VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ PRŮMYSLOVÁ ŠKOLA
A JAZYKOVÁ ŠKOLA S PRÁVEM STÁTNÍ JAZYKOVÉ ZKOUŠKY
KUTNÁ HORA, MASARYKOVA 197

MATURITNÍ PRÁCE

METEOROLOGICKÁ STANICE

ROK 2022

MARCEL KUBÍN

PODĚKOVÁNÍ

Nejprve bych chtěl poděkovat vedoucímu práce Ing. Stanislavu Moravcovi za přínosné konzultace a jeho ochotu. Při jakémkoliv dotazu mě byl vždy schopen navést správným směrem. Dále děkuji mému spolužáku Martinu Mandovi za pomoc a celkové umožnění výroby plošných spojů. Bez jeho pomoci by výroba plošných spojů byla mnohem časově i finančně náročnější. V neposlední řadě děkuji rodině za částečné financování projektu.

ABSTRAKT

Záměrem práce bylo navrhnout, naprogramovat a zkonstruovat bezdrátovou meteorologickou stanici založenou na mikrokontroleru ATmega328p pro domácí použití. Stanice se skládá z venkovní jednotky napájené síťovým zdrojem a vnitřní jednotky napájené z baterií. Vnitřní jednotka je schopna měřit teplotu a relativní vlhkost vzduchu, přijímat data z venkovní jednotky pomocí 433MHz přijímače a následně všechny tyto hodnoty zobrazit na příslušném displeji. Venkovní jednotka je vybavena 433Mhz vysílačem, senzorem pro měření teploty, relativní vlhkosti a atmosférického tlaku, v neposlední řadě venkovní jednotka disponuje samostatným displejem pro zobrazení naměřených hodnot.

ABSTRACT

The aim of the work was to design, program and construct a weather station based on ATmega328p microcontroller for home use. The weather station consists of an outdoor unit powered by power supply and indoor unit powered with batteries. Indoor unit is able to measure temperature and relative humidity, receive data from the outdoor unit using a 433Mhz receiver and then display all of these values on the appropriate display. The outdoor unit is equipped with a 433MHz transmitter, a sensor for measuring temperature, relative humidity and atmospheric pressure, last but not least, the outdoor unit has a separate display for displaying measured values.

KLÍČOVÁ SLOVA

meteorologická stanice, měření, teplota, vlhkost, tlak, ATmega328p, Microchip Studio

KEYWORDS

weather station, measurement, temperature, humidity, pressure, ATmega328p, Microchip Studio

OBSAH

Úvod

Cílem práce byla kompletní realizace meteorologické stanice, skládající se z venkovní a vnitřní jednotky s použitím bezdrátové komunikace. Stanice by měla být snadná na používání, aby ji mohl bez problému používat každý. Zároveň ale musí měřit všechny potřebné hodnoty.

Smysl mé práce však nespočívá pouze v závěrečném výrobku, ale z velké části také v procesu výroby. Již na začátku jsem se rozhodl nejít cestou předpřipravených desek typu Arduino a porozumět tak více samotné funkci a vnitřní stavbě mikrokontroleru. Zároveň jsem se rozhodl, že chci co největší část softwaru pro mou meteorologickou stanici napsat sám. To znamená nejen hlavní program, ale i jednotlivé knihovny pro komunikaci se senzory a displeji, měření času, výpočet CRC a další. Tyto knihovny jsou za normálních okolností již předpřipravené uživateli, ale já chtěl zjistit, jak doopravdy fungují a na jakém principu pracují.

1 METEOROLOGICKÁ STANICE

Meteorologická stanice [1] je zařízení určené k měření meteorologických údajů. Naměřené hodnoty mohou sloužit k předpovědi počasí, či k přímé prezentaci naměřených hodnot uživateli. Mezi nejčastěji měřené údaje patří teplota, vlhkost vzduchu, atmosférický tlak, směr a rychlost větru, popřípadě množství vodních srážek. Pokročilejší stanice jsou schopné vyhodnocovat i další meteorologické jevy. Profesionální meteorologické stanice používané meteorology mohou být velice nákladná a složitá zařízení. Naopak komerčně prodávané stanice používané v domácnostech primárně pro zobrazení aktuálních hodnot jsou vyráběny co nejjednodušeji s co nejnižšími náklady.

1.1 Přístroje používané v meteorologii

- Teploměr – měření teploty
- Barometr – měření atmosférického tlaku
- Vlhkoměr – měření relativní vlhkosti vzduchu
- Anemometr – měření rychlosti a směru větru
- Pyranometr – měření slunečního záření
- Srážkoměr – měření srážkových úhrnů

Některé stanice mohou být osazeny i několika přístroji stejného typu najednou.

Obr. 1 Profesionální meteorologická stanice

Obr. 2 Meteorologická stanice pro domácí použití

2 VÝBĚR A PŘÍPRAVA KOMPONENTŮ

2.1 Mikrokontroler

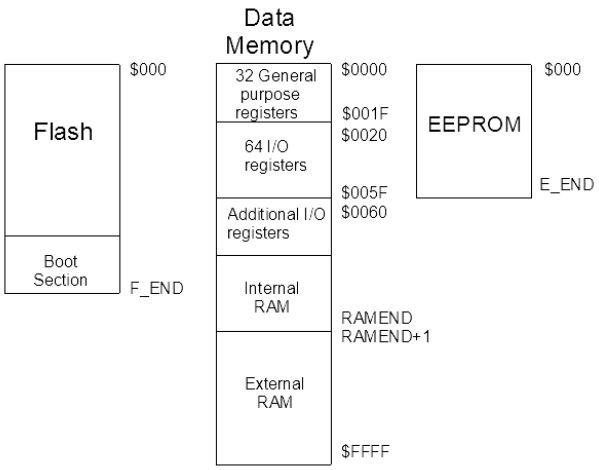
Jako první krok bylo nutné zvolit mikrokontroler [2] pro řízení jednotlivých jednotek. Mikrokontroler je zpravidla jeden monolitický integrovaný obvod. Mikrokontrolery jsou většinou používány v aplikacích pro velice specifické a předem dané operace. Základní části mikrokontroleru jsou:

- Procesor
- Paměť
- Programovatelné vstupy/výstupy

Procesor je zodpovědný za vykonávání jednotlivých instrukcí daných programem. Ke každému mikrokontroleru lze dohledat tzv. instrukční sadu. Instrukční sada je list instrukcí, které je daný mikrokontroler schopný vykonat.

