ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Embedded Linux – HW realizace

Hynek Šolar **2014**

Abstrakt

Předkládaná diplomová práce je zaměřena na programování pro zařízení s Embedded Linuxem

Klíčová slova

Embedded linux, Intel Galileo, GPIO, programování

Abstract

This master thesis presents is focusing on programming for devices with Embedded Linux

Key words

Embedded Linux, Intel Galileo, GPIO, programming

Prohlášení

Prohlašuji, že jsem tuto diplomovou/bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské/diplomové práce, je legální.

.....

podpis

V Plzni dne 8.5.2014 Jméno příjmení

Poděkování

Tímto bych rád poděkoval vedoucímu diplomové práce Ing. Petru Weissarovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Seznam obrázků

Seznam symbolů a zkratek

GPIO..... General purpose input-output

SOC..... System on a chip

PWM..... Pulzně šířková modulace

OS..... Operační systém

ADC..... Analogově – číslicový převodník

UART..... Universal asynchronous reciever-transmitter, sběrnice

SPI..... Serial peripherial interface, sběrnice

I2C..... Inter integrated circuit, sběrnice

Úvod

Předkládaná práce je zaměřena na programování pro hardware používající embedded linux. Ukázky programů jsou vytvořeny pro platformu Intel Galileo, ale jsou snadno upravitelné pro použití na jiných platformách.

Text je rozdělen do tří částí; první se zabývá platformou Intel Galileo a jejími možnostmi. Druhá popisuje vzniklý kód a popisuje principy programování pro zmíněnou platformu. Třetí část obsahuje návody pro práci s platformou a použití knihovny jež vznikla v rámci této práce.

Zařízení s Embedded Linuxem

Jedná se o zařízení jednodušší a především úspornější nežli klasické počítače, přesto na rozdíl od jednočipových mikropočítačů schopné běhu pod operačním systémem a nabízející výhody z toho plynoucí.

Práce byla tvořena na kitu Intel Galileo, proto její velká část pojednává o tomto kitu. Většina faktů je ale platná i pro jiná zařízení sdílející stejnou filosofii (např. Raspberry Pi)

1.1 Intel Galileo

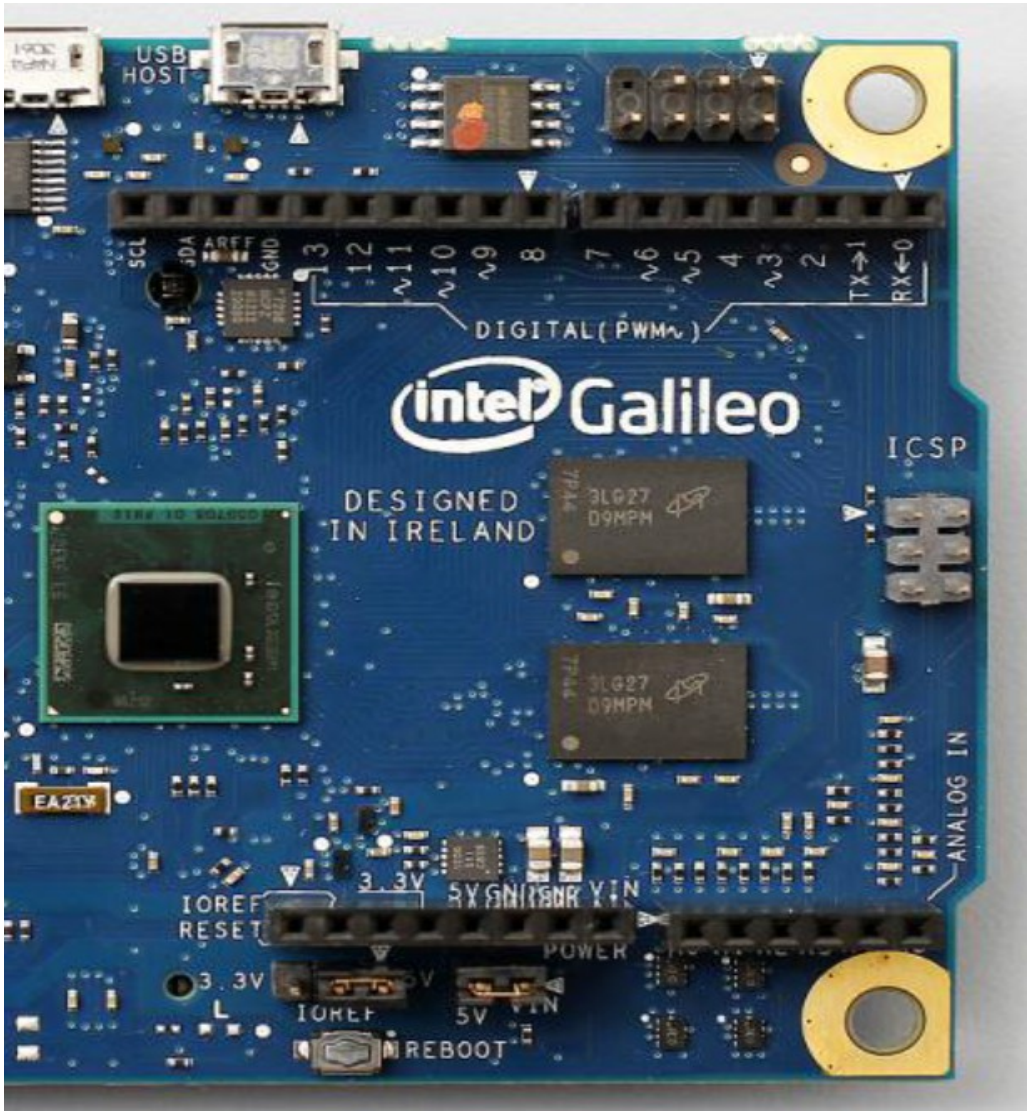
Je vývojový kit vzniklý ze spolupráce firem Intel a Arduino. Vše je postaveno kolem procesoru Intel Quark SOC X1000. Procesor je postaven na x86 architektuře.

Zařízení je fyzicky navrženo tak, aby bylo kompatibilní s přídavnými deskami pro jiné Arduino kity (zařízení používají stejné rozvržení pinů). Jelikož se ale jedná o GPIO, lze zde připojit libovolný jiný hardware a využívat vlastností pinů zde vyvedených. (Dle dokumentace lze pin zatížit až 10mA při pracovním napětí 3,3 nebo 5V)

1.2 Procesor Intel Quark SOC X1000 [1]

Jedná se o 32-bitový system on a chip jednojádrový procesor třídy Intel Pentium s následujícími vlastnostmi:

- ***Pracovní frekvence 400Mhz (lze snížit pro snížení spotřeby)***
- ***16KB L1 cache paměti***
- ***Podpora ACPI sleep modů***
- ***RTC hodiny (vyžaduje připojení 3V baterie)***
- ***10/100 Ethernet port***
- ***mini-PCIe* 2.0 port***
- ***USB 2.0 Host port s možností připojení až 128 koncových zařízení***
- ***USB 2.0 Client port (lze použít k programování kitu)***
- ***SDIO port pro připojení externí paměti***
- ***I2C, UART a SPI sběrnice***
- ***výstup pro JTAG***

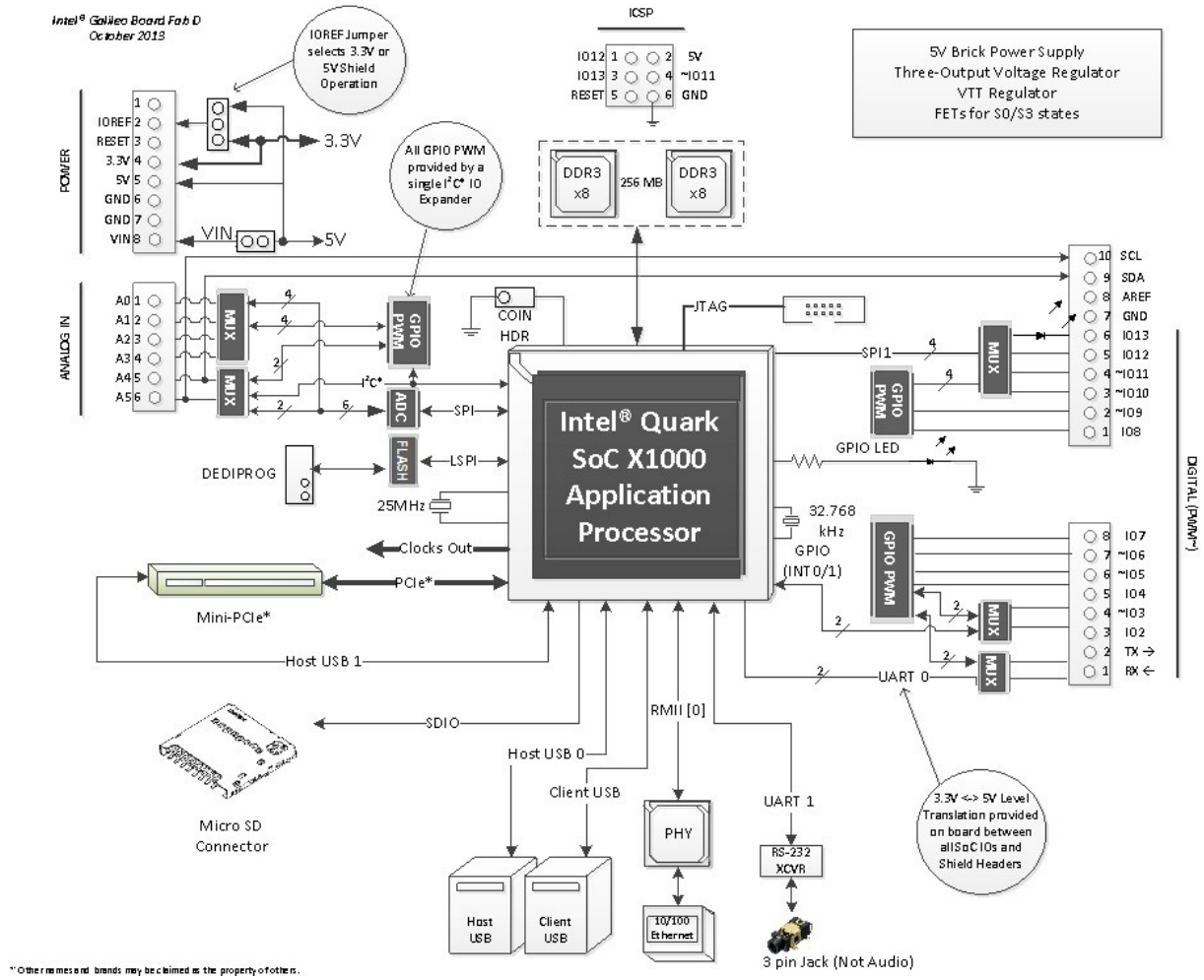


Obr. 1: Detail na Arduino patici a procesor [1]

1.3 Vlastnosti kitu Intel Galileo [1]

Kit nejen přivádí jednotlivé vstupy a výstupy na patřičné konektory, ale také přidává nětkeré další parametry:

- ***256MB DRAM paměti***
- ***8MB flash paměti***
- ***analogově – číslicový převodník***
- ***IO expandér s možností PWM***



Obr. 2: Periferie připojené k procesoru

1.4 Vlastnosti a možnosti GPIO kitu Intel Galileo

Jak již bylo řečeno, zařízení je kompatibilní s jinými Arduino zařízeními, z čehož vyplývají vlastnosti GPIO konektoru. Konkrétně je zde:

- ***14 digitálních vstupů/výstupů (z toho 6 s možností PWM)***
- ***6 analogových vstupů***
- ***2 piny pro I2C***
- ***2 piny pro UART (sdílející piny s digital 0 a 1)***
- ***4 piny pro SPI (sdílející piny s digital 10 – 13)***
- ***napájecí napětí a signál reset***

Všechny GPIO piny jsou připojeny přes IO expandér Cypres CY8C9540A. Tím jsou také dány proudové a napěťové možnosti pinů – 3,3 nebo 5V pracovní režim (volený

jumperem na kitu) a výstupní proud do 10mA

Některé piny v sobě skrývají více vlastností. Tyto jsou vybírány nastavováním multiplexerů v IO expandéru.

Kit lze používat buď s operačním systémem předpřipraveným v paměti na desce (byť tento má do jisté míry omezené možnosti – např. absence SSH serveru) nebo lze pracovat s operačním systémem na paměťové kartě (viz později). Na stránkách kitu je k dispozici verze 0.7.5 založená na Linuxové distribuci Yocto. Existují také oficiální nepodporované distribuce, ať už vyšší verze oficiálního systému nebo například YAD4galileo1 založený na Linuxivé distribuci Debian (tato práce byla vytvářena ponejvíce s použitím tohoto systému).

1.5 Srovnání s dalšími kity s možností použití Embedded linuxu pro ovládání GPIO

Kity s embedded linuxem používají stejnou filozofii přístupu k GPIO. Proto by měly být programy mezi nimi přenositelné. Liší se však fyzicky v množství portů a jejich označení. Z tohoto důvodu je zapotřebí upravit tabulku s adresací jednotlivých portů a zkontrolovat si, jestli port disponuje danými vlastnostmi (v případě použití PWM, ADC..). S tímto na paměti lze využít jeden kód na více platformách.

Údaje zapisované do souborů jsou shodné.

Pro všechny tři srovnávané kity existuje velké množství shieldů, které dále rozšiřují jejich možnosti a dělají z nich takřka univerzální zařízení (v rámci omezení výkonu procesoru). Můžeme zde najít kity připojující rozličné ovládací prvky, kity se zobrazovači (od sedmisegmentovek po dotykové LCD), kity pro ovládání motorů, kity s rozhraním pro další sběrnice (CAN), univerzální vývojové kity (s nepájivým polem), kity pro bezdrátovou komunikaci a mnohé další.

Kit Intel Galileo oproti ostatním srovnávaným disponuje portem miniPCI express (lze použít například pro wi-fi) a hodinami reálného času (po připojení 3V baterie). Výhodou je také možnost pracovat na GPIO s napětím 3.3V nebo 5V.

Zvláštností kitu Intel Galileo je použití procesoru s x86 (CISC) architekturou. U zařízení tohoto typu (jako například u dvou srovnávaných kitů) se obvykle setkáváme s ARM procesory (pravděpodobně z důvodu lepší optimalizace spotřeby)

	Intel Galileo	**Raspberry Pi**	**Beaglebone Black**
Procesor	Intel Quark X1000	ARM1176JZF-S	AM335x ARM Cortex-A8
Pracovní frekvence	400Mhz	700Mhz	1Ghz
Paměť RAM	256MB	512MB	512MB
Grafický výstup	ne	HDMI k dispozici grafický procesor	HDMI k dispozici 3D akcelerátor
Počet univerzálních GPIO	20	21	65
Pracovní napětí GPIO	3.3 nebo 5V volba jumperem na kitu	3.3V	3.3V 1.8V pro ADC
PWM	6 kanálů	1 kanál	8kanálů
Analogové vstupy	6 kanálů (12bit) max 5V	ne	7 kanálů (12bit) max.1.8V
UART	3 x	1 x	4x
SPI	1 x	2 x	2 x
I2C	1 x	1 x	2 x
Integrovaná paměť programu	8MB	ne	2GB
Cena	<60 USD	35USD	45USD

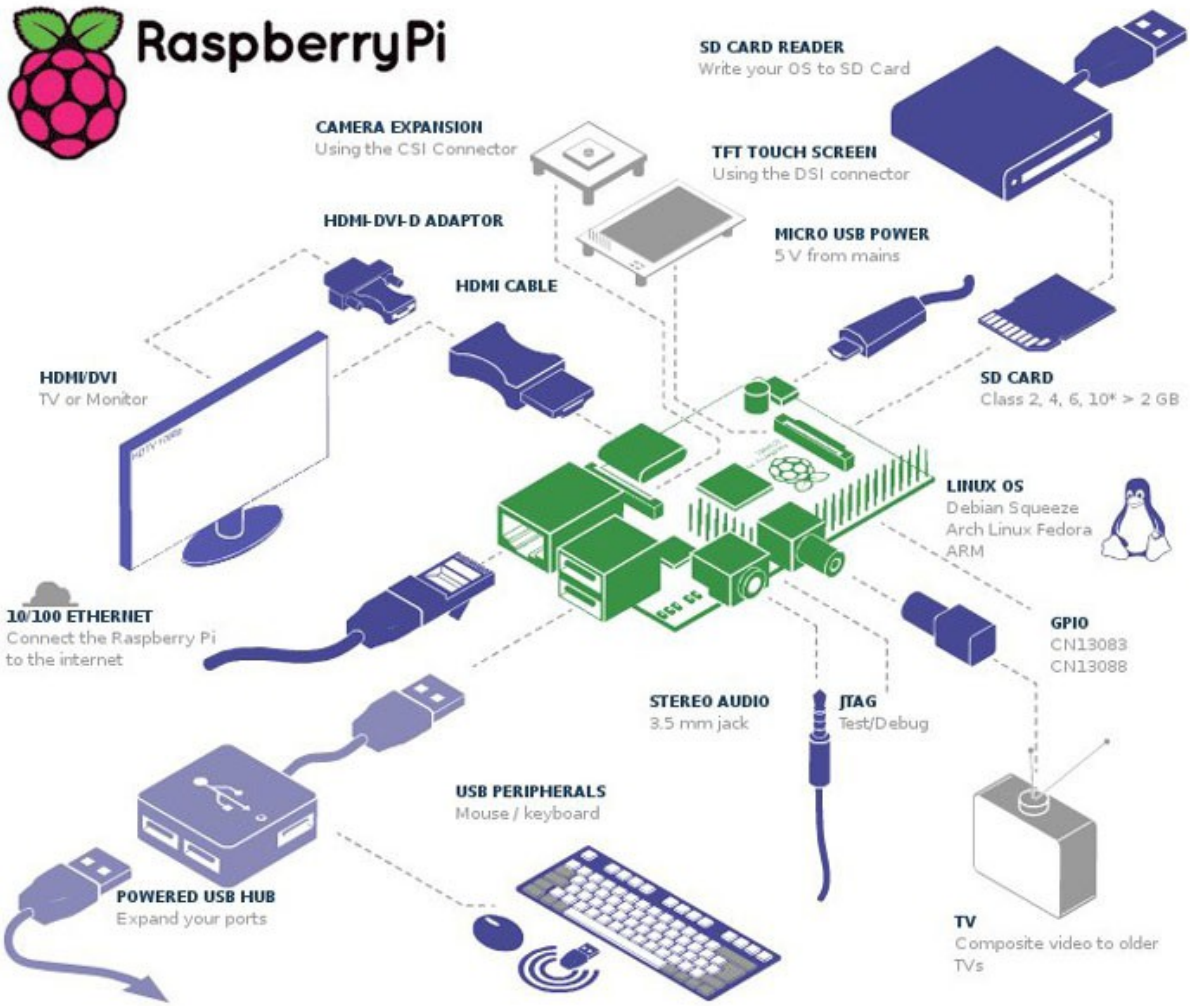
1.5.1 Raspberry Pi

Disponuje 26 piny ve dvouřadém konektoru plus dalšími osmi v druhém dvouřadém konektoru s rozestupy 2.54mm. GPIO pracuje na napětí 3.3V a při připojení 5V signálu může dojít k poškození.

Oproti kitu Intel Galileo tedy nabízí vyšší výkon a grafický výstup. Na druhou stranu je chudší, co se možností GPIO týče. Nevýhodou je též absence paměti přímo na kitu – před spuštěním je zapotřebí připravit si SD kartu s operačním systémem.

Kit Rasberry Pi je oproti kitu Intel Galileo vhodnější k multimediálním aplikacím a obecně aplikacím vyžadujícím grafický výstup – je slabší, co se týče GPIO, zato nabízí vyšší výpočetní a grafický výkon.

Kromě velkého množství vlastních periférií připojitelných skrze GPIO existuje pro Raspberry Pi také shield, který mu umožňuje používat Arduino shieldy.



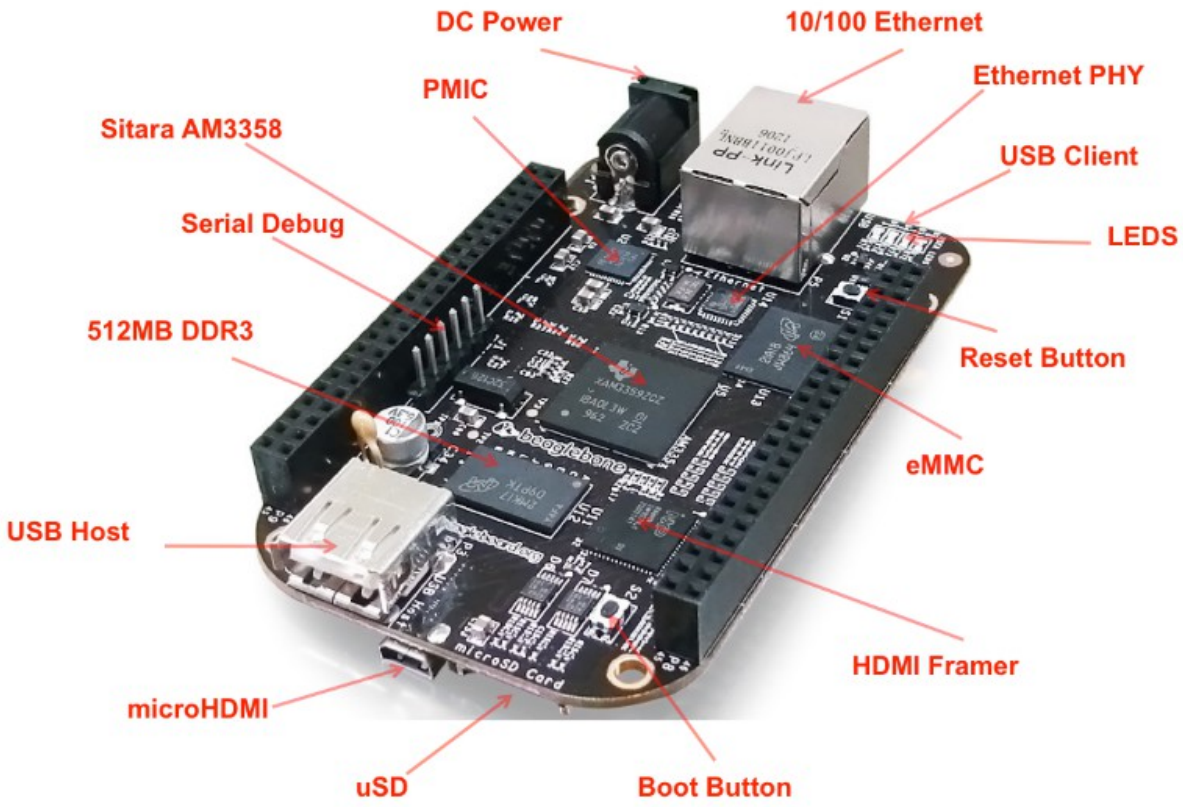
Obr. 3: Paspberry Pi a jeho dostupné porty [6]

1.5.2 BeagleBone Black

Nabízí až 65 GPIO. Některé z nich disponují alternativními funkcemi. GPIO jsou stejně jako u Raspberry Pi navrženy na napětí 3.3V

Z porovnávaných disponuje nejvyšším výkonem i největším množstvím GPIO a sběrnic. Je nejvhodnější pro aplikace s větším množství periférií. Grafický výkon je výrazně slabší nežli Raspberry Pi.

Existuje množství periférií připojitelných skrze GPIO, výrobce tyto nazývá „capes“ . Nedosahují však takového množství jako periferie pro Raspberry Pi



Obr. 4: Beaglebone Black a jeho komponenty [7]

Práce s kitem Intel Galileo

Existuje několik možností programování kitu: Jelikož nemá kit grafický výstup a jeho parametry nejsou dostatečné pro běh grafického prostředí, jeví se mi jako nejvhodnější řešení psát program na jiném počítači a v kitu jej kompilovat a/nebo pouze spouštět.

1.6 Použití Arduino IDE

Software vytvořený pro programování Arduino kitů. Program zde napsaný by měl být zkompilovatelný do libovolného Arduino zařízení (po jeho výběru z menu). Používá se zde jazyk wiring. Výhodou je jednoduchost a dostupnost množství knihoven a příkladů.

Verze softwaru použitá při této práci byla problematická – odmítala pracovat v jiném než anglickém nastavení operačního systému a to jak pod OS Windows 7 tak pod OS Linux Ubuntu.

Upload programu probíhá prostřednictvím USB Client portu – kit lze tedy připojit k počítači prostřednictvím standardního USB portu. Arduino IDE pak obsahuje instalaci softwaru pro emulaci sériového portu, který je pro komunikaci používán.

Arduino IDE také nabízí možnost aktualizace firmwaru v kitu. Tato je důležitá – verze v kitu není příliš optimální a nepodporuje mnohé z možností kitu.

1.7 Komunikace s Embedded Linuxem na kitu prostřednictvím ethernetu

Naskýtá se několik možností komunikace s kitem. Aktualizovaný firmware se dokáže připojit prostřednictvím běžného síťového routeru. K portům je pak možno přistupovat prostřednictvím zápisu/čtení souborů reprezentujících vlastnosti těchto portů (vstup/výstup, frekvence PWM, hodnota na ADC)

1.7.1 Použití protokolu telnet

Toto vyžaduje spuštění serveru pomocí Arduino IDE (viz str. 30). Pak se lze připojit s použitím libovolného klienta, např. programem Putty.

1.7.2 Použití protokolu SSH

Při použití operačního systému pro SD kartu je SSH server spuštěn již od začátku a stačí se jen připojit, opět lze použít program Putty. Při práci bez SD karty není SSH k dispozici.

1.7.3 Komunikace s prací v grafickém prostředí

V tomto případě je zapotřebí doinstalovat vnc server a xrdp. Pak lze použít například nástroj „Připojení ke vzdálené ploše“, jež je součástí OS Windows.

1.7.4 Přenos souborů prostřednictvím SCP

Vhodné například pokud je nový program napsán jinde a na kitu jen zkompilován a spuštěn

Ve všech těchto případech jsme připojeni k terminálu operačního systému na kitu. Toto nám nabízí několik možností práce s kitem:

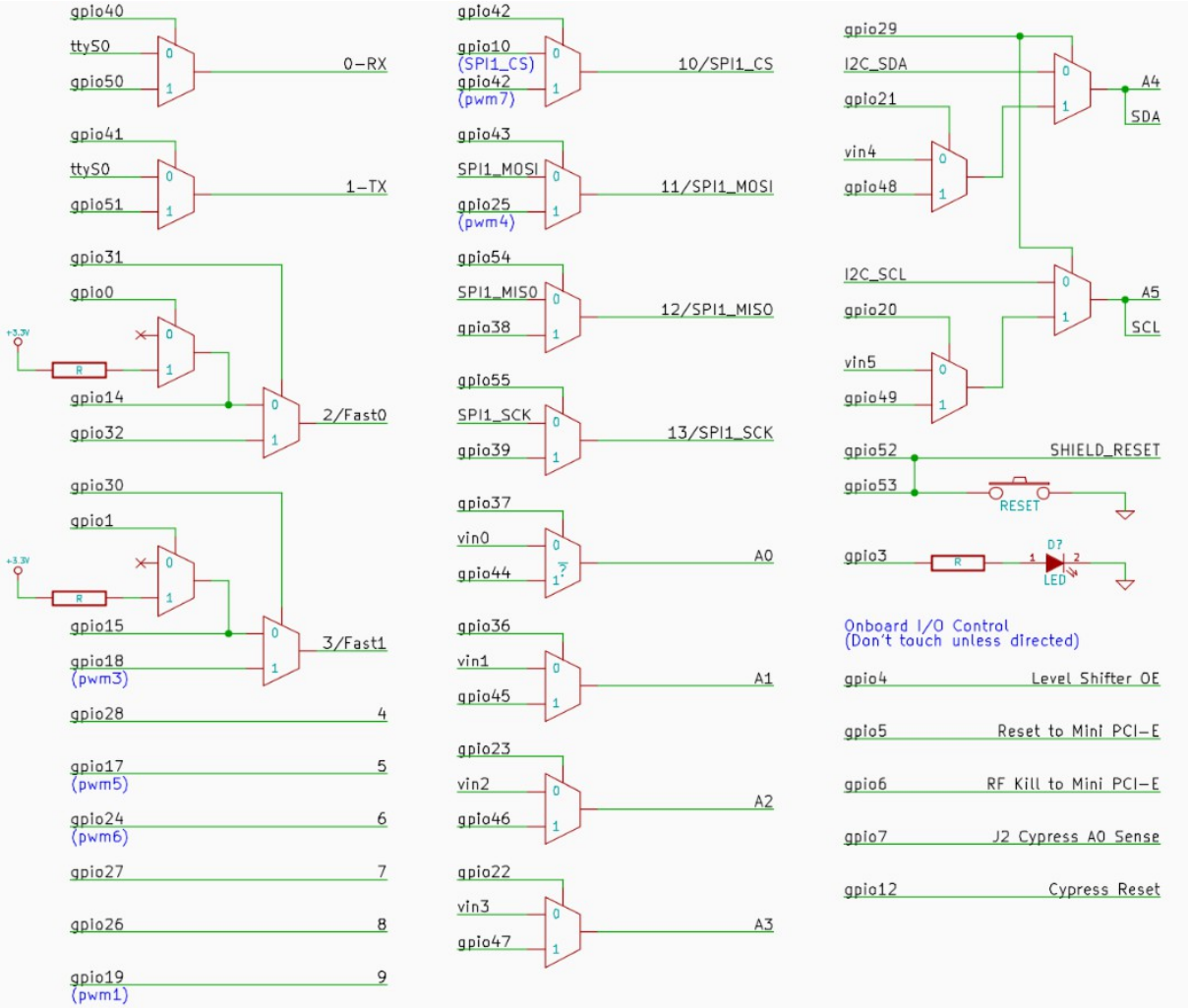
1.8 Způsoby ovládání GPIO z Embedded Linuxu

1.8.1 Přístup k GPIO

Každý z parametrů jednotlivých GPIO je reprezentován souborem a stav daného GPIO odpovídá datům v tomto souboru. Ve výchozím stavu však tyto soubory nejsou vytvořeny.

Zápisem čísla GPIO do souboru „export“ dojde k vytvoření složky se zmíněnými soubory pro daný GPIO.

Bohužel názvy GPIO na kitu neodpovídají názvům používaným systémem. Názvy jednotlivých GPIO jsou patrné z následujícího obrázku:



Obr. 5: Názvy GPIO [3]

1.8.2 Používání GPIO z terminálu

Je z možných způsobů nejjednodušší, ale také nenabízí žádně možnosti programování, jelikož každá změna / přečtení znamená zápis příkazu do terminálu. Příkazem *echo* můžeme přepsat hodnotu v souboru. Příkaz *cat* vypíše obsah souboru do terminálu. (Způsob použití bude vysvětlen později)

1.8.3 Vytvoření srkiptu

Skriptový soubor může obsahovat zmíněné příkazy pro terminál, čímž se zjednodušuje jejich zadávání. Stále se jedná spíše o sérii příkazů nežli o program. Soubor lze vytvořit v libovolném editoru a poté jej z terminálu spustit.

1.8.4 Program v jazyce C

Opět jej lze vytvořit v libovolném editoru. Kompilaci je lepší provádět na kitu příkazem *gcc nazev.c -o nazev* Vytvořený soubor pak spustíme příkazem *./nazev*

Obdobně jako u ovládání GPIO z konzole i zde zapisujeme/čteme soubory a tímto měníme nebo zjišťujeme stav jednotlivých vstupů/výstupů. Velkou výhodou je možnost zjednodušení zápisu díky funkcím, které zpracují opakující se části.

Program může obsahovat další části, které budou navázány nadata přijatá z portů (přepočet dat z ADC) nebo naopak měnit vstupy podle jiných, složitějších pravidel (periodické změny, změny v závislosti na stavu vstupů..)

1.8.5 Program v jazyce C s použitím modulu galileo_sh

V podstatě nabízí stejné možnosti jako předešlé, jen zápis je ještě více zjednodušen. Knihovna obsahuje funkce pro nastavení/čtení GPIO portů, kontroluje přístupnost zmíněných portů a převádí z názvů na kitu na názvy používané systémem.

Modul galileo_sh

Knihovna slouží k usnadnění práce s kitem Intel Galileo prostřednictvím programů psaných v jazyce C. Funkce této knihovny pokrývají běžné činnosti s GPIO porty a od uživatele vyžadují jen označení portu uvedené na kitu a parametr, který má být nastaven (výstup, log. „0“, perioda PWM...)

Dále budou rozebrány jednotlivé funkce knihovny a jejich vstupní parametry a návratové hodnoty.

int GPIO_prep(int gpio, char rw)

připraví GPIO k použití - ujisti se, ze je exportován (k použití), případně exportuje, nastaví směr (in[R]/out[W]). Do terminálu vypíše, pokud narazí na chybu nebo pokud exportuje GPIO.

Jako vstupní parametr slouží systémové číslo pinu a volba, zda má být vstupem nebo výstupem. Je-li pin nastaven jako výstup, vrací hodnotu 1, při nastavení na vstup vrací hodnotu 2. Jinak (tedy v případě chyby) vrací záporné číslo.

Tato funkce je využívána funkcí GPIO_open, stačí tedy použít jen té.

int GPIO_open(int gpio, char rw)

Funkce nastaví a otevře port(po té, co se ujistí, že je dostupný). Pokud narazí na chybu, předá informaci prostřednictvím terminálu.

Jako vstupní parametr slouží systémové číslo pinu a volba, zda má být vstupem nebo výstupem. Vrací file handler (typu integer) pro práci s portem. V případě chyby vrací záporné číslo.

void GPIO_write(int fHval, int val)

Zapíše jeden bit (tedy nulu nebo jedničku) do souboru daného hodnotou file handleru. Nemusí se jednat jen o změnu stavu GPIO, může jít například o přepnutí multiplexeru (např. pro připojení ADC) Při pokusu o zápis jiné hodnoty než 0 nebo 1 neudělá nic.

int gpio_read(int fHval)

Slouží ke přečtení jednoho bitu ze souboru daného hodnotou file handleru. Vrátí buďto hodnotu 0 nebo 1, případně -1 pokud soubor obsahuje něco jiného.

int GPIO_unexport(int gpio)

Navrátí GPIO do výchozího stavu, odstraní složku se soubory pojícími se k danému GPIO. Jako parametr slouží číslo portu.

int PWM_open(int pwm_port)

Připojí port pro práci s PWM. Zjistí, zda je port exportován, případně jej exportuje. Vypisuje na terminál informaci o exportu, případně o chybách.

Parametrem funkce je číslo GPIO portu. Funkce vrací jedničku v případě, že proběhne v pořádku. V opačném případě je navrácena hodnota mínus jedna.

int PWM_set(int pwm_port, int period, int duty)

Nastavuje parametry PWM výstupu pinu. Předpokladem je, že jsou dostupné soubory pro zápis – nejsou-li, funkce vypíše chybu na terminál.

Parametry funkce jsou číslo portu, doba trvání periody a doba sepnutí – jak dlouho po začátku periody je na pinu jednička. Proběhne-li v pořádku, funkce vrací jedničku. V opačném případě vrací hodnotu mínus jedna.

int PWM_unexport(int pwm_port)

Navrátí GPIO do výchozího stavu, odstraní složku se soubory pojícími se k danému GPIO. Jako parametr slouží číslo portu.

int ADC_open(int adc_port)

Připraví port k použití analogově – číslicového převodníku. Přepne multiplexer daného portu. Vrací mínus jedničku při chybě, jinak jedničku.

int ADC_read(int adc_port)

Zkontroluje dostupnost souboru s daty z převodníku. V případě jeho nenalezení vypíše chybu na terminál.

Parametrem je číslo portu. Vrací změřenou hodnotu v milivoltech nebo mínus jedničku v případě chyby.

int ADC_unexport(int adc_port)

Navrátí GPIO do výchozího stavu – přepne multiplexer zpět do výchozího stavu . Jako parametr slouží číslo portu.

int port_tab(int aport)

Tabulka pro dohledání systémového čísla pinu. Vstupním parametrem je číslo pinu na kitu.

int mux_tab(int port)

Tabulka pro dohledání systémového čísla pinu, který ovládá multiplexer před AD převodníkem. Vstupním parametrem je číslo analogového vstupu uvedené na kitu (bez úvodního písmene A)

Vzorový program ukazka.c

```
#include "galileo_sh.h"

int main(void){

	int pwm_channel = 11;			//vyber portu kde bude PWM   tedy pin 11
	int pwm_duty = 30;			// procenta periody
	int pwm_duty2 = 50;
	int pwn_cycle = 10;			//perioda 10ms

	int adc_port = 0;			//vyber portu pro adc
	int gpio = port_tab(13);		//vyber portu pro gpio
	int led = 3;				//led umistena na kitu ma cislo portu 3
	int fH1, fH2;				//file handlery pro praci s gpio

	fH1 = GPIO_open(gpio,'W');		//priprava gpio k zapisu
	fH2 = GPIO_open(led, 'W');

	GPIO_write(fH1, 1);			//zapis do gpio
	GPIO_write(fH2, 0);
	usleep(500000);				//cekani v mikrosekundach
	GPIO_write(fH1, 0);
	GPIO_write(fH2, 1);
	usleep(500000);
	GPIO_write(fH1, 1);
	GPIO_write(fH2, 0);
	usleep(500000);

	PWM_open(pwm_channel);			//priprava pwm
	PWM_set(pwm_channel, pwn_cycle, pwm_duty);	//nastaveni pwm
	sleep(2);				//cekani v sekundach
	PWM_set(pwm_channel, pwn_cycle, pwm_duty2);
	sleep(2);

	ADC_open(adc_port);			//priprava adc
	ADC_read(adc_port);			//precteni hodnoty z adc
	ADC_unexport(adc_port);			//odpojeni adc
	PWM_unexport(4);			//odpojeni pwm
	GPIO_unexport(fH1);			//odpojeni gpio
	GPIO_unexport(fH2);
	close(fH1);				//uzavreni souboru
	close(fH2);


	return 0;
}
```

Postupy práce s kitem Intel Galileo

1.9 Instalace Arduino IDE

Instalace byla ověřena pro verzi programu 1.5.3 dostupnou na https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=23171

1.9.1 Pod OS Windows7

Pro úspěšné spuštění je zapotřebí mít Windows přepnuté na anglickou lokalizaci. Tento problém by měl být vyřešen v příští verzi programu.

Program stačí pouze rozbalit ze staženého archivu. Doporučuje se umístit program do kořenového adresáře (C:\ )

Dále je potřeba doinstalovat ovladač pro emulaci sériového portu. Chvilku po připojení kitu k PC automatická aktualizace ovladačů ohlásí chybu.

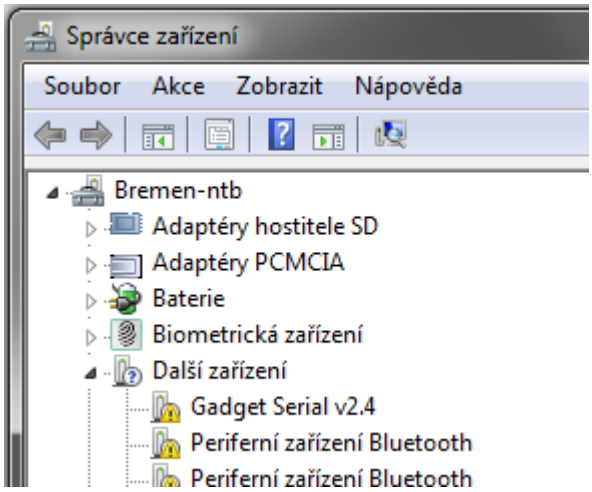
- Otevřete Ovládací panely/Správce zařízení

- Pod položkou Další zařízení by se měla nacházet položka Gadget serial v2.4 (může se nacházet též pod Porty (COM a LPT))

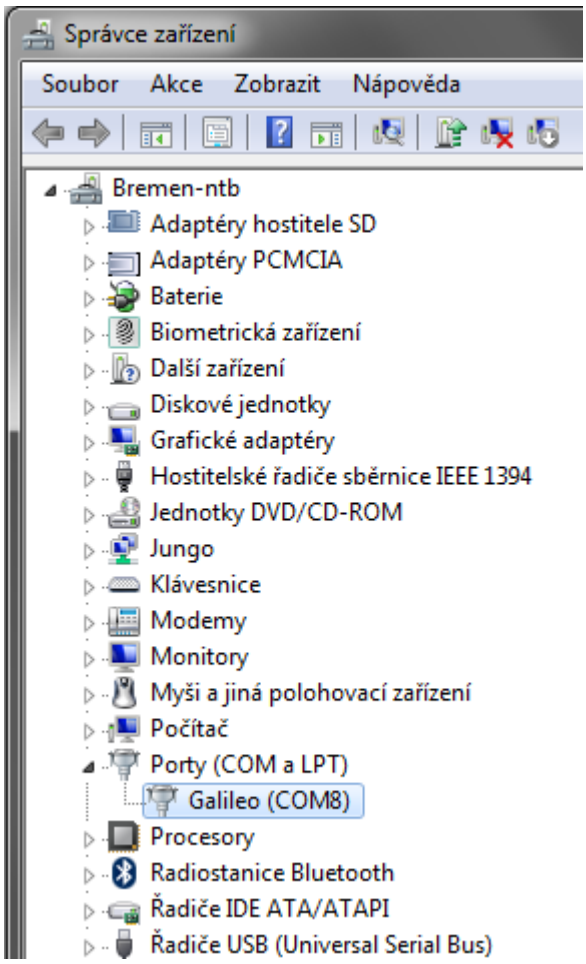
-Pravým tlačítkem zvolte „Aktualizovat software ovladače“ a vyberte „Vyhledat software v počítači“ a nechte jej vyhledat na umístění arduino-1.5.3\hardware\arduino\x86\tools

-Po úspěšné instalaci se zařízení hlásí jako Galileo s číslem portu v závorce

-Nyní lze spustit Arduino IDE (arduino-1.5.3\arduino.exe) a začít v něm pracovat



Obr. 6: Zařízení před instalací ovladače



Obr. 7: Zařízení po instalaci ovladače (číslo portu se může lišit)

1.9.2 Pod OS Linux

Pro úspěšné spuštění je zapotřebí mít Linux přepnutý na anglickou lokalizaci. Tento problém by měl být vyřešen v příští verzi programu.

Program potřebuje pro svůj běh prostředí Java. To by však mělo již být součástí operačního systému. V případě problémů s během programu doporučuji toto zkontrolovat, případně Javu aktualizovat.

Po rozbalení lze program spustit zadáním ./arduino ve složce programu. Další potřebné ovladače program nainstaluje sám při prvním překladu programu.

1.10 Update firmwaru

Nová verze firmwaru řeší některé potíže s kitem, doporučuji ji proto provést co nejdříve. Postup:

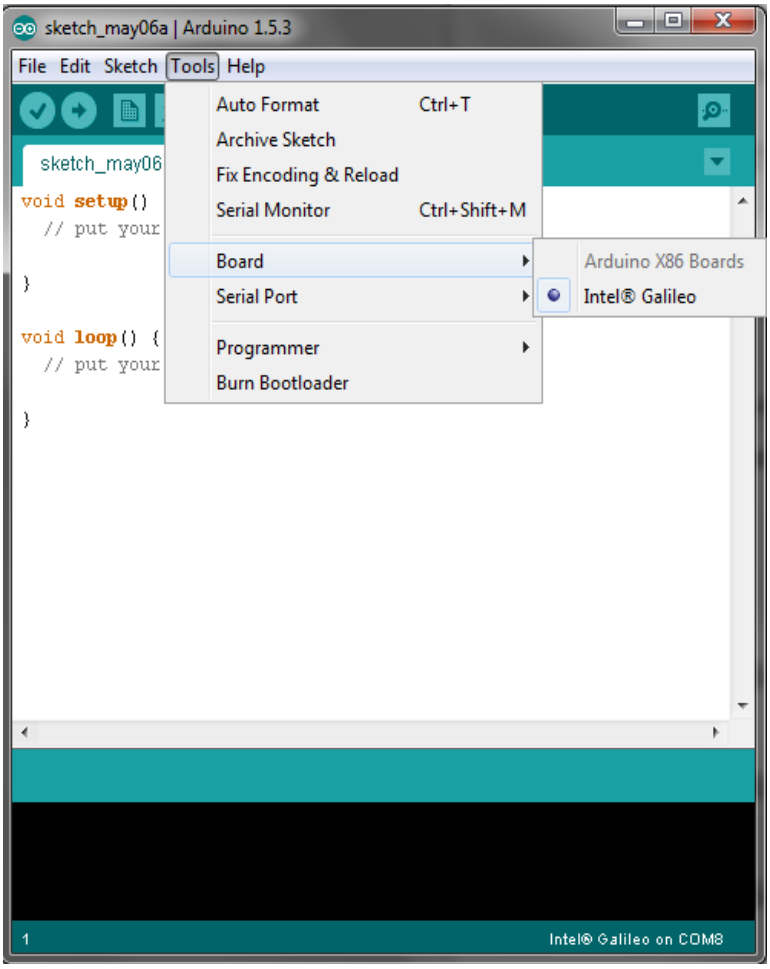
-Odpojte všechny kabely od kitu a znovu je připojte. Vyjměte SD kartu, pokud je přítomna.

-Spusťte Arduino IDE a zkontrolujte, zda je vybráno Intel Galileo (menu Tools/Board) a správný port (menu Tools/Serial Port). Číslo používaného portu lze zjistit ve správci zařízení (viz instalace Arduino IDE)

-Vyberte menu Help/Firmware Update

-Program se dotáže, zda je připojeno napájení – potvrďte

-Potvrďte verzi firmwaru pro update a vyčkejte na dokončení (IDE oznámí dokončení)



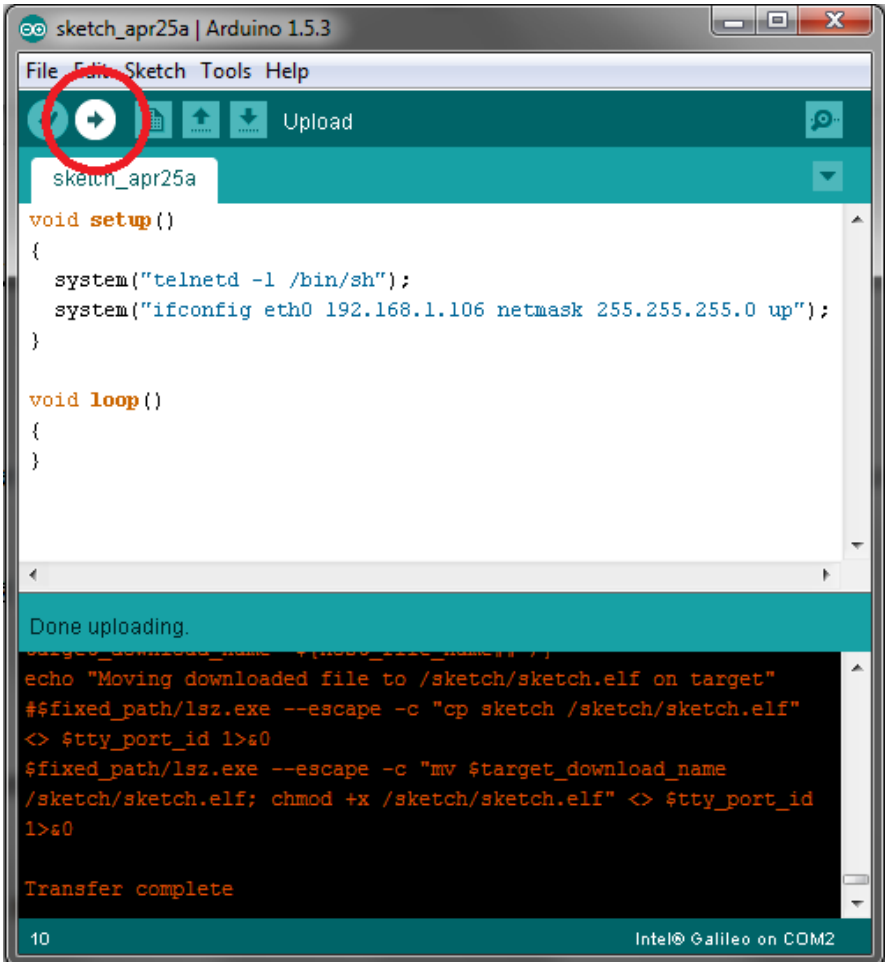
Obr. 8: Vybráno Intel Galileo v menu Tools/Board

1.11 Připojení protokolem TELNET

Nejprve je zapotřebí spustit telnet v kitu. To lze provést prostřednictvím Arduino IDE spuštěním následujícího programu (ip adresu a masku může být nutno upravit dle parametrů používané sítě):

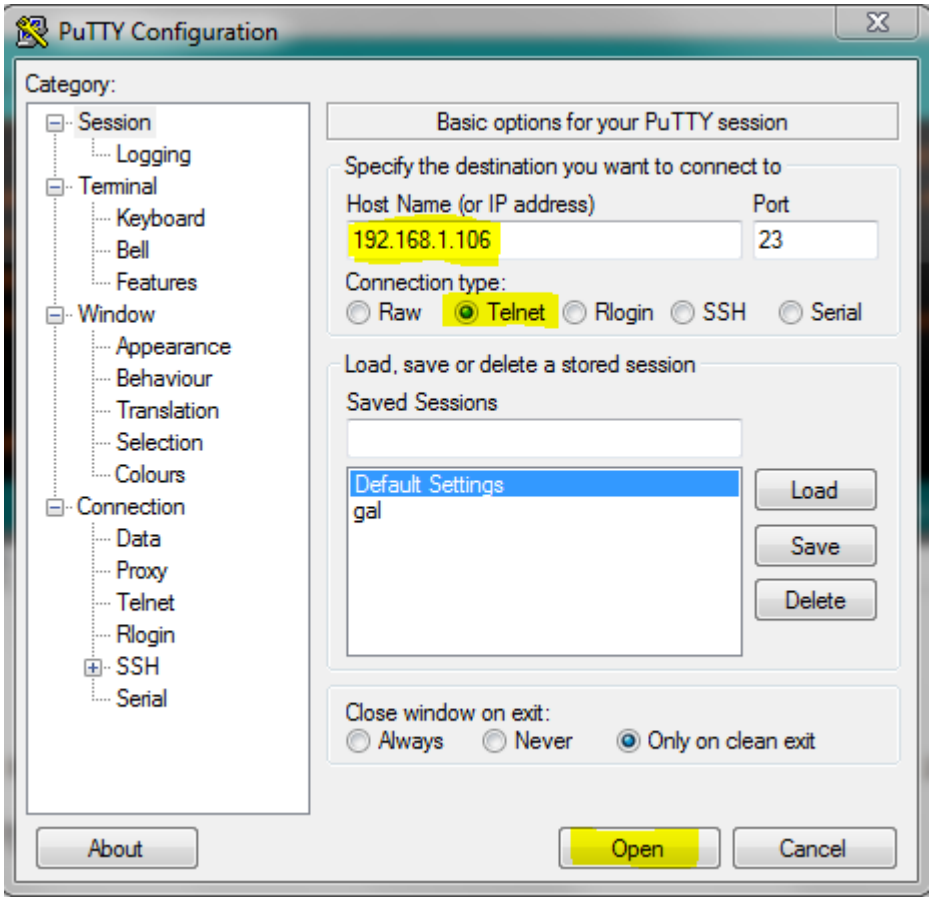
```
void setup()
{
  system("telnetd -l /bin/sh");
  system("ifconfig eth0 192.168.1.106 netmask 255.255.255.0 up");
}
void loop()
{
}
```

Program je do zařízení nahrán po stisknutí tlačítka upload (tlačítko se šipkou).



Obr. 9: Upload programu prostřednictvím Arduino IDE

Nyní se již lze připojit, například pomocí programu PuTTY – zde zadáme ip adresu, vybereme typ připojení – telnet a stiskneme open.



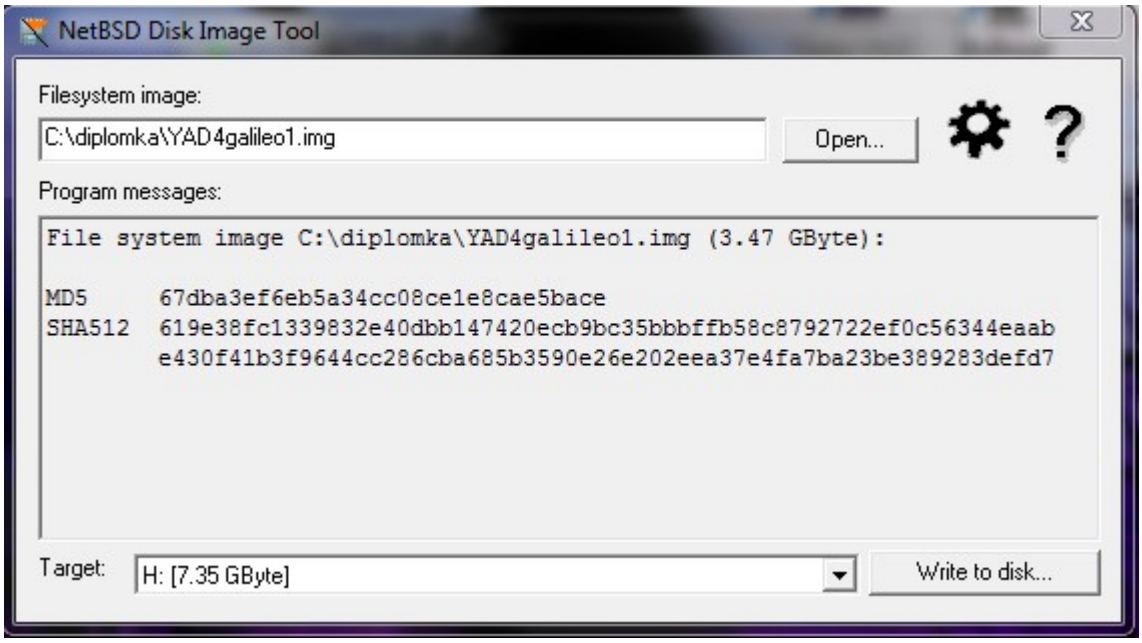
Obr. 10: Program PuTTY, Telnet připojení

1.12 Spuštění kitu z µSD karty

Pro nabootování z µSD karty je zapotřebí updatovaný firmware. Po vložení karty a restartu kitu již systém sám nabootoje z karty.

Pro použití distribuce LINUX_IMAGE_FOR_SD_Intel_Galileo_v0.7.5 (dostupné na https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=23171) postačí pouze zkopírovat obsah staženého archivu na paměťovou kartu. Ta by měla mít souborový systém FAT nebo FAT32 a měla by být nastavena jako aktivní oddíl. Doporučuji nejprve kartu naformátovat.

Pro distribuci yad4galileo1 je zapotřebí přenést na médium stažený obraz disku. Zde bohužel nelze použít kopírování. Pod MS Windows můžeme použít například program Rawrite32. Zde stačí jen vybrat obraz s Linuxem a cílový disk.

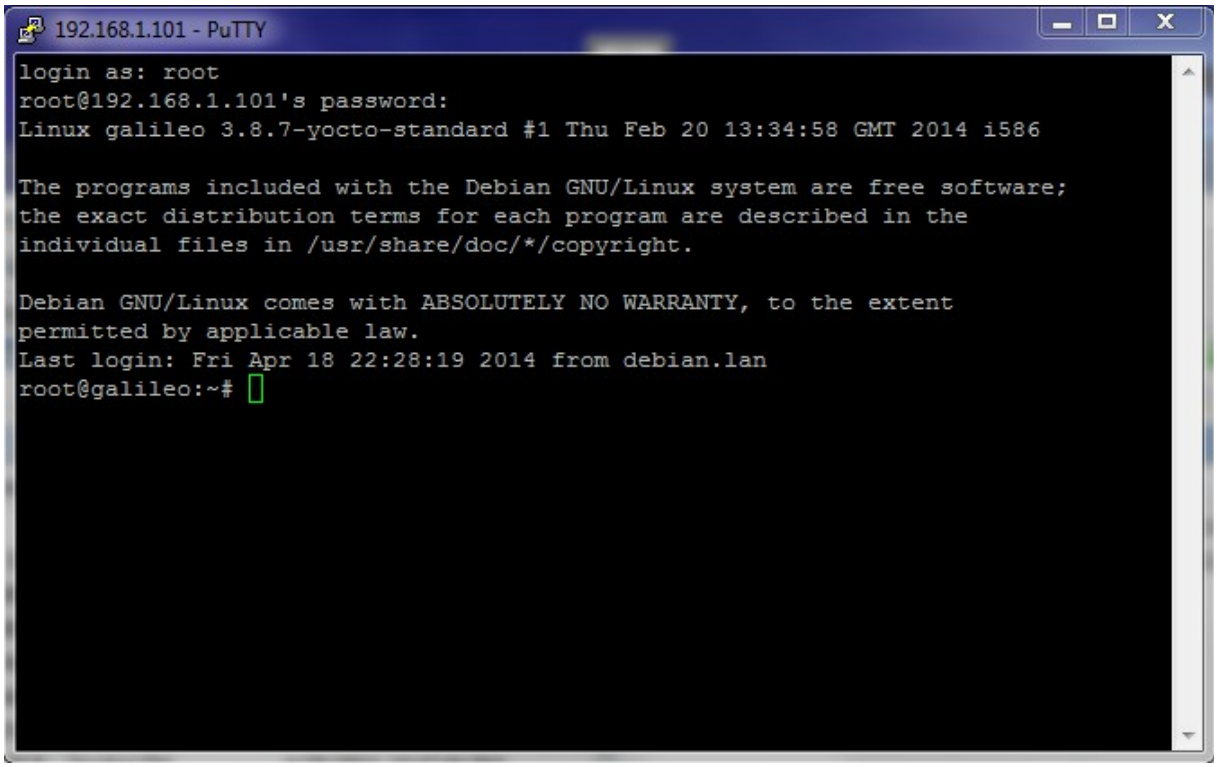


Obr. 11: Program Rawrite32

1.13 Připojení protokolem SSH

Pro použití SSH serveru je zapotřebí nabootovat z µSD karty. Obě testované distribuce (yad4galileo1 a LINUX_IMAGE_FOR_SD_Intel_Galileo_v0.7.5) již SSH server obsahovaly.

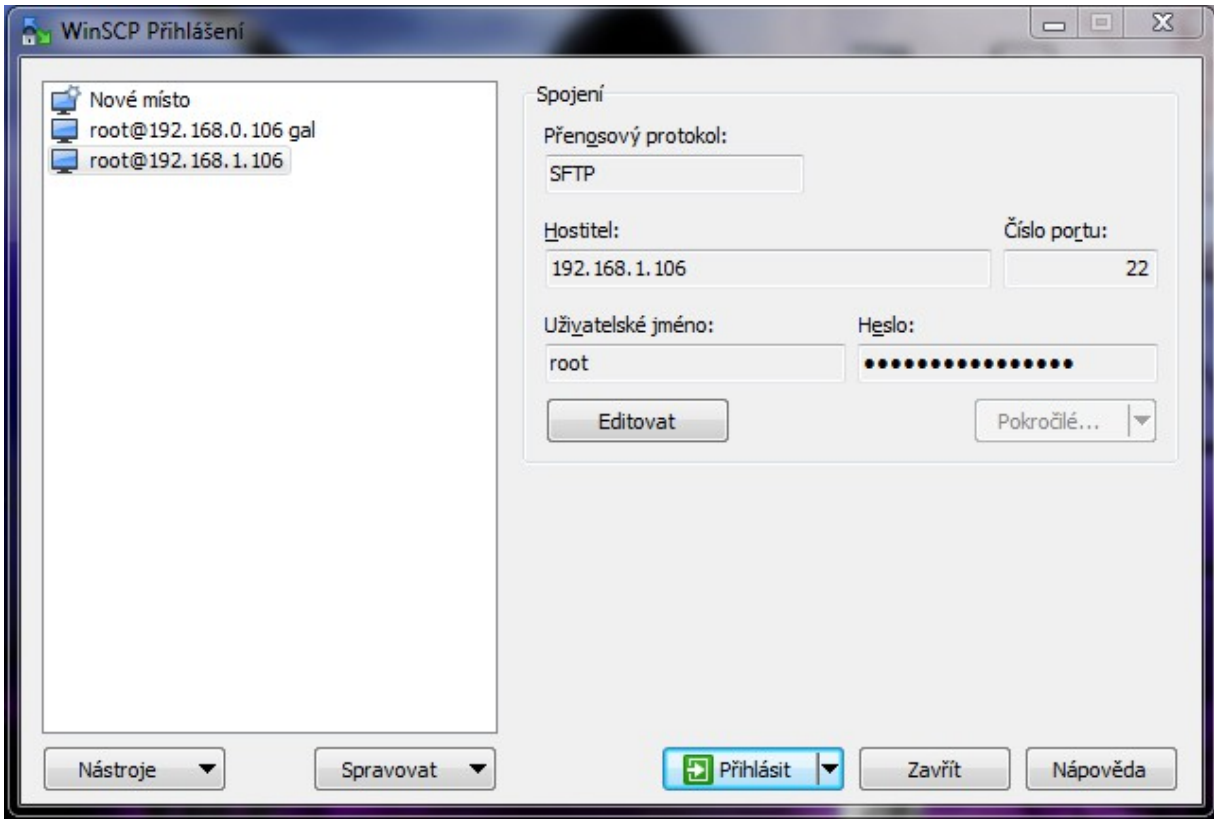
Připojit se lze opět například programem PuTTY, postup je obdobný jako při použití Telnetu, pouze je zapotřebí vybrat SSH a po přihlášení se systém dotáže na přihlašovací údaje.



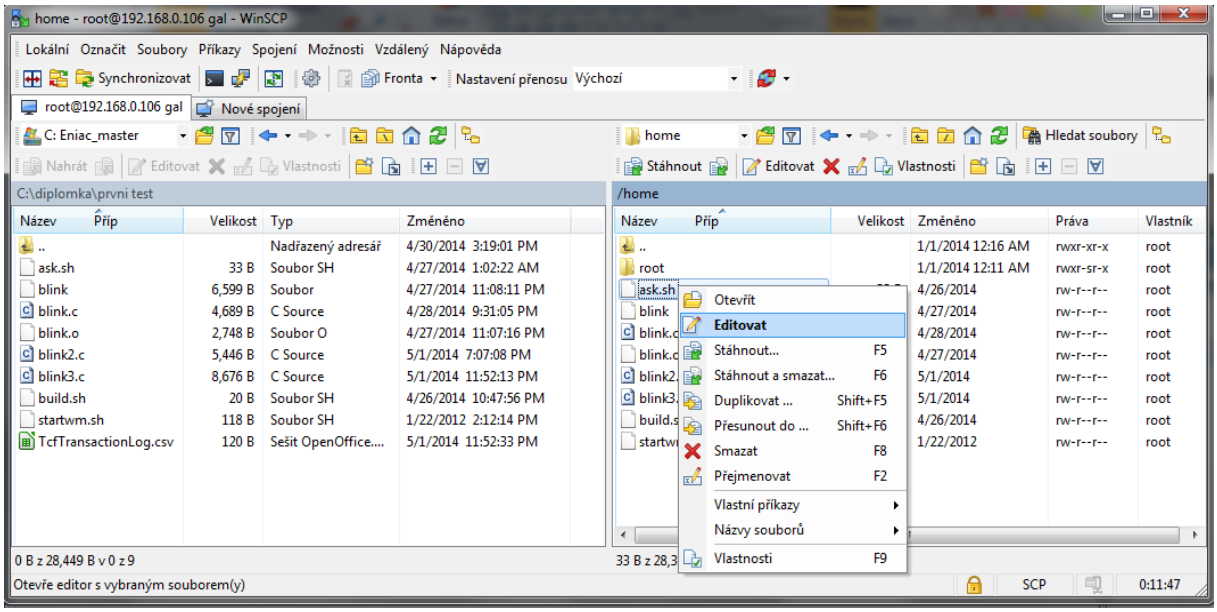
Obr. 12: Připojení programem PuTTY

1.14 Přenos souborů protokolem SCP

Zde doporučuji použít program WinSCP. Po vyplnění přihlašovacích údajů (IP, použitý protokol, login) je jeho požití stejné jako u běžného Commanderu pro Windows. Kromě kopírování lze v programu WinSCP soubory i upravovat – dají se zde rychle udělat drobné změny v programu nebo upravit skripty (viz dále)



Obr. 13: Přihlašovací obrazovka WinSCP



Obr. 14: Program WinSCP nabízí i možnost editace souboru

1.15 Instalace a spuštění xrdp

testováno pro yad4galileo1 distribuci

Instalace potřebných severů, příkazy zadejte do terminálu v uvedeném pořadí

apt-get install openssh

apt-get install vnc4server

apt-get install xrdp

Úprava používaného portu v ini souboru

v souboru /etc/xrdp/xrdp.ini změnit port=-1 (červeně) na jiný v použitelném rozsahu (například 5901)lze použít např. vi(přímo v konzoli) nebo winSCP (program typu windows commander používající SSH a SCP pro práci se vzdálenými soubory)

```
[globals]
bitmap_cache=yes
bitmap_compression=yes  port=3389
crypt_level=low
channel_code=1

[xrdp1]
name=sesman-Xvnc
lib=libvnc.so
username=ask
password=ask
ip=127.0.0.1
port=-1
```

restart xrdp

Zadáním příkazu *service xrdp restart* do terminálu

spuštění vnc4server

Zadáním příkazu *vnc4server* do terminálu

Při prvním spuštění může vyžadovat vyplnění přihlašovacích údajů

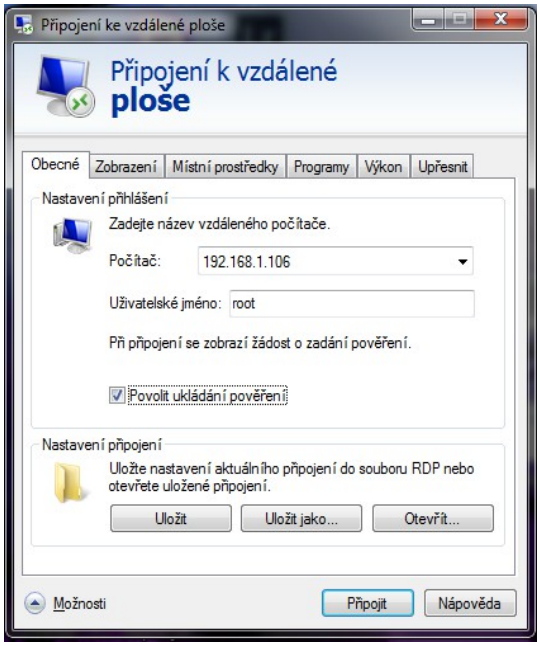
spuštění klienta, přihlášení

ve Windows7 například přes defaultního klienta pro vzdálenou plochu

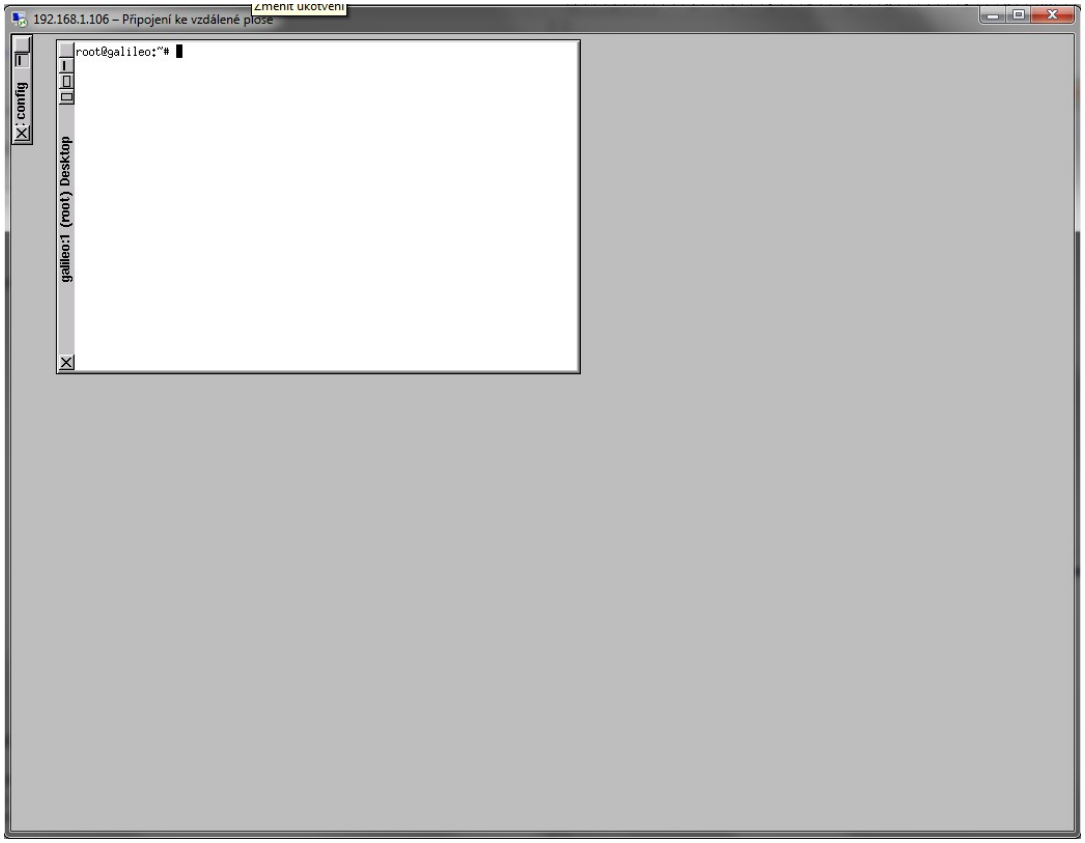
spustit Start/Všechny programy/Příslušenství/Připojit ke vzdálené ploše

nastavení ip cíle a loginu, stisknutí „připojit“

zadání nastaveného hesla



Obr. 15: nastavení připojení ke vzdálené ploše



Obr. 16: Výsledek - vzdálená plocha po přihlášení (zmenšeno)

-

1.16 Ovládání GPIO z terminálu

Jak již bylo zmíněno, jednotlivé GPIO kitu jsou reprezentovány soubory, do nichž se dají zapisovat hodnoty nastavení jednotlivých pinů nebo z nich lze číst jejich stav.

Nejprve je zapotřebí zjistit, zda je daný gpio exportovaný k použití. Čili je potřeba zjistit, zda ve složce sys/class/gpio existuje složka s číslem tohoto pinu (např. gpio37). To lze provést buďto otevřením dané složky v terminálu (do složky se dostanu požíváním příkazu cd a následně vypíši položky ve složce příkazem ls – viz obrázek) nebo požitím programu WinSCP

Neexistuje-li složka pro potřebný pin, je třeba ji exportovat. To se provede zápisem čísla GPIO do souboru export ve složce sys/class/gpio příkazem echo (viz dále).

1.16.1 Příkaz echo

Tento příkaz slouží k vypsání textu. Jeho výstup je však možno přesměrovat do souboru a tím jej používat k ovládání gpio. Zadává se ve formátu *echo „text“ > jmeno_souboru*

Tedy například *echo -n „39“ > export* zadaný ve složce sys/class/gpio zapíše číslo 39 do souboru sys/class/gpio/export a tím exportuje port 39 (tento odpovídá pinu 13 na kitu). Je-li příkaz zadáván v jiné složce, je třeba vypsat celou cestu k souboru a začít lomítkem. Tedy *echo -n „39“ > /sys/class/gpio/export*

Přepínač *-n* potlačuje znak ukončení řádky

```
root@galileo:/home# echo "39" > /sys/class/gpio/export
root@galileo:/home#
```

Obr. 17: Zápis do souboru z libovolné složky

```
root@galileo:/sys/class/gpio# echo "39" > export
root@galileo:/sys/class/gpio#
```

Obr. 18: Zápis do souboru ze stejné složky

1.16.2 Příkaz cat

Slouží k vypsání obsahu souboru do terminálu a je tedy vhodný ke zjišťování stavu GPIO. Zadává se ve formátu *cat jmeno_souboru*

Tedy například cat value zadaný ve složce sys/class/gpio/gpio39 vypíše hodnotu pinu 39 do terminálu (tento odpovídá pinu 13 na kitu). Je-li příkaz zadáván v jiné složce, je třeba vypsat celou cestu k souboru, tedy *cat /sys/class/gpio/gpio39/value* Cesta opět musí začínat lomítkem.

```
root@galileo:/sys/class/gpio/gpio39# cat value
1
root@galileo:/sys/class/gpio/gpio39#
```

Obr. 19: Přečtení hodnoty v souboru

1.16.3 Příkaz cd

Slouží ke změně složky. Zadává se ve formátu *cd ..* pro vystoupení do nadřazené složky nebo *cd jmeno_slozky* pro vstup do složky.

```
root@galileo:/sys/class/gpio/gpio39# cd ..
root@galileo:/sys/class/gpio#
root@galileo:/sys/class/gpio# cd gpio39
root@galileo:/sys/class/gpio/gpio39#
```

Obr. 20: Použití příkazu cd

1.16.4 Příkaz ls

Vypíše do terminálu obsah aktuální složky.

```
root@galileo:/sys/class/gpio# ls
export  gpio39  gpiochip0  gpiochip16  gpiochip2  gpiochip8  unexport
root@galileo:/sys/class/gpio#
```

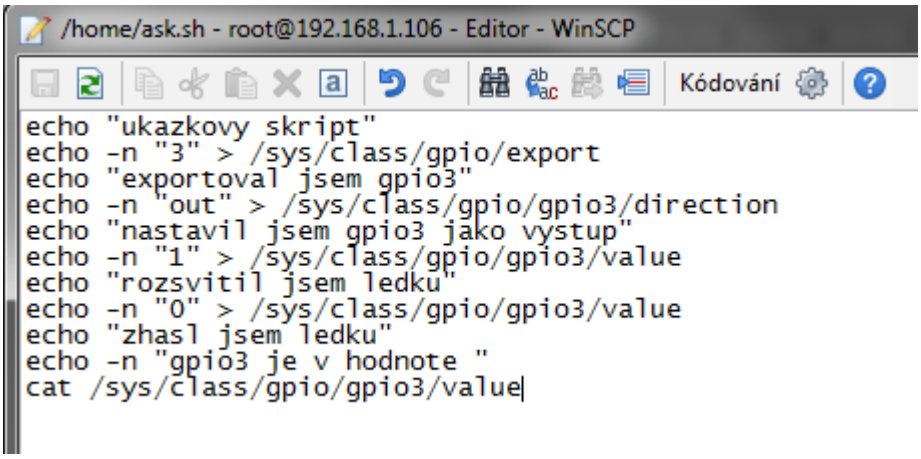
Obr. 21: Použití příkazu ls

1.17 Tvorba a použití jednoduchých skriptů

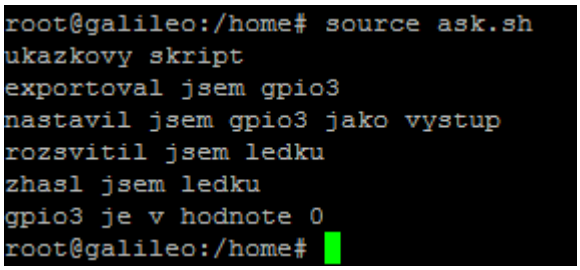
Pro zjednodušení zadávání příkazů do terminálu doporučuji namísto nich používat skripty. Jedná se o soubor s příponou *.sh . Ten obsahuje příkazy ve stejném formátu v jakém jsou zadávány do konzole. Upravovat jej lze v libovolném jednoduchém textovém editoru, například v poznámkovém bloku (ne ve Wordu).

Soubor je následně spuštěn příkazem *source jmeno.sh* zadaným ve složce, kde je tento soubor umístěn.

Jako velmi praktické se mi jeví napsat si skript pro překlad a spuštění programu napsaného v jazyce C, obzvlášť v případech, kdy se používají změněné parametry překladu (optimalizace kódu a.j.)



Obr. 22: Skript jež blikne ledkou a vypíše stav jejího gpio



Obr. 23: Výsledek skriptu

1.18 Kompilace a spuštění programu v jazyce C

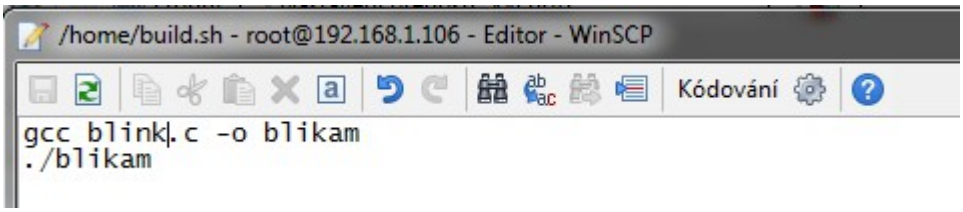
Po napsání a přenesení nového programu do kitu je zapotřebí jej přeložit. To lze provést příkazem *gcc jmeno_souboru.c -o jmeno_prelozeneho_souboru_bez_pripony.* Například *gcc blink.c -o blikam* Tento příkaz přeloží program blink.c a vytvoří z něj soubor blikam.

Systém následně vypíše seznam chyb a varování i s čísly řádků v kódu, kde k nim došlo. V případě úspěšného překladu se nic nevypisuje, jen se po chvilce znovuobjeví příkazová řádka.

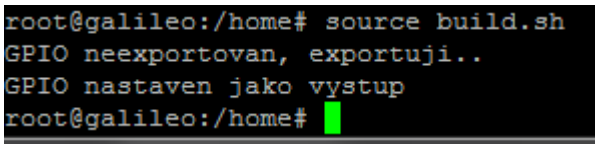
Po úspěšném překladu lze pokračovat spuštěním programu. To provedeme zadáním *./jmeno_prelozeneho_souboru_bez_pripony* Tedy *./blikam*

Oba tyto příkazy je zapotřebí zadávat ve složce, kde jsou umístěny používané soubory.

Provádění souboru lze zastavit stiskem Ctrl+C (například při nekonečné smyčce).

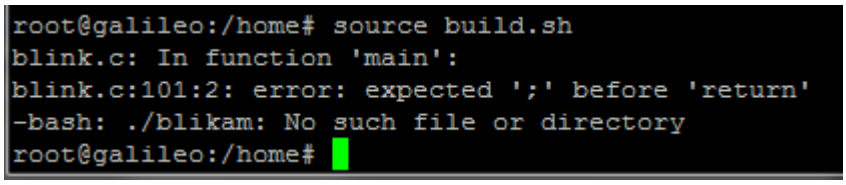


Obr. 24: Skript pro kompilaci a spuštění programu



Obr. 25: Průběh tohoto skriptu

Při překladu se nic nevypisuje, text na obrázku je vypisován běžícím programem.

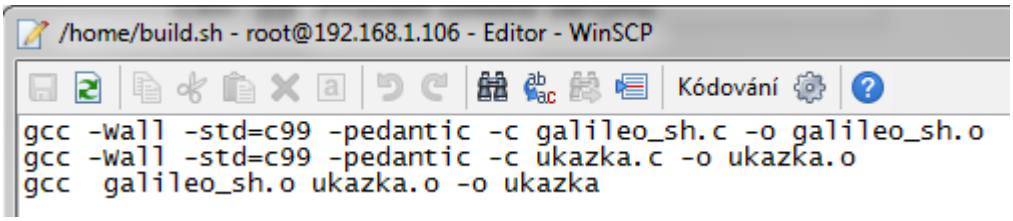


Obr. 26: Neúspěšný překlad vypíše chybu

1.18.1 Použití modulu galileo_sh

Pro použití modulu je zapotřebí přidat na začátek programu #include „galileo_sh.h“

Dále je zapotřebí modul zkompilovat a připojit k programu. Doporučuji vytvoření následujícího skriptu (název programu *ukazka* nahradit vlastním názvem).



Obr. 27: skript pro kompilaci s modulem galileo_sh

Závěr

Zpočátku byla práce s kitem Intel Galileo docela problematická. Po updatu firmwaru a nabootování z µSD karty však byla většina těchto problémů odstraněna. Další používání probíhalo přes protokol SSH. Zde již bylo prostředí vcelku uživatelsky přívětivé.

Věřím, že návody a příklady uvedené v této práci dokáží přiblížit práci s tímto zařízením (a podobnými) i studentům, kteří s programováním a Linuxem teprve začínají. Ovládání GPIO je vcelku intuitivní. Jediným problémem jsou názvy jednotlivých GPIO – je zapotřebí dohledávat konkrétní čísla v tabulce. Použití modulu galileo_sh tento problém odstraní.

