Střední průmyslová škola elektrotechnická, Havířov, CZECH REPUBLIC
Zespół Szkół Technicznych, Mikołów, POLAND

METEOSTANICE

Co-funded by the Erasmus+ Programme of the European Union

Student: Bořek Tuleja
Tutor: Ladislav Opiol

Student: Marek Krosny
Tutor:

Obsah

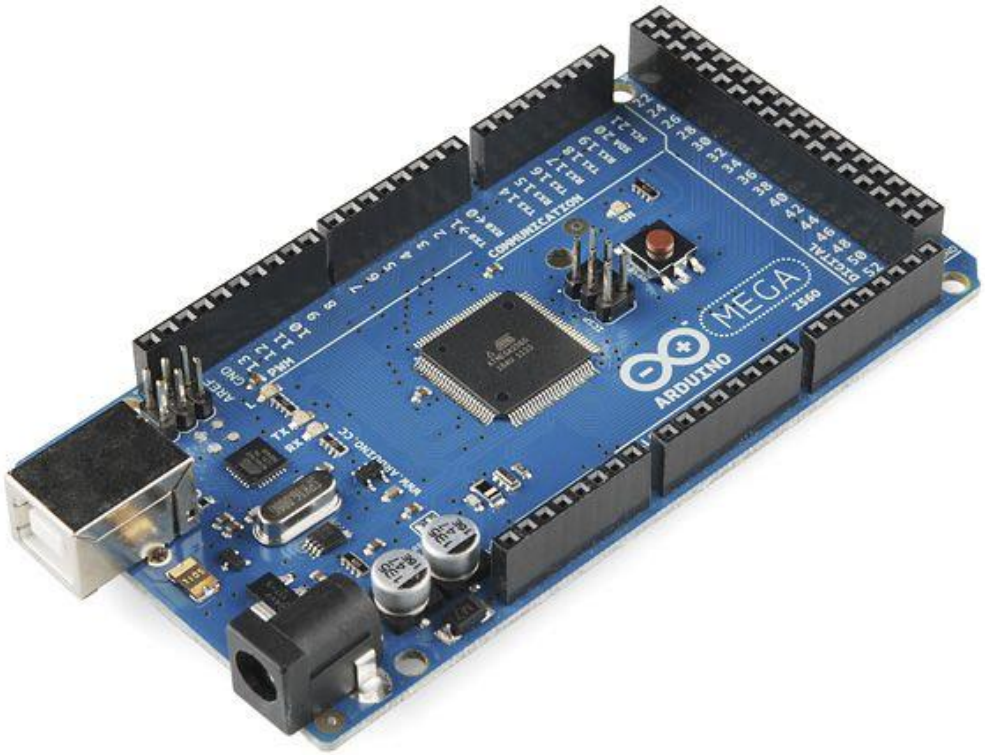
2 Úvod

Cílem našeho projektu bylo sestavit meteostanici, která je schopna měřit teplotu, tlak, vlhkost a rychlosti and směr větru. Používáme 2 typy senzorů pro čtení potřebných dat a LCD displej pro jejich zobrazení.

Meteostanici je možno použít k získání informací o veličinách zmíněných nahoře. Měla by být umístěna někde venku, protože uvnitř budov není žádný vítr, ale je to rozhodnutí majitele.

Bořek:
Zvolil jsem si tento projekt, protože jsem nikdy předtím na ničem podobném nepracoval. Na druhou stranu je velmi jednoduché zjistit, zdali vše funguje tak jak má na základě jiných, již funkčních zařízení, které umí číst informace o počasí. Zároveň jsem, ale také během vývoje narazil na spoustu problémů, které jsem úspěšně vyřešil a získal tak nové zkušenosti.

Marek:
Projekt jsem si vybral, protože vypadal zajímavě a chtěl jsem se naučit něco nového o Arduinu, měření, počasí, programování v C a nerad moc pracuji.



3 Teoretická část

3.1 Průzkum trhu

Většina ostatních meteostanic, které si je možno koupit na internetu poskytují více funkcí a mohou ukazovat například také čas či datum.

Každopádně, ne každá stanice umí měřit rychlost a směr větru. To je jedna z výhod naší meteostanice. Také protože jsme použili operační systém reálného času, každá změna, třeba teploty, vlhkosti atd. se zobrazí okamžitě bez zpoždění.

Mezi cenami jednotlivých meteostanic jsou velké rozdíly. Ty menší se pohybují od 20 Euro a cena následně stoupá i s velikostí třeba až na 200 Euro. Vetší stanice jsou také schopné změřit vlastnosti větru, stejně jako ta naše, ale jejich cena je většinou větší něž by stál náš projekt.

Mezi nevýhody bych začadil velikost, protože používáme dvě krabice, které jsou spojeny přes kabel, který je možno velmi lehce přeseknout a také se všechna měřená data nezobrazují najednou.

Našel jsem také produkty, které umožnují připojit se k internetu, nebo Wi-Fi.

Myslím si však, že je náš projekt vhodný pro venkovní měření za férovou cenu a jsme schopni změřit a zobrazit většinu důležitých dat.

3.2 Technický rozbor

Našim cílem bylo vytvořil použitelnou meteostanici, ale zároveň nechat věci tak jednoduché, jak to jen jde. Vybrali jsme senzory, které jsou schopné měřit více veličin najednou.

3.2.1 Měření informací o počasí

Pro měření všech potřebných dat o počasí používám senzor BME280. Je velmi jednoduchý na použití, protože generuje okamžité hodnoty v digitální podobě. Jediné co jsme museli udělat je připojit ho k mikrořadiči a mohli jsme měřit.

3.2.2 Výpočet rychlosti větru

Jednou z úloh které jsme měli vyřešit byl výpočet rychlosti větru a zobrazení této hodnoty uživateli. Pro vyřešení tohoto úkolu jsme se rozhodli použít hallovy sondy. Tyto senzory generují buď logickou „1“, nebo „0“. Generovaný výstup je ovlivněn magnetickým polem. Celé problematika hallových senzorů bude rozebrána později. Vítr tedy otáčí směrovou šipkou na meteostanici a ta otáčí magnet skrytý pod ní. Jak se magnet otáčí, ovlivňuje výstup hallovy sondy. Program registruje moment ve kterém senzor začal generovat logickou „0“ (časový údaj). Vítr otáčející šipka otáčí i magnet. Když senzor začne generovat logickou „1“, tento moment se opět registruje a nyní víme, že až se na výstupu sondy opět objeví logická „0“, můžeme výpočet ukončit a začít znova. Když je tedy výpočet ukončen, vyvodíme si za jak dlouho se magnet otočil kolem. Teď jsem musel použít trochu fyziky.

Pro výpočet rychlosti můžete použít tuto rovnici, pokud znáte dráhu a čas:

$$v = \frac{s}{t}$$

v … rychlost (m / s)
s … dráha (m)
t … čas (s)

V tento okamžik jsem měl pouze čas. Jednoduše jsme tedy změřili poloměr kruhu kolem kterého se magnet otáčí. S těmito hodnotami můžete vypočítat obvod kruhu – v našem případě je obvod roven hledané délce.

$$O = 2 \times \pi \times r$$

O … obvod (m)
r … poloměr (m)

Získal jsem všechny hodnoty potřebné k vypočítání rychlosti větru. Mimochodem finální čas, který jsme vyhodnotili byl v milisekundách, takže jsme je museli převést na sekundy.

Každopádně, toto řešení je složité, protože je potřeba sledovat výstup sondy neustále. Toho nebylo možné, protože jsme v programu používali tzv. „delay“ funkce. Tyto příkazy zastaví program na určitou dobu a čekají na dané instrukci. Pokud by jsme to tak nechali, mělo by to obrovský dopad na můj výpočet, protože by během počítání vznikali vysoké odchylky. K řešení tohoto problému jsem se rozhodl napsat si vlastní operační systém reálného času, který bude představen níže.

3.2.3 Výpočet směru větru

Pro výpočet směru větru jsme použili obdobnou metodu jako předtím. Máme 8 hallových senzorů, které fungují úplně stejně jako senzor pro výpočet rychlosti větru. Každý z těchto senzorů představuje jednu světovou stranu na kompasu.

Směr
Sever
Severovýchod
Východ
Jihovýchod
Jih
Jihozápad
Západ
Severozápad

Když fouká vítr, otáčí dalšími 3 magnety blízko sebe. Tyto magnety opět ovlivňují výstupní hodnoty 8 senzorů. Magnet uprostřed má opačný pól oproti 2 zbylým magnetům. Když se tedy 3 magnetky točí, ten prostřední mění výstupní hodnotu sond na logickou „0“ a ostatní 2 přepínají předchozí a následující sondu do logické „1“. Senzor generující logickou „0“ signalizuje směr, kterým fouká vítr.

3.2.4 Zobrazení dat

Na zobrazení vypočtených dat jsme použili jednoduchý LCD displej (20x4). Po změření stačilo data jen poslat na displej.

