ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

VÝUKOVÁ STAVEBNICE ARDUINO TINYLAB NA VYSOKÉ ŠKOLE

BAKALÁŘSKÁ PRÁCE

Šimon Karban
Přírodovědná studia, obor Informatika se zaměřením na vzdělávání

Vedoucí práce: PhDr. Tomáš Jakeš, Ph.D.

Plzeň 2022

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 28. dubna 2022

vlastnoruční podpis

PODĚKOVÁNÍ

Děkuji PhDr. Tomáši Jakešovi, Ph.D. za odborné vedení práce, dobré rady i věcné připomínky při vypracování bakalářské práce.

Obsah

Úvod

Aktuální trendy informačních a komunikačních technologií směřují k IoT, digitalizaci a automatizaci práce. To vše jsou prvky nejen Průmyslu 4.0.

Jedna z reakcí škol je nákup hardwarových vývojových kitů, které umožňují žákům rozvíjet potřebné dovednosti v těchto oblastech. Tyto stavebnice se zaměřují na aktivním využití hardwarových prvků v praxi a tím poskytují bohaté zkušenosti uplatnitelné na dnešním trhu práce.

Hlavní motivací mojí práce je zvýšit zájem o toto odvětví pomocí ekonomicky a uživatelsky dostupných prostředků s širokou podporou. Pro tento účel využijeme prototypovací desku TinyLab postavenou na open-sourcovém jednodeskovém počítači Arduino.

Cílem práce je představit prototypovací stavebnici TinyLab, popsat její moduly, jejich význam, řízení a propojení, a sestavit pestrou prakticky orientovanou sadu výukových aktivit sloužících k pochopení jejich užití.

Práce je rozdělena do čtyř kapitol. V první se nejdříve seznámíme s vlastní vývojovou deskou TinyLab a Arduino, na kterém je postavena, popisem a funkcí jednotlivých modulů na stavebnici. V následující kapitole s doprovodným softwarem, vývojovým prostředím a programovacím jazykem pro Arduino.

Třetí kapitola je složena z dílčích úloh na bázi modulů, jejich aplikace v reálném životě, motivace k rozvoji z hlediska elektrotechniky a programování. Poslední kapitola kombinuje nasbírané poznatky v podobě komplexních úkolů využívajících více periferií a pokročilejších programovacích technikách. Veškeré zdrojové kódy jsou veřejně dostupné v repozitáři.

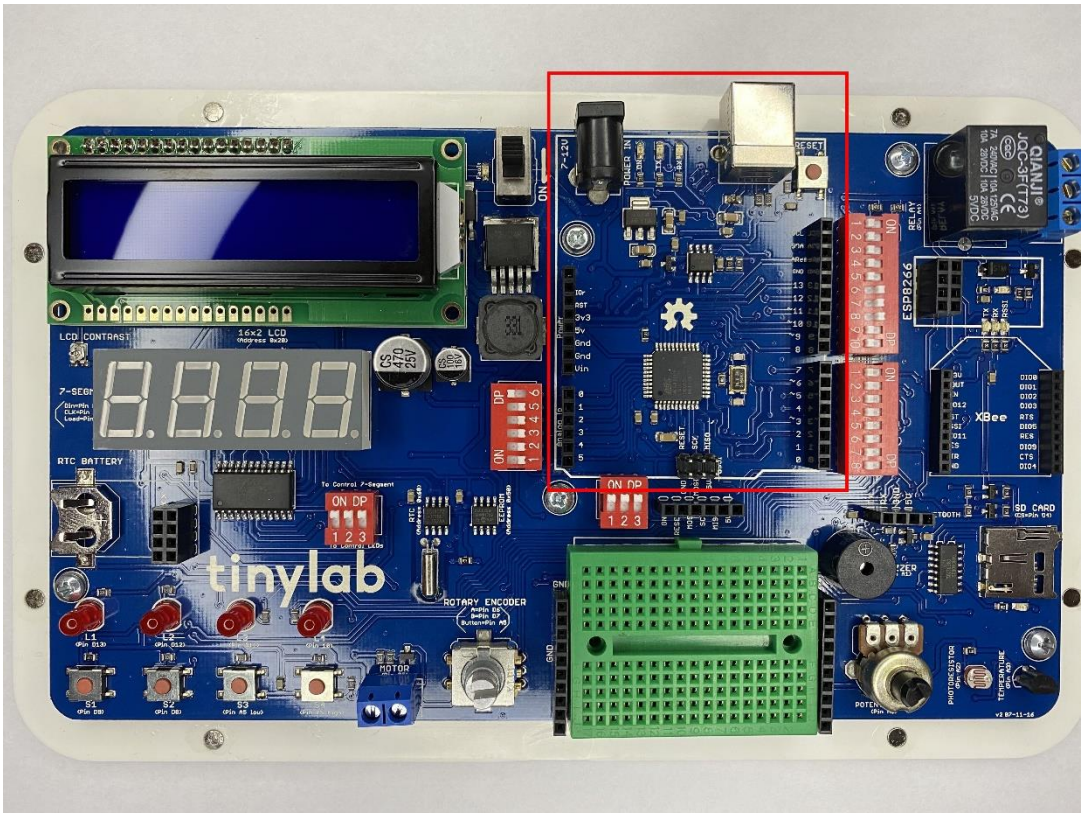
K utvrzení uplatnění v praxi jsou cvičení uvozena příklady využití v běžném životě. Úkoly postupně gradují v obtížnosti, jsou doplněny o postup řešení včetně odůvodnění kroků, cíli, samotným kódem a možnostmi rozšíření.

1 TINYLAB

Prototypovací vývojová deska TinyLab vznikla jako crowdfundingový[1] projekt na serveru Kickstarter týmem Sixfab. Jeho záměrem bylo integrovat nejpoužívanější periferní zařízení a ovládací prvky přímo do jednoho zařízení tak, aby odpadlo zdlouhavé propojování, množství použitých propojovacích vodičů a nutnosti využití nepájivých polí.

Deska TinyLab se skládá z Arduina Leonardo s veškerou jeho funkčností doplněnou o možnost fyzicky přepínat mezi porty Arduina a ovládacími prvky na rozšířené desce. Jednotlivým modulům se budeme věnovat v následujících kapitolách.

Veškerá dokumentace a programové bloky jsou volně dostupné[2].



Obrázek 1: Vývojová deska TinyLab se zvýrazněným mikropočítačem Arduino Leonardo

[1] Skupinově financovaný
[2] https://github.com/sixfab/tinylab

1.1 ARDUINO

Arduino je open-sourcová elektronická platforma postavená na základě snadno použitelného hardwaru a softwaru. [1]

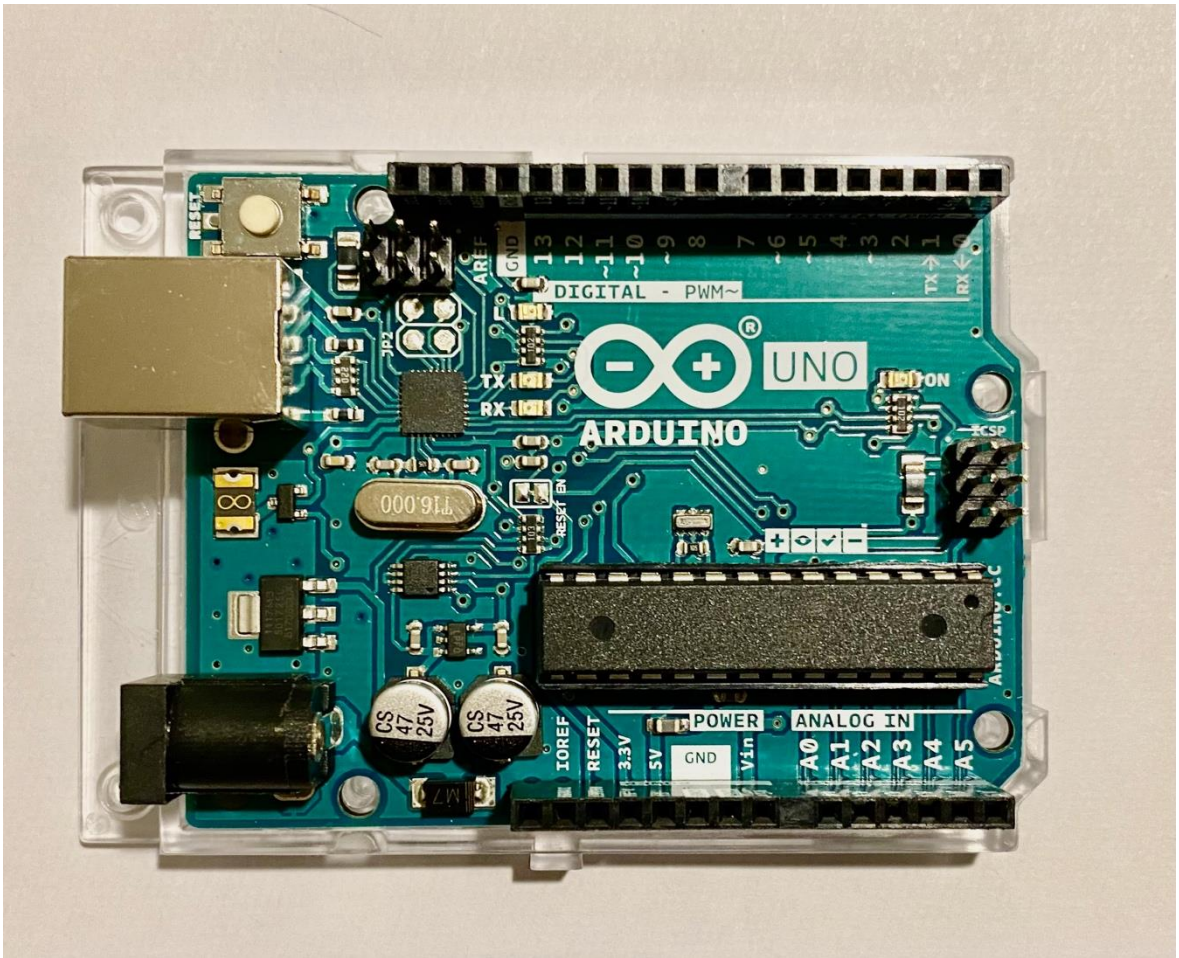
Projekt Arduino byl vytvořen zejména jako jednoduché a finančně dostupné zařízení pro začínající i mírně pokročilé zájemce o programování hardware bez nutnosti rozsáhlých technických znalostí. Umožňuje práci s mikroprocesory bez předchozích znalostí architektury nebo podrobností o funkci celého systému. TinyLab tuto myšlenku ještě více rozvíjí a k Arduinu integruje často využívané moduly.

Arduino a jeho nástavba TinyLab jsou tedy ideálními deskami pro výuku. Jsou cenově dostupné, otevřené, dobře zdokumentované ze strany hardwaru i softwaru, kompatibilní se širokým spektrem modulů a senzorů.

Vývojové desky Arduino jsou dostupné různých v provedeních, která se liší velikostí, mikroprocesorem, počtem vstupů a výstupů, velikostech pamětí. Desky jsou řízeny procesorem od firmy Atmel, ATMega.

Otevřenost softwaru i hardwaru umožňuje výrobcům adaptovat desky pro specifické účely, modernizovat je, minimalizovat nebo rozšiřovat. Arduino komunita je rozsáhlá, má dobře zdokumentované postupy a poskytuje mnoho návodů a řešení problémů.

Jedny z nejběžnějších desek jsou modely Arduino UNO a Arduino Leonardo, na kterém je postaven TinyLab.



Obrázek 2: Arduino UNO – vývojová deska

1.1.1 DIGITÁLNÍ A ANALOGOVÉ PORTY MIKROPOČÍTAČE

Arduino je mikropočítač řízený programem, který pracuje se vstupními a výstupními porty. Jejich přehled nalezneme v tabulce níže a poslouží nám k pochopení jejich významu.

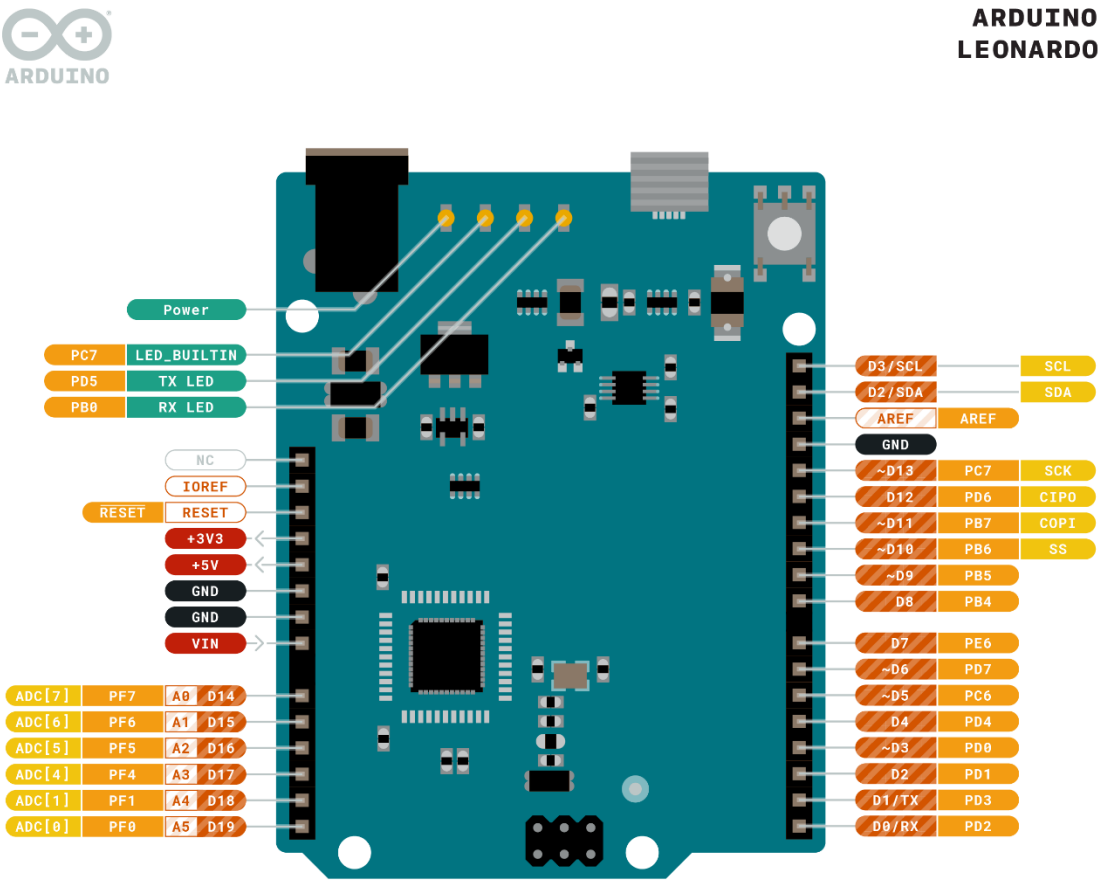
Tabulka 1: Arduino Leonardo – popisky portů

Značení	Význam
A#	Analogový port č. # (např. A0-A5)
D#	Digitální port č. #
~D#	Digitální port č. # s podporou PWM
D0/RX	Port pro přijímaná data přes sběrnici UART
D1/TX	Port pro odeslaná data přes sběrnici UART
SDA	Port pro data přenášená přes I2C sběrnici (Serial Data)
SCL	Port pro časový signál I2C sběrnice (Serial Clock)
CS	Port pro Chip Select pro rozhraní SPI
CE	Port pro Chip Enable pro rozhraní SPI
CSN	Port pro Chip Select/Enable pro rozhraní SPI
GND	Zemnicí port

Značení	Význam
+5V	Napájecí port 5 voltů
3V3	Napájecí port 3,3 volty

1.1.2 UMÍSTĚNÍ PORTŮ NA ZÁKLADNÍ DESCE MIKROPOČÍTAČE

Ukázka jednotlivých portů na základní desce mikropočítače Arduino Leonardo.



Obrázek 3: Arduino Leonardo – pinout diagram (Zdroj: https://docs.arduino.cc/static/e62e15c564ec48ac82db2d311d20fb1a/A000057-pinout.png)

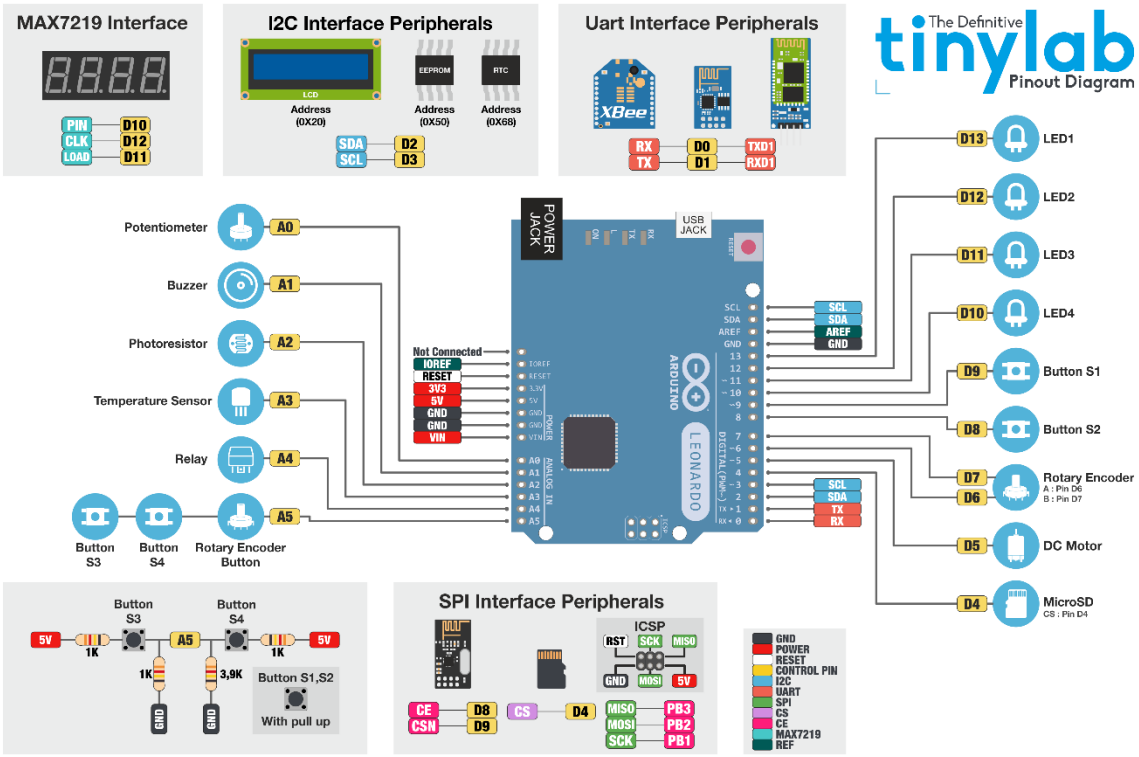
1.2 MODULY STAVEBNICE TINYLAB

Základní deska TinyLab se skládá z mikropočítače Arduino Leonardo a dalších pevně osazených komponent, tzv. modulů, např. LED, tlačítka, displeje, a paticemi pro komunikační periferie jako Bluetooth nebo XBee. Dále obsahuje slot pro micro SD kartu a přihrádku na knoflíkovou baterii pro samostatné napájení RTC.

