VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘÍDICÍ PANEL PRO AUTOMATIZACI BUDOV

CONTROL PANEL FOR SMART HOUSE AUTOMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Pavel Širůček
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Stanislav Věchet, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Pavel Širůček**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Řídicí panel pro automatizaci budov

Stručná charakteristika problematiky úkolu:

Podstatou práce je navržení a realizace řídícího panelu pro automatizaci budov. K panelu bude možné připojovat vstupní a výstupní zařízení. Celá sestava bude sloužit k automatizování některých procesů běžně se vyskytujících v domácnostech a malých podnicích, jako je například měření a ovládání teploty, řízení osvětlení nebo průtoku vody. K realizaci řešení bude použit open source hardware Arduino a k němu kompatibilní příslušenství.

Cíle diplomové práce:

1) Seznámit se s Arduinem a příslušnými periferiemi.
2) Navrhnout a vytvořit desku elektroniky pro řídicí panel.
3) Napsat řídicí program a ověřit jeho funkčnost na praktické aplikaci.
4) Popsat možnosti použití systému a jeho limity.

Seznam doporučené literatury:

BALÁTĚ, J.: Technické prostředky automatického řízení. Praha, SNTL 1986.

ROS.org. ROS.org | Powering the world's robots. [online]. 2.11.2016 [cit. 2016-11-02]. Dostupné z: http://www.ros.org/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Práce pojednává o automatizaci budov, zaměřením produktů v této oblasti a jejich použitím. Dále se věnuje konkrétnímu řešení vyvíjenému v rámci této práce pro automatizaci budov, jeho hardwarovým a softwarovým vlastnostem. Výsledný panel pro automatizaci budov je kompaktní zařízení, zamýšlené především pro nekomerční použití. Jedná se o víceúčelové zařízení, jehož konkrétní funkci nastavuje uživatel volbou řídících instrukcí. Je zde uveden jejich popis, software pro jejich sestavení a příklady praktických aplikací.

ABSTRACT

This thesis deals with smart house automation, the application of products in this field, and their use. It also describes a specific solution developed in this work for smart house automation and its hardware and software properties. The developed smart house automation panel is a compact device designed primarily for non-commercial use. It is a multipurpose device whose specific function is set by the user by selecting control instructions. This work contains a description of these instructions, the software for building them, and examples of practical applications.

KLÍČOVÁ SLOVA

Automatizace budov, řídící instrukce, SpeedAPI, panel pro automatizaci budov, Arduino, PLCduino

KEYWORDS

Smart house automation, control instructions, SpeedAPI, control panel for smart house automation, Arduino, PLCduino

BIBLIOGRAFICKÁ CITACE

ŠIRŮČEK, P. *Řídicí panel pro automatizaci budov.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 73 s. Vedoucí diplomové práce doc. Ing. Stanislav Věchet, Ph.D..

PODĚKOVÁNÍ

Chtěl bych poděkovat mému vedoucímu práce doc. Ing. Stanislavu Věchetovi, Ph.D. za odborné vedení, Bohu, mé přítelkyni a rodině za zázemí, pomoc a podporu a Jiřímu Březinovi za odbornou radu.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Stanislava Věcheta, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 26. 5. 2017 ……

Pavel Širůček

OBSAH

1 ÚVOD

Automatizace budov se zabývá automatizováním procesů, běžně se vyskytujících v obytných a kancelářských budovách, jako jsou řízení osvětlení, větrání, vytápění, bezpečnosti osob a přístupu osob a to jak v samostatných místnostech, tak v celých budovách. [1] Na trhu se nachází množství produktů, které řeší tuto problematiku a liší se jak zaměřením, tak cenou. Seznámení s alespoň některými z nich a srovnání jejich vlastností je první částí této práce. Zvolená řešení jsou rozčleněna do dvou kategorií, dle rozsahu použití, na zařízení univerzální a zařízení určená k automatizaci konkrétních procesů. V následující části je vysvětleno, jaká oblast automatizace budov není v této oblasti dle autora ještě zcela zaplněna a navrhnuto vlastní řešení, zaměřující se právě na tuto oblast.

Vlastním řešením je pak kompaktní přístroj umožňující automatizovat některé z výše uvedených procesů. Základní myšlenkou bylo navrhnout zařízení, jež lze sestavit z běžně dostupných dílů a jehož nastavení k zvolené činnosti je možné přímo uživatelem. Zařízení je postaveno na rozšířené jednočipové platformě Arduino [2], ke které je množství veřejně dostupné dokumentace. Řízení zařízení je realizováno prostřednictvím instrukcí, uložených na datové kartě, jenž je čtena čtečkou SD karet. Jednotlivým částem hardwaru a jejich propojení je věnována první část vlastního řešení. V následující části je pak popsán software, v němž byly programovány jednotlivé části řešení. Součástí je i grafické aplikační rozhraní sloužící k tvorbě instrukcí pro zařízení a zjednodušující jejich tvorbu, protože jejich manuální vytváření prostřednictvím textového editoru je velmi nepřehledné. V poslední části jsou nastíněny příklady použití, ukazující jak schéma zapojení, tak sadu instrukcí nutnou pro správný běh konkrétního příkladu.

2 POUŽÍVANÁ ŘEŠENÍ AUTOMATIZACE BUDOV

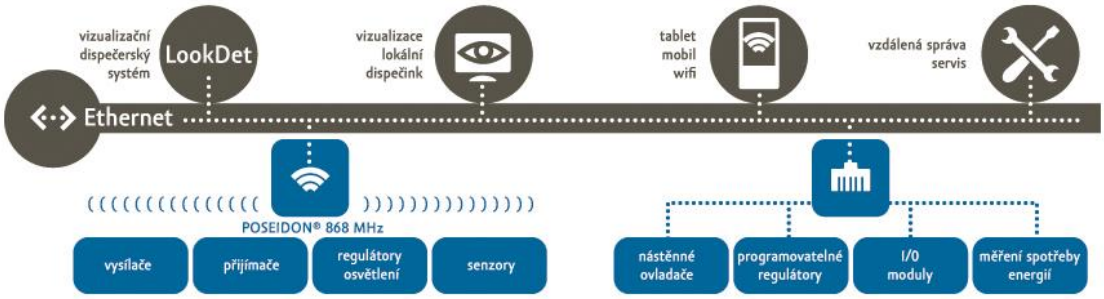
V dnešní době je k dispozici velké množství řešení pro automatizaci budov, lišících se jak přístupem k problému, tak cenou. V následující části budou rozebrány vybrané z nich a přiblíženo jejich rozdělení a použití. U těch řešení, kde výrobce udává technické parametry a cenu, jsou tyto parametry uvedeny v tabulce. Parametry jsou získané ze stránek výrobce. Cena se může lišit v závislosti na zemi, prodejci a době nákupu a je k ní tedy ve všech případech třeba přistupovat pouze jako k orientační položce. Následující seznam by měl ukázat jakým směrem se často automatizace budov ubírá a více přiblížit, proč bylo rozhodnuto navrhnout vlastní řešení. Z hlediska přístupu lze rozdělit řešení pro automatizaci budov do dvou hlavních kategorií, které můžeme nazvat jako komplexní řešení a kompaktní řešení, lišící se vzájemně především rozlohou celého systému.

2.1 Komplexní řešení

Přistupují k automatizaci budov jako k automatizaci celé budovy, případně jejích větších částí. Jejich hlavním znakem je centrální řídící systém umístěný uvnitř budovy a řídící její funkce, k němu pak mohou, nebo nemusí být, připojeny rozšiřující moduly, snímače, akční členy a přístupové body. Mezi hlavní výhody tohoto systému patří, že použitím jediného systému pro celou budovu je zaručena jeho kompatibilita a že takové systémy bývají snadno dále rozšiřitelné. Nevýhodou pak je vyšší cena, oproti řešením jednoúčelovým a často i nutnost počítat s instalací takového systému už při navrhování celé budovy, což značně limituje použití tohoto systému u budov již postavených. Je tedy vhodné zavádět takové řešení jen pokud chce uživatel zautomatizovat větší celky. Pro automatizaci jedné zásuvky, případně osvětlení v jedné místnosti je určitě lepší zvolit řešení kompaktní. Tyto řešení se v případě automatizace obytných prostor zaměřují především na oblast větrání, vytápění, svícení, vytahování a stahování rolet, zabezpečení, zapínání spotřebičů a ozvučení. [3] Mnohdy je možné ovládat je prostřednictvím chytrého telefonu, případně nástěnných ovládacích panelů. Dalším typickým znakem komplexních řešení je nutnost jejich odborné instalace. Celou síť musí nejprve nainstalovat technik s příslušným elektrikářským vzděláním a před uvedením do provozu bývá nutné ještě její naprogramování. Pokud chce uživatel v průběhu životnosti sítě měnit některé její funkce, například, který vypínač spíná jaké světlo, musí opět vyhledat programátora, který tuto změnu v softwaru sítě provede. Tato skutečnost tedy dále prodražuje celé řešení. Za komplexní řešení jsou v této práci označeny všechny řešení, které při instalaci počítají s použitím dvou a více zařízení, jež spolu nějakým způsobem komunikují a jejichž funkčnost by byla bez společného použití nemožná, nebo alespoň velmi omezená.

2.1.1 Automatizace budov od firem Anika a Emit

Základ systému tvoří více řídících jednotek, které spolu komunikují prostřednictvím sítě Ethernet (Obr. 2.1), k těm je možné připojit členy monitorující a regulující teplotu, vlhkost, přítomnost, intenzitu osvětlení, kvalitu vzduchu, ale také aktuální pozici



Obr. 2.1 schéma sítě pro automatizaci budov [4]

venkovních žaluzií, oken či dveří. Tyto členy mohou být připojeny buď také pomocí sítě ethernet, nebo bezdrátově přes sběrnici Poseidon [4]. Součástí řídícího systému je i uživatelská a dispečerská vizualizace. Bezdrátové moduly lze instalovat například do již existujícího osvětlení všech druhů. Systém lze obsluhovat přes mobil nebo také vzdálenou správou. Systém je určený především k řízení osvětlení, zabezpečení, vlhkosti a teploty a při jeho nasazení slibuje značné energetické úspory. Počítá se, že systém bude zahrnut do budovy už při jejím projektování, možnost instalace systému do budov již existujících není na stránkách výrobce uvedena, zároveň však nikde neuvádí, že by to možné nebylo. [3]

2.1.2 I-bus od ABB

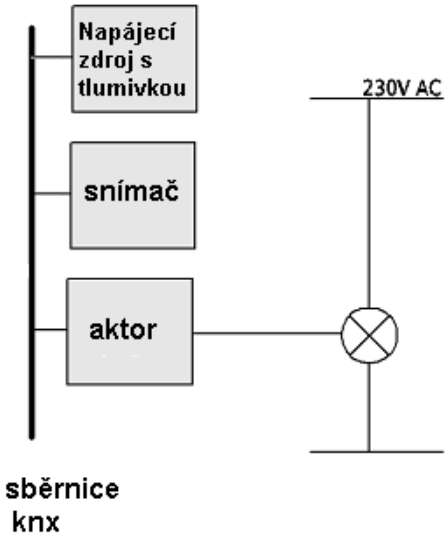
Jedná se o systém pro automatizaci budov založený na standardu KNX, takže je zaručena vzájemná kompatibilita systému v rámci zařízeních využívajících tento standard. Asociace KNX byla založena v roce 1999 jako sdružení tří společností a zavázala se poskytovat podporu systémům s KNX po nezbytně nutnou dobu. V roce 2003 pak byl tento standard odsouhlasen jako evropská norma. [5]

Nejmenší systém na standardu KNX a tím pádem i řešení i-bus od ABB musí obsahovat tyto prvky (Obr. 2.2):

- Napájecí zdroj 30V DC - Typy zdrojů (160mA, 320mA, 640mA a 1280mA)
- Snímač – např. tlačítko, snímač teploty, vlhkosti a podobně
- Akční člen – např. relé, ventil topení, stmívač apod.
- Sběrnicové vedení – kroucená dvoulinka

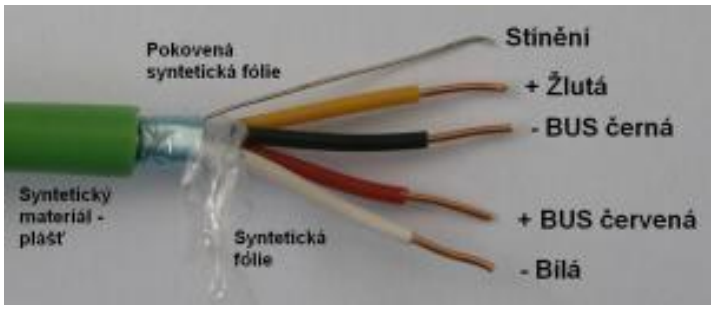
Před uvedením takové instalace do provozu je nutné provést ještě zadání individuálních adres jednotlivým přístrojům, parametrizace příslušného softwaru snímače nebo akčního členu a přiřazení skupinových adres, což znamená provázání funkcí snímačů a akčních členů. V jedné instalaci lze použít maximálně 58000 přístrojů. Ty se ke sběrnici připojují

pomocí sběrnicových svorkovnic, umožňujících jejich odebrání bez přerušení sběrnice. Jako přenosové medium se používá kabel YCYM 2x2x0 8 (Obr. 2.3), který splňuje požadavky KNX. Topologie sběrnice může být realizována jako lineární, stromová,



Obr. 2.2 nejmenší možný systém na sběrnici KNX [6]

hvězdicová, případně kombinace těchto topologií. Struktura KNX je uspořádána do linií, které jsou vzájemně propojeny liniovými spojkami. Zařízení zapojená do příslušných linií jsou napájeny napětím 30V. [5]



Obr. 2.3 kabel pro KNX instalaci [6]

Systém i-bus [7] nabízí prvky pro vytápění, klimatizaci, ovládání žaluzií, osvětlení, komunikaci, vzdálenou správu, energetický management a zabezpečení. Programování sítě, její nastavení a údržba probíhá v softwaru ETS. Jedná se o grafické programovací prostředí, umožňující přidání komponent pomocí metody “drag and drop” a následné nastavení jejich parametrů. I tento programovací nástroj umožňuje programovat prvky i od jiných výrobců, splňují-li normu KNX.

2.1.3 Big Clown

Je řešení vyvíjené jako vedlejší projekt firmy Jablotron.[8] Ve své podstatě může být k automatizaci použit pouze jediný modul z tohoto řešení a lze ho tak řadit i k řešením kompaktním, v praxi se však počítá s použitím většího množství modulů a tak se nachází

v této sekci. Jednotlivé hardwarové komponenty tohoto řešení je možné zakoupit ze stránek výrobce [9], software je veřejně dostupný pod licencí MIT. Základním prvkem systému je “node” čili uzel, sestávající z volitelné kombinace zařízení, jako jsou modul pro napájení, senzorické moduly a komunikační moduly. Základem každého “node” je takzvaný “Core module” čili jádrový modul. (Obr. 2.4) Jedná se o jednočipový procesor s jádrem Cortex-M0+, konkrétně jde o STM32L083CZ. Jde o obvod z řady procesorů STM32 a oproti Arduinu má větší paměť jak pro proměnné, tak pro program. Podobně jako u Arduina Nano je na něm integrováno USB s „bootloaderem“, což je součást nutná pro nahrávání programu do zařízení. Obsahuje i dvě komponenty pro kryptografii, a to TRNG a AES-128. Výrobce udává, že právě zabezpečení má být silnou stránkou tohoto řešení. Přímo na čipu “Core module” je integrován teploměr TMP112, tříosý akcelerometr LIS2DH12 a bezdrátový komunikační modul SPSGRF. Sběrnice umístěné na modulu jsou I2C, SPI a 1-Wire. Podobně jako u Arduina se zde nacházejí i digitální vstupně/výstupní piny, není jich však k dispozici takové množství. K “Core module” lze připojit další moduly, jedním z nich je například “Tag module”, sloužící k připojení maximálně šesti zařízení komunikujících po sběrnici I2C. Těmito zařízeními jsou desky osazené senzory intenzity osvětlení, vlhkosti, teploměr, nebo Co2 snímač. Může se jednat ale i o další moduly. Jádrový modul lze připojit k počítačům Raspberry Pi, nebo Turris Omnia, lze jej však připojit i do bezdrátové sítě využívající frekvenci 868MHz nebo 915MHz. Více jádrových modulů v tomto případě komunikuje s jedním jádrovým modulem, který je připojen k počítači. Systém je možné propojit s internetem a službami jako Azure IoT, Amazon IoT, Thingspeak či s vlastním řešením. [8]

Obr. 2.4 jádrový modul řešení Big Clown [8]

Komunikačním standardem používaným v tomto řešení je MQTT. Přenos probíhá pomocí TCP. Používaným návrhovým vzorem je „publisher – subscriber“, což znamená, že existuje jeden centrální bod, který předává zprávy mezi klienty. Zprávy jsou tříděny do takzvaných témat a zařízení buď publikuje v daném tématu, čili posílá data do centrálního bodu (nazvaného jako “broker”), nebo je přihlášeno k odběru témat, čili „broker“ všechny zprávy s daným tématem posílá do tohoto zařízení. Jedno zařízení může zároveň některé zprávy přijímat a jiné odebírat. Obsah zprávy není nijak předem určen, jedná se přímo o binární data. [8]

Oproti ostatním komplexním řešením počítá Big Clown s tím, že zvolená zařízení bude programovat přímo uživatel ve vývojovém prostředí vyvinutém pro Big Clown. Pro naprogramování vlastního software je tedy nutné se v tomto vývojovém prostředí naučit pracovat. Programovací jazyk vychází z C++ a podobně jako u Arduina jsou do něj přidány vlastní příkazy. Ty jsou ve srovnání s Arduinem delší a složitější, což může být pro začínající uživatele nevýhodou. Výhodou je naopak dokumentace a podpora dostupná ze stránek výrobce.
Cena: 90 - 2500Kč za modul [9]

2.2 Kompaktní řešení

Za kompaktní řešení jsou v této práci označena zařízení, jež ke své plnohodnotné funkci nepotřebují žádná další speciální zařízení. Vystačí si tedy pouze s napájením, nějakým způsobem připojení pro přenos dat a terminálem pro jejich obsluhu, který je často reprezentován aplikací pro chytré telefony nebo osobní počítače, případně obojí. Kompaktní řešení bývají většinou jednoúčelová, případně je oblast jejich použití značně omezena. Jedná se o zásuvky s bluetooth připojením, inteligentní konvice, ale lze sem zařadit i domovní asistenty, jako například Google Home [10] od společnosti Google, u nichž je možnost použití víceúčelová, pro provoz jim však stačí pouze jedno kompaktní zařízení. V žádném případě tedy nelze říci, že by řešení kompaktní nebyla dále rozšiřitelná, případně nebyly dodávány rozšiřující moduly a kompatibilní aplikace, pouze tyto moduly nejsou pro základní plánovanou funkčnost zařízení nezbytné. Dalším charakteristickým znakem kompaktních řešení bývá nemožnost změny jejich softwaru uživatelem. Zařízení je dodáváno s jednou už předem definovanou funkcí a s ní spojeným programem, ke kterému nemá uživatel přístup. Vzhledem k tomu, že se kompaktní řešení zabývají pouze určitou omezenou oblastí automatizace, bývá jejich cena oproti zařízením komplexním nižší. Toto lze však říci jen při srovnání pořizovací ceny určitého kompaktního řešení a nasazení komplexního řešení pro jednu určitou činnost, jde-li o automatizaci více různých činností, případně automatizaci určité činnosti na větší rozloze, nelze toto již tvrdit. Rovněž náročnost jejich instalace bývá malá. Kompaktní řešení jsou navrhována tak, aby pro jejich zprovoznění po zakoupení nebylo potřeba dalších odborníků a jejich uživatelské prostředí bývá snadno ovladatelné pro běžného uživatele. Za nevýhodu lze označit častá nekompatibilita mezí jednotlivými výrobci. Jsou-li v jednom bytě instalovány například inteligentní zásuvky od jednoho výrobce a chytrá konvice od jiného výrobce, je velice pravděpodobné, že obě zařízení nepůjde ovládat prostřednictvím jedné mobilní aplikace, ale pro jejich obsluhu bude nutné nainstalovat aplikace dvě a pak mezi nimi přepínat.

2.2.1 Elgato Eve Energy EU

Jedná se o bezdrátový senzor a zásuvku v jednom pouzdře. (Obr. 2.5) Zařízení se propojí mezi instalovanou zásuvku a spotřebič, k instalaci nejsou potřeba odborné znalosti ani žádné další komponenty, nebo nástroje. Takto vybavenou zásuvku pak lze spínat a

vypínat například pomocí mobilního telefonu, spárovaného pomocí Bluetooth přes aplikaci HomeKit, která slouží k ovládání i dalších Apple produktů pro inteligentní domácnosti. [11] Zásuvka měří informace o proudu a napětí. Tyto informace je možné vykreslit do grafu na displeji mobilního telefonu. Inteligentní zásuvka je kompatibilní s produkty značky Apple a lze ji obsluhovat pomocí virtuální asistentky Siri, takže zásuvka reaguje mimo klasické ovládání i na hlasové povely. Nevýhodou je, že kompatibilita s jinými zařízeními zcela chybí, takže bez chytrého telefonu, nebo tabletu značky Apple není možné zásuvku ovládat. Zásuvka je kompatibilní s českými zástrčkami i zásuvkami a lze ji u nás zakoupit. Maximální dovolený proud zásuvkou je 11A. Přenos dat přes Bluetooth je šifrován, takže by zásuvka měla být chráněna proti neautorizovanému ovládání. [11] Zjištěné technické parametry lze vidět na následující tabulce. (Tab. 2.1)

Šířka	49 mm
Přenos dat	Jednosměrný
Bezdrátový standard	Bluetooth
Hloubka	73 mm
Komponenty Smart Home	Bezdrátový spínač
Výška	49 mm
Provozní napětí	230 V
Zatížení:	maximálně 11 A / 2500 W
Cena:	1350Kč

Tab. 2.1 technické parametry chytré zásuvky [11]

Obr. 2.5 chytrá zásuvka [11]

2.2.2 Rychlovarná konvice od firmy Xiaomi

Rychlovarná konvice (Obr. 2.6), kterou je možné ovládat prostřednictvím chytrého telefonu pomocí aplikace MiHome. V aplikaci lze nastavit na jakou teplotu se má ohřát voda v konvici a jestli má na této teplotě setrvávat, či se po dosažení teploty vypnout. V aplikaci je také možné sledovat aktuální teplotu vody v konvici. Konvice komunikuje s telefonem prostřednictvím Bluetooth a dokáže udržovat vodu na požadované teplotě nejvýše 12 hodin. Aplikace je kompatibilní s mobilními telefony s operačním systémem Android. Nalít vodu do konvice a ohřátou jí z ní opět vylít musí uživatel. Tento proces zatím není nijak automatizován a dle dostupných informací jeho automatizace ani není plánována. Společnost vyrábějící konvici zaručuje výdrž konvice na nejméně 10 let běžného provozu. Vnitřek konvice je nerezový, vnější obal je plastový a mezi nimi je tenká vrstva vzduchu sloužící jako tepelná izolace. Víko konvice lze otevřít do polohy 45°, nebo 80° pro lepší čištění. Konvice uvaří 1,5l vody za 4 minuty, přičemž toto množství je zároveň jejím maximálním objemem. (Tab. 2.2) Kabel ke konvici je opatřen navijákem, čímž lze regulovat jeho délku. [12] Konvici momentálně výrobce na území

České republiky nenabízí, je jí však možné zakoupit v zahraničí. Konvici lze kromě automatizace budov řadit i do takzvaného internetu věcí, jako běžný domovní spotřebič však do automatizace budov nepochybně také patří.

Vnitřní průměr	13 cm
Bezdrátový standard	Bluetooth
Objem	1,5 l
Provozní napětí	230 V
Orientační cena	30 dolarů

Tab. 2.2 technické parametry inteligentní konvice [12]

Obr. 2.6 inteligentní konvice [12]

2.2.3 Google Home

Jedná se o jednoho z domácích asistentů, kterých se v posledních letech objevilo již více. Pojmem domácí asistent je v tomto případě myšleno digitální zařízení, které prostřednictvím senzorů poslouchá komunikaci uživatelů a pokud ji vyhodnotí jako určenou sobě, odpoví, nebo vykoná některou z dalších přednastavených akcí. Takové zařízení je často napojené na neuronovou síť a může tak průběžně měnit své reakce s cílem zlepšit se a přizpůsobit se uživateli.

Pouzdrem Google Home je kulatá bílá krabice se čtyřmi diodami, ovládaná hlasem. Kromě diod se na ní nachází ještě reproduktor, mikrofon a tlačítko pro umlčení. Software je tvořen Google asistentem, který je už nyní běžný v mobilních telefonech Android. Pro plnohodnotnou funkci zařízení je nutné jeho připojení k internetu. S ním dokáže asistent odpovídat na různé otázky, zjistit aktuální počasí, ukládat a načítat události z kalendáře, spustit překladač, nebo přehrát hudbu. Je-li k dispozici domácí termostat Nest, je možné s ním asistenta propojit. Plánované je propojení i s dalšími zařízeními, jako je televizor, nebo ovládání světel. Zařízení je zatím stále ve vývoji, a tak nejsou známy bližší technické specifikace ani konečná cena. [10]

Obr. 2.7 Google Home [10]

3 MOTIVACE K VÝVOJI VLASTNÍHO ŘEŠENÍ

Z předchozí kapitoly vyplývá, že existující řešení automatizace budov jsou většinou buď náročná na zprovoznění a instalaci, nebo se zabývají pouze jednou určitou funkcí. Může tedy nastat okamžik, kdy uživatel potřebuje zautomatizovat pouze jeden prvek, případně soubor malého množství prvků v oblasti automatizace budov a kompaktní řešení pro jeho konkrétní problém nelze sehnat. Poté musí přistoupit k řešení komplexnímu, kde však rapidně roste cena a náročnost instalace. Například pokud by chtěl uživatel zautomatizovat pouze jeden pokoj z celé budovy, má na výběr buď umístit do něj množství vzájemně nekompatibilních řešení a ty nezávisle na sobě používat, nebo umístit do pokoje rozvaděč a předělat veškerá dosavadní vedení pro umístění řešení komplexního. Bylo tedy rozhodnuto o tvorbě vlastního řešení, které by mělo pokrývat přesně tuto oblast. Takové řešení by mělo být levné na pořízení a díly použité k jeho výrobě by měly být běžně dostupné například v internetových obchodech. I jejich spojení a zprovoznění by mělo být snadné, aby instalaci zvládl sám uživatel, bude-li alespoň nějak technicky zdatný. Stejně tak programování by mělo probíhat nejlépe pomocí grafického editoru, aby nebylo nutné učit se pro zprovoznění programovací jazyk, a to buď existující, nebo vlastní. Je totiž možné, že při vývoji takového řešení se nutnost, či spíše vhodnost nového programovacího jazyka projeví. Naopak nebylo cílem stvořit zařízení, které by bylo určené k automatizaci složitých a náročných řešení, případně řešení značně rozsáhlých, jako jsou celé komplexy budov. Veškerá dokumentace k takovému zařízení by pak měla být dostupná veřejně a celé zařízení není zamýšleno jako komerční produkt, spíše určeno technickým nadšencům a kutilům, kteří ho budou schopni sami dle návodu zkonstruovat, případně dokonce rozšířit. Jako návod může sloužit i tato práce, kde je navíc zaručena i veřejná dostupnost. V průběhu práce na tomto řešení byly prostudovány podobné projekty, které podobně jako tento cílí přímo na koncové uživatele a při jejich zavádění do praxe počítají s jistou technickou zručností. Jedním z těchto řešení je například systém pro automatizaci budov od firmy Big Clown [9]. Oproti řešení popisovanému v této práci nabízí více funkcí a lepší bezpečnost, naopak jeho nevýhodou je vyšší cena za jednotlivé komponenty a nutnost znalosti programovacího prostředí, které je na rozdíl od grafického programovacího prostředí, navrženého přímo pro tuto práci, méně uživatelsky přívětivé. Z těchto skutečností bylo odvozeno, že zvolené řešení cílí na jinou uživatelskou skupinu, než řešení podobné a poskytuje oproti nim jisté výhody. Ve vývoji tohoto řešení se tedy dalo pokračovat.

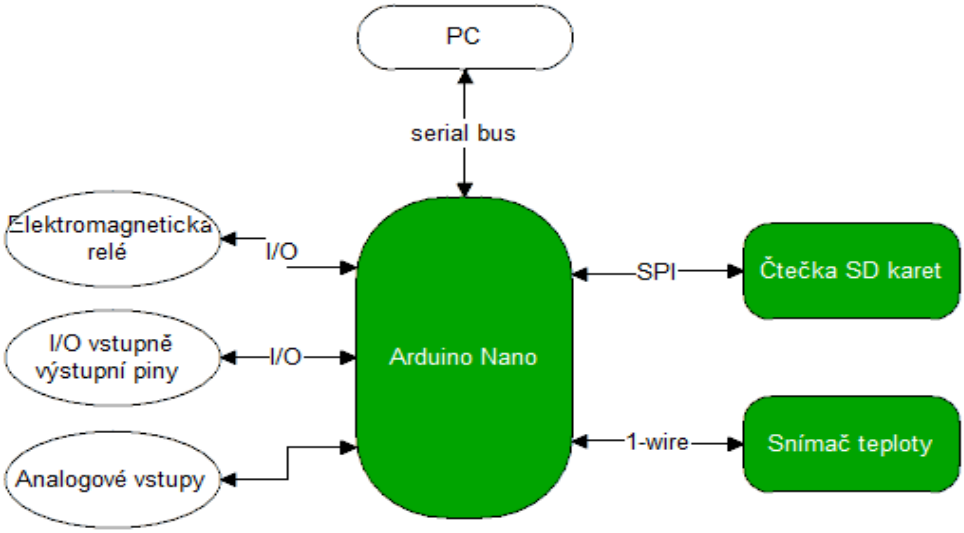
Navržené zařízení by mělo umět spínat relé pro připojení prvků používajících napětí 230V, samo napájet binární zobrazovací prvky, jako například diody a komunikovat s počítačem pro vypsání chyb, případně spuštění složitějších procesů umístěných například přímo v počítači, nebo posílání dat na web. Vhodné by bylo, kdyby bylo zařízení opatřeno i displejem, vzhledem k povaze zařízení to však není nezbytné. Vstupy by byly binární, ale i analogové, pro připojení například teplotních čidel PT100, vhodným dodatkem by byl už přímo připojený teploměr, s kterým by uměl program

pracovat, případně další konkrétní periferie, jako vlhkoměr. Pokud by zařízení umělo do PC data posílat, je vhodné, aby je z něj umělo i získávat a využít je například jako virtuální binární vstup. Software takového zařízení by měl pracovat s běžnými funkcemi používanými v automatizaci, jako jsou logický AND, OR, NOT, [13] čítače, časovače a další běžné prvky. Tvorba programu pro zařízení by pak měla probíhat v grafickém editoru, kde by se tyto jednotlivé prvky vyskytovaly a bylo by je možné řetězit do větších celků. Výhodou by bylo umístění programu na nějakém vyměnitelném datovém nosiči, aby bylo možné měnit funkci zařízení pouze změnou tohoto nosiče.

Na základě těchto požadavků byl zahájen vývoj zařízeni, které by je splňovalo. V průběhu řešení se pak především logika programování tohoto zařízení začala velmi podobat logice PLC a výsledné zařízení tedy dostalo pojmenování PLCduino jako kombinace slov PLC a Arduino, tvořící základ zařízení. Dále je tedy název PLCduino používán k označení výsledného zařízení.

4 HARDWARE VLASTNÍHO ŘEŠENÍ

Zařízení sestává z Arduina Nano, které se stará o vykonávání programu a k němu připojených periferií. Těmi jsou čtečka SD karet, na níž se nachází instrukce řídící běh programu, vstupně výstupní piny, elektromagnetická relé sloužící k spínání výstupů o vyšší zátěži, než jakou je schopno napájet samo Arduino a digitální teploměr. Celé řešení je pojato jako návrh zařízení, který je do jisté míry modifikovatelný uživatelem. Počítá se s tím, že každý, kdo by se v této práci navržené a sestavené zařízení pokusil replikovat, ho v potřebné míře upraví pro své účely. Počítá se i s možností přidání dalších kompatibilních periferií. Všechny použité součásti jsou proto běžně dostupné a jejich přesná montáž by měla být možná i v domácích podmínkách. Přehled o zapojení použitých periferií dává následující schéma (Obr. 4.1) Příklad celkového zapojení Arduina [2] i se všemi sběrnicemi je pak možné vidět v příloze (Příloha A).

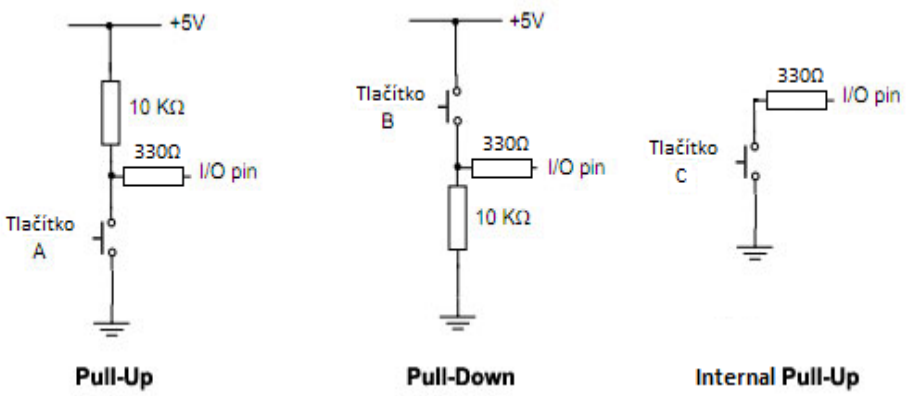


Obr. 4.1 schéma periferií a sběrnic

Po zapojení sběrnice 1-wire a sběrnice SPI zbývá Arduinu ještě 15pinů pro vlastní účely, ty mohou být použity pro připojení elektromagnetických relé nebo jako digitálních vstupně výstupních pinů. Osm z nich lze místo toho použít jako vstupy analogové. Není nutné použít přímo Arduino Nano, Arduino Uno [2] se liší pouze rozměry, ale počty pinů zůstávají stejné, tudíž je možné jím Arduino Nano plnohodnotně nahradit, pouze se tak zvětší rozměr výsledného zařízení.

Zapojení digitálních vstupů je možné provést několik způsoby, ten nejjednodušší je připojit vstup přímo na pin Arduina, je však potřeba pamatovat na to, že i s použitím napájení dodávaného Arduinem je přímo před vstupní pin nutné zařadit ještě rezistor s malou hodnotou odporu, protože jinak by proud mohl Arduino poškodit. V této práci jsou k tomuto účelu použity odpory o hodnotě 330Ω, jejichž použití se již v praxi osvědčilo. Neměl by však být problém zaměnit je za jinou, podobnou hodnotu. Při tomto zapojení

však po vypnutí vstupu může docházet k jeho náhodným opětovným zapnutím. To je dáno tím, že pin není při vypnutí v tomto zapojení připojen k žádné referenční hodnotě napětí, a tak se na něm v této pozici může vyskytovat šum. Zamezení výše popsaného problému je možné takzvaným "PULL-UP" nebo "PULL-DOWN" zapojením (Obr. 4.2). Při „PULL-UP" zapojení je v klidovém stavu pin takzvaně tažen k napájecímu napětí. (V případě této práce je z důvodu použitého hardwaru jako toto napětí všude používáno napětí o hodnotě 5V.) To znamená, že je vstup třeba spínat jeho uzemněním k hodnotě napětí 0V. Takto zapojený pin se tedy chová inverzně a pokud je sepnut, posílá do zařízení logickou nulu, pokud je vypnut, tak logickou jedna. Zapojení "PULL-DOWN" funguje opačně a v klidovém stavu je tak vstup tažen k napětí 0V. Funguje tedy tak, že pokud je vstup sepnutý, posílá do zařízení logickou jedna a pokud vypnutý posílá logickou nulu. V praxi se tohoto zapojení dosáhne tak, že se pin připojí do série s uzemněním přes odpor o hodnotě například 10kΩ. Vstup je pak k tomuto pinu zapojen paralelně vůči tomuto odporu. U zapojení "PULL-UP" je vše podobné, jen je pin spojen do série s napětím 5V a sepnutí vstupu, například tlačítka, ho uzemní. [14]



Obr. 4.2 náhled používaných zapojení I/O pinů [15]

V obou případech je zamezeno vznikání šumu na vstupu z předem uvedených důvodů. Obě tato zapojení však mají určitou nevýhodu, že při navrhování konkrétního zapojení zařízení je nutné mít již předem rozmyšlené, které 1/0 piny budou použity jako vstupy, a které jako výstupy, protože připojení výstupu na vstup ani v jednom zapojení není možné. Tento problém je však v Arduinu vyřešen možností softwarově aktivovat interní "PULL-UP" rezistor. V tomto případě se vnější zapojení omezí na poslední schéma na (Obr. 4.2) a takto vybavený vstup je možné dodatečně zaměnit za výstup, bez změny plošného spoje, za předpokladu, že je na něj software připraven a funkci výstupu neomezuje sériové zařazení odporu o hodnotě 330Ω k tomuto výstupu. Bohužel v praxi toto zapojení nefunguje, neboť šum na pinech vzniká stále, alespoň co se týče v této práci používaného klonu Arduina - Sanduino. Z jakého důvodu tento problém vzniká nebylo zjištěno, neboť v dokumentaci se o tomto problému nic nepíše. Z tohoto důvodu bylo zvoleno řešení, kdy je nutné zapojit vstupní piny jako "PULL-DOWN" a mít tedy dopředu rozmyšlené, jaké piny budou ve finální podobě zařízení vstupní a jaké výstupní. [16]

4.1 Arduino Nano

Při výběru hlavního procesoru celého projektu připadalo v úvahu několik řešení. Jednou z podmínek při výběru bylo, aby tento procesor byl už součástí vývojové desky, měl zabudované připojení pro MicroUSB a jeho součástí byl programátor. Pokud má být celé řešení volně přístupné a použitelné jako alternativa k automatizaci budov pro technicky nadané uživatele, je vhodné, aby jednotlivé části hardware byly běžně dostupné, buď v tuzemských obchodech, nebo prostřednictvím internetových portálů a jejich spojení do finálního celku nebylo náročné jak na vybavení, tak znalosti. Nakonec byla pro tento projekt zvolena vývojová deska s názvem Sanduino, což je totožná kopie Arduino Nano [17] (Obr. 4.3). Výhodou této desky je kompatibilita s deskou Arduino UNO, kdy jak piny, tak základní čip jsou totožné, desky se od sebe liší pouze velikostí a periferiemi, kdy například u Arduino Nano, není součástí desky otvor pro napájení. Dalším důležitým parametrem byla velikost a Nano zde svými rozměry zcela vyhovuje – je však možné jej úspěšně zaměnit za Arduino UNO bez nutnosti úprav zapojení, pouze se tak celé zařízení zvětší, a to bez jakéhokoliv zlepšení parametrů. [2] Další podstatnou výhodou platformy Arduino je jeho rozšířenost a z toho plynoucí dostupnost návodů a knihoven. Kdyby chtěl kdokoliv další do programu zasahovat, najde v porovnání s jinými vývojovými deskami množství odladěných knihoven. Navíc je programátorské prostředí Arduina – Arduino IDE dostupné volně ke stažení. V neposlední řadě byla klíčovým parametrem cena, ta se u tohoto klonu Arduina pohybuje okolo 50 -70Kč (počítáno při nákupu z Číny, přes portál *Ebay.com*). Nevýhodou tohoto řešení je pak značně malá paměť, jak pro program (32kB), tak pro proměnné (2kB). Do této paměti se však celé řešení nakonec úspěšně vešlo a ještě zbylo místo pro jeho rozšíření, pro které zbývá zhruba 30-35% celkové paměti.

Obr. 4.3 Arduino Nano [18]

Základem Arduina Nano je procesor Atmega328. [19] Programování probíhá přes sériovou linku UART, fyzicky je Arduino osazeno konektorem Micro USB, přes který se i programuje. Pokud probíhá nahrávání programu do procesoru, není možné, aby po této sériové lince probíhala jakákoliv další komunikace. Sériová linka je vybavena vlastním

bufferem, jenž je možné ovládat pomocí software nahraném na Arduinu. Arduino Nano je osazeno jedním tlačítkem pro jeho fyzický restart a čtyřmi diodami. Odshora dolů, při pohledu shora a umístění textu na Arduinu tak, aby byl čitelný, jsou to dvě diody signalizující komunikaci, vždy jedna pro každý směr. Dále dioda zobrazující, je-li Arduino pod napětím a nakonec dioda na pinu 13, kterou lze využít k vlastním účelům. K dalším periferiím patří sběrnice ISP, nacházející se na pinech 10-13, jenž zde bude rozepsána podrobněji v dalších kapitolách, sběrnice 1-Wire (opět využitá v tomto projektu) a celkem 20 vstupně výstupních pinů, z nichž 8 může být použito jako analogové vstupy, zbytek je digitální, avšak jako digitální lze použít i piny analogové. 6 pinů lze použít jako PWM, což však v tomto projektu není využito. Arduino může být napájeno napětím od 3,3V do 5V, maximální výstupní proud je 200mA. Maximální výstupní proud na jeden pin je 40mA, ale doporučený pouze 20mA. Maximálním odběrem na více než pěti pinech je tedy možné Arduino zničit. Umístění pinů a jejich význam je vidět na schématu v příloze (Příloha B).

4.2 Čtečka Micro SD karet na sběrnici SPI

Bylo zvoleno řešení od výrobce Catalex s názvem MicroSD card adapter (Obr. 4.4) a to především pro své kompaktní rozměry, které jsou 4,2x2,3 cm. Důležitou roli hrála také dostupnost knihoven, které jsou pro toto zařízení ke stažení přímo v Arduino IDE. Cena tohoto zařízení se pohybuje na portálu *Ebay.com* okolo 17Kč, včetně poštovného. Tato čtečka je již vybavena sběrnicí SPI a k jejímu zprovoznění je tedy nutné správně připojit piny SCK, MISO, MOSI, VCC, GND a CS. Čtečka je napájena přímo z Arduina a to napětím 5V. které je pak přímo na adaptéru převedeno na 3,3V pomocí konverzního čipu. S takto sníženým napětím pak čtečka dále pracuje. Podporovány jsou karty standartu Micro SD a Micro SDHC, přičemž adaptér je schopen jak čtení, tak zápisu dat. Na desce adaptéru jsou umístěny 4 montážní díry pro šrouby M2. Držák karty je vybaven mechanismem pro její zajištění po zasunutí a pružinou pro její vysunutí.

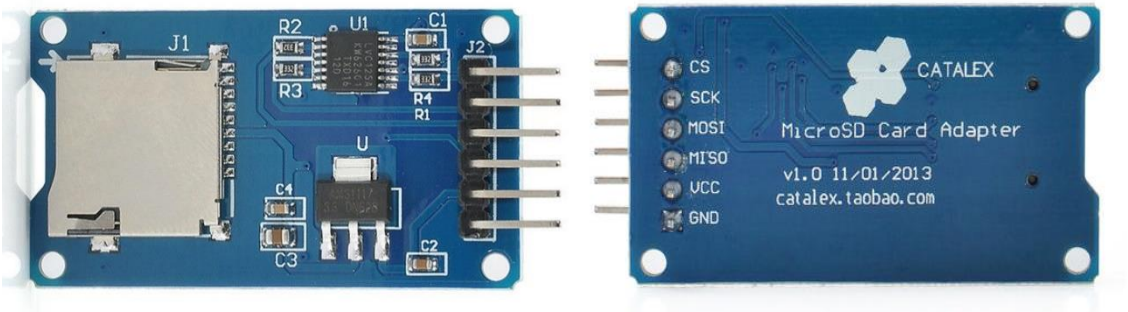
Sběrnice SPI je synchronní sběrnice řízená hodinovým signálem (SCK) s maximální možnou frekvencí až 70MHz, což je rychlost několikrát převyšující potřeby navrhovaného zařízení. Sběrnice zvládá komunikovat s více uzly, přičemž k jejich výběru slouží takzvaný Chip Select - CS. Jeden z uzlů je pak vždy typu Master – neboli takzvaný řadič sběrnice, který zároveň vysílá hodinový signál, jímž se ostatní uzly řídí. Sběrnice umožňuje plně duplexní komunikaci „full duplex“, v tomto projektu je však použita pouze komunikace půl duplexní „half duplex“. Výhodou této sběrnice je, že každý použitý port (pin) je vždy pouze jednosměrný. Výčet pinů a jejich popis lze přehledně vidět v následující tabulce. (Tab. 4.1)

Mezi výhody SPI patří její jednoduchost a snadná implementace. Jednoduché je elektrické rozhraní sběrnice, ale i přenosový protokol, který na rozdíl například od sběrnice RS-323C nevyžaduje žádné start, ani stop bity. Tím že komunikace probíhá zcela synchronně a po jednom vodiči vždy v jednom směru není ani nutné řešit přepínání mezi vysílání a příjmem. [20]

VCC	napájení, 5V
GND	zem, 0V
MISO	Master IN, Slave OUT
MOSI	Master OUT, Slave IN
CS	výběr slave chipu
SCK	hodinový signál

Tab. 4.1 přehled kabelů na sběrnici SPI

Naopak mezi nevýhody patří, že pouze jedno zařízení může být typu Master, to však v tomto konkrétním případě, kdy jsou na sběrnici připojeny pouze dvě zařízení není problém. Další nevýhodou sběrnice je, že je možné ji používat pouze na malé vzdálenosti, je totiž nutné aby jak data tak hodinový signál dorazili ve stejný čas a nedošlo k jejich desynchronizaci. Poslední pak poměrně velké množství kabelů a tím vzrůstající cena instalace sběrnice, což však není při použití na malé vzdálenosti zásadní problém. [20]



Obr. 4.4 Čtečka SD karet s označením pinů [21]

4.3 Teploměr na sběrnici 1-Wire

PLCduino je osazeno jedním digitálním teplotním snímačem Dallas Temperature Sensor DS18B20 (Obr. 4.5), po menších úpravách programu by však mělo být možné na stejné sběrnici použít teploměrů mnohem více, což však nebylo pro řešení vyhodnoceno jako potřebné. Jednou z hlavních výhod snímače je, že již obsahuje logiku, která se stará o převod měřeného napětí na stupně a výstup je tedy přímo ve stupních, což usnadňuje práci se zařízením a šetří výpočetní výkon Arduina, kterého je již tak omezené množství.

Snímač s Arduinem komunikuje digitálně. V zařízení je instalován teplotní snímač ve vodě odolném provedení s přívodním kabelem dlouhým 1m, vyrábí se ale i varianta bez vodě odolného provedení, která je rozměrově mnohem menší, avšak měla by být zcela kompatibilní, neboť řídící elektronika je totožná, [20] při instalaci teplotního čidla přímo ve schránce PLCduina by však mohlo dojít k ovlivnění měření ohřevem od procesoru a proto toto řešení není použito a ani se jeho použití nedoporučuje. Důležité technické parametry jsou pro lepší přehlednost opět uvedeny v tabulce a v textu již nejsou znovu vypsány. (Tab. 4.2)

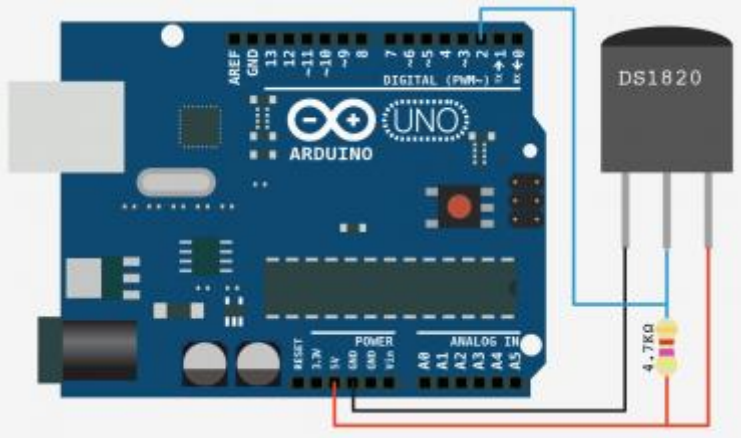
Minimální měřitelná teplota	-55°C
Maximální měřitelná teplota	+125°C
Přesnost	+/- 0,5°C
Rozsah vstupního napětí	3V - 5,5V
Velikost těla snímače	6 X 50 mm
Průměrný napájecí proud	1mA
Čas potřebný ke změření teploty	750ms

Tab. 4.2 přehled technických parametrů teplotního snímače

Obr. 4.5 digitální snímač teploty s vysvětlením barvy vodičů [22]

Snímač vyžaduje ke správné funkci tři vodiče. VCC pro napájení snímače napětím 5V. (Je možné použít napájení přímo z Arduina.) GND s napětím 0V pro uzemnění a signálový vodič DQ, připojený na pin D2, kde se u použitého Arduina Nano počítá s použitím sběrnice 1-Wire. Při připojení signálového vodiče na jiný pin je zapojení nefunkční. (Obr. 4.6) Na daném obrázku je pro lepší názornost připojení k Arduinu UNO, ale u Arduina Nano je zapojení analogické. Pro správnou funkci snímače jsou vodiče DQ a VCC spojeny přes rezistor o hodnotě 4,7kΩ. [16] Snímač je možné zapojit i takzvaně parazitně, kdy se vodiče VCC a GND spojí do jednoho a snímač je pak napájen přes signálový vodič, čímž lze ušetřit jeden vodič, což je výhodné, pokud by byl teplotní snímač daleko od Arduina. Toto zapojení je umožněno kondenzátorem umístěným ve

snímači, který napájí zařízení během doby, kdy jsou po datovém kabelu posílány data a tak nemůže sloužit k napájení. Při tomto zapojení je však sběrnice mnohem citlivější na okolní rušení a i vliv materiálu a průřezu vodičů na měření se zvýší. [20]



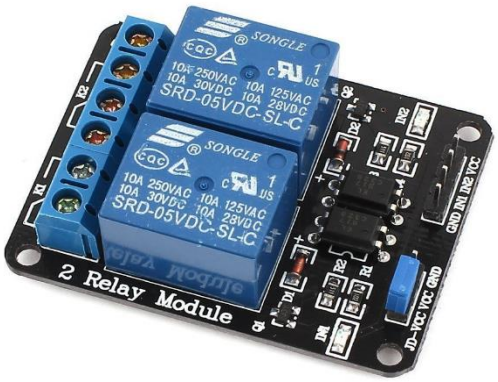
Obr. 4.6. korektní zapojení snímače k Arduinu Uno [20]

Sběrnice 1-Wire je sběrnice navržená firmou Dallas Semiconductor corporation pro komunikaci zařízení s nízkou datovou rychlostí na velké vzdálenosti. V tomto případě je Arduino připojeno jako Master a teplotní snímač jako Slave. Vysílání začíná pomlkou, kdy je datový vodič stáhnut k nule. To ho restartuje a Master začíná vysílat, po odesláni dat z Masteru Slave odpoví. V případě, kdy by bylo použito více zařízení na jedné sběrnici, bylo by nutné provést ještě výběr Slave zařízení, s kterým se má komunikovat, tato možnost však nebyla využita. Data jsou proti ztrátě zabezpečena technologií CRC. Maximální délka sběrnice testovaná výrobcem je 300m, je však možné že sběrnice bude fungovat i na větší vzdálenosti. [23]

4.4 Výstupní relé

Pro spínání větších zátěží, než které je schopné napájet přímo Arduino je použit reléový modul s dvěma elektromagnetickými relé (Obr. 4.7). Maximální proud každým z nich na spínané straně je 10A při 250V střídavého napětí, nebo 10A při 30V stejnosměrného napětí. Modul je napájen napětím 5V přímo z Arduina, to je přímo na modulu snižováno na 1,5V s kterým pak modul dále pracuje. Příkon každého z relé je 15-20mA. Rozměry jsou 50x38x17mm, deska je vybavena čtyřmi montážními otvory pro šrouby. Přímo u relé se nachází svorkovnice k připojení spínaného obvodu, což značně usnadňuje montáž. K připojení spínaného obvodu do svorkovnice je možné použít jak plochý, tak křížový šroubovák. Spínací a spínaný obvod jsou od sebe galvanicky odděleny. Modul se zprovozní po připojení GND a VCC na příslušné piny, jež jsou na modulu přehledně popsány. Po přivedení logického signálu 1 z Arduina na vstup IN1, nebo IN2 dojde k sepnutí, případně rozepnutí výstupní části relé, dle připojení spínaného obvodu na relé. Záleží na uživateli, na které piny tento modul připojí, případně použije-li modul s více relé. Rozhodně by však měl respektovat zapojení sběrnic popsané dříve v této kapitole.

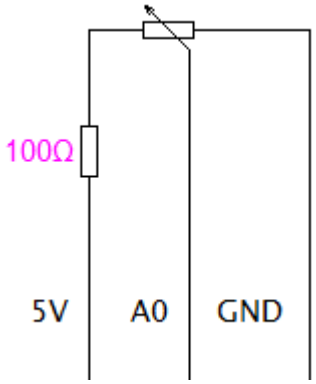
Pro připojení pinů Arduina na piny relé není nutné vkládat mezi ně žádné další zařízení a to ani jinak u vstupů doporučovaný rezistor o hodnotě 330Ω. Cena tohoto relé modulu se na internetových obchodech pohybuje okolo 30Kč. Při připojování střídavého síťového napětí na relé je třeba dbát zvýšené bezpečnosti. Jedná se o napětí, které může být člověku nebezpečné. [24]



Obr. 4.7 reléový modul s dvěma elektromagnetickými relé [25]

4.5 Analogové vstupy

Jako analogové vstupy lze využít porty Arduina označené jako A0-A7, které je však v instrukcích nutné indexovat jako piny 14-21, o čemž je více informací v kapitole "Software vlastního řešení". Arduino je schopno analogově měřit napětí mezi 0-5V. Toto napětí měří v Arduinu takzvaný "Analog-to-digital" konvertor, který ho převádí na hodnoty mezi 0 a 1023 (je tedy desetibitový). Při napětí na pinu 0V je hodnota měřená Arduinem 0, při napětí 5V je měřená hodnota 1023. Číslo měřené Arduinem tedy odpovídá poměrné hodnotě napětí na pinu. K analogovým vstupům jsou v případě Arduina nejčastěji připojovány potenciometry a teplotní čidla, jelikož řešení popisované v této práci již jeden teploměr obsahuje, počítá se především s použitím potenciometrů. [2]



Obr. 4.8 příklad zapojení potenciometru

Zapojení potenciometru je vidět na schématu (Obr. 4.8) Mezi napětí a potenciometr je vhodné zařadit ještě odpor malé hodnoty, většina potenciometrů při otočení do jedné krajní pozice nemá již žádný odpor a není vhodné pustin na pin Arduina plný proud z vývodu 5V na Arduinu, z důvodu možného zničení pinu. V tomto zapojení pak není možné dosáhnout přímo horní hodnoty měřeného napětí 1023 dílků, což však lze u většiny aplikací zanedbat. V praxi pak bylo zjištěno, že i při konstantním natočení potenciometru měřená hodnota mírně kolísá v rozsahu +/- 3 dílků s čímž je vhodné počítat při návrhu programu. Uvedená hodnota se může lišit v závislosti na použitém potenciometru a vnějších vlivech.

5 SOFTWARE VLASTNÍHO ŘEŠENÍ

Lze rozdělit do tří částí a vznikal ve třech odlišných vývojových prostředích a jazycích. Jako první byl naspán software pro Arduino Nano a to v jazyce Wiring, což je jazyk C++ pouze doplněný o příkazy přímo pro Arduino. Tento software má na starost korektní chování výsledného zařízení. Druhý jazyk není programovacím jazykem v pravém slova smyslu, jedná se ve své podstatě o data, které čte Arduino a řídí se jimi. Z tohoto důvodu byly tyto data označeny jako instrukce, ty musí být ve správném formátu a musí zachovávat danou strukturu, aby je bylo Arduino schopné korektně přečíst. Jejich řetězením dle daných pravidel pak vzniká program, jímž se PLCduino řídí. Sady instrukcí ke čtení jsou uloženy na SD kartě v textových souborech *ins.txt* a *setup.txt*. Třetím použitým jazykem byl C# a vývojovým prostředím Visual Studio, v něm je vytvořeno grafické programovací prostředí sloužící k snadnějšímu vytváření programu z jednotlivých instrukcí. Toto grafické prostředí je dále nazýváno jako SpeedAPI.

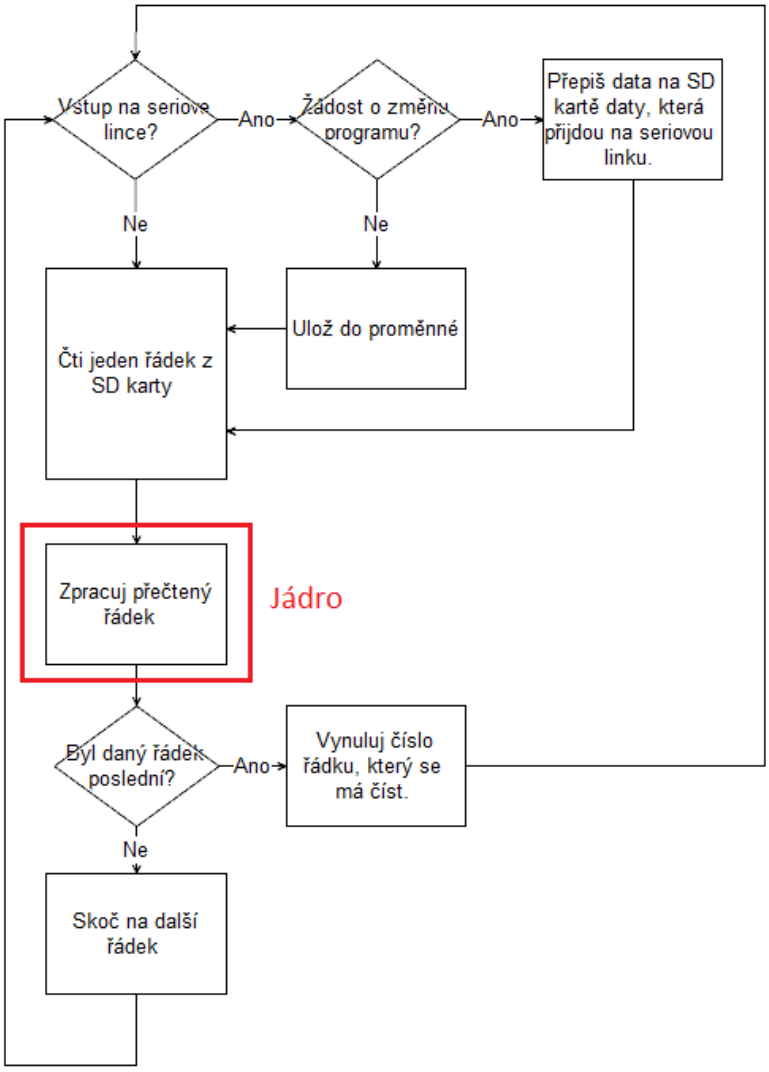
5.1 Program v Arduino Nano

Program v Arduinu je napsán v jazyce C++, ve vývojovém prostředí Arduino IDE. Arduino IDE lze stáhnout ze stránek *www.arduino.cc* a je zdarma. V tomto vývojovém prostředí lze buď rovnou psát, nebo k němu připojit externí editor. Po připojení externího editoru okno v Arduino IDE určené k psaní programu zašedne a editace programu je pak možná pouze v připojeném externím editoru. Prostředí se stará o kompilaci programu a jeho nahrání na vývojovou desku. Obsahuje i další užitečné nástroje jako vložení knihoven, nebo sériový monitor. [2]

Program na Arduinu se stará jak o komunikaci s počítačem a čtečkou datových karet, tak o provedení instrukcí uložených na datové kartě. Po své kompilaci zabírá 75 % úložného místa pro program. Proměnné zabírají 65 % místa pro proměnné. Program je jedno vláknový a probíhá v nekonečné smyčce stále dokola, jak je vidět na následujícím blokovém schématu. (Obr. 5.1)

Program nejprve zkontroluje vstup ze sériové linky. Nenalezne-li na ní nic, je celá tato část přeskočena a pokračuje se k části, která přečte a následně upraví jeden řádek z textového souboru *inx.txt* na SD kartě do tvaru zpracovatelného dále v programu. Každý řádek uložený na SD kartě v korektním tvaru je nazýván jako instrukce. V následujícím bloku se takto získaná instrukce zpracuje, případně proběhne nastavení vstupů a výstupů. Tato část se také nazývá jako jádro a je jí věnována celá samotná sekce, protože v ní probíhá hlavní logika programu. Ostatní části programu jsou pak nezbytně nutné přípravné a dokončovací práce pro toto jádro. Pokud se jednalo o poslední instrukci v souboru, skočí program na instrukci první, v opačném případě pokračuje k následující instrukci, kterou opět zpracuje. Toto schéma se opakuje neustále dokola. V případě že je na sériové lince zjištěn vstup, mohou nastat dvě varianty. Pokud se jedná pouze o takzvaný virtuální vstup, se kterým program předem počítá, uloží se jeho hodnota do

proměnné a v určitém bodě programu je použit při zpracování instrukcí v jádru. V druhém případě může být na sériové lince zaznamenána žádost o změnu dat na SD kartě. Touto žádostí jsou původní soubory *setup.txt* a *ins.txt* smazány. Data poslaná na sériovou linku za touto žádostí jsou pak uložena do nově vzniklých souborů setup.txt a *ins.txt*. Tímto způsobem je možné měnit instrukce uložené na SD kartě i za běhu programu.



Obr. 5.1 blokové schéma programu

5.1.1 Použité knihovny a proměnné

Pro chod programu je nezbytně nutné mít nainstalované některé knihovny, obsahující funkce použité v tomto projektu. Všechny potřebné knihovny jsou popsány v této části. Knihovny *SPI.h* a *SD.h* (Obr. 5.2) získá člověk spolu s instalací Arduino IDE, lze je najít v záložce *Project->include library*. Knihovna *SPI.h* není v programu nikde přímo použita, nebudou bez ní však fungovat funkce z knihovny *SD.h*. Tato knihovna je na poměry Arduina velká, při použití datového typu *FILE*, jež je nadefinován v této knihovně, zabírá program pouze s touto knihovnou zhruba 50% volného místa pro program. Na internetu lze najít a stáhnout knihovny plnící stejnou funkci o mnohem

menší velikosti, pracuje se s nimi ale odlišně, pro jejich použití by tedy byly nutné výrazné změny v kódu. Nakonec bylo rozhodnuto pro použití takto velké knihovny, protože je jakožto součást oficiální Arduino databáze knihoven dobře zdokumentovaná a odladěná. Arduino navíc obsahuje poměrně starý kompilátor, který je již dobře odladěný a kompilace programu je velmi úsporná, i při použití knihovny tedy na Arduinu zůstalo ještě dost místa pro dodatečné úpravy. Další dvě knihovny *OneWire.h* a *DallasTemperature.h* pakl bylo nutné stáhnout a před samotným přidáním do Arduina rozbalit do složky *~/arduino/libraries/*. [2]

```
1  //knihovny
2  #include <SPI.h>
3  #include <SD.h>
4  #include <OneWire.h>
5  #include <DallasTemperature.h>
6  #define ONE_WIRE_BUS 2
7  OneWire oneWire(ONE_WIRE_BUS);
8  DallasTemperature sensors(&oneWire);
```

Obr. 5.2 část programu ukazující výčet knihoven

Další částí kódu jsou proměnné. (Obr. 5.3) Celý kód i se všemi z nich je možné vidět v příloze (příloha C), zde jsou popsány pouze ty nejdůležitější. První proměnná *memory* je typu *pole long* a slouží k uchování referenční hodnoty časovače a teploměru při jeho nastavení, nebo aktuální hodnoty čítače, je však možné do ní ukládat i množství jiných hodnot, jak je nejlépe vidět v sekci věnující se instrukcím. Právě z důvodu použití této paměti i pro nastavené hodnoty teploměru není možné, aby byla proměnná *unsigned,* neboť jsou používány i záporné hodnoty. Z výše uvedeného vyplívá, že paměť pro jmenované tři prvky je společná a je tedy nutné dávat pozor při její indexaci. Další proměnou je *m_set*, zde se uchovává informace o tom, jestli je místo v paměti proměnné *memory* o daném indexu již použito. Je tedy nutné, aby toto pole mělo stejný počet míst jako pole předchozí. Podobně jako *memory* funguje *sr_block*, jen slouží jako paměť pouze pro set/reset bloky.

```
12 //globální proměnné
13 long memory[20]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
14 bool m_set[20]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
15 bool sr_block[30]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
16 long t=0; //pro ukladani casu
17 int i=0; //pro udrzeni pozice citace a casovace
18 int teplota=0; //pro teplomer
```

Obr. 5.3 důležité globální proměnné

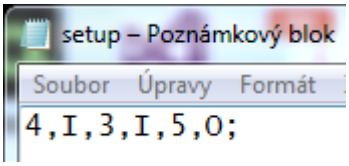
Velikost všech třech těchto proměnných je možné navýšit, pokud by to uživatel vyžadoval, program je na to připraven. Naprostá většina všech proměnných jsou proměnné globální, aby se zamezilo tomu, že při běhu programu zaberou dynamicky

vytvářené proměnné příliš mnoho místa a program zkolabuje. Mezi další důležité proměnné patří *ins_number*, *ins_param_one* a *ins_param_count*. V nich je uložena aktuálně načtená instrukce z SD karty a to tak, že každá část instrukce oddělená znakem čárky "," je uložena do jedné z proměnných v pořadí, v jakém jsou zde vypsány.

5.1.2 Komunikace s periferiemi

Nejprve se program pokusí inicializovat SD kartu, (Výpis 5.1) pokud se mu to podaří, vypíše o tom zprávu na sériovou linku a pokračuje dále, pokud ne, také o tom vypíše zprávu a dále nepokračuje, jelikož komunikace s SD kartou je po celou dobu běhu programu zcela zásadní. Na sériovou linku jsou kromě tohoto hlášení vypisovány i další informace a lze ji tedy použít jako náhled k tomu, co se uvnitř programu děje, případně kde se vyskytla chyba.

V dalším kroku se načtou data z textového souboru *setup.txt*. Zde jsou uloženy informace o tom, které piny mají být nastaveny jako vstupy a které jako výstupy. Vždy se nejprve zapisuje číslo pinu a pak na co se má nastavit. Pokud je za číslem znak velké tiskací I, nastaví se pin jako vstup, pokud velké tiskací O, nastaví se jako výstup. Čtení končí ve chvíli, kdy je v souboru nalezen středník, nebo program došel až na konec souboru. Na následujícím obrázku (Obr. 5.4) je vidět příklad správně zapsaného souboru *setup.txt*. V tomto případě se piny 4 a 3 nastaví jako vstupy a pin 5 jako výstup. Pokud by bylo potřeba využít jako vstupy, nebo výstupy i analogové piny, musí zde být očíslovány jako piny 14–21, s tím že pin 14=A0, 15=A1...až pin 21=A7. Jejich indexování standartní zápis s písmenem A by program nepřečetl korektně.



Obr. 5.4 příklad konfiguračního souboru

Tato část kódu se provede pouze jednou, vždy po restartu zařízení a dále už se neprovádí, tudíž není možné vstupy a výstupy měnit bez restartu, což však není omezení, v programu je totiž zabudován softwarový restart, [26] který se provede při nahrání nových dat za běhu programu, zapsáním zprávy *write;* na sériovou linku. Při tomto restartu se tedy vykoná opětovně zde popisovaná část programu, viditelná níže, a piny se mohou opět přestavit, byl-li nahrán jiný program *setup.txt.*

```
vypiš "Inicializace SD karty..."
pokud nebyla karta nalezena
        vypiš "Karta nenalezena..."
jinak
        vypiš "Karta inicializovana..."
        otevři setup.txt pro čtení
        dokud je v souboru co číst
                přečti aktualni byte a skoč na další
```

```
                    pokud je byte čárka
                        posuň se na další parametr
                    pokud je středník, ukonči čtení
                    jinak
                        pokud jsi na
                        prvním parametru
                            spočítej z bytů číslo pinu
                        druhém parametru
                            pokud je byte znak O
                                nastav pin jako výstup
                            pokud je byte znak I
                                nastav pin jako vstup
                    konec podmínky
                konec podmínky
                zavři soubor
            konec podmínky
```

Výpis 5.1 inicializace karty a pinů

Dále následuje kód, (Výpis 5.2) který již probíhá cyklicky do restartu, nebo vypnutí napájení zařízení, v nekonečné smyčce. Nejdříve se zkontroluje, jestli na sériovou linku nepřišly data. O to se stará standartní knihovní funkce *Serial.available()* o které je poněkud chybně v oficiální dokumentaci uvedeno, že vypisuje, kolik místa ještě zbývá v bufferu na sériové lince. Sériová linka je vybavena bufferem, který hromadí příchozí data a posílá je do Arduina tak rychle, aby je dovedlo zaznamenávat a nedocházelo tak ke ztrátě dat. Ve skutečnosti tato funkce nevypisuje kolik bytů je ještě volných, ale kolik bytů bufferu je již obsazeno, to znamená, kolik bytů právě po sériové lince dorazilo. Zjistí-li se tedy, že na sériovou linku něco přišlo, provede se následující kód.

```
pokud na sériovou linku něco přišlo
        přečti co to je, dokud nepřijde znak středníku
        pokud je to zpráva "write"
                nastav setup.txt na zápis
                vypiš na sériovou linku "Nastavujete konfigurační data."
                dokud nepřijde zpráva "end;"
                zapisuj co přijde do tohoto souboru
                konec podmínky
                nastav ins.txt na zápis
                vypiš na sériovou linku "Nastavujete program"
                dokud nepřijde zpráva "end;"
                zapisuj co přijde do tohoto souboru
                konec podmínky
        konec podmínky
konec podmínky
```

Výpis 5.2 zápis instrukcí do textového souboru

Nejprve se zjišťuje, jestli se jedná o zprávu *write;*, pokud ano je nejprve smazán soubor *setup.txt* a pak vytvořen nový prázdný se stejným jménem, do kterého jsou zapsány zaslané instrukce až do zprávy *end;*. Analogicky to funguje i pro soubor *ins.txt*. Dokud nepřijde zpráva *end;*, není pokračováno ve vykonávání ostatních částí programu. O tom, který soubor je právě upravován je uživatel informován prostřednictvím sériové linky. Zpráva *write;* nemusí být vždy zachycena na první pokus, Arduino je zaneprázdněno vykonáváním zbytku kódu a na zprávu nečeká, aby se tím vykonávání kódu příliš nezpomalovalo, je tedy možné, že se vyskytne nutnost poslat tuto zprávu vícekrát. Že byla zpráva přijata lze zjistit tak, že se na sériovou linku vypíše: "Nastavujete konfigurační data" V opačném případě se nevypíše nic a je potřeba poslat zprávu znovu.

Kromě *write;* může na seriovou linku přijít ještě zpráva *input*, (pozor, tentokrát bez středníku) po přijetí této zprávy Arduino očekává kladné celé číslo od 0 do 256, které simuluje virtuální vstup a může být použito dále v programu. Po přijetí tohoto čísla je nutné opět ukončit čekání zprávou *end;* aby se pokračovalo ve vykonávání dalšího kódu.

Soubor *ins.txt* se měl původně jmenovat *instructions.txt* a odtud tedy pochází jeho název, takto dlouhé pojmenování však nešlo použít, protože knihovna pro práci s SD kartou neumí pracovat s názvy delšími než 8 znaků.

Následuje kód pro správné čtení instrukcí ze souboru *ins.txt* jehož úkolem je každou instrukci načíst a správně rozdělit do proměnných. (Výpis 5.3)

```
načti jeden byte ze souboru ins.txt a nastav kurzor na další byte
pokud je byte čárka
        posuň se na další parametr
pokud je byte středník
        rozběhni jádro
jinak
        pokud znak není číslo
                vypiš chybové hlášení
        konec podmínky
        pokud jsi na prvním parametru
                ulož zjištěné číslo do ins_number
        pokud jsi na druhém parametru
                ulož zjištěné číslo do ins_param_one
        pokud jsi na třetím parametru
                ulož zjištěné číslo do ins_param_count
        konec podmínky
konec podmínky
```

Výpis 5.3 načítání instrukcí ze souboru

Instrukce se čtou po bytu a postupně se jimi plní proměnné. Vždy jedna proměnná je plněna byty ze souboru, dokud nepřijde znak čárka. Kód obsahuje mechanismus na přepočet těchto bytů na konkrétní číslo, který je jasně patrný v příloze (příloha C). V kódu je implementována i část, která hlídá, jestli jsou v souboru pouze platné znaky. Pokud nejsou, zobrazí se chybové hlášení, ukazující, na kterém řádku textového souboru k chybě

došlo. Toto hlášení se opět vypíše na sériovou linku. Program ze souboru nečte zbytek řádky následující za středníkem, je tedy teoreticky možné právě sem umístit poznámky k jednotlivým instrukcím. V praxi se ale počítá s použitím externího editoru, který teprve vytvoří potřebné textové soubory a tím pádem by tyto soubory měly být již bez chyb a poznámky by zde neměli být potřeba.

5.1.3 Jádro

Je část programu následující hned za komunikací s periferiemi. Neznamená to však, že by se v jádře žádná komunikace s periferiemi nemohla objevit, jen je jeho účel primárně jiný. Jádro přesto komunikuje například s digitálním teploměrem, nebo vypisuje na sériovou linku.

Jádro není ani třídou ani funkcí, jedná se pouze o další část programu, která má na starost samotné zpracování instrukcí, které jsou již teď získané z SD karty a uložené do proměnných. Jádro je vlastně jeden velký *switch* [27], kde se vždy dle proměnné *ins_number* zjistí, který jeho *case* se má vykonat a další proměnné jsou pak v tomto *case* případně použity, pokud to zrovna daný případ potřebuje. Celý program je tedy možné označit za datově řízený, protože co se stane určují právě data získaná z SD karty. Jak přesně která část jádra funguje je popsáno v samostatné kapitole řídící instrukce, protože pro ovládání PLCduina je znalost jednotlivých částí *switche* a tím pádem i znalost správného vytváření instrukcí naprosto klíčová.

5.2 Řídící instrukce

Instrukce řídící PLCduino jsou uloženy na SD kartě, mimo samotné Arduino, které se stará o zpracování instrukcí. Arduino cyklicky čte textový soubor *ins.txt* a nastavuje svoje vnitřní proměnné a výstupy podle instrukcí v něm obsažených. Program v Arduinu je tedy daty řízený. Pro lepší orientaci jsou tyto instrukce pojmenovány jako Speed, ale nejedná se o programovací jazyk v pravém slova smyslu.

5.2.1 Veličina logický signál

Zásadní proměnou při čtení instrukcí je logický signál. Ten nabývá hodnot 0 nebo 1 (je typu *bool*) a má vliv na to, jak zafunguje následující instrukce. Podle něj lze rozdělit instrukce do tří hlavních skupin:

1. **instrukce pouze ovlivňující hodnotu logického signálu** – zdroj, NOT, vstupy, OR, virtuální vstupy, sledování časovače, sledování čítače
2. **instrukce měnící pouze paměť a výstupy** – hodnotu logického signálu nemění, vykonávají nějakou jinou funkci – výstupy, nastavení časovačů, přičítání do čítače, matematické operace s pamětí, přidávání čísla do paměti, zobrazení paměti, uložení analogových vstupů, restart paměti, virtuální výstupy
3. **kombinované instrukce** – například *SET/RESET, teploměr* a ve SpeedAPI pak instrukce vzniklé sdružením více základních instrukcí (typicky sdružení instrukcí *sledování čítače* a *přičítání do čítače* v jednu instrukci s názvem *čítač*)

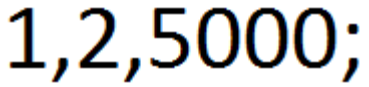
Při čtení instrukcí se tedy mění proměnná logický signál dle právě prováděné instrukce a ovlivňuje vždy instrukci po ní přímo následující.

Instrukce se dále liší tím, jak reagují na změnu logického signálu. Existují instrukce, které se řídí náběžnou hranou logického signálu a instrukce, které se řídí pouze hodnotou logického signálu. Jsou-li zanedbány instrukce *OR* a *NOT*, lze s jistou mírou zjednodušení rozdělit instrukce do dvou skupin, dle toho, jestli reagují na náběžnou hranu logického signálu, nebo jen na jeho hodnotu:

1. **instrukce reagující na náběžnou hranu** – jsou vždy instrukce operující s pamětí pro čítače, časovače a teploměr. Na začátku běhu programu se taková instrukce jednou provede, pokud je do ní přiveden logický signál o hodnotě 1, čímž se inicializuje místo v paměti, na nějž odkazuje. Aby se tato instrukce provedla znovu, musí instrukcí projít logický signál o hodnotě 0, čímž se zruší inicializace daného paměťového slotu, ale hodnota uložená na něm zůstane. Projde-li tedy neinicializovanou instrukcí znovu logický signál o hodnotě 1, instrukce se opět jednou provede. Tyto instrukce jsou: *přičítání k čítači po jedné, inicializace časovače a přičtení čísla do paměti*. Používání těchto instrukcí může být složité pro pochopení a doporučuje se tedy jen pokročilým uživatelům programu, existují však takové případy, kdy se praktické aplikace bez těchto instrukcí neobejdou.
2. **instrukce reagující na hodnotu logického signálu** – vykonají určitou činnost pokud je hodnota logického signálu 1 a pokud není, vykonají činnost jinou. Tyto instrukce jsou všechny ostatní.

5.2.2 Význam částí instrukce

Každá instrukce se skládá z řídících znaků a vlastních instrukcí, které jsou ve formátu celých čísel. V tomto formátu nejsou pro většinu lidí příliš intuitivně čitelné a nepočítá se tedy s tím, že by někdo programoval zařízení přímo, bez použití nějakého externího editoru. Jsou zapsány ve formátu vhodném především pro strojovou čitelnost a výpočetní rychlost. Každá instrukce musí začínat na novém řádku a být ukončena znakem středníku. (Obr. 5.5)



Obr. 5.5 příklad instrukce

První číslo – značí číslo instrukce, podle něj kód v Arduinu pozná, jaký úsek kódu má vykonávat, ostatní čísla jsou pak parametry této instrukce. (například: 1 - nastav časovač, 2 - sleduj časovač, 3 - sleduj digitální vstup) Toto číslo musí být vyplněno vždy. Pouze kladná celá čísla. Je mu vyhrazen formát *unsigned integer*.

Čárka -, - odděluje jednotlivé části instrukce

Druhé číslo – první parametr instrukce. Ne všechny instrukce ho musí mít. Používá se například pro zjištění čísla pinu, který chceme digitálně číst, zadání místa v paměti a podobně. Pouze kladná celá čísla. Je mu vyhrazen formát *unsigned integer*.

Třetí číslo – druhý parametr instrukce. Ne všechny instrukce ho musí mít. V Arduinu je mu vyhrazen formát *long*. Používá se pro zápis větších hodnot než druhé číslo. Může obsahovat i znaménko minus, je tedy používán například pro zadání sledované teploty na teploměru, hodnoty pro čítač, nebo času pro časovač. Lze do něj zapisovat i mínusové hodnoty, avšak znaménko mínus musí být zapsáno až za číslem.
Čtvrté číslo – třetí parametr funkce, podobný jako druhé číslo. Použije se pouze výjimečně u funkcí, vyžadujících více než 2 parametry. Například u teploměru, nebo sčítání pamětí.
Středník -; - označuje konec instrukce, pokud na něj program narazí skáče na další řádek, cokoliv za tímto znakem tedy nebude přečteno. Toho lze s výhodou použít na umístění komentáře.

5.3 Seznam instrukcí

Timer set/Inicializace časovače- 1,X,Y;

Pokud je logický signál 1 a časovač ještě nebyl inicializován, nastaví časovač X na čas Y + hodnota uložená do této doby na paměťovém místě 0. Na tomto paměťovém místě bude vždy 0, nebylo-li již s tímto paměťovým místem manipulováno pomocí jiných instrukcí. Čas Y je zadáván v milisekundách a může nabývat pouze kladných hodnot. Pro časovač je vyhrazeno 20 míst v paměti. Je tedy možné mít nejvýše 20 různých časovačů v programu. Je důležité pamatovat na to, že paměť pro časovače je sdílená s pamětí pro čítače a teploměr. Čítač stejného čísla X jako je časovač tedy svojí inicializací přepíše hodnotu uloženou v časovači. Hodnota X může být v rozsahu od 0 do 19 a může nabývat pouze celých kladných hodnot. Hodnota Y může být v rozsahu od nuly do 2,147,483,647, takže nejdelší možný časovatelný úsek je něco přes 24 dnů. Pokud bylo dané místo v paměti již inicializováno, nastaví se časovač na hodnotu na tomto místě.

- Ve SpeedAPI má název Timer.
- Tato instrukce neovlivňuje hodnotu logického signálu.

Timer check/sleduj časovač 2,X;

Pokud je logický signál 1 a časovač X napočítal do hodnoty Y nastaví logický signál na 1, jinak na 0. V SpeedAPI je tato funkce sloučena do jedné spolu s předchozí s názvem Timer.

Digital input/digitální vstup 3,X;

Pokud je logický signál 1 a na vstupu X napětí, nastav logický signál na 1, jinak na 0. Možno použít všechny piny Arduina, kromě pinu 2 na kterém je digitální teploměr a pinů D10-D13 na nichž se nachází komunikace s SDkartou.

- Ve SpeedAPI má název Input.

Digital output/digitální výstup 4,X;

Pokud je logický signál 1, nastav výstup X na 1, pokud je logický signál 0 nastav výstup X na 0.

Pro výstupy lze použít stejné množství pinů jako pro vstupy se stejným omezením. Žádný pin nesmí být zároveň vstupní a výstupní.

- Ve SpeedAPI má název Output.
- Tato instrukce neovlivňuje hodnotu logického signálu.

Source/zdroj signálu - 5;

Tato instrukce nastaví hodnotu logického signálu na 1. Funguje tedy jako zdroj logického signálu. Touto instrukcí by měl začínat každý funkční program napsaný ve Speedu.

- Ve SpeedAPI má název Source.

Memory reset/Reset paměti pro čítač, časovač a teploměr - 6,X;

Pokud je logický signál 1, tato instrukce vypne časovač/čítač X a také vynuluje jeho paměť.

- Tato instrukce neovlivňuje stav logického signálu.
- Ve SpeedAPI má název Memory reset.

Counter/přičítání k čítači - 7,X;

Pokud je logický signál na náběžné hraně, zvedni hodnotu čítače X o jedna. Pro čítač je vyhrazeno 20 míst v paměti. Je tedy možné mít nejvýše 20 různých čítačů v programu. Je důležité pamatovat na to, že **paměť pro časovače je sdílená s pamětí pro čítače**. Čítač stejného čísla X jako je časovač tedy svojí inicializací přepíše hodnotu uloženou v časovači a naopak. Hodnota X může být v rozsahu od 0 do 19 a může nabývat pouze celých kladných hodnot. Hodnota Y může být v rozsahu od nuly do 2,147,483,647. Po průchodu nulou se inicializace daného paměťového místa vyruší, ale hodnota zde zůstane uložena.

- Tato instrukce neovlivňuje stav logického signálu.
- Ve SpeedAPI má název Counter.

Counter check/sleduj čítač - 8,X,Y;

Pokud je logický signál 1 a čítač X napočítal do hodnoty, která je stejná, nebo větší, než Y nastav logický signál na 1, pokud ne, nastav logický signál na 0. Ve SpeedAPI je tato instrukce sloučena s instrukcí Counter. Hodnota Y může nabývat pouze kladných hodnot.

SR block set/SR blok set - 9,X;

Pokud je logický signál 1 nastav hodnotu SR bloku X na 1. Pokud je logický signál 0, nedělej nic. Pro SR bloky je vyhrazeno v paměti 30 míst. Je tedy možné mít nejvíce 30 různých SR bloků. Paměť pro SR bloky není s ničím jiným sdílená. Hodnota X může být v rozsahu od 0 do 29 a může nabývat pouze celých kladných hodnot. SR bloky můžou nabývat hodnot buď 1 nebo 0, v případě že je SR blok na hodnotě 1 nastavuje hodnotu logického signálu také na 1, pokud je na hodnotě 0, nastavuje hodnotu logického signálu

také na 0. Je-li tedy touto instrukcí SR blok zapnut, nastavuje logický signál na 1, dokud není pomocí funkce SR block reset zase vypnut.

- Ve SpeedAPI má tato instrukce název S/R Set.

SR block reset/SR blok reset - 10,X;

Pokud je logický signál 1 nastav hodnotu SR bloku X na 0. Pokud je logický signál 0, nedělej nic. SR bloky můžou nabývat hodnot buď 1 nebo 0, v případě že je SR blok na hodnotě 1 nastavuje hodnotu logického signálu také na 1, pokud je na hodnotě 0, nastavuje hodnotu logického signálu také na 0.

- Ve SpeedAPI má tato instrukce název S/R Reset.

OR - 11;

Pokud je logický signál 1 přeskočí vykonání následující instrukce, pokud 0 nedělá nic. Je dobré dávat pozor, kde je tato instrukce v kódu umístěna.

- Ve SpeedAPI má tato instrukce název OR.

NOT - 12;

Invertuje hodnotu logického signálu.

- Ve SpeedAPI má tato instrukce název NOT.

Virtual input/Virtuální vstup - 13,X;

Pokud je logický signál 1 a na sériovou linku poslána zpráva ve formátu:
input;
Z;
end;
a *Z* ve zprávě se rovná X nastavenému v instrukci, nastaví logický signál na 1, pokud ne, nastaví logický signál na 0. **Je vhodné pamatovat, že od zadání slova *input;* do zadání slova *end;* vykonávání kódu Arduinem stojí.**

- Ve SpeedAPI má název Virtual IN.

Virtual output/Virtuální výstup - 14,X;

Pokud je logický signál 1, pošli na sériovou linku X.

- Tato instrukce neovlivňuje logický signál.
- Ve SpeedAPI má název Virtual OUT.

Temperature measure/Měření teploty - 15,X,Y,Z;

Pokud je logický signál 1, měří teplotu. Teplotu je možné tímto způsobem měřit pouze pomocí připojeného snímače od Dallas Instruments se signálním pinem připojeným na pin Arduina D2 (u Arduina UNO/Nano), kterému zde přísluší sběrnice OneWire. Je důležité pamatovat, že z důvodu použitých knihoven vyvolá zahájení měření časovou prodlevu (přibližně 100ms) po kterou vykonávání kódu stojí. Teplota je měřena pouze v

celých stupních Celsia. Hodnota Y je omezena teplotním rozsahem použitého teplotního snímače. Pro měření záporných hodnot je znaménko mínus zapsáno až za požadovanou hodnotu. Hodnota Z určuje na které místo v paměti se uloží hodnota Y. Hodnota Y se na dané místo uloží, jen pokud toto místo ještě nebylo inicializováno a jejím uložením se dané místo inicializuje. Bylo-li již na tomto místě něco uloženo, vezme se tato uložená hodnota jako hodnota Y. Pro teploměr je vyhrazeno 20 míst v paměti. Paměť pro teploměr je sdílená s pamětí pro časovače a čítače. Teploměr stejného čísla Z jako je časovač, nebo čítač tedy svojí inicializací přepíše hodnotu uloženou v časovači nebo čítači. Hodnota Z může být v rozsahu od 0 do 19 a může nabývat pouze celých kladných hodnot.

- Pokud X=0 a změřená teplota je nižší než Y, nastav logický signál na 1. Pokud ne, nastav logický signál na 0.
- Pokud X=1 a změřená teplota je vyšší než Y, nastav logický signál na 1. Pokud ne, nastav logický signál na 0.
- Pokud X=2 zapiš změřenou teplotu na sériovou linku.
- Ve SpeedAPI má název Thermometer

Příklad: 15,1,12-,1; Význam: Pokud je logický signál 1, měř teplotu a pokud je vyšší než -12°C nastav logický signál na 1, jinak na nula. Mezní hodnotu teploty (-12°C) ulož do paměti na pozici 1.

Analog input/Analogový vstup - 16,X;

Pokud je logický signál 1, čte hodnotu na analogovém vstupu X. Analogové vstupy u Arduina Nano jsou A0-A7, je však nutné je indexovat čísly 14-21. Jedná se o poměrnou hodnotu napětí na pinu, vyjádřenou kladnými celými čísly od 0 do 1033.

- Ve SpeedAPI má název Analog in

Analog save/Ulož analogový signal - 17,X;

Pokud je logický signál 1, uloží poslednou změřenou analogovou hodnotu do paměti pro čítače, časovače a teploměr na pozici X. Pro tuto paměť je vyhrazeno 20 míst. Místo v paměti stejného čísla X jako je časovač, nebo čítač tedy svojí inicializací přepíše hodnotu uloženou v časovači nebo čítači. Hodnota X může být v rozsahu od 0 do 19 a může nabývat pouze celých kladných hodnot. Místo v paměti se touto instrukcí inicializuje.

- Ve SpeedAPI je tato funkce sloučena s Analog Input.
- Tato instrukce neovlivňuje logický signál.

Memory show/Zobraz místo v paměti - 18,X;

Pokud je logický signál 1 vypíše obsah paměti na místě X pro čítače, časovače a teploměr na sériovou linku. Pokud je paměť prázdná, zobrazí se nula.

- Ve SpeedAPI má název Memory show
- Tato instrukce neovlivňuje logický signál.

Memory math/Matematické operace s pamětí - 19,X,Y,Z;

Pokud je logický signál 1, podle čísla Y provede s pamětovými místy X a Z paměti pro čítače, časovače a teploměr požadovanou operaci. Pro pamět je vyhrazeno 20 míst. Čísla X a Z mohou nabývat kladných celých hodnot od 0 do 19.

- Pokud Y=0 vynásobí hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=1 vydělí hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=2 přičte hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=3 odečte hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=4 umocní hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=5 odmocní hodnotu na paměťovém místě X hodnotou z paměťového místa Z.
- Pokud Y=6 nahradí hodnotu na paměťovém místě X hodnotou z paměťového místa Z. A inicializuje toto paměťové místo.
- Pokud Y=7 a hodnota na pamětovém místě X je větší než hodnota na paměťovém místě Z, nastaví logický signál na 1, jinak na 0.
- Pokud Y=8 a hodnota na pamětovém místě X je menší než hodnota na paměťovém místě Z, nastaví logický signál na 1, jinak na 0.
- Pokud Y=9 a hodnota na pamětovém místě X je rovna hodnotě na paměťovém místě Z, nastaví logický signál na 1, jinak na 0.
- Ve SpeedAPI má název Memory math
- Tato instrukce nemění inicialiazci žádného paměťového místa, pouze ovlivňuje hodnotu na některém z nich.
- Ve všech případech platí, že hodnota Z se nijak nezmění.

Příklad: 19,1,5,2; Vysvětlení: Pokud je logický signál 1, odmocni hodnotu na paměťovém místě X hodnotou na paměťovém místě Z.

Memory add/přičtení čísla do paměti - 20,X,Y;

Pokud je logický na nástupné hraně, přičte na místo X paměti pro čítače, časovače a teploměr hodnotu Y. Hodnota Y může nabývat celých hodnot a má-li být mínusová, musí být znaménko - zapsáno až za číslo. Pro paměť je vyhrazeno 20 míst. Čísla X a Z mohou nabývat kladných celých hodnot od 0 do 19. Pří průchodu nástupnou hranou se dané paměťové místo inicializuje, pokud tato instrukce zaznamená průchod nulou, inicializace místa se opět vynuluje, ale hodnota zde zůstane již uložena.

- Ve SpeedAPI má název Memory add
- Tato instrukce neovlivňuje logický signál.

Příklad: 20,1,1-; Vysvětlení: Pokud je logický signál na nástupné hraně, odečti od čísla uloženého v paměťovém bloku 1 hodnotu -1.

5.4 API

Neboli rozhraní pro programování aplikací je napsáno v jazyce C# [28], jedná se o jazyk vycházející z jazyka C, je objektově orientovaný a velmi podobný jazyku C++, což je i jeden z důvodů, proč byl pro tvorbu API zvolen. [27] Arduino je programováno v C++ a je tedy pravděpodobné, že uživatel znalý C++ se rychle zorientuje i v jazyce C#. Jazyk ke svému chodu vyžaduje platformu NET framework, běžně se vyskytující na Windows, z čehož plyne i jeho nevýhoda. Programy pod jiným operačním systémem, než je Windows nepoběží. Pro drtivou převahu operačních systémů Windows na osobních domácích počítačích však bylo rozhodnuto pro tvorbu API v jazyce C# i přes tuto nevýhodu. V potaz byl brán ještě Action Script vycházející z Javy a programovací prostředí Game Maker, případně produkty třetích stran dovolující vytvářet vlastní funkce pro Arduino ve svém vlastní grafickém rozhraní. Žádná z těchto možností se však v praxi neosvědčila.

Programovací prostředí, konkrétně Microsoft Visual Studio 10, ve kterém je API vytvořeno disponuje rozhraním pro snadnou tvorbu okenních aplikací ve Windows, takzvaných „Windows form“. Jednotlivé prvky, které se mají ve výsledném programu objevit se přidávají na pracovní plochu metodou “Drag and Drop”. Programovací prostředí obsahuje všechny ve Windows běžně dostupné grafické prvky a jejich funkce. Verze, v níž API vznikalo je již několik let stará, tudíž je zaručena kompatibilita programu i na starších operačních systémech, jako je Windows XP.

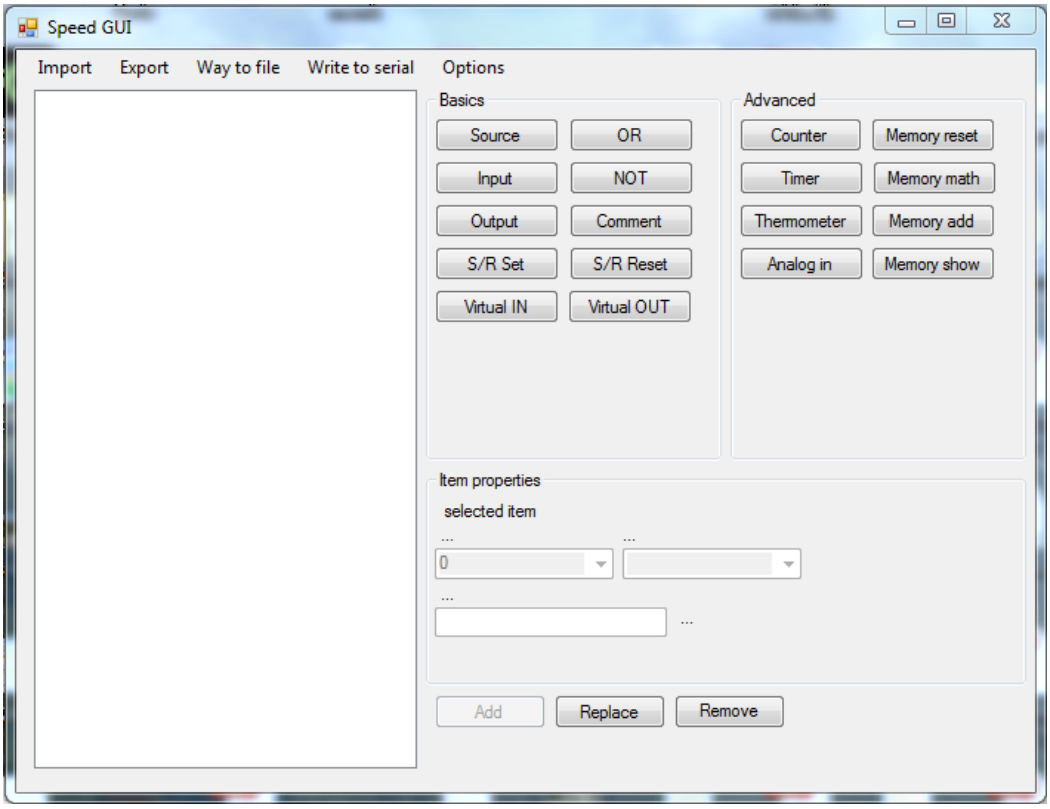
API nese název SpeedAPI, jak již bylo řečeno, podle souhrnného názvu jazyka instrukcí, jimiž se ovládá PLCduino. Jak již bylo zmíněno v minulých kapitolách, není nezbytně nutné používat k psaní instrukcí toto jediné API spíše se počítá s případnou tvorbou dalších, vhodných pro konkrétní účely. Instrukce jsou záměrně v takovém tvaru, aby jejich vytvoření a uložení právě pomocí API bylo co nejjednodušší. Úkolem SpeedAPI je zjednodušit tvorbu těchto instrukcí a orientaci v nich, protože jejich manuální zapisování do textového souboru je velmi nepřehledné. S použitím popisovaného API by měl zvládnout tvorbu základních aplikací i uživatel, který před použitím tohoto API prošel buď rychlým, nebo žádným školením.

5.4.1 Ovládání API

Po dvojitém poklepání na ikonu SpeedAPI (Příloha C) se zobrazí přímo hlavní okno programu (Obr. 5.4) a je možné začít rovnou vytvářet vlastní Speed program. Nahoře se nachází systémová lišta s možností importovat program Speed z textového souboru, exportovat právě vytvářený program do textového souboru, změnit cestu k tomuto souboru, nebo zapsat program do PLCduina přímo přes sériovou linku. Poslední možností je záložka *Možnosti*, kde lze nastavit další podrobnosti, jako například volba sériového portu. V současném stavu vývoje fungují pouze záložky *Export* a *Cesta k souboru*. Pokud

má PLCduino textový soubor správně rozeznat je nutné, aby nesl název *ins.txt*. API ale umožňuje pojmenovat soubor i jinak, aby bylo možné vytvářet například různé verze programu ve stejné složce. Soubor je možné uložit přímo na datovou kartu, je-li k počítači připojena. Exportovaný soubor *ins.txt* uložený na SD kartě a obsahující instrukce pro běh programu se pak vloží přímo do čtečky SD karet na PLCduinu a po nastavení konfiguračního souboru je již možné PLCduino využívat. V současné době není konfigurační soubor s nastavením vstupů a výstupů *setup.txt* vytvářen automaticky při exportu programu do *ins.txt* a je tedy nutné zapsat do něj požadované vstupy a výstupy ručně, tak aby korespondovali s programem, který má aktuálně na PLCduinu běžet.

Dále se v programu nachází sekce *Basics* s jednoduchými funkcemi, sekce *Advanced* s funkcemi složitějšími jak na správné použití, tak na program a pod nimi záložka *Item properties*. Okno, v němž je vypisován současný stav programu se nachází úplně vlevo. Některé instrukce se do tohoto okna přidají okamžitě po stisknutí daného tlačítka. Konkrétně jsou to *Source, OR* a *NOT*. Jedná se o instrukce, u nichž není možné, a ani potřebné, provádět žádné další nastavení. Při kliknutí na jakékoliv jiné tlačítko s instrukcí, se zobrazí nabídka v *Item properties*. Instrukce se po požadovaném nastavení parametrů v této sekci přidá do programu tlačítkem *Add,* umístěným níže. Kromě přidávání instrukcí do programu je možné je nahrazovat jinými pomocí tlačítka *Replace,* nebo mazat tlačítkem Remove. Před odstraněním nebo nahrazením instrukce v programu je nutné ji nejprve kliknutím myši, nebo pomocí kurzorových šipek, označit. Při přidávání instrukcí nových se instrukce přidávají vždy **nad** aktuální označenou instrukci a pokud není žádná instrukce označena, tak na konec programu.



Obr. 5.4 hlavní okno SpeedAPI

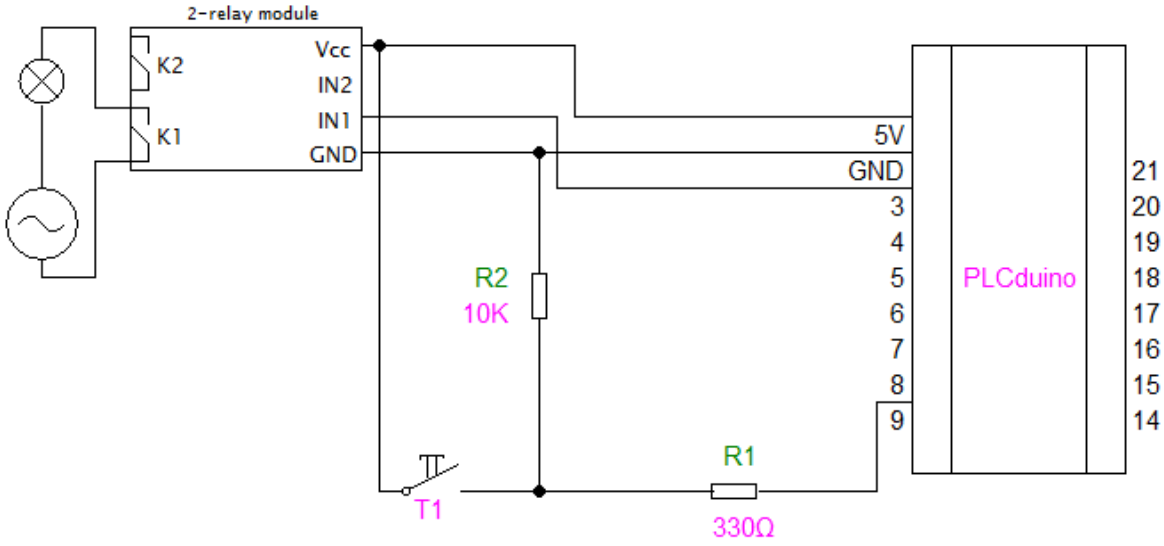
6 PŘÍKLADY PRAKTICKÝCH APLIKACÍ

V následující kapitole je uvedeno několik praktických aplikací vyvinutého zařízení. Je u nich vždy ukázán vzhled v API, popsána jejich logika a vysvětlena funkce. U několika z nich je také zobrazeno zapojení. Rozhodně se nejedná o všechny existující funkce, těch je mnohem více, tato kapitola má tak spíše nastínit, k jakým účelům lze zařízení použít a jak se zařízení konfiguruje a programuje.

6.1 Ovládání schodišťového osvětlení

Zadání: Navrhnout schodišťové osvětlení tak, aby po stisknutí tlačítka na schodišti svítilo jím ovládané schodišťové osvětlení, realizované jednou žárovkou, ještě 10 vteřin a poté se vypnulo.

Řešení: Nejprve je PLCduino zapojeno dle schématu níže (Obr. 6.1), modul s dvěma relé je zde připojen k pinu 3, využije se tedy jen jedno ze dvou elektromagnetických relé na modulu. Tlačítko, v tomto případě spínač schodišťového osvětlení je připojeno na pin 9 jako "PULL-DOWN". Napájení PLCduina není v tomto schématu zakresleno.



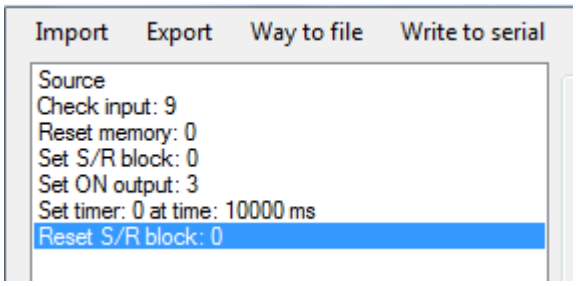
Obr. 6.1 zapojení PLCduina s relé a jedním tlačítkem

Poté je do příslušného konektoru počítače vložena SD karta, která má být použita jako úložiště instrukcí pro PLCduino, vytvořen na ní textový soubor setup.txt a vložen do něj následující řádek*: 9,I,3,O;* Neboli, nastavit pin 9 jako vstup a pin 3 jako výstup.

Následně je otevřeno SpeedAPI a začíná se s tvorbou programu (Obr. 6.2). Jak už bylo řečeno dříve, program je čten cyklicky od shora dolů. Nejprve je třeba vložit prvek *Source*, který vždy při průchodu tímto prvkem nastaví logický signál na 1, hned za ním následuje instrukce *Check input*, která zkontroluje, je-li na pinu 9 napětí 5V, tzn. že je spínač sepnut. Pokud se tak stalo nechá logický signál na hodnotě 1, jinak ho nastaví na

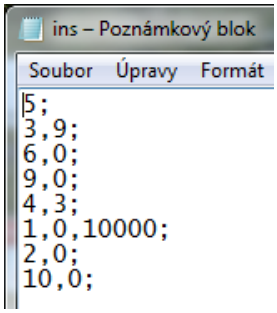
nulu. Pokud je logický signál na nule, žádná z následujících instrukcí se neprovede. Při logickém signálu na hodnotě 1 se restartuje paměťový slot 0 instrukcí *Reset memory*. Na tomto slotu je umístěn dále v programu časovač, bude tedy stiskem tlačítka restartován, bez tohoto kroku by, po prvním vypnutí světla, světlo svítilo už pouze při stisknutém spínači. Na tomto příkladu jde vidět, že paměťové sloty pro S/R bloky nejsou sdílené s ostatními paměťovými bloky, je tedy možné využít pro ně také paměťový slot s indexem 0. Jako čtvrtá instrukce je zde nastavení S/R bloku na set, aby světlo svítilo i po puštění tlačítka. Následně se sepne výstup 3, což je v tomto případě relé ovládající žárovku a v dalším kroku se nastaví časovač na 10s, po uplynutí tohoto času je logický signál o hodnotě 1 "propuštěn" dál a resetuje S/R blok 0. Světlo zhasne.

Výhodou zobrazeného řešení je fakt, že vždy při stisknutí tlačítka se odpočet restartuje a je-li tedy tlačítko stisknuto například jednu vteřinu před zhasnutím světla, světlo bude svítit i následujících 10 vteřin.



Obr. 6.2 hotový program ve SpeedAPI

Kliknutím na *Way to file* je otevřeno dialogové okno pro výběr umístění souboru. Cesta k souboru zvolena jako cesta na vloženou SD kartu, soubor je pojmenován jako *ins*, stisknuto *OK* a po zavření dialogového okna stisknuto tlačítko *Export*. Po úspěšném exportu by se měl kurzor zastavit na posledním řádku programu, jak je patrné z obrázku. (Obr. 6.2) Pokud je soubor *ins.txt* otevřen, jsou vidět instrukce přímo ve Speedu. (Obr. 6.3) Každý řádek zde odpovídá jednomu řádku ve SpeedAPI, pouze instrukce *Set timer* zabírá řádky dva (konkrétně řádek 6 a 7). Ve skutečnosti se totiž jedná o dvě samostatné instrukce (Inicializace časovače a hlídání, zda časovač přetekl), které jsou však ve SpeedAPI z důvodu snazšího ovládání sloučeny do jedné.



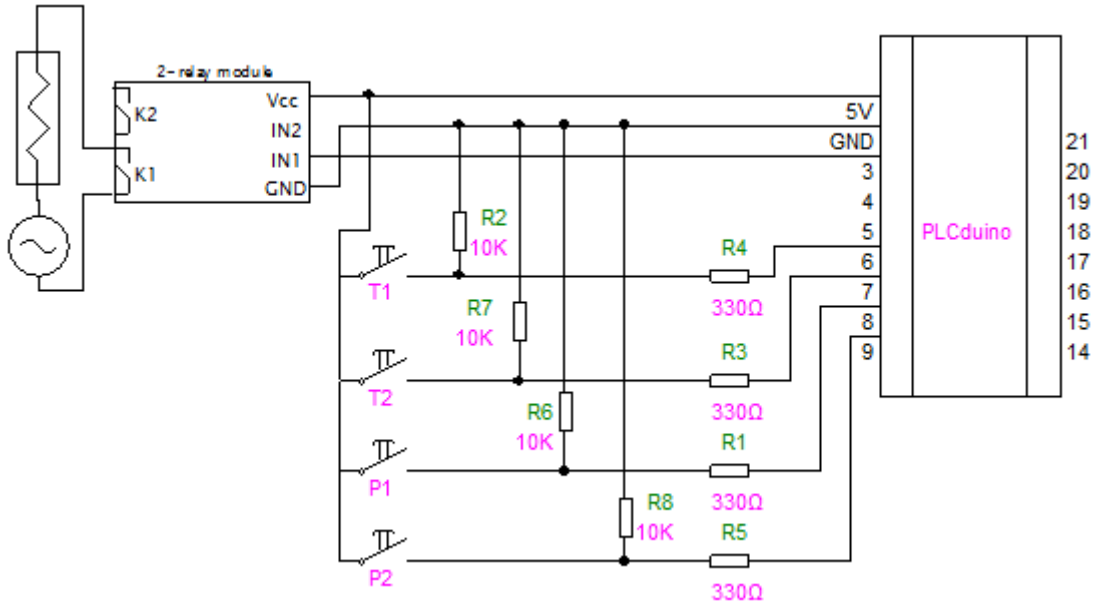
Obr. 6.3 program zobrazen ve Speedu

Vložením SD karty zpět do PLCduina a připojením napájení (případně restartem), lze ověřit, že daný kód funguje.

6.2 Termostat

Zadání: Navrhnout termostat, který bude sledovat teplotu v místnosti a pokud teplota poklesne pod požadovanou hodnotu spustí topení. Panel termostatu bude vybaven tlačítky pro změnu požadované teploty a tlačítky pro manuální ovládání topení.

Řešení: Nejprve je opět zapojeno PLCduino podle schématu. (Obr. 6.4) Zapojení zůstává velmi podobné, jen žárovka je zde vyměněna za elektrické topné těleso, (při použití modulu relé popisovaného v této práci se jedná o topné těleso s příkonem do 10A) také přibyly tři tlačítka. Je zvoleno, že tlačítkem na pinu 6 bude zvolená teplota zvyšována, tlačítkem na pinu 7 snižována, pokud bude sepnut přepínač na pinu 8 bude termostat ve stavu manuálního ovládání a pokud v tomto stavu bude sepnut i přepínač na pinu 9, bude topení topit. (je tedy nutné, aby P1 a P2 byly přepínače a vhodné aby T1 a T2 byly tlačítka)

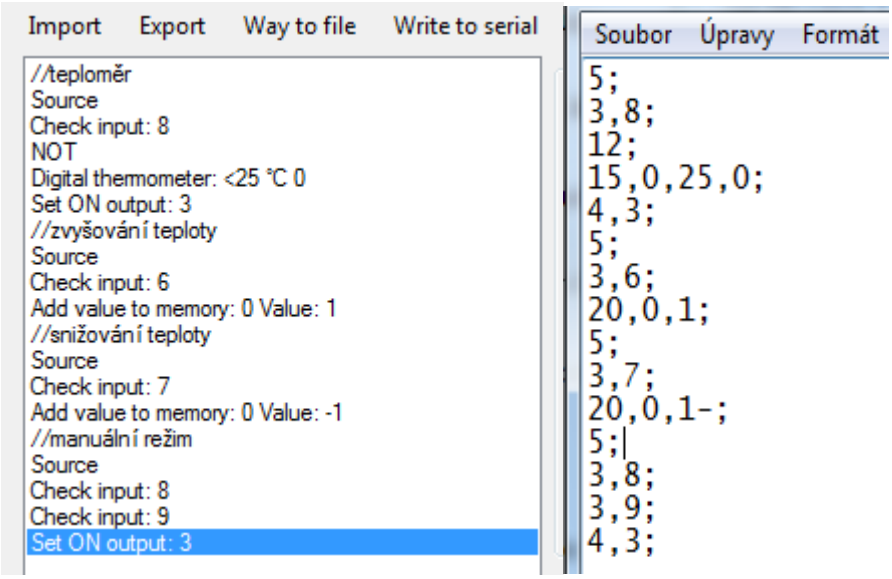


Obr. 6.4 schéma zapojení PLCduina jako termostat

Po zapojení je vytvořen kód ve SpeedAPI. (Obr. 6.5) První viditelná instrukce je poznámka, jelikož bude tento program již o něco delší než v předchozím příkladě, jsou zde poznámky použity pro lepší orientaci, nemají však na výsledný program žádný vliv. Nejprve tedy program zkontroluje, jestli není termostat v manuálním režimu, pokud není, měří teploměr teplotu a její požadovanou hodnotu má uloženou v paměti na místě 0. Pokud je teplota nižší než 25°C, spustí se topení. V tomto příkladu se počítá s umístěním termostatu v běžném obývacím pokoji, systém má tedy dostatečnou setrvačnost a není třeba nastavovat rozmezí hodnot ve kterém má termostat spínat a vypínat. Následuje zvyšování požadované teploty, při držení tlačítka na pinu 6 se k hodnotě uložené na paměťovém slotu 0 bude přičítat jednička. Na paměťovém slotu 0 je uložená právě uživatelem zvolená požadovaná teplota a bude se tedy touto instrukcí zvyšovat. Naopak po stisknutí tlačítka 7 se bude jednička odečítat. Nakonec je zde manuální režim, v tomto zápisu fungují spínače na pinech 8 a 9, jako by byly zapojeny sériově, realizují tedy funkci

AND, pokud je zvolen manuální režim a je stisknutý přepínač na pinu 9, topení bude topit, pokud je přepínač na pinu 9 vypnutý, nebude a pokud je přepínač na pinu 8 vypnutý, bude termostat fungovat v autonomním režimu.

Nevýhodou v tomto případě je, že zvolená teplota není nikde vidět což může způsobit neúmyslné zvýšení, nebo snížení teploty o více stupňů, než uživatel požadoval. Oba tyto problémy však lze vyřešit. Místo tlačítek je možné zařadit analogový vstup a na něm měnit teplotu otáčením potenciometru, což celý program ještě zjednoduší a hodnotu teploty uložené na paměťovém slotu 0 lze vypisovat na sériovou linku pomocí příkazu *Memory show*. Také je možné potenciometr označit tak, aby se teplota nastavená na termostatu dala z něj přímo odečíst, nutností je pak přepočet hodnoty odečítané na potenciometru na hodnotu lépe odpovídající měřenému rozsahu teplot.

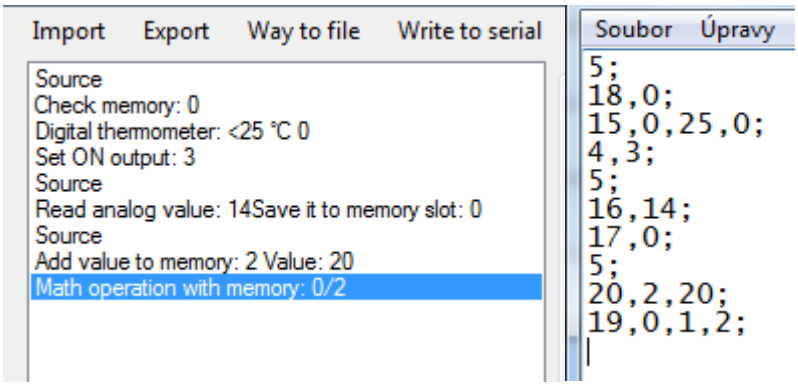


Obr. 6.5 instrukce pro termostat ve SpeedAPI (vlevo) a přímo ve Speedu (vpravo)

Pokud budou navrhované úpravy provedeny, bude nutné změnit zapojení a program bude vypadat následovně. (Obr. 6.6) Manuální režim z něj byl pro větší přehlednost odstraněn, je v něm tedy vidět pouze část s nastavením požadované teploty pomocí potenciometru a její vypsání na sériovou linku. V druhé instrukci je vypsána na sériovou linku paměť na slotu 0, následně se nastaví požadovaná teplota opět na 25°C a v každém průchodu programem se kontroluje, jestli není měřená teplota nižší než tato hodnota. Uložením hodnoty 25°C do slotu 0 se tento slot inicializuje. Na šestém řádku probíhá čtení analogové hodnoty z pinu 14 a její uložení do slotu 0, ten se tímto krokem přepíše z hodnoty 25 na hodnotu nastavenou na potenciometru a proběhla by i jeho inicializace, pokud by se tak již nestalo dříve. Hodnota odečtená z potenciometru nemusí svou velikostí ani rozsahem odpovídat teplotě měřené teploměrem, je tedy vhodné ji dodatečně upravit.

To se děje na předposledním řádku, kde se do paměťového slotu 2 uloží hodnota 20, tím se tento slot inicializuje, a protože je přímo před touto instrukcí zařazena instrukce *Source*, instrukce *Add value to memory* už nikdy neprojde nulou, inicializace se tak

nezruší a číslo uložené na slotu 2 se v dalších průchodech již nebude zvyšovat. Jednoduše řečeno, v tomto případě se instrukce *Add value to memory* provede pouze jednou. Poslední instrukce se naopak bude provádět při každém průchodu programem a hodnota na paměťovém slotu 0 v ní bude vydělena hodnotou na paměťovém slotu 2. V případě že by tedy na potenciometru byla odečítána hodnota tisíc, bude termostat udržovat teplotu na 50°C. V dalším průchodu programem se pak pomocí instrukce *Memory show* ukáže teplota, již se termostat snaží udržovat a nevýhody uvedené zde dříve tak byly odstraněny. Jedinou zůstávající nevýhodou je, že není nijak zobrazená aktuální měřená teplota, což lze ale také řešit, a to vypisováním této teploty na sériovou linku pomocí příkazu *Digital thermometer* a jeho nastavením na *Send temperature to serial.*

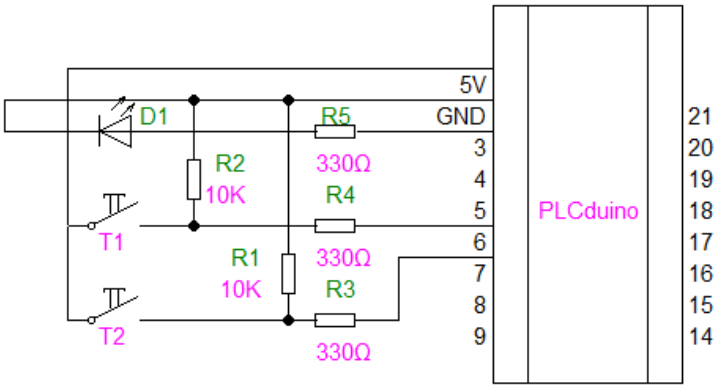


Obr.6.6 instrukce pro termostat ovládaný potenciometrem.

6.3 Rozsvěcení LED

Zadání: Navrhnout kontrolní LED diodu, která bude svítit, pokud bude alespoň jeden ze dvou spínačů sepnutý.

Řešení: PLCduino je zapojeno dle schématu. (Obr. 6.7) Tlačítka zapojena jako vstupy typu "PULL-DOWN" například na piny 6 a 7, diodu jako výstup na pin 3.



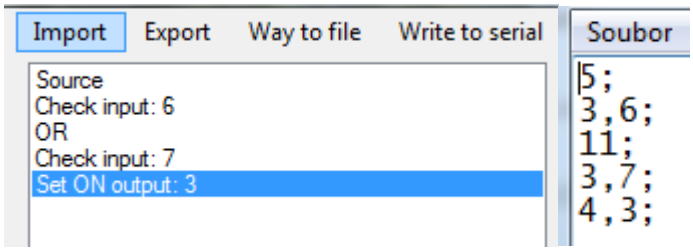
Obr. 6.7 schéma PLCduina s jednou diodou a dvěma tlačítky

Konfigurační soubor bude nyní vypadat následovně:

6,I,7,I,3,O;

Program bude mít nyní pouze 5 řádků. (Obr. 6.8) Použitím instrukce source se logický signál nastaví na 1, pokud bude tlačítko na vstupu 6 sepnuté, instrukce *OR* způsobí přeskočení následující instrukce a rovnou se provede instrukce pátá, v tomto případě poslední, která způsobí rozsvícení svítivé diody na pinu 3. Pokud by tlačítko na pinu 6 zůstalo nesepnuté, nastavil by se logický signál opět na nulu a putoval by k instrukci *OR*, která by ho nastavila opět na 1, záleželo by tedy na stavu spínače na pinu 7. Pokud by zůstal nesepnutý, logický signál by se opět přepnul na nulu a dioda by nesvítila, pokud by ale byl sepnutý, logický signál by zůstal na hodnotě jedna a dioda by se opět rozsvítila. Tento příklad je zde uveden především z důvodu lepší ilustrace funkce *OR*, která funguje tak, že pokud je na řadě a logický signál je na hodnotě 1, následující instrukci přeskočí, jinak nastaví logický signál z nuly na jedna. Z toho důvodu nemůže být za funkcí *OR* zařazen jako následující instrukce například časovač, pokud tedy není žádoucí, aby byl v případě že na *OR* dojde logický signál 1 přeskočen. Funkci *OR* lze tedy cíleně využít i jako funkci pro vynechání části programu.

Vpravo na (Obr. 6.8) je opět vidět vzhled programu přímo ve Speedu, každý řádek zde odpovídá jedné instrukci ve SpeedAPI.



Obr. 6.8 implementace instrukce OR

6.4 Automatizace pivovaru

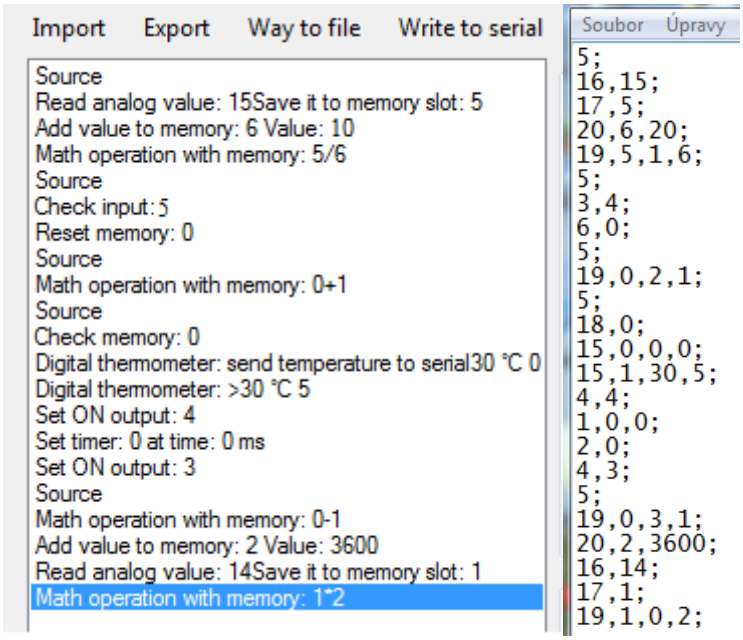
V průběhu vaření piva je nutné opakovaně ohřívat budoucí pivo na určité teploty a udržovat je na těchto teplotách po určitý časový úsek. Mírnou úpravou a rozšířením následující ukázky je možné automatizovat většinu pivovarnických procesů, tato ukázka se však věnuje pouze jedné části problému, celé problematice se pak věnuje jiná práce. [29]

Zadání: Navrhnout řešení umožňující nastavit teplotu, na níž má být ohřáta tekutina s možností udržování tekutiny na této teplotě po stanovenou dobu.

Řešení: Schéma zapojení není v tomto případě uvedeno z toho důvodu, že zapojení ohřevu je stejné jako na schématu (Obr. 6.4) zapojení tlačítek pak bylo v této práci rozebíráno již vícekrát a stejně tak zapojení analogového vstupu je podrobně popsáno na (Obr. 4.8) Nejprve se tedy zapojí příslušné periferie. Elektromagnetické relé například na pin 4, analogové vstupy pak na piny 14 a 15. Kontrolní dioda bude umístěna na pinu 3. Potenciometrem na pinu 15 bude regulována teplota, potenciometrem na pinu 14 pak

doba, po kterou má být po dosažení této teploty tato teplota udržována. Pokud se dosáhne požadované doby, rozsvítí se dioda na pinu 3, které upozorní obsluhu, že je třeba postoupit k následujícímu kroku. Rozdíl je zde v elektromagnetickém relé, které je tentokrát zapojeno inverzně, při sepnutí se tedy ohřev vypne. Tato změna je zde zařazena opět především kvůli přehlednosti kódu, v praxi je ji možné nevyužít.

První část programu (Obr. 6.9) je stejná, jako v alternativním řešení termostatu v sekci *6.2. Termostat.* Program zde čte hodnotu napětí na potenciometru a vhodnými úpravami ji mění na požadovanou teplotu. Jedinou změnou je zde hodnota 10, uložená na paměťovém místě 6, aby teplotní rozsah nastavitelný potenciometrem umožňoval navolit i hodnoty kolem 100°C. Změna začíná na řádcích 5-7, kde je část kódu pro restart paměťového slotu 0. Po stisknutí tlačítka umístěném na pinu 5 dojde k restartu paměťového místa sloužícího v tomto příkladu jako paměť pro časovač. To umožňuje navolit jiný čas i po tom, co již byla kontrolní dioda na pinu 3 rozsvícena.



Obr. 6.9 program pro řízení ohřevu a udržování teploty

Nejdůležitější část kódu však začíná až na řádku 9 a je také nejsložitější na vysvětlení. K hodnotě časovače je zde přičtena hodnota uložená na paměťovém místě 1. Na toto místo se ukládá hodnota napětí na potenciometru na pinu 15, opět vhodně upravená, tentokrát vynásobená 3600, dále v kódu. Číslo 3600 je zde proto, aby se při nastavování potenciometru dalo dosáhnout časů odpovídajících přibližně jedné hodině. (Což je při vaření piva nejdelší doba, po kterou je nutno udržovat tekutinu na požadované teplotě.) Na řádcích jedenáct a dvanáct je pak pouhé vypsání aktuální měřené teploty a hodnoty slotu 0 na sériovou linku. Následuje instrukce, kontrolující, jestli je měřená teplota nad nastavenou hodnotou. Pokud ano, sepne se relé a vypne se ohřev a v následující instrukci se nastaví časovač na čas 0 milisekund + čas uložený na slotu 0. Po uplynutí této doby se rozsvítí kontrolní dioda na pinu 3, upozorňující obsluhu, že již uplynul nastavený čas. V poslední části je pak kód pro uložení požadovaného času na paměťový slot 0. Nejdříve je

od paměťového slotu 0 odečtena hodnota slotu 1, přičtená na začátku programu. To proto, aby se hodnota na slotu 0 nezvyšovala v každém průběhu programu o hodnotu z paměťového slotu 1 a zůstávala na nastavené hodnotě. Následně je zde zařazen již známý kód pro úpravu hodnoty napětí čtené na pinu 14.

Shrnutí: Nastavení požadovaného času tedy probíhá tak, že nejprve je přičtena k časovači hodnota uložená na slotu 1, například 5000ms, ta se na konci programu opět odečte, aby se v každém průchodu nezvyšovala hodnota časovače o dalších 5000ms. Jakmile je dosaženo požadované teploty, inicializuje se časovač na hodnotu aktuální čas běhu programu od restartu + 5000ms a po pěti vteřinách od dosažení požadované teploty se rozsvítí kontrolní dioda.

7 ZÁVĚR

Z analýzy uvedené rešeršní části vzniklo zařízení, určené především pro amatérský zájem o automatizaci budov, zaměřené na uživatelskou přívětivost a jednoduchou aplikaci. Práce ukazuje, proč bylo zvoleno právě toto řešení, popisuje jeho návrh, hardwarové a softwarové provedení. V práci je zahrnut popis ovládání a komunikace se zařízením a ukázky praktických aplikací.

Základem zařízení je mikroprocesorová jednotka Arduino Nano spojená s čtečkou SD karet, na níž jsou uloženy data. V Arduinu je nahrán program, který cyklicky čte data uložená na SD kartě a řídí jimi svůj chod, tyto data jsou v této práci označeny jako instrukce. K zařízení je možné v současném stavu připojit digitální a analogové vstupy, teploměr, digitální výstupy a elektromagnetická relé pro ovládání zařízení v domovní rozvodné síti. Zařízení je schopno ovládat činnosti domovní automatizace jako zapínání a vypínání světel, vytápění, či otvírání a zavírání oken. Součástí zvoleného řešení je popis sady instrukcí, jimiž se zařízení ovládá a grafické aplikační rozhraní pro vytváření těchto instrukcí. Záleží na uživateli, jaké instrukce budou v zařízení uloženy. Podle toho se může úplně změnit i použití zařízení, jak nastiňují praktické příklady v této práci. Zařízení je stavěné s ohledem na jeho možnou rozšiřitelnost. Tou se myslí například přidání funkcí pro komunikaci s digitálním displejem, nebo přidání dalších funkcí navržených uživatelem. V současném stavu je možné zařízení propojit s osobním počítačem. Jednou z praktických aplikací tohoto zařízení je jeho plánované použití v Escape room v Adamově, dalšími jsou pak časovač schodišťového osvětlení, termostat, nebo řízení ohřevu tekutin.

8 SEZNAM POUŽITÉ LITERATURY

[1] MERZ, Hermann, Thomas HANSEMANN a Christof HÜBNER. *Automatizované systémy budov: sdělovací systémy KNX/EIB, LON a BACnet*. Praha: Grada, 2008. Stavitel. ISBN 978-80-247-2367-9.

[2] VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[3] *Automatizace budov* [online]. [cit. 2017-05-22]. Dostupné z: http://www.automatizacebudov.cz/#o-nas

[4] AN34: Příklady použití Poseidon Industrial Bus. *HW group* [online]. [cit. 2017-05-22]. Dostupné z: http://www.hw-group.com/support/an34/index_cz.html

[5] *NATIONAL KNX* [online]. [cit. 2017-05-22]. Dostupné z: https://www.knx.org/cz/

[6] Úvod do KNX. *Automatizace.hw.cz* [online]. [cit. 2017-05-22]. Dostupné z: http://automatizace.hw.cz/teorie-a-praxe/knx.html

[7] ABB i-bus® KNX. *ABB* [online]. [cit. 2017-05-22]. Dostupné z: http://www117.abb.com/index.asp?thema=5886

[8] BigClown se představuje: IoT otevřeně a hlavně bezpečně. *ROOT.CZ* [online]. [cit. 2017-05-22]. Dostupné z: https://www.root.cz/clanky/bigclown-se-predstavuje-iot-otevrene-a-hlavne-bezpecne/

[9] Big Clown: obchod. *Big Clown* [online]. [cit. 2017-05-22]. Dostupné z: https://obchod.bigclown.cz/

[10] *Google home* [online]. [cit. 2017-05-22]. Dostupné z: https://madeby.google.com/home/

[11] Elgato Eve Energy EU – bezdrátový senzor a zásuvka. *Czc.cz* [online]. [cit. 2017-05-22]. Dostupné z: https://www.czc.cz/elgato-eve-energy-eu-bezdratovy-senzor-a-zasuvka/199610/produkt?gcli

[12] Xiaomi představilo pod obchodní značkou MIJIA inteligentní rychlovarnou konvici. *Gizchina* [online]. 2016 [cit. 2017-05-22]. Dostupné z: https://gizchina.cz/2016/06/14/xiaomi-predstavilo-pod-obchodni-znackou-mijia-inteligentni-rychlovarnou-konvici/

[13] BALÁTĚ, Jaroslav. *Technické prostředky automatického řízení*. Praha: SNTL – Nakladatelství technické literatury, 1986.

[14] PROČ JE U TLAČÍTKA REZISTOR? *ARDUINO.CZ* [online]. [cit. 2017-05-22]. Dostupné z: https://arduino.cz/proc-je-u-tlacitka-rezistor/

[15] Pull-Up ve Pull-Down Nedir? *TOPRAK HATTI* [online]. [cit. 2017-05-22]. Dostupné z: http://www.toprakhatti.com/pull-up-ve-pull-down-nedir/

[16] RAMBOUSEK, J., S. NEČÁSEK a J. JANEČEK. *ELEKTRONICKÉ A ELEKTROAKUSTICKÉ SOUČÁSTKY: Jejich volba a použití*. Praha, 1980. ISBN 04-523-80.

[17] Arduino compatible Nano board, Free USB cable. *EEZONE* [online]. [cit. 2017-05-22]. Dostupné z: http://eezone.co.uk/blog/arduino/arduino-compatible-nano-board-free-usb-cable-2.html

[18] Nano V3.0 FT232. *AliExpress* [online]. ATMEL [ATMEL Corporation] [cit. 2017-05-22]. Dostupné z: https://www.aliexpress.com/item/Nano-V3-0-FT232-chip-with-usb-cable-for-Arduino-free-shipping/32309876432.html?spm=2114.40010408.3.1.UY7V9D&s=p

[19] ATMEGA328 Datasheet (PDF) - ATMEL Corporation. *ALLDATASHEET.COM* [online]. ATMEL [ATMEL Corporation] [cit. 2017-05-22]. Dostupné z: http://www.alldatasheet.com/datasheet-pdf/pdf/392243/ATMEL/ATMEGA328.html

[20] Inteligentní internetový termostat. *ROOT.CZ* [online]. 2015 [cit. 2017-05-23]. Dostupné z: https://www.root.cz/serialy/inteligentni-internetovy-termostat/

[21] New Micro SD Storage Board Mciro SD TF Card Memory Shield Module SPI For Arduino. *PIC* [online]. [cit. 2017-05-23]. Dostupné z: https://picclick.com/New-Micro-SD-Storage-Board-Mciro-SD-TF-311713029903.html

[22] 1M Waterproof Digital Thermal Probe Temperature Sensor DS18B20 Arduino Sensor. *Ebay.com* [online]. [cit. 2017-05-23]. Dostupné z: http://www.ebay.com.au/itm/1M-Waterproof-Digital-Thermal-Probe-Temperature-Sensor-DS18B20-Arduino-Sensor-/190738017226

[23] Sběrnice 1-Wire. *Vyvoj.hw.cz* [online]. [cit. 2017-05-23]. Dostupné z: http://vyvoj.hw.cz/navrh-obvodu/rozhrani/sbernice-1-wiretm.html

[24] Relé 5V / 220V - samostatné. *Santy.cz* [online]. [cit. 2017-05-23]. Dostupné z: http://www.santy.cz/rele-c26/relay-5v-alone-i144/

[25] Smart Home Sensor Kit. *OSOYOO* [online]. [cit. 2017-05-23]. Dostupné z: http://osoyoo.com/2016/08/12/smart-home-sensor-kit/

[26] Reset arduino with code. *ARDUINO* [online]. [cit. 2017-05-24]. Dostupné z: https://forum.arduino.cc/index.php?topic=49581.0

[27] VIRIUS, Miroslav. *Jazyky C a C++: kompletní průvodce*. 2., aktualiz. vyd. Praha: Grada, 2011. Knihovna programátora (Grada). ISBN 978-80-247-3917-5.

[28] PETZOLD, Charles. *Programování Microsoft Windows Forms v jazyce C#*. Přeložil Karel VORÁČEK. Brno: Computer Press, 2006. ISBN 9788025110584.

[29] ŠIRŮČEK, P. Automatizace domácího mikropivovaru. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 42 s. Vedoucí bakalářské práce Ing. Tomáš Marada, Ph.D..

9 SEZNAM ZKRATEK

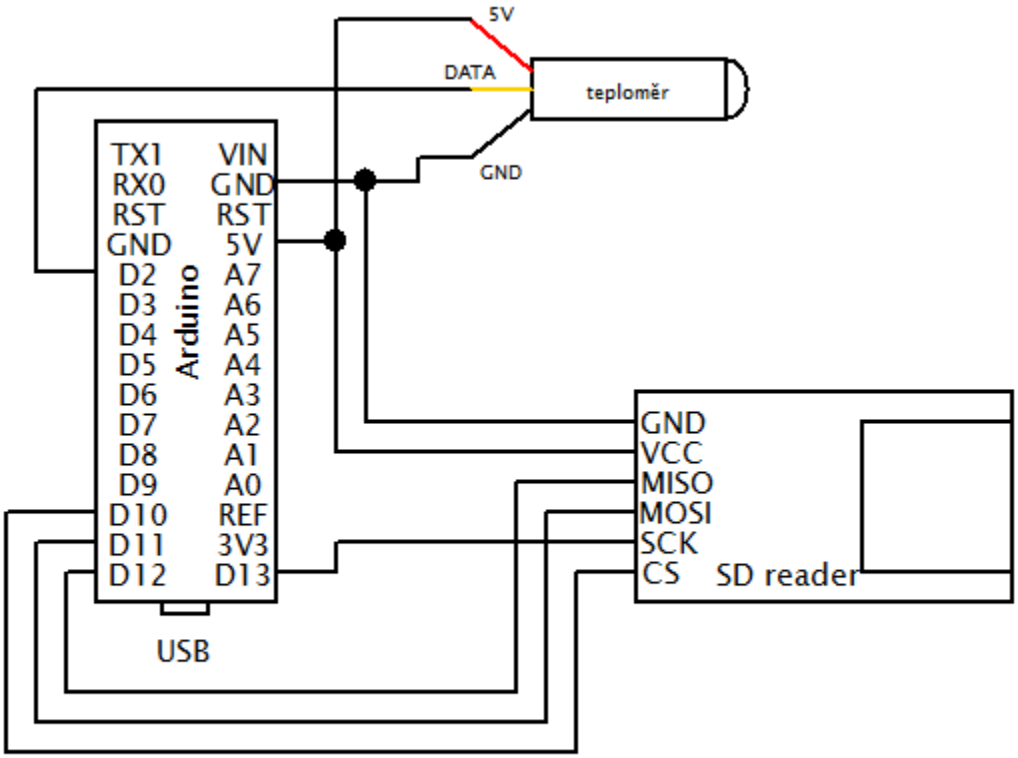
PWM – pulzně šířková modulace
USB – univerzální sériová sběrnice
IDE – integrované vývojové prostředí
SPI – sériové periferní rozhraní
1-wire – sběrnice one-wire
API – rozhraní pro programování aplikací

10 SEZNAM PŘÍLOH

Příloha A: Schéma zapojení PLCduina
Příloha B: Schéma pinů Arduina Nano [15]
Příloha C: CD s programem pro Arduino a SpeedAPI

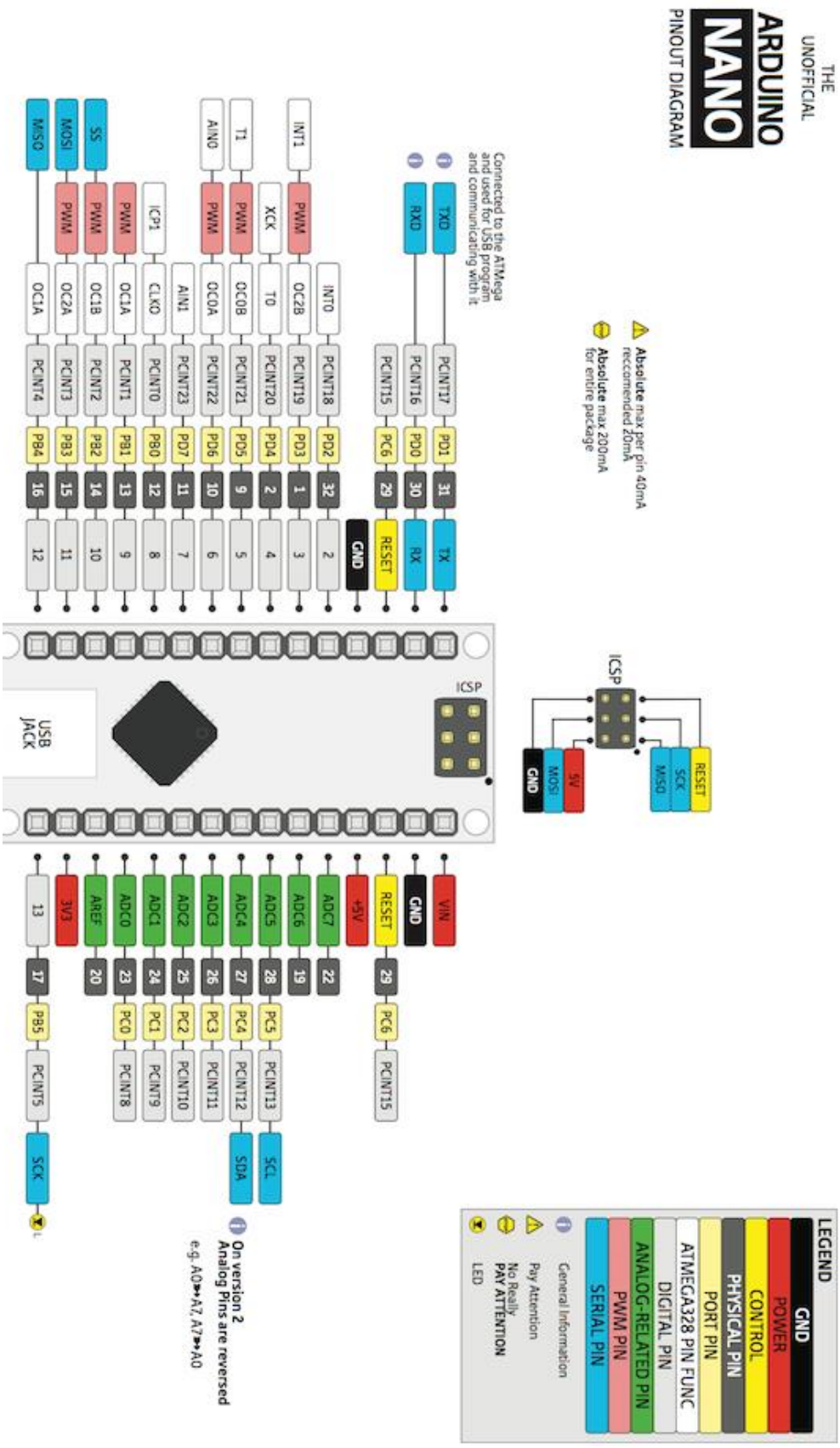
Příloha A:

Schéma zapojení PLCduina



Příloha B:

Schéma pinů Arduina Nano [15]



Příloha C:

CD obsahující:

- Program pro Arduino s příponou *.ide*
- Spustitelný program SpeedAPI pro editaci instrukcí s příponou *.exe* a projekt tohoto programu v řešení Microsoft Visual Studio