3.2.5 Jak vše komunikuje

Všechny části projektu mezi sebou komunikují, protože jsou připojeny přímo k mikrořadiči.

3.3 Použité technologie

3.3.1 Operační systém reálného času

Jak jsem se již zmínil nahoře, museli jsme dosáhnou efektu multitaskingu pro úspěšný výpočet rychlosti větru. První věc co mě napadla, když jsem přemýšlel nad řešením tohoto problému, bylo použít operační systém reálného času. Bohužel pro Arduino není žádný oficiální multitaskingový systém, takže jsem byl donucen si napsat vlastní.

Opravdové multitaskingové systémy používají takzvaný „Process control block“ pro manipulaci běžících procesů, jejich spuštění na procesoru atd. Na Arduinu není možnost programovat na této úrovni, takže jsem se rozhodl vytvořit „falešné“ rešení.

Můj systém umožňuje tvorbu vlastních tasků a jejich spuštění. Každý task obsahuje 3 záznamy.

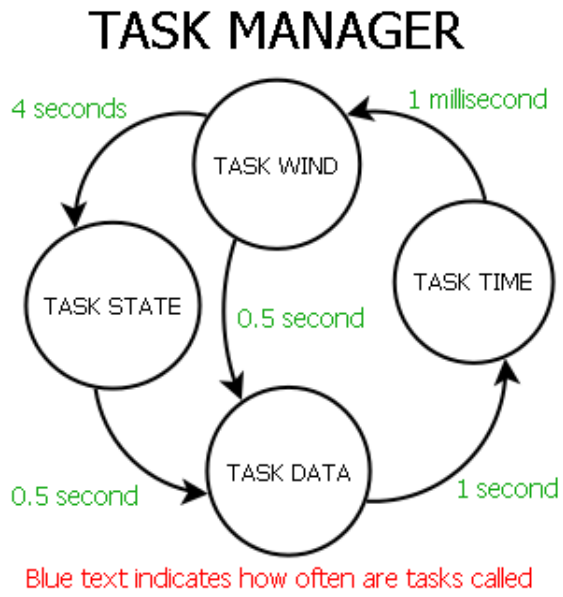
Záznam	Datový typ	Popis
Function	Pointer na funkci	Z jakou funkcí je task spojen
DelayTime	Unsigned long	Jak často je task volán
LastRepeatTime	Unsigned long	Poslední čas, kdy by task volán

Potom je zde manažerský objekt. Ten má vlastní frontu čekajících tasků, které jsou připraveny ke spuštění. Manažerský objekt umožňuje task do fronty přidat, odebrat jej, nebo uvolnit z paměti. Objekt také opakovaně volá všechny tasky ve frontě a ptá se každého z nich, zda je možné ho volat. Task je spuštěn, pokud je momentální čas mínus LastRepeatTime tasku menší, nebo větší než DelayTime tasku. Toto znamená, že jsou tasky spouštěny v časových intervalech.

Toto není opravdový multitasking, protože všechny tasky jsou spouštěny postupně a program čeká dokud se celý neprovede a zároveň projet celou frontu než se task spustí znova. Pro nás to, ale bylo dostačující, protože se teď nepohybujeme mezi odchylkami v sekundách, ale mikrosekundách. Výsledek výpočtu je tedy velmi blízko realitě.

Verze mého systému je stále velmi jednoduchá a podporuje užití jen 8 tasku v 1 frontě na manažérský objekt.

V programu používám 4 tasky, abych všechno obsloužil.



1. Task state – tento task mění stav program. To znamená, že jsou v každém stavu zobrazená jiná data na displeji.
2. Task data – task se ptá na nová data senzoru BME280.
3. Task time – task pro výpočet uplynulého času od začátku programu.
4. Task wind – task použitý pro výpočet rychlosti větru.

3.4 Finanční analýza

3.4.1 Rozpočet

Zde je tabulka všech částí, které jsme koupili na internetu a jejich cena v Eurech.

