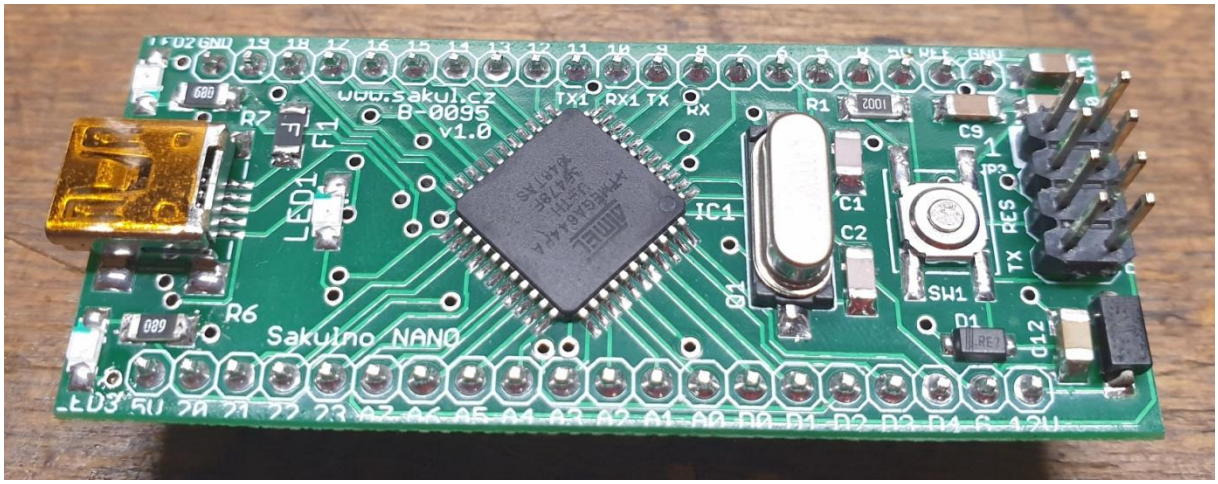
Sakulno NANO 644 (EA)

Lukáš Kořínek – www.sakul.cz – SakulRaider@seznam.cz

Poslední aktualizace: 26.08.2020 – PCB: B-0095



Tato deska vznikla, když jsem řešil potřebu nového mikropočítače pro moji konstrukci stopek. Bohužel žádný běžně podporovaný mikropočítač pro vývojové rozhraní Arduino IDE mi nevyhovoval, protože měl buď málo vývodů, nebo naopak až zbytečně mnoho. Proto jsem chvilku hledal na internetu, až jsem narazil na **MightyCore**. To přidává do vývojového prostředí podporu nových mikropočítačů: ATmega8535, ATmega16, ATmega32, ATmega164, ATmega324, ATmega644 a ATmega1284, přičemž právě ATmega644 mě zaujal asi nejvíce. Za rozumné peníze nabízí dostatek vývodů, paměti Flash i paměti SRAM. Následně mě tedy napadlo udělat maličkou desku inspirovanou deskou Arduino NANO, ale s mikropočítačem ATmega644 Sakulno NANO 644. Nicméně je možno osadit kterýkoli ze zmíněných mikropočítačů, záleží jen na Vašich preferencích.

Technické specifikace:

PCB	B-0095
Napájecí napětí	7-15V DC (označeno jako 12V)
Provozní napětí	5V (max 500mA)
Mikropočítač	ATmega644 (všechny varianty: P, PA, A)
Paměť Flash	64KB (z toho zabírá 1KB bootloader)
Paměť SRAM	4KB
Paměť EEPROM	2KB
Takt procesoru	16MHz
Analogové vstupy	8x
Digitální I/O	32x (6 s podporou PWM)
Maximální proud výstupu	40mA (doporučeno max 20mA na vývod a 200mA celkem)
Komunikační rozhraní	2xUART, I2C, SPI
Připojení	Mini USB přes CH340G na UART0
Velikost	60x15x10mm (výška je bez osazených konektorů)

Popis konstrukce:

V první řadě je nutné upozornit, že tato konstrukce ve své podstatě nemá žádnou funkci. Svou funkci získá až teprve poté co ji použijete v nějakém Vašem zařízení a nahrajete do ní Váš firmware. Ve své podstatě jde pouze o mikropočítač umístěný na PCB s tím, že jsou přidány nezbytné komponenty, aby mohl fungovat. Je osazen mini USB konektor, přes který je možno

tuto desku napájet a nahrávat do mikropočítače firmware. Dále jsou vyvedeny po stranách desky všechny vývody mikropočítače, takže je možno celou tuto desku zasadit do nějaké složitější konstrukce. Samozřejmě aby tato konstrukce byla i nějak přínosná je u ní vyřešena konstrukční chyba většiny Arduino desek, kde je závažná chyba v obvodu napěťového managementu. Většina těchto desek totiž propouští napětí na USB konektor, pokud jsou napájeny z externího zdroje. Toto je dle specifikací USB naprosto nepřípustné, a proto to na tomto zařízení bylo vyřešeno a nehrozí závada vlivem zavlečení napětí.

Schéma zapojení:

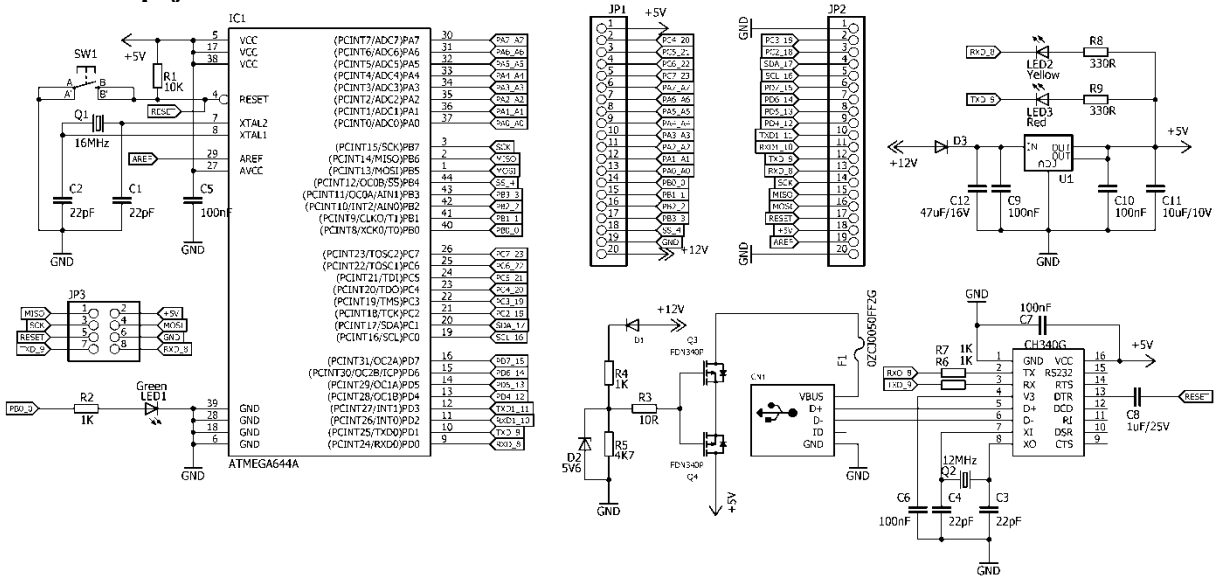


Schéma zapojení. Obrázek v plném rozlišení je součástí elektronické dokumentace.

Popis zapojení:

Jak je ze schématu patrné všemu dominuje použitý mikropočítač ATmega644 (IC1). Nicméně bez jakýchkoli úprav je možno použít i jiný kompatibilní mikropočítač jako například: ATmega8535, ATmega16, ATmega32, ATmega164, ATmega324, ATmega644 a ATmega1284. Avšak varianty ATmega8535, ATmega16 a ATmega164 asi nedávají příliš smysl kvůli malé paměti Flash. Většina vývodů použitého mikropočítače je vyvedena na pájecí body po stranách PCB, kam je možno osadit kolíkové lišty JP1 a JP2. Dále je zde obvody krystalového oscilátoru tvořeného krystalem Q1 (16MHz) a kondenzátory C1 a C2. Krystal je použit o frekvenci 16MHz, ale je možno použít i jiné frekvence, případně je možno mikropočítač přepnout na interní oscilátor. Podrobný rozpis použitelných krystalů najdete dále. Obvod resetu mikropočítače je pak realizován pomocí tlačítka SW1 (manuální reset) a rezistoru R1, případně přes C8 z vývodu DTR obvodu CH340G (U$1), což se používá při aktualizaci firmware přes USB.

