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

DÍLČÍ OKRUHY NÍZKONÁKLADOVÉ AUTOMATIZACE BUDOVY
SUB-SECTIONS OF COST-EFFECTIVE BUILDING AUTOMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Lukáš Holoubek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Zdeněk Němec, CSc.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Lukáš Holoubek**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Zdeněk Němec, CSc.**
Akademický rok:	2019/20

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:

Dílčí okruhy nízkonákladové automatizace budovy

Stručná charakteristika problematiky úkolu:

Problematika se zabývá nízkonákladovým řešením řízení předmětných systémů domu s cílem zabezpečit a usnadnit jeho komfortní chod a obsluhu.
Základem je použití mikrokontroléru Arduino, pomocí kterého je zajištěn automatický provoz s minimální údržbou, aplikace vhodných senzorů a elementární vstup uživatelů k ovládání pomocí bezdrátové sítě.

Cíle diplomové práce:

Popište mikrokontrolér Arduino a možnosti jeho využití.
Vyberte a popište předmětné okruhy řízení.
Navrhněte přístrojové a programové vybavení s nízkými náklady.
Realizujte vybrané okruhy řízení pomocí mikrokontroléru Arduino.
Navrhněte vhodné ovládání zařízení pro snadnou obsluhu.
Vyhodnoťte výhody a nevýhody navrženého řešení.

Seznam doporučené literatury:

Matlab & Simulink. Informace o produktech fy The Math Works. Dostupné z: : //www.humusoft.cz [online: 1.10.19].

ŠVARC, a kol.: Automatické řízení, CERM, Brno, 2011.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

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

ABSTRAKT

Cílem této práce je automatizace budovy, ve které jsou jednotlivé okruhy řízeny pomocí jednodeskového počítače Arduino a celý systém řízení je ovládán pomocí mobilního telefonu. V teoretické části je popsán jednodeskový počítač Arduino, jeho části, funkce, možnosti jeho použití a komunikace s ním. V praktické části jsou jednotlivé okruhy popsány, realizovány a je vytvořen funkční model domu, který prakticky demonstruje průběh řízení jednotlivých okruhů simultánně, s možností ovládat a ovlivňovat chování celého systému pomocí mobilním telefonu.

ABSTRACT

The goal of this thesis is a building automation in which the circuits are controlled by a single-board computer Arduino and the entire system is operated by a mobile phone. The theoretical part describes a single-board computer Arduino, its parts, functions, capabilities and communication with it. In the practical part, the controlled circuits are described, implemented and a functional model of a house is created. The model of the house practically demonstrates behavior of the controlled circuits simultaneously with a possibility to manage and influence the system using a mobile phone.

KLÍČOVÁ SLOVA

automatizace domu, automatizace budovy, automatizace domácnosti, Arduino, řízení budovy, řízení domu, řízení domácnosti, chytrý dům, chytrá domácnost, mikrokontroler, ovládání domu, ovládání budovy, internet věcí

KEYWORDS

house automation, building automation, home automation, Arduino, building control, house control, home control, smart house, smart home, microcontroller, internet of things, IoT

BIBLIOGRAFICKÁ CITACE

HOLOUBEK, Lukáš. *Dílčí okruhy nízkonákladové automatizace budovy,* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky.

PODĚKOVÁNÍ

Děkuji Ing. docentu Zdeňkovi Němcovi CSc. za pomoc při vedení diplomové práce. Rád bych také poděkoval své rodině za podporu a trpělivost.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. docenta Zdeňka Němce CSc. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 1. 5. 2020 ……

Lukáš Holoubek

OBSAH

1 ÚVOD

Automatizace různých okruhů v domech a průmyslových budovách je stále více žádaná a cenově dostupnější. Existuje celá řada technologií, které řízení dílčích okruhů mohou zajišťovat. Vhodným elektronickým komponentem pro řízení mohou být např. mikročipy, přesněji mikrokontrolery. Celá řada typů mikrokontrolerů je již delší dobu hojně využívaná, a to nejenom pro automatizaci okruhů v domech, přesto uživatelský přístup a aplikace těchto mikročipů není triviální a ve většině případů vyžaduje pokročilé znalosti v oblasti elektroniky. Nahrávání programu do paměti mikrokontroleru, i další komunikace s ním je obtížná bez připojení dalších elektronických členů, které tento přístup zajišťují. S příchodem jednodeskových počítačů je však právě tato komunikace mezi uživatelem a mikrokontrolerem usnadněna. Jednodeskové počítače jsou modulární, nízkonákladové, uživatelsky přívětivé a kontinuálně otevírají nové možnosti, jak vytvářet sofistikovanější okruhy řízení snadněji. Jejich výhodou je také skladnost a malá spotřeba elektrické energie. Pro tuto práci byl vybrán významný zástupce těchto jednodeskových počítačů Arduino, který nabízí vlastní prostředí pro psaní kódu a jeho deska už obsahuje vše potřebné ke snadné komunikaci. Výhodou Arduina je, že spousta výrobců na trhu nabízí stále nové inovativní cenově dostupné vstupní í výstupní periferie, které jsou pro desky Arduino určené a použití těchto periferii nebývá komplikované. Arduino nabízí také možnost využívat různé bezdrátové moduly, které jsou s kombinací s mobilními zařízeními, například telefonu dnes velmi žádané např. pro řízení tzv. chytrých domů.

Cílem této práce je využití potenciálu jednodeskového počítače Arduino při řízení vybraných okruhů v domě s možností ovládat tento systém pomocí mobilního zařízení, například telefonu. V teoretické části jsou podrobněji popsány mikrokontrolery, které jsou předmětnou částí Arduina a důvodem jeho existence. V dalších kapitolách jsou popsány funkce a možnosti jednodeskových počítačů, později zaměřené konkrétněji na Arduino. V závěru teoretické části je podrobněji popsán způsob komunikace Arduina s mobilními zařízeními pomocí bezdrátových modulů.

V praktické části jsou realizovány jednotlivé zvolené okruhy řízení, jejich podrobný popis a reálná aplikace na funkčním modelu domu. Všechny použité elektronické periferie jsou podrobně popsány, pro každý z nich je vytvořeno schéma zapojení a k závěru je sestaveno celkové schéma zapojení celého projektu, včetně jeho grafického znázornění.

2 AUTOMATIZACE OKRUHŮ BUDOV

V současnosti jsou domy běžně vybaveny systémy, které alespoň částečně zajišťují samočinný chod různých řízených okruhů. Takové systémy jsou stále pokročilejší a jako celek jsou častěji nazývány pojmy chytrá domácnost, chytrý dům apod. V domech je možné automatizovat chytré osvětleni, termostaty, meteostanice, zásuvky, spínače, zabezpečeni, vytápěni apod. Automatizované okruhy mohou být řízeny centrálně i jednotlivě. Komunikace uživatele k ovlivňovaní činnosti systému chytrého domu je velmi často realizována použitím bezdrátové sítě a různých nástěnných dotykových monitorů, tabletů anebo mobilních telefonů, které má uživatel stále po ruce. Přesto, že algoritmy řízených okruhů mohou být často velmi složité, přístup uživatele by měl být vždy navržen tak, aby byl jednoduchý, přehledný a srozumitelný. Okruhy ovládaných systémů můžeme chápat jako souhrn senzorů, které předávají informace o stavech řízených okruhů do výpočetních zařízení, které tyto signály zpracovávají a vyhodnocují. Vyhodnocená data jsou dále použita pro žádané řízení těchto okruhů a také k informování uživatele o stavech a průbězích jednotlivých zařízení. [1, 32]

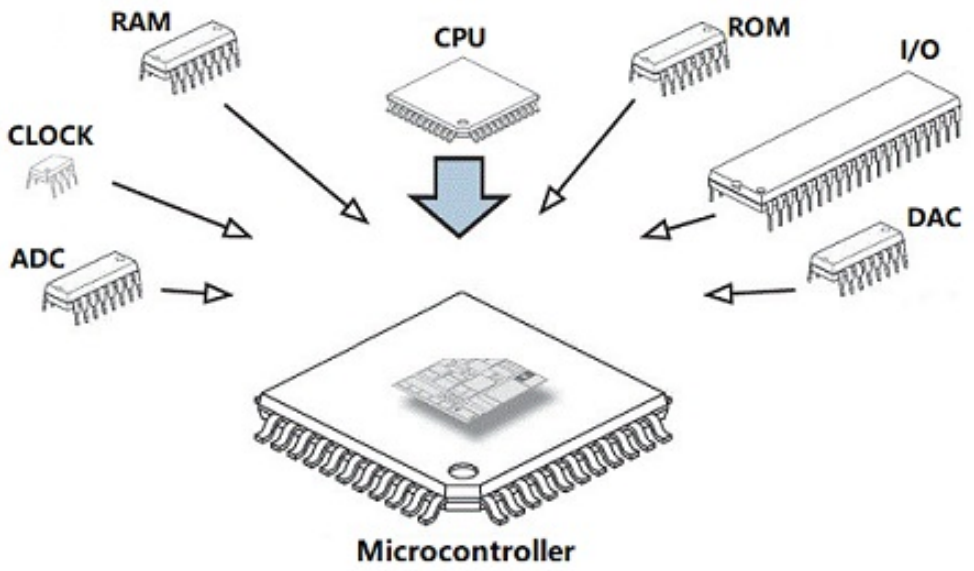
Obr. 1: internet věcí “IoT“ [13]

Využití internetové sítě při automatizaci okruhů v domech a průmyslových objektech nazýváme dnes již známým pojmem “Internet of Things“ zkratkou “IoT“. Výraz je přeložen z angličtiny jako “Internet věcí”. Tento koncept poprvé vznikl už v roce 1982, kdy byl modifikován výdejní automat na Coca-Colu na Melton univerzitě (PA) ve Spojených Státech Amerických tak, aby podával informace o teplotě a dalších stavech prodávaných produktů správci těchto zařízení. Pro přenos dat byla použita sít označovaná jako předchůdce dnešního internetu ARPANET. [1, 32]

3 MIKROKONTROLERY

3.1 Úvod

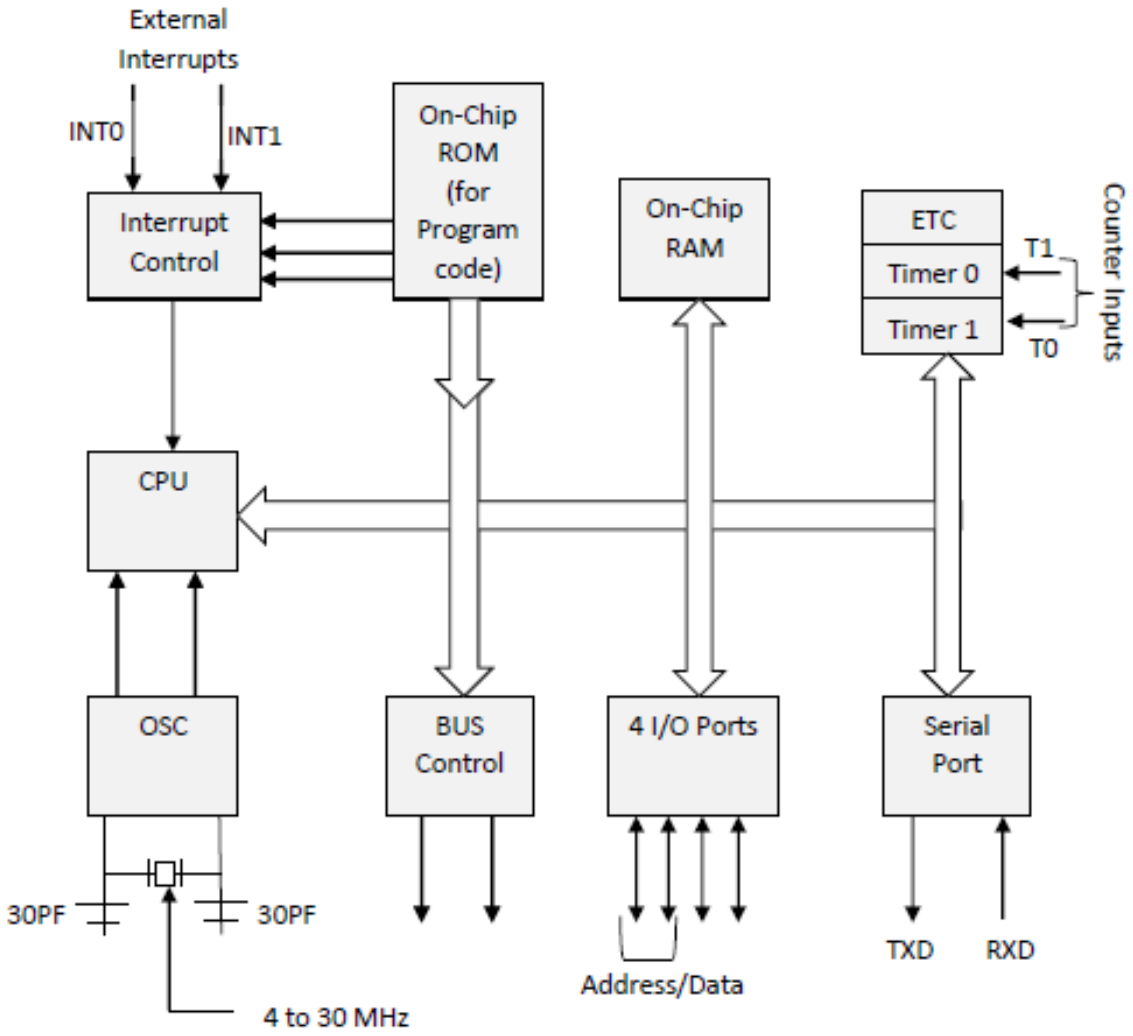
Mikrokontroler, nebo také “microcontroller unit“, dále jen MCU je integrovaný obvod monolitického typu, také nazývaný jako jednočipový mikropočítač. MCU je vyráběný technologií “Very-large-scale integration“ (VLSI), tj. výroba polovodičových prvků s velmi vysokou mírou integrace. Běžný MCU obsahuje mikroprocesor, paměť, vstupy a výstupy (I/O), vše je integrované na jednom spoji. Dnes se můžeme setkat na trhu s různými typy MCU podle datových typů, např. 4bit, 8bit, 64bit i 128bit. [4, 5]



Obr. 2: MCU [21]

3.2 Historie

První MCU byl vyvinut v sedmdesátých letech společností Texas Instruments a nesl označení TMS1802NC. Cílem bylo spojení více elektronických funkčních bloků do jednoho čipu a první aplikace TMS1802NC byla použita při výrobě kalkulaček. Později vyvinutý TMS1000 již obsahoval paměť RAM a ROM. Následoval vývoj MCU společností Intel s označením 8048 a následně velmi rozšířený 8051. Za zmínku stojí také MCU firmy Motorola 68HCxx. MCU se poté začaly vyrábět po miliardách kusů ročně a staly se běžnou součástí všech odvětví. [4, 5]



Obr. 3: Architektura MCU Intel 8051 [6]

3.3 Popis částí

3.3.1 Mikroprocesor

Jádrem MCU je mikroprocesor, ten vykonává aritmetické a logické operace, obsahuje aritmetickou logickou jednotku (ALU), řadič a registry. Taktovací frekvence určuje, kolik cyklů instrukcí je schopen vykonat za sekundu, v současnosti je to v MCU běžně v MHz. Důležitý je také stabilní oscilátor, který se pomocí integrovaného krystalu stará o správnou délku taktovacích pulzů, zajišťuje stabilní hodinový signál a synchronizaci procesoru. [4, 5, 9]

3.3.2 Paměť

Paměť RAM (random-access memory) slouží k ukládání proměnných a výpočtů. Neměnná paměť ROM (read-only memory) slouží k ukládání a spouštění programu a programovatelná FLASH paměť pro rychlé ukládání dat přes I/O porty. [4, 5, 9]

3.3.3 Napájení

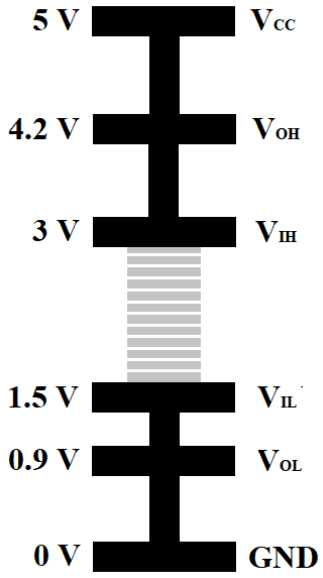
MCU jsou napájeny malým napětím, tj. do 24 V. Řada MCU ATMega použitá v deskách Arduino může být napájena napětím 5 V a rozsah maximálního napětí je 6–20 V.
[4, 5, 9]

3.3.4 Sběrnice

Systémová sběrnice se stará o přenos dat a je navržena tak, aby optimálně zvládla přenos dat ve správném poměru výkonu k procesoru a dalších částí MCU. [4, 5, 9]

3.3.5 Vstupy a výstupy

Vstupní a výstupní porty, tj. I/O porty slouží k propojení MCU s různými externími periferiemi. Každý port se zpravidla stará o velký počet obousměrných pinů. Porty dále dělíme na paralelní, sériové, sběrnicové, A/D a D/A (analogové vs digitální převody), pulsní šířkové modulace (PWM), apod. [4, 5, 9]



Obr. 4: Logické pásmo MCU ATMega [12]

3.3.6 Logické pásmo

I/O porty MCU např. u řad ATMega použitých v deskách Arduino operují v logickém pásmu v rozmezí 0-5 V. Logické pásmo citlivosti je znázorněné na Obr. 4. [12]

3.3.7 Resetovací pojistka

Většina desek Arduino, MEGA2560 nevyjímaje, mají zabudovanou resetovací pojistku, ta chrání desku před průchodem proudů vyšších hodnot, jenž by mohly desku poškodit. Pokud prochází např. USB portem proud vyšší než 500 mA, pojistka přeruší kontakt s deskou a je znovu aktivována až v případě, když proud klesne zpět pod maximální povolenou hodnotu. [11]

3.4 Hlavní výrobci MCU

Mezi hlavní výrobce MCU patří Atmel Corporation (po akvizici Microchip Technology), Texas Instruments, Silicon Labs, Renesas Technology Corp, Intel Corporation, STMicroelectronic, Freescale Semiconductor, Dallas Semiconductor, Fujitsu Semiconductor Europe, ZiLog Company a dále mnoho dalších. [10]

4 ARDUINO

4.1 Jednodeskové počítače Arduino

Arduino bylo vytvořeno vývojáři Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino a David Mellis v Italii na začátku roku 2000 v Ivrea Interaction Design Institute. První Arduino deska byla představena veřejnosti v roce 2005 a jejím cílem bylo usnadnit práci studentů při komunikaci fyzického světa s digitálním. Jedná se tedy o vývojovou desku Arduino, která významně usnadňuje přístup a programováni mikrokontroleru, který je předmětnou částí této desky a jako celek je tato deska často nazývaná jednočipovým, nebo jednodeskovým počítačem. Dnes je oficiální stránka a výrobce produktů Arduina "arduino.cc".

V deskách Arduino jsou používané mikrokontrolery od firmy Atmel a to je ATMegaX, kde X je označení řady 8, 168, 328, 1280, 2560. Desky Arduino obsahují nepájivé připojení vodičů, které nazýváme piny. Kromě vodičů lze také na celou desku Arduino připojovat tzv. Shield nadstavby, které většinou obsahují stejné rozložení nepájivých pinů a k tomu navíc i např. nepájivé pole a další. Shield nadstavby jsou také využívaný pro další rozšíření např. wifi, ethernet apod. Připojení Arduina k počítači je často realizováno USB portem, který pomocí softwarového řešení simuluje linku RS-232 C. Informace o deskách Arduino a jejich softwaru jsou volně dostupné tzv. "open source" a zároveň s rozsáhlou podporou Arduino komunity. Jeho "open source" vlastnost také v případě softwaru nabídne velké množství knihoven vytvořených dalšími zkušenými programátory na celém světe a v případě hardware velké množství dostupných periferii. Všechny desky Arduino běží na operačních systémech MAC, Windows a Linux.

Arduino desky, stejně jako většina dostupných periferii jsou nízkonákladové a vhodné i pro všechny typy projektů, kde je tato vlastnost esenciální. Na trhu je dnes spousta typů desek Arduino dostupných samostatně, ale je možné zakoupit i kompletní stavebnice, tzv. kity, které obsahují nejpoužívanější senzory a další periferie v jednom balení společně s deskou. [11]

4.2 Open-source Arduino Software (IDE)

Software IDE je běžně dostupný na stránkách Arduina zdarma. Kompletní balíček obsahuje ovladače, knihovny, základní množství ukázkových hotových kódů (Examples) a především vývojové prostředí pro vytváření kódů. Je dostupný pro všechny platformy a stále vychází nové aktualizované verze s dalšími vylepšeními. Software IDE je vytvořen jako univerzální prostředí pro většinu vývojových desek Arduina, pouze je nutné v programu správně nastavit typ desky Arduino a typ komunikace. V prostředí Softwaru IDE je kladen důraz na jednoduchost a přehlednost.

Doporučené jazyky pro Programování Arduina jsou C nebo C++, přesto je možné programování i v dalších jazycích, je ale nutné použití dodatečných konverzí napsaného

kódu. V takových případech se často stává, že i minoritní úpravy kódu po hotové konverzi jsou méně přehledné. V automatizaci bývá často využíváno například programovací prostředí MATLAB / Simulink, kde je možné kromě čistého kódu vytvářet logické prostředí pomocí funkčních bloků.

Prostředí IDE po spuštění nabízí dvě základní části pro psaní samotného kódu, tj. funkce void Setup() a void loop(), zároveň jsou tyto dvě funkce v prostředí IDE povinné. Jak už samotné názvy napoví prostor pro psaní kódu void Setup() slouží k počáteční inicializaci, která proběhne pouze jednou při spuštění programu a void loop() k exekuci samotného programu, který (pokud není definováno jinak) probíhá v opakujících se cyklech. Nahrávání samotného programu je většinou prováděno pomocí USB portu, který je k tomu určený, tento port slouží zároveň jako napájení desky Arduina, nebo k dalším sériovým přenosům. Existují i jiné možnosti nahrávání kódu do desky Arduina pomocí dalších komunikačních kanálů, např. pokud je Arduino vybaveno ethernetovým připojením, nebo bezdrátovým modulem apod. [11]

5 ARDUINO MEGA 2560

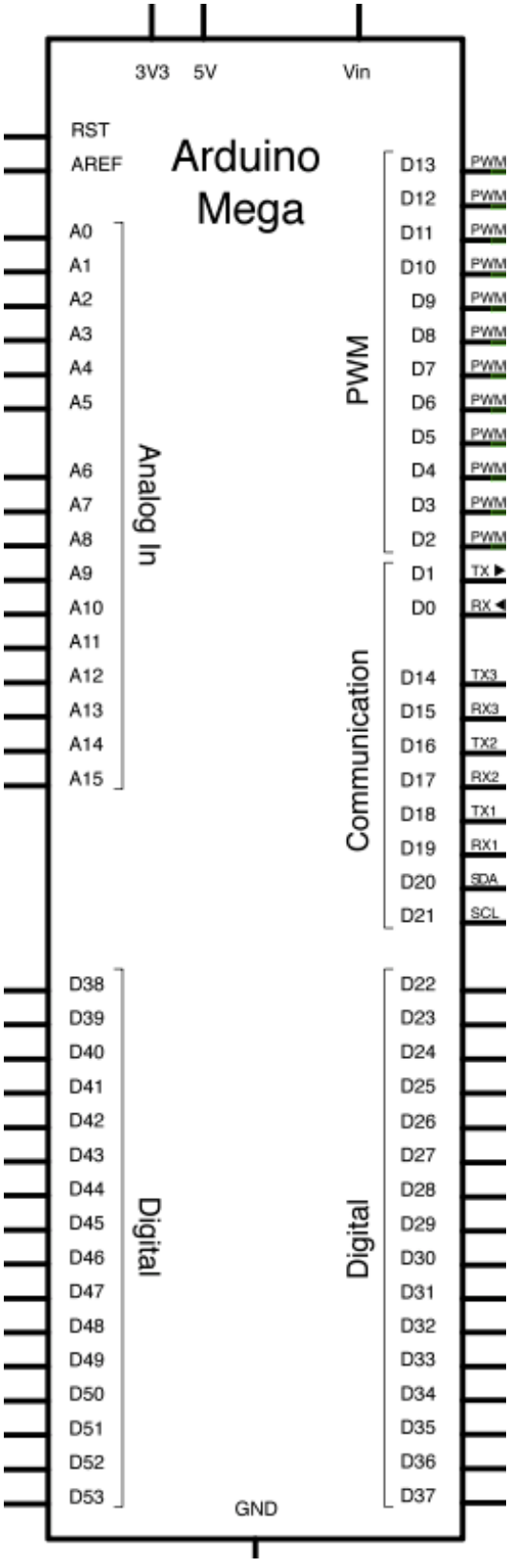
Vývojová deska Arduino MEGA 2560 je přizpůsobena projektům větší složitosti a hardwarové náročnosti. Je založena na MCU ATMEGA2560. Deska má celkem 54 vstupů a výstupů, z toho 15 jsou piny, kde je možnost využívat pulsní šířkovou modulaci (PWM), 16 analogových vstupu a 4 UART hardwarové sériové porty. Deska využívá 16 MHz krystalový oscilátor, USB přístup, napájecí vstup, tzv. ICSP header, který se stará o firmware/program a resetovací spínač. [11, 12]

Tab. 1: Specifikace Arduino MEGA2560 [12]

PŘEHLED PINŮ	
Digitální I/O Piny	54 (z kterých 15 poskytuje PWM)
Analogové vstupní piny	16
Uzemňovací GND piny	5
3.3 V pin	1
5 V pin	5
Reset spínač	1
USART	4 Sériová hardwarová komunikace
ISP programovací pin	6
Krystalový oscilátor vestavěný o kmitočtu	16MHz
USB kabelový port	1
ICSP header	Určený pro nahrávání programu z počítače
Napájecí vstup	1
Resetovací pojistka	Jištění překročení limitu vstupního proudu
NAPĚŤOVÉ A PROUDOVÉ SPECIFIKACE	
Operační napětí	5 V
Vstupní napětí (doporučené)	7–12 V
Vstupní napětí (maximální)	6–20 V
DC proud per I/O Pin	20 mA
DC proud pro 3.3V Pin	50 mA
PAMĚŤOVÉ SPECIFIKACE	
FLASH Paměť	256 KB
SRAM	8 KB
EEPROM	4 KB
VÁHA A ROZMĚRY	
Délka	101.5 mm
Šířka	53.3 mm
Váha	37 g

Většina desek Arduino má kromě originálu prvovýroby, který je garantovaný a vytvořený společností Arduino různé klony / kopie, které jsou ve většině případů téměř identické. Deska je napájena externím konektorem, do kterého lze připojit baterii, nebo AC-DC adaptér. Napájet lze také pomocí USB kabelu a sériového portu např. přímo z desktopového počítače. Pracovní napětí je 5–9 V, uvedené maximální vstupní napětí 20 V by nemělo být dlouhodobě překročeno, hrozí přehřátí desky. [11, 12]

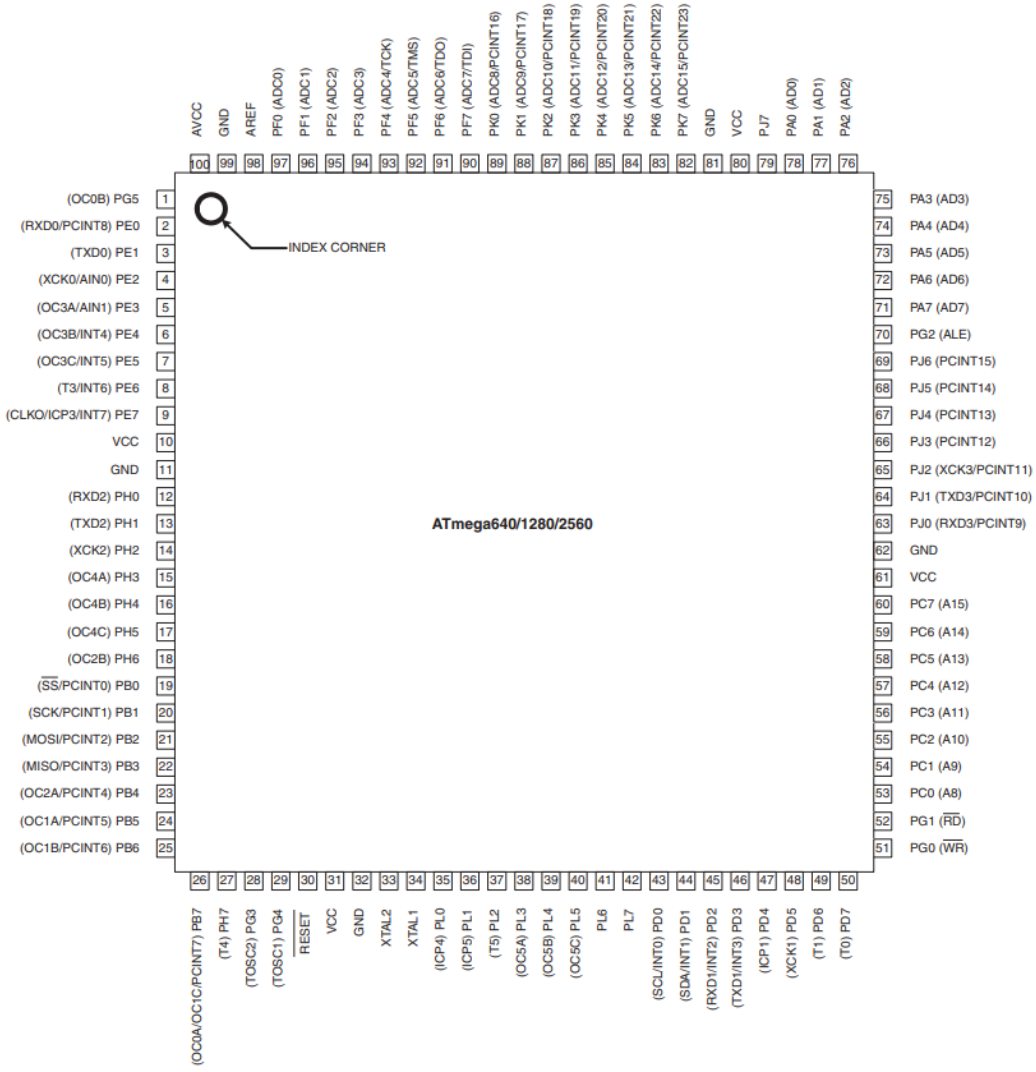
5.1 Přehled I/O pinů Arduino MEGA 2560



Obr. 5: Arduino MEGA 2560 [11]

5.2 MCU Atmel ATMEGA2560-16AU

Čip ATMEGA2560 od výrobce Americké firmy Atmel, mateřské firmy Microchip Technology je předmětnou částí desky Arduino MEGA2560. Rychlost zpracování, tj. rychlost jednoho taktovacího cyklu při velmi nízkém příkonu osmibitové CMOS technologie AVR je 1MIPS / 1MHz. Pro dosažení optimálního výkonu a souběžnosti používá technologie AVR Harvardskou architekturu, tzn. oddělené paměti a sběrnice na program a data. [12]

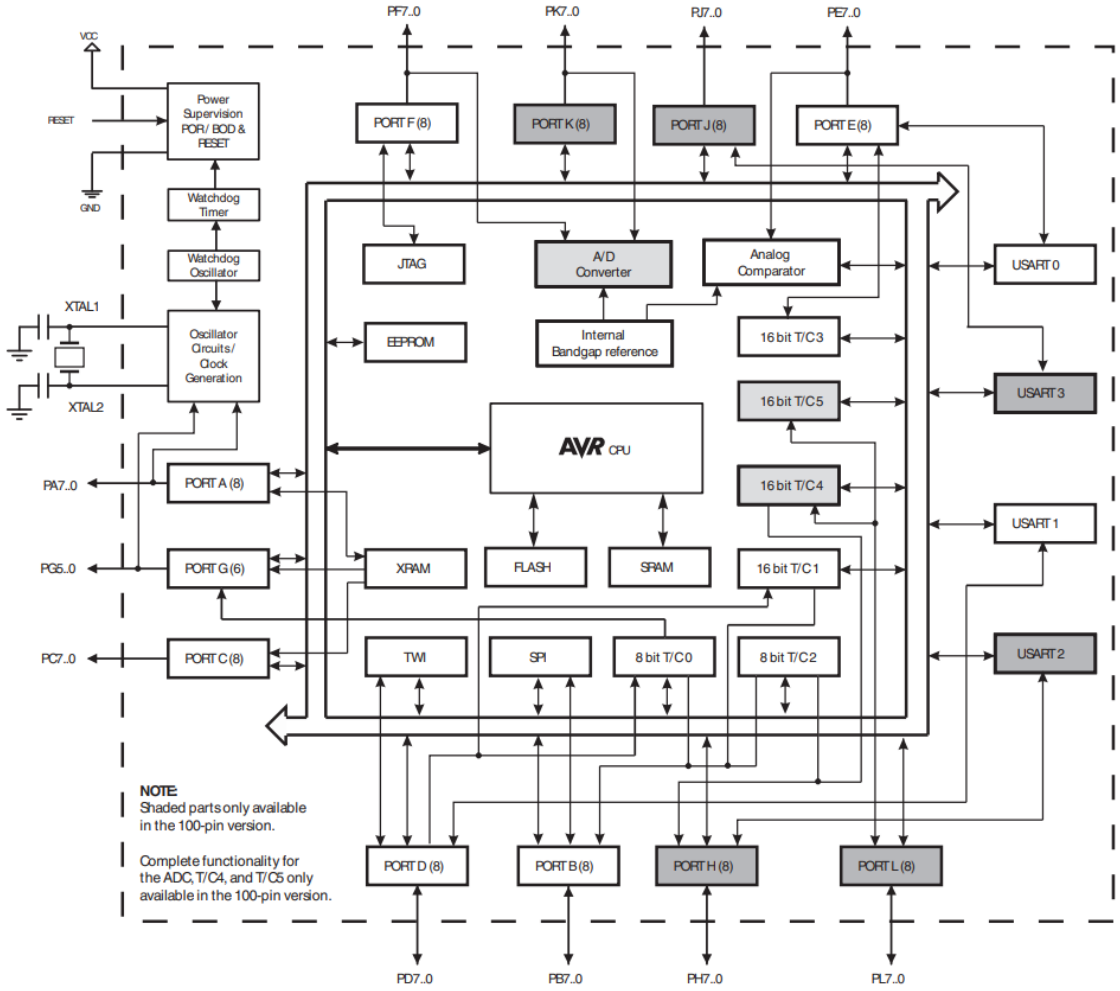


Obr. 6: MCU ATmega1281/2561 pin diagram [12]

Čip ATmega2560 obsahuje jedenáct obousměrných osmibitových portů A až L, všechny jsou vybaveny “pull-up“ rezistory, které jsou aktivovány i při resetu nebo přerušení taktu. Porty F a K navíc obsahují A/D převodník, celkem má 8/16 kanálů ADC převodníky. viz Obr. 7. Pracovní teplota čipu je -40–85 stupňů Celsia. Čip obsahuje rezonátor 32KB a programovatelný watchdog timer s odděleným čipovým oscilátorem, vestavěný oscilátor je kalibrovaný. Dále obsahuje analogový komparátor, Interrupt funkci při změně signálu pinů. Celkem má 86 I/O pinů, z toho 12 PWM a 4 sériové USART. [12]

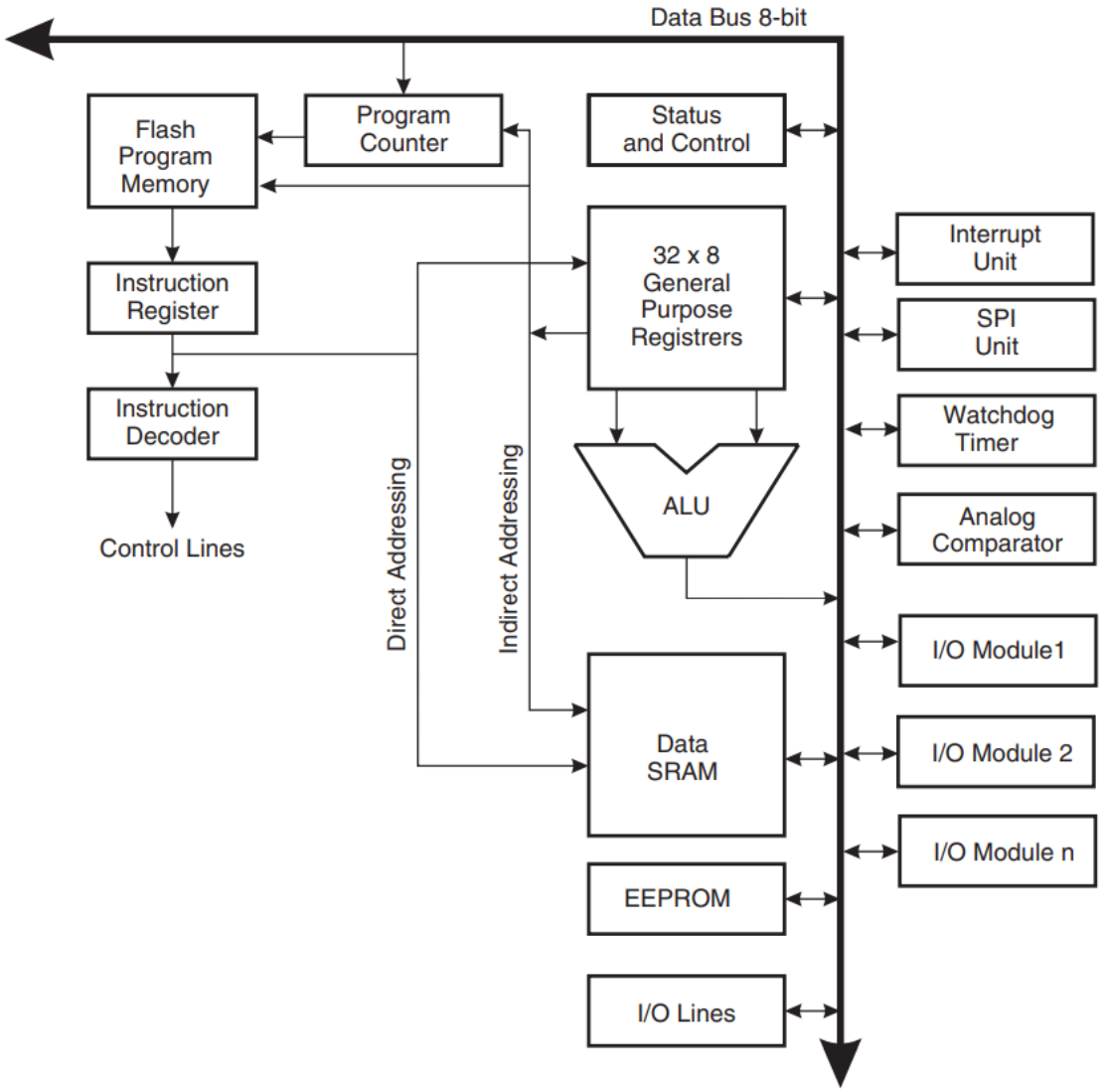
Tab. 2: Specifikace ATmega2560 [12]

PARAMETR	POPIS
Architektura	Vyspělá RISC Architektura
Počet instrukcí	135 Výkonných Instrukcí
Registry	32 × 8 pracovních registrů
Processor	Až 16 MIPS Throughput při 16MHz
Násobič	Integrovaný 2 cyklový
Segmenty paměti	Odolné nevolatilní
Pamět Flash	256 KB programovatelná
Pamět EEPROM	4 KB
Pamět SRAM	8 KB Interní
Cykly Zápisu/Výmazu	10,000 Flash EEPROM
Zachování dat	20 let při 85 °C/ 100 let při 25 °C
Boot kod	Volitelná sekce s nezávislými lock bity
Boot programování	Pomocí integr. boot programu
bezpečnost softwaru	Zámek programování
externí paměťový prostor	64Kbytes
Knihovny	Podpora Atmel® QTouch®



Obr. 7: Blokové schéma ATmega2560 [12]

Technologie AVR pracuje tak, že když je jedna instrukce prováděna další je už předchystána v paměti programu, tím je umožněno provádět instrukce v každém prováděném cyklu programu. AVR umožnuje využití třiceti dvou instrukcí řízené aritmetickou logickou jednotkou (ALU), zároveň jsou zpřístupněny dva registry během každého hodinového cyklu pro jednu instrukci. ALU zprostředkovává logické a aritmetické operace mezi registry nebo mezi konstantou a registrem. Většina instrukcí technologie AVR má jeden 16bitový slovní formát a všechny adresy programové paměti obsahují 16–32bitové instrukce. [12]



Obr. 8: Technologie AVR [12]

Programová paměť FLASH se dělí na dvě části, boot program a aplikační program. ALU operuje ve spojení se 32 obecnými účelové pracovními registry a v během jednoho hodinového cyklu jsou mezi těmito registry prováděny a počítány aritmetické operace. Operace jsou rozděleny na tři kategorie, aritmetické, logické a bitové. [12]

6 KOMUNIKACE S ARDUINEM

6.1 Sériová linka USB port

Výhoda vývojové desky Arduino je, jak již bylo zmíněno, především ve snadné komunikaci mezi uživatelem a MCU, který vykonává předmětné logické operace. Desku Arduino je možné připojit pomocí USB kabelu přímo k počítači, v němž je vytvořen a kompilován samotný program v prostředí IDE. Pří připojení Arduina je nutné pouze nastavení příslušného portu, přes který probíhá přenos dat a USB TTL převodník už zajistí vše potřebné. Tento převodník, který zajišťuje převod dat do MCU bývá zpravidla na pevno vestavěný v desce Arduina, přesto se můžeme setkat s případy, kdy je používán externí převodník a v takovém případě se často používá šest vodičů, do kterých se převodník připojuje. Sériový přenos do Arduina pomocí USB portu lze využívat i k další komunikaci, tj. např. kontrola běhu programu a vykonávaných procesů, nebo přímé ovládání Arduina z počítače pomocí textových příkazů.

Přenos dat je udávaný v baudech. Baud nebo také modulační jednotka se používá k měření počtu přenesených symbolů v čase, jeden baud udává jeden symbol přenesený za jednu sekundu. Podporované hodnoty baud pro desky Arduina jsou 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, 38400, 57600, and 115200. [11]

6.2 Bezdrátový přenos Wifi

V případě využití bezdrátové sítě wifi je nutné k desce Arduino připojit příslušný modul, který tento bezdrátový přenos zajišťuje. Nejčastěji je využíván 32bit modul s označením ESP8266 výrobce Espressif Systems. Tento modul je možné připojit k většině typů desek Arduino pomocí osmi vodičů. Modul ESP8266 je levný čip malé váhy a rozměrů, pracuje na napětí 3.3 V. Nejpoužívanější desky např. Arduino Uno nebo MEGA2560 mají 3.3V výstup k dispozici. V případě použití desky, která tento výstup nenabízí, je nutné použití napěťového převodníku, např. převodník logické úrovně IIC I2C.

Využití modulu ESP8266 je podmíněno způsobu jeho aplikace, lze ho využívat i bez použití desky Arduino. Pokud bychom použili modul samotný, například pro bezdrátové měření teploty, je vhodné zakoupení takového typu, který má integrovaný USB převodník, díky kterému je s modulem snadnější komunikace přímo přes USB port.

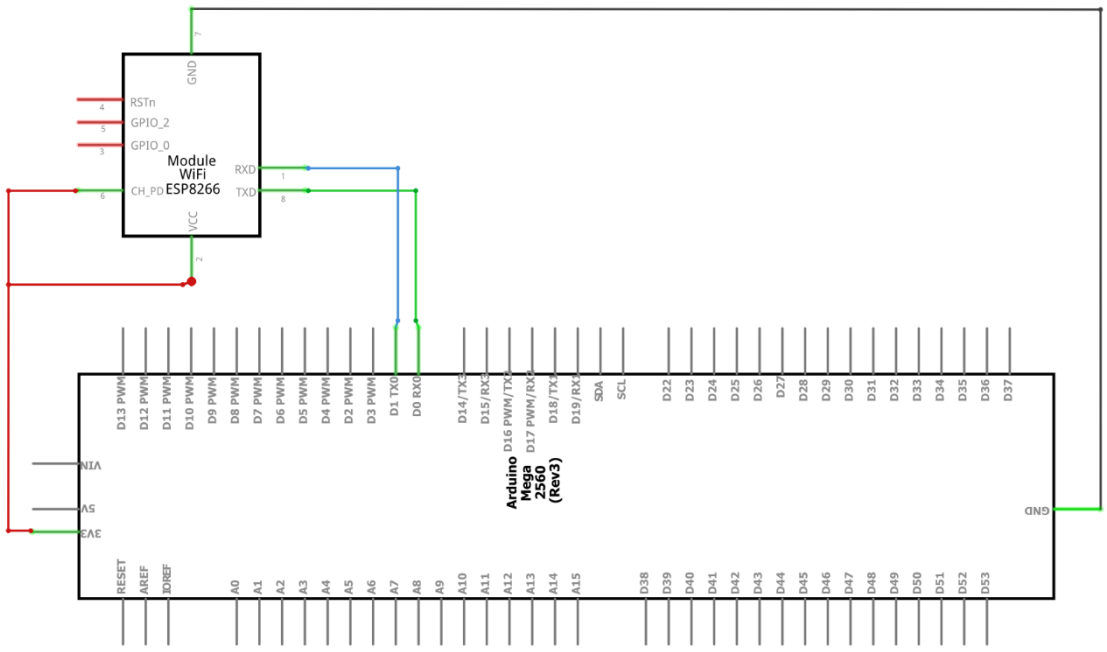
Programovací prostředí pro psaní kódů do ESP8266 může být především Arduino IDE. Další vhodné programovací jazyky jsou např. Micropython nebo Wiring. Pro nastavení ESP8266 v prostředí IDE je nutné nastavení URL desky, přidání desky přes managera desek a stažení vhodných knihoven.

Další modul, který zajišťuje bezdrátové wifi sítě je deska NodeMCU. Na trhu jsou také různé sofistikovanější modifikace desek, jako např. kompletní deska Arduino MEGA2560 s integrovaným wifi modulem ESP8266. [14]

6.2.1 Specifikace wifi ESP8266

Tab. 3: Specifikace ESP8266 [14]

PARAMETR	POPIS
Modulace	802.11 b / g / n
Wifi AP	Wi-Fi Direct (P2P), soft-AP
Komunikační protokol	Vestavěný TCP / IP protokol
PLL	Vestavěný PLL, napěťový regulátor
Pásmo	802.11b mod. + 19.5dBm výstup
Vestavěné periferie	Sensor teploty, integrovaná anténa
Procesor	Nízkoenergetický 32-bit
I/O	SDIO 2.0, SPI, UART
Bezdrátová komunikace	STBC, 1x1 MIMO, 2x1 MIMO
Rámce	A-MPDU, A-MSDU
Pakety	2ms, spojení a přenos datových paketů
Standby režim	1.0mW (DTIM3)



Obr. 9: Schéma zapojení ESP8266 [26]

6.3 Bezdrátový přenos Bluetooth

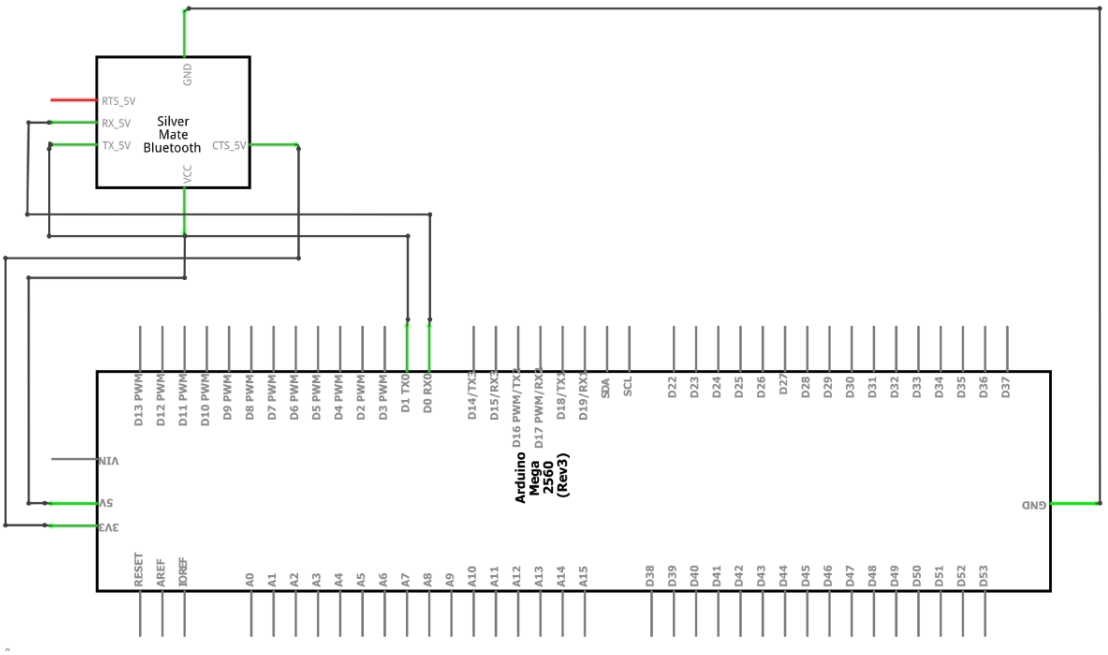
Pro bezdrátový přenos Bluetooth je využíván komunikační modul HC-05. Modul pracuje s napětím od 3.3 do 5 V a je vybaven verzí Bluetooth 2.0. Tento modul je vybaven malou integrovanou anténou a dosah bezdrátové sítě je proto poměrně malý. V otevřeném prostoru se dosah pohybuje obvykle do 10 metrů. Pro připojení modulu k desce Arduino

jsou zapotřebí pouze čtyři vodiče, tj. napájení VCC, uzemnění GND, TXD pro odesílání a RXD pro přijímání dat. Přenos HC-05 je duplexní tzn. dvoucestný a lze jej využívat např. pro spojení dvou desek Arduino, nebo s mobilním zařízením (telefon, tablet apod.), které disponuje funkcí Bluetooth. Modul není vhodný pro přenos obsáhlejších souborů, například multimédia, tj. fotografie a zvuk. Pro přístup ke konfiguraci AT modulu je nutné uzemnění klíčového "key" pinu. [17]

6.3.1 Specifikace Bluetooth HC-05

Tab. 4: Specifikace Bluetooth HC-05 [17]

PARAMETR	POPIS
Napájecí napětí	3,6 - 6 V
Odebíraný proud	40 mA, po spárování klesá na 10 až 15 mA
Komunikační přenos	UART
Rychlost přenosu	MAX 1382400bps
Výkonový parametr	Třída 2 (cca 4dBm / 2,5mW)
Dosah přenosu	10 m
Nastavení	Arduino AT
Režim	Master – Slave



Obr. 10: Schéma zapojení Bluetooth HC-05 [26]

6.4 Další bezdrátové periferie

Mezi další možnosti bezdrátových komunikačních modulů patří sada modulů NiceRF 433MHz, které obsahují vysílač STX882, přijímač SRX887 a k tomu set pružinových antén. Všechny komponenty jsou schopny pracovat s napájecím napětím

od 3.3 do 6 V, u kterého proudový odběr může dosáhnout až 35 mA. Moduly pracují na frekvenci 433/315 MHz. Tyto typy modulů jsou využívané pro propojení dvou MCU, ale běžně se používají i samostatně ke spínaní silových reléových prvků. Moduly jsou hojně využívány např. pro dálkové otevírání bran, závor apod. Vysílač STX882 i přijímač SRX887 jsou připojeny pomocí tří vodičů, vždy napájení, uzemnění a jeden pin na příjímání / vysílání signálu. [19, 20]

V případě Projektů, kde je výhodné využití GSM sítě, je k dispozici deska shield SIM900 výrobce SIMCom. SIM900 má podobné chování, jako například mobilní telefon. S desky Arduina připojené k SIM900 je možné volání na jiná mobilní čísla, připojení k internetu pomocí GPRS apod. Výhoda desky SIM900 jsou jeho frekvence GSM850, EGSM900, DCS1800 a PCS1900, které mají celosvětovou dostupnost, deska SIM900 vyhledává vždy dostupnou frekvenci automaticky. SIM900 dále disponuje sériovým portem, který je schopen přenášet data 1200bps až 115200bps. Zde je možnost použití externího USB – TTL/UART převodníku, který je zapojen pomocí tří vodičů. Pro použití desky SIM900 je potřeba vložení SIM karty. Pracovní napájecí napětí desky SIM900 je od 3 do 5 V. Zapojení desky SIM900 je pomocí čtyř vodičů, tj. napájení 5 V, uzemnění a piny pro přenos dat RX a TX. [18, 20]

U bezdrátových periferií, je dobré zmínit možnost využít set infračerveného přijímače s dálkovým ovládáním HX1838. Tento přijímač je zapojen k Arduinu pomocí tří vodičů, tj., napájení, uzemnění a výstupní vodič signálu. K virtuálním spínačům na dálkovém ovladači je možné přiřadit různé proměnné a ty potom využívat v kódu programu k dalším zásahům. [20]

7 OVLÁDÁNÍ POMOCÍ MOBILNÍHO TELEFONU

Základní komunikace s deskou Arduino je propojením s počítačem přes sériovou linku pomocí USB portu, pomocí které lze načítat hodnoty, jak se program chová a zároveň chod programu ovlivňovat textovými příkazy. Dynamika a vizualizace takového řešení je na nízké úrovni, a především velký desktopový počítač nebo notebook nejsou stále po ruce. V takovém případě je výhodné ovládání desky Arduino pomocí mobilního telefonu, nebo tabletu. Mobilní telefon umožnuje nepřetržitý přistup s možností instantně ovládat okruhy řízení a zajištuje uživateli komfort a informovanost o jeho projektu. Arduino aktuálně nenabízí žádné uživatelské rozhraní pro komunikaci s mobilním telefonem a v takovém případě je možné využít externí aplikace, které tuto komunikaci umožnuji.

V této práci je uvedeno několik příkladů aplikací, které byly testovány a mohly by zmíněný přistup k ovládání Arduina zprostředkovat.

7.1 Aplikace Blynk

Aplikace Blynk od společnosti Blynk, dále jen AB byla vytvořena pro ovládání IoT projektů jednodeskových počítačů typu Arduino, které jsou propojeny s operačním systémem Android nebo iOS, v reálném čase. AB je použitá pro ovládání projektu v této diplomové práci. Použití této aplikace je nízkonákladové jen v případě, že není využívaná komerčně, tato podmínka je v této práci splněna. Volně dostupná verze AB se nazývá “Prototype” a nabízí základní kredit dvou tisíc jednotek. Kredit je nutný k zakoupení grafických komponentů, tzv “widgets“, které nabídnou předmětné funkce, např. jeden spínač v AB stojí dvě stě jednotek kreditu. V této práci bylo použito přes deset tisíc kreditů, celkem bylo zakoupeno dvacet tisíc kreditů za celkovou hodnotu pět set korun. AB nabízí kvalitní vizuální prostředí (“drag and drop UI“) a přímo v aplikaci není nutné psaní kódů. Grafické komponenty, které byly zmíněny, jsou snadno dostupné přetažením z nabídky v menu přímo na pracovní plochu projektu v aplikaci. Jsou to např. spínače různých velikostí a vlastností, posuvníky apod. Stejným způsobem jsou dostupné další komponenty, např indikační a zobrazovací komponenty pro vizualizaci dat, například virtuální diody nebo různé displeje. AB také nabízí různé notifikační komponenty a mnoho dalších. [25]

7.1.1 Nastavení aplikace Blynk

Aby bylo možné AB využívat, je potřeba aplikaci nainstalovat do mobilního zařízení, a je nutná registrace. Pokud byla aplikace úspěšně instalována, je nutná instalace potřebných knihoven v prostředí programovacího soft. Arduina IDE. Nejdříve je nutné pro propojení hardwarového zařízení s AB získat kód tzv. “device Authentication Token”, který je generován pro každého uživatele, jako jedinečný autentizační klíč. Jakmile je úspěšně vytvořen uživatelský účet, je možné se ke stejnému účtu přihlásit

paralelně až z dvaceti mobilních zařízení. Ovládat projekt z jednoho účtu mohou tedy i další členové rodiny při řízení domácnosti.

