FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Platforma pro měření efektivity komunikace mikrokontroleru Arduino s externími senzory
Student:	Matěj Kocourek
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1) Prozkoumejte existující řešení pro měření efektivity komunikace při obsluze většího množství senzorů.
2) Navrhněte vlastní zařízení umožňující připojení většího množství externích senzorů k platformě Arduino.
3) Navržené řešení bude umožňovat analýzu využité paměti, procesorového času a elektrické energie v závislosti na počtu připojených externích senzorů.
4) Pro komunikaci mezi mikrokontrolerem Arduino a externími senzory použijte protokol RS485.
5) Navržené řešení zrealizujte a řádně otestujte.
6) V jazyce C++ napište obslužnou aplikaci Arduina.
7) Pro účely testování a analýzy spotřeby zdrojů vytvořte aplikaci v C++ pro OS Windows emulující velké množství senzorů.
8) Na základě získaných výsledků navrhněte algoritmus pro efektivní obsluhu velkého množství senzorů s pomocí mikrokontroleru Arduino.

Elektronicky schválil/a doc. Ing. Jan Janoušek, Ph.D. dne 16. února 2022 v Praze.

Bakalářská práce

PLATFORMA PRO MĚŘENÍ EFEKTIVITY KOMUNIKACE MIKROKONTROLERU ARDUINO S EXTERNÍMI SENZORY

Matěj Kocourek

Fakulta informačních technologií
Katedra teoretické informatiky
Vedoucí: Ing. Pavel Kubalík, Ph.D.
9. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především vedoucímu této práce, Ing. Pavlovi Kubalíkovi, Ph. D. za veškerou pomoc, důležité rady a ochotu mi pomáhat tuto práci dokončit. Dále chci poděkovat mým rodičům za jejich podporu a za to, že celou dobu drželi se mnou a byli mi nápomocni, když se zrovna nedařilo. Nakonec bych rád poděkoval svým přátelům a spolužákům, a to zejména Matoušovi Kulhanovi a Matějovi Frnkovi, bez kterých by příprava na zkoušky a studium celkově neprobíhalo tak hladce a za to, že mi pomohli tuto práci zkontrolovat. Díky vám všem.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2022 .....

Abstrakt

Práce se zabývá analýzou a měřením sítě senzorů a jiných modulů, centrálně řízenou mikrokontrolérem Arduino (modely Uno a Mega 2560) přes RS485. Je navrhnut jednoduchý komunikační protokol, který je následně použit pro účely práce a vyvinut software pro desku Arduino, který protokol využívá. Pro účely měření vytíženosti systémových prostředků desky Arduino je vytvořena aplikace na PC emulující jednotlivá zařízení v senzorové síti. Takto navržené řešení je doplněno o monitorovací prvky hardwarového i softwarového typu a jsou provedeny testy vytížení systémových prostředků. Měření různých druhů scénářů zatížení centrální desky jsou zanalyzovány a porovnány. Výsledkem jsou poznatky o dopadu provozování takovéto sítě na zatížení centrálního zařízení a rozhodovací algoritmus, jak postupovat při její implementaci pro vlastní využití.

Klíčová slova senzorová síť, monitorování zatížení, komunikace vestavěných systémů, externí moduly, analýza spotřeby zdrojů, Arduino

Abstract

This thesis deals with an analysis and measurement of a sensor network driven by the Arduino microcontroller (models Uno and Mega 2560) over RS485. For the purposes of this thesis, a simple communication protocol is designed alongside with a software for the Arduino board implementing this protocol. A PC application emulating sensors is created for resource monitoring of the Arduino board. This solution is then extended with both hardware and software monitoring elements and is subjected to resource consumption tests. The measurements of different load scenarios are analyzed and compared between each other. The thesis result consists of various observations about the use of the Arduino microcontroller for driving a sensor network, and a decision algorithm describing how to proceed when implementing network such as this for own use.

Keywords sensor network, resource monitoring, embedded systems communication, external modules, resource consumption analysis, Arduino

Seznam zkratek

IoT	Internet of Things (internet věcí)
CPU	central processing unit (centrální procesorová jednotka)
RAM	random-access memory (paměť s náhodným přístupem)
EEPROM	electrically erasable programmable read-only memory (elektricky vymazatelná paměť pouze pro čtení)
I/O	input and output (vstup a výstup)
PWM	pulse width modulation (pulzně šířková modulace)
UART	universal asynchronous receiver-transmitter (univerzální asynchronní přijímač-vysílač)
PC	personal computer (osobní počítač)
USB	universal serial bus (univerzální sériová sběrnice)
API	application programming interface (rozhraní pro programování aplikací)
UI	user interface (uživatelské rozhraní)

Úvod

Chytrá zařízení a systémy jsou v dnešní době čím dál rozšířenější, ať už jde o chytrou domácnost, bezdrátové systémy nebo podnikové sítě. Čím dál více se spoléháme na různé druhy senzorů a jinou elektroniku, která nám pomáhá v monitorování a ovládání dalších zařízení.

Nezřídka kdy je takováto elektronika zapojována do větších sítí, řízená a vyhodnocovaná centrálním prvkem. Tyto senzorové sítě nabízejí výhody, jako například snadnější údržba, lepší přehled a jednodušší nasazení. Pokud ale nebude síť dobře a efektivně navržena, mohou negativa snadno překonat výhody a síť se stane téměř nepoužitelnou, nebo minimálně drahou na provoz.

Není již potřeba dále zdůrazňovat, že důležitou částí při plánování nasazení takové sítě je právě její návrh. Ten musí mimo jiného zohledňovat potřeby konkrétní implementace a robustnosti, a to s použitím dostatečně výkonných zařízení, která budou výhodná nejen na koupi, ale i na provoz. Takovou roli zastávají různé mikrokontroléry, lišící se výkonem, možnostmi i cenou.

Jedním z nejpopulárnějších a nejdostupnějších mikrokontrolérů současné doby je platforma Arduino. Jedná se o open-source projekt pocházející z Itálie, který představuje stejnojmenný mikrokontrolér založený zejména na mikročipech ATMega od firmy Atmel. Cílem projektu je dostupná programovatelná elektronika, se kterou je jednoduché začít, ale zároveň nabízí i pokročilé možnosti návrhu [1].

Tyto mikrokontroléry umožňují být v roli koncových zařízení, ale i zařízení řídících ostatní. V tomto projektu je již zmíněná platforma nasazena do právě této role. Platforma má velmi nízkou spotřebu, ale má i relativně omezené systémové prostředky. Nabízí se proto otázka, kolik externích zařízení zvládne reálně řídit a kolik systémových prostředků u toho spotřebuje. Na tuto otázku se bude snažit odpovědět tato bakalářská práce.

Práce začíná rešerší podobných řešení, což je možné najít v první kapitole. V dalších kapitolách jsou analyzována úskalí a výhody jednotlivých přístupů k měření a následně je vytvořeno zvolené řešení. To je pak důsledně otestováno a výsledky z tohoto testování (včetně komentářů) je možné vidět v následující kapitole. Nakonec jsou celkové výsledky rozebrány a je podle nich navržen algoritmus, jak postupovat při návrhu senzorové sítě.

Cíle

Prvotním cílem je prozkoumání existujících řešení. Následuje návrh platformy umožňující připojení externích senzorů k platformě Arduino, která bude poté realizována a otestována. Po realizaci je dalším cílem vytvoření počítačové aplikace emulující externí senzory.

Tato aplikace bude využita k dalšímu, pro práci klíčovému, cíli, čímž je měření využití systémových prostředků platformy (procesor, paměť, spotřeba) v závislosti na počtu senzorů a následná analýza výsledků, což je hlavním cílem práce.

Posledním cílem je pak navržení algoritmu pro obsluhu senzorové sítě na základě naměřených hodnot.

Kapitola 1
Existující řešení

Cílem kapitoly je prozkoumání existujících řešení pro měření efektivity komunikace při obsluze většího množství senzorů.

Zadání práce je poměrně unikátní a neexistuje mnoho materiálů zabývající se touto problematikou. Materiály, které se zabývají větším množstvím senzorů, většinou nemají prvky řízeny centrálně z vestavěného zařízení a nesledují tak stejné cíle jako tato práce. Naopak materiály pracující s platformou Arduino (stejně jako tato práce) většinou neřeší efektivitu komunikace s externími senzory, natož s jejich velkým počtem. Nalezené materiály související se zadáním jsou uvedeny níže a zařazeny do kategorií.

1.1 Komunikace mezi senzory

Velká část článků a ostatních materiálů k tématu se zabývá komunikací senzorů. Ačkoli navržení efektivního algoritmu komunikace je jeden ze stanovených cílů této práce, většina materiálů zabývajících se tématem protokolů jsou mimo její rozsah. Standard pro komunikaci je rovněž již stanoven v zadání práce a jedná se o drátovou komunikaci, zatímco tyto materiály většinou berou v potaz protokoly bezdrátové.

Takovými materiály jsou například články [2] a [3], částečně od stejných autorů. První ze zmíněných se zabývá zejména navržením robustního protokolu s citlivostí na ztráty, přičemž provádí i analýzu spotřeby elektrické energie senzorů. Druhý materiál se zaměřuje na navržení co nejefektivnějšího protokolu pro komunikaci se senzory po stránce času, čehož se snaží dosáhnout dynamickou volbou cesty paketu a nižší režií při komunikaci.

Obě tyto práce jsou ale pouze o navrhování protokolů, což není cílem této bakalářské práce i pokud je pominut fakt, že jedná o protokoly čistě bezdrátové a tato práce má v zadání zvolen standard RS485, který je drátový.

1.2 Protokoly pro IoT zařízení

Některé zdroje se zabývají komunikačními protokoly určenými pro IoT zařízení. Existují např. práce srovnávající různé protokoly, drátové i bezdrátové, z různých hledisek.

Jedno z nejobsáhlejších porovnání lze najít ve článku [4]. Autoři zde popisují a porovnávají rozšířené drátové i bezdrátové protokoly. Zaměřují se na výhody a nevýhody nejpoužívanějších z nich, ale i energetickou náročnost nebo možnost zabezpečení. Standard RS485, který je využívaný v této bakalářské práci, autoři sice specifikovali, ale dále podrobněji nerozvádějí.

Dalším porovnáním je článek [5]. Stejně jako první článek obsahuje podrobné porovnání komunikačních protokolů pro IoT, na rozdíl od něj se ale zaměřuje pouze na protokoly bezdrátové.

1.3 Spotřeba elektrické energie

Existuje mnoho materiálů zabývající se spotřebou (většinou bezdrátových) externích senzorů a jak jí omezit. Toto téma je sice související, ale není to předmětem této bakalářské práce. Jako zástupce je vybrána jedna z takových prací, která je obsahem nejpodobnější.

Tímto materiálem je práce [6]. Zde autoři vytvořili nástroj, pomocí kterého lze simulovat elektrickou spotřebu senzorů zapojených do rozsáhlé sítě. I přes na první pohled podobně znějící obsah je zde několik zásadních bodů, které tuto práci odlišují. Za prvé se zmiňovaný výzkum zaměřuje na spotřebu samotných senzorů, zatímco tato práce se zabývá měřením (mimo jiných metrik) spotřeby zařízení, ke kterému se senzory připojují. Dále autoři práce projekt navrhovali zejména pro senzory typu Mica2. V této bakalářské práci jsou senzory zobecněny, mohou být implementovány na libovolných zařízeních a důraz je opět kladen zejména na zařízení Arduino, ke kterému se senzory připojují. Nakonec je také dobré zmínit, že práce je skoro 20 let stará a vlivem rychlého vývoje v oblasti vestavěných zařízení už nemusí být relevantní pro dnešní potřeby.

Kapitola 2
Analýza

V kapitole je proveden rozbor stavebních prvků platformy a analýza možných dílčích řešení. Uveden je používaný hardware a odhad jeho silných a slabých míst.

Pro přenos je zadaný standard RS485, který umožňuje komunikaci více zařízení v modu half-duplex, a to za maximální teoretické rychlosti 10 Mbit/s a maximální vzdálenosti až 1,2 km. Limit počtu zařízení a délky je možné zvyšovat použitím opakovačů, každý opakovač přidá možnost dalších 32 zařízení a prodlouží maximální délku o 1,2 km. Zařízení jsou připojována do sběrnicové topologie a může jich být na jednu linku připojeno až 32. Toto vše standard umožňuje za pomocí jednoho páru kroucené dvojlinky (pouze dvou vodičů) [7].

Opakovače mohou v reálných podmínkách zpomalit přenos nebo zvýšit odezvu, ale pro zjednodušení se práce nezabývá úskalím opakovačů a předpokládá, že limit počtu zařízení je možné zvyšovat do nekonečna.

Navrhované řešení se skládá ze dvou typů zařízení, na principu server a klient.

Centrálním prvkem komunikace je platforma Arduino. Jejím úkolem je sběr dat ze senzorů, provedení výpočtu, jaká akce má nastat v závislosti na přijatých datech a následné případné ovládání aktivních prvků.

Klientským zařízením je myšleno obecné zařízení, které dokáže komunikovat po lince RS485 a odpovídat (podle specifikace) na dotazy centrálního prvku.

Klientskými zařízeními jsou senzory a aktivní prvky. Senzor je takové zařízení, které na vyžádání oznamuje stav jedné ze svých hodnot, které udržuje a vnitřně může aktualizovat podle vnějších vlivů. Aktivní prvek je takové zařízení, které umožňuje na příkaz vykonat jednu ze svých předem předdefinovaných akcí.

2.1 Komunikační protokol

Protože standard RS485 přímo nespecifikuje žádný protokol na vyšší úrovni (definuje pouze přenos jednotlivých bajtů), je nutné ho pro práci vytvořit. Měl by být jednoduchý s co nejnižší režií, aby nezpomaloval komunikaci, ale zároveň musí umožňovat komunikaci co největšího počtu zařízení.