Tím jsme se dostali k USB/TTL převodníku, jež převádí komunikaci z USB na UART0. Tento převodník byl zvolen CH340G pro svoji dobrou podporu například Windows 10, jeho dobrou dostupnost a příznivou cenu. Převodník používá krystal Q2 (12MHz) plus několik kondenzátorů C3, C4, C6, C7 a již zmíněný C8. Jako USB konektor byl zvolen mini USB, což sice není zcela běžný konektor, ale doma se mi jich válí celkem dost, takže jsem je potřeboval zužitkovat. Do budoucna je možné použití micro USB, který je mnohem rozšířenější.

Napětí 5V z tohoto konektoru je vedeno přes PTC pojistku F1 do obvodu napěťového managementu tvořeného dvěma tranzistory mosfet s vodivostí P (FDN340P). Tyto tranzistory jsou pak ovládány přes napěťový dělič R4 a R5 se zenerovou diodou D2 (5V6). Správná polarita je zajištěna diodou D1. Tento obvod zajistí, že se nemůže napětí 5V z integrovaného

stabilizátoru U1(LM1117) dostat na USB konektor. No a tímto jsme se propracovali až ke stabilizátoru napětí, jež umožňuje funkci i na napětí v rozsahu cca 7-15V. Vstup stabilizátoru je chráněn diodou D3 proti nechtěnému přepólování. Kondenzátory C9-12 jsou filtrační.
Zbývají nám už jen signalizační LED diody LED2 a LED3, jež signalizují probíhající komunikaci na sériovém portu UART0. LED1 je stejně jako u klasické Arduino desky použita na jednom výstupním pinu procesoru pro signalizaci.
Samozřejmě nesmím zapomenout na konektor JP3. Ten slouží k připojení ISP programátoru, pokud potřebujeme do mikropočítače nahrát například bootloader, nebo přímo firmware (pokud bootloader nepoužíváme). Nicméně ti pozorní si jistě všimli, že na běžný ISP konektor má tento nějak moc pinů. To je způsobeno tím, že já používám konektor, kde je ještě vyveden port UART0, což se může hodit při nahrávání firmware pomocí externího převodníku.
No a to je tedy, co se týká zapojení vše.

Schéma osazení:

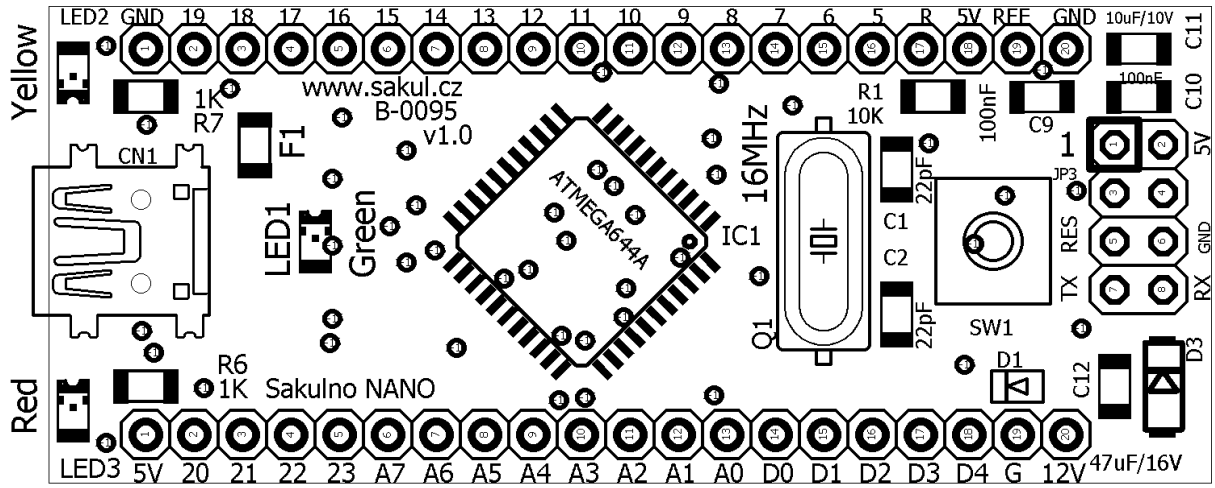


Schéma osazení TOP strany.

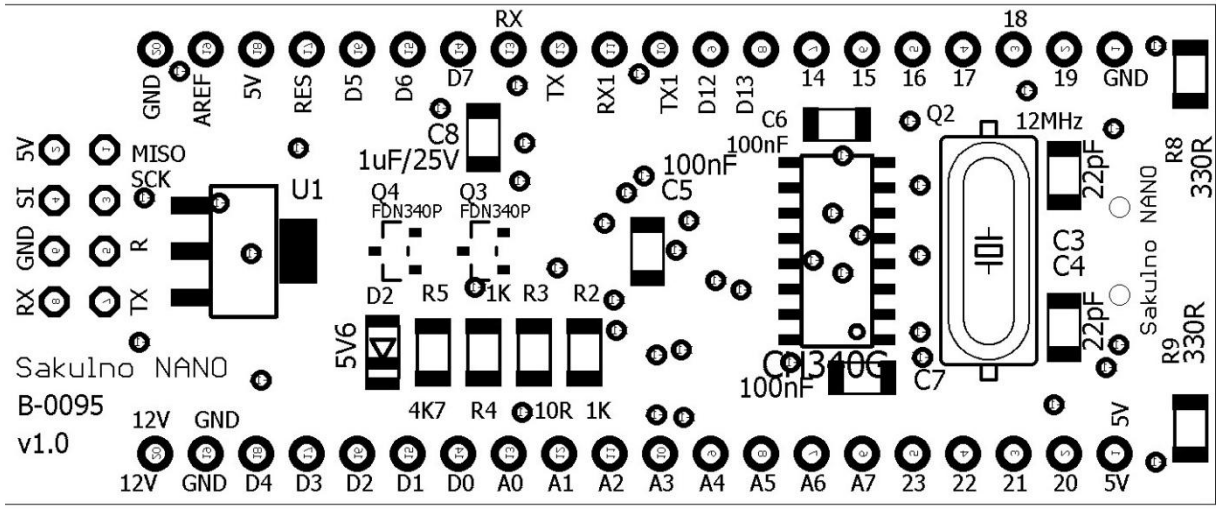


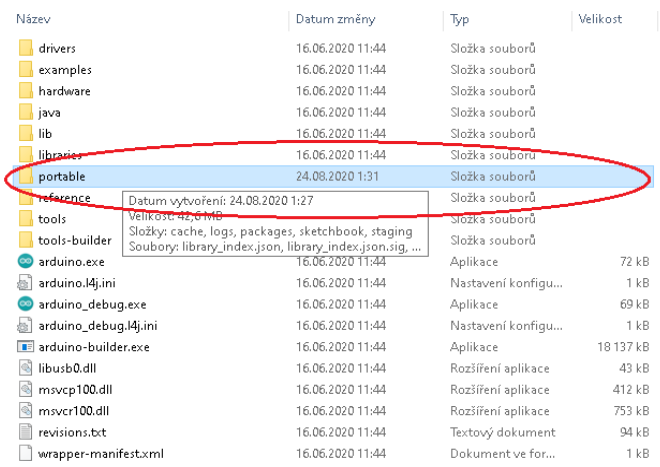
Schéma osazení BOTTOM strany.

Výkresy spojů PCB neuvádím neb sou součástí elektronické dokumentace a výroba této PCB v domácích podmínkách je dost těžko řešitelná díky prokoveným otvorům.