Očekával jsem, že nevýhodou kitu Intel Galileo bude absence grafického výstupu. V průběhu jsem však zjistil, že ovládání prostřednictvím protokolu SSH je plně dostačující.

Seznam literatury a informačních zdrojů

[1] Gallileo Getting Started. Sparkfun [online]. 2013 [cit. 2014-05-07]. Dostupné z: https://learn.sparkfun.com/tutorials/galileo-getting-started-guide

[2] Blinking LEDs – manipulating digital GPIOs on the Intel® Galileo board with the IoT Development Kit. Intel Developer zone [online]. 2013 [cit. 2014-05-07]. Dostupné z: https://software.intel.com/en-us/articles/blinking-leds-manipulating-digital-gpios-on-the-intel-galileo-board-with-the-iot

[3] Intel Galileo - Programming GPIO From Linux. Sergey's Blog [online]. 2013 [cit. 2014-05-07]. Dostupné z: http://www.malinov.com/Home/sergey-s-blog/intelgalileo-programminggpiofromlinux

[4] XRDP via SSH session. My E-Health Technology Blog [online]. 2013 [cit. 2014-05-07]. Dostupné z: http://ehealth-aussie.blogspot.cz/2012/07/xrdp-via-ssh-session.html

[5] Yet Another Debian available. Intel comunities [online]. 2013 [cit. 2014-05-07]. Dostupné z: https://communities.intel.com/thread/50862?start=0&tstart=0

[6] Raspberry Pi requirment of Peripherals. Education federal [online]. [cit. 2014-05-08]. Dostupné z: http://educationfederal.com/raspberry-pi/

[7] BeagleBone: open-hardware expandable computer. BeagleBoard.org [online]. [cit. 2014-05-08]. Dostupné z: http://beagleboard.org/Support/bone101

Přílohy

Příloha A – Modul galileo_sh.c

```
#include "galileo_sh.h"

int GPIO_prep(int gpio, char rw){

	int fHexp, fHdir;                                   //file handler
	char buff[256];                                          //buffer

	sprintf(buff, "/sys/class/gpio/gpio%d/direction", gpio);      //cesta k souboru
                                                                  do bufferu
	if((fHdir = open(buff, O_WRONLY)) == -1){       //lze otevrit "direction"
                                                        portu?  =  je port dostupny?
	sprintf(buff, "GPIO%d neexportovan, exportuji..",gpio);
	puts(buff);                          //vypis do konzole
	sprintf(buff, "/sys/class/gpio/export");
	fHexp = open(buff, O_WRONLY);
	sprintf(buff, "%d", gpio);
	write(fHexp, buff, strlen(buff));                   //zapis do souboru
	close(fHexp);
	sprintf(buff, "/sys/class/gpio/gpio%d/direction", gpio);
	if((fHdir = open(buff, O_WRONLY)) == -1){
		sprintf(buff, "chyba, nepovedlo se exportovat GPIO%d", gpio);
		puts(buff);
		return -1;
	}
}
if(rw=='R'){
	write(fHdir,"in",2);
	close(fHdir);
	sprintf(buff, "GPIO%d nastaven jako vstup", gpio);
	puts(buff);
	return 1;
}
if(rw=='W'){
	write(fHdir,"out",3);
	close(fHdir);
	sprintf(buff, "GPIO%d nastaven jako vystup", gpio);
	puts(buff);
	return 2;
}
close(fHdir);
sprintf(buff, "GPIO%d nenastaven, spatne parametry", gpio);
return -2;
}

int GPIO_open(int gpio, char rw){

	int fHval;
	char buff[256];

	if(GPIO_prep(gpio, rw)<0) return -1;
	sprintf(buff, "/sys/class/gpio/gpio%d/value", gpio);
	if(rw=='W'){
		if((fHval = open(buff, O_WRONLY)) == -1){
			puts("port nedostupny");
			return -2;
		}
```

```
		return (fHval);
	}
	if(rw=='R'){
		if((fHval = open(buff, O_RDONLY)) == -1){
			puts("port nedostupny");
			return -2;
		}
		return (fHval);
	}
	puts("GPIO nenastaven, spatne parametry");
	puts("nastavte R pro vstupni port nebo W pro vystupni");
	return -3;
}

void GPIO_write(int fHval, int val){
	if(val == 0) write(fHval, "0",1);
	if(val == 1) write(fHval, "1",1);
}

int gpio_read(int fHval){
	char val;
	read(fHval, &val, 1);
	if (val == '1' )return 1;
	if (val == '0' )return 0;
	return -1;
}


int GPIO_unexport(int gpio){
	int fHuexp;
	char buff[256];

	sprintf(buff, "/sys/class/gpio/unexport");
	fHuexp = open(buff, O_WRONLY);
	sprintf(buff, "%d", gpio);
	write(fHuexp, buff, strlen(buff));
	close (fHuexp);
	return 1;
}

int PWM_open(int pwm_port){

	int fHexp, fHen;			//file handler
	char buff[256];
					//buffer
	pwm_port = pwm_tab(pwm_port);
	sprintf(buff, "/sys/class/pwm/pwmchip0/pwm%d/enable", pwm_port);//cesta k
							souboru do bufferu
	if((fHen = open(buff, O_WRONLY)) == -1){	//lze otevrit "enable" portu?
							=  je port dostupny?
		puts("PWM neexportovan, exportuji..");		//vypis do konzole
		sprintf(buff, "/sys/class/pwm/pwmchip0/export");
		fHexp = open(buff, O_WRONLY);
		sprintf(buff, "%d", pwm_port);
		write(fHexp, buff, strlen(buff));		//zapis do souboru
		close(fHexp);
		sprintf(buff, "/sys/class/pwm/pwmchip0/pwm%d/enable", pwm_port);
		if((fHen = open(buff, O_WRONLY)) == -1){
			puts("chyba, nepovedlo se exportovat PWM");
			return -1;
		}
	}
```

```
	write(fHen,"1",1);
	close(fHen);
	return 1;
}

int PWM_set(int pwm_port, int period, int duty){
	int fHper, fHdut;                                   //file handler
	char buff[256];                                     //buffer

	pwm_port = pwm_tab(pwm_port);
	period = period * 1000000;
	duty = duty * period / 100;
	sprintf(buff, "/sys/class/pwm/pwmchip0/pwm%d/period", pwm_port);//cesta k
                                                            souboru do bufferu
	if((fHper = open(buff, O_WRONLY)) == -1){           //je port dostupny?
		puts("port nedostupny, pouzijte PWM_open(int pwm)");
		return -1;
	}
	sprintf(buff, "/sys/class/pwm/pwmchip0/pwm%d/duty_cycle", pwm_port); //cesta k
                                                            souboru do bufferu
	if((fHdut = open(buff, O_WRONLY)) == -1){           //je port dostupny?
		puts("port nedostupny, pouzijte PWM_open(int pwm)");
		return -1;
	}
	write(fHper, buff, strlen(buff));
	sprintf(buff, "%d", duty);
	write(fHdut, buff, strlen(buff));
	sprintf(buff, "pwm nastaveno na period %d ns, duty cycle %d ns", period, duty);
	puts(buff);
	close(fHper);
	close(fHdut);
	return 1;
}

int PWM_unexport(int pwm_port){
	int fHuexp;
	char buff[256];

	sprintf(buff, "/sys/class/pwm/pwmchip0/pwm%d/enable", pwm_port);
	fHuexp = open(buff, O_WRONLY);
	write(fHuexp, "0", 1);
	close(fHuexp);
	sprintf(buff, "/sys/class/pwm/pwmchip0/unexport");
	fHuexp = open(buff, O_WRONLY);
	sprintf(buff, "%d", pwm_port);
	write(fHuexp, buff, strlen(buff));
	close (fHuexp);
	return 1;
}



int ADC_open(int adc_port){
	int mux_port, fHadc;

	mux_port = mux_tab(adc_port);
	if((fHadc = GPIO_open(mux_port,'W')) < 0){
		puts("nelze otevrit multiplexer pro zapis (prepnuti)");
		return -1;
	}
	GPIO_write(fHadc,0);
	puts("ADC pripraven..");
```

```
	close(fHadc);
	return 1;
}

int ADC_read(int adc_port){
	int fHadc,res;                                  //file handler
	char buff[256];                                 //buffer
	int result = -1;

	sprintf(buff, "/sys/bus/iio/devices//iio:device0/in_voltage%d_raw", adc_port);
                                                //cesta k souboru do bufferu
	if((fHadc = open(buff, O_RDONLY)) == -1){       //je port dostupny?
		puts("port nedostupny, pouzijte ADC_open(int adc_port)");
		return -1;
	}
	puts("adc reading..");
	read (fHadc, &buff, 4);
	result = atoi(buff);
	close(fHadc);
	res = result * 1000 * UREF;
	res = res / RANGE;
	sprintf(buff, "vysledek: %d = %d mV", result, res);
	puts(buff);
	return result;
}

int ADC_unexport(int adc_port){
	int mux_port, fHadc;

	mux_port = mux_tab(adc_port);
	if((fHadc = GPIO_open(mux_port,'W')) < 0){
		puts("nelze otevrit multiplexer pro zapis (prepnuti do puvodniho
stavu)");
		return -1;
	}
	GPIO_write(fHadc,1);
	GPIO_unexport(mux_port);
	puts("ADC odpojen");
	close(fHadc);
	return 1;
}

int port_tab(int aport)	{
	switch(aport){
		case 0: return 50;
		case 1: return 51;
		case 2: return 32;
		case 3: return 18;
		case 4: return 28;
		case 5: return 17;
		case 6: return 24;
		case 7: return 27;
		case 8: return 26;
		case 9: return 19;
		case 10: return 42;
		case 11: return 25;
		case 12: return 38;
		case 13: return 39;
	}
	return -1;
}
```

```
int mux_tab(int port)       {
       switch(port){
              case 0: return 37;
              case 1: return 36;
              case 2: return 23;
              case 3: return 22;
              case 4: return 21;
              case 5: return 20;
       }
       return -1;
}

int pwm_tab(int aport){

       switch(aport){
              case 3: return 3;
              case 5: return 5;
              case 6: return 6;
              case 9: return 1;
              case 10: return 7;
              case 11: return 4;
       }
       return -1;
}
```

Příloha B – Hlavičkový soubor galileo_sh.h

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>   //práce se soubory
#include <unistd.h>  //knihovna obsahující sleep
#include <string.h>

#define UREF 5       //referencni napeti pro ADC
#define RANGE 4095 //rozsah vysledku ADC (12bitu)


int GPIO_prep(int gpio, char rw); //pripravi gpio k pouziti - ujisti se, ze je
                                  exportovan (k pouziti), pripadne exportuje, nastavi
                                  smer (in[R]/out[W])
int GPIO_open(int gpio, char rw); //nastavi a otevre port   Volba smeru :
                                  (in[R]/out[W]), vraci file handler portu
void GPIO_write(int fHval, int val);
int gpio_read(int fHval);
int GPIO_unexport(int gpio);
int PWM_open(int pwm_port);       //pripoji PWM port
int PWM_set(int pwm_port, int period, int duty); //nastavi periodu a duty cycle.
                                  Perioda je v milisekundach, duty v procentech
int PWM_unexport(int pwm_port);
int ADC_open(int adc_port);
int ADC_read(int adc_port);
int ADC_unexport(int adc_port);
int port_tab(int aport);    //prevod z cisel portu na desce (arduino pinout) na cisla
                            portu adresovana systemem
int mux_tab(int port);      //vraci cislo multiplexeru pred ADC
int pwm_tab(int aport);
```