

PRŮVODCE SVĚTEM ARDUINA

ZBYŠEK VODA & TÝM HW KITCHEN



Obsah

I Seznámení s Arduinem

1	O Arduinu	2
2	Typy desek	3
2.1	Arduino Mini	3
2.2	Arduino Nano	4
2.3	Arduino Micro	4
2.4	LilyPad Arduino	5
2.5	Arduino Fio	5
2.6	Arduino Uno	6
2.7	Arduino Leonardo	6
2.8	Arduino Yún	7
2.9	Arduino Mega2560	7
2.10	Arduino Due	8
2.11	Arduino Esplora	8
2.12	Arduino Robot	9
2.13	Arduino Intel Galileo	9
2.14	Arduino Tre	10
3	Arduino Shieldy	11
4	Arduino klony	12
II	Programujeme Arduino	13
5	Výběr a seznámení	15

6	Arduino IDE	17
6.1	Historie	17
6.2	Stažení a instalace	17
6.3	Používání	18
6.4	Programovací jazyk	19
7	Blikáme LED	20
7.1	Budeme potřebovat	20
7.2	Připojení Arduina a PC	20
7.3	Zapojení	21
7.4	Program	21
III	Základní struktury jazyka Wiring	22
8	Sériová komunikace	24
9	Proměnné	26
9.1	Práce s proměnnými	27
9.2	Datové typy	28
10	Pole	29
10.1	Deklarace pole	29
10.2	Přístup k hodnotám v poli	29
11	Digitální vstup a výstup	30
11.1	Vstup nebo výstup?	30
11.2	Ovládání výstupu	30
11.3	Čtení vstupu	31
12	Příklad: Tlačítko a LED dioda	32
IV	Pokročilejší struktury jazyka Wiring	34
13	Konstanty	36
13.1	Logické konstanty	36
13.2	Typ digitálního pinu	36
13.3	Proud na digitálních pinech	38

14 Analogový vstup a výstup	39
14.1 analogWrite()	39
14.2 analogRead()	40
15 Příklad: Regulace jasu LED	41
16 Podmínky	43
16.1 Porovnávací operátory	43
16.2 Složené podmínky	44
16.3 if()	45
16.4 else if()	45
16.5 else	45
16.6 Switch	46
17 Příklad: Pás LED diod	47
V Sériová komunikace a cykly	49
18 Sériová komunikace	51
18.1 Zahájení komunikace – Serial.begin()	51
18.2 Odeslání dat – Serial.print() a Serial.println()	52
18.3 Čtení dat – Serial.available() a Serial.read()	53
18.4 Ukončení komunikace – Serial.end()	54
19 Cykly	55
19.1 Složené operátory	55
19.2 Cyklus while()	56
19.3 Cyklus do...while()	57
19.4 Cyklus for()	58
20 Příklad: Had z LED diod	59
VI Užitečné funkce	60
21 Čas	62
21.1 delay()	62

21.2	delayMicroseconds()	63
21.3	millis()	63
21.4	micros()	63
22	Matematické funkce	64
22.1	Matematické operátory	64
22.2	min()	65
22.3	max()	65
22.4	abs()	66
22.5	constrain()	66
22.6	map()	66
22.7	pow()	67
22.8	sqrt()	67
22.9	Goniometrické funkce	67
22.9.1	sin()	68
22.9.2	cos()	69
22.9.3	tan()	70
23	Náhodná čísla	71
23.1	random() a randomSeed()	71
24	Příklad: Hrací kostka	72
VII	Uživatelsky definované funkce	75
25	Definice funkce	77
26	Volání funkce	78
27	Funkce, které vrací hodnotu	79
28	Převody datových typů	81
28.1	char()	81
28.2	byte()	81
28.3	int(), long(), float()	81

29 Zvuk a tón	82
29.1 tone()	83
29.2 noTone()	83
29.3 Příklad: Třítónový bzučák	84
30 Segmentové displeje	85
30.1 Sedmisegmentový displej	85
30.2 Vícesegmentové displeje	88
31 Příklad: Klavír	89
VIII Arduino jako klávesnice a myš	91
32 Úvod	93
33 Arduino Leonardo	94
34 Mouse	95
34.1 Mouse.click()	96
34.2 Mouse.move()	97
34.3 Mouse.press(), Mouse.release() a Mouse.isPressed()	97
34.4 Příklad: Myš	97
35 Keyboard	99
35.1 Keyboard.write()	99
35.2 Keyboard.press(), Keyboard.release() a Keyboard.releaseAll()	100
35.3 Keyboard.print() a Keyboard.println()	101
36 Arduino Esplora	102
36.1 Joystick	103
36.2 Směrová tlačítka	103
36.3 Lineární potenciometr	104
36.4 Mikrofon	105
36.5 Světelný senzor	105
36.6 Teploměr	105

36.7 Akcelerometr	106
36.8 Piezo bzučák	107
36.9 RGB LED	107
36.10 Neoficiální pinout	107
IX Processing	108
37 Úvod	110
38 Seznámení	111
39 Základ	112
39.1 Datové typy	112
39.2 Pole	112
39.3 Výpis hodnot	113
40 Kreslení	114
40.1 Vytvoření kreslicího plátna	114
40.2 Barvy	115
40.3 Bod	116
40.4 Úsečka	117
40.5 Čtyřúhelník	118
40.6 Zvláštní případy čtyřúhelníků	119
40.7 Trojúhelník	122
40.8 Elipsa a kružnice	123
40.9 Část kružnice a elipsy	124
41 Interakce s myší	125
41.1 Tlačítka myši	125
41.2 Poloha kurzoru	126
X Arduino a Processing	127
42 Úvod	129

43 Firmata	130
43.1 Nastavení	130
43.2 Propojení	130
43.3 Programování	131
44 Vlastní způsob komunikace	134
44.1 Sériová komunikace	134
45 Příklad: Ovládání bodu joystickem	138
46 Příklad: Graf hodnot ze slideru	140
 XI Propojujeme Arduino s jinými zařízeními	 142
47 Sériová linka	144
47.1 Propojení	144
48 Bluetooth	147
48.1 Odeslání a zpracování dat z potenciometru	148
49 Arduino a Android	149
50 Sběrnice i2c	152
50.1 Funkce pro práci s i2c	153
50.2 Přenos master ->slave	154
50.3 Přenos slave ->master	155
 XII Arduino a displeje	 157
51 Úvod	159
52 Maticové LED displeje	160
52.1 Teorie řízení	161
52.2 Zapojení	162
52.3 Programování	162
53 RGB teoreticky	165

54 Rainbowduino	166
54.1 Funkce	167
54.2 Propojujeme Rainbowduina	168
54.3 Zobrazení obrázku z PC	171
55 LCD displeje	177
55.1 Grafické monochromatické LCD	183
55.2 Barevné grafické LCD	185
XIII Projekt: 2048	191
56 SD karta	193
56.1 Příprava Arduina	194
56.2 Funkce	194
56.3 Příklad 1.: Zápis hodnot	196
56.4 Příklad 2.: Výpis dat ze souboru	197
57 Hra 2048	198
57.1 Hodnoty	198
57.2 Jdeme na to	199
XIV Arduino a Ethernet shield	210
58 Ethernet Shield	212
58.1 Funkce	214
58.2 Vytváříme server	216
58.3 Sosáme data	219
58.4 Ovládání přes síť	220
XV Náš první klon Arduina	223
59 Příprava Arduina	225
60 Čipy ATtiny	226

61 Čipy ATmega	229
XVI Projekt: Programátorská klávesnice	231
62 Keypad	233
62.1 Zapojení a programování	234
63 Bezpečnostní systém	236
64 Programátorská klávesnice	239
XVII Projekt: Robotická ruka	241
65 Servomotory	242
66 Robotická ruka	243
XVIII WiFi shield	251
67 Seznámení	252
68 Firmware shieldu	254
68.1 Zjištění verze firmware	254
68.2 Aktualizace firmware	255
69 Údaje potřebné pro připojení k WiFi	258
70 Přehled funkcí pro práci s WiFi	259
70.1 Třída WiFi	260
70.2 Třída WiFiServer	261
70.3 Třída WiFiClient	262
71 Příklady	263
71.1 Připojení k síti	263
71.2 Interakce se serverem	265
XIX Zdroje obrázků	267

V průběhu roku 2014 postupně vznikal na serveru HW Kitchen seriál článků o Arduinu. Postupně byly představeny základní dovednosti potřebné pro zvládnutí práce s ním. Seriál se také podrobně věnoval některým ze shieldů pro Arduino. Tato publikace obsahuje osmnáct dílů tohoto seriálu. Text byl mírně upraven pro potřeby ebooku, ale jinak zůstaly články nezměněny.

Milý čtenáři. I přes všechnu moji snahu není vyloučeno, že se do textu vloudily chyby. Pokud na některou narazíš, napiš mi prosím na email zbysekvoda@seznam.cz.

Část I

Seznámení s Arduinem

Když se v současné době začátečník podívá na trh s vývojovými platformami, může ho čekat nemilé překvapení. Existuje totiž celá řada více či méně vhodných desek a čipů, které výrobci nabízí. Počínaje samostatnými čipy (např. PICAXE), k jejichž programování stačí pouze sériový kabel a výkonnými platformami s možností běhu přizpůsobeného operačního systému konče. Ve světě asi nejrozšířenější platformou je Arduino. To nabízí různé typy desek od méně výkonných a malých modelů po kompletní soustavy obsahující USB, HDMI, Ethernet, či audio porty. V tomto článku si některé z desek představíme a povíme si, co dovedou.

Kapitola 1

O Arduinu

Vývoj prvního Arduina započal v roce 2005, když se lidé z italského Interaction Design Institute ve městě Ivrea rozhodli vytvořit jednoduchý a levný vývojový set pro studenty, kteří si nechtěli pořizovat, v té době rozšířené a drahé desky BASIC Stamp. Mezi studenty se Arduino uchytilo, a tak se tvůrci rozhodli poskytnout ho celému světu. (V roce 2010 vznikl zajímavý dokument o vzniku Arduina s rozhovory s jeho tvůrci: Arduino The Documentary (2010) English HD.) A to nejenom prodejem vlastních desek, ale i sdílením všech schémat a návodů (jedná se o Open Source projekt). Programová část Arduina byla založena na Processing, což je programovací jazyk s vlastním editorem, určený k výuce programování. V dnešní době se prodalo již několik stotisek desek Arduino. Důkazem, že tato platforma není mrtvá, může být i to, že nedávno byl ohlášén vývoj nové a výkonné desky Arduino Galileo, která vzniká ve spolupráci s Intelem. Za osm let vývoje již vzniklo spoustu různých typů Arduina. Jelikož se jedná o opensource projekt, vznikalo společně s hlavní linií projektu i spoustu dalších, neoficiálních typů, takzvaných klonů. Nejdříve si ale představíme oficiální desky.



Obrázek 1.1: Oficiální logo platformy Arduino

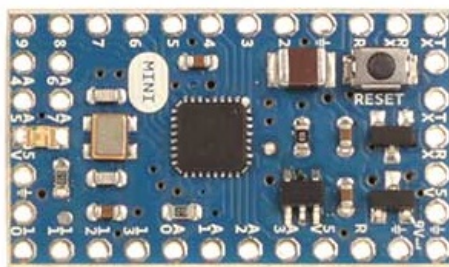
Kapitola 2

Typy desek

Srdcem každého Arduina je procesor od firmy Atmel, který je obklopen dalšími elektronickými komponenty. Pro celou řadu desek je typické jednotné grafické zpracování s převažující modrou barvou. V eshopech, i na oficiálních stránkách Arduino arduino.cc se můžeme setkat s deskami, které mají za svým názvem ještě přidáno například Rev3, nebo R3. Jedná se o číslo verze dané desky. Mezi jednotlivými verzemi se mohlo například mírně změnit rozložení součástek, nebo design. Nejedná se však o velké změny, které by si vyžádaly vznik další desky. Na většině desek je mimo hlavního čipu ještě převodník, který umožňuje komunikaci mezi PC (USB) a čipem. Setkám se však s typy, které převodník nemají. Může to být ze dvou důvodů. Prvním z nich je úspora místa a následná nutnost použití externího převodníku. Druhým typem jsou ty, jejichž čip má v sobě tento převodník zabudovaný.

Nyní si předvedeme jednotlivé desky, které jsou pro přehlednost seřazeny od těch nejmenších po největší.

2.1 Arduino Mini



Obrázek 2.1: Arduino Mini

Arduino Mini je asi nejmenší oficiální verze Arduina, navržena pro úsporu místa. Daní za malé rozměry je však absence USB portu. K programování je tedy nutné použít externí USB 2 Serial převodník. Jeho výkon však nijak nezaostává za většími deskami. Běží na procesoru ATmega328 (dříve ATmega168) s taktem 16 MHz. Pro své malé rozměry je vhodný k použití například v chytrých vypínačích, dálkových ovladačích a podobně.

2.2 Arduino Nano



Obrázek 2.2: Arduino Nano

Arduino Nano se od svého menšího sourozence výbavou moc neliší. Největším rozdílem je zde však přítomnost USB portu a převodníku, kvůli němuž je celkové provedení o něco větší. Odpadá tak nutnost mít společně s deskou ještě další programovací prostředek.

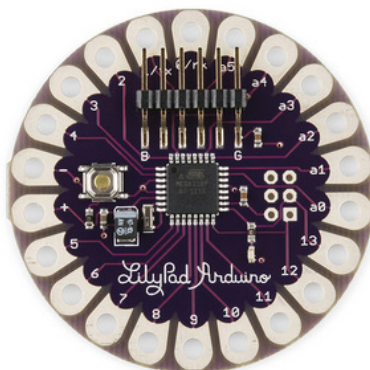
2.3 Arduino Micro



Obrázek 2.3: Arduino Micro

Arduino Micro je jedna z desek, která má čip obsahující převodník. Tímto čipem je ATmega32u4. Jeho výhodou je, že se může pro počítač tvářit jako myš, nebo klávesnice a posílat příkazy, jako jsou stisk klávesy a posunutí myši. To je sice možné i s ostatními deskami, ale tato operace vyžaduje přeprogramování převodníku (nejčastěji založeném na čipu ATmega16u2, nebo ATmega8u2), což nemusí být úplně jednoduché. S touto deskou je tedy velice jednoduché vytvořit si vlastní klávesnici, nebo herní ovladač.

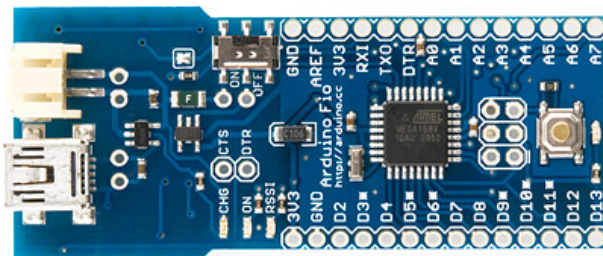
2.4 LilyPad Arduino



Obrázek 2.4: Arduino Lilypad

Již při prvním pohledu je jasné, že LilyPad Arduino není úplně typické. Jedná se totiž o verzi přizpůsobenou k nošení na textilu, kdy jsou spoje tvořeny vodivou nití. Tak se dá vyrobit například cyklistická mikina s přišitými blinkry. Existuje více druhů této desky. Můžeme se setkat s verzí s USB a čipem ATmega32u4, nebo bez USB ve verzi ATmega328 a dalšími.

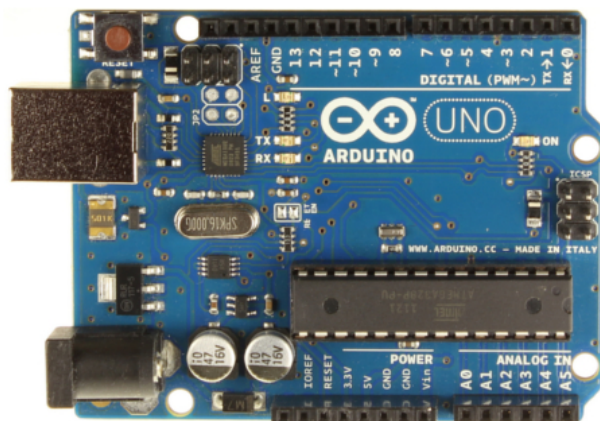
2.5 Arduino Fio



Obrázek 2.5: Arduino Fio

Tato deska je přizpůsobená k připojení různých bezdrátových modulů (XBee moduly). Srdcem je procesor ATmega328P, který běží na frekvenci 8MHz. Napětí je zde kvůli kompatibilitě s moduly sníženo oproti většině ostatních desek z 5V na 3,3V.

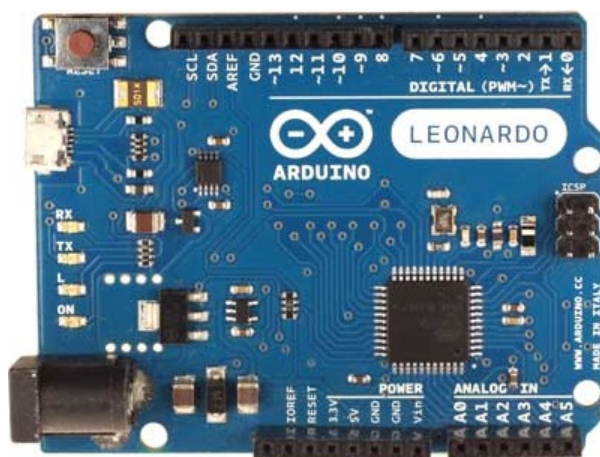
2.6 Arduino Uno



Obrázek 2.6: Arduino Uno

Arduino Uno je v současné době asi nejčastěji používaný typ desky. Je přímým pokračovatelem hlavní vývojové linie, která započala prvním Arduinem se sériovým portem místo USB, pokračující přes Arduino Extreme, NG, Diecimila a Duemilanove až k dnešnímu Uno. Na desce najdeme procesor ATmega328 a již klasické USB. Z této hlavní linie se vyvinuly i další dvě speciální desky. První z nich je Arduino Ethernet, které má stejnou výbavu jako Uno. Místo USB portu zde ale najdeme Ethernet port pro připojení k síti. Příjemná je přítomnost slotu pro microSD karty. Druhou deskou je Arduino Bluetooth. Jak už název napovídá, místo USB zde najdeme bluetooth modul pro bezdrátovou komunikaci. Velmi odlehčenou verzí Arduina Uno je Arduino Pro. To postrádá USB port a je tedy nutné ho programovat externím převodníkem. Je určeno spíše k pevnému zabudování do nějakého projektu.

2.7 Arduino Leonardo



Obrázek 2.7: Arduino Leonardo

Arduino Leonardo designově navazuje na Arduino Uno. Liší se však použitým čipem. Tím je ATmega32u4, který byl popsán již u Arduino Micro. Specifika tohoto čipu si popíšeme dále.

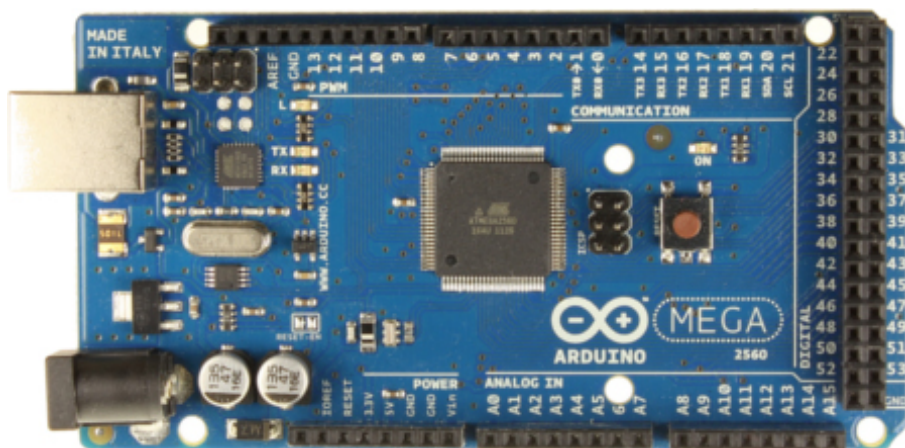
2.8 Arduino Yún



Obrázek 2.8: Arduino Yún

Model Arduino Yún sice také designově navazuje na Arduino Uno, jedná se však o naprostého průkopníka. Mimo již zmíněného čipu ATmega32u4, na kterém běží jádro Arduina, zde totiž najdeme i čip Atheros AR9331, který je schopný běhu odlehčeného linuxu Linino. Ve výbavě je softwarový bridge (prostředník, most), který zajišťuje komunikaci mezi oběma čipy. V kompaktním obalu tedy získáme v porovnání s velikostí velmi výkonný stroj. Na desce najdeme mimo microUSB pro programování ATmega32u4 i normální USB pro potřeby linuxu a Ethernet port pro připojení k síti. Můžeme tedy například posílat naměřené hodnoty přímo na webový server.

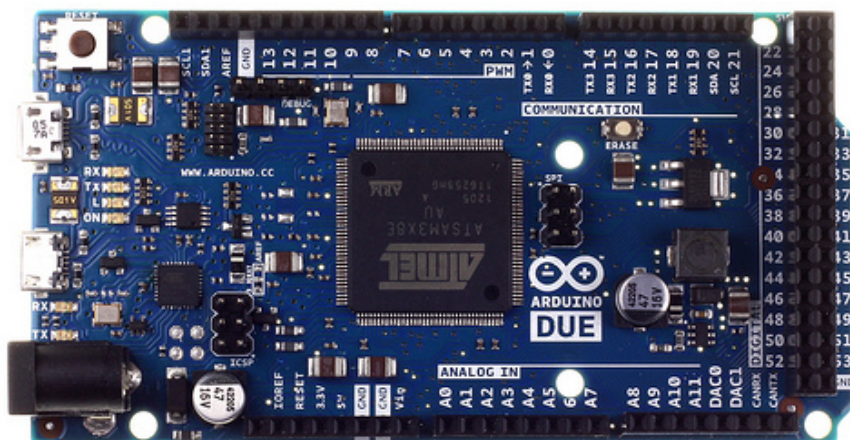
2.9 Arduino Mega2560



Obrázek 2.9: Arduino Mega2560

S Arduino Mega2560 se dostáváme do skupiny desek, jejichž vzhled vznikl prodloužením designu Arduina Uno. Zvětšení rozměrů přináší prostor pro větší a výkonnější čipy a také více pinů (zdiřek). Předchozí verzí bylo Arduino Mega1280. Hodí se tam, kde je zapotřebí většího výpočetního výkonu. Zajímavou odnoží této desky je Arduino Mega ADK vybavené jedním USB navíc pro připojení zařízení s Androidem.