Postup osazení:
Pokud se rozhodnete si PCB osadit sami je asi nejlepší metoda osadit kompletní stranu TOP SMD součástkami a nechat zapájet v peci metodou Refflow (případně horkým vzduchem) a následně ručně osadit a zapájet stranu BOTTOM. Jako poslední pak osadíme THT součástky (vesměs konektory). Samozřejmě není problém celou desku zapájet ručně pomocí mikro páječky. V takovém případě je nutné postupovat pečlivě, aby nedošlo k nějakým zkratům nebo nekvalitním spojům (studeňáky). Při osazování doporučuji nejprve osadit mikropočítač a pak od nejmenších po největší součástky na straně TOP. Následně stejným postupem osadíme i stranu BOTTOM.
Po kompletním osazení doporučuji vše zkontrolovat.

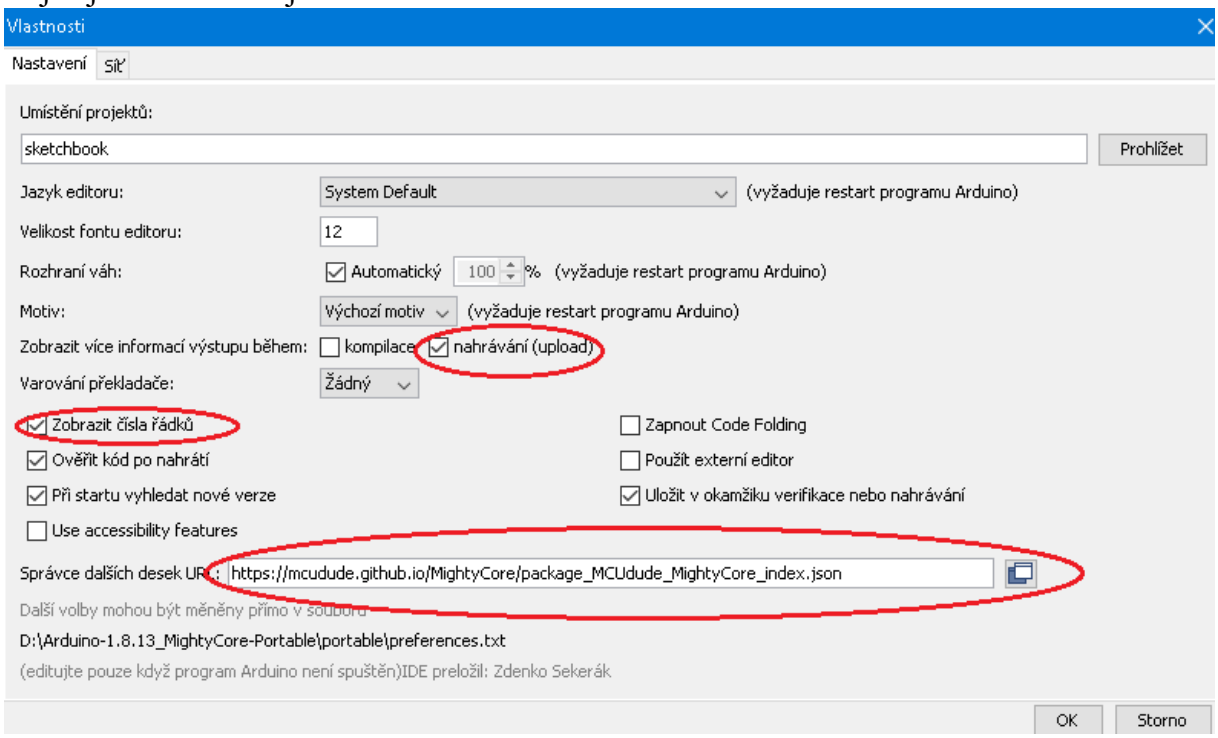
Oživení:
Pokud je vše správně osazeno a nenašli jsme při kontrole žádné problémy, můžeme přistoupit k oživení. Nejprve připojíme napájení pomocí USB kabelu. Na některý z pinů označených 5V a GND připojíme voltmetr a zkontrolujeme, že je přítomno správné napětí. Pokud je vše v pořádku, odpojíme USB a připojíme ideálně 12V z laboratorního zdroje na pin označený 12V a GND. Na zdroji nastavíme proudové omezení na 50mA (nebo nejmenší jaké zdroj umožňuje) a zapneme ho. Odebíraný proud by se měl pohybovat kolem 10mA. Pokud je výrazně vyšší znamenalo by to nějaký problém. Pokud je tedy odběr v pořádku, provedeme změření napětí 5V. Pokud je v pořádku změříme, že na pojistce F1 není žádné napětí.
Tímto máme provedenu první kontrolu a můžeme přistoupit k oživení mikropočítače.

MightyCore:
Než se pustíme do oživování mikropočítače, musíme si do vývojového prostředí Arduino IDE, doinstalovat podporu námi použitého mikropočítače. Osobně doporučuji použít Arduino IDE pokud možno v nejnovější verzi. Pro použití MightyCore je nutná minimálně verze 1.8.7. Já jsem již toto prostředí připravil za Vás a je součástí dokumentace. Stačí ho rozbalit ideálně programem 7-zip, do libovolného umístění ve Vašem PC a můžeme ho hned začít používat. Nicméně pro úplnost popíšu i samotnou instalaci MightyCore.
V první řadě doporučuji ze stránky https://www.arduino.cc/ stáhnout nejnovější verzi Arduino IDE. Stahujte variantu: **Windows ZIP file for non admin install**. Po stažení archiv rozbalte do libovolného umístění ve Vašem PC. Ještě než vývojové prostředí spustíte souborem arduino.exe, vytvořte v adresáři, kde se nachází i tento exe soubor, novou složku pojmenovanou **portable** (viz následující obrázek).

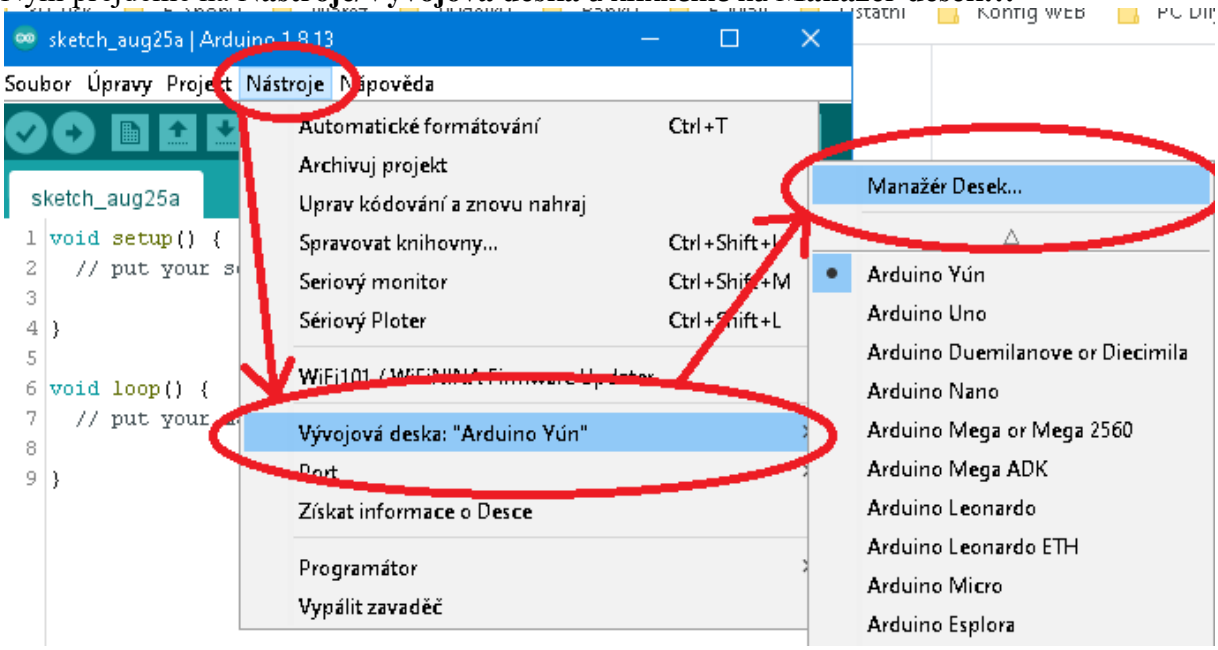


Vytvoření této složky způsobí, že si Arduino IDE nebude ukládat dodatečné soubory kamsi do Vašeho uživatelského profilu, ale právě do této složky. To Vám umožní nahrát toto prostředí například na Flash Disk a přenášet ho mezi různými počítači a mít všechny Vaše projekty stále uložené jen na tomto Flash Disku.
Jakmile je složka portable vytvořena, můžeme spustit Arduino IDE souborem arduino.exe. Předpokládám, že jste již někdy Arduino IDE používali, takže doporučuji nejprve aktualizovat všechny definice desek a knihoven. Jakmile jsou všechny aktualizace hotové, přejdeme na kartu **Vlastnosti** (Soubor/ Vlastnosti), kde si můžeme například zapnout zobrazení čísla řádků

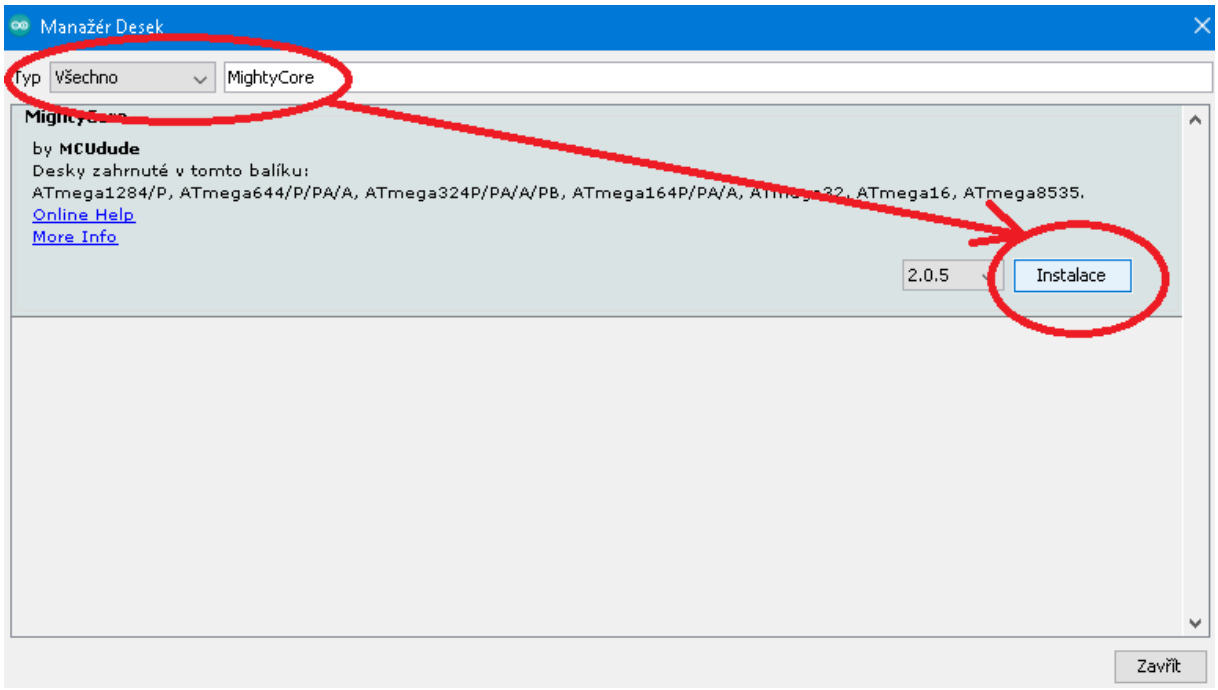
programu, zobrazení podrobnějšího výpisu při nahrávání projektu a to nejdůležitější, do řádku **Správce dalších desek URL**: zadáme následující odkaz:
https://mcudude.github.io/MightyCore/package_MCUdude_MightyCore_index.json
Stejně jako na následujícím obrázku.



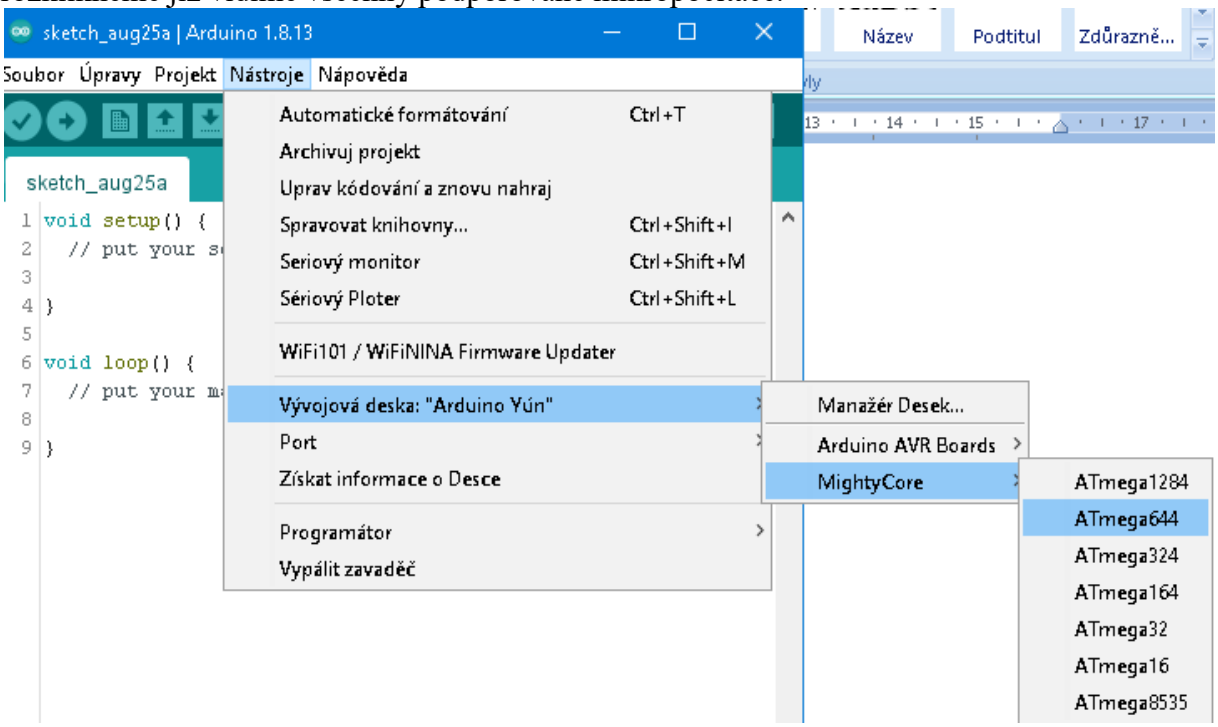
Vše potvrdíme OK.
Nyní přejdeme na **Nástroje/Vývojová deska** a klikneme na **Manažér desek…**



Otevře se nová karta, kde můžeme zpravovat (instalovat, aktualizovat nebo odinstalovat) podporu jednotlivých desek nebo přesněni mikropočítačů. V poli **Typ** vybereme **Všechno** a do pole hledání hned vedle napíšeme **MightyCore**. Automaticky bude vyhledán balíček obsahující námi preferované mikropočítače. Nyní již stačí kliknout na tlačítko **Instalace** a počkat až se vše stáhne a nainstaluje.
Ukázka jak to má vypadat je opět na následujícím obrázku.



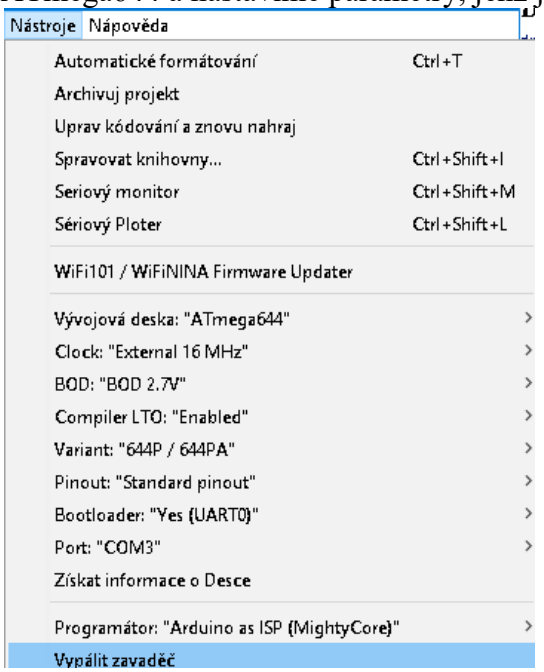
Následně klikneme na **Zavřít**.
Nyní se nám již v **Nástroje/Vývojová deska** objevila další položka **MightyCore**, kterou když rozklikneme již vidíme všechny podporované mikropočítače.



Vybereme tedy mikropočítač, jenž máme osazen v Sakulno NANO (nejspíš tedy ATmega644).

Nastavení pojistek a nahrání Bootloaderu:
Jakmile máme doinstalovánu podporu **MightyCore**, je nutné nastavit správné parametry a nahrát do mikropočítače bootloader. A to i v případě, že bychom bootloader nehodlali používat. V momentě, když dáte nahrát bootloader jsou současně nastaveny i pojistky. Proto při první inicializaci mikropočítače ho musíme vždy nahrát.

Opět si tedy rozklikneme nabídku **Nástroje**, kde máme od minula vybrán mikropočítač ATmega644 a nastavíme parametry, jenž jsou na následujícím obrázku.



Clock: Zde by mělo být již přednastaveno na **External 16MHz**. V Sakulnu je taktéž použit krystal 16MHz, takže tuto položku máme dobře a můžeme se posunout k další. Nicméně pokud si tuto položku rozklikneme vidíme všechny podporované varianty oscilátorů a jejich frekvencí. Pro bližší informace doporučuji prostudovat dokumentaci k **MightyCore**.

BOD: Touto položkou se nastavuje minimální napětí mikropočítače, než se řízeně vypne (Brown out detection). Každá část mikropočítače vyžaduje nějaké minimální napětí, aby mohla správně pracovat. Pokud pak napájecí napětí klesne pod tuto hodnotu, může dojít k nestabilitě. V naprosté většině vystačíme s nastavením, jež je defaultní a to **2,7V**. Opět pro více informací si prostudujte ideálně datasheet daného mikropočítače.

Compiler LTO: Zde doporučuji přepnout na **Enabled**, což může někdy docela významně zmenšit velikost obsazené paměti mikropočítače. Nicméně aby tato funkce fungovala je nutné vývojové prostředí Arduino IDE ve verzi 1.6.11 nebo novější. Na starších verzích zapnutí této volby vrátí chybové hlášení při kompilaci.

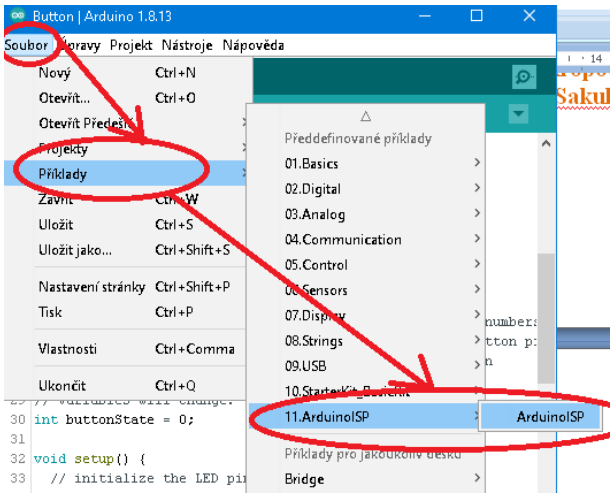
Variant: Zde je nutné vybrat přesnou variantu použitého mikropočítače. Toto označení je přímo na pouzdře součástky. Například v použitém prototypu Sakulno NANO 644 je použit mikropočítač **ATmega644PA**. Pokud zadáte špatnou variantu, tak při nahrávání firmware nebo bootloaderu obdržíte chybové hlášení, že nesouhlasí signatura (ID) připojeného čipu.

Pinout: Já používám **Standard pinout**, ale můžete použít některý jiný. Toto nastavení je velice důležité pokud používáte při definování hardware ve Vašem projektu pouze čísla. Typicky, například vývod mikropočítače, jež je běžně u Arduino desek vyveden na signalizační LED (D13). Zápis v programu pak může být například takovíto: **const int ledPin = 13;** Zde právě ta 13ka určuje, o jaký vývod mikropočítač jde. Jenže definice jen čísla 13 je naprosto nic neříkající z pohledu mikropočítače. Proto musí vývojové prostředí správně přiřadit při kompilaci danému číslu správný vývod mikropočítače. Z toho důvodu existuje tabulka pinout, která říká který vývod mikropočítače má jaké číslo. **MightyCore** umožňuje definovat daný vývod mikropočítače i jeho skutečným jménem. Například **PIN_PB0** odpovídá v **Standard pinoutu** číslu **0**. Nicméně například v **Bobuino pinoutu** je to číslo **4**. Opět pro bližší info doporučuji prostudovat dokumentaci **MightyCore**.

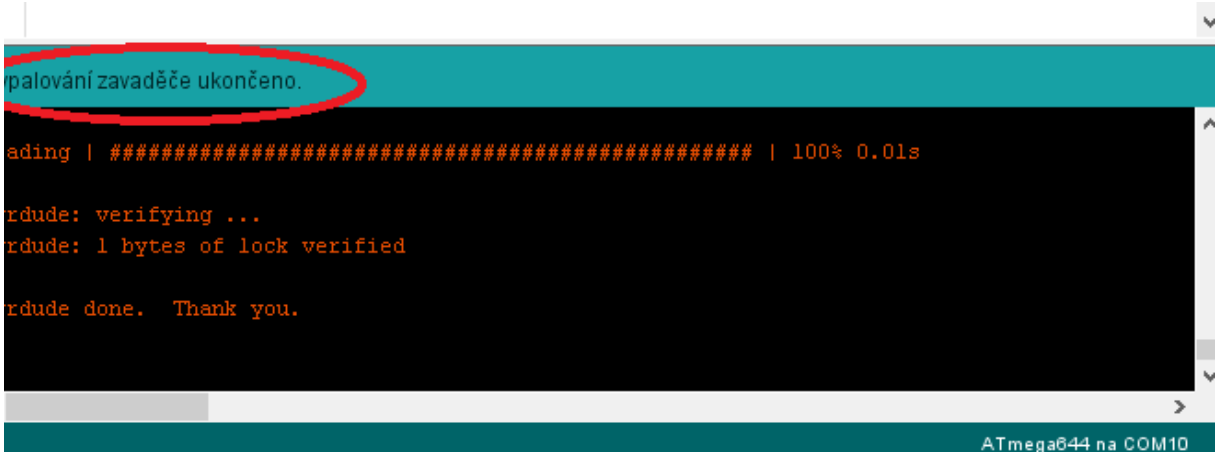
Bootloader: Tato položka nastavuje, zda bude současně s pojistkami do mikropočítače nahrán i bootloader, který umožňuje následné nahrání firmware pomocí USB nebo zvoleného portu UART bez použití programátoru. V případě Sakulna NANO je nutné nastavit tuto volbu na **Yes (UART0)**. Je to z toho důvodu, že USB/TTL převodník je fyzicky připojen právě k portu UART0. Pokud bychom bootloader nehodlaly používat s tím, že budeme do mikropočítače nahrávat program pouze programátorem, zadáme volbu **No Bootloader**. V takovém případě se do mikropočítače nahraje pouze konfigurace pojistek.

Port: Tak toto již všichni jistě znáte. Zde se nastavuje ComPort, kterým Vaše PC komunikuje s deskou Sakulno NANO. **Nicméně nyní, když děláme první inicializaci mikropočítače se nastavení tohoto portu týká připojeného ISP programátoru, nikoli desky Sakulno NANO.**

Výběr programátoru ISP:
Tímto plynule přecházíme k volbě programátoru, kterým nahrajeme do mikropočítače



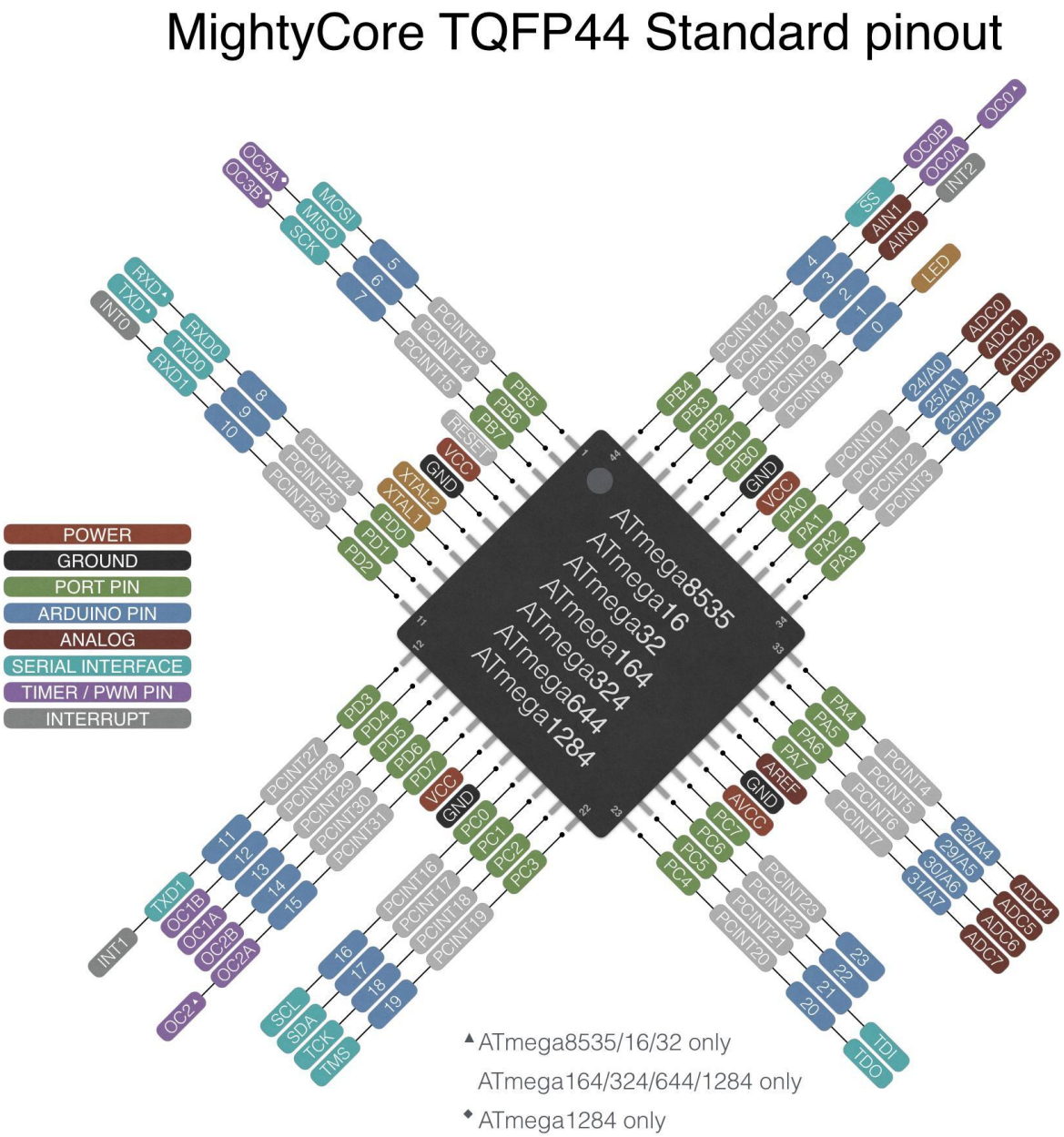
bootloader a nastavení pojistek. Já s oblibou používám desku Arduino NANO, které jsem přehrál bootloaderem od desky UNO a následně do něj nahrál firmware z nabídky **Soubor/Příklady/ArduinoISP**. Tím se z běžné desky Arduino stane programátor. To proč jsem přehrával bootloader z desky UNO do NANO je proto, že pokud jsem měl v desce bootloader od desky NANO nechtěl mi potom firmware **ArduinoISP** fungovat správně. Takže asi doporučuji, si programátor vyrobit přímo z desky UNO. To jak se pak propojí deska UNO s deskou Sakulno NANO (konektor JP3) je popsáno přímo ve firmware **ArduinoISP**. Normálně propojíte vývody MOSI, MISO, SCK, 5V a GND z desky UNO na Sakulno NANO + Reset na desce Sakulno NANO připojíte na pin D10 na desce UNO. Pokud i přesto nevíte, zkuste YouTube, kde je těchto návodů jak udělat z Arduina UNO ISP programátor asi miliarda.
Nicméně, samozřejmě pokud máte nějaký jiný podporovaný programátor, můžete použít ten. Já to však pro jednoduchost popíšu za použití programátoru Arduino as ISP.
Vybereme tedy:
Programátor: Arduino as ISP(MightyCore). A pokud máte programátor propojen s deskou Sakulno NANO a současně programátor připojen k PC, vybereme v Arduino IDE daný ComPort Vašeho ISP programátoru a můžeme kliknout na poslední položku **Vypálit Zavaděč**. Pokud je vše v pořádku, během několika vteřin je do nového mikropočítače v desce Sakulno NANO nahrán bootloader včetně konfigurace pojistek. To, že vše proběhlo správně, vidíme ve stavovém hlášení vývojového prostředí a současně by se měla na desce Sakulno NANO rozblikat LED1.



Ukázka stavového hlášení po úspěšném nahrání bootloaderu.

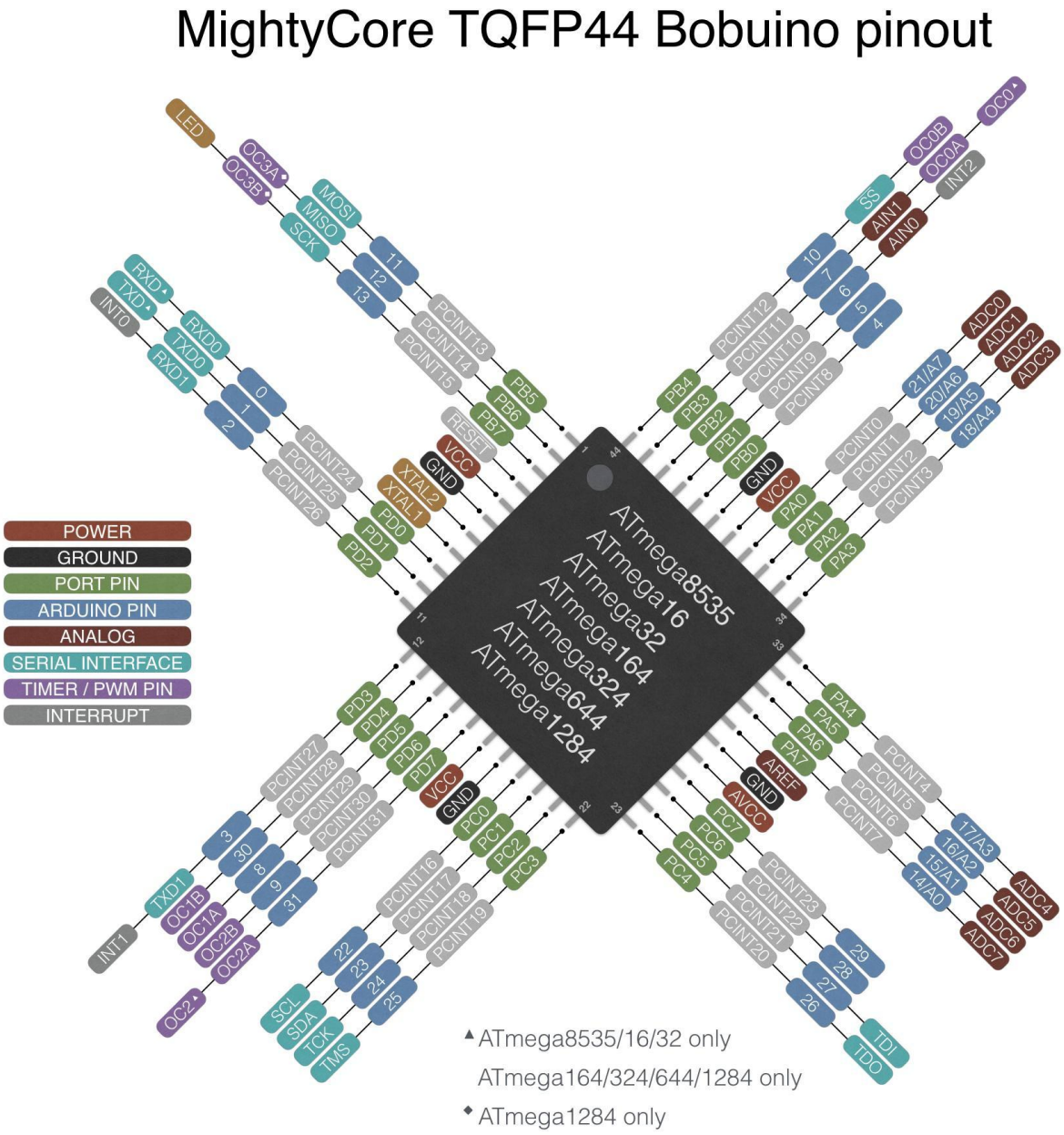
V tuto chvíli se z desky Sakulno NANO stává jakákoli běžná Arduino deska a již ji můžeme programovat pouze za pomocí připojeného USB kabelu.

Jednotlivé pinouty:

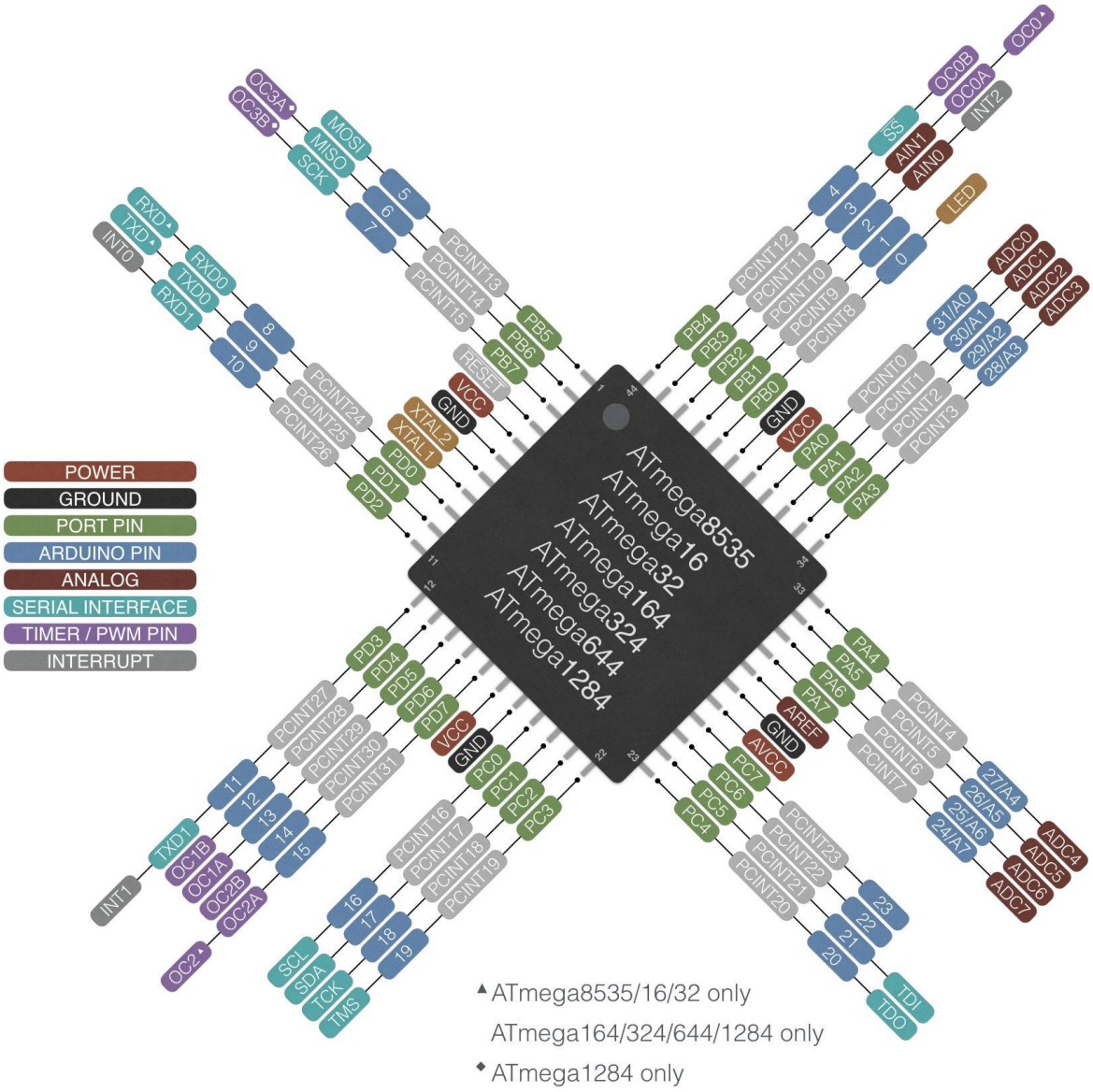


http://github.com/MCUdude/MightyCore

Tento pinout v plném rozlišení najdete v elektronické dokumentaci.



Tento pinout v plném rozlišení najdete v elektronické dokumentaci.

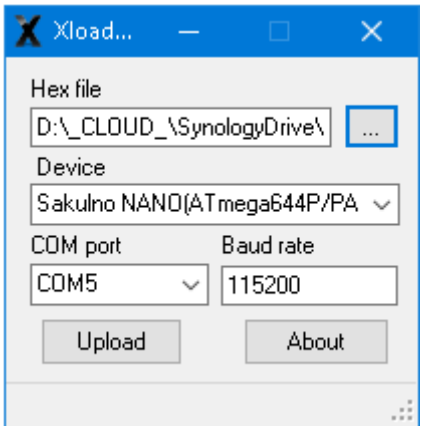


http://github.com/MCUdude/MightyCore

Tento pinout v plném rozlišení najdete v elektronické dokumentaci.

Použití Xloaderu:

Program Xloader slouží primárně k nahrání firmware ve formátu HEX (tedy již zkompilovaného) do desek Arduino. Bohužel v defaultní konfiguraci nepodporuje námi použitý mikropočítač ATmega644PA. Ovšem je celkem snadné do něho přidat tuto podporu. Takže jsem do seznamu podporovaných desek přidal i **Sakulno NANO** a to hned ve dvou variantách lišících se použitým mikropočítačem. První varianta by měla podporovat mikropočítače ATmega**644** a **644A**. Druhá pak varianty ATmega644**P** a **PA**. Takže tento program s upravenými deskami Sakulno NANO najdete v elektronické dokumentaci. Já ho používám celkem rád pro jeho jednoduchost, hlavně u projektů, kde potřebuji možnost uživatelského updatu firmware v případě, že vydám novější verzi. Tímto prográmkem dokáže pak firmware v daném zařízení aktualizovat i běžný uživatel bez toho, aby musel firmware složitě kompilovat.



Řešení problémů:

Při uvádění do provozu může nastat nespočet problémů, a proto bych tu některé chtěl zmínit.

1. **Problém s resetem** procesoru při nahrávání firmware přes USB. S tímto problémem jsem se setkal po začátku, kdy jsem začal používat mikropočítače z **MightyCore**. Já v mých konstrukcích používám většinou USB/TTL převodních CH340G, který provádí reset pomocí vývodu DTR. Tento vývod je v běžném provozu ve stavu **H**. V momentě, kdy dochází k uploadu firmware, přejde do stavu **L**. Bohužel v tomto stavu je po celou dobu uploadu. To by však celou dobu drželo mikropočítač v resetu (vypnutém stavu). Proto se zařazuje mezi vývod DTR a Reset programované součástky (mikropočítače) kondenzátor, většinou s hodnotou 100nF. To způsobí pouze krátký přechod resetu mikropočítače do stavu L, aby mohl být spuštěn bootloader, který následně aktualizuje nový firmware. Jak se ukázalo, tak tato hodnota je pro použití s mikropočítači ATmega644 a ATmega1284 (jiné jsem dosud netestoval) nedostačující. Při testech jsem zjistil, že kapacita kondenzátoru musí být minimálně 220nF, ale ideálně 330nF. Proto jsem ve finále použil tento (ve schématu je to C8) kondenzátor s kapacitou 1uF. Používám běžný keramický kondenzátor v pouzdře SMD1206 s maximálním napětím 25V.
2. **Při první inicializaci mikropočítače (nahrání bootloaderu a konfigurace pojistek) obdržíme hlášení, že připojená součástka má špatné ID.** Toto je celkem běžný problém. Ať už při nahrávání bootloaderu nebo následně již firmware se totiž vždy před samotným uploadem zkontroluje, zda připojená součástka (mikropočítač) odpovídá součástce, pro kterou byl firmware zkompilován (pokud ho nahráváme přes Arduino IDE). No, a pokud nesouhlasí předpokládané ID s ID přečteným z programované součástky, upload se nezdaří. Toto může mít nejčastěji 3 příčiny. Buď je nastavena v Arduino IDE špatná součástka (typicky třeba místo ATmega644**PA** je nastaveno ATmega644) než je skutečně ta připojená. V takovém případě je potřeba zkontrolovat a nastavit správnou součástku. Druhou možností je, že ID přečtené z programované součástky je zcela neplatné, nebo patří zcela jiné součástce. Zde je už potřeba zbystřit, protože toto už může být celkem zásadní problém, který může mít hned několik příčin. V první řadě je nutné zkontrolovat veškerou kabeláž, zda vše správně převádí a současně zkontrolovat, že napětí programované součástky je po celou dobu v provozním rozsahu. V případě, že je vše napájeno jen z USB počítače a

to třeba přes delší kabel nevalné kvality, může na něm být celkem zásadní úbytek napětí (klidně 2-3V) což způsobí nestabilitu programované součástky. Proto pokud máte tento problém, připojte ještě nějaké externí napájecí napětí. To ve většině případů vyřeší většinu problémů. No a poslední třetí možností je, že programovaná součástka vůbec nevrací žádné ID (respektive vidíme samé nuly). Pokud to nemá příčinu v nedostatečném napájení, je nejspíše zapojena špatně kabeláž a proto nemohla být navázána komunikace s programovanou součástkou.

Seznam použitých komponent:

Part	Value	Device	Package
C1	22pF	C-EUC1206	C1206
C2	22pF	C-EUC1206	C1206
C3	22pF	C-EUC1206	C1206
C4	22pF	C-EUC1206	C1206
C5	100nF	C-EUC1206	C1206
C6	100nF	C-EUC1206	C1206
C7	100nF	C-EUC1206	C1206
C8	1uF/25V	C-EUC1206	C1206
C9	100nF	C-EUC1206	C1206
C10	100nF	C-EUC1206	C1206
C11	10uF/10V	C-EUC1206	C1206
C12	47uF/16V	C-EUC1206	C1206
CN1	10033526-n3212mlf/	USBMINIB	USB-MINIB
D1	1N4148W-LIT	DIODE_SOD-123FL	SOD-123FL
D2	5V6	ZENER-DIODESOD80C	SOD80C
D3	US1M-TP	DIODE-DO214AC	DO214AC
F1	0ZCJ0050FF2G	PTCFUSE-1206	R1206
IC1	ATMEGA644PA	ATMEGA644	TQFP44
JP1	Kolíková lišta 1x20	PINHD-1X20-BIG	1X20-BIG
JP2	Kolíková lišta 1x20	PINHD-1X20-BIG	1X20-BIG
JP3	Kolíková lišta 2x4	PINHD-2X4	2X04
LED1	Green	LEDCHIPLED_1206	CHIPLED_1206
LED2	Yellow	LEDCHIPLED_1206	CHIPLED_1206
LED3	Red	LEDCHIPLED_1206	CHIPLED_1206
Q1	16MHz	CRYSTALSM49	SM49
Q2	12MHz	CRYSTALSM49	SM49
Q3	FDN340P	SMD-MOSFET-P	(SOT-23) SOT-23
Q4	FDN340P	SMD-MOSFET-P	(SOT-23) SOT-23
R1	10K	R-EU_R1206	R1206
R2	1K	R-EU_R1206	R1206
R3	10R	R-EU_R1206	R1206
R4	1K	R-EU_R1206	R1206
R5	4K7	R-EU_R1206	R1206
R6	1K	R-EU_R1206	R1206
R7	1K	R-EU_R1206	R1206
R8	330R	R-EU_R1206	R1206
R9	330R	R-EU_R1206	R1206
SW1	1301.9314	SPST_TACT-EVQQ2 SPST	
U$1	CH340G		SOIC16
U1	LM1117IMP-5.0	V_REG_LM1117SOT223	SOT223

Závěrečné prohlášení:
Autor této konstrukce se zříká jakékoli odpovědnosti za chování této konstrukce a jakékoli škody, která může vzniknout použitím této konstrukce. Veškerou odpovědnost přebírá provozovatel zařízení.

Co znamená (EA / FINAL) v nadpisu konstrukce:
Jde o zkratku **Early Access** neboli předběžný přístup. Většina mých projektů začíná fází **předběžného přístupu**, kdy je daná konstrukce uvolněna (zveřejněna), ale stále nejde o finální provedení. Některé funkce nemusí být ještě integrovány, případně se v konstrukci mohou vyskytovat chyby. Nicméně již jde o použitelnou konstrukci, která se dále vyvíjí a zdokonaluje. V momentě, kdy uznám, že je již vše funkční a odladěné, přechází konstrukce do **Finální** fáze (označeno jako FINAL). Předem upozorňuji, že konstrukce zveřejněné v režimu EA nemusí nikdy přejít do verze FINAL a nelze reklamovat jejich funkcionalitu.
Tím, že si tuto konstrukci pořídíte, zároveň stvrzujete, že jste seznámeni s aktuální funkcionalitou a případnými chybami, jež může konstrukce obsahovat a akceptujete je.

Technická podpora:
Veškerá podpora pro tuto konstrukci je řešena výhradně formou diskuse. Proto pokud máte jakýkoli dotaz týkající se této konstrukce, obracejte se do fóra:
https://forum.sakul.cz/viewtopic.php?f=10&t=48
https://forum.sakul.cz/viewtopic.php?f=10&t=1184

Donate (příspěvek/dar):
Pokud Vám tento manuál pomohl, zvažte možnost příspěvku libovolné částky. Právě díky třeba Vašemu příspěvku bude moci vzniknout nějaký další manuál nebo celá konstrukce.
Přispět je možno na bankovní účet: 670100-2208863541/6210
Nebo na PayPal: SakulRaider@seznam.cz

Zajímavé odkazy:
Můj Patreon - https://www.patreon.com/sakul
Sakul WORLD - https://www.sakul.cz/
Sakul Fórum - https://forum.sakul.cz/
Digispark: Malé Arduino - https://www.sakul.cz/digispark-male-arduino/n
GRBL Board - https://www.sakul.cz/grbl-board/n
Stopky pro hasiče - https://www.sakul.cz/stopky-pro-hasice-pe11-2011/n
Stopky pro hasiče v1.5 SMD - https://www.sakul.cz/stopky-pro-hasice-smd/n/
GPS hodiny - https://www.sakul.cz/gps-hodiny-v2-pe2-2015/n
Počítadlo YouTube odběratelů - https://www.patreon.com/posts/36304881
Velký displej nejen pro stopky - https://www.sakul.cz/velky-displej-nejen-pro-stopky/n