Každé zařízení má určitý počet napřímo připojených součástek, které chceme vzdáleně spravovat. Předpokládá se, že takovýchto součástek má každé zařízení průměrně až 10. Koncová zařízení jsou navržena maximálně jednoduše a větší počet vstupů by zvyšoval jejich nutnou komplexitu.

Adresa zařízení by měla být pevné délky. Tuto adresu přečtou všechna zařízení, čímž zjistí, komu je zpráva určena. Jelikož komunikace přes RS485 probíhá po bajtech, nabízí se několik logických možností adresace podle počtu bajtů.

Jednobajtová adresa by byla nejjednodušší na implementaci a umožňovala by 256 zařízení na síti. Tento počet zařízení je ale moc malý, v práci se předpokládá, že bude možné obsluhovat alespoň tisíc zařízení.

Dvoubajtová adresace tuto situaci řeší – nabízí 65 536 různých adres. Malý rozsah zde není problém, ale je však otázka, jestli není takové množství až zbytečné – centrální zařízení má omezenou paměť a tolik externích zařízení není schopné spravovat.

Pro zefektivnění je možné sloučit adresu zařízení a adresu přilehlého senzoru do dvou bajtů. Tímto způsobem bude splněn rozsah adresace a zároveň optimalizována velikost. Detaily tohoto sloučení budou probírány později v části návrhu.

2.2 Hardware

Jak je možné vidět již ze zadání práce, centrálním prvkem komunikace je platforma Arduino.

Existuje mnoho desek Arduino s různou výbavou pro různá použití. Z důvodu možnosti rozdílných výsledků kvůli různé výbavě desek budou v práci prozkoumány dvě Arduino varianty. Zvoleny byly desky `Arduino Uno R3` a `Arduino Mega 2560 Rev3`. Tyto desky byly zvoleny zejména z toho důvodu, že se jedná o nejpopulárnější desky Arduino v době psaní [8] a také se mezi sebou do značné míry liší výbavou.

Tabulka 2.1 Porovnání Arduino Uno R3 [9] a Arduino Mega 2560 Rev3 [10]

	CPU Freq.	RAM	Flash	EEPROM	Dig. I/O	Anal. I.	PWM	UART
`Uno`	16 MHz	2 KiB	32 KiB	1 KiB	13	6	6	1
`Mega`	16 MHz	8 KiB	256 KiB	4 KiB	54	16	15	4

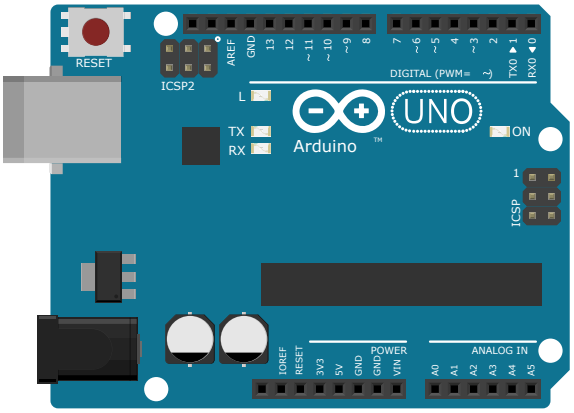
2.2.1 Arduino Uno

Arduino Uno, které je nejpopulárnější deskou [8] a stalo se de facto ikonou celé platformy, je v porovnání s variantou Mega značně slabší po hardwarové stránce.

Obě desky sice mají stejnou frekvenci procesoru, ale výrazné omezení z hlediska počtu I/O pinů, 4x menší paměť RAM a 8x menší programová paměť, to vše kromě dalších parametrů jsou jednoznačně viditelná místa, kde deska Uno v porovnání se silnější Mega značně postrádá (Tabulka 2.1).

Úskalím této desky, které je podstatné pro tento projekt, je pouze jedna sériová linka UART. Kvůli nutnosti monitorování zdrojů v této práci je tato linka použita pro hlášení stavu zařízení, jako např. zatížení procesoru. Z tohoto důvodu je linka již obsazena pro komunikaci s PC po USB a nemůže tak být využita pro senzorovou síť. Sériovou komunikaci tedy musí na této desce nahrazovat softwarově Arduino knihovna `SoftwareSerial`, která je součástí základních knihoven od projektu Arduino [11].

Hlavní výhodou desky je však cena, která je v porovnání s výkonnější variantou téměř dvakrát nižší [8]. Nabízí se tedy otázka, zda se pro určitý počet senzorů vyplatí použití desky Uno, nebo Mega. I odpovědí na tuto otázku by měla být právě tato práce.

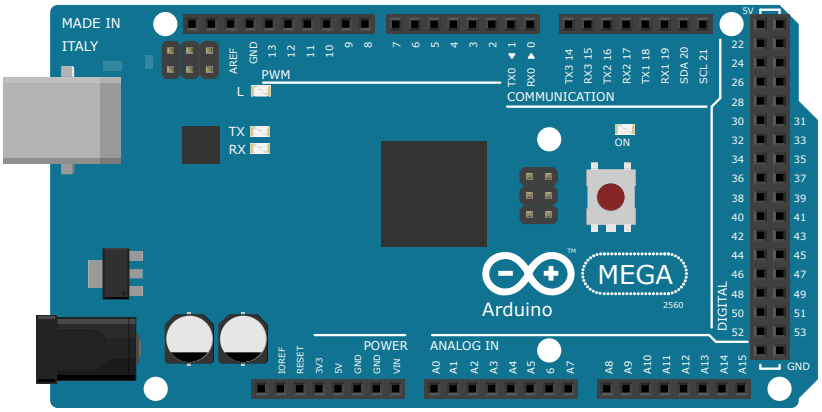


Obrázek 2.1 Arduino Uno v měřítku 1:1 [12]

2.2.2 Arduino Mega 2560

Deska Arduino Mega 2560 je mezi Arduino deskami zařazená do kategorie Enhanced Features, a to z dobrého důvodu. V porovnání s Uno, které je označeno ještě jako Entry Level, nabízí zvýšení téměř všech parametrů (Tabulka 2.1). Jedním z mála parametrů, které se nedočkaly oproti verzi Uno zlepšení, je frekvence procesoru. Tato hodnota je stále poměrně nízká a jedná se o možné úskalí pro tento projekt.

Co se týká ostatních parametrů, větší počet I/O pinů, které deska Mega 2560 nabízí, je sice příjemnou výhodou, ale pro návrh, který je v této práci zvolen, není moc relevantní. To se už ale nedá říci o větší operační paměti, což je pro tuto práci naopak klíčovou výhodou právě této desky.

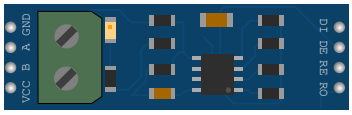


Obrázek 2.2 Arduino Mega 2560 v měřítku 1:1 [12]

2.2.3 Transceiver pro Arduino

Modul RS485 od společnosti Maxim Integrated je transceiver pro komunikaci po lince RS485.

Toto zařízení obsahuje celkem 6 pinů určených pro připojení k desce (v tomto případě Arduino), dva sloužící k napájení a další 4 piny obsluhující komunikaci – jelikož je přenos po RS485 half-duplex, kromě pinů na samotnou datovou komunikaci jsou zde zbylé dva piny pro přepínání mezi odesíláním a přijímáním. Program nahraný na desce Arduino si tak vždy musí vybrat, zda chce přijímat nebo odesílat a tento stav dát modulu vědět přes tyto dva piny.



Obrázek 2.3 TTL transceiver MAX485 v měřítku 1:1

2.2.4 USB Interface

Jelikož je součástí práce i emulace libovolného množství senzorů přes PC, je nutné komunikační linku platformy spojit s PC. K tomuto účelu slouží jednoduchý převodník do USB, který na straně PC emuluje COM port a lze tedy jeho prostřednictvím jednoduše komunikovat pomocí zavedeného API operačního systému.

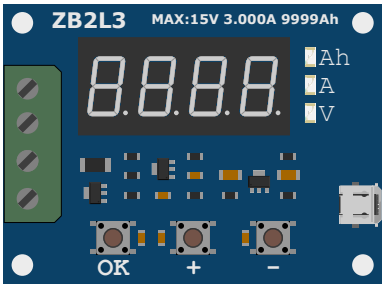
Vybraný převodník je typu `ICSH012A` a pohání ho dobře zavedený čip `CH340C`, který je často také používaný i v deskách Arduino. Tento převodník umožňuje podle výrobce dosáhnout vysoké teoretické maximální rychlosti přenosu 115 200 B/s a dosahuje vzdálenosti dané standardem RS485 [13], neměl by tedy v tomto řešení být limitujícím faktorem.

Obrázek 2.4 Převodník USB na RS485 typu ICSH012A v měřítku 1:1

2.2.5 Zařízení měřící spotřebu energie

Pro měření spotřeby elektrické energie platformy je využívaný modul `ZB2L3`, někdy také označovaný jako `HW-586`. Jedná se o zařízení měřící proteklý proud, využívané především pro měření kapacity baterií.

Pro tuto práci byl zvolen z toho důvodu, že ze svého principu měří proteklý proud v delším časovém úseku, neměl by být tedy příliš ovlivněn možnými rychlými výkyvy spotřeby desky Arduino danými změnou chování programu, který je na desku nahraný.



Obrázek 2.5 Modul ZB2L3 v měřítku 1:1

2.3 Software

Pro práci je nutné vytvořit dva programy. Prvním z nich je program na samotné Arduino, který bude vyřizovat aktualizaci hodnot senzorů a veškerý provoz na senzorové síti. Druhý program slouží pro emulaci senzorů v senzorové síti a poběží na připojeném PC.

2.3.1 Arduino

Program na Arduinu musí být jednoduchý, aby bylo zatížení procesoru a spotřeba paměti co nejmenší.

Interval, po kterém dochází k aktualizaci hodnot senzorů, je možné rozdělit na 3 možnosti:

1. Každý senzor má přidělenu dobu, po které dochází k jeho aktualizaci. Kvůli úspoře místa v paměti a nepotřebě vysoké granularity je možné použít pouze jeden bajt, který reprezentuje počet milisekund vydělených koeficientem. Pro tento koeficient byla zvolena hodnota 256. Doba obnovení je tedy v rozsahu od 256 do 65 535 ms po násobcích 256.
2. Je stanovena doba, po které jsou aktualizovány všechny senzory najednou. Tato hodnota je minimálně 1 ms.
3. Všechny senzory jsou neustále aktualizovány po sobě bez žádné prodlevy.

Z těchto 3 typů je možné vyvodit následující poznatky:

- Pouze pro první typ je potřeba ukládat interval a příští čas aktualizace u každého senzoru. Pro všechny kromě prvního typu je tímto možné zmenšit strukturu o přibližně 4 B, což je při vyšším počtu senzorů velmi významná část.
- Protože u třetího typu jsou senzory aktualizovány bez prodlevy, je zde zbytečné měřit využití CPU, bude vždy 100 %.

2.3.2 PC

Součástí práce je i aplikace na PC, konkrétně na OS Windows, která emuluje senzory na lince RS485. Tento program je napsaný v jazyce `C++` a jeho úkolem je odpovídat na dotazy platformy Arduino.

Požadavkem na program je vysoká rychlost – musí odpovídat s co nejmenším zpožděním, aby bylo následné měření spotřeby zdrojů pokud možno co nejvíce konzistentní a nebylo ovlivněno výkonem PC. Kvůli tomu by návrh programu neměl být příliš komplexní a měla by být udržena jednoduchost.

Program by po obdržení dotazu s danou adresou měl pouze nahlédnout do paměti na požadovanou hodnotu (konstantní složitost) a tuto hodnotu vrátit. Paměť vyžadovaná pro udržování pole není na současných PC problém, v případě 16bitové adresy zařízení bude celé pole zabírat v paměti stále méně, než 1 MB.

Hodnoty by mělo být možné nastavovat z uživatelského prostřední a provádět tak emulaci změn hodnot fyzických senzorů.

Pro zpřehlednění by bylo vhodné při každém požadavku od Arduina zobrazit informační zprávu v příkazové řádce, tato funkce má ale zásadní nevýhodu, a tou je zpomalení celého programu. I při použití efektivního způsobu zápisu na standardní výstup v jazyce C++ je výstup limitujícím faktorem a program by tak byl zásadně zpomalován, čímž by byl porušen nejdůležitější požadavek na program – co nejnižší odezva.

Možnost výpisu požadavků je tedy vhodné nechat spíše na vyžádání, pro řešení potíží je to stále užitečná funkce.

Aby bylo možné detekovat chyby u přenosu, mohl by program umožňovat kontrolu obdržených dotazů vůči definovaným pravidlům. Tato kontrola, jako například porovnání dvou hodnot, je velmi rychlá a neměla by program zpomalovat.

2.4 Způsoby měření

Tato část práce se zabývá systémovými prostředky, které budou v práci měřeny, proč je důležité je měřit a jakým způsobem toho bude docíleno.

2.4.1 Využití paměti

Paměť je v případě vestavěných zařízení jako Arduino typicky velmi omezeným zdrojem, je tedy důležité analyzovat, kolik jí navrhované řešení bude využívat.

Jeden senzor sice v paměti zabírá pouze několik bajtů, ale v kombinaci s velikostí paměti zvolených desek Arduino se paměť může rychle stát hlavním omezením počtu možných senzorů, které deska zvládne obsluhovat.

V případě této práce bude program na Arduino desce velmi jednoduchý a spotřebu paměti lze vypočítat již během kompilace. Je tedy snadné určit, jestli bude paměť zařízení stačit a kolik jí zůstane pro zbytek programu.

Tento výpočet zčásti provádí vývojové prostřední Arduina automaticky. Po kompilaci zobrazí paměť využitou globálními proměnnými s informací, jestli je paměť zařízení plně dostačující, přesahuje dostupnou paměť, nebo varovnou hlášku se zprávou, že paměť sice pojme globální proměnné, ale nemusí už dostačovat na lokální proměnné při běhu programu. V práci jsou dále uvažovány pouze případy, kdy kompilátor vyhodnotí využití paměti bezpečné pro celý běh programu.