Veškeré moduly jsou podrobně popsány v následujících kapitolách.

1.3 PROPOJENÍ MODULŮ S MIKROPOČÍTAČEM

Pro lepší přehled se můžeme podívat na schéma zapojení TinyLab. U většiny komponent je uvedená adresa portu/ů nutných ke správné funkci přímo na desce, nicméně některé nejsou označené. Moduly jsou přímo napojeny na Arduino, tím odpadá nutnost propojovat moduly pomocí vodičů. Vedle každého portu je přepínač, který umožňuje zvolit funkci napojeného modulu nebo připojení na port Arduina přímo. Dále jsou zde sdílená rozhraní, která určité periferie využívají.



Obrázek 4: TinyLab – pinout digram (Zdroj: https://tinylab.cc/wp-content/uploads/2016/05/tinylab-pinout-diagram2.png)

1.4 VÝSTUPNÍ ZAŘÍZENÍ

V této sekci si představíme jednotlivé periferie na desce TinyLab společně s ovládacími prvky a vybranými příkazy z knihoven.

Tečky před příkazy značí tzv. tečkovou notaci tedy přístup k metodám a vlastnostem objektu.

1.4.1 LED

Na desce TinyLab se nacházejí čtyři světlo vyzařující diody (LED). Diody jsou připojeny k digitálním portům D10 až D13. Množství světla vyzařované diodou diody L1, L3, L4 lze regulovat PWM (Pulse-Width Modulation) hodnotami 0-255, L2 má pouze stavy vypnuto/zapnuto.

LED sdílí porty se segmentovým displejem, D10-D13. Pro jeho aktivaci přepneme sadu přepínačů nacházející se mezi segmentovým displejem a LED na polohu s označením „To Control 7-Segment“.

Tabulka 2: LED – příkazy

`pinMode(číslo pinu, INPUT/OUTPUT)`	Přiřazení
`digitalWrite(číslo pinu, HIGH/LOW)`	Rozsvítí/zhasne LED
`analogWrite(číslo pinu, 0-255)`	Logaritmicky rozsvítí LED (PWM)

1.4.2 BZUČÁK (BUZZER)

Piezoelektrický bzučák má vyhrazený analogový port A1. Umožňuje přehrávat tóny v rozsahu 50 Hz-14 kHz. Pokud nezadáme rozšiřující parametr pro dobu trvání tónu, můžeme jej zastavit pomocí dalšího příkazu.

Tabulka 3: Bzučák – příkazy

`pinMode(číslo pinu, INPUT/OUTPUT)`	Přiřazení
`tone(číslo pinu, frekvence, (případně) trvání)`	Vyvolání tónu o dané frekvenci, případně trvání
`noTone(číslo pinu)`	Ztlumení tónu

1.4.3 RELÉ (RELAY)

Relé slouží ke spínání/rozepínání externího obvodu nebo periférie. Na desce je model JQC-3F(T73). Výstup má 3 porty, kde uprostřed je společný vodič (COM), vlevo sepnutý (NC) a vpravo rozepnutý (NO) pin. Při aktivaci se piny přepnou do opačného stavu. Dokáže spínat až 230 V, to pro výukové účely není vhodné ani bezpečné.

Relé je zapojeno na Analog IN 4, která sekundárně funguje jako digitální port D18.

Tabulka 4: Relé – příkazy

`pinMode(číslo pinu, INPUT/OUTPUT)`	`Přiřazení`
`digitalWrite(číslo pinu, HIGH/LOW)`	`Sepnutí/rozepnutí relé`

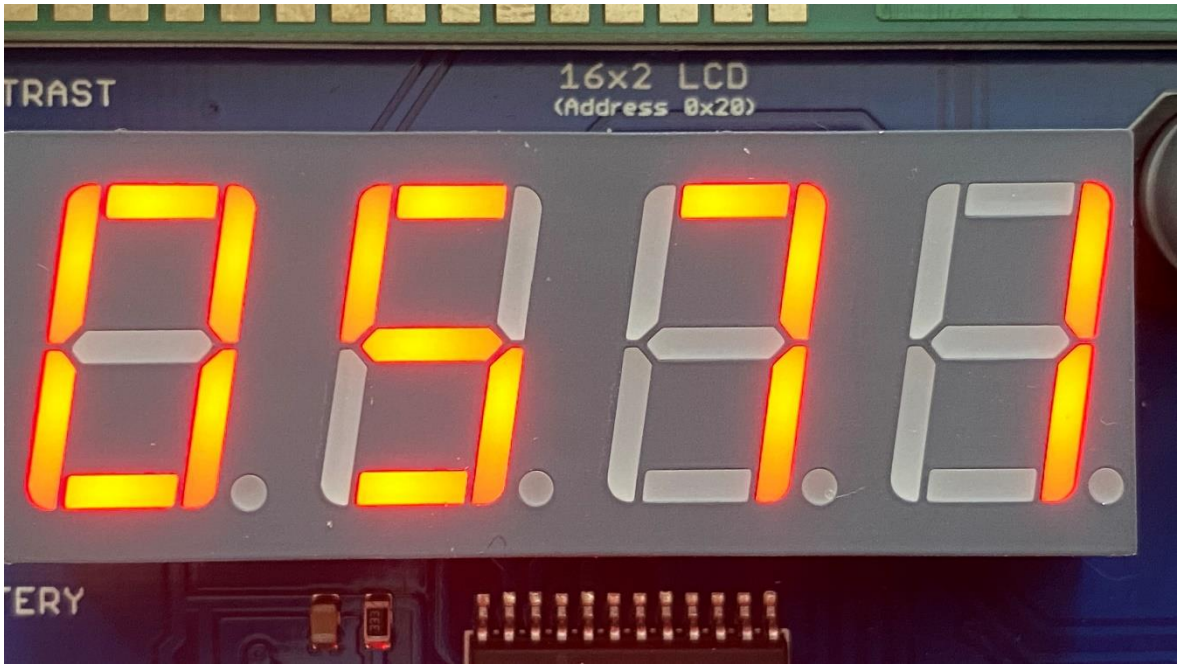
1.4.4 DC MOTOR

Na desce jsou k dispozici terminály pro řízení DC motoru (D5). Pomocí PWM můžeme ovládat šířku pulzu. Hodnoty napětí 0-5 V jsou reprezentovány digitálně v intervalu 0-255.

DC motory obecně potřebují vyšší proud, než je Arduino schopno poskytnou a zároveň může vytvořit škodlivé výkyvy napětí ve funkci generátoru. Abychom tomuto předešli, musíme prakticky motor izolovat od desky a napájet jej externě. [3]

1.4.5 SEGMENTOVÝ DISPLEJ (7-SEGMENT)

Číslice na segmentovém displeji jsou tvořeny sedmi dílky (segmenty). Standardně by se musel každý segment ovládat zvlášť, ale v našem případě, jak můžeme vidět z digramu (Obrázek 4), je použit MAX7219 Interface. Ten ovládá celý displej najednou a tím si ušetříme digitální porty na desce.



Obrázek 5: TinyLab – Segmentový displej

Abychom mohli displej jednoduše ovládat, použijeme knihovnu LedControl[3]. Uvedeme si zde základní příkazy, kompletní sadu lze najít v repozitáři projektu. Parametr adresa odpovídá pořadí displeje (pokud jich je více, začínáme od nuly).

Segmentový displej sdílí porty se všemi LED, D10-D13. Pro jeho aktivaci přepneme sadu přepínačů nacházející se mezi segmentovým displejem a LED na polohu s označením „To Control 7-Segment“.

Tabulka 5: Segmentový displej – příkazy

`LedControl(Din, CLK, Load, počet čipů)`	**Přiřazení/deklarace displeje**
`.shutdown(adresa, T/F)`	**Zapnutí/vypnutí (T/F) displeje**
`.setIntensity(adresa, <0-15>)`	**Nastavení jasu displeje (logaritmicky)**
`.clearDisplay(adresa);`	**Vyčištění displeje**
`.setDigit(adresa, pozice, číslo, desetinná tečka T/F)`	**Zobrazení číslice na dané pozici**

1.4.6 LCD DISPLEJ

Tento displej využívající mikročip MCP23008 se skládá z matice 16x2 obdélníčků. Obdobně jako segmentový předchůdce je ovládán pomocí sběrnice, tentokrát I²C. Ta je sdílena s RTC a EEPROM.

Potřebné knihovny jsou Wire[4] pro správnou funkcionalitu I²C/TWI sběrnice. Dále budeme potřebovat LiquidTWI2[5] na ovládání displeje.

Sběrnice I²C využívá porty D2 (SDA) a D3 (SCL), adresa displeje na I²C sběrnici je 0x20, model displeje je LTI_TYPE_MCP23008.

[3] https://github.com/wayoda/LedControl
[4] https://github.com/esp8266/Arduino/tree/master/libraries/Wire
[5] https://github.com/lincomatic/LiquidTWI2

Tabulka 6: LCD displej – příkazy

`LiquidTWI2 <název>(adresa sběrnice)`	Deklarace displeje
`.setMCPType(LTI_TYPE_MCP23008)`	Nastavení typu displeje
`.begin(počet znaků, počet řádků)`	Spuštění displeje a nastavení počtu znaků a řádků
`.setBacklight(LOW/HIGH)`	Nastavení podsvícení (ON/OFF)
`.clear()`	Vyčištění displeje a reset kurzoru na pozici 0, 0
`.setCursor(x, y)`	Nastavení kurzoru na dané souřadnice

1.5 VSTUPNÍ ZAŘÍZENÍ

1.5.1 TLAČÍTKO

Tlačítko jeden z nejběžnějších ovládacích prvků. Při stisknutí spojí obvod a při puštění opět rozpojí. Hodnoty tlačítka v obou stavech jsou ovlivněny zapojeným rezistorem. Pokud je zapojen k zemi, tzv. pull-down rezistor, hodnota se blíží nule. Pokud je tomu naopak a rezistor je napojen k napájecímu napětí, pull-up, vstup se blíží k logické 1. [2]

V případě desky TinyLab jsou obě digitální tlačítka zapojena jako pull-up na portech D8 a D9, tudíž při sepnutí indikují logickou 0. Tlačítka 3 a 4 jsou zapojeny na společný port A5 v režimu pull-down, každé však s odlišným rezistorem (viz Obrázek 4 vlevo dole). Proto je možné rozlišit úrovní napětí na vstupu kombinace jejich stisknutí. Na port A5 je také připojeno tlačítko rotačního enkodéru.

Tabulka 7: Tlačítko – příkazy

`pinMode(číslo pinu, INPUT/OUTPUT)`	Přiřazení
`digitalRead(číslo pinu)`	Čtení digitálního tlačítka
`analogWrite(číslo pinu)`	Čtení analogového tlačítka

1.5.2 POTENCIOMETR

Potenciometr je odporový napěťový dělič, který působí jako proměnný rezistor. Má tři terminály, z pohledu na otočný element zleva jsou to: země, výstupní napětí, vstupní napětí. Nabývá hodnot 0-1023 na analogovém portu A0.

Tabulka 8: Potenciometr – příkazy

analogRead(číslo pinu)	**Čtení potenciometru**

1.5.3 ROTAČNÍ ENKODÉR

Malý (2017, s. 308) enkodér popisuje: „Taková součástka má uvnitř podobný kotouček s Grayovým kódem. Má dva vývody, na kterých se střídají hodnoty 00 – 01 – 11 – 10, nebo 00 – 10 – 11 – 01, to podle směru otáčení. Na jednu otáčku mívají typicky 20 kroků, a navíc bývají kombinovány s tlačítkem, takže lze zařízení ovládat pomocí otáčení do dvou směrů a mačkání tlačítka.“ [4]

Zjednodušeně rotační enkodér zaznamenává směr a krokově i otáčení voliče, které nám poskytuje jako digitální hodnotu.

Vyhrazuje si porty D6-D7 na otáčení a A5 pro tlačítko.

Tabulka 9: Rotační enkodér – příkazy

pinMode(číslo pinu, INPUT/OUTPUT)	**Přiřazení**
digitalRead(číslo pinu)	**Čtení otoček**
analogRead(čísolo pinu)	**Čtení tlačítka**

1.5.4 FOTOREZISTOR

Fotorezistor je variabilní rezistor reagující na světlo, kde se v závislosti na množství na něj dopadajícího světla mění hodnota jeho elektrického odporu. Ten má vliv na napětí přiváděné na připojený portu A2.

Tabulka 10: Fotorezistor – příkazy

analogRead(číslo pinu)	**Čtení fotorezistoru**

1.5.5 TEPLOTNÍ SENZOR (TEMPERATURE)

Deska TinyLab obsahuje teplotní senzor LM35 napojený na analogový port A3. Podle rozsahu napětí se mění měřitelné maximum a minimum. Změna jednoho stupně odpovídá 10 mV.

Tabulka 11: Teplotní senzor – příkazy

analogRead(číslo pinu)	**Čtení teplotního senzoru**

1.5.6 RTC

Hodiny reálného času DS1307 udržují údaj o aktuálním čase, který je schopen uchovat s pomocí baterie velmi dlouho dobu (v řádech let). Hodiny se musí po resetu vždy nastavit.

Pro ovládání jsou nutné knihovny Wire[6] kvůli I²C sběrnici a RTClib[7] pro samotné ovládání hodin.

Tabulka 12: RTC – příkazy

RTC_DS1307	Deklarace
.adjust(čas a datum)	Nastavení času
.begin()	Spuštění
.isrunning()	Dotaz, zdali hodiny běží (jsou nastaveny)
.now()	Vrátí aktuální čas

1.6 VSTUPNĚ VÝSTUPNÍ ZAŘÍZENÍ

1.6.1 EEPROM

Elektronická programovatelná nevolatilní paměť na sběrnici I²C. Tento typ paměti je zpravidla integrovaný na zařízení, které si potřebuje zachovat předchozí stavy během vypnutí, nebo logy.

Kvůli I²C potřebujeme knihovnu Wire[8] a I2C_eeprom[9] pro paměť.

Tabulka 13: EEPROM – příkazy

I2C_eeprom <název>(0x50, I2C_DEVICESIZE_24LC256);	Deklarace
.isConnected()	Kontrola připojení
.writeByte()	Zápis
.readByte()	Čtení

[6] https://github.com/esp8266/Arduino/tree/master/libraries/Wire
[7] https://github.com/adafruit/RTClib
[8] https://github.com/esp8266/Arduino/tree/master/libraries/Wire
[9] https://github.com/RobTillaart/I2C_EEPROM

1.6.2 ČTEČKA SD KARET (SD CARD)

Čtečka SD karet slouží podobnému účelu jako EEPROM modul, tedy k ukládání dat.

Malý (2019, s. 70): „Hlavní výhoda, kterou SD karty mají, je ta, že mají velmi jednoduché rozhraní – nepotřebujete přesné časování jako u USB, stačí vám pouze nějak naemulovat rozhraní SPI, a to zvládne i pomalý osmibit.“ [5]

Pro správnou funkcionalitu budeme potřebovat dvě knihovny SD[10], která je ještě závislá na SPI[11].

Tabulka 14: Čtečka SD karet – příkazy

`.begin(číslo pinu)`	Inicializace
`.open(název souboru, čtení/zápis)`	Otevření souboru k čtení/zápisu
`.close()`	Zavření souboru

1.6.3 XBEE MODUL

XBee je IoT modul umožňující komunikace na běžné frekvenci 2.4 GHz. Mohou tedy integrovat Bluetooth, WiFi a RF vysílání. Zapojen je prostřednictví sériového portu UART skrze porty D0/RX a D1/TX.

Ke správnému nastavení a použití je nutný doprovodný software k dispozici na stránkách www.digi.com.[12]

Tabulka 15: XBee – příkazy

`<linka>.begin(baudy za sekundu)`	Spuštění vybrané sériové linky
`<linka>.available()`	Ověření dostupnosti linky
`<linka>.write(data)`	Zápis dat na linku
`<linka>.read()`	Čtení dat z linky

1.6.4 WIFI MODUL (ESP8266)

Na desce se nachází patice pro WiFi modul ESP8266 na UART sběrnici, které je vyhraněna část sériové linky – Serial1 (porty D0/RX a D1/TX).

[10] https://github.com/arduino-libraries/SD
[11] https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/SPI
[12] https://hub.digi.com/support/products/xctu/

Tabulka 16: WiFi – příkazy

`<linka>.begin(baudy za sekundu)`	Spuštění vybrané sériové linky
`<linka>.available()`	Ověření dostupnosti linky
`<linka>.write(data)`	Zápis dat na linku
`<linka>.read()`	Čtení dat z linky

1.6.5 BLUETOOTH MODUL

Poslední z trojice komunikačních modulů na sběrnici UART. S tímto modulem můžeme komunikovat přímo s PC a sledovat tak veškerou komunikaci mezi oběma zařízeními.

Tabulka 17: Bluetooth – příkazy

`<linka>.begin(baudy za sekundu)`	Spuštění vybrané sériové linky
`<linka>.available()`	Ověření dostupnosti linky
`<linka>.write(data)`	Zápis dat na linku
`<linka>.read()`	Čtení dat z linky

1.6.6 RÁDIOVÝ MODUL (NRF24LXX)

Radiofrekvenční modul nRF24L01 vysílá na frekvenci 2,4 GHz. Můžeme jím skenovat frekvence, komunikovat s dalšími moduly, přenášet mezi nimi informace.

Knihovna nutná pro ovládání je RF24[13] s dependencí SPI[14] a to kvůli sběrnici.

[13] https://github.com/nRF24/RF24

[14] https://github.com/arduino/ArduinoCore-avr/tree/master/libraries/SPI

Tabulka 18: Rádiový modul – příkazy

RF24 <název>(CE, CSN)	Deklarace
.begin()	Spuštění
.setPALevel(síla signálu)	Nastavení signálu
.setPayloadSize(sizeof(data))	Nastavení velikosti přenesených dat
.openWritingPipe(vysílač)	Nastavení vysílače
.openReadingPipe(1, vysílač /přijímač)	Nastavení přijímače
.startListening()	Přepnutí na přijímač
stopListening()	Přepnutí na vysílač
.write(&data, sizeof(data))	Odeslání dat o velikosti
.read(&data, sizeof(data))	Příjem dat o velikosti

1.6.7 NEPÁJIVÉ KONTAKTNÍ POLE

Speciálním případem je nepájivé kontaktní pole. Umožňuje zapojení různých součástek a periferií a ty potom ovládat připojením přímo s Arduinem.

2 VÝVOJOVÉ PROSTŘEDÍ A PROGRAMOVACÍ JAZYK

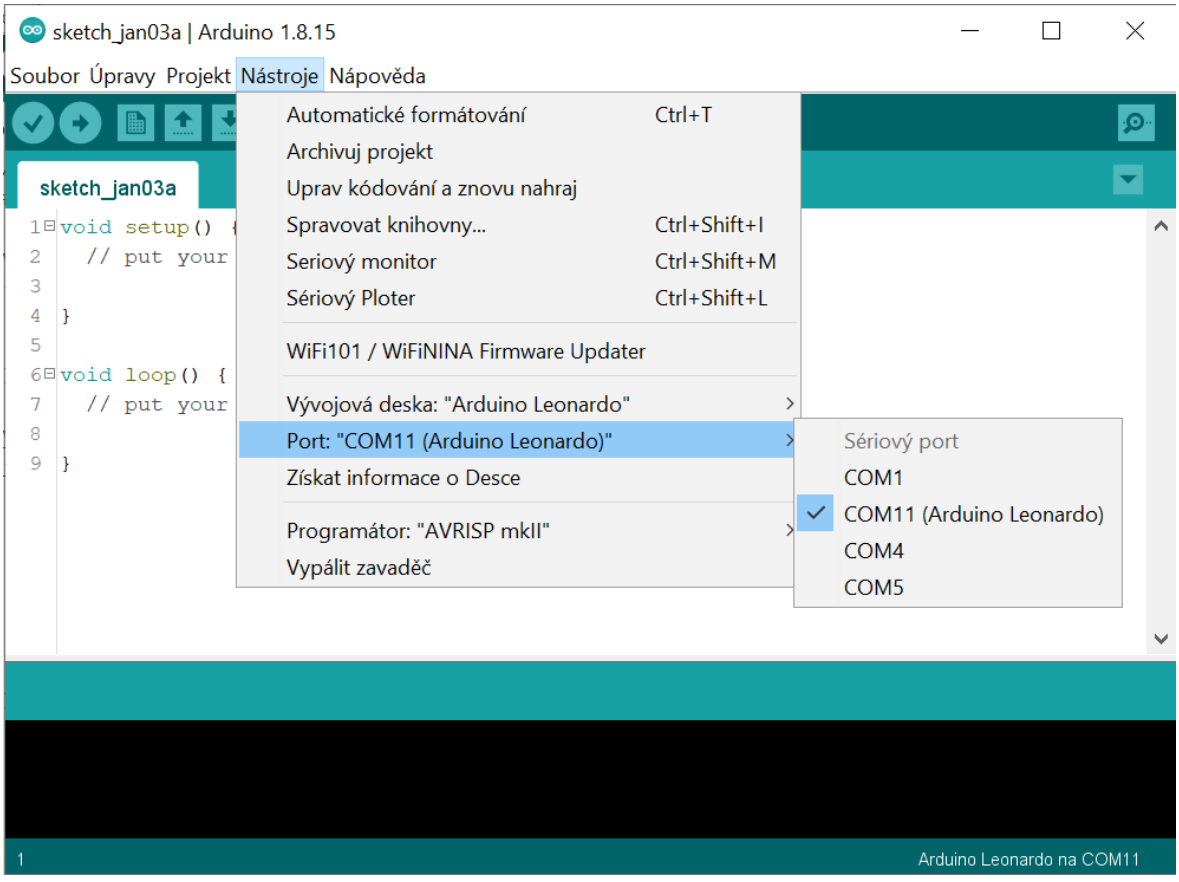
Nedílnou součástí prototypování je psaní kódu, k tomu nám poslouží volně dostupné vývojové prostředí Arduino IDE. Seznámíme se jeho funkcionalitou, prvky pro sledování průběhu programu, programovými bloky a jazykem.

2.1 VÝVOJOVÉ PROSTŘEDÍ

Jako vývojové prostředí neboli IDE (Integrated Development Environment) si zvolíme oficiální Arduino IDE. Je podporováno všemi mainstreamovými operačními systémy (Windows, Mac OS, Linux) a má velkou komunitu. Výhodou je prostředí s českou lokalizaci.

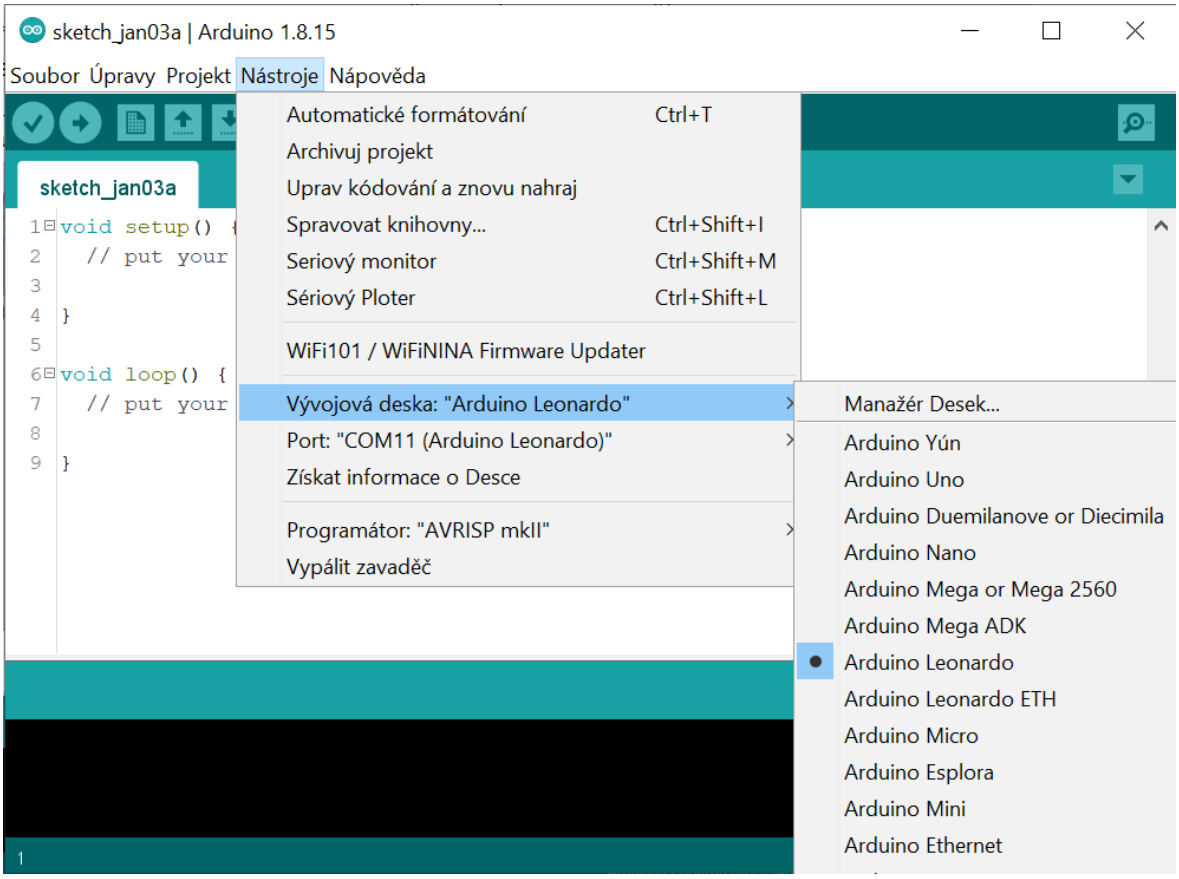
2.1.1 ÚVODNÍ NASTAVENÍ

Po připojení vývojové desky Arduino je nutné zkontrolovat na jakém portu se nachází. Po stisknutí Nástroje> Port> COM #. Pokud je Arduino připojené a detekované, bude vedle čísla COM portu napsaný jeho název.



Obrázek 6: Arduino IDE – Volba COM portu

Dále musíme nastavit správný druh desky, kde ve stejném menu zvolíme položku Vývojová deska. Zde vybereme desku, jejíž název můžeme najít u zmíněného COM portu, případně na desce samotné.



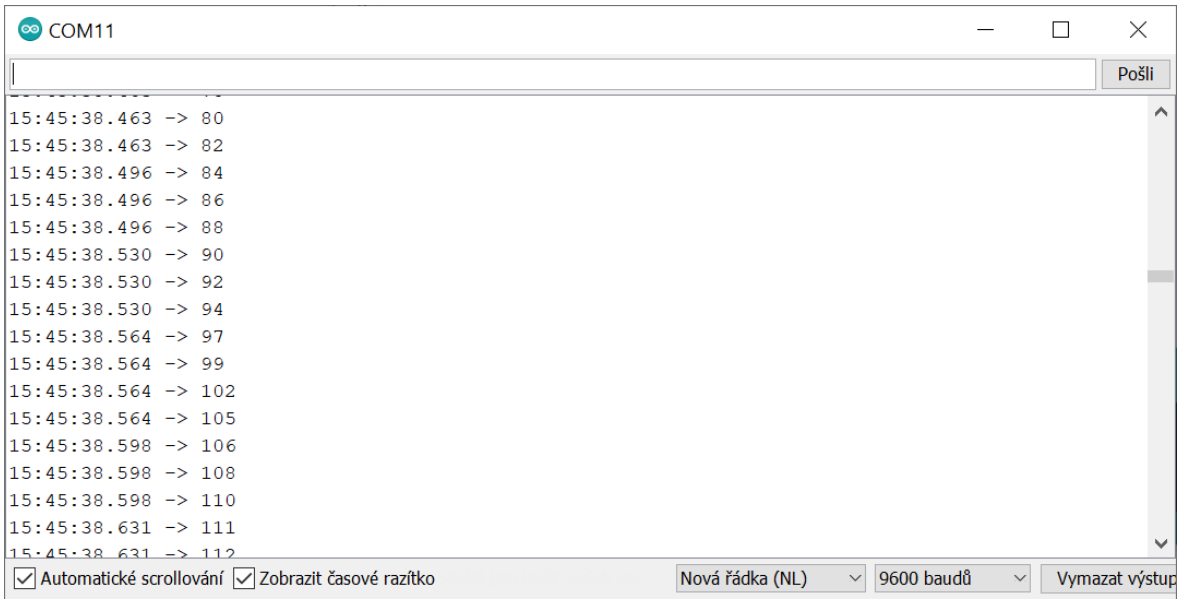
Obrázek 7: Arduino IDE – Volba Vývojové desky

2.1.2 NÁSTROJE PRO SLEDOVÁNÍ A LOGOVÁNÍ

Sériový monitor

Velmi důležitou komponentou je sériový monitor, který umožňuje výpis a sledování hodnot a také jejich zadávání.

Najdeme jej v hlavním menu pod položkou Nastavení> Sériový monitor a také pod zkratkou **Ctrl + Shift + M**.

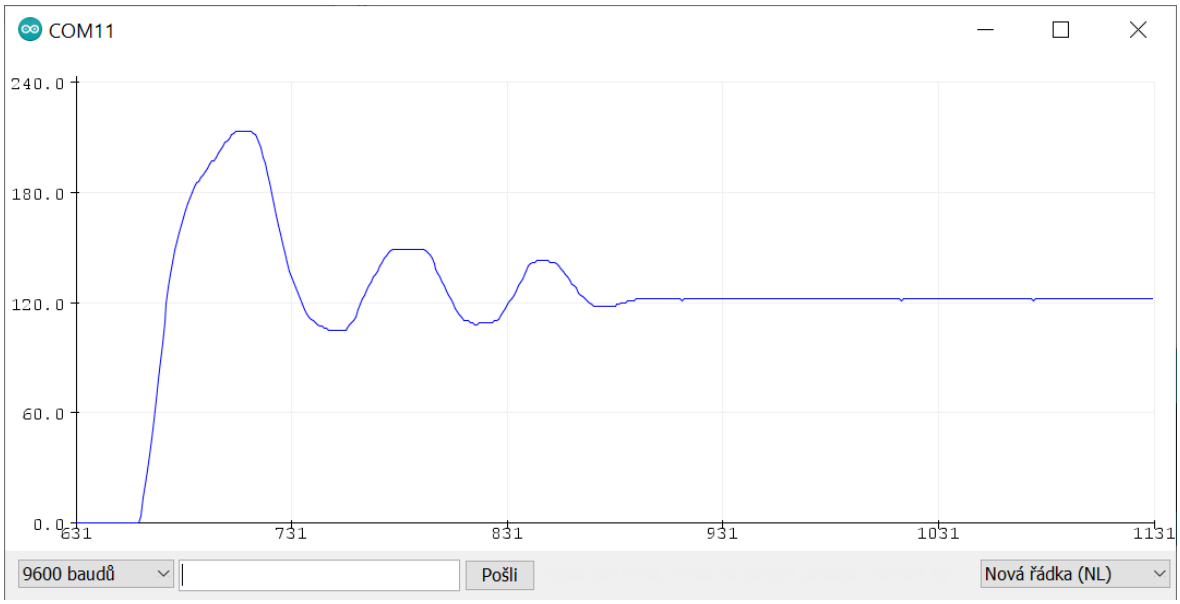


Obrázek 4: Arduino IDE – Sériový monitor

Sériový plotter

Dalším užitečným nástrojem je sériový plotter. Slouží k vykreslení grafů na základně hodnot definovaných proměnných.

Nachází se v Nastavení> Sériový plotter, nebo také pomocí **Ctrl + Shift + L**.



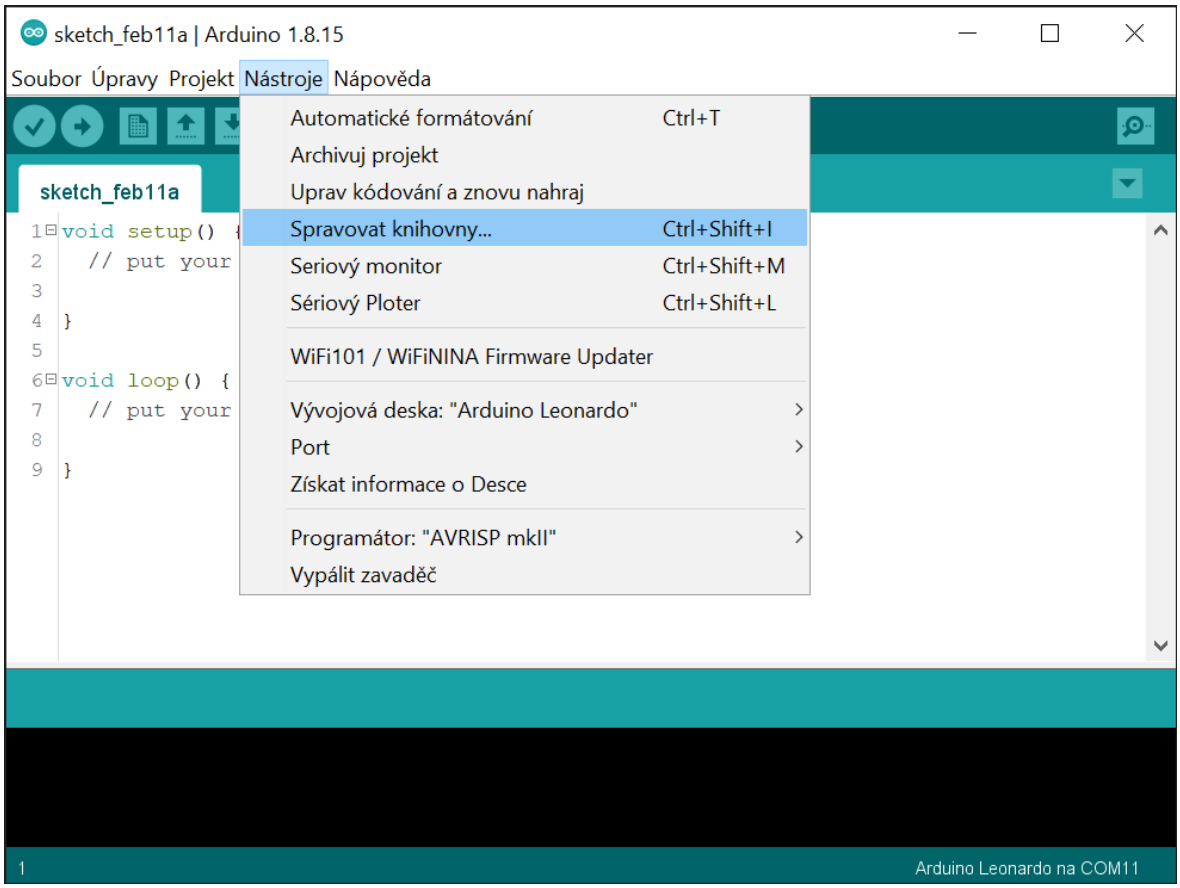
Obrázek 8: Arduino IDE – Sériový plotter

2.1.3 SPRÁVCE KNIHOVEN

Pro snadnější využití modulů či tvorbu pokročilejších programů lze využít knihovny. Jsou to balíčky metod a funkcí pro specifická témata. Ovládání modulů, manipulace s daty, výpočty, komunikace a další.