AB nabízí velké množství různých hotových projektů nazývané “samples”, které pomáhají s vývojem vlastních nápadů a podrobnou dokumentaci ke komponentům a jejich implementaci při psaní kódu programu. Dále je zde možnost být součástí rozsáhlé Blynk komunity s velkým počtem dalších uživatelů podělit o své nápady, projekty a čerpat další informace. [11, 25]

7.1.2 Programování v prostředí IDE s aplikací Blynk

Při programováni s AB nelze použít některé běžně zažité techniky psaní kódu, na které jsme při standardní práci s Arduinem v IDE prostředí zvyklí. Krystalový oscilátor v desce Arduino udává kmitočet 16MHz za sekundu. Reálná naměřená rychlost cyklu je díky dalším procesům, např. načítaní knihoven ve funkci “void loop()“ mnohem nižší, přibližně 117 kHz za sekundu. AB využívá svůj cloudový server přes který zpracovává požadavky mezi mobilním telefonem a deskou Arduino. V případě, že bychom psali kód programu přímo do funkce “void loop()“, tak jak je v IDE prostředí běžné, by došlo k zahlcení serveru AB a přerušení spojení. Nabízí se řešení jednoduše použít příkaz “delay()“. Příkaz delay má přirozeně nežádoucí efekt, který způsobí, že se běh cyklu programu zcela zastaví po dobu použitého parametru příkazu. Zastavení cyklu programu způsobí přerušení funkce Blynk.run, která průběžně zajišťuje spojení AB s Arduinem a znemožní tím, aby se data dostala do cloud serveru AB. Řešením je zajistit, aby funkce “void loop()“ obsahovala pouze funkci “Blynk.run“. Funkce udržuje spojení se serverem AB v reálném čase a všechny další příkazy a procedury by mely být umístěny do jiných funkcí, které jsou volány ve stanovených časových intervalech. Samotné vytváření programu je tedy prováděno objektovým programováním, kterým jsou vytvářeny funkce pro každou požadovanou samostatnou proceduru. Tyto funkce jsou volány v intervalech podle zadané časové hodnoty tak, aby nedošlo k zahlcení přenosu. Cyklus je proveden pomocí příkazu “timer. setInterval(long d, timer callback f)”, který je zadán v inicializační funkci programu “void setup()” pro každou volanou funkci. Je také nutné přidat do funkce “void loop()“ funkci ”timer.run()“, která v každém cyklu provede inicializační interval funkce. Příkaz “ timer. setInterval“ je velmi variabilní, zde je možné předmětnou funkci volat pouze jednou, či jiným definovaným počtem intervalů apod.
[11, 25]

7.1.3 Grafické komponenty v aplikaci Blynk

Po úspěšné instalaci AB v telefonu je po spuštění a přihlášení k registrovanému účtu nutné vytvořit nový projekt. V novém projektu je přiřazen hardware a spojení, které bude v projektu použité. V případě této práce se jedná o desku Arduino MEGA2560 a přenos pomocí sériového portu USB. Tímto je vše připravené k selekci komponentů, které budou vykonávat a přijímat příkazy. V AB se každý grafický komponent přiřadí k pinům podle požadované funkce. V mnoha případech je možné použití zvoleného komponentu tak, že pouze přímo vysílá, nebo přijímá data z Arduina a v takovém případě

se přiřadí přímo k fyzickým digitálním pinům Arduina značených písmenem D a číslem pinu, tj. v případě desky MEGA2560 1-53. Stejný druhý typu pinu, v tomto případě analogového typu je označený písmenem A, opět reflektuje číslem počet analogových pinů dané desky. Z těchto pinů načte potřebné hodnoty, nebo naopak na nich daný příkaz rovnou vykoná. Jako příklad můžeme uvést přidání komponentu spínače. Spínač při stisknutí vyšle jednocestně signál, který zapne nebo vypne LED diodu. Z mnoha důvodů je žádoucí, aby komponent přijímal data a zároveň je i vysílal, např. při použití spínače. Spínač po stisknutí pošle příkaz k zapnutí LED diody a ve stejný okamžik přijímá informaci, jestli tato LED dioda už svítí nebo ne a podle toho příkaz vykoná, či změní svůj stav, barvu apod. Tento "obousměrný" typ pinu je nazýván virtuální pin, a je označený písmenem V, stejně jako v předešlých případech, také číslem pinu. Rozdíl ve využití virtuálních pinů je v tom, že nemusí reflektovat fyzické možnosti použité desky. AB nabízí až 127 virtuálních pinů, které je možné takto využívat. [11, 25]

7.1.4 Nastavení komunikace aplikace Blynk s Arduinem

Nastavení komunikace na straně AB je definováno při vytváření projektu. Na straně programu v prostředí IDE je zpravidla nutné více kroků, které budou popsány v jednotlivých sekcích typů komunikace. Obecně lze také použít možnost automatického nastavení inicializace programu na stránkách "examples.blynk.cc", kde pouze vybereme typ komunikace a typ jednodeskového počítače a dostaneme vygenerované nastavení inicializačních parametrů. Parametry vložíme do vlastního programu v prostředí IDE. Dále budou podrobněji popsány tři nejčastěji používané typy komunikace, sériový port USB, Bezdrátová wifi sít a Bezdrátová komunikace pomoci Bluetooth. [11, 25]

7.1.5 Sériový port USB

Univerzální řešení komunikace AB s Arduinem pomocí běžného desktopového počítače je přes sériovou linku v USB portu. Pro tento typ komunikace je nutné nastavení portu v počítači, přes který bude komunikace probíhat. Je nutné brát zřetel na fakt, že není možné používat USB port k nahrávání programu do Arduina a zároveň mít aktivně připojenou AB k Arduinu, pouze jeden typ komunikace je přes tuto linku možný.

Při nahrávaní knihoven ve standardním nastavení AB obdržíme v adresáři knihoven aplikace "My Documents\Arduino\libraries\Blynk\scripts" script, který umožní spuštění, nastavení a dedikování příslušného portu k Arduinu. Zde je umístěn spouštěcí soubor "blynk-ser.bat", který definuje parametry pro vymezení portu k Arduinu. V souboru je také možnost tyto parametry měnit, např. nastavení požadované rychlosti přenosu "baud rate", apod. Po spuštění souboru "blynk-ser.bat" je ještě nutné definovat, který port bude pro komunikaci používán. Pokud není zřejmé, který USB port Arduino využívá, tak lze tuto informaci zjistit např. přes "Device Manager/ports" (v případě použití OS Windows). V neposlední řadě je potřeba, aby použitý program v prostředí IDE obsahoval definici knihovny "#include <BlynkSimpleStream.h>" a do funkce "void setup()" vložíme "Blynk.begin(Serial, auth);" [11, 25]

7.1.6 Bezdrátová komunikace pomocí Wifi sítě

Komunikace přes bezdrátovou wifi sít je, jak již bylo zmíněno realizovaná připojením příslušného modulu ESP8266. V případě, že byl modul úspěšně nastaven, lze tento typ komunikace použít samostatně bez nutnosti dalšího propojení počítače s deskou Arduino. Deska je tedy pouze napájena externím napájením nezbytným k její činnosti. Přes tento modul je možné vytvoření vlastní bezdrátové wifi sítě, nebo připojení na wifi sít již existující, např. přes domácí router. Připojení na router se jeví výhodné v případě potřeby ovládat okruhy v domě s kteréhokoliv místa na světě, kde je k dispozici internet. V programu v prostředí IDE je nutné přidání potřebných knihoven pro modul ESP8266, ale také pro tento typ komunikace, a tj. “#include <BlynkSimpleEsp8266.h>”. Dále je potřeba přidat “wifi credentials“, tj. definováno pomocí příkazů “char ssid[] = "YourNetworkName";“ a “char pass[] = "YourPassword";“ a do funkce “void setup()” vložíme “Blynk.begin(auth, ssid, pass);” [11, 25]

7.1.7 Bezdrátová komunikace pomocí Bluetooth

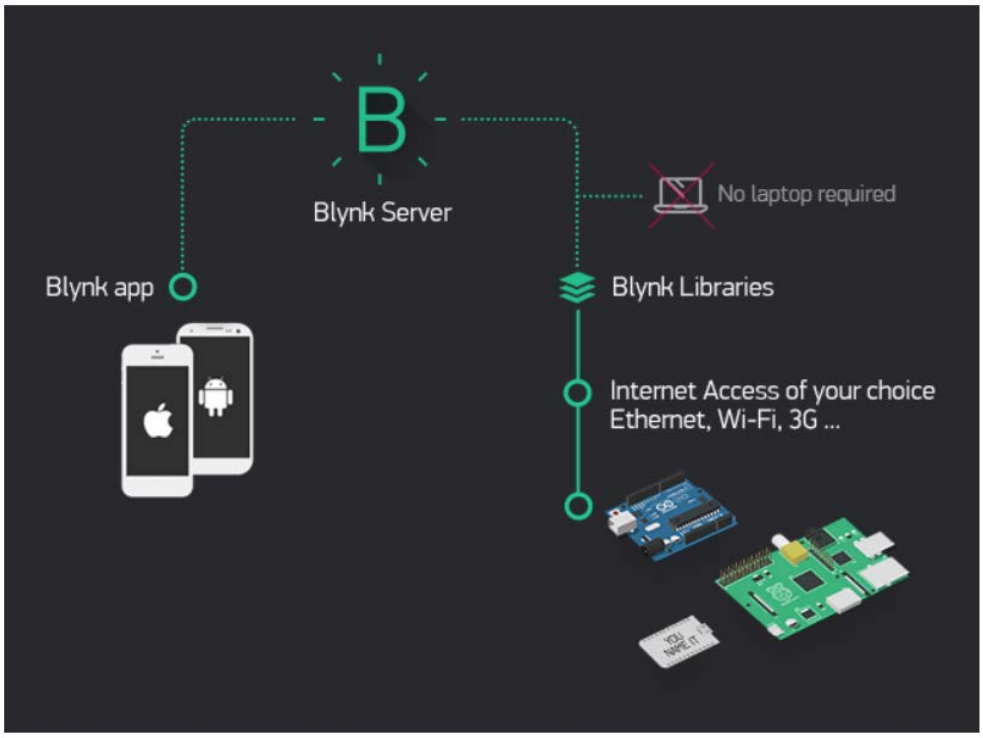
Nastavení bezdrátové sítě Bluetooth je obdobné jako v předchozích případech. Jakmile je úspěšně nastaven Bluetooth modul, nastavíme inicializační kód, který bude obsahovat knihovny, které přidáme příkazy “#include <BlynkSimpleSerialBLE.h>“, “#include <SoftwareSerial.h>“, dále příkaz pro použité piny přenosu “SoftwareSerial SerialBLE(10, 11); // RX, TX“, potom do funkce “void setup()” vložíme příkaz pro rychlost přenosu “SerialBLE.begin(9600);”. Nakonec je nutné přidat verifikační parametr “Blynk.begin(SerialBLE, auth);”. [11, 25]

7.2 Aplikace RemoteXY

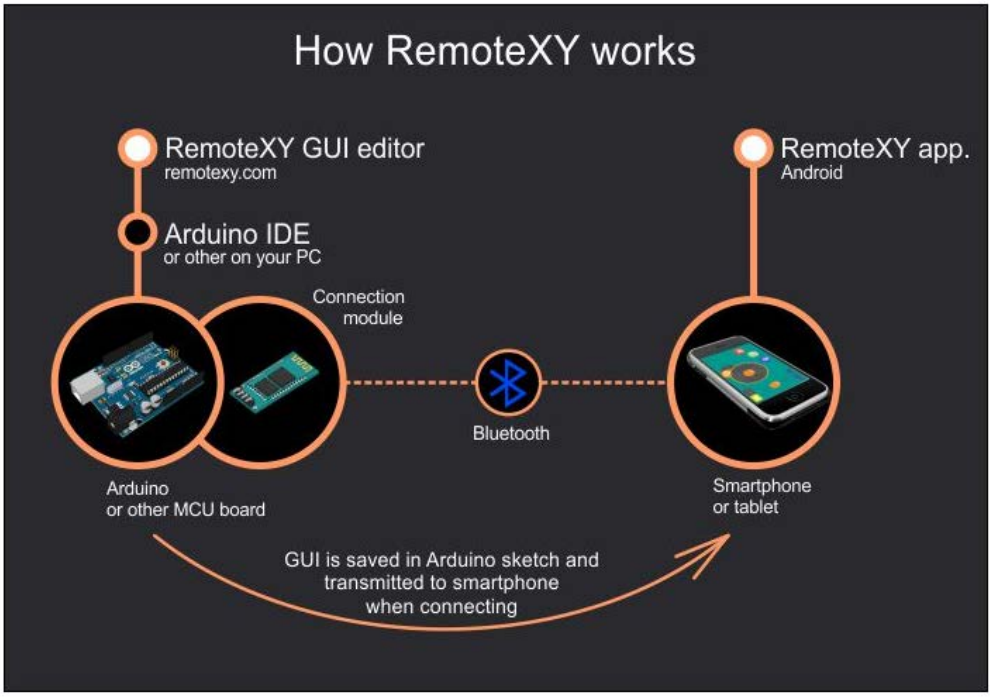
Aplikace Remote XY je další aplikací, která nabízí levné řešení a snadnou komunikaci mezi mobilním zařízením a deskou Arduino. Na stránkách této aplikace je k dispozici editor, který nabídne přímou konfiguraci grafických komponentů, např. spínače, indikační a zobrazovací komponenty. Princip nastavení probíhá obdobným způsobem, viz kapitola 7.1.4 o aplikaci Blynk. Ve webovém editoru je také možnost nastavení konfigurace použité komunikace mezi mobilním zařízením a Arduinem. Po dokončení nastavení je získán potřebný zdrojový kód se všemi parametry, ten je nasledně vložen do vlastního kódu v prostředí IDE. Je nutné stažení aplikace, která nabídne přímé spojení s daným projektem.

Vizuální prostředí aplikace RemoteXY je ve srovnání s aplikací Blynk na nižší úrovni a ve většině případů je komunikace komponentů pouze jednosměrná. Podobnost lze spatřit v popsaném procesu Aplikace Blynk viz kapitola 7.1.3, kde je použití pouze fyzických pinů, tedy bez možnosti použít virtuální piny. V tomto případě jsou tedy možnosti aplikace RemoteXY značně omezené. Zároveň je použití aplikace RemoteXY nízkonákladové a snadné bez nutnosti složitějšího kódování použitých komponentů

v prostředí IDE. Registrace je zdarma a pro neomezené použití všech nabízených komponentů je nutné zaplatit pouze symbolickou částku. [11, 22]



Obr. 11: Aplikace Blynk princip činnosti [25]



Obr. 12: Aplikace RemoteXY princip činnosti [22]

8 NÁSTROJE PRO VIZUALIZACI ZAPOJENÍ

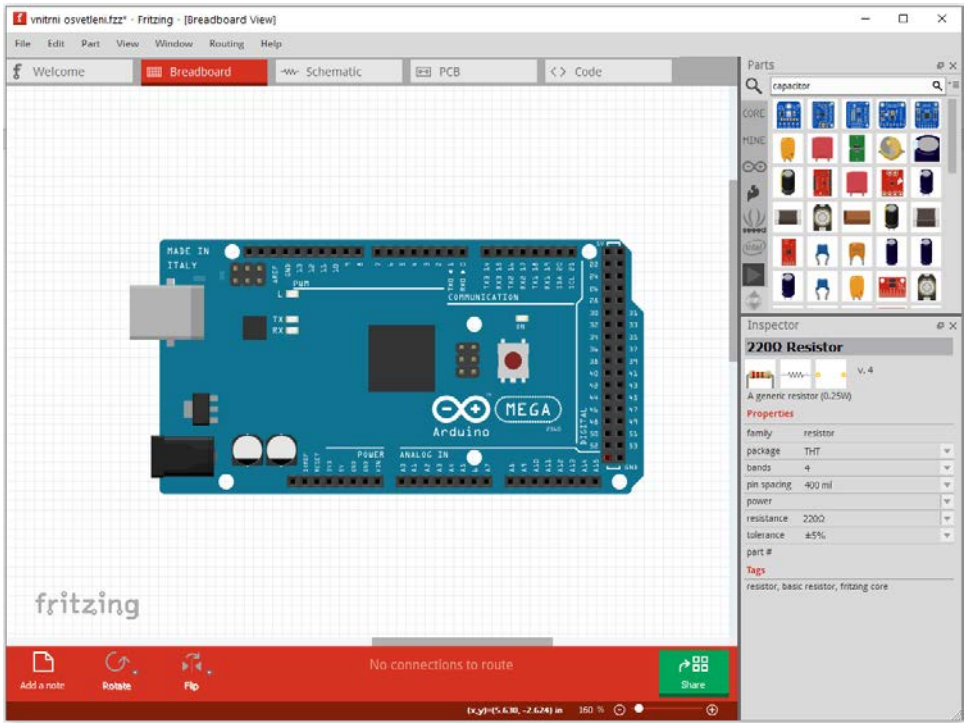
Nástroje pro simulaci a vizualizaci zapojení, především tvorba schémat a grafických řešení, jsou nedílnou součástí při práci s elektronikou. Existují obecné nástroje, ty dovolují pouze použití obecných komponentů, jako jsou běžné elektronické součástky např. odpory, kondenzátory, operační zesilovače, diody apod. V případě desek Arduino je však vhodnější využívat nástroj, který umožní použít knihovny, které obsahují komponenty určené pro tyto desky.

8.1 Nástroj Fritzing

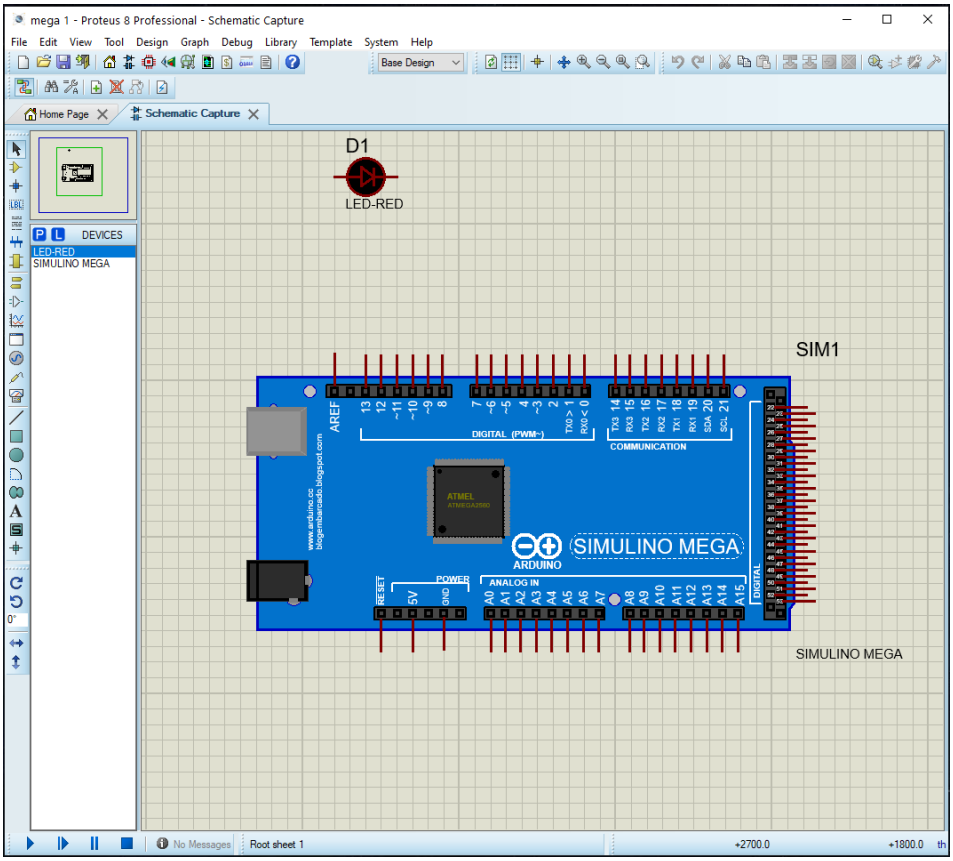
Nástroj Fritzing je "open-source" software vytvořený pro všechny typy projektů založených především na jednodeskových počítačích typu Arduino. Tento nástroj lze používat pro grafické zobrazení projektů jednoduchým přetáhnutím komponentů do projektu. Z hotových grafických zapojení lze vygenerovat hotové schéma a v případě potřeby i "Printed circuit board" na základě kterého lze nechat dané zapojení vyrobit. K dispozici s nástrojem je po registraci také rozsáhlé fórum. Na fóru lze komunikovat s ostatními uživateli a vyplnit mezery v nabraných znalostech. Fritzing je nízko nákladový nástroj a jeho aktivace stála pouze symbolickou částku. Před použitím je ještě nutné dodatečně vyhledat, stáhnout a nainstalovat potřebné knihovny pro použité komponenty v této práci. Tento nástroj bohužel nemá simulační prostředí, přesto pro jeho variabilitu, dostupnost knihoven a komponentů je použitý pro vizualizaci všech zapojení v této práci. [26]

8.2 Nástroj Proteus

Další kvalitní nástroj, který podporuje desky Arduino je nástroj Proteus od firmy Labcenter Electronics Ltd. Proteus je velmi pokročilý nástroj, a má i rozsáhlé simulační prostředí. Grafické prostředí a práce s komponenty vyžadují pokročilé znalosti projektování. Všechny komponenty nejsou dostupné v grafické formě, je však možnost je manuálně vytvořit v nástroji Proteus. Proteus je především komerční nástroj, pro běžné uživatele cenově nedostupný a není tím pádem splněna podmínka nízkonákladového řešení v této práci pro jeho použití. [24]



Obr. 13: Nástroj Fritzing [26]



Obr. 14: Nástroj Proteus [24]

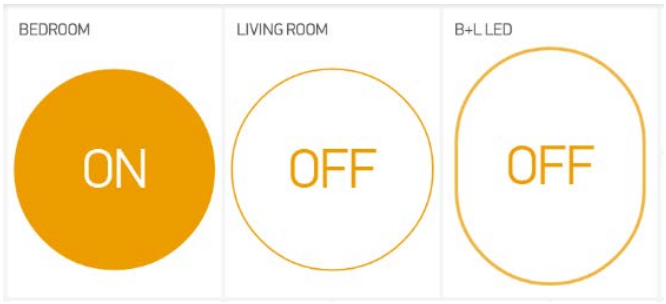
9 VLASTNÍ ŘEŠENÍ

Projekt, který byl vytvořen pro tuto práci obsahuje jednodeskový počítač Arduino MEGA2560 a komponenty, které tento počítač ovládá nebo z nich zpracovává informace za účelem řízení zvolených okruhů v domě. Tento projekt je nízkonákladový, je řízen pomocí mobilního telefonu a bude prakticky demonstrován na malém modelu domu. Zvolené okruhy řízení jsou do modelu domu implementovány a je testováno jejich chování a funkčnost. V dalších řádcích vlastního řešení jsou popsány všechny zvolené okruhy řízení, použité součástky a schéma zapojení.

9.1 Ovládání spínání vnitřního osvětlení

Základní popis

V okruhu ovládání vnitřního osvětlení jsou aplikována dvě světla. Ovládání je realizováno pomocí fyzických spínačů, nebo virtuálních spínačů dálkově s použitím mobilního telefonu. Fyzické i virtuální spínače mají identickou funkci, stiskem je světlo zapnuté, dalším stiskem spínače vypnuté a všechny spínače pracují simultánně. V případě potřeby ovládání všech světel současně je aplikován spínač stejného typu, který zapíná, nebo vypíná všechna světla zároveň. Tento způsob ovládání všech světel může být praktický například při odchodu z domu.

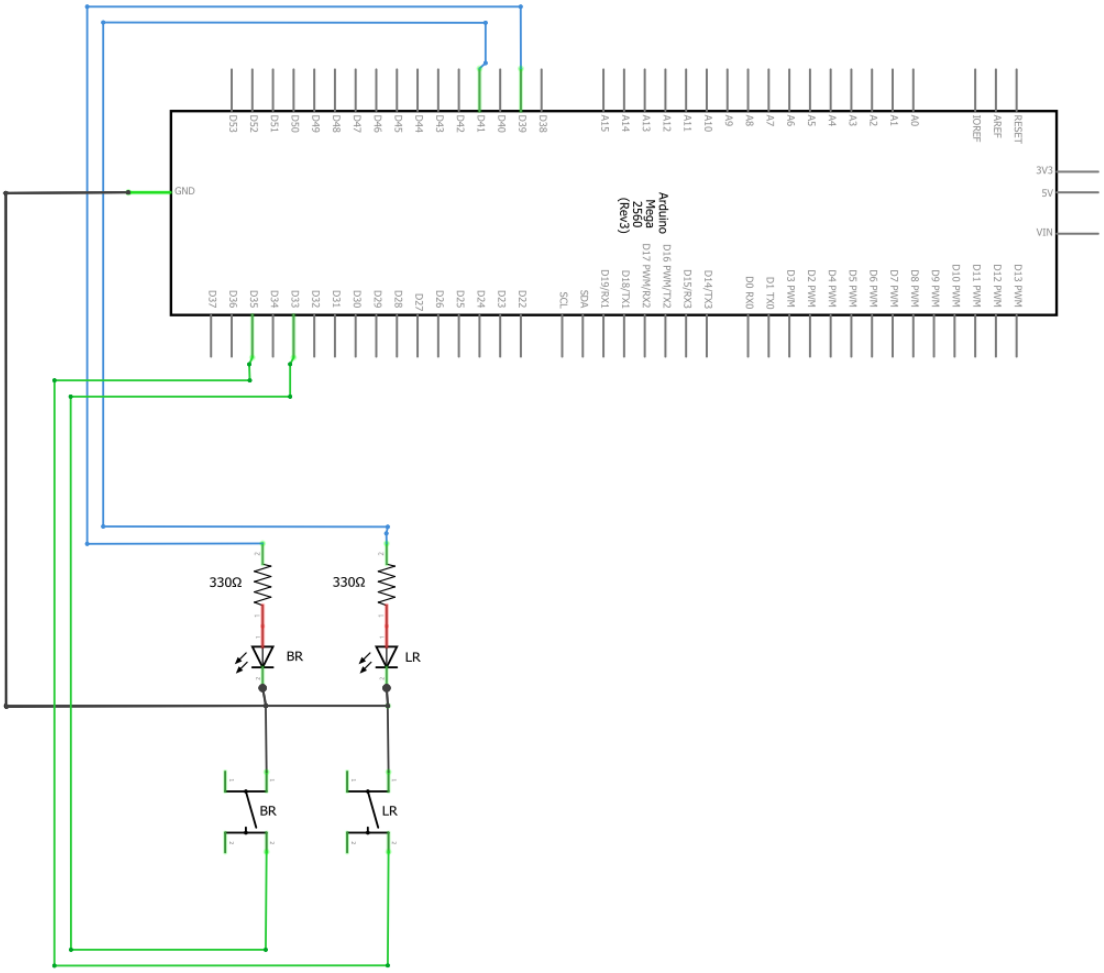


Obr. 15: Komponent virtuálního spínače vnitřního osvětlení [25]

Princip činnosti

Při stisku virtuálního, nebo fyzického spínače je světlo aktivováno. Pokud je v programu indikován stav zapnutý, tak je při stisku spínače světlo vypnuto a v opačném případě je zapnuto. Stav je vždy indikován v mobilním telefonu přímo na virtuálním spínači, kde je indikován zapnutý stav žlutou barvou s nápisem ON a vypnutý s nápisem OFF. Stejný princip platí i pro druhý virtuální spínač, který vypíná všechna světla zároveň. Funkce všech světel je ovládána pouze z mobilního telefonu. Při použití virtuálního spínače všech světel je programem monitorován stav zapnuto / vypnuto obou světel současně. Pokud jsou obě světla vypnutá, tak se zapnou, pokud jsou obě světla zapnutá, tak se vypnou a v jakémkoliv jiném případě jsou světla vypnuta.

Schéma zapojení



Obr. 16: Schéma zapojení vnitřního osvětlení [26]

Použité součástky a jejich specifikace

Pro světla jsou použité LED diody žluté barvy. Výpočtem podle Ohmova zákona a specifikace použité diody je stanovena optimální velikost proudu 15 mA. Pro dodržení této hodnoty je nutné předřadit rezistor, byl vybrán keramický typ 330 Ohm. Spínače jsou použité standardní jednopólové mikrospínače se čtyřmi konektory. Sepnutí probíhá přes "pull-up" rezistor již integrovaný na digitálních vstupech desky, respektive MCU ATmega2560. Na pinech spínačů je vedena logická 1, při sepnutí spínače je pin uzemněn přes pull-up rezistor, hodnota pinu je změněna na logickou 0, na základě, které program provede sepnuti pinu, který aktivuje LED diodu.

Tab. 5: Použité součástky spínání vnitřního osvětlení

SOUČÁSTKA	SPECIFIKACE
2x Mikrospínač	Napětí 250 VAC, max. proud 50 mA
2x Keramický rezistor	330 Ohm, výkon 0,25W, přesnost 1 %
2x LED dioda žlutá	Max. proud v p. směru 20 mA, napětí v p. s. 2 V

Ukázka kódu programu spínání jedné LED diody

```
 1 void Button_1()
 2 {
 3 buttonNew = digitalRead(buttonPinRead);
 4 if (buttonNew == 0)
 5 {
 6   if(LEDState == 0)
 7   {
 8     digitalWrite(LEDPin, HIGH);
 9     LEDState = 1;
10     Blynk.virtualWrite(V4, HIGH);
11   }
12   else
13   {
14     digitalWrite(LEDPin, LOW);
15     LEDState = 0;
16     Blynk.virtualWrite(V4, LOW);
17   }
18 }
19 }
20
21 BLYNK_WRITE(V4)
22 {
23 virtualNew = param.asInt();
24 if (virtualNew == 1)
25 {
26   if(LEDState == 0)
27   {
28     digitalWrite(LEDPin, HIGH);
29     Blynk.virtualWrite(V4, HIGH);
30     LEDState = 1;
31   }
32   else
33   {
34     digitalWrite(LEDPin, LOW);
35     Blynk.virtualWrite(V4, LOW);
36     LEDState = 0;
37   }
38 }
39 }
```

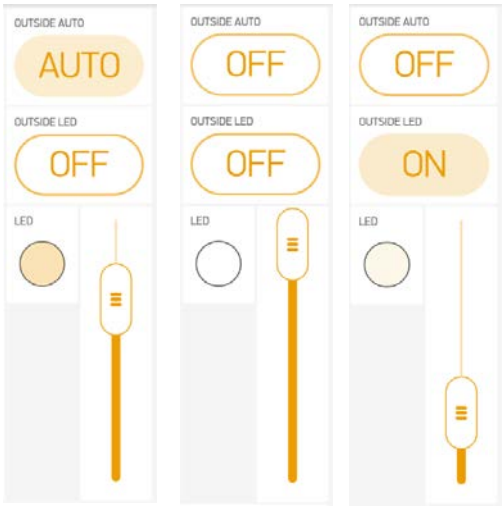
9.2 Samočinné řízení a ovládání venkovního osvětlení

Základní popis

Okruh samočinného řízení venkovního osvětlení automaticky zesiluje / zeslabuje intenzitu LED diody venkovního osvětlení podle přirozeného venkovního světla snímané fotorezistorem. Aktivace samočinného venkovního osvětlení je realizována pomocí mobilního telefonu s použitím virtuálního spínače zapnutí / vypnutí automatického režimu řízení.

Princip činnosti

Pokud je automatický režim aktivován, nepřetržitě snímá pomocí fotorezistoru intenzitu přirozeného venkovního osvětlení a nepřímo úměrně zesiluje / zeslabuje svit LED diody. Druhá možnost je ovládání intenzity svitu LED diody pomocí analogového posuvníku.



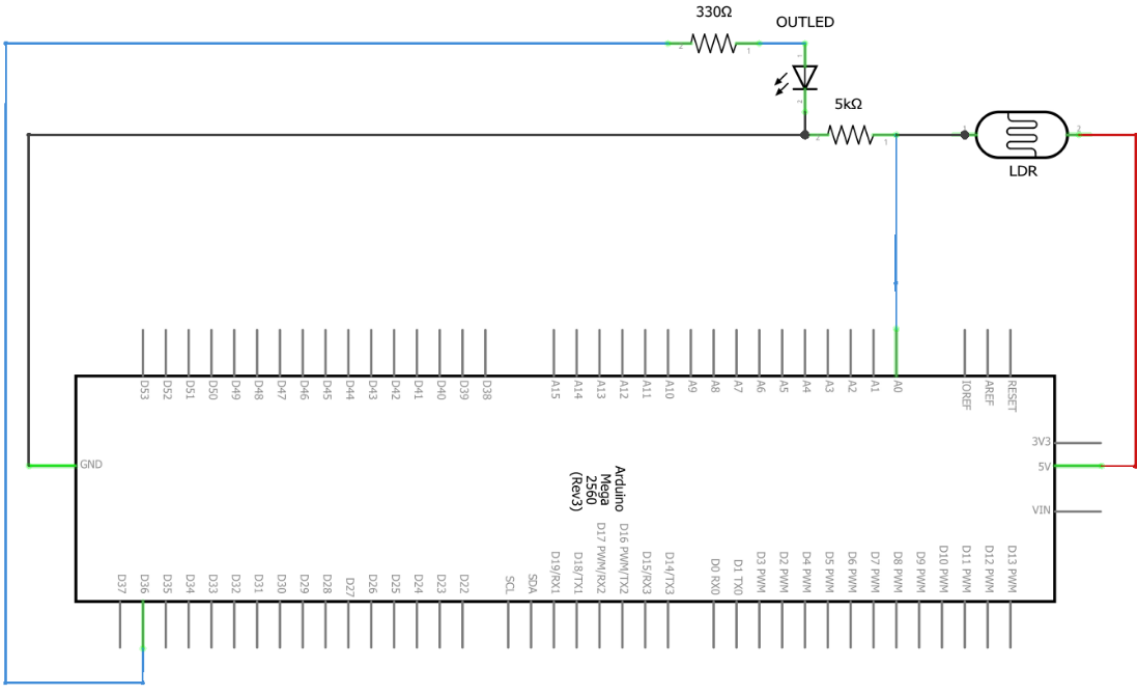
Obr. 17: Komponenty virtuálního spínače venkovního osvětlení [25]

Vizuální Indikace aktuálního stavu je realizována v mobilním telefonu pomocí komponentů virtuálních spínačů. Při zapnutém automatickém režimu je spínač vyplněn žlutou barvou s nápisem ”ON”. Uživatel je informován, zdali venkovní světlo svítí indikační diodou, která mění sytost barvy podle intenzity venkovního osvětlení.

Použité součástky a jejich specifikace

Rezistory a LED dioda jsou použité obdobným způsobem, jak je již zmíněno v kap. 9.1 vnitřního osvětlení. Pro snímání intenzity venkovního osvětlení je použitý fotorezistor GL125. Fotorezistor je pasivní elektronická součástka, která funguje na principu změny odporu podle intenzity světla tzn., že elektrický odpor se snižuje, pokud se intenzita světla zvyšuje. Na trhu jsou dostupné různé varianty fotorezistorů, ve většině případů se liší plochou snímání foto-vodivého materiálu. Rychlost odezvy fotorezistorů nebývá velká, má pomalý nástup a postupně exponenciálně zrychluje. Na směru proudu, který prochází fotorezistorem nezáleží. Citlivost je velká, a to je kromě jeho nízké ceny a dostupnosti jedna z výhod. Nevýhodou fotorezistoru může být zmíněná odezva, stárnutí, a především silně nelineární odezva. [27]

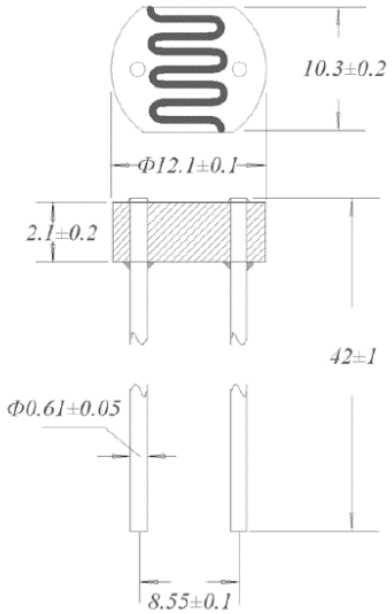
Schéma zapojení



Obr. 18: Schéma zapojení venkovního osvětlení [26]

Tab. 6: Použité součástky spínání venkovního osvětlení [27]

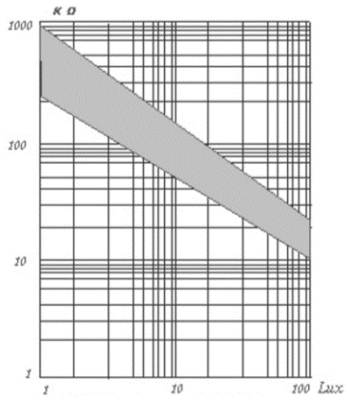
SOUČÁSTKA	SPECIFIKACE
Keramický rezistor	330 Ohm, výkon 0,25W, přesnost 1 %
Keramický rezistor	5k Ohm, výkon 0,25W, přesnost 1 %
Fotorezistor GL125	Max. napětí 250 V, rezistivita světla dána logaritmicky
LED dioda zelená	Max. proud v p. směru je 20 mA, napětí v p. sm. 2 V



Obr. 19: Rozměry fotorezistor GL125 [27]

Tab. 7: Specifikace fotorezistoru GL125 [27]

SOUČÁSTKA	PARAMETR	SPECIFIKACE
Fotorezistor GL125	Max. napětí	250VDC
	Spotřeba energie	200mw
	Rezistivita světlo	5 až 100Kohms
	Rezistivita tma	přes 1Mohms
	Pracovní teplota	-30 až +70 °C



Obr. 20: Závislost odporu na intenzitě světla fotorezistor GL125 [27]

Ukázka kódu programu venkovního osvětlení

```
 1 void LDR_Stepper()
 2 {
 3 sensorValue = analogRead(A0);
 4 outputValue = 1023 - sensorValue;
 5 outputValue2 = 1023 - sensorValue;
 6
 7   if(outputValue < minimum)
 8   {
 9   outputValue = 0;
10   }
11   else if (outputValue > maximum)
12   {
13   outputValue = 1023;
14   }
15   else if (outputValue2 < minimum_OUT)
16   {
17   outputValue2 = 0;
18   }
19   resultValue = map(outputValue, 1023, 0, 0, Stepper_Distance);
20   OUTLEDindicator = map(outputValue2, 1023, 0, 100, 0);
21 }
22
23 BLYNK_WRITE(V10)
```

```
24 {
25   OUTLEDVirtual = param.asInt();
26 }
27 BLYNK_WRITE(V15)
28 {
29   OUTLEDON = param.asInt();
30 }
31 BLYNK_WRITE(V5)
32 {
33   OUTLEDSLIDER = param.asInt();
34 }
35
36 void OUTLED_LIGHT()
37 {
38     if (OUTLEDVirtual == HIGH && OUTLEDON == LOW)
39     {
40     analogWrite(OUTLED, OUTLEDindicator);
41     }
42     else if (OUTLEDVirtual == HIGH && OUTLEDON == HIGH)
43     {
44     digitalWrite(OUTLED, LOW);
45     }
46     else if (OUTLEDVirtual == LOW && OUTLEDON == HIGH)
47     {
48     analogWrite(OUTLED, OUTLEDSLIDER);
49     WidgetLED outledD(V12);
50     outledD.setValue(OUTLEDSLIDER);
51     }
52     else if (OUTLEDVirtual == LOW && OUTLEDON == LOW)
53     {
54     digitalWrite(OUTLED, LOW);
55     }
56 }
57
58 void OUTLED_INDICATION()
59 {
60     if (OUTLEDVirtual == HIGH && OUTLEDON == LOW)
61     {
62     Blynk.virtualWrite(V5, OUTLEDindicator);
63     WidgetLED outledD(V12);
64     outledD.setValue(OUTLEDindicator);
65     }
66     else if (OUTLEDVirtual == HIGH && OUTLEDON == HIGH)
67     {
68     Blynk.virtualWrite(V10, LOW);
69     Blynk.virtualWrite(V15, LOW);
```

```
70      OUTLEDON = LOW;
71      OUTLEDVirtual = LOW;
72      WidgetLED outledD(V12);
73      outledD.setValue(0);
74      }
75      else if (OUTLEDVirtual == LOW && OUTLEDON == LOW)
76      {
77      Blynk.virtualWrite(V5, 255);
78      WidgetLED outledD(V12);
79      outledD.setValue(0);
80      }
81  }
```

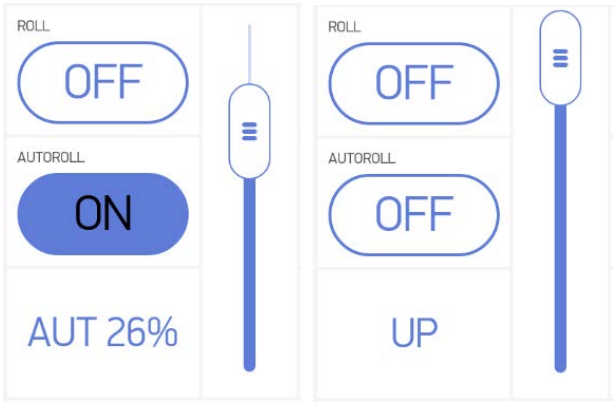
9.3 Samočinné řízení rolety

Základní popis

Okruh pro samočinné řízení rolety realizuje automatické zatahování / vytahování venkovní rolety podle intenzity venkovního osvětlení. Automatický režim je aktivován pomocí virtuálního spínače v mobilním telefonu. Celkový rozsah polohy rolety byl zjištěn heuristickou metodou a je definován v programu.

Princip činnosti

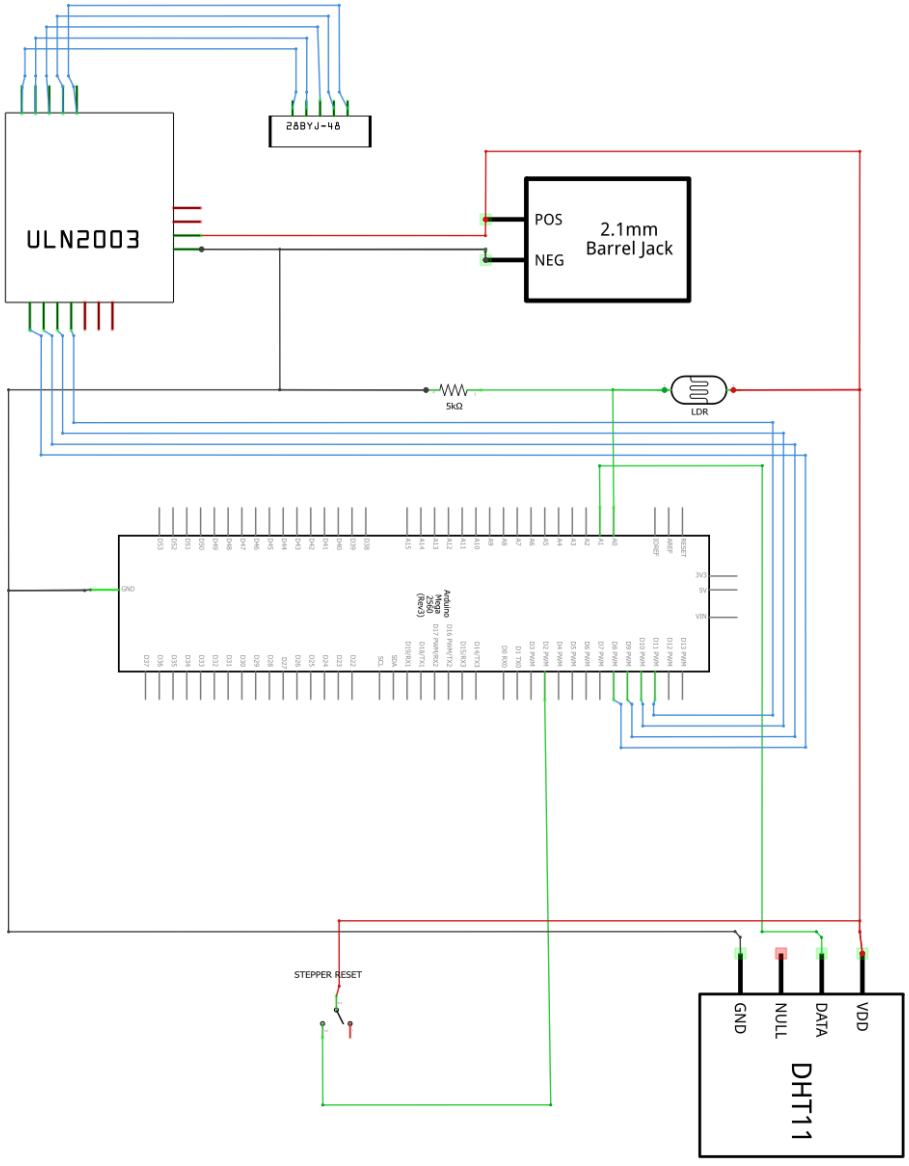
Pokud je automatický režim aktivován, snímá fotorezistor intenzitu přirozeného venkovního osvětlení a roleta je analogicky nepřímo úměrně nastavována krokovým motorem do polohy dle této intenzity. Princip je, že čím vyšší je intenzita venkovního osvětlení, tím víc je roleta ztažena směrem dolů s cílem zabránění průniku světla do objektu. Automatický režim obsahuje podmínku, pokud je teplota v objektu nižší, než programem nastavená hodnota (20 °C), tak je činnost automatického polohování rolety podle intenzity světla ukončena a roleta je vytažena směrem nahoru s cílem většího průniku světla do místnosti a tím zvýšení teploty v místnosti. Tato podmínka trvá, dokud teplota nestoupne zpět nad zmíněnou hodnotu, pak je opětovně aktivován předchozí režim polohování rolety.



Obr. 21: Komponent virtuálního spínače automatického řízení rolety [25]

Informace o stavech nejen samočinné rolety indikuje komponent textového pole. Pokud je automatický režim aktivován je indikován text “AUT“ a poloha rolety v procentech. Pokud je režim samočinné rolety aktivní, zároveň je splněna podmínka nízké teploty v místnosti a roleta je vysunuta nahoru, je indikován text “UP<20°C“. Přerušení automatického režimu nastane při vypnutí virtuálního spínače auto. režimu, nebo při sepnutí fyzického, nebo virtuálního spínače ovládání rolety. Dojde k vypnutí režimu a zároveň se roleta přesune do nové polohy plně vysunuta nebo zasunuta. Auto. režim lze aktivovat kdykoliv, bez ohledu na aktuální polohu rolety. V případě použití tlačítka reset v průběhu činnosti automatického režimu, který má za úkol aktivovat kalibraci polohy motoru, je po kalibraci roleta navrácena zpět do zapnutého automatického režimu a pokračuje v něm, dokud není stanoveno jinak. Analogový posuvník neustále automaticky ukazuje, kde se roleta nachází. [25]

Schéma zapojení



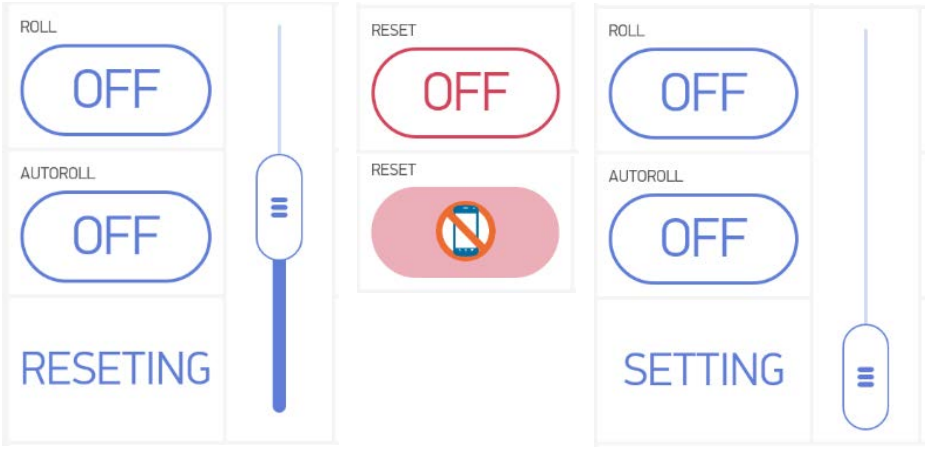
Obr. 22: Schéma zapojení krokového motoru samočinné rolety [26]

Použité součástky a jejich specifikace

Pohyb rolety v této práci zajišťuje krokový motor typ 28BYJ-48 od firmy Kiatronics. Z důvodu, že desky Arduino nejsou vybaveny napájením, pomocí kterého je možné odebírat vyšší proudy, je nutné většinu motorů napájet externím zdrojem. Proud potřebný k napájení motorů bývá zpravidla ve stovkách miliampérů. Další potřebná součástka při použití motoru je ovládací čip, kterým motor ovládáme. Pro tento motor je použitý modul s čipem ULN2003, který už obsahuje celkové zapojení vstupů a výstupů na jednom tištěném spoji. Krokový motor je řízen pomocí pulsní šířkové modulace na PWM výstupech v desce Arduina. [28. 29]

Reset pozice krokového motoru

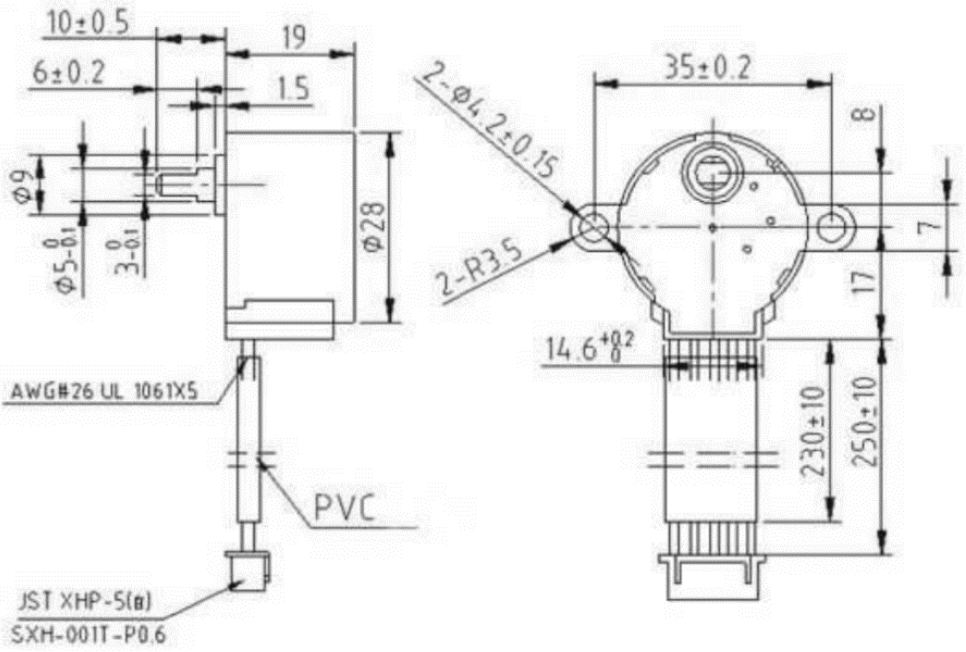
Je důležité, aby krokový motor, který roletu ovládá měl při každém novém zapnutí desky Arduino, nebo v případě výpadku napájení nastavenou startovací polohu, tzv. reset. Reset není možné realizovat softwarovým řešením a je nutné zavádět další elektronické prvky do okruhu tak, aby reset umožnily. Zpravidla se jedná o různé senzory polohy, v této práci byl použitý kontaktní mikrospínač s jazýčkem. Motor je programem řízený tak, aby se při aktivaci desky Arduina, nebo jiném přerušení napájení začal pohybovat směrem k mikrospínači tak dlouho, dokud hřídel rolety fyzicky nenarazí do jeho jazýčku. Po sepnutí mikrospínače dojde ke změně napětí na pinu v Arduinu a je volán příkaz přerušení cyklu programu. Příkaz přerušení zavolá funkci STOP(), zastaví motor, posune ho do startovací pozice a nastaví všechny parametry motoru pro jeho provoz. Zastavení motoru mikrospínačem je možné i použitím běžné funkce v cyklu programu, ale v případě, že by došlo k nějaké neočekávané chybě v běhu programu, roleta by prorazila mikrospínač a došlo by k poškození systému. Dále se po dokončení resetu přívod k ovládacímu čipu motoru vypne, aby se zbytečně nezahříval a spořil energii.



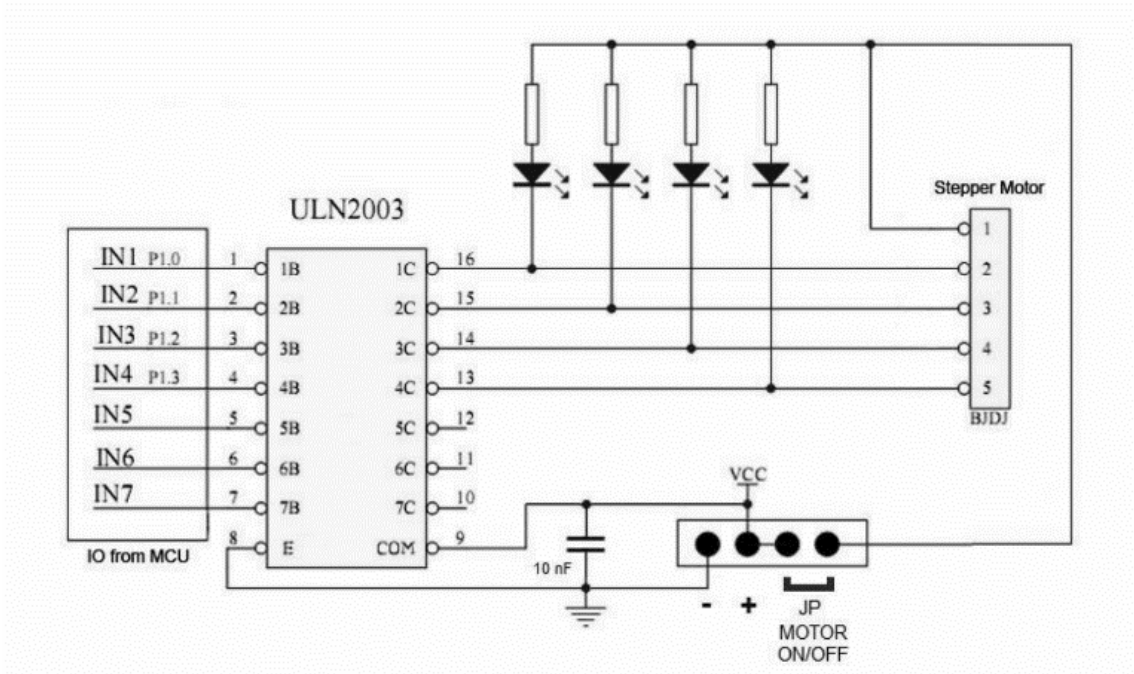
Obr. 23: Komponent resetu krokového motoru [25]

Nastane-li při běžném používání rolety situace, že se z nějakého důvodu motor zastaví v nežádoucí pozici, např. nedovření rolety, je v mobilním telefonu komponent, viz obr. 23 uprostřed, kterým je možné reset motoru vyvolat uživatelem a znovu kalibrovat počáteční polohu motoru. Tento komponent je virtuální spínač, kterým je vyvolán resetovací režim. Vždy, když resetovací režim probíhá, komponent virtuálního

spínače svítí vyplněnou červenou barvou a indikační textové pole vyobrazí nápis “RESETING“. Při odklánění rolety od resetovacího mikrospínače je v textovém poli vyobrazen text “SETTING“.



Obr. 24: Rozměry krokového motoru 28BYJ-48 [28]



Obr. 25: Ovládací modul ULN2003 [29]

Tab. 8: Použité součástky samočinné rolety [28]

SOUČÁSTKA	SPECIFIKACE	
Krokový motor 28BYJ-48	Napájecí napětí	5-12 V DC
	Maximální proud	500 mA
	DC Rezistivita	50 Ohm
	Počet vodičů	4
	Převodový poměr	Jan-64
	Krokový uhel	5.625 / 64
	Frekvence I/O	600 / 1000 Hz
	Moment	34.3 Nm
	Hlasitost	35 dB
Fotorezistor GL125	Max. napětí 250 V	
Modul ULN2003	5 V DC	
Rezistor	5k Ohm	

Ukázka kódu automatického řízení rolety

```
void Stepper_Reset()
{
  RESET_Switch_Read = digitalRead(resetSwitch);
  if (RESET_Switch_Read == HIGH)
  {
  while (RESET_State == true)
      {
       myStepper.enableOutputs();
       myStepper.setAcceleration(200);
       myStepper.setMaxSpeed(reset_speed);
       myStepper.moveTo(10000);
       myStepper.run();
      }
  }
}
void STOP()
{
  if(RESET_State == true)
  {
  RESET_State = false;
  RESET_Away = true;
  WATERSTATUS = true;
  myStepper.setCurrentPosition(0);
  myStepper.setMaxSpeed(reset_speed);
  myStepper.setAcceleration(200);
  }
}

```

```
29 void RESETLED()
30 {
31   if (RESET_State == false && RESET_Away == false)
32   {
33    Blynk.virtualWrite(V17, LOW);
34   }
35   else if (RESET_State == true)
36   {
37    Blynk.virtualWrite(V17, HIGH);
38    Blynk.virtualWrite(V23, "RESETING");
39    Blynk.virtualWrite(V21, 0);
40    STOPPER = 0;
41   }
42 }
43 void STOP_RESET_MOVE()
44 {
45   while (RESET_Away == true)
46   {
47   Blynk.virtualWrite(V23, "SETTING");
48   myStepper.moveTo(-500);
49   myStepper.runToPosition();
50     if(myStepper.currentPosition() == -500)
51     {
52     RESET_Away = false;
53     myStepper.setCurrentPosition(0);
54     myStepper.setMaxSpeed(normal_speed);
55     myStepper.setAcceleration(150);
56     myStepper.disableOutputs();
57     AutoVirtual = 0;
58     Blynk.virtualWrite(V3, LOW);
59     }
60   }
61 }
62 BLYNK_WRITE(V3)
63 {
64   AutoVirtual = param.asInt();
65 }
66 void AUTOMAT()
67 {
68     if (AutoVirtual == 1 && temperature > LOW_TEMP)
69     {
70     myStepper.enableOutputs();
71     myStepper.moveTo(resultValue);
72     myStepper.runToPosition();
73     STOPPER = resultValue;
74     RollState = 0;
```

```
    myStepper.disableOutputs();
    }
    else if (AutoVirtual == 0 && RollState == 0 && STOPPER == 0)
    {
    myStepper.enableOutputs();
    myStepper.moveTo(0);
    myStepper.runToPosition();
    myStepper.disableOutputs();
    }
    else if (AutoVirtual == 1 && temperature < LOW_TEMP)
    {
    myStepper.enableOutputs();
    myStepper.moveTo(0);
    myStepper.runToPosition();
    myStepper.disableOutputs();
    }
}
void DHT_S()
{
  humidity = dht.readHumidity();
  temperature = dht.readTemperature();
  Blynk.virtualWrite(V14, humidity);
  Blynk.virtualWrite(V13, temperature);
  TEMP = map(temperature, 25.0, 35.0, 70, 255);
}
void LDR_Stepper()
{
sensorValue = analogRead(A0);
outputValue = 1023 - sensorValue;
outputValue2 = 1023 - sensorValue;

  if(outputValue < minimum)
  {
  outputValue = 0;
  }
  else if (outputValue > maximum)
  {
  outputValue = 1023;
  }
  else if (outputValue2 < minimum_OUT)
  {
  outputValue2 = 0;
  }
  resultValue = map(outputValue, 1023, 0, 0, Stepper_Distance);
  OUTLEDindicator = map(outputValue2, 1023, 0, 100, 0);
}
```

9.4 Ruční ovládání rolety

Základní popis

Tento okruh je ovládán fyzickým spínačem, virtuálním spínačem a analogovým posuvníkem v telefonu. Zvolený způsob ovládání reflektuje možnou potřebu otevírat, nebo zavírat roletu přímo spínačem umístěného na zdi v domě a zároveň z mobilního telefonu, který zajistí dálkové ovládání rolety.

Obr. 26: Komponent virtuálního spínače ovládání rolety [25]

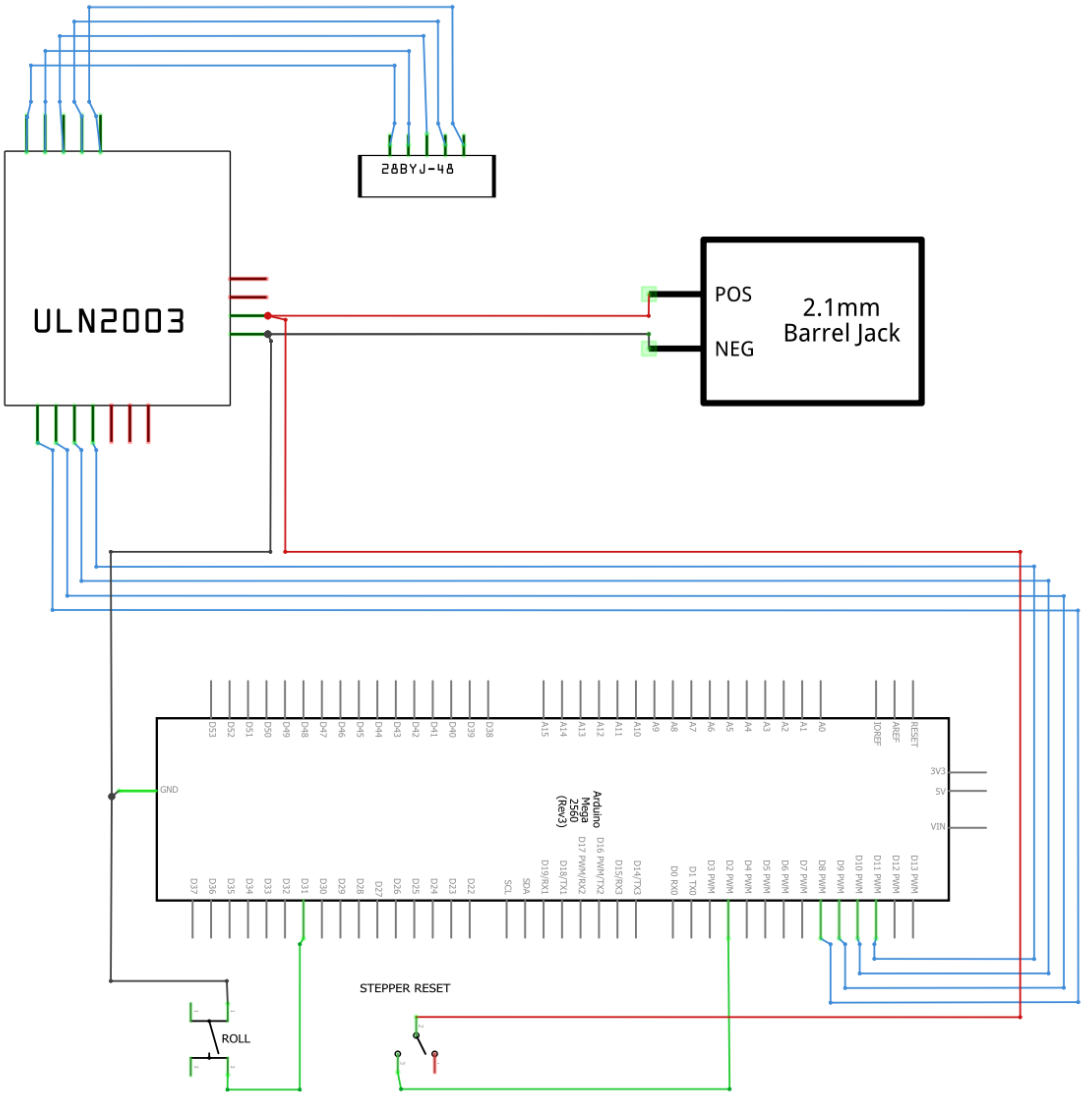
Princip činnosti

Spínací režim je již popsán v kapitole 9.1, princip činnosti. Virtuální spínač při sepnutí reflektuje stav rolety. Pokud je roleta v počáteční poloze, tj. vysunuta nahoře, je na spínači vyobrazen text s nápisem UP. Při stisknutí spínače je roleta stažena. Když je roleta stažena, je v textovém indikačním poli nápis DOWN. Změny v indikačním textovém poli, jsou stejným způsobem indikovány i při použití spínače fyzického, tzn. že virtuální komponent vždy reflektuje aktuální polohu rolety. Také analogový virtuální posuvník se posune vždy tam, kde se roleta nachází. Posunutím rolety pouze pomocí analogového posuvníku bez použití spínačů do jakékoliv polohy ihned přesune do této polohy i roletu a uživatel tudíž může roletu přesunout do kterékoliv pozice v jejím rozsahu v procentech. Pokud se roleta nachází kdekoliv mezi maximální horní a spodní hranicí, tj. z pravidla použitím analogového posuvníku, je její poloha indikována v procentech v textovém poli.

Použité součástky a jejich specifikace

Použité součástky jsou krokový motor, mikrospínač, specifikace těchto součástek viz kapitola 9.3, použité součástky a jejich specifikace.

Schéma zapojení



Obr. 27: Schéma zapojení krokového motoru ovládání rolety [26]

Ukázka kódu programu ovládání rolety

```
1  void ROLL_VIRTUAL_INDICATION()
2  {
3    if(AutoVirtual == 0)
4    {
5      if (myStepper.currentPosition() == Stepper_Distance)
6      {
7      Blynk.virtualWrite(V23, "DOWN");
8      Blynk.virtualWrite(V21, myStepper.currentPosition());
9      }
10     else if (myStepper.currentPosition() == 0
11     && RESET_State == false)
12     {
```

```
    Blynk.virtualWrite(V23, "UP");
    }
    else if (myStepper.currentPosition() != 0
    && STOPPER != 0)
    {
    int ROLL_MAP = map(STOPPER, 0, -6230, 0, 100);
    Blynk.virtualWrite(V23, ROLL_MAP,"%");
  }
  }
  else if (AutoVirtual == 1 && STOPPER != 0
  && temperature > LOW_TEMP)
  {
  Blynk.virtualWrite(V23, "AUTO");
  Blynk.virtualWrite(V21, STOPPER);
  }
  else if (AutoVirtual == 1 && STOPPER != 0
  && temperature < LOW_TEMP)
  {
  Blynk.virtualWrite(V23, "UP","<",LOW_TEMP,"°","C");
  Blynk.virtualWrite(V21, 0);
  }
  else if (AutoVirtual == 1 && STOPPER == 0
  && temperature < LOW_TEMP)
  {
  Blynk.virtualWrite(V23, "UP","<",LOW_TEMP,"°","C");
  Blynk.virtualWrite(V21, 0);
  }
}

void ROLL_BUTTON()
{
RollNew=digitalRead(rollPinRead);
if  (RollNew == 0)
  {
  myStepper.enableOutputs();
  myStepper.setMaxSpeed(normal_speed);
  myStepper.setAcceleration(150);
  STOPPER = 0;
  AutoVirtual = 0;
  Blynk.virtualWrite(V3, LOW);
  Blynk.virtualWrite(V21, STOPPER);
  if(RollState == 0)
  {
    myStepper.moveTo(Stepper_Distance);
    RollState = 1;
  }
```

```
  else
  {
    myStepper.moveTo(0);
    RollState = 0;
  }
}
myStepper.runToPosition();
myStepper.disableOutputs();
}
BLYNK_WRITE(V6)
{
virtualRollNew = param.asInt();
if  (virtualRollNew == 1)
  {
  myStepper.enableOutputs();
  myStepper.setMaxSpeed(normal_speed);
  myStepper.setAcceleration(150);
  STOPPER = 0;
  AutoVirtual = 0;
  Blynk.virtualWrite(V3, LOW);
  Blynk.virtualWrite(V21, STOPPER);
  if(RollState == 0)
  {
    myStepper.moveTo(Stepper_Distance);
    RollState = 1;
  }
  else
  {
    myStepper.moveTo(0);
    RollState = 0;
  }
}
myStepper.runToPosition();
myStepper.disableOutputs();
}

BLYNK_WRITE(V21)
{
  STOPPER = param.asInt();
}
void ROLLSLIDER()
{
  if (STOPPER != 0 && AutoVirtual == 0)
  {
   RollState=0;
   myStepper.setMaxSpeed(normal_speed);
```

```
105    myStepper.setAcceleration(150);
106    myStepper.moveTo(STOPPER);
107    myStepper.runToPosition();
108    myStepper.disableOutputs();
109   }
110   else if (STOPPER == 0 && AutoVirtual == 0
111   && RollState == 0)
112   {
113    myStepper.setMaxSpeed(normal_speed);
114    myStepper.setAcceleration(150);
115    myStepper.moveTo(0);
116    myStepper.runToPosition();
117    myStepper.disableOutputs();
118   }
119 }
```

9.5 Samočinné bezkontaktní měření objemu vody retenční nádrže

Základní popis

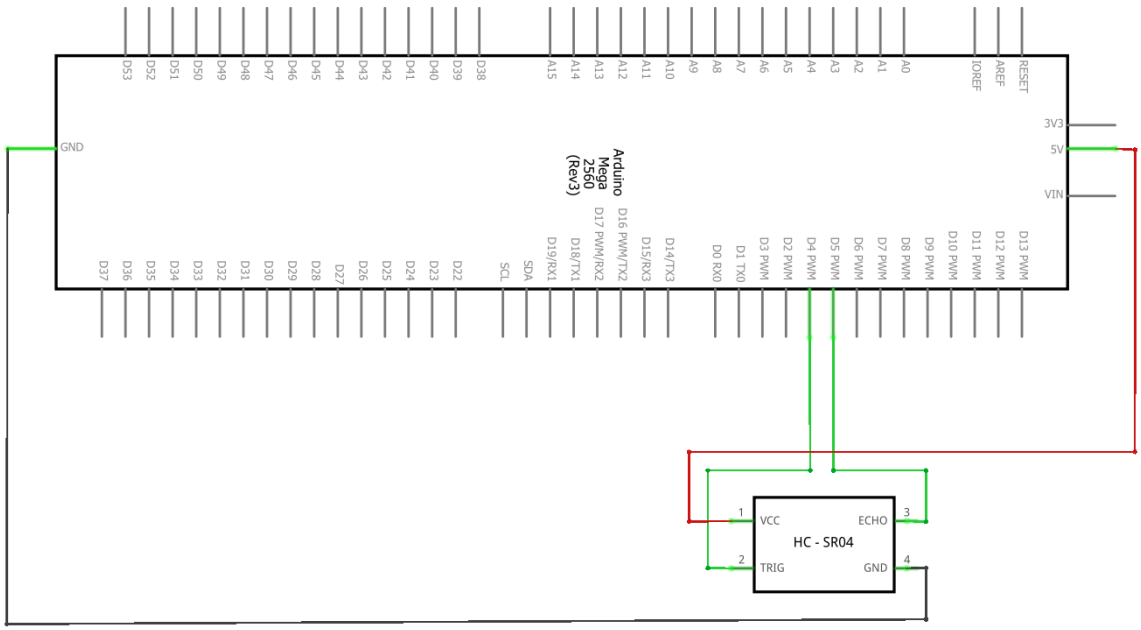
Samočinné měření objemu vody v retenční nádrži je prováděno bezkontaktním měřením pomocí ultrazvukového senzoru typu HC-SR04. Měření je prováděno v programem stanovených ekvidistantních časových intervalech a stav nádrže, tj. její objem v litrech je znázorněn graficky i numericky v mobilním telefonu.

Princip činnosti

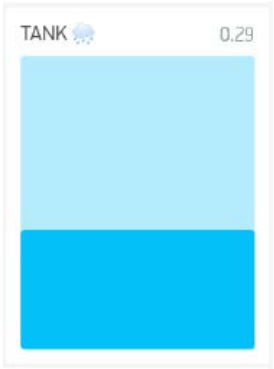
V programu je zadán parametr konkrétní nádrže, tj. její objem. V této práci je použitá nádrž ve tvaru válce a všechny jeho rozměry jsou dané. Ultrazvukový senzor vyšle zvukový impulz směrem k hladině válce a pokud byla detekována vzdálenost větší než nula, je měření pro větší přesnost opakováno ještě pětkrát a přičítáno do proměnné vzdálenosti. Následně je součet těchto pěti měření vydělen pěti a je spočítán průměr. Tato hodnota je použita jako konečná vzdálenost v daný moment a zapsána do proměnné výsledné vzdálenosti. Celková výška válce je sečtena s hodnotou malé vzduchové rezervy před senzorem. Od této hodnoty je odečtena vzdálenost naměřená senzorem, vynásobena obsahem a je dán výsledný objem nádrže v metrech krychlových. Hodnota je vynásobena tisícem a je získán výsledný objem v litrech.

Hodnota objemu nádrže je aktualizována průběžně v delších 30 sekundových časových okamžicích, aby byl co nejméně zatěžován přenos. V případě, že je tvar nádrže jiný, je nutný zásah do kódu programu a stávající výpočet objemu zde nahradit novým vzorcem. Při reálné aplikaci je nejčastěji používaný eliptický tvar retenční nádrže protažený válcem podél středu nádrže. Pro tento typ nádrže byl senzor také úspěšně vyzkoušený, bylo ale nezbytné provést poměrně složitý výpočet objemu nádrže podle polohy hladiny.

Schéma zapojení



Obr. 28: Schéma zapojení senzoru HC-SR04 [26]



Obr. 29: Komponent měření objemu retenční nádrže [25]

Použité součástky a jejich specifikace

Tab. 9: Použité součástky měření objemu vody v nádrži [15]

SOUČÁSTKA	SPECIFIKACE	
HC-SR04	Napájecí napětí	5 V DC
	Maximální proud	2 mA
	Uhel rozsahu senzoru	Max 15 stupňů
	Efektivní vzdálenost měření	2–450 cm
	Přesnost měření	3 mm
	Rozměry senzoru	45 x 20 x 15 mm
	Hmotnost senzoru	8 g

Ukázka kódu programu měření objemu vody

```
 1 void Tank()
 2 {
 3 vzd = sonar.ping_cm();
 4  if (vzd > 0)
 5  {
 6   vzd = 0;
 7     for (int i = 0; i < 20; i++)
 8     {
 9     vzd += sonar.ping_cm();
10     Millis();
11     }
12     vzd = (vzd / 20)*0.01;
13
14 float d = 0.05;
15 float r = d / 2.0;
16 float v_valec = 0.275;
17 float kominek = 0;
18 float obsah = M_PI * pow(r,2);
19 float Vvalec = obsah * (v_valec+kominek-vzd);
20 float Litr = Vvalec*1000;
21
22 Blynk.virtualWrite(V1, Litr);
23 }
24 }
```

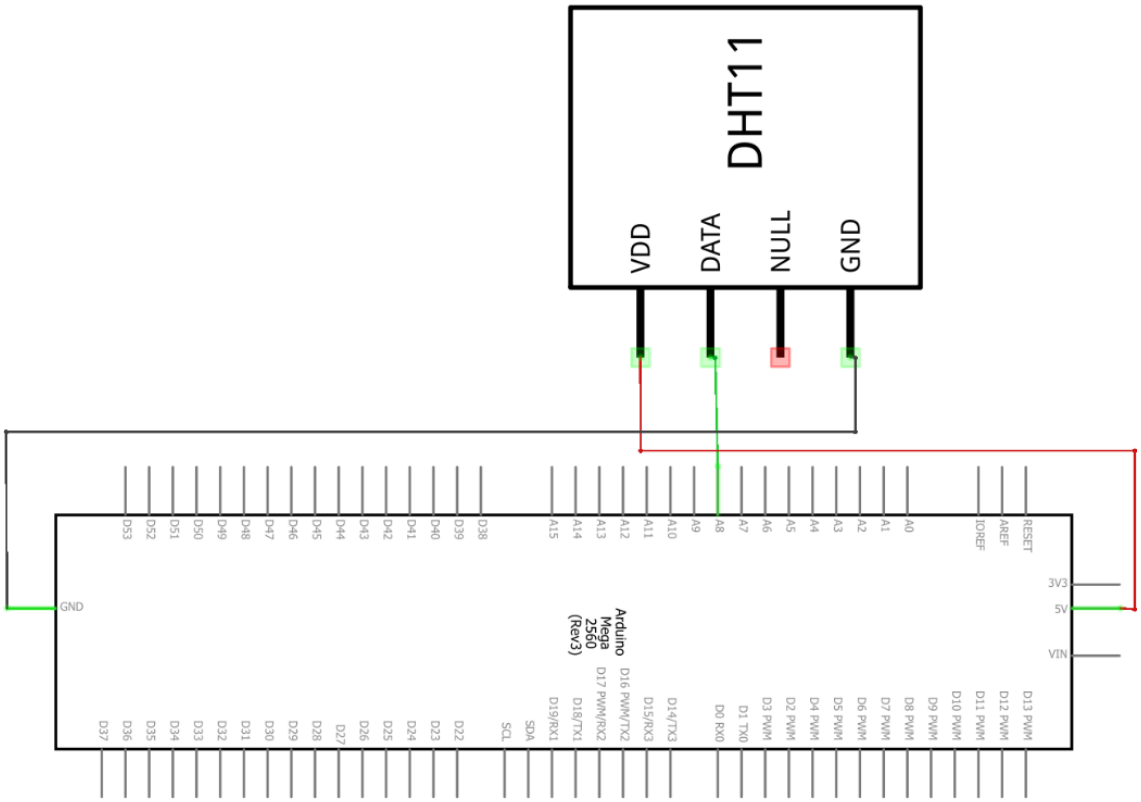
9.6 Měření teploty a vlhkosti

Základní popis

Okruh měření teploty a vlhkosti ve zvolené místnosti je realizován pomocí jednoho senzoru typu DHT11 od firmy Aosong Electronics, který podporuje obě tyto funkcionality současně. Stav teploty a vlhkosti je průběžně zaznamenáván v podobě dvou čísel do mobilního telefonu.

Obr. 30: Komponent měření teploty a vlhkosti [25]

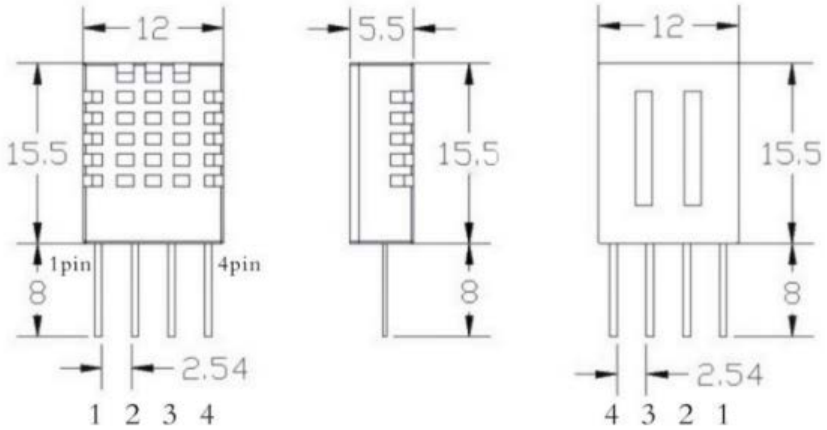
Schéma zapojení



Obr. 31: Schéma zapojení senzoru DHT11 [26]

Princip činnosti

Hodnota teploty a vlhkosti je analogově snímána pomocí senzoru DHT11. V mobilním telefonu jsou dva komponenty, které zobrazují stav teploty a vlhkosti v podobě číselných hodnot. Tyto hodnoty jsou průběžně aktualizovány. Pro tento typ měření je použitý analogový vstup na desce Arduino.



Obr. 32: Rozměry DHT11 [30]

Použité součástky a jejich specifikace

Tab. 10: Použité součástky měření teploty a vlhkosti [30]

SOUČÁSTKA	SPECIFIKACE	
DHT11	Napájecí napětí	3.3-5.5 V DC
	Provozní proud	0.3 mA
	Rozsah teplot	0-50 stupňů celsia
	Rozsah vlhkosti	20-90 %
	Přesnost měření teploty	+/- 1 stupeň celsia
	Přesnost měření vlhkosti	+/- 5 %

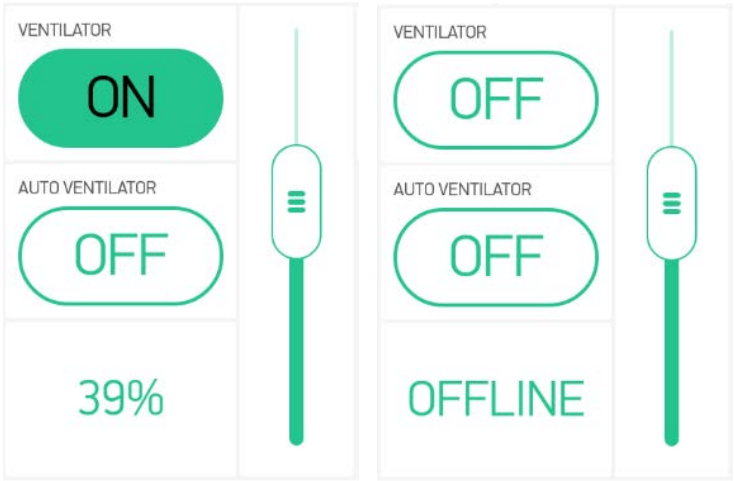
Ukázka kódu programu měření teploty a vlhkosti

```
1 void DHT_S()
2 {
3   humidity = dht.readHumidity();
4   temperature = dht.readTemperature();
5   Blynk.virtualWrite(V14, humidity);
6   Blynk.virtualWrite(V13, temperature);
7   TEMP = map(temperature, 25.0, 35.0, 120, 255);
8 }
```

9.7 Ovládání ventilátoru

Základní popis

Ventilátor je ovládán z mobilního telefonu. V telefonu jsou dva komponenty – spínač pro spouštění ventilátoru a posuvník pro regulaci otáček. Ventilátor lze zapnout / vypnout při pevném nastavení otáček, nebo lze tyto otáčky průběžně měnit posuvníkem.

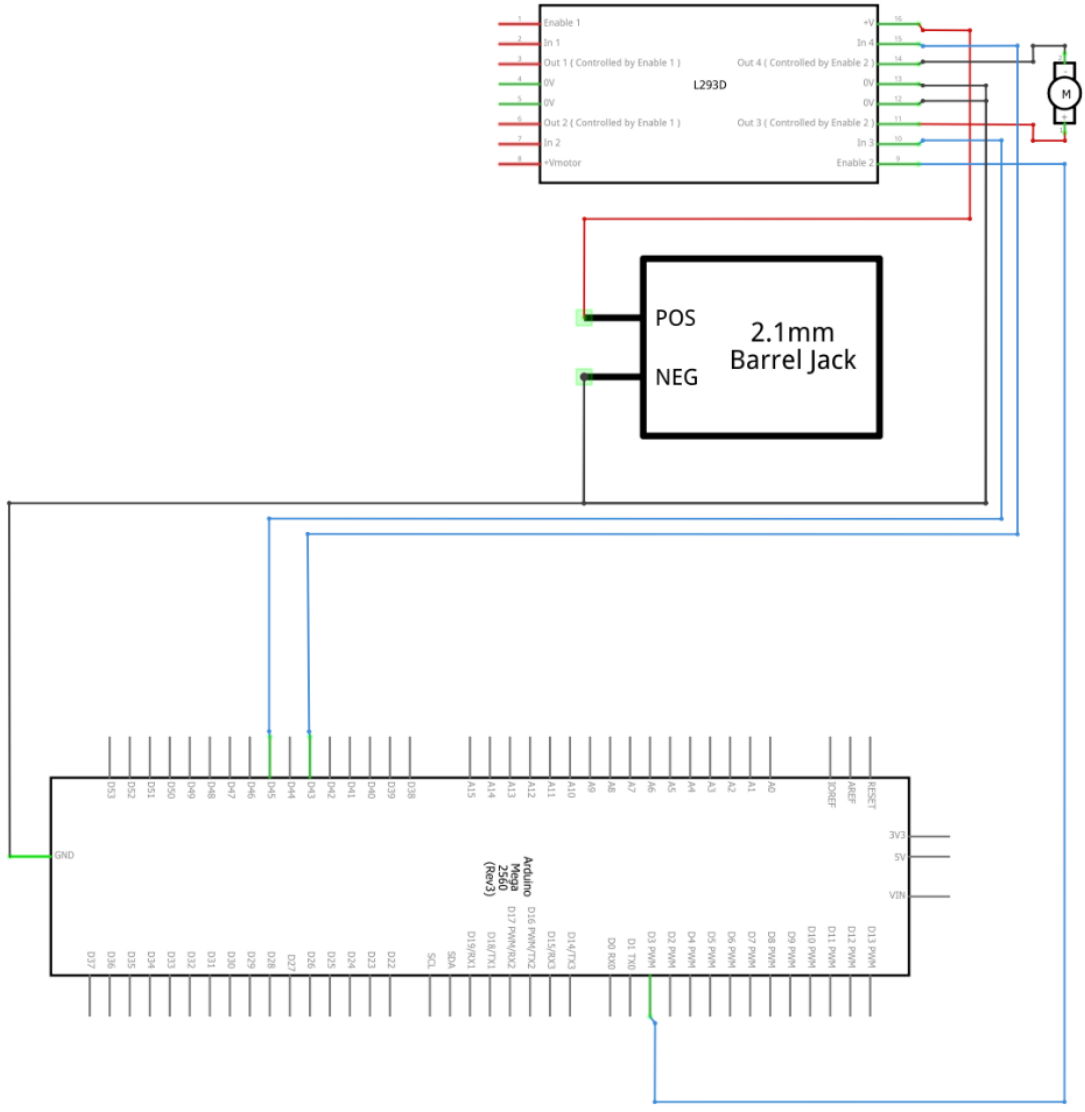


Obr. 33: Komponenty ovládání ventilátoru [25]

Princip činnosti

DC motor, kterým je ventilátor v této práci ovládán je možné zapnout / vypnout pomocí komponentu spínače ON / OFF. Při zapnutí DC motoru je možné plynule regulovat otáčky pomocí komponentu analogového posuvníku. Regulace otáček je realizována pomocí pulsní šířkové modulace na PWM výstupu Arduina, přepočtena a indikována v procentech v indikačním textovém poli. Pro ovládání motoru je použitý modul L239D. Směr otáčení je nastaven jedním směrem, ale v případě potřeby modul umožnuje směr otáčení měnit. Změna této funkcionality je provedena zásahem do funkce ventilátoru v kódu programu. Motor je nutné napájet externím napájením, princip viz kapitola 9.3, použité součástky a jejich specifikace.

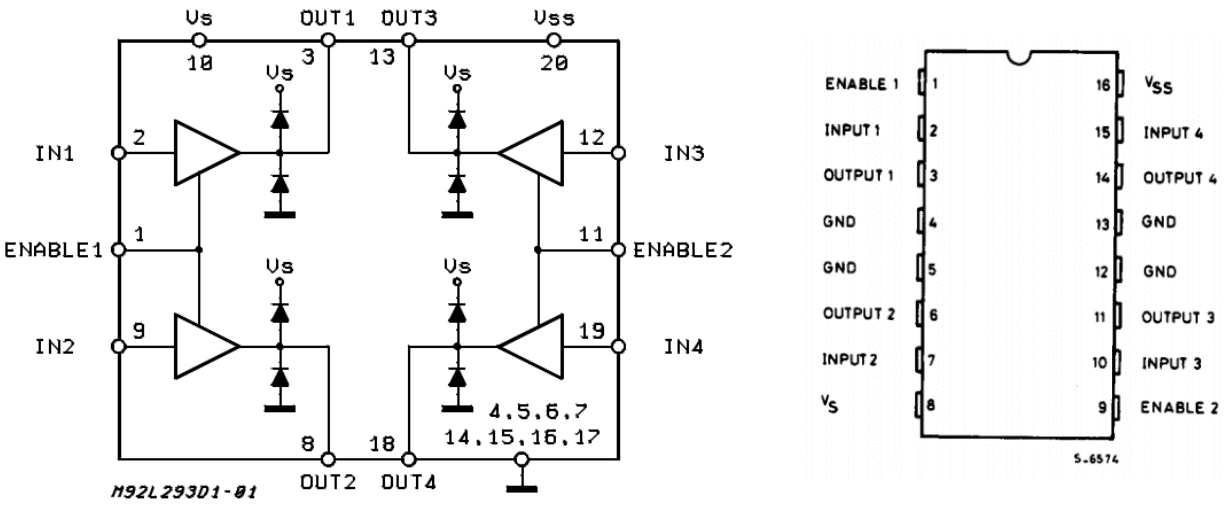
Schéma zapojení



Obr. 34: Schéma řízení DC motoru pomocí modulu L293D [26]

Použité součástky a jejich specifikace

Pro tento okruh je použitý malý DC motor. Napájení je zajištěno externě přes AC-DC adaptér, který napájí modul L293D.



Obr. 35: Schéma ovládacího modulu L293D [29]

Tab. 11: Použité součástky ovládání ventilátoru [29]

SOUČÁSTKA	SPECIFIKACE	
DC Motor RE-260RA	Provozní napětí	6 V
	Proud naprázdno	0.3-0,4 A
	Max. otáčky	20800 RPM
	Průměr	20 mm
	Délka	27 mm
	Průměr osy	2 mm
L293D	Vstupní napětí	4.5–25 V DC
	Max. proud	600 mA
	Ochrana proti přetížení	Ano
	Max. kmitočet modulace	5 kHz

Ukázka kódu programu ovládání ventilátoru

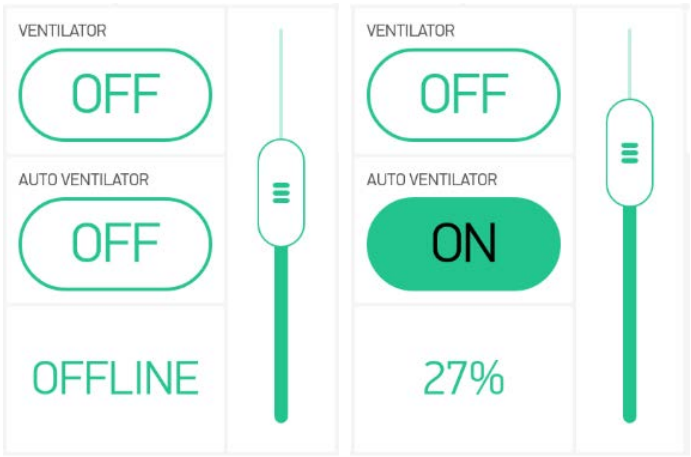
```
BLYNK_WRITE(V19)
{
  AUTOVENT = param.asInt();
}
BLYNK_WRITE(V9)
{
  DC_Motor_Virtual_Slider = param.asInt();
}
BLYNK_WRITE(V8)
{
  DC_Motor_Virtual = param.asInt();
}
void DC_Motor()
{
```

```
15   if (DC_Motor_Virtual == HIGH)
16   {
17   AUTOVENT = LOW;
18   analogWrite(SpeedPin, DC_Motor_Virtual_Slider);
19   Blynk.virtualWrite(V8, HIGH);
20   Blynk.virtualWrite(V19, LOW);
21   RPM = map(DC_Motor_Virtual_Slider,0,255,0,100);
22   Blynk.virtualWrite(V22, RPM,"%");
23   }
24   else if (DC_Motor_Virtual == LOW && AUTOVENT == LOW)
25   {
26   analogWrite(SpeedPin, 0);
27   Blynk.virtualWrite(V8, 0);
28   Blynk.virtualWrite(V22, "OFFLINE");
29   int T1 = timer.setInterval(100, STOP_RESET_MOVE);
30   timer.enable(T1);
31   }
32 }
```

9.8 Samočinné řízení ventilátoru podle teploty v místnosti

Základní popis

Automatický režim ventilátoru je aktivován virtuálním spínačem v mobilním telefonu ON / OFF. Při zapnutém režimu je monitorována aktuální teplota v místnosti a otáčky motoru jsou proporcionálně přizpůsobeny podle hodnoty teploty.



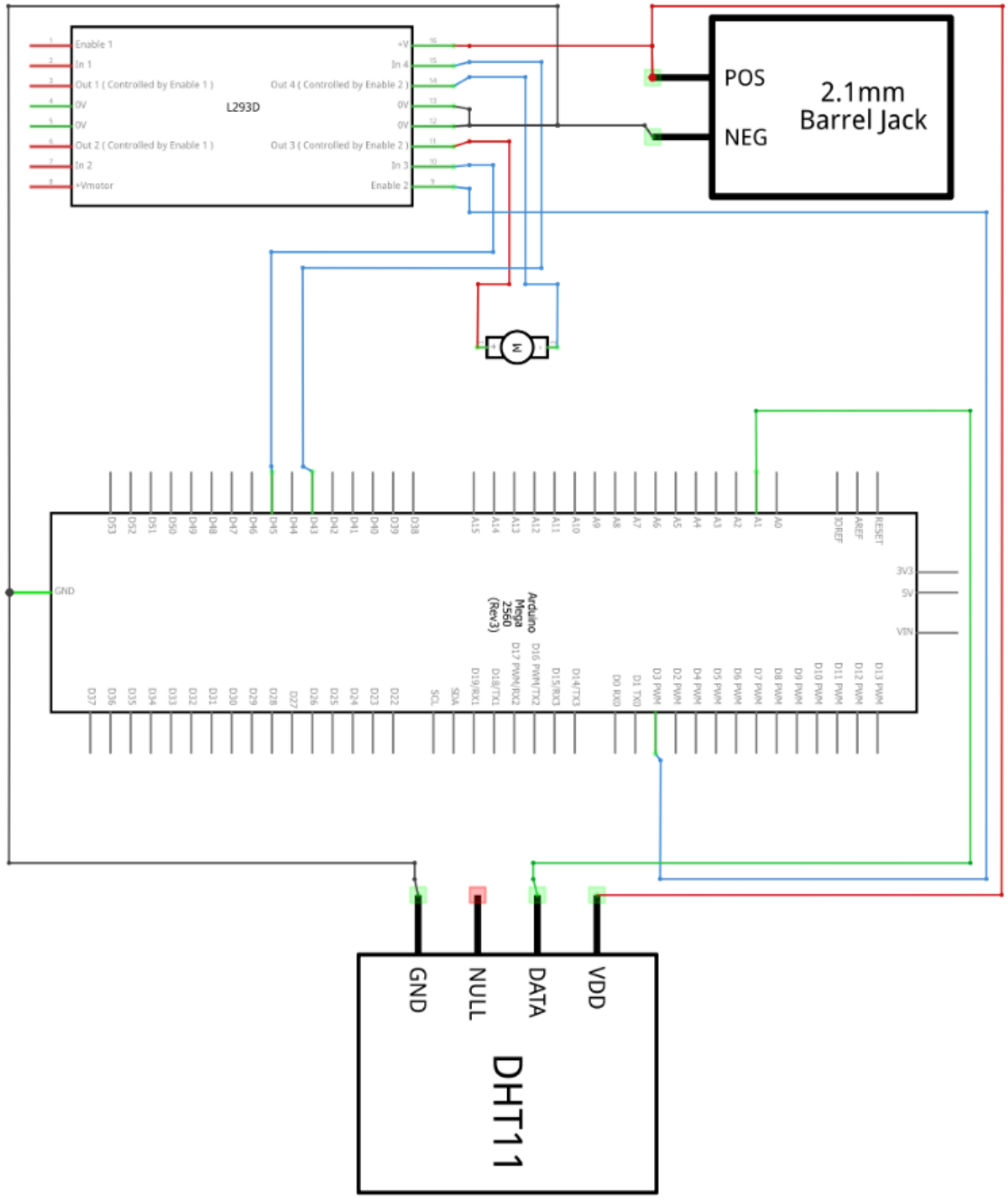
Obr. 36: Komponent ovládání automatického režimu ventilátoru [25]

Princip činnosti

Pokud je teplota vyšší nebo rovna 25 stupňům celsia, ventilátor se začne točit na minimální otáčky stanovené programem. DC motor je řízen pulsní šířkovou modulací. Jeden puls je přibližně 81.5 ot. / min. Rozmezí otáček ventilátoru je mapováno programem v šířce pulsů 0–255 / perioda. Z důvodu, že se motor není schopen rozběhnout

pod hodnotou šířky pulsů menší než 70, bylo stanoveno pracovní rozmezí jeho otáček pulsů 70-255. Přepočtem na otáčky motoru je to 5705–20800 ot. / min. k poměru rozmezí teplot 25-35 stupňů celsia. Při 25 stupních celsia se tedy otáčí 5705 ot. / min. až do 35 stupňů celsia, kdy má maximální otáčky, tj. 20800 ot. / min.

Schéma zapojení



Obr. 37: Schéma řízení DC motoru pomocí modulu L293D a senzoru DHT11 [26, 30]

Použité součástky a jejich specifikace

Použité součástky viz kapitola 9.6 a 9.7, použité součástky a jejich specifikace.

Ukázka kódu programu řízení ventilátoru podle teploty

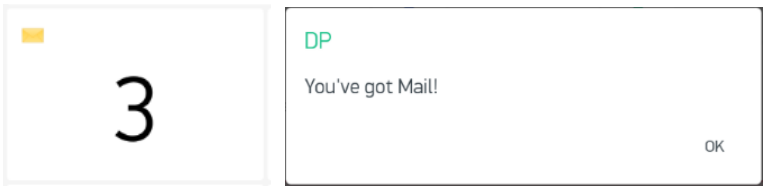
```
BLYNK_WRITE(V19)
{
  AUTOVENT = param.asInt();
}
BLYNK_WRITE(V9)
{
  DC_Motor_Virtual_Slider = param.asInt();
}
void AutoVentilator()
{
  if (AUTOVENT == HIGH)
  {
    if(TEMP >= 25.0)
    {
    DC_Motor_Virtual = LOW;
    analogWrite(SpeedPin, TEMP);
    RPM = (TEMP/255)*100;
    Blynk.virtualWrite(V22, RPM,"%");
    Blynk.virtualWrite(V9, TEMP);
    Blynk.virtualWrite(V19, HIGH);
    Blynk.virtualWrite(V8, LOW);
    }
  }
}

void DHT_S()
{
  humidity = dht.readHumidity();
  temperature = dht.readTemperature();
  Blynk.virtualWrite(V14, humidity);
  Blynk.virtualWrite(V13, temperature);
  TEMP = map(temperature, 25.0, 35.0, 70, 255);
}
```

9.9 Chytrá poštovní schránka

Základní popis

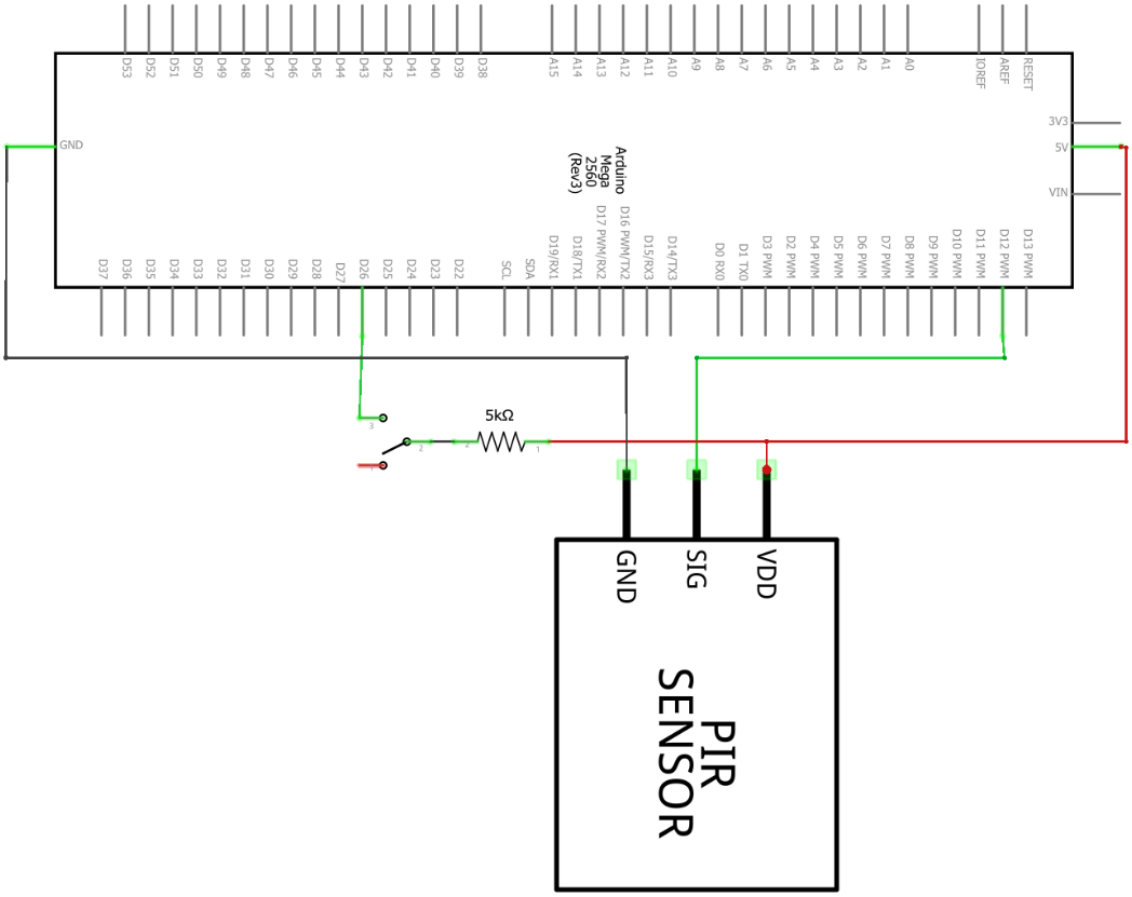
Chytrá poštovní schránka je okruh, který i přes svou jednoduchost byl zhotoven až po kompletním seznámení s možnostmi využití desky Arduino a AB v mobilním telefonu. Účelem chytré poštovní schránky bylo především vytvořit inovativní a praktický nápad k tématu této práce. Princip funkce chytré poštovní schránky je podávat

informaci uživateli domu o příchozích dopisech do mobilního telefonu, bez nutnosti frekventované kontroly poštovní schránky.



Obr. 38: Komponenty chytré poštovní schránky [25]

Schéma zapojení



Obr. 39: Schéma zapojení HC – SR501 [26]

Princip činnosti

Do poštovní schránky je umístěn sensor HC – SR501, který zajistí na digitální vstup Arduina logickou hodnotu 1 pokaždé vždy, když je zaznamenán v poštovní schránce pohyb. Každý nový příchozí dopis aktivuje notifikaci, která upozorní v telefonu, že přisel nový dopis. Algoritmus programu je napsán tak, aby každou změnu hodnoty pinu na logickou jedničku, kterou sensor HC–SR501 do Arduina pošle inkrementoval do proměnné, a tím je v této proměnné ukládán aktuální počet změn, tj. počet vložených objektů (dopisů) do poštovní schránky. Hodnota této proměnné je posílána

do komponentu číselného pole v AB, která čte a zobrazí hodnotu, tzn., že je v mobilním telefonu zobrazen aktuální počet dopisů. Ve schránce se nachází mikrospínač, který sepne při otevření dvířek poštovní schránky a změní logickou hodnotu na pinu v desce Arduino. V takovém případě program zajistí, že je proměnná, která ukládá počet dopisů vynulována a sensor není aktivní, dokud nejsou dvířka schránky opětovně uzavřeny. Tím je zajištěno, že pokud jsou dopisy ze schránky vybrány, je počet dopisů 0 a zároveň je zajištěno, aby při vybírání schránky nebyla tato aktivita mylně zaznamenávána jako příchozí dopis. Pro správnou funkci je nutné vhodně kalibrovat citlivost senzoru HC – SR501 a nastavit optimálně zpoždění po prvním zaznamenaném pohybu, při kterém by například docházelo k inkrementaci dopisů takovou rychlosti, že by za dobu vkládání jednoho dopisu bylo inkrementováno více dopisů než jeden. Zároveň je tato vlastnost citlivosti senzoru také silně ovlivněna jeho okolním prostředím, kdy reaguje i na změnu světelných podmínek.

Použité součástky a jejich specifikace

Tab. 12: Použité součástky chytrá poštovní schránka [31]

SOUČÁSTKA	SPECIFIKACE	
HC – SR501	Provozní napětí	5-20 V DC
	Klidový proud	50uA
	Výstup	3.3 V HIGH / 0 V LOW
	Nastavitelné zpoždění	3 s–5 min
	Rozměry	32 mm*24 mm
	Uhel záběru	110 stupňů
	Provozní teplota	-15 - +70 stupňů
Mikrospínač		
Rezistor	10k Ohm	

Ukázka kódu programu poštovní schránky

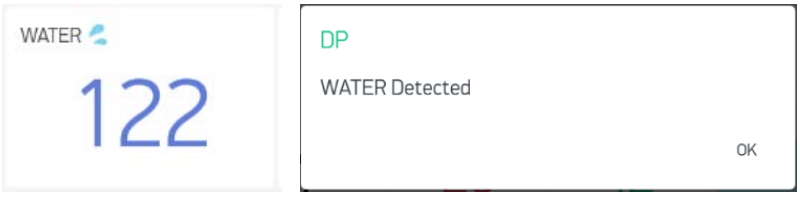
```
 1 void PIR()
 2 {
 3 int pirValue = digitalRead(pirPin);
 4   if (pirValue == HIGH)
 5     {
 6     Blynk.notify("You've got Mail!");
 7     Blynk.virtualWrite(V2, eventCount);
 8     Millis2();
 9     eventCount++;
10     }
11 }
12 void MAIL_RESET()
13 {
14 MailReset = digitalRead(MailReset_Button);
```

```
15 if (MailReset == HIGH)
16   {
17   Blynk.virtualWrite(V2, 0);
18   eventCount = 0;
19   }
20 }
```

9.10 Detekce přítomnosti vody

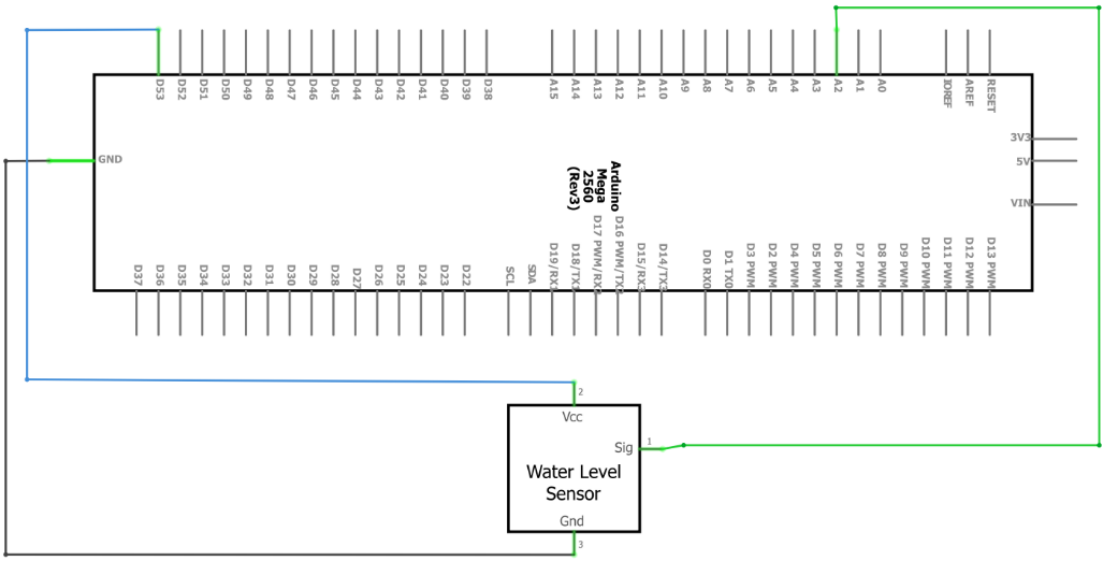
Základní popis

Detekce přítomnosti vody v různých částech domu, zejména tam, kde jsou přípojky vody ke spotřebičům je velmi užitečná. Včasné odhalení uniku vody pomůže řešit drobné vady, např. poškozené těsnění ventilů a různých spojů a později může zabránit i větším škodám na majetku. V případě typů senzorů, které zajišťují bezpečnost objektu je velmi důležité být o změnách stavu informován notifikací přímo do mobilního telefonu.



Obr. 40: Komponenty detekce vody [25]

Schéma zapojení



Obr. 41: Schéma zapojení senzoru HW-038 [26]

Princip činnosti

Do rizikového prostoru je umístěn modul dešťového senzoru HW-038, který pracuje s hodnotami rezistivity mezi vodiči umístěnými na malé ploše senzoru. Pokud je na ploše

senzoru i malé množství vody je rezistivita mezi vodiči senzoru snížena. Pokud dojde k překročení programem nastavené hodnoty rezistivity je v telefonu vyobrazena notifikace se zprávou o této skutečnosti. V telefonu je také pomocná informace v číselné podobě o aktuální hodnotě rezistivity. Nutnost snímat detekci vody byla stanovena v půl minutových intervalech. V každém intervalu je senzor aktivován napájecím pinem a je změřena aktuální hodnota. Jakmile je měření ukončeno, je napájení senzoru do začátku dalšího intervalu deaktivováno.

Provozní teplota senzoru HW-038 je značně omezena. Na základě hodnot stanovené výrobcem je použití vhodné především ve vnitřních prostorách domu, tak jak bylo aplikováno v této práci.

Použité součástky a jejich specifikace

Tab. 13: Dešťový senzor HW-038 [11]

SOUČÁSTKA	SPECIFIKACE	
HW-038	Provozní napětí	3.3~5 V DC
	Klidový proud	20 mA
	Výstup	3.3 V HIGH / 0 V LOW
	Rozpětí vlhkosti	10 % ~ 90 %
	Rozměry	65 mm×20 mm×8 mm
	Výstupní signál	0~4.2 V
	Provozní teplota	10℃~30℃

Ukázka kódu programu detekce přítomnosti vody

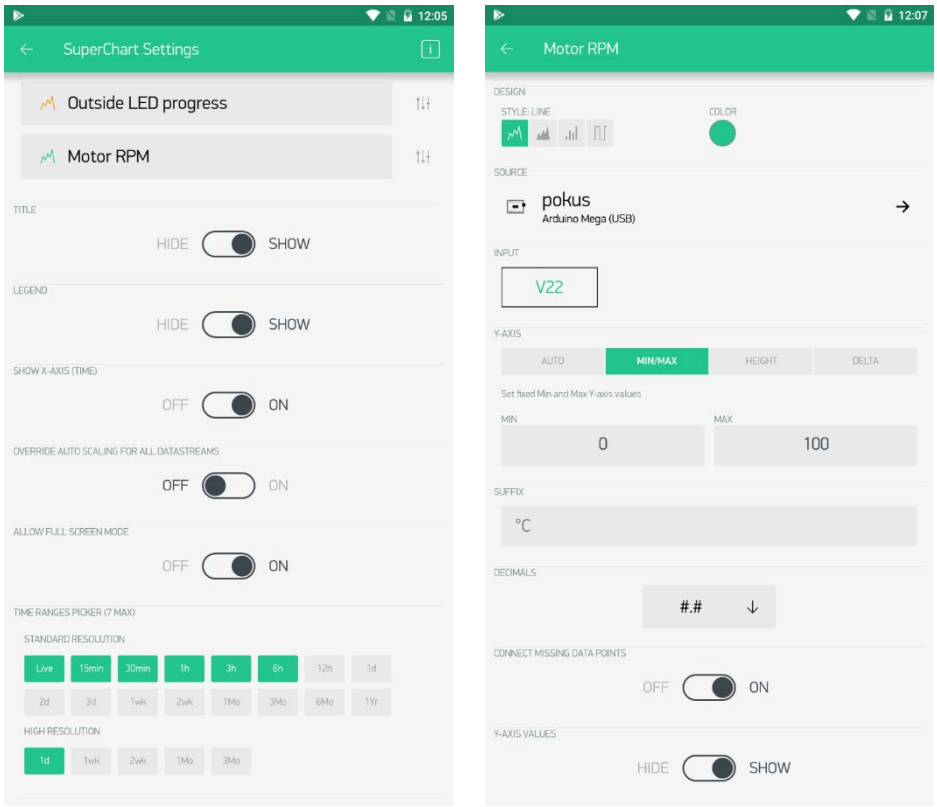
```
 1 void WATER()
 2 {
 3 digitalWrite(waterPOWER, HIGH);
 4 int waterREAD = analogRead(waterDATA);
 5 if (waterREAD < 15)
 6 {
 7   waterREAD = 0;
 8 }
 9 Blynk.virtualWrite(V20, waterREAD);
10 digitalWrite(waterPOWER, LOW);
11     if (waterREAD > 120 && WATERSTATUS == true)
12     {
13       Blynk.notify("WATER Detected");
14       WATERSTATUS = false;
15     }
16 }
```

9.11 Možnosti a komponenty sledování průběhů zvolených okruhů v čase

Základní popis ovládání grafu a funkcí

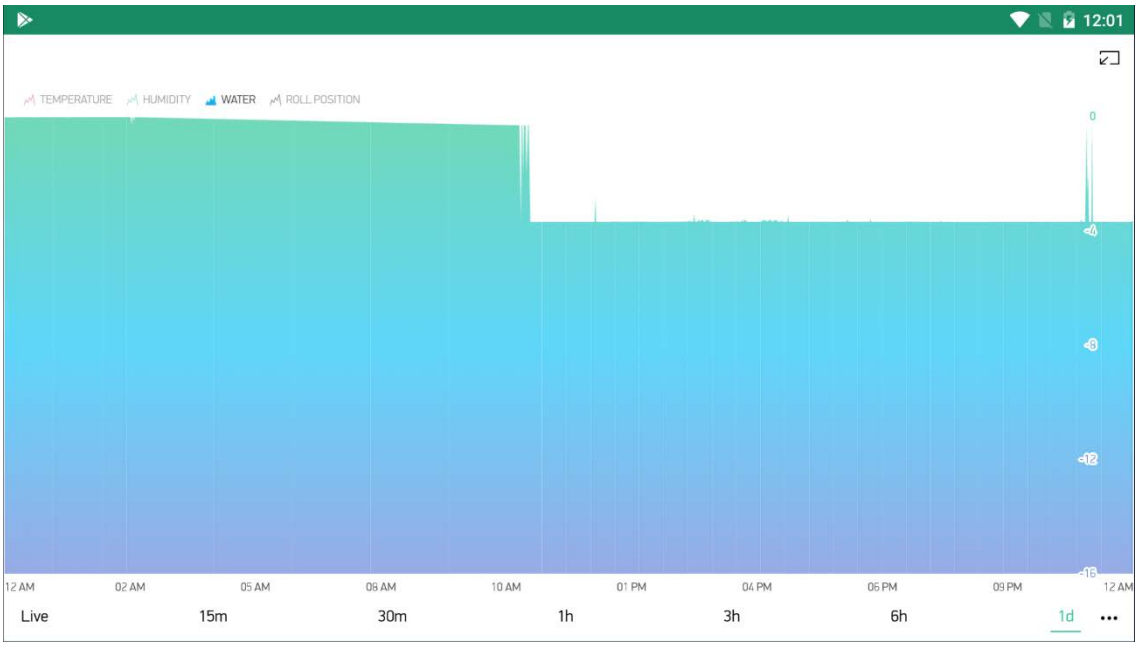
Aplikace Blynk umožnuje aplikovat virtuální komponent, který zaznamenává průběhy zvolených okruhů ve zvoleném čase. V této práci je virtuální komponent nastaven pro průběh teplot, vlhkosti, objemu vody v nádrži a spínání fotorezistoru. Nastavení virtuálního grafu je možné kdykoliv změnit. Všechny průběhy je možné exportovat do CSV formátu a dále použít pro další kalkulace. Průběhy jsou velmi užitečné například při analýze změn objemu vody v retenční nádrži v čase. Z těchto dat je následně možné odhadnout výdej vody pro zahradu oproti plnění nádrže z deště. Lze také počítat např. změny teploty v určitou dobu během dne a přizpůsobit tomu nastavení vytápění. Rozlišení jednotek času nastavené na grafech je pro optimální monitorování možné v aplikaci průběžně měnit. Nastavit lze také celkový vzhled grafů na tmavé nebo světlé pozadí a barvy jednotlivých průběhů. [25]

Vizuální možnosti Grafů průběhů v AB



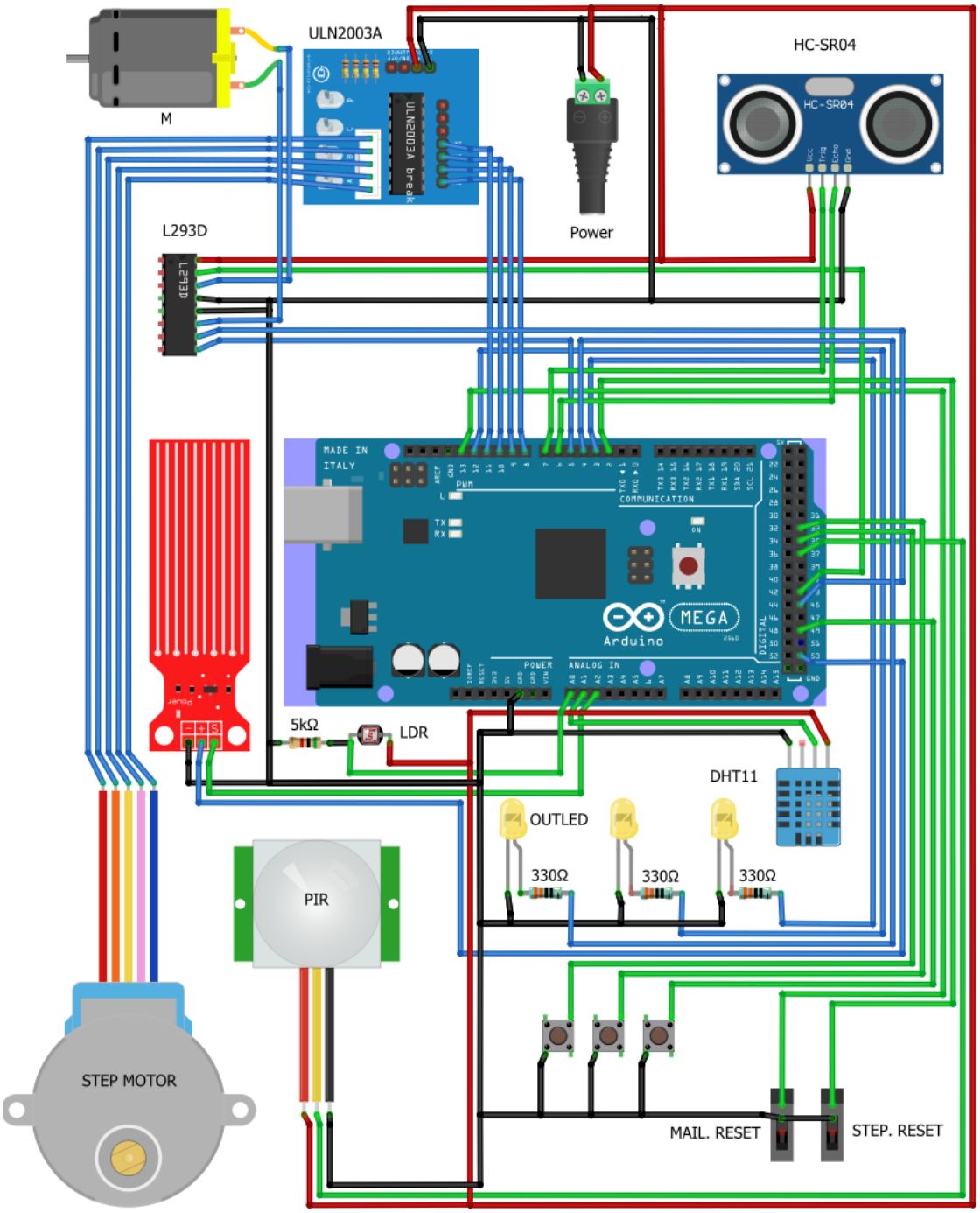
Obr. 42: Komponent graf průběhů nastavení [25]

Obr. 43: Komponent graf průběhů teploty a vlhkosti



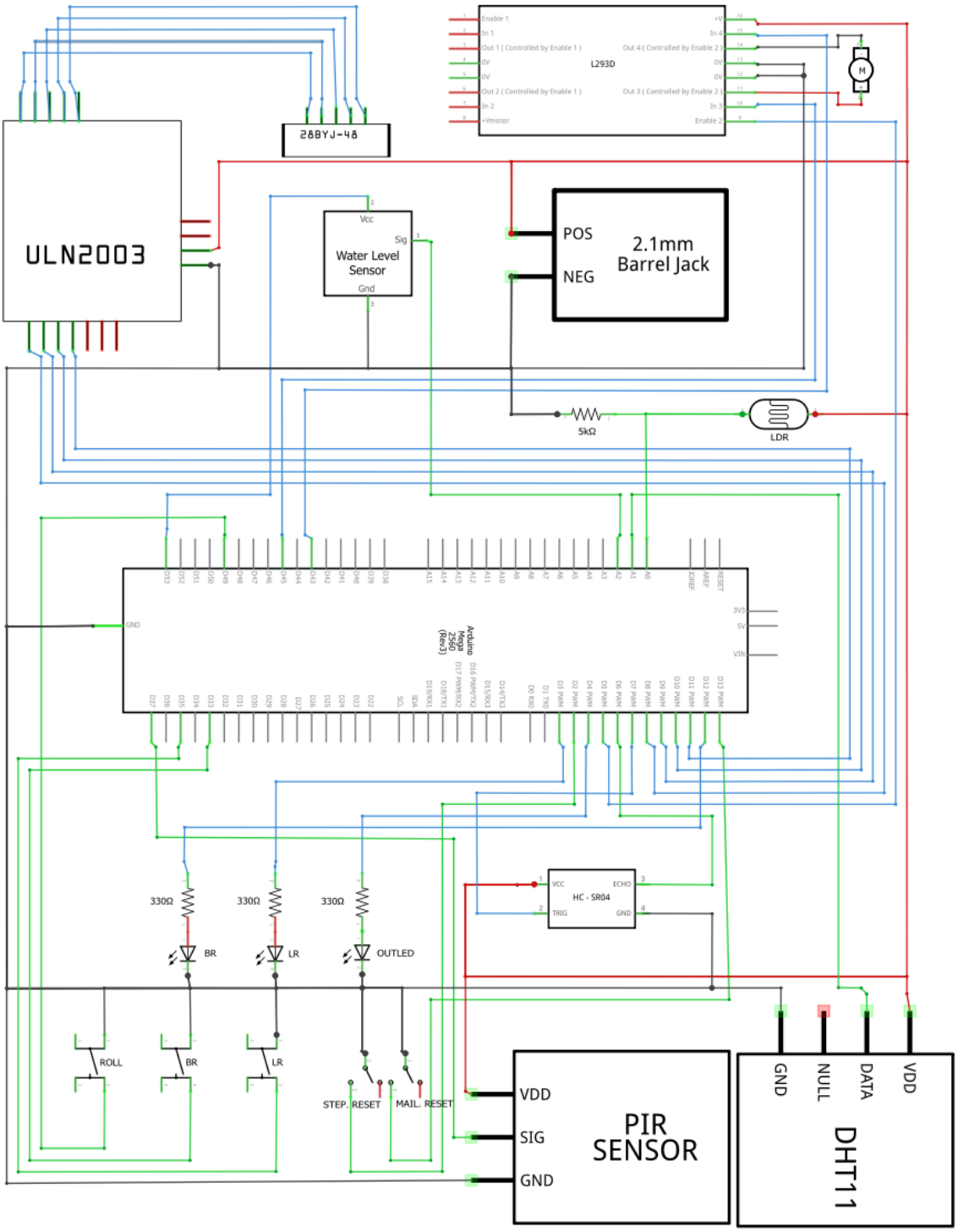
Obr. 44: Komponent graf průběhů objemu vody v nádrži.

9.12 Grafické znázornění zapojení celého projektu



Obr. 45: Grafické schéma zapojení celého projektu [26]

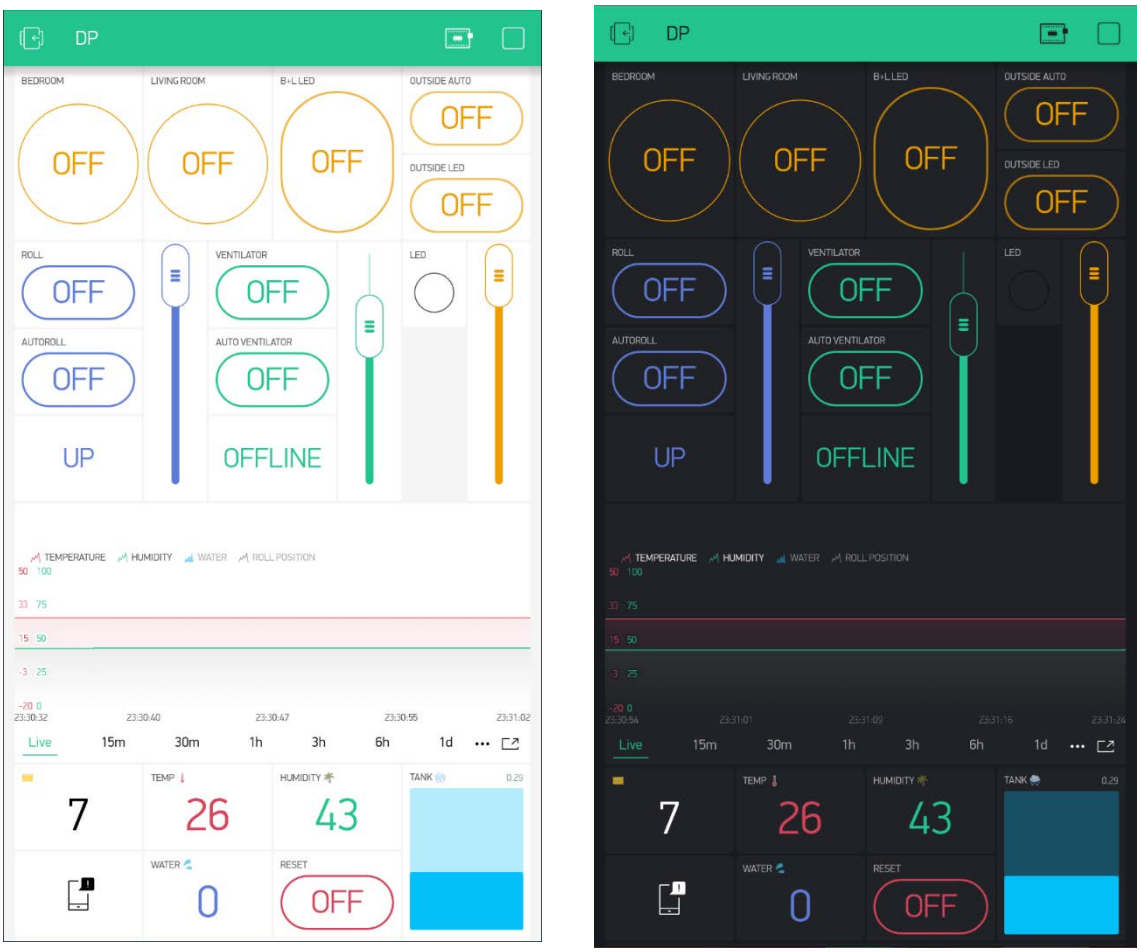
9.13 Blokové schéma zapojení celého projektu



Obr. 46: Blokové schéma zapojení celého projektu [26]

9.14 Vizualizace ovládacího menu v telefonu celého projektu

Ovládací menu AB v telefonu je možné barevně i graficky přizpůsobit potřebám uživatele. Barvy je možné nastavit na jednotlivých komponentech různé při sepnutých nebo rozepnutých stavech a také pozadí. Změny vzhledu komponentů mohou být dynamicky změněny voláním příkazu z programu při různých událostech. Každý uživatel může používat paralelně více různých projektů s různými komponenty a sdílet přistup s dalšími uživateli.



Obr. 47: Ovládání celého projektu v telefonu v různých barevných nastaveních [25]

Menu na obr. 47 není nutně ukončené souborem zde použitých komponentů, ale je možné přidávat další komponenty a posunovat celé menu posuvníkem dál směrem dolů. Komponenty a jejich stavy jsou aktualizovány instantně ve stejný moment na všech mobilních telefonech současně.

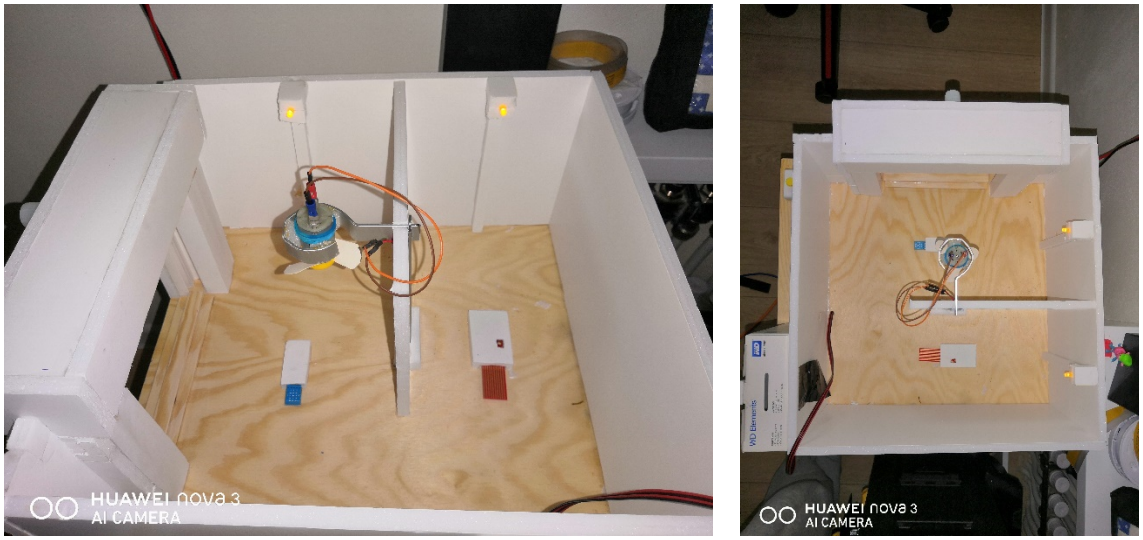
9.15 Praktická realizace pomocí modelu domu

Model domu je osazen všemi senzory popsaných v kapitolách vlastního řešení a vše je zapojené jako funkční celek. Dům je zhotoven v kombinaci s materiálem dřeva a kapa desek. Jednodeskový počítač Arduino je umístěn v kryté spodní části, kde je uzavřen a chráněn. Spodní část lze odklopit, aby byl umožněn přístup k Arduinu a jeho zapojení. Umístění senzorů je osazeno následovně.

Senzor teploty a vlhkosti DHT11 je umístěn na podlaze v přední části domu. Senzor detekce vody HW-038 je umístěn na podlaze v zadní části domu. Fotorezistor GL125 je umístěn na levé straně čelní stěny domu. Diody vnitřního osvětlení jsou umístěny na levé straně přední a zadní části domu a venkovní dioda na čelní stěně domu. Poštovní schránka je umístěna na pravé straně domu. Zkušební laboratorní válec pro ověření přesnosti měření hladiny nádrže je společně s ultrazvukovým senzorem HC-SR04 volně položen u domu. Fyzické spínače pro ovládání rolety a vnitřního osvětlení jsou umístěna na pravé straně domu. Ventilátor je umístěn v centrální části domu v hliníkovém kruhu na pěnových podpěrách, aby byla snížena hlučnost a vibrace.

Postup analýzy průběhů a chování systému

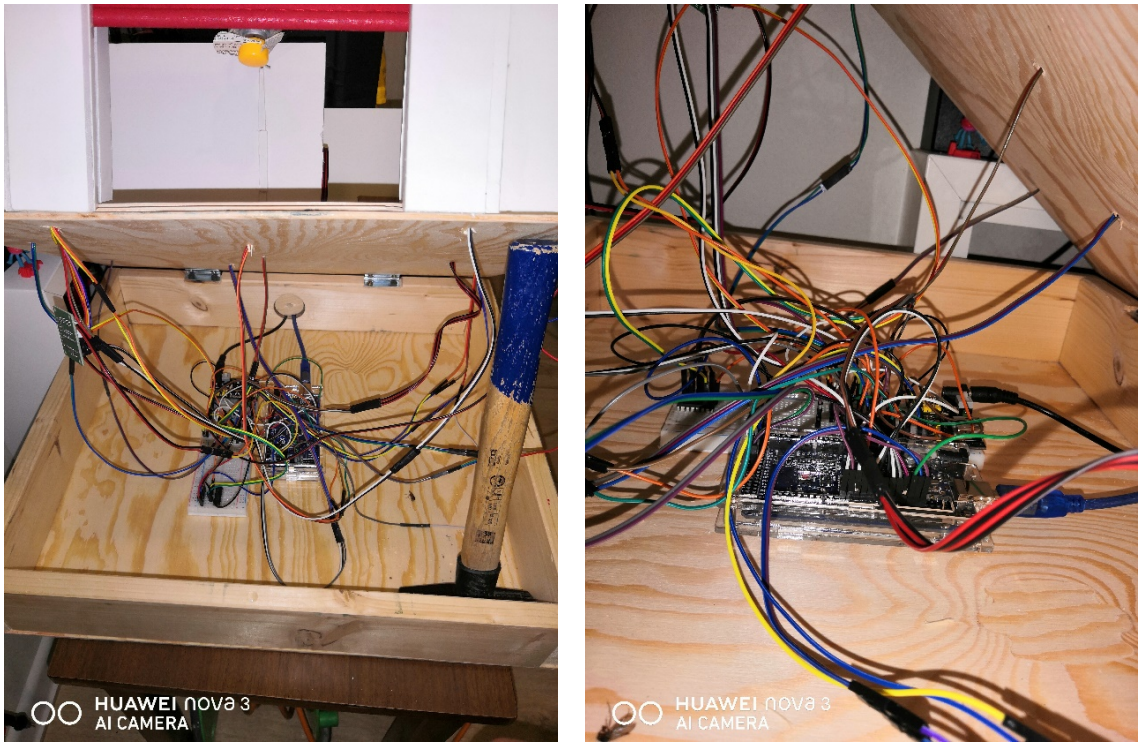
Dům byl umístěn do venkovního prostředí a vystaven reálným podmínkám. Během 24 hodin je analyzován celkový průběh chování okruhů automatického řízení rolety a venkovního osvětlení. Ostatní okruhy byly testovány jednorázově pro ověření jejich bezproblémové činnosti a funkčnosti. Grafy průběhů 24 h testu budou zaznamenány a následně vyhodnoceny. Všechny stavy jsou průběžně foceny a zdokumentovány.



Obr. 48: Model domu svislý pohled

Očekávaný průběh okruhu automatického řízení rolety je, že roleta bude při malé intenzitě světla, tj. ve večerních hodinách zcela vytažena nahoru. Při ranním rozbřesku bude roleta postupně stahována do polohy odpovídající intenzitě světla a zároveň musí

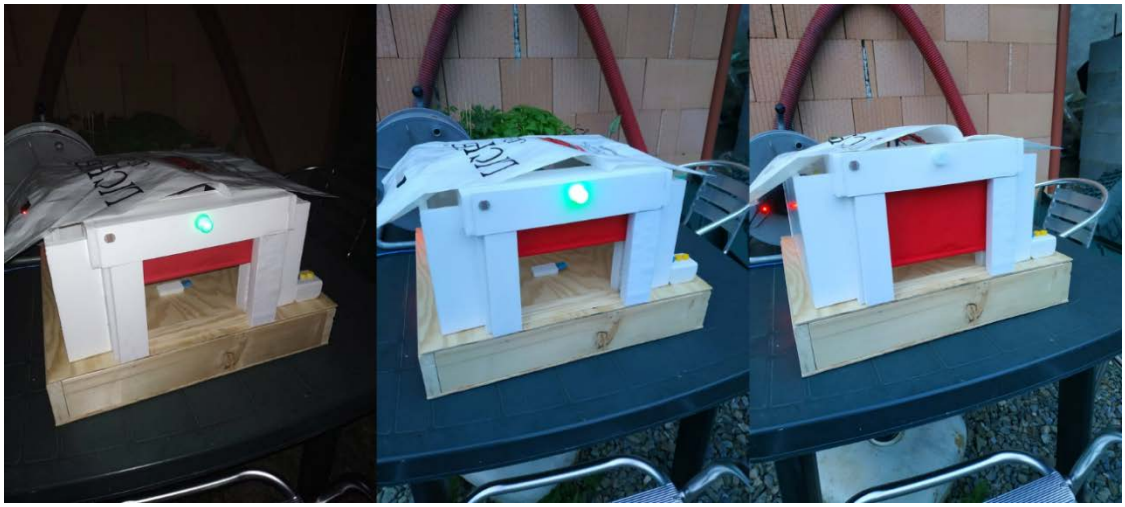
být splněna podmínka, že roleta zůstává vytažena nahoře, pokud je teplota v domě nižší, než 20° C.



Obr. 49: Vizualizace zapojení modelu domu

Obr. 50: Zelená dioda venkovního osvětlení v nočních hodinách

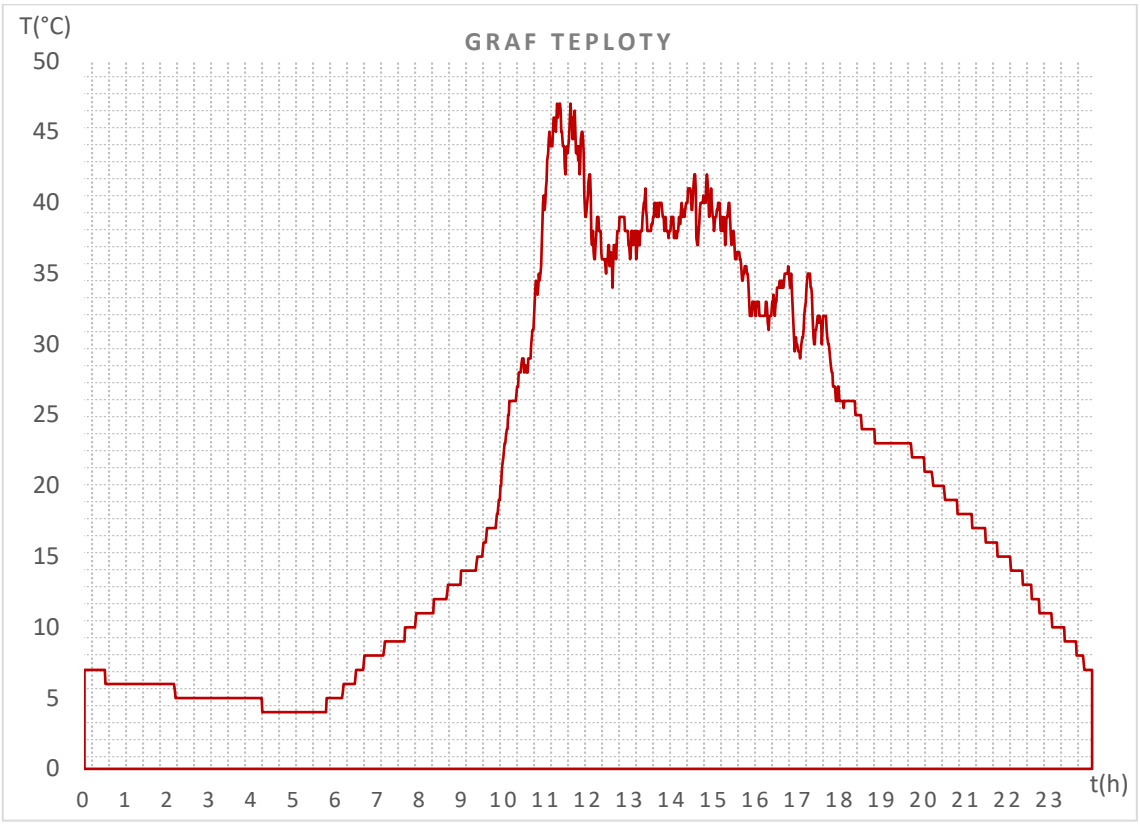
Obr. 51: Stažení rolety v průběhu testu v ranních hodinách



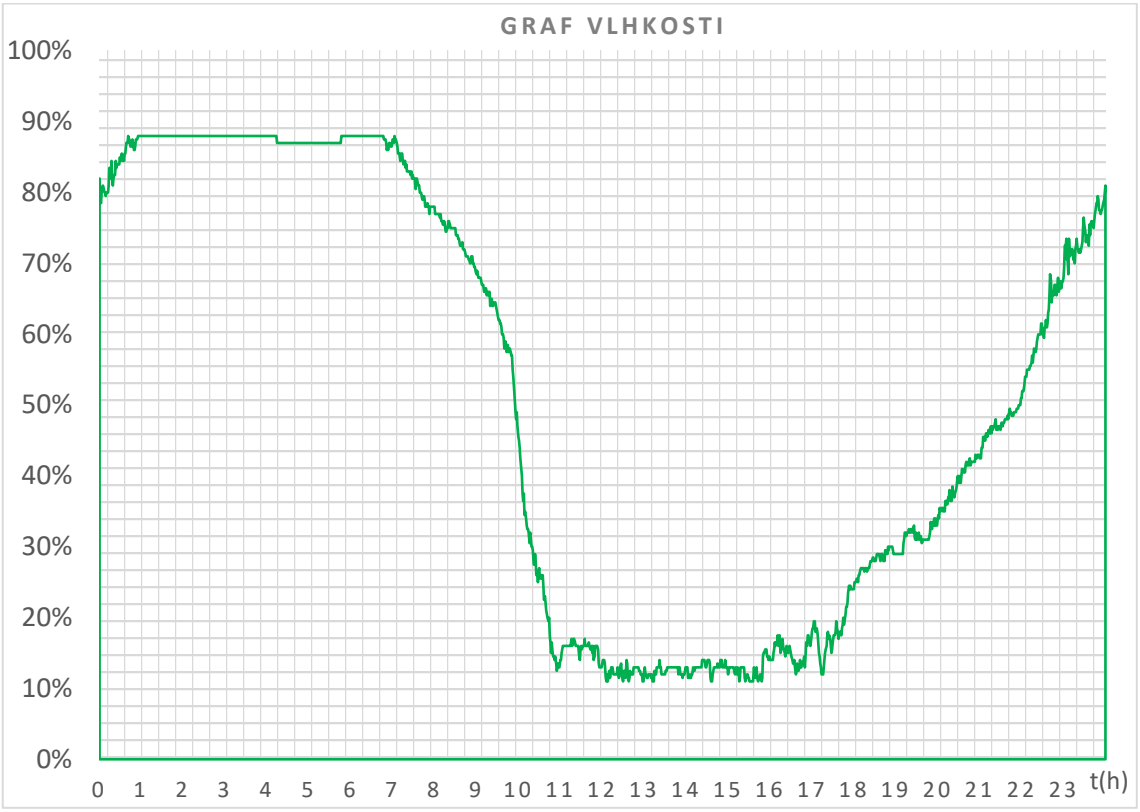
Obr. 52: Změna polohy rolety vpravo ve 20:00h, uprostřed ve 20:30h a vlevo ve 21:30h

Na konci testu nebylo možné měřit delší průběh kontinuálního vytahování rolety při klesající intenzitě světla z důvodu nízké teploty, která rychle klesla pod stanovenou hranici, kdy platí podmínka okamžitého vytažení rolety. Jednotlivé průběhy byly zaznamenávány do grafů v minutových intervalech, které odpovídají volání funkce fotorezistoru každých 60 sekund. Data byla exportována do CSV souboru a z nich byly vytvořeny grafy v aplikaci MS Office / Excel (https://www.microsoft.com/).

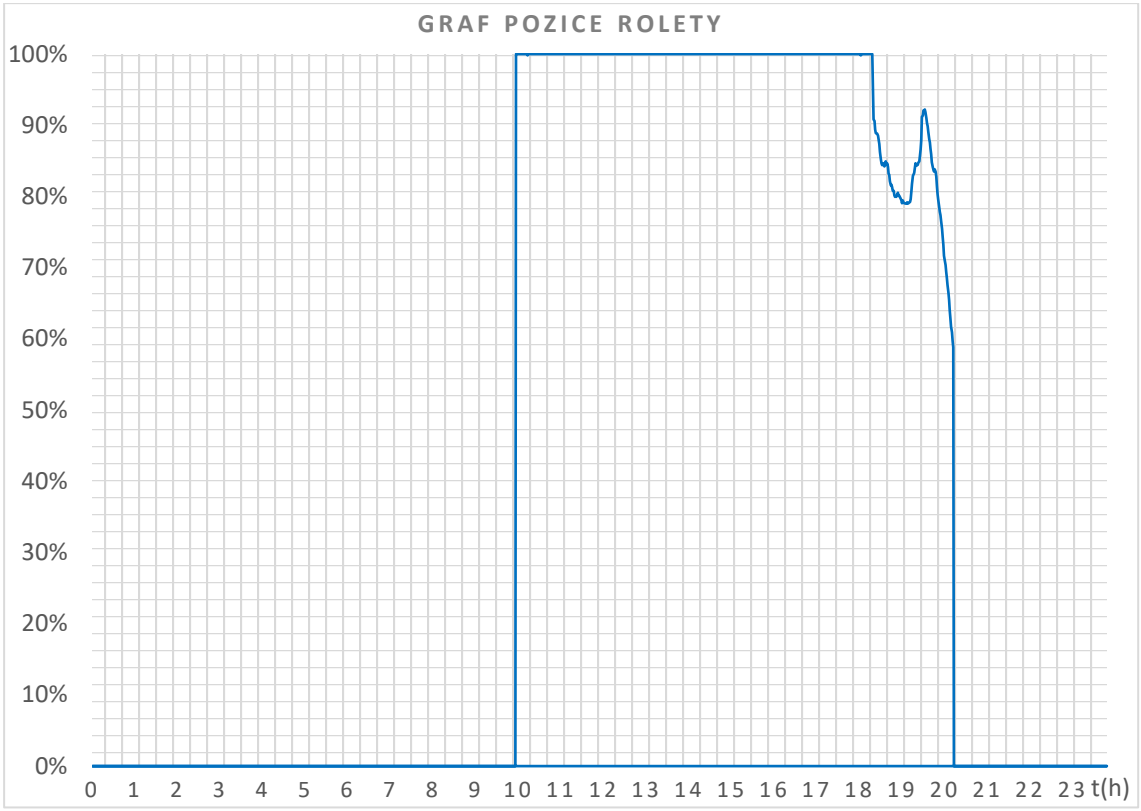
Grafy průběhů během 24 hodin



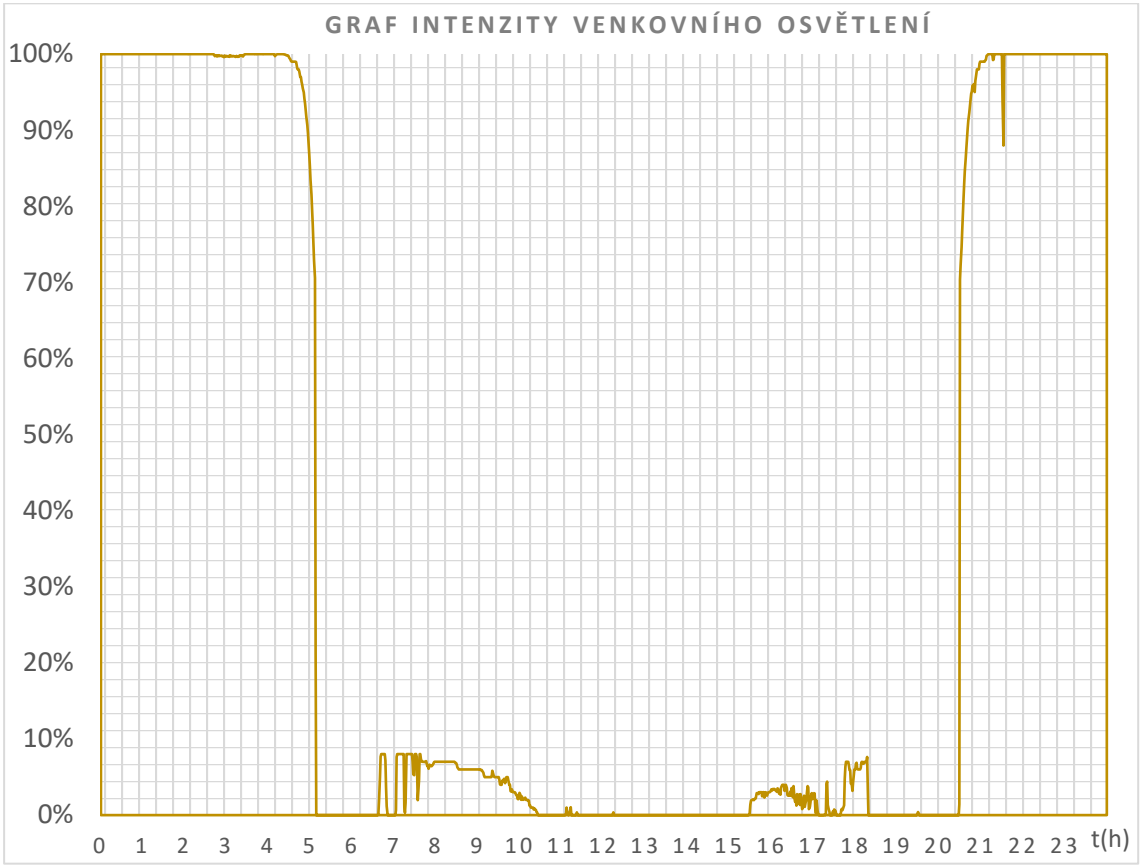
Obr. 53: Teplota v závislosti na čase



Obr. 54: Vlhkost v závislosti na čase



Obr. 55: Pozice rolety v závislosti na čase (100 % je roleta dole)



Obr. 56: Intenzita venkovního osvětlení v závislosti na čase

Vyhodnocení grafů a chování systému podle naměřených hodnot

Roleta byla aktivována v 0:00h viz obr. 55. Absence venkovního světla aktivovala okamžité vytažení rolety nahoru, tj. staženo 0 %. Následující den v ranních hodinách byla velká intenzita světla už po 7:00 hodině ráno, přesto z důvodu nízké teploty zůstává roleta do 9:50h vytažena nahoře. V čase 9:50h stoupla teplota nad stanovených 20 °C a roleta je plně stažena dolů na 100 %. Plné stažení bylo způsobeno přímým slunečním zářením na dům. Roleta zůstává stažena do 18:30h, a potom se postupně začíná vytahovat s klesající intenzitou venkovního světla, slunce je dočasně zastíněno mraky. V čase od 19:15h do 19:30h se objeví malý výkyv opětovného stahování rolety způsobené krátkodobé absence mraků. Od času 19:30h do 20:15h už slunce zapadá a roleta je postupně vytahována nahoru. V čase 20:15h teplota klesá pod stanovenou hranici a roleta je promptně plně vytažena nahoru.

Venkovní osvětlení je aktivováno v 0:00h viz obr. 56. Dioda je díky absenci světla promptně rozsvícena na 100 %. V čase od 4:25h do 5:10h při ranním rozbřesku intenzita svítivosti LED diody postupně klesá. V čase 5:10h se dostává odpor fotorezistoru pod hranici 700k Ohmů a dioda venkovního osvětlení je okamžitě vypnuta. V čase od 6:30h intenzita světla kolísá na hranici vypnutí a LED dioda je aktivována do 7 %. V 10:24h je venkovní osvětlení zcela deaktivováno, a kromě menších procentních výkyvů od 15:25h do 18:30h zůstává LED dioda ven. osvětlení vypnutá. V čase 20:30h je ven. os. opětovně sepnuto a opět strmě logaritmicky stoupá intenzita svítivosti LED diody do 21:25, kdy dosáhne 100 % intensity.

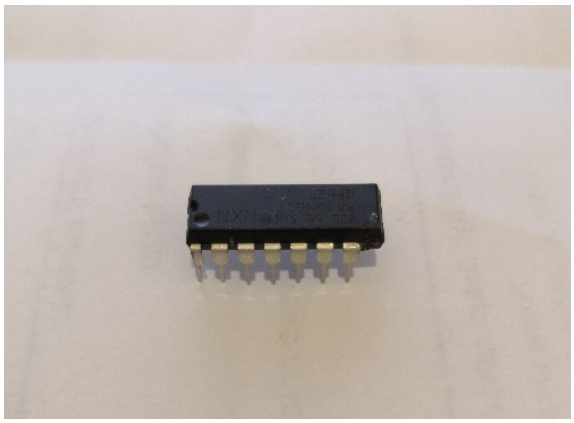
Teplota a vlhkost viz obr. 53 a obr. 54 je výrazně změněna v porovnání s nočními hodinami přímým slunečním zářením přímo na dům v časech od 10:00h ranní do 22:00h večerní. Z důvodu zakrytí domu plastovou folii, chránící před deštěm se teplota v denním režimu vyšplhala až na 47 °C a maximální vlhkost přes noc stoupla až na 85 %.

9.16 Poznatky z řešení, realizační problémy

Při práci s deskou Arduino, citlivými konektory a periferiemi může být možnou nevýhodou malá robustnost celého zařízení. Různé periferie a součástky použité v projektu jsou malé, křehké a vyžadují alespoň minimální zkušenosti a s elektronikou. V opačném případě může dojít k jejich snadnému poškození a znehodnocení. Konektory často končí ulámané nebo ohnuté, a to především mikročipy například typ L293D, který má měkké milimetrové nožičky. K rychlému znehodnocení součástek dochází také při překročeni dovoleného napětí, nezřídka některé periferie dovolují maximální vstupní napětí do 3.3 V a při překročení dojde k jejich okamžitému zničení.

Během přepínání stavů na pinech se objevil problém s nežádoucím zbytkovým napětím. Tento jev je v elektronice poměrně známý a způsobuje, že piny, které jsou v momentální nečinnosti indikují zbytkové napětí. Tato vlastnost může zapříčinit nežádoucí spínání, zejména při použití příkazů “RISING” a “FALLING”, které pracují se změnou logického stavu. Řešením je zajistit, aby piny, které jsou v momentální nečinnosti byly uzemněné.

Některé procedury přímo závislé na rychlosti cyklu mohou při použití s aplikací Blynk způsobovat obtížně řešitelné úkony. Při práci s krokovým motorem, bylo nutné použít příkazy, které motoru stanoví přerušení programu k dokončení úkonu. Příkladem může být příkaz runToPosition(), který zadá motoru finální počet kroků, program je přerušen dokud motor úkon nedokončí. Jsou případy, kdy by byl vhodnější příkaz run(), který je závislý na cyklu bez přerušení programu. Takový příklad může být při držení stisknutého tlačítka, při kterém je splněna podmínka pro proběhnutí cyklu funkce a dokud je tlačítko drženo, funkce je cyklována a motor se pohybuje stanoveným směrem. Tento typ funkce nebylo možné realizovat s aplikací Blynk, která způsobuje zpomalení při komunikaci s Arduinem a frekvence cyklu je nepravidelně zpožďována. Krokový motor zde vyžaduje minimální frekvenci cyklů 0.2ms, čehož nebylo možné dosáhnout.



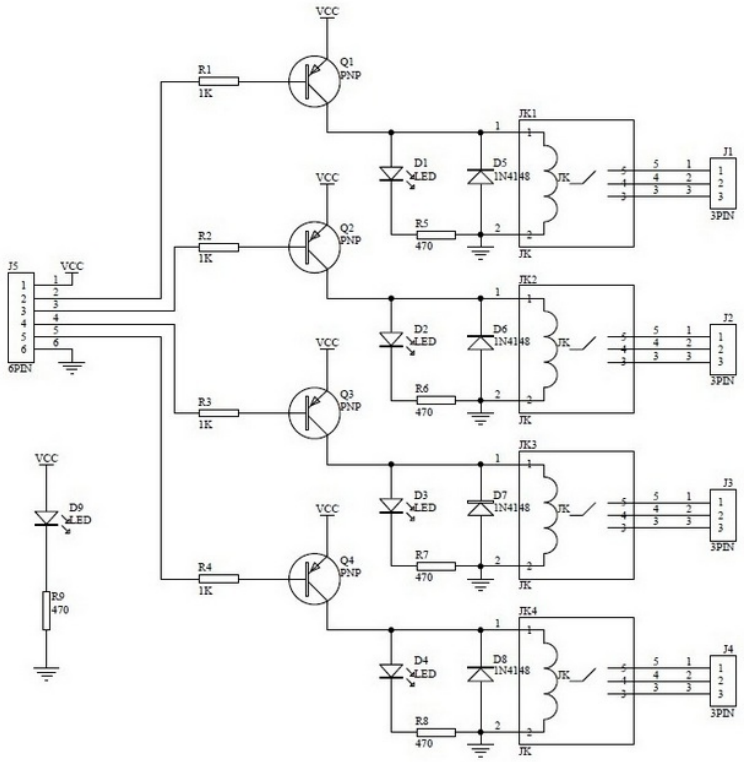
Obr. 57: Poškozený čip L293D

Fyzická tlačítka, která spínají různé okruhy přes Arduino často způsobují tzv. ”bounce” efekt. Spínaná plocha kontaktu spínače nemá ideální styk a způsobí díky velké rychlosti cyklu zakmitání hodnoty signálu. Existují různá řešení, jak tento problém řešit, softwarově i hardwarově. Při testování chování Arduina během této práce bylo úspěšně vyzkoušeno hardwarové řešení, které s pomocí kondenzátorů zlepší průběh náběhu hrany logické hodnoty. Realizováno a ponecháno bylo softwarové řešení, které pracuje se zpožděním cyklu funkce, spínač získá čas k sepnutí a zakmitání je odstraněno.

Architektura kódu programu byla během této práce několikrát zásadně změněna z důvodu složitosti při volání jednotlivých funkcí. Časovač “SimpleTimer“, který je doporučený pro použití s AB a umožnuje volat jednotlivé funkce ve zvoleném čase má limit všech volaných funkcí maximálně 16. Jelikož tento projekt vyžaduje mnohem více funkcí, bylo nutné vyhledat a instalovat další časovač, který umožní zbývající funkce volat. Struktura obou časovačů není stejná a zatím co “SimpleTimer“ čeká na dokončení příkazů volané funkce, než volá funkci další, ostatní běžné časovače tuto vlastnost nemají. Taková kombinace způsobí, že některé funkce nejsou exekučně dokončeny a čekají na další cyklus programu a takové zpoždění je nežádoucí. Tento problém nebyl optimálně vyřešen, pouze byl kód dodatečně napsán tak, aby každé funkci dal alespoň 40ms na dokončení v každém cyklu. [11]

9.17 Řízení okruhů nízkého napětí pomocí výkonových relé

V reálném prostředí jsou často důležitá řešení, která operují s provozním napětím vyšším, např. střídavé napětí do 250 V, proud do 10 A. Pro tyto účely jsou dostupné elektromagnetické tranzistorové reléové moduly. Tyto moduly jsou nabízené v různých variantách a v případě desek Arduino se jedná o moduly spínané pěti volty. Varianty těchto modulů jsou dále dostupné v počtech kanálů 1/2/4/6/8/16 a zároveň je možné využívat i další funkce, např. časovač se zpožděním. Pro sofistikovanější okruhy lze vícekanálová relé také spínat pomocí tranzistorového pole např. typ ULN2803APG.
[16, 23]



Obr. 58: Schéma čtyř kanálového reléového modulu [16]

Proud nutný k napájení jedné cívky relé modulů je 15-20 mA a vzhledem k tomu, že 200 mA je maximální dovolený proud pro celou desku Arduino, je vhodnější podobně jako např. při použití motorů, napájet relé moduly externě. Některá vícekanálová relé mimo jiné obsahuji optočlen, který zajistí elektrické oddělení kontaktů.

Reléové moduly jsou připojeny třemi vodiči, tj. napájení, uzemnění a komunikační pin, který slouží jako signál k sepnutí. Spínaní komunikačního pinu je dostupné i s obrácenou logikou, tzn. 0 = zapnuto / 1 = vypnuto. [16]

10 ZÁVĚR

Shrnutí všech částí jednodeskového počítače Arduino, především jeho mikrokontroleru, stanovilo v teoretické části předpoklad možností, kterých lze s touto deskou dosáhnout. Programování v jazycích C a C++ je optimální pro většinu projektů při práci s deskou Arduino. Další řádky budou věnovány závěrečné analýze jednotlivých okruhů a praktických postřehů při jejich realizaci.

Okruh ovládání dvou světel vnitřního osvětlení pomocí logiky klopného obvodu byl realizován úspěšně a splnil očekávání. Na tento okruh byly použité pouze dvě LED diody. V případě, že by LED diod bylo více, je z důvodu většího odběru proudu a následného zahřívání desky vhodnější použít externí napájení. Pokud by ovládání světel bylo realizováno pouze fyzickými spínači bez komunikace s mobilním zařízením, při kterém dochází ke zpoždění přenosu dat, byla by reakce světel výrazně rychlejší a pružnější.

Okruh samočinného řízení venkovního osvětlení se chová standardně. Dynamika a rychlost vyhodnocení intenzity světla splňují stanovený předpoklad.

Okruh samočinného řízení rolety byl realizován úspěšně. Velkou výhodou je přesnost krokového motoru. Nevýhodou řešení je silně nelineární chování fotorezistoru, které způsobuje skokové chování rolety. Tyto nelinearity byly částečně potlačeny diskrétní úpravou volání funkce programu v delších časových intervalech. Tento okruh byl z hlediska obtížnosti náročnější a výhodou jsou pokročilé praktické zkušenosti s krokovými motory. Vytvoření kódu a použití vhodných knihoven bylo pro jeho úspěšné řízení částečně obtížné.

Okruh samočinného bezkontaktního měření objemu vody v nádrži vznikl na základě skutečné potřeby vody v retenční nádrži u rodinného domu měřit. Bezkontaktní měření hladiny vody se jeví jako nejméně poruchové. Přesto, že při reálné aplikaci jsou senzory a systémy bezkontaktního měření vzdálenosti často velmi drahé, při použití Arduina bylo dosaženo dobrých výsledků při nízkých nákladech.

Okruh měření teploty a vlhkosti proběhl dle očekávání. Tento okruh byl realizován pro automatizaci dalších okruhů, např. řízení ventilátoru podle teploty.

Okruh ovládání ventilátoru byl i s pomocným ovládáním rychlosti otáček pomocí pulsní šířkové modulace úspěšně realizován. Krátké pulzy šířkové modulace někdy způsobují u DC motorů pískání. Realizace kódu programu ventilátoru nebyla obtížná, pouze kombinace analogového posuvníku a otáček vyžadovala více testů reakcí motoru.

Okruh samočinného ovládání ventilátoru, respektive jeho otáček podle teploty v místnosti, dosáhl uspokojivých výsledků. Tento okruh může mít i jiné využití, např. tam, kde je nutné spojité chlazení, a tím udržování stanovené teploty. Kód programu, který tento okruh řídí vyžaduje vhodné nastavení závislosti otáček na teplotě, kdy je nutné optimálně určit rozmezí teplot a otáček motoru.

Okruh chytré poštovní schránky splnil očekávání. Tento okruh skvěle demonstruje potenciál Arduina při méně obvyklých projektech, které však mohou být velmi užitečné.

Okruh detekce přítomnosti vody byl realizován úspěšně. Vzdálené upozornění na detekci nežádoucí vody je pro systémy bezpečnosti mandatorní. Identickým způsobem je možné aplikovat další nespočet senzoru typu detekce CO2, kouře a další.

Výhody použití jednodeskového počítače Arduino je nízkonákladovost, dostupnost periferií, dostupnost informací a variabilita použití.

Hlavní nevýhodou celého použitého řešení je poměrně pomalá komunikace mezi Arduinem a aplikací Blynk. Způsobuje zpomalení frekvence cyklu, a tudíž omezení celého výkonu desky Arduino. Pomalá komunikace je potencionálně řešitelná nastavením lokálního serveru, který aplikace Blynk poskytuje, takový způsob nastavení nebyl testován. Elegantním řešením by také mohlo být vytvoření vlastního webového serveru a prostředí, které by tuto aplikaci nahradilo, takové řešení ale vyžaduje pokročilé znalosti v této oblasti nad rámec této práce.

SEZNAM POUŽITÉ LITERATURY

[1] MATLAB & SIMULINK. Informace o produktech fy The Math Works. Dostupné z: www.humusoft.cz [online: 1.10.19], [cit. 2020-04-03]. 2011.

[2] ŠVARC, Ivan a MATOUŠEK Radomil, a ŠEDA Miloš a VÍTEČKOVÁ Miluše. Automatické řízení. Brno: CERM – Akademické nakladatelství, 2011. ISBN 978-80-2144398-3.

[3] GORDON, Oliver. IBS ELECTRONICS & SECURITY. How To Be A Home Automation Super Hero: Home Security Systems [online]. [cit. 2020-04-03]. Dostupné z: https://www.ibselectronics.net/how-to-be-a-home-automation-super-hero/

[4] ROUSE, Margaret. TECHTARGET. *Microcontroller* [online]. 2019 [cit. 2020-04-03]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/microcontroller

[5] MARSHALL, Brain. HOWSTUFFWORKS. *How Microcontrollers Work* [online]. [cit. 2020-04-03]. Dostupné z: https://electronics.howstuffworks.com/microcontroller.htm

[6] TUTORIALSPOINT. *Microcontrollers - 8051 Architecture* [online]. [cit. 2020-04-03]. Dostupné z: https://www.tutorialspoint.com/microprocessor/microcontrollers_8051_architecture.htm

[7] HRBÁČEK, Jiří. *Komunikace mikrokontroleru s okolím*. 1. vyd. Praha: BEN – technická literatura, 1999, 159 s. ISBN 80-86056-42-21. [cit. 2020-04-03]. Dostupné z: https://eluc.krolomoucky.cz/verejne/lekce/857

[8] GOYETTE, Richard. *Microchip AVR Hacking* [online]. [cit. 2020-04-03]. Dostupné z: http://www.richardgoyette.com/Projects/AVRPgming/AVRPgming.html

[9] ELECTRONICS HUB. *Microcontroller Types and Applications* [online]. 9 Srpen. 2015 [cit. 2020-04-03]. Dostupné z: https://www.electronicshub.org/microcontrollers/

[10] WORDPRESS. *Top Ten Microcontroller Manufacturing Companies across the Globe: Electronic components* [online]. 25 Září. 2013 [cit. 2020-04-03]. Dostupné z: https://www.electronicshub.org/microcontrollers/

[11] ARDUINO. [online]. [cit. 2020-04-03]. Dostupné z: https://www.arduino.cc

[12] MICROCHIP TECHNOLOGY. *ATMega* [online]. [cit. 2020-04-03]. Dostupné z: https://www.microchip.com

[13] GETTYIMAGES. *Home Automation: Obr. 1* [online]. [cit. 2020-04-03]. Dostupné z: https://www.gettyimages.com

[14] ESPRESSIF. *ESP8266* [online]. [cit. 2020-04-03]. Dostupné z: https://www.espressif.com/en/products/hardware/esp8266ex/overview

[15] HERTZ, Daniel. MAKERPRO. *HC-SR04 ultrasonic proximity sensor, highlighting its specs and common applications: HC-SR04 Datasheet* [online]. 2 Květen. 2019 [cit. 2020-04-03]. Dostupné z: https://maker.pro/custom/tutorial/hc-sr04-ultrasonic-proximity-sensor-datasheet-highlights

[16] HOBBYIST.CO.NZ. *Weaving the relay modules with the dancing lights: 4 Channel relay* [online]. [cit. 2020-04-03]. Dostupné z: https://www.hobbyist.co.nz/?q=music-plays-and-lights-dance

[17] ITEAD. HC-05 [online]. [cit. 2020-04-03]. Dostupné z: https://www.itead.cc/

[18] COMPONENTS101. *SIM900A: Module Datasheet* [online]. 8 Říjen. 2018 [cit. 2020-04-03]. Dostupné z: https://components101.com/wireless/sim900a-gsm-module

[19] NICERF. *G-NiceRF Bluetooth Module* [online]. [cit. 2020-04-03]. Dostupné z: https://www.nicerf.com/productslist_119_NiceRF%20.html

[20] ARDUINO SHOP. [online]. [cit. 2020-04-03]. Dostupné z: https://navody.arduino-shop.cz/

[21] RAFIQUZZAMAN, M. APOGEEWEB. *Microcontroller Applications and Its Principle* [online]. [cit. 2020-04-03]. Dostupné z: http://www.apogeeweb.net/article/58.html

[22] REMOTEXY. [online]. [cit. 2020-04-03]. Dostupné z: https://remotexy.com

[23] TOSHIBA CORPORATION. *ULN2803APG* [online]. [cit. 2020-04-03]. Dostupné z: https://www.toshiba.co.jp/worldwide/index.html

[24] LABCENTER. *Proteus* [online]. [cit. 2020-04-03]. Dostupné z: https://www.labcenter.com/

[25] BLYNK. [online]. [cit. 2020-04-03]. Dostupné z: https://blynk.io

[26] FRITZING. [online]. [cit. 2020-04-03]. Dostupné z: https://fritzing.org/home/

[27] DATASHEET PDF. *Photorezistor GL125: Datasheet* [online]. [cit. 2020-04-03]. Dostupné z: http://www.datasheet-pdf.com

[28] KIATRONICS. *Motor 28BYJ-48: Datasheet* [online]. [cit. 2020-04-03]. Dostupné z: https://www.kiatronics.co.nz/

[29] TEXAS INSTRUMENTS. *ULN2003: Datasheet* [online]. [cit. 2020-04-03]. Dostupné z: http://www.ti.com

[30] MOUSER ELECTRONICS. *DHT11: Datasheet* [online]. [cit. 2020-04-03]. Dostupné z: https://eu.mouser.com/

[31] EPITRAN. HC–SR501: Datasheet [online]. [cit. 2020-04-03]. Dostupné z: https://www.epitran.it/ebayDrive/datasheet/44.pdf

[32] TEICHER, Jordan. IBM INDUSTRIES. The little-known story of the first IoT device: Home Security Systems [online]. 7 Únor, 2018. [cit. 2020-04-03]. Dostupné z: https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/

SEZNAM PŘÍLOH

Program.zip