TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Výukový model chytrého domu s inteligentní elektroinstalací

Diplomová práce

Studijní program: N2301 – Strojní inženýrství
Studijní obor: 2301T049 – Výrobní systémy a procesy

Autor práce: **Bc. Jiří Koudelka**
Vedoucí práce: Ing. Radek Votrubec, Ph.D.

Liberec 2019

Technická univerzita v Liberci
Fakulta strojní
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jiří Koudelka**
Osobní číslo:	**S17000220**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Výrobní systémy a procesy**
Název tématu:	**Výukový model chytrého domu s inteligentní elektroinstalací**
Zadávající katedra:	**Katedra výrobních systémů a automatizace**

Zásady pro vypracování:

Navrhněte a realizujte řídicí systém výukového modelu chytrého domu s použitím mikroprocesoru Arduino, který by splňoval podmínky definované v následujících bodech:
1. Navrhněte model chytrého domu s inteligentní elektroinstalací (osvětlení, topení, alarm).
2. Pro realizaci řídicího systému použijte vhodnou desku Arduino.
3. Ovládání všech prvků umožněte lokálně manuálně a vzdáleně pomocí telefonu s Android nebo iOS. Ovládací aplikaci navrhněte s ohledem na komfort a nenáročnost uživatele.

Rozsah grafických prací: **podle potřeby**

Rozsah pracovní zprávy: **50**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Začínáme s Arduinem: příručka. In: Snail Instruments: www.hobbyrobot.cz [online]. 2014 [cit. 2015-01-09]. Dostupné z: http://www.snailshop.cz/literatura/1537-zaciname-s-arduinem-prirucka.html.
[2] Arduino Learning: Getting Started with Arduino. In: Arduino [online]. 2014 [cit. 2015-01-09]. Dostupné z: http://arduino.cc/en/Guide/HomePage
[3] BALÁTĚ, J. Automatické řízení. 1. vyd. Praha: BEN - technická literatura, 2003, 663 s. ISBN 978-80-247-4116-1.

Vedoucí diplomové práce: **Ing. Radek Votrubec, Ph.D.**
Katedra výrobních systémů a automatizace

Datum zadání diplomové práce: **15. listopadu 2018**

Termín odevzdání diplomové práce: **15. května 2020**

prof. Dr. Ing. Petr Lenfeld
děkan

L.S.

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 15. listopadu 2018

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

2. 5. 2019

Bc. Jiří Koudelka

Poděkování

Chtěl bych poděkovat všem, kteří mě podporovali při tvorbě diplomové práce. Především vedoucímu této práce panu Ing. Radku Votrubcovi, Ph.D. Rád bych také poděkoval své partnerce Kateřině Hálové a mé rodině za podporu během celého studia.

Abstrakt

Diplomová práce se zabývá návrhem a realizací řídicího systému výukového modelu chytrého domu s inteligentní elektroinstalací (osvětlení, topení, alarm). Pro realizaci řídicího systému je použita deska Arduino. Ovládání všech prvků je umožněno lokálně, nebo vzdáleně pomocí telefonu s operačním systémem Android. V úvodní části diplomové práce jsou vysvětleny pojmy chytrá domácnost a inteligentní elektroinstalace, následuje popis vývojové desky Arduino a dalších použitých součástek. V praktické části je popsán způsob zapojení a řízení všech prvků modelu chytrého domu.

Klíčová slova

Chytrý dům, inteligentní elektroinstalace, Arduino

Abstract

The master thesis deals with development of a control system for educational model of smart home with intelligent wiring (lighting, heating, alarm). The Arduino board is used to implement the control system. All components can be controlled locally or remotely using an Android phone. The first part of the thesis explains the concepts of smart home and intelligent wiring, followed by a description of the development board Arduino and other used components. The practical part describes the way of connecting and controlling all components of the smart home.

Keywords

Smart home, intelligent wiring, Arduino

Obsah

Seznam obrázků

Seznam tabulek

Seznam termínů a zkratek

3D	trojdimenzionální
C++	programovací jazyk
CGRAM	paměť obsahující znaky využívané pro zápis na displej
DC	stejnosměrný proud
DDRAM	paměť obsahující znaky aktuálně zobrazené na displeji
E	potvrzení
FS	fakulta strojní
GND	uzemnění
I/O	vstup / výstup
I^2C	integrovaná datová sběrnice
ICSP	sériové programování mikrokontroléru
IDE	vývojové prostředí
LCD	displej z tekutých krystalů
LED	elektroluminiscenční dioda
PC	osobní počítač
PIR	pasivní infračervené čidlo
PWM	pulzně šířková modulace
R/W	čtení / zápis
RST	reset
RP	reálný přepínač
RS	zvolení registru
RX	přijatá data
SCL	hodinový signál
SDA	datový signál
SELV	bezpečné malé napětí
TUL	Technická univerzita v Liberci
TX	odeslaná data
USB	univerzální sériová sběrnice
VCC	napájecí napětí
VP	virtuální přepínač
Wi-Fi	typ bezdrátové komunikace

Úvod

Tématem této práce je sestavení výukového modelu chytrého domu s inteligentní elektroinstalací. Při návrhu elektroinstalace je snaha propojit dílčí prvky modelu domu s centrální jednotkou, která umožní jejich vzájemnou komunikaci, kontrolu a řízení. Řídicí systém je realizován užitím vývojové desky Arduino. Tyto jsou v současné době velmi oblíbené, a to díky jejich jednoduchosti, nízké ceně, početné uživatelské komunitě a volně přístupnému softwaru na oficiálních stránkách výrobce.

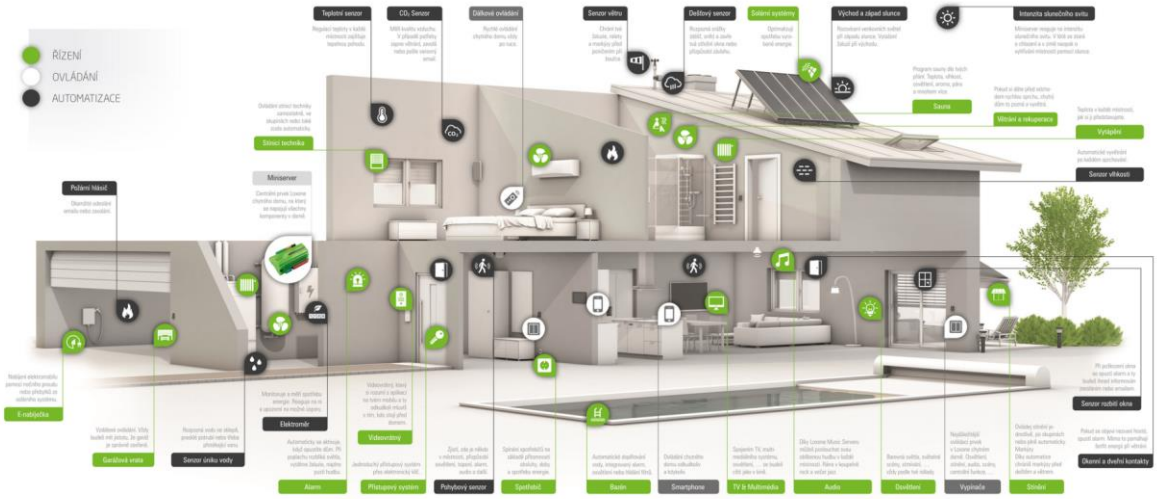
Součástí modelu domu je osvětlení všech místností, světlo před domem na časový spínač, čidlo teploty, topení a domovní alarm. Alarm se skládá z čidla pohybu, klávesnice pro zadání bezpečnostního hesla a displeje. Všechny zmíněné součásti lze ovládat jak lokálně manuálně pomocí tlačítek, přepínačů, popřípadě klávesnice, nebo vzdáleně pomocí mobilního telefonu. Telefon komunikuje s centrální jednotkou přes internet pomocí aplikace Blynk, v níž je vytvořeno uživatelské prostředí pro snadné ovládání, kontrolu a pozorování stavu jednotlivých prvků domu.

Při návrhu modelu je kladen důraz na přehledné zapojení všech elektrických součástek. Vodiče jsou vedeny po stěnách modelu domu a jsou barevně oddělené podle typu použití. Jednotlivé prvky chytrého domu jsou programovány jako stavové automaty. Chování těchto automatů lze přehledně vyjádřit grafem, což s kombinací s reálným modelem vytváří vhodnou výukovou pomůcku pro pochopení chování řídicího systému chytré domácnosti jako celku.

1 Chytrý dům

Pojem chytrý dům, chytrá domácnost, někdy také digitální domácnost bývá definován mnoha způsoby, ale obecně označuje moderní bydlení vybavené počítačovou a komunikační technikou, která předvídá a reaguje na potřeby obyvatel. Cílem je uživatelům takové domácnosti zvýšit komfort a pohodlí, zaručit co nejvyšší bezpečí a snížit energetické náklady prostřednictvím inteligentní elektroinstalace, jež umožní propojení a řízení jednotlivé techniky v domácnosti.

Sjednocením techniky v domácnosti a díky její vzájemné komunikaci získáme jednotné ovládání, které lze přizpůsobit na míru domu a jeho obyvatelům. Ovládání může být realizováno lokálně ve formě nástěnného systémového dotykového panelu, nebo dálkově například přes internet pomocí chytrého telefonu.



Obrázek 1: Funkce chytré domácnosti [1]

Důležitou vlastností chytrého domu je, aby i přes veškerý komfort a automatizaci, kterou chytrá domácnost zajišťuje, člověk neztrácel kontrolu nad jednotlivými prostředky, které se v případě potřeby dají i nadále ovládat standardním způsobem (např. rozsvícení a zhasnutí světel pomocí běžných přepínačů). Výsledný pocit z užívání chytrého bydlení by měl působit přirozeně a jednoduše. [2]

1.1 Výhody chytrých domů

Mezi základní východy, které chytré domácnosti přináší jejím uživatelům, jsou:

- komfortní funkce,
- úspora energií,
- bezpečnost,
- automatické funkce.

1.1.1 Komfortní funkce

Příkladem komfortních funkcí chytrého domu může být osvětlení se stmíváním, přednastavené světelné scény či regulace teploty v každé místnosti zvlášť (pokud to tedy topný systém umožňuje). Další užitečnou funkcí je možnost ovládání a kontrola jednotlivých zařízení na dálku pomocí mobilního telefonu. [3]

1.1.2 Úspora energií

Spotřebu energií může chytrá domácnost ovlivnit regulací vytápění a klimatizace, chytrou regulací osvětlení, kdy je intenzita osvětlení závislá na světelných podmínkách okolí, nebo blokováním spotřebičů při vysokém tarifu v elektrické síti. [3]

1.1.3 Bezpečnost

Základní funkcí bezpečnostního systému bývá detektor pohybu s alarmem, který v případě sepnutí okamžitě pošle tuto informaci uživateli do chytrého telefonu, popřípadě informuje bezpečnostní agenturu. Dalším bezpečnostním opatřením může být senzor zatopení, nebo kouřový senzor. A v neposlední řadě chytrá domácnost umožňuje pomocí chytrého telefonu zjistit jaký spotřebič nebo zařízení je v provozu, a popřípadě ho na dálku vypnout.

1.1.4 Automatické funkce

Chytrá domácnost umožňuje automaticky vykonávat operace na základě zvolené veličiny jako je čas, teplota, nebo osvětlení. Například při setmění automaticky zatáhne žaluzie, rozsvítí světla a podobně. [3]

1.2 Nevýhody a rizika chytrých domů

Mezi nevýhody nebo omezení chytrých domácností patří potřebné stavební úpravy s tím spojené pořizovací náklady. Také je třeba zmínit často diskutované téma, čímž jsou bezpečnostní rizika. Uvedené zápory lze rozdělit do tří skupin:

- stavební úpravy,
- náklady,
- bezpečnost.

1.2.1 Stavební úpravy

Pro realizaci chytrého domu je potřeba vzájemná komunikace jednotlivých zařízení. To může být problém zejména u již postaveného domu. Chytrá domácnost může sice být zcela bezdrátová, ale za spolehlivější bývá někdy brána chytrá elektroinstalace, která vyžaduje stavební úpravy. [4]

1.2.2 Náklady

Častou překážkou v pořízení chytré domácnosti jsou vysoké ceny spotřebičů, systémů a instalace. Kvalitní chytrá elektroinstalace se může pohybovat ve statisícových částkách. Investice může uživateli zajistit určité pohodlí a komfort, ale po finanční stránce se následnými energetickými úsporami investované peníze většinou nevrátí. [4]

1.2.3 Bezpečnost

O bezpečnosti v kladném slova smyslu bylo psáno již v kapitole (1.1.3), ale z druhého pohledu je chytrá domácnost připojena k internetu, což může mít za následek velké bezpečnostní riziko. Některé chytré domy posílají svá data na externí servery, kde mohou být dál zpracovávány. Potenciál ke zneužívání informací je tedy velmi vysoký. [4]

2 Typy elektroinstalací

Na základě použití přístrojů a zařízení v budovách a bytech lze současné způsoby elektroinstalace rozdělit do dvou skupin:

- klasická elektroinstalace,
- systémová elektroinstalace, která se dle druhu použitého systému dělí na:
  - centralizované systémy,
  - decentralizované systémy,
  - hybridní systémy. [5]

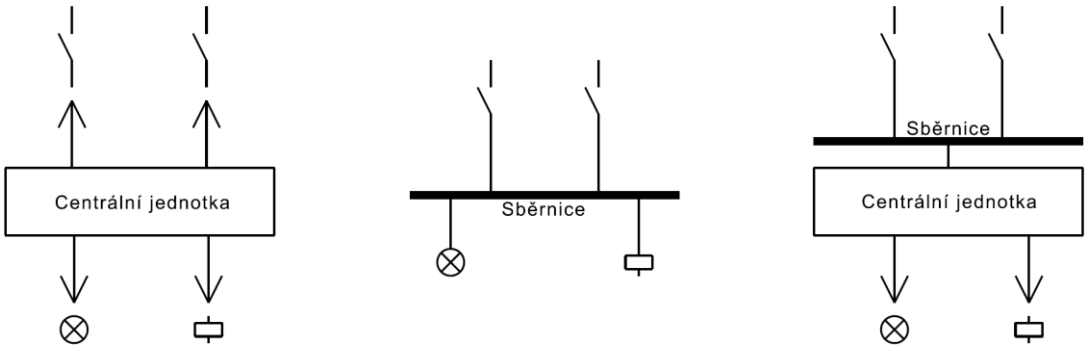
2.1 Klasická (konvenční) elektroinstalace

Klasická elektroinstalace se skládá z různých samostatných obvodů, jako je obvod pro ovládání osvětlení, topení, žaluzií a tak dále. K řízení se nepoužívají žádné sběrnice, ale každý prvek se ovládá samostatně, což má za následek další náklady a stavební úpravy při jakékoliv změně v elektroinstalaci. [5]

2.2 Systémová (inteligentní) elektroinstalace

Tento typ elektroinstalace je navržený tak, že jsou všechny provozně technické funkce propojeny do jednoho funkčního celku, kde mezi sebou mají vazby umožňující jejich vzájemnou komunikaci. Tento systém pak může řešit vše od řízení a ovládání osvětlení, řízení pohonu rolet až po vizualizaci celé použité technologie. [5]

Inteligentní elektroinstalací jsou propojeny jednotlivé prvky systému pomocí sběrnicového kabelu. Podle typu tohoto propojení se sběrnicové systémy dělí na centralizované a decentralizované. Kombinací těchto dvou systému vzniká systém hybridní. Rozdělení inteligentních elektroinstalací je znázorněno na obrázku 2.



Obrázek 2: Centralizované, decentralizované a hybridní sběrnicové systémy

2.2.1 Centralizované řídicí systémy

U těchto systémů jsou všechny vstupy (senzory) a výstupy (aktory) připojeny hvězdicově k centrální jednotce. Ke vzájemné komunikaci mezi připojeným zařízením může docházet jen přes tuto centrálu. Snímač předá informaci centrální řídicí jednotce, ta přijatou informaci zpracuje, a na základě toho pošle příkazy akčnímu členu, který například rozsvítí žárovku. Centrální řízení se vyznačuje vysokou spolehlivostí komunikace; naopak nevýhodou je přerušení veškeré vzájemné komunikace dílčích prvků při výpadku centrální jednotky. Další nevýhodou může být větší spotřeba kabelů oproti sběrnicové topologii.

2.2.2 Decentralizované řídicí systémy

V decentralizovaném systému spolu účastníci (senzory a aktory) komunikují prostřednictvím sběrnice, na kterou jsou připojeny. Každý z těchto účastníků má svoji řídicí jednotku. Není zde žádné centrální řízení. Každý účastník může neustále přijímat nebo odesílat informace, které jsou důležité pro splnění požadované činnosti. Výhodou sběrnicové topologie u decentralizovaných systémů je ta, že po výpadku jednoho prvku zůstanou funkční zbylé prvky na sběrnici. Další výhodou může být poměrně jednoduché připojování dalších prvků do systému. Nevýhodou je vyšší cena připojených zařízení, které musí mít vlastní řídicí jednotku. [6]

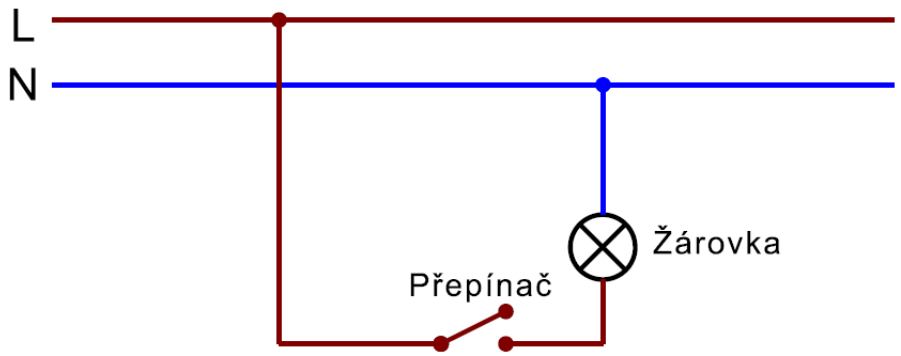
2.2.3 Hybridní (smíšené) řídicí systémy

Hybridní řídicí systémy jsou kombinací centralizovaných a decentralizovaných systémů. U těchto systémů jsou vstupy připojeny na sběrnici a výstupy do centrální jednotky. Součástí systému je tedy řídicí jednotka, která umožňuje ovládat dané prvky z jednoho místa a zároveň se minimalizují náklady na vstupní čidla. [6]

2.3 Porovnání klasické a systémové elektroinstalace

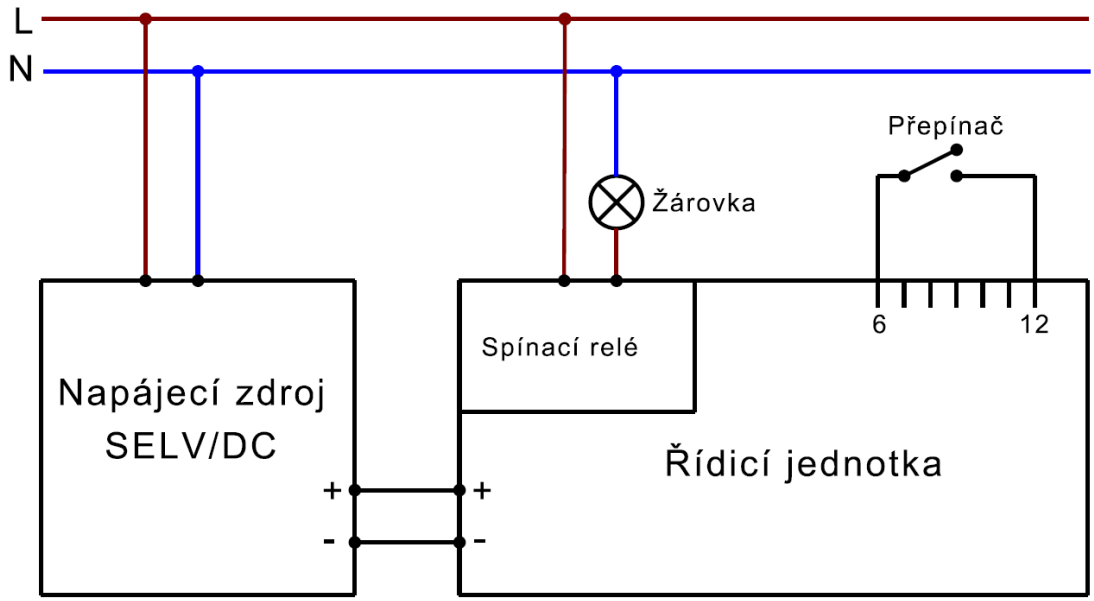
Hlavním rozdílem mezi klasickou a systémovou elektroinstalací je, že u té klasické se ovládacím prvkem spíná přímo příslušný spotřebič, zatímco u systémové ovládací prvek zasílá pouze povely pro spínání. Princip funkce obou typů instalací lze přehledně popsat na příkladu spínání žárovky – viz obrázky 3 a 4.