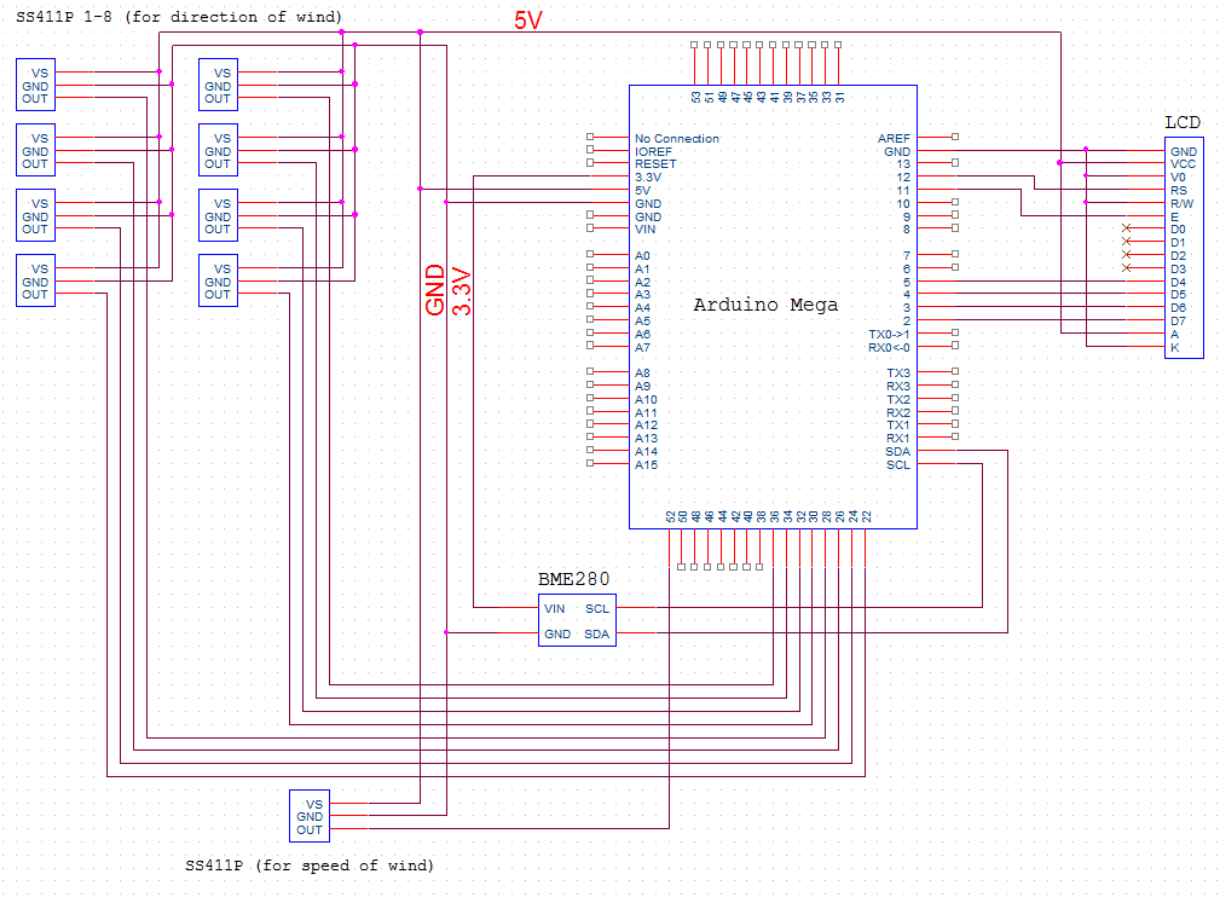
Jméno	Cena
Arduino Mega2560 Clone	15 €
BME280	11 €
9x SS411P	5,75 €
LCD	5, 72 €
Vodiče	2 €
Vše dohromady	39,47 €

3.4.2 SWOT analýza

Silné stránky	Slabé stránky
Vysoká přesnost Rychlá odpovídá Pěkný vzhled RTOS	Krabice jsou velké Meteostanice je tvořena 2 krabicemi
Příležitosti	**Nebezpečí**
Vylepšení senzorů a procesoru Rozšíření Propojení s internetem Ukládání dat do databáze	Tvorba nových a lepších senzorů

4 Praktická část

Toto je schéma našeho projektu. Je zde ukázáno jak jsou všechny součásti vzájemně propojeny.



4.1 Elektronika

Nejprve jsem se rozhodl vypsat tabulku veškeré použité elektroniky, součástí, periferií a senzorů a následně je jednotlivě popsat.

Jméno	Typ
Arduino Mega2560	Mikrořadič
BME280	Senzor
SS411P	Senzor
Arduino LCD 20x4	Displej

4.1.1 Arduino Mega2560

Toto je mikrořadič, který jsme použili pro naprogramování a propojení všech součástek. Museli jsme použít verzi Mega, protože má hodně digitálních pinů a také spoustu paměti pro kód.

Arduino mikrořadiče je možno programovat v jazycích C/C++. Výrobce tohoto mikrořadiče také napsal speciální knihovnu, která obsahuje funkce pro programování této desky s procesorem. Pro programování jsem použil oficiální IDE přímo od Arduina, které nám v mnohém pomohlo, třeba při lazení.

Arduino Mega2560 generuje 5V a 3.3V. Použili jsme obě napětí k napájení připojených komponent. Tyto části musí také být uzemněné.

Digitální piny 22 – 36 a 52 jsou použity jako vstup pro výstup z hallových sond SS411P.

4.1.2 BME280

BME280 je digitální senzor, který je schopen číst teplotu, vlhkost a tlak. Byl vyvinut firmou Bosch. Senzor umožnuje komunikaci pomocí sběrnice I2C, nebo SPI. Rychlost komunikace se může pohybovat až na frekvenci 3,4 MHz, pokud je použita I2C sběrnice, nebo 10 MHz při užití SPI. Pracovní rozsah je od -40 °C do +85 °C pro teplotu, 0 – 100% pro vlhkost a 300 – 110 hPa pro tlak. Odchylka měření teploty je +/- 1 °C, +/- 3% vlhkost a +/- 1 Pa u tlaku. Napájecí napětí je 1,71 – 3,6 V.

Senzor má 4 piny. Napájení (VIN), které jsme připojili na 3,3 V pin mikrořadiče. Zem (GND). SDA, což je Seriál Data In / Master Out Slave In pin, který senzor používá pro posílání dat. SCK je pin pro hodinový signál SPI.

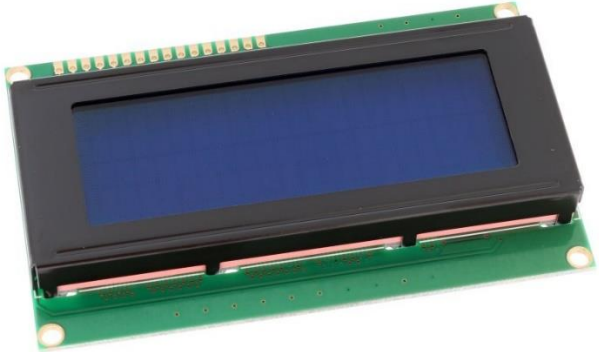
Součástka je vysoce efektivní, potřebuje jen velmi malé napájení a je optimalizovaná, pro měření co nejpřesnějších hodnot z co nejmenší odchylkou. Čas potřebný pro měření je velice malý (12 ms v průměru).

Použili jsme fanouškovskou knihovnu, která komunikuje ze senzorem a poskytla nám velice jednoduché funkce, které čtou data.

4.1.3 SS411P

Malé, univerzální digitální zařízení, operující pomocí magnetického pole. Tento senzor reaguje pokaždé, když se v blízkosti nachází magnet. Senzory jsou bipolární a mají zvýšenou citlivost, takže jsou vhodné i pro použití s menšími a levnějšími magnety. Malé, olovnaté a ploché TO-92 (SS411P) nezabírá moc prostoru na plošné spoji. Citlivé, bipolární magnety reagují na změnu severního a jižního pólu a to dělá tento produkt vhodný pro RPM měření. Vestavěný pull-up rezistor umožňuje zařízení jednoduše použít s jinými součástkami bez nutnosti přidávat jiné zařízení, což ve finále snižuje cenu. RoHS materiál je v souladu se směrnicí 002/95/EC.

4.1.4 Arduino LCD 20x4



LCD displej, který jsem vybral má modré podsvícení, 4 řádky a 20 znaků na řádku. Napájení je 4 – 5,5 V, takže naší jedinou možností bylo displej připojit k 5 V. LCD je schopno vypsat základní ASCII znaky, které jsou uloženy v paměti, a také znaky čínské. Displej má 16 pinů a 8 jich je použito jako datové sběrnice. To znamená, že může zobrazovat až 256 rozdílných znaků (2^8). Všechny funkce, které je LCD schopno vykonávat potřebují jen pár mikrosekund pro zpracování. Použitý materiál je sklo, organický tmel, organické kapaliny a polarizér.

Ostatní piny jsou RS a R/W, které LCD říkají, jaké instrukce se mají právě vykonávat. Potom E, což je povolení displeje, a nakonec anoda a katoda pro napájení podsvícení.

Pin	Level	Funkce
RS	log. „0“	Povolí instrukční mód
RS	log. „1“	Povolí datový mód
R/W	log. „0“	LCD je v režimu zápisu
R/W	log. „1“	LCD je v režimu čtení

Rozdíl mezi instrukčním a datovým módem je ten, že v případě povolení instrukčního módu je vstup displeje reprezentován jako instrukce. To znamená třeba „vymaž displej“, nebo „přesuň kurzor“. V datovém módu je veškerý vstup reprezentován jako 8 bitové znaky.