Dále pak mikroprocesor disponuje zpravidla hned několika druhy již zmíněné paměti. Prvním typem je programová paměť (ROM). Programová paměť, jak už název napovídá je určena k trvalému uchování uživatelského programu. Část paměti ROM je vyhrazena pro bootloader (velikost této sekce lze zpravidla specifikovat pomocí nastavení pojistek). Druhým typem je operační paměť (RAM), tento druh je určen

pro dočasné uchování hodnot. Jsou zde uloženy mimo jiné hodnoty proměnných. Některé mikrokontrolery jsou doplněny ještě například o paměť typu EEPROM. Paměť tohoto typu umožňuje uživateli trvale uchovávat data navzdory výpadkům napájení, nebo resetu mikrokontroleru.



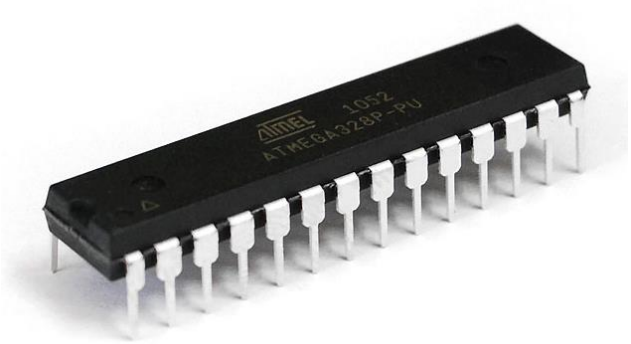
Obr. 3 Příklad paměťové mapy mikrokontroleru

Jednotlivé mikrokontrolery mají různý počet vstupů/výstupů. Vstupy je mikrokontroler schopen číst a následně zpracovat. Výstupy jsou naopak ovládané pomocí programu.

Další vybavení mikrokontroleru je specifické pro jednotlivé modely. Běžně se setkáváme s A/D převodníky, časovači/čítači a komunikačními rozhraními jako jsou USART, I^2C, nebo SPI a další.

2.1.1 Výběr

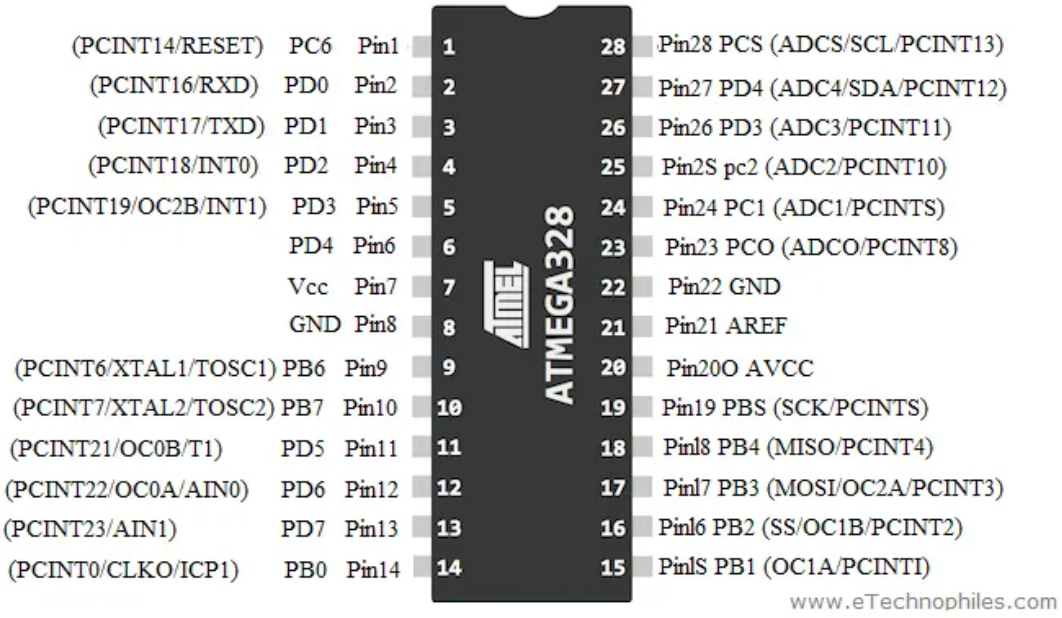
Při výběru mikrokontroleru je důležité si uvědomit co od něj požadujeme a co už není pro aplikaci tak důležité. V mém případě byl výběr poměrně snadný. Potřeboval jsem mikrokontroler, který bude mít dostatečně malou spotřebu pro použití na bateriích, ale také dostatek paměti pro použití OLED displeje. Mikrokontroler musel být také vybaven komunikačním rozhraním I^2C (TWI), které používají displeje i jednotlivé senzory. Pro svou aplikaci jsem tedy zvolil mikrokontroler ATmega328p z rodiny AVR [3], protože jsem jich několik měl již doma a zároveň splňují všechny kritické požadavky. Pro zjednodušení jsem zvolil stejný univerzální mikrokontroler do vnitřní i venkovní jednotky.



Obr. 4 ATmega328p v pouzdře PDIP-28

2.1.2 Příprava mikrokontroleru

Před prvním naprogramováním bylo nutné mikrokontroler nejprve připravit. Jako první krok jsem si připravil základní zapojení, kde jsem připojil pull-up rezistor na resetovací pin, mezi piny XTAL1 a XTAL2 zapojil piezokrystal s frekvencí 16 MHz pro vnitřní jednotku a 4 MHz pro vnější jednotku. Piny MOSI, MISO, SCK, RESET, GND a Vcc jsem si vyvedl na jedno místo pro snazší programování.



Obr. 5 uspořádání pinů ATmega328p

Následovalo naprogramování pojistek (fuse bitů) pomocí SPI komunikačního rozhraní. To jsem provedl pomocí nástroje AVRDUDE. Pro vnitřní jednotku jsem nastavil pojistky pro externí 8–20 MHz krystal, prodlevu po zapnutí na 16K CK a prodlevu po resetu 14 CK + 4,1 ms. Část paměti pro bootloader jsem nastavil na co nejmenší a povolil možnost zapsání do programové paměti pomocí SPI. Venkovní jednotka má pojistky nastaveny naprosto identicky, s výjimkou frekvence krystalu. Pro venkovní jednotku jsem nastavil pojistky pro externí 3–8 MHz krystal. Ten je zde použit v zájmu šetření energie.

Kompletní pojistky pro vnitřní jednotku ve formátu parametrů pro AVRDUDE:

```
-U lfuse:w:0xef:m -U hfuse:w:0xdf:m -U efuse:w:0xff:m
```

Kompletní pojistky pro venkovní jednotku ve formátu parametrů pro AVRDUDE:

```
-U lfuse:w:0xed:m -U hfuse:w:0xdf:m -U efuse:w:0xff:m
```

Při nahrávání pojistek bylo nutné nástroji AVRDUDE zadat používaný mikrokontroler a programátor. Celý příkaz tak v mém případě vypadal takto:

```
avrdude -p m328 -c usbasp -U lfuse:w:0xef:m -U hfuse:w:0xdf:m -U
efuse:w:0xff:m
```

Zprvu se mi pojistky nepodařily nahrát. Ukázalo se, že je to způsobeno továrním nastavením pojistek mikrokontroleru a programátorem. V základu je totiž ATmega328p nastavena na vnitřní 8 MHz oscilátor s zapnutou děličkou 8. To způsobilo, že programátor byl příliš rychlý.