Na obrázku 3 je zobrazeno schéma principu spínání žárovky klasickým způsobem, kdy stiskem přepínače dojde k uzavření obvodu mezi pracovním vodičem L a nulovým vodičem N, a tím pádem i k rozsvícení žárovky.



Obrázek 3: Princip spínání žárovky klasickým způsobem

Příklad spínání žárovky se systémovou elektroinstalací je znázorněn na obrázku 4. Řídicí jednotka je napájena ze SELV[1] zdroje. Na tomto nízkém napětí je na vstupu mezi piny 6 a 12 připojen přepínač, jehož sepnutím dostane řídicí jednotka impulz, aby na výstupu sepnula relé[2]. Tím se propojí okruh a žárovka se rozsvítí.



Obrázek 4: Princip spínání žárovky se systémovou elektroinstalací

[1] SELV zkratka z anglického (Safety Extra Low Voltage) – bezpečné malé napětí
[2] Nemusí se vždy jednat o relé, výstupy mohou být například tranzistorové, nebo SSR výstupy.

3 Stavový automat

Automat popisuje velmi jednoduchý počítač, kterým lze modelovat požadované chování systému. Stavový automat obsahuje určité množství stavů, přičemž informace o právě aktuálním stavu si ukládá do paměti. Díky té není výstup závislý pouze na aktuální vstupní informaci, ale je pro něj podstatná celá kombinace předchozích vstupů. Chování automatu lze vyjádřit grafem skládajícím se ze:

- stavů,
- přechodů při splnění podmínky,
- akcí.

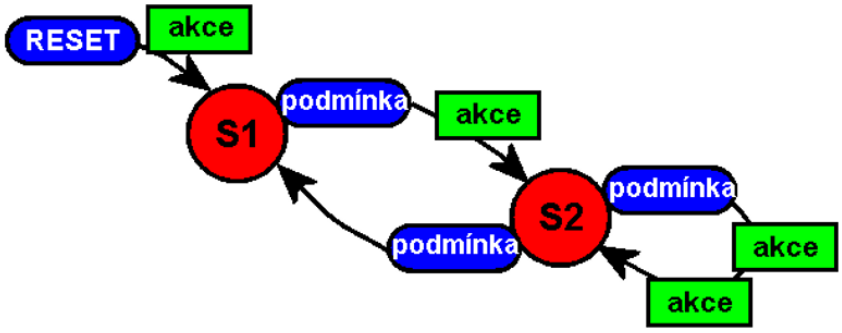
Stavem se rozumí ustálené trvání určitých vlastností systému. Dojde-li ke změně těchto vlastností, změní se i stav. Změnou stavu může být rozsvícení žárovky a podmínkou pro ni může být například doběhnutí časovače.

Další částí stavového grafu jsou přechody při splnění podmínky. Značí se šipkou, která směřuje od předchozího stavu k aktuálnímu. Tím může být někdy i tentýž stav. Jak již vyplývá z názvu, k přechodu mezi stavy dojde při splnění podmínky pro změnu stavu. Výjimkou může být počáteční stav S1, ke kterému je veden přechod z bodu RESET.

Akcí je pokyn pro vykonání určité činnosti, například zhasni, rozsviť nebo nastav časovač. Akce se vykoná při zahájení přechodu mezi stavy, přičemž nemusí proběhnout při každém přechodu, a zároveň naopak může proběhnout více akcí při jednom přechodu.

Automat může být realizován například pomocí mikroprocesoru, který má na výstupech připojená světla, bzučáky a podobně, a na vstupech tlačítka. Grafy automatů se tvoří podle zásad[3], jenž jsou zobrazeny na obrázku 5. Graf vychází z podmínky RESET, z níž vede přechod do prvního stavu S1. Jednotlivé stavy jsou zobrazovány v kruhovém poli. Na výstupu z každého stavu musí být podmínka. Podmínky jsou ohraničeny v oválném poli. Přechody mezi jednotlivými stavy jsou zobrazovány šipkou, ta je orientována vždy jen jedním směrem, a to z výchozího stavu do následujícího. Akce, které mají být při přechodu mezi stavy vykonány, jsou zobrazovány v obdélníkovém poli, jenž je umístěno za podmínkou na šipce přechodu. Akcí může být také nastavení časovače na určitou hodnotu.

[3] Tyto zásady se dodržují v předmětu Programovatelné logické systémy na FS v Liberci. Zásady pro tvorbu grafů automatů se mohou v jiných zdrojích lišit.



Obrázek 5: Graf automatu

4 Model domu

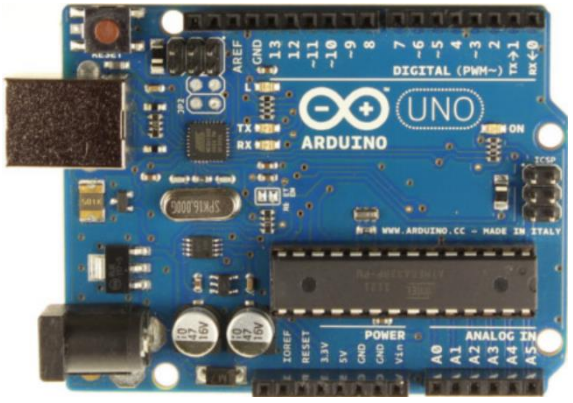
Níže je zobrazen model domu, na kterém bude navržena a realizována inteligentní elektroinstalace. Původní majitel již neměl pro tento tomek využití, a tak ho poskytl ke studijním účelům. Dům se skládá ze dvou pater, z nichž každé má dvě místnosti. Místnosti jsou od sebe odděleny tenkou dřevěnou stěnou. Nosnou konstrukci tvoří čtyři sloupky umístěné v rozích a dva v čele. Model je po obvodu otevřený, což je zásadním požadavkem pro přehledné vedení elektroinstalace. Jednotlivé místnosti byly očíslovány podle obrázku 6. Z pravé horní strany, proti směru hodinových ručiček, místnost číslo jedna, dva, tři a čtyři.

Obrázek 6: Poskytnutý model domu

5 Arduino

Arduino je otevřená elektronická platforma. Skládá se z jediné desky plošných spojů, jejíž základem je mikroprocesor od firmy Atmel, ke kterému jsou připojeny další elektronické komponenty. Arduino nelze brát jako samostatný počítač, jedná se pouze o vývojovou desku. Aby Arduino deska vykonávala, co se po ní chce, je potřeba vytvořit řídicí program a nahrát ho přes převodník do mikroprocesoru dané desky. Pro tvorbu řídicího programu lze využít oficiální volně dostupný software Arduino IDE. Pro připojení dalších součástí a obvodů obsahuje Arduino určité množství pinů s různými funkcemi a vlastnostmi. K napájení obvodů podle potřeby a typu zařízení mohou sloužit $5\ [V]$ nebo $3{,}3\ [V]$ piny. Pokud je do obvodu potřeba dodat vyšší napětí, je nutné použít externí zdroj. [7]

Podle požadavků si lze vybrat mezi několika typy Arduino desek, které se mezi sebou liší velikostí, počtem vstupů a výstupů, výkonem, velikostí paměti nebo typem procesoru. Pokud je hlavním kritériem malá velikost, lze použít desky Mini, Mikro a Nano. Mezi komplexnější a zároveň nejpoužívanější desku patří Arduino Uno; požadavek na větší počet pinů splňuje například Arduino Mega2560. Kromě originálních Arduino desek, existuje spousta dalších neoficiálních, tak zvaných klonů. Mohou to být napodobeniny oficiálních modelů nebo mohou být speciálně uzpůsobeny konkrétní činnosti. Rozšířit možnosti jednotlivé vývojové desky o další funkce lze pomocí přídavných modulů tak zvaných shieldů, které lze nasunout do zdířek na Arduinu. Mezi oficiální shieldy patří například Ethernet Shiled, WiFi Shield, nebo Motor Shiled. [7]



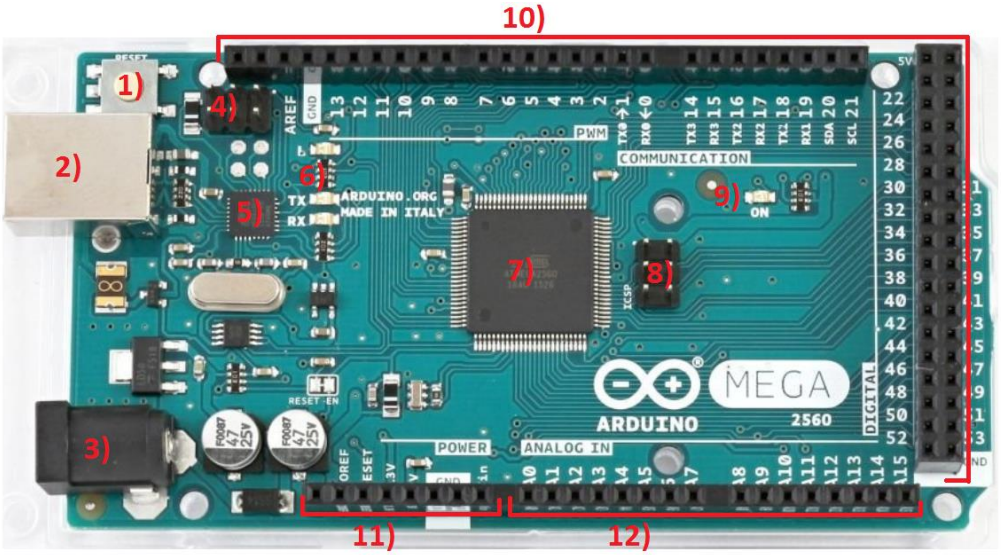
Obrázek 7: Arduino Uno

Obrázek 8: Ethernet Shield

Požadavkem na řídicí systém modelu chytrého domu byl vysoký počet digitálních I/O pinů, proto byla zvolena vývojová deska Arduino Mega2560.

5.1 Arduino Mega2560

Tato vývojová deska je založená na mikrokontroléru ATmega2560. Obsahuje 54 digitálních vstupů/výstupů z toho 15 použitelných jako PWM[4] výstupy, 16 analogových vstupů a 4 hardwarové sériové porty. Na obrázku 9 jsou očíslovány a níže i popsány základní prvky této desky.



Obrázek 9: Arduino Mega2560

1. Resetovací tlačítko. Po jeho stisknutí se spustí nahraný řídicí program znovu od začátku.
2. USB konektor typu B. Slouží k propojení Arduina s PC. K propojení je potřeba USB kabel a PC s USB portem. Přes USB kabel lze Arduino napájet a nahrát do něj příslušný řídicí program.
3. Napájecí konektor. Lze jej využít místo napájení z USB. Podle výrobce je doporučené vstupní napětí $5\ [V]$ až $12\ [V]$ DC a maximální $20\ [V]$.
4. ICSP hlavice. Slouží pro externí programování USB-serial převodníku. [7]
5. USB-serial převodník. Obstarává komunikaci mezi mikroprocesorem a PC.
6. Indikační LED diody. Diody s popisem Tx a Rx signalizují komunikace po

PWM (Pulse Width Modulation, pulsně šířková modulace). Jedná se o jakousi digitální „náhražku“ analogového signálu. Blíže popsáno v kapitole (5.3.1). Ovládání analogového vstupu a výstupu.

sériové lince. Nad nimi je dioda s popisem L, ta je připojena k pinu číslo 13 a slouží spíše ke kontrolním účelům.

7. Mikroprocesor.
8. Druhá ICSP hlavice. Slouží pro externí programování mikroprocesoru. [7]
9. Indikační LED dioda. Svítí, pokud je připojeno napájení.
10. Převážně digitální piny. Většina z těchto pinů jsou čistě vstupně/výstupní digitální piny, část z nich podporuje PWM výstup, některé slouží ke komunikaci.
11. Převážně napájecí výstupy.
12. Analogové vstupy.

5.2 Vývojové prostředí Arduino IDE

Integrované vývojové prostředí Arduino IDE je volně dostupný software sloužící pro psaní programů, jimiž se daná deska Arduino bude po jeho nahrání řídit. Podle operačního systému si lze na stránkách výrobce stáhnout tento program pro Windows, Linux nebo Mac OS. V případě Windows si lze stáhnout ZIP archiv a spustit program bez nutnosti instalace. Pokud použijeme právě poslední zmíněný způsob, získáme po rozbalení archivu na zvoleném místě tři složky. První je složka *Drivers*, která obsahuje ovladače pro komunikaci mezi Arduinem a PC. Ve složce *Examples* jsou uloženy příklady kódů a do *Libraries* se ukládají knihovny, které jsou podrobněji popsány níže v podkapitole (5.2.3). Po spuštění vývojového prostředí se zobrazí okno, do kterého lze ihned psát kód. [7]

Obrázek 10: Vývojové prostředí Arduino IDE

5.2.1 Popis vývojového prostředí

Vývojové prostředí je zobrazeno na obrázku 10. V horní liště je důležitá zejména nabídka *Tools*, která obsahuje nastavení pro připojení desky. Pod hlavní navigační lištou je řádek s funkcemi. Popsáno zleva:

- *Verify*. Tato funkce spustí kontrolu kódu. Pokud bude nalezena nějaká chyba v programu, vypíše ji do černého okna na spodní části obrazovky s odhadem na pozici chyby. Pokud kód neprojde touto kontrolou, nelze ho do Arduina nahrát.
- *Upload*. Zkontroluje kód a přenese jej do připojeného Arduina.
- *New*. Vytvoří nový projekt.
- *Open*. Slouží pro nahrání uloženého projektu.
- *Save*. Uloží současný projekt.
- *Serial Monitor*. Stiskem této funkce dojde po nahrání kódu do Arduina k výpisu vybraných hodnot do okna sériového monitoru, a to při každém proběhnutí cyklu.

Bílý prostor pod těmito funkcemi slouží k zápisu kódu a černý prostor dole zobrazuje případné chybové nebo informační výpisy.

5.2.2 Nahrávání kódu

Pro nahrání kódu do Arduina je potřeba v nabídce *Tool*s zvolit na jakém USB portu je Arduino k PC připojeno a o jaký typ vývojové desky jde. Poté se stiskne již zmíněná funkce *Upload*. Připojená deska se automaticky resetuje a začne nahrávání kódu. V průběhu nahrávání blikají na desce diody Tx a Rx a v dolní části vývojového prostředí se zobrazí stav.

5.2.3 Knihovny

Knihovny jsou jakési balíčky kódu, které lze do programu nahrát, a tím jej rozšířit o potřebnou funkci. Také zjednodušují a zpřehledňují samotný program. Připojíme-li nějakou součást, například display, lze pro něj nahrát příslušnou knihovnu, a tím si značně ulehčit programování. Některé knihovny jsou již součástí programovacího prostředí, další lze dohledat na internetu. Pro použití stažené knihovny je potřeba

ji nahrát. Nejjednodušším způsobem je v horní liště vývojového prostředí rozbalit záložku *Sketch*, zvolit *Import Library* a vybrat požadovanou funkci. [8]

5.3 Programovací jazyk

Nejrozšířenějším způsobem programování Arduina je použití knihovny zvané Wiring, jež je součástí programovacího jazyka *C++*. Tato knihovna rozšiřuje jazyk o příkazy určené pro přímé řízení hardwarových součástek, a tím v kombinaci s dalšími knihovnami vývojáři velmi ulehčuje práci. Díky komplexnosti této knihovny je často nazývána samostatným programovacím jazykem. [7]

5.3.1 Struktura „jazyka“ Wiring

Na ukázce zdrojového kódu číslo 1 jsou zobrazeny dvě základní funkce, které musí být vždy součástí kódu. Jsou to funkce *setup()* a *loop().* Složené závorky vymezují začátek a konec jednotlivých funkčních bloků (funkcí) a za dvojitá lomítka lze psát komentář. V této kapitole budou dále popsány základní prvky používané při tvorbě programu řídicí jednotky chytrého domu. Podrobnější popis je například na oficiálních stránkách Arduina [9].

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Zdrojový kód 1: Základní funkce

Funkce *setup()*

Tato funkce je vyvolána pouze jednou na začátku programu. To může být po nahrání programu do Arduina, připojení napájení nebo restartu. Slouží především k definování vstupů a výstup na jednotlivých pinech.

Funkce *loop()*

Do této funkce se zapisuje kód, který se bude po vykonání předešlé funkce *setup()* neustále opakovat v nekonečné smyčce, dokud nedojde k odpojení zařízení. Tím je programu umožněno reagovat na vstupní podněty a řízení desky Arduino.

Proměnné

Proměnná je místem, ve kterém lze uchovávat určitou informaci za běhu programu. Tato informace může být známá nebo neznámá a nazývá se hodnota. K vytvoření (deklaraci) proměnné je potřeba kromě její hodnoty určit také její název a datový typ.

Datové typy

Datový typ určuje, jaké hodnoty mohou být v proměnné uschovány. Může se jednat o čísla, logické hodnoty nebo třeba znaky. Mezi číselné datové typy patří například *byte*, *integer*, *long* nebo *float*. *Boolean* je logický datový typ a *char* znakový. Na ukázce zdrojového kódu číslo 2 je zobrazena deklarace proměnných jednotlivých datových typů.

```
byte a = 20;         // byte
int b = 200;         // integer
long c = 20000000;   // long
float d = 2.222;     // float
boolean e = true;    // boolean
char f = 'A';        // char
```

Zdrojový kód 2: Datové typy

Proměnná datového typu *byte*

Proměnná typu *byte* slouží k uchování celých čísel, má velikost 8 bitů a její rozsah je 0 až 255.

Proměnná datového typu *integer*

Proměnná s tímto datovým typem slouží k uchování celých čísel, její velikost se odvíjí od velikosti daného procesoru, například u procesoru Atmega je to 16 bitů.

Proměnná datového typu *long*

V proměnné s tímto datovým typem je možné uchovat číselnou hodnotu o velikosti až 32 bitů, tedy od $-2\,147\,483\,648$ do $2\,147\,483\,647$.

Proměnná datového typu *float*

Proměnná tohoto typu slouží k uchování čísel s desetinou čárkou, v jazyce wiring s desetinou tečkou, o hodnotě až 32 bitů.

Proměnná datového typu *boolean*

V proměnné tohoto typu lze uchovávat pouze logickou hodnotu TRUE (pravda) nebo logickou hodnotu FALSE (nepravda).

Proměnná datového typu *char*

Proměnná tohoto znakového typu slouží k uchování jednoho textového znaku.

Konstanty

Od tvůrců Arduina bylo přednastaveno několik hodnot, které se nazývají konstanty. Slouží především k zpřehlednění programu.

Konstanty typu TRUE a FALSE

Jedná se o logické konstanty, používají se v případě, že je potřeba rozhodnout mezi dvěma stavy. Konstanta FALSE má hodnotu 0, konstanta TRUE je definována jako 1, ale muže nabývat i jiných hodnot kromě nuly.

Konstanty typu HIGH *a* LOW

Tyto konstanty se používají pro čtení nebo zápis jedné z hodnot na digitální piny. Hodnotu HIGH lze chápat jako logickou 1 a LOW jako logickou 0.

Konstanty typu INPUT a OUTPUT

Jedná se o konstanty určující, zda bude digitální pin plnit funkci vstupu INPUT, nebo výstupu OUTPUT.

Nastavení digitálního vstupu a výstupu

Arduino vždy obsahuje určitý počet digitálních pinů, které lze podle potřeby nastavit jako vstupní, nebo výstupní. K tomuto nastavení slouží funkce *pinMode()*.

Funkce *pinMode()*

Jak lze vidět na ukázce zdrojového kódu číslo 3, mezi závorky této funkce je vždy potřeba zadat číslo pinu a dále konstantu, která určí, jestli se daný digitální pin bude chovat jako výstup – OUTPUT, anebo jako vstup – INPUT.

```
void setup() {
  pinMode(2, OUTPUT);         // Nastavení pinu 2 jako OUTPUT.
  pinMode(3, INPUT);          // Nastavení pinu 3 jako INPUT.
  pinMode(4, INPUT_PULLUP);   // Nastavení pinu 4 jako INPUT_PULLUP.
}
```

Zdrojový kód 3: Nastavení digitálních pinů

Další možností je nastavit digitální pin jako INPUT_PULLUP. Tím dojde k připojení interního rezistoru mezi daný pin a $+5\ [V]$. Digitální pin se tedy bude chovat jako vstup, který přijímá hodnotu HIGH. Hodnotu LOW lze poté získat připojením pinu na GND.

Ovládání digitálního vstupu a výstupu

Pokud jsou digitální piny nastaveny, je možné s nimi dál pracovat. Pro zápis a čtení digitálních pinů se používají funkce *digitalWrite()* a *digitalRead()*. Jejich zápis je zobrazen na ukázce zdrojového kódu číslo 4.

Funkce *digitalWrite()*

Tato funkce se používá k ovládání digitálních výstupů, lze s ní nastavit daný pin na hodnotu HIGH $5\ [V]$ nebo LOW $0\ [V]$.

Funkce *digitalRead()*

Tato funkce slouží ke čtení vstupních pinů. Po zadání čísla pinu vrátí jeho aktuální hodnotu. Při čtení je u většiny Arduino desek napětí vyhodnoceno jako HIGH, pokud je větší než $3\ [V]$, a LOW, pokud je napětí menší než $2\ [V]$.

```
int A;                    // Proměnná pro uchování hodnot.
void loop() {
  A = digitalRead(3);     // Do proměnné A se uloží stav vstupu číslo 3.
  digitalWrite(2, HIGH);  // Nastavení výstupu číslo 2 na HIGH.
}
```

Zdrojový kód 4: Ovládání a čtení digitálních pinů

Ovládání analogového vstupu a výstupu

Arduino desky obsahují určitý počet pinů sloužící pro práci s analogovými hodnotami. Pro analogové vstupy jsou vyhrazeny piny označené[5] písmenem A, ke čtení hodnot na těchto vstupech se používá funkce *analogRead()*. Pro výstup jsou vyhrazeny piny označené[6] jako PWM a pro zápis slouží funkce *analogWrite()*. Analogové piny na rozdíl od digitálních není třeba nastavovat jako vstupní, nebo výstupní.

Funkce *analogRead()*

Tato funkce načte hodnotu analogového pinu a uloží ji do příslušné proměnné. Načtená hodnota je u většiny Arduino desek v desetibitovém rozlišení, tedy celé číslo v rozsahu od 0 do 1023. [7]

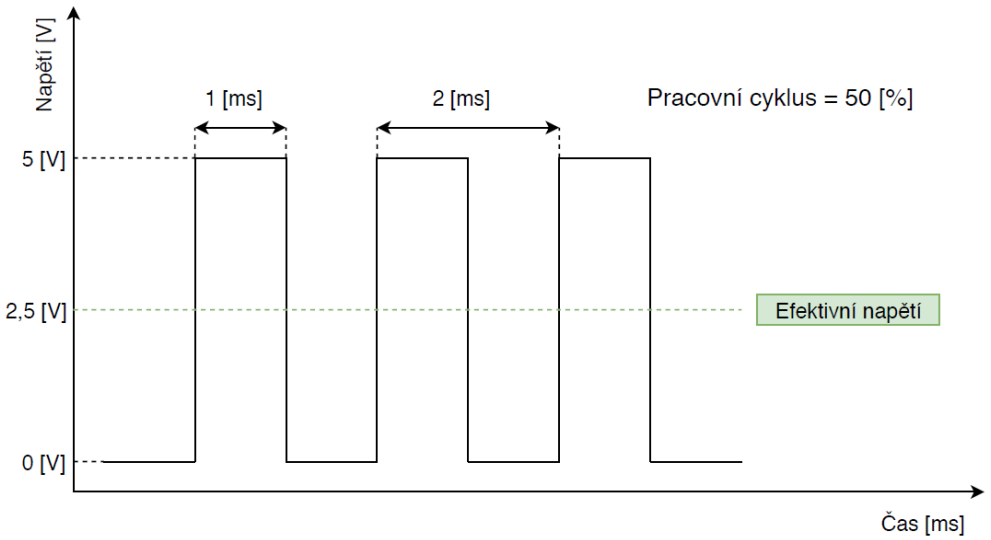
[5] Arduino Mega2560 má vyhrazené piny pro analogové vstupy od $A0$ do $A15$.
[6] U Arduina Mega2560 podporují PWM digitální piny 2 až 13.

```
int B;                  // Proměnná pro uchování hodnot.
void loop() {
  B = analogRead(A0); // Do proměnné B se uloží stav vstupu číslo A0.
  B = B/4;            // Úprava rozsahu.
  analogWrite(5, B);  // Nastavení na výstup číslo 5 odpovídající PWM.
}
```

Zdrojový kód 5: Ovládání a čtení analogových pinů

Funkce *analogWrite()*

Tato funkce neslouží doslovně k nastavení analogového výstupu, jak by se z jejího názvu mohlo zdát. Arduino totiž samo o sobě analogové výstupy nemá, místo nich používá metodu pulzně šířkové modulace signálu, tedy PWM. Princip šířkové modulace signálu jako náhražky analogového hodnoty lze vysvětlit na jednoduchém příkladu, kdy neustále rozsvěcíme a zhasínáme žárovku. Žárovka bude přirozeně blikat, ale pokud by se zvyšovala frekvence přepínaní, tak v určité chvíli vlivem setrvačnosti vlákna přestane stíhat reagovat na průběh signálu, a následkem toho se sníží intenzita jejího svitu. Arduino pracuje na stejném principu. Na výstupu rychle[7] střídá hodnoty $0\ [V]$ a $5\ [V]$ a podle poměru času, ve kterém je na výstupu jedna z těchto hodnot, se chová připojený akční člen, jako by byl napájen napětím přímo úměrným právě tomuto poměru. [7]



Obrázek 11: Průběh signálu PWM, při pracovním cyklu 50 [%]

[7] Arduino generuje signál PWM o frekvenci přibližně $500\ [Hz]$, perioda je tedy po přepočtu přibližně $2\ [ms]$. [10]

Při zápisu funkce *analogWrite()* je potřeba určit číslo pinu a hodnotu plnění PWM. Tato hodnota se zadává v rozsahu od 0 do 255 a určuje podíl doby zapnutí vůči době celé periody. Příklad použití této funkce je na ukázce zdrojového kódu číslo 5. Na této ukázce je do proměnné B ukládána analogová hodnota, například z potenciometru. Dalším krokem je převod načtené hodnoty na hodnotu vhodnou pro zápis. Následným zápisem této hodnoty se na daném pinu generuje PWM signál. Možný průběh tohoto signálu je ilustrován na obrázku číslo 11. V tomto případě je na výstupu stejně dlouho stav $0\ [V]$ i $5\ [V]$. Doba periody je tedy dvakrát větší než doba zapnutí. A efektivní napětí odpovídá hodnotě $2{,}5\ [V]$. Pro zápis tohoto napětí pomocí funkce *analogWrite()* je potřeba zadat parametr plnění PWM odpovídající hodnotě 127.

Porovnávací operátory

Tyto operátory porovnávají dvě hodnoty. Pokud pro obě hodnoty platí, že jsou v souladu s daným operátorem, bude výsledkem porovnávací operace logická hodnota TRUE; v opačném případě logická hodnota FALSE.

```
A == B  // Pokud A je rovno B, vrátí hodnotu TRUE.
A != B  // Pokud B není rovno B, vrátí hodnotu TRUE.
A > B   // Pokud je A větší než B, vrátí hodnotu TRUE.
A < B   // Pokud je A menší než B, vrátí hodnotu TRUE.
A >= B  // Pokud je A větší nebo rovno B, vrátí hodnotu TRUE.
A <= B  // Pokud je A menší nebo rovno B, vrátí hodnotu TRUE.
```

Zdrojový kód 6: Porovnávací operátory.

Logické operátory

Tyto operátory slouží k vyhodnocení stavu mezi více porovnávacími operátory. Jejich výsledkem je opět logická hodnota TRUE, nebo FALSE.

```
(A > 0) && (B > 0)  /* Logické AND (konjunkce). Vyhodnotí jako TRUE,
                       pokud je výsledek obou porovnání TRUE. */

(A > 0) || (B > 0)  /* Logické OR (disjunkce). Vyhodnotí jako TRUE,
                       pokud je alespoň jedno porovnání TRUE. */

(!A > 0)            /* Logické NOT (negace).  Vyhodnotí jako TRUE,
                       pokud je výsledek porovnání FALSE. */
```

Zdrojový kód 7: Logické operátory

Podmínky

Často je při psaní programu potřeba, aby se část kódu přehrála pouze za určité situace. Právě k tomu se používá podmíněný příkaz *if()*. Mezi závorky se píše podmínka. Splněním podmínky je umožněno provést dané příkazy. Pro zápis podmínek slouží výše zmíněné porovnávací a logické operátory. Podmíněný příkaz *if()* lze dále rozšířit o zápis *else if()* a *else*. Příklad zápisu podmínek je na ukázce zdrojového kódu číslo 8. Podle tohoto zápisu by byla nejprve vyhodnocena podmínka číslo 1. Pokud by byla vyhodnocena jako TRUE, splnily by se příkazy číslo 1. Pokud jako FALSE, došlo by k vyhodnocení podmínky 2. Jestliže by byly obě podmínky vyhodnoceny jako FALSE, splnily by se příkazy číslo 3.

```
if (podmínka 1) {
  příkazy 1;
}
else if (podmínka 2) {
  příkazy 2;
}
else {
  příkazy 3;
}
```

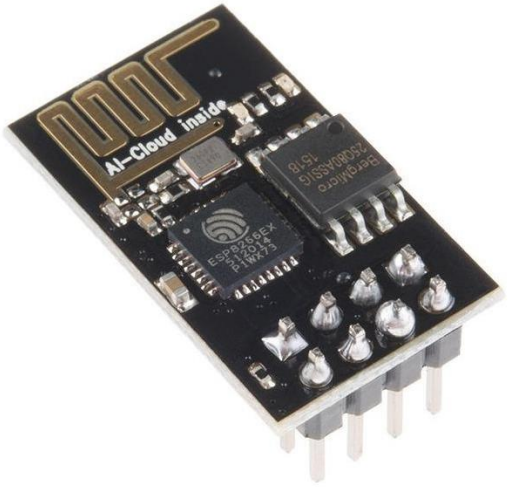
Zdrojový kód 8: Podmínky

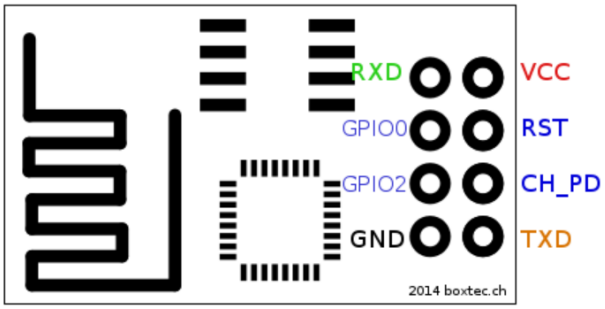
6 Další použité součástky

Nejdůležitější součástí modelu chytrého domu je zcela jistě mikropočítač Arduino, který měl pro svůj popis vyhrazenou celou předešlou kapitolu. Mezi další součástky patří maticová klávesnice, displej, pohybové čidlo, senzor teploty, Wi-Fi čip a další.

6.1 Wi-Fi modul ESP8266 verze ESP-01

Modul ESP8266 umožňuje poměrně snadné a levné připojení různých zařízení s mikroprocesorem k Wi-Fi síti. Bylo vytvořeno hned několik verzí tohoto modulu, které se od sebe liší velikostí desky, počtem pinů nebo například výkonem antény. Nejrozšířenější verzí je ESP-01. Tato verze modulu je zobrazena na obrázku číslo 12 a schéma s popisem jednotlivých pinů na obrázku 13.





Obrázek 12: ESP-01

Obrázek 13: ESP8266 – popis pinů [11]

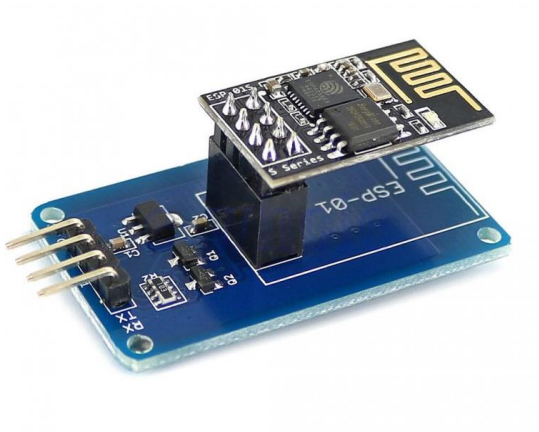
Na zobrazeném modulu lze vidět 8 pinů. Piny RXD a TXD slouží pro ovládání čipu jiným zařízením přes sériovou linku. VCC a GND jsou napájecí piny. Pin RST slouží pro restartování čipu a CH_PD pro zapínání a vypínaní čipu. Poslední dva piny GPIO0 a GPIO2 lze využít tehdy, pokud bude modul používán samostatně, tedy nebude ovládán jiným zařízením. [7]

6.1.1 ESP8266 a Arduino

Pokud má ESP8266 zprostředkovávat připojení Arduina k Wi-Fi síti, je potřeba jejich vzájemného propojení. Důležitou věcí, na niž je potřeba si dávat při zapojení pozor je, že ESP8266 modul pracuje na napětí $3{,}3\ [V]$, kdežto většina Arduin pracuje na napětí $5\ [V]$. Arduino sice má $3{,}3\ [V]$ výstup, kterým lze Wi-Fi čip napájet[8], problém ale nastává u sériové linky, jež u většiny Arduin pracuje na $5\ [V]$. Je tedy potřeba zajistit, aby vzájemná komunikace po sériové lince probíhala na napětí $3{,}3\ [V]$. Toho lze docílit například napěťovým děličem nebo vhodným adaptérem. Další komplikací může být rozmístění pinů ve dvou řadách blízko sebe, zejména pokud je potřeba čip zapojit do nepájivého kontaktního pole. [7]

[8] Pro napájení ESP8266 je vhodné použít silný stabilizovaný zdroj napětí. Silný zdroj je požadován především proto, že si čip při rozběhu může nárokovat elektrický proud řádově až ve stovkách miliampér. Arduino tedy jako zdroj napětí, není vhodné řešení.

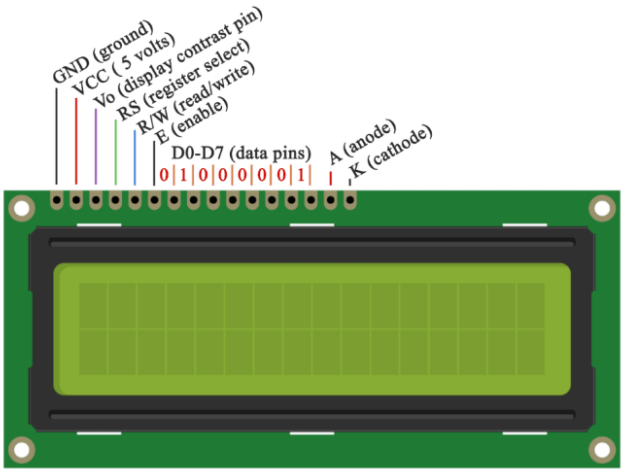
Oba zmíněné problémy, jak problém s rozdílným napětím, tak s rozmístěním pinů, lze jednoduše vyřešit použitím speciálního modulu adaptéru pro ESP-01 Wi-Fi platformu s integrovaným $3{,}3\ [V]$ stabilizátorem napětí.



Obrázek 14: ESP8266 Adaptér s ESP-01 modulem

6.2 LCD Displej

Na obrázku 15 je zobrazen znakový LCD displej 16×2. Jedná se tedy o dva řádky po šestnácti znacích. K ovládání displeje se používá řadič. Tento displej má integrovaný řadič HD44780 od firmy Hitachi. Display má dále paměť dělenou na DDRAM a CGRAM. Paměť DDRAM uchovává informaci o aktuálním obrazu, tedy jaký znak má být zobrazen na daném místě. V paměti CGRAM je uložena sada předdefinovaných znaků. Těmi jsou mimo jiné malá a velká písmena, numerické znaky, popřípadě přidané vlastní znaky. Tento display má šestnáctipinový konektor, jednotlivé piny jsou popsány na obrázku 15. Piny GND a VCC slouží k napájení, pomocí pinu V0 lze nastavit kontrast. Další tři piny jsou určeny řadiči, RS zvolí mezi instrukčním nebo davovým registrem, R/W určí, zda půjde o čtení nebo zápis a E potvrdí odeslání dat nebo instrukcí. Následujících osm pinů slouží pro datovou komunikaci. Poslední dva piny jsou určeny pro připojení anody podsvícení displeje a katody podsvícení displeje. [12]



Obrázek 15: LCD displej

6.3 LCD Displej a Arduino

Arduino IDE obsahuje knihovnu *LiquidCrystal.h*, která velmi ulehčuje komunikaci s LCD displeji. Ovládání displeje umožňuje tato knihovna pomocí jednoduchých příkazů. Níže je popsáno několik základních funkcí, které budou používány v této práci. Kompletní popis příkazů nalezneme například v odkazu [13] nebo česky v [7].

Funkce *LiquidCrystal lcd()*

Tato funkce vytvoří objekt pro práci s displejem. Parametry jsou piny připojené k displeji.

Funkce *lcd.begin()*

Zahájí práci s displejem. Parametry jsou počet řádků a sloupců.

Funkce *lcd.clear()*

Vyvoláním této funkce se z displeje smažou všechny zobrazené znaky a kurzor se přesune do levého horního rohu.

Funkce *lcd.setCursor()*

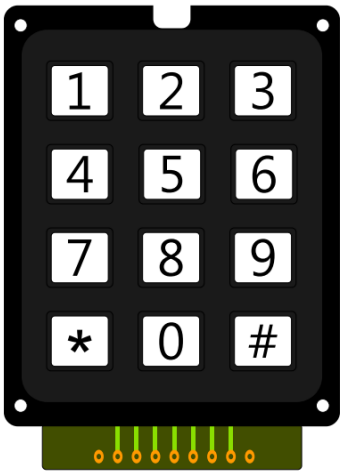
Tato funkce přesune kurzor na zvolenou pozici. Pozice se zadává pomocí dvou parametrů. Jeden parametr určuje řádek a ten druhý sloupec maticového displeje.

Funkce *lcd.prind()*

Vypíše na displej text, jenž uživatel zadá mezi závorky této funkce. Poté se pozice kurzoru přesune za poslední zobrazený znak. Předpokladem je, že zvolené znaky se nachází v CGRAM paměti displeje.

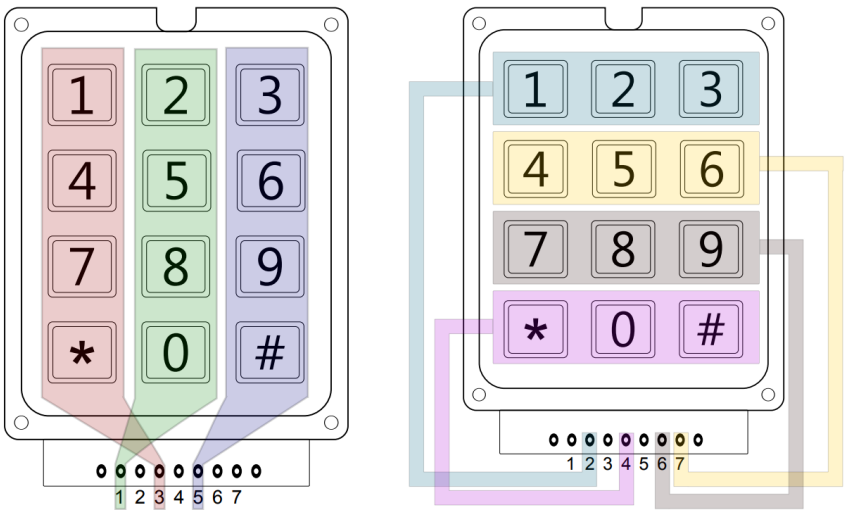
6.4 Maticová klávesnice

Na obrázku 16 je dvanácti tlačítková maticová klávesnice 3×4. Tlačítka na této klávesnici jsou uspořádána do třech sloupců a čtyřech řádků. Každému z těchto řádků a sloupců připadá jeden pin.



Obrázek 16: Maticová klávesnice 3×4 [14]

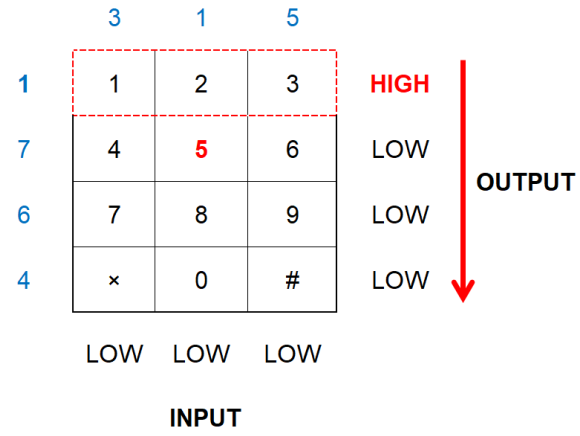
Seřadí-li se piny zleva doprava a očíslují se od jedné do sedmi, jak je tomu na obrázku 17, tak lze vidět, že piny odpovídající jednotlivým řádkům a sloupcům nejsou seřazeny postupně vedle sebe. Sloupcům odpovídají piny $1,3,5$ a řádkům piny $2,4,6,7$.



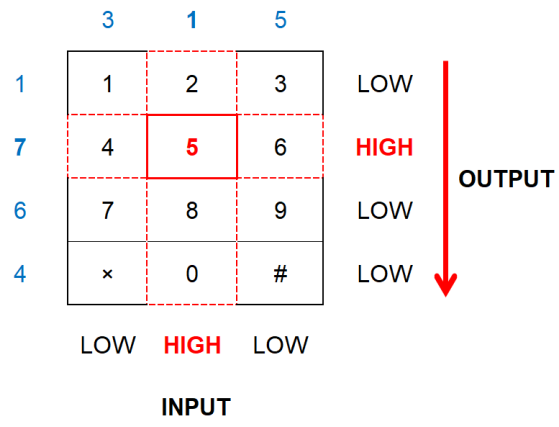
Obrázek 17: Vnitřní propojení klávesnice [14]

6.5 Klávesnice a Arduino

Co se týká klávesnice, je úkolem Arduina detekovat jaká klávesa byla stisknuta. K detekci stisknuté klávesy existuje řada procesů, které se však principiálně velmi podobají. Základem je vždy rozdělení jednotlivých řádků a sloupců na vstupy a výstupy. Na obrázcích 18 a 19 je ilustrován příklad detekce stisknu klávesy číslo 5. Sloupce jsou připojeny na vstupy Arduina a přes rezistor na GND. Řádky jsou připojeny na výstupy. Arduino postupně střídá hodnoty napětí na jednotlivých výstupech. Ve chvíli, kdy hodnota HIGH některého z výstupních pinů bude detekována na vstupu, je zřejmé, jaká klávesa byla stisknuta. Na uvedeném příkladu je detekován stisk klávesy číslo 5 hodnotou HIGH na pinech 1 a 7.



Obrázek 18: Detekce stisku tlačítka (a)



Obrázek 19: Detekce stisku tlačítka (b)

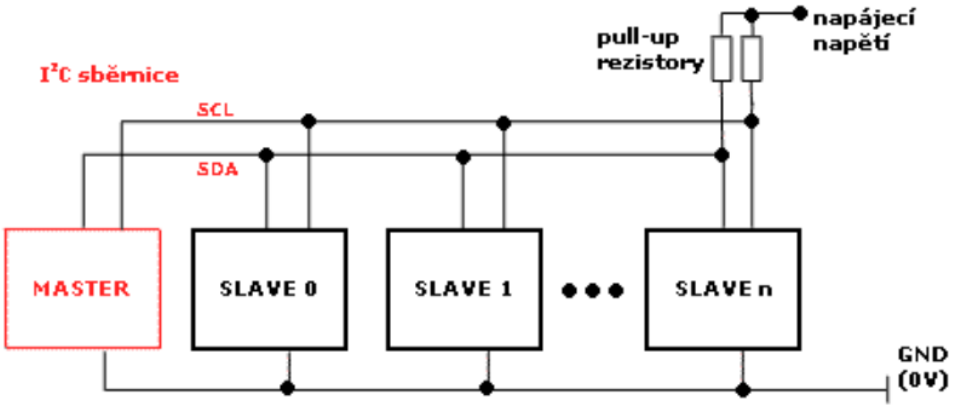
6.6 I²C expandér PCF8574

Modul PCF8574 umožňuje přes I²C sběrnici připojit k Arduinu dalších osm vstupně-výstupních pinů. Expandér dále obsahuje tři takzvané *jumpery*, které umožňují adresovat až osm dalších zařízení tohoto typu na jedné sběrnici. Pro připojení tohoto modulu (např. k Arduino) jsou z boku vyvedeny čtyři piny. Piny SDA a SCL slouží pro komunikaci po sériové sběrnici I²C. Piny VCC a GND jsou pro napájení modulu.

Obrázek 20: I²C expandér

6.6.1 I²C Sběrnice

Sběrnice slouží pro přenos dat mezi jednotlivými integrovanými obvody. I²C sběrnice potřebuje k přenosu dat pouze dva signálové vodiče. První z nich přenáší data (SDA – *serial data*), druhý přenáší hodinový signál (SCK – *seriál clock*). Řízení sběrnice obstarává zařízení, které je nastaveno do režimu MASTER. Toto zařízení při jakémkoli přenosu generuje hodinový signál. Ostatní zařízení jsou v režimu SLAVE, tato zařízení jsou řízená, musí mít definovanou adresu a nemohou sama od sebe spustit přenos dat. [15]



Obrázek 21: Schéma sběrnicového zapojení [16]

Na obrázku 21 je příklad zapojení zařízení na sběrnici I^2C. Zařízení typu MASTER by mohl být mikroprocesor, který přímá data z ostatních SLAVE zařízení. Všechna zařízení jsou propojena vodiči SDA, SCL a připojeny na společnou zem. Jelikož spínání na sběrnici probíhá logickou nulou, je potřeba zajistit, aby byla v klidovém stavu na signálních vodičích SDA a SCL logická jednička. Toho je docíleno připojením dvou *pull-up* rezistorů mezi signální vodiče a napájecí napětí.

Obrázek 22: Schéma přenosu dat po sběrnici [15]

Samotný přenos dat je zobrazen na obrázku 22. Na počátku je klidový stav zajištěn logickými jedničkami na obou vodičích. Přenos se zahájí takzvaným *start bitem,* tedy změnou logické úrovně SDA z logické jedničky na logickou nulu. Poté zařízení MASTER pošle na sběrnici sedmibitovou adresu a osmý bit, který určí, zda cílové zařízení má data vysílat nebo přijímat. Následně porovná každé SLAVE zařízení vyslanou adresu s tou svojí. V případě shody potvrdí dané zařízení, že se na sběrnici nachází. Potvrzení proběhne tak, že přijímač pošle zpět na zařízení MASTER logickou nulu. Následuje přenos dat, přenáší se osm bitů. Při přenosu dat se logická úroveň na SDA mění pouze, pokud je SCL v logické nule. Každým pulzem na SCL je přenesen jeden bit. Komunikace se ukončí *stop bitem*. Ten je vygenerován stejně jako *start bit* změnou úrovně SDA z logické jedničky na nulu, zatímco je na SCL logická jednička. [15] a [16]

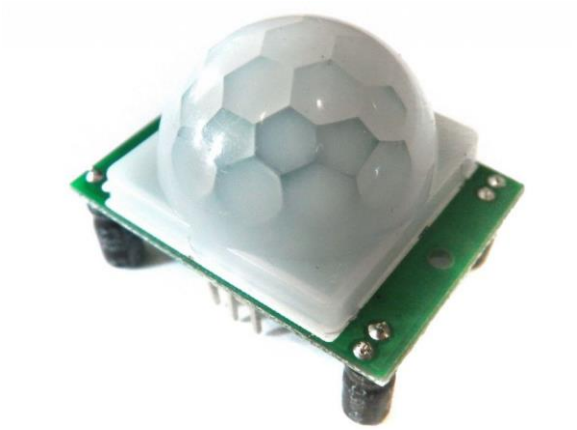
6.7 Teploměr a vlhkoměr DHT22

Na obrázku 23 je zobrazen DHT22 digitální senzor teploty a vlhkosti. Tento senzor je přizpůsobený pro připojení na Arduino desku. K jeho připojení postačí pouze tři vodiče. Dva pro napájení a jeden datový. Zapojený senzor měří teplotu v rozsahu od $-40\ [°C]$ do $80\ [°C]$ a relativní vlhkost vzduchu v plném rozsahu hodnot. Pro práci s tímto modulem je vhodné použít knihovnu *DHT.h*. [17]

6.8 Pohybové čidlo HC-SR501

Pohybové čidlo tohoto typu obsahuje pyroelektrický senzor, jenž detekuje infračervené záření z okolních zdrojů. Pokud se skokově změní hodnota dopadajícího infračerveného záření, například projde snímanou oblastí člověk, senzor jej detekuje, a dojde k sepnutí čidla. Podle potřeby lze na čidlu nastavit citlivost snímání a dobu sepnutí. Pro připojení čidla k Arduinu postačí tři vodiče. Dva pro napájení a třetí pro přenos dat.

Obrázek 23: DHT22 teploměr a vlhkoměr



Obrázek 24: Pohybové čidlo

7 Návrh a realizace modelu chytrého domu

Požadavkem na návrh chytrého domu je přehledné zapojení elektroinstalace – osvětlení, topení, čidlo teploty a alarm. Řídicí jednotkou pro veškeré chytré prvky v domě je vývojová deska Arduino.

Každá místnost má své vlastní osvětlení ovládané přepínačem připevněným ke stěně. U vchodu do domu je dále venkovní osvětlení zapínané stiskem tlačítka. Dobu, po kterou bude toto světlo svítit, než zhasne, lze ovlivnit opakovaným stiskem tlačítka. Jak vnitřní, tak venkovní osvětlení je hvězdicově propojeno s řídicí jednotkou a tvoří centralizovanou elektroinstalaci. Výjimku tvoří pouze osvětlení v místnosti čtyři, které bylo pro možnost porovnání zapojeno klasickým konvenčním způsobem.

Součástí výukového modelu chytrého domu je také bezpečnostní systém reagující na pohyb v domě. Skládá se z maticové klávesnice pro zadání hesla, PIR detektoru pohybu připevněnému na otáčivém kloubu, bzučáku, který sepne při detekci pohybu a grafického rozhraní, které tvoří LED display. Další součástí je vytápění domu. Jelikož je model domu otevřený a není tedy možné v něm teplo udržet, byla pro ilustraci vytápění jako zdroj tepla zvolena žárovka. V blízkosti žárovky, dostatečné pro zaznamenání změny teploty, je umístěn senzor teploty. Žárovka je spínaná přes tranzistor a regulovat teplotu je možné pomocí potenciometru.

Kromě lokálního ovládání, lze jednotlivé prvky ovládat i vzdáleně pomocí mobilního telefonu. K tomu slouží čip ESP8266 umožňující připojení Arduina k Wi-Fi síti. Když je řídicí systém i mobilní telefon připojen k internetu, lze pomocí aplikace Blynk navázat přes vzdálený server vzájemnou komunikaci. Pomocí mobilní aplikace lze dálkově ovládat osvětlení, regulovat topení, či sledovat aktuální teplotu a vlhkost v domě. Mobilní aplikace nás upozorní při sepnutí alarmu nebo výpadku proudu.

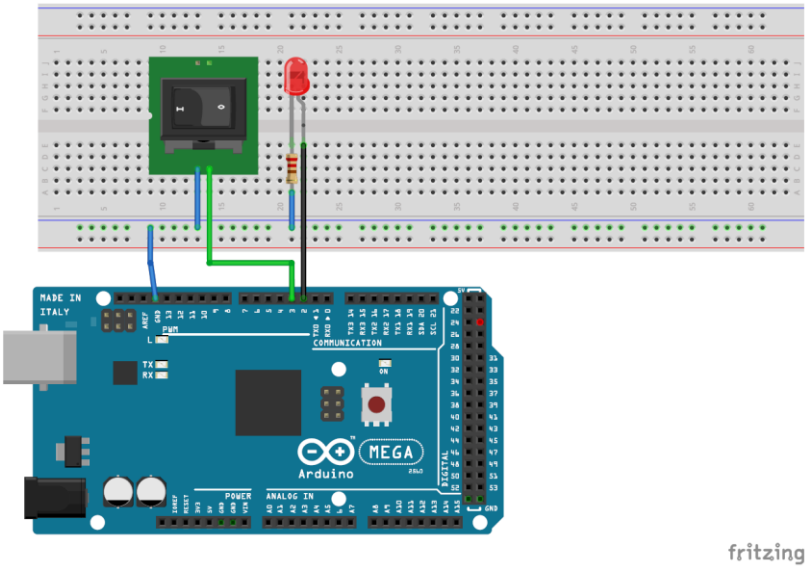
Při návrhu byl kladen důraz na fakt, že se jedná o výukový model chytrého domu, který má sloužit primárně jako výuková pomůcka pro studenty. Především se jedná o trasování vodičů, které nejsou schovány ve zdech, nebo lištách. Ale naopak jsou vedeny otevřeně a přehledně vedle sebe tak, aby bylo na první pohled zřejmé, k čemu jaký vodič je a odkud kam vede. Vodiče se dělí podle barev. Modrý je zemnící, černý je pracovní. A zbylé jsou signální.

7.1 Osvětlení

Veškeré osvětlení je realizováno pomocí LED diod. Místnosti jedna, dva a tři jsou opatřeny inteligentní elektroinstalací, v místnosti čtyři je světlo zapojeno konvenčně. Vnitřní osvětlení je ovládáno přepínači, venkovní tlačítkem.

7.1.1 Vnitřní osvětlení

V případě inteligentní elektroinstalace je třeba použít čtyři vodiče. Dva pro diodu a dva pro přepínač. Jedná se tedy o dva různé obvody. Obvod s diodou je řízený a obvod s přepínačem řídicí. Na obrázku 25 je znázorněno zapojení světla v prvním pokoji.

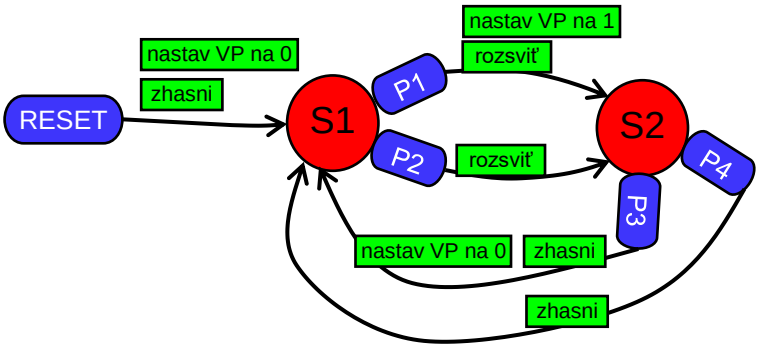


Obrázek 25: Schéma zapojení vnitřního osvětlení

Přepínač je přes modrý zemnící vodič připojen ke GND a přes zelený signální vodič k digitálnímu vstupu číslo 3. K tomuto vstupu je připojený interní *pull-up* rezistor, který na něm v nesepnutém stavu udržuje logickou jedničku. Pokud dojde k sepnutí přepínače, propojí se oba vodiče a vstupní hodnota na pinu 3 se změní na logickou nulu. Pokud Arduino vyhodnotí, že má rozsvítit diodu, nastaví na výstupní pin číslo 2 hodnotu HIGH. Tím pustí do obvodu s diodou napětí o hodnotě 5 $[V]$. Aby nedošlo tímto napětím k poškození LED diody, je k ní ještě sériově připojený rezistor.

Složitější situace nastane při spínaná diody mobilním telefonem. Přeskočí-li se celá problematika propojení mobilního telefonu s Arduinem, tak ve výsledku pouze

přibyl další přepínač, který je v tomto případě virtuální. Oba přepínače musí mít mezi sebou vzájemnou vazbu. Při sepnutí reálného přepínače není problém jeho aktuální stav převést na virtuální. Naopak při sepnutí virtuálního je převod stavu na reálný přepínač komplikovaný. Proto si musí řídicí jednotka pamatovat aktuální stav a dokázat spínat jak logickou jedničkou, tak logickou nulou. Pro ovládání vnitřního osvětlení se Arduino řídí následujícím stavovým automatem.



Obrázek 26: Stavový automat – vnitřní osvětlení

Tabulka 1: Popis stavového automatu vnitřního osvětlení

Zkratka	Popis
S1	Zhasnuto
S2	Rozsvíceno
P1	Změna hodnoty na RP
P2	Změna z log. 0 na log.1 na VP
P3	Změna hodnoty na RP
P4	Změna z log. 1 na log. 0 na VP

Ihned po zapnutí nebo restartování Arduina přejde automat do stavu S1, kdy je zhasnuto, a virtuální přepínač VP je v logické nule. K přechodu do stavu S2 je nutné splnit jednu ze dvou podmínek. První podmínkou P1 je změna hodnoty na reálném přepínači, ta je doprovázena následnou synchronizací s virtuálním přepínačem. Druhou podmínkou P2 je přepnutí virtuálního přepínače na logickou jedničku. Ve stavu S2 dioda svítí. Návrat do stavu S1 je obdobný. Buď podle P3 přepnutím reálného přepínače, nebo podle P4 přepnutí VP na logickou nulu. Níže je popsán zdrojový kód tohoto automatu.

```
// vnitřní osvětlení 1 (inteligentní elektroinstalace)
int prepinacPin1 = 3;              // reálný přepínač (pin)
int ledPin1 = 2;                   // LED dioda (pin)
boolean prepinacState1 = 0;        // proměnná která uchovává stav přepínače
boolean lastPrepinacState1 = 0;    // proměnná aktuálního stavu přepínače
boolean counterPrepinacState1 = 0;// brání znovunačtení – prepinacState1
int pinValue1 = 0;                 // proměnná stavu virt. přepínače (Blynk)
boolean sa1 = true;                // stav sa1 (S1 pro 1. pokoj) - zhasnuto
boolean sa2 = false;               // stav sa2 (S2 pro 1. pokoj) - rozsvíceno
BLYNK_WRITE(V1) {                  // načte data z V1 (virt. přepínač 1)
  pinValue1 = param.asInt();       // uloží data do proměnné pinValue1
}
void setup(){
  pinMode(ledPin1, OUTPUT);
  pinMode(prepinacPin1, INPUT_PULLUP);
  digitalWrite(ledPin1, LOW);
}
```

Zdrojový kód 9: Vnitřní osvětlení prvního pokoje – prvotní nastavení

V první části kódu jsou deklarovány všechny proměnné. Jednička za názvy proměnných značí, že se jedná o první pokoj. Pin 2 je napojený na diodu a nastavený jako OUTPUT. Pin 3 je napojený na přepínač a nastavený jako INPUT_PULLUP.

```
if (sa1) {                         // stav 1 - zhasnuto
  if (counterPrepinacState1 == 0)  // pokud není zabráněno načtení přepínače
  {                                // načti stav přepínače ↓
    prepinacState1 = digitalRead(prepinacPin1);
    counterPrepinacState1 = 1;     // zabraň znovunačtení a přepsání stavu
  }                                // načti poslední stav přepínače ↓
  lastPrepinacState1 = digitalRead(prepinacPin1);
  if (prepinacState1 != lastPrepinacState1) { // pokud se stavy nerovnají:
    digitalWrite(ledPin1, HIGH);   // rozsvítí LED
    counterPrepinacState1 = 0;     // povolí znovunačtení a přepsání stavu
    sa1 = false;                   // ukončí stav S1
    sa2 = true;                    // povolí přechod do stavu S2
    Blynk.virtualWrite(V1, HIGH);  // nastaví virtuální přepínač na HIGH
    pinValue1 = 1;                 // vysvětlení pod ukázkou kódu
  }                                // pokud byl stisknut virt. přepínač: ↓
  if (pinValue1 == 1 && sa1 == true) {
    digitalWrite(ledPin1, HIGH);   // rozsvítí LED
    counterPrepinacState1 = 0;     // povolí znovunačtení a přepsání stavu
    sa1 = false;                   // ukončí stav S1
    sa2 = true;                    // povolí přechod do stavu S2
} }
```

Zdrojový kód 10: Vnitřní osvětlení prvního pokoje – stav S1 (sa1)

Druhá část kódu je již uvnitř smyčky *loop()*. Popisuje stav S1 (*sa1*) a přechod do stavu S2 (*sa2*). Algoritmus je jednoduchý. Do proměnné *prepinacState1* se načte poloha přepínače. Zabrání se přepisování této proměnné. V každém cyklu se porovná uložená poloha přepínače s tou aktuální. Pokud Arduino vyhodnotí změnu na přepínači, dá povel pro rozsvícení diody a přejde do stavu S2.

Do proměnné *pinValue1* se ukládá stav virtuálního přepínače, jeho stiskem dojde ke změně hodnoty proměnné z 0 na 1, nebo naopak. Ke změně hodnoty ale nedojde, pokud se mění stav přepínače pomocí příkazu *Blynk.virtualWrite()*. Proto je nutné za tímto příkazem ještě nastavit proměnnou *pinValue1* na hodnotu odpovídající aktuálnímu stavu virtuálního přepínače. Třetí část kódu popisuje přechod nazpátek do stavu S1. Principiálně se ale neliší od druhé části.

```
if (sa2)                            // stav - 2 rozsvíceno
{
  if (counterPrepinacState1 == 0)   // pokud není zabráněno načtení přepínače
  {                                 // načti stav přepínače ↓
    prepinacState1 = digitalRead(prepinacPin1);
    counterPrepinacState1 = 1;      // zabraň znovunačtení a přepsání stavu
  }                                 // načti poslední stav přepínače ↓
  lastPrepinacState1 = digitalRead(prepinacPin1);
  if (prepinacState1 != lastPrepinacState1) // pokud se stavy nerovnají:
  {
    digitalWrite(ledPin1, LOW);     // zhasne LED
    counterPrepinacState1 = 0;      // povolí znovunačtení a přepsání stavu
    sa1 = true;                     // povolí přechod do stavu S1
    sa2 = false;                    // ukončí stav S2
    Blynk.virtualWrite(V1, LOW);    // nastaví virtuální přepínač na LOW
    pinValue1 = 0;                  // stav virtuálního přepínače
  }                                 // pokud byl stisknut virt. přepínač: ↓
  if (pinValue1 == 0 && sa2 == true)
  {
    digitalWrite(ledPin1, LOW);     // zhasne LED
    counterPrepinacState1 = 0;      // povolí znovunačtení a přepsání stavu
    sa1 = true;                     // povolí přechod do stavu S1
    sa2 = false;                    // ukončí stav S2
  }
}
```

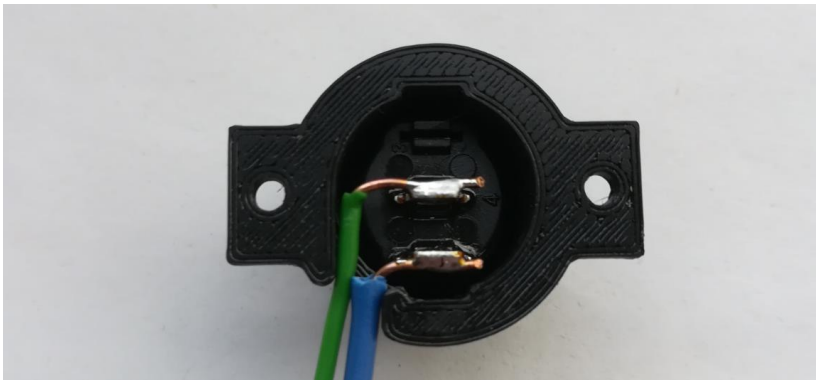
Zdrojový kód 11: Vnitřní osvětlení prvního pokoje – stav S2 (sa2)

Pro upevnění přepínačů na stěny domu byly navrženy a vytisknuty tyto držáky. Vnitřní průměr držáku odpovídá profilu přepínače. Drážka na vnitřním průměru zaručuje správné nasazení přepínače a zamezuje jeho následnému pootočení. Spodní

Obrázek 27: Držák přepínače

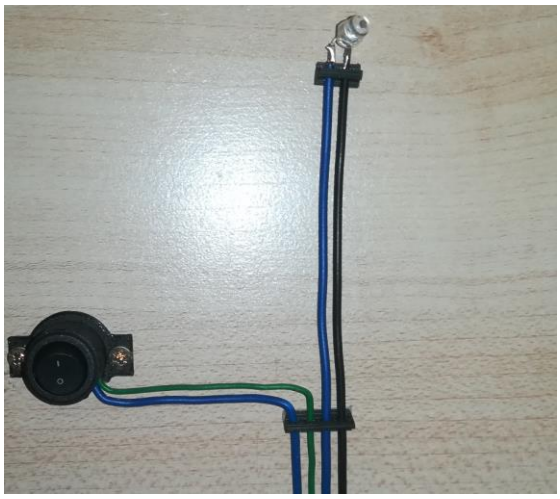
Obrázek 28: Držák přepínače (detail)

lišta se dvěma otvory slouží k přišroubování držáku na stěnu. A malý otvor dole k přívodu vodičů ke konektorům přepínače. Přívod vodičů tímto otvorem a napájení na konektory je zobrazeno na obrázku 29.



Obrázek 29: Napájení vodičů ke konektorům přepínače

Na obrázku 30 je inteligentní zapojení osvětlení, kdy přepínač a světlo mají každý svůj obvod. Na obrázku 31 je zobrazeno osvětlení ve čtvrté místnosti domu. Jedná se o konvenční zapojení pomocí dvou vodičů. Přepínač je zapojený v sérii s LED diodou, přepnutím na přímo spojí nebo rozpojí obvod.

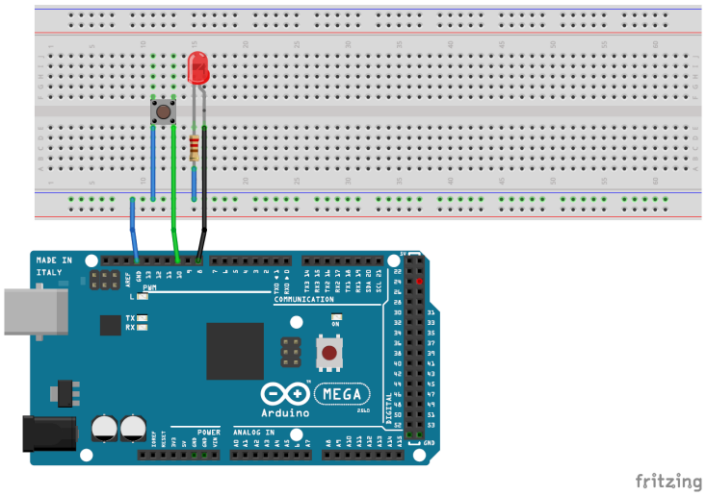


Obrázek 30: Inteligentní elektroinstalace vnitřního osvětlení

Obrázek 31: Konvenční elektroinstalace vnitřního osvětlení

7.1.2 Venkovní osvětlení

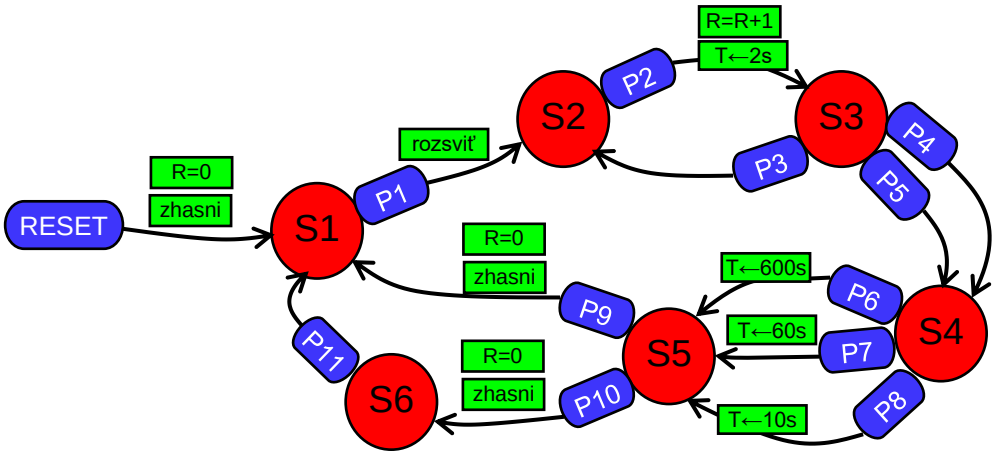
Venkovní osvětlení je umístěno u vchodu do domu. Pro ovládání je v tomto případě použito tlačítko. Požadavkem na toto osvětlení je, aby se po určité době samo zhaslo. Dobu, po kterou bude světlo svítiti, lze nastavit opakovaným stiskem tlačítka. Celkem je možné nastavit tři různé sekvence. Krátkou na deset vteřin, středně dlouhou na minutu a dlouhou na deset minut.



Obrázek 32: Schéma zapojení venkovního osvětlení

Venkovní osvětlení je také rozděleno na dva samostatné obvody. Jeden pro tlačítko, druhý pro LED diodu. Tlačítko je připojeno na vstupní pin číslo 10. K tomuto vstupu je připojený interní *pull-up* rezistor, který na něm v nesepnutém stavu

udržuje logickou jedničku. Stiskem tlačítka je zaznamenána na vstupu logická nula. Dioda je připojena na výstupní pin 8. Venkovní osvětlení je řízeno následujícím stavovým automatem.



Obrázek 33: Stavový automat – venkovní osvětlení

Tabulka 2: Popis stavového automatu venkovního osvětlení

Zkratka	Popis
S1, S6	Zhasnuto
S2, S3, S4, S5	Rozsvíceno
P1, P3, P10	Stisk tlačítka
P2, P11	Uvolnění tlačítka
P4, P9	Doběhnutí časovače
P5, P8	R = 3
P6	R = 1
P7	R = 2

Po zapnutí nebo restartování Arduina přejde automat do stavu S1. V tomto stavu je zhasnuto a proměnná R je rovna nule. Do této proměnné se bude ukládat počet stisků tlačítka, reálného i virtuálního. Prvním stiskem se rozsvítí světlo a tím se přejde do stavu S2. Uvolněním tlačítka se změní hodnota proměnné na R+1, nastaví se časovač na dvě vteřiny a přejde se do stavu S3. Pokud bude ještě před doběhnutím časovače znovu stisknuto tlačítko, automat se navrátí do stavu S2. Po uvolnění opět naroste hodnota proměnné o R+1 a nastaví se časovač. Jsou dvě možnosti, jak přejít do stavu S4 první je doběhnutí časovače, druhou je nárust proměnné R na maximální hodnotu tři. Podle hodnoty proměnné se nastaví čas, po jakou dobu bude světlo svítit.

Ve stavu S5 je již nastaveno, jak dlouho bude světlo svítit, a čeká se na doběhnutí časovače. Pokud je potřeba světlo okamžitě vypnout, stačí znovu stisknout tlačítko. Po jeho uvolnění se automat navrátí do stavu S1. Podmínka P11 je nutná proto, aby automat nepřeskočil ze stavu S6 ihned do stavu S2. Níže je popsán zdrojový kód tohoto automatu.

```
// venkovní osvětlení (inteligentní elektroinstalace)
int buttonPin = 8;        // reálné tlačítko (pin)
int ledPin4 = 10;         // LED dioda (pin)
boolean buttonState = 0; // proměnná stavu tlačítka
int pinValue4 = 0;        // proměnná stavu virtuálního tlačítka (Blynk)
int R = 0;                // proměnná počtu stisků tlačítka R(1-3)
unsigned long time1;      // časovač
// stavy pro automat venkovního osvětlení
boolean sd1 = true;       // (S1)zhasnuto a nesepnuté tlačítko
boolean sd2 = false;      // (S2)rozsvíceno a sepnuté tlačítko
boolean sd3 = false;      // (S3)rozsvíceno a uvolněné tlačítko, nezn. R(1-3)
boolean sd4 = false;      // (S4)rozsvíceno, známe R, nezn. t(10,60,600 sec)
boolean sd5 = false;      // (S5)rozsvíceno a známe t
boolean sd6 = false;      // (S6)zhasnuto a sepnuté tlačítko
BLYNK_WRITE(V4) {         // načte data z V4 (virtuální tlačítko)
  pinValue4 = param.asInt();
}
void setup() {
  pinMode(ledPin4, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  digitalWrite(ledPin4, LOW);
}
```

Zdrojový kód 12: Venkovní osvětlení – prvotní nastavení

V úvodní části jsou deklarovány všechny proměnné. Pin 10 je napojený na diodu a pin 8 na tlačítko. Další ukázky kódu patří k jednotlivým stavům a jsou již součástí nekonečné smyčky *loop()*.

```
if (sd1) {                                    // stav 1 - zhasnuto
  buttonState = digitalRead(buttonPin);       // načte stav tlačítka
  if (!buttonState == HIGH || pinValue4 == 1) // stisk tlačítka:
  {
    digitalWrite(ledPin4, HIGH);              // rozsvítí led
    sd1 = false;                              // ukončí stav S1
    sd2 = true;                               // povolí přechod do stavu S2
} }
```

Zdrojový kód 13: Venkovní osvětlení – stav S1 (sd1)

V prvním stavu se čeká na stisk tlačítka. Před konstantou *buttonState* je vykřičník, ten neguje porovnávací operátor. Je zde z důvodu, že tlačítko je napojeno na interní *pull-up* rezistor a spínání tedy probíhá logickou nulou. Podmínka je splněna stiskem reálného nebo virtuálního tlačítka.

```
if (sd2)                                          // stav 2 - rozsvíceno
{
  buttonState = digitalRead(buttonPin);           // načte stav tlačítka
  if (!buttonState == LOW && pinValue4 == 0)      // uvolnění tlačítka:
  {
    R = R + 1;                                    // čítač stisků
    time1 = millis() + 2000;                      // nastaví časovač na 2 sec
    sd2 = false;                                  // ukončí stav S2
    sd3 = true;                                   // povolí přechod do stavu S3
  }
}
```

Zdrojový kód 14: Venkovní osvětlení – stav S2 (sd2)

Ve druhém stavu se čeká na uvolnění tlačítka. Poté se přičte jeden stisk a nastaví časovač.

```
if (sd3)                                          // stav 3 - rozsvíceno
{
  if (millis() > time1 || R == 3)                 // pokud R = 3 nebo doběhl
  {                                               // časovač, tak:
    sd3 = false;                                  // ukončí stav S3
    sd4 = true;                                   // povolí přechod do stavu S4
  }
  else                                            // jinak:
  {
    buttonState = digitalRead(buttonPin);         // načte stav tlačítka
    if (!buttonState == HIGH || pinValue4 == 1)// stisk tlačítka:
    {
      sd3 = false;                                // ukončí stav S3
      sd2 = true;                                 // navrátí stav S2
    }
  }
}
```

Zdrojový kód 15: Venkovní osvětlení – stav S3 (sd3)

Pokud aktuální čas přeběhne hodnotu *time1* nebo počet stisků je roven třem, přejde automat do stavu S4. Další možností je stisk tlačítka, což navrátí stav S2.

```
if (sd4) {                                    // stav 4 - rozsvíceno
  if (R == 1) {                               // pokud R = 1, tak:
    time1 = millis() + 10000;                 // nastaví časovač na 10 ses
  }
  else if (R == 2) {                          // pokud R = 2, tak:
    time1 = millis() + 60000;                 // nastaví časovač na 60 sec
  }
  Else {                                      // jinak:
    time1 = millis() + 600000;                // nastaví časovač na 600 ses
  }
  sd4 = false;                                // ukončí stav S4
  sd5 = true;                                 // povolí přechod do stavu S5
}
```

Zdrojový kód 16: Venkovní osvětlení – stav S4 (sd4)

Uvnitř čtvrtého stavu se podle počtu stisků nastaví časovač, po jakou dobu bude světlo svítit.

```
if (sd5) {                                    // stav 5 - rozsvíceno
  buttonState = digitalRead(buttonPin);       // načte stav tlačítka
  if (millis() > time1) {                     // pokud doběhne časovač:
    digitalWrite(ledPin4, LOW);               // zhasne led
    R = 0;                                    // nastaví R na 0
    sd5 = false;                              // ukončí stav S5
    sd1 = true;                               // navrátí stav S1
  }
  if (!buttonState == HIGH || pinValue4 == 1) {// stisk tlačítka:
    digitalWrite(ledPin4, LOW);               // zhasne led
    R = 0;                                    // nastaví R na 0
    sd5 = false;                              // ukončí stav S5
    sd6 = true;                               // povolí přechod do stavu S5
} }
```

Zdrojový kód 17: Venkovní osvětlení – stav S5 (sd5)

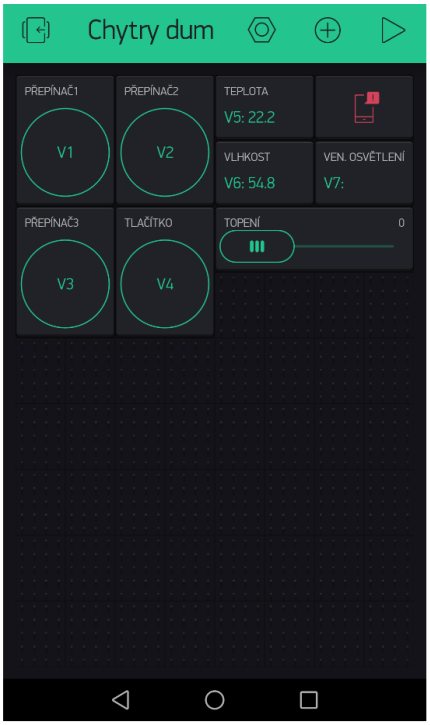
Při doběhnutí časovače zhasne dioda a přejde se do stavu S1. Při stisku tlačítka zhasne dioda a přejde se do stavu S6.

```
if (sd6) {                                    // stav 6 - zhasnuto
  buttonState = digitalRead(buttonPin);       // načte stav tlačítka
  if (!buttonState == LOW && pinValue4 == 0) { // uvolnění tlačítka:
    sd6 = false;                              // ukončí stav S6
    sd1 = true;                               // navrátí stav S1
} }
```

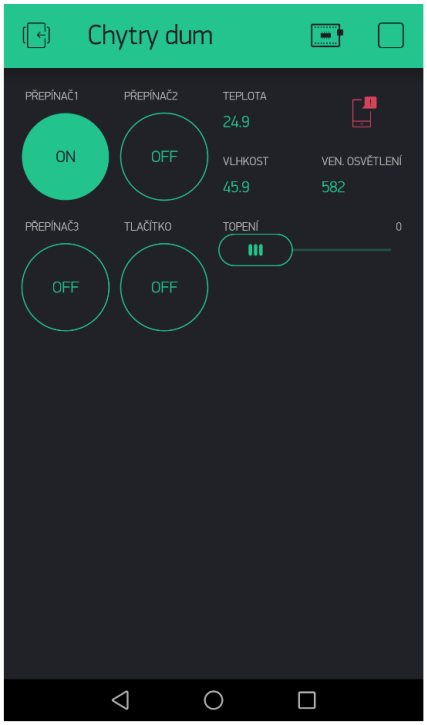
Zdrojový kód 18: Venkovní osvětlení – stav S6 (sd6)

7.1.3 Ovládání osvětlení mobilním telefonem

Dálkové řízení chytrého domu mobilním telefonem je zprostředkováno pomocí aplikace Blynk. V levém horním rohu jsou tři virtuální přepínače pro vnitřní osvětlení a jedno virtuální tlačítko pro venkovní osvětlení. Vpravo uprostřed je rámeček s virtuálním pinem V7, v tomto poli je časovač venkovního osvětlení. Na obrázku 34 je pracovní prostředí aplikace Blynk, zde se kompletují jednotlivé ovládací prvky v jeden funkční celek. Funkčním celkem může být uživatelské prostředí zobrazené na obrázku 35. Zde je v tuto chvíli rozsvíceno světlo v místnosti jedna a venkovní světlo, které, pokud nedojde k přerušení, bude svítit ještě dalších 582 vteřin.



Obrázek 34: Blynk, pracovní prostředí



Obrázek 35: Blynk, uživatelské prostředí

Na rozdíl od reálných přepínačů, kde se podle potřeby spíná jak logickou jedničkou, tak nulou, je u virtuálních zajištěna synchronizace s aktuálním stavem světla. Je-li rozsvíceno, je virtuální přepínač vždy sepnut (ON), a v opačném případě vždy odepnut (OFF). Je tedy z mobilní aplikace vždy zřejmé, jaké světlo svítí a jaké ne. Virtuální tlačítko se stejně jako reálné po stisknutí a následném uvolnění vrací zpátky do původního stavu.

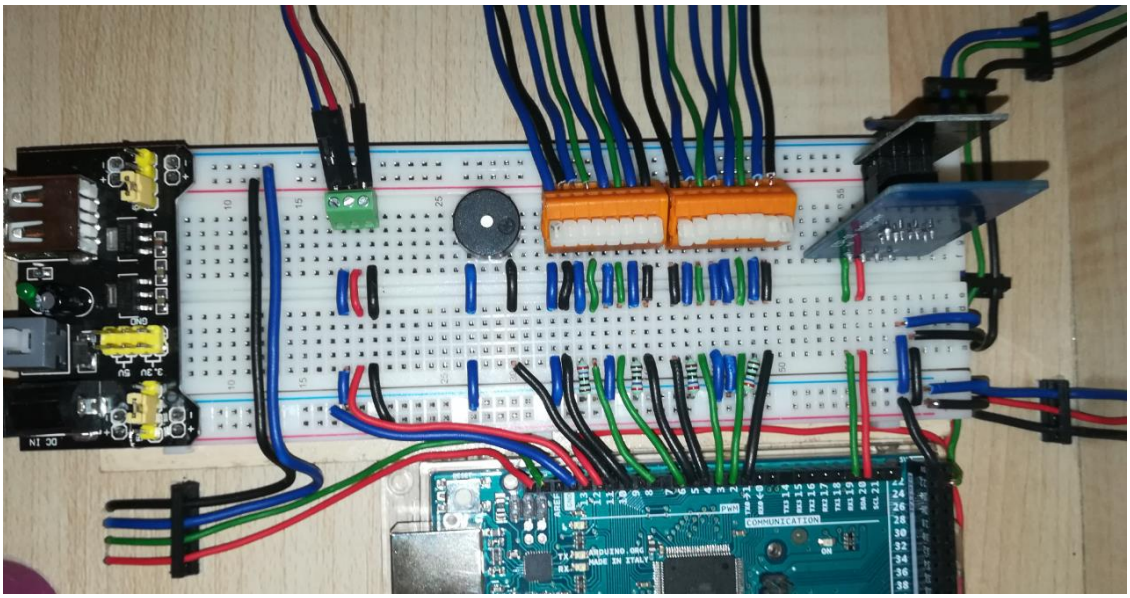
7.1.4 Celkový přehled zapojení osvětlení

V předchozích kapitolách bylo popsáno osvětlení prvního pokoje a venkovní osvětlení, v tabulce 3 je kompletní seznam zapojení veškerého osvětlení v domě.

Tabulka 3: Osvětlení – přehled zapojení

Pin	Funkce	Pin	Funkce
2	Světlo – první pokoj	10	Světlo – venkovní osvětlení
3	Přepínač – první pokoj	V1	Virtuální přepínač–první pokoj
4	Přepínač – druhý pokoj	V2	Virtuální přepínač – druhý pokoj
5	Světlo – druhý pokoj	V3	Virtuální přepínač – třetí pokoj
6	Světlo – třetí pokoj	V4	Virtuální tlačítko – venkovní osvětlení
7	Přepínač – třetí pokoj	VCC	Konvenční osvětlení
8	Tlačítko – venkovní osvětlení	GND	Konvenční osvětlení

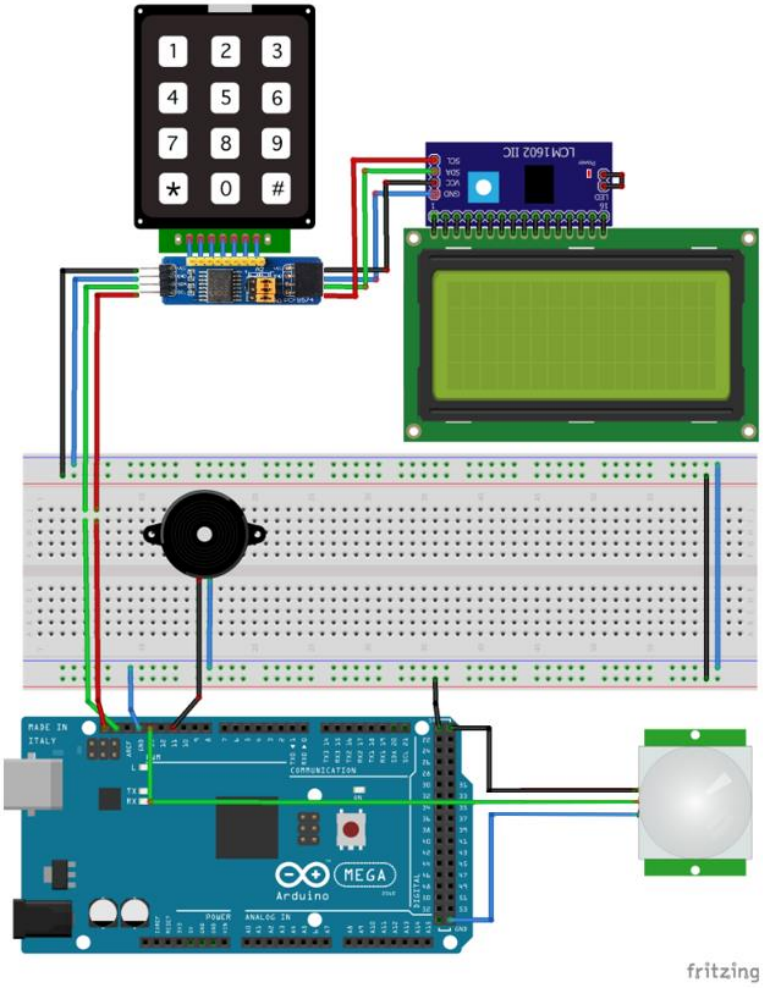
V nepájivém elektrickém poli jsou zapojeny dvě oranžové svorkovnice. Z jedné strany jsou připojené k Arduinu a z druhé vyvádí vodiče k jednotlivým světlům. První čtveřice vodičů zprava vede do prvního pokoje, druhá do druhého pokoje, třetí do třetího a čtvrtá k venkovnímu osvětlení. Konvenční osvětlení je připojeno pouze k VCC a GND.



Obrázek 36: Zapojení osvětlení

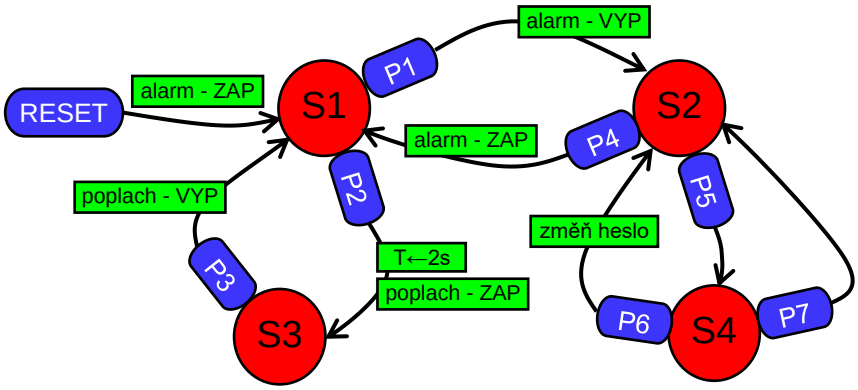
7.2 Bezpečnostní systém

Bezpečnostní systém detekuje pohyb v domě. Pokud je alarm aktivní, tak při detekci pohybu spustí poplach a informuje uživatele přes mobilní aplikaci. Alarm lze aktivovat nebo deaktivovat zadáním hesla. Na obrázku 37 je schéma zapojení bezpečnostního systému. Klávesnice a displej jsou s Arduinem propojeny přes I^2C sběrnici. To velmi redukuje počet vodičů. Šestnáct vodičů z displeje a sedm z klávesnice jsou nahrazeny čtyřmi vodiči. Dva jsou komunikační, červený SCL a zelený SDA. Druhé dva jsou napájecí, černý je připojen k VCC a modrý ke GND. Digitální pin číslo 13 je nastaven jako vstup a přijímá informace s PIR detektoru pohybu. Digitální pin 11 je výstupní, ten při detekci pohybu přívodem napětí $5\,[V]$ spustí bzučák.



Obrázek 37: Schéma zapojení bezpečnostního systému

Princip bezpečnostního systému lze schematicky popsat pomocí stavového automatu, který je ve zjednodušené podobě zobrazen na obrázku 38. Po spuštění systému je alarm aktivní. Pokud čidlo detekuje pohyb, přejde se do stavu S3, kdy je nastaven časovač a sepnutý bzučák. Po doběhnutí časovače přejde automat do původního stavu. Zadáním správného hesla je alarm deaktivován – stav S2. Zde je možné zažádat o změnu hesla, tím se přejde do menu pro změnu hesla – stav S4. Potvrzením starého hesla a zadáním nového je heslo změněno a navrácen stav S2. Zadáním aktuálního hesla lze alarm znovu aktivovat, tím se automat vrátí do stavu S1.



Obrázek 38: Stavový automat – bezpečnostní systém

Tabulka 4: Popis stavového automatu bezpečnostního systému

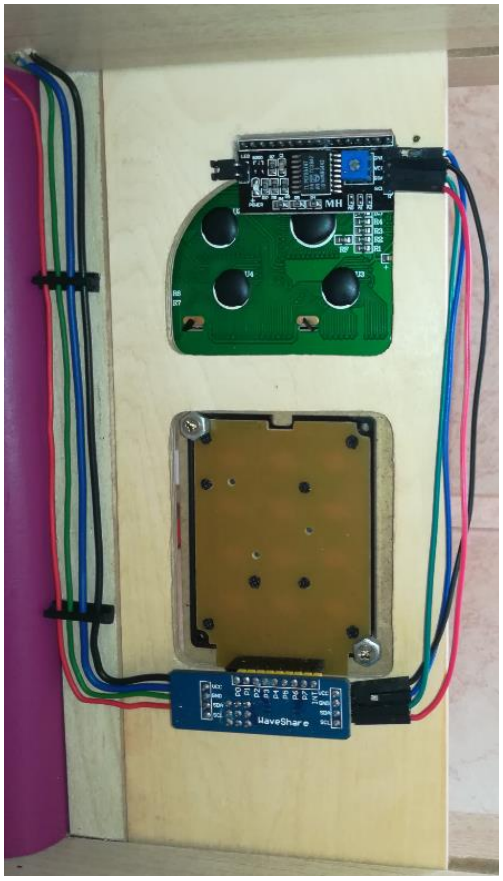
Zkratka	Popis	Zkratka	Popis
S1	Alarm aktivní	P2	Sepnuté čidlo pohybu
S2	Alarm neaktivní	P3	Doběhnutí časovače
S3	Poplach	P5	Požadavek na změnu hesla
S4	Změna hesla	P6	Zadání nového hesla (potvrzení starého)
P1, P4	Zadání hesla	P7	Ponechání starého hesla

7.2.1 Instalace bezpečnostního systému do modelu chytrého domu

Při instalaci bezpečnostního systému bylo potřeba navrhnout rozmístění a mechanické uchycení jednotlivých prvků, tak aby celé zapojení bylo přehledné a ovládání jednoduché. Na obrázku 39 je zapojení displeje a klávesnice z pohledu před vchodem do domu. Klávesnice je připevněna k tenké liště a ta je společně

s displejem přišroubována k boční zdi domu. Na následujícím obrázku jsou obě součásti zobrazeny z opačné strany. Na piny klávesnice je připájen I^2C expandér. K displeji byl expandér připojen již od výrobce. Na pravé straně jsou čtyřmi vodiči propojeny oba expandéry a na levé jsou vodiče vedoucí k Arduinu.

Obrázek 39: Upevnění displeje a klávesnice zepředu



Obrázek 40: Upevnění displeje a klávesnice zezadu

Dalším komponentem je PIR pohybové čidlo, pro které byla navržena a vytisknuta speciální příchytka. Ta je opatřena kulovým kloubem pro možnost natočení a ohybu čidla do požadované polohy. Kloub je napevno spojen s objímkou, která slouží k uchycení čidla. Druhá část držáku obsahuje jamku, do které přesně kloub zapadá. Po nastavení drží oba díly v dané poloze. To je zajištěno hrubou strukturou, která vznikla při nanášení jednotlivých vrstev při 3D tisku. Upevnění čidla k modelu domu je znázorněno na obrázku 42. Spodní rovinná plocha držáku je přiložena ke stropu domu a připevněna šroubem. K čidlu jsou přivedeny pletené kabely, které jsou velmi tvárné a umožňují snadné otočení čidla až o $360°$. Poslední součástí bezpečnostního systému je bzučák, ten je zapojen pouze do nepájivého kontaktního pole.

Obrázek 41: Příchytka PIR čidla

Obrázek 42: Upevnění PIR čidla

7.2.2 Ukázka funkčnosti bezpečnostního systému

Po zapnutí Arduina je alarm aktivní, k deaktivaci je potřeba zadat heslo. Tím je čtyřmístný číselný kód. Jak lze vidět na obrázku 43, při zadávání kódu se na displeji nezobrazují čísla, ty jsou nahrazeny hvězdičkami. Na klávesnici jsou krom čísel i dva znaky, hvězdičkou se potvrdí zadaný kód a křížkem se vymaže. Pokud se správné heslo potvrdí stiskem hvězdičky, alarm se deaktivuje a na displeji se zobrazí hlavní menu.

Obrázek 43: Alarm – menu pro zadávání hesla

V hlavním menu jsou využity všechny řádky displeje. První dva říkají, že stiskem hvězdičky se zobrazí menu pro aktivaci alarmu a stiskem křížku menu pro změnu hesla. Poslední dva řádky jsou využity pro zobrazení aktuální teploty a relativní vlhkosti v domě.

Obrázek 44: Alarm – hlavní menu

Na dalším obrázku je zobrazeno menu pro změnu hesla. Nejdříve je potřeba zadat aktuální heslo a poté heslo nové. Původní heslo je zakryto hvězdičkami, nové je zobrazeno číselně. Změna se potvrdí hvězdičkou. Stiskem křížku se vymažou všechny zadaná čísla, opětovný stisk křížku navrátí systém do hlavního menu bez změny hesla.

Obrázek 45: Alarm – menu pro změnu hesla

Jelikož se jedná o výukový model, tak je vhodné, aby se po restartování Arduina vždy obnovilo původní heslo. To je *1234*. Trvale je heslo možné změnit pouze ve zdrojovém kódu. Po minutě nečinnosti se vypne podsvícení displeje, znovu se zapne stiskem libovolné klávesy.

7.2.3 Zdrojový kód bezpečnostního systému

Zdrojový kód bezpečnostního systému je poměrně obsáhlý. Proto byly pro popis vybrány úseky kódu, které nejlépe demonstrují funkci systému. Kompletní kód je obsažen v přiloženém CD k diplomové práci. Na ukázce 19 je popsáno úvodní nastavení klávesnice a displeje. U klávesnice je nastaven počet řádků a sloupců, ty jsou přiřazeny k jednotlivým pinům. Poté je vytvořeno znakové pole a nastavení adresy pro sběrnicovou komunikaci. U displeje je nastaven počet řádků, sloupců a adresa.

```
#include <PCF8574.h>                    // načte knihovnu PCF expandéru (I2C)
#include <Keypad_I2C.h>                 // načte knihovnu klávesnice (I2C)
#include <LiquidCrystal_I2C.h>          // načte knihovnu displeje (I2C)
#define I2CADDR 0x38                    // adresa I2C klávesnice je 0×38
const byte ROWS = 4;                    // klávesnice má 4 řádky
const byte COLS = 3;                    // klávesnice má 3 sloupce
char keys[ROWS][COLS] = {               // nastaví znakové pole klávesnice
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}};

byte rowPins[ROWS] = {5, 0, 1, 3};      // přidělení řádků k pinům (PCF8574)
byte colPins[COLS] = {4, 6, 2};         // přidělení sloupců k pinům (PCF8574)
Keypad_I2C kpd(makeKeymap(keys), rowPins, colPins, ROWS, COLS, I2CADDR,
PCF8574);                               // spojení všech parametrů klávesnice
LiquidCrystal_I2C lcd(0x27, 20, 4);     // nastavení adresy I2C (0x27), počet
                                        // řádků a sloupců, LCD (20x4)
```

Zdrojový kód 19: Bezpečnostní systém – klávesnice a displej

Na ukázce zdrojového kódu 20 je zobrazen stav, kdy je alarm aktivní. Pokud senzor zaznamená pohyb, spustí se poplach a odešle se hlášení do mobilní aplikace. Algoritmus se neustále ptá na to, jestli je heslo ověřeno. Pokud ano, tak se deaktivuje alarm a nastaví se vše, co je nutné k přechodu do dalšího stavu, tím je hlavní menu. Heslo ověřuje funkce s názvem *enterPassword()*, ta se nachází naspod ukázky kódu, při jejím načtení, přesměruje systém k ověřujícímu algoritmu. Ten je popsán na následující ukázce zdrojového kódu.

```
if (alarmActivated == true)             // pokud je alarm aktivován, tak:
{
  cidloAlarm = digitalRead(senzorPin); // načte stav pohybového senzoru
  digitalWrite(bzucak, cidloAlarm);    // při pohybu spustí poplach
  if (cidloAlarm == 1 && millis() > timeCidloAlarm)
  {                                     // pokud je detekován pohyb, tak:
    Blynk.notify("Pozor Alarm!");       // pošle informaci na mobilní telefon
    timeCidloAlarm = millis() + 5000;  // nastaví časovač
  }
  if (hesloOvereno)                     // pokud je heslo ověřeno, tak:
  {
    alarmActivated = false;             // deaktivuje alarm
    lcd.clear();                        // vyčistí displej
    lcd.setCursor(0, 0);                // nastaví kurzor do polohy (0,0)
    lcd.print(" *** alarm off *** ");  // zobrazí se na displeji text
    delay(2000);                        // počká dvě vteřiny
    lcd.clear();                        // vyčistí displej
    lcd.setCursor(0, 0);                // nastaví kurzor do polohy (0,0)
    lcd.print("* aktivace alarmu ");    // zobrazí na daném místě text
    lcd.setCursor(0, 1);                // nastaví kurzor do polohy (0,1)
    lcd.print("# zmena hesla ");        // zobrazí na daném místě text
    lcd.setCursor(0, 2);                // nastaví kurzor do polohy (0,2)
    lcd.print("teplota=");              // zobrazí na daném místě text
    lcd.setCursor(9, 2);                // nastaví kurzor do polohy (9,2)
    lcd.print(teplota);                 // na daném místě zobrazí teplotu
    lcd.print(" C ");                   // pokračuje v textu za teplotou
    lcd.setCursor(0, 3);                // nastaví kurzor do polohy (0,3)
    lcd.print("vlhkost=");              // zobrazí na daném místě text
    lcd.setCursor(9, 3);                // nastaví kurzor do polohy (9,3)
    lcd.print(vlhkost);                 // na daném místě zobrazí vlhkost
    lcd.print(" % ");                   // pokračuje v textu za vlhkostí
    modeMenu = true;                    // umožní přechod do hlavního menu
  }
  else                                  // pokud heslo není otevřeno, tak:
  {
    enterPassword();                    // funkce, která ověřuje heslo
  }
}
```

Zdrojový kód 20: Bezpečnostní systém – aktivní alarm

Ověření hesla začíná nastavením prostředí pro jeho zadávání. Proměnná *k* nastavuje číslo sloupce, na jakém se bude kurzor nacházet. Proměnná *zadHeslo* je datového typu *string*, což je řada, která uchovává posloupnost znaků zadaného hesla. Následuje detekce stisku klávesy, pokud je stisknuta klávesa s číselnou hodnotou, uloží se dané číslo do proměnné a posune se kurzor. Pokud je stisknuta hvězdička

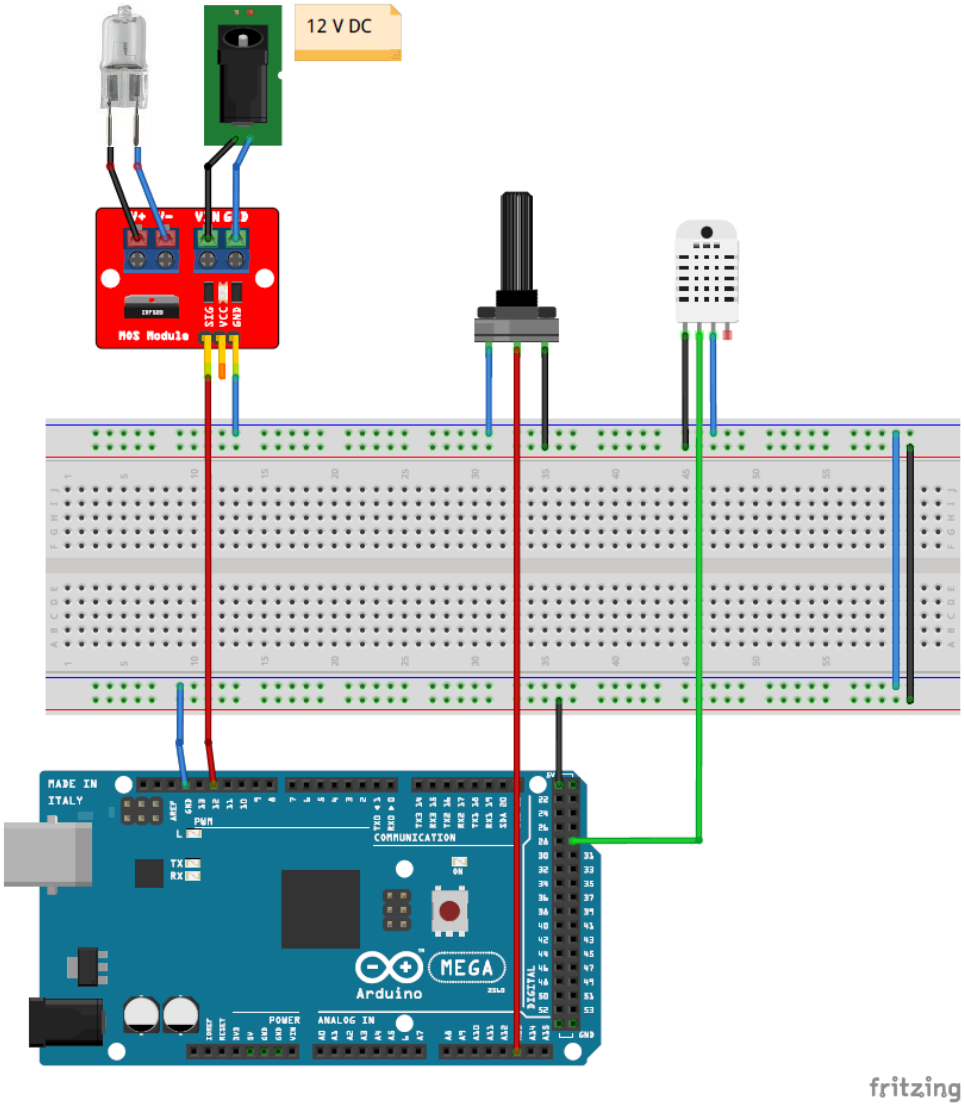
a heslo se shoduje, tak algoritmus vrátí informaci, že je heslo ověřeno. Pokud se heslo neshoduje nebo byl stisknut křížek, obnoví se menu pro zadávání hesla.

```
void enterPassword()
{
  if (nastaveniProZadaniHesla == 0)    // nastaví menu pro zadávání hesla:
  {
    k = 5;                             // heslo se zadává od 6. sloupce
    zadHeslo = "";                     // smaže zadané heslo
    lcd.clear();                       // vyčistí displej
    lcd.setCursor(0, 0);               // nastaví kurzor do polohy (0,0)
    lcd.print(" *** ALARM *** ");      // zobrazí se na displeji text
    lcd.setCursor(0, 1);               // nastaví kurzor do polohy (0,1)
    lcd.print("Pass>");                // zobrazí se na displeji text
    nastaveniProZadaniHesla = 1;       // brání znovunastavení tohoto menu
  }
  if (keypressed != NO_KEY)            // pokud bylo stisknuté tlačítko:
  {
    if (keypressed == '0' || keypressed == '1' || keypressed == '2' ||
        keypressed == '3' || keypressed == '4' || keypressed == '5' ||
        keypressed == '6' || keypressed == '7' || keypressed == '8' ||
        keypressed == '9')             // reaguje na stisk číslice
    {
      zadHeslo += keypressed;          // uloží číslo a posune se o 1 pozici
      lcd.setCursor(k, 1);             // nastaví kurzor na pozici - k
      lcd.print("*");                  // zobrazuje místo čísla hvězdičku
      k++;                             // k = k+1;
    }
  }
  if (k > 9 || keypressed == '#')      // zadáno 5. číslo, nebo stisk křížku
  {
     nastaveniProZadaniHesla = 0;      // obnoví menu pro zadávání hesla
  }
  if (keypressed == '*')               // stisk hvězdičky:
  {
    if (zadHeslo == heslo)             // pokud je zadané správné heslo:
    {
      hesloOvereno = true;             // heslo je ověřeno
    }
    else if (zadHeslo != heslo)        // pokud je zadané špatné heslo:
    {
      lcd.setCursor(0, 1);             // nastaví kurzor do polohy (0,1)
      lcd.print("Wrong! Try Again");   // zobrazí se na displeji text
      delay(2000);                     // počká 2 sec.
      nastaveniProZadaniHesla = 0;     //obnoví menu pro zadávání hesla
} } }
```

Zdrojový kód 21: Bezpečnostní systém – ověření hesla

7.3 Topení

Jako zdroj tepla byla zvolena halogenová žárovka o výkonu 20 $[W]$. Ta je napájena z externího zdroje napětím 12 $[V]$. Schéma zapojení je na obrázku 46. Červená součástka je modul s MOSFET tranzistorem. Do svorkovnice vpravo je připojeno vstupní napětí a na levé straně je připojena spínaná žárovka. Vstup pro spínání tranzistoru je připojen k pinu číslo 12. Na vstupní analogový pin $A13$ je připojen potenciometr. A na digitální pin 23 je připojen senzor teploty. Eses MOSFET modul zde plní funkci regulátoru. Na vstup je přiváděn PWM signál z Arduina, kterým je regulováno výstupní napětí, a tím i povrchová teplota žárovky. Hodnotu PWM signálu lze nastavit otočným potenciometrem.



Obrázek 46: Schéma zapojení a řízení topení

Topení lze také řídit přes mobilní aplikaci, ta obsahuje virtuální potenciometr. Úkolem řídicího systému je reagovat na oba potenciometry tak, aby se navzájem nevyrušovali. Toho je docíleno tak, že se sice načítají hodnoty z obou potenciometrů, ale pouze jedna z nich je převedena na výstupní PWM signál. A to je hodnota toho potenciometru, na kterém byla jako posledním zaznamenána změna.

7.3.1 Zdrojový kód řízení topení

```
int mosfet = 12;                    // MOSFET tranzistor (pin)
int potenciometr = A13;             // potenciometr (analogový pin)
int prepocetState;                  // přep. analogových hodnot (uložených)
int prepocet = 0;                   // přep. analogových hodnot (aktuálních)
boolean counterPrepocetState = 0;   // zabrání přepsání - prepocetState
int pinValue8State;                 // hodnota na Blynku (uložená)
int pinValue8 = 0;                  // hodnota na Blynku (aktuální)
boolean counterBlynk = 0;           // zabrání přepsání - pinValue8State
int rizeniTopeni = 0;               // kdo řídí: 1 - potenciometr, 2 - Blynk
unsigned long casSynchro = 0;       // čas synchronizace mezi potenciometry
BLYNK_WRITE(V8) {                   // načte data z V8 (virt. potenciometr)
pinValue8 = param.asInt(); }        // a uloží do proměnné
```

Zdrojový kód 22: řízení topení – prvotní nastavení

V úvodní části kódu je nastavení všech proměnných. Pin $A13$ bude přijímat analogovou hodnotu z potenciometru, ta bude následně přepočítána a odeslána na MOSFET tranzistor. V druhé části je zobrazen obsah funkce *loop()*. Topení je řízeno algoritmem, kdy je v první řadě uložena hodnota potenciometru a zabráněno jejímu přepsání. Při každém opakování cyklu dojde k porovnání uložené a aktuální hodnoty. Stejné porovnání proběhne i na virtuálním potenciometru. Pokud na jednom z nich dojde ke změně aktuální hodnoty, bude topení řízeno tímto potenciometrem a na druhém bude povoleno přepsání uloženého stavu. Tím je zajištěna reakce systému na oba potenciometry, aniž by se navzájem vyrušovali. Pokud bude topení řízeno reálným potenciometrem, bude navíc každé dvě vteřiny probíhat synchronizace hodnot obou potenciometrů.

```
if (counterPrepocetState == 0) {   // není zabráněno uložení stavu pot.:
  prepocetState = analogRead(potenciometr)/4; // ulož stav potenciometru
  counterPrepocetState = 1;        // zabraň znovunačtení a přepsání stavu
}
prepocet = analogRead(potenciometr)/4; // načti poslední stav potenciometru
// pokud je rozdíl uloženého a posledního stavu potenciometru větší jak 5: ↓
if (rizeniTopeni != 1 && abs(prepocetState - prepocet) > 5 ) {
  rizeniTopeni = 1;                // topení je řízeno potenciometrem
  counterBlynk = 0;                // umožní znovunačtení - pinValue8State
}
if (rizeniTopeni == 1) {           // pokud je topení řízeno potenciometrem:
  analogWrite(mosfet, prepocet);   // nastav topení podle hodnoty na pot.
  if (millis() > casSynchro)
    Blynk.virtualWrite(V8, prepocet); // synch. virt. pot. podle reálného
    casSynchro = millis() + 2000;  // každé dvě vteřiny
  }
}
if (counterBlynk == 0) {           // není zabráněno uložení stavu virt. pot
  pinValue8State = pinValue8;      // ulož stav virt. potenciometru
  counterBlynk = 1;                // zabraň znovunačtení a přepsání stavu
}                                  // pokud je rozdíl uloženého a posledního
                                   // stavu virt. potenciometru větší jak 1:
if (rizeniTopeni != 2 && abs(pinValue8State - pinValue8) > 1 ) {
  rizeniTopeni = 2;                // topení je řízeno virt. potenciometrem
  counterPrepocetState = 0;        // umožní znovunačtení - prepocetState
}
if (rizeniTopeni == 2) {           // pokud je topení řízeno virt. pot.:
  analogWrite(mosfet, pinValue8);  // nastav topení podle hodnoty na virt.
}                                  // potenciometru
```

Zdrojový kód 23: řízení topení – algoritmus

Níže je zobrazen zdrojový kód senzoru teploty a vlhkosti. Pro práci s ním byla využita knihovna *DHT.h*. Ta zpracovává načtená data ze senzoru. Každých pět vteřin jsou hodnoty přeposlány do aplikace Blynk.

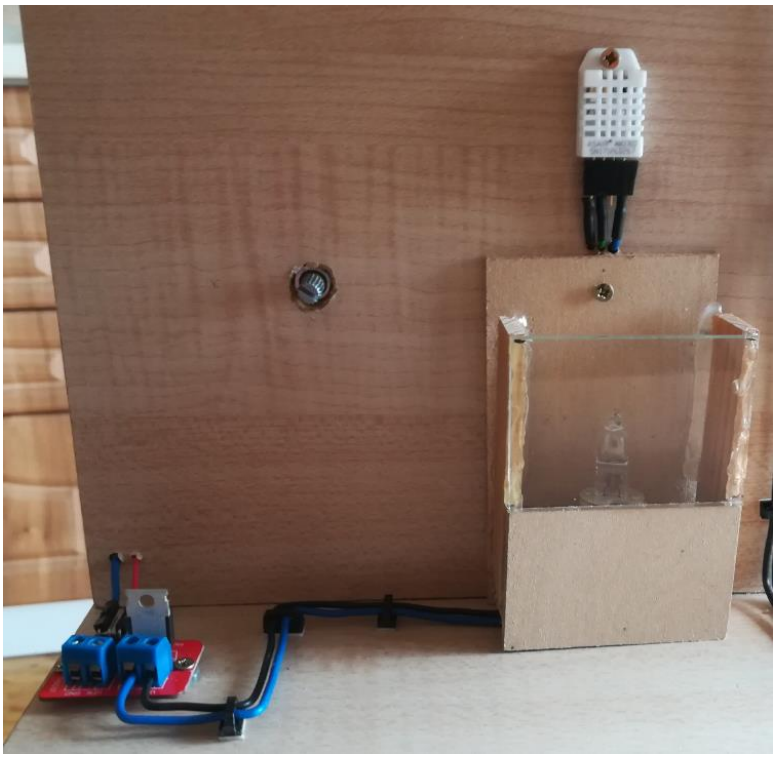
```
#include "DHT.h"                   // načte knihovnu DHT
#define pinDHT 23                  // DHT čidlo (pin)
DHT dht(pinDHT, DHT22);            // nastavení DHT
float teplota;                     // ukládá teplotu
float vlhkost;                     // ukládá vlhkost
unsigned long timeDHT = 0;         // časovač
void setup() {
  dht.begin();                     // rozběhnutí senzoru
}
```

```
void loop() {
  if (millis() > timeDHT) {           // po doběhnutí časovače:
    teplota = dht.readTemperature(); // uloží teplotu do proměnné
    vlhkost = dht.readHumidity();    // uloží vlhkost do proměnné
    timeDHT = millis()+ 5000;        // nastaví časovač na 5 sec.
    Blynk.virtualWrite(V5, teplota); // pošle hodnotu do aplikace Blynk
    Blynk.virtualWrite(V6, vlhkost); // pošle hodnotu do aplikace Blynk
  }
}
```

Zdrojový kód 24: DHT senzor

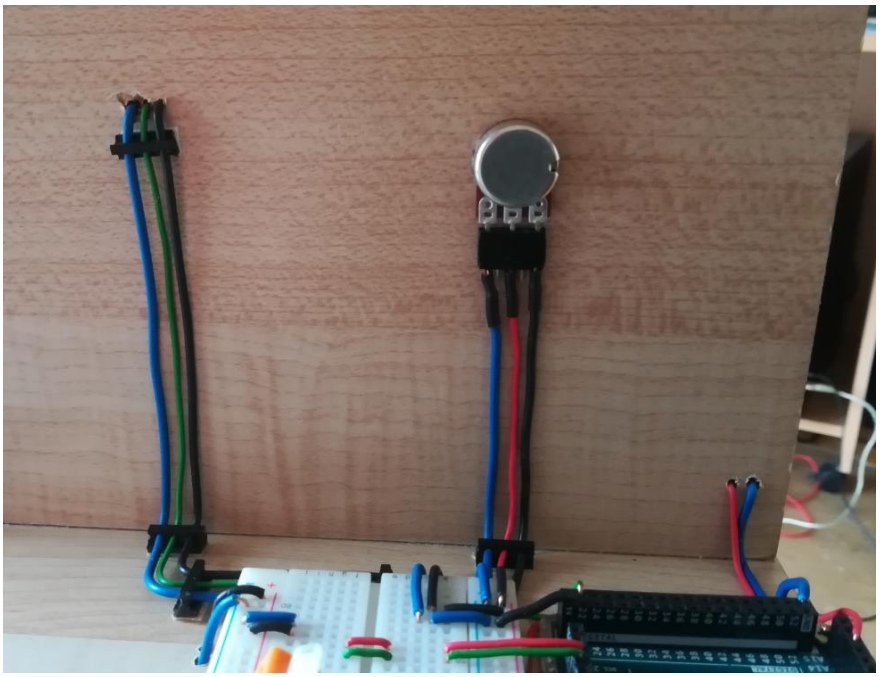
7.3.2 Zapojení topení

Žárovka je zapojena uvnitř dřevěného rámu s proskleným čelem, ten tvoří maketu kamen. Ze spodu jsou k žárovce přivedeny napájecí kabely. Nad konstrukcí je zapojen senzor teploty.



Obrázek 47: Zapojení vytápění

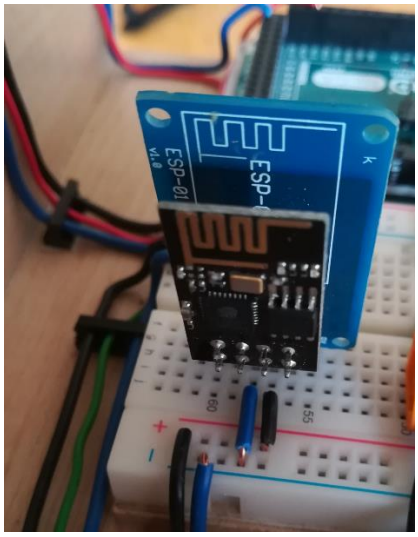
Vodiče pro tranzistor a DHT senzor jsou přivedeny z druhé strany zdi. Potenciometr je dokonce zapojený úplně na druhé straně a je vyvedena pouze otočná matice. Pohled z druhé strany je zobrazeno na obrázku 48.



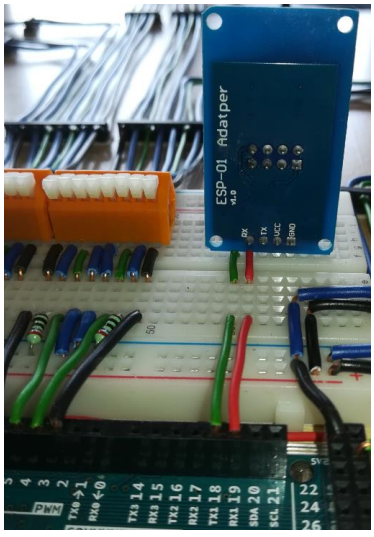
Obrázek 48: Zapojení vytápění – přívod vodičů

7.4 Připojení chytrého domu k internetu

Připojení k internetu zajišťuje ESP8266 Wi-Fi modul. Ten je zapojen do nepájivého pole podle obrázku 49 a 50. Z jedné strany jsou přivedeny napájecí vodiče a z druhé jsou piny modulu Rx a Tx křížově propojeny s Arduino piny Rx1 a Tx1.



Obrázek 49: Zapojení ESP – napájení



Obrázek 50: Zapojení ESP – komunikace

7.4.1 Vzdálené ovládání mobilním telefonem

Vzdálené ovládání mobilním telefonem zprostředkovává aplikace Blynk. Pro navázání vzájemné komunikace mezi touto aplikací a Arduinem byl sestaven následující zdrojový kód.

```
#define BLYNK_PRINT Serial            // posílá info. na sériový monitor
#include <ESP8266_Lib.h>              // načtení knihovny ESP modulu
#include <BlynkSimpleShieldEsp8266.h> // načtení knihovny Blynk
char auth[] = "c58f18e7065e411783c1545bedde339c"; //token Blynk aplikace
char ssid[] = "HOTSPOT_DUM";          // název Wi-Fi
char pass[] = "123456789";            // heslo Wi-Fi
char server[] = "blynk-cloud.com";    // vzdálený server
int port = 8080;                      // tcp port
#define EspSerial Serial1             // sériová komunikace na RX1 a TX1
#define ESP8266_BAUD 115200           // rychlost přenosu
ESP8266 wifi(&EspSerial);
unsigned long cas_reconnect;          // čas dalšího pokusu připojení Wi-Fi
```

Zdrojový kód 25: Nastavení připojení k Wi-Fi – nastavení

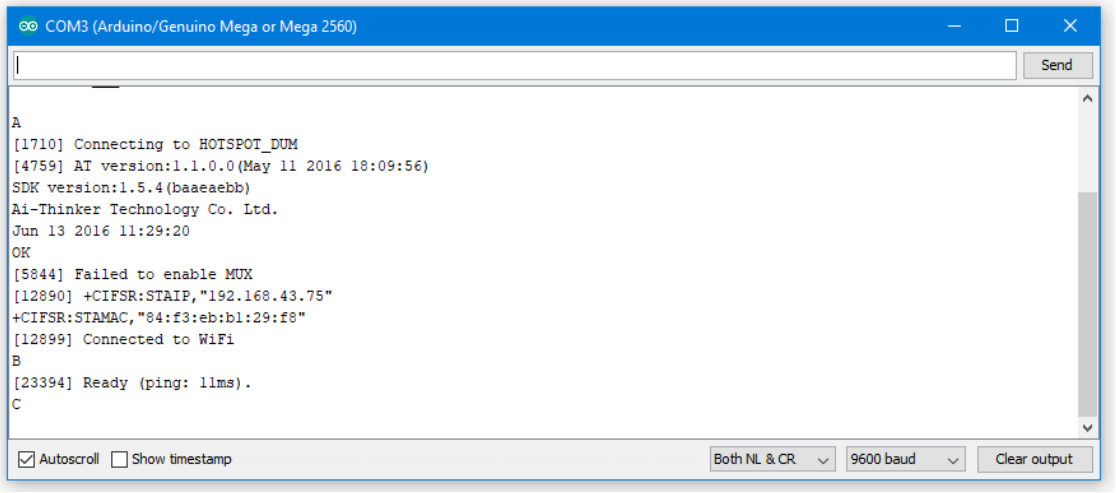
V úvodní části jsou nastaveny všechny knihovny a proměnné, které jsou nezbytné pro připojení k Wi-Fi. Důležité jsou zejména řádky s názvem a heslem Wi-Fi, ke které se bude ESP modul připojovat.

```
void setup() {
  Serial.begin(9600);
  EspSerial.begin(ESP8266_BAUD);      // start sériové komunikace s ESP
  Blynk.config(wifi, auth, server, port);// nastavení připojení
  Serial.println("A");
  Blynk.connectWiFi(ssid, pass);      // připojení k Wi-Fi
  Serial.println("B");
  Blynk.connect();                    // připojení přes vzdálený server
  Serial.println("C");
  cas_reconnect = millis() + 600000;  // čas mezi pokusy o připojen k Wi-Fi
}
```

Zdrojový kód 26: Nastavení připojení k Wi-Fi – setup

Navázání vzdálené komunikace se skládá ze tří kroků. Nastavení síťového připojení příkazem *Blynk.config()*, připojení k Wi-Fi příkazem *Blynk.connectWiFi* a připojení ke vzdálenému serveru příkazem *Blynk.connect()*. Pro ilustraci byly mezi příkazy vloženy body A, B, C a následně spuštěn sériový monitor. Ten je zobrazen

na obrázku 51. Za bodem A začíná pokus o připojení k Wi-Fi síti, po připojení se zobrazí hláška: *Connected to WiFi*. Za bodem B probíhá připojování přes internet ke vzdálenému serveru. Připojení v tomto případě proběhlo úspěšně s odezvou 11 $[ms]$.



Obrázek 51: Sériový monitor připojení k Wi-Fi

Poslední část kódu zajišťuje, že se po výpadku připojení k internetu Arduino bude pokoušet znovu připojit. Připojení neproběhne ihned, ale jednou za deset minut. Důvodem je základní vlastnost Arduina, že nedokáže vykonávat více činností najednou. Připojení k Wi-Fi trvá několik vteřin a po tu dobu není Arduino schopné reagovat na žádné další podněty.

```
void loop() {
  if (Blynk.connected()){              // pokud je připojeno k Wi-Fi:
    Blynk.run();                       // Blynk běží
  }
  else if (millis() > cas_reconnect)   // není připojeno a doběhl časovač:
  {
    Blynk.connectWiFi(ssid, pass);     // pokusí se připojit
    Blynk.connect();                   // pokusí se zahájit komunikaci
    cas_reconnect = millis() + 600000;// nastaví časovač
} }
```

Zdrojový kód 27: Nastavení připojení k Wi-Fi – loop

Závěr

V této práci byl navržen výukový model chytrého domu s inteligentní elektroinstalací (osvětlení, topení, alarm). Pro realizaci řídicího systému byla zvolena vývojová deska Arduino Mega2560. Ovládání všech prvků je umožněno lokálně manuálně a vzdáleně pomocí mobilního telefonu s operačním systémem Android nebo iOS. Každá místnost má vlastní osvětlení. To je realizováno pomocí LED diod. Tři místnosti jsou opatřeny inteligentní elektroinstalací a jedna pro možnost porovnání konvenční elektroinstalací. Další světlo je umístěno před vchodem do domu. Práce obsahuje schéma zapojení, stavový automat a zdrojový kód vnitřního i venkovního osvětlení. Součástí modelu je také alarm, který upozorní uživatele na pohyb uvnitř domu. Součástí alarmu je klávesnice pro zadání bezpečnostního hesla, displej, pohybové čidlo a bzučák, který se spustí při detekci pohybu. Dalším systémem v domě je topení. Jako zdroj tepla byla použita žárovka. Ta je napájena z externího zdroje. Ke spínání byl použit tranzistor a k regulaci otočný potenciometr.

Při návrhu elektroinstalace byla snaha o přehledné zapojení všech elektrických součástek. Vodiče jsou trasovány tak, aby bylo na první pohled zřejmé k čemu jaký vodič je a odkud kam vede. Všechny součástky, se kterými se bude mechanicky manipulovat, jsou přišroubovány ke stěnám modelu. Většina součástek byla k montáži již sama přizpůsobena. Pro některé však bylo nutné navrhnout vlastní uchycení. Příkladem mohou být kolébkové přepínače a PIR senzor, pro které byly navrhnuty speciální držáky. Ty byly následně vytisknuty na 3D tiskárně.

Ovládání chytrého domu působí přehledně a jednoduše. Společně se schématy zapojení jednotlivých systémů, grafy stavových automatů a přehledným zdrojovým kódem, tvoří celá práce vhodnou výukovou pomůcku pro pochopení centralizovaného řídicího systémů chytrého domu.

Seznam zdrojů:

[1] *Chytrý dům* [obrázek] [online]. [cit. 2019-28-1]. Dostupné z: https://www.loxone.com

[2] *Inteligentní budovy* [online]. [cit. 2019-28-1]. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/lekce/990

[3] *Co je a jak funguje chytrý dům* [online]. [cit. 2019-28-1]. Dostupné z: https://www.lupa.cz/clanky/co-to-je-a-jak-funguje-chytry-dum-chytry-byt-a-chytra-domacnost/

[4] *Výhody a rizika chytré domácnosti* [online]. [cit. 2019-28-1]. Dostupné z: https://www.living.cz/jake-vyhody-rizika-ma-chytra-domacnost/

[5] *Systémová technika budov* [online]. [cit. 2019-28-1]. Dostupné z: http://fei1.vsb.cz/kat420/vyuka/Bakalarske/STB/1_klasicka_elektroinstalace_a_systemova_technika_budov.pdf

[6] *Informační systémy pro správu budov* [Online]. [cit. 2019-28-1]. Dostupné z: https://is.muni.cz/th/xwul2/TEXT/thesis-petr-kutalek-2008.xhtml

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

[8] *Arduino knihovny* [Online]. [cit. 2019-28-1]. Dostupné z: https://navody.arduino-shop.cz/zaciname-s-arduinem/arduino-knihovny.html

[9] *Language References* [Online]. [cit. 2019-28-1]. Dostupné z: https://www.arduino.cc/reference/en/

[10] *PWM* [Online]. [cit. 2019-28-1]. Dostupné z: https://www.arduino.cc/en/Tutorial/PWM

[11] *Programming the ESP8266* [Online]. [cit. 2019-28-1]. Dostupné z: https://ubidots.com/blog/esp8266-arduino-ide-tutorial/

[12] *LCD* [Online]. [cit. 2019-28-1]. Dostupné z: http://www.mipi.chytrak.cz/lcd.php

[13] *LiquidCrystal Library* [Online]. [cit. 2019-28-1]. Dostupné z: https://www.arduino.cc/en/Reference/LiquidCrystal

[14] *Datasheet maticové klávesnice* [Online]. [cit. 2019-28-1]. Dostupné z: https://cdn.sparkfun.com/datasheets/Components/General/SparkfunCOM-08653_Datasheet.pdf

[15] *Komunikace po sériové sběrnici I2C* [Online]. [cit. 2019-28-1]. Dostupné z: https://www.root.cz/clanky/komunikace-po-seriove-sbernici-isup2supc/

[16] *Stručný popis sběrnice I2C* [Online]. [cit. 2019-28-1]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256

[17] *Teploměr a vlhkoměr DHT11 a DHT22* [Online]. [cit. 2019-28-1]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/teplotni-senzor-dht11.html