Fakulta strojní

Ústav mechaniky, biomechaniky a mechatroniky

Průmyslové komunikační sítě

Bakalářská práce

Jan Matouš

Studijní program: Teoretický základ strojního inženýrství
Vedoucí práce: Ing. Martin Nečas, MSc., Ph.D.
Rok: 2024

ZADÁNÍ BAK PRÁCE

ANOTACE

Autor	Jan Matouš
Název práce	Průmyslové komunikační sítě
Thesis title	Industry communication networks
Akademický rok	2023/2024
Ústav/odbor	Ústav mechaniky, biomechaniky, mechatroniky, odbor mechaniky a mechatroniky
Vedoucí práce	Ing. Martin Nečas, MSc., Ph.D.
Údaje o práci	Počet stránek: 72 Počet obrázků: 75 Počet tabulek: 19 Přílohy: 1x CD
Klíčová slova	LoRa, rádiová komunikace, mesh, Arduino, měření teploty a vlhkosti
Keywords	LoRa, radio communication, mesh, Arduino, temperature and humidity measurements
Abstrakt	Tato bakalářská práce se věnuje experimentální realizaci a testování LoRa mesh sítě na platformě Arduino. Hlavním cílem bylo vytvořit funkční síť, která umožňuje bezdrátovou komunikaci s nízkou spotřebou energie na vzdálenosti ca. 1 km. První část práce se soustředí na praktickou realizaci LoRa mesh sítě, včetně detailního popisu použitého hardwaru a softwaru. Druhá část zahrnuje simulační ověření funkčnosti sítě, kde byly zkoumány různé scénáře provozu sítě. V třetí části práce byly provedeny experimentální měření teploty a vlhkosti. Na závěr jsou dílčí výsledky analyzovány a diskutovány.
Abstract	This bachelor thesis focuses on the experimental implementation and testing of a LoRa mesh network on the Arduino platform. The main objective was to create a functional network that enables reliable communication with low power consumption at distances ca. 1 km. The first part of the thesis focuses on the practical implementation of the LoRa mesh network, including a detailed description of the hardware and software used. The second part contains a simulation verification of the network functionality, where different scenarios of network operation were investigated. In the third part of the work, experimental measurements of temperature and humidity were performed. Finally, the partial results are analyzed and discussed.

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu práce Ing. Martinu Nečasovi, MSc., Ph.D., za jeho neustálou ochotu, trpělivost a cenné rady. Také mu děkuji za jeho kladný přístup a za jeho neutuchající touhu pro dokončování věcí. Také děkuji své rodině a blízkým přátelům za podporu, kterou mi neustále dopřávají. Děkuji.

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem Průmyslové komunikační sítě vypracoval samostatně pod vedením Ing. Martina Nečase, MSc., Ph.D. a použil jsem pouze literaturu a zdroje uvedené v seznamu literatury.

ABSTRAKT

Tato bakalářská práce se věnuje experimentální realizaci a testování LoRa mesh sítě na platformě Arduino. Hlavním cílem bylo vytvořit funkční síť, která umožňuje bezdrátovou komunikaci s nízkou spotřebou energie na vzdálenosti ca. 1 km. První část práce se soustředí na praktickou realizaci LoRa mesh sítě, včetně detailního popisu použitého hardwaru a softwaru. Druhá část zahrnuje simulační ověření funkčnosti sítě, kde byly zkoumány různé scénáře provozu sítě. V třetí části práce byly provedeny experimentální měření teploty a vlhkosti. Na závěr jsou dílčí výsledky analyzovány a diskutovány.

Klíčová slova: LoRa, rádiová komunikace, mesh, Arduino, měření teploty a vlhkosti

This bachelor thesis focuses on the experimental implementation and testing of a LoRa mesh network on the Arduino platform. The main objective was to create a functional network that enables reliable communication with low power consumption at distances ca. 1 km. The first part of the thesis focuses on the practical implementation of the LoRa mesh network, including a detailed description of the hardware and software tools used. The second part contains a simulation verification of the network functionality, where different scenarios of network operation were investigated. In the third part of the work, experimental measurements of temperature and humidity were performed. Finally, the partial results are analyzed and discussed.

Keywords: LoRa, radio communication, mesh, Arduino, temperature and humidity measurements

SEZNAM ZKRATEK A SYMBOLŮ

AES	Advanced Encryption Standard
APM	Alternative Pulse Modulation
ARM	Advanced RISC Machine
ASi	Actuator Sensor Interface
Ax	Analog pin x
Buffer	Temporary Storage
ca	Cirka
CAN	Controller area network
CS	Chip Select
CIP	Common Industrial Protocol
DC	Distributed Clock
DDL	Data Definition Language
DIO	Digital Input Output
DSSS	Direct Sequence spread spectrum
DSUB	D-subminiature
Dx	Digital pin x
EEPROM	Electrically Erasable Programmable Read Only Memory
FHSS	Frequency Hopping Spread Spectrum
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying
GND	Ground
GPS	Global Position System
GSM	Global System for Mobile Communications
GPRS	General Packet Radio Service
HART	Highway Addressable Remote Transducer
HSE	High Speed Ethernet
HV	High Voltage
ID	Identification
IDE	Integrated Development environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INT	Interrupt
IO	Input Output
IoT	Internet of Things
IP	Internet protocol
IT	Information Technology
ISM	Industrial, Scientific and Medical

JSON	JavaScript Object Notation
LAN	Local Area Network
LEPC	Low Energy Power Control
LV	Low Voltage
LoRa	Long Range
MAC	Media Access Control
MBP	Manchester Encoded Bus Powered
MBAP	Modbus Application Protocol
MISO	Master In Slave Out
MKR	Microcontroller Key Radio
MOSI	Master Out Slave In
MQTT	Message Queue Telemetry Transport
MSK	Minimum Shift Keying
Node	Individual part of a larger data structure
N	Node
NSS	Network Slave Select
OSI	Open Systems Interconnection
PC	Personal Computer
PLC	Programmable Logic Controllers
RAM	Random Acces Memory
RS	Recommended Standart
RSSI	Received Signal Strength Indicator
RST	Reset
RX	Receiver pin
SRAM	Static Random Acces Memory
SAE	Society of Automotive Engineers
SCK	Serial Clock
SPI	Seriál Peripheral Interface
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TX	Transmitter pin
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UCMM	Unconnected Message Manager
VCC	Voltage at the Commom Collector
WAN	Wide Area Nettwork
WiFi	Wireless fidelity
WLAN	Wireless Local Area Network

OBSAH

OBRÁZKY

TABULKY

1. Úvod

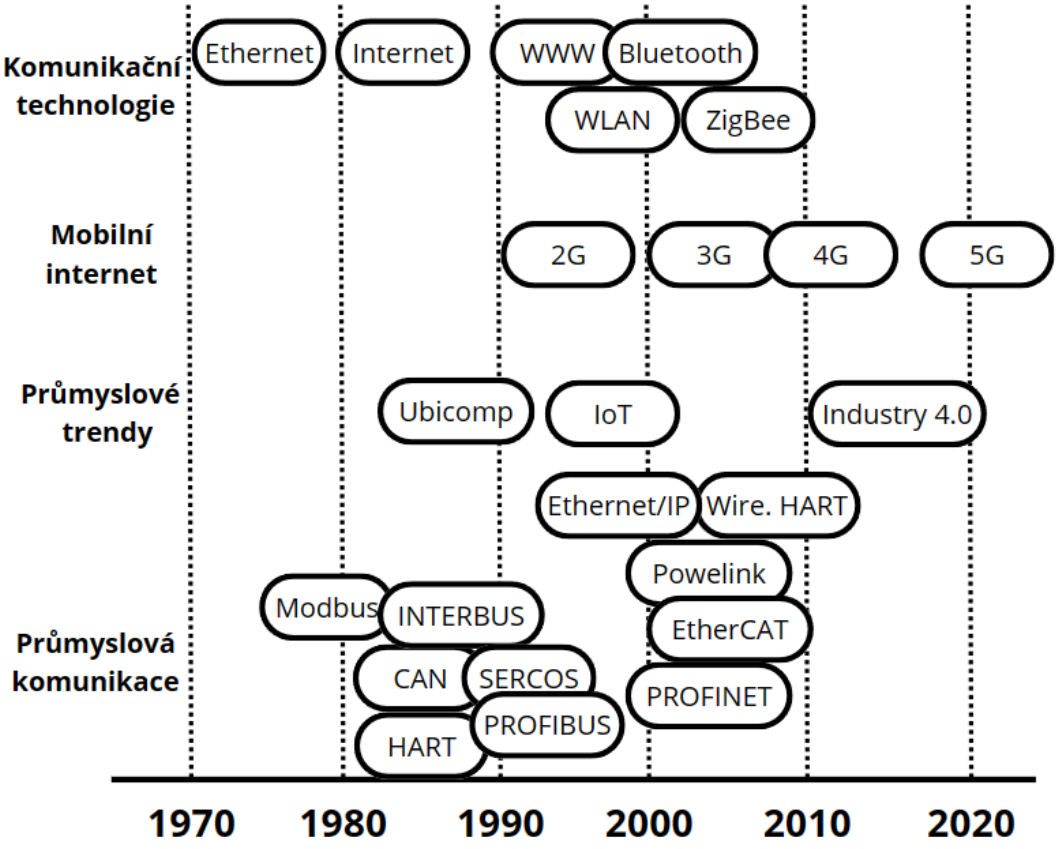
V průběhu historie byl průmysl nucen se neustále adaptovat na nové technologie. V 18. století příchod uhlí a parních strojů zapříčinil rozsáhlou mechanizaci procesů. V 19. století díky spalovacím motorům a energii v podobě elektřiny došlo opět k významným změnám v průmyslu. Příchod telefonů a nových dopravních prostředků zkrátil vzdálenosti a vedl k výrazné proměně fungování lidské společnosti. Dalším velmi významným průmyslovým pokrokem byl vznik výpočetní techniky, který nastal v druhé polovině 20. století. V současné době průmyslových trendů v podobě Průmyslu 4.0 dochází k masivní digitalizaci a propojování vzdálených výrobních linek v reálném čase. Hranice mezi fyzickou a virtuální realitou se postupně ztenčuje, moderní technologie představují významnou složku našeho běžného života. Technologie jako IoT, cloud a umělá inteligence umožňují průmyslu realizovat nové a pokročilejší způsoby výroby. Současné výrobní linky umožňují sběr velkého množství dat v reálném čase s nutností tyto informace v reálném čase sledovat, vyhodnocovat a často, ze vzdálených míst, na ně adekvátně reagovat.

2. Komunikační sítě

Aby byla zajištěná správně fungující komunikace mezi přístroji, musí být zvolen stejný komunikační jazyk na všech zařízeních. Při konfrontaci navzájem odlišných jazyků dochází k nesprávné komunikaci. Komunikační protokoly tento problém řeší. Definují sadu pravidel, za kterých může komunikace dvou nebo více entit probíhat bez výraznějších chyb. Stanovují syntax, sémantiku a synchronizaci jazyku. Protokoly mohou nabývat podoby hardwarové, softwarové nebo jejich kombinací. [1]

2.1. Historie

Jednou z prvních průmyslových implementací protokolů byla technologie fieldbus, která umožnila snížit počet zapojených kabelů. Fieldbus nevyžaduje bodové propojení mezi každým zařízením a kontrolérem. K další změně došlo kolem tisíciletí s příchodem internetu. V důsledku toho využití ethernetu začalo prudce stoupat. Nevhodnost ethernetu pracovat v reálném čase vedla ke vzniku mnoha rozdílných implementací snažící se tuto nedostatečnost vyřešit, například systém EtherCAT. Tato realtime implementace dokázala splnit velmi náročné požadavky na nízkou latenci řízení pro pohybové implementace. [2]

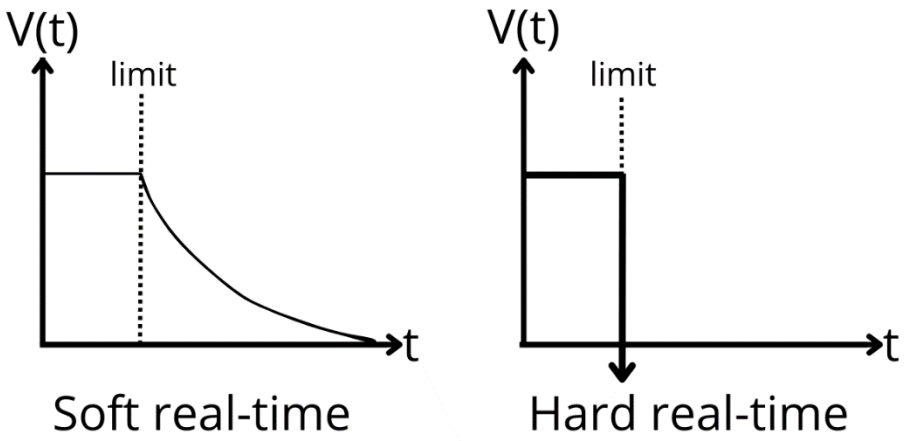


Obr. 2.1: Historické milníky významných technologií v evoluci industriální komunikace [2]

2.2. Základní rozdělení

Jednotlivé protokoly můžeme vzájemně rozlišovat na základě následujících vlastností:

- **Chování v reálném čase:**
  - Soft real-time: doba časového cyklu je volitelná, používané v aplikacích, kde při nedodržení limitu nehrozí žádné ohrožení.
  - Hard real-time: systémy od 1 do 10 ms, používané pro systémy s kritickou důležitostí dodržení limitů.
  - Isochronous real-time: systémy od 250 µs do 1 ms, dochází k vzájemné koordinaci časování pomocí synchronizovaných hodin. [3]



Obr. 2.2: Grafická reprezentace real-time systémů

- **Distribuce:**
  - LAN: sítě s omezeným dosahem o vysokých rychlostí.
  - WAN: sítě o fungující na velké dosahy o menších rychlostech.
- **Homogenita:**
  - Homogenní: jednotlivé komunikační jednotky jsou navzájem identické. Provádí stejné úkony a jsou vytvořeny ze stejných součástek.
  - Heterogenní: v síti se vyskytují alespoň dvě odlišně komunikační jednotky. [4]
- **Druh instalace:**
  - Kabelové
  - Bezdrátové

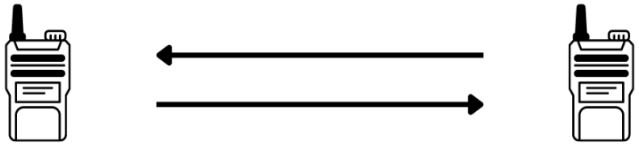
2.3. Průmyslové požadavky

- Fungování v reálném čase
- Funkční bezpečnost
- Zabezpečení: zvolený systém obsahuje víceúrovňového zabezpečení.
- Různá prostředí: důraz kladen na citlivě zvolené řešení pro danou situaci/místo. [3]

2.4. Posílání a přijímání zpráv

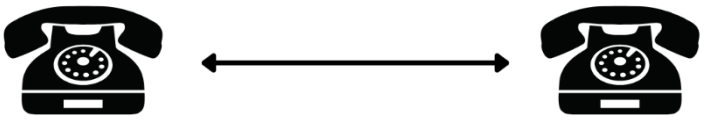
Samotnou komunikaci můžeme rozdělit podle přístupu k toku dat. Rozlišujeme dva hlavní druhy a jejich podkategorie.

- **Simplex:** jednosměrná komunikace přes jeden kanál
- **Duplex:** dvousměrná komunikace přes jeden kanál
    - **Half-duplex:** pouze jedna strana může v daný moment komunikovat přes kanál. Komunikaci můžeme připodobnit k vysílačkám. Vyskytuje se například u technologie Wi-Fi a Ethernetu 10Base5 [5]



Obr. 2.3: Schéma half-duplex komunikace pomocí vysílaček

    - **Full-duplex:** obě strany mohou současně přijímat i posílat data. Tuto technologii můžeme připodobnit k hovoru po telefonu. Full-duplex využívají technologie Ethernet 100Base-TX a LTE. [5]



Obr. 2.4: Schéma full-duplex komunikace pomocí telefonů

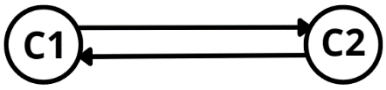
2.5. Topologie

Topologie udává fyzickou nebo logickou podobu komunikační sítě v provozním prostředí. Jedná se o tvar, který jednotlivé členy zaujímají. Dle situace se může jednat například o hvězdu, kruh, mesh apod. Zvolený tvar následně určuje silné a slabé stránky komunikace. Zvolením vhodného tvaru lze snížit spotřebu energie a zvýšit množství přenosu dat. Ke každému návrhu je proto třeba přistupovat individuálně. [6]

- **Bodová**: jedná se o nejjednodušší možnost, kde se vyskytují pouze dva členy. Jeden informaci posílá, druhý informaci přijímá. [7]

Výhody	Nevýhody
Jednoduchost Bezpečnost	Nelze rozšířit o další členy Porušení má za následek nefunkční komunikaci

Tabulka 2.1 – Výhody a nevýhody bodové topologie [8]

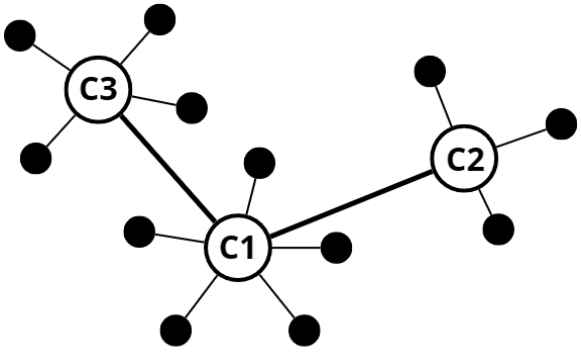


Obr. 2.5: Bodová topologie

- **Hvězda:** každá část má určené přijímače a vysílače. Disponuje vysokým výkonem, jelikož dochází k minimálním odrazům a odchylkám. Například LAN. [5] Centrální člen kontroluje veškeré funkce, shromažďuje zprávy a také se chová jako opakovač signálu. [7]

Výhody	Nevýhody
Jednoduchost Každý člen je spojen s centrálním členem pouze 1x Chyba v jednom spojení neohrozí zbytek sítě	Při poruše centrálního členu, přestane fungovat Drahá instalace Výkon závislý pouze na centrálním členu

Tabulka 2.2 – Výhody a nevýhody topologie do hvězdy [8]

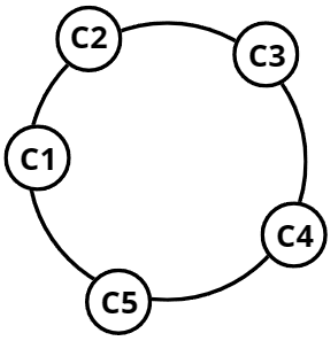


Obr. 2.6: Topologie do hvězdy

- **Kruh:** podobné jako uspořádání do hvězdy, každý segment má stanovené přijímací a vysílací členy, směr šíření dat je ale pouze v kruhu. Pokud uzel 1 pošle informaci uzlu 3, informace projde i přes uzel 2. (viz Obr. 2.7). [5]

Výhody	Nevýhody
Vysoká rychlost přenosu informace	K celkovému selhání stačí porucha 1 členu
Nedochází ke kolizím dat	Horší ochrana proti napadení
Levnější než topologie do hvězdy	Obtížné hledání chyb

Tabulka 2.3 – Výhody a nevýhody topologie do kruhu [8]

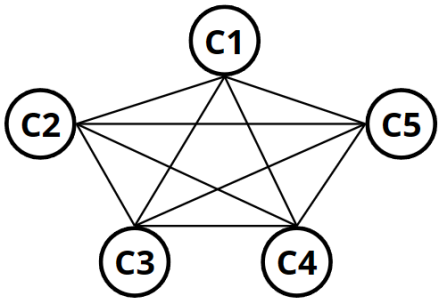


Obr. 2.7: Topologie do kruhu

- **Mesh:** každá část je spojena k jiné části pomocí jednoho kanálu, jedná se o decentralizovaný systém, ve kterém existuje větší počet cest mezi jednotlivými členy. Pokud člen C1 vyšle data do členu C4, tak zpráva se bude šířit nejvýhodnější cestou. To v některých případech může znamenat, že zpráva projde přes jiné členy. Viz Obr. 2.8. [5]

Výhody	Nevýhody
Soukromí a zabezpečení	Častá údržba
Robustní síť	Vhodnější pro menší počet zařízení
Jednoduché odhalení chyb	Obtížná instalace

Tabulka 2.4 – Výhody a nevýhody topologie mesh [8]

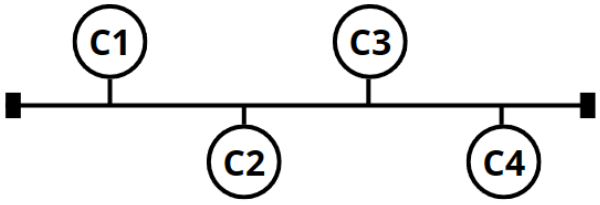


Obr. 2.8: Topologie mesh

- **Bus:** veškeré členy jsou spojeny jedním kabelem, do kterého jsou vyvedeny v uzlech jednotlivé odbočky. Každá odbočka vnáší do systému nechtěné signálové odrazy signálu. Jedná se o dvousměrný systém. Pravidly je stanovena maximální délka jedné odbočky a zároveň celková délka odboček. [5] Ideální pro aplikace s menším počtem členů. Na obou koncích kabelu je umístěn terminátor sloužící pro pohlcení signálu. [7]

Výhody	Nevýhody
Historicky nejvíce používaná topologie	Nejpomalejší topologie z hlediska přenosu dat
Jednoduchá odstranitelnost jednotlivých členů	Obtížné hledání poruch
Efektivní pro menší systémy	Dochází k častým ztrátám zpráv

Tabulka 2.5 – Výhody a nevýhody topologie bus [8]

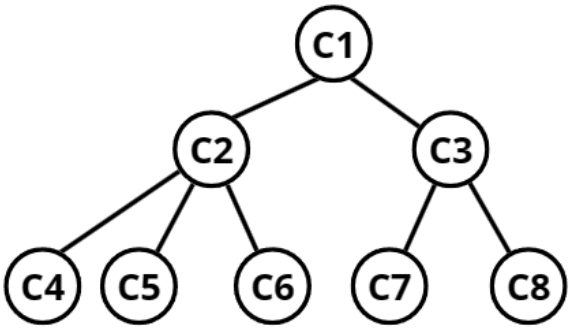


Obr. 2.9: Topologie bus

- **Strom:** tato topologie vznikne připojením několika topologií hvězd k jednomu centrálnímu členu, nebo spojením hvězd přes topologii bus. [7] Obsahuje víceúrovňový tok dat. Topologie připomíná strom díky své struktuře. Každá část se může nadále větvit při připojení nových členů. [8]

Výhody	Nevýhody
Jednoduchá detekce chyb	Funkčnost topologie závisí na hlavním členu
Snadné přidávání nových členů	Obtížná konfigurace
Menší vzdálenost cest signálu	Při větším počtu členů pomalejší přenos dat

Tabulka 2.6 – Výhody a nevýhody topologie strom [8]

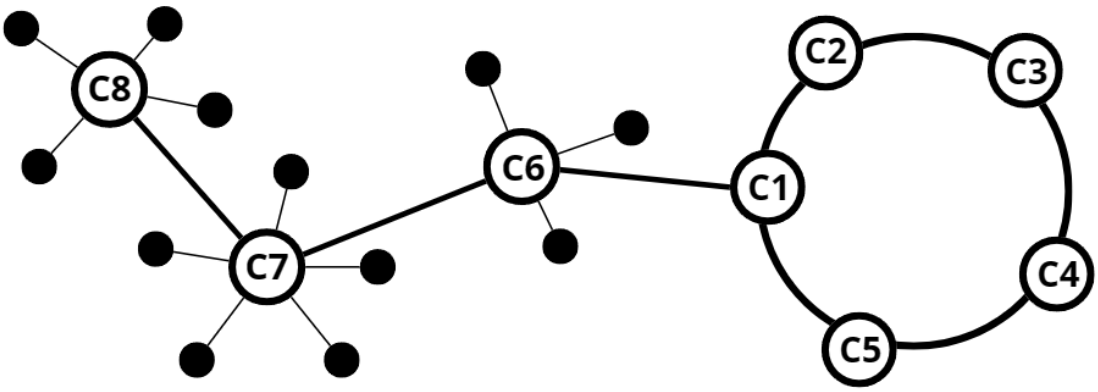


Obr. 2.10: Topologie strom

- **Hybrid:** jednotlivé topologie lze nadále spojovat dohromady za účelem vzniku nových hybridních. Jsou navrhovány k minimalizaci slabých stránek a maximalizaci stránek silných. Využívaná pro komplikovanější sítě. [8]

Výhody	Nevýhody
Dobrá přizpůsobitelnost topologie Snadné přidávání nových členů	Obtížná topologie na vytvoření Drahé

Tabulka 2.7 – Výhody a nevýhody topologie hybrid [8]



Obr. 2.11: Topologie hybrid

2.6. OSI Model (Open Systems Interconnection)

Model popisující sedm vrstev, které počítače používají pro síťovou komunikaci. Představen byl již v roce 1983.

7	**Aplikační**	Interakční vrstva mezi počítačem a člověkem, vstupní bod aplikace do sítě
6	**Prezentační**	Prezentace dat a šifrování
5	**Relační**	Udržuje komunikaci a zodpovídá za jednotlivé porty
4	**Transportní**	Posílaní dat pomocí TCP a UDP protokolů
3	**Síťová**	Definuje konkrétní fyzickou cestu dat
2	**Spojová**	Definuje formát dat
1	**Fyzická**	Posílá binární signál přes fyzické médium

Tabulka 2.8 – OSI model a základní charakteristika jeho vrstev

2.7. Průmyslové kabelové sítě

2.7.1. Úvod

Kabelové digitální komunikace měly na posun vývoje kontrolních systémů velký vliv. Každý podnikový IT systém musel umožnit uživatelům přístup k datům a zároveň implementoval jejich odlišné požadavky na podobu sítě. Například: velikost sítě, počet podporovaných zařízení, šířka pásma sítě, doba odezvy, vzorkovací frekvence apod. V průběhu času proto vznikaly různé typy průmyslových komunikačních systémů: [3]

- Sítě senzor/aktuátor – na úrovni senzorů
- Sítě sběrnic – na úrovni procesních dat
- Sítě řídících jednotek – na úrovni řídících jednotek
- Rozlehlé sítě (WAN) – na úrovni podniku

2.7.2. Sítě senzor/aktuátor

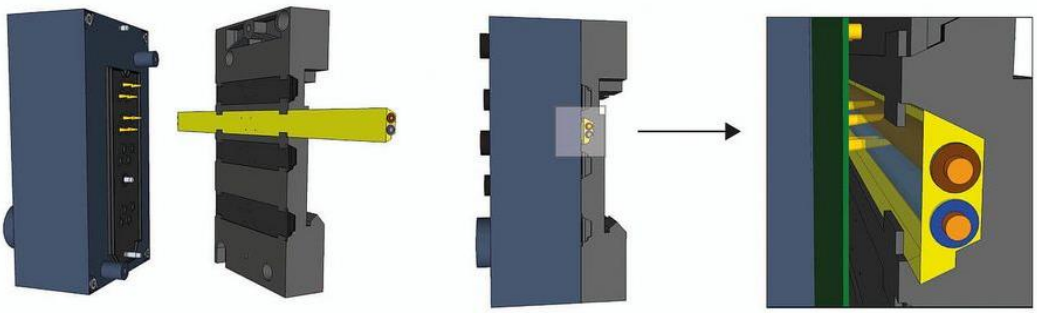
V tomto odvětví se historicky zavedly následující implementace:

- HART
- ASi
- IO Link
- CAN

- **HART:** protokol pracuje v proudovém v rozsahu 4-20 mA, je schopný obousměrné bodové komunikace. Oproti jiným technologiím stejného rozsahu nabízí lepší výkon a širší použitelnost, jelikož umožňuje průchod procesní veličiny analogovým signálem 4-20 mA a současně digitálně přenáší informace týkající se například kalibrace, diagnostiky zařízení apod. Využívá klíčování frekvenčním posuvem dle standardu Bell 202. Protokol komunikuje rychlostí 1200 bps. Funguje na principu master/slave. V jednu chvíli je možno používat až 2 masters. Příkazy lze zadávat pomocí DDL (Data Definition Language). [9] V dnešní době se HART řadí již k přežitým technologiím, avšak stálé existuje. [10]

- **ASi:** protokol fungující pro síť aktuátorů a senzorů. K vzájemnému propojení slouží kroucená dvojlinka a kabel pro napájení (max. 2 A). Zvolená konstrukce umožňuje propojit až 31 dalších zařízení na úrovni slave. Finální podobou topologie je hvězda, kruh apod. K přenosu dat se využívá technologie APM (Alternative Pulse Modulation). Rozhraní Asi kontroluje každý signál na paritu.

Implementuje se proto do velmi rušných prostředí. Jako master zařízení se využívají PC nebo PLC automaty. Výměna zpráv probíhá ve dvou nastaveních. Ve stavu jedné 4bitové zprávy, nebo ve formě více za sebou poslaných 4bitových zpráv. Pro snadnou a bezpečnou instalaci je používaná piercing-technology, při které dochází k průniku propichovacími kolíky do žil kabelu pro vytvoření spolehlivého kontaktu. [11]

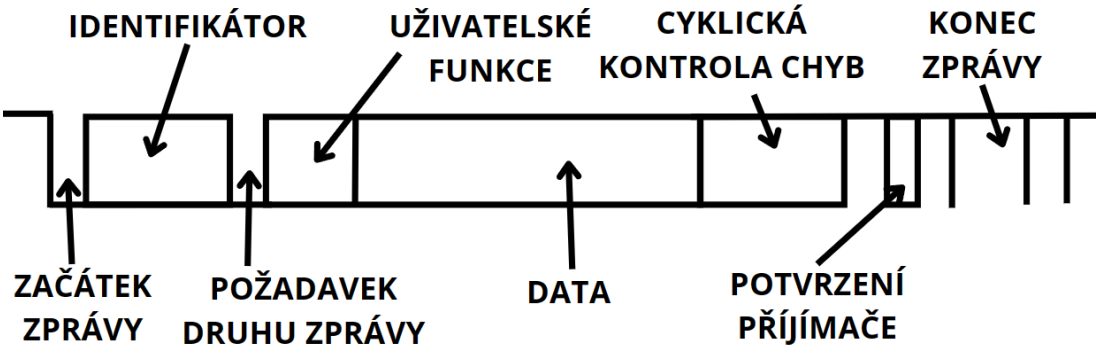


Obr. 2.12: Piercing technology [12]

- **IO-Link:** digitální komunikační protokol propojující aktuátory/senzory s PLC. Založen na mezinárodním standardu IEC 61131-9, který umožňuje posílání binárních nebo analogových signálů. IO-Link je schopen tvořit pouze bodové topologie na úrovni master/slave. Využívá třížilový kabel pro přenos informací, uzemnění a napájení o hodnotách proudu nepřesahujících 200 mA. Pro případ aktuátorů by takové napájení nebylo dostatečné, připojuje se proto navíc pětižilový kabel. Nad parametrická data umožňuje protokol navíc odesílání diagnostických dat. Data ovšem nesmějí přesahovat 32 bitů. [13]

Obr. 2.13: Zařízení IO-Link [14]

- **CAN:** komunikační standard operující rychlostí až 1 Mb/s v topologii bus. Všechna připojená zařízení slyší veškeré zprávy. Data se posílají jako zprávy obsahující 0 až 8 bajtů. Na začátek zprávy se navíc připojí jedinečné 11bitové číslo (v některých verzí až 29bitové číslo, například Extended CAN) určující důležitost zprávy a zároveň fungující jako filtr pro jednotlivá zařízení. Celkově obsahuje 47 bitů. V sítí se obvykle nachází vícero různých zpráv, takže vysílací systém stále porovnává jednotlivé zprávy podle důležitosti, tu s největší prioritou pak odesílá. Maximální délka propojovacího kabelu se odvíjí od požadované rychlosti sítě. Pro rychlost 1 Mb/s nesmí kabel překročit délku 40 m, pro rychlost 10 kb/s je povolená délka kabelu až 6 km (podmíněno rychlostí světla). Každý CAN systém podléhající standardu ISO 11898 musí obsahovat alespoň jeden terminátor pro udržení optimálního napětí a pro odstranění signálů před odrazem. Terminátor je rezistor o jmenovitém odporu 120 Ω. Opět dle standardu ISO 11898 musí propojovacím kabelem být kroucená dvojlinka. Existuje avšak i standard SAE J2411, který umožňuje pro spojení použít jen jedné linky, avšak na úkor robustnosti sítě. Jako konektory se v průmyslu používají tyto: devítipinový DSUB, pětipinový Mini-C a Micro-C, šestipinový Deutsch konektor. [15],[16]



Obr. 2.14: Schéma a popis datové struktury CAN 2.0A

- **CAN FD**: rozšířený CAN protokol. Umožňuje přenos dat o různých velikostech a frekvencích. Původně vznikl čistě pro automatizační systémy. Oproti původnímu CAN protokolu nabízí následující vylepšení:
    1. Vyšší přenos dat (až 10 Mbps)
    2. Posílaní dat o různých velikostech (1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 48, 64 bajtů)
    3. Kompatibilnost s již existujícími CAN systémy
    4. Vylepšená detekce chyb [17]

2.7.3. Sítě sběrnic

Používané technologie jsou v tomto odvětví plně průmyslově standardizované, existuje však velké množství koncepcí. Normy IEC 61158 a 61784 obsahují 10 možných druhů technologií založených na:

- vlastním protokolu – Profibus, Interbus, P-Net, WorldFIP, SwiftNet, Foundation Fieldbus H1
- Ethernetu – High Speed Ethernet, Ethernet IP, PROFINET/CBA

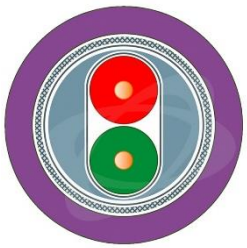
Dále také technologie DeviceNet, která nesplňuje výše uvedené normy, v průběhu času však byla používána.

- **Profibus:** univerzální systém uplatnitelný v různých odvětvích průmyslu. Založen na standardu IEC 61158 a IEC 61784. Dokáže rozdělovat členy na master/slave. Díky své popularitě poskytuje širokou podporu různých zařízení. Podporuje také různé metody přenosu dat, jako jsou: [3]

  - **RS 485**: vhodné pro automatizace, maximální počet stanic 32, přenosové rychlosti od 9,6 kb/s až po 12 000 kb/s. [19]

  - **MBP**: varianta vytvářejí různé topologie (bus, strom, hvězda). MBP poskytuje proudové připojení o velikostech 10–15 mA. Obsahuje navíc čipy, které dokáží ovlivnit distribuci elektrického proudu ve vytvořené síti, aby nedocházelo v určitých místech topologie k nedostatku proudu. Přenosová rychlost se pohybuje kolem 32 kb/s. [19]

  Profibus rozlišuje dvě úrovně masterů:
  - **Master první úrovně** (kontrolér): cyklicky si vyměňuje údaje s nižšími stanicemi sítě (slaves).
  - **Master druhé úrovně** (slave-slave komunikace): výměna dat probíhá o konfiguraci a stavu sítě. [20]



Obr. 2.15: Kabel PROFIBUS [18]

- **DeviceNet:** komunikační síť používaná hlavně mezi kontroléry a IO zařízeními (senzory, displeje, …). Principiálně funguje na technologii CAN a CIP. Podporuje víceúrovňovou komunikaci (master -> slave a slave -> slave). Technologie je standardizována normou IEC 62026-3. K propojení zařízení dochází v bus topologii. Celkově lze propojit 64 modulů. Na vzdálenost 500 m dosahuje DeviceNet rychlosti 125 kb/s, na vzdálenost 100 m 500 kb/s. [21] Každý člen vytvoří buď nepropojeného správce zpráv (UCMM), nebo Group 2-nepropojený port. Obě možnosti zajistí vytvoření vlastního identifikátoru. Při zvolení UCMM nebo Group 2 portu se následně ustálí komunikace. Tato komunikace je nadále používána pro posílání zpráv z jednoho členu na druhý. [3]
  V současné době se od této technologie ustupuje, jelikož se zcela nahrazuje Ethernetem, který umožňuje větší tok dat a rychlejší komunikaci v síti. DeviceNet zároveň vyžaduje externí software pro jeho konfiguraci a mapování hardwarových úkonů. [22]

2.7.4. Sítě řídících jednotek

Skupinu technologií rozdělíme z pohledu Ethernetu na:

- Lokální soft přístupy v reálném čase
- Hard přístupy v reálném čase
- Izochronní přístupy v reálném čase

Zmíněné technologie řadí do standardu IEC 61158. [3]

2.7.4.1. Lokální soft přístupy v reálném čase

Tato kategorie využívá TCP/IP mechanismy přes sdílené Ethernet sítě. Nadále se rozděluje podle různých funkcí TCP/IP a mechanismů aplikačního procesu. Tyto technologie nabízí reakční čas v řádech nižších milisekund. Zástupci této skupiny jsou:

- **MODBUS TCP/IP**: protokol na úrovni aplikační vrstvy pro komunikaci klienta/serveru s připojenými zařízeními přes různé sítě (reakce na příkaz). Jedná se o adaptaci Modbus technologie s možností navíc komunikovat přes Ethernet. Protokol je velmi využívaný v různých aplikacích (automatizace, systémy spravující elektrickou energii, komplexní systémy průmyslových procesních automatizací). TCP je používán jako transportní protokol pro zajištění správného pořadí Modbus zpráv. Umožňuje datovou segmentaci, přeposílání dat a samotné potvrzení. Komunikační protokol spoléhá na IP vrstvu v rámci adresování, směrování a doručování. Z IP adresy získává

informaci o odesílateli a příjemci. Posílaná data mají vždy strukturu začínající MBAP hlavičkou, funkčním kódem a datová pole. MBAP hlavička tvoří unikátní identifikátor, funkční kód definuje požadovanou akci. Datové pole obsahuje další podrobnější informace o požadavku. Modbus TCP/IP v automatizaci průmyslu funguje jako propojovací protokol mezi PLC kontroléry, senzory a terminály za vzniku větší propojené sítě. V automatizaci budov je využíván pro ovládání světelných/požárních systémů nebo také systému pohybu. V systémech spravující elektrickou energii propojuje senzory se softwary. [23]

Výhody	Nevýhody
Jednoduchost systému	Zastaralý bezpečností protokol
Spolehlivost	Nedostatek pokročilých funkcí
Snadná škálovatelnost sítě	Nepodporující nové technologie

Tabulka 2.9 – Výhody a nevýhody MODBUS TCP/IP

- **Ethernet/IP:** technologie kombinuje Ethernet (IEEE 802.3) s TCP/IP protokolem pro vznik průmyslových automatizačních technologií a zároveň umožňuje přístup k internetu. Je schopna vícero topologií (například hvězda, kruh). Kompatibilnost se standardem pro IEEE Ethernet umožňuje uživateli používat rozdílné rychlosti (10 Mb/s, 100 Mb/s, 1 Gb/s) a zároveň pružnou síťovou architekturu z široce dostupných měděných a optických kabelů. Při využití CIP protokolu lze technologii použít pro izochronní systémy reálného času i v automatizaci. [24]

Výhody	Nevýhody
Široká podpora	Nevhodná technologie pro pohybové aplikace
Vysoká přenosová rychlost dat	Obtížná konfigurace zařízení
Široké využití	Pro některé industriální aplikace je Ethernet/IP předimenzovaný

Tabulka 2.10 – Výhody a nevýhody Ethernet/IP [25]

- **High Speed Ethernet (HSE):** jedná se o odvozenou aplikační vrstvu z Fieldbus Foundation H1 (digitální obousměrný komunikační systém od FieldComm Group). V rámci odvětví automatizace je HSE považován za lepší alternativu vzhledem k jeho větší rychlosti páteřní sítě a zvětšené šířce pásma. Podporuje zároveň různé kategorie zařízení (hostitelské a provozní zařízení, I/O brány) [26]

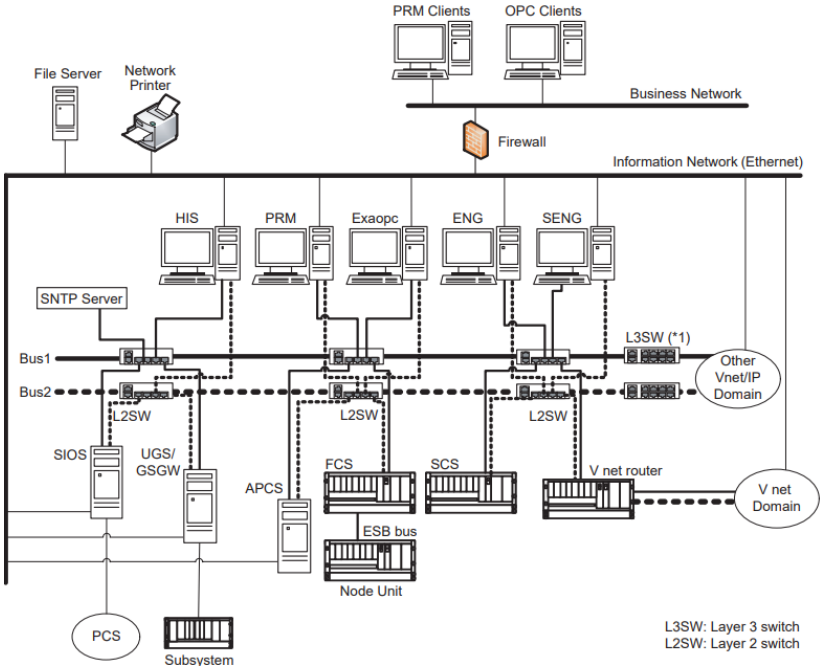
2.7.4.2. Hard přístupy v reálném čase:

Nad samotnou MAC vrstvou tyto přístupy zároveň využívají middleware (software) k ustanovení vyhlazovacích funkcí. Průmyslovými zástupci jsou:

- **PROFINET:** komunikační protokol založený na Ethernetu a jeho standardech. Ustanovuje cyklické a acyklické komunikace mezi jednotlivými členy o pevně stanovené časové délce od 250 µs do 512 ms. Profinet zajišťuje diagnostiku, funkční bezpečnost a systémové varování. Lze ho dále kombinovat i s dalšími Ethernetovými protokoly. Vzhledem k jeho spolehlivosti je používán i pro kritické aplikace. Během poruchy systém přechází do bezpečného stavu, kdy je schopný provádět alespoň minimálně potřebné funkce. [27]

- **Time-Critical Control Network (Tcnet, Toshiba):** Tcnet specifikuje v aplikační vrstvě společnou paměť pro časově kritické aplikace. Společná paměť je virtuálně sdílena mezi všemi členy. Toto řešení zajišťuje časovou a prostorovou soudržnost distribuce dat. Společná paměť je rozdělena do bloků o různých délkách, které jsou následně z jednoho místa rozesílaný mezi všechny zúčastněné členy sítě. Za obnovení bloků zodpovídá mechanismus cyklického vysílání. Společná paměť se proto skládá z určitých oblastí pro vysílaná data. Každý účastník má dostupný rychlý přístup k distribuovaným datům. Aplikační protokol se skládá ze tří podprotokolů (service protokol, application relationship protokol a data link mapping protokol). [3]

- **Vnet:** protokol založen na Ethernetu. Nabízí vysokou spolehlivost, ovládání v reálném čase a ochranu proti externím útokům na síť. Spolehlivost je zajištěna díky dvěma redundantním sběrnicím, které se skládají z nezávislých podsítí (sběrnice 1 a sběrnice 2). Řízení je obvykle zajištěno pomocí sběrnice 1, pokud ale dojde na dané síti k náhlé chybě, systém okamžitě přepne na sběrnici 2. Vnet obsahuje protokol zodpovědný za vysoce kvalitní komunikaci UDP protokolu. Funkce zabraňuje zpoždění přenosu a ztrátě informací tím, že omezuje hromadění zpráv. Odezva v reálném čase je umožněna prioritizací mezi komunikacemi. V rámci bezpečnosti Vnet provádí ověřování pomocí sdíleného tajného klíče, který se pravidelně aktualizuje. Mezi vyskytující se topologie patří hvězda, kruh a strom. Datová rychlost se pohybuje kolem 1 Gb/s. Jednotlivá data lze v rámci sítě rozdělit podle čtyř priorit: urgentní, důležitá, normální a při dostatku času. [28]



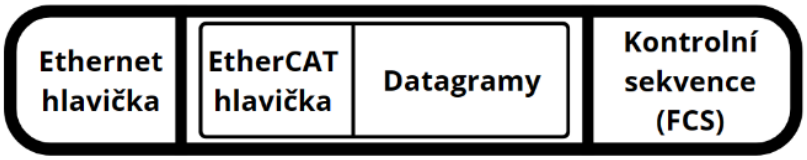
Obr. 2.16: Příklad systémové konfigurace Vnet [29]

2.7.4.3. Izochronní přístupy v reálném čase

- **Powerlink:** technologie kombinuje Ethernet, CANopen a systém reálného času. Spojuje až 240 členů. Jedná se čistě o software technologii, pokud uživatel vlastní jiný standardizovaný hardware, tak Powerlink se sofwarově doimplementuje. Powerlink spoléhá na mixed-polling a time-slot proceduru umožňující v danou chvíli přenášet data jen jednomu členu. Řídící člen stanovuje čas pro synchronizaci všech členů. Postupně se dotazuje každého členu pomocí zpráv PollRequest. Své data může propisovat pomocí funkce PollResponse. Tímto systém předchází vzniku kolizí. Celý cyklus se skládá ze tří částí. V první fázi se ustáli komunikace a zařízení se synchronizují. Následuje izochronní fáze a odesílání důležitých dat. Na závěr nastává asynchronní fáze, kde dojde k odesílání méně důležitých dat. (např. diagnostické a konfigurační data). [30]

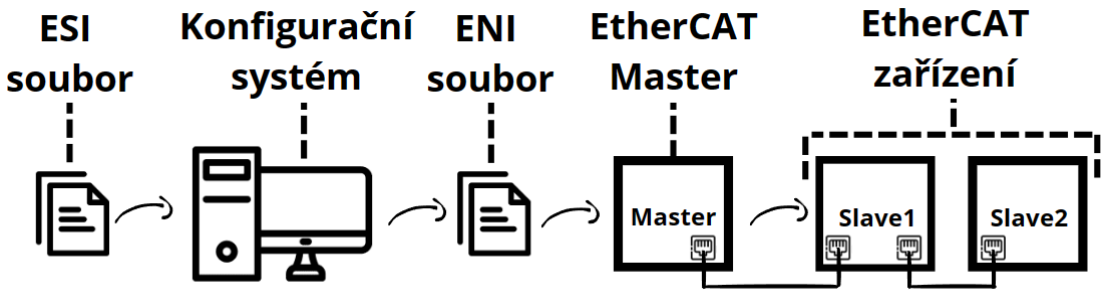
- **EtherCAT:** založen na technologiích Ethernetu a bus. Poskytuje spolehlivou a efektivní komunikaci pro celou řadu automatizací. Odpovídá standardu Ethernetu IEEE 802.3. Využívá master/slave hierarchii ke komunikaci mezi členy své sítě. Vlivem jeho standardizace a otevřenosti se jedná o cenově dostupnou technologii. EtherCAT navíc podporuje metodu distribuovaných hodin (DC), která umožňuje časovou synchronizaci celé sítě v rámci nanosekund. To umožňuje EtherCAT široké uplatnění v robotice pro zdravotnictví a také v automatizačním průmyslu.

EtherCAT pro komunikaci využívá funkci Ethernetu optimalizující šířku pásma a krátký procesní cyklus dat. Nevyužívá objemnějších UDP/IP nebo TCP/IP protokolů. Rámec zprávy obsahuje hlavičku Ethernetu, EtherCAT data a je zakončen kontrolní sekvencí rámce (FCS). Viz Obr. 2.17. EtherCAT data obsahují hlavičku definující celkovou délku a typ následujících datagramů, které obsahují skutečná data. Celý rámec může obsahovat až 1 498 bitů. Pokud dojde k překročení limitu, master vyšle větší počet rámců dat, které budou přijímací člen informovat o dalších pokračujících rámcích. EtherCAT zprávy zpracovává „za letu“ (on the fly). To umožňuje odeslání pouze jednoho rámce ke všem členům sítě. Master sestaví rámec a odešle jej. Daný rámec putuje po sítí, projde všemi členy a vrátí se. Jak rámec putuje, jednotlivá zařízení rámec čtou, pokud se v rámci nachází některá data s jejich adresou, data vyjmou a vloží data nová.



Obr. 2.17: Rámec EtherCAT zprávy

EtherAT je schopen různých topologií. Každé zařízení slave obsahuje minimálně dva další připojovací porty. Výsledná síť může např. zaujímat topologii hvězdy, kruhu nebo stromu. EtherCAT ovládá full-duplex komunikaci. Systémová architektura je zobrazena na Obr. 2.18.



Obr. 2.18: Systémová konfigurace EtherCAT

ESI soubor – popisuje funkcionalitu a konfiguraci slave zařízení, nezbytná pro správné utvoření komunikace

ENI soubor – popisuje topologii dané sítě, vytvářeno EtherCAT systémem

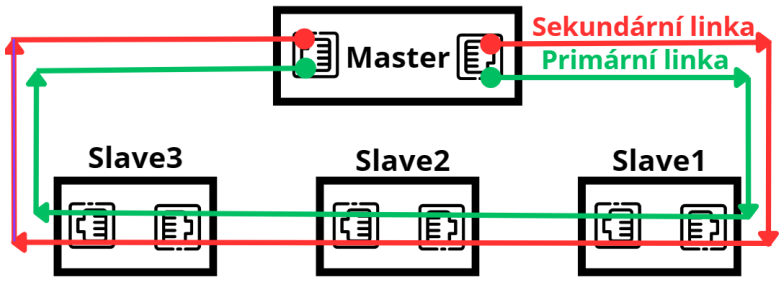
Master – PC/mikroprocesor/PLC zodpovědný za správnou komunikaci s dalšími zařízeními a synchronizaci.

Zařízení/slave – senzory, aktuátory apod. [31], [32]

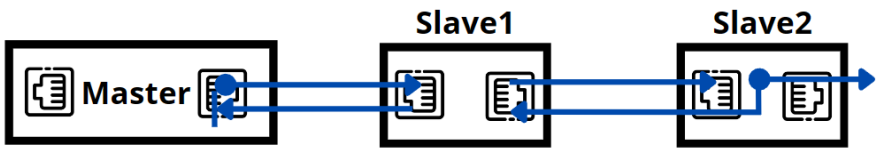
- **PROFINET IO:** technologie využívá plně duplexního Ethernetu o rychlosti 100 Mb/s. Profinet IO přidává ke kanálům reálného času navíc izochronní kanál, který umožňuje výměnu cyklických dat. Další mechanismy pro synchronizaci času a plánování jsou umístěny v rámci a nad MAC vrstvou Ethernetu. Šířka pásma je pro cyklický provoz oddělena. V rámci jednoho cyklu existují časově oddělené oblasti pro cyklický hard reálný čas, soft reálný čas a pro synchronizaci mechanismu. Čas jednoho cyklu se obvykle pohybuje kolem 250 µs až 1 ms. Při použití optických vláken dokáže Profinet IO komunikovat až na dálku 3 km. [3]

- **Ethernet/IP:** rozšíření Ethernet/IP využívá CIP synchronizační protokol umožňující přenos izochronních dat. Při použití CIP protokolu lze dosáhnout synchronizace času až s přesností na 500 ns.

- **SERCOS III:** síť na úrovni master/slave vytvořena za účelem ovládání pohybu. Zařízení slave obsahují opakovače s konstantním zpožděním. Z hlediska topologie se převážně jedná o kruh nebo bus. Jednotlivé členy, kde každý má nezaměnitelné dva komunikační porty, jsou vzájemně propojeny přenosovou linkou. Pro kruhovou topologii platí, že je tvořena dvěma kanály. Všechny členy posílají zprávu dál před sebe, viz Obr. 2.19. Bus topologie využívá primárního nebo sekundárního kanálu. Poslední člen tvoří zpětnou smyčku, všechny ostatní posílají pouze vpřed, viz Obr. 2.20. [33]



Obr. 2.19: Sercos III kruhová topologie



Obr. 2.20: Sercos III bus topologie

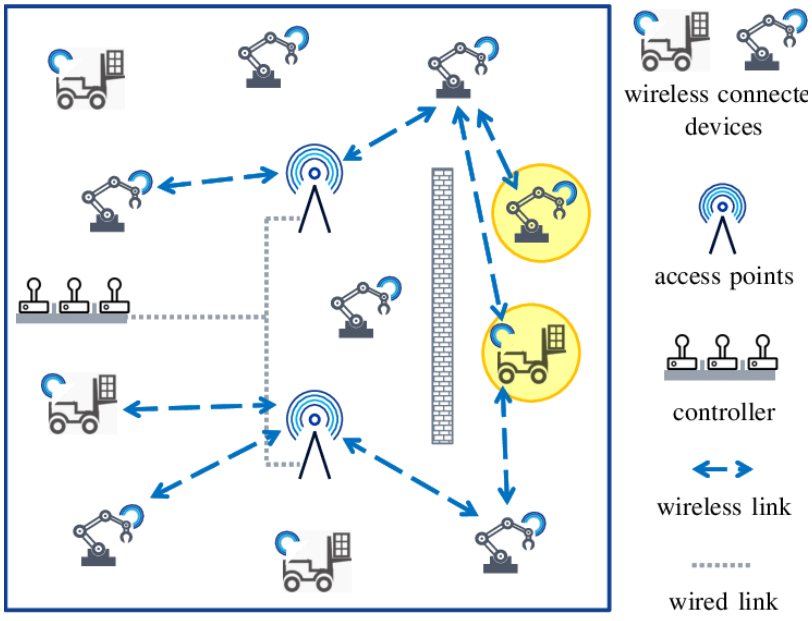
2.8. Průmyslové bezdrátové sítě

2.8.1. Úvod

Jakákoliv komunikace mezi technologiemi je digitální reprezentací analogového průběhu zatížena o různá zpoždění a odchylky. Realizovaná drátová komunikace vykazuje nižší citlivost na rušení ve srovnání s bezdrátovou komunikací. Odesíláním dat na sdílené síti se problém stává mnohem složitějším. V takovém případě je třeba řešit komunikační protokoly, mechanismy MAC a přenosové rychlosti jednotlivých signálů. V praxi se proto používají protokolové sítě, které se snaží dosáhnout relativně konstantního zpoždění za využití různých povoleních k regulaci přístupu do sítě.

Bezdrátové sítě se sítím drátovým dokáží výkonově vyrovnat, avšak jsou častokrát odkázány na neoptimální podmínky prostředí. Průmyslové zařízení se vyskytují na malých vzdálenostech. Takový stav vede k zvýšenému výskytu ruchů a vytváření zpoždění bezdrátové komunikace v porovnání s kabelovou komunikací. Bezdrátové protokoly s regulací přístupu do sítě se ukazují jako vhodnou alternativou pro řídící systémy, jelikož omezují rušení v sítí a zároveň poskytují rozumnou přenosovou rychlost. Mezi základní standardy bezdrátové komunikace patří technologie: [34]

- **Mobilní komunikace** – GSM, GPRS, UMTS
- **Standardy nižších vrstev** (IEEE 802.11 a 802.15) – WLAN, personální sítě (sloužící primárně jako základ lokálních rádiových sítí)
- **Standardy vyšších vrstev** (aplikační vrstvy nad IEEE 802.11 a 802.15.4) – WiFi, Bluetooth, bezdrátový HART, ZigBee
- **Rádiové technologie** – WiSA, LoRa



Obr. 2.21: Schéma bezdrátové sítě pro průmyslové bezdrátové řízení [35]

2.8.2. Moderní bezdrátové sítě v průmyslu

- **WLAN/WiFi:** bezdrátová verze Ethernetu. Standard IEEE802.11 specifikuje, že se jedná o infračervenou, přímo sekvenčně rozprostřenou (DSSS) a frekvenčně přeskakující (FHSS) fyzickou vrstvu. Využívá speciální frekvenční pásma, která sdílí médium s ostatními uživateli. V historii byla založena WiFi Alliance za účelem zajištění kompatibility mezi různými klienty WLAN. Sítě používají bezlicenční frekvenční pásmo. Tato technologie je implementovaná nejen v osobních počítačích, ale i ve vestavěných systémech po celém světě. WLAN umožňuje bezdrátový přístup k sítím LAN fungujících na Ethernetu. Technologie nabízí vysokorychlostní přenos dat, umožňuje větší flexibilitu a je cenově dostupná. Maximální povolená síla přenosu nesmí překročit 100 mW. Maximální dosah takové sítě se uvádí jako 100 m (při použití směrových antén i více). Využívá rádiových frekvencí v oblasti 2,4 GHz a 5 GHz. Spotřeba energie není optimalizována. Umožňuje vytvářet vysoký počet nepřekrývajících se kanálů. To usnadňuje konfiguraci sítě a je možné vytvořit i více přístupových bodů, u kterých nedochází k vzájemnému rušení. [36],[37]

  WLAN/WiFi sítě se ukazují jako užitečné pro monitorování (strojů/systémů), avšak kvůli jejich nedostatečné rychlosti, velké latenci a omezené stabilitě připojení se omezuje jejich použití v kritických aplikací řízení strojů. Konkrétně se technologie používá například pro: čtečky čárových kódů, snímače pohybu, sledování stavu strojů (senzory teploty, tlaku, vlhkosti, …) [38]

- **WirelessHART:** síť založena na rádiovém přenosu o frekvenci 2,4 GHz vhodná pro topologie mesh, hvězdu, strom. Obsahuje Time synchronized mesh protocol (TSMP), který umožňuje jednotlivým členům časovou synchronizaci a stanovuje konkrétní čas pro vysílání. WirelessHART využívá různých kanálů pro komunikaci. Oproti ZigBee jsou konečná zařízení sítě schopna společné komunikace o unikátním síťovém identifikátoru. Vícero sítí může vzájemně koexistovat, i když se nachází na stejné rádiové frekvenci. Každý člen si vede vlastní komunikační seznam. V rámci bezpečnosti jsou zprávy šifrovány 128bitovým mechanismem AES. [3], [39]

- **ZigBee:** nad základní standard IEEE 802.15 přidává další sítě a aplikační vrstvy umožňující tvořit sítě nízké spotřeby o menších rychlostí. ZigBee síť je schopna tvořit různé topologie a je vhodná pro nenáročné aplikace. V Evropě používá DSSS radiový signál v rozmezí 868 MHz a také 2,4 GHz v ISM pásmu. Maximální rychlost přenosu dat je 250 Kb/s. K vyhýbání se kolizím signálů síť využívá algoritmus podobný jako je u standardu 802.11, kde každé zařízení poslouchá daný kanál před tím, než začne posílat vlastní signál. U ZigBee nedochází ke změnám vysílacího kanálu. V případě signálové kolize například s velmi využívaným WiFi sítí může dojít k zhoršení komunikace. [37], [39]

  Obecně rozlišuje mezi třemi typy ZigBee zařízení:

  1) **Koordinátor** uschovává informaci o síti a bezpečnostní klíče. Je zodpovědný za propojení dalších zařízení do sítě.
  2) **Router** rozesílá data dalším členům sítě.
  3) **Konečné zařízení**, posílá data routeru/koordinátorovi. [3]

- **Bluetooth:** pracuje v ISM pásmu 2,4 GHz, při vysílání dochází ke změnám kanálu, není tím pádem náchylné na rušení. Výkonově se pohybuje od 1 mW do 100 mW. Bluetooth je schopno různých topologií. Od nejnovější generace (Bluetooth 5.0 a výš) umožňuje přenosovou rychlost 2 Mb/s, zároveň zlepšuje vzájemnou toleranci s ostatními 2,4 GHz sítěmi a posouvá maximální komunikační vzdálenost až na 240 metrů (při menších přenosových rychlostech 125/500 Kb/s). To hlavně díky FEC (Forward Error Correction). Tato funkce umožňuje přijímači opravit chyby na přijímaných datech, které například vznikly interferencí s jinými signály. Bluetooth operuje ve třech stavech (inzerování, snímání, propojení). Inzerování dat probíhá v nynější verzi až o velikostech 255 bitů na jeden paket. To dovoluje posílá dat více zařízením najednou. Verze 5.2 obsahuje LEPC (LE power control), tato technologie optimalizuje vysílací signál na nejnižší možný výkon při stále stabilní komunikaci a tím omezuje velikost spotřeby. Oproti ZigBee umožňuje vetší datové přenosy na úkor vetší spotřeby. S verzí Bluetooth 5.3 přišlo zlepšení optimalizace spotřeby energie spolehlivosti propojení a také novější bezpečnostní standardy. Nejnovější verze 5.4 nabízí kromě dalších výkonových a bezpečnostních vylepšeních i funkci PAwR (Periodic Advertising with Response). Tato nová funkce umožní propojení jednoho vysílacího zařízení s mnoha přijímači. [40], [41],[42]

  Existuje Bluetooth Low Energy (BLE) protokol pro snižování spotřeby a zvýšení komunikační vzdálenosti. Momentální verze BLE nepodporuje ale data v reálném čase a je pouze limitována na topologii do hvězdy. [43]

- **LoRa/LoRaWAN**: rádiová technologie schopná komunikovat na rozsáhlé vzdálenosti při nízké spotřebě a nízké přenosové rychlosti. Své místo nachází pro aplikace se senzory s omezenou baterií. Senzorové členy odesílají data koncovému členu, který data odesílá dále. Při ideálních podmínkách je technologie schopna odesílat data až na vyšší jednotky kilometrů. LoRa modulace využívá techniky rozprostřeného spektra a také chirp spektra s dopřednou chybovou korekcí.
  LoRa pracuje v nižších pásmech ISM rádiových vln (v Evropě o hodnotách 433 MHz a 868 MHz). Lze využít různých topologií.
  LoRaWAN je standardizovaný MAC protokol schopný pouze topologie do hvězdy. Jedná se o otevřený standard řízený organizací LoRa Alliance. Dosahuje přenosových rychlostí od 0,3 Kb/s do 50 Kb/s vhodných pro senzorové uplatnění v odvětvích automatizace. Rozlišuje tři druhy zařízení:
  - **Třídy A**: bateriově nabíjené zařízení, mají nejnižší spotřebu na úkor největší latence. Převážně se jedná o nenáročné senzory.
  - **Třídy B**: bateriově nabíjené zařízení, mají nízkou spotřebu, avšak nyní je důležité časté provedení komunikace. Například aktuátory/senzory.
  - **Třídy C**: obousměrně komunikující zařízení připojené k externímu zdroji energie. Většinu svého času přijímají vysílané signály, až na okamžiky, kdy samo vysílá.

  O bezpečnost se stará protokol na bázi IEEE 802.15.4, který využívá dva bezpečnostní klíče. Každý o 128 bitech. [44]

2.8.3. Porovnání LoRa, ZigBee, Bluetooth BLE, Wi-Fi

Parametr	LoRa	ZigBee	Bluetooth BLE	Wi-Fi
Přenosová rychlost [Kb/s]	50 – 300	20 – 250	125 – 2 000	54 000 – 1 300 000
Přenosová vzdálenost [m]	2 000 – 20 000	2 000	10 – 1 500	15 – 100
Frekvenční pásmo [MHz]	434/868	2 400	2 400	2 400/5 000
Maximum použité energie [mW]	2	500	1	1
Spotřeba energie	nízká	nízká	střední	střední

Tabulka 2.11 – Srovnání vybraných parametrů LoRa, ZigBee, Bluetooth a Wi-Fi technologií [45], [46]

3. Realizace LoRa mesh sítě

K vytvoření funkční mesh sítě je použita LoRa technologii s topologií mesh. K implementaci je využito vývojové prostředí Arduino IDE a mikrokontrolery Arduino NANO.

3.1. Arduino – obecné informace

Arduino umožňuje používání open source hardwaru a softwaru a je vhodné pro rychlý vývoj prototypových aplikací.

V roce 2003 v Itálii bylo Arduino založeno skupinou akademiků se záměrem přiblížit studentům svět automatizace. Díky nízkým pořizovacím nákladům se Arduino stalo velmi preferovanou možností mezi studenty a domácími kutily po celém světě. Silná stránka Arduina je v široké kompatibilnosti s mnoha operačními systémy (Windows, macOS, Linux). Oproti běžným PLC kontrolérům ztrácí však na celkovém výpočetním výkonu. [47]

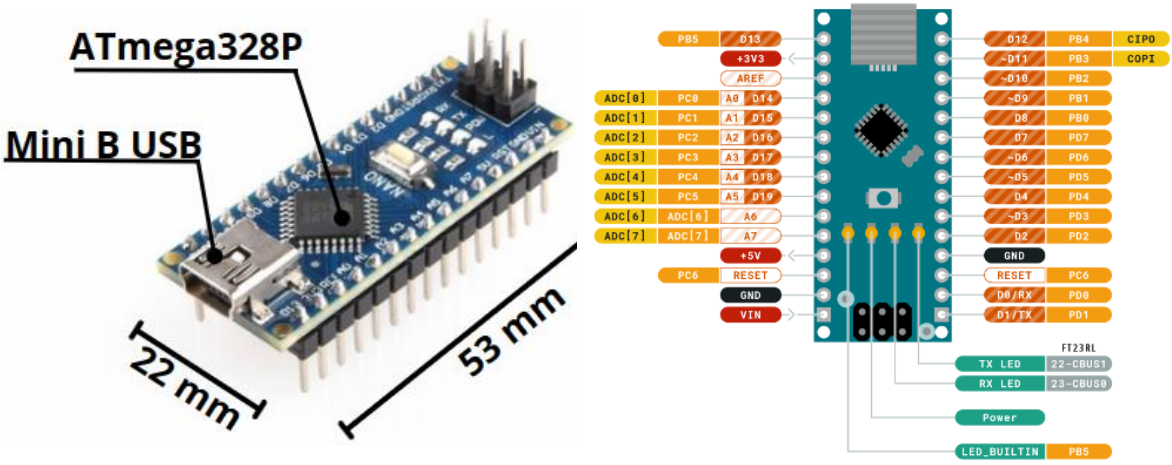
Po hardwarové stránce Arduino dělí své kontroléry do 3 skupin:

- **Nano (SAMD21):** kompaktní, levné desky s nízkou spotřebou elektrické energie postavené na ARM procesorech s možností bezdrátového připojení skrze Bluetooth a WiFi. Pracují s napěťovou úrovni 3,3V.
- **MKR:** kontroléry vybudované na ARM procesorech a slouží převážně jako rozšiřující moduly o nové funkce pro další Arduino desky. Například modul pro GPS, Ethernet nebo měření teploty.
- **Classic:** desky s procesory ATmega o větších velikostech než Nano.

Všechny Arduino platformy je možné programovat v integrovanému vývojovému prostředí Arduino IDE. Programovací jazyk je založen na C a C++.[48]

3.2. Použité mikroprocesory Arduino

V rámci minimalizace vstupních nákladů se pro realizaci použily klony vývojové desky typu Arduino Nano 3.0, které se na českých internetových obchodech cenově pohybují kolem 200 Kč za kus. Deska obsahuje procesor ATmega328P, řadí se do procesorové skupiny Classic. Max frekvence CPU je 20 MHz. Deska má 32 KB Flash, 2 KB SRAM a 1 KB EEPROM. Na bočních stranách desky lze nalézt 14 digitálních I/O pinů, 8 analogových a další. Vstupní napětí o 5 V umožňuje napájet podobné zařízení 5 V nebo 3,3 V. K připojení k počítači je použito Mini B USB. Deska je schopna jednoduché sériové komunikace na digitálních pinech 0 (RX) a 1 (TX). [50]



Obr. 3.1: Arduino Nano R3 s připájenými piny a pinový diagram [49], [51]

3.3. Použité moduly pro rádiovou komunikaci

Pro bezdrátovou komunikaci byl vybrán radiový modul Ra-02 s adaptérem, který obsahuje vysílač Semtech SX1278 umožňující LoRa komunikaci. Dle výrobce dosahuje citlivosti signálu až kolem -148 dBm. Modul je vyroben firmou AI-Thinker. Pracovní frekvence se pohybuje mezi 420–450 MHz. Modul je schopen vícero modulací (FSK, GFSK, MSK, LoRa). Výkonově se vysílač pohybuje kolem +20 dBm a přenosová rychlost dat je menší než 300 kb/s. Potřebný proud pro: režim ve spánku (0,2 µA), režim přijímaní (max. 10,8 mA), režim vysílání (max. 120 mA). Dle výrobce je modul schopen dosáhnout komunikace až na vzdálenost 10 km při otevřeném terénu a při využití směrových antén. U českých internetových prodejců se momentální cena za kus pohybuje kolem 200 Kč. [52]

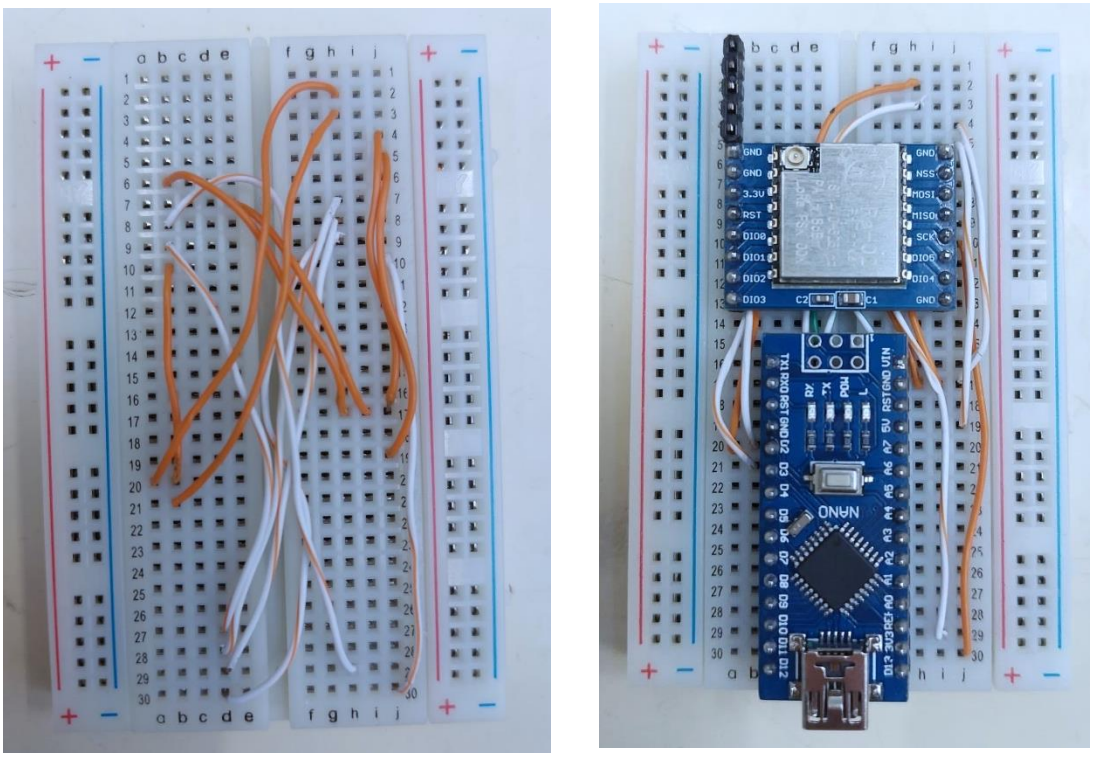
Obr. 3.2: Bezdrátový komunikační modul 433 MHz SX1278 LoRa RA-02 [53]

3.4. Propojení Arduino Nano 3.0 s rádiovým modulem RA-02

Jednotlivé části systému byly vzájemně propojeny s využitím nepájivého pole, viz Obr. 3.3. Jednotlivé spojené piny jsou:

Nano	RA-02
3,3 V	3,3 V
GND	GND
D9	RST
D2	DIO0
D10	NSS
D11	MOSI
D12	MISO
D13	SCK

Tabulka 3.1 – Propojené piny z Arduino Nano 3.0 k modulu RA-02



Obr. 3.3: Vytvořené kabelové propojení na nepájivém poli bez modulů a s moduly

Propojení modulu Arduina a modulu LoRa Ra-02 je realizováno pomocí sběrnice SPI s full-duplex spojením. Spojení je vytvořeno prostřednictvím 4 signálových vodičů. Prvním je Slave Select (SS), druhým Serial Clock (SCLK), třetím Master Out Slave In (MOSI), čtvrtým Master In Slave Out (MISO). SS linkou zahájí master komunikaci. Poté zahájí operaci na SCLK generováním hodinového signálu. V každém cyklu master pošle a zároveň obdrží 1 bit. Po odeslání dat komunikace dále probíhá, nebo je ukončena přerušením SS signálu. [54]

3.5. Kód LoRa komunikace v Arduino IDE

K samotnému fungování kódu je využito dodatečných externích knihoven. V komunikační sítí se vyskytuje vícero členů, každý s vlastním unikátním číslem (ID). Číslo/ID uložíme do semi-permanentní paměti EEPROM.

Je využito podknihoven patřící do skupiny RadioHead, která se používá pro různé implementace rádiových zařízení s mikrokontrolery. Knihovna RHRouter se zaměřuje na budování bezdrátových mesh sítí s funkcemi routování. RHMesh rozšiřuje funkce o automatické routování, adresování a spolehlivost sítě. RH_RF95 je poslední podknihovna vytvořená pro modul SX1278 umožňující nastavovat parametry komunikace. [55]

```
2   #include <EEPROM.h>
3   #include <RHRouter.h>
4   #include <RHMesh.h>
5   #include <RH_RF95.h>
```

Obr. 3.4: Použité externí knihovny v kódu LoRa komunikace

Jsou deklarovány základní proměnné N_NODES, nodeId, routes a rssi a jejich datové typy. Uint8_t umožňuje ukládat 8bitový integer, hodnoty od 0 do 255. Naproti tomu int16_t je 16bitový integer uchovávající hodnoty od -32 768 do 32 767. [56] Deklarováním N_NODES je určena velikost sítě. Proměnná nodeId pracuje s číslem nodu. Routes[N_NODES] a rssi[N_NODES] je pole hodnot s informacemi o routování a síle signálu.

```
9   #define N_NODES 4
10  uint8_t nodeId;
11  uint8_t routes[N_NODES];
12  int16_t rssi[N_NODES];
```

Obr. 3.5: Deklarované proměnné v kódu LoRa komunikace

Následně jsou vytvořený dva objekty. Zaprvé RH_RF95, to je třída rádiového ovladače na modulu rf95. Zadruhé ukazatel manager pro jednodušší manipulaci s objektem RHMesh. Řádek 17 na Obr. 3.6 inicializuje pole znaků o maximální velikosti zprávy.

```
15  RH_RF95 rf95;
16  RHMesh *manager;
17  char buf[RH_MESH_MAX_MESSAGE_LEN];
```

Obr. 3.6: Deklarované proměnné a objekty v kódu LoRa komunikace

V setupové části kódu, se řádkem 52 definuje frekvence sériové komunikace. Na řádku 53 se uloží do nodeId unikátní číslo z EEPROM paměti. Dále na řádku 60 dojde k vytvoření nové instance objektu RHMesh s konkrétními ovladači a s id nodu. Kód na řádku 66 udává vysílací sílu signálu v db, kde je nutné respektovat místně příslušná legislativní omezení na sílu vysílajícího signálu. Řádek 67 určuje operační frekvenci v MHz. Poslední řádek na Obr. 3.7 specifikuje, jak dlouho bude modul čekat před tím, než začne vyhodnocovat zaplnění kanálu a případně začne vysílat.

```
52    Serial.begin(115200);
53    nodeId = EEPROM.read(0);
54    if (nodeId > 10) {
55      Serial.print(F("EEPROM nodeId invalid: "));
56
57      Serial.println(nodeId);
58      nodeId = 1;
59    }
60    manager = new RHMesh(rf95, nodeId);
61    if (!manager->init()) {
62      Serial.println(F("init failed"));
63    } else {
64      Serial.println("done");
65    }
66    rf95.setTxPower(10, false);
67    rf95.setFrequency(433.0);
68    rf95.setCADTimeout(500);
```

Obr. 3.7: Setup kód LoRa komunikace

Dále v programu jsou definovány jednotlivé funkce. Funkce printNodeInfo (uint8_t node, char*s), zobrazí do sériového monitoru diagnostickou zprávu příslušného nodu.

```
139   void printNodeInfo(uint8_t node, char *s) {
140     Serial.print(F("node: "));
141     Serial.print(F("{"));
142     Serial.print(F("\""));
143     Serial.print(node);
144     Serial.print(F("\""));
145     Serial.print(F(": "));
146     Serial.print(s);
147     Serial.println(F("}"));
148   }
```

Obr. 3.8: Funkce printNodeInfo

Funkce updateRoutingTable() zodpovídá za aktualizaci routovací tabulky a za obdržení signálu každého nodu (pole routes a rssi). Deklarace začne for smyčkou, která iteruje přes každý nod v síti. Na řádku 108 dojde k aktualizaci informace do routovací tabulky pro momentální nod za pomocí funkce getRouteTo(). Je-li shoda čísla ID nodu s číslem iterace, nod získá identifikátor s hodnotou 255, tímto je zaručena samoidentifikace. Pokud první podmínka neplatí, je nodu přiděleno číslo příslušné číslu aktuální iterace. Poslední podmínka testuje existenci cesty k určitému nodu n. Pokud cesta neexistuje uloží hodnotu signálu rovnu 0, viz Obr. 3.9.

```
106  void updateRoutingTable() {
107    for(uint8_t n=1;n<=N_NODES;n++) {
108      RHRouter::RoutingTableEntry *route = manager->getRouteTo(n);
109      if (n == nodeId) {
110        routes[n-1] = 255; // self
111      } else {
112        routes[n-1] = route->next_hop;
113        if (routes[n-1] == 0) {
114          // if we have no route to the node, reset the received signa
115          rssi[n-1] = 0;
116        }
```

Obr. 3.9: Funkce updateRoutingTable

Funkce GetRouteInfoString vrátí proměnou typu string obsahující zprávu o cestě ke každému nodu. For cyklus následně projde všechny nody a přiloží do zprávy jejich číslo n a jejich sílu signálu rssi.

```
122  void getRouteInfoString(char *p, size_t len) {
123    p[0] = '\0';
124    strcat(p, "[");
125    for(uint8_t n=1;n<=N_NODES;n++) {
126      strcat(p, "{\"n\":");
127      sprintf(p+strlen(p), "%d", routes[n-1]);
128      strcat(p, ",");
129      strcat(p, "\"r\":");
130      sprintf(p+strlen(p), "%d", rssi[n-1]);
131      strcat(p, "}");
132      if (n<N_NODES) {
133        strcat(p, ",");
134      }
135    }
136    strcat(p, "]");
137  }
```

Obr. 3.10: Funkce getRouteInfoString

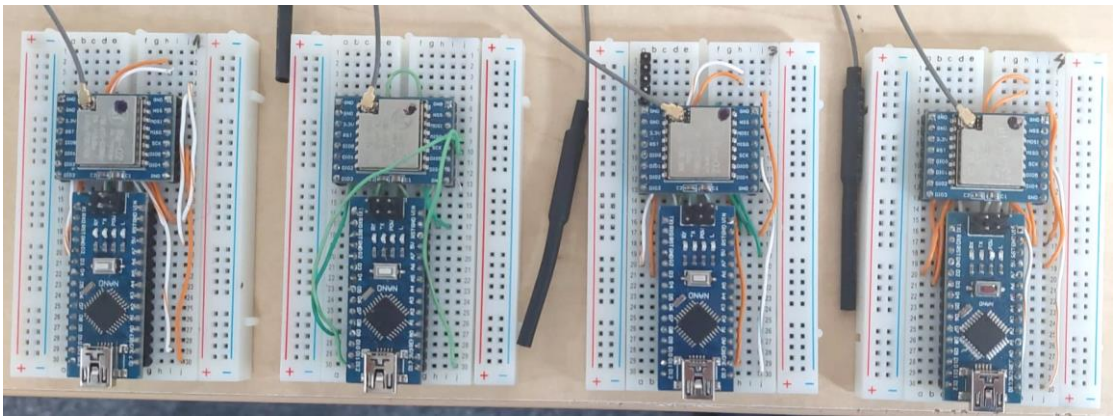
Kód dále v nekonečné smyčce posílá, přijímá zprávy a zajišťuje chod sítě. Kód na Obr. 3.11 postupně proiteruje přes veškeré nody mesh sítě voláním funkce updateRoutingTable(), tak dochází k postupné aktualizaci routovací tabulky. Voláním funkce getRouteInfoString() dochází k přeposílaní jednotlivých zpráv. Kód Manager->sendtoWait() je zodpovědný za odeslání zprávy cíli. Pokud zpráva byla úspěšně odeslána, program získá aktualizovanou hodnotu rssi signálu. Následně je v intervalu 3-5 s provedeno přijímání zpráv od nodů v aktuálním signálovém dosahu. Pokud nod obdrží zprávu od jakéhokoliv dalšího nodu v síti, přepíše aktuální hodnotu signálu rssi novou hodnotou. Při dokončení, se celý cyklus identifikace sítě opakuje, je tak možné sledovat dynamickou změnu topologie sítě.

```
151   for(uint8_t n=1;n<=N_NODES;n++) {
152     if (n == nodeId) continue; // self
153     updateRoutingTable();
154     getRouteInfoString(buf, RH_MESH_MAX_MESSAGE_LEN);
155     Serial.print(F("->"));
156     Serial.print(n);
157     Serial.print(F(" :"));
158     Serial.print(buf);// send an acknowledged message to the target node
159     uint8_t error = manager->sendtoWait((uint8_t *)buf, strlen(buf), n);
160     if (error != RH_ROUTER_ERROR_NONE) {
161       Serial.println();
162       Serial.print(F(" ! "));
163       Serial.println(getErrorString(error));
164     } else {
165       Serial.println(F(" OK"));
166       // we received an acknowledgement from the next hop for the node we tri
167       RHRouter::RoutingTableEntry *route = manager->getRouteTo(n);
168       if (route->next_hop != 0) {
169         rssi[route->next_hop-1] = rf95.lastRssi();}}
170     if (nodeId == 1) printNodeInfo(nodeId, buf); // debugging   // listen for
171     unsigned long nextTransmit = millis() + random(3000, 5000);
172     while (nextTransmit > millis()) {
173       int waitTime = nextTransmit - millis();
174       uint8_t len = sizeof(buf);
175       uint8_t from;
176       if (manager->recvfromAckTimeout((uint8_t *)buf, &len, waitTime, &from))
177         buf[len] = '\0'; // null terminate string
178         Serial.print(from);
179         Serial.print(F("->"));
180         Serial.print(F(" :"));
181         Serial.println(buf);
182         if (nodeId == 1) printNodeInfo(from, buf); // debugging // we receive
183         RHRouter::RoutingTableEntry *route = manager->getRouteTo(from);
184         if (route->next_hop != 0) {
185           rssi[route->next_hop-1] = rf95.lastRssi(); }
186       }
```

Obr. 3.11: Loop kód LoRa komunikace

3.6. Ověření funkčnosti realizované LoRa mesh sítě

Pro ověření správného fungování sítě, je proveden jednoduchý pokus o čtyřech komunikačních jednotkách. Na každém Arduinu Nano je nahraný stejný kód (viz kapitola 3.5). Napájené moduly jsou následně rozmístěny v různých vzdálenostech.



Obr. 3.12: Čtyři sestavené komunikační jednotky pro ověření funkčnosti sítě

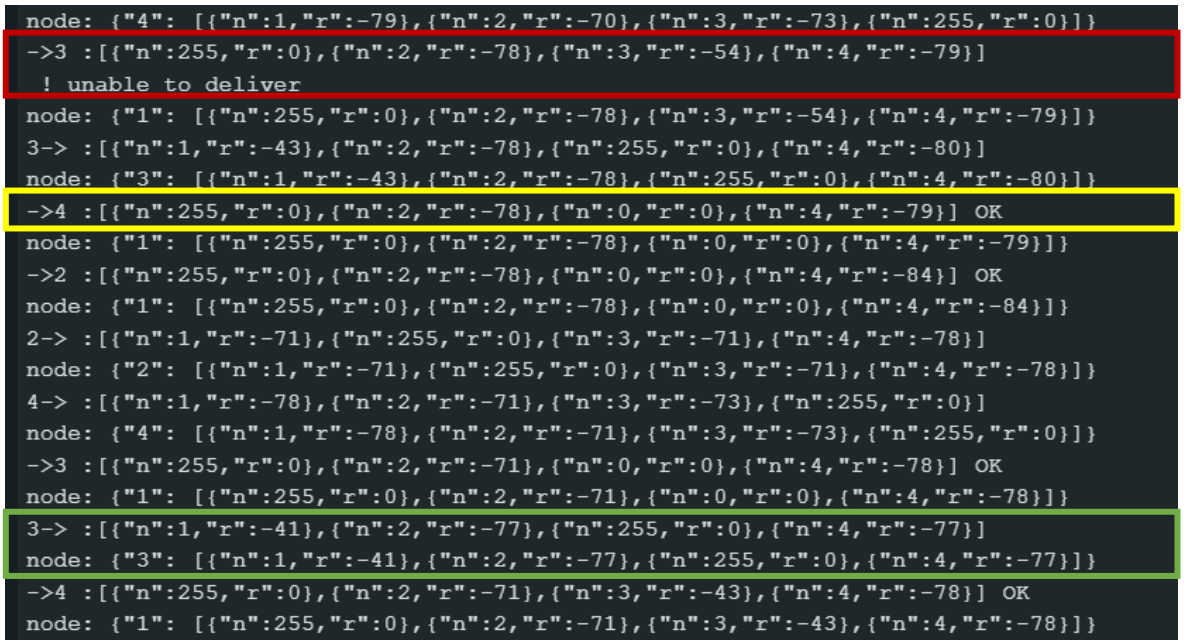
Při zapnutí Serial monitoru v Arduino IDE členu 1 systém obdržel následující zprávy.

```
node: {"1": [{"n":255,"r":0},{"n":2,"r":-79},{"n":3,"r":-43},{"n":4,"r":-79}]}
->4 :[{"n":255,"r":0},{"n":2,"r":-79},{"n":3,"r":-54},{"n":4,"r":-79}] OK
node: {"1": [{"n":255,"r":0},{"n":2,"r":-79},{"n":3,"r":-54},{"n":4,"r":-79}]}
2-> :[{"n":1,"r":-71},{"n":255,"r":0},{"n":3,"r":-71},{"n":4,"r":-79}]
node: {"2": [{"n":1,"r":-71},{"n":255,"r":0},{"n":3,"r":-71},{"n":4,"r":-79}]}
->2 :[{"n":255,"r":0},{"n":2,"r":-71},{"n":3,"r":-54},{"n":4,"r":-86}] OK
node: {"1": [{"n":255,"r":0},{"n":2,"r":-71},{"n":3,"r":-54},{"n":4,"r":-86}]}
4-> :[{"n":1,"r":-79},{"n":2,"r":-70},{"n":3,"r":-73},{"n":255,"r":0}]
node: {"4": [{"n":1,"r":-79},{"n":2,"r":-70},{"n":3,"r":-73},{"n":255,"r":0}]}
```

Obr. 3.13: Odesílané a přijímané zprávy z pohledu nodu 1 v prostředí Arduino IDE

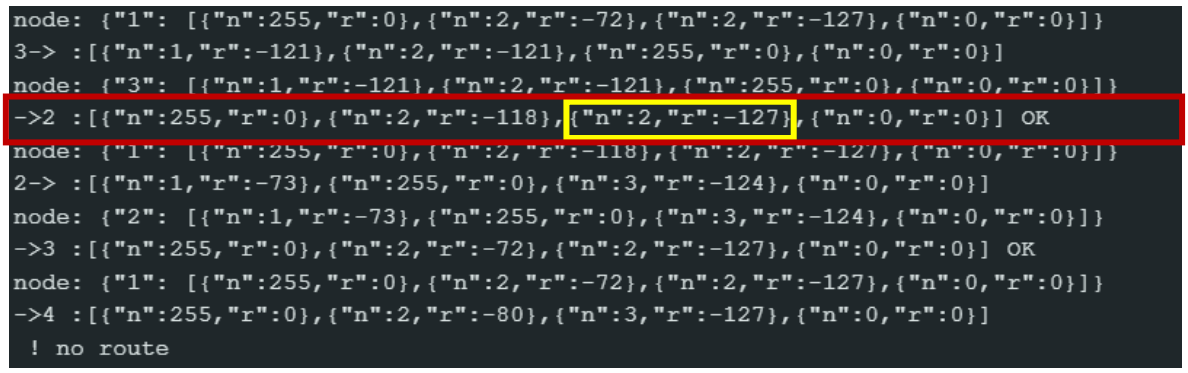
Síť si dynamicky rozesílá zprávy o nodech účastnících se komunikace a o jejich hodnotě signálu. Zpráva na druhém řádku (viz Obr. 3.13 červený obdélník), udává, že člen 1 navazuje komunikaci se členem 4 (posílá mu zprávu). První složená závorka v sobě uchovává informaci o členu samotném, proto n = 255 a síla signálu r = 0. Dále posílá informaci o členu 2 se sílou signálu -79 dBm, o členu 3 se sílou signálu -54 dBm a o členu 4 se sílou signálu -79 dBm. Na konci řádku vidíme OK. Zpráva byla úspěšně přijata členem 4.

Dále nastane jev neúspěšné komunikace. Nod 1 se snažil odeslat zprávu nodu 3 (viz červený řádek Obr. 3.14). Obdržena byla proto chybová zpráva (unable to deliver). Nod 1 rozesílal v dalších zprávách informaci o nemožnosti navázání komunikace s nodem 3 (viz žlutý řádek Obr. 3.14). O pár řádku níže je však komunikace znovu navázána (viz zelený řádek Obr. 3.14) a síť je opět zcela propojená.



Obr. 3.14: Přerušení a obnovení komunikace z pohledu nodu 1 v prostředí Arduino IDE

Pro ověření schopnosti nodů komunikovat nepřímo je provedeno následující. Nody 1, 2 a 3 jsou rozmístěny tak, aby nedošlo k přímé komunikaci mezi nody 1 a 3. To vše při stále zapnutém Serial monitoru nodu 1. Po určité době nastane nepřímá komunikace (viz červený řádek Obr. 3.15). Nod 1 není schopen přímé komunikace s nodem 3, tudíž používá nod 2 pro předání zprávy. Tento jev je znázorněn jiným číslem nodu ve zprávě, než je jeho pořadí (viz žlutý řádek Obr. 3.15).



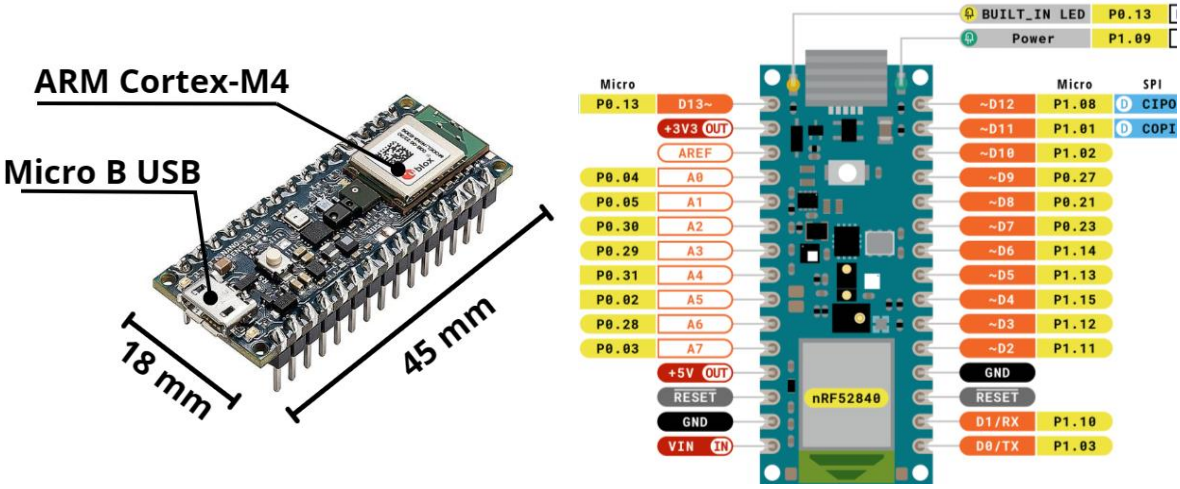
Obr. 3.15: Nepřímá komunikace mesh sítě

4. Experimentální distribuované měření teploty a vlhkosti

V této kapitole je navázáno na vytvořenou síť mesh přidáním sensoru teploty, vlhkosti a přidáním vstupní brány k připojení k internetu. Soustava změří hodnoty teploty a vlhkosti, které následně posílá přes síť až do členu, který data odešle přes Wi-Fi do MQTT serveru ze kterého je možné následně zprávy prostřednictví modelu publish/subscribe nezávisle číst.

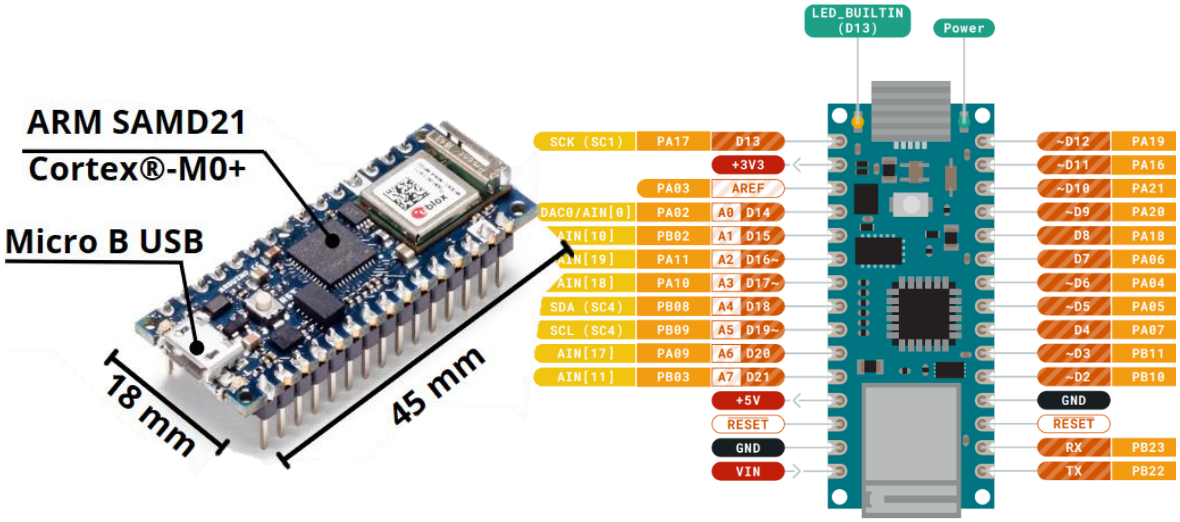
4.1. Použitý hardware

Pro měření teploty/vlhkosti je použito Arduino Nano 33 BLE Sense, které obsahuje zabudovaný senzor teploty a vlhkosti HTS221. Výrobce senzoru je STMicroelectronics.



Obr. 4.1: Arduino Nano 33 BLE Sense a pinový diagram [57], [58]

Jako vstupní bránu na internet je použita deska Arduino NANO 33 IoT. Deska obsahuje modul NINA-W102 schopný připojení k Wi-Fi. Modul je vyroben společností U-blox.

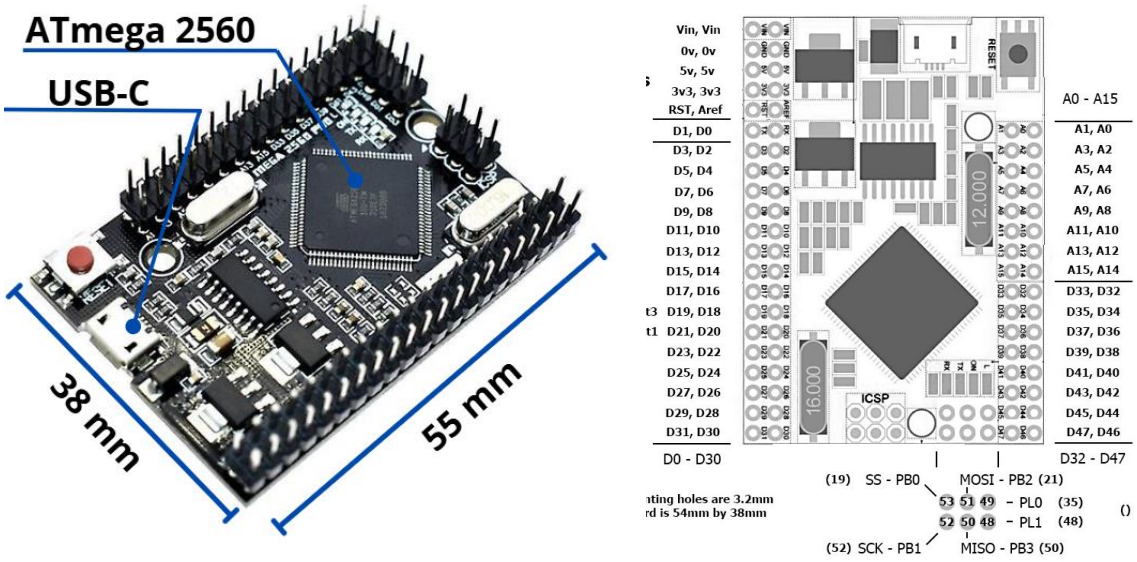


Obr. 4.2: Arduino NANO 33 IoT a pinový diagram [59], [60]

SRAM na deskách Arduino Nano R3 o velikosti 2 KB neumožňuje další zvětšování kódu nad úroveň LoRa mesh sítě o velikosti ca. 4 nodů (viz kapitola 3). Implementace byla proto řešena prostřednictvím desky MEGA 2560 PRO Mini. Jelikož nejvýznačnějším rozdílem z hlediska implementace LoRa mesh sítě je velikost SRAM paměti pro uchování routovací tabulky. Hlavní rozdíly mezi deskami jsou:

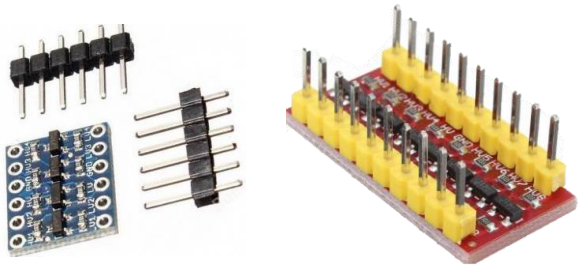
Specifikace	Arduino Nano R3	Arduino 2560 PRO Mini
Mikroprocesor	ATmega328P	ATmega2560
Počet digitálních IO pinů	14	54
Počet analogových IO pinů	8	16
Paměť Flash	32 KB	256 KB
SRAM	2 KB	8 KB
EEPROM	1 KB	4 KB
Rozměry	18 x 45 mm	38 x 55 mm
Váha	7 g	10 g
Vstup	Mini-USB	USB-C
Spotřeba energie	19 mA	70 mA
Počet Sériových portů	1	4

Tabulka 4.1: Hlavní rozdíly desek Arduino Nano R3 a Arduino 2560 PRO Mini [50], [61]



Obr. 4.3: Arduino 2560 PRO Mini s napájenými piny a pinový diagram [62], [63]

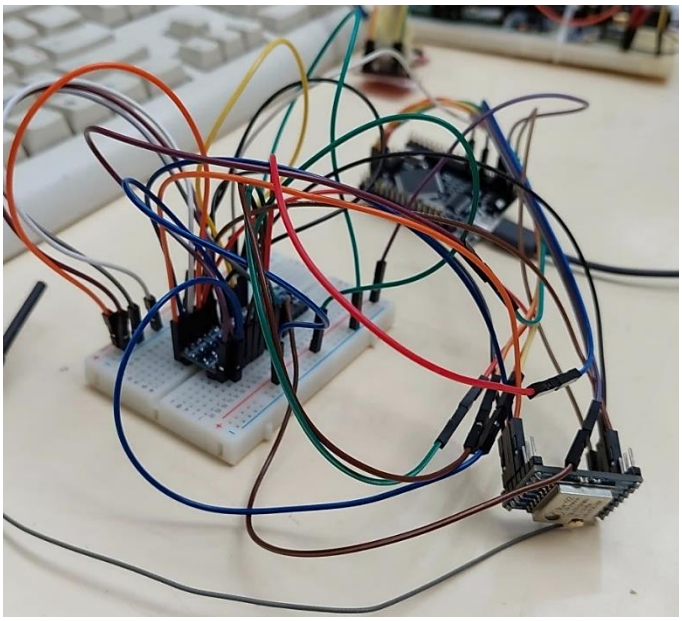
Vzhledem k odlišným pracovním napětím použitých modulů je nutné použít převodníky logických úrovní. Deska Arduino MEGA 2560 PRO Mini pracuje s 5 V, Arduino NANO 33 BLE Sense, Arduino 33 IoT a rádiový modul Ra-02 s 3,3 V.



Obr. 4.4: Obousměrný převodník logické úrovně čtyřkanálový a osmikanálový [64], [65]

4.2. Propojení MEGA 2560 PRO Mini a RA-02

Vzniklé propojení přes logický převodník vyžaduje nepájivé pole a další kabely, kterými propojíme jednotlivé piny součástek. Výsledná podoba viz Obr. 4.5.



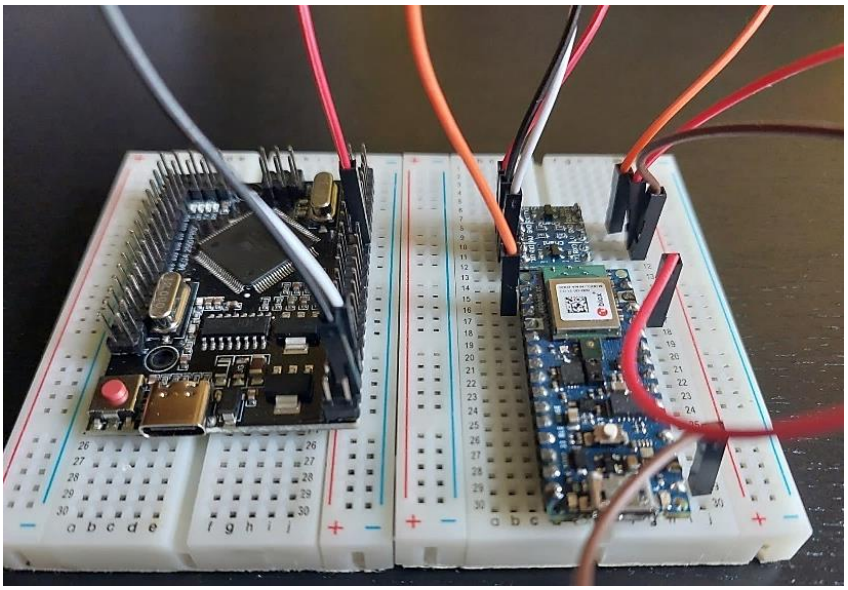
Obr. 4.5: Propojené Arduino MEGA 2560 PRO Mini s modulem RA-02 přes převodník logických úrovní na nepájivém poli

MEGA 2560 PRO Mini	Převodník	Převodník	RA-02
5 V	HV	LV	3,3 V
GND	GND	GND	GND
D2	HV1	LV1	DIO0
D9	HV2	LV2	RST
D50	HV3	LV3	MISO
D51	HV4	LV4	MOSI
D52	HV5	LV5	SCK
D53	HV6	LV6	NSS

Tabulka 4.2: Přehled propojených pinů Arduino MEGA 2560 PRO Mini s modulem RA-02

4.3. Propojení MEGA 2560 PRO Mini a Nano 33 BLE Sense

Rádiový komunikační nod (MEGA 2560 PRO Mini) je oddělen od měřícího členu (Nano 33 BLE Sense). Výměna informace o teplotě a vlhkosti z jedné desky na druhou je zajištěna přes sériový port. Nano 33 BLE Sense měří hodnoty a sériovou komunikací data předává do MEGA 2560 PRO Mini. Přehled propojených pinů s využitím převodníku logických úrovní viz Tabulka 4.3



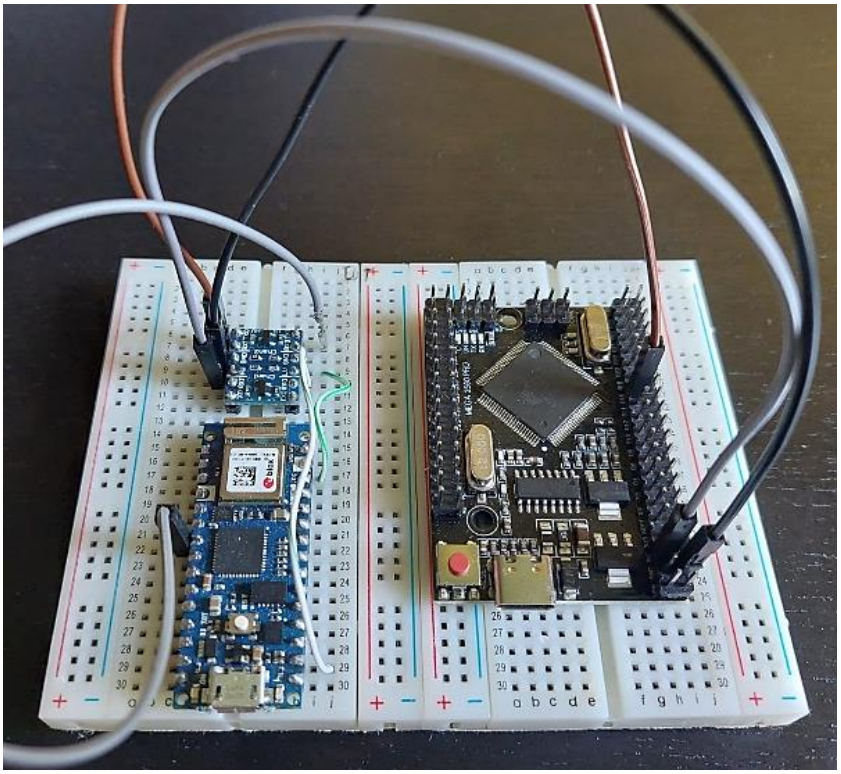
Obr. 4.6: Spojení MEGA 2560 PRO Mini a Nano 33 BLE Sense pomocí převodníku logických úrovní a kabelů za účelem sériové komunikace

MEGA 2560 PRO Mini	Převodník	Převodník	Nano 33 BLE Sense
5 V	HV	LV	3,3 V
GND	GND	GND	GND
D17	TX1	TX0	TX1

Tabulka 4.3: Přehled propojených pinů Arduino MEGA 2560 PRO Mini a Arduino Nano 33 BLE Sense

4.4. Propojení MEGA 2560 PRO Mini a Nano 33 IoT

Pro přenesení informace o teplotě a vlhkosti z mesh sítě do Nano 33 IoT je využívána sériová komunikace z MEGA 2560 PRO Mini. Na Nano 33 IoT není k dispozici sériový port pro externí komunikaci. Sériový port je však možno softwarově nakonfigurovat prostřednictvím sériového multiplexeru SERCOM obsaženého v procesorech SAMD, zde na pinu D7 (TX) a pinu D4 (RX).



Obr. 4.7: Spojení MEGA 2560 PRO Mini a Nano 33 IoT pomocí převodníku logických úrovní a kabelů za účelem sériové komunikace

Nano 33 IOT	Převodník	Převodník	MEGA 2560 PRO Mini
3,3 V	LV	HV	5 V
GND	GND	GND	GND
D4	TX0	TX1	D16

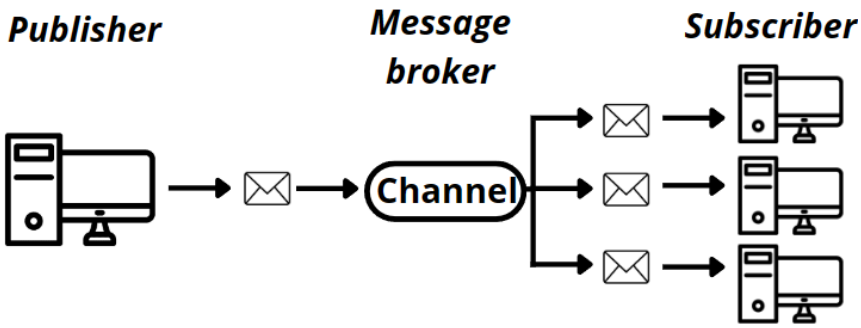
Tabulka 4.4: Přehled propojených pinů Arduino MEGA 2560 PRO Mini Arduino a Nano 33 IoT

4.5. MQTT a důvod použití

Message Queuing Telemetry Transport je otevřený messaging protokol založený na TCP/IP protokolu. Používá se pro zařízení s nízkou šířkou pásma, velkou latencí a nespolehlivým spojením. V principu se snaží zajistit komunikaci o minimální vlnové šířce a minimální spotřebované energie. Posílání dat funguje na konceptu publish/subscribe.

- MQTT klient – zařízení s aktivní MQTT knihovnou
- MQTT broker – server starající se o připojení/odpojení klienta a odesílání dat na vydaný požadavek.
- Topic – třída sloužící pro odlišení dat

Výměna dat probíhá navázáním spojení klienta s brokerem pomocí TCP/IP protokolu a provedením bezpečnostního ověření pomocí přihlašovacích údajů. Dále klient odešle/přijme konkrétní data, broker je obdrží a přepošle je dál k adresátovi. [49]



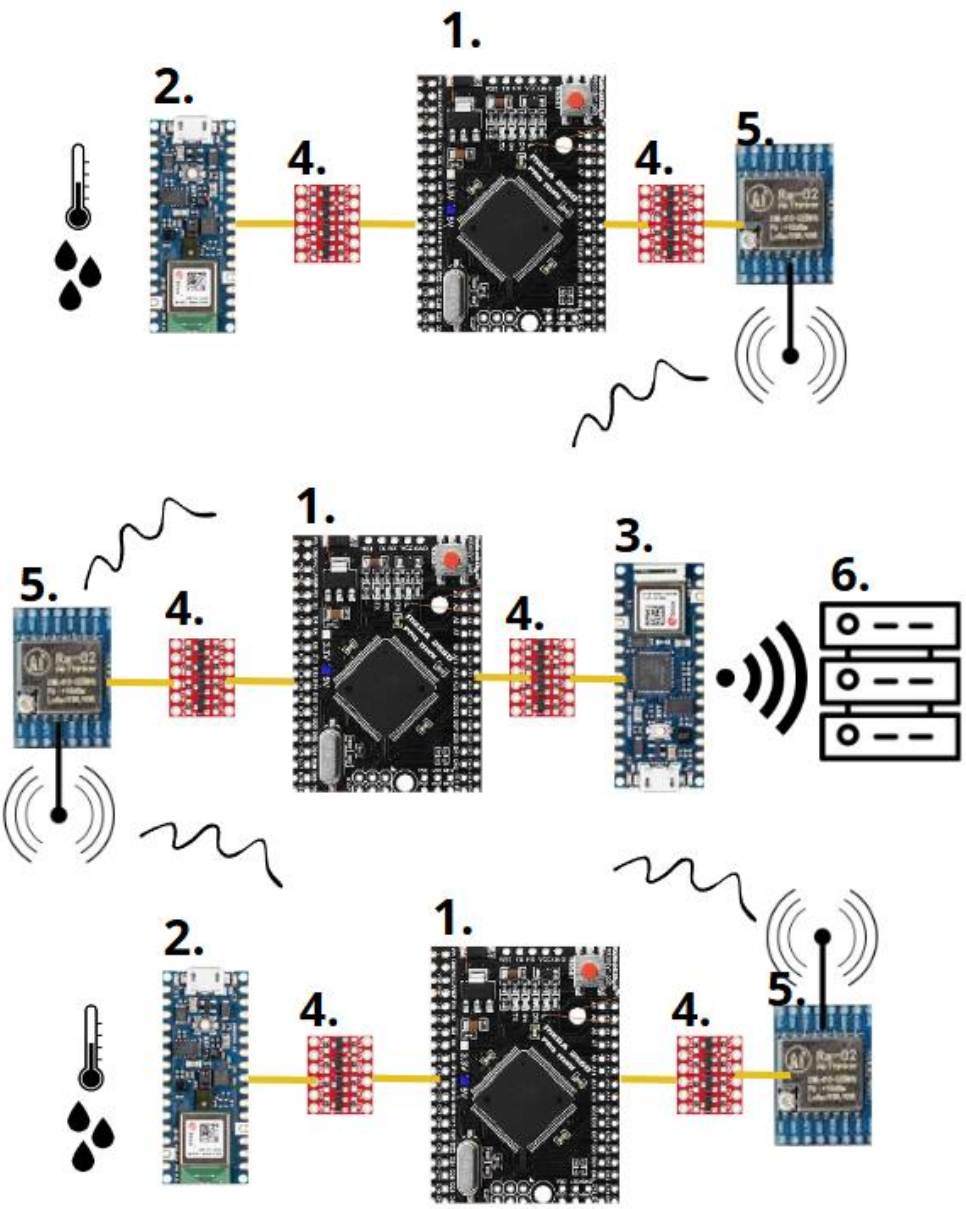
Obr. 4.8: Schéma publish/subscribe principu

MQTT je využito k přenosu informací o stavu sítě a měřených hodnot teplot/vlhkosti pro následné zpracování dat. Arduino poskytuje mnoho různých předpřipravených knihoven. [66]

Qubitro: Jedna z mnohých dostupných cloudových platforem pro IoT projekty. Pro limitovaný počet zařízení se jedná o neplacenou službu. V našem případě Qubitro využijeme pro vytvoření MQTT brokera a vygenerování přístupových údajů. Stránka také nabízí základní funkce pro práci s daty. Uživatel si tak může vytvořit základní graf o jedné proměnné, nebo základní ovládací panel. [67]

Obr. 4.9: Logo Qubtro [67]

4.6. Zjednodušené schéma vytvářené sítě



Obr. 4.10: Zjednodušené schéma vytvářené sítě

1 - Arduino MEGA 2560 PRO Mini
2 - Arduino Nano 33 BLE Sense
3 - Arduino Nano 33 IoT
4 – Převodník logických úrovní 5V 3,3V
5 - Rádiový modul Ra-02
6 - Server

5-5: Rádiová komunikace mezi RA-02
3-6: MQTT komunikace mezi Nano 33 IoT a serverem
1-4, 2-4, 3,4, 4-5: Spojení vodiči
2: Měření teploty a vlhkosti

4.7. Kód Nano 33 BLE Sense pro měření teploty a vlhkosti

Je využito Arduino_HTS221 knihovny, která je kompatibilní s obsaženým senzorem HTS221 na Nano 33 BLE Sense. Pomocí 5. řádku na Obr. 4.11 je inicializováno spuštění senzoru. Řádkem 11 a 12 jsou uloženy hodnoty teploty a vlhkosti. Následně pomocí Serial1 deska odešle hodnoty do komunikačního nodu. Tyto operace provádíme ve smyčce.

```
1  #include <Arduino_HTS221.h>
2  void setup() {
3    Serial.begin(115200;
4    Serial1.begin(115200);
5    if (!HTS.begin()) {
6      Serial.println("Failed to initialize humidity tempe
7      while (1);
8    }
9  }
10 void loop() {
11   float temperature = HTS.readTemperature();
12   float humidity = HTS.readHumidity();
13
14   Serial.print(temperature);
15   Serial.print(",");
16   Serial.print(humidity);
17
18   Serial1.print(temperature);
19   Serial1.print(",");
20   Serial1.print(humidity);
21   delay(3900); // Wait for 1 second before taking the n
```

Obr. 4.11: Kód na Nano 33 BLE Sense pro měření teploty a vlhkosti

4.8. Kód MEGA 2560 PRO Mini pro přenos informací do Nano 33 IoT

Je využit LoRa mesh kód z kapitoly 3, do kterého je přidán kód k zajištění schopnosti pracovat s teplotou a vlhkostí. Knihovna RF95 pro LoRa komunikaci má v základním nastavení jinak definované piny (CS – Chip select, RST – Reset button, INT – interrupt). Kód na řádcích 15, 16, 17 reflektuje změnu těchto pinů. Dále na řádku 21 a 22 jsou vytvořeny proměnné typu float pro teplotu a vlhkost každého nodu. Poslední změnou je řádek 25, při volání knihovny RF95 jsou deklarované změněné piny.

```
15 #define RFM95_CS 53
16 #define RFM95_RST 9
17 #define RFM95_INT 2
18 uint8_t nodeId;
19 uint8_t routes[N_NODES]; // full routing table for me
20 int16_t rssi[N_NODES]; // signal strength info
21 float temp[N_NODES]; // temperature
22 float humi[N_NODES]; // humidity
23 // Singleton instance of the radio driver
24 //RH_RF95 rf95;
25 RH_RF95 rf95(RFM95_CS, RFM95_INT);
26 // Class to manage message delivery and receipt, usin
27 RHMesh *manager;
```

Obr. 4.12: Změněný kód MEGA 2560 PRO Mini pro přenos informací do Nano 33 IoT

4.9. Kód MEGA 2560 PRO Mini pro zpracování teploty a vlhkosti

Kód z kapitoly 4.8 je rozšířen o funkci, která přečte a uloží sériově posílané hodnoty teploty a vlhkosti z Nano 33 BLE Sense. Hlavní změna je ve funkci updateRoutingTable, kde na řádku 134 na Obr. 4.13 je doplněna if podmínka pro čtení sériové komunikace. Na řádcích 137-141 jsou uloženy přijímané hodnoty teploty a vlhkosti.

```
129  void updateRoutingTable() {
130    for(uint8_t n=1;n<=N_NODES;n++) {
131      RHRouter::RoutingTableEntry *route = manager->getRouteTo(n);
132      if (n == nodeId) {
133        routes[n-1] = 255; // self
134        *if (Serial2.available() > 0) {
135          // Read the incoming line
136
137            temp[n-1] = Serial2.parseFloat();
138            Serial.print(" -->");
139            Serial.print(temp[n-1]);
140            while (Serial2.read() != ','); // Skip the comma
141            humi[n-1] = Serial2.parseFloat();
142      }
```

Obr. 4.13: Přidaný kód I na MEGA 2560 PRO Mini pro zpracování teploty a vlhkosti

Do nekonečně běžící smyčky kódu (void loop) jsou přidány řádky, které umožní uložení dat o vlhkosti a teplotě. Pomocí knihovny ArduinoJson je vytvořena proměnná pro ukládání dat. Funkce rozloží zprávu z bufferu a hodnoty uloží do jednotlivých kategorií. Podmínkou je zajištěno nepřepisování dat starými hodnotami, které komunikační nod odeslal dříve. Ostatní části zprávy o teplotě a vlhkosti uložíme do polí temp a humi.

```
248          StaticJsonDocument<1024> doc;
249          DeserializationError error = deserializeJson(doc, buf);
250          if (error) {
251          Serial.print(F("deserializeJson() failed: "));
252          Serial.println(error.f_str());
253          return;
254          }
255          for (JsonObject obj : doc.as<JsonArray>()) {
256            int n = obj["n"];
257            float t = obj["t"];
258            float h = obj["h"];
259
260            if (n == nodeId) {
261            continue;
262            }
263            temp[index] = t;
264            humi[index] = h;
265            index++;
```

Obr. 4.14: Přidaný kód II na MEGA 2560 PRO Mini pro zpracování teploty a vlhkosti

4.10. Kód Nano 33 IoT pro přenos teploty, vlhkosti pomoci Wi-Fi na MQTT

K správnému fungování je využito následujících dostupných knihoven v Arduinu IDE. Viz Obr. 4.15 a Tabulka 4.5.

```
1  #include <Arduino.h>
2  #include <SPI.h>
3  #include <Wire.h>
4  #include <ArduinoJson.h>
5  #include <PubSubClient.h>
6  #include <WiFiNINA.h>
7  #include <QubitroMqttClient.h>
8  #include "wiring_private.h"
9
```

Obr. 4.15: Obsažené knihovny kódu pro Nano 33 IoT

Arduino.h	základní knihovna Arduina, obsahuje základní funkce
SPI.h	knihovna slouží pro komunikaci s dalšími připojenými zařízeními
Wire.h	umožňuje použití nové sériové komunikace
ArduinoJson.h	pro vytváření a rozkládání JSON dat z MQTT komunikace
PubSubClient.h	slouží pro inicializaci a správu MQTT komunikace
WiFiNINA.h	obsahuje funkce pro zahájení a správu WiFi připojení
QubitroMqttClient.h	zjednodušuje práci s MQTT při využívaní platformy Qubitro cloud
Wiring_private.h	použita pro manipulaci hardwarového připojení desky

Tabulka 4.5: Přehled použitých knihoven na desce Nano 33 IoT

Kód na řádku 10 na Obr. 4.16 definuje konstantu MAX_STRING_SIZE o hodnotě 160 bajtů. Rozměr, který je využit k stanovení velikosti pole znaků newData na řádku 11. Kód na řádku 12 vytvoří pole znaků payload o velikosti 256 bajtů. Deklarování, že se jedná o static char, umožňuje uložit danou hodnotu a zároveň hodnotu inicializovat pouze jednou. Na řádku 15 je vytvořen nový UART objekt pod názvem mySerial využívající sercom0 na pinech 4 = RX, 7 = TX pro sériovou komunikaci na SERCOM blocích SERCOM_RX_PAD_3 a UART_TX_PAD_2.

```
10 #define MAX_STRING_SIZE 160
11 char newData[MAX_STRING_SIZE];
12 static char payload[256];
13
14 //Sercom communication
15 Uart mySerial (&sercom0, 4, 7, SERCOM_RX_PAD_3, UART_TX_PAD_2);
16
```

Obr. 4.16: Kód I v Nano 33 IoT

Dále na řádcích 18 a 19 na Obr. 4.17 kód uloží do dvou znakových polí konkrétní deviceID a deviceToken sloužící pro autentizaci MQTT brokeru. Řádek 21 a 22 kódu slouží k deklaraci údajů k připojované WiFi síti. Vytvořením WiFiClient na řádku 23 vznikne instance, která řídí WiFi připojení tím, že ovládá základní TCP/IP stack potřebný pro síťovou komunikaci. Řádek 24 je instance QubitroMqttClient pojmenovaná mqttClient využívající wifiClient pro síťovou komunikaci. Tento řádek iniciuje MQTT klienta, který se připojí k MQTT brokeru přes WiFi.

```
18  char deviceID[] = "";
19  char deviceToken[] = "";
20
21  const char* ssid = "";
22  const char* password = "";
23  WiFiClient wifiClient;
24  QubitroMqttClient mqttClient(wifiClient);
```

Obr. 4.17: Kód II v Nano 33 IoT

Je vytvořen také IrqHandler zodpovídající za přerušované poslouchání SERCOM0. Díky tomu bude mikroprocesor schopen provádět ostatní funkce kódu a přijme zprávu ze sériového portu jen v případě, kdy nějaká data dorazí.

```
17  // Attach the interrupt handler to the SERCOM
18  void SERCOM0_Handler(){
19      mySerial.IrqHandler();
20  }
```

Obr. 4.18: Funkce SERCOM0_Handler v Nano 33 IoT

Ve funkci setup_wifi je nejdůležitější řádek 65, kterým funkce započne připojení k WiFi. Předložením ssid a password se modul pokusí připojit k WiFi, pokud ověřením je správné, získá hodnotu IP adresy pro síťovou komunikaci.

```
59  void setup_wifi(){
60    delay(100);
61    Serial.println();
62    Serial.print("Connecting to ");
63    Serial.println(ssid);
64    delay(100);
65    WiFi.begin(ssid, password);
66    while( WiFi.status() != WL_CONNECTED){
67      delay(500);
68      Serial.print(".");
69      WiFi.begin(ssid, password);
70      delay(500);
71    }
72    randomSeed(micros());
73    Serial.println("");
74    Serial.println("WiFi connected");
75    Serial.println("IP address: ");
76    Serial.println(WiFi.localIP());
```

Obr. 4.19: Funkce setup_wifi v Nano 33 IoT

Dále je vytvořena funkce storeData, která uloží zprávu získanou přes sériový port. Tato funkce obdrží na vstupu string. Proiteruje postupně každým znakem stringu a uloží si pozice znaků „[“ a „]“. Na řádku 35 na Obr. 4.20 zkopíruje znaky mezi „[“, „]“ a kódem na řádku 36 přidá ohraničení pomocí nul. Touto funkcí získáme sjednocený tvar zprávy.

```
22  void storeData(const char* inputString) {
23    // Find the position of '[' and ']'
24    int startPos = 0;
25    int endPos = 0;
26    for (int i = 0; i < strlen(inputString); i++) {
27      if (inputString[i] == '[') {
28        startPos = i;
29      } else if (inputString[i] == ']') {
30        endPos = i;
31        break;
32      }
33    }
34    // Extract the desired substring
35    strncpy(newData, inputString + startPos, endPos - startPos + 1);
36    newData[endPos - startPos + 1] = '\0';
```

Obr. 4.20: Funkce storeData v Nano 33 IoT

Ve funkci quibitro_init je stanovena adresa brokera a hodnota portu. Na řádku 81 je uložen identifikátor zařízení. Řádek 82 slouží k zabezpečenému připojení k MQTT. Pokud připojení neproběhne správně, klient vrátí číslo chyby, v opačném případě začne zařízení odebírat zprávy.

```
78  void qubitro_init() {
79    char host[] = "broker.qubitro.com";
80    int port = 1883;
81    mqttClient.setId(deviceID);
82    mqttClient.setDeviceIdToken(deviceID, deviceToken);
83    Serial.println("Connecting to Qubitro...");
84
85    if (!mqttClient.connect(host, port))
86    {
87      Serial.print("Connection failed. Error code: ");
88      Serial.println(mqttClient.connectError());
89      Serial.println("Visit docs.qubitro.com or create a new issue
90      delay(500);
91    }
92    else {
93    Serial.println("Connected to Qubitro.");
94    mqttClient.subscribe(deviceID);
95    delay(500);
```

Obr. 4.21: Funkce qubitro_init v Nano 33 IoT

V setupu je stanoven výstupní pin pro zabudovanou diagnostickou LED na řádku 119. Na řádcích 121, 122 je přiřazena pinům 4, 7 funkce SERCOM komunikace. Sériová komunikace je zahájena přes Serial.begin a mySerial.begin. Funkce k připojení k WiFi a MQTT jsou následně zavolány.

```
118  void setup() {
119    pinMode(LED_BUILTIN, OUTPUT);
120    // Reassign pins 5 and 6 to SERCOM alt
121    pinPeripheral(4, PIO_SERCOM_ALT);
122    pinPeripheral(7, PIO_SERCOM_ALT);
123    // Start my new hardware serial
124    mySerial.begin(115200);
125    Serial.begin(115200);
126    setup_wifi();
127    qubitro_init();
128  }
```

Obr. 4.22: Funkce setup v Nano 33 IoT

V kódu, který běží v nekonečné smyčce, je na řádku 136 ověřena dostupnost dat na vytvořeném sériovém portu z MEGA 2560 PRO Mini. Při obdržené zprávě, je vytvořen buffer pro uložení obdržených dat. Dále na řádku 143 je znovu ověřena dostupnost dat a je započato čtení dat až do doby, než je naraženo na „\n“. Přečtená data jsou uložena. V tomto případě je vyžíváno dynamicky alokované paměti.

```
135  void loop() {
136    if (mySerial.available() > 0) {
137      Serial.print("-");
138      char* buffer = NULL;
139      size_t bufferSize = 0;
140      char incomingChar;
141      // Read characters until newline is encountered
142      while (true) {
143        if (mySerial.available() > 0) {
144          incomingChar = mySerial.read();
145          if (incomingChar == '\n') {
146            break;
147          }
148          buffer = (char*)realloc(buffer, bufferSize + 2);
149          if (buffer == NULL) {
150            Serial.println("Memory allocation failed!");
151            return;
152          }
153          buffer[bufferSize++] = incomingChar;
154          buffer[bufferSize] = '\0'; // Null-terminate the
155        }
156      }
```

Obr. 4.23: Kód I v nekonečné smyčce v Nano 33 IoT

Na řádku 158 je vytvořen ukazatel const char* message na buffer ke čtení. Je ověřen výskyt symbolu „>“. Když je obsažen, je zavolána funkce storeData.

```
158      const char* message = buffer;
159      // Check if the input string contains the desired label
160      if (strstr(message, ">") != NULL) {
161        Serial.print("> ");
162        storeData(message);
```

Obr. 4.24: Kód II v nekonečné smyčce v Nano 33 IoT

Na řádku 165 je deklarována třída StaticJsonDocument o velikosti 320 bajtů. Řádkem 166 je zavolána funkce deserializeJson z knihovny ArduinoJson.h, která rozloží zprávu newData a uloží ji do proměnné typu StaticJsonDocument s názvem doc_it.

```
165        StaticJsonDocument<320> doc_it;
166        deserializeJson(doc_it, newData);
```

Obr. 4.25: Kód III v nekonečné smyčce v Nano 33 IoT

V další části kódu je zkontrolována stálost připojení MQTT klienta, v případě odpojení dojde k znovupřipojení. Řádek 174 kódu inicializuje odchod MQTT zprávy. Použitím Begin.Message(deviceID) je vytvořena zpráva identifikovaná obsahem proměnné deviceID, dále se přiloží upravená zpráva newData a zpráva je ukončena.

```
171     if (!mqttClient.connected()){
172       qubitro_init();
173     }
174     mqttClient.poll();
175     mqttClient.beginMessage(deviceID);
176     mqttClient.println(newData);//payload
177     Serial.println(newData);
178     mqttClient.endMessage();
179     }
180     free(buffer);
181   }
```

Obr. 4.26: Kód IV v nekonečné smyčce v Nano 33 IoT

4.11. Dynamický graf sítě z MQTT zpráv

Obdržená data z MQTT o síle signálu a jednotlivých nodech jsou využita k realizaci dynamického grafu z pohledu nodu 2 pro vizuální reprezentaci stavu sítě. Implementace je provedena v jazyce Python. Využité knihovny v programu jsou viz Obr. 4.27 a Tabulka 4.6.

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3  import matplotlib.animation as animation
4  import paho.mqtt.client as mqtt
5  import json
```

Obr. 4.27: Použité knihovny pro vytvoření dynamického grafu sítě v prostředí Python

Networkx	poskytuje nástroje na vytváření komplexních grafů a sítí
Matplotlib.pyplot	umožňuje vytvářet vizualizace a grafy
Matplotlib.animation	nadstavba matplotlib balíčku
Paho.mqtt.client	propojuje Python s MQTT brokerem
Json	umožňuje základní operace s daty ve formátu JSON

Tabulka 4.6: Použité knihovny a základní charakteristika

Pro vytvoření propojeného grafu díky networkx je nejprve nutno definovat počet nodů a jednotlivá pravidla pro propojování nodů mezi sebou. Graf by měl umožňovat:

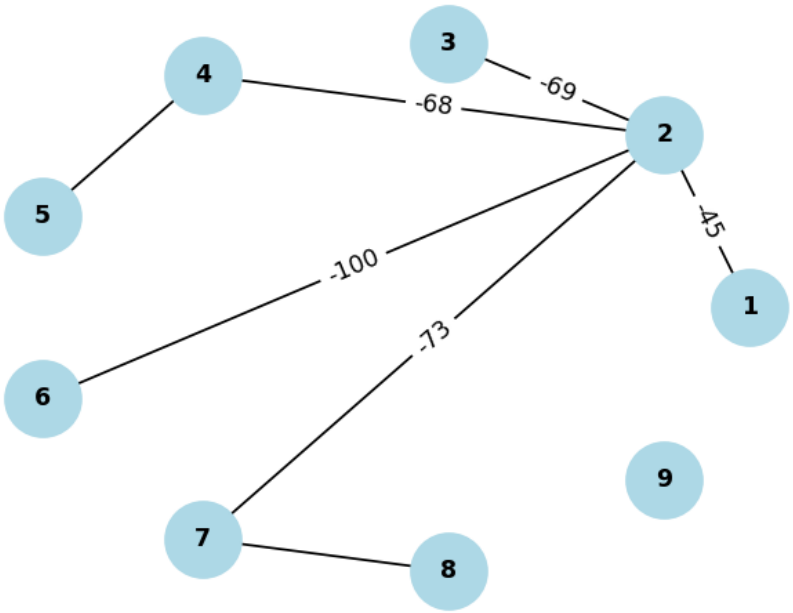
1) Vytvoření každého nodu komunikace
2) Při úspěšné komunikaci vytvoření vizuální propojení nodů
3) Při ztrátě signálu (odpojení nodu), reflektuje změnu

Rozhodovací logika a její implementace v Pythonu vypadá následovně:

```
for data in [message]:
    if data[1]["n"] == 255:
    #if data[interval]["n"] == 255:
        for idx, item in enumerate(data):
            # if item["n"] == 255:
                    # node = 2  # Node 255 is represented as node 2

            if item["n"] != 255:
                node = item["n"]

            # for adding nodes which are not connected straight to centra
            if node != idx + 1 :  # Assuming line numbers start from 1
                node = idx + 1

            G.add_node(node)
            # G.nodes[node]["Text"] = item["Text"]
            if item["n"] != 255 and item["r"] != 0:
                G.add_edge( u_of_edge: 2, node, length=item["r"])  # Edge fr
            elif item["n"] != 255 and item["n"] != 0  and item["r"] == 0:
                G.add_edge(item["n"], node)  # Give edge to not directly
```

Obr. 4.28: Kód pro vytváření vizuální podoby sítě v prostředí Python

Výsledný graf smyšlené sítě například o 9 nodech:



Obr. 4.29: Dynamický graf pro vizuální reprezentaci propojení smyšlené LoRa mesh sítě o 9 nodech z pohledu nodu 2 s příslušnými rssi hodnotami (vysílání)

4.12. Graf pro vykreslení změřených teplot a vlhkostí

Je využit jazyk Python pro vytvoření grafu pro zobrazení průběhu teploty a vlhkosti v čase. Využité knihovny jsou shodné s knihovnami z kapitoly 4.10.

```
1  import json
2  import paho.mqtt.client as mqtt
3  import matplotlib.pyplot as plt
4  from matplotlib.animation import FuncAnimation
```

Obr. 4.30: Použité knihovny pro vytvoření grafu pro vykreslení změřených teplot a vlhkostí

Ve programu je definován počet nodů síti a zároveň jsou vytvořeny seznamy pro teplotu a vlhkost na ukládání hodnot z obdžených MQTT zpráv.

```
14  nodes =   # add number of current nodes
15  # Initialize lists to hold temperature and humidity data
16  temp_data = [[] for _ in range(nodes)]
17  humidity_data = [[] for _ in range(nodes)]
```

Obr. 4.31: Definování počtu nodů a seznamů pro graf změřených teplot a vlhkostí

Funkce jménem update (viz Obr. 4.32) na vstupu dostane parametr frame od animace grafu. Proiteruje všemi nody a i-té složce teploty přiřadí souřadnici x o hodnotě velikosti momentální délky seznamu a souřadnici teploty y přiřadí i-tou hodnotu. Na podobném princupu funguje i aktualizování vlhkosti s tím rozdílem, že hodnoty ukládá na jiné pozice v seznamu lines, aby se mohla závislost vlhkosti zobrazit samostatně od závistlosti teploty. Na konec pouze vrátí seznam lines, který se použíje pro aktualizování grafu.

```
30  # Update function for the animation
31  def update(frame):
32      for i in range(nodes):
33          lines[i].set_data(range(len(temp_data[i])), temp_data[i])
34          lines[i+nodes].set_data(range(len(humidity_data[i])), humidity_data[i])
35      return lines
```

Obr. 4.32: Definování funkce update pro graf změřených teplot a vlhkostí

Funkce on_message je vytvořena pro zpracování zpráv. Při obdržené zprávě od MQTT brokera, funkce zprávu zpracuje a uloží ji do proměnné typu slovník pod jménem payload.

```
37  # MQTT callback function
38  def on_message(client, userdata, message):
39      global temp_data, humidity_data
40      print(str(message.payload.decode))
41      payload = json.loads(message.payload.decode())
42      print("Received message: "+str(payload))
```

Obr. 4.33: Definování funkce on_message pro graf změřených teplot a vlhkostí

Dále projde všechny položky slovníku payload. Proměnným n přiřadí všechny hodnoty z proměnné data s klíčem n. Dále pokud n je menší než počet nodů přidá hodnoty teploty a vlhkosti do jejich listů. Při více jak 100 hodnotách, dojde k odstranění nejstarší hodnoty a nahrazení novou hodnotou.

```
45      for data in payload:                                    #
46          n = data['n'] -1
47          if n < nodes:  # Ensure we have only valid node
48              temp_data[n].append(data['t'])
49              humidity_data[n].append(data['h'])
50              # Keep only the last 100 data points
51              if len(temp_data[n]) > 100:
52                  temp_data[n].pop(0)
53              if len(humidity_data[n]) > 100:
54                  humidity_data[n].pop(0)
```

Obr. 4.34: Pokračování v definování funkce on_message pro graf změřených teplot a vlhkostí

Pro smyšlenou síť o 7 nodech a 6 měřících členech může graf pro časový průběh teploty a vlhkosti nabývat této podoby:

Obr. 4.35: Graf závislosti teploty a vlhkosti na čase pro smyšlenou síť

4.13. Provedení experimentálního měření teploty a vlhkosti

K provedení experimentálního měření teploty a vlhkosti je použito vytvořeného zařízení z kapitoly 4. Soustava monitoruje průběh veličin ve skleníku, který se nachází zhruba 6 metrů od domu. Každých 30 minut soustava změří a odešle údaje. Pro porovnání paralelně je monitorován průběh daných veličin i ve vnitřních prostorech domu.

Obr. 4.36: Měřící soustava teploty a vlhkosti v prostorech skleníku

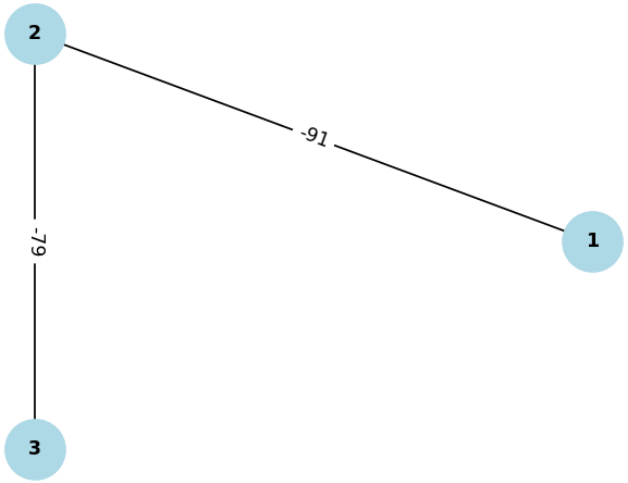
Soustava odesílá data a prostřednictvím MQTT brokeru odebíráme data. Celková sít se skládá ze 3 komunikačních nodů. První zpráva je z času 08:00, druhá 08:30, třetí 09:00, čtvrtá 09:30, atd. Celkové měření hodnot provedeme od času 8:00 do času 20:00.

```
{'n': 1, 'r': -88, 't': 19.97, 'h': 80.27}, {'n': 255, 'r': 0, 't': 0, 'h': 0)}, {'n': 3, 'r': -79, 't': 23.75, 'h': 63.75}]
{'n': 1, 'r': -91, 't': 24.26, 'h': 62.93}, {'n': 255, 'r': 0, 't': 0, 'h': 0)}, {'n': 3, 'r': -77, 't': 24.04, 'h': 62.83}]
{'n': 1, 'r': -89, 't': 27.10, 'h': 64.01}, {'n': 255, 'r': 0, 't': 0, 'h': 0)}, {'n': 3, 'r': -79, 't': 22.60, 'h': 68.18}]
{'n': 1, 'r': -91, 't': 27.83, 'h': 58.50}, {'n': 255, 'r': 0, 't': 0, 'h': 0)}, {'n': 3, 'r': -79, 't': 23.46, 'h': 66.61}]
```

Obr. 4.37: Obdržené MQTT zprávy od měřící soustavy v prostředí Python

4.14. Výsledek měření

Výsledná podoba sítě při měření:



Obr. 4.38: Výsledná podoba sítě s rssi hodnotami v experimentu měření teploty a vlhkosti (nod 2 přijímá zprávy)

kde nod 1 je umístěn ve skleníku, nod 3 v obývacím pokoji domu a nod 2 s MQTT komunikací v jiném pokoji domu.

Při interpretaci hodnot signálu je zřejmé, že jednotlivé komunikační členy mají dostatečně silný signál. V průběhu měření nedošlo k žádnému nedoručení měřených hodnot. Vytvořená síť je proto robustní a pro náš případ dostatečná. Naměřené hodnoty:

	Skleník		Domov			Skleník		Domov	
Čas	Teplota [°C]	Vlhkost [%]	Teplota [°C]	Vlhkost [%]	Čas	Teplota [°C]	Vlhkost [%]	Teplota [°C]	Vlhkost [%]
8:00	19,97	80,27	23,75	63,75	14:30	31,64	52,31	23,01	65,10
8:30	24,26	62,93	24,04	62,83	15:00	32,08	44,20	23,28	63,89
9:00	27,10	64,01	22,60	68,18	15:30	31,81	50,34	23,35	64,52
9:30	27,83	58,50	23,46	66,61	16:00	31,54	48,23	23,08	65,90
10:00	29,10	46,58	23,02	66,08	16:30	30,82	45,14	22,79	67,41
10:30	30,15	44,08	22,81	65,32	17:00	30,53	43,84	22,66	66,84
11:00	28,73	53,01	23,29	65,32	17:30	30,36	49,83	22,39	67,11
11:30	26,99	66,40	23,33	65,22	18:00	28,62	55,15	22,30	68,37
12:00	28,55	36,68	22,97	65,62	18:30	29,44	46,89	22,53	67,90
12:30	28,88	32,64	22,89	65,35	19:00	29,00	53,28	22,22	68,73
13:00	30,72	43,01	22,13	69,02	19:30	25,60	64,07	22,45	68,13
13:30	30,96	47,71	22,58	65,53	20:00	24,19	74,93	22,18	68,13
14:00	31,42	49,25	22,71	65,41					

Tabulka 4.7: Naměřené hodnoty z experimentálního měření teploty a vlhkosti

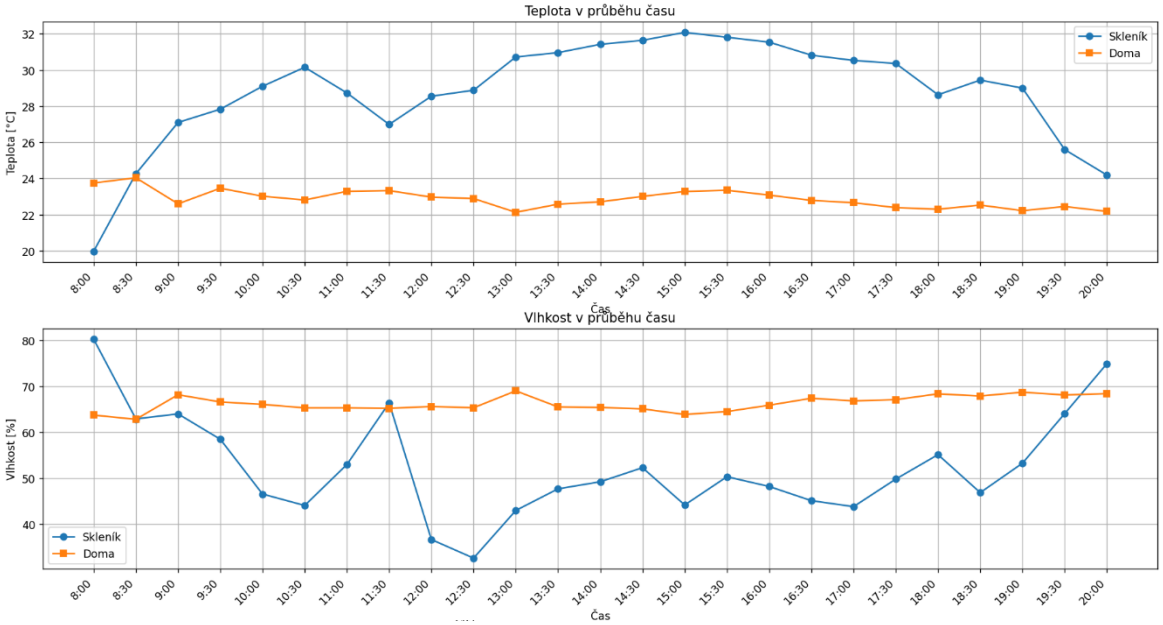
Pro interpretaci změřených dat je vypočtena průměrná hodnota a nejistota měření dle standartní chyby průměru, hodnoty jsou také vyneseny do grafu:

$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.1}$$

$$\tilde{x}_{chyb} = \sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(x_i - \tilde{x})^2} \tag{4.2}$$

kde n je počet měření, $\tilde{x}$ průměrná hodnota, x_i jednotlivá měření

Časový průběh hodnot v grafu:



Obr. 4.39: Graf závislosti relativní vlhkosti a teploty na čase v prostředí skleník a domov

Průměrná teplota skleníku v daném čase:

$$\widetilde{t_s} = 28{,}8 \pm 0{,}6\ °\mathrm{C}$$

Průměrná relativní vlhkost ve skleníku v daném čase:

$$\widetilde{h_s} = 52{,}5 \pm 2{,}2\ \%$$

Průměrná teplota v domu v daném čase:

$$\widetilde{t_d} = 22{,}9 \pm 0{,}1\ °\mathrm{C}$$

Průměrná relativní vlhkost v domě v daném čase:

$$\widetilde{h_d} = 66{,}3 \pm 0{,}3\ \%$$

Tyto výsledky ukazují, že teplota ve skleníku je v průměru vyšší než teplota v domě, což je očekávané vzhledem k tomu, že skleníky slouží k udržování vyšších teplot pro podporu růstu rostlin. Z průměrné hodnoty relativní vlhkosti vzduchu ve skleníku můžeme usoudit, že by mělo dojít ke zvýšení hodnoty relativní vlhkosti, jelikož doporučená hodnota relativní vlhkosti se pohybuje kolem 60-70 %. Je ale důležité zmínit, že měření ve skleníku byla prováděna za slunečného dne s otevřenými okny a dveřmi. To mohlo přispět k vyšší variabilitě naměřených hodnot, protože sluneční záření a přístup vnějšího vzduchu mohou výrazně ovlivnit teplotu a relativní vlhkost ve skleníku.

Při pohledu na hodnoty z měření pro domácí prostředí je patrno, že teplota a relativní vlhkost jsou stabilnější v porovnání s prostředím skleníku. Teplota v domácím prostředí se neměnila ani v odpoledním čase, kdy bychom očekávali maximální vnější teploty v daný den. Z toho můžeme usoudit, že izolační prvky domu fungují v tomto ohledu správně.

Z vytvořeného grafu je dobře vidět vzájemné ovlivňování teploty a relativní vlhkosti. Teplejší vzduch totiž může zadržovat více vodní páry něž vzduch chladnější. To znamená, že při zvýšení teploty se relativní vlhkost vzduchu sníží, pokud množství vodní páry zůstane konstantní. Naopak při ochlazení vzduchu se relativní vlhkost zvýší.

5. Závěr

Tato bakalářská práce měla za účel sestavit funkční rádiovou síť LoRa pro topologii mesh na platformě Arduino a využít ji pro distribuované měření teploty a vlhkosti. Díky tomu může být vytvořená soustava použita pro dálkové monitorování veličin. Možných aplikací této sítě je vícero, jelikož pouhou změnou senzorů může síť začít monitorovat veličiny jiné.

První část práce byla věnována průmyslovým komunikačním sítím. V krátkosti byla zmíněna jejich historie. Navázáno bylo uvedením existujících topologií a jejich charakteristiky. Následně byly popsány průmyslové kabelové a bezdrátové sítě, značný prostor byl vyčleněn i vybraným zástupcům. První cíl práce tím byl splněn.

V další části bakalářské práce byla realizována a experimentálně ověřena funkčnost LoRa mesh sítě na platformě Arduino. Byl popsán použitý hardware a využívaný kód. Zároveň byly ozkoušeny základní komunikační vlastnosti mesh sítě jako je například nepřímá komunikace. Splněn byl cíl druhý a třetí.

Dále bylo implementováno řešení pro měření teploty a vlhkosti pomocí dalších vývojových Arduino desek. Vytvořena byla také MQTT komunikace, která nám umožnila dále zpracovat obdržená data ze sítě, například ve formě dynamického grafu o stavu síly signálu sítě a také k vynášení měřených hodnot teploty a vlhkosti do časově aktualizovaného grafu. Proveden byl dále základní experiment měření teploty a vlhkosti. Měřící soustava byla otestována pro měření v prostorech skleníku a paralelně s tím pro prostory v domě. Výsledek měření je shrnut v kapitole 5. Splněn byl jak čtvrtý, tak i pátý cíl bakalářské práce.

Použitá literatura

[1] Starkey Lab inc; Oticon as. Wireless communication protocol. USA. US7529565B2. Přihlášeno 2004-12-01. Uděleno 2009-05-05.

[2] M. Wollschlaeger, T. Sauter and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," in IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17-27, March 2017, doi:10.1109/MIE.2017.2649104.

[3] Pereira, C.E., Neumann, P. (2009). Industrial Communication Protocols. In: Nof, S. (eds) Springer Handbook of Automation. Springer Handbooks. Springer, Berlin, Heidelberg.https://doi.org/10.1007/978-3-540-78831-7_56.

[4] V. Mhatre and C. Rosenberg, "Homogeneous vs heterogeneous clustered sensor networks: a comparative study," 2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577), Paris, France, 2004, pp. 3646-3651 Vol.6, doi: 10.1109/ICC.2004.1313223.

[5] Perry S. Marshall a John S. Rinaldi, 2005. Industrial Ethernet. Second Edition. ISA. ISBN 9781556178924.

[6] Deep Medhi a Karthik Ramasamy, 2018. In: Network Routing. Second edition. Morgan Kaufmann, s. 2-29. ISBN 978-0-12-800737-2.

[7] Singh, P., & Verma, S. (2015). Network Topologies. *IJRDO -Journal of Computer Science Engineering*, *1*(5), 01-09. https://doi.org/10.53555/cse.v1i5.1184

[8] Types of Network Topology, 2023. GeeksforGeeks [online]. [cit. 2024-03-31]. Dostupné z: https://www.geeksforgeeks.org/types-of-network-topology/

[9] HELSON, Ronald B., 1996. The HART® protocol an enabler for improved plant performance. In: ISA Transactions. Volume 35, Issue 2. ISA, s. 159-164. ISSN 0019-0578.

[10] HART Communication Based Distributed Measurement System, 2019. International Journal of Research And Analytical Reviews [online]. 6(1), 196-200 [cit. 2024-04-02]. ISSN 2348 –1269.
Dostupné z: http://ijrar.com/upload_issue/ijrar_issue_20543720.pdf

[11] AS-International Association e.V. [online], 2024. [cit. 2024-04-02]. Dostupné z: https://www.as-interface.net/en

[12] Piercing technology, 2024. In: As-interface [online]. [cit. 2024-05-28]. Dostupné z: https://www.as-interface.de/en/as-interface/technology/

[13] IO-Link Design Guideline, 2018. IO-Link [online]. [cit. 2024-04-02]. Dostupné z: https://io-link.com/share/Downloads/Planung/IO-Link_Design_Guideline_eng_2018.pdf

[14] Ucelené řešení IO-Link, 2024. In: Pepperl+fuchs [online]. [cit. 2024-05-28]. Dostupné z: https://www.pepperl-fuchs.com/czech_republic/cs/io-link.htm

[15] Tindell, Hansson and Wellings, "Analysing real-time communications: controller area network (CAN)," 1994 Proceedings Real-Time Systems Symposium, San Juan, PR, USA, 1994, pp. 259-263, doi: 10.1109/REAL.1994.342710.

[16] The CAN Bus Protocol Tutorial, 2024. Kvaser [online]. [cit. 2024-04-03]. Dostupné z: https://www.kvaser.com/can-protocol-tutorial/

[17] Yong Xie, Gang Zeng, Kurachi Ryo, Guoqi Xie, Yong Dou a Zhili Zhou, 2017. An optimized design of CAN FD for automotive cyber-physical systems,. Journal of Systems Architecture, [online]. (81), Pages 101-111 [cit. 2024-05-14]. ISSN 1383-7621. Dostupné:https://www.sciencedirect.com/science/article/pii/S1383762116301771

[18] Kabel datový PROFIBUS 1x2x0,64 PVC fialový, 2024. In: Sonepar [online]. [cit. 2024-05-28]. Dostupné z: https://www.sonepar.cz/kabel-simatic-net-6xv1830-0eh10-1x2x064profibus-l2

[19] G. Gabor, C. Pintilie, C. Dumitrescu, N. Costica and A. T. Plesca, "Application of Industrial PROFIBUS-DP Protocol," 2018 International Conference and Exposition on Electrical And Power Engineering (EPE), Iasi, Romania, 2018, pp. 0614-0617, doi: 10.1109/ICEPE.2018.8559857.

[20] PROFIBUS [online], 2024. [cit. 2024-04-03]. Dostupné z:https://www.profibus.com/technologies/profibus#tab1-232296

[21] DeviceNet – designed for factory automation, 2024. CAN in Automation [online]. 1992-2024 [cit. 2024-04-03]. Dostupné z: https://www.can-cia.org/can-knowledge/hlp/devicenet/

[22] Why should i upgrade from controlnet/devicenet to ethernet/ip?, 2020. JCS [online]. [cit. 2024-04-03]. Dostupné z: https://www.jcs.com/blog/why-should-i-upgrade-from-controlnet/devicenet-to-ethernet/ip

[23] Understanding Modbus TCP-IP: An In depth Exploration, 2023. Wevolver [online]. [cit. 2024-04-04]. Dostupné z: https://www.wevolver.com/article/understanding-modbus-tcp-ip-an-in-depth-exploration

[24] Ethernet/IP - ODVA, 2024. ODVA [online]. [cit. 2024-04-04]. Dostupné z: https://www.odva.org/technology-standards/key-technologies/ethernet-ip/

[25] The hateful eight: pros and cons of Ethernet/IP, 2016. Rinaldi, John. RTAutomation [online]. [cit. 2024-04-04]. Dostupné z: https://www.rtautomation.com/rtas-blog/the-hateful-eight-pros-and-cons-of-ethernet-ip/

[26] Vincent, Sean J., 2001. Foundation™ Fieldbus High Speed Ethernet Control System. Fieldbus inc [online]. [cit. 2024-04-05]. Dostupné z: https://www.fieldbusinc.net/downloads/hsepaper.pdf

[27] Bowne, Michael, 2021. What is Profinet - Profinet explained. Profinet [online]. [cit. 2024-04-05]. Dostupné z: https://us.profinet.com/profinet-explained/

[28] Vnet/IP Network Construction Guide, 2012. Yokogawa [online]. [cit. 2024-04-05]. Dostupné z: https://web-material3.yokogawa.com/TI30A10A05-01E.pdf

[29] Vnet/IP system configuration, 2019. In: Vnet/IP Network Construction Guide [online]. [cit. 2024-05-28]. Dostupné z: https://web-material3.yokogawa.com/TI30A10A05-01E.pdf

[30] Powerlink, 2024. Br-automation [online]. [cit. 2024-04-05]. Dostupné z: https://www.br-automation.com/cs/technologie/powerlink/

[31] EtherCAT - the Ethernet Fieldbus, 2024. EtherCAT [online]. [cit. 2024-04-06]. Dostupné z: https://www.ethercat.org/en/technology.html

[32] Introduction to EtherCAT Technology and the EtherCAT Protocol, 2024. Acontis [online]. [cit. 2024-04-06]. Dostupné z: https://www.acontis.com/en/what-is-ethercat-communication-protocol.html

[33] Lutz, Peter, 2024. Communicate flexibly and continuously with Sercos. Sercos [online]. [cit. 2024-04-06]. Dostupné z: https://www.sercos.org/technology/what-is-sercos/

[34] Nicholas Kottenstette, Panos J. Antsaklis: Communication in Automation, Including Networking and Wireless. Handbook of Automation 2009: 237-248

[35] Wireless network model for industrial wireless control, 2020. In: ResearchGate [online]. [cit. 2024-05-28]. Dostupné z: https://www.researchgate.net/figure/Wireless-network-model-for-industrial-wireless-control-Wireless-devices-with-weak_fig1_344335295

[36] Shakib, J., & Muqri, M. R. (2011, June), Wireless Technologies in Industrial Automation Systems Paper presented at 2011 ASEE Annual Conference & Exposition, Vancouver, BC. 10.18260/1-2--18823

[37] Silvestre-Blanes, Javier, 2010. Factory automation. InTech. ISBN 978-953-51-5911-7.

[38] Comparing Wireless Protocols for Industrial Automation, 2024. DigiKey [online]. 1995-2024 [cit. 2024-04-09]. Dostupné z: https://www.digikey.cz/en/articles/comparing-wireless-protocols-for-industrial-automation

[39] S. Palanisamy, S. S. Kumar and J. L. Narayanan, "Secured wireless communication for industrial automation and control," 2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, India, 2011, pp. 168-171

[40] Bluetooth technology is the smart choice for industrial IoT (IIoT), 2020. Avnet [online]. [cit. 2024-04-10]. Dostupné z: https://www.avnet.com/wps/wcm/connect/onesite/4ba765fa-5b8b-427e-8489-797596c9ef20/Avnet-Nordic-nRF52840.pdf?MOD=AJPERES&CVID=nidLD1M&attachment=false&id=1600105911283

[41] Bilstrup, Urban & Wiberg, P.-A. (2000). Bluetooth in industrial environment. 239 - 246. 10.1109/WFCS.2000.882555.

[42] Ready for the New Bluetooth® 5.4? – What You Should Know First, 2024. Silicon Labs [online]. [cit. 2024-05-14]. Dostupné z: https://www.silabs.com/blog/the-new-bluetooth-5-4-what-you-should-know-first

[43] G. Patti, L. Leonardi and L. Lo Bello, "A Bluetooth Low Energy real-time protocol for Industrial Wireless mesh Networks," IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 2016, pp. 4627-4632

[44] LoRa - A Survey of Recent Research Trends, 2018. Pori, Finland ** Yokohama, Japan. Survey. Tampere University of Technology, Pervasive Computing ** Keio University/Graduate School of Media and Governance.

[45] A. S. Bharadwaj, R. Rego and A. Chowdhury, "IoT based solid waste management system: A conceptual approach with an architectural solution as a smart city application," 2016 IEEE Annual India Conference (INDICON), Bangalore, India, 2016

[46] Wireless Connectivity Options for IoT Applications – Technology Comparison, 2020. AFANEH, Mohammad. Bluetooth [online]. [cit. 2024-04-11]. Dostupné z: https://www.bluetooth.com/blog/wireless-connectivity-options-for-iot-applications-technology-comparison/

[47] Armenta, Antonio, 2022. Introduction to Arduino: History, Hardware, and Software. *Control Automation* [online]. 8. 9. 2022 [cit. 2024-05-05]. Dostupné z: https://control.com/technical-articles/introduction-to-arduino-history-hardware-and-software/

[48] Arduino [online], 2024. [cit. 2024-05-05]. Dostupné z: https://www.arduino.cc/

[49] Arduino Nano R3, ATmega328 Klon, Připájené piny, 2024. In: Laskakit [online]. [cit. 2024-05-28]. Dostupné z: https://www.laskakit.cz/arduino-nano-r3--atmega328p-klon--pripajene-piny/

[50] Arduino Nano, 2024. Arduino store [online]. [cit. 2024-05-05]. Dostupné z: https://store.arduino.cc/products/arduino-nano

[51] Pinout Diagram Arduino Nano, 2024. In: Store Arduino [online]. [cit. 2024-05-28]. Dostupné z: https://store.arduino.cc/products/arduino-nano

[52] Ai-Thinker RA-02 SX1278 433MHz LoRa modul s adaptérem, 2024. Laskakit [online]. [cit. 2024-05-05]. Dostupné z: https://www.laskakit.cz/ai-thinker-ra-02-sx1278-433mhz-lora-modul-s-adapterem/

[53] Bezdrátový komunikační modul 433MHz SX1278 LoRa RA-02, 2024. In: Hadex [online]. 2011-2024 [cit. 2024-05-28]. Dostupné z: https://www.hadex.cz/m349c-bezdratovy-komunikacni-modul-433mhz-sx1278-lora-ra-02/

[54] Mishra, Sanjeeb, Neeraj Kumar SINGH a Vijayakrishnan Rousseau, 2015. Chapter 10 - Sensor Interfaces. In: System on Chip Interfaces for Low Power Design. Morgan Kaufmann, s. 331-344. ISBN 9780128016305.

[55] Mccauley, Mike, 2024. RadioHead Packet Radio library for embedded microprocessors [online]. [cit. 2024-05-06]. Dostupné z: https://www.airspayce.com/mikem/arduino/RadioHead/

[56] Chen, Yunji, Ling Li, Wei Li, Qi Guo Zidong Du a Zichen Xu, 2024. AI Computing Systems. In: AI Computing Systems [online]. Morgan Kaufmann, Pages 123-166 [cit. 2024-05-07]. ISBN 9780323953993. Dostupné z: https://www.sciencedirect.com/science/article/pii/B978032395399300010X

[57] Arduino Nano 33 BLE Sense, 2024. In: Electronics-Lab [online]. [cit. 2024-05-28]. Dostupné z: https://www.electronics-lab.com/arduino-nano-33-ble-sense/

[58] Nano 33 BLE Sense Rev2 Cheat Sheet, 2024. In: Arduino Docs [online]. [cit. 2024-05-28]. Dostupné z: https://docs.arduino.cc/tutorials/nano-33-ble-sense-rev2/cheat-sheet/

[59] Arduino Nano 33 IoT - with soldered headers, 2024. In: Opencircuit [online]. [cit. 2024-05-28]. Dostupné z: Nano 33 BLE Sense Rev2 Cheat Sheet, 2024. In: Arduino Docs [online]. [cit. 2024-05-28]. Dostupné z: https://docs.arduino.cc/tutorials/nano-33-ble-sense-rev2/cheat-sheet/

[60] Nano 33 IoT Pinout, 2024. In: Arduino Docs [online]. [cit. 2024-05-28]. Dostupné z: https://docs.arduino.cc/hardware/nano-33-iot/

[61] Mega 2560 PRO, ATmega2560-16AU, CH340G, 2024. Laskakit [online]. [cit. 2024-06-11]. Dostupné z: https://www.laskakit.cz/mega-2560-pro-mini--atmega2560-16au--ch340g/

[62] Mega 2560 Pro Mini 5V Development Board for Arduino Mega, 2024. In: Fruugo [online]. [cit. 2024-06-11]. Dostupné z: https://www.fruugo.cz/mega-2560-pro-mini-5v-vyvojova-deska-pro-arduino-mega/p-69028588-138870140

[63] Mega 2560 PRO - handy chart, 2022. In: Arduino Forum [online]. [cit. 2024-06-11]. Dostupné z: https://forum.arduino.cc/t/mega-2560-pro-handy-chart/981446

[64] IIC I2C 5V na 3.3V Obousměrný převodník logické úrovně, 2024. In: Dratek [online]. [cit. 2024-05-28]. Dostupné z: https://dratek.cz/arduino/1481-iic-i2c-5v-na-3.3v-obousmerny-prevodnik-logicke-urovne.html

[65] I2C IIC Osmikanálový, obousměrný konvertor - Modul Logic Level, 2024. In: Dratek [online]. [cit. 2024-06-11]. Dostupné z: https://dratek.cz/arduino/1575-i2c-iic-konvertor-osmikanalovy-obousmerny-modul-logic-level.html

[66] MQTT: The Standard for IoT Messaging [online], 2022. [cit. 2024-05-21]. Dostupné z: https://mqtt.org/

[67] Qubitro IoT Device and Data Platform [online], 2024. [cit. 2024-06-12]. Dostupné z: https://www.qubitro.com/