Tento problém jsem vyřešil pomocí jednorázového nastavení parametru -B (bitclock) na nižší hodnotu.

2.2 Senzory

2.2.1 Požadavky

Žádná meteorologická stanice by se neobešla bez senzorů, je ale nutné vybrat správně. Mé požadavky byly, aby senzory nebyly zbytečně drahé, ale dostatečně přesné pro jednotlivá použití v domácích podmínkách. Senzor pro venkovní jednotku musel být schopen pracovat i na nižším napětí (3,3 V) a především měl malý odebíraný proud v režimu spánku. V ideálním případě senzory také měli komunikovat pomocí I^2C (TWI). Minimální požadavky na měřené veličiny byly teplota a vlhkost pro vnitřní i venkovní jednotku.

2.2.2 Výběr

Následně, když jsem věděl, jaké senzory hledám, jsem mohl přistoupit k samotnému výběru. V procesu výběru jsem si porovnával několik senzorů dle parametrů uvedených v datasheetu. Po důkladném srovnání jsem usoudil, že senzor SHT31 [4] bude vhodný pro použití ve vnitřní jednotce. Disponuje rozhraním I^2C, je schopen měřit teplotu s přesností na 0,3 °C a relativní vlhkost s přesností 2 %. Pro venkovní jednotku jsem zvolil senzor BME280 [5]. Taktéž disponuje rozhraním I^2C. Měří teplotu a relativní vlhkost a atmosférický tlak. Naměřené hodnoty sice nedosahují takové přesnosti, jako u senzoru SHT31, nicméně jsem usoudil, že pro venkovní měření jsou dostatečné. Tento senzor jsem vybral z důvodu možnosti měřit atmosférický tlak a z důvodu velmi malého odebíraného proudu v režimu spánku i během měření. Řádově µA.

Obr. 6 Senzor SHT31 jako SMD

Obr. 7 Senzor BME280 jako SMD

2.3 Displeje

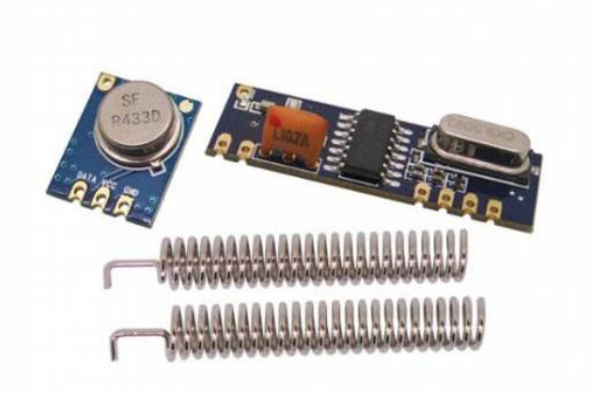
Vzhledem k tomu, že se jedná o meteorologickou stanici určenou pro domácí použití, bylo nutné ji vybavit nějakým zobrazovacím prvkem. Tedy displejem. Nabízelo se hned několik variant, například LCD (Liquid Crystal Display), OLED (Organic Light Emitting Diode), popřípadě displej typu e-Paper. Displeje e-Paper jsem vyloučil jako první. Mají sice velmi malou spotřebu, ale jsou velmi drahé, a i malý displej by svou cenou nejméně zdvojnásobil cenu stanice. Zbyly tedy displeje typu LCD a OLED. Ve své aplikaci jsem se nakonec rozhodl použít displej OLED, ačkoliv LCD displej by byl dle mého názoru též dobrou volbou. OLED displej jsem zvolil z důvodu výborného rozlišení, dobré viditelnosti, a také jsem si chtěl vyzkoušet práci s něčím s čím jsem ještě nepracoval. Displeje sice mají vyšší spotřebu, ale jen když jsou zapnuté. Nespornou výhodou bylo také že OLED displeje se běžně prodávají s I^2C komunikačním rozhraním (stejné jako u senzorů). Konkrétní použité čipy jsou SSH1106 [6] (vnitřní jednotka) a SSD1306 [7] (venkovní jednotka).

Obr. 8 SSH1106 OLED displej 128 × 64

Obr. 9 SSD1306 OLED displej 128 × 32

2.4 Bezdrátová komunikace

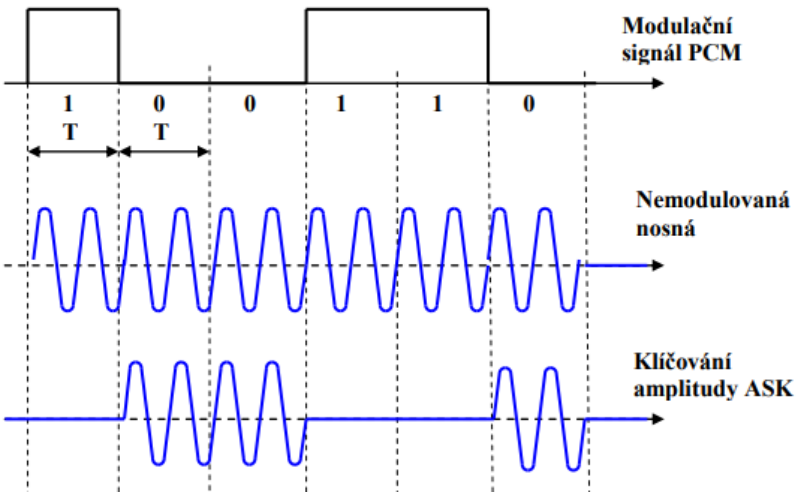
Jak již bylo řečeno, rozhodl jsem se vnitřní a venkovní stanici propojit bezdrátově, protože drátové spojení by bylo krajně nepraktické. Pro bezdrátovou komunikaci jsem zvolil moduly STX882 (vysílač) a SRX822 (přijímač) pracující na frekvenci 433 MHz za použití amplitudového klíčování. Tento druh komunikace jsem zvolil, protože moduly pracují jednoduchém principu a jsou levné. Vzhledem k tomu, že datový tok mé aplikace je minimální, moduly plně dostačují všem požadavkům.