4.2 Program

Pro naprogramování stanice jsme použili jazyk C++, oficiální knihovny i knihovny třetí strany. C++ je velmi rychlý, objektově orientovaný jazyk, který se používá pro programování procesorů na celém světe.

Protože program pracuje s mnoha hodnotami, které jsou reprezentovány ve formě čísla, rozhodl jsem se všechny tyto hodnoty reprezentovat pomocí konstant, tak, aby byl kód přehlednější a já věděl co zrovna nastavuji. To je také jeden z důvodů, proč jsem použil Arduino Mega2560. Spousta paměti je vhodná pro zavedení konstant, enumů a jiných struktur.

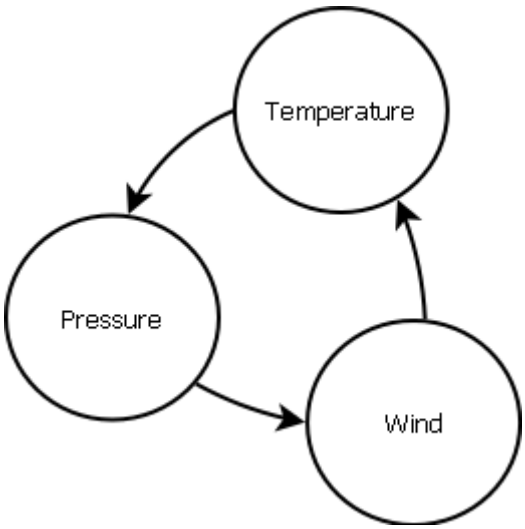
4.2.1 Stavy programu

Kromě tasků jsem program rozdělil na stavy. Celkem jsem použil 3 stavy, a v každém z nich jsou na displeji zobrazována jinačí data. Tady je tabulka všech stavů:

Stav	Co je zobrazeno na LCD
Teplota	Teplota a vlhkost
Tlak	Tlak
Vítr	Rychlost a směr větru

Definoval jsem si enum stavů, abych se vyvaroval používání čísel a kód byl tak čitelnější.

Základním stavem programu je „Teplota“. Stav programu se mění pokaždé, když je volán stavový task. Používám switch větvení, abych zjistil, v jakém stavu se program nachází a následně stav přepnul na následující hodnotu. Další diagram ukazuje, jak se stavy přepínají:



Protože je stavový task volán co 4 sekundy, také stav programu se mění co 4 sekundy.

4.2.2 Operační systém reálného času

Abych naprogramoval operační systém reálného času, rozhodl jsem se vše kontrolovat pomocí tzv. „Manažerského objektu“. Ten je představen jako instance C++ třídy, která umožňuje správu tasků a dalších operací. Zde je tabulka všech funkcí, které je možno volat na instanci manažerského objektu:

Funkce	Popis
CreateTask	Alokuje v paměti nový task a vratí na něj pointer
AddTask	Přidá task do fronty tasků manažerského objektu
RemoveTask	Odstraní task z fronty manažerského objektu
FreeTask	Uvolní task z paměti
Run	Provede veškeré připravené tasky ve frontě

Manžerský objekt také obsahuje frontu tasků. Každý vytvořený task může být do této fronty přidán a následně spuštěn. Pokud již task není potřebný po určitou dobu, je možno jej z fronty vyhodit, nebo přímo uvolnit z paměti. Fronta tasků je pole pointrů na strukturu tasku. Protože je pole staticky alokováno, jeho velikost je pouze 8, c čehož ted vyplývá, že každá fronta může obsahovat pouze 8 tasků.

Pro reprezentaci tasků v programu jsem zvolil formu struktury, protože zde není potřeba používat žádné funkce, pouze proměnné. Každá z proměnných ve struktuře tasku již byla popsána.

Systém také deklaruje nový datový typ – pointer na funkci. To je vhodné řešení jak tasku přidělit funkci, které je s ním spojena.

Když manažerský objekt provádí všechny task, nejsou volány „paralelně“, ani se chvilkově nestřídají na procesoru. Jsou prováděny postupně, sekvenčně, ale nejsou zde žádné spožďovací funkce, takže vzniká iluze multitaskingu. Největší výhodou tohoto řešení je, že během provádění tasků nevznikají žádné kritické sekce, takže není potřeba zavádět nástroje jako mutexy, semafory a jiné řešení, pro ošetření globálních proměnných.