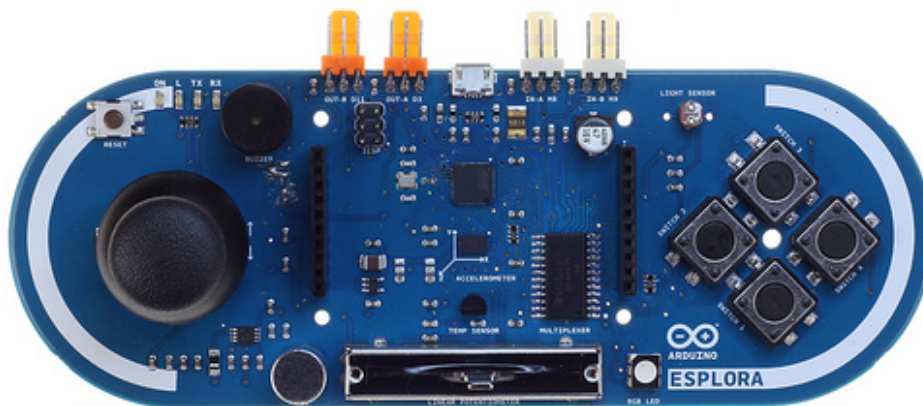
2.10 Arduino Due



Obrázek 2.10: Arduino Due

Arduino Due je pokračovatelem Arduina Mega, avšak s tím rozdílem, že běží na daleko výkonnějším čipu. Je jím Atmel SAM3X8E, který tiká na taktovací frekvenci 84Mhz a jeho jádro je 32-bitové, což je oproti ostatním deskám s 8-bity a maximálně 16MHz opravdu velký skok. Na desce nalezneme dva microUSB konektory. Jeden pro programování čipu, druhý pro připojení zařízení, jako jsou myši, klávesnice, telefony a jiné.

2.11 Arduino Esplora



Obrázek 2.11: Arduino Esplora

Arduino Esplora je první z desek, které by se daly zařadit do kategorie „hybridní“. Na první pohled je viditelný joystick, tlačítka a posuvný potenciometr. Nalezneme zde ale také piezzo bzučák, teploměr, tříosý akcelerometr, nebo piny pro připojení LCD displeje. Jedná se totiž o typ Arduina, se kterým se dá vytvořit samostatný herní set, nebo vlastní konzole pro ovládání her. Jednoduchou komunikaci s PC zajišťuje procesor ATmega32u4.

2.12 Arduino Robot



Obrázek 2.12: Arduino Robot

Jak už název napovídá, jedná se o set pro vytvoření vlastního chytrého robota. Jeho mozkiem je procesor ATmega32u4. Zajímavostí je přítomnost kompasu.

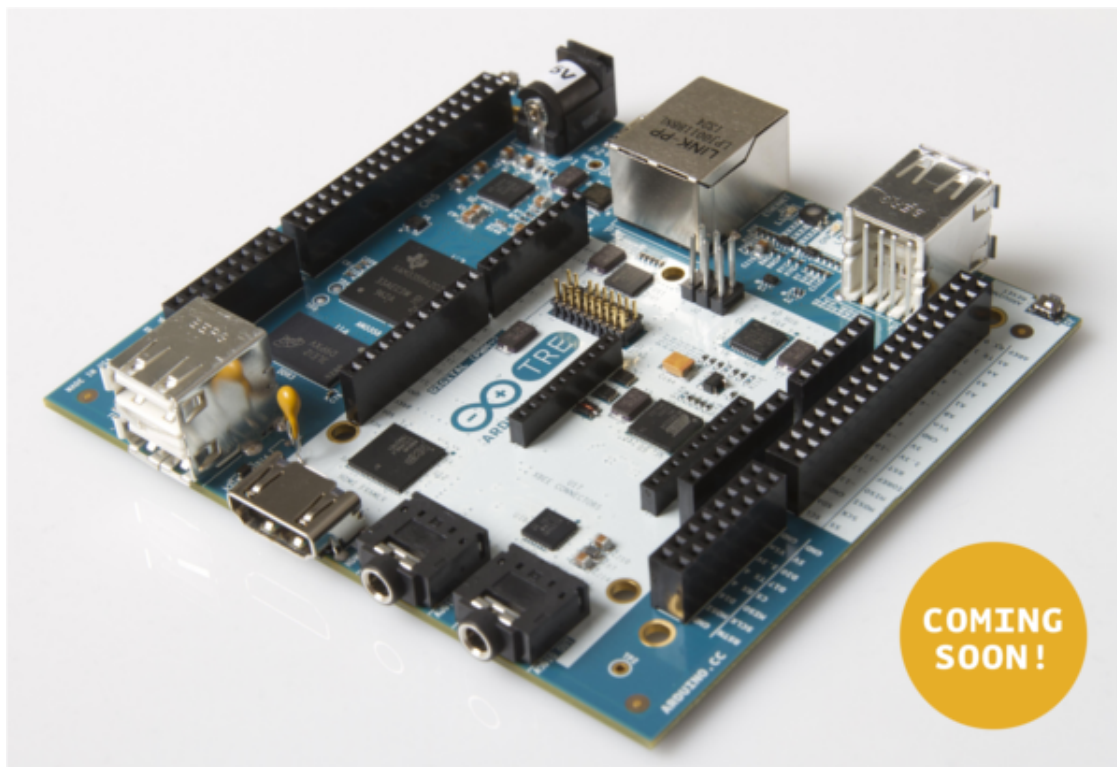
2.13 Arduino Intel Galileo



Obrázek 2.13: Arduino Intel Galileo

Tato verze vznikla ve spolupráci se společností Intel. Jedná se o první desku, která běží na čipu Intel® Quark SoC X1000, což je 32-bitový procesor s frekvencí 400 MHz. Najdeme zde dvě USB, microSD slot i Ethernet port. Užitečná může být také přítomnost mini-PCI Express slotu, pro připojení různých přídatných karet.

2.14 Arduino Tre



Obrázek 2.14: Arduino Tre

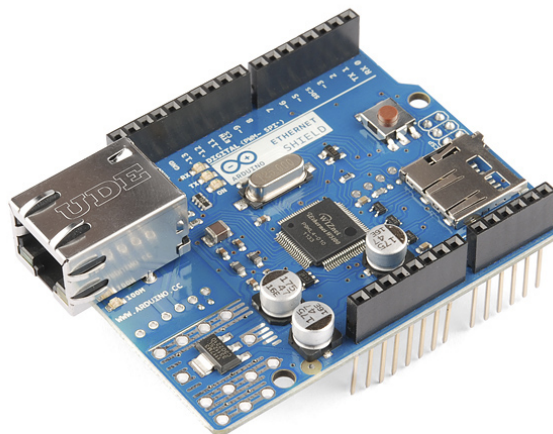
V současné době chystané Arduino Tre bude zatím nejvýkonnější typ. Mělo by obsahovat 1 GHz procesor, schopný běhu velmi náročných výpočetních aplikací. Stejně jako Arduino Yún bude obsahovat dva procesory. Jeden pro jádro Arduina a druhý pro linux. Na desce také nalezneme HDMI port, dva audio konektory, jeden USB port pro programování a 4 USB porty pro připojení dalších zařízení k linuxu. Už z hardwarové výbavy je patrné, že bude moci Arduino Tre konkurovat i jiným menším počítačům jako je například Raspberry Pi. Původně výrobce sliboval dostupnost na jaře 2014. V říjnu 2014 ale stále není jasné, kdy se skutečně začne prodávat.

Kapitola 3

Arduino Shieldy

Když se chceme na běžném stolním počítači připojit k WiFi, většinou nemáme jinou možnost, než si dokoupit WiFi kartu. Když chceme poslouchat, nebo nahrávat dobrou hudbu, musíme připojit kvalitní zvukovou kartu. A stejné to je u Arduina. Když něco nezvládne, nemusí být ještě všemu konec. Stačí si vybrat z rozsáhlé nabídky tzv. shieldů a vybraný shield poté nasunout do zdířek na Arduinu. Stejně jako desek existuje i celá řada shieldů. Z těch oficiálních jsou to ale Ethernet Shield, Wifi Shield, Motor Shield a Další. Při výběru je však nutné dát si pozor na to, aby byl vybraný shield s Arduinem kompatibilní.

Na obrázku vidíte, jak vypadá takový Ethernet shield.



Obrázek 3.1: Arduino Ethernet Shield

Kapitola 4

Arduino klony

Jak už jsem naznačil dříve, společně s oficiální řadou existuje ještě spousta dalších, neoficiálních desek. Jedná se o takzvané klony. Poznáme je podle toho, že mají často v názvu -duino (název Arduino je chráněný autorskými právy, -duino a podobné části jsou v názvu přípustné). Jelikož jsou všechna schémata, součástky i software dostupné online zdarma, může si prakticky každý sestavit své Arduino takřka „na koleni“. Můžeme se tedy setkat s klony tvarově a výbavou totožnými s oficiálními modely. Není to však pravidlem. Často jsou k vidění i desky, které jsou uzpůsobené ke konkrétní činnosti. Příklady klonů jsou:

- ArduPilot – navržený pro ovládání autonomních létajících zařízení (letadla, kvadrokoptéry...)
- Freaduino, Seeeduino – o něco levnější kopie originálních desek
- Rainbowduino – připravené k nasazení a řízení maticového RGB LED displeje, je možné je sestavovat do větších celků
- A další...

Část II

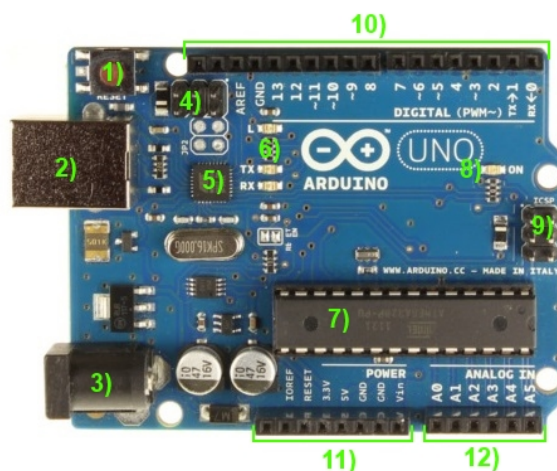
Programujeme Arduino

V první kapitole jsme se mohli dozvědět něco o historii Arduina. Také jsme si ukázali desky z oficiální řady a zmínili jsme i existenci shieldů. V závěru článku přišla řeč na neoficiální klony. Dnes už se tedy můžeme pustit do praktického použití. Na začátku si vybereme vhodné Arduino a na něm si ukážeme, co všechno se dá na desce najít. Řekneme si něco o programovacím jazyku a o vývojovém prostředí pro Arduino. Na závěr si vyzkoušíme první program, který bude blikat LED diodou.