Obr. 10 433 MHz vysílač STX882 + 433 MHz přijímač SRX882

2.4.1 Klíčování amplitudovým posuvem (ASK)

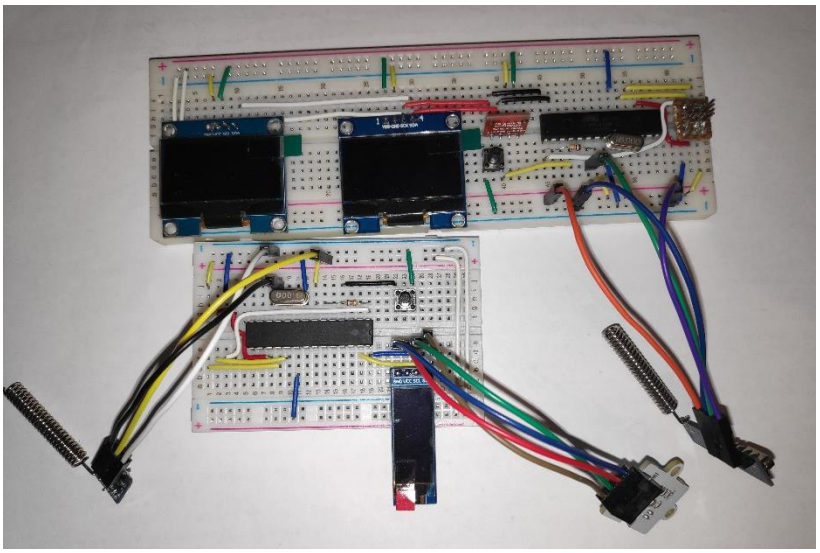
Klíčování amplitudovým posuvem je forma amplitudové modulace. Díky ASK jsem schopni poslat digitální data za pomoci změn amplitudy nosné vlny. Nevýhodu je, že tato forma modulace není odolná vůči rušení. Velká náchylnost na amplitudové poruchy. Základní varianta 2 – ASK je dvoustavová. Má dva symboly, jeden pro 0 a druhý pro 1.



Obr. 11 Příklad ASK

2.5 Když je vše vybrané

Poté, co všechny komponenty byly vybrané, jsem si všechny důležité části zapojil na nepájivém poli. Během zapojování jsem se orientoval podle datasheetů jednotlivých komponentů. Když bylo vše takto připravené, mohl jsem přejít k samotnému programování.



Obr. 12 Vnitřní a venkovní jednotka zapojená na nepájivém poli

3 PROGRAM

Následující kapitola se věnuje přípravě programovacího prostředí a samotnému naprogramování jednotlivých jednotek. Celý program je napsán v programovacím jazyce C. Programy jednotek jsou složeny vždy z hlavního programu a několika knihoven. Popisy programů jsou jen orientační. Celé zdrojové kódy lze nalézt v příloze.

Knihovna je samostatný soubor obsahující většinou funkce pro zpracování určitých operací. Kromě vlastního zdrojového kódu knihovny je nutné ještě vytvořit tzv. hlavičkový soubor. Pomocí tohoto souboru lze jednotlivé knihovny propojit s hlavním programem. V mé aplikaci se knihovny zpravidla specializují na jednotlivé komunikace, senzory atd. Hlavním zdrojem informací při programování knihovny mi vždy byl datasheet daného zařízení.

Do hlavního programu, jak již bylo řečeno, jsou zahrnuty knihovny pomocí hlavičkových souborů. Knihovny je tedy nutné napsat dříve než hlavní program. Samotný běh mikrokontroleru se tedy řídí hlavním programem, ve kterém jsou čteny jednotlivá tlačítka, volány funkce knihoven, volány funkce samotného hlavního programu atd.

3.1 Příprava programovacího prostředí

Pro naprogramování mikrokontrolerů jsem si zvolil programovací prostředí Microchip Studio. Po instalaci prostředí jsem ho spustil a založil první projekt. Při vytváření projektu je nutné zvolit mikroprocesor. V mém případě ATmega328p. Poté v samotném projektu zvolit frekvenci na které mikrokontroler pracuje. Před nahráním programu ještě zbývá přidat do Microchip Studia programátor. Jako programátor jsem opět použil již zmíněný nástroj AVRDUDE.

3.2 Knihovny pro komunikaci

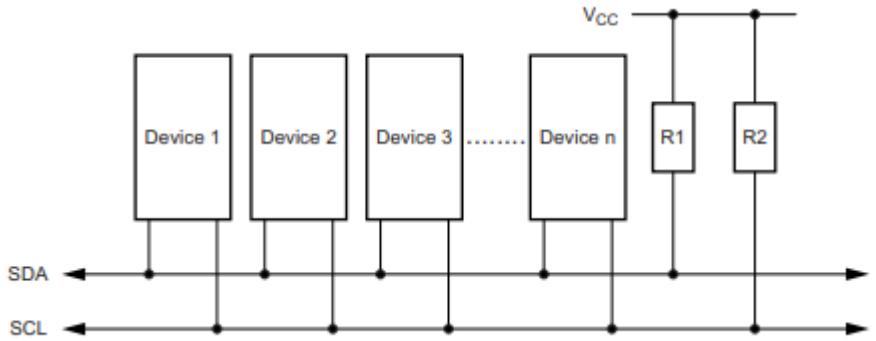
3.2.1 TWI

TWI knihovna umožňuje komunikaci pomocí dvoudrátového rozhraní (Two Wire Serial Interface). TWI je víceméně identické s I^2C rozhraním, které používají senzory a displeje. Tato knihovna umožňuje všechnu komunikaci mezi zařízeními.

Zařízení jsou na TWI připojena sériově jako Master (řídí) nebo Slave (je řízen). Každý Slave musí mít svou adresu. TWI, stejně jako I^2C, používá čtyři vodiče z toho dva pro přenos dat.

Jednotlivé vodiče jsou:

- SDA – přenos binární reprezentace dat
- SCL – přenos hodinového signálu
- Vcc – napájení pro jednotlivá zařízení
- GND – napěťová reference



Obr. 13 Připojení jednotlivých zařízení na TWI (na obrázku jsou pouze datové vodiče)

Při programování této knihovny jsem použil hardwarovou část v ATmega328p určenou pro toto komunikační rozhraní. Knihovna se skládá s několika funkcí, které obstarávají jednotlivé části komunikace.