When the manager executes all the tasks in its front they are actually not called „parallelly“ nor the processor switches between them. They are all executed sequentially but there are no delays at all so it

creates an illusion of multitasking. The biggest advantage of this solution is the fact that no critical sesctions are made through the executing so there are no mutexes, semaphores and other solutions needed.

4.2.3 Tasky

State task – Tento task již byl představen výše. Je volán co 4 sekundy a slouží ke změně stavu programu. Používá switch strukturu pro zjištění stavu programu a následné vyhodnocení stavu následujícího. Task také vymaže celý LCD displej.

Data task – Task je zodpovědný za výpis dat na LCD displej. Je volán, co půl sekundy, aby uživatel dostal co nejpřesnější naměřená data za krátkou dobu od změny, ale zase aby funkce nebyla volána neustále. Task také využívá stav programu, aby věděl, jaká data má vypsat na LCD displej. Zobrazovaná data jsou čtena ze senzoru BME280, když je task aktivní.

Time task – V tomto tasku probíhá výpočet času, který uběhl od spuštění programu. Je volán každou sekundu, protože nejmenší zobrazovaná jednotka času jsou sekundy. Funkce využívá matematických řešení a rovnice pro převod milisekund na formát hodiny-minuty-sekundy a potom časový údaj vypíše na LCD.

Wind task – Toto je asi nejdůležitější task, protože je zodpovědný za správný výpočet rychlosti směru větru. Metoda výpočtu již byla popsána a zde je naprogramována. Protože chceme rychlost co nejpřesnější, task je volán co 10 milisekund.

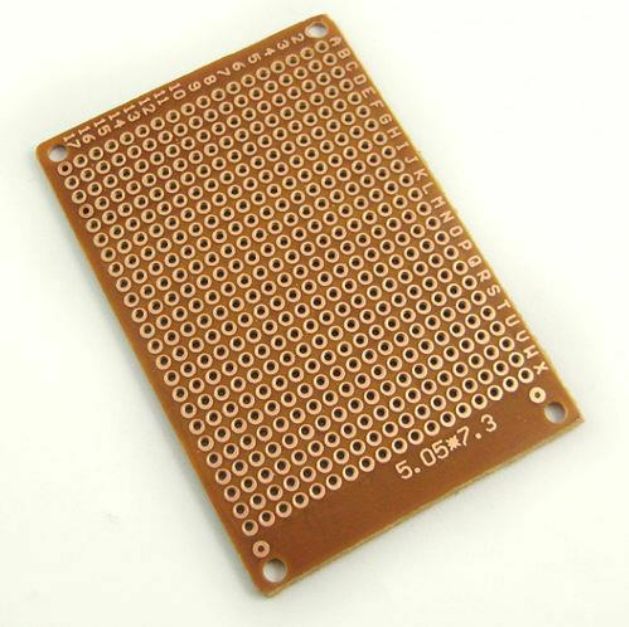
4.3 Mechanická část

Meteostanice se skládá ze dvou částí krabic. V první z nich je vložen plošný spoj, procesor, LCD displej a senzory. Druhá krabice obsahuje mechanismy a senzory pro výpočet směru a rychlosti větru.

Část s procesorem byla zhotovena českou stranou a krabice s hallovými sondami dělala polská skupina.

4.3.1 Krabice s plošným spojem

Abychom dali veškeré zbývající části do krabice, museli jsme vyměřit rozměry a rozmístit součástky tak, aby vše fungovalo. LCD displej jsme museli dát na mikrořadič, protože na vrchu krabice je pro něj vyřezaná díra. Dostali jsme zelenou desku vhodnou pro vytváření plošných spojů a pájení a k ní jsme mikrořadič upevnili. Abychom získali data z druhé krabice, použili jsme velký kabel s 12 vodiči. 9 vodičů bylo použito jako výstup hallových sond a dva další pro 5V a zem. Kvůli použití pouze 2 vodičů pro 5V a zem jsme museli spojit a spájet všechny vodiče hallových sond určených pro napájení a uzemnění s kabelem. Protože je senzor BME280 připojen přímo k mikrořadiči, žádné speciální úprava zde nebyla nutná. Museli jsme však vodič 5V a zem z kabelu připojit k LCD a mikrořadiči. Udělali jsme plošný spoj, který se vše propojilo. Datové vodiče z hallových sond jsou připojeny přímo k Arduinu. Když jsme byli hotovy ze vším pájením, desku s plošným spojem jsme upevnili v krabici.



4.3.2 Krabice s hallovými sondami

Nejprve jsem se podíval po kabelech, které se už nepoužívají. Potom jsem vytvořil projekt, který ukazoval jak bude krabice vypadat. Vyřezal jsem dvě díry pro otáčející se motory a ještě jednu pro vyvedení kabelů. Také jsem vyřezal díru pro LCD displej. Na vytvoření šipky, které určuje směr větru a ostatní plastové části pro rychlost větru jsem použil frézku, abych je vyřezal z plastových plátků. Mnoho částí je znovu použito z jiných zařízení. Třeba magnety a krabice.

4.4 Testování

V průběhu vývoje jsme udělali mnoho testů, abychom zjistili zda vše funguje, tak, jak má.

Začali jak nejjednodušeji, jak to jen jde – ze senzorem BME280. Součástka je velmi jednoduchá na použití, ale paradoxně jsme zde ztratili nejvíce času a to protože jsme použili verzi Arduino Mega. Tento mikrořadič je trochu jinačí od verze Uno, kterou senzor nativně podporuje a všechna schémata zapojení jsou tedy pro ni. Mega má potřebné registry na jiných pinech a strávil jsem zde alespoň 20 minut, než jsem vyřešil problém, proč nemůžeme získat žádná data. Když jsem však senzor zapojil správně, vše už fungovalo.

Potom jsme se rozhodli přesunout na LCD displej. S tím nebyli žádné potíže, protože, když je LCD správně zapojeno, začne svítit zadní světlo. Museli jsme LCD jen napájet a zobrazit pár znaků.

Poslední testovanou částí bylo zda hallovy senzory generují nějaký signál. Napsal jsme menší program, který konfiguruje potřebné piny pro výstup hallových sond a potom čte hodnotu na těchto pinch and zobrazuji ji na LCD displeji. Vše fungovalo a mohli jsme se přesunout k dalšímu programování.

Když bylo vše správně zapojeno a propojeno napsal jsem program a přesunuli se k testování finálního projektu. Opět se neobjevili žádné problémy.

5 Závěr

Na závěr bych řekl, že náš projekt byl úspěšný. Naprogramovali jsme všechno potřebné chování, vytvořili model a výsledek otestovali.

Bořek:

Dle mého byla meteostanice úspěch, hlavně protože všechny mé výpočty a rovnice fungují a také protože jsem byl donucen si napsat vlastí RTOS, čímž jsem získal nové zkušenosti v oblasti programování. Ale teď vidím, že by projekt mohl být mnohem lepší, kdybych přidal více částí. Jednou z nich je třeba časový modul, který by umožnil zobrazovat momentální čas, nebo i datum. Další částí, co by mohla být změněna je LCD. Chtěl jsem použít typ mobilního displeje, abych projekt vizuálně zdokonalil a vypadal profesionálněji a zobrazovaný text by vypadal pěkně s barvami, fonty a jinou velikostí textu. Posledním nápadem pro rozšíření je implementace Wi-Fi modulku. Protože znám jazyky jako PHP, HTML, CSS a trochu JavaScript, mohli bychom vytvořit webovou stránku a zobrazovat naměřená data zde. Také by se mohla vytvořit databáze pro prohlídnutí historie měření.

V budoucnosti bych chtěl tyto rozšíření přidat, ale všechno záleží na faktu, zda je v krabici dostatek prostoru. Také nevím jak přesně funguje Wi-Fi modul na Arduinu, takže bychom možná potřebovali klávesnici pro ovládání.

6 Odkazy

Informace o LCD displeji:
https://www.hwkitchen.cz/user/related_files/lcd-displej-20x4-modry-s-podsvetlenim-ds-c2004a3-btsswe-naa.pdf

Informace o všech halových sondách:
https://sensing.honeywell.com/index.php?ci_id=45288

Informace o senzoru BME280:
https://navody.arduino-shop.cz/navody-k-produktum/senzor-bme280-mereni-teploty-relativni-vlhkosti-a-barometrickeho-tlaku.html

RTOS:
https://en.wikipedia.org/wiki/Real-time_operating_system

Všechno o Arduinu, pinech a jeho verzích
https://www.arduino.cc/

Informace o ostatních meteostanicích na trhu
https://www.google.com/search?q=weather+station&client=firefox-b-d&source=lnms&tbm=shop&sa=X&ved=0ahUKEwirkoDz_7HhAhUbTRUIHeiJD1EQ_AUIDygC&biw=946&bih=958

7 Přílohy

Toto byl můj nápad pro vytvoření metody na výpočet rychlosti větru: