UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Adam Dvořák

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Využití jednodeskových počítačů pro výuku programování na středních školách

Bakalářská práce

2024 Adam Dvořák

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Adam Dvořák**
Osobní číslo:	**I21143**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Využití jednodeskových počítačů pro výuku programování na středních školách**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Práce se zabývá využitím jednodeskových počítačů,především pak počítače Arduino Uno R3, při výuce algoritmizace, programovánía robotiky pro studenty středních průmyslových škol. V práci je zmíněno jak jsou v současné době takovéto počítače na středních školách využívány.
V teoretické části je přesněji popsán hardware tohoto počítače a část kompatibilních periférií.
V praktické části jsou ukázany základy jazyka Wiring, praktické příklady zaměřené na výuku a procvičení programování,které jsou vhodné pro stundenty středních škol.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

SELECKÝ, Matúš. Arduino: Uživatelská příručka. Computer Press, 2016. ISBN 978-80-251-4840-2.
NOVÁK, Milan a PECH, Jiří. Robotika pro střední školy: Programujeme Arduino. Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, 2020. ISBN 978-80-7394-786-6.
OBDRŽÁLEK, David; KOPPENSTEINER, Gottfried; MERDAN, Munir; BALOGH, Richard a LEPUSCHITZ, Wilfried. Robotics in Education Methodologies and Technologies. Springer International Publishing, 2021. ISBN 978-3-030-67410-6.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem Využití jednodeskových počítačů pro výuku programování na středních školách jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 01. 04. 2024

Adam Dvořák v.r.

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu práce, Ing. Janu Panušovi, Ph.D. za jeho cenné rady, čas a pomoc během psaní této práce. Jeho podpora byla pro mě neocenitelná.

ANOTACE

Práce se zabývá využitím jednodeskových počítačů, především pak počítače Arduino Uno R3, při výuce algoritmizace, programování a robotiky pro studenty středních průmyslových škol. V práci je zmíněno, jak jsou v současné době takovéto počítače na středních školách využívány. V teoretické části je přesněji popsán hardware tohoto počítače a část kompatibilních periférií. V praktické části jsou ukázány základy jazyka Wiring, praktické příklady zaměřené na výuku a procvičení programování, které jsou vhodné pro studenty středních škol.

KLÍČOVÁ SLOVA

Programování, výuka programování, Arduino, Arduino Uno R3, Wiring, jednodeskové počítače, vzdělávání ve středních školách

TITLE

Utilization of single-board computers for teaching programming in secondary schools

ANNOTATION

This work focuses on the use of single-board computers, particularly the Arduino Uno R3, in teaching algorithms, programming, and robotics to students of vocational secondary schools. It discusses how such computers are currently utilized in secondary schools. The theoretical part describes in detail the hardware of this computer and a range of compatible peripherals. The practical section presents the basics of the Wiring language, practical examples aimed at teaching and practicing programming, which are suitable for secondary school students.

KEYWORDS

programming, programming education, Arduino, Arduino Uno R3, Wiring, single-board computers, secondary school education

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

ČR	Česká republika
DC	direct current, stejnosměrný proud
EEPROM	electrically erasable programmable read-only memory, elektricky vymazatelná paměť pouze pro čtení
HW	hardware
IDE	integrated development environment, vývojové prostředí
IMU	inertial meansurement unit, inercionální měřící jednotka
IO	input-output, vstupně-výstupní
IoT	internet of things, internet věcí
IT	informační technologie
LED	light-emitting diode, světlo-emitující dioda
PC	personal computer, osobní počítač
PWM	pulse width modulation, pulzně šířková modulace
RFID	radio frequency identification, identifikace na rádiové frekvenci
RGB	red – green – blue, červená – zelená - modrá
RTC	real time clock, hodiny reálného času
SBC	single board computer, jednodeskový počítač
SD	Secure Digital
SMD	surface mount device, součástka určená pro povrchovou montáž
SŠ	střední škola
UART	universal asynchronous receiver-transmitter, univerzální asynchronní přijímač-vysílač
USB	universal serial bus, univerzální sériová sběrnice

TERMINOLOGIE

case senstive	konvence pro pojmenování prvků v programu, kde se rozlišují velká a malá písmena (např. v jazyce Wiring nebude název funkce „Moje_Funkce“, ale lepší bude označení „mojeFunkce“)
combobox	prvek grafického rozhraní, který slouží k výběru položky z předdefinovaného seznamu (rozbalitelný seznam)
pinout	plánek pinů, jejich vlastností, propojení a účelu
scope	rozsah prvku v programovacím jazyku
shield	připojitelný modul poskytující dodatečnou funkcionalitu

ÚVOD

Cílem této bakalářské práce je bližší seznámení s jednodeskovými počítači, jejich hardwarem, softwarem a analyzovat možnosti jejich programování. Po seznámení s těmito informacemi následuje část s praktickými úlohami, jež je možné využít pro výuku například programování mikroprocesorů, algoritmizace anebo robotiky.

První část se zabývá rolí jednodeskových počítačů při výuce programování na středních průmyslových školách. Jsou zde reprezentovány konkrétní případy, kde se již jednodeskové počítače při výuce používají a způsob, jak se výrobci těchto počítačů staví k využívání jejich produktů jako výukové pomůcky.

Druhá část popisuje nejčastěji pro výuku využívaný programovatelný počítač a tím je Arduino. Dále v této části naleznete verze tohoto zařízení, které lze zakoupit a k čemu jsou vhodné. Následuje popis periférií. V něm jsou popsány jak snímací, tak i akční členy.

Třetí část se zaměřuje přímo na způsob, jak programovat počítač Arduino, konkrétněji pak na jazyk Wiring. Tato část ukazuje verze tohoto jazyku, základní syntaxi a strukturu. Kromě toho je zde popsáno i vývojové prostředí přímo určené k programování tohoto jednodeskového jednoduchého počítače.

Poslední část ukazuje praktické příklady na procvičení logického myšlení, algoritmizace, naučení základní látky týkající se programování mikroprocesorů, robotů a automatizovaných systémů. Nachází se zde jak jednodušší příklady, tak i pokročilejší. Proto lze tyto úlohy použít pro výuku odborných předmětů na některých středních průmyslových školách.

1 Jednodeskové počítače v roli výukových pomůcek

Jednodeskové počítače mohou být skvělou pomůckou při výuce programování, algoritmizace, mikroprocesorové techniky anebo robotiky. Většina známých a rozšířených jednodeskových počítačů totiž pracuje s IO periferiemi. Díky tomu lze žáky upoutat tím, že práce s nimi neznamená jen vytvoření kusu kódu, který něco provede uvnitř osobního počítače. Naopak. Výstupem práce je pak většinou nějaký na pohled viditelný, slyšitelný anebo hmatatelný projev a výsledek. To je to, v čem tkví ta výhoda, proč se jedná o perfektní výukové pomůcky. Sety, jež obsahují zmíněný jednodeskový počítač neboli SBC, a zároveň i zmíněné periferie a propojovací prostředky se nazývají stavebnice anebo vývojové kity. Tyto stavebnice představují dostupné možnosti pro rozvoj všech aspektů spojených s programováním a elektrotechnickým vývojem. V současnosti je již podobných stavebnic využíváno jak ve školství, tak v zájmových kroužcích anebo v rámci samostudia. To potvrzují i výsledky ankety Cena bastlířů, kde v roce 2018 vyhrál kategorii nejlepší produkt právě SBC Arduino Uno. Výsledky této ankety jsou k nalezení pouze na Wikipedii [8], ale sami autoři ankety na svojí facebookové stránce na ni odkazují [9].

1.1 Výuka programování na středních školách

Výuku programování na středních školách je nutno rozřadit do dvou směrů, zejména podle toho, jak jsou jednotlivé obory zkonstruovány. Jeden je vývoj webových, desktopových a mobilních aplikací, což se týká oborů jako hlavně IT. Pak se jedná o programování například v jazycích jako je Java, C#, Kotlin atd. Druhý směr je jakýsi praktický a týká se oborů jako například automatizace apod. Zde je cílem vytvářet praktické řídící systémy. V pozdějších fázích takového studia lze žáky seznámit s PLC a jazyky jako je LD, ST anebo FBD. V ranné fázi lze například vyučovat za pomocí stavebnice Lego Mindstorms. Ta je však využívána i na některých druhých stupních základních škol pokročilejšími žáky anebo žáky na některých středních školách pro demonstraci fyziky [10]. Ti ji mohou programovat v grafickém rozhraní. Jedná se o abstraktnější a jednodušší stavebnici. Alternativou mezi stavebnicí jako právě Lego Mindstorms a mezi PLC jsou jednodeskové počítače. Mezi nejpoužívanější a nejznámější patří již zmíněné Arduino Uno, dále Raspberry Pi anebo Odroid. V současné době se již SBC na některých školách využívají. Není tomu tak jenom v ČR, ale jedná se o světový fenomén. Programování pomocí jazyka Wiring v Arduino IDE si procvičují i studenti v Indonésii [11].

1.2 Využití SBC při výuce

Využití jednodeskových počítačů při výuce nemusí být pouze u předmětů týkajících se programování. Podobně jako u robotické stavebnice od firmy Lego, jak je již zmíněno v předchozí části, lze SBC využít například při hodinách fyziky pro demonstraci. To samozřejmě nemusí být ani na průmyslové střední škole. Je ale jasné, že pro výuku programování lze nalézt v SBC mnohem větší potenciál.

Na elektrotechnických středních školách by byl přinos veliký. Z praxe by se dal uvést příklad toho, pro jaké všechny předměty, by se dalo Arduino Uno anebo jiný jednodeskový počítač použít. Například na SPŠE Pardubice se vyučují tyto předměty: průmyslová automatizace, praxe, programování, mikroprocesorová technika, robotika, atd [12]. Pro každý z nich by se SBC dal alespoň částečně využít a blíže tak žákům přiblížit danou problematiku. V prostředí průmyslové automatizace by se SBC dal využít pro tvorbu řídích a monitorovacích systémů, u předmětu praxe se dá využít pro realizaci vlastních projektů a získání tak praktické zkušenosti jak s hardwarem, tak i se softwarem. Pro výuku mikroprocesorové techniky je SBC jako Arduino Uno, Raspberry Pi anebo jiný perfektní platformou pro pochopení základních principů a architektur mikroprocesorů. Při robotice by se dal jednodeskový počítač využít jako mozek robota a sloužil by tak pro jeho řízení. Při elektrotechnice se dá využít jako například testovací prvek. Lze s ním snímat určitou veličinu a například ji přehledně zobrazovat na displeji anebo data ukládat do souboru.

Jedná se tedy o velice flexibilní nástroj, jež přináší mnohé výhody. Jednou z nich je i zvýšené zapojení studentů, jež SBC nabízejí. Jedná se o praktický způsob výuky, což může způsobit pozdvižení zájmu žáků o danou problematiku. Navíc by se díky mnohým rozšířením a modulům studenti mohli seznámit s technologiemi z praxe a připravit se více na svoje budoucí povolání. Jedná se pro příklad o ultrazvukové snímání vzdálenosti anebo o práci s RFID.

1.3 Dostupnost pro výuku

Dostupnost jednodeskových počítačů včetně rozšiřujících kitů je poměrně dobrá. Z finančního hlediska lze konstatovat, že cena je více než vstřícná. V porovnání oproti klasickému PLC anebo srovnatelnějšímu Legu Mindstorm se lze dostat na mnohem menší cifru. Navíc k tomu například Arduino Uno je nelicencovaná platforma a její volné šíření způsobilo výrobu spoustu klonů této stavebnice v Číně. V důsledku toho lze takovouto stavebnici pořídit v zahraničí až čtyřikrát levněji než u nás.

Dalším z faktorů je rozsah výukových dokumentů a podpora ze strany výrobců. Ta je také na velice vysoké úrovni. Zůstaneme-li přímo u SBC Arduino, tak na oficiálních stránkách *www.arduino.cc* lze naleznout sekci education. Zde je na výběr ze tří podsekcí. Po vybrání podsekce je možné vybírat konkrétní druh výukové stavebnice. V sekci docs se nachází oficiální dokumentace ke všem vydaným druhům Arduino desek a shieldů. Navíc jako veliký bonus je zde i ke každé vývojové desce řada tutoriálů na procvičení a pochopení technologií, jež lze využívat spolu s deskou. Řeč byla sice o Arduinu, ale velmi podobně jsou na tom i jiní výrobci. Navíc další tutoriály, dokumentace a jiné materiály jsou k nalezení v různých fórech, bastlířských komunitách anebo ve webových magazínech.

2 Arduino Uno

Arduino je jeden z nejznámějších jednodeskových počítačů, jež jsou určeny pro výuku a seberozvoj v oblasti mikroprocesorové techniky, robotiky, automatizace a řídící techniky. Historie tohoto jednodeskového počítače sahá až do roku 2003. V této době italský student jménem Hernando Barragán pracoval na diplomové práci. Jejím cílem bylo zjednodušení práce pro lidi, jež vyvíjejí elektroniku. Mezi základní požadavky patřilo vytvoření jednoduchého IDE, jednoduchý programovací jazyk pro mikroprocesory, bootloader pro nahrávání programů a další. Práci vedli Massimo Banzi a Casey Reas. Výsledkem bylo vytvoření několika prototypů, z nichž jeden nesl název Wiring. V letech 2003 až 2005 se toto zařízení dokonce prodávalo. V projektu byl též zapojen David Mellis a byl to právě on, kdo se spolu s Banzim od projektu odtrhli a dali tak vzniknout vlastnímu konceptu, tentokrát už s názvem Arduino. Smutným faktem je to, že i když projekt vychází z projektu Wiring, samotný Hernando k Arduinu nikdy přizván nebyl. Později došlo i ke sporům uvnitř Arduina, a to se rozdělilo na Arduino SRL a Arduino LLC. Projekt ale drží dodnes pospolu pouze si společnosti mezi sebou rozdělili globální trh. Posledním faktem z historie, který je zde zmíněn je to, že název Wiring se zachoval a je po něm pojmenován jazyk, ve kterém lze Arduino desky programovat.[4]

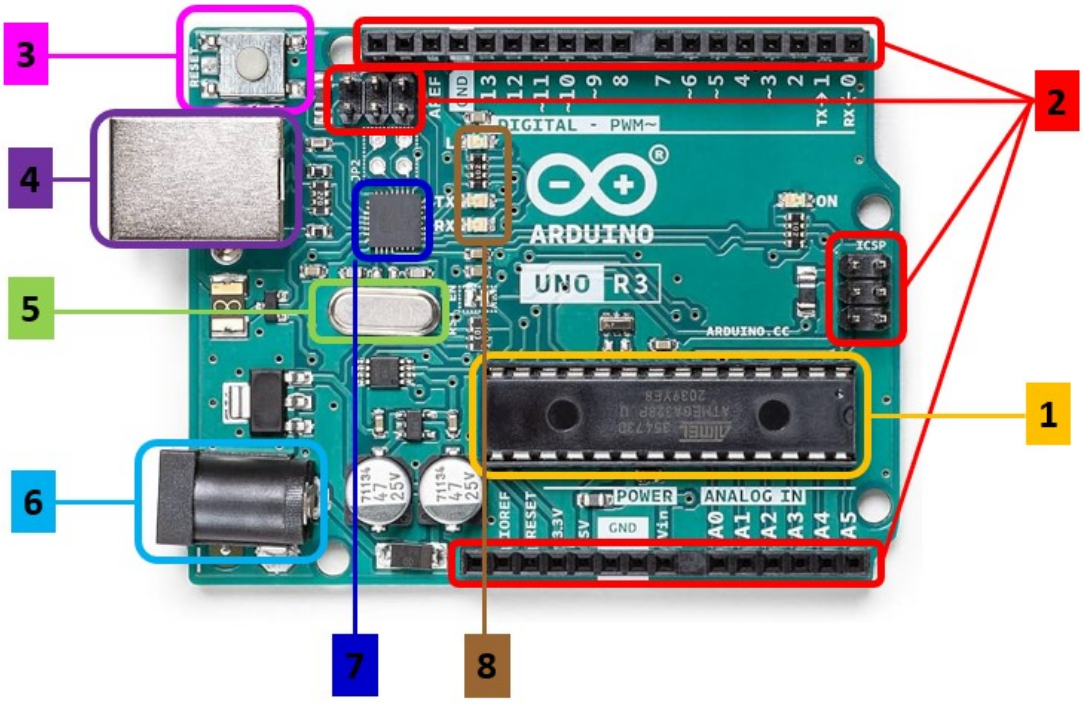
2.1 Platforma Arduino a druhy počítačů

Jak již bylo zmíněné jedná se o opensource platformu, která nabízí jednoduchou počítačovou desku a vlastní software. Jedná se o nízkonákladový mikrokontrolér pomocí jehož se dají tvořit rozsáhlé a zajímavé projekty. Postupným vývojem této platformy vzniklo několik finálních a prodejných verzí počítačů Arduino. Na popis desky, využitelných technologií a na verze tohoto počítače se zaměřují následující kapitoly.

2.1.1 Arduino Uno R3

Nejrozšířenější deskou od firmy Arduino je deska Arduino Uno R3. Deska se ze základu skládá z mikrokontroléru, krystalu, napájecí části a z části, která obstarává propojení přes sériové rozhraní. Osazeným mikrokontrolérem je Atmel ATMega 328p. Jedná se o 8-bitový AVR mikrokontrolér. Disponuje pamětí FLASH o velikosti 32kB, pamětí EEPROM o velikosti 1kB a SRAM o velikosti 2kB. Jeho IO vývody jsou vyvedeny do dutinkových a kolíčkových lišt, jež jsou kompatibilní s kabelovými svazky s konektory dupont. Dále deska obsahuje ATMega 16U2, což je čip, jež umožňuje komunikaci mezi Arduinem a PC pomocí sériového rozhraní USB. Tento čip převádí data z USB rozhraní na sériové rozhraní Arduina.

Pro připojení USB rozhraní se zde nachází USB konektor typu B a může sloužit i jako napájecí konektor. Kromě toho lze počítač napájet i pomocí souosého konektoru pomocí DC napětí v hodnotách 7 – 12V, přičemž ideální je konstantních 9V [3]. Dále zde nalezneme tlačítko RESET. Po stisku dojde k uvedení mikrokontroléru do výchozí stavu a program, jež byl nahrán do paměti se začne vykonávat od začátku. Další z důležitých součástí je krystal. Jeho frekvence je 16MHz a je součástí tzv. oscilátoru, jehož úkolem je dodávat hodinový signál pro mikrokontrolér. Tímto signálem se vlastně určuje, jakou rychlostí se budou vykonávat jednotlivé instrukce. Poslední ze zmíněných komponent jsou SMD LED. Na desce vyobrazené níže jsou zvýrazněny tři. Jedna z nich může docela ulehčit práci pro začátečníky, protože je napojena pin 13 a lze s ní trénovat základní programové konstrukce. Další dvě ze zvýrazněných jsou označeny Rx a Tx (Receive, Transmit). Jedná se o indikaci přijímání nebo odesílání dat přes sériové rozhraní.



Obrázek 1 Deska Arduino Uno R3 [13]

1) Mikrokontrolér Atmel ATMega 328p
2) IO vstupy a výstupy
3) Resetovací tlačítko
4) USB-B konektor pro napájení a přenos dat

5) 16Mhz krystal
6) Souosý konektor pro napájení
7) Čip Atmel ATMega 16U2
8) Led pro indikaci přijmu a odesílání dat / LED pinu D13

Pro vysvětlení podporovaných technologií a dostupných možností je vhodné si představit dostupné vstupy a výstupy. Následující 3 kapitoly se proto zabývají těmito záležitostmi u Arduino Uno R3.

2.1.1.1 Vstupy a Výstupy

První věcí, kterou je nutné se zabývat, je napájení. Mezi základní vstupy patří určitě napájecí souosý konektor. Ten umožňuje napájení pomocí stejnosměrných 7 až 12V, jak již bylo zmíněno v předchozí kapitole. Další způsob, je napájet desku pomocí USB konektoru, a tedy i pomocí 5V. Tento vstup je ale zejména určen pro sériovou komunikace a přes snímaných dat anebo nahrávání programu do paměti mikroprocesoru. Poslední validní cestou, jak lze napájet desku je pomocí externího zdroje k pinům Vin a GND pomocí 7-12V DC. Desku lze také napájet pomocí pinu 5V a 3,3V, ale nedoporučuje se to, protože se pak obcházejí ochranné prvky desky a může dojít ke zničení. Tyto piny mají sloužit jen jako napájení pro další periférie, součástky anebo shieldy.

Další pin, který souvisí s napájením je IOREF. Ten poskytuje referenční napětí. Jedná se o napětí, jež je dodáváno na IO piny. Podobné je to s AREF. To dodává referenční napětí pro analogové vstupy. [15]

Mezi vstupní prvky lze považovat i tlačítko RESET. Jeho funkcionalita je již vysvětlena v předchozí kapitole, ale z názvu je celkem zřejmé o co jde.

Arduino Uno disponuje šesti analogovými vstupy. Ty přijímají 0 až 5V. Toto napětí reprezentuje rozsah hodnot 0 – 1023 a tyto hodnoty pak, lze využívat v programu. Jedná se například o stav senzorů.

Dále je zde 13 digitálních OI pinů. To znamená, že mohou sloužit jako vstup i výstup. Pracují pouze s dvěmi hodnotami a to je logická nula (úroveň LOW) a logickou jedna (úroveň HIGH). Lze pomocí jich například rozsvěcet LED, spínat relé anebo přijímat informaci o stisku tlačítka. Pro ulehčení začínajícím vývojářům anebo zkrátka pro využití jako způsob indikace je na pin D13 napojena SMD LED. U pinů D0 a D1 si lze povšimnout označení Tx a

Rx. Jedná se o piny využitelné při asynchronní sériové komunikaci UART mezi Arduinem a jinými počítači, deskami anebo periferiemi.

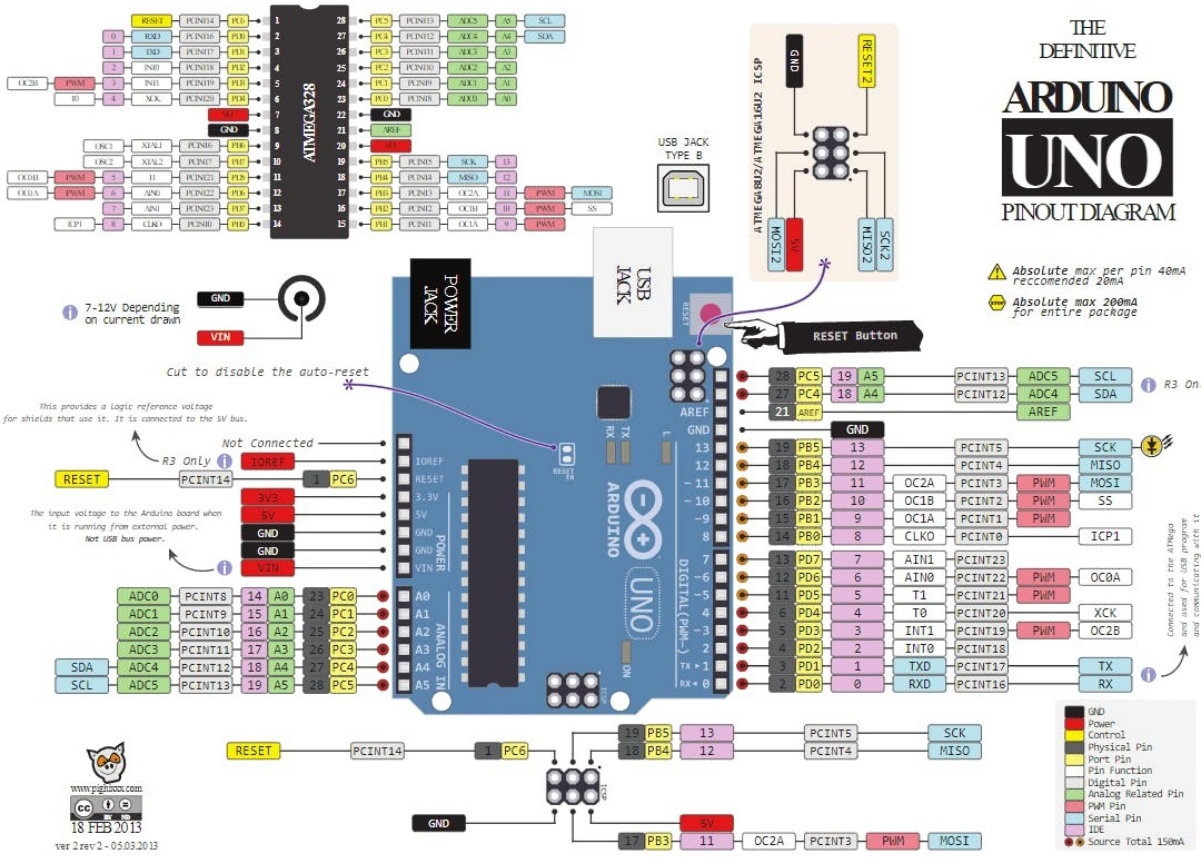
Poslední z nejpodstatnějších informací k digitálním pinům je to, že některé podporují PWM, což je vlastně jakási simulace analogového výstupu. Arduino totiž jinak nedisponuje přímo analogovým výstupem, a proto je zde tato technologie. Není však dostupná na všech digitálních pinech. Jedná se jen o ty jež u sebe mají označení vlnovky.

Dále piny 10-13 jsou piny, které podporují SPI komunikaci.

Piny 2 a 3 jsou schopny zachytit přerušení.

Dále jsou tu dva piny označené jako SDA a SCL. Jedná se o piny, jež se využívají při komunikaci po I2C sběrnici. Slouží tak pro usnadnění připojení periférií a k úspoře IO pinů. Podpora I2C je také zahrnuta na pinech A4 a A5.

V poslední řadě na desce lze naleznout 2 ICSP konektory (3 x 2 kolíčky). Umožňují programování ATMega 328 například USB programátorem a jeden podporuje opět rozhraní SPI.



Obrázek 2 Pinout desky Arduino Uno R3 [16]

2.1.2 Podporované technologie a principy

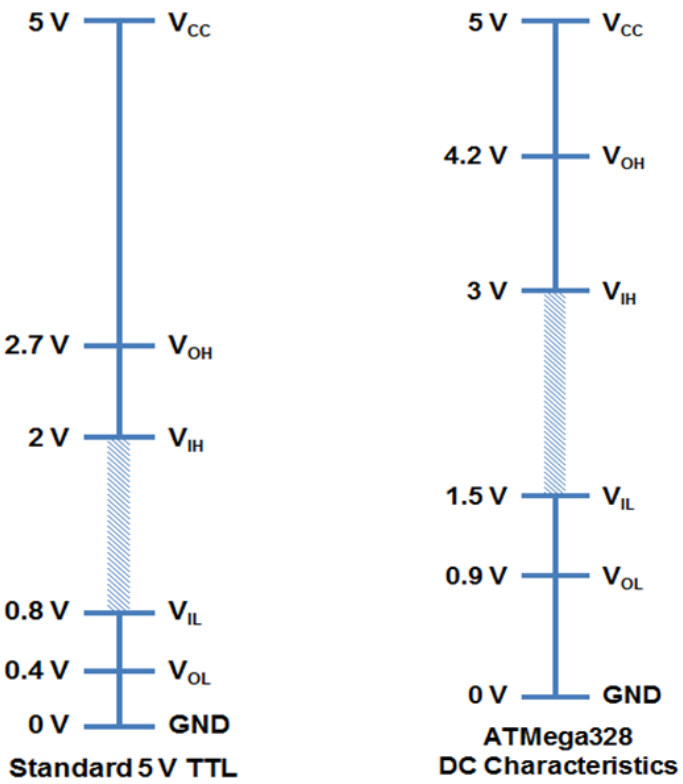
Arduino umožnuje pracovat s různými technologiemi, sběrnicemi a funguje s jistými technologickými aspekty. Ty jsou vysvětleny v této kapitole.

2.1.2.1 Analogový a digitální signál, Logické úrovně

V principu se rozlišují dva typy signálů. Každý signál má svoji napěťovou úroveň, které může dosahovat. V případě Arduina Uno je to 0 – 5 V.

Obecná definice analogového signálu by se dala formulovat takto: Analogový signál je sestaven z nepřetržitého průběhu funkce, která se mění v čase, a může být přenášen různými prostředky. Na rozdíl od digitálního signálu není omezen na diskrétní časové intervaly. [17] V praxi to vypadá takto: Analogový signál dosahuje několika úrovní. V případě Arduina je to konkrétně 1024 (0 až 1023), kdy je 5 V rozděleno a určité napětí odpovídá určité úrovni.

Digitální signál je signál, který je reprezentován diskrétní hodnotou. Ta je vyjádřena v binární podobě. Jedná se tedy o sekvenci nul a jedniček. Místo 0 a 1 lze používat označení LOW a HIGH. Arduino využívá něco jako TTL logiku, kde nulu a jedničku reprezentuje vždy určité napětí. Některé desky Arduino využívají 5V logiku a některé 3,3V. Níže je porovnání TTL a logiky Arduino Uno, respektive mikroprocesoru ATMega 328.



Obrázek 3 Standartní TTL vs Arduino napěťová logika [18]

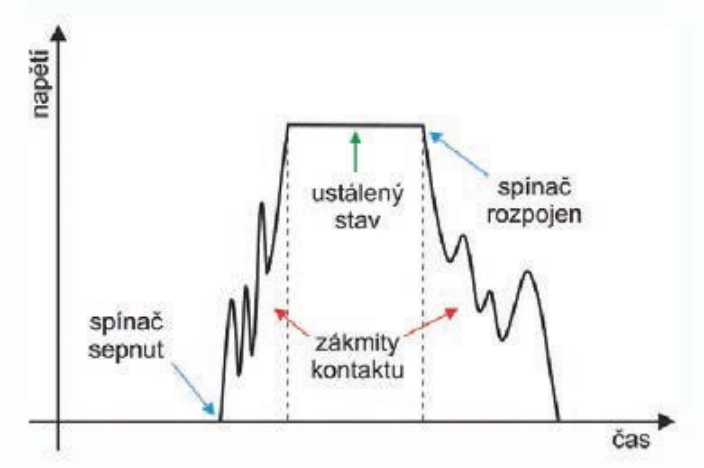
Tabulka 1 Porovnání napěťových logik [18]

	Standart 5V TTL		ATMega328 DC Characteristic	
	Na vstupu	**Na výstupu**	**Na vstupu**	**Na výstupu**
LOW	0 – 0,8 V	0 – 0,4 V	0 – 1,5 V	0 – 0,9 V
HIGH	2 – 5 V	2,7 – 5 V	3 – 5 V	4,2 – 5 V

Z tabulky a obrázku výše je zřejmé, že jsou rozlišné rozmezí chápání logické jedničky a nuly na vstupu a na výstupu zařízení. Z porovnání logik lze vyvodit, že standartní TTL má užší oblast vnímání úrovně LOW a širší oblast vnímání HIGH úrovně. Arduino to má přesně naopak a pro obě úrovně má přiměřenější rozsah napěťových hodnot.

2.1.2.2 Ošetření zákmitů

Při praktickém procvičování s Arduinem a tlačítky, by každý postupem času zjistil, že narazil na problém. Ten spočívá v zákmitech. V anglicky psané literatuře se tento stav označuje pojmem bouncing. Projevuje se to tak, že při stisku tlačítka 1x program zaregistruje například 3 stisky. To je způsobeno konstrukcí tlačítka, které má membránu a několikrát po stisku zakmitá a může tak způsobit vícero násobné stisknutí. Tento problém lze nejjednodušeji ošetřit softwarově. V principu jde o to, že po prvním stisku tlačítka je ignorováno, co se děje na vstupu dalších pár milisekund. Vstup opět vnímá tlačítko až po časové prodlevě. V programu se k tomu využívá funkce *delay*, která ale zastaví běh celého programu. To není příliš vhodné, a proto se využívá spíše funkce *millis*. Ta měří milisekundy od začátku programu. V programu se zavolá 2x a kontroluje se jejich rozdíl. Dokud není časový rozdíl dostatečně velký, vstup je zanedbáván.



Obrázek 4 Zakmitávání kontaktu (bouncing) [19]

2.1.2.3 Obsluha přerušení

Přerušení je znamení pro procesor, které mu dává vědět, aby okamžitě přestal dělat vykonávaný program a aby začal vykonávat obslužný program. Využití obsluhy přerušení se hodí při externím přerušení, kdy je například při stisku tlačítka vykonat nějakou činnost. Výhodné je to v tom, že systém reaguje na stisk tlačítka mnohem rychleji a programátorovi odpadá nutnost do programu implementovat algoritmy na kontrolu stisku tlačítka.

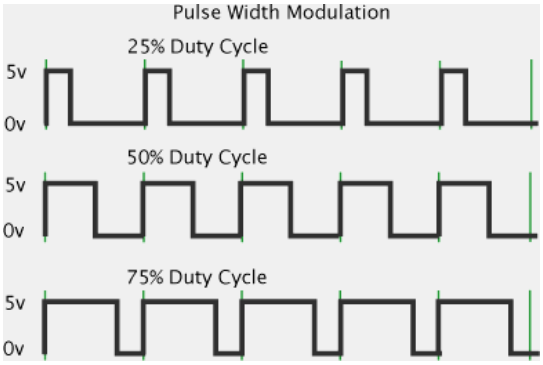
U Arduino Uno lze přerušení vyvolat na dvou digitálních vstupech. Jsou to piny D2 a D3. Sledovat lze čtyři druhy událostí:

- LOW – přerušení je vyvoláno kdykoliv je na pinu signál na úrovni LOW
- CHANGE – přerušení je vyvoláno kdykoliv přejde signál z jedné úrovně na druhou
- RISING – přerušení je vyvoláno kdykoliv na pinu stoupne signál z úrovně LOW na HIGH
- FALLING – přerušení je vyvoláno kdykoliv na pinu spadne signál z úrovně HIGH na LOW

2.1.2.4 PWM

Pulzní šířková modulace (pulse width modulation) je technika, kdy se dá pomocí digitálního signálu generovat v podstatě analogový signál. Princip je prostý. Když nelze měnit hodnotu napětí, tak lze HIGH úrovně a LOW úrovně střídat a vysílat je tak s určitou frekvencí. Jedná se o periodické střídání 0 a 1. Pokud bude čas zapnutí a vypnutí stejný, bude se připojené zařízení chovat, jako by na něj bylo přivedeno napětí poloviční. Tomu se říká napětí zdánlivé. Pokud bude doba zapnutí a vypnutí v jiném poměru, bude podle i úměrně velké zdánlivé napětí. U Arduina to lze využít pro řízení toho, jak moc bude LED svítit anebo pro snížení

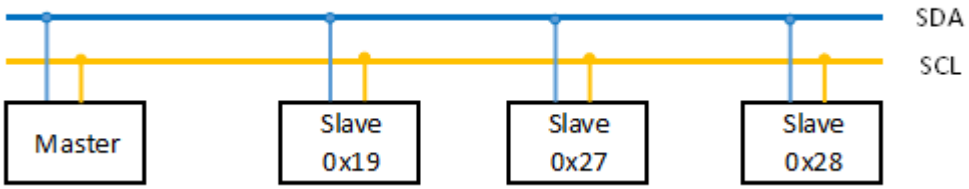
otáček DC motorů. U Arduina Uno lze pro generování PWM signálu využít piny číslo 3, 5, 6, 9, 10, 11.



Obrázek 5 Grafické znázornění průběhu signálu PWM

2.1.2.5 I2C sběrnice

Jedná se o sériový způsob komunikace. Sběrnice je sestavena ze dvou datových vodičů. Jedno nebo více zařízení je označené jako Master a ostatní jako Slave, kdy master je zodpovědný za řízení komunikace a ostatní jen odpovídají. Master je například deska Arduino. Sběrnice I2C v Arduinu umožňuje připojit až 128 zařízení. Aby bylo možné rozpoznat s jakým zařízením se komunikuje, má každé přiřazenou svoji adresu. Největší výhodou je samozřejmě ohromná úspora pinů, které by musely být využity na připojení všech zařízení.



Obrázek 6 I2C sběrnice

2.1.2.6 SPI

SPI neboli Serial Peripheral Interface je další sériová komunikační sběrnice. Na rozdíl od I2C se skládá ze čtyř signálů. To jsou SCLK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out) a SS/CS (Slave Select/Chip Select). Výhodou je větší rychlost oproti I2C a menší spotřeba energie. Nevýhodou je pak zmíněný větší počet vodičů, na jedné sběrnici lze zapojit menší počet zařízení a SPI neověřuje, že jsou data přijata správně.

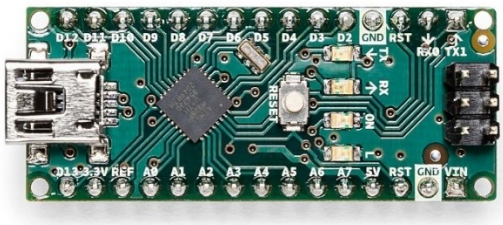
2.1.3 Verze desek Arduino

Firma Arduino nabízí nejrůznější verze desek. Každá je něčím specifická. Jedná se buď u technologie, kterými disponuje bez žádných dalších modulů a shieldů anebo je deska specifická například svými rozměry anebo připojovacím rozhraním.

Obecně dle desky řadit do rodin. Těch je celá řada a většina z nich je uvedena zde.

2.1.3.1 Nano Family

Nano rodina je specifická zejména svými rozměry. Obyčejné Arduino Nano je v podstatě totožné s Arduino Uno. Rozdíl je v rozměrech, takže se hodí do projektů, kde je tento aspekt podstatný. Vizuálně znatelný rozdíl je v tom, že mikrokontrolér již není zasazen v zásuvce pro integrované obvody. Nachází se zde v menší verzi a je pouze povrchově osazen. Zmenšený musí být i konektor pro nahrání programu a napájení. Oproti klasickému Unu je zde pouze jeden a je typu mini B USB. V rodině jsou k nalezení desky, jež jsou již například osazeny Wi-Fi modulem ESP32, což je Arduino Nano IoT anebo Arduino Nano RP2040 Connect. To se ale od klasického konceptu desek Arduino dosti odchyluje. Mikrokontrolér ATMega 328p od firmy Atmel je úplně nahrazen mikrokontrolérem RP2040. Tento mikrokontrolér je od firmy Raspberry Pi a dosahuje větších výkonů. To i proto že RP2040 obsahuje 133MHz 32bitový procesor Arm Cortex M0+. Tato deska je vhodná také pro IoT projekty, protože má již zabudovaný WiFi modul anebo IMU, což je v podstatě gyroskop.



Obrázek 7 Arduino Nano [14]

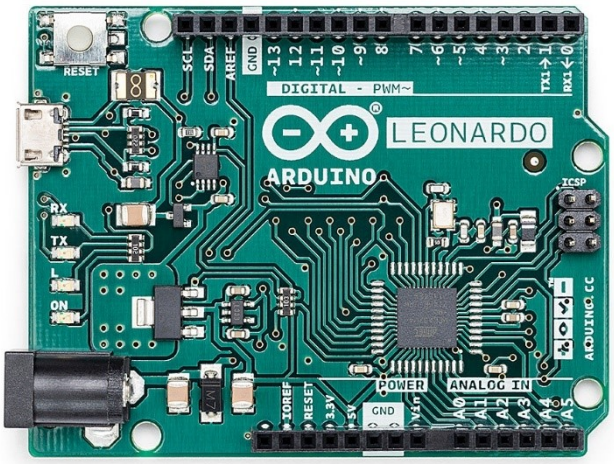
2.1.3.2 MKR Family

MKR je velmi specifická řada. V této rodině se nacházejí desky, které mají opět největší využitelnost v IoT odvětví. Tyto desky mají malé rozměry a spotřebu. Někteří zástupci z této rodiny mají svůj tvar až dosti atypický. Navíc jsou pro ně dostupné různé senzory v podobě lehce připojitelných shieldů.

Obrázek 8 Arduino MKR IoT Carrier Rev2 [14]

2.1.3.3 Classic Family

Classic family je zástupce standartního typu desek. Jedná se o druhou rodinu desek, pro které se vyrábí shieldy. Zbytek rodin je zatím oficiálně nemá. Typickým příkladem počítače z této rodiny je Arduino Uno. To má různé revize od R2 do R4. Je zde verze R2 minima, která disponuje mikroprocesorem RA4M1 od firmy Renesas, což desce dodává větší výkon. Další zajímavý exemplář je verze Leonardo, tím že má vestavěnou komunikaci přes USB a při připojení k PC se jeví stejně jako myš nebo klávesnice, což může být užitečné pro projekty k ovládání her a podobně. Zmenšenou verzí Leonarda je Micro. Přestože má menší rozměry, stále patří sem, a ne do rodiny Nano. Dále zde nalezneme i verze desek se zabudovaným WiFi modulem anebo jiným systémem napájení.

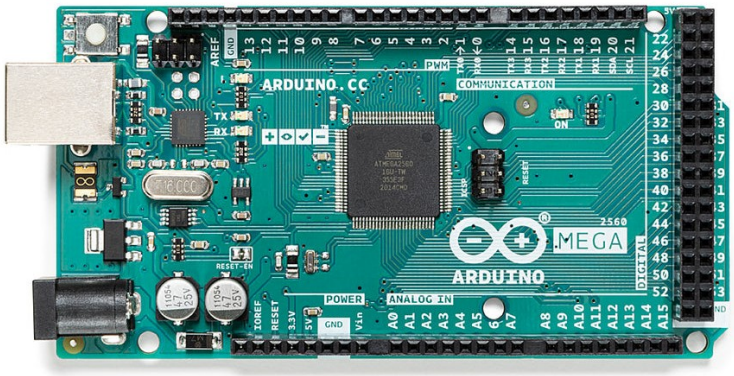


Obrázek 9 Arduino Leonardo [14]

2.1.3.4 Mega Family

Jak již vypovídá název tato rodina se odlišuje tím, že desky jsou rozměrově větší. Mají za to ale jisté výhody. Mezi ně patří 54 IO vstupů z nichž 15 umožňuje PWM a 16 analogových

vstupů. Hodí se tedy pro rozsáhlejší projekty. Bonusem je to, že jsou některé tyto SBC kompatibilní se shieldy z rodiny Classic. Naleznout zde také Arduino Due, které se odlišuje nejen svým procesorem, ale i napájecím napětím, které není oproti jiným deskám 5V ale 3,3V. S tímto napětím dále pracuje i na IO pinech.



Obrázek 10 Arduino Mega 2560 Rev3 [14]

2.1.3.5 Arduino Pro

Arduino Pro nelze označit jako rodinu, nýbrž jako samotnou platformu, jež obsahuje další rodiny. Jedná se o rodiny desek cílených pro průmysl. Výrobce konkrétně udává, že jejich výrobky jsou vhodné pro letectví, industry 4.0, robotiku, automobilový průmysl, chytré zemědělství a mnoho dalších. Desky jsou koncipovány pro vyšší výkony, a proto mají jiné mikrokontroléry. V principu se jedná profi provedení standartní platformy. V této platformě nalezneme i řadu shieldů anebo sofistikovanými senzorovými deskami. Novinkou je pak Arduino Opta. Jedná se o mikro PLC vyvinuté Arduinem ve spolupráci se společností Finder. Toto zařízení by mělo sloužit pro snadnou automatizaci budov a průmyslu. Bonusem je, že se dá programovat jako jiná PLC pomocí pěti programovacích jazyků definovaných normou IEC 61131-3.

Obrázek 11 Arduino Opta

2.2 Periferie

Pro všechny verze desek Arduino existuje mnoho druhů vstupních a výstupních zařízení, které lze připojit. Může se jednat o různé senzory. Snímat lze například vzdálenost, vlhkost, infračervené signály anebo obyčejné signály od tlačítek. Vstupní signály mohou být analogové, potom se připojují na analogové vstupy, anebo mohou být ve formě digitálních signálů, potom se digitální piny musí nastavit jak vstupy a lze na ně signály přivádět.

U výstupních signálů je u Arduina dostupný pouze signál digitální, díky PWM lze ovládat zařízení podobně jako s analogem. Spínat lze různá relé, ať už jednokanálové či více, anebo různé akční členy, jako servomotory, krokové motory, bzučáky, displeje atd. K displejům a jiným zařízením je pro připojení většinou výhodnější použít sběrnici, pokud to lze. Díky relé anebo jiným spínacím zařízením je možné s přehledem ovládat i nějaká výkonová zařízení. Jelikož je těchto periférií velká řada, jsou v následujících podkapitolách uvedeny jen ty, jež jsou nejznámější anebo ty, jež jsou využity pro ukázku v praktických úlohách.

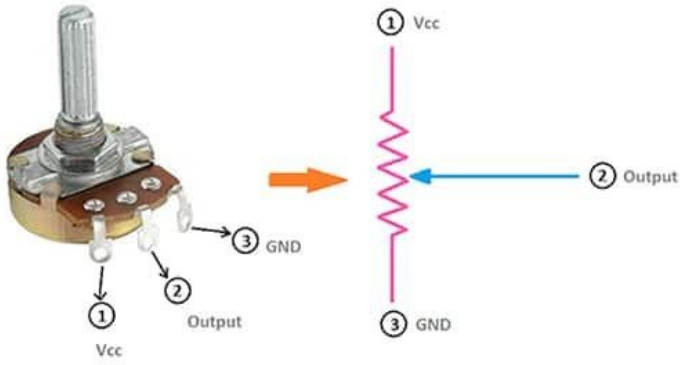
2.2.1 Vstupní periferie

2.2.1.1 Tlačítko

Tlačítko slouží jako jeden z nejzákladnějších vstupů. Většinou je založeno na mechanickém principu, kde je stiskem tlačítka způsobeno prohnutí membrány ke kontaktu. Tímto se sice rozpozná stisk, ale dojde tím také k zákmitům (rozkmitání membrány a nechtěnému opětovnému stisku) a ty je nutné ošetřovat. Mechanická tlačítka se i proto v některých použitích nahrazují kapacitními tlačítky.

2.2.1.2 Potenciometr

Potenciometr je součástka, jež umožnuje vyvinout elektrický odpor a měnit jeho velikost v určitém rozsahu. Výstupem z něho je tedy analogová hodnota. Potenciometr má 3 vývody. Krajní z nich jsou konce odporové dráhy a prostřední z nich je spojena s jezdcem. Odporová dráha bývá realizována vrstvou uhlíku anebo navinutým odporovým drátem. Rozlišují se potenciometry otočné a lineární.



Obrázek 12 Otočný potenciometr [21]

2.2.1.3 Senzory

Existuje mnoho druhů senzorů a lze s nimi snímat nejrůznější veličiny. Může se jednat o snímaní teploty, vlhkosti, vzdálenosti, detekci světla, barev, senzory plynů, detekci pohybu, váhové senzory a tak dále.

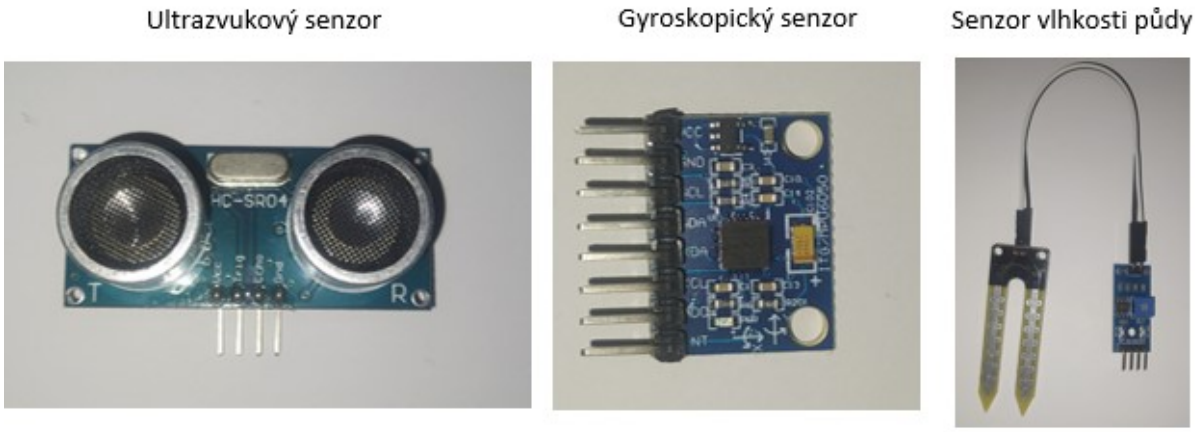
Teplotní senzory jsou založeny nejčastěji na termistorech. To jsou rezistory, jejichž odpor závisí na okolní teplotě. Dále mohou fungovat na principu bimetalových spojených plíšků, přičemž se snímá ohnutí plíšků.

Vlhkostní senzory se rozlišují na měření vlhkosti půdy a ovzduší. Z principu využívají měření rezistence mezi dvěma body.

Čidla vzdálenosti jsou založena na principu vysílání a odrazu. Využívají k tomu například ultrazvukových vln, infračerveného záření anebo laserových paprsků. Podobných principů se využívá i při snímání pohybu.

Pro detekci barev a světla se využívá fotorezistorů, u kterých odpor závisí na okolním osvětlení, dále se využívá fotodiod anebo komplexnějších rgb senzorů.

Všechny senzory zkrátka fungují na principu vnímání snímané veličiny a jejím převodu na elektrické napětí anebo proud, defacto i odpor.



Obrázek 13 Ukázka senzorů (Zdroj: vlastní zpracování)

2.2.1.4 RFID čtečka

RFID je technologie, jež k přenosu dat využívá rádiových vln. Jedním z nejčastějších využití RFID je použití této technologie pro přístupové systémy, kde každý uživatel má svůj takzvaný tag. Tag je fyzická karta nebo čip a pokaždé uchovává svoji adresu. Pomocí přiložení ke čtečce a přečtení adresy se dá vyhodnotit, zda má uživatel k určité činnosti nebo vstupu práva či ne. Tyto tagy fungují většinou na frekvenci 125 kHz anebo 13,56 MHz. Při práci s touto technologií je tedy vhodné si nejprve zjistit s jakou frekvencí se bude pracovat.

2.2.2 Výstupní periferie

2.2.2.1 LED

LED neboli světlo emitující dioda je polovodičová součástka, která při připojení napájení emituje světlo. Je založena na PN přechodu. Oproti žárovkám je mnohonásobně úspornější, kompaktnější, navíc má i delší životnost a lze ji sehnat v mnoha barevných variantách. Bohužel u ní ale není lhostejná polarita. Při otočení polarity, dojde k jejímu zničení. Kladné elektrodě se říká anoda a z praxe se jedná o delší vývod. Záporná elektroda se nazývá katoda, jedná se o kratší vývod a nachází se u zploštělého konce LED diody.

2.2.2.2 Relé

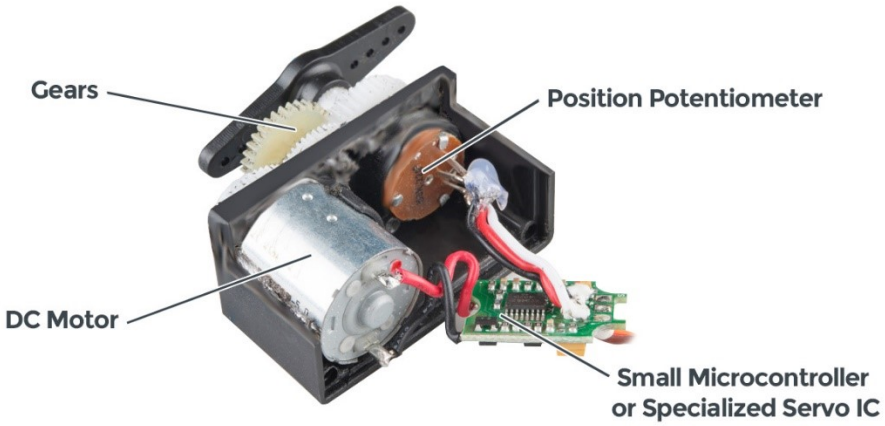
Relé je elektromechanický spínací prvek. Slouží k ovládání jednoho elektrického obvodu pomocí druhého. Pomocí řídícího signálu lze spínat například obvody s motory, silnoproudou elektronikou atd. Skládá se z principu z cívky a kontaktů. Jelikož se jedná o mechanické zařízení patří mezi nevýhody spínací rychlost a omezená životnost. Další nevýhodou je i rušení způsobené spínáním. Výhodou pak ovšem bývá snadná nahraditelnost a jejich jednoduchost. V dnešní době je nahrazují SSR relé založené na polovodičích.

2.2.2.3 Mosfet

Mosfet je zkratka pro Metal-Oxide-Semiconductor Field-Effect Transistor, což je typ polovodičového tranzistoru, který se používá k řízení a zesilování elektrických signálů v mnoha elektronických zařízeních. V podstatě se dá použít i jako relé. Má výhody v tom, že je menší, má menší spotřebu, delší životnost a rychlejší spínání. Navíc neprojevuje rušení. Nevýhodou, pokud bude určen pro spínání silnoproudých obvodů, bude jeho cena.

2.2.2.4 Servomotor

Jedná se o motor, který za pomocí řídícího signálu uvede hřídel do určité pozice. Aby se dalo vyvodit, v jaké pozici se zrovna hřídel nachází, je nutno sledovat její pozici. U malých servomotorů, jakých bývá využito třeba právě při práci s Arduinem, je hřídel motoru spjata přes převodovku spolu s potenciometrem. Na základě odporu se pak vyhodnotí natočení.



Obrázek 14 Průřez servomotorem [23]

2.2.2.5 Displej

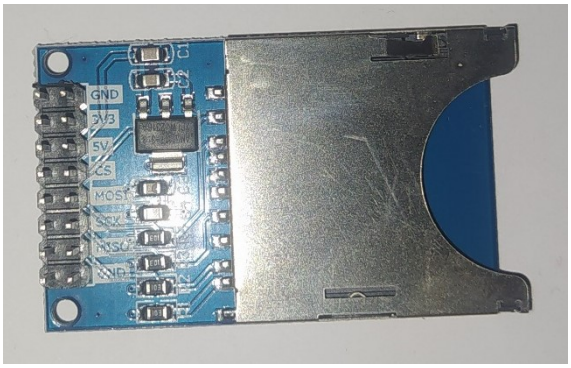
Displej je obecně prostředek pro vizuální zobrazení informací. U Arduina se nejčastěji používají LCD displeje anebo IPS. Zjednodušeně LCD displej funguje na principu kapalných krystalů a pomocí jejich natáčení elektrickým polem se zajišťuje zobrazování obrazu. Zespodu displeje je vyzařováno světlo a pomocí krystalů buď světlo prochází anebo je zastíněno. Základním displej má rozměry 16x2 digitů, lze u něj spínat a regulovat podsvícení a lze jej zakoupit ve variantě se zabudovaným převodníkem na I2C sběrnici. To je vhodné pro úsporu pinů mikroprocesoru.

2.2.3 Vstupně výstupní periferie

2.2.3.1 Čtečka SD karet

Čtečka karet pro mikrokontrolery bývá navržena pro sériovou komunikaci přes SPI sběrnici. Pomocí této čtečky lze číst i zapisovat na SD kartu. K obojímu je zapotřebí využití funkcí ze

speciální knihovny. Její využitelnost je v projektech, kde je nutné ukládat logy aneb nějaké reporty, popř. i obrázky.



Obrázek 15 Čtečka SD karet pro jednodeskové počítače (Zdroj: vlastní zpracování)

2.2.3.2 RTC modul

Modul hodin reálného času je zařízení, které obsahuje obvod, jež dokáže uchovávat datum i čas. Díky vlastní baterii udrží tyto hodnoty i při výpadku proudu. Při vybití baterie lze čas a datum nastavit znovu anebo některé moduly mají možnost automatické synchronizace s internetovými časovými servery.

2.2.3.3 Ethernet modul

Ethernet modul nejčastěji obsahuje typický konektor RJ45. Pro jeho využívání je opět potřeba využít speciálních knihovních funkcí. Pomocí síťového rozhraní lze například odesílat emaily anebo data na vzdálené úložiště. Tyto moduly obvykle podporují řadu protokolů, jako je TCP/IP, UDP, DHCP, DNS. Napájení tohoto modulu lze zajistit externí anebo přímo z jednodeskového počítače.

3 Programování v jazyku Wiring

Tato kapitola se zabývá základními znalosti o jazyku Wiring. Popisuje vývojové prostřední a stručně definuje vzhled programu. Pro obsáhlejší informace lze použít oficiální dokumentaci, konkrétně sekci programming. Dostupné na https://docs.arduino.cc/programming/ .

3.1 Jazyk Wiring

Arduino lze programovat v jazycích C nebo C++. Nejlepší a nejjednodušší způsob, jak lze psát programy pro tyto počítače je s využitím knihovny Wiring. Kvůli její rozsáhlosti se, ale považuje za programovací jazyk. [3]

3.1.1 Dostupné datové typy a objekty

Dostupné datové typy, které jsou přímo výčtově popsány v dokumentaci jsou v tabulce níže. Tabulka se vztahuje k datovým typům při použití desky Arduino Uno R3. Při použití jiných desek se rozsahy datových typů mohou lišit.

Tabulka 2 Dostupné datové typy a objekty jazyku Wiring při použití Arduino Uno R3 [22]

Datový typ	Užití/popis	Rozsah
Array	Množina oindexovaných proměnných stejného typu	Dle dostupné paměti a dle datového typu prvků
Bool	Pravdivostní hodnota	True/False
Boolean	Pravdivostní hodnota, nestandartní alias pro bool	True/False
Byte	Celé číslo	-127 až 128
Char	Hodnota znaku	1 znak ASCII (0 – 255)
Double	Desetinné číslo	-3,4028235E+38 až 3,4028235E+38
Float	Desetinné číslo	-3,4028235E+38 až 3,4028235E+38
Int	Celé číslo	-32 768 to 32 767
Long	Celé číslo	-2 147 483 648 až 2 147 483 647
Short	Celé číslo	-32 768 až 32 767
size_t	Velikost libovolného objektu v bajtech (celé číslo)	Min. 0 – 65 535

String	Pole znaků ukončené nulovým znakem	Dle dostupné paměti
String()	Třída pro práci s textovými řetězci	Dle dostupné paměti
unsigned char	Hodnota znaku	1 znak ASCII (0 – 255)
unsigned int	Celé kladné číslo včetně nuly	0 až 65 535
unsigned long	Celé kladné číslo	0 až 4 294 967 295
Word	Celé kladné číslo	0 až 65 535

Pokud bude pro vývoj použita jiná deska, například Arduino Due, dojde k tomu, že rozsah datového typu *int* bude 4 bajty, namísto dvou a to odpovídá rozsahu od -2 147 483 648 až 2 147 483 647. Adekvátně se změní i *unsigned int*. Další změnou bude například rozsah pro *double*, který bude 8 bajtový, atd.

3.1.2 Základní konvence a pravidla v jazyku

Jazyk Wiring je odvozený od jazyka C++. Ten sice umožňuje i objektový přístup, ale Wiring pouze strukturovaný. To znamená, že kód je zpracován sekvenčně. Je v něm podpora proměnných, konstant, funkcí, cyklů a podmínek.

Za každý neblokovým příkazem je nutné psát středník.

Co se týká case senstive se u tohoto jazyka rozlišují velká a malá písmena v identifikátorech jako jsou názvy proměnných, funkcí, tříd atd. Nejčastěji lze v ukázkách kódu naleznout použitý styl CamelCase, kdy je více slov spojeno do jednoho pojmenování, ale se zachováním velkých počátečních písmen.

3.1.3 Komentář

Komentář je část kódu, která není vykonávána počítačem jako součást programu. Jedná se jisté poznámky, psány lidským jazykem, dokumentující určitou oblast kódu. Slouží ke zlepšení orientace v kódu, usnadnění jeho čtení jiným vývojářům anebo obsahuje připomínky ke kódu nebo informace o tom, co je potřeba v programu doimplementovat. V jazyce Wiring se používá dvou typů komentářů. První z nich je komentář jednořádkový, který začíná dvěma lomítky. Druhý je komentář více řádkový, který začíná lomítkem a hvězdičkou a končí těmito symboly v opačném pořadí.

```
//Toto je jednořádkový komentář

/*
 *Toto je
 *víceřádkový
 *komentář
 */
```

Obrázek 16 Ukázka komentářů

3.1.4 Proměnná a konstanta

Proměnné a konstanty slouží k uchování a manipulaci s daty. Proměnným lze v průběhu programu hodnotu měnit, u konstant ji lze nastavit pouze jednou. Při práci s nimi je nutné nastavit jejich scope, což znamená nastavit jejich viditelnost. Toho se docílí tak, že globální proměnná je deklarovaná mimo bloky *setup* a *loop*. Lokální je pak uvnitř. Lze definovat i takzvané makro, což je konstantní hodnota, která umožňuje programátorovi pojmenovat konstantní hodnotu před kompilací programu a nezabírá místo v paměti programu. Dále je třeba při deklaraci volit správný datový typ (například pro celočíselnou hodnotu zvolit int).

```
int cislo = 1;          //globální proměnná
const int cislo2 = 2;   //globální konstanta
#define cislo3 3        //definování makra (konstantní hodnoty)

void setup(){
  //zde by mohly být definované lokální proměnné a konstanty
}

void loop(){
  //zde by také mohly být definované lokální proměnné a konstanty
}
```

Obrázek 17 Proměnné a konstanty v jazyce Wiring

3.1.5 Větvení – Podmínka, switch

Větvení slouží k provádění různých akcí na základě nějaké podmínky. Rozlišují se dvě konstrukce. První z nich je *if-else*, neboli podmíněný výraz. Struktura kódu vypadá tak, že začíná slovem *if* a v závorce následuje podmínka. Pokud je podmínka splněna, program pokračuje vykonáním kódu v sekci ve složených závorkách za koncem podmínky. Pokud ne, program pokračuje blokem kódu *else.* Lze vyhodnocovat více podmínek za sebou, pro to je vhodné užití *else if*. Podmínky lze do sebe i vnořovat.

```
int hodnota = 75;

if(hodnota < 50) {
  //provede se, pokud hodnota je menší než 50
  Serial.println("Hodnota je menší než 50");
}else if (hodnota >= 50 && hodnota < 100){
  //provede se, pokud hodnota je v rozmezí od 50 do 99
  Serial.println("Hodnota je v rozmezí 50 až 99");
}else{
  //provede se, pokud žádná z předchozích podmínek není splněna
  Serial.println("Hodnota je 100 nebo větší");
}
```

Obrázek 18 Podmínka v jazyce Wiring

Druhou konstrukcí je takzvaný switch-case. Umožňuje provádět různé akce na základě hodnoty nějaké proměnné. Je tedy vhodná v případě, kdy může proměnná nabývat mnoha hodnot a je nutné na jejich základě provádět mnoho různých akcí.

```
int cislo = 2;

switch (cislo){
  case 1:
    //provede se tento blok, pokud je cislo rovno 1
    break;
  case 2:
    //provede se tento blok, pokud je cislo rovno 2
    break;
  default:
    //provede se tento blok, pokud nebyla splněna žádná z předchozích podmínek
    break;
}
```

Obrázek 19 Switch konstrukce v jazyce Wiring

3.1.6 Cykly

Cykly slouží k opakovanému provádění určité části kódu. Ten se má provádět vždy dokud je podmínka platná. V jazyce Wiring se v zásadě nacházejí tři druhy cyklů.

První je *for* cyklus. Po klíčovém slovu následují v obyčejných závorkách nejprve počáteční podmínka, tj. obvykle inicializace proměnných, za středníkem následuje podmínka. Ta, když je splněna, blok kódu v těle cyklu se provede. A za další středníkem se nachází aktualizace, což bývá nejčastěji inkrementace počátečně definované proměnné.

```
//Cyklus for pro výpis čísel od 0 do 9
for(int i = 0; i < 10; i++) {
  Serial.println(i);
}
```

Obrázek 20 For cyklus v jazyce Wiring

Dalším cyklem je *while* cyklus a jak již název napovídá, opakuje se tolikrát, dokud platí podmínka. Struktura je jednodušší a za klíčovým slovem se v závorkách nachází hned podmínka.

```
//Cyklus while pro výpis čísel od 0 do 9
int cislo = 0;
while (cislo < 10) {
  Serial.println(cislo);
  cislo++;
}
```

Obrázek 21 While cyklus v jazyce Wiring

Posledním je vlastně otočený cyklus while. Jmenuje se *do-while* a otočený je proto, protože se nejprve vykonává blok kódu a až následně se kontroluje platnost podmínky. Pokud je platná, provede se blok kódu znovu.

```
//Cyklus do-while pro výpis čísel od 0 do 9
int cislo = 0;
do {
  Serial.println(cislo);
  cislo++;
} while (cislo < 10);
```

Obrázek 22 Do-while cyklus v jazyce Wiring

Provádění cyklu lze ukončit, pokud se narazí při vykonávání na *break*.

3.1.7 Pole

Pole je indexovaný jedno nebo více rozměrný seznam určitých dat. Slouží jak k uchování, tak manipulaci s těmito daty. Struktura zápisu kódu pro pole je následující. Nejprve se určí, jakého datového typu budou data, následuje název a po něm hranatá závorky s rozměry. Pole lze i při deklaraci inicializovat, což by znamenalo, že po rozměrech a po znaku rovná se, následují ve složených závorkách konkrétní hodnoty, oddělené čárkou.

```
//Deklarace a inicializace pole
int pole[5] = {10, 20, 30, 40, 50};
```

Obrázek 23 Pole v jazyce Wiring

3.1.8 Funkce

Funkce je blok kódu, který provádí určitou činnost a může být volán z jiné části programu, z jiné funkce anebo i z jiné knihovny. Struktura funkce sestává ze stanovení návratového typu, což je některý z dostupných datových typů, anebo *void*, což značí, že funkce nic nevrací. Následuje název, v závorkách parametry a ve složených závorkách tělo funkce. To, pokud se jedná o metodu, co má vracet nějakou hodnotu, musí obsahovat slovo *return* a za ním hodnotu, jež funkce vrací. Funkci lze také ukončit pouze pomocí *return;* . Funkce se deklarují mimo části *setup* a *void.*

```
//Deklarace a definice funkce
int secti (int a, int b) {
  return a + b;
}

void setup(){
  // inicializace
}

void loop() {
  //Volání funkce
  int vysledek = secti(3, 5);
  Serial.println(vysledek);
}
```

Obrázek 24 Funkce v jazyce Wiring

3.1.9 Implementované a knihovní funkce

V zabudované knihovně najdeme objekt *Serial.* Z něho se nejčastěji využívá funkce *print* anebo *println* pro výpis do konzole. Dále pak *begin* pro inicializaci sériového přenosu a nastavení přenosové rychlosti.

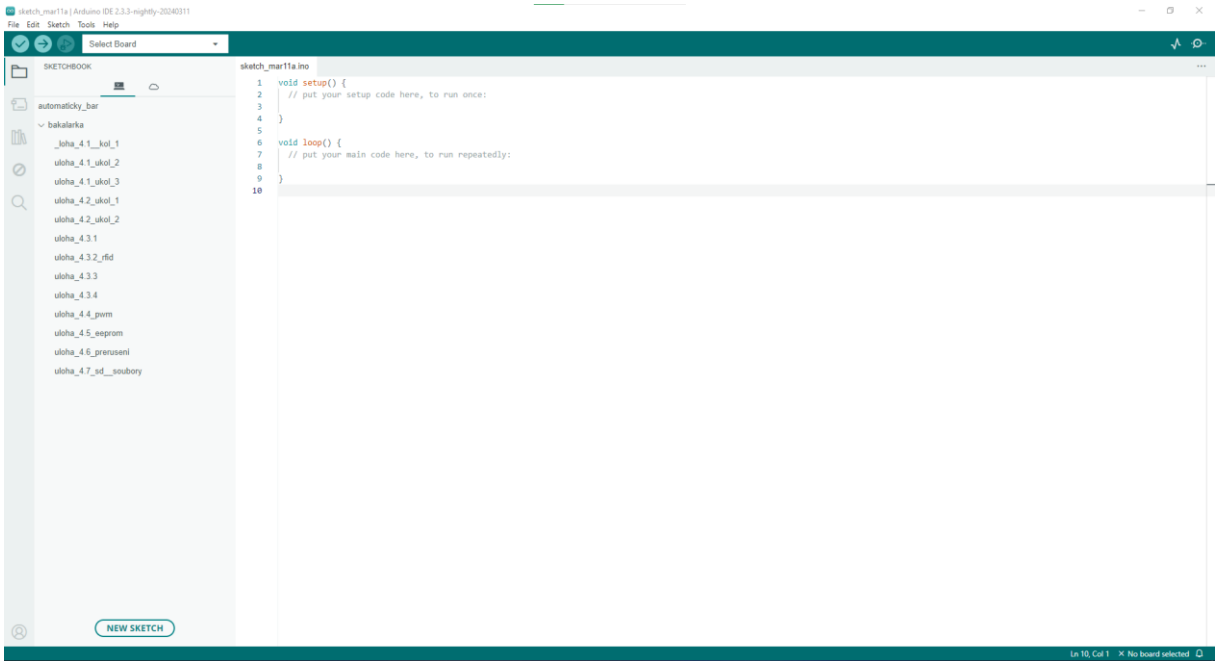
Další hodně používanou funkcí je *delay*, která slouží k pozastavení programu na určitý časový okamžik. Dále používanou funkcí je *millis*, která měří uplynulý čas od startu běhu programu.

Nesmí chybět zmínka o *digitalWrite* a *digitalRead.* Ty slouží k zápisu a čtení hodnot na digitálním pinu. Podobně jen pro analogové piny existují *analogWrite* a *anologRead*. S tím

souvisí funkce *pinMode*, která požadovaný pin nastaví tak, aby se choval buď jako vstup nebo jako výstup.

3.2 Vývojové prostředí

Vývojové prostředí pro jazyk Wiring a programování Arduino desek se jmenuje Arduino IDE. Existuje verze 1.0 a verze 2.0. Novější generace, nyní na počátku roku 2024 konkrétně verze 2.3.2 oproti starší první řadě vývojového prostředí nabízí například našeptávač, příjemnější uživatelské prostředí (např. volba mezi světlým a tmavým motivem) anebo podpora Arduino Cloudu. V aplikaci zabírá většinu plochy prostor pro psaní kódu. V dolní části se při určitých akcích zobrazí konzole. Pod menu nabídkou se nachází lišta a na té tlačítko pro kompilaci kódu a nahrání jej do paměti mikrokontroléru. Následuje tlačítko pro debugování a combobox pro výběr připojené desky. Dále je zde možnost zobrazení sériového monitoru anebo grafu. V levé části se pak nachází manažeři pro nedávné projekty, knihovny, desky, debugging a vyhledávač.



Obrázek 25 Vývojové prostředí Arduino IDE 2.3.2

Existují i další aplikace. Pro PLC řadu desek existuje například Arduino PLC IDE anebo dalším softwarem je Arduino CLI, které slouží jako uploader, správce desek atd. přes příkazový řádek.

3.3 Struktura kódu pro Arduino

Struktura kódu v jazyce Wiring je celkem jednoduchá. Program lze dělit do čtyř částí. První z nich je deklarace globálních proměnných, definování konstant a importování knihoven.

Další částí programu je funkce *setup()*. Tato část kódu se provede vždy pouze jednou po spuštění. Typicky se v ní inicializuje sériová komunikace anebo IO piny.

Třetí část tvoří funkce *loop()*. Jak již název napovídá, jedná se o neustále se opakující část programu. Ukončit ji lze pouze ukončením programu. Zde se typicky kontrolují vstupy a ovládají výstupy.

Pod touto částí je prostor pro definování vlastních funkcí, jež je možné zavolat z druhé anebo třetí části.

4 Praktické programování a úlohy pro Arduino

V této kapitole se nachází sestava úloh, které mají cíl obeznámení se základy programování jazyka Wiring, rozvoje logického myšlení a rozvoje vědomostí spojených s programováním jednodeskových počítačů.

Pro každou úlohu je nutné využít vhodné IDE, jako například Arduino IDE 2.2.1 anebo zastaralejší Arduino IDE 1.8.19. Z hardwaru je pak nutné mít desku Arduino UNO, USB kabel k sériové komunikaci mezi deskou a PC, ideálně nepájivé pole, řadu kompatibilních vodičů. Další nutné komponenty, které jsou potřebné k sestavení úlohy, jsou vždy uvedeny v požadavkách na začátku každé podkapitoly. Lze použít i jiné verze desky Arduino, ale není zde zaručeno, že bude řešení úloh vždy řešitelné anebo stejné.

Nejvhodnější variantou z pohledu HW je použití nějakého ze základních startovacích kitů a popřípadě dokoupení periferií. [1]

Obrázek 26 Startovací kit Arduino pro začátečníky [1]

Před každou úlohou je vhodné studenty obeznámit se základní problematikou kapitoly, s využívanými technologiemi, senzory anebo akčními členy. Též je nutné představit způsob používání těchto prvků. Řešení úloh jsou vždy ukázková, což znamená, že může existovat více správných řešení.

4.1 Práce s LED, funkce millis(), výpis do konzole

4.1.1 Potřebný hw

V této úloze je nutné mít kromě základních komponent navíc diodu typu LED a předřadný rezistor. Zde je použita červená LED 5mm a předřadný odpor 220Ω.

4.1.2 Zadání

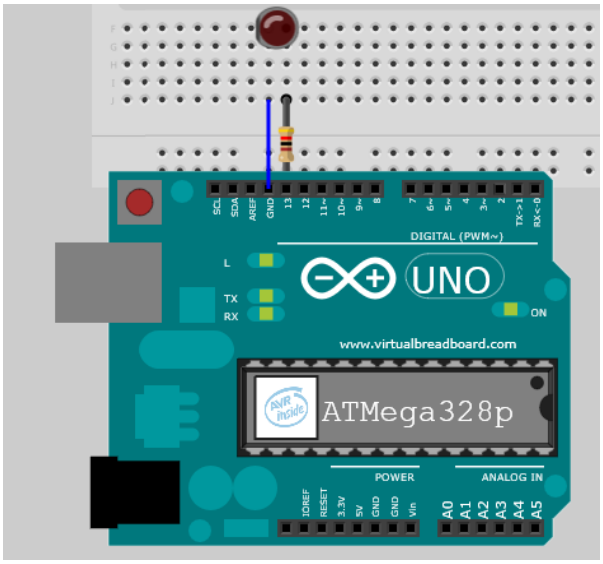
Tato úloha se skládá z tří úkolů.

Úkol č.1: Realizujte výpis do konzole vývojového prostředí každé 2 sekundy. Vypisovaný řetězec bude „Hello World“. Časování řešte pomocí funkce *delay*. Před každý řetězec vypisujte o kolikátý výpis se jedná.

Úkol č.2: Modifikujte úkol č.1 tak, že místo funkce *delay* využijete funkci *millis*.

Úkol č.3: Realizujte zapojení vyobrazené níže a zkuste nejprve rozsvítit LED na 5 sekund, poté pomocí cyklu zajistěte, aby LED po dobu 3s zablikala s frekvencí 10Hz a poté zajistěte, aby začala blikat frekvencí 1Hz.

4.1.3 Zapojení



Obrázek 27 Zapojení úlohy 4.1

4.1.4 Jak řešit

Úkol č.1: První věc, která je nutná udělat při řešení, je nastavení sériové komunikace, například na hodnotu 9600 baudů. Toho lze docílit pomocí *Serial.begin*. Ta stačí inicializovat pouze jednou, takže se nachází ve funkci *setup*. Dále je možné realizovat výpis do konzole pomocí *Serial.print* a časování pomocí *delay*. Nyní chybí už jen počítadlo pro indexaci výpisu. Na začátku programu je proto definována globální proměnná typu integer, v části *setup* je definována na počáteční hodnotu a vždy na konci funkce *loop je* inkrementována.

Úkol č.2: Pro vyřešení úlohy je nejprve z kódu z předchozího úkolu odmazáno volání funkce *delay.* Dále je stanovena globální proměnná pro uchování posledního času výpisu do konzole a další proměnná na začátku funkce *loop*, do které se vždy uloží aktuální hodnota času od

začátku běhu programu. Následně je nutné vytvořit *if-cyklus,* kde se porovnáním aktuálního a předchozího času, zjišťuje, zda uplynulo alespoň 2000ms od posledního výpisu do konzole. V případě, že ano, provede se aktualizace posledního času výpisu na aktuální, provede se výpis a inkrementace počítadla.

Úkol č.3: K splnění tohoto úkolu je nutné si nejprve definovat pomocí globální proměnné, ke kterému digitálnímu pinu, je výstupní periferie připojena. V tomto případě je to pin D13. Dále je nutné nastavit chování tohoto pinu jako výstupní. Toho lze docílit pomocí příkazu *pinMode(ledPin, OUTPUT)*. Ten stačí provést pouze jednou, a je tak proto ve funkci *setup.* Dále je nutností si přepočítat frekvenci blikání na jednotlivé fáze. To je například, že blikání s frekvencí 1 Hz rovná se tomu, že se LED 1x za sekundu rozsvítí a 1x zhasne. Rozdělit se to dá ideálně rovnoměrně, tedy 500 ms svítí a 500ms nesvítí. K rozsvícení a zhasnutí se použije volání funkce *digitalWrite*. K tomu, aby po určitou dobu činnost vykonávala lze použít *delay*, např. *delay (500)*. Blikání s f = 10 Hz po dobu t = 3 s pomocí cyklu proveďte *for-cyklem.* Tato frekvence vlastně určuje velikost periody T na 100ms, což je doba kdy LED svítí i nesvítí. Tedy 50 ms a 50 ms. Pakliže, se tento děj opakuje v cyklu 30krát, dostaneme 3 s.

4.1.5 Řešení v jazyce Wiring

```
int counter;

void setup() {
  Serial.begin(9600); // inicializace sériové komunikace
  counter = 1;     // nastavení počítadla výpisů na 1
}

void loop() {
  Serial.println(String(counter) + " Hello world"); // výpis spolu s enterem
  delay(2000); // zastavení programu na 2000ms
  counter++; // inkrementace počítadla
}
```

Obrázek 28 Programové řešení úlohy 4.1 - úkol 1

```
int counter = 1;
unsigned long previousMillis = 0; // ukládá poslední čas, kdy se provedla akce

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

void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= 2000) {
    previousMillis = currentMillis; // uložení času poslední akce
    Serial.println(String(counter) + " Hello world");
    counter++;
  }
}
```

Obrázek 29 Programové řešení úlohy 4.1 - úkol 2

```
const int ledPin = 13; // LED je připojena k digitálnímu pinu 13

void setup() {
  pinMode(ledPin, OUTPUT); // Nastavíme pin s LED jako výstupní

  // 1. část: Rozsvítit LED na 5 sekund
  digitalWrite(ledPin, HIGH); // Rozsvítíme LED
  delay(5000);                // Počkáme 5 sekund (5000 milisekund)

  // 2. část: Blikání LED s frekvencí 10Hz po dobu 3 sekund
  for (int i = 0; i < 30; i++) { // 10Hz blikání po dobu 3 sekund (10*3)
    digitalWrite(ledPin, HIGH);   // Zapneme LED
    delay(50);                    // Počkáme půl cyklu (1000ms / (2 * 10Hz))
    digitalWrite(ledPin, LOW);    // Vypneme LED
    delay(50);                    // Počkáme půl cyklu
  }
}

void loop() {
  // 3. část: Blikání LED s frekvencí 1Hz
    digitalWrite(ledPin, HIGH); // Zapneme LED
    delay(500);                // Počkáme půl sekundy (půl cyklu 1Hz)
    digitalWrite(ledPin, LOW); // Vypneme LED
    delay(500);                // Počkáme půl sekundy (půl cyklu)
}
```

Obrázek 30 Programové řešení úlohy 4.1 - úkol 3

4.2 Práce s tlačítky, ošetření zákmitů

4.2.1 Potřebný hw

Základní potřeby, LED, předřadný odpor 220Ω, 2x tlačítko

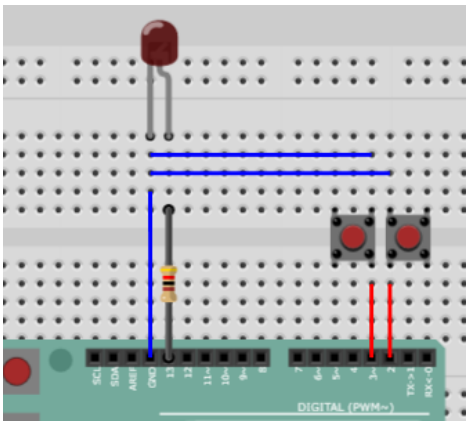
4.2.2 Zadání

Tato úloha se skládá ze dvou úkolů.

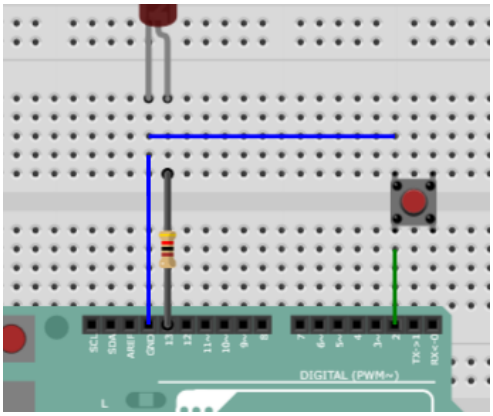
Úkol č.1: Realizujte zapojení 4.2 a). Dále napište program, který při stisku jednoho tlačítka rozsvítí LED a druhé tlačítko bude diodu zhasínat.

Úkol č.2: Modifikujte zapojení tak, aby odpovídalo zapojení 4.2 b), tedy odeberte jedno tlačítko. Napište program, který pomocí jednoho tlačítka dokáže rozsvítit a i zhasnout LED. Příklad: Spustí se program a LED nesvítí, stiskne se 1x tlačítko a LED se rozsvítí, stiskne se podruhé, tak LED zhasne. K tomu využijte softwarové ošetření zákmitů.

4.2.3 Zapojení



Obrázek 31 Zapojení úlohy 4.2 a)



Obrázek 32 Zapojení úlohy 4.2 b)

4.2.4 Jak řešit

Úkol č.1: Pro vyřešení tohoto úkolu je nutné inicializovat pin, na který je připojena LED jako výstupní, a to pomocí příkazu *digitalWrite(„číslo pinu“,OUTPUT).* V případě pinů, jež jsou připojena na tlačítka je nutné si pomocí této funkce definovat piny jako vstupní. Pakli-že je dodrženo zapojení, jež je vyobrazeno na přiloženém zapojení, je nutné pin definovat jako INPUT_PULLUP. To znamená, že Arduino využije zabudovaný rezistor a přes něj přivádí na digitální pin logickou 1. Tento pin je připojen na jeden vývod tlačítka. Ten druhý vede na GND pin desky. To je i důvodem, proč stisk tlačítka nastane, když je stav digitálního vstupu LOW a ne HIGH. Kontrolu stisknutí, tedy zda je na vstupu logická nula, je provedeno pomocí podmínky *if*. Následně je pak LED vypnuta anebo zapnuta.
Úkol č.2: Pro řešení tohoto úkolu je nutné vypracovat předešlý úkol a upravit zapojení dle schématu. Aby bylo možné spolehlivě ovládat rozsvícení a zhasnutí LED pomocí jednoho tlačítka, je nutné využít takzvané ošetření zákmitů. Při stisku tlačítka nedojde totiž pouze k tomu, že se spojí dva vodivé kusy materiálu, ale ve skutečnosti dojde zakmitání a opětovnému rozpojení a spojení těchto 2 vodivých ploch. Proto je ošetření zákmitů nutností. Existuje jednodušší způsob, kdy se kontroluje, zda došlo ke stisku tlačítka a v případě, že je tak vyhodnoceno, tak hned po splnění této podmínky ignoruje vstup na pár set milisekund. K tomu lze využít funkci *delay*. Tento úkol je však řešen sofistikovanější cestou, kdy se používá *millis()* místo *delay* a nezastaví se tak běh programu. Tato funkce zjišťuje, kolik milisekund nastalo od začátku běhu programu. Ukládá se, kdy došlo ke změně stavu tlačítka a ukládá se i samotný stav. V praxi to znamená, že když uživatel stiskne tlačítko, tak se uloží čas stisku a sleduje se doba od stisku tlačítka, což podobně jako obyčejné *delay* ignoruje dobu, kdy dochází k zákmitům.

4.2.5 Řešení v jazyce Wiring

```
const int ledPin = 13;       // LED je připojena k pinu 13
const int buttonOnPin = 2;   // Tlačítko pro zapnutí LED je připojeno k pinu 2
const int buttonOffPin = 3;  // Tlačítko pro vypnutí LED je připojeno k pinu 3

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonOnPin, INPUT_PULLUP);
  pinMode(buttonOffPin, INPUT_PULLUP);
  digitalWrite(ledPin, LOW);
}

void loop() {
  if (digitalRead(buttonOnPin) == LOW) {
    // Pokud je tlačítko pro zapnutí stisknuto, rozsvítí se LED
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonOffPin) == LOW) {
    // Pokud je tlačítko pro vypnutí stisknuto, LED se vypne
    digitalWrite(ledPin, LOW);
  }
}
```

Obrázek 33 Programové řešení úlohy 4.2 - úkol 1

```
const int buttonPin = 2;    // tlačítko je připojeno na pin D2
const int ledPin = 13;      // LED je připojena na pin D13
int buttonState;            // aktuální stav tlačítka
int lastButtonState = LOW;  // předchozí stav tlačítka
int ledState = LOW;         // stav LED

unsigned long lastDebounceTime = 0;  // poslední doba odskoku

void setup() {
  pinMode(buttonPin, INPUT_PULLUP); // inicializace pinu tlačítka jako vstup s
pull-up rezistorem
  pinMode(ledPin, OUTPUT);          // inicializace pinu LED jako výstup
  digitalWrite(ledPin, ledState);   // nastavení výchozího stavu LED
}

void loop() {
  // čtení stavu tlačítka
  int reading = digitalRead(buttonPin);

  // kontrola, zda se stav tlačítka změnil
  if (reading != lastButtonState) {
    // reset času poslední změny
    lastDebounceTime = millis();
  }

  // pokud je čas od poslední změny větší než zpoždění pro odstranění odskoku
  if ((millis() - lastDebounceTime) > 50) { //50ms pro ignoraci zákmitů
    // pokud se stav tlačítka liší od stavu LED
    if (reading != buttonState) {
      buttonState = reading;

      // pokud bylo tlačítko stisknuto (změna na LOW při použití pull-up
rezistoru)
      if (buttonState == LOW) {
        // změna stavu LED
        ledState = !ledState;
      }
    }
  }

  digitalWrite(ledPin, ledState);

  // uložení stavu tlačítka pro příští porovnání
  lastButtonState = reading;
}
```

Obrázek 34 Programové řešení úlohy 4.2 - úkol 2

4.3 Praktické implementace s periferiemi

Úlohy v této kapitole jsou zaměřeny na práci se vstupními a výstupními periferiemi. Jedná se o pro studenty zábavnější a komplexnější úlohy.

4.3.1 Zavlažovací systém

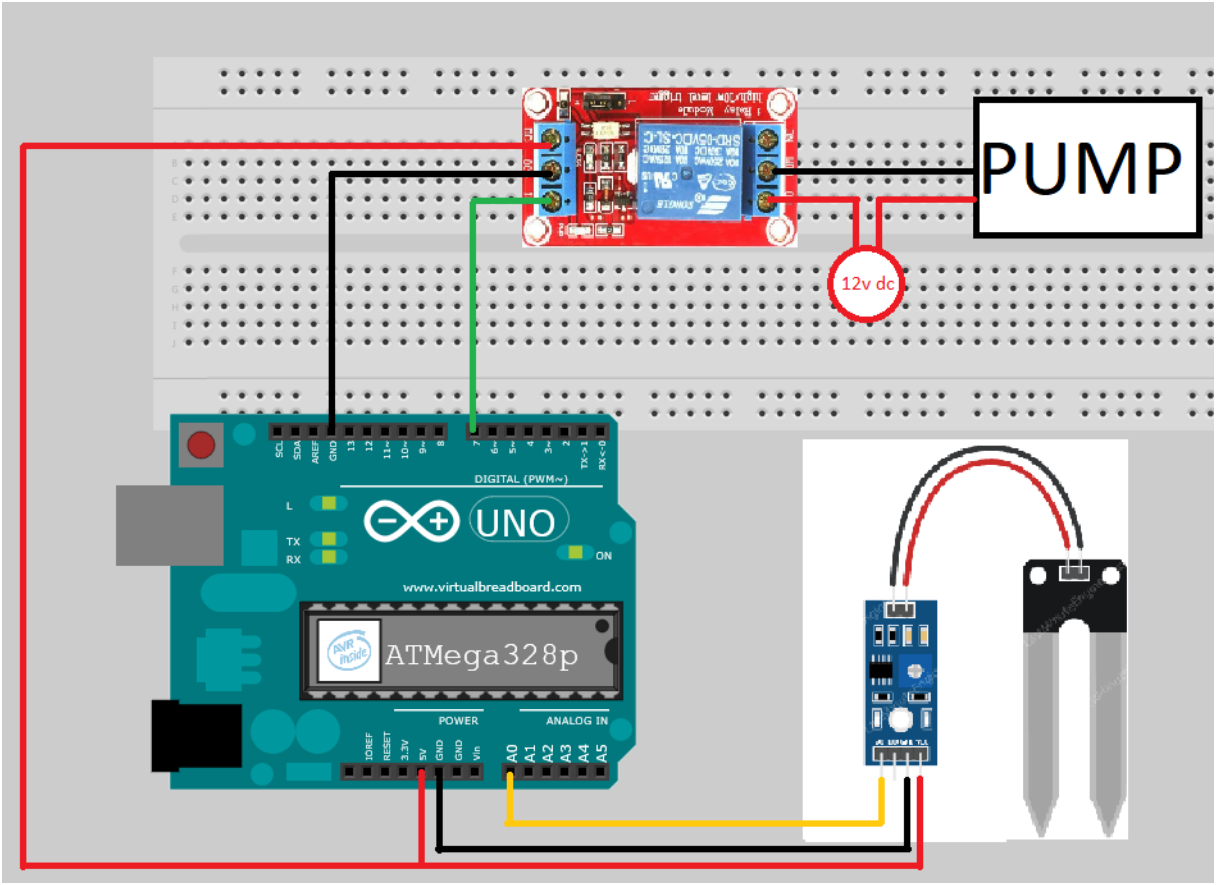
4.3.1.1 Potřebný hw

Základní set, půdní vlhkoměr, relé modul, čerpadlo na vodu.

4.3.1.2 Zadání

Zrealizujte zapojení níže a vytvořte jednoduchý program pro zavlažovací systém. Program jednou za hodinu zkontroluje vlhkost půdy. Použijte analogový výstup senzoru. Pokud bude analogová hodnota získaná ze senzoru menší jak 650, lze usoudit, že je půda dostatečně vlhká. Pokud nastane opak (analog. hodnota >= 650), zajistěte sepnutí relé, popř. čerpadla. Relé (čerpadlo) se vypne v okamžiku, kdy senzor vlhkosti vrací hodnotu 350 a menší.

4.3.1.3 Zapojení



Obrázek 35 Zapojení úlohy 4.3.1

4.3.1.4 Jak řešit

Program, který je řešením obsahuje na svém začátku inicializaci digitálního pinu jako výstup a analogového jako vstup. Dále je zde definována konstanta pro interval kontroly a proměnná pro sledování uplynulého času. Tělo smyčky *loop()* začíná definicí proměnné, jež získá aktuální čas od začátku běhu programu. Dále se pomocí *analogRead*, přečte analogová hodnota získaná ze senzoru vlhkosti půdy. Následuje podmínka, která kontroluje, zda od posledního zalití (poslání logické 1 na digitální výstup) uběhl stanovený časový interval 1 hodina, dále se kontroluje že hodnota získaná ze senzoru je větší nebo rovno než 650, to znamená, že půda je natolik suchá, aby se mohla zalít. Zároveň se kontroluje i skutečnost, že pin již není nastaven na HIGH. V případě, že je tato podmínka splněna, sepne se čerpadlo a aktualizuje se čas posledního zalití. Po této podmínce následuje *else-if* jež se vyhodnotí pravdivě, pokud je půda dostatečně zalitá, tj. hodnota se senzoru je menší nebo rovno 350 a zároveň je čerpadlo sepnuta. Po úspěšném vyhodnocení se čerpadlo vypne pomocí *digitalWrite.*

4.3.1.5 Řešení v jazyce Wiring

```
const int soilMoisturePin = A0;  // Senzor vlhkosti půdy připojený k pinu A0
const int relayPin = 7;          // Relé (čerpadlo) připojeno k pinu 7
const unsigned long checkInterval = 3600000;  // Interval kontroly vlhkosti (1
hodina = 3600000 milisekund)

unsigned long previousMillis = 0;  // Proměnná pro sledování uplynulého času

void setup() {
  pinMode(soilMoisturePin, INPUT);  // Nastaví analogový pin jako vstup
  pinMode(relayPin, OUTPUT);        // Nastaví pin relé jako výstup
  digitalWrite(relayPin, LOW);      // Zajistí, že relé je vypnuté při startu
}

void loop() {
    unsigned long currentMillis = millis();
    int soilMoistureValue = analogRead(soilMoisturePin);  // Přečte hodnotu ze
senzoru

    if (soilMoistureValue >= 650 && digitalRead(relayPin) == LOW &&
currentMillis - previousMillis >= checkInterval) {
      digitalWrite(relayPin, HIGH);  // Zapne relé (čerpadlo), pokud je půda
suchá a od posledního zalití uběhla alespoň hodina
      previousMillis = currentMillis;  // Aktualizace času poslední kontroly
    }
    else if (soilMoistureValue <= 350 && digitalRead(relayPin) == HIGH) {
      digitalWrite(relayPin, LOW);   // Vypne relé (čerpadlo), pokud je půda
dostatečně vlhká
    }
  }
```

Obrázek 36 Programové řešení úlohy 4.3.1

4.3.2 Chytré otevírání dveří

4.3.2.1 Potřebný hw

Základní kit, sada RFID modul RC522 a chipové karty nebo klíčenky, relé modul (popř. přímo elektromagnetický zámek)

4.3.2.2 Zadání

Naimportujte si do svého IDE knihovnu MFRC522, návod, jak na to naleznete zde: [5] .

Jako v předchozích úlohách, opět zapojte komponenty podle přiloženého schématu. Dále doplňte níže přiložený program. Ten v současné době přečte a vypíše do konzole osmimístnou adresu chipu. Spusťte program a naskenujte svůj chip, zjistíte tak jeho adresu.

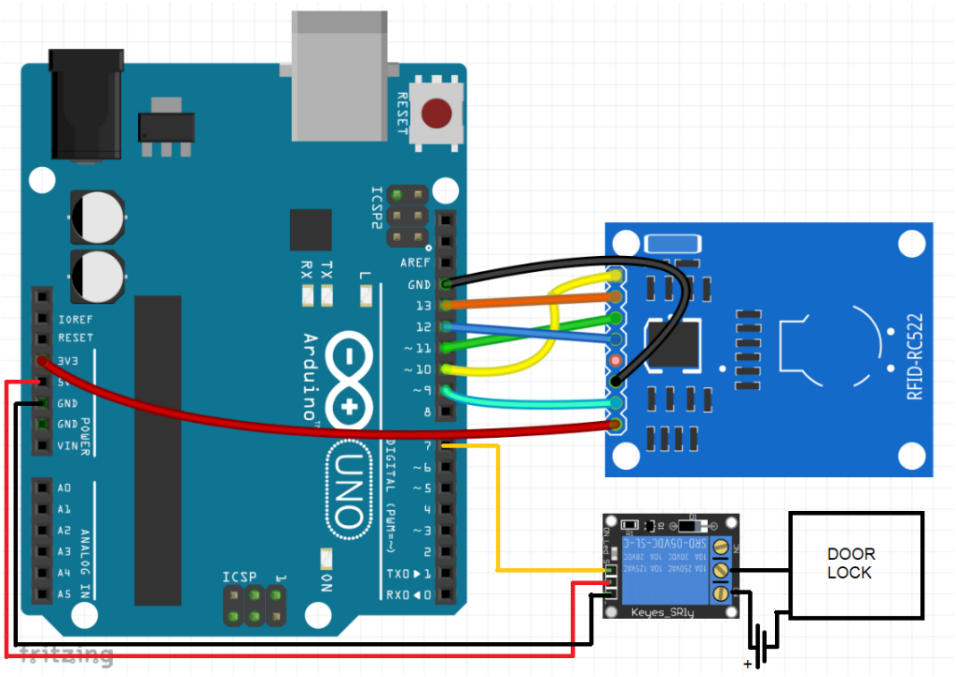
Do kódu doplňte pole, ve kterém budete uchovávat adresy povolených chipů. Dále doplňte inicializaci výstupu d7 pro relé. Po načtení chipu se zjistí, zda jeho adresa odpovídá nějakému v poli. Pokud ano, vypíše se do konzole „Přístup povolen“ a relé se sepne na 5 vteřin. V opačném případě se vypíše „Přístup zamítnut“ a relé se nesepne.

Kód k doplnění:

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Vytvoření instance MFRC522.

void setup() {
  Serial.begin(9600); // Inicializace sériové komunikace.
  SPI.begin();        // Inicializace SPI sběrnice.
  mfrc522.PCD_Init(); // Inicializace čtečky.
}

void loop() {
  // Hledání nové karty.
  if ( ! mfrc522.PICC_IsNewCardPresent())
    return;

  // Vybrání jedné ze zjištěných karet.
  if ( ! mfrc522.PICC_ReadCardSerial())
    return;

  // Zobrazení UID čtené karty.
  Serial.print("Card UID:");
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();

  // Ukončení čtení karty.
  mfrc522.PICC_HaltA();
}
```

Obrázek 37 Kód k doplnění u úlohy 4.3.2

4.3.2.3 Zapojení



Obrázek 38 Zapojení úlohy 4.3.2 [6]

4.3.2.4 Jak řešit

Pro vyřešení této úlohy je postup následující. Nejprve je nutné si definovat a inicializovat digitální pin jako výstup. Jedná se o přidání direktivu *#define*. Jedná se o něco jako konstantu s globálním dosahem, oproti ní má však výhodu v tom, že není uložena v paměti a nezabírá v ní tak místo. Během kompilace se přímo nahrazuje konkrétní hodnotou. Dále je zde vytvořeno dvourozměrné konstantní pole s globálním dosahem typu *byte*. Každá řádka má obsahovat 4 členy. Tento jeden řádek uchovává hexadecimální adresu chipu (V ukázce řešení tj.: 03 C9 B2 0D). Adresa je tedy tvořena čtyřmi bajty. Část kódu, která kontroluje, zda se adresa přiloženého chipů nachází v seznamu povolených je implementována na konci smyčky *loop*. Je nutné počítat s tím, že se v seznamu povolených bude nacházet více jak jeden chip, a proto se musí procházet po řádcích celé toto pole. Tento fakt je implementován pomocí smyčky *for*. Ta se opakuje tolikrát, kolik se nachází řádků v poli. Počet řádků je vypočítán jako podíl celkové velikosti pole v bajtech lomeno velikost jednoho řádku v bajtech. Kontrola toho, zda se daná adresa v poli shoduje s adresou přiloženého chipu je provedena v *if-else* větvi. Podmínka obsahuje funkci *memcmp*, která porovnává po bajtech zadané čtyřbajtové bloky. Pokud jsou shodné, tak tato funkce vrací nula a dále se stane podmínka splněnou a jsou

splněny příslušné kroky popsané v zadání. Pokud není splněna a vstoupí se do *else* větve, tak se vypíše „Přístup zamítnut“ do konzole dle zadání.

4.3.2.5 Řešení v jazyce Wiring

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
#define RELAY_PIN 7
MFRC522 mfrc522(SS_PIN, RST_PIN);

//pole povolených chipů
const byte allowedChips[][4] = {
  {0x03, 0xC9, 0xB2, 0x0D}
};

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();

  pinMode(RELAY_PIN, OUTPUT);
}

void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent())
    return;
  if ( ! mfrc522.PICC_ReadCardSerial())
    return;

  Serial.print("Card UID:");
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();

  for (byte i = 0; i < sizeof(allowedChips) / sizeof(allowedChips[0]); i++) {
    if (memcmp(allowedChips[i], mfrc522.uid.uidByte, 4) == 0) {
      Serial.println("Přístup povolen");
      digitalWrite(RELAY_PIN, HIGH); // Zapnutí relé.
      delay(10000);                  // Počkání 10 sekund.
      digitalWrite(RELAY_PIN, LOW);  // Vypnutí relé.
      break;
    }else{
      Serial.println("Přístup zamítnut");
    }
  }
  mfrc522.PICC_HaltA();
}
```

Obrázek 39 Programové řešení úlohy 4.3.2

4.3.3 Hodiny – I2C sběrnice

4.3.3.1 Potřebný hw

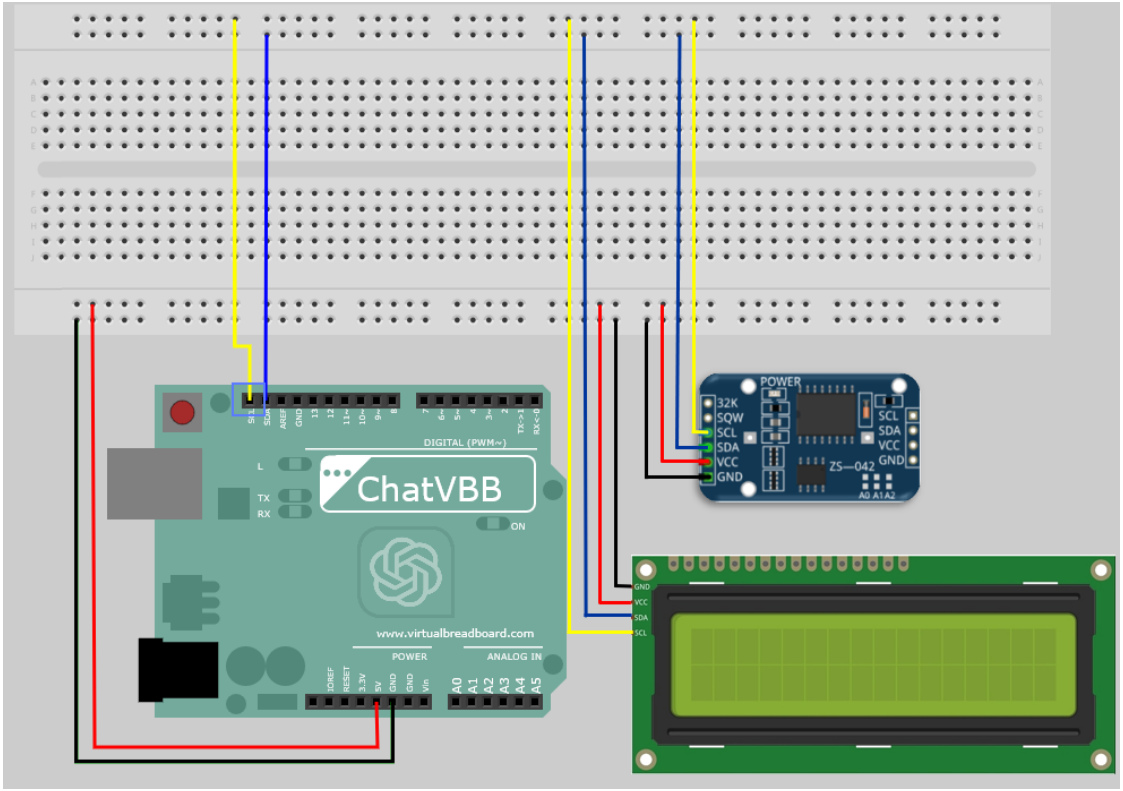
LCD 16x2 s I2C sběrnicí, RTC

4.3.3.2 Zadání

Nejprve nainstalujte knihovnu Wire, LiquidCrystal_I2C a RTClib.

Vyzkoušejte si práci s I2C sběrnicí. Použijte ji k připojení displeje a modulu reálného času. Zapojení viz níže. Realizujte výpis aktuálního času na lcd displej každou 1 vteřinu. V případě, že RTC modul nebude nalezen, vhodně ošetřete výpis na lcd a zajistěte, aby se program nepokoušel číst čas.

4.3.3.3 Zapojení



Obrázek 40 Zapojení úlohy 4.3.3

4.3.3.4 Jak řešit

K vyřešení této úlohy je nutné nejprve importovat knihovny zmíněné v zadání. Následně je nezbytnou součástí kódu provedení instance displeje a modulu reálného času. Jelikož se jedná o sběrnici, mají jednotlivá zařízení svou adresu, u displeje to bývá ve většině případů 0x27. Další dvě číselné hodnoty udávají počet znaků v řádku a počet řádků displeje. V *setup* se provede inicializace rozhraní k displeji pomocí *lcd.begin*. U displeje se též musí zapnout

podsvícení a následně se provede inicializace rtc modulu. Ta je vložena v *if* podmínce a je znegována. To znamená, že když je inicializace neúspěšná (např. modul není připojen či jiná chyba), tak rtc.begin vrátí *false*. Aby se podmínka provedla právě při tomto stavu, je nutné mít zde zmíněnou negaci. Vypíše se text do konzole a to, aby program neskočil do *loop* metody, se zajistí pomocí *while(1)*, který se provádí v podstatě do nekonečna. V *loop* sekci kódu se z *rtc.now* zjistí aktuální čas. Následuje posloupnost příkazů pro výpis jednotlivých digitů hodin a pauza 1000ms pro aktualizaci času na displeji.

4.3.3.5 Řešení v jazyce Wiring

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

// Instance zařízení
LiquidCrystal_I2C lcd(0x27, 16, 2);
RTC_DS3231 rtc;

void setup() {
  lcd.begin(); // Inicializace sběrnice
  lcd.backlight(); // zapnutí podsvícení lcd
  if (!rtc.begin()) {
    lcd.print("Nelze najit RTC");
    while (1);
  }
}

void loop() {
  DateTime now = rtc.now(); // Přečtení aktuálního času z RTC

  // Zobrazení času na LCD
  lcd.setCursor(0, 0);
  lcd.print(now.hour(), DEC);
  lcd.print(':');
  lcd.print(now.minute(), DEC);
  lcd.print(':');
  lcd.print(now.second(), DEC);

  delay(1000); // Aktualizace času každou vteřinu
}
```

Obrázek 41 Programové řešení úlohy 4.3.3

4.3.4 Ovládání servomotoru přes gyroskop

4.3.4.1 Potřebný hw

Servomotor kompatibilní s Arduinem, model gyroskopu + akcelerometru MPU-6050 s podporou I2C

4.3.4.2 Zadání

V této úloze si vyzkoušíte praktickou implementaci zejména z odvětví robotiky. Pomocí snímání hodnoty osy Y budete ovládat otočení hřídele servomotoru. (To lze v praxi využít při vzdáleném ovládání robotické paže atd.).

Nejprve si naimportujte knihovny: Wire, MPU6050, Servo

Dále realizujte zapojení přiložené k této úloze. Po-té převezměte následující kód a doplňte ho. Nyní kód obsahuje základní připojení a inicializace knihoven, servomotoru a gyroskopu. Vy přidejte kód, který zajistí výpis aktuální hodnoty ay do konzole. Ay bude dosahovat hodnoty v intervalu od -32768 do 32767. Běžným otočením gyroskopu však dosahuje nejvíce hodnot od -16000 do 16000. Zajistěte tak, aby při:

- Ay < -16000 byla pozice servomotoru rovna 0°
- -16000 <= Ay <= 16000 byla pozice rovnoměrně rozdělena mezi 0 až 180° (pomocí funkce map)
- Ay > 16000 byla pozice rovna 180°

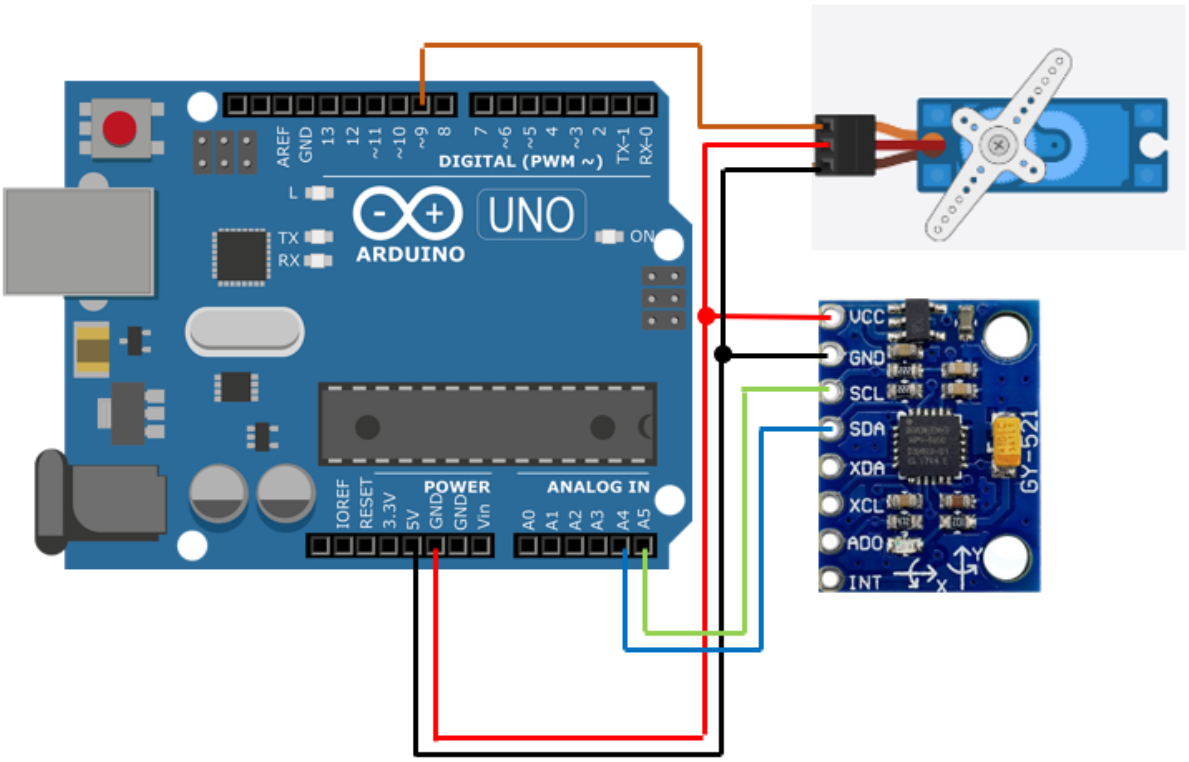
To zajistí rovnoměrné otáčení. Nyní si ale všimněte, že i když držíte senzor téměř nehybně, tak servomotor stále pracuje a o nepatrný úhel hýbe osou na jednu a na druhou stanu. To ošetříte tak, že budete zkoumat, zda rozdíl mezi poslední a aktuální hodnotou ay je větší jak 2000.

Kód k doplnění:

```
#include <Wire.h>
#include <MPU6050.h>
#include <Servo.h>

MPU6050 mpu;
Servo myServo;
int ax, ay, az;
int gx, gy, gz;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
  myServo.attach(9);

  if (!mpu.testConnection()) {
    Serial.println("MPU6050 není připojen");
    while (1);
  }
}

void loop() {
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  //TODO výpis hodnoty

  //TODO ovládání serva podle ay

  //TODO optimalizace pohybu serva a oprava cukání

  delay(100); // Krátká pauza, aby bylo možné sledovat výstup
}
```

Obrázek 42 Kód k doplnění u úlohy 4.3.4

4.3.4.3 Zapojení



Obrázek 43 Zapojení úlohy 4.3.4

4.3.4.4 Jak řešit

Při řešení této úlohy je dobré začít tím nejjednodušším, a to je výpis gyroskopické hodnoty náklonu osy do konzole. Získání hodnot z gyroskopu je již zajištěno. Proto je sekci *loop* po získání hodnot realizován výpis hodnoty ay do konzole pomocí *Serial.print(" Y: "); Serial.println(ay).* Vytvořena je dále proměnná typu int, jež má uchovávat hodnotu od 0 do 180, což reprezentuje stupeň natočení hřídele servomotoru ve stupních. O hodnotě, jež se do této proměnné uloží, se rozhoduje v nadcházejícím *if-else if-else* větvení. Podmínky jsou stanoveny podle zadání. To znamená, že zanedbáváme příliš velký náklon gyroskopu. V *else* větvi je přepočtena analogová hodnota z gyroskopu na stupně natočení pomocí funkce *map*. Ta převádí rovnoměrně číslo z jednoho rozsahu do úměrně velikého čísla z druhého rozsahu.

Posledním krokem je ošetření drobných zákmitů gyroskopu a zároveň vyslání příkazu k natočení hřídele. V rámci 100 ms po kterých se provádí kontrola, zda se hodnota gyroskopu nezměnila, se musí ukládat vždy předchozí hodnota gyroskopu. Natočení hřídele se provede až tehdy, když je rozdíl náklonů v absolutní hodnotě větší než hodnota určená zadáním (2000). Zároveň se musí vždy aktualizovat předchozí hodnota na aktuální.

4.3.4.5 Řešení v jazyce Wiring

```
#include <Wire.h>
#include <MPU6050.h>
#include <Servo.h>

MPU6050 mpu;
Servo myServo;
int ax, ay, az;
int gx, gy, gz;
int lastAy = 0; // Přidáme proměnnou pro sledování předchozí hodnoty ay

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
  myServo.attach(9);

  if (!mpu.testConnection()) {
    Serial.println("MPU6050 není připojen");
    while (1);
  }
}

void loop() {
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  Serial.print(" Y: "); Serial.println(ay);

  int servoPos;
  if (ay <= -16000) {
    servoPos = 0;
  } else if (ay >= 16000) {
    servoPos = 180;
  } else {
    // Převede hodnotu gyroskopu Y na rozsah 0-180 pro servomotor
    servoPos = map(ay, -16000, 16000, 0, 180);
  }

  // Kontroluje, zda se hodnota změnila o 2000 nebo více
  if (abs(ay - lastAy) > 2000 ) {
    myServo.write(servoPos);
    lastAy = ay; // Uložení aktuální hodnoty ay pro porovnání v dalším cyklu
  }

  delay(100); }
```

Obrázek 44 Programové řešení úlohy 4.3.4

4.4 PWM

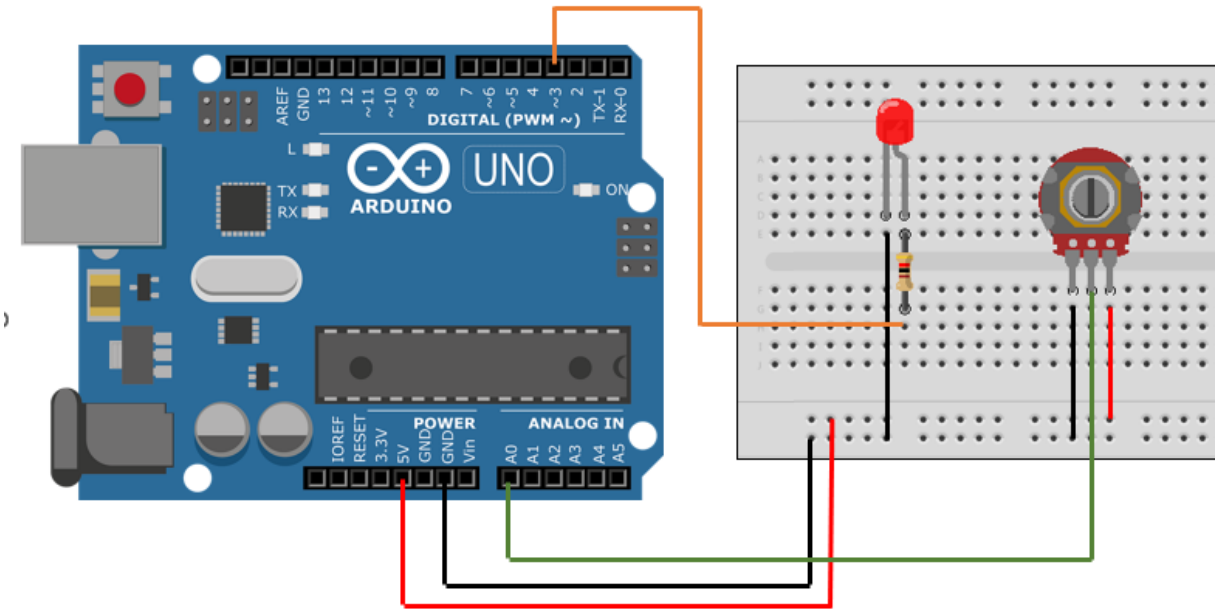
4.4.1 Potřebný hw

Dioda typu LED 5mm, předřadný rezistor 220Ω, potenciometr 10K Ω

4.4.2 Zadání

Zrealizujte zapojení níže a využijte funkce pulzní šířkové modulace k regulaci svítivosti LED v závislosti na hodnotě analogového vstupu z potenciometru (Hodnota by měla být v intervalu od 0 do 1023). Do konzole každých 100 ms vypisujte hodnoty jak z analogového vstupu, tak i hodnoty, jež používáte pro PWM. Pro ovládání tímto způsobem použijte funkce *analogWrite* a *analogRead.*

4.4.3 Zapojení



Obrázek 45 Zapojení úlohy 4.4

4.4.4 Jak řešit

Nejprve je pro řešení úlohy nutné, definovat si vstupy a výstupy. Důležité je, že digitální pin, který je využit, podporuje pwm, neboli pulzní šířkovou modulaci. Pak následuje započetí sériové komunikace. V *loop* funkci se čte analogová hodnota získaná z potenciometru a pomocí funkce *map* se převádí na hodnotu z intervalu od 0 do 255. Práce tyto hodnoty lze pro pwm využít. Jde o to, že při stejném napětí, lze měnit výkon výstupních zařízení pomocí šířky vysílaných impulzů. Tímto principem se v praxi můžou řídit například některé DC motory.

Zde je praxe předvedena na LED a lze si povšimnout, jak led v závislosti na otočení hřídele potenciometru mění svou svítivost.

4.4.5 Řešení v jazyce Wiring

```
int potPin = A0; // Analogový pin, ke kterému je připojen potenciometr
int ledPin = 3;  // Digitální PWM pin, ke kterému je připojena LED

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int potValue = analogRead(potPin);  // Přečte hodnotu z potenciometru (0 -
1023)
  int ledValue = map(potValue, 0, 1023, 0, 255); // Přemapuje hodnotu na
rozsah PWM (0 - 255)

  analogWrite(ledPin, ledValue);  // Nastaví úroveň svícení LED pomocí PWM

  Serial.print("Potenciometer: ");
  Serial.print(potValue);
  Serial.print(" => LED: ");
  Serial.println(ledValue);

  delay(10);  // Krátká pauza pro stabilitu
}
```

Obrázek 46 Programové řešení úlohy 4.4

4.5 Práce s pamětí EEPROM

4.5.1 Potřebný hw

Pouze základní vybavení

4.5.2 Zadání

Vyzkoušejte si přístup k paměti EEPROM pomocí funkcí stejnojmenné knihovny. Do paměti nejprve zapište libovolnou byte hodnotu na adresu 0. Poté ji přečtěte, vypište do konzole, vypočítejte efektivním algoritmem její faktoriál a tuto hodnotu také vypište. Vzhledem k tomu, že hodnota faktoriálu roste v násobcích a datový typ byte pro výsledek nemusí stačit, zvolte vhodný datový typ. Program provádějte pouze jednou po spuštění.

4.5.3 Zapojení

Pouze deska Arduino Uno připojená přes USB do PC

4.5.4 Jak řešit

Jelikož se má program provádět jen jednou, bude se kód nacházet jen v metodě *setup*. Nejprve se provede inicializace sériové komunikace, následně se definuje proměnná, jež ukládá informaci o tom, na jakou adresu v eeprom paměti chceme přistupovat. Následně je definovaná proměnná typu byte, jež uchovává hodnotu pro zápis (v tomto případě je to číslo 5). Zápis se provede jednoduše pomocí příkazu *EEPROM.write*. Parametry této funkce jsou adresa kam se má zapsat a hodnota k zapsání. Čtení je intuitivně podobné. Následuje výpis do konzole. Poté je zde definována hodnota typu unsigned long na 1. Unsigned znamená, že odpadá možnost zapsat do proměnné zápornou hodnotu, zato ale zdvojnásobí svůj rozsah v kladném intervalu. Dále je zde jeden z efektivních algoritmů pro výpočet faktoriálu. Jedná se cyklus *for*. Ten začíná na čísle 1 a pokračuje až do hodnoty čísla přečteného z paměti. Při každém opakování se pak stávající hodnota proměnné *faktoriál* vynásobí indexem aktuální iterace. Následuje už jen výpis do konzole konečné hodnoty faktoriálu.

4.5.5 Řešení v jazyce Wiring

```
#include <EEPROM.h>

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

  int address = 0; // Adresa v EEPROM
  byte valueToWrite = 5;
  EEPROM.write(address, valueToWrite); // Zápis hodnoty do EEPROM

  byte valueRead = EEPROM.read(address); // Čtení hodnoty z EEPROM
  Serial.print("Hodnota přečtená z EEPROM: ");
  Serial.println(valueRead);

  // Výpočet faktoriálu
  unsigned long factorial = 1;
  for (int i = 1; i <= valueRead; i++) {
    factorial *= i;
  }

  Serial.print("Faktoriál ");
  Serial.print(valueRead);
  Serial.print(" je ");
  Serial.println(factorial);
}

void loop() {  // Zde se nic neděje.}
```

Obrázek 47 Programové řešení úlohy 4.5

4.6 Obsluha přerušení

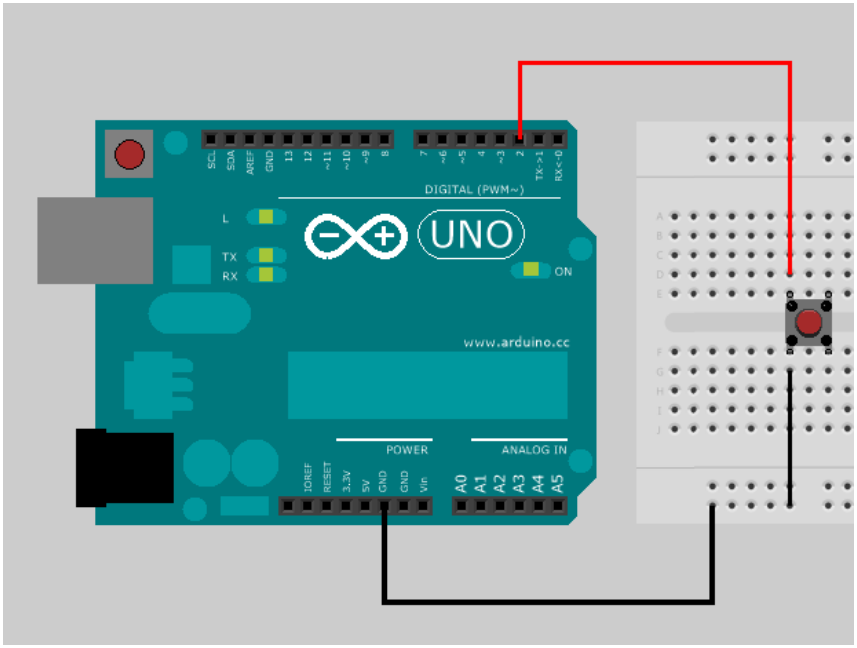
4.6.1 Potřebný hw

tlačítko

4.6.2 Zadání

Zapojte hardware dle zapojení. Vytvořte cyklus, který opakovaně vypisuje čísla od 1 do 10 do konzole. Aby bylo sledování výpisu přívětivější, realizujte čekání programu po dobu 350ms po každém výpisu. Vyřešte obsluhu přerušení při příchodu náběžné hrany na pin D2. Při stisku tlačítka se vypíše do konzole „PŘERUŠENÍ“.

4.6.3 Zapojení



Obrázek 48 Zapojení úlohy 4.6

4.6.4 Jak řešit

Pro vyřešení této úlohy obsahuje kód funkcionalitu pro inicializaci sériové komunikace a inicializaci tlačítka v režimu INPUT_PULLUP. Pro simulaci běžné rutinní práce procesoru se v *loop* funkci opakuje pomocí *for* cyklu počítání od 1 do 10 přičemž výpis jedné číslice trvá 350 ms. Při programovém řešení pomocí obyčejné podmínky při stisku tlačítka, by musela být podmínka například za for cyklem a reakce na stisk tlačítka by nebyla okamžitá. Právě proto existuje obsluha přerušení. V dolní části kódu mimo sekci *loop* je vytvořena vlastní funkce s návratovým typem void, jež simuluje práci prováděnou procesorem po stisku tlačítka. Propojení stisku tlačítka a provedení této obslužné funkce se nachází ještě v metodě *setup*. Provedeno je to tímto příkazem *attachInterrupt(digitalPinToInterrupt(2),interrupt, RISING).*

V prvním parametru se uvádí identifikátor přerušení. Zde je funkce *digitalPinToInterrupt(2),* která převádí číslo pinu právě na tento identifikátor. Dále je zde uveden název obslužné prováděné funkce. A poslední parametr stanovuje, kdy má být přerušení provedeno. Zde je použito RISING, to znamená, že je přerušení vyvoláno v okamžiku přechodu signálu z LOW na HIGH.

4.6.5 Řešení v jazyce Wiring

```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2),interrupt, RISING);
}

void loop() {
  for(int i=1; i<=10; i++){
    Serial.println(i);
    delay(350);
  }
}

void interrupt(){
  Serial.println("PŘERUŠENÍ");
  delay(350);
}
```

Obrázek 49 Programové řešení úlohy 4.6

4.7 IO práce se souborem, zápis na SD kartu

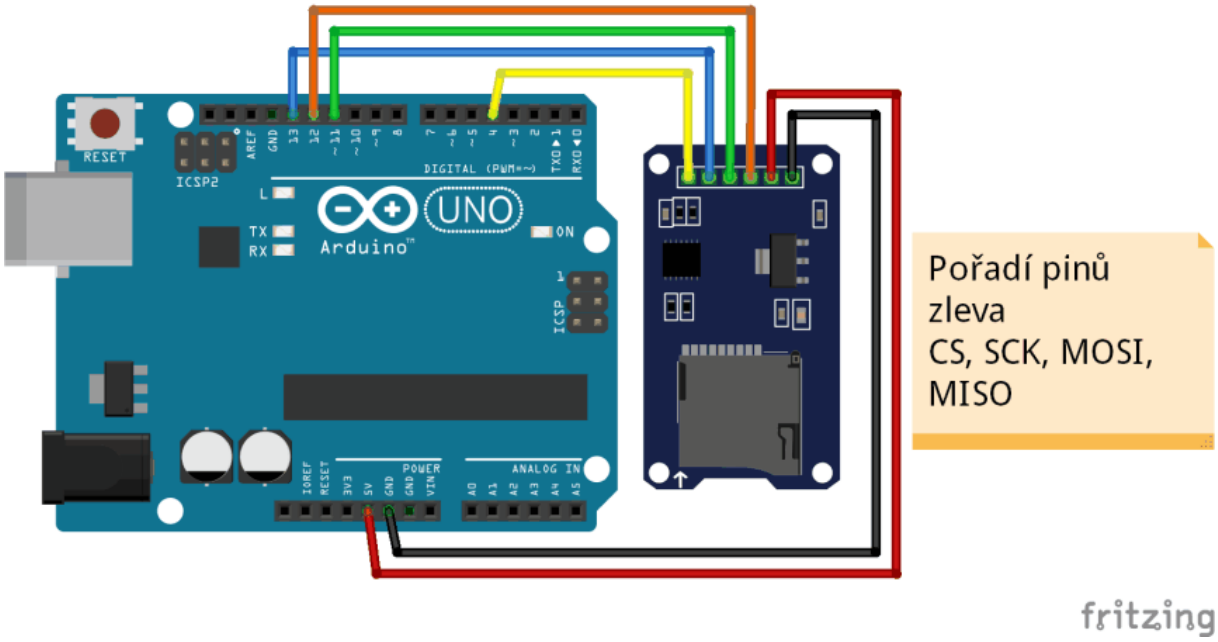
4.7.1 Potřebný hw

Čtečka sd karet SPI

4.7.2 Zadání

Opět realizujte zapojení k této úloze. Nejprve zformátujte svoji SD kartu tak, aby systém souborů byl FAT32 a velikost alokační jednotky byla 32kB. Vytvořte program, který pomocí knihovny SD vytvoří nebo otevře soubor *test.txt* a zapíše do něj testovací textový řetězec. Následně realizujte čtení a výpis do konzole. Po celou dobu programu do konzole vhodně informujte o aktuálním stavu práce s SD kartou. V případě nenalezení karty či jiných chyb výjimky ošetřete. Využijte třídu File pro uchování souboru, pro vytvoření pak metody z knihovny SD, konkrétně *SD.open.*

4.7.3 Zapojení



Obrázek 50 Zapojení úlohy 4.7 [7]

4.7.4 Jak řešit

Po úspěšném naformátování sd karty a realizaci zapojení, lze pokračovat v řešení úlohy programově. Nejprve je vytvořena proměnná datového typu File a konstanta uchovávající číslo pinu pro chipSelect. Je započata sériová komunikace a proveden výpis. Poté se v podmínce provádí inicializace komunikace s paměťovou kartu. V případě nezdaru se program zastaví na *while (1)* a ještě předtím vypíše chybovou hlášku do konzole. V případě úspěchu vypíše *"initialization done."*. Následně se soubor otevře pro režim zápisu a pokud existuje, lze do něj zapsat pomocí *myFile.println("Testing 1, 2, 3.")*. Nelze opomenout, že je žádoucí dokument zavřít pomocí *close*. Čtení ze souboru lze realizovat pomocí *SD.open*, jež přijímá jako parametr název souboru. Pokud nenastane chyba, lze číst soubor až do konce (dokud je co číst). To je zajištěno *while* cyklem spolu s *myFile.available()*. Opět se soubor musí po konci práce s ním zavřít.

4.7.5 Řešení v jazyce Wiring

```
#include <SPI.h>
#include <SD.h>

File myFile;
const int chipSelect = 4;

void setup() {
  Serial.begin(9600);
  Serial.print("Initializing SD card...");

  if (!SD.begin(chipSelect)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");

  myFile = SD.open("test.txt", FILE_WRITE);

  if (myFile) {
    Serial.println("Writing to test.txt...");
    myFile.println("Testing 1, 2, 3.");
    myFile.close();
    Serial.println("done.");
  } else {
    Serial.println("error opening test.txt");
  }



// přečtění souboru:
  Serial.println("Reading...");
  myFile = SD.open("test.txt");
  // pokud se soubor načte a otevře, tak:
  if (myFile)
  {
    Serial.println("file contains: ");

    // čti ze souboru, dokud je co
    while (myFile.available()){
      Serial.write(myFile.read());
    }
    myFile.close();
  }
  else
  {
    Serial.println("Error opening txt file");
  }
```

```
}

void loop() {
  // nic se zde neděje
}
```

Obrázek 51 Programové řešení úlohy 4.7

ZÁVĚR

Jednodeskové počítače jsou v dnešní době poměrně trendem. Edukativní dostupné počítače jsou používány hlavně v hobby segmentu, ale transformují se i do profesionálního prostředí a o to více i do edukace.

To je i důvod, proč cílem této práce bylo seznámení s jednodeskovými počítači, s jejich hardwarem, periferiemi, technologiemi a používaným softwarem. Práce prezentuje jednodeskové počítače jako výukové pomůcky a zabývá se více aspekty. Práce je rozdělena do čtyř sekcí.

V první části se zabývá tím, jak v současné době vypadá výuka elektrotechnických a IT oborů na středních průmyslových školách jak v ČR, tak v zahraničí. Dále představuje vhodné prostředky pro výuku a nastiňuje jejich praktické využití a zabývá se i dostupností těchto pomůcek.

Druhá a třetí část je též teoretická, ale její znalost je poměrně klíčová ke zvládnutí praktické práce s mikrokontroléry na jednodeskových počítačích. Jedná se o popis desky Arduino Uno a základy jazyka Wiring.

Poslední část je částí praktickou. Nalézá se zde deset úloh k procvičení jak základních programovacích úkonů, tak i pokročilejších. Po projití touto částí, je předpoklad, že student bude lépe chápat a ovládat z části technologie používané i v praxi.

Lze se samozřejmě v tomto směru vzdělávat více. K tomu je vhodné využít oficiální dokumentace anebo existuje řada knížek a webů.

POUŽITÁ LITERATURA

[1] IOT Starter kity. DRÁTEK.CZ [online]. [cit. 2024-01-18]. Dostupné z: https://dratek.cz/187-arduino-starter-kity/#sc=300

[2] OBDRŽÁLEK, David, Gottfried KOPPENSTEINER, Munir MERDAN, Richard BALOGH a Wilfried LEPUSCHITZ. Robotics in Education Methodologies and Technologies. Springer International Publishing, 2021. ISBN 978-3-030-67410-6.

[3] NOVÁK, Milan a Jiří PECH. Robotika pro střední školy: Programujeme Arduino. Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, 2020. ISBN 978-80-7394-786-6.

[4] SELECKÝ, Matúš. Arduino: Uživatelská příručka. Computer Press, 2016, 344 s. ISBN 978-80-251-4840-2.

[5] Arduino knihovny. DRÁTEK.CZ [online]. [cit. 2024-01-21]. Dostupné z: https://navody.dratek.cz/zaciname-s-arduinem/arduino-knihovny.html

[6] 27_rfid_ctecka_schema.png. In: DRÁTEK.CZ [online]. [cit. 2024-01-21]. Dostupné z: https://navody.dratek.cz/images/obr_clanky/27_rfid_ctecka/27_rfid_ctecka_schema.png

[7] Schema-zapojeni-ctecka-sd-karet-arduino-navody.png. In: Hwkitchen.cz [online]. [cit. 2024-01-24]. Dostupné z: https://cdn.myshoptet.com/usr/www.hwkitchen.cz/user/documents/upload/arduino-navody/modul-sd-karty-arduino-navody/schema-zapojeni-ctecka-sd-karet-arduino-navody.png

[8] Cena bastlířů. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2022, 21. 12. 2022 [cit. 2024-02-19]. Dostupné z: https://cs.wikipedia.org/wiki/Cena_bastl%C3%AD%C5%99%C5%AF?fbclid=IwAR3-qJ_GAKj0Z3GnZjccfC4Td6XGB9GfM3O3DVkfMN-LKW36mbr0qnT5Dgo

[9] Cena bastlířů [online]. [cit. 2024-02-19]. Dostupné z: https://www.facebook.com/100039213978485/posts/2079519668752887/

[10] Využití moderních technologií ve výuce fyziky na SŠ. Hradec Králové, 2022. Dostupné také z: https://theses.cz/id/z68et0/STAG97805.pdf. Diplomová práce. Univerzita Hradec Králové, Přírodovědecká fakulta, Katedra fyziky.

[11] High school students are learning to program with the Arduino IDE software. ResearchGate [online]. 2017 [cit. 2024-02-19]. Dostupné z:

https://www.researchgate.net/figure/High-school-students-are-learning-to-program-with-the-Arduino-IDE-software_fig2_321909218

[12] SPŠE - SŠ - studijní obory. Spše [online]. b.r. [cit. 2024-02-19]. Dostupné z: https://www.spse.cz/obory-ss.php

[13] 71z22cRPeeL.__AC_SY300_SX300_QL70_FMwebp_.jpg: Arduino Uno R3. In: Amazon [online]. © 1996-2024 [cit. 2024-02-21]. Dostupné z: https://m.media-amazon.com/images/I/71z22cRPeeL.__AC_SY300_SX300_QL70_FMwebp_.jpg

[14] Boards and Modules. Arduino.cc [online]. [cit. 2024-02-24]. Dostupné z: https://store.arduino.cc/collections/boards-modules

[15] SINGH, Rajesh, Anita GEHLOT, Bhupendra SINGH a Sushabhan CHOUDHURY. Arduino-Based Embedded Systems: Interfacing, Simulation, and LabVIEW GUI. CRC Press, 2017, 330 s. ISBN 9781315162881.

[16] AN INTRODUCTION TO ARDUINO UNO PINOUT: Arduino Uno Pinout Guide. In: Circuito.io [online]. b.r. [cit. 2024-02-26]. Dostupné z: https://www.circuito.io/blog/arduino-uno-pinout/

[17] Analogový signál. It-slovnik [online]. © 2008 - 2024 [cit. 2024-02-27]. Dostupné z: https://it-slovnik.cz/pojem/analogovy-signal

[18] Logic Levels. SPARKFUN ELECTRONICS. SparkFun [online]. 2013 [cit. 2024-02-28]. Dostupné z: https://learn.sparkfun.com/tutorials/logic-levels

[19] MECHATRONIKA, ČÁST 3 SENZORY. DPS Elektronika od A do Z [online]. 2022, 2022(3), 1 [cit. 2024-02-28]. Dostupné z: https://www.dps-az.cz/clanky/id:85889/z-aktualniho-vydani-mechatronika-cast-3-senzory

[20] [PIC programování] – 7. díl. TRÁVNÍČEK, Samuel, ed. Allcomp.cz [online]. 2018 [cit. 2024-02-28]. Dostupné z: https://blog.allcomp.cz/pic-programovani-7-dil/

[21] Základy digitálních potenciometrů a jak je používat. Vývoj.hw [online]. 2021 [cit. 2024-03-03]. Dostupné z: https://vyvoj.hw.cz/zaklady-digitalnich-potenciometru-a-jak-je-pouzivat.html

[22] Language Reference. Arduino.cc [online]. © 2024 [cit. 2024-03-06]. Dostupné z: https://www.arduino.cc/reference/en/?_gl=1*16cdso1*_ga*MjExNDEwMDEyLjE3MDU1NzI5NzU.*_ga_NEXN8H46L5*MTcwOTcxNTQwNC4xNi4xLjE3MDk3MTU0MTEuMC4wLjA.*_fplc*WTFFJTJGNEdyNEpIU3lyVmUlMkY0dVQ3SzNvc3pBSUpNS2ZzVyUyRkFhYlVEVlR6YUI5eFY4TGlDYlgzNm56WnlWOHFjdEVPdTQ0dll

3NGNzOTJvRkNITGcxJTJGamdIdDUlMkZlRkxHaGNaTnJXR2FJcVdnMzc1akh2cTBkZlk0V2VmSExndyUzRCUzRA..

[23] SERVOS EXPLAINED. Sparkfun [online]. b.r. [cit. 2024-03-11]. Dostupné z: https://www.sparkfun.com/servos