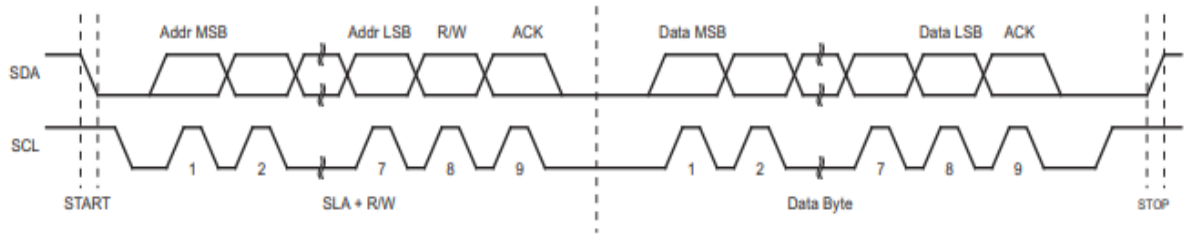
Příklad jednoduché funkce pro začátek komunikace:

```
error_code TWI_Start(){
      TWCR = ((1 << TWINT)|(1 << TWSTA)|(1 << TWEN)) //odešle START
      while(!(TWCR & (1 << TWINT)));      //čeká než je START odeslán
      if(TW_STATUS != TW_START) return TW_STATUS;    //kontrola erroru
      return SUCCES;    //funkce vrátí úspěšný průběh
}
```

Funkce dostupné v hlavním programu jsou:

```
void TWI_Init(uint16_t FREQUENCY, _Bool internal_pullup_en);
error_code TWI_Write(uint8_t Slave_adress, uint8_t *Data, uint16_t
Byte_count);
error_code TWI_Read(uint8_t Slave_address, uint8_t *Data, uint16_t
Byte_count);
```

Pomocí těchto funkcí jsem schopný inicializovat komunikaci, zapisovat i číst z jednotlivých zařízení.



Obr. 14 Typický přenos dat po TWI

3.2.2 VirtualWire

Původním plánem bylo naprogramovat vlastní knihovnu pro bezdrátový přenos. Nicméně můj první nápad odesílat data pomocí komunikačního rozhraní USART nebyl proveditelný. Hlavním důvodem bylo rušení generované na přijímači ve chvílích, kdy nebyla vysílána žádná data. Druhým nápadem bylo použití tzv. Manchester kódování. Nakonec po zvážení všech možností jsem se uchýlil k již vytvořené knihově VirtualWire, kterou jsem upravil tak, aby vyhovovala mým požadavkům.

Knihovna pro přesné vysílání a příjem dat používá vnitřních časovačů ATmega328p. Díky těmto časovačům lze realizovat komunikaci na téměř kterémkoliv pinu.

3.3 Knihovny pro senzory

3.3.1 SHT31

Knihovna SHT31 zprostředkovává data ze senzoru SHT31, za použití knihovny TWI. Tento senzor nevyžaduje žádnou inicializaci. Knihovna tak kromě dvou malých funkcí obsahuje jen jednu hlavní funkci pro načtení a přepočet naměřených hodnot. Společně s naměřenými hodnotami senzor vrací také cyklický redundantní součet, který je zpracován pomocí knihovny CRC_8.

Hlavní funkce pro načtení naměřených hodnot:

```
struct Measurments SHT31_Measure(){
	struct Measurments meas;	//příprava struktury pro vrácení hodnot
	uint8_t data[6];		//datové pole pro nová data

	meas_again:			//v případě chyby měření proběhne znovu
	SHT31_SendCOM(0x2C, 0x06);	//příkaz pro změření aktuálních hodnot
	TWI_Read(SHT31_ADDRESS, data, 6);  //přečtení změřených hodnot

	//kontrola cyklického redundantního součtu teploty
	if(CRC_8_Check(0x31, 0xff, 0x0, data, 3) != 0) goto meas_again;
	//zápis teploty do structu
	meas.Temperature = ((-45 + 175 * ((((uint16_t)data[0] << 8) |
	data[1]) / (float)65535)) * 100);

	//kontrola cyklického redundantního součtu vlhkosti
	if(CRC_8_Check(0x31, 0xff, 0x0, data+3, 3) != 0) goto meas_again;
	//zápis vlhkosti do struktury
	meas.Humidity = (100 * ((((uint16_t)data[3] << 8) | data[4]) /
	(float)65535));

	return meas;			//vracení celé struktury
}
```

3.3.2 BME280

Knihovna BME280 obstarává senzor BME280 použitý ve vnější jednotce. Knihovna postavena na knihovně TWI. Na rozdíl od SHT31 je u tohoto senzoru nutná prvotní inicializace a také použití kompenzačních formulí pro výpočet správných hodnot. Kompenzační parametry použité v těchto výpočtech jsou uloženy přímo v senzoru. Samotná knihovna umožňuje měřit veličiny jednotlivě, nebo všechny na jednou.

Příklad kompenzace atmosférického tlaku:

```
uint32_t BME280_PressCOMP(){
	int32_t adc_P = (BME280_ReadREG(BME280_PRESS_DATA, 3, BIG_ENDIAN) >>
	4);   //načtení dat z registrů senzoru


	//kompenzace atmosférického tlaku
	int64_t var1, var2, p;
	var1 = ((int64_t)t_fine) - 128000;
	var2 = var1 * var1 * (int64_t)BME280_Comp.dig_P6;
	var2 = var2 + ((var1 * (int64_t)BME280_Comp.dig_P5) << 17);
	var2 = var2 + (((int64_t)BME280_Comp.dig_P4) << 35);
	var1 = ((var1 * var1 * (int64_t)BME280_Comp.dig_P3) >> 8) + ((var1 *
	(int64_t)BME280_Comp.dig_P2) << 12);
	var1 = (((((int64_t)1) << 47) + var1)) *
	((int64_t)BME280_Comp.dig_P1) >> 33;
	if(var1 == 0) return 0;
	p = 1048576 - adc_P;
	p = (((p << 31) - var2) * 3125) / var1;
	var1 = (((int64_t)BME280_Comp.dig_P9) * (p >> 13) * (p >> 13)) >> 25;
	var2 = (((int64_t)BME280_Comp.dig_P8) * p) >> 19;
	p = ((p + var1 + var2) >> 8) + (((int64_t)BME280_Comp.dig_P7) << 4);


	return (uint32_t)p;
}
```

3.4 Knihovny pro displeje

3.4.1 SSD1306

Tuto knihovnu jsem naprogramoval pro zobrazení jednotlivých znaků na displeji. Rozhodl jsem se tak, protože jsem chtěl více pochopit funkci OLED displejů, což by s použitím složitých grafických knihoven nebylo možné. Nejedná se tedy o grafickou knihovnu, ani o nijak univerzální knihovnu. Nicméně upravení knihovny pro jinou aplikaci by nebyl velký problém.

Stejně jako u senzorů je knihovna postavena na knihovně TWI. V knihovně jsou funkce určené k zobrazení jednotlivých znaků, ale také inicializační sekvence.

Příklad funkce, která rozloží číselnou proměnnou na jednotlivé číslice a ty jsou následně převedeny na příslušné znaky:

```
//zobrazení vlhkosti
_Bool SSD1306_PrintHUM(uint32_t hum){
	if((hum < 0)||(hum > 100)) return ERROR; //kontrola zda je hodnota možná
	SSD1306_Clear_buffer();			//vyčištění bufferu
	SSD1306_Print_su(HUMIDITY);		//načtení symbolu a jednotky
	if(hum / 10) SSD1306_Print_n(hum / 10, 65);	//výpočet desítek
	SSD1306_Print_n(hum % 10, 83);		//výpočet jednotek
	SSD1306_Send_buffer();			//odeslání dat displeji
	return SUCCESS;				//funkce vrátí úspěšný průběh
}
```

3.4.2 SH1106

Knihovna SH1106 je modifikovanou verzí SSD1306. Čipy jsou téměř totožné. Hlavním rozdílem je nutnost zvyšovat počítadlo řádků OLED displeje manuálně. Dalším rozdílem je absence některých příkazů, které bylo potřeba nahradit.

3.5 Ostatní knihovny

3.5.1 Millis

Knihovna millis slouží k jednoduchému účelu, a tím je počítání času v milisekundách. K počítání knihovna používá jeden čítač nastavený tak, aby přetekl každou milisekundu a vygeneroval přerušení.

Funkce dostupné v knihovně jsou:

```
void millis_Init();     //inicializace knihovny
void millis_Start();    //spuštění časovače
void millis_Stop();     //zastavení počítání
void millis_Reset();    //reset celého počítání
uint64_t millis();      //tato funkce vrací aktuální čas
```

3.5.2 CRC_8

Knihovna CRC_8 je určena pro kontrolu a výpočet cyklického redundantního součtu. Cyklický redundantní součet je sled několika poměrně jednoduchých matematických operací určených k detekci chyb. [8]

Funkce pro výpočet cyklického redundantního součtu:

```
uint8_t CRC_8_Compute(uint8_t poly, uint8_t init, uint8_t f_XOR, uint8_t
*data, uint8_t byte_count){
      uint8_t CRC = init;
      for(uint8_t b = 0; b < byte_count; b++){
            CRC ^= data[b];
            for(uint8_t i = 0; i < 8; i++){
                  if((CRC & 0x80) != 0){
                        CRC = ((CRC << 1) ^ poly);
                  }
                  else CRC <<= 1;
            }
      }
      return (CRC ^ f_XOR);
}
```

3.6 Hlavní program

V této podkapitole slovně zpracovávám hlavní programy jednotek. Celé zdrojové kódy lze opět nalézt jako přílohu.

3.6.1 Vnitřní jednotka

V první části hlavního programu pro vnitřní jednotku nejprve připojím všechny hlavičkové soubory potřebných knihoven. Následují prototypy funkcí, deklarace globálních proměnných a struktur.

Další částí programu je samotný main (hlavní část). V této části nejprve zapínám 2s watchdog, vypínám všechny nepotřebné části mikrokontroleru a provádím prvotní nastavení knihoven, popřípadě komponentů. V hlavní části je také umístěna smyčka, která se neustále opakuje. V této smyčce kontroluji zda nepřišly data z venkovní jednotky, sepnutí tlačítka, obnovuji displeje a následně přecházím do režimu spánku.

Poslední část celého hlavního programu vnitřní jednotky jsou dvě funkce. Jedna zprostředkovává zapnutí již zmíněného watchdogu a druhá ukládá data z vnitřního senzoru do předpřipravené struktury.

3.6.2 Vnější jednotka

V úvodní části hlavního programu vnější jednotky provádím obdobné operace jako u vnitřní jednotky. Oproti vnitřní jednotce zde používám i enumeraci. V hlavní části programu opět zapínám 2s watchdog, vypínám všechny nepotřebné části mikrokontroleru a provádím prvotní nastavení knihoven, popřípadě komponentů.

V opakující se smyčce odesílám data v pravidelných intervalech. Tyto intervaly jsou generovány watchdog časovačem, který vždy za určitou dobu vygeneruje přerušení. Dále je ve smyčce čtení tlačítka, obnovování displeje a přechod do spánku mikrokontroleru.

Na rozdíl od vnitřní jednotky je v programu vnější jednotky více funkcí. Kromě již zmíněné funkce pro obnovení dat ze senzoru a zapnutí 2s watchdogu se v programu nachází funkce pro uspání mikrokontroleru, nebo funkce pro poskládání a následné odeslání celého datového packetu.

Poslední částí jsou obslužné rutiny pro jednotlivá přerušení. V programu je použito externí přerušení pomocí tlačítka a přerušení generované watchdog časovačem.

4 PLOŠNÉ SPOJE

Následující kapitola se věnuje návrhu a výrobě plošných spojů.

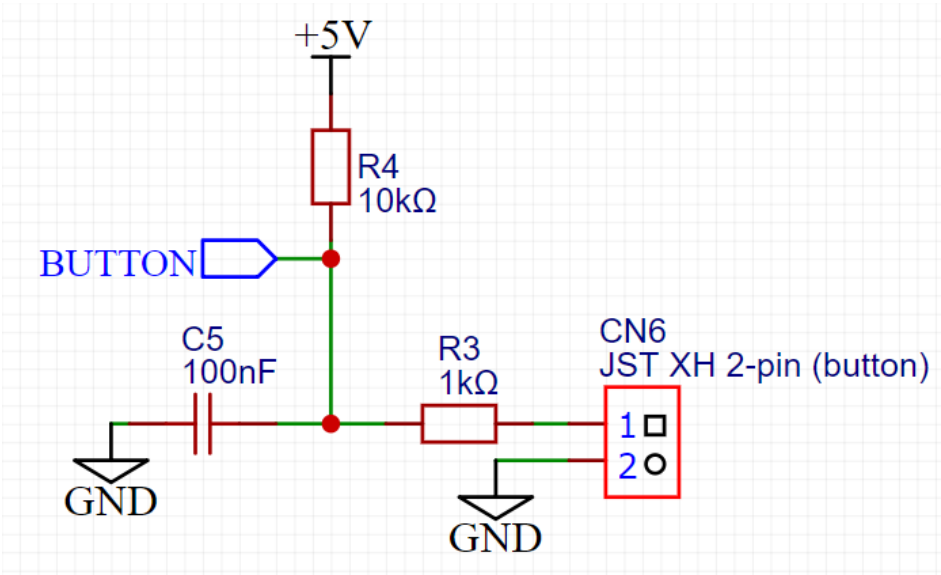
4.1 Návrh

Prvním krokem pro vytvoření plošného spoje je jeho návrh. Pro návrh jsem použil program EasyEDA. Tento program jsem použil, protože jsem si chtěl vyzkoušet práci v programu, který jsem ještě nepoužíval. Bohužel tento program je dle mého názoru určen spíše pro navržení a následné objednání plošných spojů u výrobce. Pro pouhý návrh plošných spojů bych příště zvolil rozhodně jiný program.

