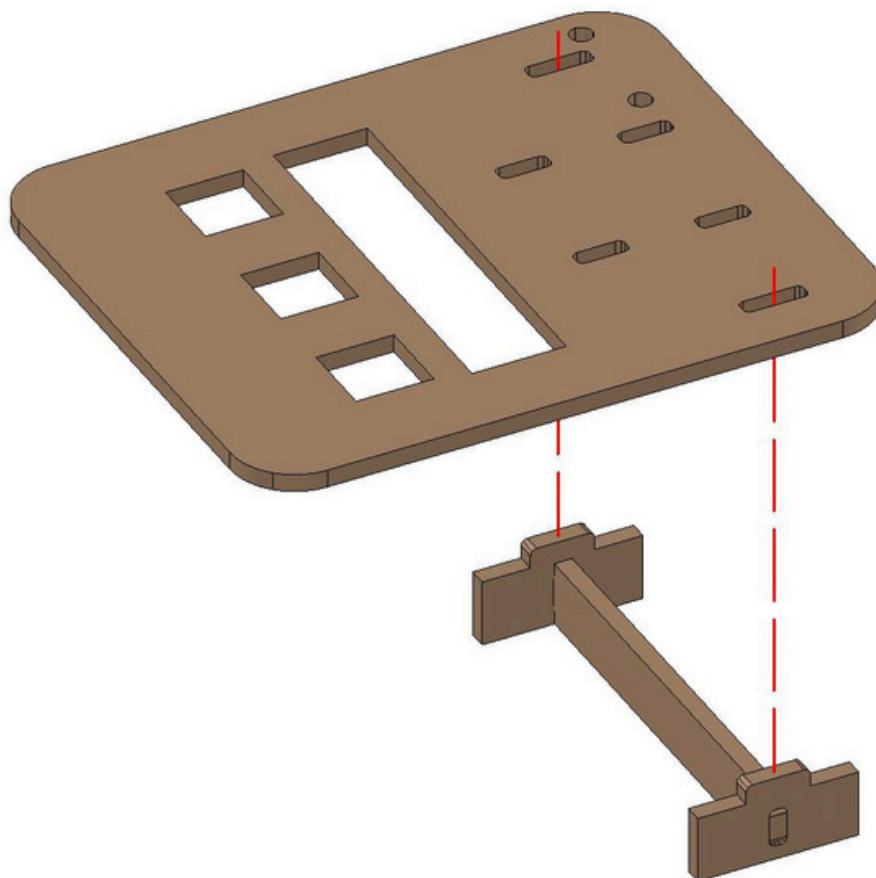
Krok 12 Instalace podpěrné konzoly základny ventilátoru

Ilustrace



Krok 13 Sestavte základnu ventilátoru

Ilustrace



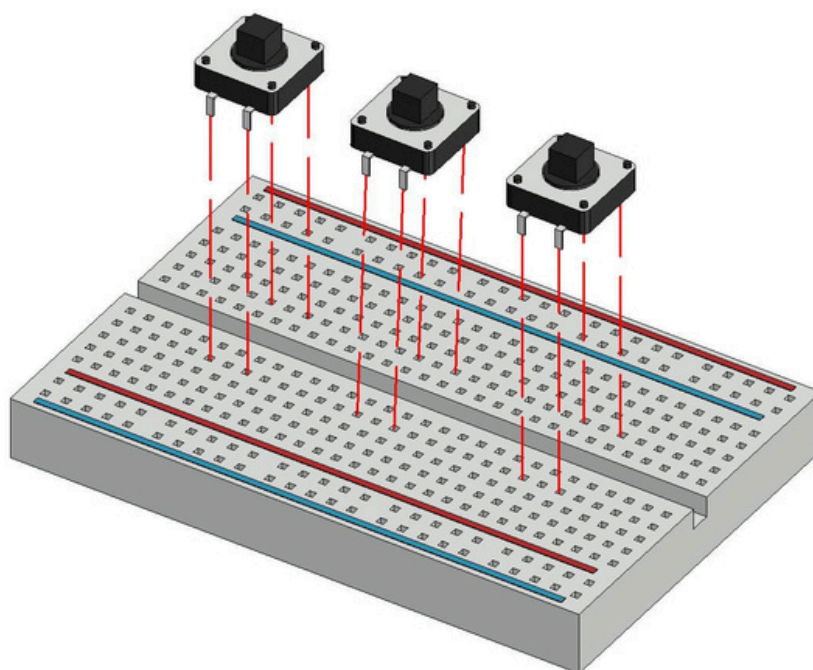
Krok 14 Instalace tlačítka

Seznam dílů

Tlačítka*3

Prkénko*1

Ilustrace

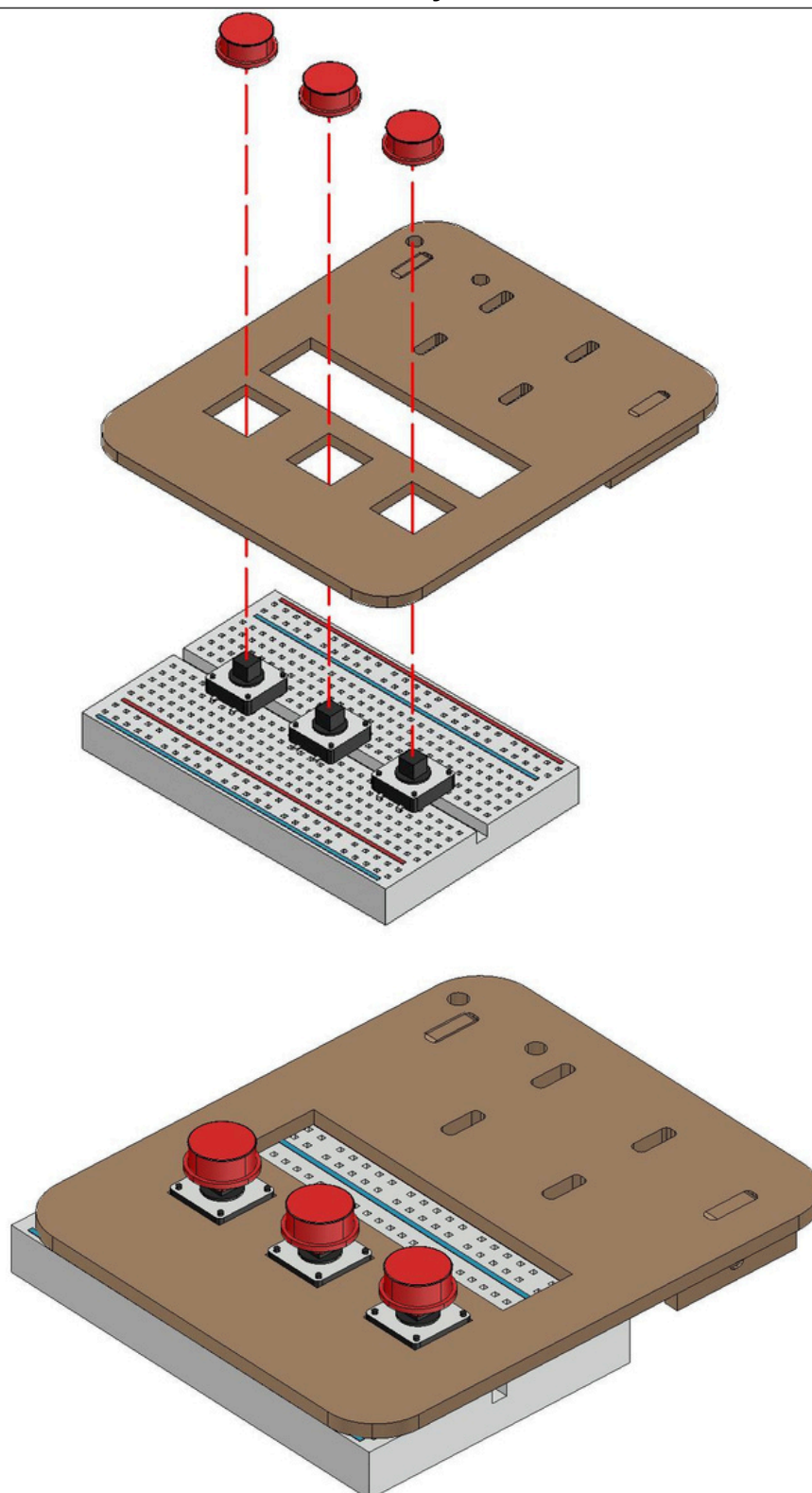


Krok 15. Instalace kláves

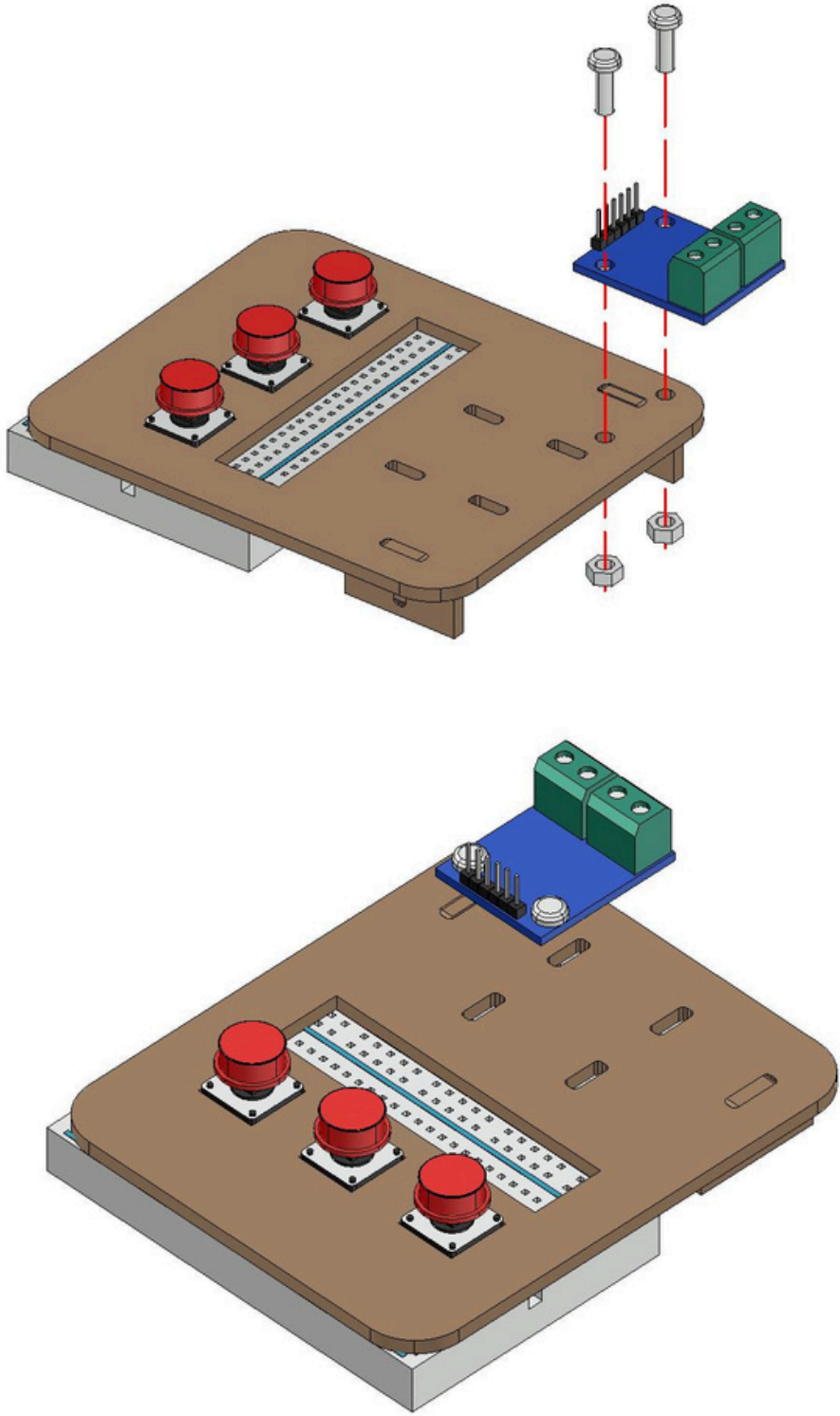
Seznam dílů

Klávesy*3

Ilustrace

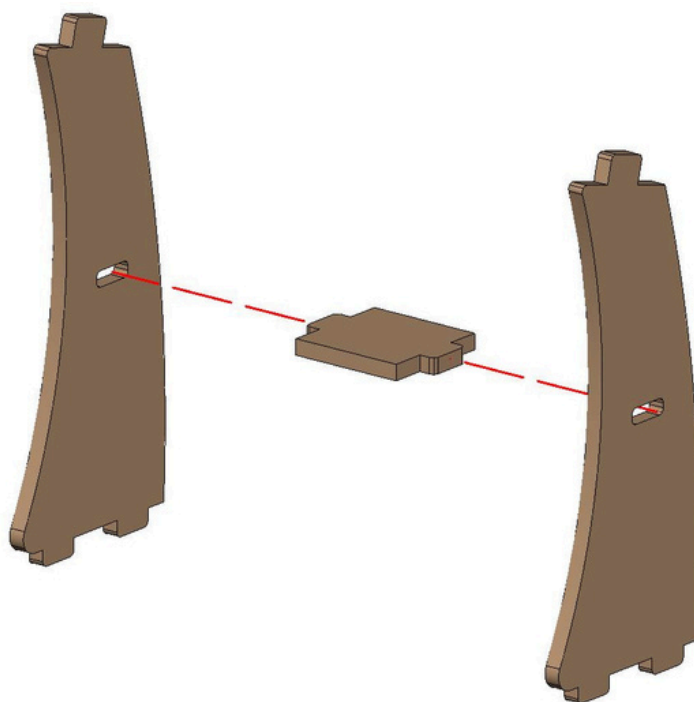


Krok 16 Instalace desky ovladače motoru

Seznam dílů	M3*10mm kulatá hlava Šroub*2	Matice M3*2	Ovladač motoru Deska*1
Ilustrace			

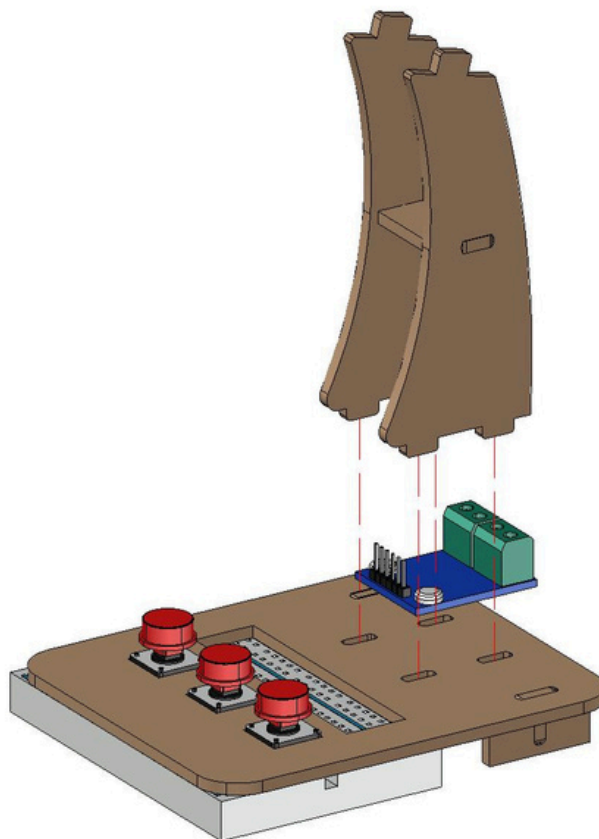
Krok 17 Sestavte podpůrné sloupky

Ilustrace



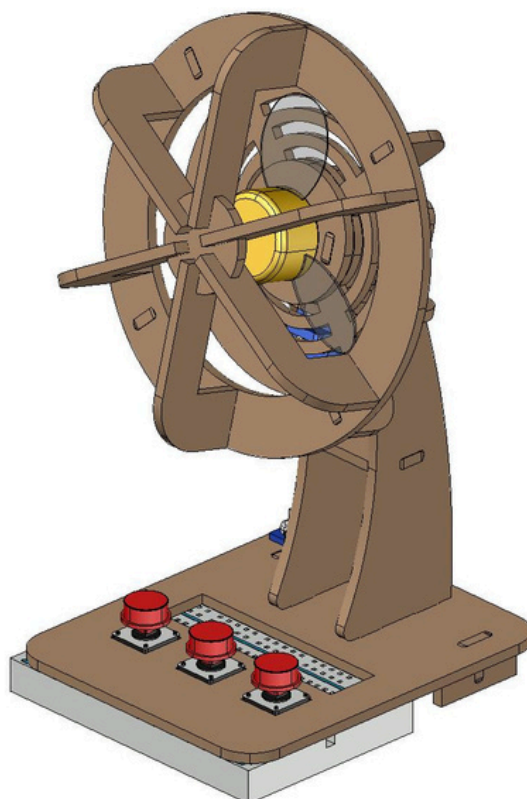
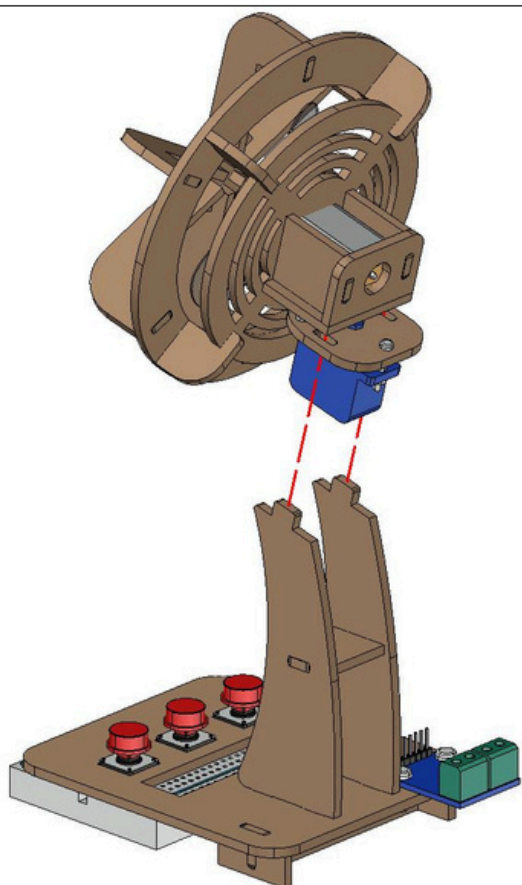
Krok 18 Instalace podpůrných sloupků

Ilustrace

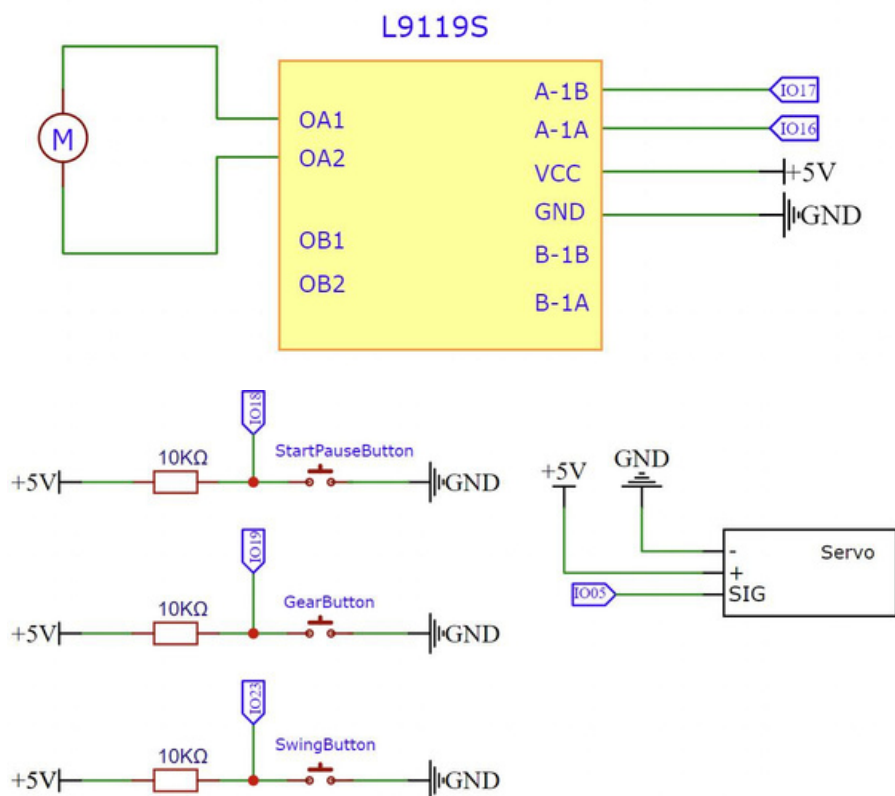


Krok 19 Zajistěte sběrnici ventilátoru

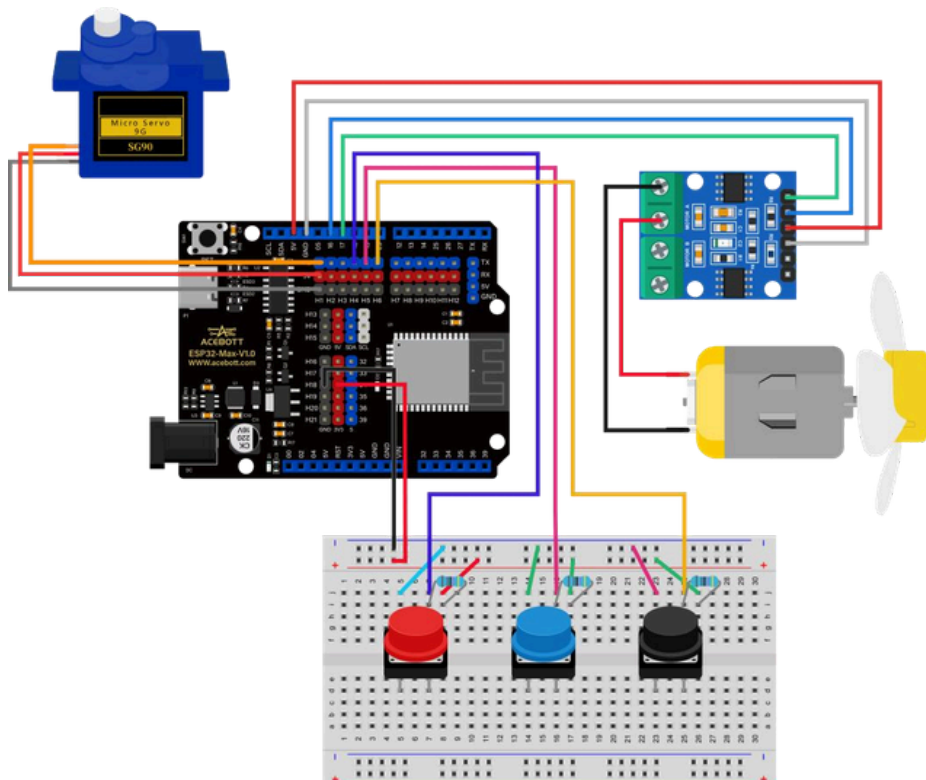
Ilustrace



4. Schéma zapojení



5. Hardwarová připojení



6. Referenční program

Otevřete "[Oscilating Fan.ino](#)" soubor v adresáři „English\Arduino(Experienced Learner)\2.Code\lesson 376 Oscilating Fan\Oscillating_Fan“ nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu podle pokynů dříve naučené kroky, klikněte  nahrát program.

```
#include <ESP32Servo.h>

const int servoPin = 5; // Servo connected to pin 5 on ESP32
const int fanPin = 16; // Fan control pin
const int unusedPin = 17; // Unused pin, kept LOW
const int buttonPin1 = 18;
const int buttonPin2 = 19;
const int buttonPin3 = 23;

Servo myServo; // Create a servo object
bool isMoving = false; // Whether the servo is currently moving
float angle = 90;
int dutyCycle = 0;
int currentFanLevel = 0;
int isWaving = 0;
int is_speedcontrol = 0;
unsigned long lastDebounceTime = 0; // Last debounce time
const unsigned long debounceDelay = 50; // Debounce delay in milliseconds
int reading_1;

int buttonState1 = 0; // Current state of button 1
int lastButtonState1 = 0; // Previous state of button 1
int buttonToggleState1 = 0; // Toggle state for button 1 (0 or 1)

int buttonState2 = 0; // Current state of button 2
int lastButtonState2 = 0; // Previous state of button 2
int buttonToggleState2 = 0; // Toggle state for button 2 (unused in logic)

int buttonState3 = 0; // Current state of button 3
int lastButtonState3 = 0; // Previous state of button 3
int buttonToggleState3 = 0; // Toggle state for button 3 (0 or 1)

void setup() {
  Serial.begin(115200); // Initialize serial communication for debugging
  pinMode(buttonPin1, INPUT); // Set button 1 as input
  pinMode(buttonPin2, INPUT); // Set button 2 as input
  pinMode(buttonPin3, INPUT); // Set button 3 as input
  pinMode(fanPin, OUTPUT);
```

```
pinMode(unusedPin, OUTPUT);

// Set up PWM for fan using ledcWrite
ledcSetup(0, 20000, 8); // Channel 0, 20kHz frequency, 8-bit resolution
ledcAttachPin(fanPin, 0); // Attach fan pin to channel 0

// Configure servo library
ESP32PWM::allocateTimer(1); // Allocate a timer for PWM use
myServo.setPeriodHertz(50); // Standard 50Hz for servo
myServo.attach(servoPin, 500, 2500); // Attach servo to pin and define
pulse width range

digitalWrite(unusedPin, LOW); // Ensure unused pin stays LOW
myServo.write(90); // Set initial servo angle to 90°
delay(1000); // Wait for servo to stabilize
}

void loop() {
  button(); // Check button states

  // If servo is set to move, start waving
  if (isMoving) {
    startWaving();
  }
}

void button() {
  // Read button 1 state
  reading_1 = digitalRead(buttonPin1);
  if (reading_1 != lastButtonState1) {
    lastDebounceTime = millis(); // Timestamp for debounce
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading_1 != buttonState1) {
      buttonState1 = reading_1;
      if (buttonState1 == HIGH) {
        if (buttonToggleState1 == 0) {
          buttonToggleState1 = 1;
        } else if (buttonToggleState1 == 1) {
          buttonToggleState1 = 0;
        }
      }
    }

    // Toggle logic for button 1
    if (buttonToggleState1 == 1) {
      Serial.println("000");
      isWaving = 1;
      is_speedcontrol = 1;
    }
  }
}
```

```

currentFanLevel = 3;
speed_control(currentFanLevel);
} else if (buttonToggleState1 == 0) {
Serial.println("111");
currentFanLevel = 0;
speed_control(currentFanLevel);
isWaving = 0;
is_speedcontrol = 0;
isMoving = false; // Stop servo movement
}
}
}
lastButtonState1 = reading_1;

// Read button 2 state
buttonState2 = digitalRead(buttonPin2);
if (buttonState2 == LOW && lastButtonState2 == HIGH) { // On button press
currentFanLevel += 1;
if (currentFanLevel > 3) {
currentFanLevel = 1;
}
delay(50); // Simple debounce
speed_control(currentFanLevel);
}
lastButtonState2 = buttonState2;

// Read button 3 state
buttonState3 = digitalRead(buttonPin3);
if (buttonState3 == LOW && lastButtonState3 == HIGH) { // On button press
buttonToggleState3 = !buttonToggleState3;
if (buttonToggleState3 == 1 && isWaving == 1) {
isMoving = true; // Start waving
} else if (buttonToggleState3 == 0) {
isMoving = false; // Stop waving
}
delay(50); // Simple debounce
}
lastButtonState3 = buttonState3;
}

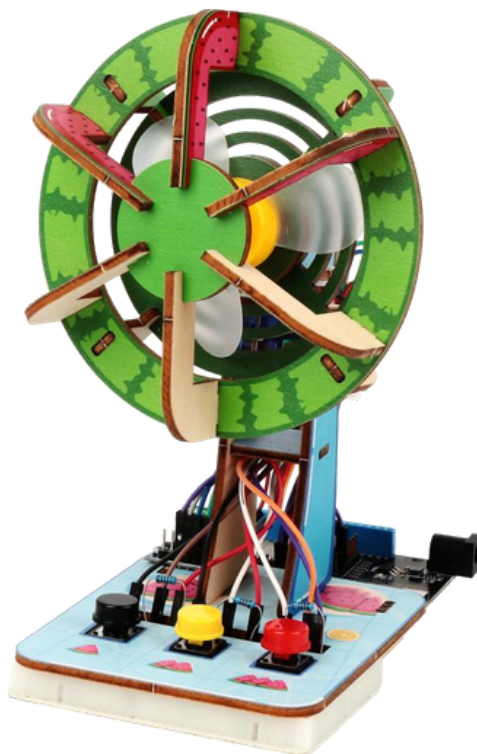
void startWaving() {
// Simple back-and-forth waving motion
static bool goingUp = true; // Controls direction

if (goingUp) {
angle += 1;
if (angle >= 170) {
goingUp = false;
}
}
}

```

```
} else {  
  angle -= 1;  
  if (angle <= 10) {  
    goingUp = true;  
  }  
}  
  
myServo.write(angle);  
delay(20); // Adjust delay to control speed of waving  
}  
  
void speed_control(int command) {  
  if (is_speedcontrol == 1) {  
    switch (command) {  
      case 0:  
        dutyCycle = 0;  
        Serial.println("Fan set to 0% speed");  
        break;  
      case 1:  
        dutyCycle = 200;  
        Serial.println("Fan set to 50% speed");  
        break;  
      case 2:  
        dutyCycle = 220;  
        Serial.println("Fan set to 75% speed");  
        break;  
      case 3:  
        dutyCycle = 235;  
        Serial.println("Fan set to 100% speed");  
        break;  
      default:  
        Serial.println("Invalid command. Use 1, 2, or 3.");  
        return;  
    }  
    ledcWrite(0, dutyCycle); // Apply PWM duty cycle to fan  
  }  
}
```

Po úspěšném nahrání programu stiskněte černé tlačítko pro spuštění ventilátoru na první rychlost. Stiskněte žluté tlačítko pro přepínání rychlostí. Stiskněte červené tlačítko pro aktivaci funkce oscilace ventilátoru. Dalším stisknutím černého tlačítka ventilátor zastavíte a vypnete.



Lekce 38 Elektronické piano



Cíle kurzu

- Pochopení fungování elektronických kláves
- Napsání programu pro tvorbu efektů pro elektronické klávesy



Úvod do kurzu

Elektronické klávesy jsou oblíbeným hudebním nástrojem. Stisknutím různých kláves vznikají zvuky různých frekvencí, které se kombinují a vytvářejí krásné melodie.

V této lekci se naučíme, jak si vyrobit „elektronickou klávesnici“, která dokáže hrát jednoduché melodie a umožní vám zažít radost z „hraní“ vlastníma rukama.



Body znalostí

1. Elektronická klávesnice

Elektronické klávesy jsou klávesový nástroj, který elektronicky produkuje zvuk. Když stisknete různé klávesy, obvod uvnitř klávesnice generuje elektrické signály o různých frekvencích. Tyto signály jsou zesíleny a poté vysílají odpovídající tóny přes reproduktory. Elektronické klávesy jsou lehké, mají bohatý tón a jsou široce používány v hudebním vzdělávání, zábavě a produkci elektronické hudby.



Jak funguje elektronická klávesnice

Základním principem elektronických kláves je převod stisknutí kláves na elektrické signály odpovídajících frekvencí a přehrávání odpovídajících tónů přes reproduktory. Základní proces je následující:

(1) Ovládání kláves

Když hráč stiskne klávesu, aktivuje se spínač uvnitř elektronické klávesnice a vyšle elektrický signál odpovídající dané klávese.

(2) Generování frekvence

Elektrický signál je zpracováván elektronickým obvodem za účelem generování elektrického signálu o specifické frekvenci. Tato frekvence odpovídá výšce zvuku, který by měl klíč vydávat.

(3) Syntéza zvuku

Syntezátor uvnitř elektronické klávesnice zpracovává tyto frekvenční signály a simuluje tak zabarvení různých nástrojů.

(4) Zesílení signálu

Zpracovaný elektrický signál je zesílen zesilovačem, aby byl zvuk dostatečně hlasitý.

(5) Zvukový výstup

Nakonec je elektrický signál převeden reproduktorem na zvuk a hráč slyší odpovídající tón.

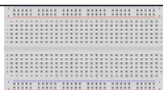


Experiment

1. Popis projektu

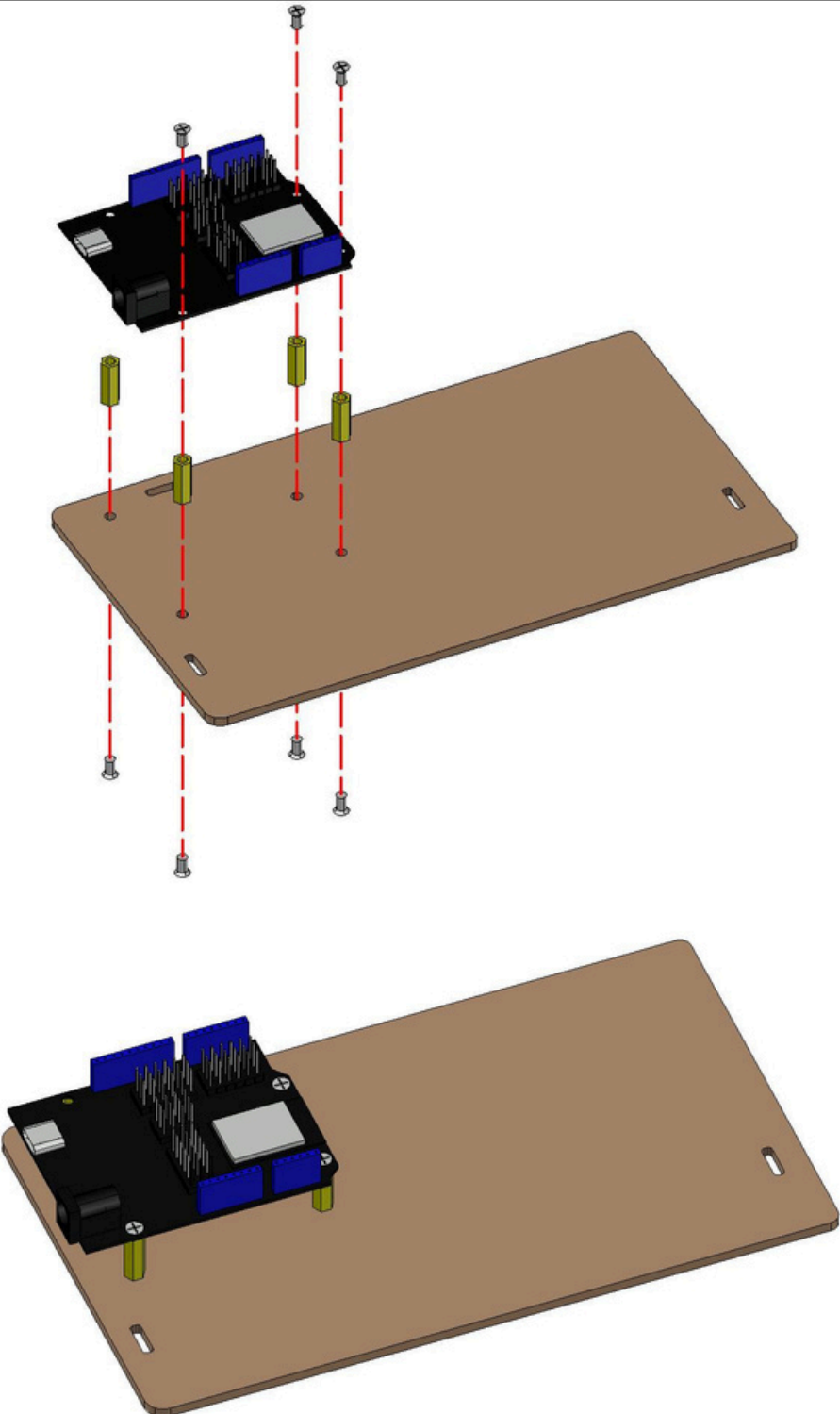
Připojením tlačítek k sedmi pinům může uživatel spustit sedm tónů (C4 až B4). Každé tlačítko odpovídá pevné frekvenci. Po stisknutí tlačítka bzučák přehraje odpovídající tón; po uvolnění tlačítka zvuk ustane.

2. Seznam hardwaru

Obráz	Jméno	Množství
	Deska řadiče ESP32	1
	Nepájivé pole s 400 spojovacími body	2
	Pasivní bzučák	1
	Tlačítko	7
	Rezistor 1 kΩ	1
	Rezistor 10 kΩ	7

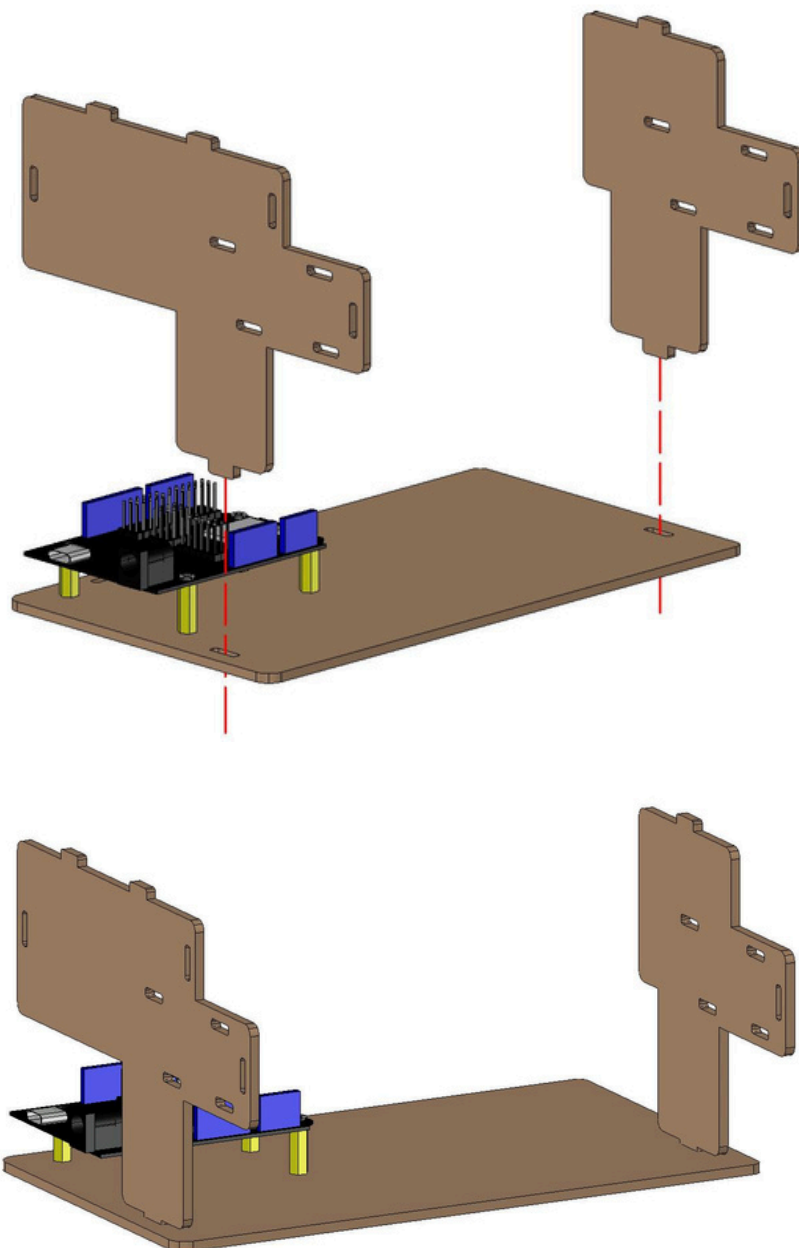
3. Sestavte elektronickou klávesnici (Pokud jste si nezakoupili stavební materiál, můžete krok montáže přeskočit a přejít přímo k připojení hardwaru a učení programu)

Krok 1 Nainstalujte desku řadiče ESP32

Seznam dílů	ESP32 Ovladač Deska*1	M3*12MM Dvouprůchodový Měděný sloup*4	M3*6MM Plochá hlava Šrouby*7
Ilustrace			

Krok 2 Instalace bočních panelů piana

Ilustrace



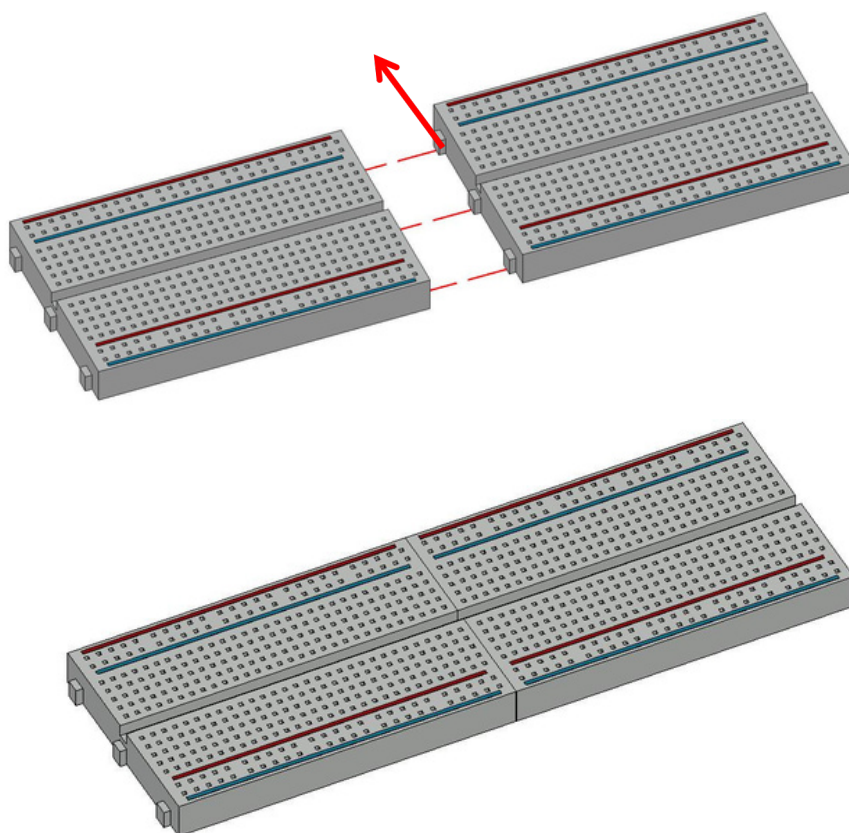
Krok 3. Sestavení nepájivého pole

Seznam dílů

400jamkové nepájivé pole*2

Ilustrace

Poznámka: Zarovnejte obě nepájivé desky tak, aby jejich strany těsně přiléhaly k sobě. Poté jemně zacvakněte stranu s vyvýšeným výstupkem směrem nahoru do drážky druhé nepájivé desky, dokud se bezpečně nezapojí. Pro snadné zapojení udržujte napájecí kolejnice zarovnané.



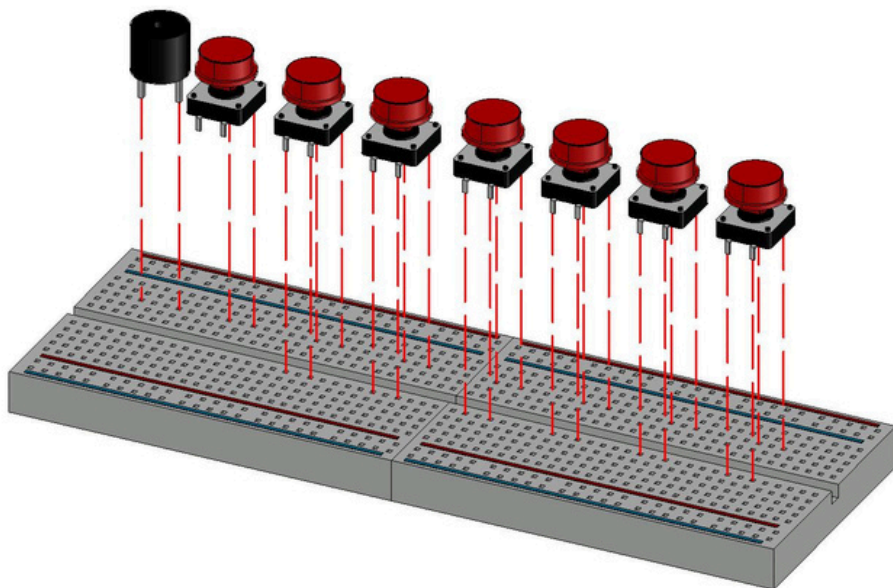
Krok 4 Instalace tlačítka a bzučáku

Seznam dílů

Tlačítko*7

Bzučák*1

Ilustrace



Poznámka: Tlačítka je nutné vkládat přesně na místa znázorněná na obrázku.

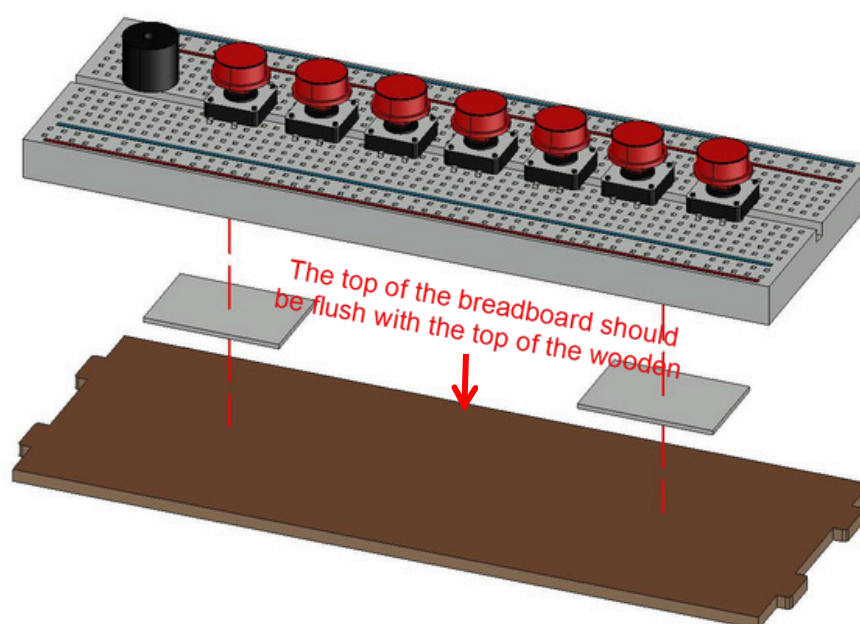
Krok 5. Zajistěte nepájivé pole

Seznam dílů

Oboustranná páska*2

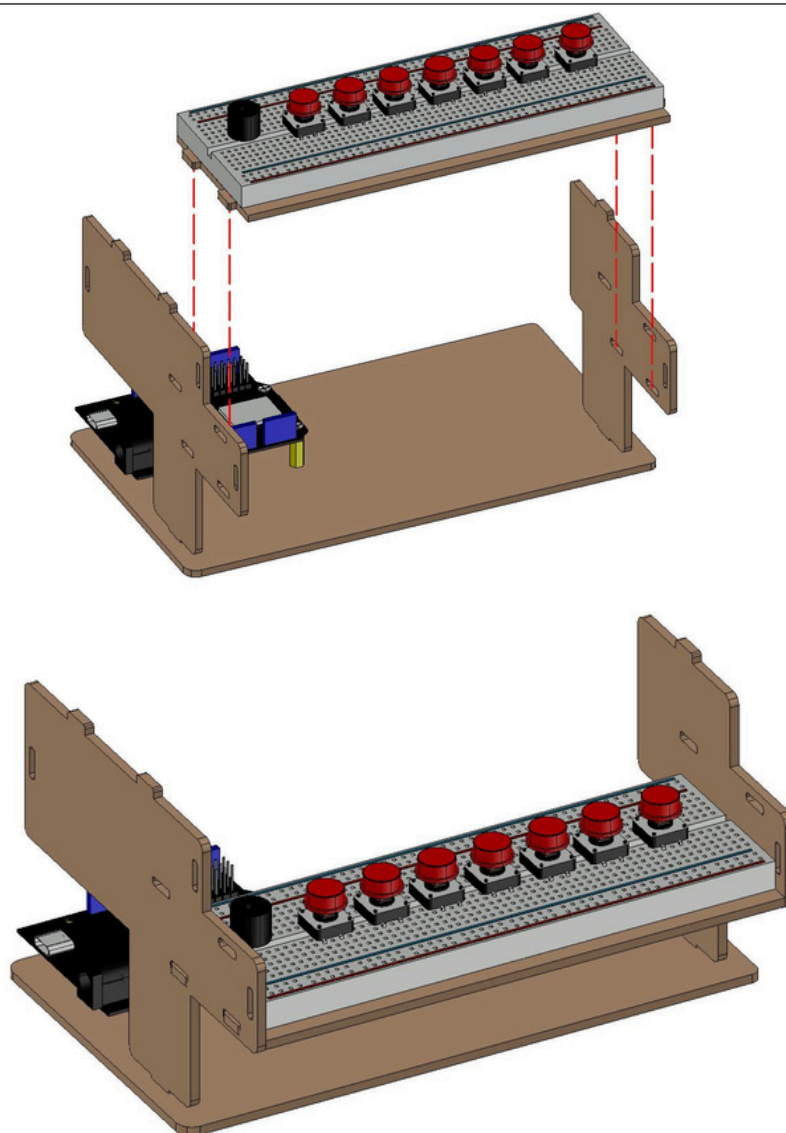
Šití

Ilustrace



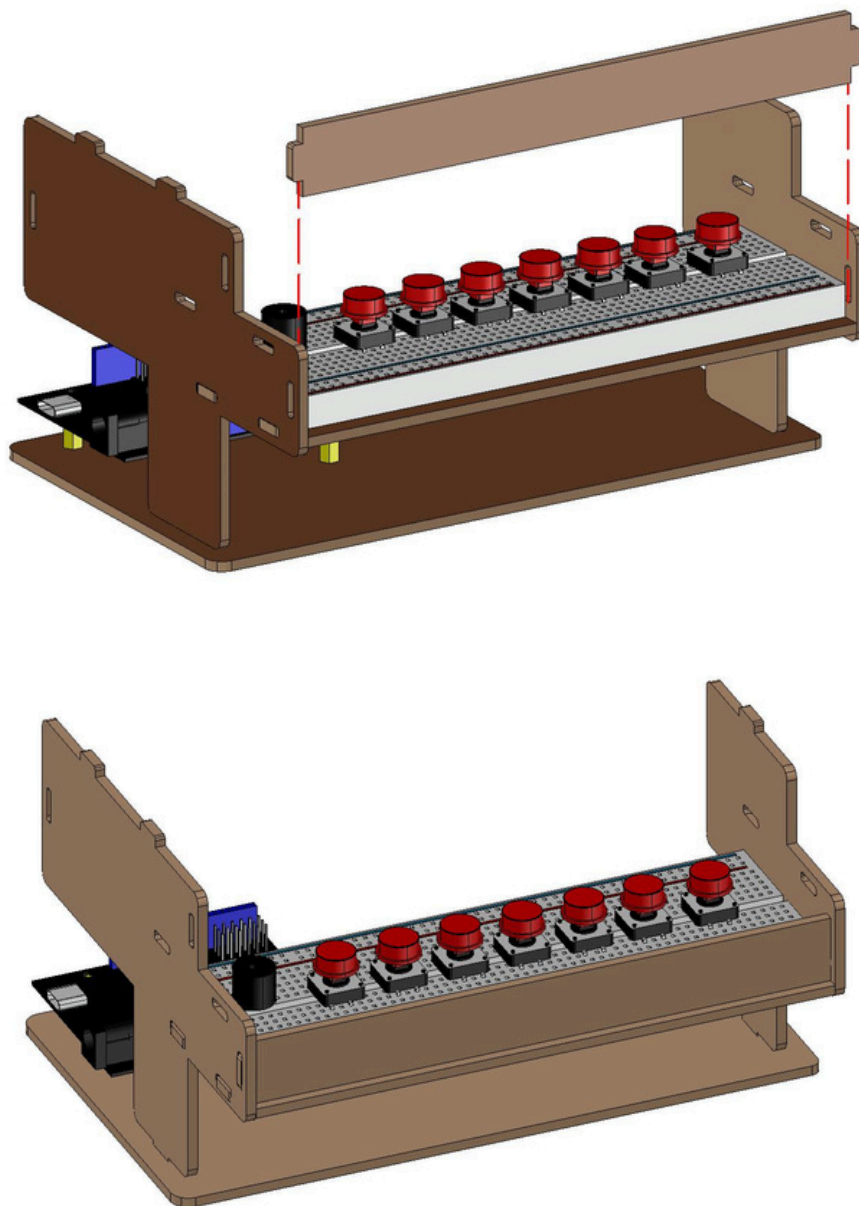
Krok 6. Instalace podpěry tlačítka

Ilustrace



Krok 7. Instalace předního panelu piana

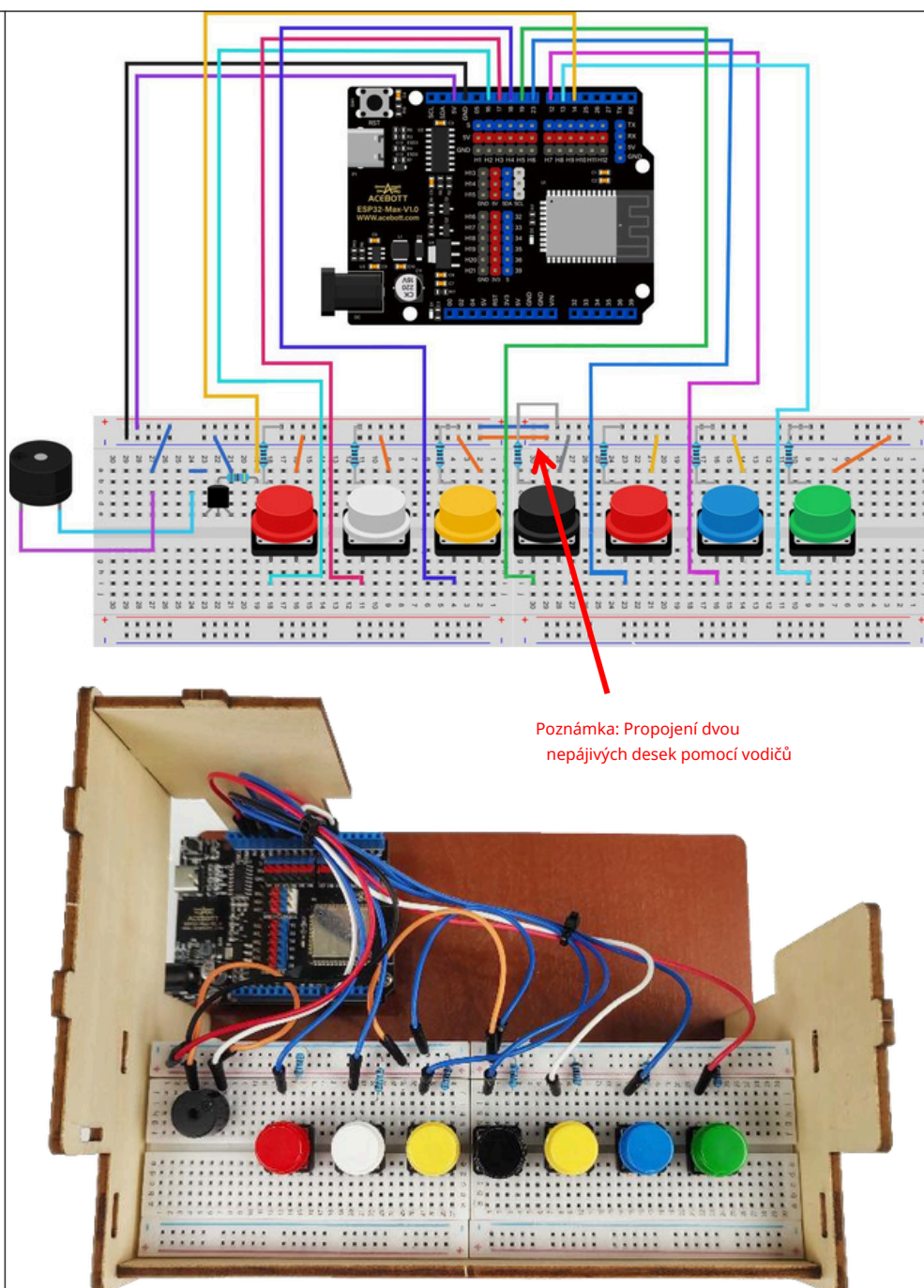
Ilustrace



Poznámka: Než budete pokračovat, zapojte vodiče podle schématu zapojení uvedeného níže.

Krok 8: Zapojení

Ilustrace



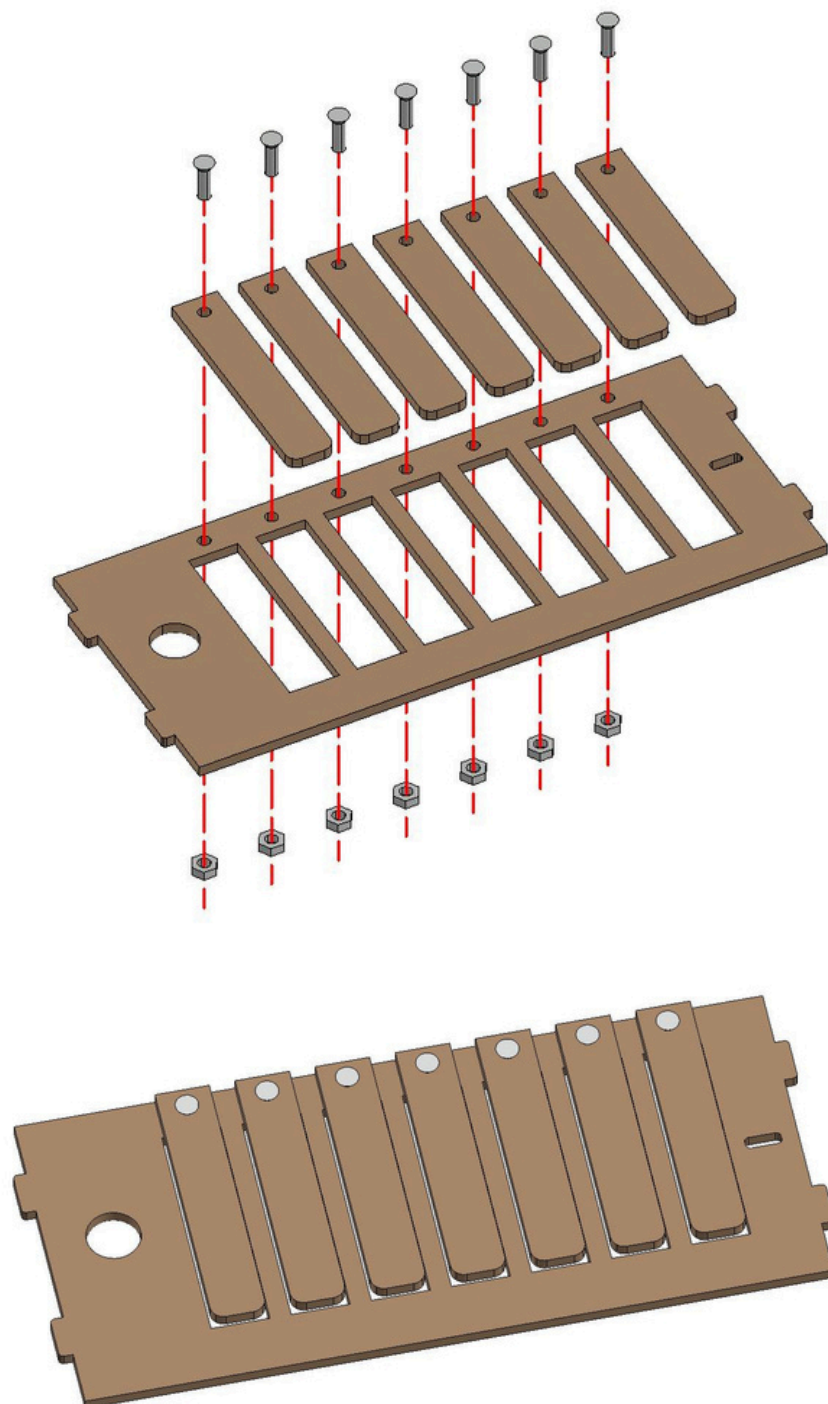
Krok 9 Instalace klíčů

Seznam dílů

M3*6mm plochá hlava
Šroub*7

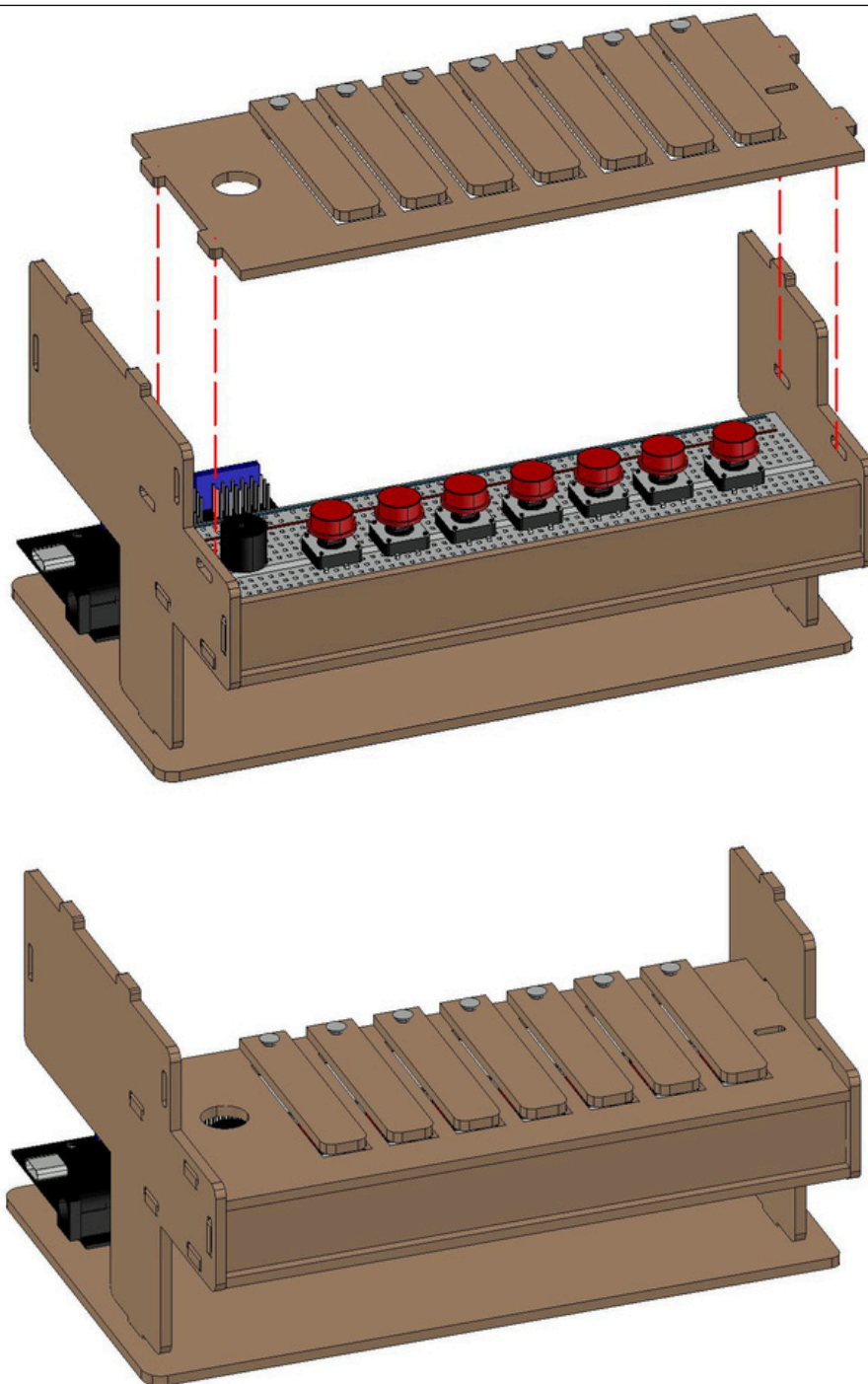
Matice M3*7

Ilustrace



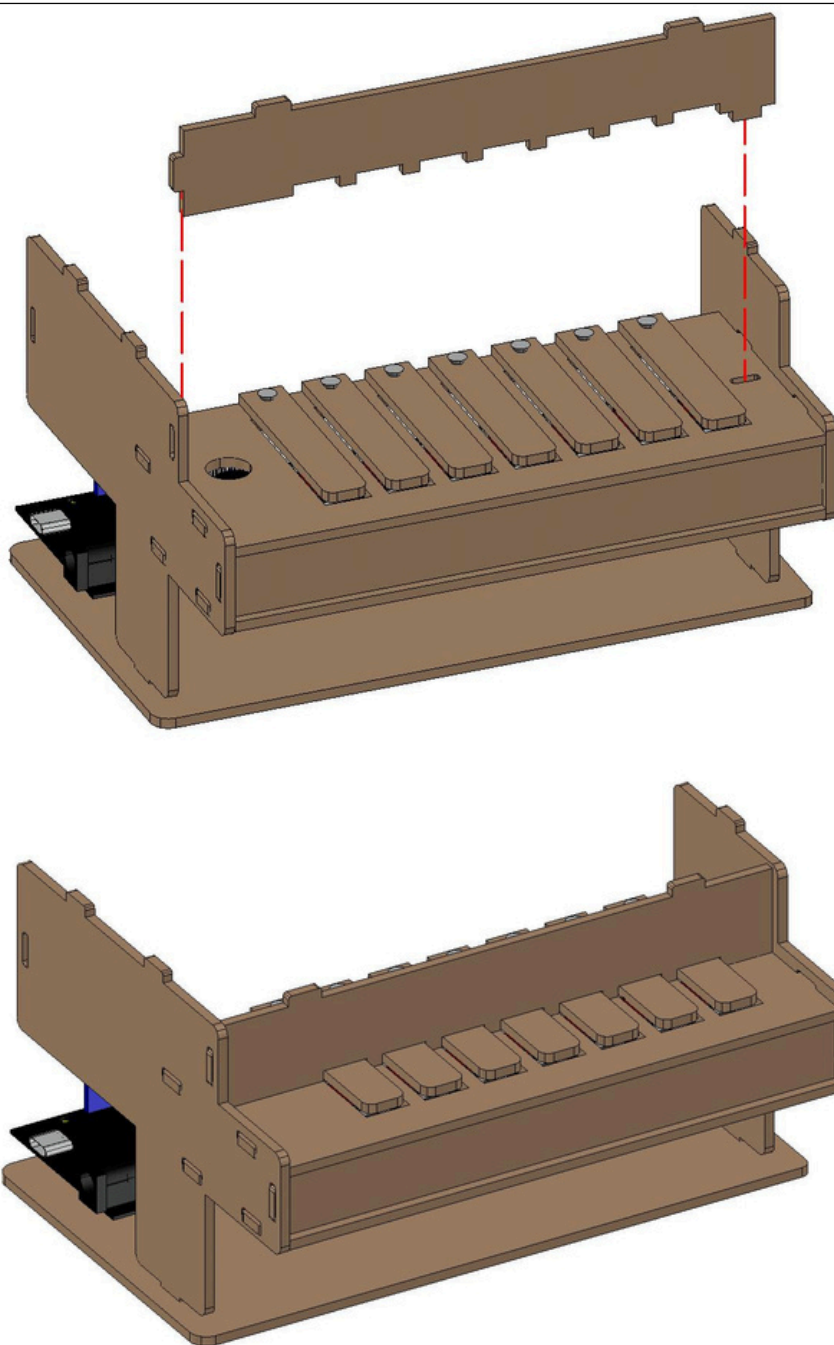
Krok 10 Instalace klávesnice

Ilustrace



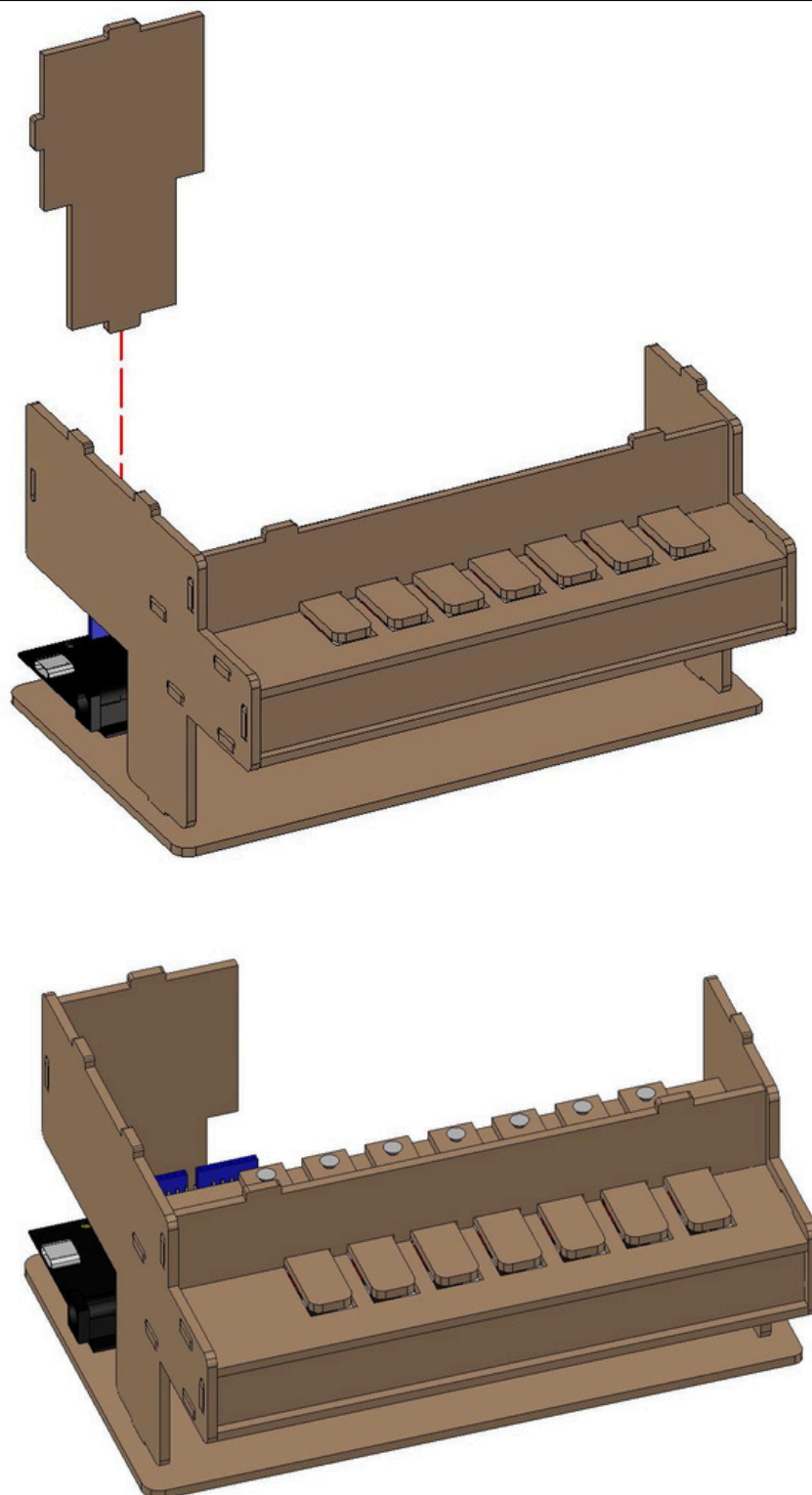
Krok 11 Instalace zvedáku klávesnice

Ilustrace



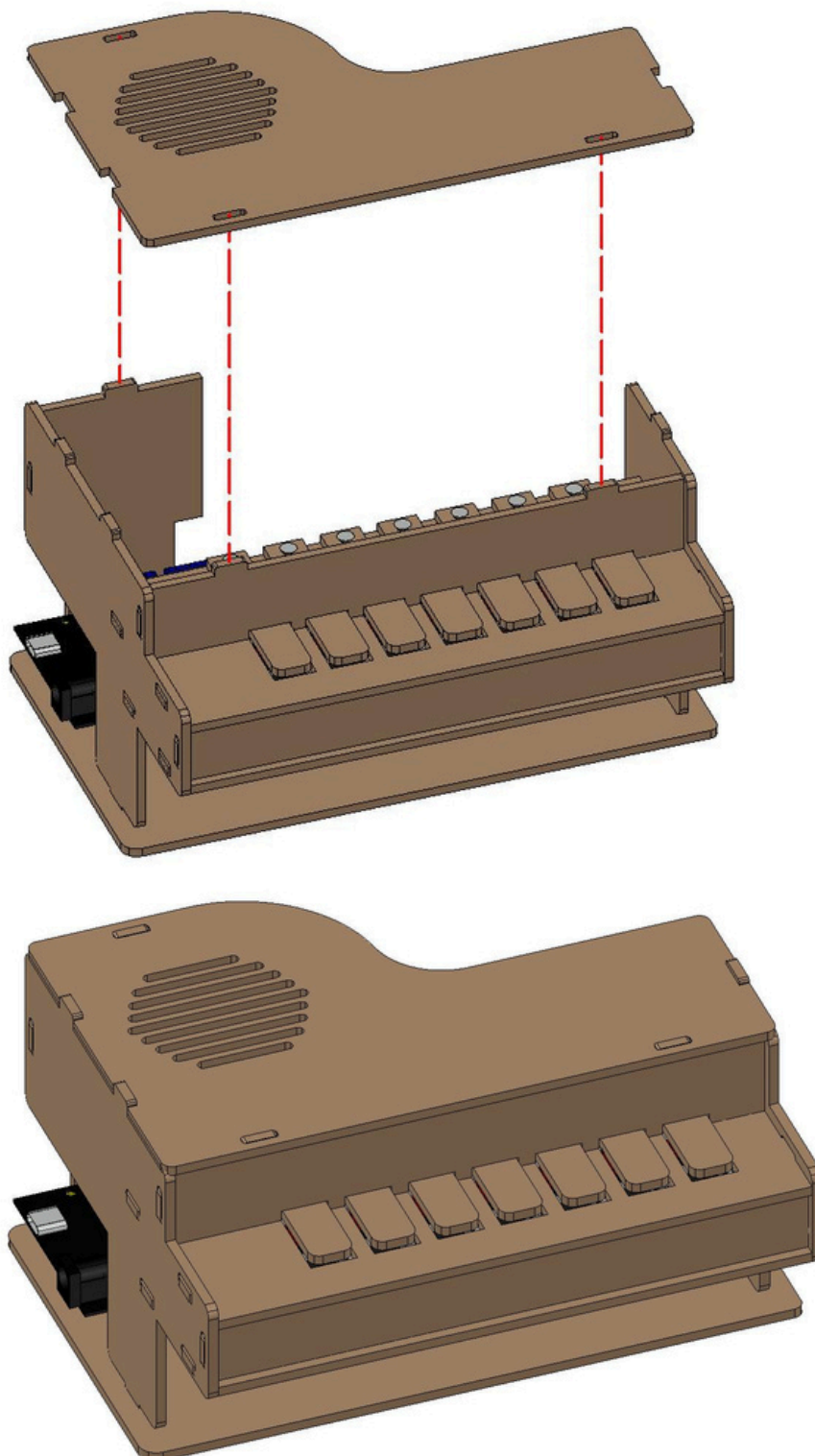
Krok 12 Nainstalujte zadní panel

Ilustrace

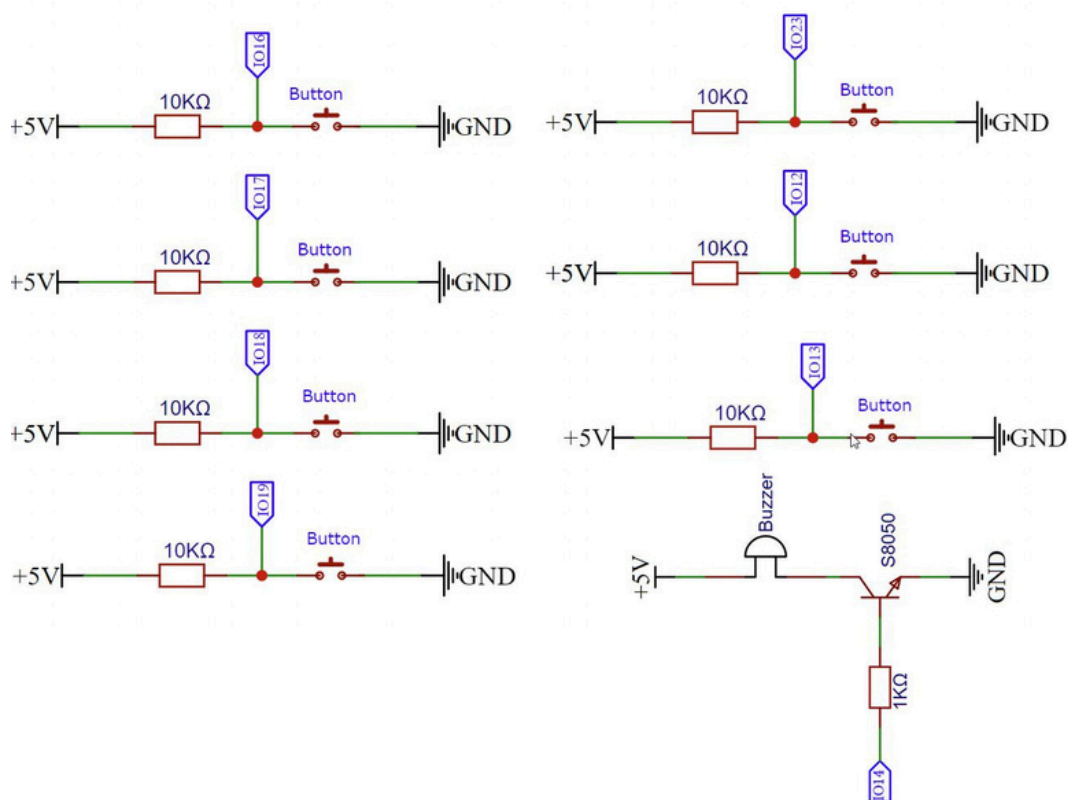


Krok 13 Nainstalujte kryt klávesnice

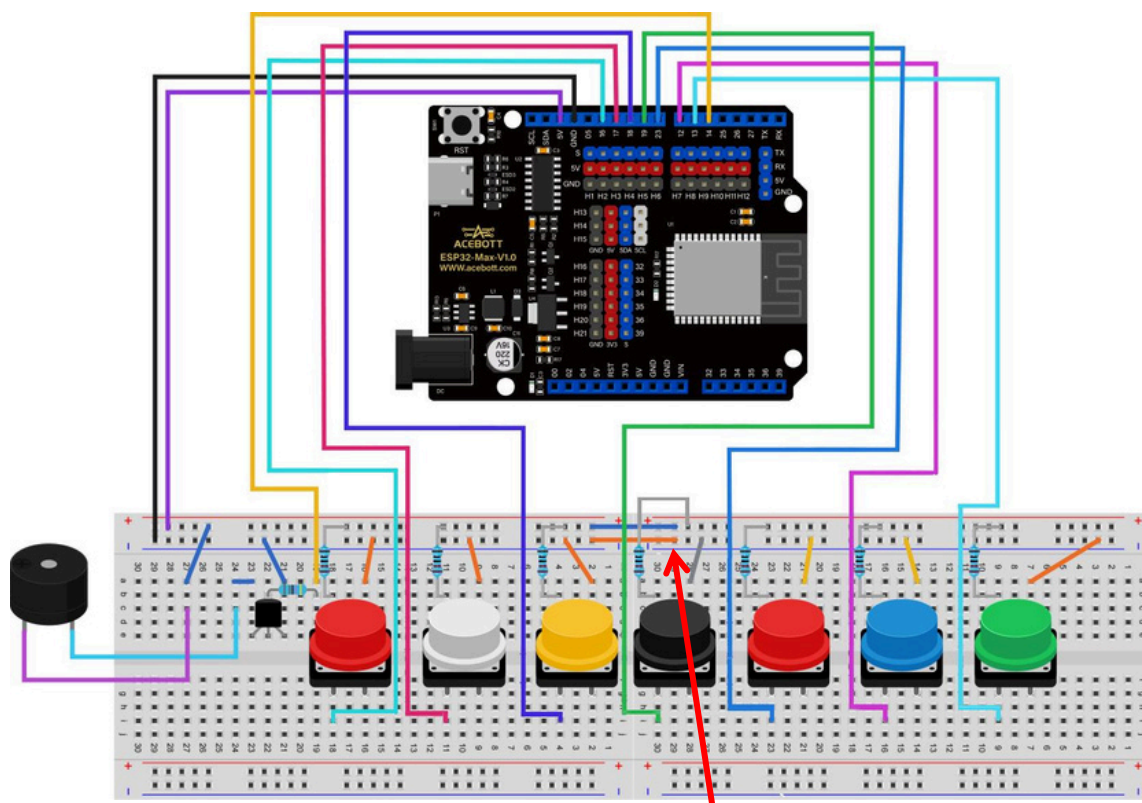
Ilustrace



4. Schéma zapojení



5. Hardwarová připojení



Poznámka: Propojení dvou
nepájivých desek pomocí vodičů

6. Referenční postupy

Otevřete "[Electronic_Piano.ino](#)" soubor v adresáři

"English\Arduino(Experienced Learner)\2.Code\lesson 38 Electronic keyboard\Electronic_Keyboard" nebo zkopírujte následující kód do Arduino IDE. Po výběru vývojové desky (ESP32 Dev Module) a příslušného portu

podle dříve naučených kroků klikněte  nahrát program.

```
#define BUZZER_PIN 14

int keys[] = {16, 17, 18, 19, 23, 12, 13}; // Key input pins
int notes[] = {523, 587, 659, 698, 784, 880, 988}; // Frequencies for C4,
D4, E4, F4, G4, A4, B4

void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(keys[i], INPUT_PULLUP); // Set each key pin as input with internal
    pull-up resistor
  }
}

void loop() {
  for (int i = 0; i < 7; i++) {
    if (digitalRead(keys[i]) == LOW) { // Check if key is pressed (active low)
      tone(BUZZER_PIN, notes[i]); // Play the corresponding tone frequency on
      buzzer pin
      while(digitalRead(keys[i]) == LOW); // Wait here until key is released
      noTone(BUZZER_PIN); // Stop playing the tone
    }
  }
}
```

Po úspěšném nahrání programu stiskněte klávesy pro přehrání odpovídajících not a bzučák vydá odpovídající zvuk.