Kapitola 5

Výběr a seznámení

Před tím, než začneme s Arduinem pracovat si musíme nějaký vhodný model vybrat. Pokud jsou cílem co nejmenší výrobky, bude nejlepší sáhnout po nějaké z menších desek (Mini, Nano, Micro). Pokud však hledáte velmi výkonné modely, které zastanou méně výkonné počítače. Těmito modely jsou například DUE, či Intel Galileo. Zlatou střední cestou jsou modely Uno a Mega 2560. Pro jednoduché zkoušení zapojení a nenáročných modulů je Uno plně dostačující. Na něj je také designována většina shieldů. Arduino Mega 2560 na druhou stranu nabídne daleko více vstupních a výstupních pinů. Vyberme si tedy pro účel následujícího popisu model Arduino Uno. Níže uvedené části najdeme v různých obměnách na většině desek.



Obrázek 5.1: Co nalezneme na desce Arduino

1. Pod číslem jedna se skrývá resetovací tlačítko. To použijeme, pokud chceme náš program spustit znovu od začátku. U různých typů Arduina se můžeme setkat i s jiným umístěním tohoto tlačítka. Většinou je ale toto tlačítko popsané nápisem RESET.
2. USB konektor typu B. U nejstaršího modelu najdeme místo USB sériový port. U některých novějších modelů se setkáme s micro USB. Na některých deskách ho vůbec nenajdeme, protože mají buď jiný způsob připojení (Ethernet, BT), nebo pro programování vyžadují připojení externího programátoru.
3. Napájecí konektor. Využijeme ho, pokud nebudeme Arduino napájet z USB.
4. ICSP hlavice pro externí programování USB-serial převodníku. Běžný uživatel ji nepoužije. U verzí bez převodníku, nebo s převodníkem obsaženým v hlavním čipu ji nenajdeme.
5. USB-serial převodník. Ten se stará o komunikaci mezi hlavním čipem a PC. Plní zde roli překladatele. U verzí bez převodníku, nebo s převodníkem obsaženým v hlavním čipu ho nenajdeme.
6. Indikační LED diody L, Rx a Tx. Dioda s popisem L je často využívána. Je totiž připojená k výstupu č. 13. S ní se tedy dá vyzkoušet blikání i bez připojené externí LEDky. Některá Arduina ji však vůbec neobsahují. Diody s popisem Tx a Rx blikají, pokud probíhá komunikace přes sériovou linku.
7. Hlavní čip celé desky. V různých podobách a typech ho najdeme na všech deskách.
8. Indikační LED dioda ON. Svítí, když je připojené napájení.
9. ICSP hlavice pro externí programování hlavního čipu. Využívají ji některé shieldy.
10. Digitální piny. Do těchto zdírek budeme připojovat všemožné obvody. Vývody označené vlnovkou podporují PWM modulaci, o které si povíme později.
11. Převážně napájecí výstupy Arduina.
12. Analogové vstupy. Sem připojíme vodiče, na kterých budeme chtít měřit nějakou analogovou hodnotu. Dají se využít i jako digitální vstupy a výstupy.

Kapitola 6

Arduino IDE

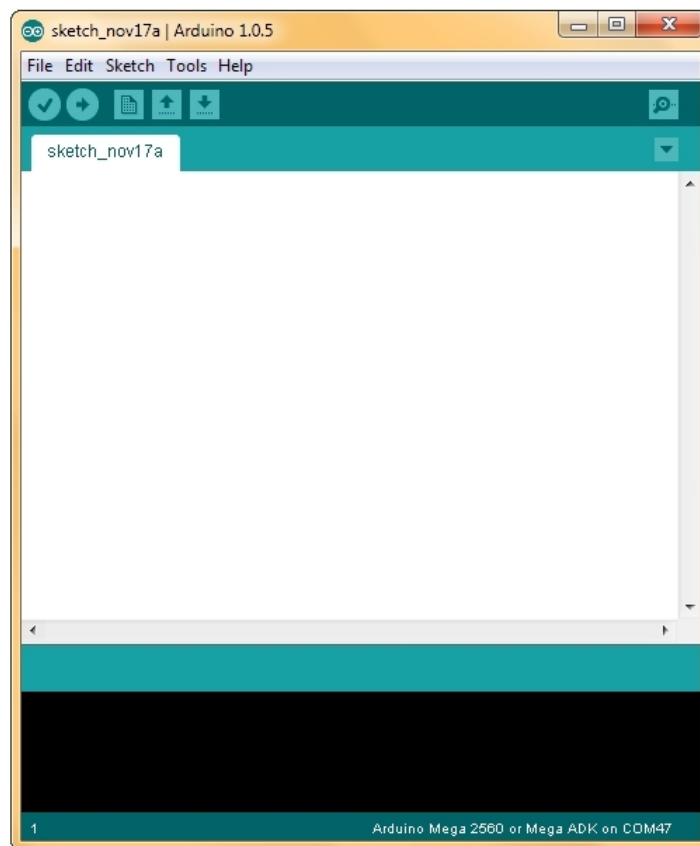
6.1 Historie

Arduino IDE (integrated development enviroment = integrované vývojové prostředí) je napsané v jazyce Java. Jedná se o software vzniklý z výukového prostředí Processing. To bylo mírně upraveno, byly přidány určité funkce a v neposlední řadě podpora jazyka Wiring.

6.2 Stažení a instalace

Arduino IDE si můžeme stáhnout zde [en]. V této době je poslední verzí Arduino 1.0.5 (1.5 je stále ve verzi Beta). Najdeme si tedy na poslední finální verzi a stáhneme ji pro požadovaný operační systém. Pro Windows je nejjednodušší stáhnout si ZIP archiv, který je po rozbalení plně funkční. Pro Linux se instalace může lišit i podle distribucí. Popis pro jednotlivé distribuce nalezneme zde[en]. Musíme také zmínit další velký operační systém, jímž je Mac OS. Návod na instalaci nalezneme zde[en]. Následující část článku se bude věnovat použití Arduina s operačním systémem Windows. Předpokládáme, že máme stažený ZIP archiv verze 1.0.6. Vybereme si složku, ve které chceme mít software pro Arduino a zde archiv rozbalíme. Po rozbalení obsahuje další složky a soubory. Složka Drivers obsahuje ovladače pro komunikaci Arduina s PC. V Examples nalezneme příklady kódů. Důležitou složkou je složka Libraries, kam se ukládají knihovny. To jsou balíky obsahující rozšiřující funkce pro programování. Ještě než si IDE spustíme si vytvoříme v uživatelské složce Dokumenty složku Arduino. Tuto složku většinou nalezneme na umístění: C:\Users\jmeno_uzivatele\Documents. Zde si vytvoříme složku Arduino a v ní složku libraries. Sem si budeme ukládat vytvořené programy a do složky libraries přidáme knihovny. Ty nám zde zůstanou stále stejné, i když přejdeme na novější verzi Arduino IDE. Nyní už nás bude zajímat pouze soubor arduino.exe ve staženém balíku, který spustí vývojové prostředí. Spustíme si ho tedy a ukažme si jeho nejdůležitější funkce.

6.3 Používání



Obrázek 6.1: Arduino IDE

Vývojové prostředí můžete vidět na obrázku. V prvním řádku navigačních prvků nás bude zajímat pouze rozbalovací nabídka Tools, ve které nalezneme nastavení pro připojení a programování desky. Její funkce si popíšeme později. V dalším řádku nalezneme několik ikon. Jako první zleva nalezneme ikonu s fajfkou – Verify. Ta po kliknutí spustí kontrolu programu a zkontroluje kód. Pokud nalezne nějakou chybu, v syntaxi ji zvýrazní. Vedle nalezneme ikonu s šipkou doprava – Upload. Ta spustí kontrolu programu, a pokud nenalezne žádné chyby, nahraje program do připojeného Arduina. Další je ikona se symbolem přeložené stránky – New, která po kliknutí vytvoří nový soubor. Další tlačítko s šipkou nahoru – Open – otevře nabídku pro otevření programů (včetně těch, které máme uložené v Dokumentech). Tlačítko s šipkou dolů – Save – uloží současný program. Ve stejném řádku nalezneme úplně vpravo ještě ikonu s lupou – Serial Monitor. Ta spustí sériový monitor, o kterém si více povíme příště. Velký bílý prostor slouží k zápisu kódu a černý prostor dole zobrazuje informační a chybové výpisy z běhu prostředí.

6.4 Programovací jazyk

Arduino je možné programovat v jazyce C nebo C++. Nejjednodušší je však používat knihovnu Wiring. Ta je v současné době pro programování Arduino velmi rozšířená. Kvůli její komplexnosti se o ní občas mluví jako o samostatném programovacím jazyku. Pro první seznámení si otevřeme příklad BareMinimum. Klikneme na ikonu Open a v rozbalovacím seznamu 01.Basics vybereme možnost BareMinimum. V editoru se nám zobrazí následující kód.

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8  
9 }
```

Na ukázkovém kódu si můžeme všimnout hned dvou věcí. První z nich je přítomnost dvou bloků programu. V horní části nalezneme blok (funkci) void setup() . Mezi složené závorky se v tomto bloku píše kód, který se provede pouze jednou na začátku programu. To znamená buď po připojení napájení, zmáčknutí tlačítka restart, nebo nahrání kódu do Arduino. Druhým blokem (funkcí) je void loop() , do jehož složených závorek se zapisuje kód, který se bude opakovat neustále dokola až do odpojení napájení. Tyto dvě části musí být v programu VŽDY – tedy i když neobsahují žádné příkazy. Při jejich absenci by program skončil chybou. Dále bychom si měli všimnout dvojitého lomítka. To nám značí komentáře v programu. Část kódu, nebo textu zapsanou za podtržítkem bude program ignorovat. Používá se, když si chceme k části kódu zapsat poznámku, nebo když chceme na chvíli vyřadit část kódu z provozu. Můžeme se setkat s dvěma typy komentářů:

```
1 jednoradkový komentář  
2 //když řádek začíná dvojitým podtržítkem, jedná se o jednoradkový komentář  
3 //vše, co je v řádku za ním program ignoruje  
4 to co se však nachází na dalším řádku je bráno jako normální kód  
5  
6 druhým typem jsou včítákové komentáře  
7 /* ty začínají podtržítkem a hvězdičkou  
8 mohou  
9 mít  
10 libovolný  
11 počet  
12 řádků  
13 a končí hvězdičkou a podtržítkem */
```

Kapitola 7

Blikáme LED

7.1 Budeme potřebovat

1. Arduino Uno(nebo jinou desku)
2. USB kabel
3. Nepájivé kontaktní pole
4. LED Dioda – stačí obyčejná za 1 Kč
5. Vodiče – dobrou kombinací obsahující nepájivé pole i vodiče je tento set.
6. 330 ohm rezistor – zapojení sice bude fungovat i bez něj, ale je nutné ho použít.

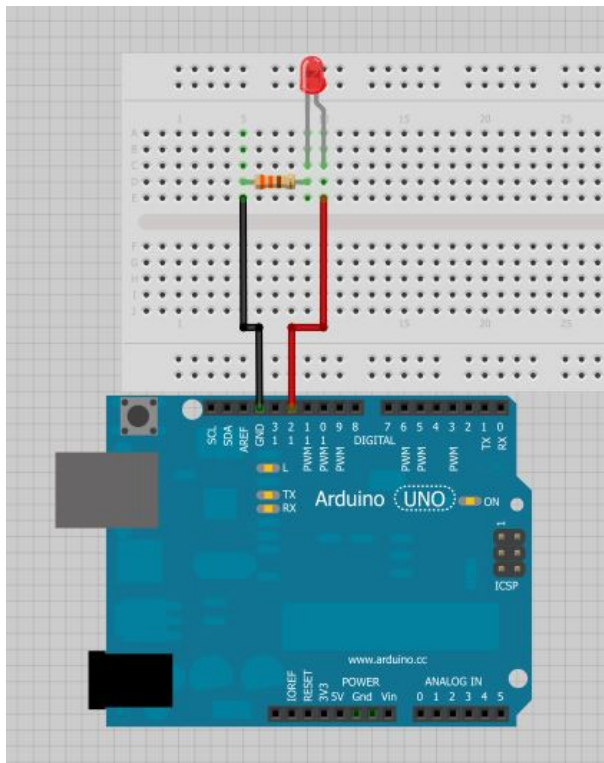
7.2 Připojení Arduina a PC

Arduino Uno připojíme pomocí USB k počítači. Mělo by se nám objevit instalační okno. Pokud se instalace nového zařízení nepovede, neděste se. Nyní přichází na řadu stažená složka Drivers. V nabídce Start -> Ovládací panely otevřeme Správce zařízení. Zde najdeme naše Arduino. Pravým tlačítkem otevřeme okno Vlastnosti a na kartě Ovladač zmáčkne tlačítko Aktualizovat ovladač (k tomu je však nutné mít oprávnění správce). Poté vybereme možnost Vyhledat ovladač v počítači a navedeme instalační program do umístění složky Drivers. Nakonec zmáčknutím tlačítka další nainstalujeme ovladač.

Nyní ještě musíme nastavit Arduino IDE. Otevřeme nabídku Tools a v seznamu Boards vybereme Arduino Uno. Poté v Tools -> Serial Port vyberte sériový port, na který je Arduino připojeno (většinou je to jediný port v seznamu). Tímto je za námi nastavování a můžeme se pustit do zapojování.

7.3 Zapojení

Vezmeme si LED diodu, vodiče a rezistor a vše zapojíme tak, jak je vidět na obrázku.



Obrázek 7.1: Schéma zapojení příkladu Blink

7.4 Program

Nyní už nám nezbývá nic jiného, než do editoru vložit následující program a stiskem tlačítka Upload nahrát kód do Arduina. Pokud vše proběhlo v pořádku, LED dioda začne blikat.

```
1 void setup() {  
2     pinMode(12, OUTPUT); //nastav pin 12 jako vystup  
3 }  
4  
5 void loop() {  
6     digitalWrite(12, HIGH); //na pinu 12 pust proud  
7     delay(1000); //pockej 1000 ms = 1 s  
8     digitalWrite(12, LOW); //na pinu 12 vypni proud  
9     delay(1000);  
10 }
```

Část III

Základní struktury jazyka Wiring

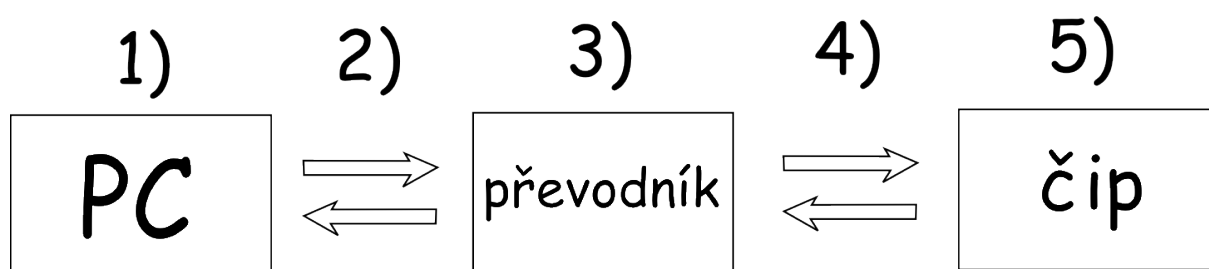
V předchozí části jsme si ukázali první program, ve kterém Arduino blikalo LED diodou. Úvodní seznámení je tedy za námi a můžeme se pustit do dalšího programování. V dnešním článku se podíváme na základní náležitosti jazyka Wiring. Na začátek si vysvětlíme, jak Arduino komunikuje s počítačem. Poté si řekneme, jak používat proměnné a jak pracovat se vstupy a výstupy Arduina.

Kapitola 8

Sériová komunikace

Aby mohlo Arduino správně komunikovat s PC, musí mít několik základních součástí. Následující popis postihuje většinu desek Arduino. Najdou se však i speciální desky, které podporují jiný způsob programování (BT, Ethernet, Wifi. . .), ale těmi se dále zabývat nebudeme. Popišme si tedy proces programování, se kterým se setkáme u většiny desek.

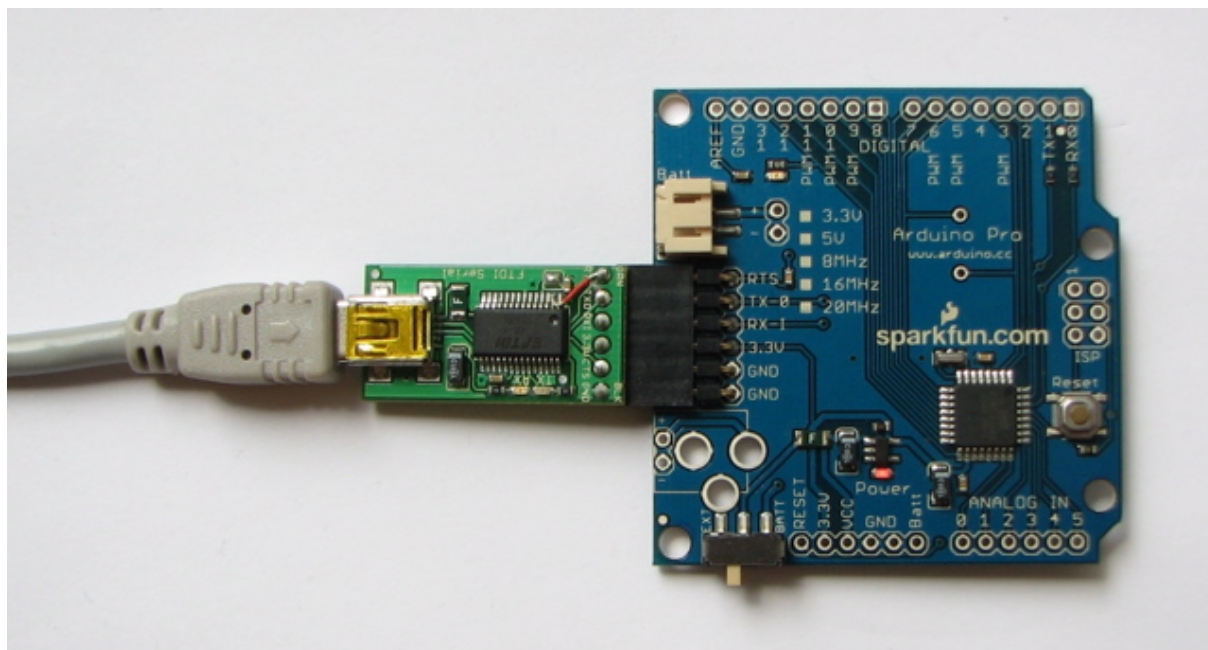
Co je ke komunikaci potřeba:



Obrázek 8.1: Schéma sériové komunikace

1. Základním předpokladem je mít PC s USB portem.
2. USB kabel
3. S převodníkem se nám schéma trochu komplikuje. Můžeme se totiž setkat se třemi základními typy převodníku. Všechny tři převodníky fungují stejně. Liší se pouze způsobem připojení.
 - (a) Převodník, který je na pevně připojený k základní desce Arduina.
 - (b) Převodník, který mají některé čipy (ATmega32u4. . .) přímo v sobě.
 - (c) Externí převodník, který musíme při programování Arduina připojit.
4. Připojení převodníku a čipu. Při použití externího převodníku se většinou jedná o šest vodičů, které vystupují z desky. U zbylých dvou typů převodníku jsou to pouze kontakty na plošném spoji, nebo propojení uvnitř čipu.
5. Mozek, který přijímá přeložené instrukce od převodníku.

Takto vypadá Arduino Pro s připojeným externím převodníkem.



Obrázek 8.2: Arduino Pro s externím převodníkem

Sériová komunikace se ale dá využít k více věcem, než jen k programování. Pomocí ní totiž můžeme komunikovat s Arduinem, i když už na něm běží náš program. Poté můžeme například číst hodnoty ze senzorů a posílat je do PC, nebo ovládat Arduino jednoduchými textovými příkazy. Používání těchto funkcí si popíšeme za chvíli, kdy už budeme mít dostatek informací k jejich pochopení.

Kapitola 9

Proměnné

Proměnná je místo ve kterém se dají uchovávat data. Každá proměnná má vlastní jméno, datový typ a hodnotu. Často se používají například tam, kde se v programu dokola opakují ty samé hodnoty. Praktické využití proměnných si ukážeme na následujícím příkladu napsaném v pseudokódu:

Představme si, že máme několik očíslovaných světel. Vždy si vybereme jedno, se kterým budeme blikat.

```
1  zapniSvetlo(10); //zapni desate svetlo
2  vypniSvetlo(10); //vypni desate svetlo
3  zapniSvetlo(10);
4  vypniSvetlo(10);
5  zapniSvetlo(10);
6  vypniSvetlo(10);
7  ...
```

Nyní si stejný program přepíšeme s užitím proměnných.

```
1  cislo A = 10;
2  //promenna se jmenuje A, je datoveho typu cislo a ma hodnotu 10
3
4  zapniSvetlo(A);
5  vypniSvetlo(A);
6  zapniSvetlo(A);
7  vypniSvetlo(A);
8  zapniSvetlo(A);
9  vypniSvetlo(A);
10 ...
```

Kdybychom chtěli změnit světlo, se kterým blikáme, museli bychom v prvním případě změnit všechny čísla 10 na jiná. V případě druhém nám stačí přepsat pouze hodnotu proměnné a program ji už sám dosadí na všechna potřebná místa.

9.1 Práce s proměnnými

Nyní už se podíváme na to, jak pracovat s proměnnými v jazyce Wiring. Jedním z číselných datových typů je typ integer (zkracuje se na int). Ukažme si tedy, jak vytvořit proměnnou, která v sobě uchová číselnou hodnotu. Abychom mohli v programu s proměnnou pracovat, musíme ji nejprve deklarovat („vytvořit“). Poté jí můžeme přiřadit hodnotu.

```
1 //deklarace promenne x
2 int x;
3 //přirazení hodnoty
4 x = 10;
5
6 //tyto dve operace se daji spojit do jedne
7 int y = 10;
```

Vytvořit proměnnou ale nemůžeme jen tak někde. Když budeme chtít používat danou proměnnou všude v programu, musíme ji vytvořit vně všech funkcí (tedy i mimo funkce setup a loop). Pokud nám stačí používat proměnnou uvnitř jedné funkce a nikde jinde ji nepotřebujeme, stačí, když ji deklarujeme uvnitř funkce.

```
1 int x = 10; //tuto promennou muzeme pouzit vsude
2
3 void setup() {
4     int y = 11;
5     //wnitr teto funkce muzeme pouzit promenne x a y
6 }
7
8 void loop() {
9     int z = 12;
10    //zde muzeme pouzit promenne x a z
11 }
```

Pokud bychom se pokusili do Arduina nahrát kód, ve kterém používáme proměnnou y ve funkci loop (nebo z ve funkci setup), překlad kódu skončí s chybovou hláškou a do Arduina se nic nenahraje.

9.2 Datové typy

Jak už jsem naznačil dříve, každá proměnná má svůj datový typ. Ten nám říká, jaká data můžeme v proměnné najít. Může se jednat o logické hodnoty (true/false), znaky, nebo čísla. Pojdme si nyní představit základní typy.

Číselné datové typy

- **byte** – Proměnná datového typu byte má velikost 8 bitů a slouží k uchování celých čísel. Její rozsah je 28 hodnot – 0 až 255
- **integer** – V programech se používá jen zkratka int. Slouží k ukládání celých čísel. Rozsah tohoto datového typu se liší podle použitého procesoru a zasahuje jak do kladných, tak i do záporných čísel. Nula leží přibližně v polovině rozsahu. U desek s procesory Atmega (tedy naprostá většina) uchovává 16-bit hodnotu – tedy od -32,768 do 32,767. U Arduino DUE s procesorem SAM je tato hodnota 32-bitová a může obsahovat čísla od -2,147,483,648 do 2,147,483,647.
- **long** – Slouží k uchování celočíselných 32 bitových hodnot od -2,147,483,648 do 2,147,483,647.
- **float** – Tento datový typ je určený pro uchování čísel s desetinnou čárkou. V jazyce Wiring se však používá desetinná tečka. Jeho velikost je 32 bitů. Můžeme v něm ukládat hodnoty od -3,4028235 * 1038 do 3,4028235 * 1038.

Logický datový typ

- **boolean** – Proměnné tohoto datového typu v sobě uchovávají pouze dvě hodnoty. Buďto true (pravda), nebo false (nepravda).

Znakový datový typ

- **char** – Tento datový typ slouží k uchování jednoho znaku textu. Znak je zde uchován jako jeho číselná hodnota v ASCII tabulce znaků. Písmena, slova i věty se píší v uvozovkách. K uchování řetězců textu slouží typ string, kterým se budeme zabývat za chvíli.

```
1 //byte
2 byte a = 12;
3 //integer
4 int b = 400;
5 //long
6 long c = 12121212;
7 //float
8 float d = 1.256;
9
10 //boolean
11 boolean e = false;
12
13 //char
14 char f = 'A';
15 char f = 65; //v ASCII tabulce znaku má A hodnotu 65
```

Existují i další číselné datové typy, ale ty se nepoužívají moc často. Jejich popis nalezneme v Arduino Reference [EN] v prostředním sloupci v části Data Types.

Kapitola 10

Pole

Pole (anglicky array) je speciální typ proměnné. Umožňuje shromáždit více hodnot do jedné proměnné. Můžete si jej představit, jako krabičku, která má jednotlivé přihrádky očíslované. Když víme, jakou krabičku používáme a do jaké přihrádky se chceme podívat, můžeme se dostat k požadované hodnotě. V programátorské terminologii se číslům přihrádek říká index.

10.1 Deklarace pole

Jejich deklarace je podobná, jako u proměnných – každé pole má datový typ hodnot, které v něm najdeme, jméno, hodnoty a navíc i velikost pole.

```
1 //pole muzeme deklarovat nekolika zpusoby
2 int jmeno[6]; //deklarace pole s sestí bunkami
3 int jmeno[] = {2, 3, 4, 5}; //prvky v poli oddelujeme carkami
4 int jmeno[4] = {2, 3, 4, 5}; //v tomto pripade velikost pole uvest muzeme, nemusime
5
6 //zvlastnim typem pole je pole znaku (nazyvane retezec – string)
7 //umoznuje totiz specificky zpusob prirazeni hodnoty
8 char jmeno[15]; //deklarace retezce
9 char jmeno[] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
10 char jmeno[7] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
11 char jmeno[] = "arduino";
12 char jmeno[7] = "arduino";
```

10.2 Přístup k hodnotám v poli

Ve většině programovacích jazyků jsou indexy v poli číslovány od nuly. Prvek na prvním místě má tedy index 0. Čtení hodnoty prvku pole poté probíhá stejně, jako u proměnných, jen musíme připojit ještě jeho index.

```
1 int a[] = {1,2,3,5,7,11,13,17}; //deklarace pole a
2
3 a[0]; //prvek s indexem 0 ma hodnotu 1
4 a[5]; //prvek s indexem 5 ma hodnotu 11
5 ...
```

Kapitola 11

Digitální vstup a výstup

Jelikož je Arduino určeno k dalšímu rozšiřování, obsahuje vstupy a výstupy (nazývané piny), ke kterým se dá vodičem připojit další obvody, čipy, relé, paměti atd.. My už jsme se s takovýmto případem setkali v minulém článku, když jsme blikali LED diodou. K práci s těmito piny má Arduino k dispozici jednoduché funkce. Nejdříve ze všeho je však potřeba programu říci, jestli s pinem pracujeme jako se vstupem, nebo výstupem.

11.1 Vstup nebo výstup?

K nastavení pinu slouží funkce `pinMode()`. Ta pro svoji správnou činnost potřebuje dva vstupní parametry – `pinMode(cislo_pinu, INPUT/OUTPUT)`; Pokud chceme daný pin používat jako vstup, bude druhý parametr `INPUT`, pokud jako výstup, bude to `OUTPUT`. Číslo pinu je většinou natištěno na desce Arduina. Ve verzi Arduino UNO tedy můžeme používat piny 0 až 13. Tímto ale nekončí, protože můžeme k digitálním operacím používat i piny označené jako `ANALOG IN`, jenom místo samotného čísla musíme před číslo napsat `A`. U Arduina UNO jsou to tedy `A0` až `A5`.

```
1 byte cislo = 13;
2
3 pinMode(cislo, OUTPUT); //nastavení pinu 13 na vystup
4 pinMode(12, INPUT); //a pinu 12 na vstup
```

11.2 Ovládání výstupu

K ovládání výstupu se používá funkce `digitalWrite()`. S touto funkcí jsme se setkali již v minulém článku, když jsme blikali LED diodou. Stejně jako `pinMode()` potřebuje i tato funkce dva parametry – číslo pinu a informaci o proudu. Pokud proud teče, je to `HIGH`, pokud ne, tak `LOW`.

```
1 digitalWrite(13, HIGH);
2 digitalWrite(12, LOW);
```

11.3 Čtení vstupu

Ke zjištění, zda proud do vstupu teče, nebo ne se používá funkce `digitalRead()`. Ta, na rozdíl od předchozích dvou funkcí, potřebuje pouze jeden parametr, kterým je číslo pinu. Tato funkce navíc vrací hodnotu. Když proud teče, vrátí hodnotu `HIGH`, když ne, tak `LOW`.

```
1 int cteni;  
2 byte vstup = 13;  
3  
4 cteni = digitalRead(vstup); //pokud proud teče, do promenne cteni se ulozi hodnota HIGH  
5 //pokud ne, tak LOW
```

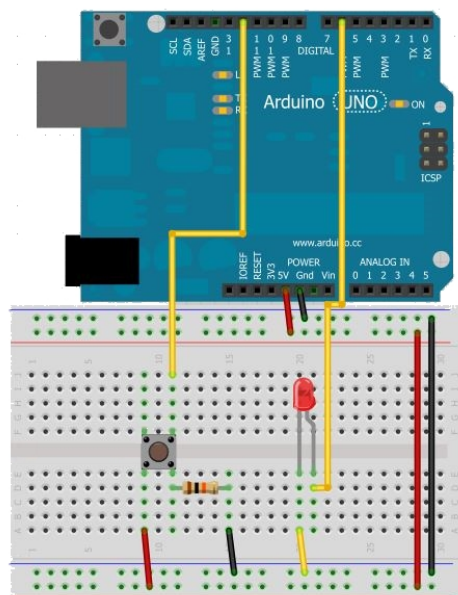
Kapitola 12

Příklad: Tlačítko a LED dioda

Ukážeme si program, který bude zjišťovat, jestli je stisknuté tlačítko. Pokud bude, rozsvítí se LED dioda. K této ukázce budeme potřebovat:

1. Desku Arduino
2. PC
3. Nepájivé kontaktní pole s vodiči
4. Tlačítko
5. 10K ohm resistor
6. LED diodu

Vše zapojíme podle schématu. Poté nahrajeme do Arduino uvedený program.



Obrázek 12.1: Zapojení tlačítka a LED diody

```

1  int cteni;
2  int led = 6;
3  int tlacitko = 12;
4
5  void setup() {
6      pinMode(led, OUTPUT);
7      pinMode(tlacitko, INPUT);
8  }
9
10 void loop() {
11     cteni = digitalRead(tlacitko);
12     digitalWrite(led, cteni);
13 }

```

Na závěr je nutno dodat, že pokud program nefunguje, nejčastější chybou je chybějící středník na konci řádku. Pokud tedy IDE napíše nějakou chybovou hlášku, zkontrolujte středníky. Ty se píší na konec řádku za: deklarací proměnné, za přiřazením hodnoty proměnné a za voláním funkce (pinMode...).

Část IV

Pokročilejší struktury jazyka Wiring

V tomto díle navážeme na informace z minula a ukážeme si další možnosti jazyka Wiring. Na začátku si řekneme, co jsou to konstanty a jak je používat. Poté si ukážeme, jak pracovat s analogovým vstupem a výstupem, pomocí něhož se dají získávat data z různých analogových senzorů. Nakonec se dostaneme k velmi důležité součásti jakéhokoliv programovacího jazyka, kterou jsou podmínky.

Kapitola 13

Konstanty

Konstanty si můžeme představit jako proměnné, které mají přednastavenou hodnotu, definovanou tvůrci Arduina. Mají za úkol zpřehlednit práci a přiblížit kód lidskému jazyku. My jsme s některými z nich pracovali již v minulém dílu. Můžeme je rozdělit do tří skupin.

13.1 Logické konstanty

Jsou pouze dvě hodnoty, a to pravda a nepravda. V programování jim odpovídají konstanty `true` a `false`. Používají se tam, kde je třeba rozhodovat pouze mezi dvěma stavy.

- `false`
 - Konstanta `false` má hodnotu 0.
- `true`
 - U konstanty `true` je situace trochu komplikovanější. Mohlo by se zdát, že má hodnotu 1, ale není to úplně přesné. Při logických operacích totiž jazyk Wiring vnímá jakékoliv nenulové číslo typu integer jako `true`.

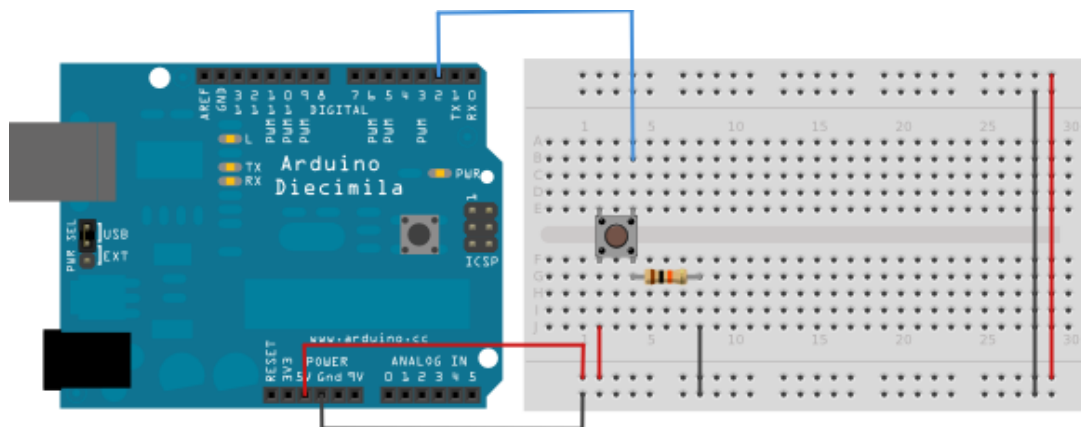
13.2 Typ digitálního pinu

V minulém díle jsme při určování, zda se bude pin chovat jako vstup, nebo jako výstup, používali ve funkci `pinMode()` dvě konstanty – `OUTPUT` a `INPUT`. Dnes si k nim přidáme ještě třetí možnost `INPUT_PULLUP`.

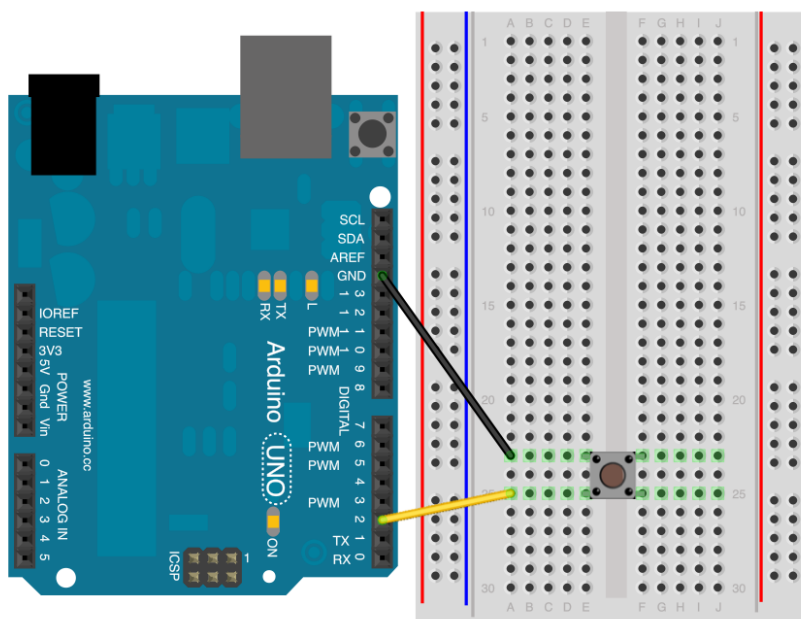
- `OUTPUT`
 - Při použití této konstanty je pin nastaven jako výstup. Ten snese proud do 40 mA. Při stavu `HIGH` tedy tento výstup může poskytnout proud do 40 mA a při stavu `LOW` může stejně velký proud přijmout.
- `INPUT`
 - Nastaví pin jako vstup. Ten se používá ke čtení hodnot z digitálních senzorů (nejjednodušší jsou tlačítka), ale i ke komunikaci. Použití s tlačítkem jsme si ukázali v minulém dílu. V jeho zapojení si všimněme, že je tento pin stále připojen ke GND přes 10k ohm resistor. Při nezmačknutém tlačítku je tedy výsledek funkce `digitalRead()` hodnota `LOW`. Po zmačknutí tlačítka dojde k připojení k +5V a změny hodnoty funkce na `HIGH`.

- INPUT_PULLUP

- Funguje podobně, jako INPUT. Rozdíl je v tom, že dojde k připojení interního rezistoru. Ten je uvnitř čipu zapojen mezi digitálním vstupem a +5V. Výchozí hodnota funkce digitalRead() je tedy HIGH. Když chceme hodnotu změnit, musíme vstup připojit na GND. Při použití příkladu s tlačítkem má tedy funkce hodnotu HIGH, když je tlačítko uvolněno a LOW, když je zmáčknuto.



Obrázek 13.1: Tlačítko - INPUT



Obrázek 13.2: Tlačítko – INPUT_PULLUP

13.3 Proud na digitálních pinech

Jsou pouze dvě možné hodnoty, jaké může mít proud při čtení a zápisu pomocí funkcí `digitalRead()` a `digitalWrite()`. Jsou to hodnoty `LOW` a `HIGH`.

- `HIGH`
 - Při čtení pomocí funkce `digitalRead()` je vyhodnocena hodnota napětí jako `HIGH`, pokud je větší než 3V. Když použijeme funkci `digitalWrite(pin, HIGH)`, na výstupu bude právě 5V.
- `LOW`
 - Při čtení je stav napětí vyhodnocen jako `LOW`, pokud je jeho velikost menší než 2V. Při zápisu je hodnota 0V. Fakticky ale může "přijmout" napětí do velikosti 5V (což nelze, pokud je nastaveno `HIGH`).

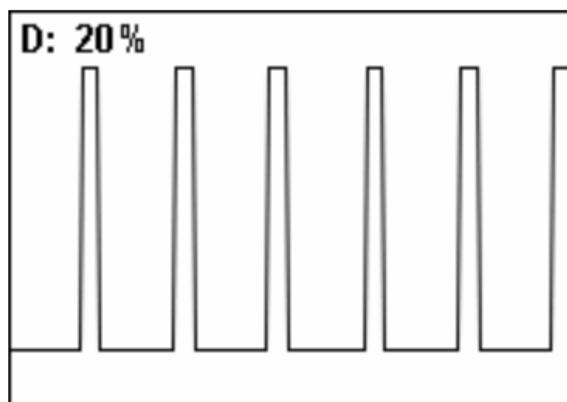
Kapitola 14

Analogový vstup a výstup

S digitálním vstupem a výstupem jsme se setkali už v předchozí části. Co když ale potřebujeme pracovat i s analogovými hodnotami? Na to má Arduino ve výbavě užitečné funkce. Ke čtení a zápisu se zde používají funkce `analogRead()` a `analogWrite()`. Ty jsou však limitovány pro použití pouze na určených pinech. Pojdme si je postupně představit.

14.1 `analogWrite()`

Jak už z názvu vyplývá, jedná se o funkci sloužící k nastavení "analogové" hodnoty na pinu. Můžeme ji použít pouze na pinech označených PWM (u Arduina UNO jsou to piny: 3, 5, 6, 9, 10, 11). Používá se u ní syntaxe `digitalWrite(číslo_pinu, hodnota)`, kdy hodnota může být v rozsahu 0 až 255. Slovo analogové jsem dal do uvozovek, protože se ve skutečnosti o žádné analogové hodnoty nejedná. Pokud bychom chtěli skutečně analogovou hodnotu v rozsahu například 0-5V, museli bychom použít externí D/A převodník. Tato funkce totiž na vybraných pinech generuje PWM signál, což je jakási digitální "náhražka" analogového signálu. Ta v praxi funguje tak, že rychle střídá 0 a 5V. To se projeví sníženou "účinností". LED svítí slaběji (ve skutečnosti rychle bliká a střídá pouze dva stavy napětí a snížená intenzita je způsobena setrvačností oka), motor se točí pomaleji atd. Podle poměru času, ve kterém je na výstupu +5V ku stavu 0V se pak odvíjí intenzita svícení LED diody a podobně. Při volání funkce `analogWrite(pin, 127)` je tedy přibližně 50% času nastaveno napětí +5V a 50% času 0V. Průběh napětí při různých hodnotách můžete vidět na obrázku.



Obrázek 14.1: Graf PWM modulace

14.2 analogRead()

Funkce `analogRead()` slouží ke čtení analogové hodnoty na vstupech k tomu určeným. Jsou to tedy piny označené písmenem A (například A2). Čtení analogových hodnot je užitečné u různých senzorů (teplota, vlhkost atd.). Většina desek Arduina má rozlišení 10 bitů, což odpovídá hodnotám od 0 do 1023. Například u Arduino DUE se ale můžeme setkat až s 12-bitovým rozlišením. Zde se dá nastavit požadované rozlišení pomocí funkce `analogReadResolution()`. My se ale budeme zabývat obyčejným Arduinem. Syntaxe je jednoduchá: `proměnná = analogRead(pin)`. Nejjednodušším příkladem použití je měření hodnot na potenciometru. Pokud bychom chtěli měřit například stav fotoresistoru, museli bychom ho zapojit do děliče napětí s vhodným resistorem. Použití obou analogových funkcí si ukážeme na zapojení s LED diodou a potenciometrem.

Kapitola 15

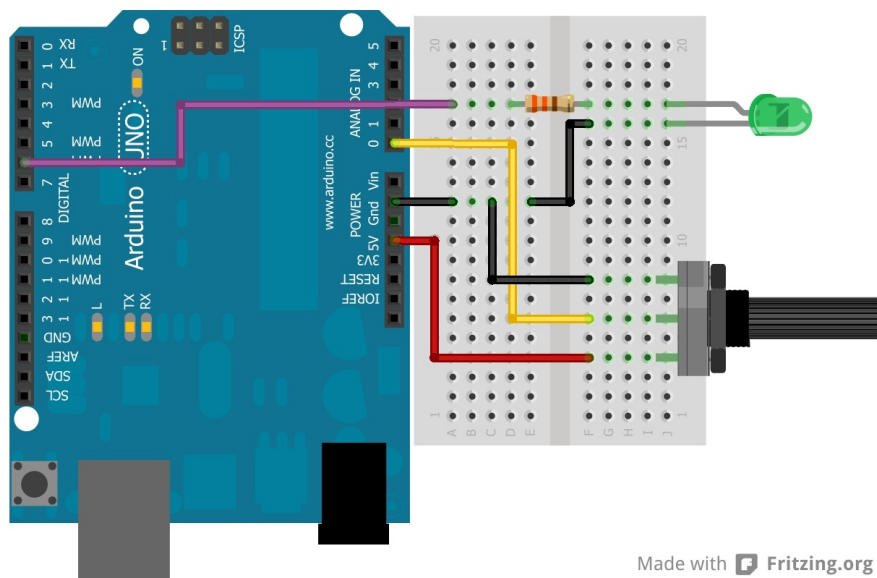
Příklad: Regulace jasu LED

Jako příklad si ukážeme zapojení, ve kterém budeme regulovat jas LED diody pomocí potenciometru.

Budeme potřebovat:

1. Deska Arduino
2. Nepájivé kontaktní pole s vodiči
3. LED dioda
4. potenciometr
5. 330 ohm resistor

Na trhu nalezneme celou řadu potenciometrů. Nejčastěji se používají ty s odporem kolem 10k ohm s lineárním průběhem (značeno B10K). Pokud máme všechny komponenty připravené, můžeme je zapojit podle následujícího schématu.



Obrázek 15.1: LED a potenciometr

V programu si musíme dát pozor na hodnoty, se kterými funkce pracují. Z funkce `analogRead()` vychází hodnoty 0 až 1023, kdežto `analogWrite()` čeká na rozsah hodnot 0 až 255. Musíme tedy zajistit převod hodnot. To je v tomto případě jednoduché, protože 256 (2^8) se vejde do 1024 (2^{10}) čtyřikrát. Nejjednodušším způsobem je tedy vydělení hodnot z `analogRead()` čtyřmi (Existuje i elegantnější způsob převodu hodnot, o kterém si povíme dále). Kód tohoto příkladu bude také velmi jednoduchý. Měření budeme provádět v těle funkce `loop()`.

```
1 byte led = 6; //pin s LED diodou
2 byte pot = A0; //pin s pripojenym potenciometrem
3 int val; //promenna pripravena k uchovani hodnot
4
5 void setup() {
6     //sem nic nepiseme
7 }
8
9 void loop() {
10     val = analogRead(pot)/4;
11     //cteni hodnoty na A0 a uprava rozsahu
12     analogWrite(led, val);
13     //generovani PWM
14 }
```

Kapitola 16

Podmínky

Pokud chceme, aby se určitá část kódu provedla pouze v určitých případech, přicházejí na řadu podmínky. Existují tři základní struktury, které rozdělují program podle zadaných podmínek. V lidské řeči by se daly vyjádřit jako:

1. Pokud platí podmínka, udělej to a to.
2. Pokud platí podmínka, udělej to a to. Pokud neplatí, udělej to a to.
3. Pokud je hodnota proměnné xy, udělej to a to. Pokud je yz, udělej to a to...

Než se však pustíme do psaní podmínek, musíme se podívat na způsob, jakým se dají v jazyce Wiring zadávat.

16.1 Porovnávací operátory

Porovnávací operátory slouží k zápisu podmínek. Jedná se o systém značek, které jsou pro počítač srozumitelné. Výsledkem porovnávací operace je logická hodnota true, nebo false. Rozlišujeme šest operátorů.

- $A == B$ – A je rovno B
 - Vrátil hodnotu true, pokud A má stejnou hodnotu, jako B.
- $A != B$ – A není rovno B
 - Vrátil hodnotu true, pokud má A jinou hodnotu než B.
- $A < B$ – A je menší než B
 - Vrátil hodnotu true, pokud je A menší než B.
- $A > B$ – A je větší než B
 - Vrátil true, pokud je A větší než B.
- $A <= B$ – A je menší nebo rovno B
 - Vrátil true, pokud je A menší nebo rovno B.
- $A >= B$ – A je větší nebo rovno B
 - Vrátil true, pokud je A větší nebo rovno B.

```

1  10 == 5 //neni pravda
2  10 != 5 //je pravda
3  10 < 5 //neni pravda
4  10 > 5 //je pravda
5  10 <= 5 //neni pravda
6  10 >= 5 //je pravda

```

16.2 Složené podmínky

Někdy dospějeme do situace, ve které je potřeba pracovat s nějakou složitější podmínkou. K tomuto účelu slouží tzv. logické operátory. Můžeme si je představit jako definici vztahu mezi více porovnávacími operátory.

- $X \ \&\& \ Y$ – a (konjunkce)
 - Výsledkem je true pouze v případě, když jsou true X i Y.
- $X \ \text{---} \ Y$ – nebo (disjunkce)
 - Výsledkem je true v případě, kdy je alespoň jedna z X a Y true.
- $!X$ – negace
 - Výsledkem je true, pokud je X false a naopak.

```

1  (1 == 2) \&\& (2 == 2) //false
2  (1 == 1) \&\& (2 == 2) //true
3
4  (1 == 2) || (2 == 3) //false
5  (1 == 2) || (2 == 2) //true
6  (2 == 2) || (2 == 3) //true
7  (2 == 2) || (3 == 3) //true
8
9  !(1 == 1) //false
10 !(1 == 2) //true
11 !(false) //true

```


16.3 if()

Ve většině programovacích jazyků se pro zápis podmínek používá slovo if. V jazyce Wiring je možné použít několik způsobů zápisu. Ty ale vždy začínají: if(podmínka).

```
1  //podminky s jedním příkazem
2
3  if(x > 120) digitalWrite(LEDpin, HIGH);
4
5  if(x > 120)
6    digitalWrite(LEDpin, HIGH);
7
8  if(x > 120){ digitalWrite(LEDpin, HIGH); }
9
10 //podminky s více příkazy – je nutné použít složené závorky
11 if(x > 120){
12     digitalWrite(LEDpin1, HIGH);
13     digitalWrite(LEDpin2, HIGH);
14     ...
15 }
```

16.4 else if()

Pokud chceme do podmínky přidat více možností, používá se zápis else if().

```
1  if (A < 800){
2      //příkazy
3  }
4  else if ((A < 500) && (A > 200)){
5      //příkazy
6  }
```

16.5 else

K části else se nepíše další podmínky. Slouží k určení příkazů, které se provedou, pokud ani jedna z předchozích podmínek není splněna.

```
1  if (A < 800){
2      //příkazy
3  }
4  else if ((A < 500) && (A > 200)){
5      //příkazy
6  }
7  else{
8      //příkazy
9  }
```

16.6 Switch

Switch je speciální druh podmínky. Speciální je v tom, že se zabývá pouze proměnnou a její hodnotou. Program prochází každou větev konstrukce switch a testuje hodnotu proměnné. Další rozdíl je v tom, že se může provést i více větví (což u if nelze). Pokud ale chceme, aby po provedení větve program pokračoval až za koncem konstrukce switch, musíme použít na konci větve příkaz break;

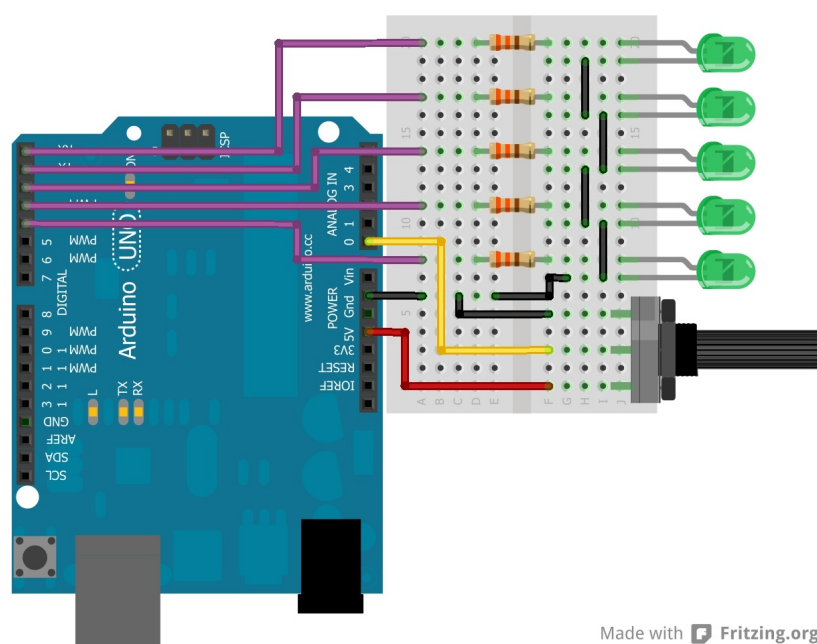
Syntaxe je následující:

```
1  switch (promenna){
2      case 1:
3          //pokud je hodnota promenne 1, provede se tato cast kodu
4          break; //po provedeni teto casti konstrukce switch konci
5      case 2:
6          //pokud je hodnota promenne 2, provede se tato cast kodu
7          break;
8      default:
9          /* pokud se hodnota promenne nerovna zadne z nabizenych moznosti,
10             provede se tato cast */
11  }
```

Kapitola 17

Příklad: Pás LED diod

Na závěr kapitoly si trochu pohrajeme s podmínkami. Vytvoříme aplikaci, která bude číst hodnotu z potenciometru a podle ní vybere led diodu. K tomuto příkladu budeme potřebovat stejné vybavení, jako u toho minulého, jen s větším počtem LED diod a resistorů. Pro předváděcí účely jsem zvolil pět diod.



Made with  Fritzing.org

Obrázek 17.1: Řada LED a potenciometr

Kód vyhodnocující data z potenciometru by mohl vypadat následovně.

```

1  byte led[] = {0,1,2,3,4}; //pole s piny pripojenych LED diod
2  byte pot = A0;
3  int val;
4
5  void setup() {
6      pinMode(led[0], OUTPUT);
7      pinMode(led[1], OUTPUT);
8      pinMode(led[2], OUTPUT);
9      pinMode(led[3], OUTPUT);
10     pinMode(led[4], OUTPUT);
11 }
12
13 void loop() {
14     val = analogRead(pot);
15
16     if(val > 800){
17         digitalWrite(led[0],HIGH);
18     }
19     else if(val > 600){
20         digitalWrite(led[1],HIGH);
21     }
22     else if(val > 400){
23         digitalWrite(led[2],HIGH);
24     }
25     else if(val > 200){
26         digitalWrite(led[3],HIGH);
27     }
28     else{
29         digitalWrite(led[4],HIGH);
30     }
31
32     delay(250);
33     digitalWrite(led[0],LOW);
34     digitalWrite(led[1],LOW);
35     digitalWrite(led[2],LOW);
36     digitalWrite(led[3],LOW);
37     digitalWrite(led[4],LOW);
38 }

```

Když se nyní podíváte na kód, jsou zde vidět opakování, ve kterých se mění pouze jedno číslo. Jak si v takovýchto případech ulehčit práci si ukážeme dále.