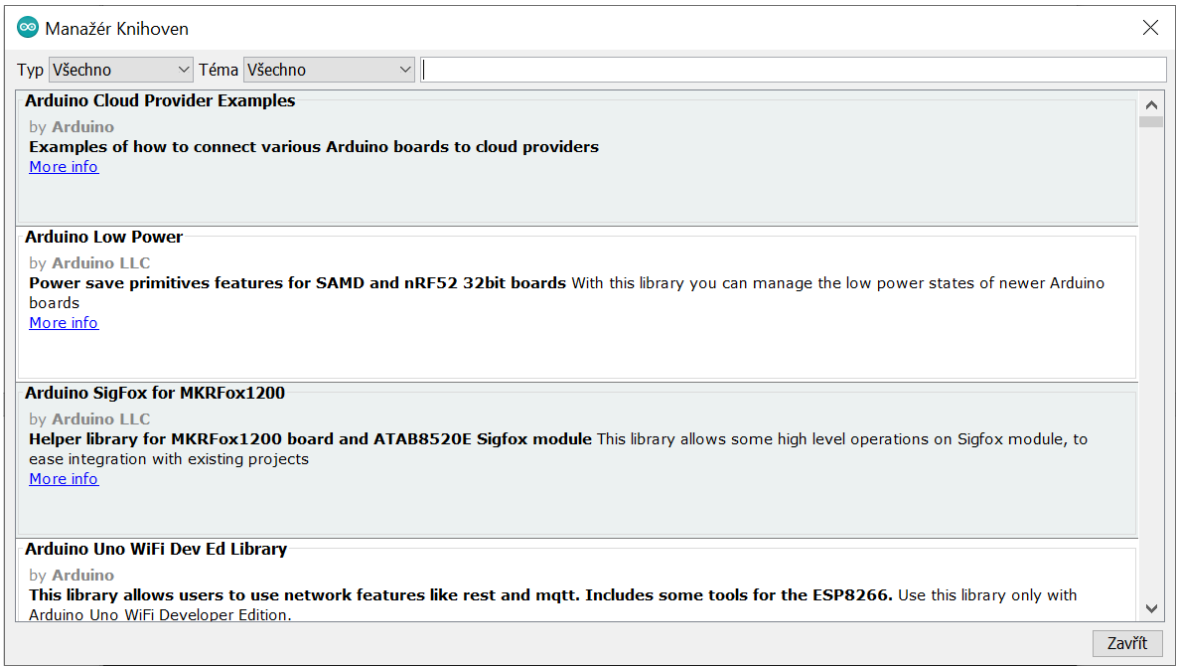
„Vývojové prostředí Arduino IDE ve své instalaci zahrnuje několik knihoven obsahujících užitečné funkce použitelné při tvorbě řešení pro různorodé projekty. Jedná se například o knihovny ošetřující práci s hardwarovými periferiemi nebo manipulaci dat. Na internetu je dostupných mnoho dalších knihoven vytvořených nadšenci Arduino komunit.“ [2]

Správce se nachází v Nástroje> Spravovat knihovny…, případně **Ctrl + Shift + I**.



Obrázek 9: Arduino IDE – Správa knihoven

Většina komponent vyžaduje knihovnu pro správnou funkcionalitu. Dají se najít v manažeru knihoven a běžně jsou k dostání na webu výrobce, případně prodejce. Knihovny specifické pro jednotlivé moduly jsou uvedeny v jejich popisu dále.



Obrázek 10: Arduino IDE – Manažer knihoven

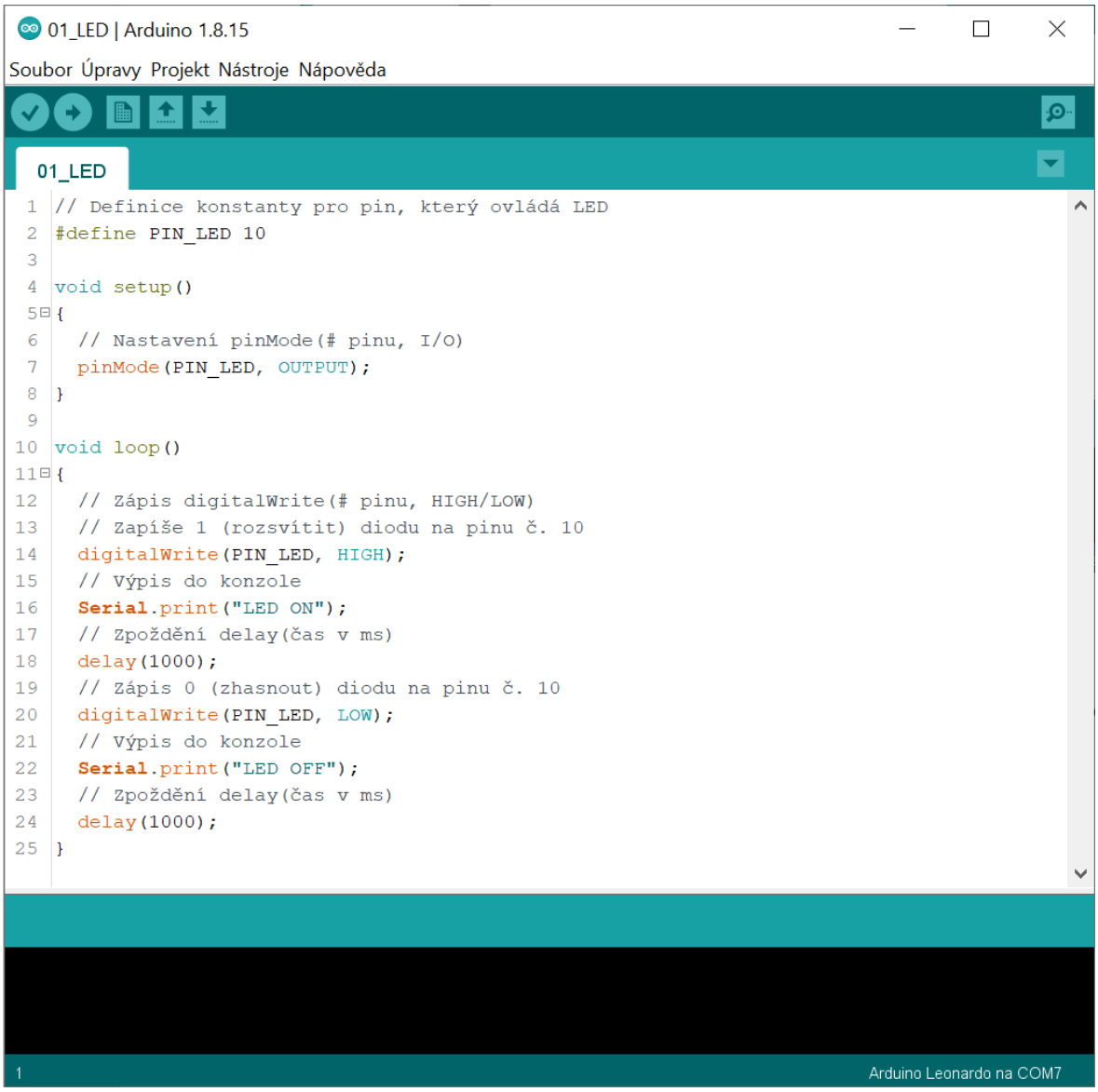
2.2 PROGRAMOVACÍ JAZYK

Programovací jazyk použitý v Arduino IDE by se dal definovat jako dialekt jazyků C/C++. Programy se zde označují jako sketche. Při zápisu klade důraz na čitelnost člověkem. Práce s moduly je zde často obsažena v externích knihovnách. Bloky jsou obaleny ve složených závorkách. Deklarace proměnných, konstant a metod je dopředná. Z pohledu překladače je kód kompilován od počátku, tudíž proměnná nebo funkce musí být definována předtím, než ji zavoláme/použijeme.

Funkce a metody využité pro ovládání komponent jsou uvedeny dále přímo u popisu jednotlivých modulů.

2.2.1 ZÁKLADNÍ METODY A BLOKY

Program můžeme strukturovat na tři základní části a jednu doplňující: deklarace, metody `setup()`, `loop()` a uživatelem definované funkce.



Obrázek 11: Arduino IDE - přehled bloků

Deklarace

První část je obvykle vymezena pro import knihoven, definici vlastních proměnných, včetně přiřazení těchto proměnných jejich periferiím.

Metoda setup()

Metoda `setup()` je volána ihned po spuštění Arduina a to pouze jednou. Využívá se zejména pro inicializaci proměnných a funkcí, přiřazení režimu portů, volání knihoven, počátek sériového přenosu včetně přiřazení datového toku.

Metoda loop()

Po skončení metody `setup()`, je v nekonečném cyklu volána metoda `loop()`. Opakovaně provádí vše, co je v ní obsaženo. V metodě se proto obvykle nachází hlavní výkonná část programu, která pracuje se vstupy a výstupy a řídí činnost desky Arduino.

Uživatelem definované funkce

Pokud plánujeme vícekrát volat stejnou sekvenci postupů, je vhodné si vytvořit vlastní funkci. Kromě potenciální adaptibility pro vstupy může funkce kód také zpřehlednit.

Jako příklad můžeme uvést výpis znaků na displej. Abychom toho docílili, musíme displej nejprve smazat, nastavit si kurzor (místo, od kterého budeme psát), vypsat text na řádek, přemístit kurzor na další a opět vypsat text. Pokud výpis dat bude volán několikrát, je vhodné si pro něj vytvořit metodu.

2.2.2 KLÍČOVÁ SLOVA

Každý programovací jazyk rezervovaná slova. Tato slova mají vždy stejný předem daný význam a uživatel je nemůže použít jako názvy svých proměnných nebo metod.

Struktura

Obecné prvky programovacího jazyka C/C++.

Preprocesor

Ještě před předáním kódu kompilátoru se mohou zavolat externí knihovny, nadefinovat konstanty a výrazy (neboli makra). Všechny tyto objekty spojuje uvození – začínají vždy „křížkem“. Jsou to příkazy jako `#import <knihovna.h>` nebo `#define NAZEV hodnota`.

Sketch

Zásadní ovládací metody pro Arduino: `setup()` a `loop()`, jejichž funkci jsme si popsali v kapitole Základní metody a bloky.

Řídicí struktury

Mezi základní ovládací prvky každého jazyka patří podmínky, cykly nebo návratové hodnoty funkcí. Tato klíčová slova jsou stejná nebo velmi podobná pro většinu jazyků: `if…(else)`, `(do)…while`, `switch…case`, `return`.

Operátory

Obecně je to sada znaků pro logické, aritmetické, porovnávací a bitové operace. Některá mohou být sloučeny za účelem ušetření místa.

Funkce

Funkce ovládají Arduino a provádějí operace/výpočty.

Čtení a zápis dat

Pokud pracujeme se periferiemi, je nutné z nich načítat data, případně data do nich odesílat a tím je prezentovat. Spadají sem `analogRead()`, `analogWrite()`, `digitalRead()`, `digitalWrite()`,`pinMode()`.

Náhodná čísla

Pro určitou nahodilost v programech se dá použít dvojice funkcí `random()` a `randomSeed()`. Pozor, jedná se pouze o pseudonáhodné funkce, nicméně k našemu použití jsou postačující.

Matematika

Základní matematické úkony jsou již v jazyce implementovány, pokud se jedná o odmocniny, absolutní hodnoty nebo třeba minima a maxima, nemusíme tyto funkce psát sami: `min()`,`max()`,`abs()`,`sqrt()`,`pow()`.

Komunikace

Objekty nutné pro komunikaci a přenos dat, `Serial` a `Stream`. Umožňují operace s daty a komunikaci na desce.

Proměnné

Proměnné je obecný název pro entitu, které můžeme přidělit hodnotu, kterou můžeme v průběhu programu měnit.

Konstanty

Jsou to klíčová slova, která obsahují hodnotu, kterou můžeme pouze volat, nikoliv ji změnit. Standardně jsou psány kapitálkami (velkými písmeny), tímto je na první pohled v kódu rozlišíme.

Například hodnota `M_PI` reprezentující hodnotu π z knihovny `<math.h>`. Dále jsou zde konstanty závislé na typu desky: napěťové `HIGH` a `LOW`, kde hodnota odpovídá typu zvolené desky. `INPUT` a `OUTPUT`, kde se přiřadí portům směr komunikace.

Jedinou výjimkou jsou logické hodnoty `true` a `false`, které se píší malými písmeny.

Datové typy

Datové typy reprezentují hodnotu, kterou proměnná obsahuje. Příkladem jsou `boolean`, `int`, `double`, `char`, `String` reprezentující logickou hodnotu, celé číslo, desetinné číslo, znak a řetězec znaků respektive.

3 PRAKTICKÉ ÚLOHY

Součástí všech úloh je kód, který je přiložen na doprovodném médiu a zároveň v repositáři na webu https://github.com/karbans/Arduino_TinyLab. První úkoly z kapitol slouží hlavně k seznámení s problematikou a k demonstraci dostupných prostředků.

Záměr následujících kapitol, pojmenovaných po samotných modulech, je seznámení s jejich funkcionalitou, možnostmi vývojového prostředí Arduino, principy programování a praktickým využitím.

Nejdříve se seznámíme s praktickým využitím modulů v běžném životě, následuje zadání samotné úlohy, použité moduly, případné knihovny a programové bloky, doporučený postup řešení a kódem, možné a předpokládané problémy, cíl úlohy a možné rozšíření.

Následující tabulka poslouží jako rozcestník a mapa modulů. U každého z nich jsou cvičení včetně obtížnosti.

Tabulka 19: Mapa modulů a úloh

Modul	Úloha	Obtížnost
LED	Ovládání LED	Snadná
	Bluetooth modul	Pokročilá
	Hlídač teploty se signalizací	Obtížná
Bzučák	Bzučák	Snadná
	Hlídač teploty se signalizací	Obtížná
Segmentový displej	Segmentový displej	Snadná
LCD displej	LCD displej	Pokročilá
	Hodiny s teploměrem	Obtížná
DC motor	DC motor	Snadná
Relé	Relé	Snadná
Tlačítko	Tlačítka	Snadná
Potenciometr	Potenciometr	Snadná
Rotační enkodér	Rotační enkodér	Pokročilá
Teplotní senzor	Teplotní senzor	Pokročilá

Modul	Úloha	Obtížnost
	Hodiny s teploměrem	Obtížná
	Hlídač teploty se signalizací	Obtížná
	Logování teploty	Obtížná
Fotorezistor	Fotorezistor	Snadná
RTC	RTC	Pokročilá
	Hodiny s teploměrem	Obtížná
	Logování teploty	Obtížná
EEPROM	EEPROM	Pokročilá
Čtečka SD karet	Čtečka SD karet	Pokročilá
	Logování teploty	Obtížná
Rádiový modul	Rádiový modul	Obtížná
WiFi modul	WiFi modul	Snadná
XBee modul	XBee modul	Pokročilá
Bluetooth modul	Bluetooth modul	Pokročilá
Nepájivé kontaktní pole	Nepájivé kontaktní pole	Snadná

3.1 VÝSTUPNÍ ZAŘÍZENÍ

3.1.1 OVLÁDÁNÍ LED

S LED, Light Emitting Diode, se v běžném životě setkáme denně. Od jednoduchého podsvícení tlačítek, indikace funkčnosti/stavu modemu po světelnou dekoraci svátečních řetězů.

Zadání

Seznamte se s deskou TinyLab. Prozkoumejte topologii a zapojení.

Vytvořte program, který rozbliká diodu.

Změny stavu zaznamenejte do konzole.

Použité moduly a knihovny

LED

Programové bloky

Tabulka 15: delay()

delay(čas v milisekundách)	Zpoždění před vykonáním další akce

Postup řešení

V programu nejprve pomocí konstanty přiřadíme pozici portu, který ovládá LED.V metodě `setup()` nastavíme `pinMode(číslo pinu, INPUT/OUTPUT)`, kam dosadíme naši konstantu a OUTPUT, protože diodu chceme ovládat.

Dále do `loop()` napíšeme příkazy `digitalWrite(číslo pinu, HIGH/LOW)`. Můžeme zde použít konstanty LOW a HIGH, které reprezentují napětí 0 a 5 voltů. Tímto příkazem pustíme do portu napětí, tím se LED rozsvítí. `Serial.print("zpráva")` nám do konzole vypíše zprávu.

Jelikož se předchozí příkazy plní prakticky bez prodlevy dokola, nejsou tak pro nás změny stavu znatelné. Je nutné mezi nimi udělat pauzu. Příkaz `delay(čas v ms)` vynutí prodlevu před zpracováním dalšího příkazu. Pokračujeme dále, akorát LED zhasneme a tento fakt vypíšeme do konzole.

Před nahráním kód zkompilujeme a na desce TinyLab ověříme, že máme přepínač u digitálního vstupu/výstupu 10 v poloze ON. Poté si zapneme sériový monitor a nahrajeme program do Arduina.

Kód

```
// Definice konstanty pro pin, který ovládá LED
#define PIN_LED 10

void setup()
{
  // Nastavení pinMode(# pinu, I/O)
  pinMode(PIN_LED, OUTPUT);
}

void loop()
{
  // Zápis digitalWrite(# pinu, HIGH/LOW)
  // Zapíše 1 (rozsvítit) diodu na pinu č. 10
  digitalWrite(PIN_LED, HIGH);
  // Výpis do konzole
  Serial.print("LED ON");
  // Zpoždění delay(čas v ms)
  delay(1000);
  // Zápis 0 (zhasnout) diodu na pinu č. 10
  digitalWrite(PIN_LED, LOW);
```

```
  // Výpis do konzole
  Serial.print("LED OFF");
  // Zpoždění delay(čas v ms)
  delay(1000);
}
```

Cíle

Seznámit se s prototypovací deskou TinyLab a vývojovým prostředím Arduino.

Možné problémy

Dávejme si pozor na polohy přepínačů u jednotlivých portů u desky Arduino. A to i přepínačům mezi LED a segmentovým displejem.