Nejprve jsem si vytvořil schéma zapojení. Oproti prvotnímu použití mikrokontroleru v nepájivém poli jsem přidal filtrační kondenzátory a také zatěžovací kondenzátory k samotnému krystalu. Velikost těchto kondenzátorů jsem vypočetl pomocí následujícího vztahu:

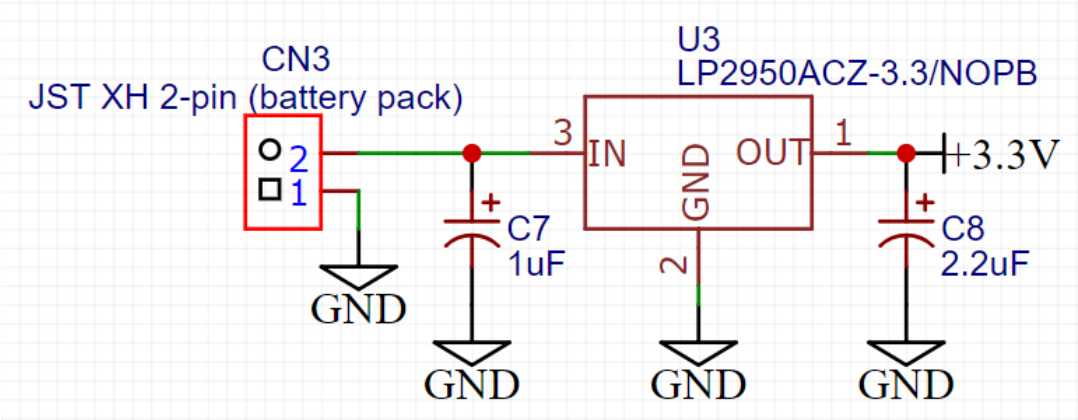
$$C_1 = C_2 = 2(C_L - C_S)$$

Po dosazení do tohoto vztahu, kde C_1 a C_2 jsou kapacity kondenzátorů, C_L je zatěžovací kapacita krystalu (udává výrobce) a C_S je kapacita cestiček (většinou se udává hodnota okolo 5 pF), jsem zjistil že použité kondenzátory by měly mít hodnotu 30 pF. Pro ošetření zákmitů tlačítek jsem použil následující zapojení. Toto zapojení vytváří RC filtr, který filtruje zákmity tlačítka.



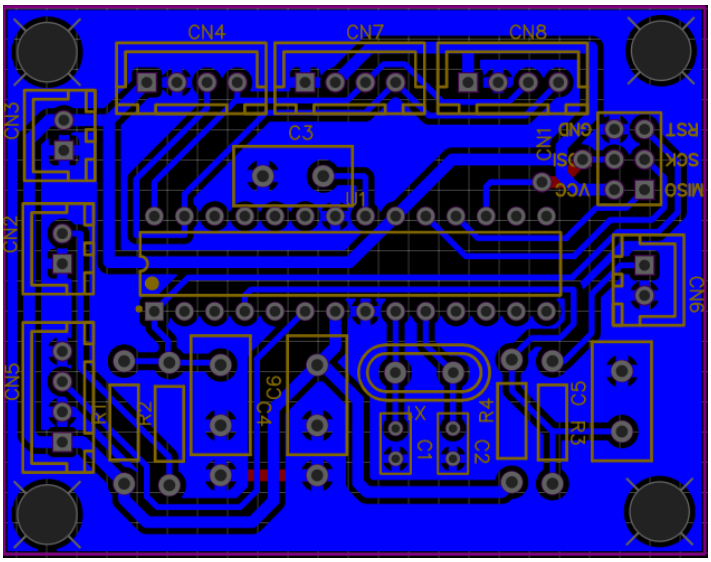
Obr. 15 Schéma zapojení tlačítka

Jednotlivé komponenty jsou připojeny k plošnému spoji pomocí JST konektorů typu XH. Na desce je také připraven ISP header pro programování mikrokontroleru. U vnější jednotky jsem navíc musel navrhnout stabilizaci napětí z baterií. Toho jsem dosáhl pomocí LDO stabilizátoru LP2950ACZ-3.3/NOPB s úbytkem napětí pouze 0,6 V a dvou elektrolytických kondenzátorů dle datasheetu stabilizátoru.



Obr. 16 Schéma zapojení LDO stabilizátoru

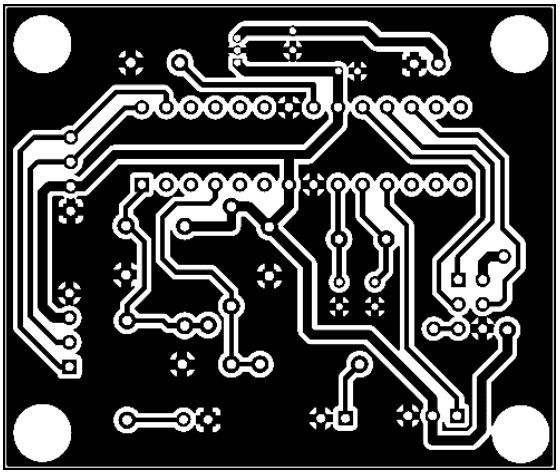
Když byly schémata zapojení hotova, mohl jsem přejí k samotnému návrhu cestiček na plošném spoji. Snažil jsem se zachovat co nejvíce kompaktní rozměry. Tloušťku cestiček jsem zvolil 1 mm pro napájecí a 0,5 mm pro datové linky.



Obr. 17 Návrh plošného spoje pro vnitřní jednotku

4.2 Výroba

Jako výrobní metodu pro zhotovení plošného spoje jsem zvolil fotocestu. Pozitiv návrhu plošného spoje jsem osvítil pomocí UV záření na fotocuprextit (cuprextit s předem nanesenou fotocitlivou vrstvou tzv. fotoresist). Jako zásadu pro vyvolání jsem použil hydroxid sodný. Následné vyleptání jsem provedl v persíranu sodném. Po vyleptání jsem desku náležitě očistil a ošetřil pájitelným lakem. Nakonec jsem vyvrtal otvory pro součástky a plošný spoj osadil dle schématu.



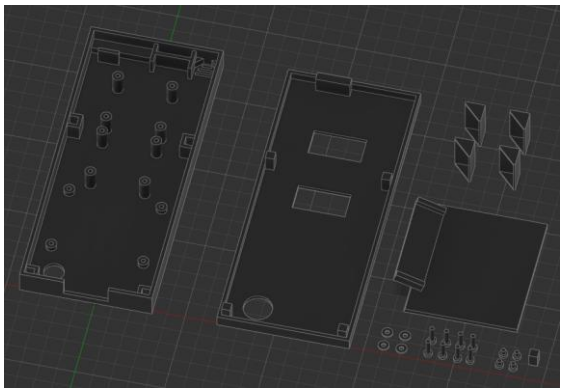
Obr. 18 Pozitiv návrhu plošného spoje vnější jednotky

Obr. 19 Osazené plošné spoje

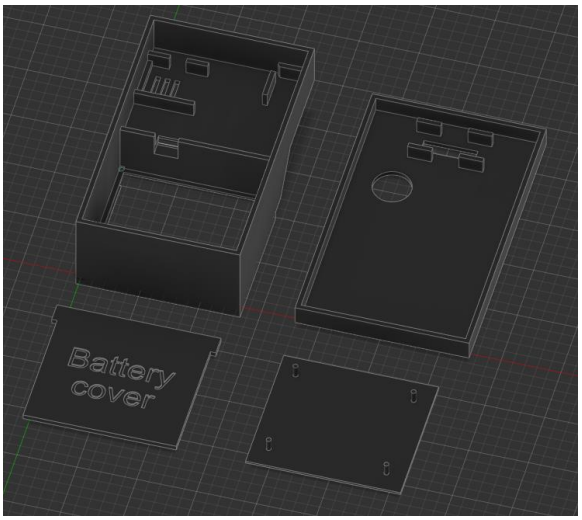
5 FINÁLNÍ VÝROBEK

5.1 Krabička

Pro plošný spoj a všechny komponenty jsem navrhl krabičku v programu Fusion 360. Modely jsem poté vytiskl na 3D tiskárně s výškou vrstvy 0,2 mm. Použitým materiálem bylo PLA. Krabička se vždy skládá ze zadního a předního krytu a dalších doplňujících částí (noha, kryt na baterie atd.).



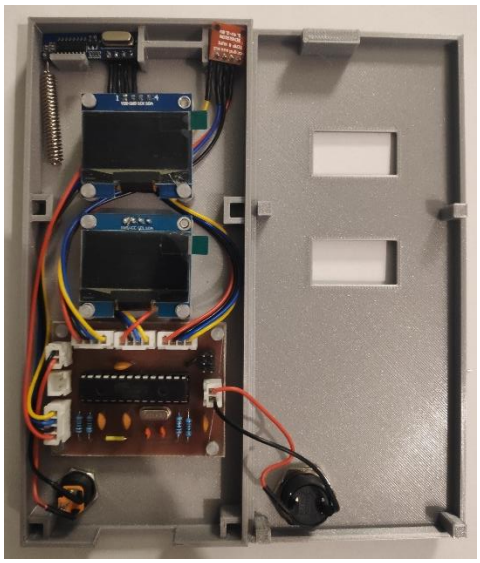
Obr. 20 Návrh krabičky pro vnitřní jednotku



Obr. 21 Návrh krabičky pro vnější jednotku

5.2 Sestavení

Posledním krokem bylo sestavení celého meteorologické stanice. Nejprve jsem do krabiček umístil plošné spoje na příslušná místa. Následně jsem do krabiček vložil a zajistil zbytek komponentů a ty připojil pomocí JST konektorů na řídící jednotku. Poté jsem sestavil zbytek krabiček a výrobek byl kompletní.



Obr. 22 Vnitřní jednotka

Obr. 23 Vnější jednotka

Obr. 24 Sestavené obě jednotky

ZÁVĚR

Celkově bych projekt návrhu a výroby meteorologické stanice hodnotil jako velmi přínosný. Během výroby stanice jsem významně obohatil své znalosti ze stavby mikrokontrolerů. Hlavním přínos mi projekt ale přinesl v oblasti programování. Celé programové vybavení stanice je napsáno v programovacím jazyce C. V průběhu vývoje programu jsem seznámil s valnou většinou klíčových slov jazyka C. Znalost těchto slov je dle mého názoru nezbytná pro využití plného potenciálu programovacího jazyka. Dále jsem se také lépe zorientoval v jednotlivých datových typech a zjistil jak jsou uchovávány v paměti mikrokontroleru. Nemalým přínosem byla také zkušenost s návrhem a výrobou plošných spojů.

Zlepšil jsem také svou schopnost organizace času a také schopnost rozdělit si práci do jednotlivých bloků. Na těchto blocích poté pracovat samostatně až do finálního složení těchto bloků do jednoho funkčního celku.

Pokud bych měl vznést nějaký návrh na vylepšení. Tímto návrhem by bylo jednoznačně zpracování celé meteostanice včetně senzorů ve formě SMD součástek. To by umožnilo výrazně menší rozměry jednotlivých jednotek a také zjednodušilo jejich vnitřní stavbu. Bohužel výroba a především osazení takového plošného spoje je značně obtížné bez potřebného vybavení.

Výsledkem mého snažení je tedy plně funkční meteorologická stanice pro domácí použití, kterou plánuji i nadále používat.

SEZNAM POUŽITÉ LITERATURY

[1] Weather station. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-03-26]. Dostupné z: https://en.wikipedia.org/wiki/Weather_station

[2] Microcontroller (MCU). TechTarget [online]. [cit. 2022-03-26]. Dostupné z: https://internetofthingsagenda.techtarget.com/definition/microcontroller

[3] ATmega328p. Microchip [online]. [cit. 2022-03-26]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[4] SHT31. Sensirion [online]. [cit. 2022-03-26]. Dostupné z: https://sensirion.com/media/documents/EA647515/61641D0C/Sensirion_Humidity_Sensors_SHT3x_Datasheet_analog.pdf

[5] BME280. Bosch-sensortec [online]. [cit. 2022-03-26]. Dostupné z: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf

[6] SH1106. Velleman [online]. [cit. 2022-03-26]. Dostupné z: https://www.velleman.eu/downloads/29/infosheets/sh1106_datasheet.pdf

[7] SSD1306. Adafruit [online]. [cit. 2022-03-26]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

[8] CRC. Sunshine2k [online]. 2015- [cit. 2022-03-26]. Dostupné z: http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

[9] VirtualWire. Github [online]. [cit. 2022-03-26]. Dostupné z: https://github.com/song940/VirtualWire

SEZNAM OBRÁZKŮ

PŘÍLOHY

V přílohách na školním EDU serveru lze nalézt kompletní programové vybavení obou jednotek meteorologické stanice a STL soubory krabiček. Zároveň je zde dostupné video dokumentující funkci meteostanice.