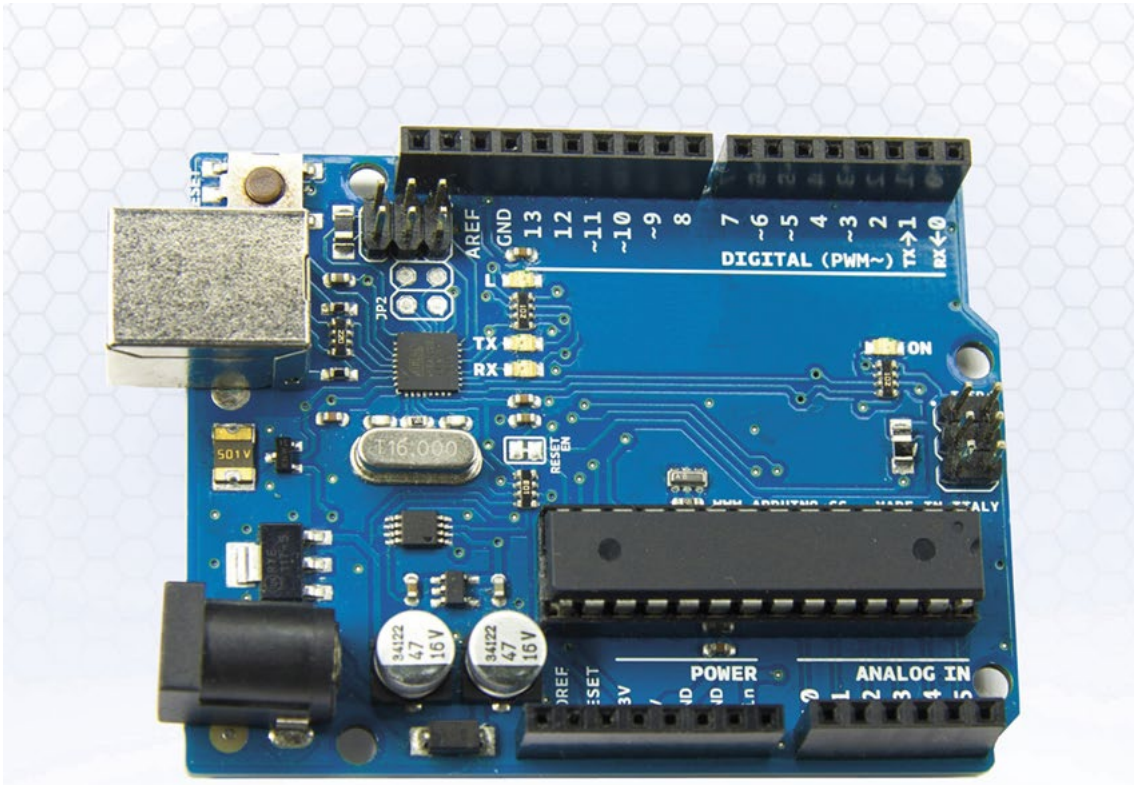


Arduino

Uživatelská příručka

Matúš Selecký

- Detailní popis možností hardwaru i softwaru
- Propojení s externími moduly
- Arduino a internet věcí

computer press

Arduino

Vyšlo také v tištěné verzi

Objednat můžete na
www.computerpress.cz
www.albatrosmedia.cz

Matúš Selecký
Arduino – e-kniha

ALBATROS MEDIA a.s.

Matúš Selecký

Arduino
Uživatelská příručka

Computer Press
Brno
2016

Obsah

O autorovi 11

O knize 11

Co v knize najdete? 12
Typografické konvence 13
Doplňující informace 14

KAPITOLA 1

Seznamte se, Arduino 15

Motivace 15
Arduino 15
Historie Arduina 16
Shrnutí 18
Licenční podmínky Arduina 18
Iniciativa Open-source 18
Arduino – ano, nebo ne? 19
Proč ano... 19
Proč ne... 20
Shrnutí 21
Produktová řada 21
Alternativní řešení 23
Emulátor Arduina 23
CodeBlocks Arduino IDE 23
Simuino 24
Autodesk 123D circuits 24
VBB4Arduino 25
Shrnutí 26
Datasheety 26
Komunity a tutoriály 27
Účast v komunitách 30
Komunikace 31
Hacker 32
Základní předpoklady 32
Potřebné znalosti 34
Shrnutí 35
Hardware hacking a reverzní inženýrství 35
Sběr informací 36
Fyzická inspekce 36
Odkrytování zařízení 36
Ovládací mechanizmus 37
Identifikace součástek 38
Výměna komponent 38
Debuggování 38
Tvorba softwaru 39
Psaní poznámek 39
Shrnutí 39
Arduino a reverzní inženýrství 40
Extrakce zdrojového kódu z Arduina 40

Konverze souboru HEX do původního kódu 41
Vývoj řešení – technický pohled 41
1. Sběr požadavků 42
2. Architektura systému 43
3. Blokový diagram 43
4. Realizace zapojení 45
5. Test-driven development 45
6. Testování a ladění 47
Vývoj řešení – pohled byznysu 47

KAPITOLA 2

Software 49

Motivace 49
Vývojové prostředí (IDE) 50
Arduino Software IDE 50
Visual Studio Community 51
Atmel Studio 51
Arduino plugin – Visual Micro 52
CodeBender 53
PlatformIO IDE 54
Shrnutí 54
Arduino IDE 55
Instalace vývojového prostředí 55
Prostředí Arduino IDE 61
První spuštění 67
Zpracování a běh kódu 69
Soubory zdrojového kódu 69
Kompilace kódu 70
Nahrání kódu 72
Běh kódu 74
Shrnutí 74
Konvence kódu 74
Názvosloví 75
Komentáře 76
Formátování 77
Programování 78
Shrnutí 78
Standardy programování 78
Kategorie: Jazyk 79
Kategorie: Předvídatelné spuštění 79
Kategorie: Defenzivní programování 81
Kategorie: Čistota kódu 81
Shrnutí 82
Pseudokód 82
Jazyk pseudokódu 82
Rozdělení pseudokódu 83
Struktura pseudokódu 83
Klíčová slova 84
Shrnutí 84
Programovací vzory 84
Imperativní programování 84
Objektově orientované programování 86
Funkcionální programování 89
Jiné typy programovacích vzorů 93

Shrnutí 94
Program (skica, sketch) 94
Globální definice 94
První program 95
Shrnutí 96
Verifikace a nahrání kódu 96
Datové typy 97
Konverze datového typu 100
Shrnutí 101
Proměnné 101
Kvalifikace proměnné 102
Volatile 102
Const 103
Vstupy a výstupy 103
pinMode() 103
digitalWrite() 105
digitalRead() 106
analogReference() 106
analogRead() 106
analogWrite() 107
analogReadResolution(), analogWriteResolution() 108
tone() 108
noTone() 108
shiftOut(),shiftIn() 108
pulseIn() 109
Serial.println() 109
Funkce 110
Testy 111
Podmínka if 111
Podmínka if...else 112
Cykly 112
Cyklus for 113
Cyklus while 114
Cyklus do...while 115
Shrnutí 116
Matematické operace 117
Ukazatel (Pointer) 120
Shrnutí 122
Pole 122
Paměť 124
Shrnutí 126
Regulární výrazy 127
Shrnutí 129
Řídicí příkazy 129
Knihovny 130
Vlastní knihovna 131
Shrnutí 134
Grafika 134
OpenFrameworks 135
Processing 135
PureData 139
Shrnutí 140
Optimalizace kódu 141
Datový typ 141

Globální a lokální proměnné 142
Spojení smyček 142
Využití cyklů 143
IF-ELSE vs. SWITCH-CASE 143
Shrnutí 145
Tipy a triky 145
1. Začít v malém a postupně rozšiřovat 145
2. Měnit jen jednu věc současně 145
3. Včasná příprava na ladění 145
4. Testování nového kódu 146
5. Rozumět upravovanému kódu 146
6. Experimentování 146
7. Zaseknutí v problému 146
8. Zmenšení programu 146
9. Zkrácený zápis funkcí 146
Hardwarové tipy 147
Diagnostika a ladění 149
Logování 149
Debuggování 150
Testy embedded zařízení 152
Arduino a unit testy 153
Test přihlašování 154
Test komunikace Arduino – PC 155

KAPITOLA 3

Hardware 157
Motivace 157
Sběrnice SPI 157
Základní specifikace 157
Programování SPI 159
Shrnutí 160
Sběrnice I2C 160
Základní specifikace 160
Programování I2C 161
I2C a Arduino 161
I2C skener adres 162
Shrnutí 163
Vstupy a výstupy 163
DDRx registr 164
PORTx registr 164
PINx registr 165
I/O piny a Arduino 165
Shrnutí 166
Asynchronní I/O 166
Multitasking – RTOS 167
RTOS a Arduino 167
Semafory 168
Arduino a semafor 169
Shrnutí 171
Časovače – milis() 171
Delay() vs. Milis() 171
Shrnutí 173
Přerušení 173
ISR rutina 174

Volání a maskování přerušení 176
Proměnné Volatile 177
Shrnutí 178
Paměti mikrokontrolérů AVR 178
Progmem 179
F() makro 180
Shrnutí 181
Watchdog 181
Princip fungování 182
Doporučení pro návrh 182
Watchdog a Arduino 182
Softwarový restart 184
Shrnutí 184
Bootloader 184
Programátor 184
Manuální instalace bootloaderu 185
Paměťové registry 186
Šetření energie 188
Způsoby šetření energie 189
Shrnutí 193
Napájení 193
Neregulovaný zdroj 193
Regulovaný zdroj 194
Baterie 194

KAPITOLA 4

Periferie 197
Motivace 197
Bezpečnost práce 198
Základní zásady 198
Elektrostatické výboje (ESD) 200
Shrnutí 201
Zobrazovací jednotky 201
LCD displej 202
OLED displej 205
Shrnutí 206
Numerická klávesnice 207
Princip fungování 207
Klávesnice a Arduino 208
Shrnutí 209
Snímače 209
Rozdělení snímačů 209
Arduino a snímače 210
Shrnutí 211
Pohony 211
Regulace otáček 212
H-můstek 214
Shrnutí 215
RFID 216
Princip fungování 216
RFID a Arduino 217
Shrnutí 218
GPS modul 218

Formát dat 218
GPS a Arduino 221
Shrnutí 223
Gyroskop a akcelerometr 223
Arduino a gyroskop 223
Shrnutí 225
Ethernet 225
Ethernet a Arduino 225
Shrnutí 228
Wi-Fi 228
Princip fungování 228
Wi-Fi a Arduino 229
Shrnutí 231
Bluetooth 231
Princip fungování 231
Bluetooth a Arduino 232
Shrnutí 234
GSM komunikace 234
Princip fungování 234
GSM a Arduino 236
Shrnutí 238
Spínání síťového napětí 238
Technický popis 239
Síťové napětí a Arduino 239
Shrnutí 242
Real-time clock (RTC) 242
RTC a Arduino 242
Shrnutí 243
Arduino callback 243
Princip fungování 243
Callback a Arduino 244
Shrnutí 245
Záznam dat 245
SD karta a Arduino 245
Databáze a Arduino 247
Shrnutí 248
Arduino Cluster 248
Princip fungování 249
Shrnutí 251
Arduino a Raspberry Pi 251
Arduino a server 254
Arduino a PowerShell 255
USB a Arduino 256
TCP a PowerShell 258
Notifikace v oznamovací oblasti 260
Shrnutí 261
Arduino a Bash 261
TCP komunikace 261
Shrnutí 262
Arduino a Python 263
Shrnutí 265
Arduino a Android/Apple iOS/Windows Mobile 265
Android 265

Apple iOS 267
Windows Mobile 268
Shrnutí 269
Arduino a SSH 269
Shrnutí 270
Vlastní moduly 271
Návrh a kreslení schémat 271
Realizace DPS 276
Shrnutí 278

KAPITOLA 5
Internet of Things 279
Motivace 279
Internet of Things 279
Message Oriented Middleware (MOM) 281
Komunikační protokoly 283
STOMP 284
AMQP 284
MQTT 284
CoAP 286
ZeroMQ 287
REST 289
Shrnutí 292
Broker server 292
Apollo 292
Rabbit MQ 295
Mosquitto 298
Shrnutí 299
Přílohy 301
Otázky na pracovním pohovoru 301
Rezervovaná slova a znaky 303
Datové typy 305
Základní tabulka ASCII znaků 305
Řecká abeceda 306
Vzorce 307
Schematické značky 308
Zapojení trojúhelník-hvězda 308
Označení rezistorů 309
Síťové adaptéry 309
Volt-ampérové charakteristiky 310
Napěťové úrovně integrovaných obvodů 311
Veličiny a jednotky 312
Převod jednotek 312
Přepočet frekvence 312
Matematika 313
Slovník 319
Bibliografie 327
Rejstřík 333

O autorovi

V oblasti ICT působím od roku 2008. Ve své profesní praxi jsem se věnoval mnoha činnostem z oblasti testování, správy zabezpečení sítí, optimalizace, automatizace a automatické verifikace systémů. Při řešení ve velké míře navrhuji, tvořím a využívám automatizované nástroje, které mi pomáhají efektivně řešit vzniklé situace a požadavky.

Před dvěma roky jsem u jednoho projektu realizovaného ve volném čase přišel do styku s Arduinem. Poznal jsem výzvy a úskalí vývoje embedded zařízení a získané znalosti a zkušenosti se snažím komplexně sumarizovat v této knize.

O knize

Pokrok moderních technologií snižuje cenu a zvyšuje dostupnost elektronických prvků, technických prostředků a zařízení pro běžné lidi. V době internetu a snadné dostupnosti informací je možné získávat znalosti téměř kdykoliv a kdekoliv. Toto všechno dává možnost i kreativním lidem a nadšencům tvořit produkty a služby, které zlepšují každodenní život. V některých případech se nápady mění v nové podnikatelské plány.

Arduino je jednou z takovýchto technologií. Řešení pochází z dílny italských inženýrů. Už i v našich zemích (Česká a Slovenská republika) existuje několik knih, článků, blogů, portálů a webových stránek popisujících tuto technologii.

Na trhu však chybí literatura, která by amatérům (lidem, kteří se touto oblastí neživí) problematiku vývoje a výroby elektronických zařízení přiblížila komplexním a čtivým způsobem. Právě tato kniha chce poskytnout takzvaný celkový přehled o problematice, v angličtině takzvaný „big picture".

Technická literatura je už v době svého psaní poměrně neaktuální, jelikož proces tvorby knihy trvá přibližně rok. U zařízení typu Arduino je životní cyklus desek a rozšiřujících modulů poměrně krátký. Navíc po vydání knihy jsou obsažené informace za rok či dva už neaktuální a v podstatě i nezajímavé a zbytečné.

Vzhledem k dostupnosti informací na internetu je poměrně jednoduché najít, co člověk potřebuje. Stačí do webového prohlížeče zadat klíčové slovo a za sekundu jsou k dispozici miliony výsledků. Pro Arduino se vrátí 26 milionů výsledků, a to vytváří další problém.

Tím je právě tato jednoduchá a rychlá dostupnost velkého množství informací, která začíná být v dnešní době více na škodu než k užitku. V době internetu převažuje kvantita obsahu

nad kvalitou. Dalším z problémů je fragmentace informací. Kniha se proto snaží poskytnout řešení ve formě sdružení relevantních informací na jednom místě.

Z těchto důvodů je koncept knihy postaven jinak než většina běžných zdrojů informací k Arduinu. Na první pohled by se mohlo zdát, že pokud kniha obsahuje informace z každé oblasti, je to o všem a zároveň o ničem. Důvodem, proč obsah knihy vytváří takový pocit, je široký rozsah obsažených témat a oblastí. Témata a jejich rozsah byly zvoleny ve snaze poskytnout úvod do vývoje embedded zařízení.

Pokud si čtenář detailně nastuduje jazyk C/C++ a technologii programování mikroprocesorů AVR od společnosti Atmel, zvládl v podstatě celé Arduino. S drobnými obměnami je schopen realizovat jakýkoliv program. Tuto knihu nebude potřebovat. Dovolím si však tvrdit, že potom bude mít na Arduino jiný pohled a zjistí, že ho v mnoha věcech deska či Arduino IDE omezují.

Schopnost programování mikroprocesorů ale nepostačuje. Mikroprocesor sám o sobě nedokáže poskytovat tak rozsáhlou funkcionalitu, jaká se požaduje například při realizaci domácího alarmu nebo amatérské meteorologické stanice.

Když se člověk postupně ponoří do problematiky, napadnou ho otázky: Jak řeší takovýto problém profesionálové v praxi? Co všechno by mělo mít kvalitní softwarové a hardwarové řešení? Jak se propojují zařízení s reálným světem? Jak nejlépe zabezpečit interakci s uživatelem? Jaké jsou moderní trendy v oblasti? Je možné hobby řešení transformovat na komerční řešení? A mnoho jiných otázek.

Tato kniha by měla být pro nezasvěcené prvním úvodem do problematiky embedded zařízení a ukázat další nasměrování na odpovídající zdroje informací. Čtenář by měl v knize najít i vysvětlení, proč jsou jednotlivé oblasti důležité a jak spolu souvisí. To, zda byly záměr a ambice knihy splněny, zůstává na čtenáři.

Co v knize najdete?

V knize se dozvíte například: Jaká pravidla při tvorbě embedded kódu označují za důležitá experti z NASA. Jak připojit Arduino do počítačové sítě bez ethernet modulu. Co potřebujete k ovládání Arduina z operačního systému Windows a Linux. Jak na emebedded systémech šetřit energii. Jakým způsobem je možné zajistit automatickou detekci a obnovu z chybových stavů. Jak začlenit Arduino do infrastruktury IoT s komunikačním modelem Publisher-Subscriber. Prozkoumáte rovněž mnoho jiných zajímavých oblastí souvisejících s vývojem embedded zařízení a Arduina.

Kniha je rozdělena do pěti kapitol a doplněna souborem příloh.

1. kapitola se věnuje seznámení s Arduinem. Seznámíte se s příběhem, který obsahuje zklamání, kopírování nápadů a interní spory mezi zakladateli projektu. Po úvodu následuje krátké zamyšlení nad vhodností a nevhodností Arduina pro projekty. V textu jsou dále zmíněny oblasti, které uvedou čtenáře do problematiky elektronických zařízení, jejich vývoje, analýzy, vylepšování či úprav v domácím prostředí. Rozšiřujícími částmi jsou témata o hackerech, hardwaru hackingu či reverzním inženýrství.

2. kapitola se zaměřuje na softwarovou část Arduina. Vysvětluje programovací jazyk, přístupy a principy. Kapitola obsahuje informace, které odhalují zajímavé nástroje k nasazení při řešení. Za zmínku stojí například ukázka regulárních výrazů pro pokročilou práci s textem či malá ochutnávka audiovizuálních frameworků pro tvorbu interaktivních rozhraní. Část textu se věnuje principům, tipům a trikům, jak tvořit a ladit zdrojový kód.

3. kapitola se věnuje hardwaru Arduina. Mezi popisovanými oblastmi jsou sběrnice, registry, operační paměť, časovače, vstupy/výstupy a jejich efektivní využívání při programování. Mezi zajímavé části patří sekce věnovaná šetření energií či principům zvýšení stability zařízení a odolnosti vůči chybám.

4. kapitola nabízí seznámení s principy propojení Arduina a externích modulů. Moduly se označují jako shieldy a rozšiřují funkcionalitu Arduina o nové technologie jako GSM, GPS či RFID. V kapitole se nezapomnělo ani na principy připojení Arduina k síti bez technologie ethernet, na Wi-Fi, Bluetooth či propojení s mobilními telefony.

5. kapitola se zaměřuje na krátký úvod do Internet of Things, což je oblast, která v posledních letech razantně získává na popularitě. V textu najdete popsaný komunikační model Publisher-Subscriber, který se dá snadno vytvořit a otestovat v domácích podmínkách.

Přílohy – knihu uzavírá soubor příloh, který má fungovat jako kolekce užitečných informací. Na seznamu se nachází například tabulky elektrotechnických značek a výpočtů či matematické vzorce. Bonusovou přílohou je ukázka otázek používaných na pohovorech na pozice typu vývojář embedded zařízení.

Typografické konvence

V knize se používá kvůli lepší přehlednosti formátování textu podle následujících typografických konvencí.

Informace související s popisovanou problematikou, které by mohly být zajímavé či důležité, jsou v textu rozděleny do tří kategorií: Tip, Poznámka a Upozornění. Všechny typy jsou zvýrazněny tímto způsobem:

Poznámka: U aritmetických operací je potřeba dávat pozor na používané datové typy. Jejich nesprávným používáním se mohou výsledky zkreslit, což má nežádoucí vliv na běh programu. Obzvlášť problémové jsou operace dělení.

Zdrojový kód, který lze aplikovat na Arduino, je formátován takto:

```
int led = 9;           // nastavení pinu LED
int brightness = 0;
int fadeAmount = 5;

void setup() {         // nastavení komunikace
  pinMode(led, OUTPUT);
}
```

```
void loop() {
  analogWrite(led, brightness);
}
```

Doplňující informace

V knize se využívá princip, který se v zahraniční literatuře označuje jako „*rule of thumb*“. Jde o příklady a principy, které se používají k vysvětlení či úvodu do problematiky. Tyto příklady a principy nejsou vytvořeny, aby byly striktně přesné a aplikovatelné na každou situaci reálného světa. Jejich účelem je jednodušší vysvětlení a pochopení textu.

Na základě zkušeností z praxe se v knize používá značné množství anglických pojmů, které se nepřekládají. Hlavním důvodem je zachování významu slov v kontextu probírané problematiky. Dalším důvodem je možnost vyhledávání doplňujících informací v zahraniční literatuře, přičemž zpětný překlad do angličtiny by nemusel být jednoduchý. Pro pobavení následuje příklad z praxe s nevhodným překladem:

Během testování softwaru byl vznesen požadavek na ověření přítomnosti stopovacích koláčků. Tento požadavek zní poměrně vtipně, když nikde v okolí nejsou na stole žádné koláčky. Při hlubší analýze otázky je zřejmé, že má jít o ověřování *tracking cookies* ve webovém prohlížeči, které se používají ke sledování uživatelů.

Dalším příkladem nevhodného překladu je v oblasti síťových komunikací přepínač CISCO. Ne každý si dokáže představit, co toto zařízení dělá. Na druhou stranu, když se zmíní CISCO switch, většina lidí z oboru (dokonce i netechnických) si dokáže představit, o co se jedná, a diskuze může plynule pokračovat dále.

Upozornění: Při realizaci jednotlivých zapojení uvedených v knize je potřeba dodržovat zásady bezpečnosti práce s elektronickými zařízeními. Vždy si důsledně zkontrolujte zapojení celého obvodu před jeho připojením k napětí.

KAPITOLA 1

Seznamte se, Arduino

V této kapitole se dozvíte o:
- Historii Arduina
- Výhodách a nevýhodách Arduina
- Iniciativě Open-source
- Licenčních podmínkách
- Arduino komunitách a portálech
- Hackingu a reverzním inženýrství
- Vývoji technického řešení

Talentovaný zasáhne cíl, který jiní zasáhnout nedokáží. Génius zasáhne cíl, který jiní nevidí.
– Arthur Schopenhauer

Motivace

První kapitola se věnuje seznámení s Arduinem. Jeho příběh je zajímavý a turbulentní. Od tématu diplomové práce k milionovému byznysu vede dlouhá cesta, která ukrývá tvrdou práci, lidské osudy, úspěchy a zklamání. Zakladatelé žijí aktivním elektronickým sociálním životem, aktivně blogují a přispívají do komunit. Odtud se dá získat většina aktuálních informací.

Od úvodního seznámení se přechází dále k práci s Arduinem. Krátké zamyšlení je věnováno otázkám, proč by si měl člověk pro svůj projekt vybrat právě Arduino, a naopak proč ne. Odkud má získávat informace, podporu, návody, řešení problémů a inspiraci pro projekty.

Pro zajímavost jsou v kapitole zmíněny oblasti jako hardware hacking či reverzní inženýrství, jelikož jde o oblasti úzce související s hobby projekty, snahou o vylepšení či úpravu existujících zařízení. Závěr kapitoly se věnuje úvodu do problematiky vývoje řešení.

Arduino

Arduino je fenoménem posledních let. V období 2005–2013 se prodalo 700 000 oficiálních zařízení Arduino. Na trhu Open-source Hardware market (OSHW), kde Arduino působí, se do roku 2011 dostalo mezi TOP 13 společností. Získaný podíl trhu se odhaduje na 3–7 %. Roční příjem se v roce 2013 pohyboval na úrovni jednoho milionu dolarů.

Arduino samo o sobě nepřináší na trh nic nového. Mikroprocesory, breadboard desky, propojení spínačů, LED diod, různých aktivních a pasivních součástek, to všechno tu bylo už dávno předtím. Dokumentace k jednotlivým mikroprocesorům, schémata zapojení, programovací

rozhraní nebo platformy jako BASIC Stamp 1 tu už byly o desítky let dříve. Arduino desky, dokumentace API, knihovny, frameworky, IDE, to všechno se podobá starším řešením, která existují již desítky let.

Čím je tedy Arduino výjimečné, že se mu tak daří? Platforma Arduino je založena na sociální inovaci. Prostřednictvím silného marketingu se podařilo vytvořit komunitu lidí, kteří se mohou prostřednictvím jednoduchého a srozumitelného vývojového prostředí dostat k programování mikroprocesorů bez nutnosti studovat složitou architekturu a logiku programování celého systému. Došlo ke zjednodušení programování mikrokontrolérů prostřednictvím IDE, které se stará o všechno na pozadí bez nutnosti zásahu uživatele.

Poznámka: Pro porovnání, například instalace bootloaderu na mikroprocesor (viz část věnovanou bootloaderu) je poměrně komplikovaná a vyžaduje relativně složité a opatrné nastavování registrů.

Tvůrci Arduina se zaměřili na mladé studenty, designéry a lidi z netechnicky orientovaných profesí, kteří mají minimální nebo žádné zkušenosti s programováním hardwaru. Arduino je platforma, která umožnuje se za jeden večer od nulových zkušeností a znalostí dostat k blikajícím diodám ovládaným mikroprocesorem.

Historie Arduina

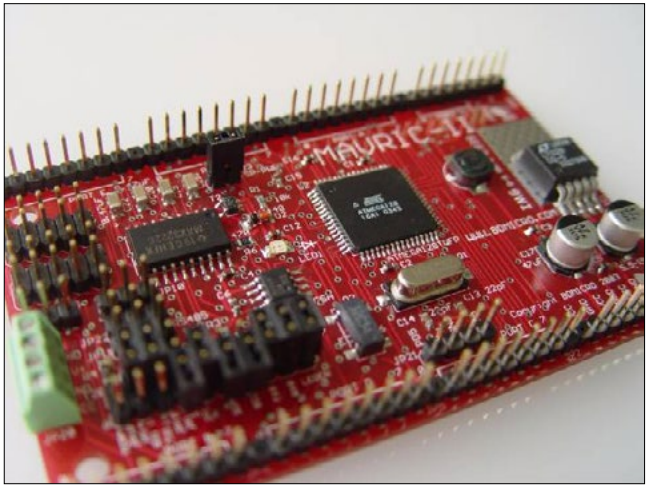
Příběh Arduina se začal psát v roce 2003, kdy student Hernando Barragán pracoval na své diplomové práci na fakultě Interaction Design Institute Ivrea (IDII) v Itálii. Cílem práce bylo ulehčení práce umělcům a designérům pracujícím s elektronikou. Umělcům a designérům mělo být ulehčeno od technických aspektů programování mikroprocesorů a mohli se tak věnovat svému primárnímu cíli – designu a umění.

Mezi základní požadavky na vytvořenou platformu patřilo:

- Jednoduché integrované vývojové prostředí (IDE) založené na jazyku Processing (*processing.org*)
- IDE funkční na systémech Microsoft Windows, Mac OS X a Linux, s jednoduchým editorem, určené k vytvoření softwarového programu
- Jednoduchý programovací „jazyk“ nebo „framework“ pro mikroprocesory
- Kompletní integrace nástrojů (transparentní pro uživatele)
- Bootloader pro snadné nahrávání programů
- Serial Monitor na kontrolu a odesílání dat z/do mikroprocesoru
- **Open-source software**
- **Open-source hardware návrhy založené na mikroprocesorech Atmel**
- Komplexní online reference příkazů a knihoven, příklady, tutoriály, fóra a ukázky projektů

Poznámka: Části s open-source požadavky jsou důležité pro další pokračování příběhu Arduina.

Diplomovou práci vedli Massimo Banzi a Casey Reas. Výstupem diplomové práce bylo zařízení s označením Wiring, které Hernando vytvořil prostřednictvím několika prototypů s různými verzemi mikroprocesoru.



Obrázek 1.1: Třetí prototyp zařízení Wiring

Hernando absolvoval v roce 2004 studium s vyznamenáním. V průběhu let 2003–2005 se dokonce finální verze zařízení Wiring prodávala přes internet do celého světa. Zde ale platforma Wiring končí.

V roce 2005 se od komercializovaného projektu Wiring odpojili Massimo Banzi a David Mellis, kteří založili vlastní projekt s názvem Arduino. Jako hlavní realizátoři projektu jsou uváděni Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino a David Mellis. Hernando nebyl součástí nového týmu, nebyl do něho ani přizván. Avšak jeho podklady posloužily k dalšímu vývoji a komercializaci projektu Arduino.

Poznámka: Název Arduino vznikl podle místního baru, kde se zakladatelé projektu setkávali. Bar nesl jméno podle italského hraběte, jenž se později stal italským králem.

V letech 2008–2014 se kvůli interním sporům zakladatelů projekt Arduino rozdělil na dvě větve: *Arduino SRL* a *Arduino LLC*. Zakladatelé se rozdělili na dvě skupiny, které udržují projekt naživu (*www.arduino.cc*, který vlastní Arduino LLC, a *www.arduino.org*, který udržuje Arduino SRL).

Vzhledem k problémům s registrací komerčního názvu Arduino (který drží v Evropě společnost Arduino SRL) vznikl nový komerční název Genuino pro trh mimo USA, který vlastní Arduino LLC, přičemž v USA se tento produkt označuje jako Arduino. Vždy se ale jedná o stejný projekt a produkt.

Poznámka: Bez ohledu na to, kdo vlastní práva pro komerční název, budou v knize nadále používány názvy Arduino, i když se bude jednat o produkty, služby nebo informace primárně od společnosti Arduino LLC, ale využívané na území EU (mimo USA).

Shrnutí

Arduino je stále živý projekt, jeho pokračování je tedy otevřeným příběhem. Každopádně bude sledování dalšího životního cyklu této velmi populární platformy zajímavé.

Licenční podmínky Arduina

K produktům značky Arduino se váží licenční podmínky, s nimiž je vhodné se před používáním seznámit. Ve stručnosti najdete níže několik nejdůležitějších bodů, které byste měli znát:

- Arduino není možné používat tam, kde bezpečnost představuje kritický aspekt funkcionality. To znamená, kde na funkcionalitě, stabilitě a bezpečnosti systému závisí lidské životy či zdraví uživatelů. Typickým příkladem jsou zdravotnické, letecké, energetické či vojenské systémy a zařízení. Je pravdou, že Arduino by se za žádných okolností nemělo dostat do podobných aplikací, jelikož na systémy z daných oblastí se kladou enormně vysoké technické požadavky.
- Zakoupením produktů Arduino na sebe uživatel bere zodpovědnost za dodržování legislativních, bezpečnostních a regulačních požadavků, které jsou aktuálně platné v dané zemi používání.
- Při integraci Arduina do komerčního řešení se nemusí zveřejnit design a logika řešení.
- Využívání a modifikace jádra Arduina a knihoven vyžaduje zachování licenčních podmínek LGPL.
- Při využití diagramů desky k vytvoření nového komerčního produktu je nezbytné, aby zůstaly zachovány současné licenční podmínky Creative Commons Attribution Share-Alike.
- Úprava a komerční redistribuce Arduino IDE není možná.

Poznámka: Pro podrobnější vysvětlení či aktuální verzi podmínek je vhodné prostudovat originální podmínky EULA, záruční podmínky a FAQ, kde bývají nejdůležitější otázky vysvětleny zjednodušeně.

Iniciativa Open-source

Open-source hardware (OSH) představuje podobnou ideologii jako open-source software (OSS). U OSS se sdílí kód softwarového řešení. U OSH jsou volně dostupné všechny podklady potřebné k designu a výrobě vlastního hardwaru. Jedná se například o mechanické výkresy, schémata, kusovníky, vrstvy desek plošných spojů, zdrojový kód HDL či data o rozvržení součástek.

Jednou z myšlenek Arduina bylo, aby lidé mohli studovat a vylepšovat hardware a ten dále sdílet s ostatními. Proto tvůrci zpřístupnili pod licencí *Creative Commons Attribution Share-Alike* (CCAS-A) všechny soubory vytvořené při vývoji Arduina.

Licence CCAS-A umožňuje fyzickým a právnickým osobám upravovat a vydávat hardware na základě získaných podkladů za podmínky odkazování na původního tvůrce a zachování licencování. V krátkosti by se dalo shrnout, že licence Creative Commons Attribution Share-Alike umožňuje:

- **Sdílení:** Umožňuje kopírovat a redistribuovat materiály na jakémkoliv médiu a v jakémkoliv formátu.
- **Úprava:** Umožňuje míchat, měnit, upravit materiály za jakýmkoliv účelem včetně komerčního.
- **Odkazování:** Při tvorbě řešení je nezbytné odkázat se na použité licencování a uvést provedené změny.
- **Zachování:** Při úpravě, změně nebo vybudování řešení je nutné zachovat původní licencování.
- **Další omezení:** Aplikováním právní legislativy nebo technických opatření nelze omezovat ostatní vývojáře v tom, co jim umožňuje licence.
- **Odvolání:** Licenční podmínky není možné odvolat.

Plné znění licence CCAS-A v3.0 najdete na portálu *Creativecommons.org* (*https://creativecommons.org/licenses/by-sa/3.0/legalcode*).

Hlavní konkurenční výhoda této platformy je postavena na ceně, relativní jednoduchosti propojení s periferiemi a jednoduchém programovacím jazyku. Arduino Software IDE je programovací prostředí, v němž lze tvořit programový kód k ovládání Arduina. IDE je taktéž distribuováno pod licencí open-source. Konkrétně pro prostředí Java je IDE vydané pod licencí GPL a pro C/C++ pod licencí LGPL.

Arduino – ano, nebo ne?

Jako všechno na světě má i Arduino své výhody a nevýhody z hlediska používání a vhodnosti či nevhodnosti pro řešení.

Proč ano...

Několik důvodů, proč si pro hobby projekt vybrat právě Arduino:

- Relativně nízká cena a dostupnost v porovnání s komerčními či průmyslovými řešeními (například PLC systémy či mikroprocesorové desky typu AVRPLC16). U hobby projektů ve většině případů rozhoduje cena.
- Široká podpora komunity, kterou tvoří nadšenci, často i lidé z praxe s technickým vzděláním a několikaletými profesními zkušenostmi.

- Open-source projekt, volně dostupné IDE, schémata, diagramy.
- Produkty, které využívají volně dostupné designové informace, se snaží přinést nové a lepší řešení, jež by zaujalo elektronické nadšence.
- Dostupnost různých modulů a flexibilita umožňující vytvořit téměř jakékoliv řešení.
- Připojitelnost a programovatelnost přes USB. Nevyžaduje programátor a znalosti jeho používání.
- Snadná dostupnost informací, návodů, schémat, zdrojových kódů.
- Relativně krátká doba vývoje funkčního řešení. Vhodné k tvorbě prototypů či proof-of-concept.
- Skvělá platforma k výuce na školách.
- Vhodný prostředek pro hackery, kteří rádi vylepšují a tvoří.

Proč ne...

I přes značnou popularitu Arduina existuje několik důvodů, proč tato platforma není vhodná k výuce či komerčním aplikacím.

- Z pohledu vývoje komerčních produktů, které se mají distribuovat, udržovat a servisovat, není tato platforma příliš vhodná a to hlavně kvůli krátkému životnímu cyklu produktů. S příchodem nového modelu přijde stará deska za rok či dva o podporu a dostupnost na trhu.
- Podobná situace panuje i v případě shieldů, které přestanou být dostupné a přijdou o podporu. S tím souvisí i podpora knihoven a ovladačů, které převážně vytváří komunita nadšenců.
- Arduino neposkytuje právě nejlepší vstup do světa programování, obzvlášť pro samouky. Uživatel může získat chybné návyky a konvence tvorby kódu, které se budou později přenášet i do dalšího kódu. Diskusní fóra, literatura či tutoriály příliš neřeší tento aspekt práce, což se v konečném důsledku může projevit v nízké kvalitě výsledků. Získání solidních základů vyžaduje kvalitní vedení zkušeným člověkem.
- Hotová řešení stažená z internetu často postrádají komentovaný kód, funkce nejsou dostatečně odladěny ani optimalizovány. Bývají funkční pro určité verze IDE a revize shieldů. (Kód funkční na rev. 2 s největší pravděpodobností nebude fungovat na rev. 3). Po zveřejnění se kód většinou už neupravuje ani neudržuje.
- Použitím kódu stylem **Ctrl+C** a **Ctrl+V** začátečník kódu neporozumí, stejně jako principům jeho fungování. Nemá tedy žádoucí pedagogický efekt.
- Při vývoji komerčních IT/SW/HW projektů se využívá sofistikovaná správa verzí a sledování pokroku či ladění chyb. Arduino IDE nabízí jen základní možnosti pro domácí uživatele, což je vzhledem k ceně (zdarma) pochopitelné. Mnoho nadšenců, kteří to myslí s programováním seriózně, využívá GitHub jako nástroj ke správě verzí.
- Komerční aplikace často vyžadují certifikaci ke splnění náročných požadavků. Obzvlášť v životně důležitých aplikacích, jako jsou vojenské, letecké či medicínské přístroje, není

Arduino ani jiný spotřebitelský produkt (Raspberry Pi, Parallella Board) použitelný. Certifikace je cenově a technicky náročný proces, který se u produktů s krátkým životním cyklem nevyplatí.

- Při masové produkci je jednoduché objednat mikroprocesory v tisících a připravit sériovou výrobu. Objednání Arduina v tisícových kusech není snadné ani levné.
- Platforma (v současnosti) nenabízí například možnost nasazení šifrovacích protokolů či jiných bezpečnostních mechanismů. Na bezpečnost se přitom v komerční praxi kladou stále vyšší nároky.
- Nevýhodou může být cena a dostupnost oficiálních modulů, které jsou často dražší a bývají rychle vyprodané. Alternativou jsou shieldy a moduly od jiných (například čínských) výrobců či komunit. Problémem však bývá kompatibilita s jinými moduly či zařízeními, ovladače či zdrojové kódy, které je často potřeba značně ladit a přizpůsobovat.

Shrnutí

Arduino je platforma vhodná pro umělce a designéry, kteří využívají elektroniku při své kreativní činnosti. Dále je velmi vhodná pro hobby projekty a nadšence z oblasti elektroniky, programátory, kteří si chtějí v domácích podmínkách a cenově rozumných relacích vytvořit svůj jednoduchý elektronický projekt.

Platforma se hodí k rychlému prototypování a výrobě proof-of-concept řešení, která je možné následně použít k prezentačním účelům potenciálním investorům či zákazníkům. Nehodí se příliš k tvorbě konečných produktů, které se budou vyrábět sériově a distribuovat zákazníkům.

Jako výuková pomůcka představuje Arduino velmi dobrý prostředek vzhledem ke své ceně, dostupnosti a relativní jednoduchosti programování. Výuka ale vyžaduje zkušeného učitele, který dokáže při vysvětlování propojit Arduino s reálným světem a požadavky, které se kladou na komerční řešení.

Systémoví inženýři, kteří se zabývají vývojem embedded zařízení, nejsou velkými příznivci této platformy, jelikož celý systém přidává novou vrstvu mezi inženýra a hardware a omezuje možnosti, jimiž je možné z hardwaru vytěžit maximum. Skeptické názory a náladu navíc podporuje i to, že Arduino z technického hlediska nepřináší nic nového.

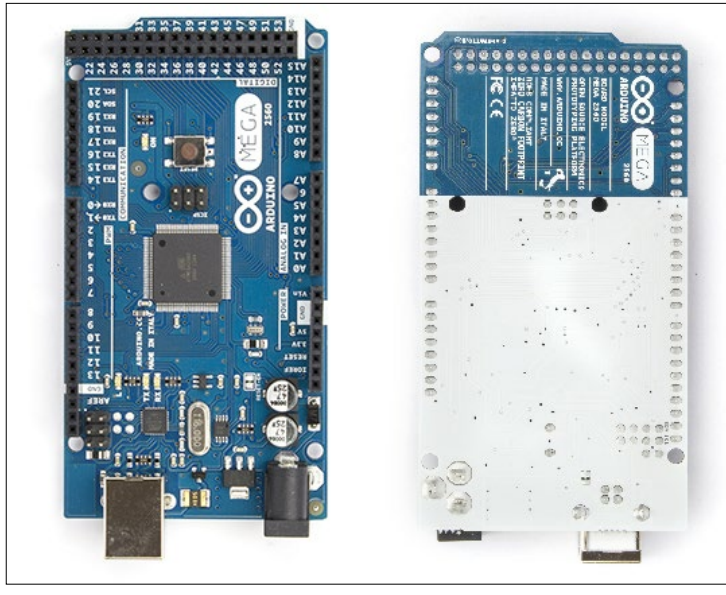
Produktová řada

Arduino je komerční produkt určený pro koncového spotřebitele. Proto má i velmi krátký životní cyklus. Od roku 2005 prošlo Arduino velmi rychlým vývojem, řadou revizí, nových verzí, různých vylepšení a obměněnými mikrokontroléry.

V době psaní knihy bylo na trhu dostupných celkem 21 verzí Arduina. Při výběru vhodného modelu Arduina je rozhodujících několik faktorů, jako jsou například rozměry desky, počet připojitelných analogových a digitálních vstupně-výstupních pinů, velikost paměti, rychlost

mikroprocesoru či možnosti rozšíření o externí shieldy. Lepší přehled o konfiguracích aktuálně dostupných verzí Arduina získáte na stránkách projektu.

Jedním z bodů, které mohou ovlivnit výběr desky, je i velikost projektu a s tím související délka vytvořeného kódu. Na diskuzním fóru (*forum.arduino.cc*) najdete několik desítek příspěvků, kde začátečníci žádají o rady při výběru vhodné desky.



Obrázek 1.2: Arduino Mega

Rozšiřující moduly

Arduino nabízí značné možnosti rozšíření funkcionality o externí moduly. Moduly navržené a označené logem Arduino se označují jako shieldy. Na trhu existuje mnoho shieldů v podobě snímačů, konvertorů mezi komunikačními protokoly či rozhraními od různých výrobců v odlišných cenových kategoriích. Mnoho modulů jiných značek (Sparkfun, Adafruit) se testuje a jsou oficiálně podporovány. Jejich prodej je často zprostředkován i z oficiálních stránek projektu Arduino.

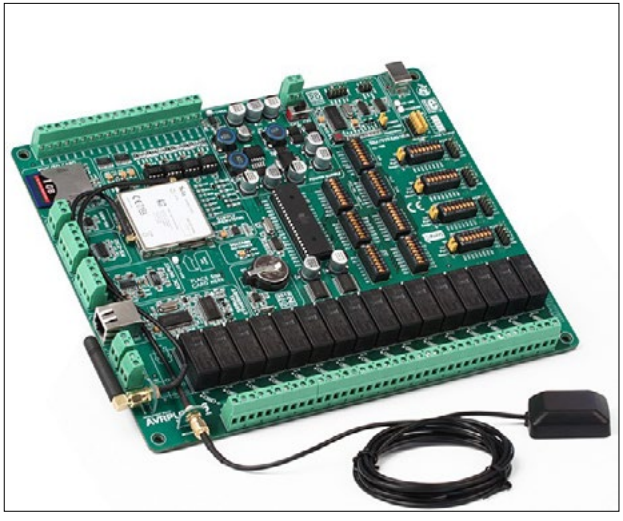
Tip: Před zakoupením jakéhokoliv externího modulu ověřte dostupnost řídicího programu. Samotná dostupnost však nezaručuje jeho funkcionalitu a kompatibilitu s používanou verzí Arduina.

Kde koupit Arduino

Arduino a jednotlivé externí moduly je možné zakoupit téměř v každém internetovém obchodě zabývajícím se elektronikou či hobby oblastí. Při zadání slova Arduino do vyhledávače se na prvních pozicích zobrazují obchody, kde můžete zakoupit všechno potřebné pro hobby projekt. Otázkou zůstává už jen cena, skladové zásoby a doba dodání.

Alternativní řešení

Na seriózní, ale i hobby vývoj embedded řešení se často z technického hlediska více hodí zařízení od výrobců mikroprocesorů a elektronických součástek jako NXP, ARM, Atmel nebo Mikroelektronika. Produkty mají k dispozici velmi detailní a kvalitní dokumentaci, širokou škálu nástrojů umožňující pokročilé programování, kompilaci a debuggování na profesionální úrovni.



Obrázek 1.3: Vývojová deska Mikroe PLC

Emulátor Arduina

S Arduinem se můžete seznámit i bez fyzické desky. Existuje několik virtuálních emulátorů a simulátorů, které virtualizují část funkcionality. Vždy se jedná jen o emulaci funkcionality reálné desky, což se projevuje v omezených možnostech v porovnání s reálnou deskou.

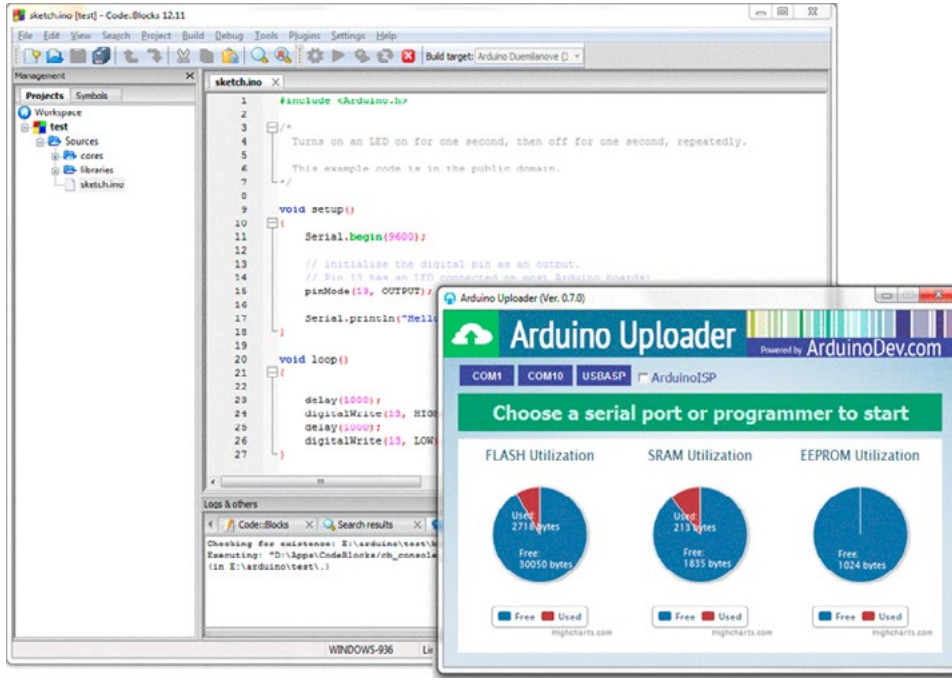
Při práci s virtualizovaným nástrojem navíc člověk přichází o fyzický kontakt s hardwarem. V některých případech se ale možnost emulace hodí, obzvlášť při cestování, kde není praktické vozit s sebou veškerý hardware a potřebujete například rychle otestovat jednu funkci či části upraveného kódu.

V současnosti na trhu existuje několik volně dostupných i placených produktů, které se liší svou kvalitou a možnostmi. Za zmínku stojí následující projekty.

CodeBlocks Arduino IDE

CodeBlocks Arduino IDE je upravené open-source programovací prostředí, které je rozšířeno o možnosti programování aplikací pro Arduino a jejich nahrávání na fyzické desky. IDE

navíc nabízí možnosti simulování Arduina přes API Arduino simulator. Tato funkcionalita je stále ve fázi vývoje. IDE je dostupné pro Windows i Linux (*http://arduinodev.com/codeblocks/*).



Obrázek 1.4: CodeBlocks Arduino IDE

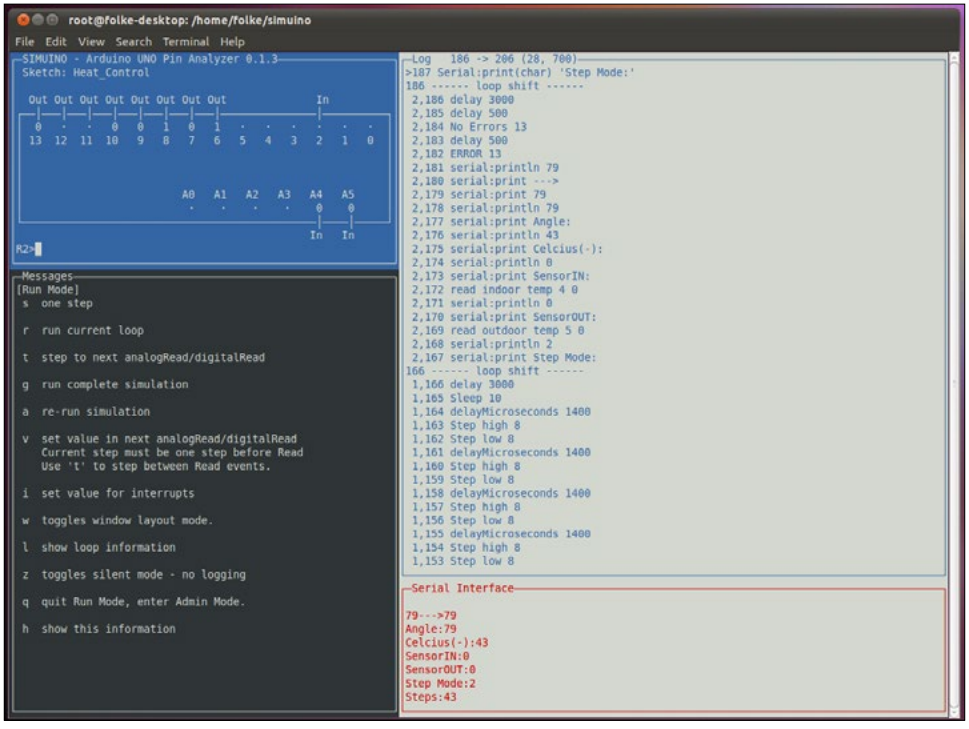
Simuino

Simuino je simulátor pinů Arduino Uno a Mega určený pro linuxové operační systémy. V současnosti je dostupná jen verze CLI, verze WEB je pozastavená (od roku 2013). Simulátor nabízí omezenou funkcionalitu pro testování kódu s funkcemi `analogRead`, `digitalRead` a `External Interrupts` (*http://web.simuino.com*).

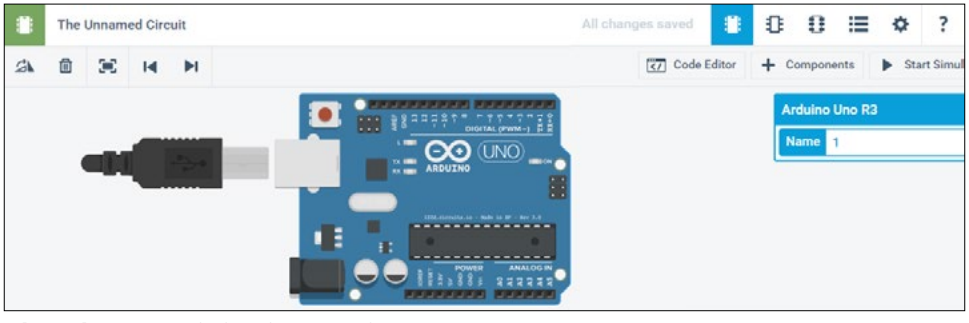
Autodesk 123D circuits

Společnost Autodesk, známá svými grafickými aplikacemi pro průmysl, zábavu či design, vytvořila projekt online aplikací určených elektrotechnickým nadšencům, kde si mohou vytvořit návrh obvodu, naprogramovat, odsimulovat, vytvořit schéma desky plošného spoje, a ten si následně mohou nechat vyrobit a zaslat domů.

Aplikace vyžaduje registraci a přihlášení, umožňuje i použití facebookového účtu. Následná tvorba schémat a programování je velmi intuitivní. Simulátor nabízí omezený počet knihoven a funkcí v porovnání s IDE. Zajímavou funkcí je debugger, který slouží k základnímu ladění a diagnostice kódu (*https://123d.circuits.io/*).



Obrázek 1.5: Simuino



Obrázek 1.6: Autodesk Arduino simulator

VBB4Arduino

VBB je placený simulátor Arduina. Simulátor umožňuje programování a diagnostiku kódu aplikací pro Arduino, k ladění problémů přes aplikaci je nutné internetové připojení, které vyhledává na internetu relevantní informace s použitím vyhledávacího API Googlu.

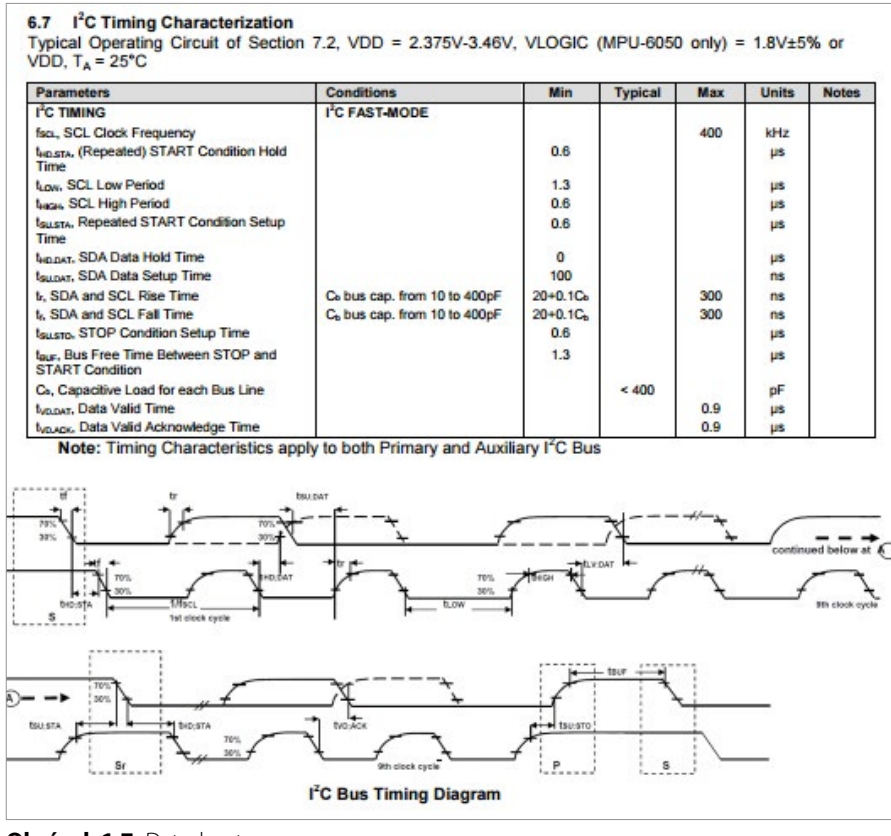
Zajímavou (v únoru 2016 připravovanou) funkcionalitou, kterou nabízí tento projekt, je nasazení zdrojového kódu Arduina na zařízení Raspberry Pi v.2 (*http://www.virtualbreadboard.com*).

Shrnutí

Simulátory a emulátory poskytují jen omezenou funkcionalitu. Jejich rozvoj není, a s velkou pravděpodobností ani nebude takový, aby překonal fyzická zařízení. Jejich použití je vhodné například při hromadných prezentacích jednoduchých aplikací či při prvotním seznámení se s technologií. Výhody hardwarové verze jednoznačně převažují nad využitím virtuálního systému.

Datasheety

Nejlepším zdrojem technických informací o produktu by měla být dokumentace. V situacích, kdy věci nejdou podle představ, se stávají návod a dokumentace nejlepšími pomocníky. U elektronických součástek a modulů se návody označují jako datasheety. Datasheet obsahuje technickou specifikaci a princip použití. Kvalita datasheetů závisí na výrobci, v některých případech obsahují datasheety téměř hotové řešení, v jiných případech je tento dokument téměř nepoužitelný a zbytečný.



6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus

I²C Bus Timing Diagram

Obrázek 1.7: Datasheet

Mezi informace, které se v datasheetech nejčastěji vyhledávají, patří:

- Označení a rozmístění pinů součástek, integrovaných obvodů a externích modulů.
- Charakteristiky popisující fungování součástky. Obvykle mají formát tabulky nebo grafu.
- Příklad zapojení či zdrojového kódu.

Na základě datsheetů se hodnotí a vybírají součástky v procesu návrhu. Během výběru součástek je potřeba zvážit i další parametry, které datasheet neobsahuje – jedná se například o cenu, skladovou dostupnost, dodací dobu a poštovné, které je často dražší než samotná součástka.

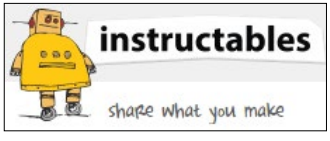
Komunity a tutoriály

Počet členů komunit Arduina se počítá na desetitisíce, což vytváří značnou databázi znalostí, zkušeností, vzdělávacích materiálů, návodů a inspirací pro projekty. Díky internetu je možné najít na diskuzních fórech řešení téměř jakéhokoliv problému souvisejícího s Arduinem či AVR mikrokontroléry. Nadšenci vytvořili několik webových portálů, kde se prezentují různé postupy, návody a hotové projekty.

Mezi velmi oblíbené portály zaměřené na Arduino patří:

Instructables (*www.Instructables.com*)

Portál založila skupina lidí pocházejících z MIT Media Lab, což je špičkové mezioborové výzkumné středisko. Projekt byl veřejně publikován v roce 2006. V současnosti nabízí množství tutoriálů a návodů, diskuzí týkajících se různých oblastí od CNC, Arduina přes Internet of Things a zdraví až po Star Wars.

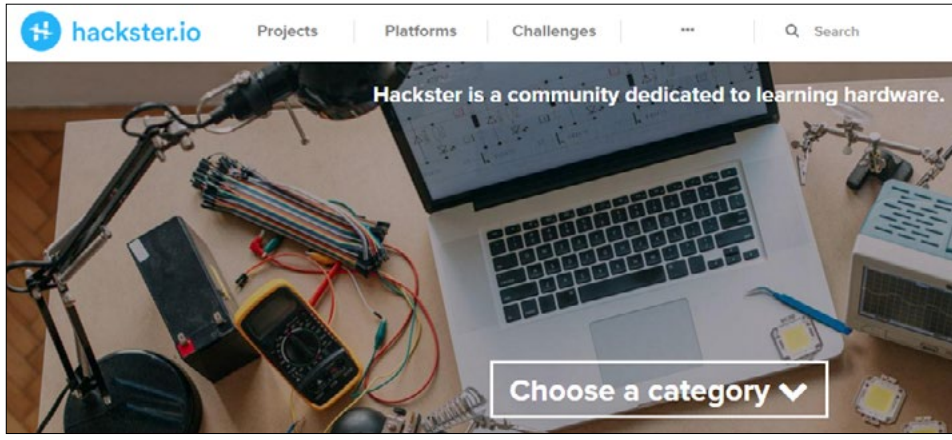


Obrázek 1.8: Portál Instructables

Jednoduché prohlížení je přístupné všem návštěvníkům stránek. Abyste mohli zasílat návody nebo stahovat ve formátu PDF, musíte se zaregistrovat. Na výběr máte dva typy registrace, bezplatnou a placenou, která nabízí rozšířené možnosti.

Hackster (https://www.hackster.io)

Komunita, která je internetově aktivní od roku 2013 s hlavním cílem vzdělávání a sdílení zkušeností, které, jak tvrdí, je důležitější než komerční cíle. Zaměřuje se na řešení reálných a velkých problémů současné doby.



Obrázek 1.9: Portál Hackster

Sociální sítě

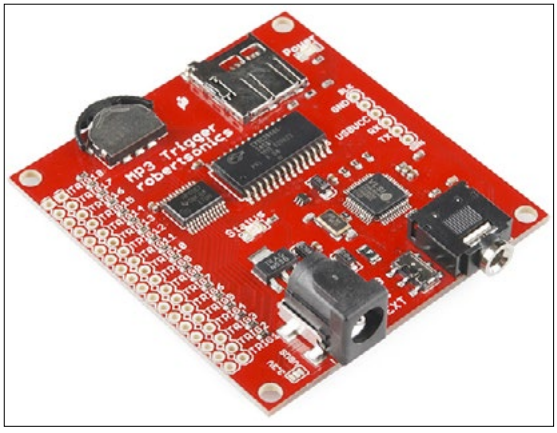
Mezi velmi aktivní patří i skupiny a komunity na sociálních sítích, jako je Twitter či Facebook:

- *https://twitter.com/arduino*
- *https://www.facebook.com/official.arduino*

Obrázek 1.10: Twitter

Sparkfun (https://www.sparkfun.com/)

Mezi portály určenými kutilům z oblasti elektroniky jsou oblíbené například i Sparkfun či Adafruit. Sparkfun je primárně online obchod zaměřený na prodej a distribuci doplňků. Na stránkách obchodu je ale možné najít hromadu zajímavých informací od návodů a tutoriálů popisujících realizaci projektů přes diskuzní fórum až po rozhovory, videa a články zaměřené na popularizaci platformy Arduino, Rasppbery Pi a souvisejících produktů vyráběných pod značkou Sparkfun, pro které je typická červená barva.



Obrázek 1.11: Hardware Sparkfun

Adafruit (https://www.adafruit.com/)

Adafruit je portál zaměřený na popularizaci a vzdělávání v oblasti amatérské elektroniky a tvorby zařízení pro oblast hobby. Portál byl založen a je řízen dámou jménem Limor „Ladyada" Fried, která se zařadila podle časopisu Glamour mezi top 35 žen, které změnily technologickou sféru byznysu.

Obrázek 1.12: Limor „Ladyada" Fried

Stack-exchange (http://arduino.stackexchange.com/)

Na Stack-Exchange získalo Arduino vlastní kategorii, kde si uživatelé vzájemně pomáhají a odpovídají na otázky a problémy, které řeší při práci s konkrétní technologií. Portál má propracovaný systém posílání otázek, hodnocení, reputační systém uživatelů (získávání bodů, medailí).

Tronixstuff (http://tronixstuff.com/tutorials/)

Portál, který obsahuje několik desítek návodů s vyčerpávajícím popisem.

educ8s.tv (https://www.youtube.com/user/educ8s/videos)

Educ8s.tv je kanál na YouTube s téměř stovkou krátkých pětiminutových tutoriálů. Videa pokrývají návody, jak propojit Arduino a externí prvky, jako jsou breadboard, snímače, displeje či jiné hotové elektronické desky, které rozšiřují Arduino o novou funkcionalitu.

EngineersGarage (http://www.engineersgarage.com/microcontroller/arduino-projects)

Dobrým zdrojem inspirace pro hobby či školní projekty je blog EngineersGarage. Stránka je sbírkou projektů pro Arduino, Raspberry Pi a mikrontroléry AVR, PIC či 8051.

ArduinoBasics (http://arduinobasics.blogspot.com.au/p/arduino-basics-projects-page.html)

Stránka s obsahem základních tutoriálů, jak pracovat a Arduinem:

HowToMechatronics (http://howtomechatronics.com/tutorials/arduino/)

Stránka s množstvím návodů a inspirací pro projekty.

JeremyBlum (http://www.jeremyblum.com/category/arduino-tutorials/)

Osobní blog autora knihy Exploring Arduino. Na blogu jsou kromě popisu zajímavých projektů prezentovány i zajímavosti a novinky z Arduino akcí a konferencí.

Lidé (http://playground.arduino.cc/Main/People)

Na stránkách *Arduino.cc* je uveden seznam lidí, kteří se podílejí na projektu Arduina či udržování aktivní Arduino komunity. Většina z těchto lidí má svůj osobní blog, kde se dají najít zajímavé informace, inspirace či návody související s Arduinem.

Účast v komunitách

Komunita se podílí i na vývoji nových knihoven či testování a nasazování nových funkcí. Pro účast na vývoji hardwaru či softwaru stačí poslat e-mail na odpovídající e-mailové adresy s konkrétní žádostí a popisem navrhované spolupráce. Jestliže se jedná o populární projekt, může odpověď přijít kvůli vytížení až později s odstupem času.

Chcete-li se zařadit do komunity zabývající se vývojem hardwaru Arduina, použijte e-mailovou adresu *distribution@arduino.cc*. Komunita, která se zabývá vývojem softwarové stránky projektu Arduino, komunikuje prostřednictvím takzvaného mailing listu, který je dostupný na adrese *https://groups.google.com/a/arduino.cc/forum/?fromgroups#!forum/developers*.

Začlenění do komunity provedete odesláním zprávy do této skupiny.

Komunikace

Základem fungování člověka v dnešním světě je komunikace. Abyste se úspěšně začlenili do komunity a získali reputaci, je nutné kromě jiného vědět, jak správně komunikovat. Pro začátečníky jsou výše popsané komunity často jedním (snad ne jediným) ze základních zdrojů informací. Pro pokročilejší je tu možnosti aktivní účasti na projektech.

Každý nějak začínal a minimálně ze začátku potřeboval radu od starších a zkušenějších. Při snaze o získání potřebných znalostí je nezbytné pokládat správné otázky. Dobré otázky posouvají kupředu obě strany. Nováček získá odpověď od zkušenějšího, která mu pomůže při řešení problému. Zkušenější, který odpovídá, si trénuje schopnosti vysvětlování, prezentace, dostává možnost ujasnit si souvislosti, získat nový pohled na problém, jak ho vidí jiní, nebo se naučit něco nového, co souvisí s diskutovanou oblastí. Učením druhých se dá zvyšovat vlastní odborná a komunikační úroveň.

Podnětné otázky

Zkušení lidé mají rádi náročné problémy a podnětné otázky. Tato kombinace je stimuluje a jsou vděční za takovouto možnost. Základním aspektem podnětných otázek je být technicky kompetentní. To znamená, že před samotným položením otázky je potřeba provést vlastní průzkum možností řešení, získat potřebné podklady a snažit se jim porozumět vlastní snahou. Otázka by měla směřovat na informace a znalosti, které z daného materiálu či návodu a procesu jeho realizace nevyplývají.

Hledání odpovědi

Při hledání odpovědi na vzniklý problém je nanejvýš vhodné ověřit, zda už někdo podobnou situaci řešil, k jakým závěrům dospěl a co mu pomohlo. Proto je vhodné před zapojením komunity do řešení:

- Hledat informace v diskuzních fórech a e-mailových konferencích.
- Hledat informace na webových stránkách, manuálech a FAQ.
- Vyzkoušet vlastní implementaci a experimentovat.
- Zeptat se známých či kolegů, kteří se zabývají podobnou oblastí.

Pokud jste stále nenašli správnou a dostatečně vysvětlující odpověď, nastal vhodný čas obrátit se na komunity. Často se totiž stává, že se kladené otázky opakují a jsou vysvětleny v manuálu či FAQ, což způsobuje rozhořčení na jedné straně a frustraci na straně druhé.

Jak (ne)klást otázky

- *Tematizace otázek* – při kladení otázek je potřeba zvolit adekvátní úroveň a odpovídající téma. Diskuzní fóra bývají tematizovaná, v každém z nich se schází otázky ze stejných oblastí.
- *Délka a podrobnost* – pokročilá témata vyžadují sofistikovanější otázky s popisem detailů, použitých postupů, vlastních zjištění. Je ale potřeba zvolit vhodný rozsah, protože příliš dlouhé texty nebude nikdo číst.

Poznámka: Na zahraničních diskuzních fórech panuje velká obliba používání anglických zkratek typu IMHO, AFAIK, TL;DR; a podobně. Právě TL;DR; se objevuje u příliš dlouhých příspěvků, které nejsou tak zajímavé, aby udržely pozornost čtenáře, a stávají se předmětem rychlého prolistování. Proto je vhodné psát otázky krátce a výstižně. To se ale snáze řekne, než realizuje.

- IMHO – in my humble opinion – Podle mého skromného názoru
- AFAIK – as far as i know – Pokud vím
- TL;DR; – too long; didn't read – Příliš dlouhé, nečetl jsem

Hacker

V kontextu výše napsaného je Arduino velmi vhodná platforma pro hackery. Pojem hacker pochází z dob vzniku internetu a vztahoval se na komunitu lidí, která pracovala primárně se softwarem.

V současnosti se slovo hacker stalo módním a často dochází k jeho nesprávné interpretaci. Za hackery bývají považováni mladí počítačoví fanatici, kteří se zabývají nabouráváním do systémů, lámáním hesel či flashováním telefonů. Tito jsou v komunitě označováni jako crackeři.

„*Zásadním rozdílem mezi hackerem a crackerem je, že hacker věci buduje a cracker věci rozbíjí.*"

Pro hackera jako takového je důležitý postoj (v angličtině označovaný jako *hacker mind-set*). Jde o snahu řešit problémy, důvěru ve svobodu a vzájemnou pomoc komunity. Hackerem se člověk nestane samozvaně, člověka nazve hackerem komunita. Hacker řeší problémy a vytváří nové věci, jeho výsledky jsou přínosem pro komunitu.

Základní předpoklady

Svět je plný zajímavých problémů, které čekají na vyřešení. Nemusí se jednat o řešení, která spasí svět. Řešení běžných problémů pomáhají změnit život obyčejných lidí tak, aby byl lepší a jednodušší. Můžeme se potom zaměřit na důležité věci. Je důležité vědět, že v současnosti není možné a ani nutné vědět vše. Problém je potřeba rozdělit na menší části a získávat jen ty poznatky, které jsou relevantní k řešení jednotlivých částí hlavního problému. Při řešení je důležité zaměřit se na výsledek a jeho užitek, ne na peníze či společenské uznání.

Není nutné znovu objevovat kolo. Mozek je úžasné „zařízení" a efektivní využívání jeho kapacity vyžaduje úsilí a čas. Ty jsou ale omezené a vzácné. Proto je potřeba omezit plýtvání a stavět na řešení jiných, sdílet vlastní řešení, která umožní jiným pokračovat.

Zároveň to ale neznamená, že na všechno existuje jediné správné řešení. Nový pohled na problém získáme i analýzou prvního řešení, které se ukáže jako neefektivní či nesprávné. Sdílením a analýzou řešení se posouváme vpřed a neděláme znovu chyby prvních řešení či pokusů. Je známo, že napoprvé se nikdy nic nepodaří udělat dokonalé a efektivní.

Experti z oblasti programování tento princip aplikují používáním existujícího kódu (dostupného na internetu či vlastního), který upravují novým podmínkám a požadavkům. Jeden student se během cvičení prohrabával v odpadkovém koši, kde hledal na papíru řešení, které

vyhodili na minulých cvičeních. Tato řešení následně upravil svým potřebám a čas strávený na řešení minimalizoval na čtvrtinu. Právě z těchto důvodů je důležité kromě psaní kódu ho umět i číst. I v běžném životě člověk více textu čte, než píše.

Je důležité zdůraznit, že se nejedná o kopírování, ale půjčování hlavní myšlenky, která se následně upraví a vylepší. Inovace (vylepšení) je vždy jednodušší, rychlejší a levnější než vynález (vymyšlení celého konceptu).

V IT firmách je psaní kódu určeno juniorským a středně pokročilým pozicím. S rostoucí úrovní a odborností se člověk přestává věnovat psaní kódu a začíná ho hodnotit, revidovat a kritizovat. Na seniorských pozicích se lidé díky své odbornosti a zkušenosti věnují čtení kódu, který vytvořili pracovníci na nižších pozicích. Revidovaný kód často vracejí k přepracování s konkrétními připomínkami.

Nudná a opakující se práce zabíjí kreativitu. Některé činnosti vyžadují opakování k získání nutných znalostí a praxe. Tyto činnosti ale (většinou) nespadají do oblasti opakujících se a nudných činností. Časté opakování by mělo být spojené s výzvou, zkoušením a objevováním možností. Například hraní šachů se dá naučit jen častým opakováním. Hrát šachy se ale člověk nenaučí opakováním stejné posloupnosti kroků. Podobně je to i s programováním či jazykem.

Za extrémně nudné, opakující se a doslova ubíjející činnosti se považuje například otrocké přepisování čísel z faktur do tabulek, manuální porovnávání tabulek a podobně. Takováto práce v dlouhodobém horizontu doslova ubíjí schopnost přemýšlet a tvořit.

Svoboda v přemýšlení je cesta, která umožňuje tvořit. Také proto mají hackeři problém s autoritou a přikazovaným způsobem řešení. Je potřeba rozlišovat autoritu, která omezuje fungování ve společnosti a státu, a manažera či vedoucího, který ví všechno nejlépe. V popisovaném kontextu jde o osoby, manažery, nadřízené či zkušenější kolegy, kteří je omezují v tom, jak přistupovat k řešení. V souvislosti se svobodou je nezbytný úsudek a kritické myšlení, které pomáhá při hodnocení a směřování, jak a kam dále.

Přístup nenahradí schopnosti. K získání schopnosti řešit problémy správně je nutný přístup, který sám o sobě nestačí. Je důležité znát dobře problematiku. Získání moudrosti (anglicky wisdom) vyžaduje inteligenci, zkušenosti, věnování se problematice a tvrdou práci. Jen časem může člověk nabýt schopnost, která se označuje jako moudrost či mistrovství v určitém oboru. Proto by měly být kompetence oceňovány, protože jejich získání vyžadovalo hodně úsilí.

Efektivnost práce. Nejdůležitějším nástrojem se v dnešní informační době stal mozek. Nezbytná je zdravá výživa a dostatek spánku. Kvalitní spánek je číslem jedna v doporučeních odborníků a expertů.

Efektivita zabezpečuje i rozumné využívání mozkové kapacity. Není nutné snažit se vstřebat a pochopit všechny detaily. Postačující je základní princip fungování a schopnost jeho aplikace v praxi. Velmi dobrým zdrojem takovýchto informací jsou zkušenější kolegové. Sdílení informací, co není správné a co nefunguje, umožňuje dělat pokroky rychleji a neučit se na vlastních chybách, pokud je už dříve udělali jiní.

Svým způsobem nejsou tyto principy ničím převratným. Často se ale ignorují nebo se na ně zapomíná.

Potřebné znalosti

Do základní báze znalostí, kterými by měl disponovat každý hacker v oblasti informačních technologií, patří:

- Schopnost programovat
- Rozumět operačním systémům
- Rozumět webu
- Umět anglicky

Úroveň nebo expertiza konkrétních oblastí záleží na tom, ve které oblasti člověk pracuje. Mít přehled a minimální znalosti i z jiných oblastí je nanejvýš vhodné. Držet krok s posledními trendy je ale značně náročné – existují desítky operačních systémů, programovacích jazyků a frameworků. Vždy se ale jedná o stejný základ, na kterém se dále staví.

Programování

Nezáleží na tom, zda je to člověk zabývající se hardwarem nebo softwarem. Všude je logika ukryta v softwaru, a proto je důležité, aby hacker rozuměl tomu, jak se software tvoří, a byl schopen ho číst a sám tvořit. Existuje několik zajímavých zdrojů, jak se naučit programovat. Samotné programování je ale jako jazyk. Je potřeba ho používat a trénovat, jinak ho člověk nedokáže efektivně využívat. Mezi doporučené jazyky patří C/C++, Java, Perl, Lisp. Bez ohledu na to, v jakém jazyku bude tvořený kód, naučí tyto jazyky různé programátorské přístupy, jejichž znalost je užitečná.

Operační systémy

Velmi dobrým způsobem, jak se naučit rozumět operačním systémům, je účast na open-source projektech. Nezáleží na tom, jestli jde o linuxové systémy, Windows či řídicí systém embedded zařízení. Open-source kód a komunita jsou velmi nápomocné při analýze a studiu logiky a principů fungování systémů.

Webové technologie

Web je trendem současnosti. Být online, prezentovat produkty, informace, služby, využívat aplikace. Všechno běží na webových serverech. Rozumět struktuře a tvorbě backendu a frontendu, schopnost vytvořit webovou stránku a kvalitní obsah patří mezi žádané kompetence. Velká část webových řešení je v současnosti postavena na frameworcích a propojování různých knihoven.

Anglický jazyk

Většina komunit, diskuzních fór, literatury, manuálů a tutoriálů je psána a prezentována v angličtině. Proto nezbývá nic jiného, než se anglicky naučit. V případě, že se chce člověk aktivně účastnit, musí ovládat jazyk na aktivní úrovni.

Tip: Velmi dobrým portálem pro získávání znalostí všemožného druhu od lidí z celého světa je web Hacker News. V současnosti existuje několik projektů, které mají podobné zaměření. Mezi obsahově nejkvalitnější a nejhodnotnější patří portály:

- *https://news.ycombinator.com*
- *https://www.quora.com*

Diskuze na těchto portálech mají skutečnou úroveň a informační hodnotu, která dokáže poskytnout zajímavý pohled na svět či diskutovanou oblast. Diskuzní fórum má často vyšší informační hodnotu než původní článek, ze kterého diskuze vychází. (Diskuze nejsou součástí portálu, kde byl diskutovaný článek zveřejněn.)

Shrnutí

Jak jste mohli vidět, zvládnutí všech popisovaných činností a oblastí vyžaduje čas a úsilí. I proto vytvářejí kompetence, které nejsou snadno nahraditelné na trhu práce. V mnoha oblastech se dlouhodobým věnováním jedné problematice vytvářejí mentální modely, které se stávají mlčky činěnými znalostmi, jež se nedají získat jinak než praktickými zkušenostmi. I proto je není možné získat přečtením jedné knihy, odehráním jedné šachové partie či napsáním jedné aplikace.

Hardware hacking a reverzní inženýrství

Hardware hacking není tak rozšířený v povědomí veřejnosti jako hacking softwarový, který se často objevuje v médiích ve smyslu útoků na síťovou infrastrukturu, či v kontextu softwarové kriminality, kdy mladí chlapci v kapucích v tmavé místnosti crackují licenční klíče ke kancelářskému softwaru či nejnovější hře. Hacking hardwaru je činnost, která spadá do jedné z následujících kategorií:

- Oprava pokaženého zařízení (cokoliv pokaženého, pro co už nejsou dostupné náhradní díly či servisní podpora).
- Úprava zařízení o nové vlastnosti, které nebyly součástí originálního designu produktu, či přizpůsobení vlastním požadavkům (připojení zařízení k internetu třeba v následujících případech: starý termostat, rychlovarná konvice).
- Zvyšování výkonu (upgrade počítačů, automobilů).
- Prolomení ochranného mechanizmu za účelem přístupu ke všem funkcím nabízeným systémem na úrovni superuživatele (například rootování telefonu či flashování firmwaru).

Reverzní inženýrství je proces získávání znalostí o designu a funkcionalitě zařízení. V případě komerční praxe se proces často využívá za účelem reprodukce zařízení či odstranění ochran proti kopírování. Reverzní inženýrství je technika použitelná i při analýze malwaru a jiného škodlivého kódu, kde lze na základě extrahovaných znalostí vytvořit signaturu pro identifikaci či postup odstranění škodlivého kódu.

Většina velkých korporací či vládních agentur (nejznámější svými praktikami jsou NSA, CIA, FBI) má skupiny expertů, kteří se zabývají komparativní analýzou (benchmarking) a reverzním inženýrstvím konkurenčních produktů. Samozřejmostí je pak popírání těchto praktik.

Poznámka: Téměř každý komerční produkt s sebou nese licenční a záruční podmínky, které nedovolují jeho otevírání, úpravu, a dokonce ani aplikování principů reverzního inženýrství. Uživatel zakoupením, instalací a používáním produktů akceptuje uvedené licenční podmínky.

Sběr informací

Cenným zdrojem informací o zařízeních jsou manuály, internetová diskuzní fóra či videa na YouTube. Velkou neznámou při analýze a úpravě zařízení jsou diagramy a schémata zapojení, která většinou výrobce dává k dispozici jen servisním organizacím. Každý výrobce se snaží konstrukční detaily o svých produktech chránit a udržovat v tajnosti, což mu pomáhá zajistit (dočasnou) konkurenční výhodu na trhu. V současnosti ale žádné zařízení nezůstane chráněné navždy.

Díky internetu je možné po chvíli vyhledávání najít minimálně základní schémata, která budou pro dané zařízení aplikovatelná. Produktové řady se odlišují jen minimálně ve velikostních a výkonnostních parametrech. Každá pračka má ale stejnou základní strukturu a funkcionalitu. Podobné je to i s auty, ledničkami, počítači a jinou elektronikou.

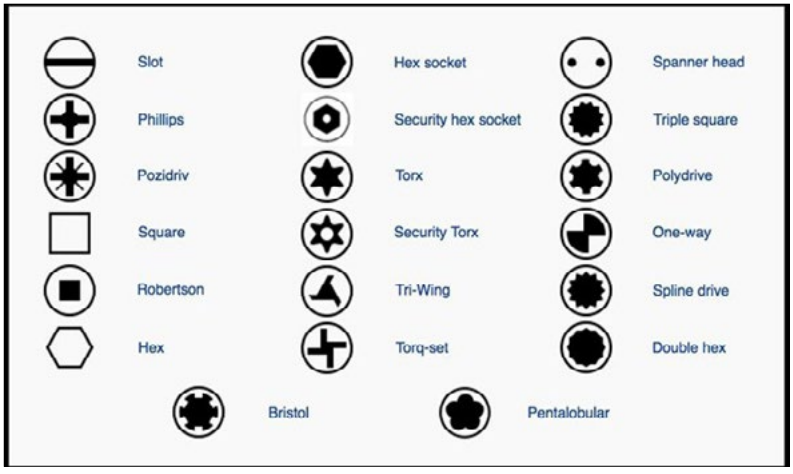
Fyzická inspekce

Analýza prvků, jako je kvalita konstrukce, fyzický design, přístupnost k jednotlivým prvkům a částem, chlazení či umístění jednotlivých komponent, může poskytnout zajímavé informace o kvalitě produktu a vytvořit obraz o společnosti a vývojovém týmu, který na produktu pracoval. U nedomyšlených či až odfláknutých prvků se dá vyvodit, že podobné nedostatky budou i jinde, jejich objevení může poskytnout cestu k požadované úpravě či průniku do systému.

Odkrytování zařízení

Odkrytování zařízení může v současnosti představovat velkou výzvu. Výrobci s oblibou používají šroubky s netradičními tvary, umisťují je na nedostupná místa nebo používají netradiční principy plastových západek, které se při nesprávné manipulaci snadno odlamují. Tato opatření mají zabránit domácím opravám a snahám o vlastní vylepšení.

Téměř nezbytnou prerekvizitou hobby hackera jsou nástroje nezbytné k rozebírání a snímání krytů. Je potřeba mít k dispozici velkou sadu nástrojů, obzvlášť šroubováků nejrůznějších tvarů a velikostí.



Obrázek 1.13: Typy hlav šroubků

Firmy se při použití spoléhají na nedostupnost takovýchto specializovaných šroubováků, případně na jejich cenu, která je příliš vysoká, než aby se šroubovák vyplatilo pořizovat kvůli třem šroubkům.

Ovládací mechanizmus

O systému se můžete mnohé dovědět pozorováním, jak funguje. Tento princip testování (poznávání) systému se označuje i jako black-box testing. Podrobnější informace k principům testování hardwaru jsou popsány v části „Testování hardwaru". Základním zdrojem informací jsou vstupně-výstupní rozhraní, která mají u komerčních produktů podobu ovládacího mechanizmu.

Většina zařízení má ovládací mechanizmus, který uživateli umožňuje zařízení ovládat. Mechanizmus může být jednoduché tlačítko (například v případě rychlovarné konvice), či složitější ovládací panel (v případě pračky). Ovládací mechanizmus je prvním vhodným místem, kde lze provádět hackovací pokusy.

Prvotní pokusy mohou být například stisknutí tlačítek a sledování, jak systém reaguje. V případě, že má systém displej, je vhodné otestovat, jak se chová, když tlačítko zůstane stisknuté dlouhou dobu, nebo jak reaguje na stisk více tlačítek dohromady. Takovýmto způsobem se dají otestovat multitaskingové systémy, schopnost watchdogu ošetřit mezní stavy, nebo zjistit případné chyby a nedostatky v softwaru. V případě chybného ošetření chybových stavů nemusí dojít k detekci, že se například klávesnice nebo displej dostaly do chybového stavu, ale základní funkcionalita zařízení zůstala nepoškozena.

Další zajímavou možností, jak hacknout zařízení, je napojení Arduina na spínací mechanizmus, který řídí zapnutí/vypnutí zařízení, umožňuje téměř jakékoliv zařízení připojit k internetu a jeho následné vzdálené ovládání, což je jeden z velmi oblíbených požadavků.

Identifikace součástek

Výrobci své součástky označují logem, namísto dlouhými nápisy. Použití loga občas komplikuje určení výrobce. Při identifikaci je možné použít stránku „How to wiki", kde se nachází databáze grafických symbolů, jež používali či používají jednotliví výrobci elektronických komponent (*http://how-to.wikia.com/wiki/Howto_identify_integrated_circuit_(chip)_manufacturers_by_their_logos/all_logos*).

Po identifikaci výrobce je trochu snazší najít konkrétní součástku a datasheet s popisem funkcionality, funkčních parametrů a schémat zapojení. Další z možností, která pomáhá zúžit kruh vyhledávaných součástek, je určení napájecích pinů – UCC a GND.

Přestože výrobci čipů a integrovaných obvodů vyrábějí speciální sady produktů s rozšířenou funkcionalitou, které jsou určeny pro výrobce elektronických zařízení, vždy se snaží minimalizovat náklady na vývoj a výrobu. To znamená, že „korporátní" součástky se příliš neodlišují od těch běžně dostupných pro veřejnost či malé výrobce. Takže datasheety k „veřejným" jsou ve většině případů použitelné i pro „korporátní".

Výměna komponent

Dalším způsobem úpravy a vylepšení zařízení je náhrada komponent či funkčních dílů. Výměna motorku v autíčku na dálkové ovládání je jednoduchá. Vyjmout mikrokontrolér nebo flash paměť z desky už není jednoduchá záležitost a vyžaduje specifické zařízení a znalosti.

Debuggování

Elektronická zařízení mají obvykle k dispozici debuggovací porty. Mezi nejzákladnější patří JTAG, BDM, ISP, SPI, I2C, CAN či NAND flash. Na trhu je k dispozici několik aplikací, které umožňují diagnostikovat data zachycená z portů.

Každé zařízení se vyrábí ve fázích, kdy se nejprve vyrobí hardware a následně se nahrává řídicí kód. Na oddělení, kde se řeší nahrávání kódu, přichází zařízení v základním stavu, je funkční a je možné ho zapnout, přičemž s programátorem realizuje základní komunikaci. Tuto funkcionalitu obvykle ošetřuje bootloader.

Výrobci v poslední době často chrání svoje zařízení a elektroniku pomocí speciálních pouzder (například BGA – ball grid array). V případě vícevrstvé desky je taktéž diagnostika obvodu značně komplikovanější.

Jedním z možných (ne však jednoduchých) přístupů je analýza pamětí EEPROM, Flash. V tomto případě je ale často nutné oddělení paměti od zbytku obvodu, což je značně komplikovaná operace vzhledem k používaným technologiím automatizovaného osazování plošných spojů.

Tvorba softwaru

Software řídí hardware a zabezpečuje komunikaci s uživatelem. Jednou z možností, jak diagnostikovat a odhalit funkcionalitu jednotlivých částí zařízení, je pomocí softwaru a jeho testování. Obzvlášť zajímavé jsou možnosti testování okrajových podmínek, které v případě chybného ošetření, respektive neošetření (ignorování), vedou k chybě. Tyto chyby se často zneužívají v zařízeních a systémech k prolomení různých ochran a získání kontroly nad zařízením (známé jako security vulnerability).

Výrobci často nabízejí možnost upgradovat firmware. Tím ukazují cestu, kterou je možné zařízení diagnostikovat a upravit pro svoje potřeby. Zabezpečení procesu upgradu firmwaru proti reverzním inženýrství u běžných komerčních zařízení, jako jsou domácí směrovače, je často minimální, pokud je vůbec nějaké implementováno.

Základní informace o funkcionalitě a logice zařízení se dají získat testováním funkcionality rozdělené do následujících částí:

- Zpracování vstupů a výstupů
- Logika hlavního programu
- Interakce s uživatelem
- Možnosti nastavení
- Zabezpečení

Psaní poznámek

Když člověk tráví hodiny a hodiny soustředěním na určitou úlohu, je vhodné dělat si poznámky průběžně během práce. Na závěr by se mohlo zapomenout na důležité detaily, které jsou kritické pro následné opakování pokusů či odhalování vzniklých chyb. Takto je možné navázat na předchozí práci o několik dní později.

Shrnutí

Proces reverzního inženýrství a hardware hackingu obnáší následující kroky:

- Zjistit co nejvíce informací o zařízení z internetu či manuálu.
- Získat katalogové a servisní informace k zařízení či produktové řadě.
- Otestovat vstupy a výstupy, různé mezní stavy.
- Rozebrat zařízení a získat informace o součástkách.
- Identifikovat schematické propojení a odhadnout funkcionalitu jednotlivých bloků a součástek.
- Pokusit se zjistit, co dělá co (hardwarová část).
- Exportovat řídicí kód a analyzovat jeho funkcionalitu.
- Pokusit se zjistit, co dělá co (softwarová část).
- Navrhnout možnosti úpravy.

- Ověřit prvotní předpoklady a domněnky pokusem a možnými úpravami.
- Pokud to zařízení „přežilo“ a stále funguje, užívat si nové funkcionality.
- Všechno zdokumentovat, a je-li to možné, se se svým úspěchem při objevování podělit. Na internetu je k dispozici mnoho diskuzních skupin a komunit, kde je k tomu více než vhodný prostor.

Arduino a reverzní inženýrství

Jednou z praktik reverzního inženýrství je extrakce programového kódu mikroprocesoru a jeho následná analýza. Jelikož Arduino využívá procesory typu ATmega, je extrakce a analýza kódu relativně jednoduchá pomocí nástroje AVRdude, který je k dispozici pro operační systémy Windows, Linux a OS X.

Nástroj AVRdude je součástí balíku AVRToolChain a je dostupný zdarma. Aktuální balíky jsou k dispozici na stránkách výrobce ATMEL (*http://www.atmel.com/tools/ATMELAVRTOOLCHAINFORWINDOWS.aspx*).

AVRToolChain je balík obsahující nástroje:

Compiler	Kompilátor zdrojového kódu C a C++
Assembler	Soubor nástrojů GNU Binutilus na práci se zdrojovým kódem. Podrobnější informace najdete v manuálových stránkách jednotlivých nástrojů: avr32-as, avr32-ld, avr32-ar, avr32-ar, avr32-ranlib, avr32-objcopy, avr32-size, avr32-nm, avr32-strings, avr32-strip, avr32-readelf, avr32-addr2line, avr32-c++filt, avr32-gdb.
Linker	
Archivátor	
Konvertor souborů	Konvertor zdrojového kódu jazyka C/C++ do formátu HEX, který lze následně nahrát do paměti mikroprocesoru.
Knihovny jazyka C	Soubor knihoven, které poskytují funkce jazyka C pro embedded zařízení.
Jiné nástroje	Mezi jinými nástroji je možné najít například debugger či simulátor.

Poznámka: Před použitím AVRToolChain je vhodné zkontrolovat dokumentaci k balíku, kde je uveden seznam podporovaných verzí mikrokontroléru.

Extrakce zdrojového kódu z Arduina

Export řídicího kódu Arduina pomocí nástroje AVRdude provedete následovně:

1. Připojte desku Arduina
2. V konzolovém okně příkazového řádku spusťte nástroj AVRdude s parametry, které specifikují typ mikroprocesoru (`pm168`), typ použitého programátoru a komunikačního protokolu (`cstk500v1`), port, na který je zařízení připojeno (*/dev/ttyUSB0*), komunikační rychlost nebo baud rate (`b19200`) či typ paměťové operace (`flash:r:program.bin:r`):

```
avrdude -F -v -pm168 -cstk500v1 -P/dev/ttyUSB0 -b19200 -D -U
  flash:r:program.bin:r
```

Podrobnější vysvětlení k jednotlivým parametrům najdete v dokumentaci nástroje (*http://www.nongnu.org/avrdude/user-manual/avrdude_4.html*), případně na linuxovém systému v manuálové stránce (*http://linux.die.net/man/1/avrdude*). Výstupem procesu je soubor HEX.

Konverze souboru HEX do původního kódu

Vzhledem k tomu, že soubor HEX je vhodný ke strojovému zpracování, jeho formát není čitelný pro člověka. Na trhu existuje několik nástrojů, které umožňují dekompilaci, žádný ale nedokáže zpětně převést soubor HEX do původního formátu vytvořeného lidským programátorem.

Potenciálním řešením je konverze souboru HEX do assembleru přes nástroj avr-objdump, kde výstupní kód nabízí low-level obraz o funkcionalitě programu. To znamená, že názvy, proměnné a komentáře nebudou k dispozici a zjistíte jen informace o běhu programu na nejnižší strojové úrovni.

Příklad použití nástroje avr-objdump:

```
avr-objdump -s -m atmega328 program.hex > program.dump
```

Dokumentaci k nástroji najdete na *http://linux.die.net/man/1/avr-objdump*.

Vývoj řešení – technický pohled

Vývoj embedded zařízení je jako sestavování dílků skládačky. Dají se spojit různým způsobem, výsledek ale nemusí být přesně podle původních představ. Puzzle lze spojit jen jedním způsobem podle výřezů. Podobně jako do sebe zapadají výřezy, i proces vývoje vyžaduje posloupnost, která do sebe zapadá. Od konceptu přes návrh a vytvoření prototypu, přes ladění a testování až po uvedení do provozu. Jednotlivé kroky lze provést v jiném pořadí nebo je lze dokonce vynechat, výsledek ale nemusí přesně odpovídat původní představě.

Poznámka: Proč je užitečné věnovat se studiu metodologií, návodů a procesních postupů? Informace získané studiem zkracují čas potřebný k vývoji. V materiálech se často popisuje, jak dopadly pokusy a experimenty s jednotlivými metodami či procesními posloupnostmi v minulosti. Z chyb minulosti se dá poučit a ušetřit si tak nemálo problémů.

Vývojový proces se skládá z následujících kroků:

1. Sběr požadavků
2. Návrh architektury (formou blokového diagramu)
3. Realizace fyzického zapojení
4. Psaní řídicího kódu
5. Testování, ladění, vylepšování
6. Opakování cyklu

1. Sběr požadavků

Prvním krokem je ujasnění představy o tom, na čem se bude pracovat. Sběr požadavků se provádí před samotnou realizací a slouží k jasnému definování výsledku – čeho se má dosáhnout na konci. Cíl by měl být co nejkonkrétnější a měl by odpovídat na otázky typu:

- Jaký problém má být vyřešen
- Kdo je cílovým uživatelem řešení
- Jaké jsou požadavky na konečné řešení:
- Cena (náklady, prodejní cena)
- Design (vzhled)
- Funkcionalita
- Vlastnosti
- Parametry
- Stabilita
- Robustnost
- Bezpečnost

Položením nesprávných otázek může celý vývoj dopadnout i následujícím způsobem:

Popis zákazníka

Návrh projektového manažera

Představa designéra

Prototyp

Testování

Cena

Výsledek

Skutečná potřeba

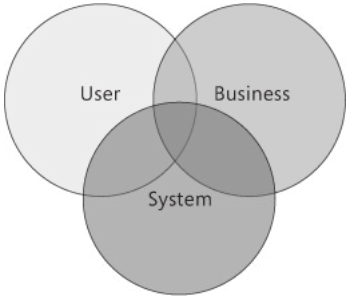
Zdroj: *http://projectcartoon.com/create/*

Součástí procesu sběru požadavků je průzkum a získávání informací o problematice, aktuálně dostupných řešeních a zjištění jejich silných a slabých stránek. Tuto fázi není dobré podceňovat, protože stabilní a kvalitní řešení může být postaveno jen na stabilních základech. Zároveň by se tato část neměla přeceňovat a věnovat jí příliš mnoho času. Na začátku se získávají informace rychle a jejich informační hodnota je velká. Postupem času ale množství nových informací a jejich informační hodnota klesá. Spotřebovaný čas však zůstává stejný.

2. Architektura systému

Architektura systému představuje tvořivou činnost, která definuje funkcionalitu systému. Bez ohledu na to, zda jde o stavební, softwarovou či hardwarovou architekturu, vždy je výsledek uměleckým dílem, které přináší kreativní řešení. Nová řešení a přístupy vycházejí z minulosti a staví na řešeních a poznatcích ověřených praxí.

Architektura musí splňovat požadavky kladené ze strany uživatele (user), obchodní stránky (business) či výrobních kapacit a omezení stanovených zákonem či normami (system). Správné řešení se nachází v průniku všech tří oblastí.



Obrázek 1.14: Požadavky na architekturu

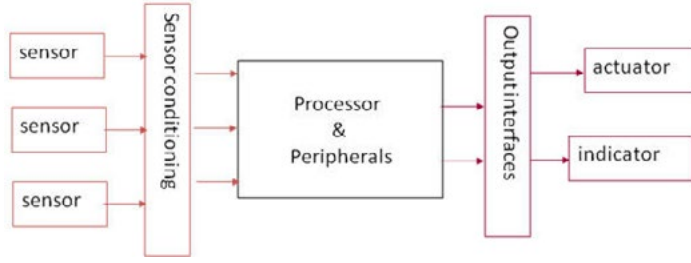
Architektura tedy vytváří most mezi třemi zájmovými skupinami. V případě, že chybí soulad s požadavky v některé z výše uvedených částí, dochází k nespokojenosti některé ze zúčastněných stran, a tím pádem i ke zhoršení vyhlídek na úspěšnost celého projektu. Z praktických zkušeností se zjistilo, že pro pocit spokojenosti s vytvořeným řešením musí být splněno aspoň 80 % vznesených požadavků každé zájmové skupiny.

Tip: Kvalitní literaturu k problematice softwarové a embedded architektury je možné najít v knihách vydaných pod hlavičkou vydavatelství O'Reilly Media a Microsoft Press.

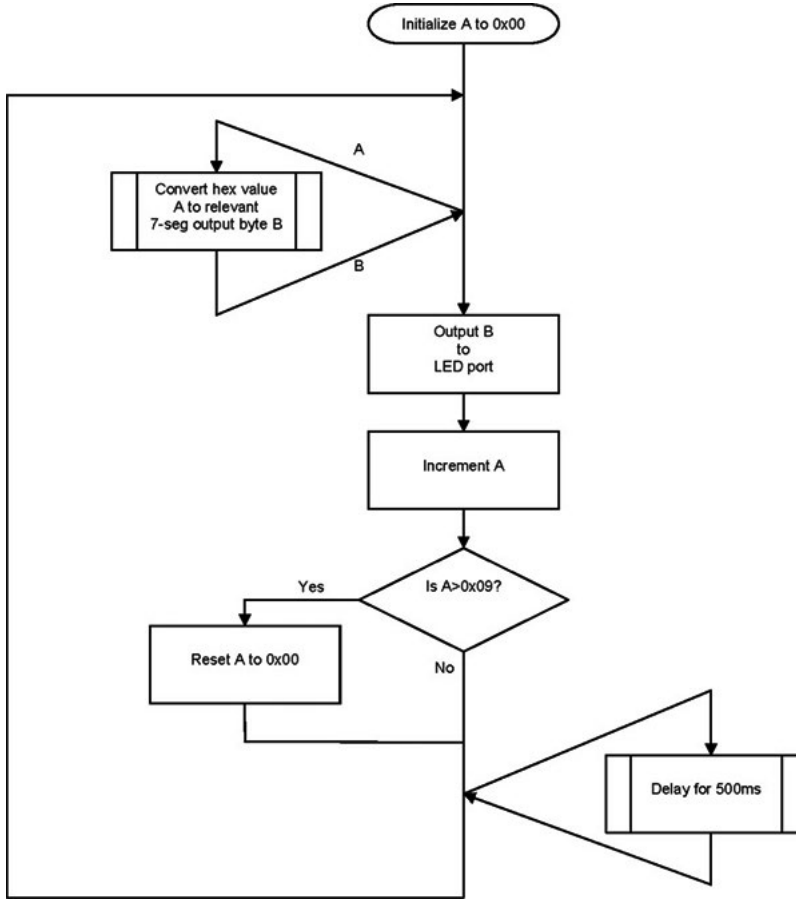
3. Blokový diagram

Blokový diagram je jeden z nástrojů pomáhajících systém dekomponovat na menší části s nákresem vazeb mezi prvky. Z nákresu je vidět, jak spolu které části komunikují. Blokový diagram je prvním náznakem objektově orientovaného přístupu při řešení. Bloky jsou abstraktní objekty, které ukrývají konkrétní funkcionalitu a způsoby její implementace. Do diagramu může být zakresleno rozhraní, přes které blok komunikuje s okolím.

Do bloků se následně dají dopisovat funkce, které budou jednotlivé bloky vykonávat. Pro jednotlivé bloky se vytvoří flow diagramy určující logiku a chování bloku.



Obrázek 1.15: Blokový diagram



Obrázek 1.16: Flow diagram

Výhodou rozdělení funkcionality do bloků je možnost delegování práce mezi více členů týmu. Každý člen bude zodpovědný za jeden blok, umožní to rychlejší vývoj a specializaci na jednu konkrétní věc. Výsledkem by měl být rychlejší a kvalitnější výsledek.

4. Realizace zapojení

Z blokového schématu se určí, jaké komponenty budou součástí řešení. Z katalogů, datasheetů či internetových obchodů se na základě technických parametrů, dodacích podmínek a ceny zvolí konkrétní části systému. V případě, že se systém skládá z modulů, je možné hotové části připojit na sběrnici a pokračovat programováním řídicího kódu. Samostatné moduly mohou vyžadovat vytvoření vlastní desky plošného spoje (DPS). Návrh a realizace DPS je samostatnou oblastí elektroniky v praxi, více o ní najdete v kapitole 4.

5. Test-driven development

V oblasti softwarového inženýrství se proces vývoje softwaru dělí do menších částí, které lze efektivněji spravovat. Jednou z metod poměrně oblíbených mezi programátory (hlavně mezi programátory na volné noze) je Test-driven development (TDD).

Základním prvkem při vývoji řešení pomocí TDD je test (například unit test). Před samotným psaním produkčního kódu se vytvoří test, kterým musí vytvořený kód následně projít. Test pomáhá hlavně při ujasnění specifikací funkcionality a odhalování nedostatků už během fáze vývoje.

Existuje i opačný přístup, kdy se nejprve vytvoří kód a ten se následně testuje. Takovýto přístup s sebou přináší rizika. Časem se většinou zjistí, že se požaduje nová (na začátku nevyslovená) vlastnost či funkcionalita aplikace. Tehdy se ukáže, že už vytvořené řešení nesplňuje tyto požadavky a je potřeba ho přepracovat. Příklad kritického rozhodnutí představuje výběr datové struktury používané v aplikaci. V některých případech jsou vhodnější pole než hashovací tabulka.

„*Je lepší přesně vědět, co jdeme stavět, než stavět a přemýšlet, co ještě dostavět.*"

Aplikace TDD v praxi má podobu vyobrazenou na obrázku 1.17. Nejprve se vytvoří test a následně se vytvoří kód. Nad hotovým kódem se spustí test. V první iteraci kód nemusí projít testem. V případě negativního výsledku testu se v kódu opraví chyby, které způsobily neúspěch v testu. Pokračuje se další iterací.

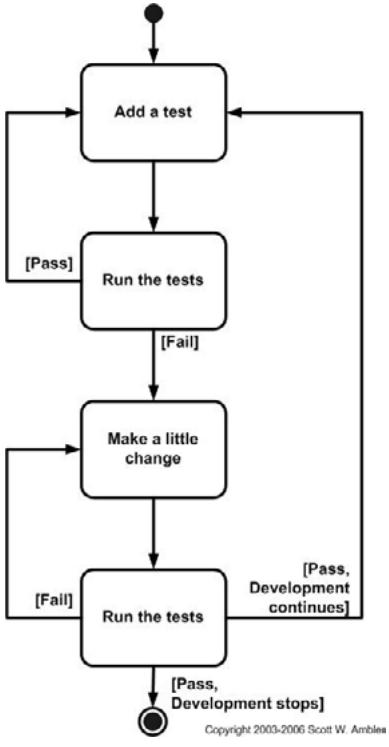
Pokud kód úspěšně prošel testem, je možné ho rozšířit o novou funkcionalitu nebo pokračovat dalšími částmi aplikace.

TDD přináší do vývoje výhody v podobě jednoduššího ladění a debuggování, jelikož se testují malé celky. Při testování se vychází z předpokladu, že ostatní části, které testem prošly, jsou funkční a není nutné se jimi zabývat při ladění nově odhalených chyb.

Další výhodu představuje použití testovacích případů pro účely dokumentace a jednodušší porozumění menším celkům kódu. Právě dokumentace a zpětná analýza řešení jsou kamenem úrazu většiny softwarových produktů. Na dokumentaci totiž obvykle nemá nikdo chuť ani čas.

Pokročilá aplikace TDD vede k ATDD, což je zkratka pro Acceptance test-driven development. ATDD je metodologie propojující funkční testování celku složeného z jednotlivých, už otestovaných částí. Otestovaný celek se tak stává předmětem předání koncovému uživateli. To

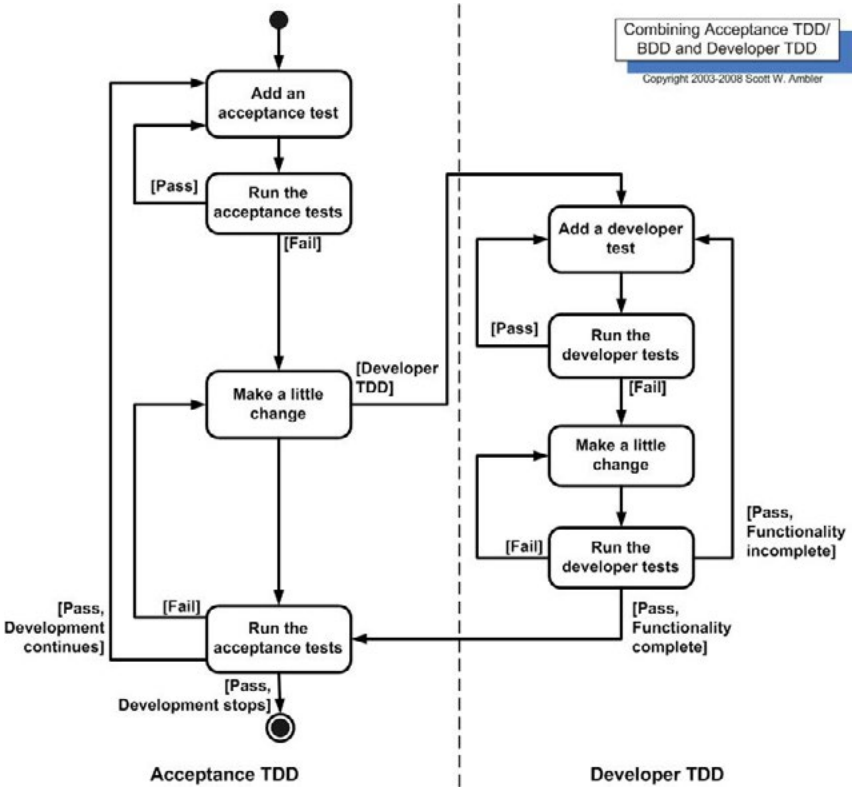
znamená, že pokud aplikace jako celek (funkčních a otestovaných malých fragmentů kódu) projde uživatelským testem (UAT – user-acceptance testing), považuje se za funkční, splňující definované požadavky.



Obrázek 1.17: TDD v praxi

Poznámka: Při testování TDD se odhalila většina chyb, které bránily funkčnosti kódu. ATDD testováním se identifikují problémy, kvůli kterým kód nesplňuje požadavky uživatele uvedené v zadání. Kód, který prošel testováním, není úplně bez chyb. Jejich navození ale může být málo pravděpodobné či zatím neznámé (například kompatibilita s jinými prvky systému). Přestože aplikace testem prošla, nemusí být uživatel s výsledkem stále spokojen.

Poznámka: Metodologie TDD je propojená s tvorbou unit testů. Podrobnější informace k těmto testům najdete v kapitole 2.



Obrázek 1.18: ATDD

6. Testování a ladění

Testování softwaru a embedded zařízení je jedním z důležitých bodů ve vývoji, jelikož testováním se odstraňují chyby, identifikují se slabá místa a navrhují se zlepšení pro další iterace či budoucí verze. Podrobnější popis testování, ladění a diagnostiky zařízení a řídicího kódu najdete v kapitole 2.

Vývoj řešení – pohled byznysu

Mnoho lidí má ambice nasazovat řešení postavené na Arduinu do komerční praxe. Z pohledu byznysu se Arduino ukazuje jako ne příliš vhodná platforma pro komerční řešení.

- **Víceúčelovost** – deska Arduina je víceúčelová, což se odráží v ceně i kvalitě provádění jednotlivých úkolů. Na vybudování prototypu je to skvělé řešení, jelikož se funkční prototyp dá postavit za několik hodin. Následně je lepší v případě potřeby vytvořit řešení s dedikovanými součástkami a konfigurací, která bude řešit jen jednu konkrétní věc. Žádné

zbytečné porty, rozšíření, nevyhovující parametry. Výhodou řešení postaveného na míru je i rychlost v porovnání s Arduinem.

- **Integrace** – integrace do existujících systémů je mírně složitější, protože systém je hotový a musí se pracovat jen s tím, co deska nabízí. Jiné vývojové desky či vlastní řešení mohou být plně přizpůsobené potřebám či požadavkům zadavatele.
- **Cílová skupina** – Arduino se zaměřuje na konečné spotřebitele (B2C), nikoliv na výrobce (B2B) hardwaru, kteří svoje produkty prodávají. Kvůli tomu je nastavená i cena, výrobní strategie, zařízení nelze objednat ve velkém množství (tisíce kusů). Embedded vývojové desky mají často širší nabídku externích modulů (shieldů) než Arduino, a to s lepšími parametry a nižšími cenami.
- **Rozměry** – rozměry Arduina nejsou příliš nakloněny moderním trendům miniaturizace. Existují desky s malými rozměry (například Arduino Mini, Micro, Nano). Stále se jedná o pevně dané rozměry, se kterými je třeba počítat.
- **Licenční podmínky** – licenční podmínky vyžadují otevřený design pro komerční řešení. Design, architektura a logika řešení jsou často předmětem konkurenční výhody či obchodního tajemství.
- **Legislativa** – na komerční produkty jsou kladeny požadavky, které určuje legislativa, normy či směrnice. Arduino často tyto požadavky nedokáže splnit. U kritických zařízení (mission critical devices), která se používají v oblastech, jako je medicína, letecký průmysl, jaderná energetika či armáda, se vyžaduje certifikace, jejíž získání je technicky, časově i finančně náročné. Proto se v těchto oblastech o Arduinu ani nemluví.

KAPITOLA 2

Software

S novým dnem přichází nové síly a myšlenky.
– Eleanor Roosevelt

V této kapitole se dozvíte o:
- Arduino IDE
- Struktuře programu
- Datových typech
- Funkcích
- Testech a cyklech
- Proměnných a konstantách
- Vstupech a výstupech
- Programovacích konvencích
- Knihovnách
- Ladění a diagnostice
- Tipech a tricích

Motivace

Druhá kapitola poskytuje komplexní přehled o oblasti programování embedded zařízení. Diskutovaná témata jsou variabilní a poskytují obraz o příležitostech a výzvách pro programátory. Prostudováním této kapitoly můžete získat více než jen informace, jak nastavit výstup na digitální pin, nebo znalosti, jak se tvoří cykly.

V kapitole se seznámíte s vývojovým prostředím Arduino Software, což je oficiální IDE na programování Arduina. Pro ty, kdo chtějí poznat, co se děje na pozadí, je určena část s vysvětlením procesu kompilace a nahrávání kódu do paměti zařízení.

Pro všeobecnou znalost a zlepšení kvality kódu zde najdete i doporučené konvence a postupy tvorby kódu. Konvence můžete aplikovat napříč všemi programovacími jazyky, jejich znalost oceníte i při dalším programování. Zajímavý je pohled na nejrozšířenější programovací vzory a ukázka jejich využití při programování Arduina. Součástí je i několik tipů a triků, které se osvědčily členům komunity při práci s hardwarem a softwarem, a návody nejčastějších přístupů při ladění a diagnostice kódu.

Implementace pokročilých funkcí se ve většině případů realizuje importem knihoven. Nezapomnělo se tedy ani na práci s knihovnami a návod na vytvoření knihovny vlastní.

Z pohledu jazyka budou představeny základní rysy jazyka C/C++, jako jsou datové typy, tvorba funkcí a cyklů, podmínky nebo práce se vstupy a výstupy. Najdete zde i další témata, jako například ukazatele, pole či dynamické alokování paměti. Tyto oblasti začátečníci často označují za problémové.

Portfolio znalostí si můžete rozšířit o základy regulárních výrazů, které využijete v praxi při řešení problémů, při vyhledávání, filtrování a verifikaci textových řetězců. Mezi umělci je vel-

mi oblíbené propojení Arduina s audio-vizuálními frameworky, které otevírají nové možnosti realizace projektů z oblasti umění, designu či neobvyklé interakce s uživatelem.

Vývojové prostředí (IDE)

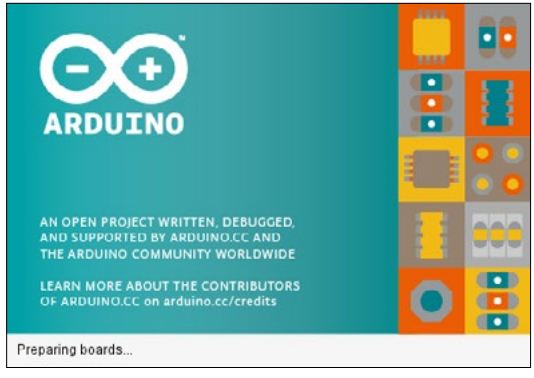
Integrované vývojové prostředí (v angličtině Integrated Development Environment) je prostředí, kde nápad dostává svou podobu programového kódu. Software je mozkem zařízení a určuje jeho kvalitu. Na kvalitě softwaru závisí funkcionalita, úroveň interakce s uživatelem, stabilita a robustnost řešení. Kvalitní IDE pomáhá:

- Tvořit kód efektivněji pomocí nápovědy
- Zlepšovat orientaci pomocí (auto)formátování
- Barevně zvýrazňovat kód
- Zobrazovat nápovědu a pomůcky u chyb během kompilace

IDE představuje soubor nástrojů pod jednou střechou, které umožňují vývoj softwarových aplikací. Prostředí obsahuje nástroje potřebné k psaní kódu, jeho kompilaci, ladění a diagnostice, nástroje na tvorbu grafického rozhraní či emulátory webových prohlížečů a mobilních zařízení.

Většina IDE představuje v současnosti univerzální nástroj. Autoři se snaží oslovit širokou komunitu programátorů tím, že jim umožní vyvíjet aplikace v různých programovacích jazycích. Vzhledem k velké nabídce IDE začneme stručným přehledem prostředí použitelných k programování Arduina.

Arduino Software IDE



Obrázek 2.1: Úvodní obrazovka Arduino IDE na Windows

Jedním z nejrozšířenějších IDE pro platformu Arduino je Arduino Software, v době psaní knihy ve verzi 1.6.5. V květnu 2016 měl software přes dva miliony stažení. Je to open-source prostředí s otevřeným kódem dostupným na portálu GitHub. Instalační balík se vydává ve verzích pro následující platformy:

- Windows Installer
- Archiv ZIP pro Windows (práce na stanici bez administrátorských práv)
- Mac OS X 10.7+
- Linux 32bit
- Linux 64bit

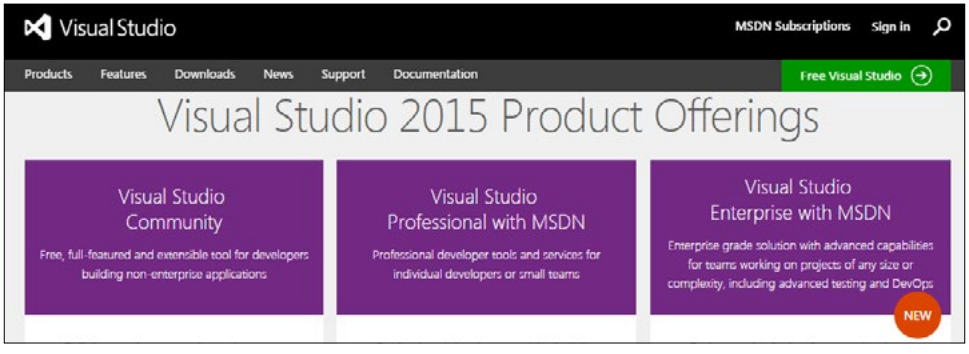
Poznámka: Většina prostředí, která zavedla podporu programování Arduina, vyžaduje instalaci Arduino IDE.

Visual Studio Community

Visual Studio je vývojové prostředí z dílny společnosti Microsoft. Jeho obrovskou výhodou je podpora jazyků pro vývoj webu, aplikací pro Windows a Linux. Novinkou je podpora jazyků pro vývoj cross-platform aplikací pro systémy iOS, Android, Windows Mobile.

Doinstalováním pluginu je možné Visual Studio používat i k vývoji aplikací pro platformu Arduino. Mezi podporovanými jazyky najdete například C#, Visual Basic, F#, C++, Python, Node.js a HTML/JavaScript. Visual Studio je dostupné na stránkách společnosti Microsoft *https://www.visualstudio.com*.

Visual Studio je dostupné v několika verzích. Zajímavou je verze Visual Studio Community, která je k dispozici zdarma pro open-source projekty, akademický výzkum, výuku a vzdělávání a malé profesionální týmy vývojářů (podmínkou je maximálně 5 uživatelů a vývoj nekomerčních aplikací).



Obrázek 2.2: Verze Visual Studia

Atmel Studio

Atmel Studio je vývojové prostředí volně dostupné ke stažení na stránkách společnosti Atmel Corporation. Prostředí je určeno k programování mikroprocesorů společností Atmel a ARM. Atmel Studio je dostupné ke stažení na stránkách společnosti Atmel *http://www.atmel.com/microsite/atmel_studio6/*.