Možnosti rozšíření

1. Zapojte více LED.
2. Kaskádově rozsviťte (jako např. dekorační LED rampouch).

3.1.2 BZUČÁK

Piezoelektrický bzučák je jednoduchá součástka sloužící zejména k signalizaci stavů či jako zvuková složka alarmu. Setkáme s ní u vyvolávacích systémů nebo platebních terminálů. Signalizuje také POST u základních desek.

Zadání

Seznamte se způsoby ovládání bzučáku.

Vytvořte program, který rozezní bzučák více než jedním způsobem.

Použité moduly a knihovny

Bzučák

Postup řešení

Nejdříve si nadefinujeme konstantu pro bzučák, kde ji zároveň přiřadíme odpovídající port, přitom si dáváme pozor, protože číslo analogového portu předchází písmeno „A“.

V iniciační metodě nastavíme mód, ve kterém port poběží `pinMode(číslo pinu, OUTPUT)`. V opakující se části kódu opakovaně rozezníme bzučák. Použijeme známý `pinMode(číslo pinu, LOW/HIGH)`, dále zkusíme rozeznít bzučák pomocí funkcí `tone(číslo pinu, čas v ms)` a `noTone(číslo pinu)`. Každou změnu zaznamenáme do konzole.

Kód

```
// Definice konstanty pro pin, který ovládá bzučák
#define BZUCAK A1

void setup()
{
  // Nastavení pinMode(# pinu, I/O)
  pinMode(BZUCAK, OUTPUT);
}

void loop()
{
  // Vyvolání tónu tone(# pinu, čas v ms)
  tone(BZUCAK, 1000);
  Serial.print("Vydávám zvuk.");
  delay(1000);
  // Zrušení tónu noTone(# pinu, čas v ms)
  noTone(BZUCAK);
  Serial.print("Jsem zticha.");
  delay(1000);
}
```

Cíle

Uvědomit si rozdíl mezi digitálním a analogovým zapojením a různými způsoby ovládání.

Možné problémy

Oproti předchozí úloze používáme analogovou součástku. Vyučujícím doporučuji ochranu sluchu.

Možnosti rozšíření

1. Použijte pokročilejších knihoven pro ovládání bzučáku.
2. Rozezněte bzučák pomocí tlačítka.

3.1.3 RELÉ

Relé (anglicky relay) je elektrotechnická součástka, která, v našem případě, pomocí elektromagnetu sepíná a rozepíná obvod. Využití je běžné v průmyslu jako proudová ochrana. Tedy pokud se v obvodu objeví nadměrné napětí, relé se rozepne. Fungovat mohou, samozřejmě, i obráceně, sepnutím obvodu. Můžeme jej také využít jako dálkově ovládaný přepínač.

Zadání

Seznamte se relé, jeho druhy a zapojte jej.

Dále správně zapojte dvě LED do terminálů relé, dbejte na správnou polaritu.

Vytvořte program, který bude periodicky aktivovat relé, změny stavu vypište do konzole a pozorujte na LED.

Jakým způsobem by bylo vhodnější LED zapojit?

Použité moduly a knihovny

Relé

Postup řešení

Nejprve zjistíme, jakou polaritu mají zdířky terminálu relé. Zapojíme do nich dvě LED a zajistíme šrouby.

V programu nejprve definujeme port pro relé `#define RELE A4`. V nastavení jej dáme do módu `OUTPUT` a spustíme sériovou linku. V cyklu budeme střídavě s prodlevou relé spínat a rozepínat.

Pokud bude vše fungovat, můžeme zkusit zapojit jiné součástky.

Kód

```
// Definice konstanty pro pin relé
#define RELE A4

void setup()
{
  pinMode(RELE, OUTPUT);

  Serial.begin(9600);
}

void loop()
{
  // Sepnutí relé
  digitalWrite(RELE, HIGH);
  delay(3000);
  // Rozepnutí relé
  digitalWrite(RELE, LOW);
  delay(3000);
}
```

Cíle

Fyzické zapojení LED, princip a funkce relé.

Možné problémy

Dáváme si pozor na polaritu součástek.

Možnosti rozšíření

1. K zapojení využijte integrované nepájivé kontaktní pole.

3.1.4 DC MOTOR

Se stejnosměrným motorem se setkáme skoro všude, kde něco rotuje. Od ventilátorů, dálkově ovládaná auta, elektrické nářadí po elektroautomobilitu. Jejich využití ale zde nekončí, pokud do motoru aktivně nejde proud, točením rotoru jej začneme vyrábět.

Zadání

Seznamte se s DC motory, jejich funkcionalitou. Dále s funkcí PWM a hodnotami, se kterými operuje.

Zapojte ovládací terminál a připojte k němu motor.

Napište program, který postupně rozběhne motor a následně jej stejným způsobem zastaví.

Pokud není k dispozici DC motor na 5 V, lze využít multimetru a sledovat na něm hodnoty.

Použité moduly a knihovny

DC motor

Postup řešení

Zapojíme terminály pro ovládání motoru a nadefinujeme si PWM port `#define MOTOR 5`.

Metodu setup() nemusíme nastavovat, pro funkcionalitu tohoto programu nemá vliv.

V opakující se části programu vytvoříme cyklus, ve kterém budeme postupně zvyšovat hodnotu, kterou analogově zapíšeme na port motoru `analogWrite(MOTOR, i)`. Zapisovaná hodnota na PWM port musí být v rozsahu 0-255. Analogicky vytvoříme cyklus zpomalení motoru.

Funkcionalitu si můžeme ověřit pomocí multimetru, případně využít pouze něj, pokud není k dispozici vhodný/žádný DC motor.

Kód

```
// Definice pinu pro zapojení motoru
#define MOTOR 5

void setup(){}

void loop()
{
  // Inkrementálně zvyšujeme otáčky motoru
  // Rozpětí pro PWM je 0-255
  for (int i = 0; i < 256; i++)
  {
    analogWrite(MOTOR, i);
    delay(100);
  }
```

```
  delay(2000);
  // Inkrementálně snižujeme otáčky motoru
  for (int i = 255; i >= 0; i--)
  {
    analogWrite(MOTOR, i);
    delay(100);
  }
}
```

Cíle

Seznámit se s PWM, DC motory.

Možné problémy

Většina motorů potřebuje víc než 5 V k funkci. Zapojený zapnutý multimetr může vést ke zpětnému proudu do desky/PC, stejně tak generátor.

Možnosti rozšíření

1. Ovládejte otáčky motoru přímo pomocí potenciometru.
2. Pokud nemáte vhodný motor, s využitím integrovaného kontaktního nepájivého pole připojte a ovládejte externě napájený motor.

3.1.5 SEGMENTOVÝ DISPLEJ

Sedmisegmentový displej můžeme spatřit na starších modelech pokladen, kalkulaček, budíků, tachometrů, dále elektronických terčích na šipky, čerpacích stanicích. Kdekoliv, kde postačí jednoduché zobrazení čísel.

Zadání

Zapojte segmentový displej.

Vytvořte program počítající čas v sekundách, kde:

1. Použijete knihovnu na ovládání displeje,
2. Vytvoříte vlastní metodu pro správné zobrazení číslic na displeji.

Použité moduly a knihovny

Segmentový displej; LedControl

Postup řešení

Nejprve zapojíme segmentový displej pomocí přepínačů u Arduina, ale i mezi LED a samotným displejem.

V prostředí Arduino importujeme pomocí manažera knihovnu LedControl, kterou následovně zahrneme do programu pomocí příkazu `#include`. Deklarujeme displej objektem z knihovny `LedControl lc = LedControl(10, 12, 11, 1)`, teď se můžeme přímo odkazovat na náš objekt displeje přes tečkovou notaci. V úvodním nastavení zapneme `lc.shutdown(0, false)`, zvolíme intenzitu podsvícení `lc.setIntensity(0, 15)` a vyčistíme jej `lc.clearDisplay(0)`.

Pro zobrazovací metodu využijeme algoritmus procházející odzadu všechny pozice na displeji a postupně vypisuje číslice.

Kód

```
// Použité knihovny: LedControl
#include <LedControl.h>

// Deklarace displeje
// LedControl(Din, CLK, Load, # čipů)
LedControl lc = LedControl(10, 12, 11, 1);

void setup()
{
  // Probuzení displeje
  // shutdown(adresa, T/F)
  lc.shutdown(0, false);

  // Nastavení podsvícení
  // setIntensity(adresa, <0-15>)
  lc.setIntensity(0, 15);

  // Smazání displeje
  // clearDisplay(adresa)
  lc.clearDisplay(0);
  Serial.begin(9600);
}

// Metoda pro zobrazení čísel na displej
// Zobrazuje čísla na správné pozici
void zobrazCislo(int in_cislo)
{
  int cislo;

  for (int i = 3; i >= 0; i--)
  {
    cislo = in_cislo % 10;
    // in_cislo = in_cislo / 10;
    in_cislo /= 10;
    // .setDigit(adresa, pozice, číslo, desetinná tečka T/F)
    lc.setDigit(0, i, cislo, false);
  }
}

void loop()
{
  for (int i = 0; i < 9999; i++)
  {
```

```
    zobrazCislo(i);
    delay(100);
  }
}
```

Cíle

Použití knihovny, tvorba vlastní metody.

Možné problémy

LED a segmentový displej sdílí sběrnici, proto je nutné dávat pozor na pozici přepínačů mezi nimi. Vlastní funkce musí předcházet blok, kde je volána.

Možnosti rozšíření

1. Zobrazte hodnoty potenciometru.
2. Vytvořte stopky.

3.1.6 LCD DISPLEJ

Tento typ displeje vypisuje každý znak zvlášť do svého políčka. Můžeme se s ním setkat u jednodušších pokladen, platebních terminálů, rádií. Využívá se také jako stavový displej u elektronických zařízení.

Zadání

Seznamte se I^2C sběrnicí a jejím využitím v zapojení periferií na desce TinyLab.

Zapojte LCD displej a napište program, který vypíše text zadaný do konzole na displej.

Použité moduly a knihovny

LCD displej LTI_TYPE_MCP23008; Wire, LiquidTWI2

Postup řešení

V tomto úkolu využijeme I^2C sběrnici, které jsou vymezeny porty D2-3. Pro správnou funkcionalitu zařízení připojených na sběrnici musíme zahrnout knihovnu `Wire.h` a další pro vlastní ovládání displeje `LiquidTWI2.h`. V deklaraci displeje jako parametr uvedeme adresu na I^2C sběrnici `LiquidTWI2 lcd(0x20)`.

V sekci `setup()` musíme nastavit typ displeje `lcd.setMCPType(LTI_TYPE_MCP23008)`, počet řádků a sloupců `lcd.begin(počet znaků/řádek, počet řádků)`, podsvícení `lcd.setBacklight(LOW/HIGH)`, vymazat jej `lcd.clear()` a nastavit kurzor na počátek (levý horní roh) `lcd.setCursor(0, 0)`.

Metoda `loop()` nám obstará opakované čtení vstupu a jeho vypsání na displej. Vytvoříme si proměnnou pro zobrazovaný text, do které rovnou přiřadíme text z konzole a tím ji vyčistíme. Vybídneme uživatele k zadání textu a počkáme na jeho vstup vložením cyklu `while (Serial.available() == 0) {}` s prázdným tělem. Tím docílíme to, že program nebude pokračovat, dokud uživatel nevloží text k zobrazení. Ten poté načteme a zobrazíme na LCD displeji.

Kód

```
// Použité knihovny: I2C komunikace, ovládání displeje
#include <Wire.h>
#include <LiquidTWI2.h>

// Deklarace displeje
// LiquidTWI2 <název>(adresa sběrnice)
LiquidTWI2 lcd(0x20);

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  // Inicializace displeje: Nastavení typu
  lcd.setMCPType(LTI_TYPE_MCP23008);
  // Spuštění a nastavení begin(# znaků, # řádků)
  lcd.begin(16, 2);
  // Spuštění podsvícení
  lcd.setBacklight(HIGH);
  // Smazání displeje
  lcd.clear();
  // Nastavení kurzoru
  lcd.setCursor(0, 0);
}

void loop()
{
  // Vyčistíme Serial tak, že jej načteme do proměnné
  String text = Serial.readString();

  Serial.println("Zadej text k zobrazeni na displeji:");

  while (Serial.available() == 0)
  {
    // Prázdná smyčka čekající na vstup
  }

  text = Serial.readString();
  Serial.println(text);
  lcd.clear();
  lcd.print(text);
  delay(1000);
}
```

Cíle

Seznámit se I^{2}C sběrnicí, způsobem a možnostmi zobrazení znaků a znakových sad.

Možné problémy

Vyčistit Serial a správně zapsat část kódu, kde se čeká na vstup uživatele.

Možnosti rozšíření

1. Vytvořte metodu pro výpis textu do každého řádku zvlášť.
2. Stavový displej: zobrazení času, teploty, světelnosti atp.

3.2 VSTUPNÍ ZAŘÍZENÍ

3.2.1 TLAČÍTKA

Tlačítko je jeden z nejběžnějších elektronických ovládacích prvků, který po dobu stisku sepíná nebo naopak rozepíná obvod. Setkáme se s nimi na každém kroku, téměř každý kousek elektroniky obsahuje tlačítko k nastavení nebo slouží jako ovládání pro spárovaný přístroj.

Zadání

Prostudujte schéma zapojení tlačítek TinyLab. Vysvětlete pojmy pull-up a pull-down.

Zapojte všechna tlačítka. Prozkoumejte hodnoty tlačítek ve všech stavech.

Napište program, který zaznamená stisknutí každého tlačítka do konzole.

Použité moduly a knihovny

Tlačítka

Postup řešení

Nejprve si vytvoříme definice pro porty propojené s tlačítky. Všimněme si, že digitální tlačítka mají každé vlastní port, kdežto analogová sdílejí jeden.

Opakující se část kódu bude sledovat stavy tlačítek a vypisovat informaci o stisknutí tlačítka. Digitální tlačítka jsou zapojena s pull-up rezistorem, tudíž sepínají s hodnotou log0. Analogová tlačítka jsou odlišná. Sdílejí analogový port, ale každé je zapojeno s rezistorem jiné hodnoty. Tímto se stisk jednoho tlačítka bude lišit hodnotou od druhého. Dohromady budou dokonce ukazovat ještě vyšší hodnotu. Tohoto využijeme při tvorbě algoritmu pro detekci stisknutých tlačítek.

Kód

```
// Definice pinů pro tlačítka
#define TLACITKO1 9
#define TLACITKO2 8
```

```
#define TLACITKO3 A5

void setup()
{
  // Nastavení módu pinů
  pinMode(TLACITKO1, INPUT);
  pinMode(TLACITKO2, INPUT);
  pinMode(TLACITKO3, INPUT);

  // Spuštění čtení pinů v hodnotě 9600 baudů
  Serial.begin(9600);
}

void loop()
{
  // Digitální tlačítka jsou zapojeny pull-up, tudíž spínají do 0
  if (digitalRead(TLACITKO1) == 0)
  {
    Serial.println("Tlačítko 1 stisknuto.");
  }

  if (digitalRead(TLACITKO2) == 0)
  {
    Serial.println("Tlačítko 2 stisknuto.");
  }

  // Po pozorování zjistíme hodnoty při stisku tlačítka/tlačítek
  // Podle hodnot se rozhodneme
  int tlacitko3 = analogRead(TLACITKO3);
  switch (tlacitko3)
  {
    case 200 ... 220:
      Serial.println("Tlačítko 3 stisknuto");
      break;
    case 500 ... 520:
      Serial.println("Tlačítko 4 stisknuto");
      break;
    case 560 ... 580:
      Serial.println("Tlačítko 3 a 4 stisknuto");
      break;
  }

  delay(100);
}
```

Cíle

Porovnat způsoby čtení tlačítek, jejich hodnoty, vyzkoušet si různé způsoby rozhodování (if, switch).

Možné problémy

Dávejte si pozor při definici tlačítek. Hodnoty u analogových tlačítek se mírně liší mezi každou deskou.

Možnosti rozšíření

1. Rozsviťte LED nad tlačítkem při stisku.

3.2.2 POTENCIOMETR

Potenciometr běžně slouží k fyzické regulaci zejména výkonu. Využití je velmi běžné u ovládání hlasitosti u přehrávačů či intenzity podsvícení.

Zadání

Zapojte potenciometr a správně jej přiřaďte.

Pomocí programu zaznamenejte jeho hodnotu a vypište ji do konzole. Dále sledujte hodnoty v plotteru.

Zjistěte minimální a maximální hodnoty potenciometru, vysvětlete je.

Použité moduly a knihovny

Potenciometr

Postup řešení

Připojíme potenciometr tak, že přepneme příslušný přepínač na desce. Všimněme si, že ovládací port je označen ANALOG IN.

Pro manipulaci s Arduinem v reálném čase je nutné dát do metody `setup()` příkaz `Serial.begin(bity za sekundu)`.

Ve smyčce vytvoříme celočíselnou proměnnou, do které přiřadíme sériově čtenou analogovou hodnotu potenciometru pomocí příkazu `analogRead(číslo pinu)`. Následně hodnotu vypíšeme do konzole.

Pomocí klávesové zkratky **Ctrl + Shift + M** si otevřeme sériový monitor a poznamenáme hodnoty. Obdobnou zkratkou **Ctrl + Shift + L** si zobrazíme sériový plotter, který nám vykreslí graf.

Při vyhodnocování zaměříme pozornost na konzoli a plotter, zejména na frekvenci čtení a hodnoty.

Kód

```
// Definice konstanty pro analogový pin, který čte potenciometr
#define POTENCIOMETR A0

void setup()
{
  // Spuštění čtení pinů v hodnotě 9600 baudů
```

```
  Serial.begin(9600);
}

void loop()
{
  // Vytvoření proměnné pro čtení hodnoty potenciometru
  // a přiřazení hodnoty z pinu A0 analogRead(# pinu)
  int hodnota = analogRead(POTENCIOMETR);
  // Výpis hodnoty do konzole
  Serial.println(hodnota);
  // Zpoždění 10 ms
  delay(10);
}
```

Cíle

Důkladné seznámení se se součástkou, její funkcionalitou, měřitelnými hodnotami.

Možné problémy

Kromě minimální a maximální naměřené hodnoty dávejte pozor na rozpětí. Zaměřte se obnovovací frekvenci reportovaných údajů v konzoli/plotteru.

Možnosti rozšíření

1. Připojte LED a rozsviťte ji úměrně k hodnotě potenciometru, porovnejte.
2. Zobrazte hodnotu potenciometru na segmentovém displeji.

3.2.3 ROTAČNÍ ENKODÉR

Zadání

S pomocí knihy Hradla, volty, jednočipy, kapitola 20.6, od Martina Malého zjistěte, jak funguje rotační enkodér. Popište, jakým způsobem zaznamená otočení a na jakou stranu.

Nejdříve správně zapojte enkodér a sledujte jeho chování na sériovém plotteru.

Vytvořte program, který při otočení enkodéru sníží/zvýší hodnotu registru a při stisku tlačítka čítač vynuluje. Hodnotu vypište do konzole.

Použité moduly a knihovny

Rotační enkodér

Postup řešení

Nejprve přiřadíme porty enkodéru, dáváme pozor na rozdíly mezi digitálním a analogovým propojením. Zapojení digitálních portů provedeme pomocí INPUT_PULLUP, aby deska reagovala na změny při otočení.

Při čtení enkodéru si zapamatujeme poslední stav jednoho z digitálních portů enkodéru. Pokud se aktuální stav neshoduje s aktuálním, porovnáme stavy obou digitálních portů, podle toho zvýšíme/snížíme inkrement. Dále sledujeme stav tlačítka enkodéru, abychom mohli resetovat čítač.

Kód

```
// Deklarace pinů enkodéru
#define ENKODER1 A5   // Tlačítko enkodéru
#define ENKODER2 6    // Kolečko A
#define ENKODER3 7    // Kolečko B

// Deklarce proměnných
int posledniStav;
int citac = 0;

void setup()
{
  // Nastavení módů
  pinMode(ENKODER1, INPUT);
  pinMode(ENKODER2, INPUT_PULLUP);
  pinMode(ENKODER3, INPUT_PULLUP);

  Serial.begin(9600);
}

void loop()
{
  int stav = digitalRead(ENKODER2);

  // Sledování stavů a změna čítače podle jejich změn
```

```
  if (stav != posledniStav)
  {
    if (digitalRead(ENKODER3) != stav)
    {
      citac++;
    }
    else
    {
      citac--;
    }
  }

  // Reset čítače
  if (analogRead(ENKODER1) != 0)
  {
    citac = 0;
  }

  posledniStav = stav;

  Serial.println(citac);
  delay(10);
}
```

Cíle

Uvědomit si princip fungování modulů, využití doprovodné literatury k tvorbě algoritmu.

Možné problémy

Rotační enkodér je připojen na digitální i analogový vstup, správně přiřaďte a čtěte vstupy.

Nastavení pinMode na INPUT_PULLUP.

Možnosti rozšíření

1. Seznamte se systémovým přerušením a přiřaďte jej enkodéru.

3.2.4 FOTOREZISTOR

S fotorezistorem se každý den setká skoro každý z nás. Naprostá většina moderních mobilních telefonů jej využívá pro automatické nastavení podsvícení. Je taktéž využíván jako senzor pro automatické osvětlení, kde se sníženou viditelností zapne světla.

Zadání

Zapojte fotorezistor. Vytvořte program pro čtení hodnot rezistoru.

Vymyslete aplikaci a její možnost realizace s pomocí desky TinyLab. Využijete k tomu dokumentaci nebo knihovny pro ovládání součástek.

Použité moduly a knihovny

Fotorezistor

Postup řešení

Tento příklad slouží k diskusi a experimentování s deskou a jejími komponenty.

Můžeme vytvořit program pro rozsvěcování LED při snížené světelnosti, případně rozsvěcovat více LED. Dále můžeme upravovat jas LED nebo segmentového displeje.

Kód

```
// Definice konstanty pro pin fotorezistoru
#define FOTOREZISTOR A2

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

void loop()
{
  Serial.println(analogRead(FOTOREZISTOR));
  delay(100);
}
```

Cíle

Využití znalosti funkce komponent k vytváření ovládacího algoritmu. Využití v praxi s použitím již známých komponent.

Možné problémy

Ne všechny periferie mají možnost ovládání jasu.

Možnosti rozšíření

1. Při snížené světelnosti rozsviťte LED.
2. Simulujte automatickou úpravu jasu u LED nebo segmentového displeje.

3.2.5 TEPLOTNÍ SENZOR

S teplotními senzory se setkáme na mnoha místech. V kuchyni jsou zastoupeny vpichovými teploměry nebo automatickými regulátory teploty v elektrických hrncích a rýžovarech. Slouží jako ochrana proti extrémním teplotám nejen u průmyslových strojů, ale i v mobilních telefonech a počítačích.

Zadání

K elektronickým součástkám a integrovaným obvodům jsou dodávány tzv. datasheety. Pomocí názvu na senzoru si vyhledejte příslušný dokument.

Zjistěte, jak senzor funguje a jaké hodnoty dokáže poskytnout.

Vytvořte teploměr, který bude do konzole vypisovat naměřené hodnoty.

Použité moduly a knihovny

Teplotní senzor LM35

Programové bloky

Tabulka 20: map()

map(vstup, min_původní, max_původní, min_nové, max_nové)	Přemapování/přepočet hodnot

Postup řešení

Nejdříve si najdeme datasheet k teplotnímu senzoru LM35. Z něj se dozvíme, že senzor měří v inkrementech 1 °C za každých 10 mV. Deska Arduino operuje v rozmezí 0-5 V, tuto znalost využijeme v tvorbě algoritmu pro převod naměřenýchhodnot na teplotu.

Zapojíme senzor teploty. Dále si připravíme funkci, která bude přijímat a vracet celočíselnou hodnotu. Vstupní hodnotu vydělíme maximální měřitelnou hodnotou senzorem 1024.0, číslo napíšeme s desetinnou tečkou, aby v průběhu výpočtu nedošlo k zaokrouhlení. Dále hodnotu vynásobíme 5000, které odpovídají rozmezí napětí na desce v milivoltech. Nakonec vydělíme 10, to odráží inkrement 1 °C. Výsledná rovnice:

```
Teplota = naměřená hodnota / 1024.0 * 5000 / 10
```

Hodnotu vypíšeme do konzole. Nakonec můžeme porovnat hodnoty vypočítané naší funkcí s nativní funkcí map().

Kód

```
// Definice konstanty pro pin teplotního senzoru
#define SENZOR A3

// Funkce pro převod naměřené hodnoty na stupně Celsia
int prepocetTeploty(int in_hodnota)
{
  // 1. vstupní hodnotu vydělíme maximální měřitelnou hodnotou,
  //    je nutné použít desetinné číslo, aby nedošlo k zaokrouhlení
  // 2. Vynásobíme maximální hodnotou napětí na desce (5 V = 5000 mV)
  // 3. Vydělíme inkrement, po kterých senzor měří (10 mV odpovídá 1 °C)
  return in_hodnota / 1024.0 * 5000 / 10;
}

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

```
void loop()
{
  int teplota = analogRead(SENZOR);

  Serial.print("Teplota pomocí funkce: ");
  Serial.println(prepocetTeploty(analogRead(SENZOR)));

  // map(vstup, min_původní, max_původní, min_nové, max_nové)
  teplota = map(teplota, 0, 1023, 0, 500);
  Serial.print("Teplota pomocí map: ");
  Serial.println(teplota);

  delay(1000);
}
```

Cíle

Naučit se využívat dokumentace od výrobce, a tak správně využít danou součástku.

Možné problémy

Ve vlastní funkci je nutné dělit desetinným číslem, jinak bude výsledek zkreslený.

Možnosti rozšíření

1. Namísto vlastní metody využijte nativní funkci map(), porovnejte hodnoty.
2. Vypište teplotu na segmentový/LCD displej.

3.2.6 RTC

Real Time Clock neboli hodiny reálného času, je komponenta generující časový signál. Tento způsob je spolehlivější a energeticky konzervativnější. Standardně má RTC možnost napájení z knoflíkové baterie, která vydrží až tři roky. Tím se zachová informace o čase i při absenci napájení desky.

Toho se dá využít v aplikacích, kde nemáme možnost Arduino napájet ze sítě. RTC si dokáže nastavit „alarm“, který v určitý čas desku probudí ze spánku. Například otevírání dvířek kurníku v určitý čas a opětovné zavření v jiný.

Zadání

Seznamte se s RTC a zapojte jej. Vytvořte program, který:

1. bude obsahovat metody pro nastavení hodin, pokud již nejsou,
2. vypíše aktuální času a datum do konzole,
3. v implementaci využijte nativní konstanty.

Použité moduly a knihovny

RTC DS1307; Wire, RTCLib

Postup řešení

Po zapojení RTC a zahrnutí knihoven pro I^2C sběrnici a vlastní hodinový modul, deklarujeme hodiny pomocí typu: `RTC_DS1307 rtc`.

Dále vytvoříme metodu pro nastavení hodin, kde nejprve zkontrolujeme, zdali hodiny neběží `rtc.isrunning()`. Pokud tomu tak není, hodiny „seřídíme“ pomocí `rtc.adjust(DateTime(__DATE__, __TIME__))`. Tento složený příkaz si rozebereme: `rtc.adjust(datum a čas)` hodiny nastaví, `DateTime(datum, čas)`, `__DATE__` a `__TIME__` jsou nativní konstanty (definice) reprezentující aktuální datum a čas.

Následující metoda čas vypíše do konzole. Nejprve si do proměnné typu `DateTime` přiřadíme aktuální čas z RTC pomocí `rtc.now()`. Z naší proměnné poté extrahujeme jednotlivé prvky času a data (hodiny, minuty, sekundy, dny, měsíce a roky) přes tečkovou notaci.

Kód

```
// Použité knihovny: Wire, RTC
#include <Wire.h>
#include <RTClib.h>

// Deklarace RTC (RTC_model)
RTC_DS1307 rtc;

// Metoda pro nastavení hodin (po absenci napájení RTC)
void hodinyNastaveni()
{
  // Pokud hodiny neběží (kvůli absenci nastavení)
  // Nastaví se datum a čas
  if (!rtc.isrunning())
  {
    Serial.println("Hodiny neběží. Probíhá nastavení.");
    // adjust(čas + datum), v tomto případě
    // DateTime((definice data), (definice času))
    rtc.adjust(DateTime(__DATE__, __TIME__));
  }
}

// Metoda pro výpis času
void hodinyVypis()
{
  DateTime ted = rtc.now();

  Serial.print("Aktuální čas: ");
  Serial.print(ted.hour());
  Serial.print(':');
  Serial.print(ted.minute());
```

```
  Serial.print(':');
  Serial.print(ted.second());
  Serial.print(' ');
  Serial.print(ted.day());
  Serial.print('.');
  Serial.print(ted.month());
  Serial.print('.');
  Serial.println(ted.year());
}

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

  rtc.begin();

  hodinyNastaveni();
  hodinyVypis();
}

void loop() {}
```

Cíle

Seznámit s funkcí a využitím RTC v praxi. Využití nativních konstant.

Možné problémy

Je nutné hodiny inicializovat. Po absenci napájení se musí hodiny opět nastavit. Je možné, že se bude muset počkat na sériovou linku.

Možnosti rozšíření

1. Využijte RTC k buzení desky.
2. Vytvořte budík.

3.3 VSTUPNĚ VÝSTUPNÍ ZAŘÍZENÍ

3.3.1 EEPROM

Electrically Erasable Programmable Read-Only Memory, elektronicky mazatelná programovatelná paměť pouze pro čtení. Tento typ nevolatilní paměti slouží převážně pro ukládání parametrů pro spuštění programů nebo k uložení stavu před vypnutím programu. I když má paměť poměrně vysokou životnost, k logování většího objemu dat jsou uzpůsobené jiné typy pamětí.

EEPROM je obsažena téměř v každém kusu elektroniky od chytrých karet, přes mikropočítače po samotné paměti typu flash.

Zadání

Seznamte se s různými typy pamětí, poté zapojte EEPROM.

Vytvořte program, který:

1. zkontroluje připojení EEPROM,
2. uloží vstup od uživatele do paměti,
3. následně jej vypíše do konzole,
4. o veškerých postupech informujte uživatele pomocí konzole.

Použité moduly a knihovny

EEPROM; Wire, I2C_eeprom

Programové bloky

Tabulka 21: .length(), char()

String.length()	Vrací délku řetězce znaků
char(int)	Vrátí znak reprezentovaný číslem

Postup řešení

Po zapojení EEPROM a importu knihoven nutných pro správnou funkcionalitu deklarujeme paměť `I2C_eeprom eeprom(0x50, I2C_DEVICESIZE_24LC256)`.

Metoda pro zápis vezme řetězec znaků jako argument. Kvůli zápisu do paměti potřebujeme znát i jeho délku, kterou zjistíme pomocí `.length()`. Jelikož se jedná o poměrně primitivní metodu, musíme každý znak zapsat postupně `eeprom.writeByte(adresa, znak)` a posouvat se. Pro zjednodušení budeme začínat na adrese 0.

Metoda čtení bude postupovat opačně a jako argument přijme délku zapsaného řetězce. Postupně bude číst každý byte na adrese paměti a překládat jeho číselnou hodnotu pomocí `char(číslo)`. Po přečtení celé délky vypíšeme řetězec do konzole.

Nastavovací metoda nejprve zkontroluje, zda je paměť připojena `eeprom.isConnected()`, pokud nebude, vyčkáme na připojení. Následně vyzveme uživatele k zadání řetězce znaků pro zápis na EEPROM, počkáme na vstup a zároveň si zjistíme jeho délku. Nakonec provedeme samotný zápis a čtení, při kterých uživatele informujeme o průběhu.

Kód

```
// Použité knihovny: I2C komunikace, ovládání EEPROM
#include <Wire.h>
#include <I2C_eeprom.h>

// Deklarace EEPROM (adresa, velikost (štítek IC))
I2C_eeprom eeprom(0x50, I2C_DEVICESIZE_24LC256);

// Metoda pro zápis do paměti
void eepromZapis(String in_text)
{
  int delka;

  delka = in_text.length();
  Serial.print("Zapisuji text o délce ");
  Serial.print(delka);
  Serial.println(" znaků do paměti.");

  if (delka != 0)
  {
    for (int i = 0; i < delka; i++)
    {
      // writeByte(místo v paměti, znak)
      eeprom.writeByte(i, in_text[i]);
    }
  }
}

// Metoda pro čtení z paměti
void eepromCteni(int in_delka)
{
  String cteni;

  Serial.println("Čtu paměť.");

  for (int i = 0; i < in_delka; i++)
  {
    // writeByte(místo v paměti)
    // Bloky musíme převést zpátky do znaků
    cteni += char(eeprom.readByte(i));
  }

  Serial.print("Přečtený text: ");
  Serial.println(cteni);
}

void setup()
{
  String text;
  int delka = 0;

  Serial.begin(115200);
  // Čekáme na start Serialu
  while (!Serial);

  // Inicializace EEPROM
  eeprom.begin();

  // Kontrola připojení
  if (!eeprom.isConnected())
```

```
  {
    Serial.println("CHYBA: EEPROM neběží/odpojena.");
    while (true);
  }

  Serial.println("EEPROM běží.");

  Serial.println("Zadej text pro zápis do paměti:");
  while (!Serial.available());
  text = Serial.readString();
  Serial.print("Zadaný text: ");
  Serial.print(text);
  delka = text.length();

  eepromZapis(text);
  eepromCteni(delka);
}

void loop() {}
```

Cíle

Čtení/zápis s využitím adresy. Interpretace přečtených binárních dat.

Možné problémy

Při výpisu dat z EEPROM nezapomeňme překládat data.

Možnosti rozšíření

1. Implementujte metodu pro naformátování paměti.
2. Vytvořte logy s pomocí RTC, ukládejte nastavení.

3.3.2 ČTEČKA SD KARET

Secure Digital karty jsou flash paměti kompaktních rozměrů. Toto miniaturní úložiště nabízí i stovky gigabajtů paměti. Používá se jako externí úložiště pro mobilní telefony, fotoaparáty, herní konzole, jednodeskové počítače.

Zadání

Seznamte se SPI a čtečkou SD karet a zapojte ji.

Vytvořte program, který:

1. zkontroluje a nastaví SD kartu,
2. vytvoří textový soubor a napíše do nej libovolný text,
3. otevře soubor a vypíše jeho obsah do konzole,
4. o veškerých postupech informujte uživatele pomocí konzole,

5. název textového souboru deklarujte jako globální proměnnou s neměnným názvem. Jak toho docílíte, proč je to vhodné a jakým způsobem se zapisuje?

Použité moduly a knihovny

Čtečka SD karet, SD karta; Wire, SD

Postup řešení

Po správném zapojení a importu knihoven si definujeme port pro komunikaci se čtečkou karet. Dále si vytvoříme globální proměnnou typu `File` pro soubor a jeho název, který opatříme klíčovým slovíčkem `const`, abychom jej nemohli v průběhu programu měnit. Jelikož je to konstanta, zapíšeme je velkými písmeny, tím je v kódu na první pohled zřejmé, o jaký typ jde.

Metoda pro kontrolu a nastavení karty zjistí, zdali čtečka komunikuje `SD.begin(SDCTECKA)`. Pokud tomu tak není, čekáme dokud tak nenastane.

Zápis provedeme tak, že otevřeme námi deklarovaný soubor v režimu zápisu `soubor = SD.open(SOUBOR_NAZEV, FILE_WRITE)`. Do souboru poté zapíšeme text stejným způsobem jako na sériovou linku `soubor.println(in_text)`. Nakonec nesmíme zapomenou soubor zavřít `soubor.close()`, jinak bychom k němu nemohli přistoupit.

Čtení provedeme obdobně jako zápis. Soubor otevřeme tentokrát v režimu čtení `soubor = SD.open(SOUBOR_NAZEV, FILE_READ)`. Soubor čteme, dokud je dostupný `while (soubor.available())` a obsah vypisujeme do konzole `Serial.print(soubor.read())`. Opět nezapomene soubor zavřít.

Kód

```
// Použité knihovny: SPI (dependence), SD
#include <SPI.h>
#include <SD.h>

// Deklarace pinu SD čtečky
#define SDCTECKA 4
// Deklarace souboru
File soubor;
String const SOUBOR_NAZEV = "test.txt";

// Metoda pro inicializaci SD karty
void sdNastaveni()
{
  Serial.println("Příprava SD karty.");
  if(!SD.begin(SDCTECKA))
  {
    Serial.println("Chyba karty.");
    while(true);
```

```
  }
  Serial.println("Karta připravena.");
}

// Metoda pro zápis na kartu
void sdZapis(String in_text)
{
  // Zápis do souboru
  // open(název, zápis/čtení)
  soubor = SD.open(SOUBOR_NAZEV, FILE_WRITE);
  // Pokud se soubor podařilo otevřít, zapíšeme do něj text
  if (soubor)
  {
    Serial.print("Zapisuji do souboru: ");
    Serial.println(in_text);
    soubor.println(in_text);
    // Po přečtení zavřeme soubor!
    soubor.close();
    Serial.println("Hotovo.");
  }
  else
  {
    Serial.println("Nepovedlo se otevřít soubor!");
  }
}

// Metoda pro čtení karty
void sdCteni()
{
  // Čtení souboru
  // open(název, zápis/čtení)
  soubor = SD.open(SOUBOR_NAZEV, FILE_READ);
  if(soubor)
  {
    Serial.println("Čtu ze souboru: ");

    while(soubor.available())
    {
      Serial.write(soubor.read());
    }
    soubor.close();
    Serial.println("Hotovo.");
  }
  else
  {
    Serial.println("Nepovedlo se otevřít soubor!");
  }
}

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

  String text = "Testovací text pro SD kartu.";

  sdNastaveni();
  sdZapis(text);
  sdCteni();
}
```

```
void loop() {}
```

Cíle

Manipulace s textovým souborem, princip fungování SPI, význam konstant.

Možné problémy

Nezapomeňte po manipulaci se souborem jej zavřít. Přepínače ICSP (mezi Arduino a nepájivým kontaktním polem) musí být přepnuty směrem k desce Arduino, jinak dojde k chybě čtení SD karty.

Možnosti rozšíření

1. Implementujte metodu pro naformátování paměti.
2. Vytvořte logy, ukládejte nastavení.

3.3.3 XBEE MODUL

XBee je komunikační protokol s širokým využitím a různými rolemi samotných modulů. Můžeme se s ním setkat v prvcích chytré domácnosti jako žárovky, spínače, zásuvky.

Zadání

Seznamte se s protokolem XBee, UART, programem XCTU, který případně nainstalujte.

Vytvořte program, který:

1. využije ke komunikaci linku Serial1/UART, kterému přiřadíte alias XB,
2. důkladně prozkoumejte komunikaci mezi TinyLab a XCTU.

Použité moduly a knihovny

XBee modul

Postup řešení

Nainstalujeme si program XCTU[15] a připojíme druhý modul do PC.

Xbee modul používá sběrnici UART, kterou sdílí s RF a BT moduly. Využívá digitální porty 0 a 1, v našem případě také RX (UART Receive) a TX (UART Transmit).

15 https://www.digi.com/resources/documentation/digidocs/90001526/tasks/t_download_and_install_xctu.htm

Pro komunikaci Arduino – PC obvykle používáme sériovou linku `Serial`. Abychom ale přenášeli informace přes XBee, musíme komunikovat zvlášť. Toho docílíme buď importem knihovny nebo využijeme praktickou linku `Serial1` ovládající pouze digitální porty 0 a 1, které jsou shodou okolností napojeny na XBee modul.

Protože `Serial` a `Serial1` Pro lepší orientaci v kódu si vytvoříme tzv. alias neboli zástupné pojmenování. Použijeme k tomu příkaz `#define XB Serial1`, po kterém můžeme volat metody `Serial` přes objekt `XB`.

V metodě `setup()` spustíme obě komunikační linky `Serial.begin(9600)` a s použitím aliasu `XB.begin()`.

Metoda loop() obstará komunikaci mezi oběma linkami tak, že pokud se něco vyskytne na sériové lince, vypíše obsah na XB linku a naopak.

Pomocí vstupu do konzole posíláme zprávy do PC. Sledujeme a analyzujeme komunikaci.

Kód

```
// Pro lepší čitelnost dáme Serial1 alias XB
// Serial1 operuje pouze na pinech 0, 1, kde je aktuálně připojený modul XBee
#define XB Serial1

void setup()
{
  // Otevření komunikace Arduino - USB
  Serial.begin(9600);
  // Otevření komunikace Arduino - XB
  // XBee modul funguje implicitně na 9600 baudech
  XB.begin(9600);
}

void loop()
{
  if(Serial.available())
  {
    XB.write(Serial.read());
  }
  if(XB.available())
  {
    Serial.write(XB.read());
  }
}
```

Cíle

Seznámit se s pojmem alias, ovládacími prvky XCTU a způsobem komunikace XBee.

Možné problémy

Je nutné nastavit XBee moduly pro jejich účel: přijímač/vysílač. Hlídejte správné nastavení komunikace mezi UART a sériovou linkou.

Možnosti rozšíření

1. Ovládejte více prvků.
2. Zprovozněte komunikaci mezi dvěma deskami.

3.3.4 WIFI MODUL

WiFi je dnes nedílnou součástí našich životů. K internetu se dnes připojuje téměř každý telefon, počítač, tablet, chytré hodinky, automobily, spínače žárovky nebo i rychlovarná konvice toaster.

Zadání

Seznamte s modulem WiFi, jeho zapojením pomocí UART, AT příkazy, dopřednou deklarací metod a zapojte modul.

Vytvořte program, který:

1. vypíše seznam AT příkazů, kterými uživatel bude schopen:
    a. najít WiFi sítě v okolí,
    b. připojit se k síti,
    c. získat informace o síti,
    d. odpojit se od sítě.
2. Vytvořte metodu pro komunikaci mezi UART/Serial1 a Serial.
3. Využije pro komunikaci UART sběrnici,
4. použije dopřednou deklaraci pro metodu komunikace mezi UART a Serial.

Použité moduly a knihovny

WiFi modul ESP8266

Postup řešení

Pro lepší orientaci v kódu si vytvoříme alias pro Serial1 `#define WF Serial1`.

Dopředná deklarace umožňuje vytvořit metodu/funkci ještě před jejím využitím v kódu s tím, že její tělo můžeme napsat dále v programu. Stačí ji definovat jako proměnnou

`void wifiKomunikace();`, nesmíme zapomenout na středník na konci. Tuto metodu umístíme na konec programu, za metodu `loop()`, standardním způsobem. Tělo metody bude sledovat komunikaci mezi linkami tak, že `Serial` bude sledovat `WF`, kterou vypíše a naopak.

V nastavovací metodě spustíme obě komunikační linky, `Serial` s frekvencí 9600 baudů a `Serial1/WF` s frekvencí 115200 baudů, na kterém operuje WiFi. Dále vypíšeme všechny příkazy a jejich význam.

V opakovací metodě zavoláme naši metodu pro komunikaci mezi linkami.

Kód

```
// Pro lepší čitelnost dáme Serial1 alias BT
// Serial1 operuje pouze na pinech 0, 1, kde je aktuálně připojený modul
WiFi
#define WF Serial1

// Dopředná deklarace, metodu popíšeme na konci
void wifiKomunikace();

void setup()
{
  // Otevření komunikace Arduino - USB
  Serial.begin(9600);
  // Otevření komunikace Arduino - WiFi
  // WiFi modul funguje implicitně na 115200 baudech
  BT.begin(115200);
  while (!Serial);

  Serial.println("Nezapomeň přepnout sériový monitor na 'Obojí NL
& CR!'");
  Serial.println("--------------------------+---------------------");
  Serial.println("Příkaz                    | Akce");
  Serial.println("--------------------------+---------------------");
  Serial.println("AT                        | Test, vrátí OK");
  Serial.println("AT+CWMODE=(1-3)           | Mód (Client/AP/C+AP");
  Serial.println("AT+CWLAP                  | Výpis sítí");
  Serial.println("AT+CWJAP=\"SSID\",\"PW\"      | Připojení k síti");
  Serial.println("AT+CIFSR                  | Info o připojení");
  Serial.println("AT+CWQAP                  | Odpojení od sítě");
  Serial.println("--------------------------+---------------------");
}

void loop()
{
  wifiKomunikace();
}

void wifiKonektivita()
{
  while (Serial.available())
  {
    WF.write(Serial.read());
  }
```

```
  while (WF.available())
  {
    Serial.write(WF.read());
  }
}
```

Cíle

Seznámení se s principem komunikace WiFi. Není vždy nutné používat knihovny k ovládání modulu.

Možné problémy

Špatná implementace metody pro komunikaci mezi deskou a WiFi modulem. Dávejte pozor na frekvenci komunikace TinyLab a komponenty WiFi.

Možnosti rozšíření

1. Vytvoření jednotlivých funkcí pro každý příkaz.

3.3.5 BLUETOOTH MODUL

S komunikačním protokolem Bluetooth se setkáváme v praktickém životě konstantně. Přes různá ovládání po chytré hodinky a bezdrátová sluchátka.

Zadání

Seznamte se s modulem Bluetooth, UART a programem PuTTY. Zapojte modul BT a LED. Pomocí Bluetooth propojte TinyLab s počítačem a navažte komunikaci programem PuTTY. Z něj pak vysílejte příkazy k rozsvícení a zhasnutí LED.

Použité moduly a knihovny

Bluetooth modul HS-05

Postup řešení

Nainstalujeme si program PuTTY[16].

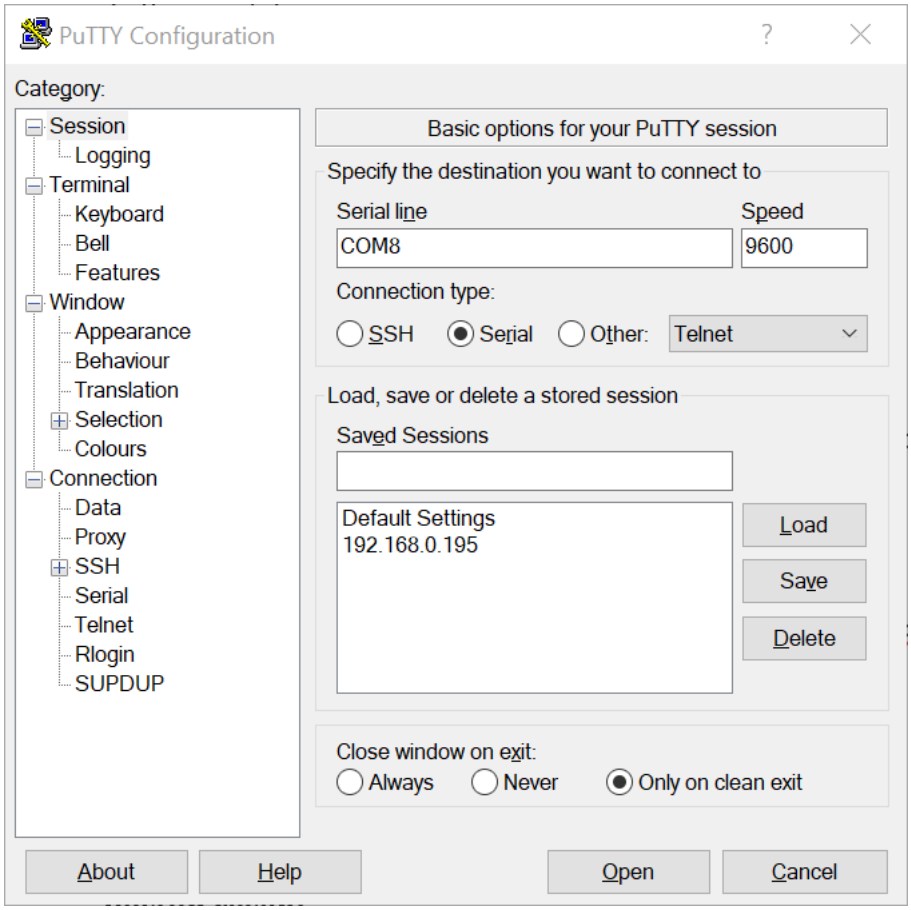
V programu si definujeme alias pro Serial1 `#define BT Serial1`, port pro LED `#define LED 10` a vytvoříme si celočíselnou proměnnou, která bude reprezentovat přenesená data.

[16] https://www.PuTTY.org/

V `setup()` nastavíme LED a spustíme sériové linky. Vybídneme druhou stranu (PC) k zadání hodnoty reprezentující ovládání LED.

Po zprovoznění Bluetooth modulu jej spárujeme s PC. V Ovládací panely\Všechny položky Ovládacích panelů\Zařízení a tiskárny najdeme modul HS-05 a zjistíme číslo jeho COM portu.

Spustíme program PuTTY a jeho pomocí se připojíme k modulu Bluetooth. V okně programu si můžeme všimnout implicitní rychlost sériové linky 9600 baudů. Po připojení se otevře okno, kde pomocí stisku 1/0 rozsvítíme/zhasneme LED.



Obrázek 12: PuTTY – konfigurace

Kód

```
// Definice konstanty pinu LED
#define LED 10
// Serial1 operuje pouze na pinech 0, 1, kde je aktuálně připojený modul
BT
#define BT Serial1
// Deklarace vstupu přes BT
int data;

void setup()
```

```
{
  // Otevření komunikace Arduino - USB
  Serial.begin(9600);
  // Otevření komunikace Arduino - BT
  BT.begin(9600);
  while (!Serial);

  pinMode(LED, OUTPUT);

  BT.println("Stiskni 1 pro rozsvícení a 0 pro zhasnutí LED.");
}

void loop()
{
  if (BT.available())
  {
    data = BT.read();
    if (data == '1')
    {
      digitalWrite(LED, HIGH);
      Serial.println("LED ON");
      BT.println("LED ON");
    }
    else if (data == '0')
    {
      digitalWrite(LED, LOW);
      Serial.println("LED OFF");
      BT.println("LED OFF");
    }
  }
  delay(100);
}
```

Cíle

Komunikace mezi TinyLab a PC. Využití programů pro komunikaci. Lokalizace komunikačních zařízení.

Možné problémy

Při psaní kódu si dávejme pozor na sériové linky, snadno se mohou plést.

Možnosti rozšíření

1. Opačná komunikace: Arduino – PC.
2. Zprovozněte komunikace mezi dvěma deskami.

3.3.6 RÁDIOVÝ MODUL

Radiofrekvenční komunikaci určitě známe díky stejnojmennému zařízení sloužícímu k přenosu hudby. Jeho původní využití bylo ke komunikaci mezi dvěma body a tento princip je stále zachován. Kromě komunikace dvou subjektů je často využita i ke komunikaci mezi zařízeními.

Příkladem jsou karty, přívěsky nebo samolepky s RF čipem. Vstupy do budov, kanceláří, zápis do elektronických píchacích hodin, ochrana před krádeží v obchodech.

Zadání

Tento příklad je pro dvojice.

Seznamte se s radiofrekvenčním modulem a způsobem funkce, dále s předávání dat odkazem. Zapojte RF modul.

Vytvořte program vysílače, který:

1. bude odesílat data přijímači,
2. data budou typu uint8_t a bude se inkrementálně zvyšovat.

Vytvořte program přijímače, který:

1. přijme vyslaná data,
2. vypíše je do konzole společně s jejich velikostí.

O veškerých postupech informujte uživatele pomocí konzole.

Použité moduly a knihovny

Rádiový modul; SPI, RF24

Programové bloky

Tabulka 22: sizeof()

sizeof(<proměnná>)	Vrací velikost argumanetu

Postup řešení

Po zapojení RF modulu importujeme knihovny RF24.h na ovládání modulu a SPI.h jako dependenci. Definujeme porty pro komunikaci `#define CE 8` a `#define CSN 9` a deklarujeme proměnné pro přijímač a vysílač `byte rfPrijimac[] = "prijimac"` a inicializujeme proměnnou pro přenos dat `uint8_t data = 0`, typ je zvolen kvůli metodě na přenos a čtení dat.

V `setup()` spustíme komunikační linku `Serial.begin(115200)` a počkáme, než začne. Spustíme také modul `rf.begin()` a nastavíme sílu signálu `rf.setPALevel(RF24_PA_LOW)`, velikost přenášených dat

`rf.setPayloadSize(sizeof(data))`, zapisovací linku `rf.openWritingPipe(rfVysilac)`, čtecí linku a poslech:

1. Vysílač
    a. `rf.openReadingPipe(1, rfPrijimac)`,
    b. `rf.stopListening()`
2. Přijímač
    a. `rf.openReadingPipe(1, rfVysilac)`,
    b. `rf.startListening()`.

V loop() pro vysílač vytvoříme proměnnou typu `bool`, která bude sledovat úspěšnost přenosu a zároveň zapisovat (vysílat) data `bool prenos = rf.write(&data, sizeof(data))`. Všimněme si ampersandu u proměnná `&data`, ten znamená referenci/odkaz na objekt. Tím si zajistíme, že předaný parametr nebude předán jako kopie, tudíž může být inkrementován.

U přijímače si vytvoříme proměnnou přenosu stejného typu jako data, `uint8_t prenos`. Pokud bude RF linka dostupná `rf.available(&prenos)` opět si všimněme reference, zjistíme velikost `uint8_t velikost = rf.getpPyloadSize()`. Přečtená data `rf.read(&data, velikost)`, reference nám umožní dosadit data jako argument a zároveň jim přiřadit hodnotu, jelikož se, opět, nepředá kopie.

Kód

Vysílač

```
// Použité knihovny: RF24, SPI (dependence)
#include <SPI.h>
#include <RF24.h>

// Definice konstant pinů RF
#define CE 8
#define CSN 9
// Deklarace vysílače <název>(CE, CSN/CS)
RF24 rf(CE, CSN);
// Deklarace přijímače a vysílače
byte rfPrijimac[] = "prijimac";
byte rfVysilac[] = "vysilac";
// Deklarace přenášených dat
uint8_t data = 0;

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

```
  while (!Serial);
  // Spuštění RF modulu
  rf.begin();
  // Kontrola funkčnosti
  if (!rf.begin())
  {
    Serial.println("RF modul nefunkční/odpojen.");
    while (true);
  }

  // Nastavení síly signálu
  rf.setPALevel(RF24_PA_LOW);
  // Nastavení velikosti přenášených dat
  rf.setPayloadSize(sizeof(data));
  // Nastavení vysílače
  rf.openWritingPipe(rfVysilac);
  // Nastavení přijímače
  rf.openReadingPipe(1, rfPrijimac);
  // Nastavení RF modulu do vysílače
  rf.stopListening();
}

void loop()
{
  // Odešleme data a zároveň zjistíme, jestli by přenos úspěšný
  // .write(ref data, velikost data)
  bool prenos = rf.write(&data, sizeof(data));
  if (prenos)
  {
    Serial.print("Odeslaná data: ");
    Serial.println(data);
    data += 1;
  }
  else
  {
    Serial.println("Přenos nevyšel");
  }
  delay(1000);
}
```

Přijímač

```
// Použité knihovny: RF24, SPI (dependence)
#include <SPI.h>
#include <RF24.h>

// Definice konstant pinů RF
#define CE 8
#define CSN 9
// Deklarace vysílače <název>(CE, CSN/CS)
RF24 rf(CE, CSN);
// Deklarace přijímače a vysílače
byte rfPrijimac[] = "prijimac";
byte rfVysilac[] = "vysilac";
// Deklarace přenášených dat
uint8_t data = 0;

void setup()
{
  Serial.begin(115200);
  while (!Serial);
  // Spuštění RF modulu
```

```
  rf.begin();
  // Kontrola funkčnosti
  if (!rf.begin())
  {
    Serial.println("RF modul nefunkční/odpojen.");
    while (true);
  }

  // Nastavení síly signálu
  rf.setPALevel(RF24_PA_LOW);
  // Nastavení velikosti přenášených dat
  rf.setPayloadSize(sizeof(data));
  // Nastavení vysílače
  rf.openWritingPipe(rfVysilac);
  // Nastavení přijímače
  rf.openReadingPipe(1, rfVysilac);
  // Nastavení RF modulu do přijímače
  rf.startListening();
}

void loop()
{
  // Parametr přenosu musí být uint
  uint8_t prenos;
  // Kontrola přenosu
  if (rf.available(&prenos))
  {
    // Velikost dat přenosu
    uint8_t velikost = rf.getPayloadSize();
    // Přečtení a dosazení do dat, proto reference
    rf.read(&data, velikost);
    Serial.print("Přijata data: ");
    Serial.print(data);
    Serial.print(" o velikosti ");
    Serial.print(velikost);
    Serial.println(" bajtů.");
  }
}
```

Cíle

Vyzkoušet si komunikace mezi dvěma zařízeními. Seznámit se s předávání dat odkazem.

Možné problémy

Dejte si pozor na nastavení přijímače a vysílače. Vzdálenost mezi moduly může způsobit problémy.

Možnosti rozšíření

1. Vytvořte skener okolních vysílačů.

3.3.7 NEPÁJIVÉ KONTAKTNÍ POLE

Hlavní účel nepájivého kontaktního pole (anglicky breadboard) je prototypování. Umožňuje zapojení součástek do desky bez nutnosti pájení. Standardně je deska rozdělena vejpůl a

dírky kolmo na střed jsou propojené. Otvory na vnější delší straně jsou propojeny vodorovně s dělicí čarou a obvykle slouží pro napájení.

Na desce TinyLab je napájení po stranách – vlevo země, vpravo 5 V.

Zadání

Zapojte LED do nepájivého pole a pomocí programu diodu rozsviťte.

Jakým způsobem LED zapojíte? Je vhodné/nutné použít další součástky?

Použité moduly a knihovny

Nepájivé kontaktní pole, LED, (rezistor), propojovací dráty

Postup řešení

Zapojíme LED do nepájivého pole tak, že delší „nohu“ připojíme k digitálnímu portu Arduina a kratší do země vedle pole. V ideálním případě mezi zdroj a LED vložíme 1KΩ rezistor, abychom zvýšili životnost diody.

V programu si definujeme port LED `#define LED 2`, kterému v nastavení přiřadíme správný mód. Smyčka obsahuje pouze kód pro rozsvícení a zhasnutí diody.

Nezapomeňme, že přepínač u portu musí být tentokrát v poloze blíže Arduinu.

Kód

```
// Definice konstanty pro pin LED
#define LED 2

void setup()
{
  pinMode(LED, OUTPUT);
}

void loop()
{
  digitalWrite(PIN, HIGH);
  delay(1000);
  digitalWrite(PIN, LOW);
  delay(1000);
}
```

Cíle

Vyzkoušet si zapojení neintegrovaných periferií.

Možné problémy

Dávejme pozor na polaritu u LED a správnou polohu přepínače na desce.

Možnosti rozšíření

1. Zapojte více LED ve tvaru dekorativního osvětlení s doprovodným řídicím programem.
2. Zapojte obvod pro ovládání motoru pomocí externího napájení a tranzistoru.

4 KOMPLEXNÍ ÚLOHY

Následující úlohy využívají více senzorů, pokročilejší styl programování nebo oboje. Jsou obtížnější než rozšiřující náměty předešlých úloh, které si můžeme stáhnout na serveru github: https://github.com/karbans/Arduino_TinyLab/tree/main/04_Pokrocile_ulohy.

4.1 HODINY S TEPLOMĚREM

Zadání

Proměňte TinyLab v informační centrum vypisující přehledně aktuální čas a teplotu v místnosti.

V řešení se pokuste se využít:

1. vlastní funkci pro každou informaci,
2. formátovaný řetězec.

Použité moduly a knihovny

Displej, RTC, teplotní senzor; Wire, LiquidTWI2, RTCLib

Postup řešení

Zapojíme všechny potřebné moduly a importujeme knihovny. S poznatky z předchozích úkolů dáme dohromady jednotlivé bloky kódu.

Pro lepší čitelnost a menší počet řádků využijeme k zobrazení času formátovaný text `sprintf(cas, "%02d:%02d:%02d\0", ted.hour(), ted.minute(), ted.second());`, resp. pole znaků `char cas[9]`. Počet znaků v poli je o jeden delší, jelikož v jazyce C je ukončen pomocí `\0`, neboli null-terminated. K ulehčení práce si explicitně přetypujeme pole na `String`, jelikož předávání polí je poněkud neobratné. Obdobný postup využijeme i data a teploty.

Nakonec všechny hodnoty zobrazíme na displeji, každou ve svém kvadrantu.

Kód

```
#include <Wire.h>
#include <LiquidTWI2.h>
#include <RTClib.h>

#define SENZOR A3

LiquidTWI2 lcd(0x20);
RTC_DS1307 rtc;

void hodinyNastaveni()
```

```
{
  if (!rtc.isrunning())
  {
    Serial.println("Hodiny neběží. Probíhá nastavení.");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}

String hodinyVypis()
{
  DateTime ted = rtc.now();
  char cas[9];

  sprintf(cas, "%02d:%02d:%02d\0", ted.hour(), ted.minute(),
ted.second());

  return (String)cas;
}

String datumVypis()
{
  DateTime ted = rtc.now();
  char datum[11];

  sprintf(datum, "%02d.%02d.%4d\0", ted.day(), ted.month(), ted.year());

  return (String)datum;
}

String teplotaVypis()
{
  char teplotaC[6];
  int teplota = map(analogRead(SENZOR), 0, 1023, 0, 500);

  sprintf(teplotaC, "%3d C\0", teplota);

  return (String)teplotaC;
}

void displejVypis()
{
  lcd.setCursor(0, 0);
  lcd.print(datumVypis());
  lcd.setCursor(0, 1);
  lcd.print(hodinyVypis());
  lcd.setCursor(11, 1);
  lcd.print(teplotaVypis());
}

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

  lcd.setMCPType(LTI_TYPE_MCP23008);
  lcd.begin(16, 2);
  lcd.setBacklight(HIGH);
  lcd.clear();
  lcd.setCursor(0, 0);
  rtc.begin();
  hodinyNastaveni();
}
```

```
void loop()
{
  displejVypis();
  delay(1000);
}
```

Cíle

Formátovaný výstup, přetypování proměnných, využití místa na displeji.

Možné problémy

Pozor na zástupné znaky ve formátovaném výpisu.

Možnosti rozšíření

1. Místo funkcí použijte metody, které si předávají ukazatele/pointery.

4.2 HLÍDAČ TEPLOTY SE SIGNALIZACÍ

Zadání

Kvůli bezpečnosti v sauně nebo inkubátoru se musí hlídat teplota. Vymyslete zapojení a vytvořte program, který spustí světelný i akustický alarm při jejím překročení.

Použité moduly a knihovny

LED, bzučák, teplotní senzor

Postup řešení

Po zapojení všech modulů a využitím znalostí z předchozích úloh a sestrojíme program.

Při rozsvícení LED a rozeznění bzučáku dáváme pozor na jejich synchronizaci. Příkaz `delay()` voláme až po nastavení kontrolek a bzučáku. Je vhodné vytvořit metodu pro reset, pokud se teplota opět dostane pod nastavenou hranici.

Pro testovací účely zvolíme takovou teplotu, kterou snadno dosáhneme prstem.

Kód

```
#define LED1 13
#define LED2 12
#define LED3 11
#define LED4 10
#define BZUCAK A1
#define TEPLOMER A3

int teplotaMax;

void alarm()
{
  digitalWrite(LED1, HIGH);
  digitalWrite(LED3, HIGH);
```

```
  digitalWrite(LED2, LOW);
  digitalWrite(LED4, LOW);
  tone(BZUCAK, 1000, 100);
  delay(100);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED4, HIGH);
  digitalWrite(LED1, LOW);
  digitalWrite(LED3, LOW);
  delay(100);
}

void alarmReset()
{
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  noTone(BZUCAK);
}

int teplota()
{
  return map(analogRead(TEPLOMER), 0, 1023, 0, 500);
}

void setup()
{
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(BZUCAK, OUTPUT);
  pinMode(TEPLOMER, INPUT);

  Serial.begin(9600);
  while (!Serial);

  Serial.println("Zadej teplotu, při které se spustí alarm:");
  while (!Serial.available())
  {
    teplotaMax = Serial.parseInt();
  }
  Serial.print("Maximální teplota nastavena na ");
  Serial.print(teplotaMax);
  Serial.println(" °C.");
}

void loop()
{
  Serial.print("Aktuální teplota: ");
  Serial.print(teplota());
  Serial.println(" °C.");

  while (teplota() >= teplotaMax)
  {
    alarm();
  }

  delay(200);
  alarmReset();
}
```

Cíle

Praktické využití teplotního senzoru. Synchronizace upozornění.

Možné problémy

Vhodně umístíme metodu alarmu do smyčky.

Možnosti rozšíření

1. Rozšiřte program o upozornění pomocí bezdrátového senzoru.
2. Při dosažení teploty se spustí větrák (DC motor).

4.3 LOGOVÁNÍ TEPLOTY

Zadání

Meteorologické stanice opakovaně měří a odesílají řadu údajů. Mezi nimi i teplotu, která může napovědět o dalším vývoji počasí. Aktivity, které se často a dlouhodobě opakují, jsou jako dělané pro stroje. Ty neřeší, zda je aktivita baví, nemají problém ji dělat opakovaně, třeba i každou minutu, ve dne v noci po celý rok.

Pomocí TinyLabu propojeného s počítačem vytvořte zapojení, které bude periodicky každou minutu do konzole i SD karty zapisovat teplotu doplněnou o aktuální čas (časovou známku). Program a jeho metody vhodně strukturujte. SD kartu vložte do PC a zkontrolujte provedené logy teplot.

Použité moduly a knihovny

Teplotní senzor, RTC, čtečka SD karet; SD, SPI, Wire, RTClib

Postup řešení

Po zapojení periferií vytvoříme dílčí metody programu. Vytvoříme si funkce na vracení času a teploty, které zavoláme v metodě pro zápis informací. Před zápisem na SD kartu se ujistíme, že je možné zapsat.

Zvolíme si vhodnou prodlevu mezi zápisy. Až nasbíráme dostatečné množství dat, zkontrolujeme obsah karty na PC.

Kód

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

```
#define SENZOR A3
#define SDCTECKA 4

RTC_DS1307 rtc;
File soubor;
String const SOUBOR_NAZEV = "Teploty.txt";
long const PERIODA = 60000;

void hodinyNastaveni()
{
  if (!rtc.isrunning())
  {
    Serial.println("Hodiny neběží. Probíhá nastavení.");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}

void sdNastaveni()
{
  Serial.println("Příprava SD karty.");
  if (!SD.begin(SDCTECKA))
  {
    Serial.println("Chyba karty.");
    while (true);
  }
  Serial.println("Karta připravena.");
}

String teplotaVypis()
{
  char teplotaC[6];
  int teplota = map(analogRead(SENZOR), 0, 1023, 0, 500);

  sprintf(teplotaC, "%3d C\0", teplota);

  return (String)teplotaC;
}

String hodinyVypis()
{
  DateTime ted = rtc.now();
  char cas[9];

  sprintf(cas, "%02d:%02d:%02d\0", ted.hour(), ted.minute(),
ted.second());

  return (String)cas;
}

void zapisTeploty()
{
  String info = hodinyVypis() + ": " + teplotaVypis();
  soubor = SD.open(SOUBOR_NAZEV, FILE_WRITE);
  if (soubor)
  {
    Serial.print("Zapisuji do souboru: ");
    Serial.println(info);
    soubor.println(info);
    soubor.close();
    Serial.println("Hotovo.");
  }
```

```
  else
  {
    Serial.println("Nepovedlo se otevřít soubor!");
  }
}

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

  rtc.begin();
  hodinyNastaveni();
  sdNastaveni();
}

void loop()
{
  zapisTeploty();
  delay(PERIODA);
}
```

Cíle

Využití logování při běhu programu.

Možné problémy

Pokud používáme formátovaný řetězec, je jednodušší jej převést na String pro snadnější předávání. Pokud i po kontrole zapojení SD karta nefunguje (neinicialuzuje se), ujistíme se, že přepínače mezi integrovanou deskou Arduino a nepájivým kontaktním polem, směřují k desce.

Možnosti rozšíření

1. Využijte nasbírané hodnoty ke zpracování denní statistiky.
2. Probouzejte Arduino před měřením a následně jej uspěte.
3. Evidujte i úroveň osvětlení.

ZÁVĚR

V první části práce jsme se seznámili s platformou TinyLab a Arduino na které je postavena. Dále moduly desky TinyLab. Jejich vlastnosti a signály. Dozvěděli jsme se, jak fungují a případně na jaké sběrnici. Zdali je potřeba knihovny pro správnou funkcionalitu a základní příkazy potřebných pro naši aplikaci. Všechny knihovny jsou doplněny o zdroj s jejich dokumentací, která je nutná pro další vývoj.

Druhou částí je Arduino IDE, neboli vývojové prostředí, ve kterém budeme pracovat, jeho možnostech a nástrojích, které jsme využili v praktických cvičeních, společně s jazykem, kterým jsou programy psány.

Třetí část je sadou praktických úloh. Nejprve jsme se seznámili s periferiemi a jejich využitím v každodenním životě. Následuje zadání příkladu spojeného s modulem, kde byl čtenář vyzván k rozvoji svých programovacích a elektrotechnických znalostí v podobě tvoření vlastního algoritmu s využitím dostupných dokumentů, implementace specifickým způsobem, komunikace s dalším zařízením atp. Součástí úkolů je návrh řešení s myšlenkovými procesy, odůvodněním, kódem, možnými chybami a jejich řešením. Úlohy jsou doplněny o pedagogické cíle a možnosti dalšího rozšíření.

Čtvrtá část rozšiřuje krátké aktivity o komplexnější projekty, které jsou navrženy pro více modulů a využití nabytých znalostí z předchozích cvičení. Lépe tak odrážejí reálné využití.

RESUMÉ

Práce seznamuje s prototypovací deskou TinyLab na bázi Arduino Leonardo a jejími dílčími moduly. Naučíme se pracovat s deskou a jejími periferiemi, programovat je pomocí Arduino IDE a to prostřednictví sady 19 cvičení a 3 komplexních projektů. Aktivity využívají LED, segmentové i LCD displeje, bzučáky, tlačítka, rotační enkodéry, teplotní senzory a fotorezistory, ale též i na znalosti náročnější komunikační Bluetooth a WiFi moduly či pro IoT navržené rozhraní XBee. Kromě vlastního zadání aktivity obsahují i návrh řešení, doprovodný kód, možná rozšíření, řešení potenciálních problémů.

RESUMÉ

This work introduces the reader with TinyLab prototyping board based on Arduino Leonardo and its individual modules. We learn how to work with the board and the peripherals, create programs using Arduino IDE via set of 19 exercises and 3 complex projects. Activities utilize LEDs, segment and LCD displays, buzzers, buttons, rotary encoders, thermal probes, photoresistors, more intermediate communication Bluetooth and WiFi modules, even IoT driven XBee interface. Assignments include solution, code, possible extensions, basic troubleshooting.

SEZNAM LITERATURY

1. ARDUINO. Arduino.cc. In: *Úvod* [online]. Turin: 5. Únor. 2018 [cit. 2022-Leden-23]. Dostupné z: https://www.arduino.cc/en/Guide/Introduction
2. SELECKÝ, M. *Arduino - Uživatelská příručka*. Brno: Albatros Media, a. s. 2016, 347 s.. ISBN 978-80-251-4849-5.
3. BLUM, J. *Exploring Arduino - Tools and Techniques for Engineering Wizardry*. Indianapolis: John Wiley & Sons, Inc. 2013, 413 s.. ISBN 978-1-118-54948-3.
4. MALÝ, M. *Hradla, volty, jednočipy - Úvod do bastlení*. Praha: CZ.NIC, z. s. p. o. 2017, 508 s.. ISBN 978-80-88168-26-3. Dostupné také z: https://knihy.nic.cz/files/edice/hradla_volty_jednocipy.pdf
5. MALÝ, M. *Porty, bajty, osmibity - Počítače na koleni*. Praha: CZ.NIC, z. s. p. o. 2019, 384 s.. ISBN 978-80-88168-39-3. Dostupné také z: https://knihy.nic.cz/files/edice/Porty__bajty_osmibity.pdf

SEZNAM OBRÁZKŮ, TABULEK, GRAFŮ A DIAGRAMŮ

PŘÍLOHY

Github repozitář

https://github.com/karbans/Arduino_TinyLab