12. ročník Žilina 2. – 4. 7. 2024
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf
OSSConf 2024: 25–34

PRIEMYSELNÉ ZARIADENIA ZALOŽENÉ NA ARDUINE, RASPBERRY PI, ESP32... GNU EMACS AKO IDE

RICHARD FABO (SK)

Abstrakt. Článok pojednáva o využití interpertra programovacieho jazyka (e)Lisp so zabudovaným textovým editorom, s názvom GNU Emacs. V krátkom opise sa vyzdvihnú jeho hlavné a najdôležitejšie vlastnosti, oblasti použitia a jeho jedinečnosť vo svete softvéru, z hľadiska funkčnosti i filozofie.

Spomenutím hlavných nedostatkov originálneho Arduino IDE sa dostaneme k spôsobu, ako nahradiť Arduino IDE ľubovoľným textovým editorom (prípade vývojovým prostredím), vo všetkých základných oblastiach (od editovania kódu, cez jeho nahrávanie do vývojových dosiek (nielen Arduino), až po nahradenie sériovej konzoly). Bližšie sa opíše, ako funguje proces inštalácie vývojových dosiek a knižníc, ako s nimi narába arduino-cli a ako generovať FQBN (plne kvalifikované názvy dosiek) pre účely kompilácie a nahrávania.

Kľúčové slová. PLC, priemysel, Arduino, ESP32, Raspberry Pi, IDE.

INDUSTRIAL DEVICES BASED ON ARDUINO, RASPBERRY PI, ESP32... GNU EMACS AS AN IDE.

Abstract. This article discusses the use of the (e)Lisp programming language interpert with a built-in text editor, called GNU Emacs. A brief description will highlight its main and most important features, its areas of application and its uniqueness in the software world, both in terms of functionality and philosophy.

By mentioning the main shortcomings of the original Arduino IDE, we will get to the way to replace the Arduino IDE with any text editor (or integrated development environment), in all basic areas (from editing code, to uploading it to development boards (not only Arduino), to replacing the serial console). It will describe how the process of installing development boards and libraries works, how arduino-cli handles them, and how to generate FQBNs (fully qualified board names) for compilation and upload purposes.

Keywords. PLC, industry, Arduino, ESP32, Raspberry Pi, IDE.

Úvod

Spoločnosť Arduino SRL zohrala veľkú úlohu pri popularizovaní programovania mikrokontrolérov. Ich platforma `Arduino` ponúka jednoduchý a priateľský spôsob pre začiatočníkov aj skúsených programátorov na vytváranie a experimentovanie s elektronickými projektami. Vďaka ich Open Source prístupu sa programovanie mikrokontrolérov stalo prístupnejším a atraktívnejším pre ľudí po celom svete.

Vydané `Arduino IDE` umožňovalo (a umožňuje) zjednodušené programovanie vďaka veľkému množstvu knižníc a príkladov, a tiež preklad a nahrávanie kódu do mikrokontroléra bez potreby znalosti tohto procesu. Už v prvých verziách mal integrovaný sériový monitor, zvýrazňoval syntax, umožňoval presun na chyby počas kompilácie...

Proces prekladu bol zásadne zjednodušený, pretože nebolo potrebné vytvárať vlastné `Makefile` súbory na kompiláciu, linkovanie, nahrávanie, s potrebou špecifikovania veľkého množstva parametrov.

Samotné vývojové prostredie bolo v porovnaní so zavedenými „dospelými“ vývojovými prostrediami (`Vim`, `Emacs`, `Visual Studio`, `Eclipse`, `Sublime Text`, `Atom` ai.) okliešténé a chýbali mu vlastnosti, ktoré zjednodušujú a zrýchľujú proces programovania, napríklad:

- doplňanie kódu (z predchádzajúcich výskytov alebo z slovníka);
- „folding“ (zbaľovanie) častí kódu;
- presun na definície, funkcie;
- integrovaná dokumentácia;
- práca v konzole ai.

Niektoré chýbajúce funkcie a vlastnosti a zlepšený výkon prinieslo až `Arduino IDE 2.0` v roku 2021.

1. Arduino-cli

`Arduino-cli` je nástroj pre príkazový riadok, ktorý umožňuje kompilovať, nahrávať a spravovať knižnice (a dosky) Arduino (a iné) bez potreby prostredia `Arduino IDE`. Zjednodušene môžeme povedať, že „obaľuje“ (zastrešuje) programy potrebné k prekladu a nahrávaniu kódu, volá ich s požadovanými parametrami, číta ich stavy. Vďaka rozhraniu príkazového riadku je ho možné využiť v rôznych vývojových prostrediach. My sa zameriame na `GNU Emacs`.

1.1. Štruktúra adresárov pre arduino-cli, nástroje

`Arduino-cli` vyžaduje, aby určité súbory (napr. definície vývojových dosiek, binárne spustiteľné súbory pre kompiláciu a pod.) boli umiestené v presne definovanej adresárovej štruktúre. Tento článok bude opisovať skutočnosti v operačnom systéme `GNU/Linux`.

Po inštalácii `arduino-cli` (viď `https://arduino.github.io/arduino-cli/`) a zavolaní príkazu `arduino-cli config init` dôjde k vygenerovaniu súboru `arduino-cli.yaml`. V našom prípade vyzerá jeho časť nasledovne (máme ho presunutý v skrytom adresári `~/.arduino15`)

```
board_manager:
  additional_urls:
```

```
    https://arduino.esp8266.com/stable/
            package_esp8266com_index.json
    http://apps.industrialshields.com/main/arduino/boards/
           package_industrialshields_index.json

directories:
  data: /home/richard/.arduino15
  downloads: /home/richard/.arduino15/staging
  user: <cesta k~našim programom pre vývojové dosky>
```

Za zmienku stoja URL adresy v sekcii `board_manager`, ktoré určujú, odkiaľ sa budú sťahovať príslušné `json` súbory pre iné ako originálne (`Arduino`) vývojové dosky a zariadenia. V našom prípade máme pridané dosky pre `ESP8266` a `PLC zariadenia Industrial Shields` (`https://famme.sk/plc.html`).

V sekcii `directories` sú nami definové cesty ku knižniciam a nástrojom pre všetky inštalované vývojové dosky a tiež k našim programom. Viac informácii k štruktúre adresárov, ktoré by presiahli rozsah tohto článku, môže láskavý čitateľ nájsť na `https://linuxos.sk/blog/richard/detail/programovanie-arduino-a-derivatov-bez-arduino/`.

Základné príkazy `arduino-cli` pre prácu s doskami

- zoznam inštalovaných dosiek `arduino-cli board listall`,
- aktualizácia zoznamu výrobcov a dosiek `arduino-cli core update-index`,
- inštalácia novej dosky `arduino-cli core install <výrobca>:<doska>`,
- vyhľadávanie v zoznamoch dosiek `arduino-cli core search <reťazec>`.

1.2. Čo je FQBN a ako ho vytvoriť?

`FQBN` je skratka z `Fully Qualified Board Name` („plne kvalifikovaný názov dosky“). Ide o jedinečný identifikátor konkrétnej dosky, ktorý zvyčajne obsahuje typ dosky, model mikrokontroléra, taktovaciu frekvenciu a ďalšie dôležité informácie. Je to jedným z nevyhnutných parametrov zadávaných pre `arduino-cli`, ktorý sa skladá z nasledovného:

```
<výrobca>:<architektúra>:<označenie dosky>[=<ďalšie možnosti,
                                           napr. typ procesora, frekvencia>]
```

Niekoľko príkladov FQBN:

- Arduino Leonardo `arduino:avr:leonardo`,
- Arduino Uno WiFi `arduino:avr:unowifi`,
- Industrial Shields Ardbox Analog HF+ `industrialshields:avr:ardbox:cpu=ardboxanaloghfplus232`,
- Industrial Shields M-Duino 58+ `industrialshields:avr:mduino:cpu=mduino58plus`.

Ako zistiť tieto FQBN?

Čitateľa v prípade záujmu o podrobnejšie informácie, ako sa tieto FQBN tvoria, ako ich manuálne zistiť zo štruktúry súborov inštalovaných dosiek, len odkážeme na odkaz `https://linuxos.sk/blog/richard/detail/programovanie-arduino-a-derivatov-bez-arduino-2/`. Nám nateraz stačí, že FQBN si vieme vygenerovať pomocou skriptu v jazyku `Tcl`, dostupného na adrese `https://gitlab.com/tcl-tk-public/arduino-fqbn-generator`.

Skript nás prevedie viacerými krokmi:

- v 1. kroku zistí a vypíše zoznam dosiek, ktoré máme nainštalované;
- po zvolení dosky (defacto výrobca) zistí, aké máme jej verzie;
- po zvolení verzie zistí, či je potrebné ešte upresnenie typu cpu – ak áno, tak je možné si tento zvoliť v následnom kroku;
- nakoniec vypíše vygenerovaný FQBN a tento uloží do súboru `fqbn.txt`, na ktorý sa odkazuje `Makefile`.

Skript spustíme príkazom:

```
tclsh <cesta ku skriptu>/arduino-fqbn-generator.tcl.
```

Pre spustenie skriptu potrebujeme mať nainštalovaný interpreter jazyka `Tcl`. S najväčšou pravdepodobnosťou, hraničiacou s istotou, bude `Tcl` v repozitároch bežnej GNU/Linuxovej distribúcie, napr. v distribúciach založených na projekte `Debian` ho nainštalujeme príkazom `sudo apt install tcl`.

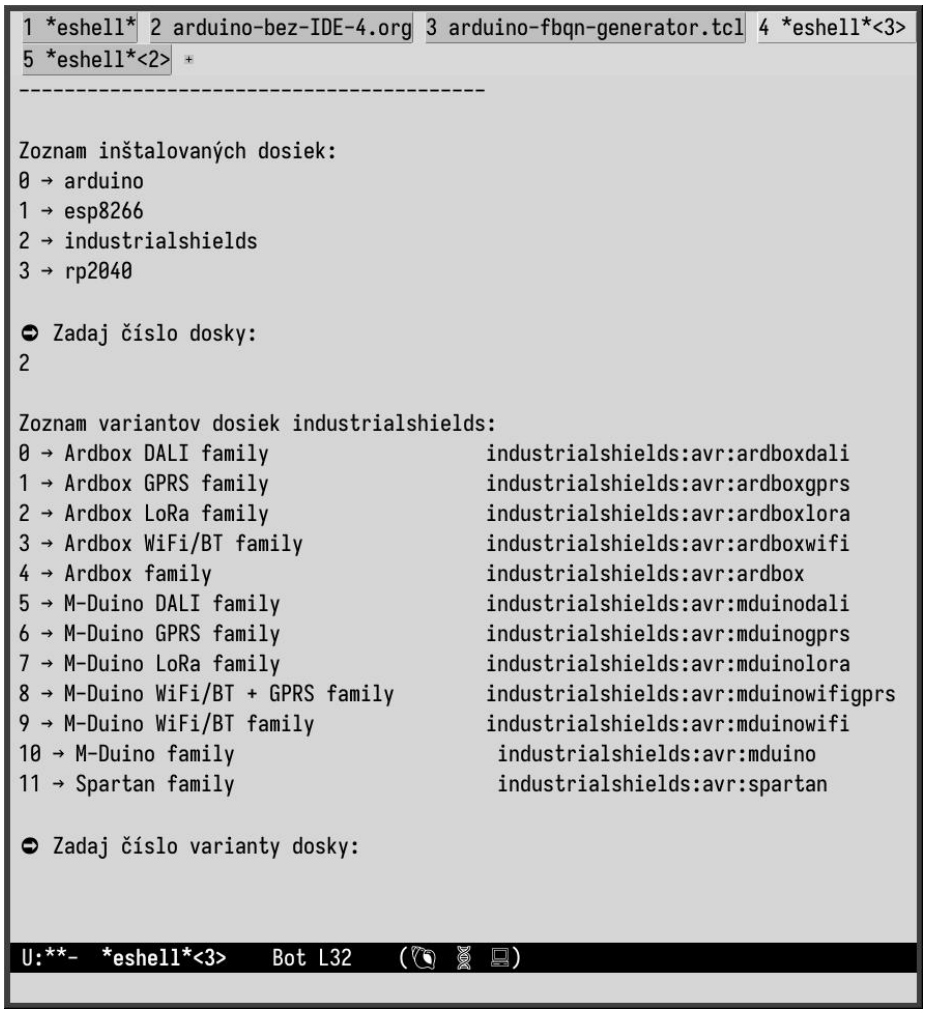
1.3. `Makefile` pre kompiláciu a nahrávanie programov

Ako bolo spomínané, na tento účel je možné v GNU Emacs použiť rozšírenie `arduino-cli-mode`. Osobne však preferujem vytvorenie `Makefile` súboru.

`Makefile` je súbor obsahujúci pravidlá a akcie určené na automatizáciu procesu kompilácie a spustenia programov v rámci vývojového prostredia. Tento súbor definuje závislosti medzi zdrojovými kódmi a objektovými súbormi, ako aj príkazy potrebné na kompiláciu a linkovanie programu. Náš `Makefile` vyzerá nasledovne (krátené, celý súbor na `https://gitlab.com/tcl-tk-public/arduino-fqbn-generator/-/tree/main/makefile_for_arduino-cli`):

```
PROGRAM ?= $(wildcard ./*.ino)

ifeq ($(wildcard /dev/ttyACM*),)
    ifeq ($(wildcard /dev/ttyUSB*),)
        PORT ?=
    else
        PORT ?= $(firstword $(wildcard /dev/ttyUSB*))
    endif
else
    PORT ?= $(firstword $(wildcard /dev/ttyACM*))
```



Obr. 1. Generátor `FQBN`

```
endif

FQBN ?= $(shell cat fqbn.txt)
# \dotskontrola, či existuje súbor fqbn.txt

ARDUINO-CLI-BIN ?= /home/richard/.arduino15/arduino-cli
# \dotskontrola, či existuje arduino-cli

kompiler-verify ?= $(ARDUINO-CLI-BIN) compile -b
kompiler-verbose ?= $(ARDUINO-CLI-BIN) compile -v -b
```

```
kompiler-upload ?= $(ARDUINO-CLI-BIN) upload -v -b

all:
 $(ARDUINO-CLI-BIN) compile -b $(FQBN) $(PROGRAM)

upload:
ifeq ($(PORT),)
 $(error Chyba: zariadenie nepripojené!)
else
 @echo "Zariadenie pripojené na: $(PORT)"; \
 $(ARDUINO-CLI-BIN) upload -v -b $(FQBN) -p $(PORT) $(PROGRAM);
endif
```

Vysvetlenie kódu:

- `PROGRAM ?= $(wildcard ./*.ino)`
  Vytvoríme si premennú `PROGRAM`, do ktorej si načítame názov súbor s príponou `ino` (môže byť len jeden taký súbor v adresári s Makefile, inak ho treba explicitne definovať).
- `ifeq ($(wildcard /dev/ttyACM*),)\dots`
  `ttyACM`, resp. `ttyUSB` sú systémové súborové symbolické odkazy, ktoré sa používajú na komunikáciu s USB zariadeniami pripojenými k počítaču cez sériový port.
- `FQBN ?= $(shell cat fqbn.txt)`
  Načítanie plne kvalifikovaného názvu dosky, vytvoreného skriptom v jazyku `Tcl`.
- `ARDUINO-CLI-BIN ?= home/richard.arduino15/arduino-cli`
  Je cesta k binárke `arduino-cli`.
- `kompiler-verify ?= $(ARDUINO-CLI-BIN) compile -b`
  `kompiler-verbose ?= $(ARDUINO-CLI-BIN) compile -v -b`
  `kompiler-upload ?= $(ARDUINO-CLI-BIN) upload -v -b`
  Sú pomocné premenné, aby sme nemuseli reťazce, ktoré reprezentujú, písať celé v neskorších receptoch. Tie sa totiž konštruujú tak, aby boli ľahko upraviteľné práve zmenami premenných.
- `all, upload`
  Sú konkrétne „recepty“, t. j. časti kódu, ktoré sa vykonajú, ak zavoláme z príkazového riadku príkaz `make`. „Recept“ `all` sa vykoná, ak je príkaz `make` bez parametrov, `verbose`, ak použijeme príkaz `make verbose` atď.

Ak to zhrnieme, postup pri zostavovaní programu pomocou `arduino-cli` s využitím `Makefile` je nasledovný:

- inštalácia príslušnej vývojovej dosky pomocou
  `arduino-cli core install <výrobca>:<doska>`;

- spustenie skriptu na vygenerovanie FQBN pomocou `tclsh <cesta ku skriptu>/arduino-fqbn-generator.tcl`;
- zostavenie programu pomocou `make` (prípadne `make verbose`, pre viac informácii počas kompilácie).

Následne nahratie do pripojeného zariadenia sa vykonáva pomocou príkazu `make upload`.

2. GNU Emacs

Niektoré definície zhrňujú:

> „GNU Emacs je textový editor so zameraním na prácu s textom a programovaním, ktorý bol vytvorený Richardom Stallmanom v roku 1984 ako súčasť projektu GNU. Je jedným z najstarších a najznámejších textových editorov vo svete s voľne dostupným zdrojovým kódom.“

Richard Stallman pracoval na MIT (Massachusetts Institute of Technology), kde sa tešil programovací jazyk `LISP` veľkej popularite. Je preto pochopiteľné, že napísal malé jadro editora v jazyku `C`, ktorého hlavný účel spočíval v interpretácii dialektu jazyka `lisp` špeciálne určeného na editovanie textu, ktorý dostal názov `Elisp` (uvádzaný tiež ELISP, elisp, (e)lisp). Vznikla tak nová (dnes sa nám zdá samozrejme prirodzená), zaujímavá paradigma vo vývoji softvéru – malé rýchle jadro + skriptovací jazyk.

V Emacse sa akákoľvek interakcia vykonáva pomocou príkazov. A tieto príkazy sú jednoduché Elisp-ové funkcie.

Hlavnou dátovou štruktúrou Emacsu sú `buffery`. Na pozadí sú spôsobom výmeny textových prúdov. V hľadiska ovládania je výhodou, že editačné a vizuálne príkazy a funkcie sú vždy rovnaké – či sa jedná o písanie kódu, alebo prácu napríklad v emulátore terminálu.

Dôležitou súčasťou sú tzv. režimy, nazývané `módy`. Je to súbor nastavení a dodatočných funkcií, ktoré menia správanie buffera a teda aj spôsob interakcie s užívateľom. Definujú také veci, ako napr. odsadenie textu, zvýrazňovanie syntaxe, priradenia klávesových skratiek.

Pochopiteľne, tento článok nie je a nemôže byť návodom na používanie Emacs-u v plnom jeho rozsahu, s využitím všetkých funkcií a možností, ktoré nám poskytuje interpreter jazyka Elisp.

2.1. arduino-mode, arduino-cli-mode

`arduino-mode` (`https://melpa.org/#/arduino-mode`) je rozšírením `c-mode` GNU Emacs-u o potrebné konštanty, kľúčové slová a úpravu odsadzovania.

2.2. Editačné funkcie, chýbajúce v Arduino IDE

Vzhľadom na rozsah článku nie je možné opísať všetky funkcie, ktoré robia Emacs unikátnym. Preto si len zdôraznime niektoré, ktoré nám editáciu kódu zásadne zlepšia. Za zmienku stoja:

Skoky na definície premenných a referencie funkcií (s náhľadom alebo bez), s využitím viacerých spôsobov: pomocou vygenerovaného indexu zdrojového kódu programom `ctags` resp. `etags`; pomocou interaktívneho zužovania textu počas zadávania vyhľadávaného reťazca (príkaz `M-x helm-swoop`) v aktuálnom bufferi alebo v rámci celého projektu (príkaz `M-x noccur`).

Prepojenie s databázou dokumentácie príkazov. S využitím projektu `Dash`, ktorý umožňuje získavanie offline dokumentácie, a jej následné zobrazenie vo vstav internetovom prehliadači; tiež je možné túto databázu využiť ako základ pre doplňovanie zadávaných príkazov pomocou slovníka (`https://linuxos.sk/blog/richard/detail/gnu-emacs-vzdy-je-co-vylepsit-2-dash/`).

Emulátor terminálu (vytvorený v Elisp-e). Jeho hlavnou výhodou, oproti zaužívaným emulátorom terminálu, je možnosť využívania všetkých editačných príkazov Emacs-u (voľný pohyb kurzora, doplňovanie, vyhľadávanie, priame spúšťanie funkcií či programov napísaných v Elisp-e, vstavaný nástroj na vzdialený prístup cez ssh, ftp...).

„Zbaľovanie“ výrazov, funkcií, vizuálne potlačenie časti kódu. Známa funkcia aj z iných vývojových prostredí zásadne zlepšuje orientáciu v kóde. Vizuálne potlačenie časti kódu (funkcie `M-x narrow-*`), na ktorom práve nepracujeme, zjednodušuje orientáciu a zlepšuje sústredenie.

Automatické doplňovanie (z predchádzajúcich výskytov, zo slovníka). Pomocou niekoľkých, navzájom sa doplňujúcich, funkcií je možné text doplňovať postupným stláčaním klávesovej skratky, alebo výberom z „drop-down“ ponuky.

Strom súborov a funkcií . Je grafickým zobrazením štruktúry celého projektu, s jednoduchým presunom po súboroch projektu, funkciách a definíciách. Nakoľko sa (stále) jedná o buffer Emacs-u, tak máme k dispozícii všetky jeho editačné funkcie.

„Snippety“ –vytvorenenie časti kódu po zadaní kľúčoveho slova. „Snippety“ (viď nNami vytvorené snippety, spolu so slovníkom pre automatické doplňovanie na `https://gitlab.com/arduino91/arduino-yasnippets-for-emacs`) sú fragmenty kódu alebo textu, ktoré je možné vytvoriť a uložiť do databázy a potom ich kedykoľvek vložiť do textu zadaním krátkeho kľúčového slova. Takto možno automatizovane vytvoriť i komplikované štruktúry, bez potreby ich pamätania.

Virtuálne stránky. Slúžia na rozdelenie kódu po ľubovoľne dlhých častiach, po ktorých je možné sa efektívne presúvať (napr. príkazom `M-x helm-pages`).

Viacnásobné kurzory sú šikovnou funkciu, ktorá umožňuje mať viacero kurzorov v texte naraz. Táto funkcia umožňuje editovať alebo zmeniť viacero častí textu naraz pomocou jedného pohybu kurzora. Tento nástroj je užitočný pre editovanie textových súborov, kde je potrebné vykonať rovnakú zmenu na viacerých miestach.

Záložky a registre. Záložky, vizuálne, v rámci jedného súboru, alebo ich databáza (obmedzená len na pripojené súborové systémy) slúžia na rýchle presuny na uložené pozície. Pritom táto pozícia nemusí byť len v rámci jedného súboru, alebo súborov v rámci jedného projektu, ale môže sa jednať i o presun do adresárov, pozície v PDF súbore, riadok v terminále, poznámku v tele e-mailu a pod. Registre sú viacnásobné schránky, uložené pod ľubovoľnými názvami, ktoré môžu uschovávať text, napríklad časť kódu. Vkladanie obsahu registrov môže sprevádzať i náhľad na ich obsah.

Dynamické expandovanie výberu je mimoriadne návyková funkcia. V podstate sa jedná o rozširovanie výberu textu, pričom po aktivácii klávesovej skratky sa zvolí slovo, následne celý výraz, blok, funkcia.

Správa verzií. Jedná sa o systém, ktorý umožňuje užívateľovi sledovať zmeny v texte a ukladať históriu verzií dokumentu, umožňuje zaznamenávať zmeny, porovnávať verzie a obnovovať predchádzajúce verzie dokumentu. Emacs podporuje rôzne systémy, ako sú Git, Mercurial, CVS, Bazaar atď.

Rýchly presun a kopírovanie v rámci viditeľnej časti. Jedná sa o presun kurzora na viditeľný text pomocou rozhodovacieho stromu založeného na znakoch. Jeho rozšírením je kopírovanie oblastí na pozíciu kurzora. Snahou je minimalizovanie počtu stlačení klávesov pri týchto častých operáciách (`https://linuxos.sk/blog/richard/detail/avy-utok-velociraptora/`).

Elektrické a farebné zátvorky Elektrické zátvorky sú funkcia, ktorá automaticky vkladá uzatváraciu zátvorku (rôznych typov) pri písaní otváracej zátvorky. Pomáha to udržiavať zátvorky správne uzavreté a minimalizovať prípadné chyby. Farebné zobrazenie rôznych zátvoriek pomáha vizuálne odlíšiť jednotlivé zanorenie blokov kódu, nakoľko sa zátvorky zobrazujú v zjednotených farbách a teda je ľahké identifikovať, ktoré zátvorky sa k sebe vzťahujú.

Integrovaný „serial monitor“ . Vstavaný sériový monitor, pri práci s ktorým opäť využívame všetky editačné možnosti Emacs-u.

Preklad a nahrávanie Samotný Emacs obsahuje príkaz `M-x compile`, ktorý zavolá `make`, v prípade potreby s doplnenými parametrami.

Na nahrávanie kódu nám postačí malá funkcia:

```
(defun arduino-make-upload (&optional arg)
  "Upload to device using Makefile"
  (interactive)
  (shell-command (concat "make upload")))
```

Kontaktná adresa

Ing. Richard Fabo, Triton Famme s.r.o., Levočská 862/28, 058 01 Poprad,
E-mailová adresa: triton@famme.sk, https://triton.famme.sk