2.4.2 Využití procesoru

Vytížení procesoru je dalším klíčovým zdrojem, který bude monitorován. Jestli procesor stíhá externí zařízení obsluhovat je sice nejdůležitější, ale užitečné je i zjištění, jak dlouhou dobu obsluha zabrala.

Z tohoto důvodu bude v práci měřeno obojí. V průběhu programu by měla být sčítána doba, při které procesor aktivně vyřizuje externí zařízení, a tato hodnota by měla být v časových úsecích vracena přes sběrnici USB do PC, které Arduino monitoruje po sériové lince.

S touto hodnotou by mělo být vraceno i maximální zpoždění, ke kterému za danou dobu došlo. Zpožděním je zde myšlen rozdíl plánované aktualizace senzoru a jejího reálného spuštění v milisekundách.

Pokud bude trend zpoždění růst, zařízení nestíhá aktualizace vyřizovat a není možné ho v daném nastavení využívat pro řízení sítě.

2.4.3 Spotřeba elektrické energie

Posledním ze zdrojů, který bude monitorován, je spotřeba elektrické energie.

Intuitivním způsobem měření spotřeby je použití běžného ampérmetru. Pokud by ale byla spotřeba měřena obyčejným ampérmetrem měřícím pouze aktuální hodnotu, naměřené hodnoty by se mohly velmi rychle měnit, a to i rychleji, než by je daný ampérmetr mohl dokázat zaznamenat.

Alternativou je modul `ZB2L3` představený v části 2.2.5. Ten zmíněné problémy do určité míry řeší, a to zjednodušeně řečeno velmi rychlým snímáním aktuálního proudu a sčítáním hodnot. Výsledkem tohoto měření je pak hodnota v mAh. Modul měří na přesnost v jednotkách mA, s napětím desky Arduino 5 V tedy přesnost vychází na 5 mW.

V této práci je cílem zjistit průměrnou spotřebu platformy, kterou lze z výsledků měření tohoto modulu v kombinaci s délkou časového úseku jednoduše vypočítat (např. při měření o délce jedné hodiny je výsledná průměrná spotřeba přímo hodnotou naměřenou modulem v mA).

Kapitola 3

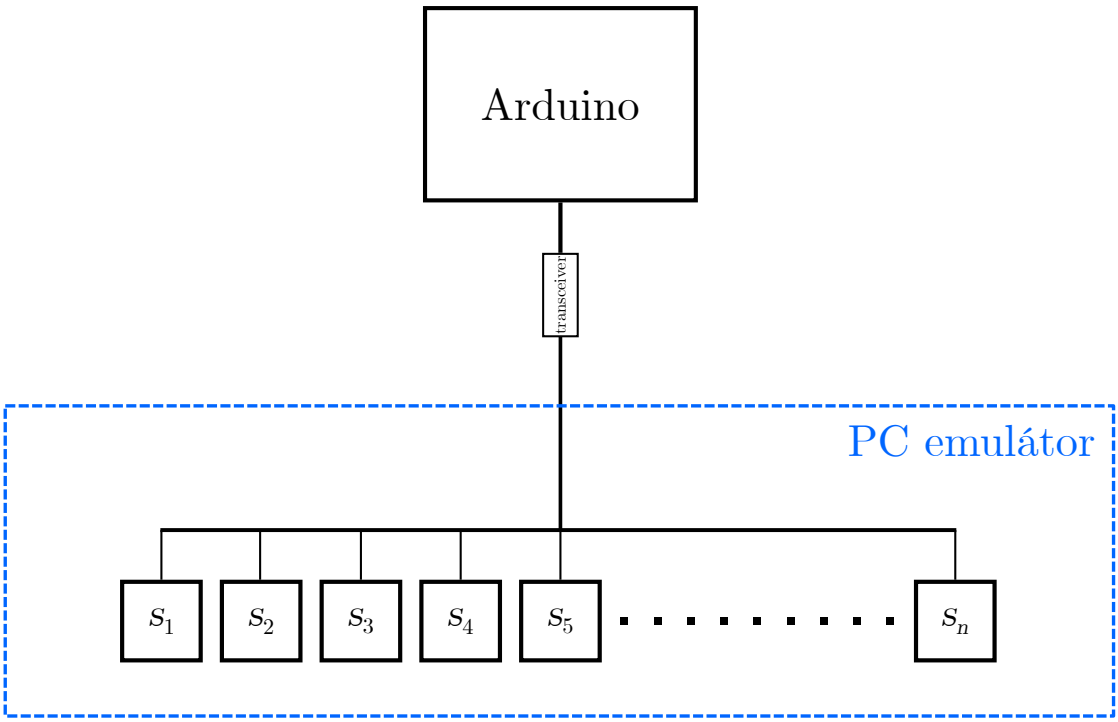
Návrh platformy a implementace

V kapitole je představena platforma, která bude následně využívána po zbytek práce. Uvedeno je zapojení hardwaru i softwarové řešení komunikace.

Centrálním prvkem je Arduino. To je připojené skrze měřič energie ZB2L3 k PC přes USB. Toto rozhraní kromě poskytování napájení slouží i na sbírání dat o provozu desky.

K Arduinu je připojený RS485 transceiver. Zapojení se liší podle typu desky. V případě Arduino Uno je modul zapojen ke standardním digitálním I/O pinům a komunikace je řešena softwarově přes zmíněnou knihovnu SoftwareSerial. U desky Arduino Mega je modul připojen k UART pinům určeným pro nativní sériovou komunikaci.

Po napojení modulu k desce už jen stačí zapojit dvouvodičový kabel, ke kterému jsou připojeny všechny senzory (nebo jejich emulátor) a komunikační síť je připravena na přenos dat (Obrázek 3.1).



Obrázek 3.1 Blokové schéma platformy

3.1 Komunikační protokol

Zvoleným řešením pro komunikační protokol je následující implementace:

Každé zařízení obsahuje flag, zda jde o senzor nebo aktivní prvek a svoji adresu v rozsahu 0–2047. Tyto hodnoty jsou do zařízení nastaveny při jeho programování a nelze je měnit za běhu.

Při komunikaci se senzorem je kromě adresy zařízení součástí i číslo senzoru v rozsahu 0–15. Při každém vyžádání hodnoty senzoru je zaslán i 16bitový parametr datového typu integer a jeho interpretace je dána na každém senzoru (může např. obsahovat dohodnutou jednotku, ve které bude hodnota vrácena, časový údaj atd.). Po přijetí a vyhodnocení senzor odpoví oznamovanou hodnotou, kterou je rovněž 16bitový integer a její význam se může lišit u každého senzoru.

Při odesílání příkazu na aktivní prvek je opět kromě adresy zařízení součástí i číslo předdefinované akce v rozsahu 0–15. Po přijetí příkazu se 16bitovým parametrem typu integer aktivní prvek odpoví 16bitovým integerem značícím úspěšnost přijetí požadavku. Pokud příkaz vyhodnotí jako platný, odpoví hodnotou 1, v opačném případě hodnotou 0. Aktivní prvek nečeká na dokončení akce, odpověď posílá okamžitě po dekódování přijatého požadavku.

Komunikaci ze strany centrálního prvku je možné shrnout do pevné délky 32 bitů:

- 1. bit značí, jestli jde o senzor nebo aktivní prvek. Hodnota 0 je senzor a 1 aktivní prvek
- 02.–12. bit definuje adresu zařízení
- 13.–16. bit definuje číslo předvolené hodnoty/akce, na kterou je dotazováno
- 17.–32. bit obsahuje parametr

Odpovědí zařízení je pak vždy 16bitový little endian integer, jehož interpretace se pro různá zařízení liší.

Z uvedeného složení adresy je možné vidět, že každá hodnota na senzoru a akce na aktivním prvku mají dohromady jasně definovanou 16bitovou adresu. Toho je využito v programu na centrálním Arduino prvku – ten totiž neřeší, na jakém ze zařízení je nějaká hodnota umístěna, nebo k jakému zařízení se váže akce, pouze pošle 16bitovou adresu (s parametrem) a přijme odpověď.

Senzory a aktivní prvky pak při každém přijetí požadavku zkontrolují, zda prvních 12 bitů souhlasí s jejich pevně danou adresou a na základě toho buď odpoví (a vykonají akci), nebo požadavek ignorují.

3.2 Software Arduino

Obslužná aplikace nahraná na centrální zařízení Arduino je vyvinuta v jazyce C++ a použitím preprocesorových maker je přizpůsobena oběma deskám, které jsou v práci využity. Úplný zdrojový kód je možné nalézt na přiloženém médiu, zde jsou vysvětlovány pouze základní principy programu.

Pro řízení aktualizací byla použita 1. varianta obnovení prozkoumána v části 2.3.1, a to kvůli univerzálnosti a největší použitelností v praxi.

Základním prvkem programu je struktura obsahující informace o jednom senzoru. Její součástí je kromě samotné adresy senzoru a proměnných určených pro synchronizaci i poslední přijatá hodnota. Velikost této struktury je 9 B.

Výpis kódu 3.1 Struktura senzoru s nastavitelným intervalem

```
struct Sensor
{
  int16_t       sensorAddress;
  int16_t       lastValue;
  uint8_t       updateInterval;
  uint32_t      nextUpdate;
};
```

Z velikosti struktury je možné rovnou udělat hrubé horní odhady maximálního počtu senzorů na desku. Například je rovnou jisté, že s reprezentací senzoru strukturou ve Výpisu kódu 3.1, která má 9 B, se do paměti Arduino Uno (zjištěné z tabulky 2.1) nikdy nemůže vejít více, než 227 senzorů ($\lfloor \frac{2048}{9} \rfloor = 227$).

Instance struktur představených ve Výpisu kódu 3.1 naplněné daty jsou následně umístěny v poli. Toto pole je v nekonečně smyčce procházeno, a pro každý prvek je zjišťováno, jestli má dojít k aktualizaci hodnoty senzoru. Pokud ano, vyšle se přes RS485 žádost s adresou zařízení a odpovídajícím parametrem a vyčkává se na odpověď. Nová hodnota senzoru se uloží do struktury a je proveden výpočet. Hodnota výsledku je pak odeslána na příslušný aktivní prvek a opět se čeká na odpověď zařízení.

Výpis kódu 3.2 Obsluha senzorů na Arduinu

```
for (int16_t i = 0; i < SENSORS; i++)
{
  uint32_t millisStart = millis();

  //Checks if sensor needs to be updated
  if (sensors[i].nextUpdate <= millisStart)
  {
    if (sensors[i].nextUpdate < millisStart)
      maxDelay = max(maxDelay, millisStart - sensors[i].nextUpdate);

    uint32_t cpuStart = micros();

    //Sends update request and waits for response
    int16_t newVal = sendRequest(sensors[i].sensorId);

    //Calculates response and sends it to an active module
    int16_t result = newVal * 42;
    sensors[i].lastValue = newVal;
    sendRequest(sensors[i].sensorId | 0b1000000000000000, result);

    //Sets next update time
    uint16_t updateMs = (((uint16_t)sensors[i].updateInterval) << 8);
    sensors[i].nextUpdate += updateMs;

    cpuUsage += micros() - cpuStart;
  }
}
```

Jak je možné vidět ve Výpisu kódu 3.2, kromě obsluhy senzorů program obsahuje i části určené pro monitoring zátěže. Tyto hodnoty jsou v nastavených intervalech odesílány k připojenému PC.

Pro zjednodušení při testování je výsledek výpočtu vždy odesílán na aktivní prvek se shodnou adresou, jako byla adresa senzoru (pouze první bit celkové adresy se liší).

3.3 Software emulace

Program na PC je vytvořený ve formě konzolové aplikace. Běží zde 2 vlákna, jedno zajišťuje UI a druhé se stará o samotnou komunikaci.

V komunikačním vlákně běží nekonečná smyčka, ve které se čeká na dotaz na senzor nebo příkaz daný aktivnímu prvku. Jakmile je přijat, je odpovězeno hodnotou uloženou u daného prvku.

Po odeslání odpovědi je zkontrolováno, jestli byla přijata validní hodnota na základně pevně daného očekávaného výsledku. Pokud není přijatá hodnota v souladu s očekáváním, je zobrazena chybová hláška.

Výpis kódu 3.3 Emulace senzorů na PC

```
while (true)
{
  //Reads address and argument
  uint16_t address = serial->read();
  if (reportAll)
    std::cout << "Address: " << std::hex << address << std::dec << ' ';
  int16_t argument = serial->read();
  if (reportAll)
    std::cout << "Argument: " << argument << std::endl;

  //Responds with corresponding value
  serial->writeAndFlush(devices[address]);

  //Checks if value is correct (arbitrary calculation)
  int16_t expected =
  address & 0b1000000000000000
  ? devices[address & 0b0111111111111111] * 42
  : 0;
  if (argument != expected){
    std::cout << "Argument not expected!: " << argument
    << ". Was expecting " << expected << std::endl;
  }
}
```

Hodnoty každého zařízení, které bude aplikace vracet, je možné měnit přes uživatelské rozhraní. Toto rozhraní přijímá následující příkazy:

- `get [address]` – zobrazí hodnotu zařízení s danou adresou
- `set [address] [value]` – nastaví novou hodnotu zařízení na dané adrese
- `debug [1/0]` – zapne resp. vypne výpis komunikace na obrazovku
- `off` – vypne aplikaci

Kapitola 4
Měření

V kapitole jsou prováděny měření různých metrik za pomoci platformy představené v předchozích kapitolách. Výsledky jsou okomentovány.

V následujících částech jsou výsledky měření provedených metodikou popsanou níže sdružovány do tabulek a grafů dle různých parametrů. Představena jsou pouze klíčová měření pro cíl práce, všechna měření je možné najít na přiloženém médiu.

Měřítka os grafů jsou pevně fixována pro lepší přehlednost přímého porovnání grafů mezi více měřeními. Osa grafů s počtem senzorů je přizpůsobena maximální hodnotě, kterou deska podporuje.

4.1 Metodika

Měření je prováděno následujícím postupem:

1. Na zvolenou desku Arduino je nahrán program s nastavenými parametry. Těmito parametry jsou počet senzorů, interval aktualizace jejich hodnot a jak jsou aktualizace rozprostřeny (náhodně/dávkově). Náhodné rozprostření znamená, že se sice všechny senzory aktualizují se stejnou frekvencí, ale ne ve stejnou chvíli – čas jejich aktualizace je náhodně rozprostřen přes daný interval. Dávkové rozprostření je toho opakem, všechny senzory se aktualizují vždy ve stejnou chvíli (v praxi pokud možno co nejrychleji za sebou).
2. Arduino je zapojeno k monitorovacímu PC skrze měřič spotřeby elektrické energie.
3. Platforma je napojena na RS485, ke kterému je připojené zařízení s běžícím programem na emulaci zařízení.
4. Je spuštěno měření elektrické energie a hodnoty jsou akumulovány po dobu 60 minut.
5. Po uplynutí je sejmuta hodnota spotřeby, zatížení procesoru a s tou související maximální odezva a tyto hodnoty jsou uloženy.

   Arduino nebylo v průběhu programu nikdy přepínáno do šetřících stavů.

4.1.1 Arduino Uno

Prvotní deskou, která je testována, je Arduino Uno.

Jak již bylo zmíněno, program nahraný na desku je upraven, aby využíval softwarovou emulaci sériové linky.

Pro baudrate sériové komunikace je zvolena hodnota 57 600 Bd. Jedná se o nejvyšší možnou hodnotu, kterou Arduino v tomto zapojení dokázalo zpracovávat. Při této rychlosti by přenesení 1 bajtu mělo podle výpočtu trvat 173,6 µs.

Maximální počet senzorů, který je v měření uvažován, je 135. Tento počet je nejvyšší možný bez varování kompilátoru o nedostatečné paměti, jak bylo uvedeno v části 2.4.1.

Kromě nulového počtu senzorů, kde kompilátor optimalizací snížil využití na ještě menší hodnotu, lze spotřeba paměti jednoduše vypočítat vyřešením výrazů $321 + n \cdot 9$, kde n je počet senzorů. Velikost struktury na jeden senzor je totiž 9 B a zbytek programu, knihovna SoftwareSerial a vestavěné Arduino funkce zabírají 321 B.

4.1.2 Arduino Mega

Testována je i deska Arduino Mega 2560.

Protože tento model na rozdíl od Uno obsahuje další hardwarovou sériovou linku, je tato linka pro komunikace využita.

Baudrate je nastaven na nejvyšší možnou hodnotu, kterou Arduino umožňuje – 115 200 Bd. Hardwarová linka na rozdíl od softwarové emulace nemá s touto přenosovou rychlostí žádné problémy a během testování nedošlo k jediné chybě při přenosu. Při této rychlosti by přenesení 1 bajtu mělo podle výpočtu trvat 86,8 µs.

Maximální počet senzorů, který je v měření uvažován, je 642. Tento počet je nejvyšší možný bez varování kompilátoru o nedostatečné paměti, jak bylo uvedeno v části 2.4.1. Jedná se o téměř pětinásobek počtu senzorů, které zvládne slabší deska.

Kromě nulového počtu senzorů, kde kompilátor optimalizací snížil využití na ještě menší hodnotu, lze spotřeba paměti jednoduše vypočítat vyřešením výrazu $361 + n \cdot 9$, kde n je počet senzorů. Velikost struktury na jeden senzor je 9 B, zbytek programu a vestavěné Arduino funkce zabírají 361 B.

4.2 Arduino Uno, rychlá aktualizace

Následující měření fixuje interval a rozprostření aktualizace a ukazuje vliv počtu senzorů na spotřebu zdrojů. Zvoleným intervalem obnovení je 256 ms a aktualizace jsou náhodně rozprostřeny.

Rychlost obnovení je nejnižší povolenou hodnotou podle vlastní specifikace v části 2.3.1. Výsledkem měření by měl být poznatek, kolik zařízení dokáže Arduino Uno obsluhovat, pokud požadujeme co nejrychlejší obnovování senzorů rovnoměrně v čase.

4.2.1 Shrnutí

Výsledky měření jsou vidět na druhé straně v Tabulce 4.1. Arduino zvládalo senzory obsluhovat až do počtu 27. Od 28 senzorů přestalo Arduino stíhat aktualizace vyřizovat (v tabulce červeně) a platforma tak není v tomto stavu již použitelná.

4.2.2 CPU

Využití procesorového času rostlo lineárně s počtem senzorů, jak je možné vidět z grafu na Obrázku 4.1. Z dat v Tabulce 4.1 lze odvodit, že jeden senzor při této rychlosti obnovování zabíral průměrně 3,6 % výkonu CPU.

Zatížení tak rostlo lineárně až do počtu 27 senzorů, kdy byl procesor zatížený na 98 %. Při přidání jednoho dalšího senzoru a dosažení počtu 28 už zatížení procesoru dosáhlo maxima a systém přestával stíhat aktualizace vyřizovat, jak je uvedeno výše.

4.2.3 RAM

V případě tohoto testu velikost paměti stačila a nebyl s ní žádný problém, ostatní prostředky byly mnohem vytíženější.

4.2.4 Spotřeba elektrické energie

Spotřeba Arduina rostla s počtem senzorů (graf na Obrázku 4.2) a lze zde pozorovat měřitelné rozdíly podle počtu senzorů.

Rozdíl mezi nulovým počtem senzorů a maximem, které deska zvládala, je 60 mW. Při rozpočítání mezi počet senzorů vychází na provoz jednoho senzoru aktualizovaného tímto způsobem přibližně 2,2 mW (bez spotřeby samotného senzoru a spotřeby nečinného Arduina).

4.2.5 Odezva

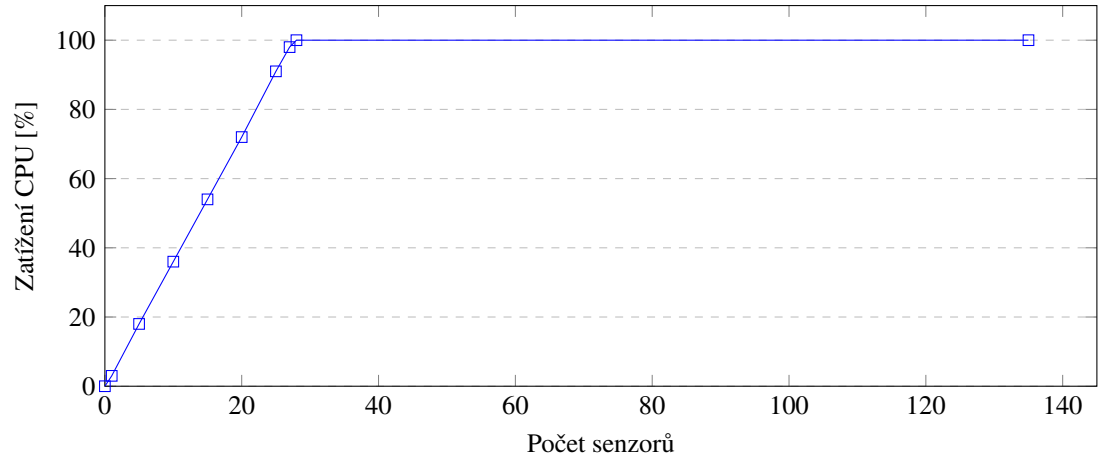
Maximální odezva (možné vidět z grafu na Obrázku 4.3) začínala na 0 ms pro jeden senzor a držela si jednocifernou hodnotu pro počet senzorů pod 10, ale s vyšším počtem už začala rapidně stoupat až na 82 ms při 27 senzorech, což byla poslední hodnota, při které Arduino stále stíhalo senzory obsluhovat.

Potom už Arduino nedokázalo nikdy vyrovnat zpoždění, které tak jenom rostlo a skončilo na hodnotě necelých 62 sekund za hodinu měření. Čím delší je doba měření, tím vyšší je tato hodnota, roste do nekonečna.

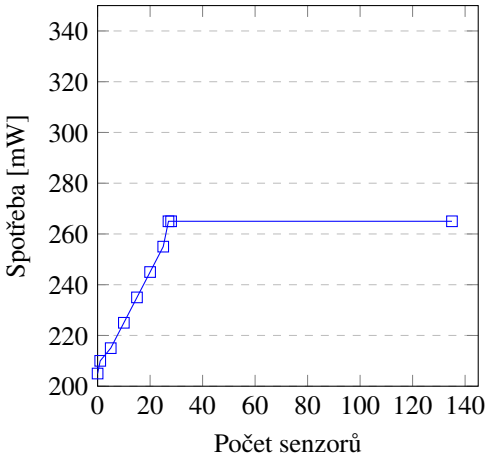
Deska	Interval aktualizace	Rozprostření aktualizací
Uno	256 ms	Náhodné

Počet senzorů	Využití RAM [KiB]	Zatížení CPU [%]	Spotřeba energie [mW]	Odezva [ms]
0	317	0	205	0
1	330	3	210	0
5	366	18	215	1
10	411	36	225	7
15	456	54	235	22
20	501	72	245	39
25	546	91	255	50
27	564	98	265	87
28	573	100	265	∞

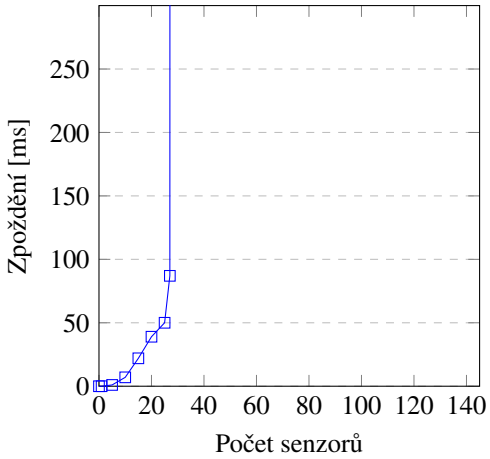
Tabulka 4.1 Přehled naměřených hodnot



Obrázek 4.1 Vliv počtu senzorů na zatížení CPU



Obrázek 4.2 Vliv p. senzorů na spotřebu



Obrázek 4.3 Vliv počtu senzorů na odezvu

4.3 Arduino Uno, rychlá aktualizace v dávce

Následující měření fixuje interval a rozprostření aktualizace a ukazuje vliv počtu senzorů na spotřebu zdrojů. Zvoleným intervalem obnovení je 256 ms a aktualizace jsou prováděny najednou v dávkách.

Rychlost obnovení je nejnižší povolenou hodnotou podle vlastní specifikace v části 2.3.1. Test je podobný jako předchozí, liší se jen rozprostřením času aktualizací. Výsledky mohou být relevantní například pro případy, kdy je z nějakého důvodu nutné mít všechny (aktuální) hodnoty senzorů dostupné ve stejnou chvíli.

4.3.1 Shrnutí

Výsledky měření jsou vidět na druhé straně v Tabulce 4.2. Stejně jako v přechozím testu Arduino zvládalo senzory obsluhovat až do počtu 27, od 28 senzorů přestalo Arduino stíhat aktualizace vyřizovat (v tabulce červeně) a platforma tak není v tomto stavu již použitelná.

U tohoto měření se vždy vnitřně střídají fáze vysokého zatížení a nečinnosti. Tento fakt se projevuje na výsledcích různě, jak je možné vidět z detailního popisu níže.

4.3.2 CPU

Stejně jako v testu minulém, využití procesorového času rostlo lineárně s počtem senzorů, jak je možné vidět z grafu na Obrázku 4.4. Z dat v Tabulce 4.2 lze odvodit, že jeden senzor při této rychlosti obnovování zabíral průměrně 3,6 % výkonu CPU.

Vytížení procesoru viditelné z tabulky a grafu je měřeno dlouhodobě, střídání fází tak není v tomto případě ve výsledcích viditelné.

4.3.3 RAM

V případě tohoto testu velikost paměti opět stačila a nebyl s ní žádný problém, ostatní prostředky byly mnohem vytíženější.

4.3.4 Spotřeba elektrické energie

Spotřeba Arduina rostla s počtem senzorů (graf na Obrázku 4.5) a lze zde pozorovat měřitelné rozdíly podle počtu senzorů.

Rozdíl mezi nulovým počtem senzorů a maximem, které deska zvládala, je 50 mW. Při rozpočítání mezi počet senzorů vychází na provoz jednoho senzoru aktualizovaného tímto způsobem přibližně 1,9 mW (bez spotřeby samotného senzoru a spotřeby nečinného Arduina).

4.3.5 Odezva

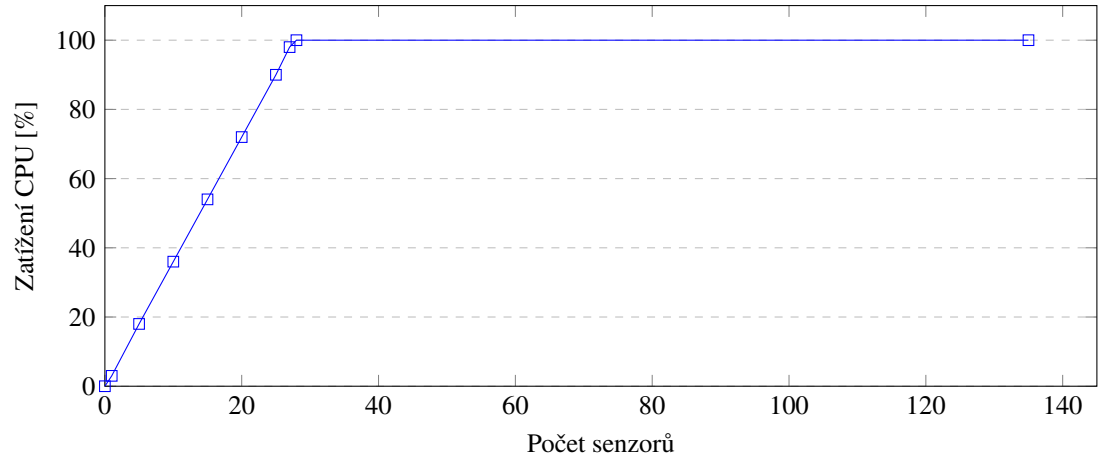
Maximální odezva (možné vidět z grafu na Obrázku 4.6) začínala na 0 ms pro jeden senzor a následně stoupala lineárně s počtem senzorů až na hodnotu 247, což byla poslední hodnota, při které Arduino stále stíhalo senzory obsluhovat.

V odezvě nastal největší rozdíl od rovnoměrného měření se stejnými parametry. Při dávkové aktualizaci odezva rostla lineárně, což se nadá říci o odezvě v minulém testu (graf na Obrázku 4.3). Zároveň také byla odezva mnohem vyšší, zejména pro nízké počty senzorů.

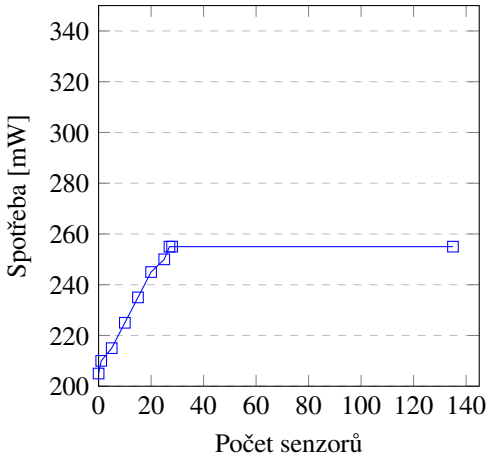
Deska	Interval aktualizace	Rozprostření aktualizací
Uno	256 ms	Dávkové

Počet senzorů	Využití RAM [KiB]	Zatížení CPU [%]	Spotřeba energie [mW]	Odezva [ms]
0	317	0	205	0
1	330	3	210	0
5	366	18	215	37
10	411	36	225	86
15	456	54	235	133
20	501	72	245	181
25	546	90	250	229
27	564	98	255	247
28	573	100	255	∞

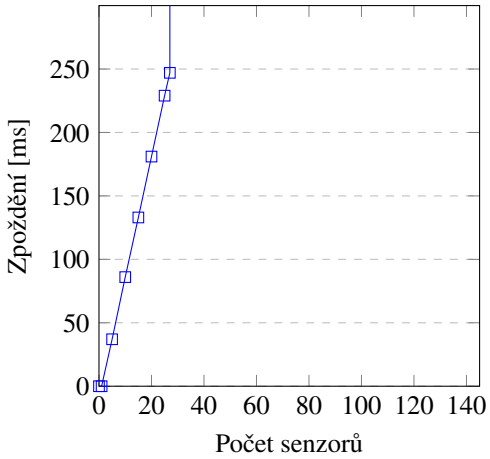
Tabulka 4.2 Přehled naměřených hodnot



Obrázek 4.4 Vliv počtu senzorů na zatížení CPU



Obrázek 4.5 Vliv p. senzorů na spotřebu



Obrázek 4.6 Vliv počtu senzorů na odezvu

4.4 Arduino Uno, maximální počet senzorů

Následující měření fixuje interval a rozprostření aktualizace a ukazuje vliv počtu senzorů na spotřebu zdrojů. Zvoleným intervalem obnovení je 1280 ms a aktualizace jsou náhodně rozprostřeny.

Rychlost obnovení je zvýšena na dostatečnou hodnotu, aby Arduino Uno zvládlo obsloužit maximální počet senzorů, který se vejde do paměti (135 senzorů). Výsledkem měření je zatížení zdrojů desky, pokud povolíme pomalejší obnovení (v rovnoměrném rozložení) za cílem možnosti správy co největšího počtu senzorů, které povoluje paměť zařízení.

4.4.1 Shrnutí

Výsledky měření jsou vidět na druhé straně v Tabulce 4.3. Arduino při této rychlosti aktualizace stíhá spravovat všechny senzory, které se vejdou do paměti.

4.4.2 CPU

Využití procesorového času rostlo lineárně s počtem senzorů, jak je možné vidět z grafu na Obrázku 4.7. Z dat v Tabulce 4.3 lze odvodit, že jeden senzor při této rychlosti obnovování zabíral průměrně 0,7 % výkonu CPU, což je výrazně nižší hodnota než v minulém měření.

Zatížení při maximálních množství nedosáhlo 100 %, je zde tedy ještě nějaká rezerva, buďto pro pár dalších senzorů při optimálnějším využití paměti, nebo složitější výpočet v průběhu aktualizace dat.

4.4.3 RAM

Na úroveň limitace pamětí v tomto testu je možné nahlížet dvěma způsoby.

Z hlediska zafixovaných parametrů testu byla paměť v tomto testu již limitující, ale vzhledem k vysokému vytížení ostatních zdrojů nebyla limitace zásadním problémem, který by bylo nutné odstranit pro vyšší efektivitu.

Z hlediska návrhu testu jako test s co nejvyšším počtem senzorů je ale paměť zásadní limitací. Pokud by byla rychlost obnovení snížena, procesor by stíhal vyřídit i větší počet senzorů, ale právě paměť zde představuje překážku.

4.4.4 Spotřeba elektrické energie

Spotřeba Arduina rostla s počtem senzorů (graf na Obrázku 4.8) a lze pozorovat měřitelné rozdíly podle počtu senzorů.

Rozdíl mezi nulovým počtem senzorů a maximem, které deska zvládala, je 60 mW. Při rozpočítání mezi počet senzorů vychází na provoz jednoho senzoru aktualizovaného tímto způsobem přibližně 0,4 mW (bez spotřeby samotného senzoru a spotřeby nečinného Arduina)

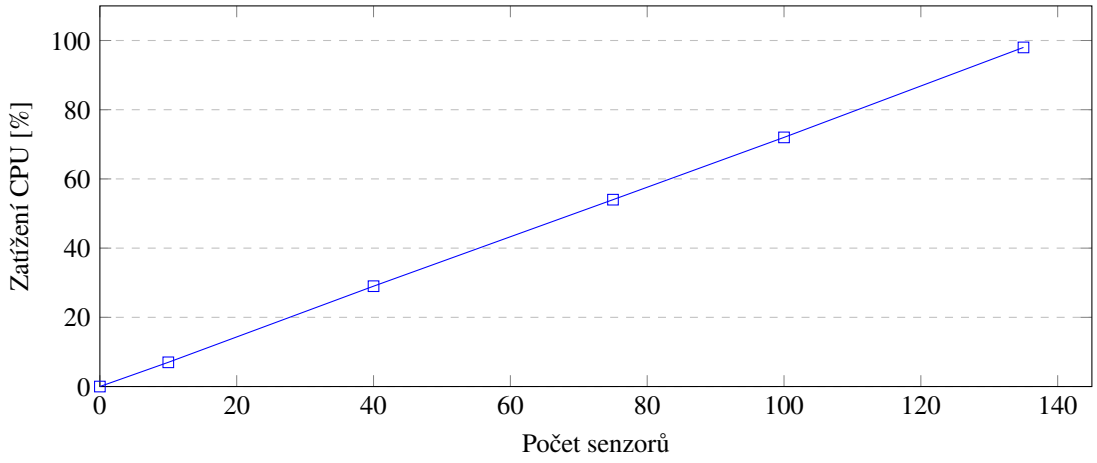
4.4.5 Odezva

Maximální odezva (možné vidět z grafu na Obrázku 4.9) začínala na 0 ms pro jeden senzor a dále se držela relativně nízko, ale s vyšším počtem už začala rapidně stoupat až na 239 ms při 135 senzorech.

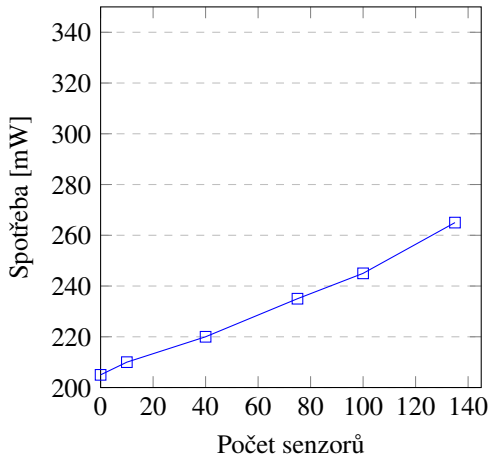
Deska	Interval aktualizace	Rozprostření aktualizací
Uno	1 280 ms	Náhodné

Počet senzorů	Využití RAM [KiB]	Zatížení CPU [%]	Spotřeba energie [mW]	Odezva [ms]
0	317	0	205	0
10	411	7	210	1
40	681	29	220	9
75	996	54	235	19
100	1 221	72	245	52
135	1 536	98	265	239

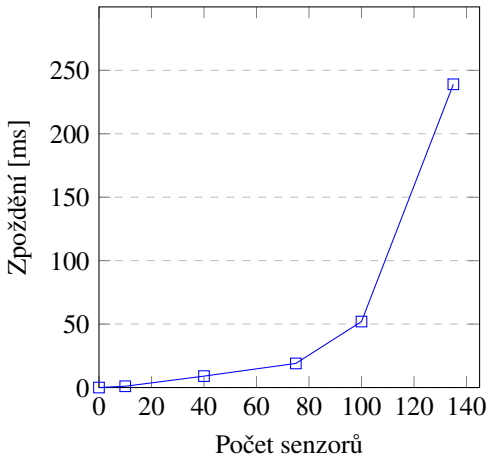
Tabulka 4.3 Přehled naměřených hodnot



Obrázek 4.7 Vliv počtu senzorů na zatížení CPU



Obrázek 4.8 Vliv p. senzorů na spotřebu



Obrázek 4.9 Vliv počtu senzorů na odezvu

4.5 Arduino Mega, rychlá aktualizace

Následující měření fixuje interval a rozprostření aktualizace a ukazuje vliv počtu senzorů na spotřebu zdrojů. Zvoleným intervalem obnovení je 256 ms a aktualizace jsou náhodně rozprostřeny.

Rychlost obnovení je nejnižší povolenou hodnotou podle vlastní specifikace v části 2.3.1. Výsledkem měření by měl být poznatek, kolik zařízení dokáže Arduino Mega obsluhovat, pokud požadujeme co nejrychlejší obnovování rovnoměrně v čase.

4.5.1 Shrnutí

Výsledky měření jsou vidět na druhé straně v Tabulce 4.4. Arduino zvládalo senzory obsluhovat až do počtu 31. Od 32 senzorů přestalo Arduino stíhat aktualizace vyřizovat (v tabulce červeně) a platforma tak není v tomto stavu již použitelná.

4.5.2 CPU

Využití procesorového času rostlo lineárně s počtem senzorů, jak je možné vidět z grafu na Obrázku 4.10. Z dat v Tabulce 4.4 lze odvodit, že jeden senzor při této rychlosti obnovování zabíral průměrně 3,1 % výkonu CPU.

Zatížení tak rostlo lineárně až do počtu 31 senzorů, kdy byl procesor zatížený na 97 %. Při přidání jednoho dalšího senzoru a dosažení počtu 32 už zatížení procesoru dosáhlo maxima a systém přestával stíhat aktualizace vyřizovat, jak je uvedeno výše.

4.5.3 RAM

V případě tohoto testu velikost paměti stačila a nebyl s ní žádný problém, ostatní prostředky byly mnohem vytíženější.

4.5.4 Spotřeba elektrické energie

Spotřeba Arduina rostla s počtem senzorů (graf na Obrázku 4.11) a lze pozorovat měřitelné rozdíly podle počtu senzorů.

Rozdíl mezi nulovým počtem senzorů a maximem, které deska zvládala, je 25 mW. Při rozpočítání mezi počet senzorů vychází na provoz jednoho senzoru aktualizovaného tímto způsobem přibližně 0,8 mW (bez spotřeby samotného senzoru a spotřeby nečinného Arduina).

4.5.5 Odezva

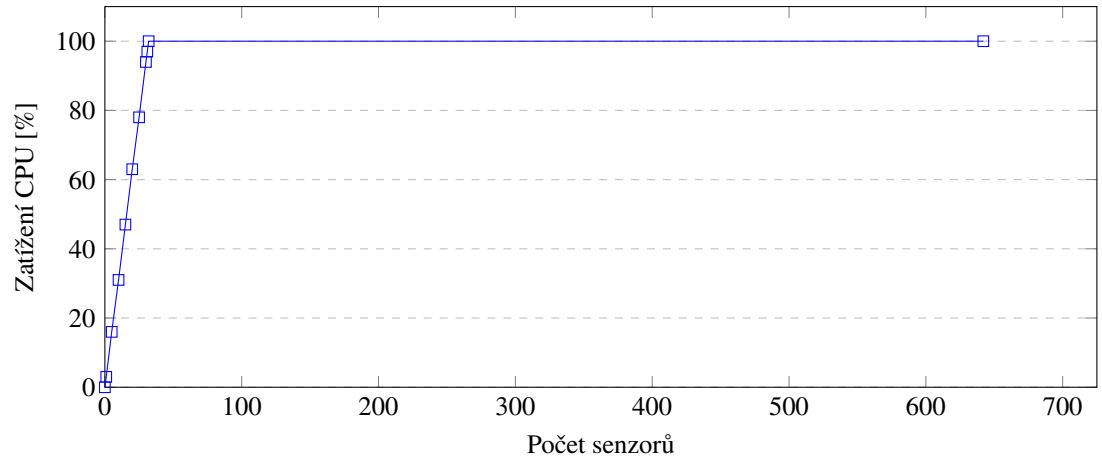
Doba odezvy (možné vidět z grafu na Obrázku 4.12) začínala na 0 ms pro jeden senzor a držela si jednocifernou hodnotu pro počet senzorů pod 15, ale s vyšším počtem už začala rapidně stoupat až na 75 ms při 31 senzorech, což byla poslední hodnota, při které Arduino stále stíhalo senzory obsluhovat.

Potom už Arduino nedokázalo nikdy vyrovnat zpoždění, které tak jenom rostlo a skončilo na hodnotě necelých 44 sekund za hodinu měření. Čím delší je doba měření, tím vyšší je tato hodnota, roste do nekonečna.

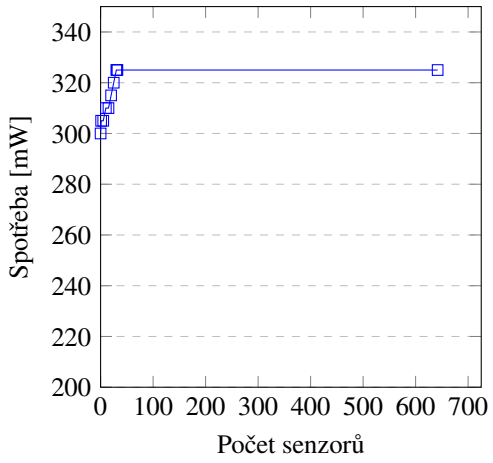
Deska	Interval aktualizace	Rozprostření aktualizací
Mega	256 ms	Náhodné

Počet senzorů	Využití RAM [KiB]	Zatížení CPU [%]	Spotřeba energie [mW]	Odezva [ms]
0	357	0	300	0
1	370	3	305	0
5	406	16	305	0
10	451	31	310	6
15	196	47	310	10
20	541	63	315	28
25	586	78	320	40
30	631	94	325	67
31	640	97	325	75
32	649	100	325	∞

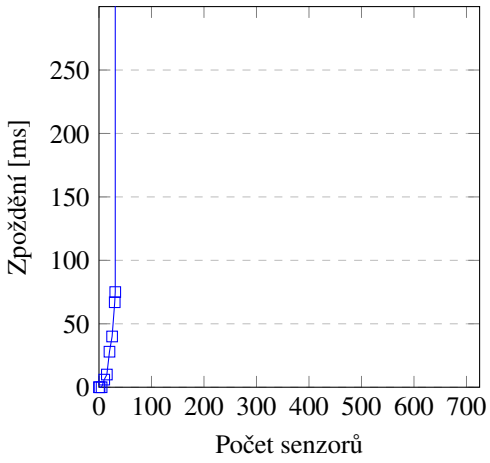
Tabulka 4.4 Přehled naměřených hodnot



Obrázek 4.10 Vliv počtu senzorů na zatížení CPU



Obrázek 4.11 Vliv p. senzorů na spotřebu



Obrázek 4.12 Vliv počtu senzorů na odezvu

4.6 Arduino Mega, maximální počet senzorů

Následující měření fixuje interval a rozprostření aktualizace a ukazuje vliv počtu senzorů na spotřebu zdrojů. Zvoleným intervalem obnovení je 5 376 ms a aktualizace jsou náhodně rozprostřeny.

Rychlost obnovení je zvýšena na dostatečnou hodnotu, aby Arduino Mega zvládlo obsloužit maximální počet senzorů, který se vejde do paměti (642 senzorů). Výsledkem měření je zatížení zdrojů desky, pokud povolíme pomalejší obnovení (v rovnoměrném rozložení) za cílem možnosti správy co největšího počtu senzorů, které povoluje paměť zařízení.

4.6.1 Shrnutí

Výsledky měření jsou vidět na druhé straně v Tabulce 4.5. Arduino při této rychlosti aktualizace stíhá spravovat všechny senzory, které se vejdou do paměti.

4.6.2 CPU

Využití procesorového času rostlo lineárně s počtem senzorů, jak je možné vidět z grafu na Obrázku 4.13. Z dat v Tabulce 4.5 lze odvodit, že jeden senzor při této rychlosti obnovování zabíral průměrně 0,15 % výkonu CPU.

Zatížení při maximálních množství nedosáhlo 100 %, je zde tedy ještě nějaká rezerva, buďto pro pár dalších senzorů při optimálnějším využití paměti, nebo složitější výpočet v průběhu aktualizace dat.

4.6.3 RAM

Na úroveň limitace pamětí v tomto testu je možné nahlížet dvěma způsoby.

Z hlediska zafixovaných parametrů testu byla paměť v tomto testu již limitující, ale vzhledem k vysokému vytížení ostatních zdrojů nebyla limitace zásadním problémem, který by bylo nutné odstranit pro vyšší efektivitu.

Z hlediska návrhu testu jako test s co nejvyšším počtem senzorů je ale paměť zásadní limitací, pokud by byla rychlost obnovení snížena, procesor by stíhal vyřídit i větší počet senzorů, ale právě paměť zde představuje překážku.

4.6.4 Spotřeba elektrické energie

Spotřeba Arduina rostla s počtem senzorů (graf na Obrázku 4.14) a lze pozorovat měřitelné rozdíly podle počtu senzorů.

Rozdíl mezi nulovým počtem senzorů a maximem, které deska zvládala, je 60 mW. Při rozpočítání mezi počet senzorů vychází na provoz jednoho senzoru aktualizovaného tímto způsobem přibližně 0,4 mW (bez spotřeby samotného senzoru a spotřeby nečinného Arduina)

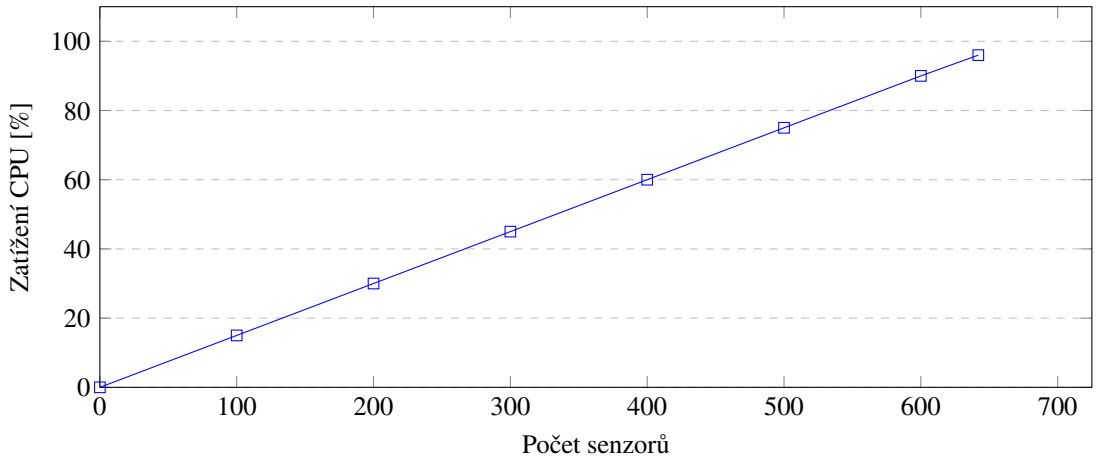
4.6.5 Odezva

Maximální odezva (možné vidět z grafu na Obrázku 4.15) začínala na 0 ms a dále se držela relativně nízko, ale s vyšším počtem už začala rapidně stoupat až na 343 ms při 642 senzorech.

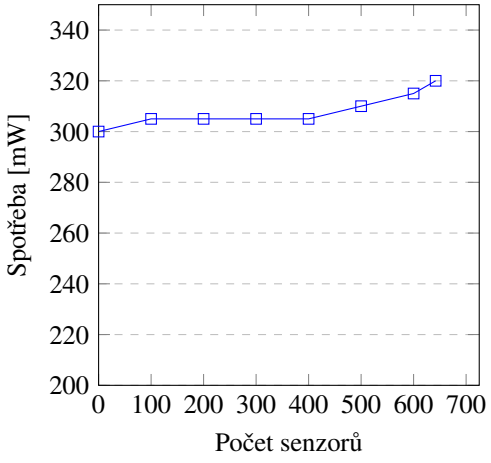
Deska	Interval aktualizace	Rozprostření aktualizací
Mega	5 376 ms	Náhodné

Počet senzorů	Využití RAM [KiB]	Zatížení CPU [%]	Spotřeba energie [mW]	Odezva [ms]
0	357	0	300	0
100	1 261	15	305	8
200	2 161	30	305	33
300	3 061	45	305	41
400	3 961	60	305	66
500	4 861	75	310	95
600	5 761	90	315	240
642	6 139	96	320	343

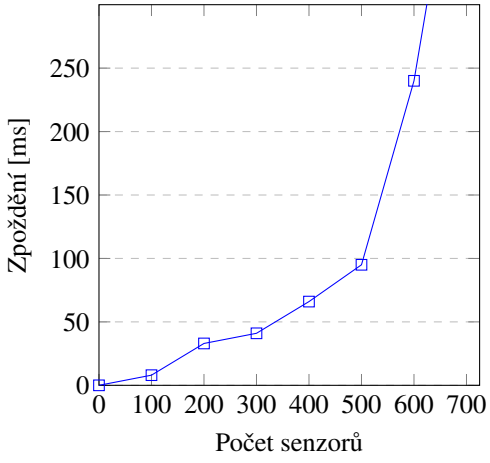
Tabulka 4.5 Přehled naměřených hodnot



Obrázek 4.13 Vliv počtu senzorů na zatížení CPU



Obrázek 4.14 Vliv p. senzorů na spotřebu



Obrázek 4.15 Vliv počtu senzorů na odezvu

Kapitola 5

Rozbor výsledků měření

V kapitole jsou shrnuty a porovnávány výsledky z předchozí kapitoly a je představen rozhodovací algoritmus postupu při návrhu senzorové sítě postavené na představené platformě.

Výsledky měření ukazují měřitelné rozdíly nejen mezi rychlostí aktualizací dat ze senzorů, ale i způsobem aktualizování. Dalším pozorováním jsou rozdílné výsledky Arduino desek, a to nejen v oblasti využití paměti, ale i výkonu a spotřeby elektrické energie.

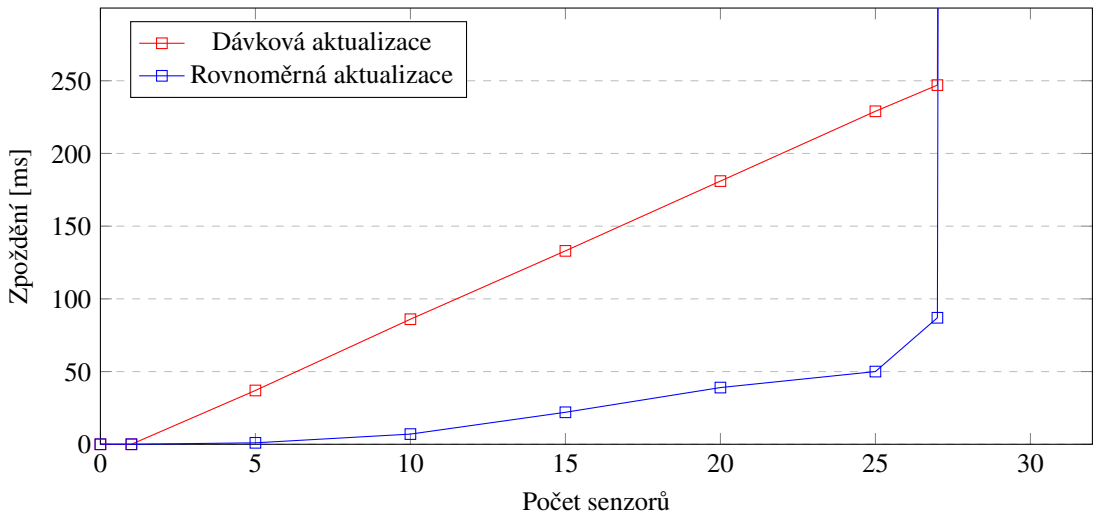
5.1 Způsob aktualizace

Na rozdíl od náhodného rozložení aktualizací, kdy bylo zatížení procesoru rozprostřené rovnoměrně přes celou dobu běhu, u dávkové aktualizace se vždy vnitřně střídají fáze vysokého zatížení a nečinnosti. Tento fakt se projevuje na výsledcích různě, jak je možné vidět z detailního popisu níže.

Největším rozdílem v měření s různými způsoby aktualizace a jinak stejnými parametry je jednoznačně zpoždění (graf na Obrázku 5.1).

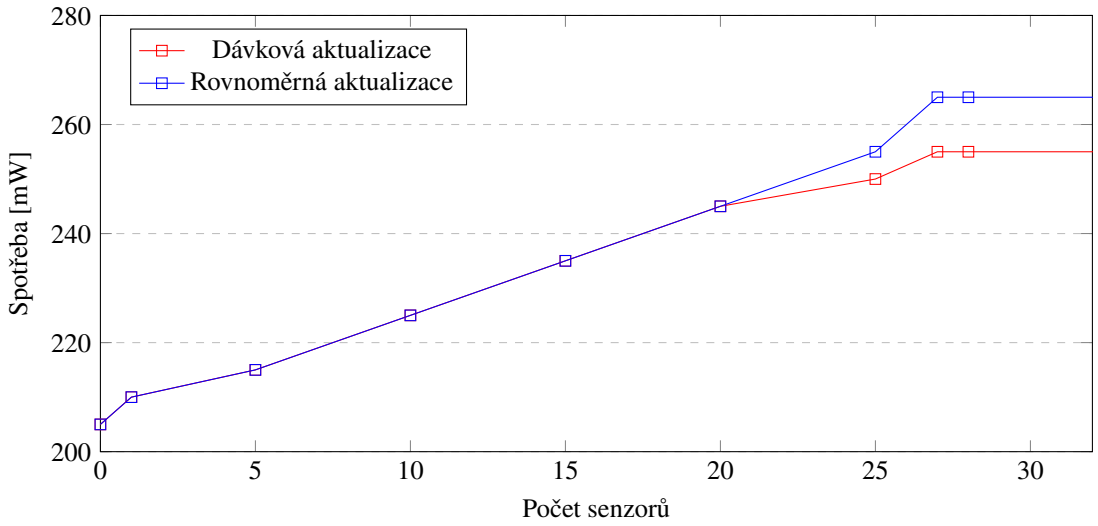
Tento fenomén je očekávaný a je možné ho vysvětlit způsobem aktualizace. Při rovnoměrném zatížení se senzory aktualizují průběžně a ve chvíli, kdy nastane požadovaný čas aktualizace libovolného senzoru, není procesor příliš vytížený.

Naopak v případě dávkové aktualizace je první senzor v dávce vždy aktualizován ve správný čas, ale ostatní senzory, které mají být aktualizovány ve stejnou chvíli, jsou zpožděny všemi ostatními. Procesor je v tu chvíli zatížen na 100 % a nestíhá aktualizace vyřizovat. Nakonec je sice zpoždění vyrovnáno, ale poslední senzor ve zvoleném pořadí má v tuto chvíli relativně vysokou hodnotu odezvy, jejíž hodnota se stane maximem a je možné jí vidět z již zmíněného grafu na Obrázku 5.1.



Obrázek 5.1 Srovnání zpoždění podle způsobu aktualizace, Uno, 256 ms

Další rozdíl je možné pozorovat ve spotřebě (graf na Obrázku 5.2). V porovnání s rovnoměrnými aktualizacemi se mírně snížila. Toto pozorování je možné vysvětlit již zmíněnými výkyvy zatížení a nečinností. Z výsledků je patrné, že tento typ zátěže je mírně šetrnější na spotřebu.



Obrázek 5.2 Srovnání spotřeby podle způsobu aktualizace, Uno, 256 ms

Jelikož zatížení CPU je měřeno po delších intervalech, než probíhá aktualizace, způsob aktualizací nemá na hodnotu vytížení CPU vliv a naměřené hodnoty jsou stejné pro oba způsoby aktualizací.

Stejně jako na zatížení CPU, ani na paměť RAM nemá způsob aktualizací žádný vliv.

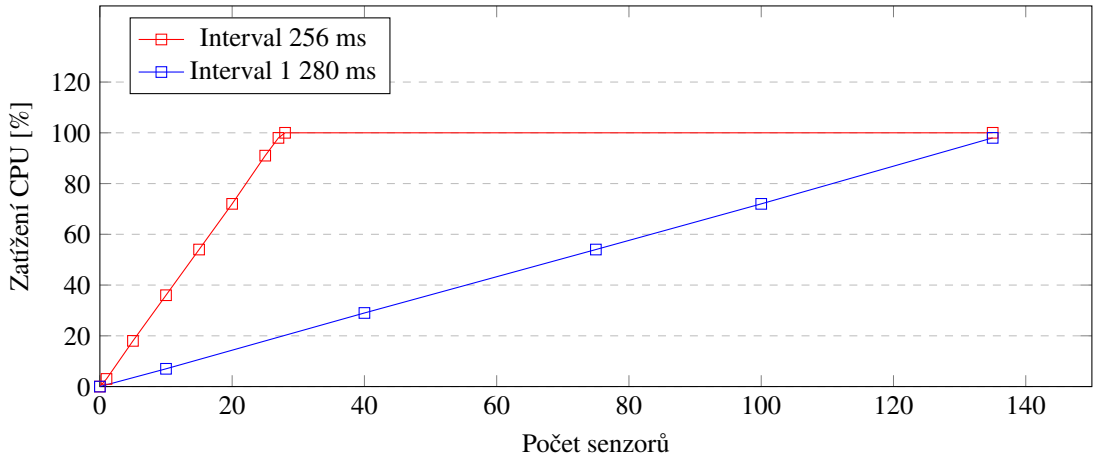
5.2 Rychlost obnovení

Rychlost aktualizací senzorů měla podle očekávání zásadní vliv na zatížení zdrojů desek. Pozorované rozdíly je možné najít v této části.

Při zvýšení rychlosti obnovení (snížení intervalu) musí procesor provádět aktualizace častěji a stane se vytíženější. Tuto skutečnost lze vidět z grafu na Obrázku 5.3.

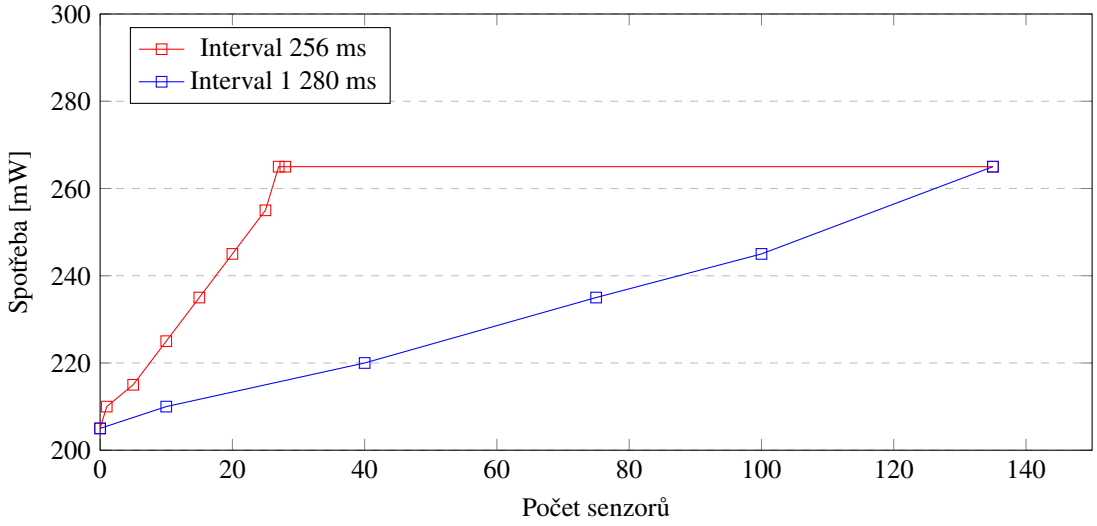
Porovnávané intervaly jsou 256 ms a pomalejších 1 280 ms, což je pětinásobek první hodnoty. Zatížení procesoru kleslo lineárně s navýšením intervalu, pomalejší interval má téměř přesně pětkrát nižší zatížení CPU.

Rozdíl v zatížení při změně rychlosti obnovení je tak významný, že při rychlejší aktualizaci se platforma stane limitovaná procesorem mnohem dříve, než při aktualizacích pomalejších.

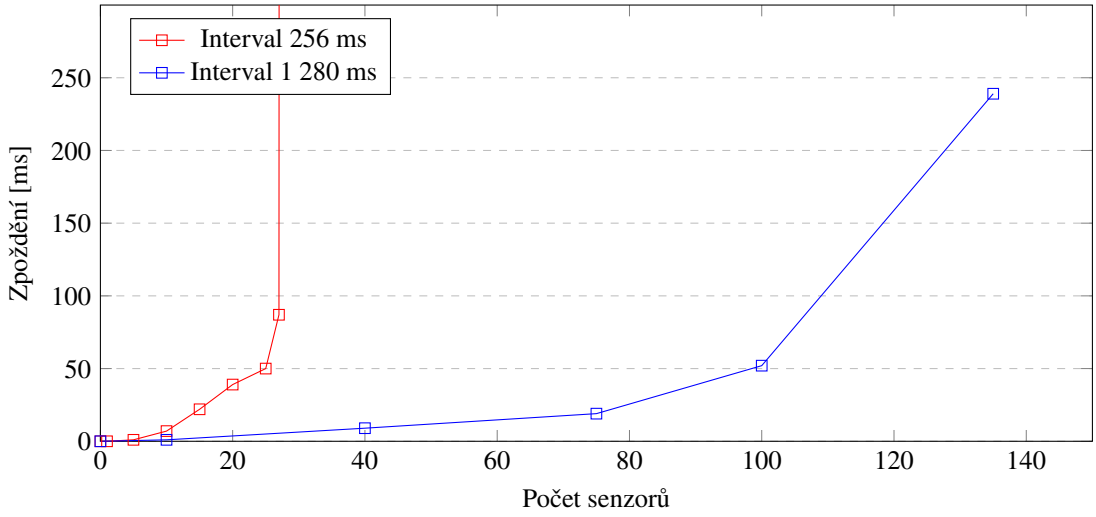


Obrázek 5.3 Srovnání zatížení CPU podle frekvence obnovování, náhodné, Uno

Podobně je na tom i spotřeba elektrické energie a odezva (grafy na Obrázku 5.4 a 5.5), které následují trend procesoru z předchozího bodu.



Obrázek 5.4 Srovnání spotřeby podle frekvence obnovování, náhodné, Uno



Obrázek 5.5 Srovnání odezvy podle frekvence obnovování, náhodné, Uno

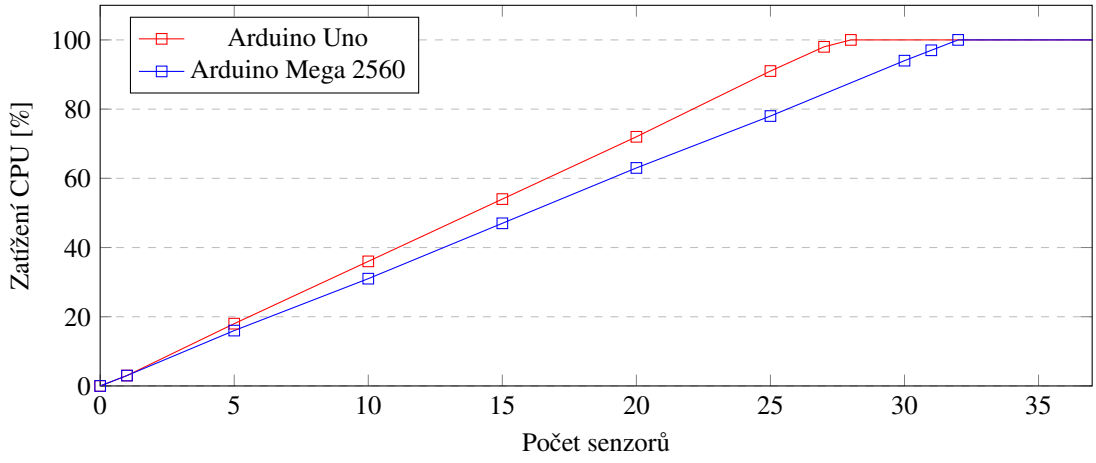
5.3 Model desky

Rozdíl mezi deskami Arduino se ukázal jako významný, a to nejen v oblasti velikosti RAM. Následující část popisuje pozorované rozdíly.

Prvotním rozdílem mezi deskami je výkon zpracování senzorů. Z grafu na Obrázku 5.6 lze vidět, že i přes stejnou frekvenci procesoru se deska Mega ukázala jako rychlejší při vyřizování komunikace, a dokáže tak obsluhovat větší počet senzorů při stejné rychlosti jejich aktualizace než Uno.

Toto zrychlení u desky Mega je možné přiřadit k typu sériové komunikace, která probíhala přes nativní linku UART a nemusela být emulována softwarově. S tím souvisel i 2x vyšší baudrate, kterého bylo možné na hardwarové lince dosáhnout.

Důsledkem je nižší zatížení procesoru desky Mega na jeden senzor (3,1 %), v porovnání s hodnotou pro desku Uno (3,6 %). Deska Mega se tak ukázala pro tento úkol přibližně o 16 % výkonnější.



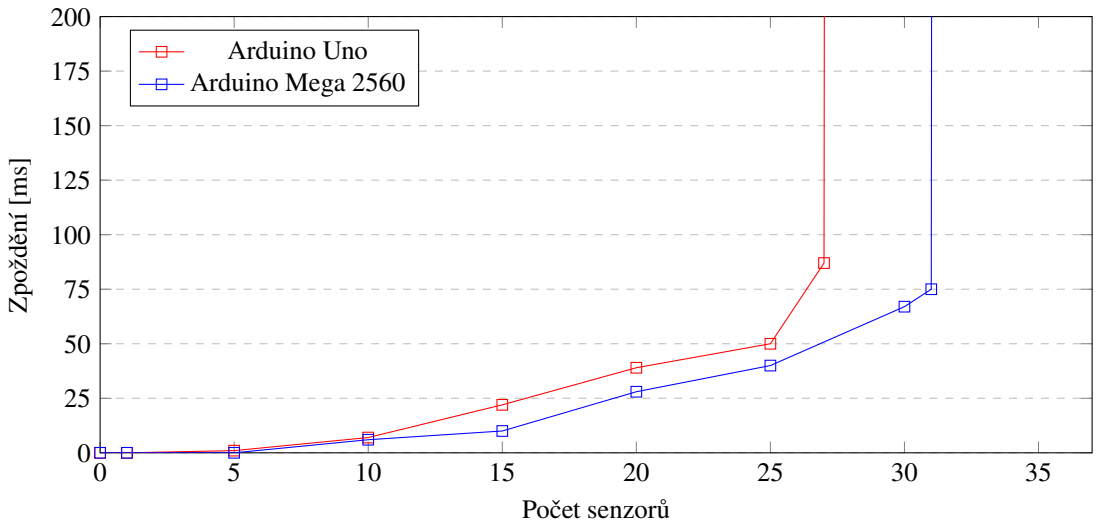
Obrázek 5.6 Srovnání zátěže CPU podle modelu desky, 256 ms, náhodné

Nejdůležitějších rozdílem se podle očekávání stala velikosti paměti RAM. Pokud přizpůsobíme ostatní parametry, obě desky se stanou limitovány právě velikostí paměti.

Limit počtu senzorů je tak 135 u modelu Uno a 642 u modelu Mega. Tento limit je patrný z grafů v minulé kapitole, konkrétně Obrázek 4.7 pro Uno, respektive Obrázek 4.13 pro Mega.

Arduino Mega (kvůli jinému způsobu komunikace přes sériovou linku) spotřebovává o 40 B více při stejném počtu senzorů než Uno. Vzhledem k velikosti paměti lze ale stále usoudit, že deska Mega má v oblasti paměti přesto výhodu.

S rozdílem v zatížení CPU souvisí i délka odezvy. Jak je možné nahlédnout v grafu na Obrázku 5.7, výkonnější Mega drželo maximální odezvu nižší než deska Uno.



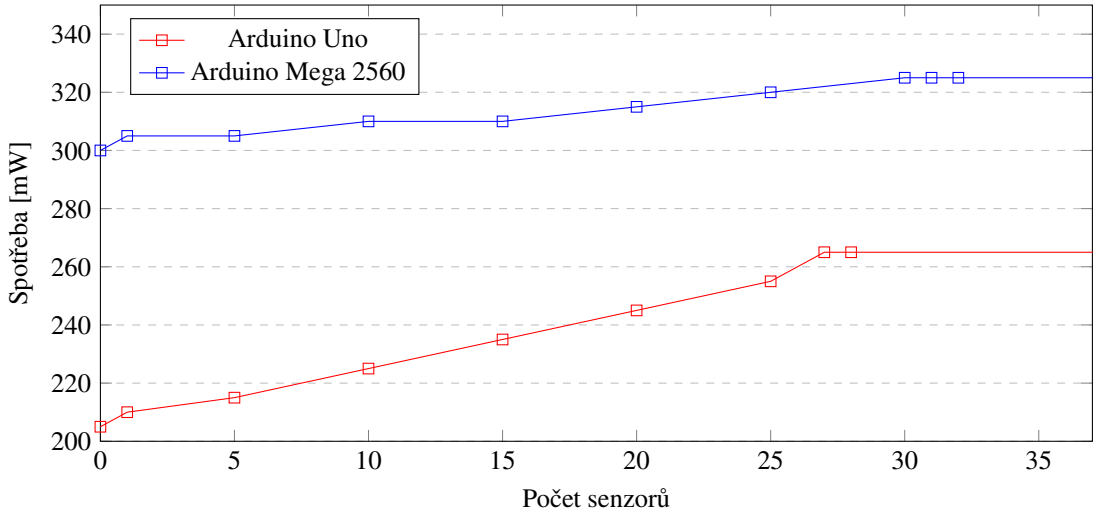
Obrázek 5.7 Srovnání odezvy podle modelu desky, 256 ms, náhodné

U obou desek je možné pozorovat zvyšování spotřeby elektrické energie při navyšování počtu senzorů.

V případě desky Uno je rozdíl markantnější – mezi nejnižším a nejvyšším počtem je rozdíl 60 mW, při plném zatížení tak spotřeba vzrostla o přibližně 29 % vůči běhu naprázdno bez senzorů.

Arduino Mega samo o sobě spotřebovávalo více energie než Uno, a to o 95 mW. Rozdíl v zatížení u této desky ale není na spotřebě tak patrný, hodnota rozdílu mezi nejnižším a nejvyšším počtem činila pouze 25 mW, což je přibližně o 8 % více než bez zatížení.

Rozdíl ve spotřebě mezi deskami se snižoval s narůstajícím počtem senzorů, přesto ale byla spotřeba desky Mega vždy výrazně vyšší (graf na Obrázku 5.8).



Obrázek 5.8 Srovnání spotřeby podle modelu desky, 256 ms, náhodné

5.4 Výsledný algoritmus návrhu

S pomocí získaných dat je možné dojít k několika závěrům a poznatkům, jak postupovat při návrhu senzorové sítě, která bude řízena mikrokontrolérem Arduino.

V následujícím textu jsou uvažovány předpoklady pro senzorovou síť definované v průběhu práce, tedy pravidelná aktualizace a udržování hodnot určitého počtu senzorů, kde každý senzor může být obnovován s různou frekvencí a v jiný čas. Po aktualizaci hodnoty senzoru se předpokládá drobný výpočet a následné odeslání požadavku na aktivní prvek, který je umístěn rovněž v této síti.

Prvotní otázkou pro navržení takové sítě je počet senzorů, které je požadované udržovat. Počet senzorů větší než 642 zachází za hranice zkoumaného hardwaru Arduino a není možné takovou síť nechávat jedním tímto mikrokontrolérem spravovat.

Pokud je požadovaný počet senzorů v rozmezí 136 a 642, je možné síť tímto způsobem provozovat, ale není možné využít Arduino Uno, které pro tento účel nemá dostatečnou paměť. Řešením je použití Arduino Mega.

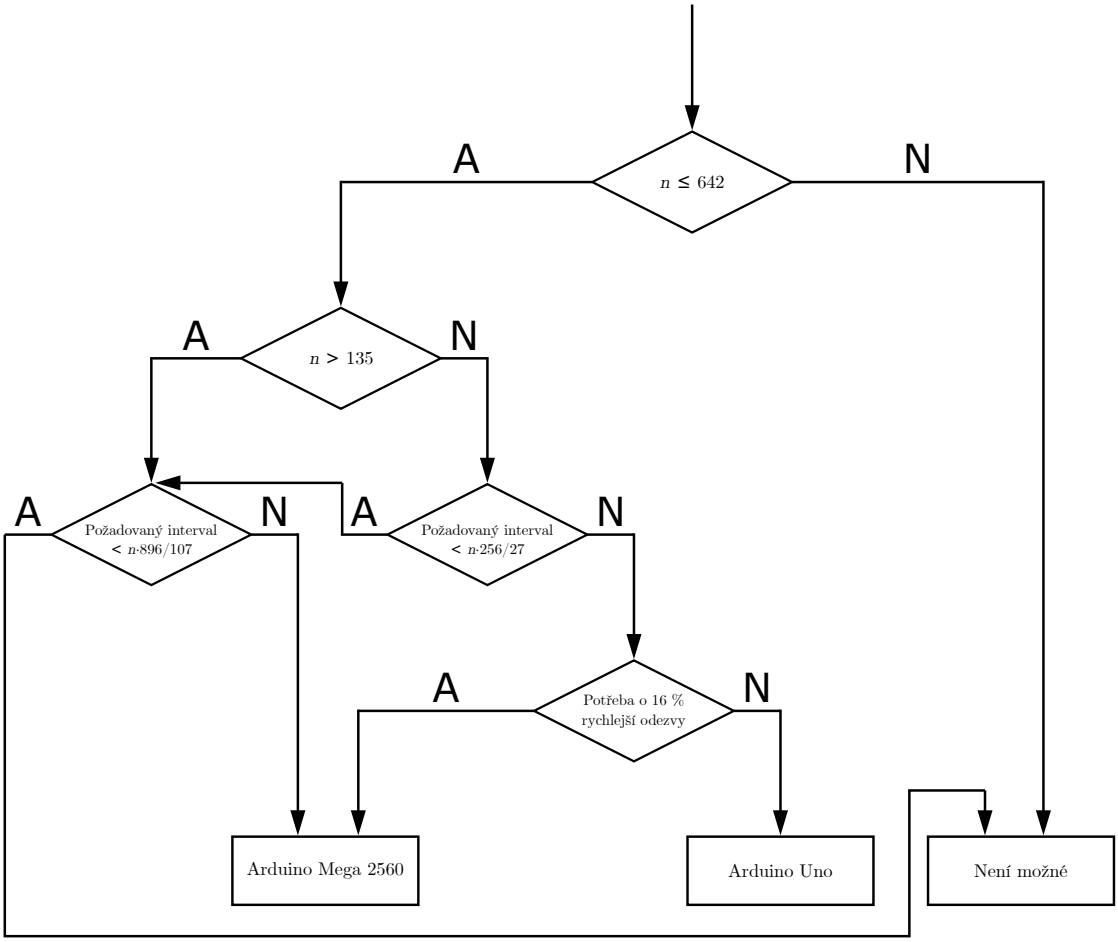
Důležitým faktorem v procesu rozhodování je interval, jak často mají být senzory aktualizovány. Pokud není nutné senzory aktualizovat častěji než jednou za 5,4 sekundy, neměl by nikdy nastat problém s přetížením Arduina pouze obnovováním hodnot.

V opačném případě je situace složitější a je nutné vzít v potaz počet senzorů v síti. Nejnižší frekvence obnovení, které je možné bezpečně dosáhnout s deskou Arduino Uno, je možné shora odhadnout výrazem $\frac{n \cdot 256}{27}$, kde n je požadovaný počet senzorů. V případě desky Arduino Mega je možné dostat stejný odhad výpočtem $\frac{n \cdot 896}{107}$.

Pokud je požadovaná rychlost obnovení vyhovující pro obě desky a není problém o 16 % vyšší odezva, doporučuje se použití desky Arduino Uno, a to kvůli až o 31 % nižší spotřebě a výrazně nižší pořizovací ceně.

V případě, že navrhované řešení senzorové sítě nepožaduje ze své podstaty aktualizace více senzorů provádět najednou, doporučuje se rozložit čas aktualizací senzorů co nejvíce rovnoměrně, kvůli rozložení zátěže mikrokontroléru a snížení tak rozdílů mezi odezvami mezi senzory.

Popsaný postup lze také názorněji vidět na Obrázku 5.9.



Obrázek 5.9 Rozhodovací diagram postupu návrhu sítě

Závěr

V této práci se podařilo navrhnout a implementovat platformu umožňující připojení externích senzorů k mikrokontroléru Arduino. Platforma byla navržena s důrazem na její efektivitu a bylo vyzkoušeno několik možností řídící jednotky v podobě různých desek Arduino.

Následně byla navržena a naprogramována jednoduchá počítačová aplikace, která emuluje připojené senzory. S pomocí této aplikace byla již zmíněná platforma otestována a bylo měřeno využití systémových prostředků. Toto měření zahrnovalo zatížení procesoru, maximální odezvu, spotřebu paměti RAM a spotřebu elektrické energie.

Po naměření těchto údajů byly výsledky shrnuty a analyzovány. Výsledkem této analýzy byl algoritmus pro efektivní návrh senzorové sítě za použití technologií využitých v práci. Na základě toho byla práce doplněna o doporučení efektivního návrhu sítě.

V budoucnu by bylo možné práci rozšířit prozkoumáním více Arduino desek sloužících k řízení sítě, nebo změnou návrhu aktualizací senzorů, který by umožnil například více možných hodnot intervalu aktualizace, nebo různě náročného výpočtu z přijatých hodnot senzorů.

Bibliografie

1. ARDUINO. *About Arduino* [online]. Arduino, 2021-09 [cit. 2022-04-13]. Dostupné z: https://www.arduino.cc/en/about.
2. YE, Fan; ZHONG, Gary; LU, Songwu; ZHANG, Lixia. GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks. *Wireless Networks*. 2005, roč. 11, s. 285–298. ISBN 978-3-540-02111-7. Dostupné z DOI: 10.1007/3-540-36978-3_44.
3. YE, Fan; CHEN, A.; LU, Songwu; ZHANG, Lixia. A scalable solution to minimum cost forwarding in large sensor networks. In: *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*. 2001, s. 304–309. Dostupné z DOI: 10.1109/ICCCN.2001.956276.
4. GLÓRIA, André; CERCAS, Francisco; SOUTO, Nuno. Comparison of communication protocols for low cost Internet of Things devices. In: *2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2017, s. 1–6. Dostupné z DOI: 10.23919/SEEDA-CECNSM.2017.8088226.
5. AL-SARAWI, Shadi; ANBAR, Mohammed; ALIEYAN, Kamal; ALZUBAIDI, Mahmood. Internet of Things (IoT) communication protocols: Review. In: *2017 8th International Conference on Information Technology (ICIT)*. 2017, s. 685–690. Dostupné z DOI: 10.1109/ICITECH.2017.8079928.
6. SHNAYDER, Victor; HEMPSTEAD, Mark; CHEN, Bor-rong; ALLEN, Geoff Werner; WELSH, Matt. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In: *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. Baltimore, MD, USA: Association for Computing Machinery, 2004, s. 188–200. SenSys '04. ISBN 1581138792. Dostupné z DOI: 10.1145/1031495.1031518.
7. TELECOMMUNICATIONS INDUSTRY ASSOCIATION (TIA). *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems systems*. Telecommunications Industry Association (TIA), 1998-03. Standard, TIA-485, Revision A. Dostupné také z: https://global.ihs.com/doc_detail.cfm?&csf=TIA&item_s_key=00032964&item_key_date=870024&input_doc_number=485&input_doc_title=&org_code=TIA.
8. ARDUINO. *Most Popular — Arduino Official Store* [online]. Arduino, 2021 [cit. 2022-04-13]. Dostupné z: https://store.arduino.cc/collections/most-popular.
9. ARDUINO. *Arduino Uno R3 — Arduino Documentation* [online]. Arduino, 2022 [cit. 2022-04-13]. Dostupné z: https://docs.arduino.cc/hardware/uno-rev3.
10. ARDUINO. *Arduino Mega 2560 Rev3 — Arduino Documentation* [online]. Arduino, 2022 [cit. 2022-04-13]. Dostupné z: https://docs.arduino.cc/hardware/mega-2560.
11. ARDUINO. *SoftwareSerial Library* [online]. Arduino, 2022. Ver. 1.0 [cit. 2022-04-21]. Dostupné z: https://docs.arduino.cc/learn/built-in-libraries/software-serial.

12. ALTHAUS. *Arduino Boards — Fritzing Parts Overview* [online]. Fritzing, 2013 [cit. 2022-04-13]. Dostupné z: `https://paulvollmer.net/FritzingParts/parts.html`.

13. ECLIPSERA. *Převodník USB na RS485 chip CH340C* [online]. Drátek.cz, 2018 [cit. 2022-04-13]. Dostupné z: `https://dratek.cz/docs/produkty/0/628/1451759885.pdf`.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `exe`
  - `emulator.exe` ..... spustitelná forma emulátoru senzorů
- `src`
  - `arduino` ..... zdrojový kód Arduino programu
  - `emulator` ..... zdrojový kód emulátoru
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `data` ..... adresář s naměřenými daty ve formátu CSV
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF