SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Experimenty s IoT sieťami - Tím LoRa

Študijný program:	Informačné a komunikačné technológie
Školiace pracovisko:	Ústav multimediálnych informačných a komunikačných technológií
Vedúci tímu:	Ing. Tomáš Páleník, PhD.

Akademický rok **2023/2024**

Obsah

Úvod

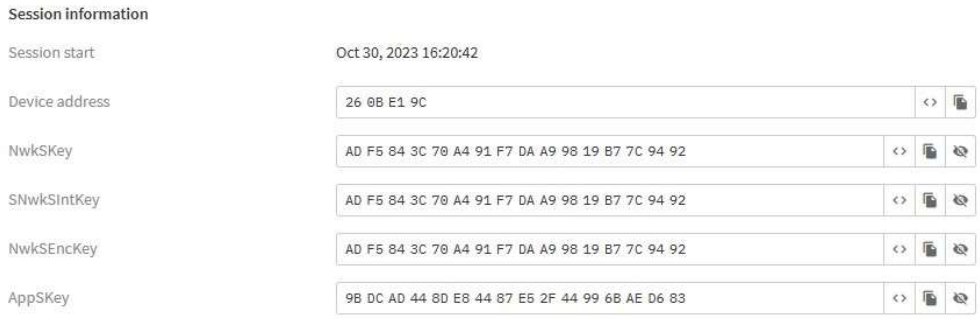
V súčasnej dobe internetu vecí je schopnosť monitorovať a analyzovať dáta z rôznych senzorov kľúčová pre rôzne aplikácie, od inteligentných domácností po priemyselné riešenia. V našom projekte sa zameriavame na vytvorenie systému na odchytávanie a spracovanie packetov vysielaných Arduinom, ku ktorému budú pripojené tri senzory: tepelný, vlhkostný a dymový senzor. Na zabezpečenie efektívneho prenosu dát plánujeme využiť The Things Network (TTN) a Dragino Gateway s RTL anténou. Našim cieľom je vybudovať systém, ktorý dokáže spoľahlivo prenášať dáta zo senzorov pripojených k Arduinu prostredníctvom bezdrôtovej siete TTN a následne tieto dáta zachytávať pomocou Dragino Gateway. Táto konfigurácia nám umožní monitorovať teplotu, vlhkosť a prítomnosť dymu v reálnom čase, čo je nevyhnutné pre široké spektrum aplikácií, vrátane bezpečnostných systémov a optimalizácie prostredia.

1. Technológia LoRa

LoRa (Long Range) je modulačná technika a spolu s LoRaWAN protokolom (Long Range Wide Area Network) tvoria bezdrôtovú technológiu, ktorá je navrhnutá na spoľahlivú a efektívnu komunikáciu v rámci Internetu (IoT). Ich kombinácia umožňuje prenos dát na veľké vzdialenosti s nízkou spotrebou energie a je ideálna pre aplikácie, kde je potrebný prenos na veľké vzdialenosti a dlhá životnosť batérií. LoRa pracuje s veľmi pomalou rýchlosťou prenosu dát a to v rozmedzí od 0,3 kbps do 5,5 kbps. Využíva sa teda prevažne na prepojenie zariadení s veľkou vzdialenosťou ako sú napr. merače teploty, vlhkosti alebo aj GPS [1].

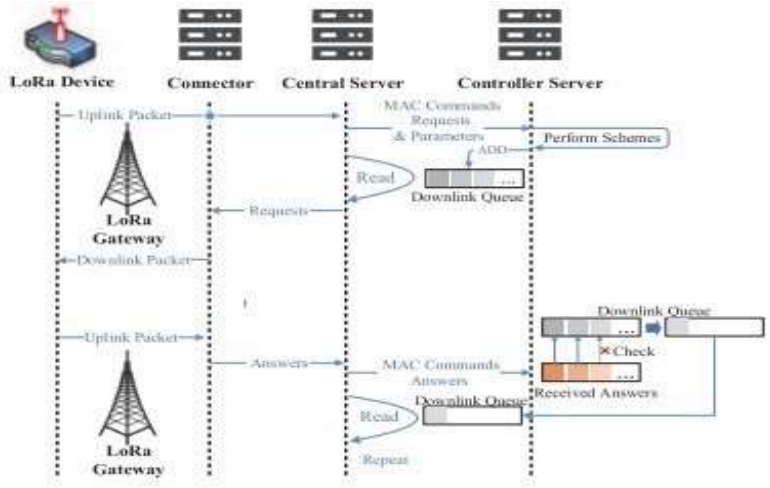
LoRaWAN na svoju prevádzku môže využiť dva typy kľúčov. V našom prípade sme používali OTAA.

1. Pri použití OTAA (Over-the-Air Activation) koncové zariadenie vstupuje do siete počas procesu inicializácie pomocou Join Request. Po odoslaní tejto správy sa zariadenie dočasne pripája k sieti pod vygenerovanými kľúčmi (NwkSKey a AppsKey) a má pridelený identifikátor (DevAddr) od sieťového servera. Výhodou je že OTAA je považovaná za bezpečnejšiu metódu, pretože kľúče nie sú predom definované a môžu byť obnovené počas každého pripojenia alebo zmenené užívateľom. Nevýhodou je že kvôli potrebným overeniam sa celý prenos spomaľuje.



Obr. č. 1 – Naša stránka TTN – informácie o automatický vytvorených kľúčoch

2. Pri použití ABP (Activation by Personalization) sú kľúče a identifikátory zariadenia predom definované a predprogramované do koncového zariadenia. Nemá proces pripojenia, a teda nie je potrebná komunikácia s Join Serverom. Kľúče sú pevne nastavené a nedajú sa zmeniť. ABP má výhodu v tom že pripojenie je rýchlejšie, pretože nevyžaduje komunikáciu s Join Serverom. Môže byť vhodná pre aplikácie, kde je dôvera medzi vopred definovanými kľúčmi. Hlavnou nevýhodou je menšia bezpečnosť, pretože kľúče sú statické a neobnovujú sa. Ak by iný užívateľ dokázal získať informáciu o kľúčoch, tak bezpečnosť dát by bola ohrozená.



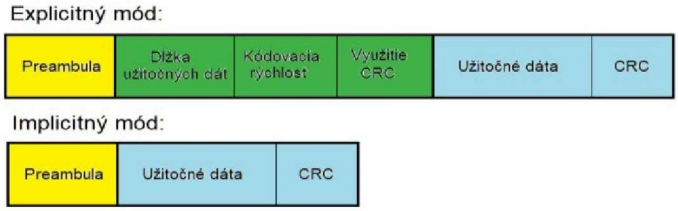
Obr. č. 2 - Praktické zobrazenie komunikácie LoRa a LoRaWAN [1]

Koncové zariadenia – rôzne typy meračov: vlhkosti, teploty alebo GPS posielajú svoje zachytené informácie do brány [1].

Brány – prijímajú a preposielajú informácie ďalej na server. Ich hlavnou úlohou je vykonať následnú kontrolu CRC. CRC je v podstate kontrolný súčet, ktorý sa pripája k dátam pred odoslaním, a pri príjme sa overuje, či dáta boli úspešne prenesené bez chýb. Ak sa vypočítaný CRC zhoduje s prijatým CRC, znamená to, že dáta sú pravdepodobne nepoškodené. V opačnom prípade môžu byť dáta chybné a môže byť vyžadované opätovné odoslanie dát [1].

Server – koncové zariadenia komunikujú s bránami, ktoré odosielajú dáta na Network Server. Network Server implementuje protokol LoRaWAN, overuje pravosť a integritu zariadení, kde duplikuje uplink správy, vyberá brány, ktoré majú byť používané pre downlink správy a odosiela príkazy ADR na optimalizáciu rýchlosti prenosu dát zariadení. Taktiež zabezpečuje kontrolu adresy zariadenia a predbežné overenie rámcov. Následne poskytuje spojenie medzi koncovým zariadením a Application Serverom, kde sa dáta spracúvajú podľa konkrétnych potrieb a aplikácií [1].

Join server - zodpovedný za proces pripojenia nových koncových zariadení do siete, generuje potrebné kľúče a identifikátory pre nové koncové zariadenia. Rieši rámcove žiadosti o pripojenie uplink kanálu. Generuje rámce pre pripojenia downlinku kanálu, čím zadáva informáciu sieťovému serveru, ktorý aplikačný server má byť pripojený ku danému koncovému zariadeniu [1].



Obr. č. 3 – Typy módov paketu LoRa [39]

V implicitnom režime sa z paketu odstráni celá hlavička, pričom užitočné dáta a rýchlosť kódu sú pevne stanovené. Tento mód zvyčajne používajú typy zariadení triedy B na odosielanie informácií o časovej synchronizácii z brán do jedného alebo viacerých koncových zariadení [3].

Pre technológiu LoRa existujú dva rozdielne open source protokoly:

1. Symphony Link™, určený pre priemyselných a podnikových používateľov, ktorí potrebujú pokročilé funkcie.

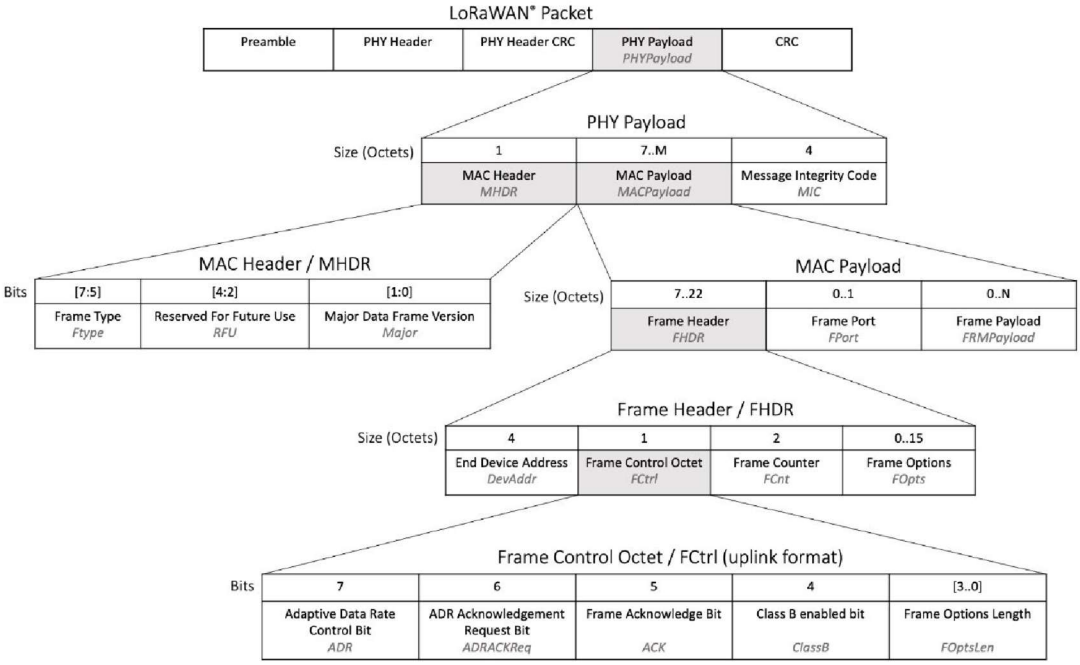
2. LoRaWAN™, určený pre mobilné siete založené na LoRaWAN, predovšetkým v Európe.

EÚ spektrum	863-870 MHz
Dosah	5-15 km
Šírka kanálu	125 kHz
Max.rádiové straty	-164,4 dB
Max.rádiové straty DL	-154,1 dB
Rýchlosť UL v EÚ	0,3 - 10 kbps
Rýchlosť DL v EÚ	0,3 - 10 kbps
UL správ za deň	neobmedzene
DL správ za deň	neobmedzene
Max. veľkosť UL dát	neobmedzene
Max. veľkosť DL dát 8 B	neobmedzene
Obojsmerný prenos	áno
Implementácia	Open-source

Tab. č. 1 – Špecifikácia LoRa [2]

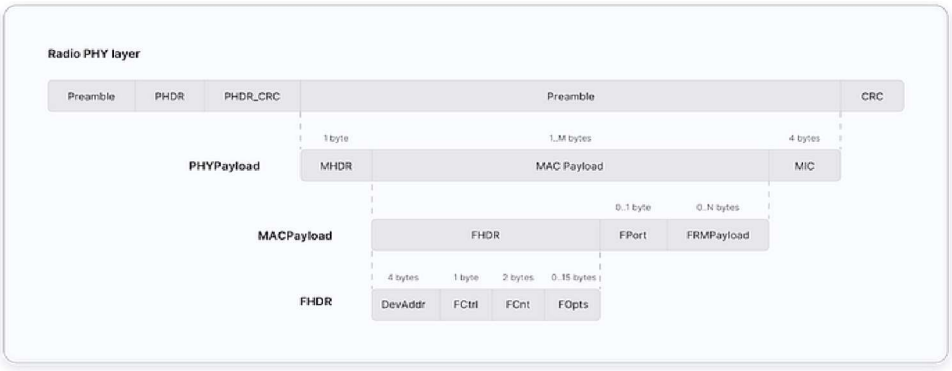
1.1 Explicitná štruktúra paketu LoRa

Explicitný režim, sme využili v našej komunikácií. Z patentu LoRa by hlavička mala byť vždy kódovaná pomocou najnižšej rýchlosti kódu (4/8 = ½), aby sa využili všetky možnosti opravy chýb hammingovho kódu. Cieľom je zvýšiť šancu na správne dekódovanie obsahu hlavičky, ktorá sa skladá z 3 hlavných častí: PHY Header, PHY Header CRC, PHY Payload [4].



Obr. č. 4 – Kompletná štruktúra paketu LoRa uplink – downlink [4]

Preambula - každý paket LoRa sa začína preambulou, ktorá slúži na detekciu prenosu, synchronizáciu rámca a synchronizáciu frekvencie. Musí pozostávať z 8 symbolov pre všetky regióny, ako sa uvádza v dokumente *LoRaWAN Regional Parameters*. Avšak k rádiovému vysielaču môže užívateľ pridať proces modulových upchirpov, pri ktorom frekvencia signálu postupne stúpa s časom. To znamená, že začiatok signálu má nižšiu frekvenciu a postupne stúpa. Upchirp sa často používa pri vysielaní dát z koncového zariadenia (end device) k bráne (gateway) alebo sieťovému serveru. Downchirp je opak, pri ktorom frekvencia signálu postupne klesá s časom. To znamená, že začiatok signálu má vyššiu frekvenciu a postupne klesá. Downchirp sa často používa pri vysielaní dát z brány alebo sieťového servera ku koncovému zariadeniu [6]. V našom prípade ako je možné vidieť na obrázku č. 20, sme využívali základňu stavbu preambuly 8.



Obr. č. 5 – Časti LoRa paketu, ktoré prakticky zachytávame v našej komunikácií - v kapitole 6.4 [3]

1. PHDR a PHDR_CRC

PHDR (Physical Header) je voliteľný prvok len v explicitnom režime, ktorý prenáša informácie ohľadom veľkosti poľa PHY Payload a CRC (Cyclic Redundancy Check). PHDR_CRC (Header CRC) je voliteľné pole, ktoré obsahuje kód na detekciu chýb a opravu chýb v hlavičke [3].

2. Payload PHY

Je kódovaný jednou z kódovacích rýchlostí (4/5, 4/6, 4/7 alebo 4/8). Kompletný rámec sa odošle pomocou jedného z faktorov šírenia SF (7 až 12) [3]. V našom prípade sme využili kódovú rýchlosť 4/5 a SF 7 viď obrázok č. 20. Maximálna veľkosť užitočných dát sa líši od DR (Data Rate), táto hodnota je špecifická pre daný región. V našom prípade pre Európu využívame šírku pásma 125kHz a naša frekvencia kanála je 868,10 MHz viď obrázok č. 20. Pre tieto parametre platí rozmedzie DR0-DR5 [39].

Hodnota	Prenosová rýchlosť (bit/s)
DR0	250
DR1	440
DR2	980
DR3	1760
DR4	3125
DR5	5470
DR6	11000
DR7	50000

Tab. č. 2 – Data range a jeho príslušné rýchlosti [39]

4. MHDR (záhlavie MAC): definovanie informácií o type správy a jej verzii formátu.

5. DevAddr: Je súčasťou paketu Frame Header. 32-bitové jedinečné ID zariadenia v sieti, ktoré prideľuje sieťový server v metóde aktivácie OTAA. V prípade aktivácie ABP je globálne jedinečné pevne zakódované v koncovom zariadení pri výrobe [3].

6. **FCtrl** (Frame Control): Patrí pod záhlavie rámca s veľkosťou 1 bajt. Kontrolné pole, ktoré nesie informácie o type rámca, potvrdení a bezpečnostných nastaveniach [3].

7. **FCnt** (Frame Counter): Patrí do záhlavia rámca s veľkosťou 2 bajty. Počítadlo, ktoré sleduje počet uplink rámcov odoslaných koncovým zariadením [3].

8. **FOpts** (Počet rámcov): FOpts (počet rámcov): Patrí pod veľkosť záhlavia rámca 2 bajty. Príkazy MAC sa môžu zdieľať v poli FOpts dátovej správy na odoslanie. Celková dĺžka príkazov MAC nesmie presiahnuť 15 bajtov [3].

9. **FPort** (Frame Port): Nepovinné pole používané na identifikáciu portu - proces presmerovania dát na koncovom zariadení na základe identifikátora aplikačného portu. Aplikačné porty umožňujú viacero služieb alebo senzorov súčasne používať jedno koncové zariadenie a jednu bránu. Keď koncové zariadenie posiela dáta do siete LoRaWAN, aplikačný port (FPort) sa používa na identifikáciu, na ktorý logický port alebo kanál by mali byť dáta doručené. Hodnota FPort 0 znamená, že pole FRMPayload obsahuje iba príkazy MAC. Hodnota FPort 1-223 znamená, že pole FRMPayload obsahuje aplikačné údaje [3].

FPort Value	Description
0	MAC commands only
1 to 223	Application-specific data
224	LoRaWAN MAC layer test protocol
255	Reserved for Future Use (RFU)

Obr. č. 7 – Význam hodnôt Frame portu [3]

10. **FRMPayload** (Frame Payload): Údaje špecifické pre aplikáciu alebo riadiace správy zapuzdrené v rámci, tieto údaje su zašifrované cez AppsKey [3].

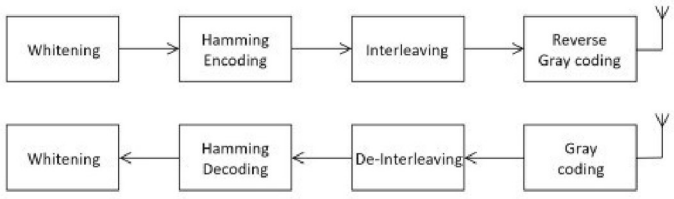
11. **MIC** (Message Integrity Code): 32-bitová hodnota zabezpečuje integritu a autentickosť správy. Kód integrity správy sa vypočíta zo všetkých polí v správe a potom sa pridá k samotnej správe. V nasledujúcom zozname je uvedené, ktoré polia sa používajú na výpočet MIC pre jednotlivé typy správ [3].

LoRaWAN version	Message Type	Fields
1.0.x	Join-request	MHDR | AppEUI | DevEUI | DevNonce
1.0.x	Join-accept	MHDR | AppNonce | NetID | DevAddr | DLSettings | RxDelay | CFList
1.0.x	Data messages (up and down)	MHDR | FHDR | FPort | FRMPayload
1.1	Join-request	MHDR | JoinEUI | DevEUI | DevNonce
1.1	Join-accept	MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay | CFList
1.1	Rejoin-request Type 0 and 2	MHDR | Rejoin Type | NetID | DevEUI | RJcount0
1.1	Rejoin-request Type 1	MHDR | Rejoin Type | JoinEUI | DevEUI | RJcount1
1.1	Data messages (up and down)	MHDR | FHDR | FPort | FRMPayload

Obr. č. 6 – Výpočet MIC pre rôzne LoRaWan verzie [3]

12. CRC: je to nepovinné pole, ktoré obsahuje kód na detekciu chýb na opravu chýb v payload správy uplinku. Je to zakódované pomocou jednej z kódovacích rýchlostí (4/5, 4/6, 4/7 alebo 4/8) [3]. Downlink správa neobsahuje CRC. V našom prípade pole CRC neevidujeme v prijatých uplinkových správach.

Proces kódovania a dekódovania dátových bitov sa vykonáva v štyroch krokoch:



Obr. č. 7 – Proces kódovania a dekódovania [5]

Gray Coding (Grayov kód):

Grayov kód alebo zrkadlový binárny kód je cyklický kód, v ktorom každé ďalšie kódové slovo sa líši od predchádzajúceho len v jednom bite a posledné slovo kódu sa líši od prvého tiež len v jednom bite. Používa sa v telekomunikáciách práve kvôli vlastnosti rozdielu vedľajších slov len v jednom bite. Pri technológií LoRa je Grayov kód používaný pre kódovanie hodnôt frekvencie alebo iných parametrov. Jedným z dôvodov je minimalizácia pravdepodobnosti chýb pri prenose signálu, najmä pri prevádzke v rušnom prostredí alebo pri prenose na dlhé vzdialenosti. Vo frekvenčnej modulácii, ktorá sa často používa v LoRa, môže byť použité Grayovo kódovanie pre mapovanie hodnôt binárneho kódu na frekvenciu. Keďže susedné hodnoty v Grayovom kóde sa líšia len v jednom bite, zmena frekvencie je plynulá a minimalizuje to riziko chýb pri dekódovaní hodnôt [5].

Binárny kód: 000 001 010 011 100 101 110 111

Grayov kód: 000 001 011 010 110 111 101 100

Interleaving (Prekladanie):

Počas prenosu môže byť symbol poškodený šumom alebo blednutím, čo vedie k viacnásobným bitovým chybám. Blednutie (fading) je jav, pri ktorom dochádza k náhlym a nepravidelným zmenám v sile signálu alebo k jeho oslabovaniu. Tento jav môže byť spôsobený rôznymi faktormi, ako sú prekážky v signálovom prostredí, interferencie, alebo odraz signálu od okolitých objektov. Keďže všetky chyby sú spôsobené jedným symbolom, sú vysoko korelované. Keď hovoríme o tom, že bity sú korelované, znamená to, že existuje nejaká matematická závislosť medzi jednotlivými bitmi v dátovej postupnosti. Korelácia môže byť pozitívna alebo negatívna. Pozitívna korelácia znamená, že existuje tendencia, že keď je jeden bit vo svojej hodnote vysoký (1), pravdepodobnosť, že ďalší bit bude tiež vysoký, je vyššia, a naopak. To znamená, že bity sú v nejakej miere korelované v smere rastúcej alebo klesajúcej postupnosti. Negatívna korelácia znamená, že existuje závislosť, že keď je jeden bit vo svojej hodnote vysoký (1), pravdepodobnosť, že ďalší bit bude nízky (0), je vyššia, a naopak. Bity sú teda korelované v opačnom smere.

Cieľom je narušiť túto koreláciu medzi chybným bitom a to rozdelením jednotlivých chýb do viacero kódových slov. Táto operácia zvyšuje účinnosť kódu na opravu chýb za cenu väčšieho oneskorenia, pretože je nutné prijať viacero symbolov skôr ako je možné obnoviť jedno úplné kódové slovo [5].

Whitening:

Aplikuje operáciu XOR na informačné bity. Odstraňuje sa ním odchýlka jednosmernej zložky (DC). Whitening v LoRa znamená aplikovanie pseudonáhodnej sekvencie bitov na pôvodné dáta pred odoslaním. Táto pseudonáhodná sekvencia má vlastnosť, že má rovnomerne distribuované hodnoty bitov, čím redukuje koreláciu medzi jednotlivými bitmi v pôvodných dátach [5]

Hamming:

Hammingovo kódovanie pracuje s dodatočnými (paritnými) bitmi, ktoré sa pridávajú k pôvodným dátam. Tieto paritné bity sa vypočítavajú na základe hodnôt pôvodných bitov a poskytujú informácie o paritnej (či nepárnej) početnosti jednotiek v určitých skupinách bitov. Hammingov kód je navrhnutý tak, aby mohol detegovať a opraviť jednotlivé chyby v prenesených dátach. Ak sa počas prenosu vyskytne chyba, Hammingov kód umožňuje lokalizovať a opraviť túto chybu. [5].

1.2 LoRaWAN

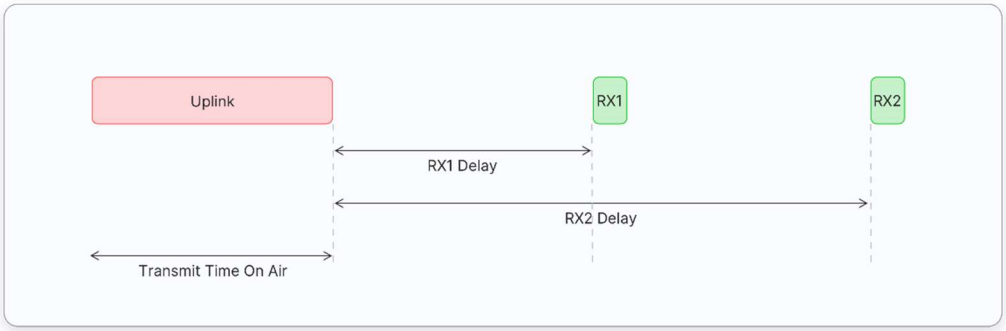
LoRaWAN (Long Range Wide Area Network) je protokolová špecifikácia postavená na základe technológie LoRa vyvinutej LoRa Alliance. Využíva nelicencované rádiové spektrum v priemyselných, vedeckých a lekárskych ISM (industry, scientific, medical) pásmach, ktoré umožňujú komunikáciu medzi diaľkovými snímačmi a bránami pripojenými k sieti s nízkym výkonom. Špecifikácia je k dispozícii zadarmo na stiahnutie z webovej stránky LoRa Alliance (www.lora-alliance.org) [2].

Siete LoRaWAN sa zvyčajne nasadzujú v topológii hviezda - hviezda, v ktorej brány prenášajú správy medzi koncovými zariadeniami a centrálnym sieťovým serverom. Brány sú pripojené k sieťovému serveru prostredníctvom štandardných pripojení IP, zatiaľ čo koncové zariadenia používajú jednorazovú komunikáciu LoRa™ alebo FSK (frequency shift keying) pre jednu alebo viac brán. Všetka komunikácia je zvyčajne obojsmerná, aj keď sa predpokladá, že prepojenie "uplink" od koncového zariadenia k sieťovému serveru bude prevládajúcou cestou. Komunikácia End-to-Gateway sa zakladá na rôznych frekvenciách kanálov a dát [2].

LoRaWAN má v skutočnosti 3-MAC triedy, ktoré pokrývajú širokú škálu aplikácií [6]:

Trieda A

Všetky koncové zariadenia LoRaWAN musia podporovať implementáciu triedy A. Zariadenie triedy A môže kedykoľvek odoslať správu uplink. Po ukončení vysielania uplink kanála zariadenie otvorí dve krátke prijímacie okná na prijímanie downlink správ zo siete. Medzi ukončením prenosu uplink spojenia a začiatkom každého prijímacieho okna je oneskorenie, známe ako oneskorenie RX1, resp. oneskorenie RX2. Ak sieťový server neodpovie počas týchto dvoch prijímacích okien, ďalšie downlink spojenie sa naplánuje hneď po ďalšom vysielaní uplink spojenia.

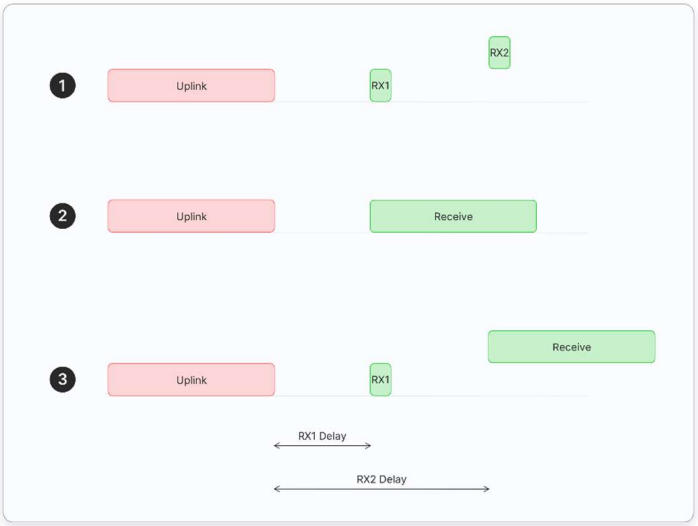


Obr. č. 8 – Komunikácia MAC triedy typu A [6]

1. Koncové zariadenie otvorí obidve okná príjmu, ale počas žiadneho z nich neprijme správu o downlink.

2. Koncové zariadenie prijíma uplink spojenie počas prvého prijímacieho okna, a preto neotvára druhé prijímacie okno.

3. Koncové zariadenie otvorí prvé prijímacie okno, ale neprijme downlink odkaz. Preto otvorí druhé prijímacie okno a počas druhého prijímacieho okna prijme downlink odkaz.



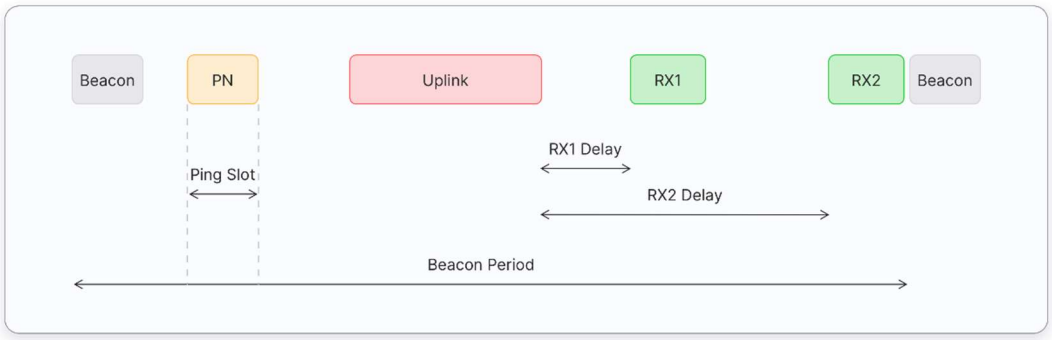
Obr. č. 9 – Typy odpovedí servera pre triedu typy A [6]

Trieda B

Zariadenia triedy B rozširujú možnosti triedy A tým, že pravidelne otvárajú prijímacie okná nazývané ping sloty na prijímanie downlink správ. Sieť periodicky vysiela časovo synchronizované signály (unicast a multicast) prostredníctvom brán, ktoré prijímajú koncové zariadenia. “Beacons” (zariadenie, ktoré dokáže prijímať a odosielať informácie cez novšiu verziu Bluetooth technológie (BLE = Bluetooth low energy)), poskytujú časovú referenciu pre koncové zariadenia, ktorá im umožňuje zladiť ich vnútorné časovanie so sieťou. To umožňuje sieťovému serveru zistiť, kedy má poslať downlink signál konkrétnemu zariadeniu alebo skupine zariadení. Čas medzi dvoma “beaconami” sa nazýva perióda “beaconu”. Po uplinku sa otvoria dve krátke prijímacie okná, RX1 a RX2, podobne ako v prípade zariadení triedy A.

Koncové zariadenia triedy B majú v porovnaní s koncovými zariadeniami triedy A nízke oneskorenie pri downlink linkách, pretože pravidelne otvárajú sloty ping. Majú však oveľa vyššiu latenciu ako koncové zariadenia triedy C. Koncové zariadenia triedy B sú často napájané z batérie. Životnosť batérie je v triede B kratšia v porovnaní s triedou A, pretože zariadenia trávia viac času v aktívnom režime kvôli prijímaniu signálov a otvoreným slotom na ping. Vzhľadom na nízku latenciu pre downlinky sa režim triedy B môže

používať v zariadeniach, ktoré vyžadujú stredne kritickú aktiváciu, ako sú napríklad merače spotreby.

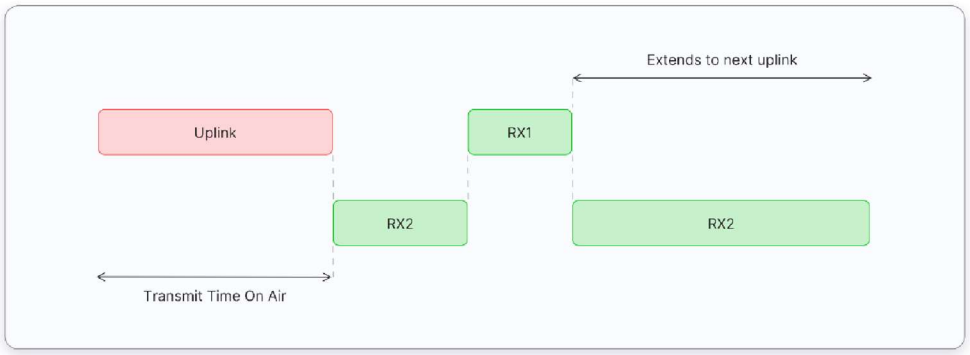


Obr. č. 10 – Komunikácia MAC triedy B [6]

Trieda C

Zariadenia triedy C rozširujú možnosti triedy A tým, že nechávajú otvorené okná príjmu, pokiaľ nevyšlú ďalší uplink. Zariadenia triedy C preto môžu prijímať downlink správy takmer kedykoľvek, čím majú veľmi nízke oneskorenie pri downlinoch. Tieto správy downlinku sa môžu použiť na aktiváciu zariadenia, napríklad na zníženie jasu pouličného osvetlenia alebo zapnutie uzatváracieho ventilu vodomeru.

Zariadenia triedy C otvárajú dve prijímacie okná, RX1 a RX2, podobne ako v triede A. Prijímacie okno RX2 však zostáva otvorené až do ďalšieho vysielania uplinku. Po odoslaní uplink spojenia zariadením sa otvorí krátke prijímacie okno RX2, po ktorom nasleduje krátke prijímacie okno RX1 a potom sa otvorí nepretržité prijímacie okno RX2. Toto prijímacie okno RX2 zostáva otvorené až do naplánovania ďalšieho uplinku. Uplinky sa odosielajú, keď neprebieha žiadny downlink.

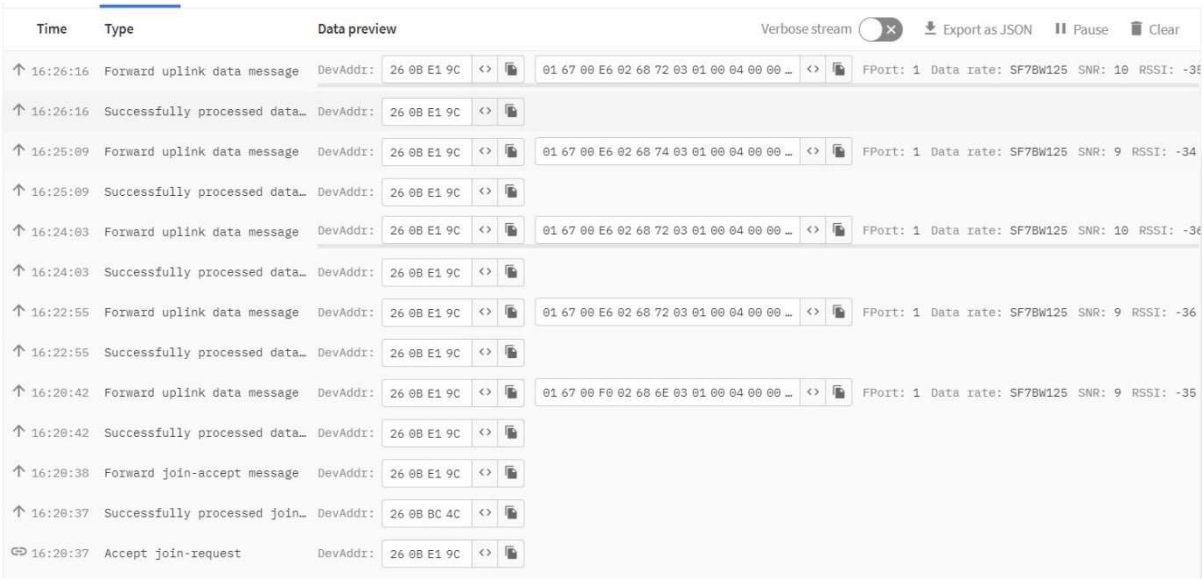


Obr. č. 11 – Komunikácia MAC triedy C [6]

Na správu siete medzi serverom a koncovým zariadením sa používajú príkazy MAC. Tieto príkazy nie sú viditeľné pre aplikácie bežiace na serveri LoRa a koncových zariadeniach. Jednoduchý dátový rámec sa skladá z jedného alebo viacerých MAC príkazov. Tieto príkazy používa koncové zariadenie alebo brána, prípadne oboje [40].

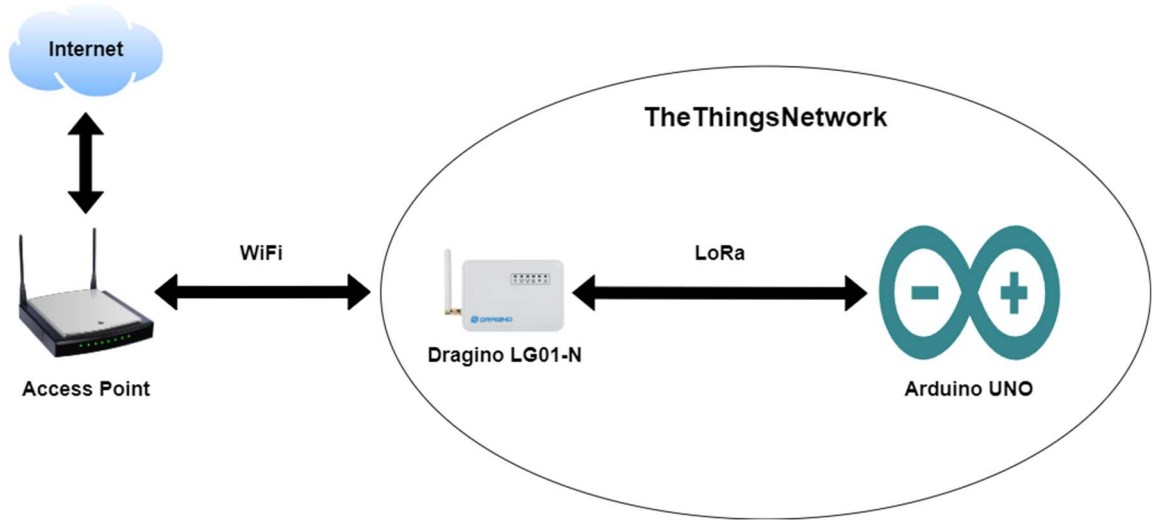
LoRaWAN MAC rozlišuje šesť rôznych typov správ pre vytvorenie komunikácie medzi koncovým zariadením a serverom [40]:

1. “Join Request” – uplink informácia - požiadavka o pripojenie (smeruje od koncového zariadenia k serveru) - používa sa pri aktivácií OTAA.
2. “Join Accept” – downlink informácia (súhlas sieťového servera s pripojením koncového zariadenia) - používa sa pri aktivácií OTAA.
3. “Confirmed Data Up” - dátový rámec pre uplink informácie – vyžaduje potvrdenie správy.
4. “Confirmed Data Down” – dátový rámec pre downlink informácie – vyžaduje potrvdenie správy.
5. “Unconfirmed Data Up” – dátový rámec pre uplink informácie - nevyžaduje potvrdenie správy.
6. “Unconfirmed Data Down” – dátový rámec pre downlink informácie – nevyžaduje potvrdenie správy.



Obr. č. 12 – Praktické zachytenie Join Requestu a začatie komunikácie nášho zapojenia

2. Kľúče a identifikátory LoRa



Obr. č. 13 – Schéma zapojenia

AppEUI - Toto je 64-bitový identifikátor aplikácie, ktorý je tiež jedinečný. Keď vytvoríte aplikáciu, server účtu The Things Network priradí AppEUI z adresového bloku nadácie The Things Network. Je to ako číslo portu v sieti Ethernet alebo Wi-Fi[15].V našej implementácií používame tento kľúč zapísaný v hexadecimálnom formáte: 0x50, 0x41, 0xD0, 0x0B, 0x1D, 0x41, 0x40, 0xA8.

DevEUI - Toto je 64-bitový identifikátor koncového zariadenia, ktorý je jedinečný a dodáva ho výrobca zariadenia. DevEUI môže byť generovaný z interných registrov ID čipu LoRa. Je to ako MAC adresa v sieti Ethernet alebo Wi-Fi[14,15]. Naše DEVEUI zapísané v skripte ako môžeme vidieť v prílohe č. :

0xAF, 0x1F, 0x06, 0xD0, 0x7E, 0xD5, 0xB3, 0x70. Znova sa kľúč zapisuje v hexadecimálnom formáte.

GatewayEUI - Toto je 64-bitový identifikátor brány, ktorý je tiež jedinečný. Brány sú vyrobené s vstavaným EUI, ktorý sa potom používa na registráciu brány v sieti The Things Network[13]. Tento identifikátor sa nachádza spravidla na zariadení, alebo krabici a je nemenný. Naše GatewayEUI je: A8 40 41 1D 0B D0 41 50.

Gateway ID - Toto je unikátny identifikátor, ktorý sa generuje pre LoRa Gateway. Je to tiež známe ako Gateway EUI a je nevyhnutné pre registráciu brány v akomkoľvek LoRa Network Server (TTN, LoRaServer) [15]. Náš identifikátor je: eui-a840411d0bd04150new.

DevAddr - Toto je 32-bitová adresa zariadenia, ktorá nie je unikátna. Táto adresa je priradená zariadeniu počas aktivácie [13]. Tento údaje je nakonfigurovaný vo webovom rozhraní TTN. DevAddr: 26 0B 85 38.

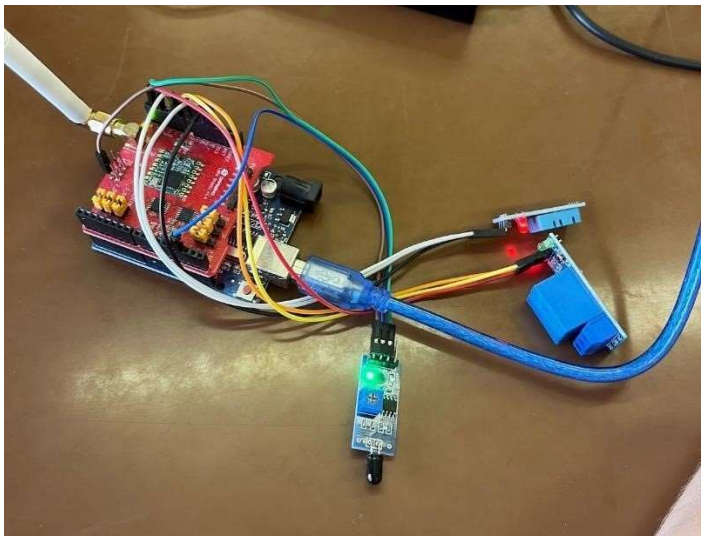
APPKEY - Toto je kľúč šifrovania, ktorý sa používa pre správy počas každej aktivácie cez vzduch. Po aktivácii sa používa AppSKey. Posluchač, ktorý pozná AppKey, môže odvodiť AppSKeyg [14]. Tento identifikátor je uložený taktiež v prostredí TTN v decimálnom formáte.

Náš identifikátor je: 41 F7 AE 05 69 CA 0D E5 BC 26 11 6B 92 30 FD 27.

Tieto identifikátory sú dôležité pre fungovanie siete LoRaWAN a umožňujú jednoznačnú identifikáciu a komunikáciu medzi zariadeniami, aplikáciami a bránami. Súčasťou procesu aktivácie je priradenie dynamických adries zariadeniam, čo sa nazýva aktivácia. Aktivácia vzduchom OTAA (Over The Air Authentication) je preferovaný a najbezpečnejší spôsob pripojenia k sieti The Things Network [13,14,15].

3. Arduino a komponenty

Zariadenie, ktoré sme počas práce na tomto tímovom projekte snažili spárovať s LoRa Gateway – LG01 je Arduino Uno spolu s LoRa Shield, Flame Sensor, Temperature a Humidity Sensor a Buzzer, všetky tieto časti boli súčasťou Single Channel LoRa IoT Kit v2 balenia [18].



Obr. č. 14 – Arduino Uno s LoRa Shield a senzormi Temperature&Humidity, Buzzer a Flame

Toto zariadenie sme sa snažili prepojiť s Dragino Gateway, za pomoci webového rozhrania TheThingsNetwork. Dragino Gateway bol rovnako prepojený s TTN a našim cieľom bolo teda preposielanie paketov s informáciami o teplote a vlhkosti zo senzorov.

Čo sa týka senzoru ohňa, ten sme z bezpečnostných dôvodov radšej netestovali, namiesto toho sme len v kóde zaviedli možnosť, kde môžeme umelo upraviť stav pinu a teda otestovať jeho funkčnosť takýmto spôsobom.

4. Arduino IDE a prerekvizity

Na tímovom projekte sme pre naše zadanie využívali vývojové prostredie Arduino IDE, ktoré je voľne dostupné na stránke Arduino [34]. Verzia IDE s ktorou sme pracovali, počas projektu je verzia 2.2.2. Toto IDE bolo nainštalované na operačnom systéme Windows 11 verzii 23H1. Pomocou IDE sme boli schopný nakonfigurovať microcontroller Arduino s ktorým sme pracovali a taktiež aj s pripojenými komponentami.

Arduino IDE slúži primárne na to, aby sme v ňom boli schopný napísať zdrojový kód, ktorý sme po pripojení samotného Arduina nahrali do jeho pamäte a spustili.

Na to aby sme vedeli pripojiť samotné Arduino do nášho zariadenia, bolo najprv potrebná inštalácia CH340 ovládača. Po úspešnej inštalácii tohto ovládača sme následne mohli vidieť v Device Manager zariadenie USB-SERIAL CH340 [18].

Následne po nainštalovaní ovládača sme boli schopný detekovať Arduino UNO v samotnom IDE, kde sme si ho zvolili. Na otestovanie funkčnosti pripojenia sme najprv spravili sériu jednoduchých testov, ako napríklad nastavenie blikania LED diódy. Po úspešnom overení funkčnosti a správneho fungovania sme teda mohli prejsť k ďalšiemu kroku, ktorým bola inštalácia knižníc a príprava kódu, na spárovanie zaradenia s TTN a Dragino Gateway.

Obr. č. 15 – Zvolenie Arduino zariadenie v Arduino IDE

```
void setup() {}
pinMode(LED_BUILTIN, OUTPUT); void
loop()
{ digitalWrite(LED_BUILTIN, HIGH);
     delay(100);
     digitalWrite(LED_BUILTIN, LOW);
     delay(100);
}
```

Príklad jednoduchého skriptu na blikanie LED diódy, ktorý sme zobrali z predpripravených ukážok v Arduino IDE. Po nahratí a spustení tohto kódu nám LED-ka začala preblikávať na Arduino doske v nastavenom intervale, čím sme potvrdili jej funkčnosť.

4.1 Potrebné knižnice

Na to aby sme mohli do Arduina nahrať kód, musíme predtým zabezpečiť prítomnosť nasledovných knižníc, ktoré sú potrebné pre komunikáciu nášho zariadenia s TTN a samotnou Dragino Gateway.

Arduino-LMIC

LMIC je skratka LoRaWAN-MAC-in-C, čo je základná knižnica na prácu s LoRaWAN v Arduino IDE. Táto knižnica nám umožnila prácu s bezdrôtovou komunikáciou tohto protokolu. Knižnica je open-source, teda dostupná bez obmedzení [35].

LoRa-raw

Knižnica LoRa-raw nám slúžila na to, aby sme boli schopný prenášať a v prípade potreby aj prijímať dáta pomocou nášho zariadenia. Taktiež dodatočne môže slúžiť aj na šifrovanie alebo ovládanie toku dát. Pomocou tejto knižnice je taktiež možné aj upravovať LoRa protokol, avšak tomu sme sa v našom projekte nevenovali. Rovnako ako v prípade LMIC je knižnica voľne dostupná a open-source [36].

DHTlib

Poslednou dôležitou knižnicou pre náš projekt je knižnica DHTlib, mocou ktorej sme schopný využívať funkcionality komponentov na meranie teploty a vlhkosti. Knižnica je ako v predošlých prípadoch voľne dostupná a open-source, čiže jej získanie nerobilo veľké problémy [37].

Pri knižniciach sme ale narazili na problém s kompatibilitou verzií, ktoré nám boli odporúčané v dokumentácii samotného Dragino kitu a najnovším Arduino IDE, bolo preto potrebné aby sme LMIC knižnicu aktualizovali, pomocou library managera v samotnom Arduino IDE. Po tejto aktualizácii už spustenie skriptu fungovalo v poriadku a boli sme schopný získať nami požadované údaje.

4.2 Zdrojový kód

Ako základ zdrojového kódu sme v Arduino IDE použili voľnedostupný kód dostupný z GitHub-u na linku [38].

Tento kód bol odporúčaný aj v príručke samotného Dragino Kitu [18] s ktorým sme pracovali a pre naše účely potreboval len pár zmien a pridaných hodnôt.

Najprv bolo potrebné aby sme zadefinovali jednotlivé AppEUI, DevEUI a AppKey, ktoré sme našej aplikácii zadali na stránke TTN.

```
  static const u1_t PROGMEM APPEUI[8]={ 0x50, 0x41, 0xD0, 0x0B, 0x1D, 0x41, 0x40, 0xA8 };

  static const u1_t PROGMEM DEVEUI[8]={ 0xAF, 0x1F, 0x06, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };

  static const u1_t PROGMEM APPKEY[16] ={ 0x6A, 0xAF, 0x39, 0xB7, 0x15, 0x78, 0x2D, 0xF1, 0x1F, 0x1F, 0xAE, 0xCD, 0x64, 0x23, 0x01, 0x07 };
```

Tieto hodnoty bolo po testovaní zadávať vo forme LSB *(teda Least Significant Bit)* , TTN konzola nám dovoľuje tieto indentifikátory ukazovať v rôznych variantoch ako Least Significant Bit, Most Significant Bit, to takže získanie týchto hodnôt nebol problém. `const unsigned TX_INTERVAL = 60;`

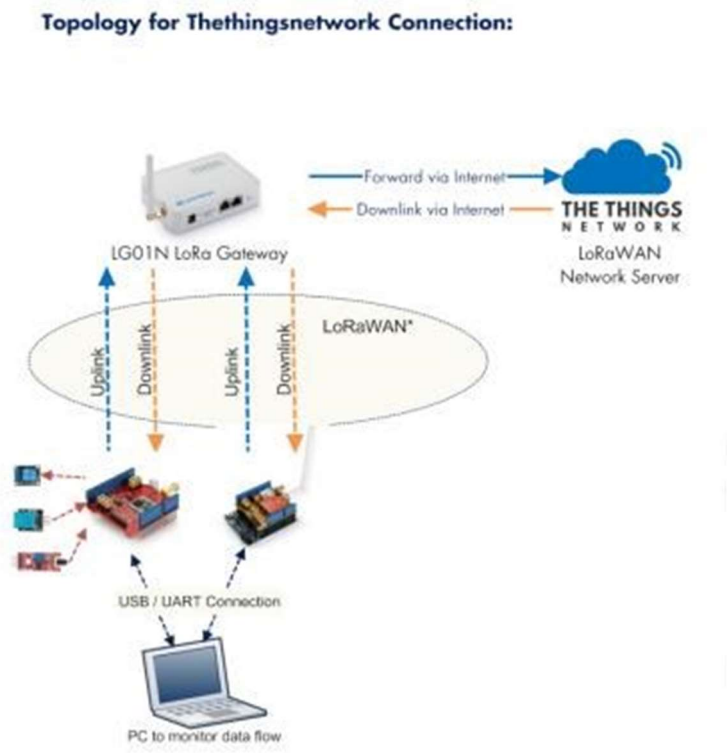
Pomocou parametru TX_INTERVAL sme boli schopný nastaviť interval odosielania sa jednotlivých paketov pomocou Arduina, tu je dôležité poznamenať, že táto hodnota v konečnom dôsledku nemusí byť presne dodržaná, kvôli povahe samotného LoRa protokolu.

```
void dhtTem()
{         int16_t tem_LPP;
temperature = DHT.read11(DHT11_PIN);
tem =
DHT.temperature*1.0;              humidity
= DHT.read11(DHT11_PIN);              hum
= DHT.humidity* 1.0;
Serial.print("###########    ");
       Serial.print("COUNT=");
       Serial.print(count);
       Serial.println("    ###########");
       Serial.println(F("The temperature and humidity:"));
       Serial.print("[");
       Serial.print(tem);
       Serial.print("℃");
       Serial.print(",");
       Serial.print(hum);
       Serial.print("%");
       Serial.print("]");
Serial.println("");
count++;         tem_LPP=tem
* 10;          LPP_data[2] =
tem_LPP>>8;
       LPP_data[3] = tem_LPP;
       LPP_data[6] = hum * 2; }
```

Následne nás zaujíma funkcia dhtTem, kde z pinov čítame teplotu a vlhkosť samotných zariadení a vypisujeme ju do konzoly, tieto dáta sa nachádzajú v poli LPP_data, kde ich reprezentujú indexy 2,3 pre teplotu a 6 pre vlhkosť, pole s indexom 4 reprezentuje stav buzzer-a a index 3 reprezentuje stav flame senzoru. V prípade ak by sme chceli zistiť informáciu o tom, či flame senzor eviduje prítomnosť otvoreného ohňa, museli by sme si zadefinovať novú funkciu, kde by sme museli čítať pomocou analogRead výstup z pripojeného pinu, a zadať si výpisy pre jednotlivé stavy, čo by mohli nastať. V tomto prípade by sa jednalo o 3 možnosti:

1. Žiadny oheň sa v blízkosti nenachádza
2. Oheň sa nachádza v blízkosti do 50cm
3. Oheň sa nachádza bližšie ako 50cm

Potom by už len daný read stačilo naformulovať na výpis a poslať. Aby sme dokázali spárovať samotné Arduino spoločne s TheThingNetwork, bolo potrebné najprv spojazdniť samotný Dragino Gateway a získať jeho unikátne ID, ktoré mu bolo pridelené.



Obr. č. 16 – Schéma prepojenia Arduina s TTN

Ako prvé bolo potrebné vytvorenie samotného zariadenia na stránke TTN. Na vytvorenie sme vytvorili nasledovný postup, ktorý sme si následne aj znovu odskúšali aby sme overili jeho funkčnosť.

1. Vstúpime na stránku https://eu1.cloud.thethings.network/console/
2. Zvolíme v pravom hornom rohu +Create Application
3. Vyplníme potrebné formality a dáme Create Application
4. Následne zvolíme v ľavej lište End Devices a Register end device
5. End device ID sa nám automaticky vygeneruje (náš prípad) alebo si ho môžeme ľubovoľne pomenovať
6. Nastavíme nasledovné parametre:
   - **Frequency Plan** – Europe 863-870MHz
   - **LoRaWAN version** – LoRaWAN Specification 1.0.3
   - **Regional Parameters version** – RP001 Regional Parameters 1.0.3 revision A
   - **LoRaWAN class capabilities** – Supports class B
   - **Activation mode** – OTAA (Over The Air Activation)
7. Následne nastavíme Activation information
   - **AppEUI** – tu nastavíme EUI, ktoré je zhodné s Dragino Gateway
   - **DevEUI** – môžeme buď nechať vygenerovať, alebo si prideliť vlastné

- **AppKey** – rovnako ako DevEUI, môžeme buď vygenerovať alebo nechať vlastné

Ďalším dôležitým nastavením pri aplikácii je nastavenie správneho payload formatteru, aby sme vedeli z prijatých dát vyčítať potrebné hodnoty teploty a vlhkosti. Na odporúčanie z literatúry sme použili formatter CayenneLPP, kde nebolo potrebné robiť žiadne dodatočné zmeny. Pôvodne sme skúšali navrhnutie vlastného JSscript formatteru, avšak toto bolo neúspešné.


```
"received_at": "2023-12-04T14:56:16.443549618Z",
"uplink_message": {
  "session_key_id": "AYw1ULmSVeCZyEO4iG4YBg==",
  "f_port": 1,
  "f_cnt": 11,
  "frm_payload": "AWcA8AJoNAMBAAQAAA==",
  "decoded_payload": {
    "digital_in_4": 0,
    "digital_out_3": 0,
    "relative_humidity_2": 26,
    "temperature_1": 24
```


Vyššie môžeme úryvok z prijatého paketu, konkrétne môžeme sledovať riadky *"relative_humidity_2": 26* a *"temperature_1": 24* ,kde sa nachádzajú potrebné hodnoty, ktoré sme chceli zistiť. Tieto hodnoty boli poslané na Dragino Gateway, kde ich môžeme následne vyčítať z konzoly. Pokiaľ by sme umelo zaviedli aj senzor ohňa, ten by bol reprezentovaný pomocou *digital_out_3*, zatiaľ čo buzzer pomocou *digital_in_4.*

5. Konfigurácia Dragino LG01-N Gateway

Pre zabezpečenie funkčnej komunikácie je nutné správne nakonfigurovať všetky sieťové prvky, ktoré daná sieť obsahuje. Začali sme konfiguráciou kľúčového sieťového prvku LoRa bránou Dragino LG01-N.

5.1 Zariadenie Dragino LG01-N

Zariadenie Dragino LG01-N je brána, ktorá funguje na báze technológie LoRa a LoRaWAN. Táto brána nám bude slúžiť na vytvorenie prepojenia medzi LoRa sieťou a sieťou založenou na IP protokole(WiFi, Ethernet...). Zariadenie podporuje štandardy WiFi 802.11 b/g/n, čo je pre naše experimenty vyhovujúce. Okrem toho disponuje 2 x RJ45 portami, takisto ako portom USB 2.0, ktorý budeme používať pri nastavovaní komunikácie medzi bránou a Arduino modulom. Brána je vybavená 400MHz procesorom typu AR9331, 64MB RAM pamäťou a 16MB Flash pamäťou.
Vbudovaný mikrokontrolér je z rodiny ATMega: ATMega 328P. Tento mikrokontrolér využíva 32KB Flash pamäť, 2KB SRAM a 1KB EEPROM pamäť [15].

5.2 Úvodné nastavenia Dragino LG01-N

Pri konfigurácii zariadenia Dragino LG01-N sme boli nútení postupovať podľa viacerých používateľských manuálov a návodov a dopĺňať ich vlastnými experimentami a poznatkami, pretože návody dostupné na internete sú už staré a nekorešpondujú s aktuálnymi verziami programov, ktoré sme používali.

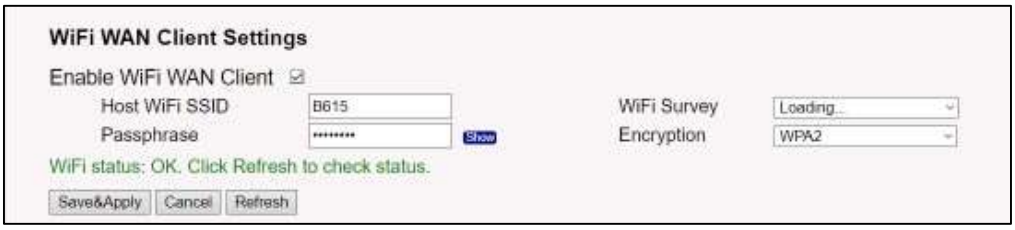
Po prvom pripojení zariadenia do siete sme sa ihneď pokúsili podľa manuálu pripojiť k WiFi sieti, ktorú si brána sama vygeneruje. Táto sieť sa nám nezobrazovala, a preto sme sa rozhodli pre reštartovanie zariadenia do továrenských nastavení, keďže na zariadení pravdepodobne bola ešte uložená konfigurácia z minulých pokusov, ktorú sme my nepotrebovali. Po reštarte do továrenských nastavení sme vykonali update firmvér-u, keďže brána nebola dlho aktualizovaná. Postupovali sme podľa návodu na internete [16].

5.3 Konfigurácia Dragino LG01-N

Po reštarte si a opätovnom pokuse o pripojenie si už brána správne vygenerovala WiFi sieť dragino2-xxxxx, na ktorú sme sa pripojili (heslo na WiFi: dragino+dragino). Po pripojení

sme sa pomocou webového rozhrania pripojili na IP adresu zariadenia (10.130.1.1, meno root, heslo dragino) a mohli sme začať so samotnou konfiguráciou v UI rozhraní zariadenia.

Prvým krokom bolo prepnúť zariadenie z aktuálneho módu WiFi Access Point do módu WiFi WAN Client. Táto zmena spôsobí, že Dragino si už nebude generovať vlastnú nezabezpečenú WiFi sieť, ale namiesto toho ju pomocou prihlasovacích údajov danej WiFi siete pripojíme na jednu z dostupných WiFi sietí, a z nej bude využívať internetové pripojenie [17].



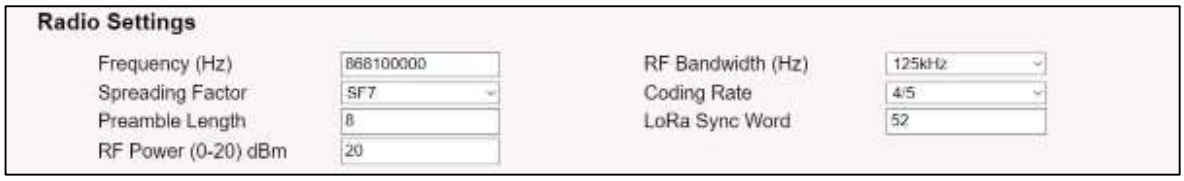
Obr.č. 17 - Nastavenie módu WiFi WAN Client

Následne si prednastavíme rôzne parametre brány, ktoré budeme neskôr využívať pri komunikácii:

- Frekvencia: **868,1 MHz**
  Frekvenciu, na ktorej bude zariadenie komunikovať sme si vybrali na základe frekvenčného pásma EU863 (863 MHz – 870 MHz) určeného pre používanie LoRa komunikácie v Európe
- Spreading Factor: **SF7**
  Spreading Factor je základným princípom modulácie LoRa. Vyššie hodnoty SF znamenajú dlhšie vysielanie, ale zvyšujú dosah a penetráciu signálu na úkor rýchlosti prenosu dát. SF7 poskytuje vyvážený pomer dosahu a rýchlosti.
- Preamble Length: **8**
  Určuje počet úvodných bitov pred začiatkom prenosu dát. 8 je štandardná hodnota.
- RF Power (0-20) dBm: **20 dBm**
  RF Power je výkon vysielania. Hodnota 20 dBm (100 mW) je maximálny povolený výkon pre LoRa zariadenia a poskytuje najlepší dosah.
- RF Bandwidth: **125 kHz**
  Tento parameter udáva použitú šírku pásma. 125 kHz je bežná pre LoRa komunikáciu.
- Coding Rate: **4/5**
  Coding Rate (Kódovacia rýchlosť) určuje, aký podiel prenosu dát je použitý na opravu chýb. 4/5 udáva, že z každých 5 bitov sú 4 bity dáta a 1 bit slúži na opravu chýb.

- LoRa Sync Word: **52**

Dvojbitové pole používané na identifikáciu siete a zariadení v sieti LoRa. Hodnota slúži na umožnenie komunikácie medzi zariadeniami v tej istej sieti a ignorovaní prenosov z iných sietí.

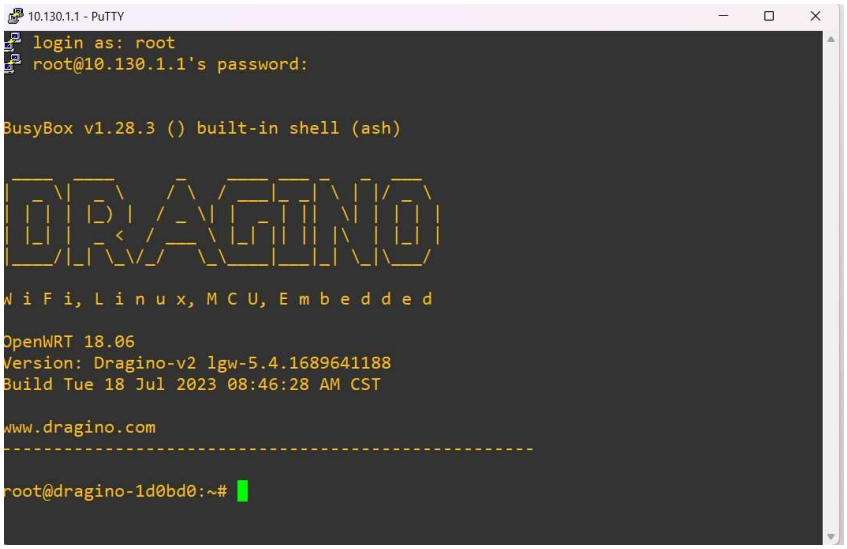


Obr.č. 18 - Konfigurácia prenosových parametrov brány

Po konfigurácii sme sa potrebovali ešte uistiť, že je správne nastavený náš hlavný LoRaWAN server. Keďže v ďalšom kroku budeme naše zariadenie registrovať do portálu TTN, náš nastavený Service Provider musí byť The Things Network (v našom prípade verzia V3) a takisto k s providerom musí korešpondovať aj jeho Server Address (v našom prípade eu1.cloud.thethings.network) [17].

5.4 Pripojenie do zariadenia pomocou klienta Putty

Dragino LG01-N je možné konfigurovať aj bez použitia webového grafického rozhrania. Jedným zo spôsobov je konfigurácia pomocou telnet/SSH klientu Putty. My sme tento spôsob konfigurácie pri našich experimentoch nevyužili, ale otestovali sme si funkčnosť pripojenia. Na pripojenie do nášho zariadenia sme použili SSH, a po zadaní IP adresy zariadenia (10.130.1.1) a portu 22 sme sa dokázali do zariadenia pripojiť (meno/heslo na pripojenie do Putty rozhrania je taktiež root/dragino).



Obr.č.19 - Pripojenie do Dragino LG01-N cez Putty

6. Konfigurácia komunikácie medzi Dragino LG01-N a TheThingsNetwork

Po nakonfigurovaní brány Dragino LG01-N bolo potrebné pripojiť zariadenie do globálnej LoRa siete. Ako poskytovateľa tejto siete sme si vybrali TheThingsNetwork (TTN). TTN je služba určená pre bezdrôtovú komunikáciu Internet of Things (IoT) zariadení. TTN nám ponúka verejne dostupné brány určené na pripojenie IoT zariadení k LoRa sieti.
Registrácia našej brány do TTN je bezplatná, a preto je pre naše experimenty TTN ideálne [18].

6.1 Vytvorenie účtu v TTN

Pre využívanie možností portálu TheThingsNetwork si potrebujeme vytvoriť používateľský účet. Tento úkon je triviálny, je potrebné iba zadať svoje používateľské meno, emailovú adresu a heslo účtu [19].

Po vytvorení a prihlásení sa do nášho TTN účtu už môžeme naplno začať využívať funkcionality tohto ekosystému (naše meno funakfilip@gmail.com, heslo Dragino1).

6.2 Registrácia zariadenia v TTN

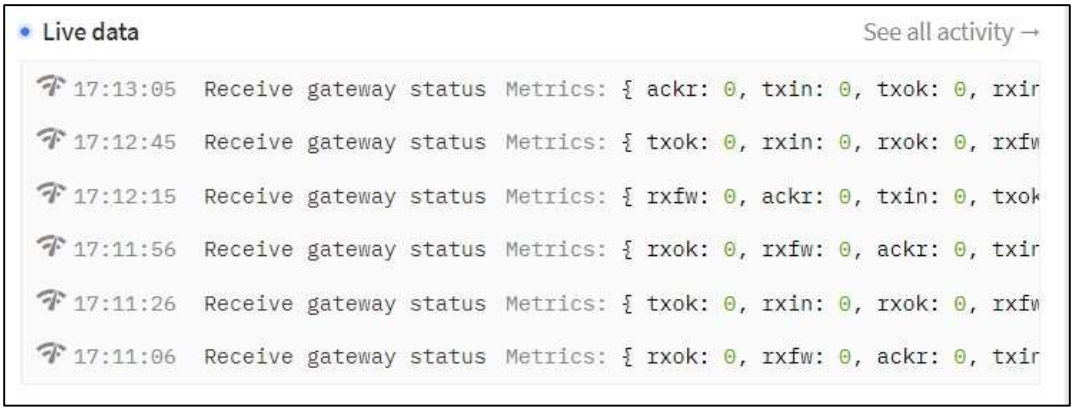
Prvým krokom po vytvorení účtu bolo zaregistrovanie nášho nakonfigurovaného Dragino LG01-N do novovytvoreného TTN účtu. Proces registrovania brány do účtu TTN je taktiež triviálny. Najprv je nutné zadať takzvané **Gateway EUI**.
Gateway EUI je 64 bitový unikátny identifikátor zariadenia. Toto číslo je unikátne pre naše zariadenie a nájdeme ho napísané priamo na našom zariadení, prípadne vo webovom rozhraní zariadenia.

V ďalšom kroku sa už len povypĺňajú zostávajúce údaje o brány, ako je **Gateway ID** (unikátne ID tejto konkrétnej registrácie zariadenia na TTN, je volené nami), Gateway Name (názov brány) a frekvenčný plán (my sme si zvolili EU863 – 870, keďže komunikačná frekvencia na našom Dragino je nastavená na 868,1 MHz). Po nastavení všetkých parametrov je zariadenie pripravené na zaregistrovanie [20].

6.3 Kontrola stavu zaregistrovaného zariadenia

Po zaregistrovaní zariadenia vieme priamo vo webovom rozhraní TTN meniť a prezerať si niektoré nastavenia brány, ktoré súvisia s TTN. Medzi tieto nastavenia patrí napríklad meno, adresa servera brány, správcovia zariadenia na TTN alebo lokalita zariadenia.

Taktiež vieme zistiť, koľko naše zariadenie prijalo uplinkov a downlinkov, a či je zariadenie online alebo nie. Pre nás je však najzaujímavejšia aktivita zariadenia v reálnom čase (tzv. Live Data), kde sa zobrazujú najnovšie správy a updaty, ktoré sa dejú počas komunikácie. Ak je všetko nastavené správne, tak by po úspešnej registrácii mala byť naša brána pripojená do TTN a mala by medzi bránou a serverom TTN prebiehať jednoduchá komunikácia. Túto komunikáciu môžeme sledovať v spomínanom Live Data module.



Obr.č.20 - Live Data modul v TTN

6.4 Popísanie typov správ počas komunikácie

Ako bolo spomenuté v predchádzajúcej kapitole, v rozhraní TTN je možné v reálnom čase sledovať prebiehajúcu komunikáciu medzi našou bránou, TTN serverom a modulom Arduino. Počas tejto komunikácie na serveri TTN môžeme sledovať viacero typov správ, ktoré brána prijíma a odosiela. Tieto správy prichádzajú v rôznych situáciách a v tejto kapitole sú podrobne vysvetlené.

6.4.1 Správa Receive Gateway Status

Správa Receive Gateway Status (Príloha A) je najbežnejšou správou, ktorá je posielaná. Táto správa je v našej sieti posielaná približne každých 30 sekúnd, a slúži na poskytnutie stavu brány. Tieto správy sú užitočné pre monitorovanie a diagnostiku stavu brány v reálnom čase [21].

Analýza správy [22]:


```
{
  "name": "gs.status.receive",
  "time": "2023-12-04T14:08:10.356758071Z",
  "identifiers": [
    {
      "gateway_ids": {
        "gateway_id": "eui-a840411d0bd04150new",
        "eui": "A840411D0BD04150"
      }
    }   ],
```


Hlavička správy: Správa začína hlavičkou správy, v ktorej sú uložené všeobecné informácie, ako je názov `name` a čas vygenerovania správy `time`. Takisto obsahuje identifikátory zariadenia, z ktorého bola správa odoslaná, menovite `gateway id` a `gateway eui`, ktoré sú už v texte vysvetlené. Štruktúra tejto hlavičky správy bola nemenná pri všetkých správach odoslaných zo zariadenia.


```
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.GatewayStatus",
    "time": "2023-12-04T14:08:10Z",
    "versions": {
      "ttn-lw-gateway-server": "3.28.1-rc1-SNAPSHOT-dcfb7b6d5"
    },
    "antenna_locations": [
      {
        "latitude": 22.7,
        "longitude": 114.24,
        "source": "SOURCE_GPS"
      }
    ],
    "ip": [
      "147.175.103.116"
    ],
```


Dátová časť správy: Obsahuje typ správy `@type` a jej časovú pečiatku `time`. Ďalej sa v nej nachádza verzia softvéru nainštalovaná na bráne `versions`, GPS súradnice brány `antenna_locations` a jej IP adresa `ip`.

```
    "metrics": {
      "rxfw": 0,
      "ackr": 0,
      "txin": 0,       "txok": 0,
      "rxin": 0,
      "rxok": 0
    }
```

Dôležitou časťou správy sú nasledujúce metriky výkonu brány:

- `rxfw`: Počet rámcov prijatých bránou a preposlaných na sieťový server
- `ackr`: Percento potvrdení, kt. brána úspešne prijala ako odpoveď na svoje uplink prenosy
- `txin`: Počet uplink rámcov, kt. brána prijala, ale ešte neboli preposlané na sieťový server
- `txok`: Počet downlink rámcov, kt. brána úspešne odvysielala
- `rxin`: Počet uplink rámcov, kt. sú momentálne spracovávané bránou
- `rxok`: Počet uplink rámcov prijatých bránou, ktoré boli správne demodulované a dekódované

Nulové hodnoty metrík značia, že brána aktuálne neprijíma ani neodosiela žiadne dáta.

```
  "correlation_ids": [
    "gs:status:01HGTJGTKMXQP7WYH343QVHZRH"
  ],
  "origin": "ip-10-100-12-105.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_GATEWAY_STATUS_READ"
    ]
  },
  "unique_id": "01HGTJGTKMDKQ7T6F71GQWRXXW"
}
```

Záhlavie správy: Po dátovej časti správy nasleduje záhlavie prijatej správy, ktoré obsahuje dodatočné informácie. Medzi tieto patrí `correlation ids`, čo je zoznam identifikátorov, ktoré môžu byť použité na koreláciu tejto správy s inými udalosťami v TTN. Takisto sa tu nachádza pôvod správy `origin` (to býva spravidla interná adresa servera TTN), identifikátor nájomcu `tenant-id` (prideľovaný TTN) a definované práva potrebné na čítanie tejto správy `rights`. Správa je zakončená jedinečným identifikátorom danej správy `unique-id`.

6.4.2 Správa Receive Uplink Message

Správa Receive Uplink Message (Príloha B) je typ uplinkovej správy vysielanej z koncového zariadenia (v našom prípade Arduino modul) vždy, keď Arduino modul posiela údaje o teplote, vlhkosti a detekcii dymu. Správa slúži na prenos údajov od zariadenia k sieti a poskytuje dôležité informácie potrebné na ich ďalšie spracovanie [21].

Analýza správy [22]:

Hlavička správy: Jej štruktúra je identická so štruktúrou hlavičky predchádzajúcej analyzovanej správy.

```
"data": {
  "@type": "type.googleapis.com/ttn.lorawan.v3.GatewayUplinkMessage",
  "message": {
    "raw_payload": "AFBB0AsdQUCoaCIG0H7Vs3A9qUfAIdk=",
    "payload": {
      "m_hdr": {},
      "mic": "R8Ah2Q==",
      "join_request_payload": {
        "join_eui": "A840411D0BD04150",
        "dev_eui": "70B3D57ED0062268",
        "dev_nonce": "A93D"
      }
    },
```

Dátová časť správy: Začína popísaním typu správy. Najdôležitejším obsahom dátovej časti je samotná prenášaná správa. Súčasťou tejto správy sú:

- Zašifrovaný obsah správy `raw_payload`
- Dekódovaný obsah správy `payload`, ktorý ďalej obsahuje:
  - Hlavičku správy `m_hdr`
  - Kontrolný súčet integrity správy `mic`
  - Detaily o žiadosti o pripojenie, ak ide o proces pripájania zariadenia do siete

```
      "settings": {
        "data_rate": {
          "lora": {
            "bandwidth": 125000,
            "spreading_factor": 7,
            "coding_rate": "4/5"
          }
        },
        "frequency": "868100000",
        "timestamp": 3913444359,
        "time": "2023-12-04T14:27:00.890571Z"
      },
```


Dátová časť správy pokračuje nastaveniami prenosu, ktoré obsahujú šírku pásma `bandwidth`, SF `spreading_factor`, kódovaciu rýchlosť `coding_rate`, vysielaciu frekvenciu `frequency` a údaje o čase.


```
      "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "eui-a840411d0bd04150new",
            "eui": "A840411D0BD04150"
          },
          "time": "2023-12-04T14:27:00.890571Z",
          "timestamp": 3913444359,
          "rssi": -85,
          "channel_rssi": -85,
          "snr": 9,           "uplink_token":
"CiUKIwoXZXVpLWE4NDA0MTFkMGJkMDQxNTBuZXcSCKhAQR0L0EFQEIfYicoOGgwItMO3qwYQwNvBsQMg
2PbX2/LoAw==",
          "received_at": "2023-12-04T14:27:00.909143488Z"
        }
      ],
      "received_at": "2023-12-04T14:27:00.909143488Z",
      "correlation_ids": [
        "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1"
      ],
      "crc_status": true
    },
    "band_id": "EU_863_870"
```


Na konci dátovej časti správy sú vypísané metadáta o príjme `rx_metadata`, vrátane časových údajov, indikátoru sily signálu `rssi`, pomeru signál/šum `snr` a autentifikačný kľúč, ktorý potvrdzuje platnosť uplink správy `uplink_token`. Po metadátach nasleduje časový údaj o prijatí dát `received_at`, kontrola CRC `crc_status` a identifikátor použitého frekvenčného pásma `band_id`.

Záhlavie správy: Štruktúra záhlavia prijatej správy je totožná so štruktúrou záhlavia predchádzajúcej analyzovanej správy.

6.4.3 Správa Send Downlink Message

Správa Send Downlink Message (Príloha C) je typ downlinkovej správy, ktorá je odosielaná z našej brány naspäť ku koncovému zariadeniu (v našom prípade Arduino modul). Správa je odpoveďou na prijatý uplink a používa sa na odoslanie pokynov, konfiguračných údajov a iných informácii späť ku koncovému zariadeniu [21].

Analýza správy [22]:

Hlavička správy: Jej štruktúra je identická so štruktúrou hlavičky predchádzajúcich analyzovaných správ.


```
"data": {
  "@type": "type.googleapis.com/ttn.lorawan.v3.DownlinkMessage",
  "raw_payload": "IPJnagRd0TiHMWKFKx9SWdFk4vZ8kaGLdoZdaxEiRb6W",
  "scheduled": {
    "data_rate": {
      "lora": {
        "bandwidth": 125000,
        "spreading_factor": 7,
        "coding_rate": "4/5"
      }
    },
    "frequency": "868100000",
    "timestamp": 3918444359,
    "downlink": {
      "tx_power": 16.15,
      "invert_polarization": true
    },
    "concentrator_timestamp": "16803346247000"
  },
  "correlation_ids": [
    "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1",
    "ns:downlink:01HGTKKCDYGEAERC5MA8CQDZRH",
    "ns:transmission:01HGTKKCDY8BJXQ2J0JWK8NRV5"
  ]   },
```


Dátová časť správy: Dátová časť správy pozostáva z typu správy `@type` a zašifrovaného obsahu správy `raw_payload` šifrovaného pomocou algoritmu Advanced Encryption Standard (AES). Obsahuje taktiež plánované prenosové nastavenia `scheduled`, ktoré budú použité pri

komunikácii. Správa definuje špecifické nastavenie pre downlink prenos, ktoré zahŕňajú použitý výkon vysielania `tx_power` a invertovanie polarizácie `invert_polarization`. V závere dátovej časti správy sú vypísané `correlation ids`, ktoré súvisia s danou správou.


```
  "correlation_ids": [
    "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1",
    "ns:downlink:01HGTKKCDYGEAERC5MA8CQDZRH",
    "ns:transmission:01HGTKKCDY8BJXQ2J0JWK8NRV5"
  ],
  "origin": "ip-10-100-12-105.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_GATEWAY_TRAFFIC_READ"
    ]
  },
  "unique_id": "01HGTKKCDZ2MANPYW85ECQJ9DQ"
}
```


Záhlavie správy: Štruktúra záhlavia prijatej správy je totožná so štruktúrou záhlavia predchádzajúcich analyzovaných správ. V záhlaví správy nám pribudli `correlation ids` týkajúce sa uplink správ, na ktoré daná downlink správa odpovedá.

6.4.4 Správa forward uplink data message

Cieľom našej práce je odchytiť správu Forward Uplink Data Message, ktorá je vysielaná z nášho Arduino zariadenia na Dragino LG01-N bránu v pravidelných intervaloch. Vysielaná správa obsahuje všetky dôležité dáta o teplote, vlhkosti a stave dymového senzora, ktoré chceme odchytiť. Táto správa má nasledujúcu štruktúru: [22]


```
{
  "name": "as.up.data.forward",
  "time": "2024-03-25T14:38:57.947669274Z",
  "identifiers": [
    {
      "device_ids": {
        "device_id": "eui-70b3d57ed0062268",
        "application_ids": {
          "application_id": "fei-tp-projekt"
        },
        "dev_eui": "70B3D57ED0062268",
        "join_eui": "A840411D0BD04150",
        "dev_addr": "260B1AAC"
      }
    }
  ],
```

Hlavička správy: Jej štruktúra je identická so štruktúrou hlavičky predchádzajúcej analyzovanej správy.


```
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.ApplicationUp",
    "end_device_ids": {
      "device_id": "eui-70b3d57ed0062268",
      "application_ids": {
        "application_id": "fei-tp-projekt"
      },
      "dev_eui": "70B3D57ED0062268",
      "join_eui": "A840411D0BD04150",
      "dev_addr": "260B1AAC"
    },
    "correlation_ids": [
      "gs:uplink:01HSV0SPPAA6RS406P0FZFM0M1"
    ],
    "received_at": "2024-03-25T14:38:57.944275711Z",
```


Dátová časť správy: Dátová časť správy sa skladá z niekoľkých dôležitých segmentov, ktoré zanalyzujeme. Úvodný segment dátovej časti správy obsahuje údaje o type dátovej štruktúry správy, takisto ako už spomínané identifikátory, correlation IDs a čas, kedy bola správa prijatá na aplikačnom serveri.


```
    "uplink_message": {
      "session_key_id": "AY518ZCYf/0hhkb7SR/Ykw==",
      "f_port": 1,
      "f_cnt": 174,
      "frm_payload": "AWcA5gJoWAMBAAQAAA==",
      "decoded_payload": {
        "digital_in_4": 0,
        "digital_out_3": 0,
        "relative_humidity_2": 44,
        "temperature_1": 23
      },
```


Nasledujúci segment je pre nás najdôležitejší a obsahuje samotný payload, ktorý sa snažíme odchytiť. Začína sa identifikátorom kľúča aktuálnej relácie. `f_port` označuje port, cez ktorý bola správa odoslaná (1 je bežne používaný) a `f_cnt`, ktorý reprezentuje počítadlo správ od začiatku komunikácie. `frm_payload` je zakódovaná forma správy v base64 a `decoded_payload` je už samotný dekódovaný payload:

- `digital_in_4`: Digitálny vstup
- `digital_out_3`: Detektor dymu (0 -> nebol detegovaný dym)
- `relative_humidity_2`: Relatívna vlhkosť v %
- `temperature_1`: Teplota v °C

```
       "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "eui-a840411d0bd04150new",
            "eui": "A840411D0BD04150"
          },
          "time": "2024-03-25T14:38:57.710297Z",
          "timestamp": 573978898,
          "rssi": -49,
          "channel_rssi": -49,
          "snr": 9,
          "uplink_token":
"CiUKIwoXZXVpLWE4NDA0MTFkMGJkMDQxNTBuZXcSCKhAQR0L0EFQEJLy2JECGgwIgZmGsAYQt+Hs3w
Ig0Ny7ntqNAQ==",
          "received_at": "2024-03-25T14:38:57.715756666Z"
        }
      ],
```


Táto časť obsahuje metadáta správy, ako sú identifikátory brány, časové údaje, intenzita prijatého signálu (Received Signal Strength Indicator) `rssi`, pomer signál/šum (Signal-to-Noise Ratio) `snr` a token pre uplink správy.


```
      "settings": {
        "data_rate": {
          "lora": {
            "bandwidth": 125000,
            "spreading_factor": 7,
            "coding_rate": "4/5"
          }
        },
        "frequency": "868100000",
        "timestamp": 573978898,
        "time": "2024-03-25T14:38:57.710297Z"
      },
      "received_at": "2024-03-25T14:38:57.738627122Z",
```


V ďalšej časti sa nachádzajú nastavenia prenosu. Medzi tieto patria nami nastavené parametre prenosu, ako šírka pásma, SF, kódovacia rýchlosť, frekvencia a časové údaje.


```
      "consumed_airtime": "0.061696s",
      "locations": {
        "user": {
          "latitude": 48.15241306020923,
          "longitude": 17.073821724010376,
          "altitude": 30,
          "source": "SOURCE_REGISTRY"
        }
      },
```


Tento segment obsahuje dáta zobrazujúce čas strávený vysielaním danej správy (`consumed_airtime`) a vopred nastavené údaje o lokácii zariadenia.

```
      "network_ids": {
        "net_id": "000013",
        "ns_id": "EC656E0000000181",
        "tenant_id": "ttn",
        "cluster_id": "eu1",
        "cluster_address": "eu1.cloud.thethings.network"
      }
    }
  },
```

Záver dátovej časti správy zahŕňa rôzne identifikátory siete, ako identifikátor siete (`net_id`), identifikátor network servera (`ns_id`), identifikátor prenajímateľa (`tenant_id`), identifikátor clusteru (`cluster_id`) a jeho adresa (`cluster_address`).

```
"correlation_ids": [
    "gs:uplink:01HSV0SPPAA6RS406P0FZFM0M1"
  ],
  "origin": "ip-10-100-6-111.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_APPLICATION_TRAFFIC_READ"
    ]
  },
  "unique_id": "01HSV0SPWVC9FSQB499A51R7VP"
}
```

Záhlavie správy: Štruktúra päty prijatej správy je totožná so štruktúrou päty predchádzajúcej analyzovanej správy.

7. RTL-SDR

RTL-SDR rádio predstavuje novú technológiu, ktorá využíva čip set RTL2832U pôvodne vyvinutý hlavne pre televízne USB adaptéry. Tento chipset bol pôvodne navrhnutý pre príjem digitálneho televízneho vysielania DVB-T (Digital Video Broadcasting – Terrestrial).Táto technológia bola rýchlo adaptovaná pre jej schopnosť fungovať ako softvérovo definovaný rádiový prijímač komunitou ľudí venujúcich sa rádiovému prenosu..

Softvérovo Definované Rádio SDR (Software Defined Radio) je technológia, ktorá umožňuje rádiovým systémom vykonávať viacero funkcií softvérovo a s väčšou flexibilitou. Využíva sa na spracovanie rádiových signálov softvérom namiesto klasických dedikovaných hardvérových komponentov [8,12].

7.1 Výhody RTL-SDR

RTL-SDR má nesporne množstvo výhod medzi, ktoré patrí najmä cenová dostupnosť zariadení, čo predstavuje jednu z najväčších výhod RTL-SDR. Obstarávacie náklady na HW sa pohybujú v desiatkach eur ďalšou výhodou je flexibilita a univerzálnosť. Je schopný pracovať s rôznymi frekvenciami a signálmi, čo umožňuje jeho použitie v rôznych aplikáciách od rádioamatérskeho hobby až po profesionálne monitorovanie signálov. Táto technológia umožňuje používateľom experimentovať, vytvárať a skúmať nové aplikácie a technológie. Vie byť teda vhodnou platformou pre inovácie a vývoj. RTL-SDR nájde svoje miesto v rôznych odvetviach, vrátane rádioamatérskeho vysielania, sledovania signálov letových správ, monitorovania bezdrôtových sietí, bezpečnostných aplikácií a viac [8,10,12].

RTL-SDR ponúka veľké množstvo otvoreného softvéru. Teda používateľ si vie vybrať z množstva aplikácií a programov na rôznych operačných systémoch, čím sa viac rozširujú možnosti jeho použitia.

7.2 SDR alebo klasické rádio

Nezávisle na tom, ktorý typ rádia využívame, vždy ho môžeme rozdeliť na tri základné komponenty, ktoré navzájom komunikujú. Týmito komponentami sú:

1. RF časť
2. Spracovač signálu
3. Používateľské rozhranie

Pri klasickom rádiu sú tieto komponenty na sebe hardvérovo závislé. Ak sa zväčšuje integrácia jednotlivých komponentov, zväčšuje sa aj ich závislosť a prepojenie. Naopak pri softvérovo definovanom rádiu sú tieto komponenty jednak hardvérovo nezávisle (môžu byť oddelené v rôznych boxoch) a jednak logicky nezávislé [10,12].

To znamená že jednotlivé komponenty môžu byť vyvíjané tak, aby nezasahovali do chodu ostatných. Ďalšou výhodu je používanie štandardných rozhraní a protokolov (USB, Firewire, TCP / IP, ...) [12].

7.3 Softvér pre RTL-SDR

Pre Windows

SDR Sharp (SDR#) je v súčasnosti najpopulárnejší RTL-SDR kompatibilný softvér. Je zdarma a oproti ostatným softvérom sa jednoduchšie nastavuje a ovláda. Je postavený na modulárnej architektúre, ktorá rozširuje jeho možnosti využitia. V základnej konfigurácii vie dekódovať RDS signál FM vysielania. HDSDR je založený na staršom programe Winrad SDR. HDSDR podporuje RTL-SDR cez použitie modulu ExtIO.dll [12].

Pre Linux

Linrad je voľne dostupný pokročilý SDR program so strmšou krivkou čítania v porovnaní s ostatnými softvérmi. Taktiež lepšie pracuje s tunerom E4000, ktorý zväčšuje jeho dynamický rozsah. Má multi-frekvenčný interval I / Q, veľmi silný "smart" šum blanker, multicast výstup, dva RF vstupné kanály, superior AGC a prispôsobiteľný regulátor automatickej frekvencie (AFC) [12].

GQRX je voľne dostupný SDR program s jednoduchým ovládaním, bežiaci pod OS Linux alebo Mac. Je veľmi podobný SDR# pokiaľ ide o vlastnosti a nenáročnosť ovládania. Prichádza so štandardným FFT spektrom, displejom a radom spoločných nastavení filtrov.

Pre Android

SDR Touch bol prvý RTL-SDR softvér na báze aplikácie pre Android zariadenia.

K dispozícii je obmedzená trial verzia. Plná sá dá zákupiť prostredníctvom obchodu Google Play. Vyžaduje minimálne Android verzie 4.0 s dobrým procesorom a OTG-USB kábel na pripojenie k zariadeniu. Wavesink Plus je ďalší z rady Android SDR-RTL softvérov s možnosťou bezplatného vyskúšania. Jeho hlavou výhodou je možnosť dekódovania DAB + DRM digitálne rádiové signály. Vyžaduje opäť minimálne Android verzie 4.0 a OTG-USB dátový kábel [8,12].

7.4 Inštalácia RTL na Linux

V rámci našej práce sme zvolili ako platformu pre použitie RTL-SDR Linux. Konkrétne išlo o nadstavbu Ubuntu verzia 22.04.3. Pomocou programu Oracle Virtual Box Manager sme vytvorili virtuálne zariadenie s obrazom Ubuntu.

Inštalácia RTL je v prostredí Linuxu naozaj veľmi jednoduchá. Nakoľko RTL-SDR sa používa najmä s týmto prostredím a je veľmi používané v tejto komunite. Pred inštaláciou sme si našu anténu do USB a následne sme mohli prejsť k inštalácií. Celá inštalácia prebieha v okne terminálu.

1. krok: Otvoríme terminál zadaním príkazu

```
ctrl+alt+T
```

2. krok: V okne terminálu spustíme aktualizáciu balíčkov pomocou definovaného príkazu.

```
sudo apt-get update
```

Po úspešnej aktualizácií balíčkov pokračujeme na krok 3

3. krok: Nainštalujeme balíčky RTL-SDR pomocou príkazu

```
sudo apt-get install rtl-sdr.
```

4. krok: Po nainštalovaní balíčkov overíme funkčnosť a správnosť inštalácie pomocou príkazu.

```
rtl_test
```

Tento krok zadávame bez príkazu sudo nakoľko už nepotrebujeme administrátorské práva. Pozor pri ostatných príkazoch nezabudnúť na `sudo` pred zadaním príkazu nakoľko inštalácia nemusí prebehnúť správne.

Po inštalácií základných balíčkov sme prešli k inštalácií programov, ktoré slúžia na prácu s RTL-SDR. V našom prípade sme nainštalovali program GQRX, o tomto programe som písal viac v kapitole 1.3.2.

1. krok: Inštalácia programu GQRX. (Optimálne, možno použiť iný)

```
sudo apt-get install gqrx.
```

2. krok: Overenie správnej inštalácie programu GQRX pomocou príkazu v termináli.

```
dpkg -l | grep gqrx.
```

Alebo

Gqrx

Prvý príkaz overí prítomnosť balíka gqrx v operačnom systéme, druhý príkaz len jednoducho otvorí program pomocou terminálu [8,9].

Pre rozšírenie využitia a možnosti RTL-SDR však môžeme nainštalovať aj iné programy určené k práci s touto technológiou. Následne sme na túto inštaláciu nadväzovali inštaláciou a implementáciou GNU Radia v prostredí Linux čo priblížime v nasledujúcich kapitolách [8,9].

8. Inštalácia RTL-SDR do GRC

8.1 RTL-SDR R280T

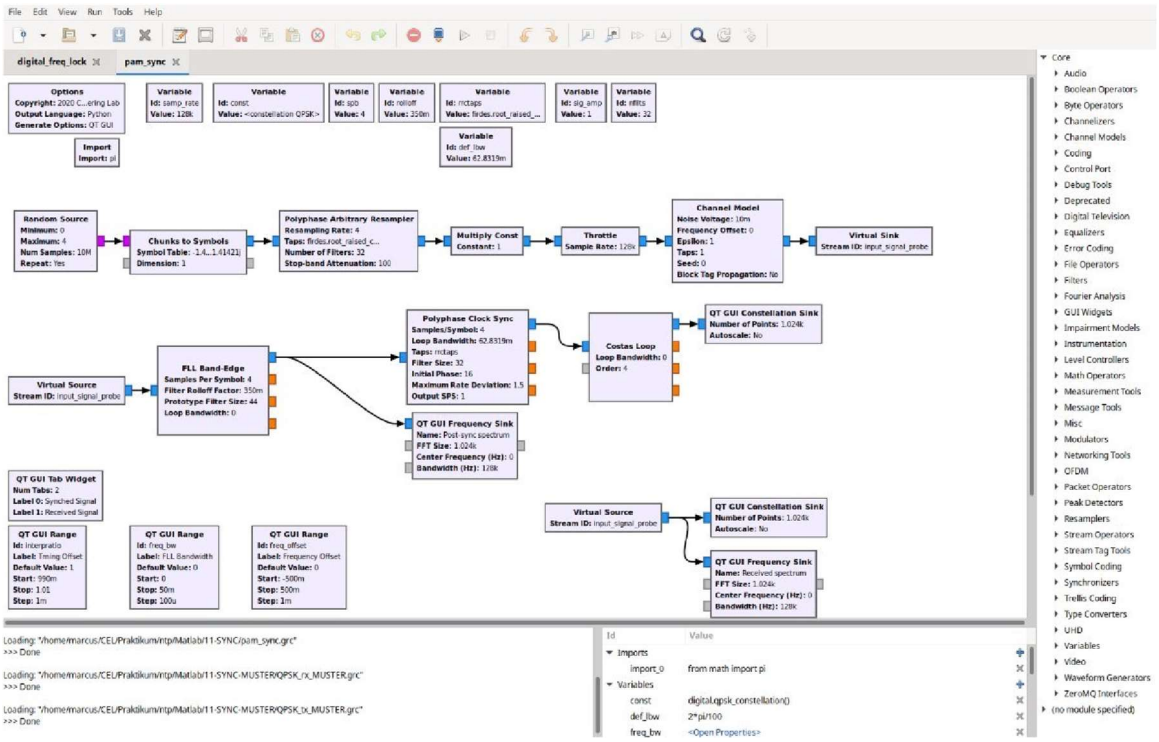
RTL-SDR (Realtek SDR) je typom SDR. Špecifický je pre svoje lacné riešenie určené pre experimentálne, školské a amatérske účely. Využíva USB dongel technológie DVB-T s radičom a tunerom, ktorý umožňuje použitie širokopásmového signálu. V našich experimentoch používame RTL-SDR R280T. Tuner s typom dongel V3 umožňuje frekvenčný rozsah od 24 – 1766 MHz. Maximálna vzorkovacia frekvencia je na hodnote 3.2 MS/s (mega vzorka za sekundu) s optimálnou hodnotou 2.56 MS/s zaručujúcu bezstratovú prevádzku. Pri použití portu USB 3.0 je táto optimálna hodnota vyššia na základe spätnej väzby užívateľov. Zariadenie je nielen nízke v rámci ceny a rovnako aj nízke v rámci odporúčaných požiadavkách, kde vyžaduje najmä aspoň dvojjadrový procesor pre zabezpečenie kritéria bezproblémového behu. Toto kritérium je dnes jednoducho splniteľné. Pre prácu s RTL-SDR zariadením budeme používať GNU Radio s jeho grafickým rozhraním GNU Radio Companion [31].

8.2 GNU radio

GNU Radio je freeware aplikácia obsahujúca súbor nástrojov na vývoj softvéru, ktorý poskytuje bloky na spracovanie signálu na implementáciu softvérových rádií. Môže sa používať s ľahko dostupným lacným externým rádiovým hardvérom na vytváranie softvérovo definovaných rádií alebo bez hardvéru v prostredí podobnom simulácii. Široko sa používa vo výskume, priemysle, akademickej sfére, vo vláde a v amatérskych prostrediach na podporu výskumu bezdrôtovej komunikácie aj reálnych rádiových systémov [28].

8.3 GNU Radio Companion

Grafické rozhranie toolkitu GNU Radio je efektívnym riešením pre konfiguráciu a spracovanie signálových flowchartov pre získavanie hodnôt a údajov vyplývajúcich z experimentov. Obsahuje konfigurovateľné bloky, ktoré sa dokážu spracovať signál a napojiť sa na výstupný flowchart. Pomocou blokov sa užívateľovi eliminuje nutnosť kódovať a namiesto toho je nutný sa iba oboznámiť s dokumentáciou pre bloky GRC. [30]



Obr. č. 21 – Prostredie GRC

8.3.1 Pracovné prostredie

Pre prácu s GRC sa budú používať osobné počítače vybavené operačným systémom Windows 11. Počas experimentovania s GRC sme zistili že platforma Windows nie je pre prácu akceptovateľná pretože po inštalácií nie sú dostupné všetky funkcie GRC ako by tak bolo pri práci v operačnom systéme Linux. Pre tento fakt sme sa rozhodli ďalej pracovať pomocou natívnej virtualizácie OS Linux a to s distribúciou Ubuntu 22.04. LTS. Toto prostredie zabezpečuje že GRC ponúkne pre prácu všetky funkcie. V prípade že počas experimentovanie nastane situácia kde natívna virtualizácia Linuxu nebude ďalej spĺňať požiadavky pre náplň experimentov je v zálohe osobný počítač s priamo nainštalovaným operačným systémom Linux distribúcie Ubuntu 22.04. LTS. Situácia môže nastať pretože aj natívna virtualizácia neobsahuje všetky funkcie, ktoré obsahuje priamo nainštalovaný operačný systém na disku osobného počítača. Doposiaľ počas štúdia dokumentácií k programu GRC nevyskytlo že by mala virtualizácia obmedzovať prácu s RTL-SDR. Po inštalácií Ubuntu nasleduje príprava prostredia pozostávajúce z nastavenie užívateľského prostredia (guest-additions) a samotnej aktualizácie OS. Takéto prostredie je ďalej pripravené pre inštaláciu softvéru GRC, jeho požadovaných knižníc ako aj nutných modulov pre experimentovanie siete typu LoRa.

8.4 Inštalácia GnuRadio

Inštalačný manuál pracuje na inštalácií cez docker. Docker je typ aplikácie, ktorá zjednodušuje proces manažmentu aplikačných procesov v kontajnery. Kontajner umožňuje tieto aplikácie spúšťať. Výhodou tejto metódy je že všetko čo potrebujeme v rámci nášho riešenia je súčasťou jedného balíčka. Toto riešenie nie je priamo popísané ani v dokumentácií samotného GnuRadio. Nevýhodou tohto riešenia je závislí na existencií samotného dockera. Docker obsahuje v sebe GnuRadio verzie 3.7.10 a LoRa balíček gr-lora v0.6. Programy budeme ukladať do adresáru **/usr/local** [48]. Postup celej inštalácie je nasledovný:

8.4.1 Závislosti

Ubuntu 20.04 image s zabezpečením sudo práv pre užívateľa. Závislosť nám určuje aj na akej verzií distribúcie Ubuntu v našom návrhu pracujeme.

8.4.2 Inštalácia Docker

Inštalácia docker je na typu z repozitára, čo zabezpečuje čo najaktualizovanejšiu verziu.

```
sudo apt-get update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
apt-cache policy docker-ce
sudo apt install docker-ce
sudo systemctl status docker
```

8.4.3 Spustenie príkazu Docker bez sumo (voliteľné)

Táto sekcia je len pre umožnenie spustenia docker príkazu bez sudo. A zároveň pridávate svoj užívateľské konto do skupiny docker pre čo najjednoduchšie. V ďalších častiach inštalácie docker nie je potrebné použiť sudo.

```
sudo usermod -aG docker ${USER}
su - ${USER}
groups
sudo usermod -aG docker username
```

8.4.4 Docker test

Táto časť nám skontroluje či inštalácia docker bola správna aby sme mohli neskôr nainštalovať GnuRadio.

```
docker run hello-world
```

Výstupná správa z tohto testu nemá nájsť image „“hello-world“ a teda ho stiahne. Niektoré príkazy vypíšu v príkazovom riadku odpovedajúcu kontrolnú správu. Správne výstupy sú súčasťou prílohy J.

8.4.5 Inštalácia gr-lora

Balíček gr-lora obsahuje GnuRadio s RTL blokmi a LoRa blokmi orientovanými na prácu s SDR správami. Zabezpečená sú funkcie na fyzickej LoRa vrstvy. Inštalácia tohto balíčka pomocou docker obsahuje v sebe všetky potrebné závislosti. Metóda tohto typu zabezpečuje že nenastanú inštalačné chyby alebo zahltenie systému.

```
git clone https://github.com/rpp0/gr-lora.git .
cd docker/
./docker_run_grlora.sh
```

Test inštalácie sa zistí pomocou spustenia súboru:

```
./lora_receive_file_nogui.py
```

Pre pripomenutie je potrebné sa nachádzať v adresári ~/docker/apps. Výstupnú správu môžete porovnať s výstupnou správou v prílohe pre inštaláciu GnuRadio. Zhoda týchto správ vypovedá o správnej inštalácií. Repozitár gr-lora obsahuje aj vzorky testovacích signálov pre učenie a testovanie prostredia GnuRadio [49].

8.5 GNURadio setup a záchyt

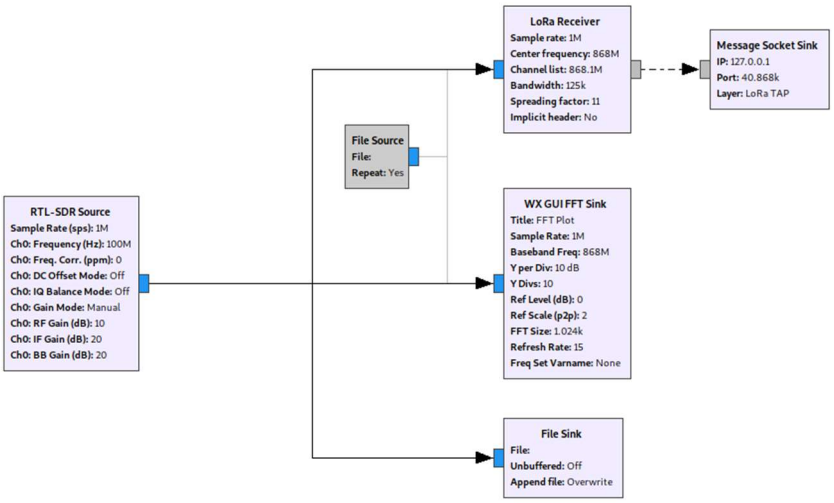
Po úspešnej inštalácii všetkých potrebných závislostí a inštalácie Dockeru s funkčnou verziu GNURadio môžeme prejsť k vytvoreniu diagramu a samotnému odchytu paketu z Arduina pomocou nasledovných krokov:

1. Otvoríme terminál Linuxu a zadáme nasledovný príkaz v adresári Dockeru
   `./docker_run_grlora.sh`
2. Otvoríme GNURadio pomocou príkazu
   sudo gnuradio-companion
3. Po spustení GNURadia zvolíme na hornej lište File->Open *(Ctrl+O)*
4. Vyberieme súbor s názvom *lora_receive_realtime.grc*

Na obrazovke sa nám zobrazí štandardná konfigurácia pre odchytávanie a zobrazenie paketu pomocou USRP alebo Osmocom Source. Keďže sme v našom projekte pracovali so zariadením RTL, je treba pridať blok RTL-Source.

5. Vyhľadáme **RTL-SDR Source** blok z lišty napravo, *(Môžeme použiť Ctrl+F)* následne ho potiahneme zo zoznamu a pridáme do schémy
6. Bloky **osmocom** a **USRP** buď vypneme *(D)* alebo zmažeme *(del)*
7. Do schémy rovnako pridáme bloky **File Sink** a **File Source** *(File Source blok dočasne vypneme, keďže budeme zatiaľ používať odchyt pomocou antény)*, ktoré sú potrebné ak si budeme chcieť daný záchyt uložiť do súboru alebo s ním pracovať v GNURadiu.
8. **RTL-SDR Source** a **File Source** výstupy prepojíme so vstupmi blokmi **LoRa Receiver** a **WX GUI FFT Sink**.
9. **RTL-SDR Source** výstup prepojíme so vstupom bloku **File Sink**.

Výsledná schéma by mala vyzerať nasledovne.



Obr. 22 Schéma GNURadio na odchyt a ukladanie paket

10. V schéme si zmeníme hodnoty premenných na nasledovné:
    capture_freq = 868.1e6
    target_freq = 868.1e6
    sf = 7
    bw = 125000
    samp_rate = 1e6
11. V RTL-SDR Source bloku zmeníme nasledovné parametre:
    Frequency = target_freq
    Bandwidth = bw
12. Do **File Sink** bloku zadáme cestu, kde si chcem prípadný .dat súbor uložiť.
13. Schéma je teraz pripravená a ak máme pripojenú RTL anténu, môžeme ju spustiť a začať odchyt paketov. Na obrazovke sa nám zobrazí okno z FFT Sinku, kde môžeme pozorovať záchyty v jednotlivých paketov a v konzole samotného GNURadio, môžeme vidieť aj ich hexaguláš *(Ten sa zobrazí len v prípade ak sme daný paket zachytili).*
14. Pokiaľ sme spokojný s množstvom zachytených paketov, môžeme schému vypnúť. Následne je potrebné si súbor z Dockeru exportovať pomocou príkazu
    `docker cp <containerId>:/file/path`

8.6 Spustenie schémy zo súboru a ďalšie informácie

Ak sme si pomocou GNURadia spravili záchyt do súboru, môžeme následne cestu k tomuto súboru zadať ako parameter File Source bloku a vypnúť bloky File Sink a RTL-SDR Source. GNURadio v tomto režime bude dookola púšťať záchyt zo súboru a ak sa v ňom nachádzajú packety, budú rovnako zobrazované v FFT Sinku a konzole GNURadia. Ďalej je vhodné spomenúť, že Docker sa pri spustení vždy obnovuje do pôvodného stavu, preto je potrebné aby sme si tú schému vždy navrhli buď nanovo, alebo ju do Dockeru po spustení exportovali a následne otvorili. To isté sa týka aj všetkých súborov zo záchytov, ktoré počas behu kontajneru spravíme, je potrebné ich uložiť na host inak sa po vypnutí Dockeru stratia. Docker po skončení môžeme jednoducho ukončiť pomocou príkazu `exit`. Každé spustenie Docker kontajneru s GNURadio generuje unikátne ID, ktoré sa stále mení.

9. Zabezpečenie LoRa komunikácie

Pre úspešný odchyt a dešifrovanie paketu v našej LoRa komunikácii musíme vedieť, aké bezpečnostné protokoly sa využívajú pri LoRa komunikácii a akým spôsobom je v LoRa sieťach vykonávané šifrovanie dát.

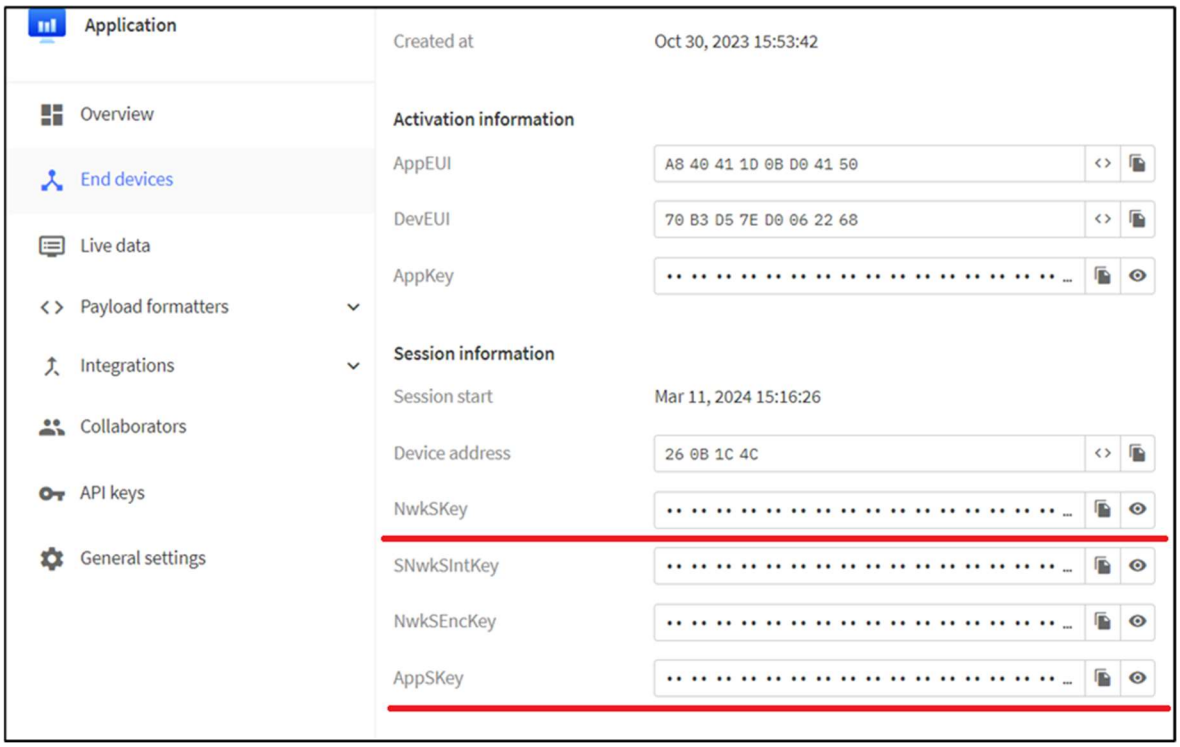
Technológia LoRa používa na zabezpečenie komunikácie šifrovací protokol AES, ktorý podporuje niekoľko prevádzkových režimov: AES-CMAC pre ochranu integrity a AES-CTR pre šifrovanie. Každému LoRaWAN zariadeniu je pridelený unikátny AES 128-bit identifikátor (AppKey) a globálny unikátny identifikátor (DevEUI založené na EUI-64), ktoré oba majú svoju úlohu pri autentifikačnom procese zariadenia. LoRaWAN payload-y sú vždy šifrované medzi koncovým uzlom a serverom [43].

9.1 Bezpečnostné kľúče používané pri LoRa komunikácii

LoRaWAN špecifikácia 1.0.x (v našom projekte používame špecifikáciu 1.0.3) využíva pre zabezpečenie komunikácie niekoľko typov bezpečnostných kľúčov. Medzi tieto kľúče patria (všetky potrebné kľúče nájdeme priamo v UI TheThingsNetwork na karte „End Devices“): [44]

a) **NwkSKey (Network Session Key)**: Network Session Key sa využíva pri interakcii medzi uzlom a sieťovým serverom. Používa sa na overenie integrity každej správy pomocou Message Integrity Check (MIC). MIC je podobný kontrolnému súčtu CRC s tým rozdielom, že zabraňuje úmyselnej manipulácii so správou. Na tento účel LoRaWAN používa AES-CMAC 128-bit. V backend-e TheThingsNetwork sa toto overenie používa na mapovanie nejedinečnej adresy zariadenia (DevAddr) na unikátne DevEUI a AppEUI. NwkSKey má hexadecimálny formát a má dĺžku 128 bitov.

b) **AppSKey (Application Session Key**): Application Session Key sa používa na šifrovanie a dešifrovanie samotného payload-u. Payload je zašifrovaný medzi uzlom a serverom TheThingsNetwork. To znamená, že nikto okrem nás nedokáže čítať obsah odosielaných alebo prijímaných správ bez toho, aby poznal dané kľúče. AppSKey má hexadecimálny formát a má dĺžku 128 bitov.



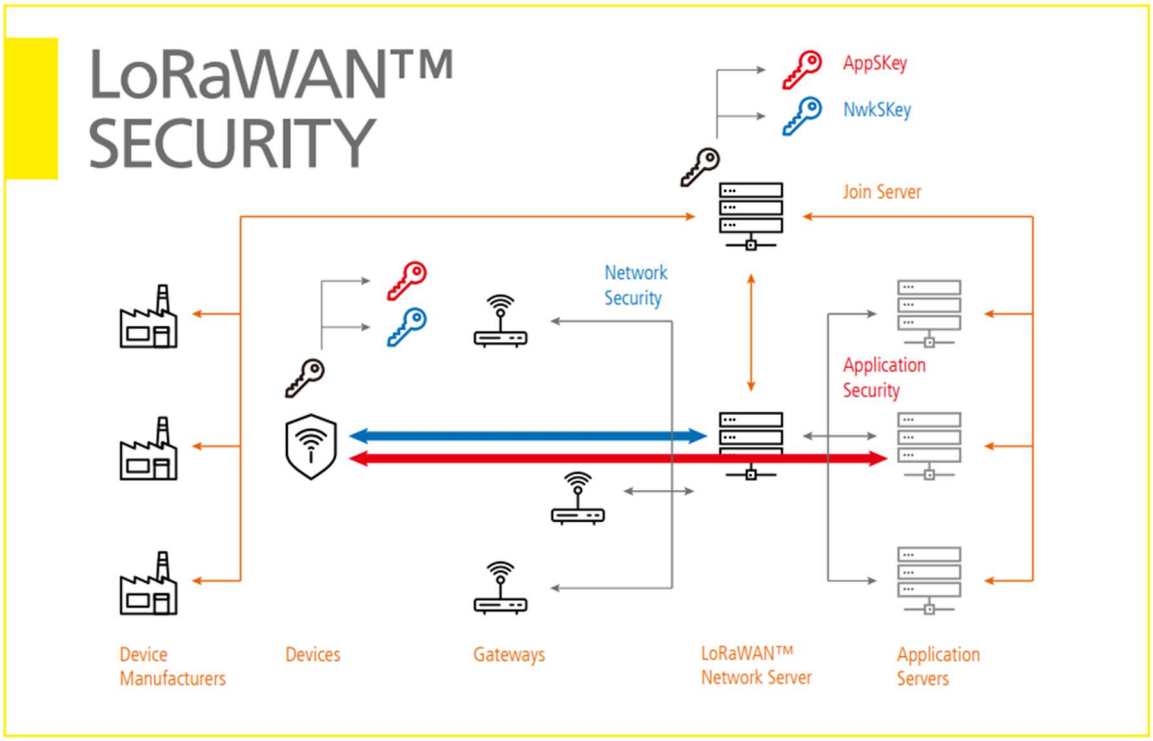
Obr.č.23 - Bezpečnostné kľúče v prostredí TTN

Keď sa zariadenie pripojí do siete (toto sa nazýva ‚Join' alebo ‚Activation'), vygeneruje sa NwkSKey a AppSKey. NwkSKey je v sieti zdieľaný, zatiaľ čo AppSKey je súkromný. Tieto kľúče budú používané počas celého trvania danej relácie. Kľúče NwkSKey aj AppSKey sú jedinečné pre každé zariadenie a každú reláciu. Ak používame Over-the-Air-Authentication (OTAA), tieto kľúče sa pri každom obnovení relácie/aktivácii generujú znova. Pri použití Activation-by-Personalization (ABP) tieto kľúče zostávajú rovnaké kým ich nezmeníme my. V našom projekte využívame Over-the-Air-Authentication (OTAA) [45].

Keďže pracujeme s rádiovým protokolom, každý dokáže so správnym zariadením našu komunikáciu odchytiť a naše správy ukladať. Tieto správy ale potencionálny útočník nedokáže prečítať bez toho, aby poznal AppSKey, pretože sú zašifrované. Takisto nie je možné s týmito správami manipulovať bez toho, aby poznal NwkSKey. Ak by sa o to pokúsil, tak to spôsobí zlyhanie kontroly integrity správy MIC. LoRaWAN správy sú vždy šifrované medzi koncovým uzlom a serverom [45].

9.2 Proces zabezpečenia LoRaWAN komunikácie

Na nasledujúcom obrázku je chronologicky opísaný proces pripojenia zariadenia do LoRaWAN siete a zabezpečenie komunikácie v danej sieti: [43]



Obr.č. 24 - Proces zabezpečenia LoRaWAN komunikácie

1. Výrobca vyrobí zariadenie (IoT senzor) a priradí mu unikátne identifikátory ako napr. AppKey.
2. Zariadenie je distribuované a nasadené do prevádzky, kde začína komunikáciu s LoRaWAN bránou.
3. Join Server spracuje požiadavku na pripojenie k sieti cez OTAA. Zariadenie odošle prostredníctvom LoRaWAN brány Join Request správu, ktorá obsahuje jeho identifikátory. Join Server použije AppKey an výpočet AES-CMAC a overuje znalosť AppKey zariadením a sieťou.
4. Po úspešnom overení Join Server vygeneruje dva kľúče: NwkSKey a AppSKey
5. Zariadenie odosiela dáta prostredníctvom LoRaWAN brány do siete. LoRaWAN Network Server spracováva prijaté správy, overuje ich autentickosť a integritu pomocou vygenerovaného NwkSKey pred odoslaním do aplikačného servera.
6. Aplikačný server prijíma zašifrované aplikačné dáta a následne používa vygenerovaný AppSKey na dešifrovanie aplikačných dát, čím sa zabezpečí ochrana dát pred neoprávneným prístupom.

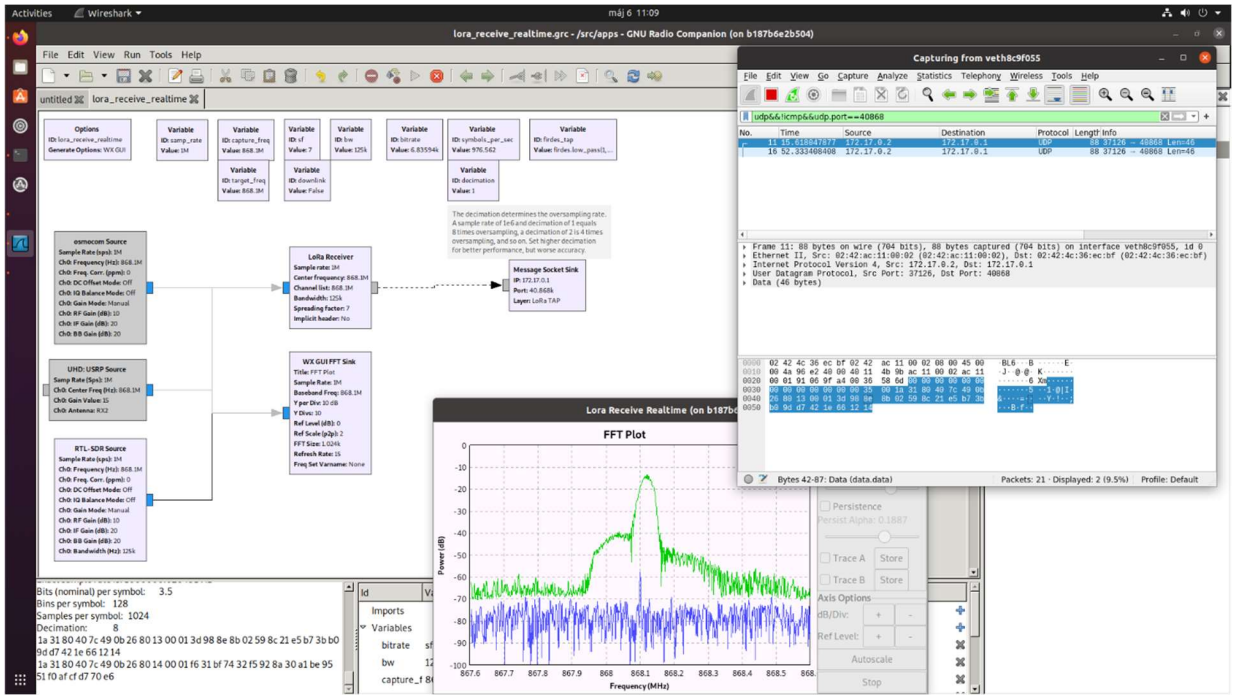
10. Dekódovanie odchyteného paketu

Po úspešnom odchytení komunikácie a zachytení neznámej štruktúry paketu, ktorý sa posiela medzi bránou Dragino LG-01N a Chipsetom Arduino, môžeme pokračovať na dešifrovanie paketu z ktorého sme sa snažili prenesenú informáciu a jej hodnoty prečítať. Ako prvé sme daný paket vypísali do konzole GNU Radio companion a následne sme nastavali odchytávanie v aplikácií Wireshark na porte UDP: 40868. LoRaWAN využíva UDP pre jeho jednoduchosť, rýchlosť a nízke režijné náklady, čo je výhodné pre IoT aplikácie, kde je požiadavka na rýchlu a efektívnu komunikáciu. Na obr.č. 25 je možné vidieť 2 úspešne odchytené príklady. Dôvod prečo iba 2 pakety udáva ten že pre správne odchytenie paketu a reakcie systémov bol nastavený časový rozostup medzi jedným a druhým paketom 30 sekúnd. Počas tohto rozostupu sa taktiež do daného odchytu v našom prípade kde sme odchyt zaznamenali do súboru zapisujú aj hodnoty, ktoré nič neodchytávajú, frekvenčné pásma môžu byť ovplyvnené rôznymi zdrojmi šumu a interferencií. Takýto súbor v príde našej frekvencie 868MHz zapíše príliš veľa Mb/s do daného súboru, čo v následnom spracovaní alebo otváraní súboru je pre operačne systémy Linux a Windows náročné.

Naše zachytené pakety sú vo formáte Hex:

Hex 1. - 1a31804048780b2680020001c4e5094f5aec992259a6e6be173690cc51e63d

Hex 2. - 1a31804048780b2680030001f66577536aafb75415047a865054fe74666f39



Obr.č. 25 – Zachytené pakety v programe WireShark a GNUradio companion

Toto sú správne kľúče po prijatým prázdneho „forward uplink message“.

NwkSkey Hex: D9B11AA8AA97C53DDE4105B63B0DA957

AppSkey Hex: 6C0257F1635228B9587B1BFD127C9174

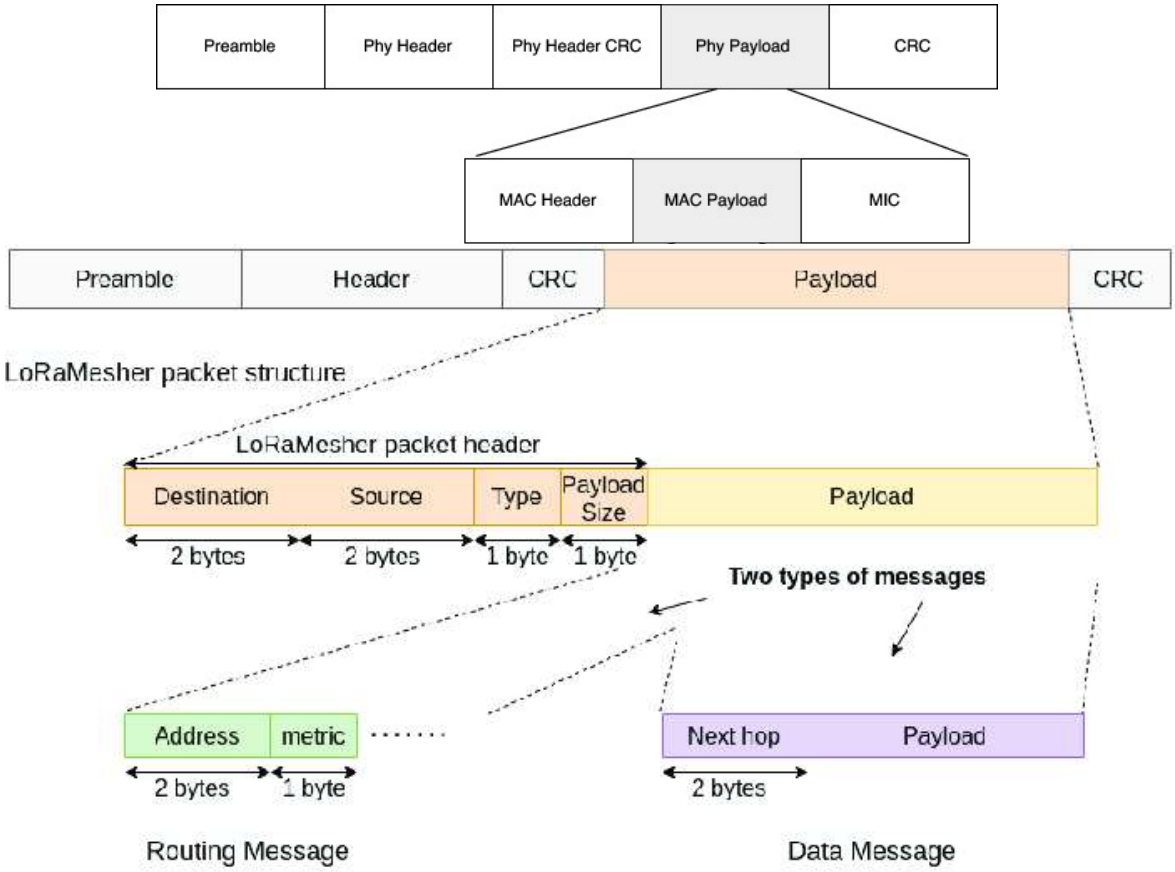
Čo je dôležite spomenúť že stránka TTN si pamätá iba posledné uložené kľúče, ktoré nastali pri poslednej komunikácií a pri zapojení novej komunikácie:

Join Request: Zariadenie posiela join request správu na LoRaWAN server

Join Accept: Server odpovie join accept správou, ktorá obsahuje kľúče pre novú reláciu.

Obr.č.26 – Správa Forward Uplink Data Message ktorá aktualizuje najnovšie kľúče

Následne po tejto akcií „Forward Uplink data message“ zariadenie stránka TTN dostáva od zariadenia úplne nové AppSkey a NwkSkey, a tieto kľúče je potrebné použiť na dešifrovanie, pretože tieto nové pregenerované kľúče patria pre danú reláciu. Odporúčanie je načítať znovu stránku pomocou „F5“ v hlavnom menu. Ak by sme použili neaktualizované kľúče tak dostaneme nesprávne dešifrovaný paket v nesprávnom formáte.



Obr.č. 27 – Štruktúra a rozloženie paketu LoRa

Dôležitým krokom po zachytení paketov je ich aj nasledovná úprava a to odstránenie prvých 6 Bytov, ktoré tvorí MAC Header a posledných 4 koncových Bytov, ktoré tvoria MIC. Tento krok je nevyhnutný pre správnu dešifrovanie paketu musíme vložiť iba čisté užitočné dáta bez týchto informácií [41].

Pakety pred úpravou nevhodné na dešifrovanie:

Hex 1. - **1a3180**4048780b2680020001c4e5094f5aec992259a6e6be173690cc51**e63d**

Hex 2. - **1a3180**4048780b2680030001f66577536aafb75415047a865054fe7466**6f39**

Výsledne pakety po úprave pripravené na dešifrovanie:

Hex 1. - 4048780b2680020001c4e5094f5aec992259a6e6be173690cc51

Hex 2. - 4048780b2680030001f66577536aafb75415047a865054fe7466

10.1 Dekódovanie LoRaWAN paketu

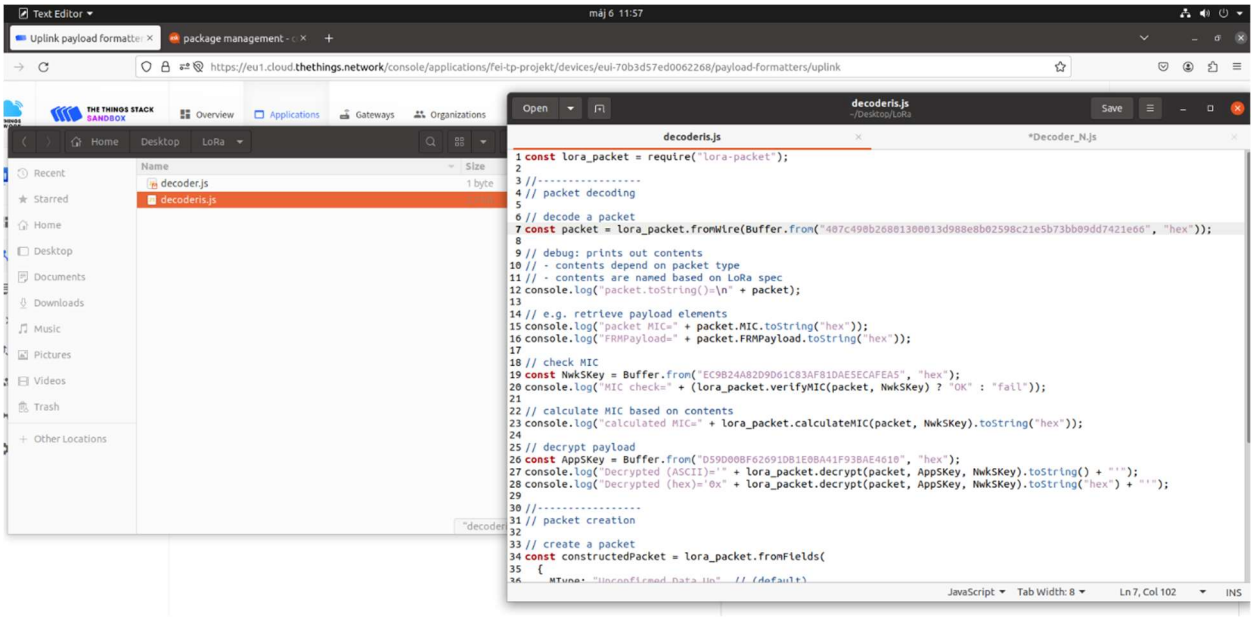
Toto dekódovanie je založené na node.js a používame knižnicu lora-packet. K samotnej inštalácií knižnice je nutné nainštalovať potrebné závislosti.

```
sudo apt install nodejs
sudo apt install npm
```

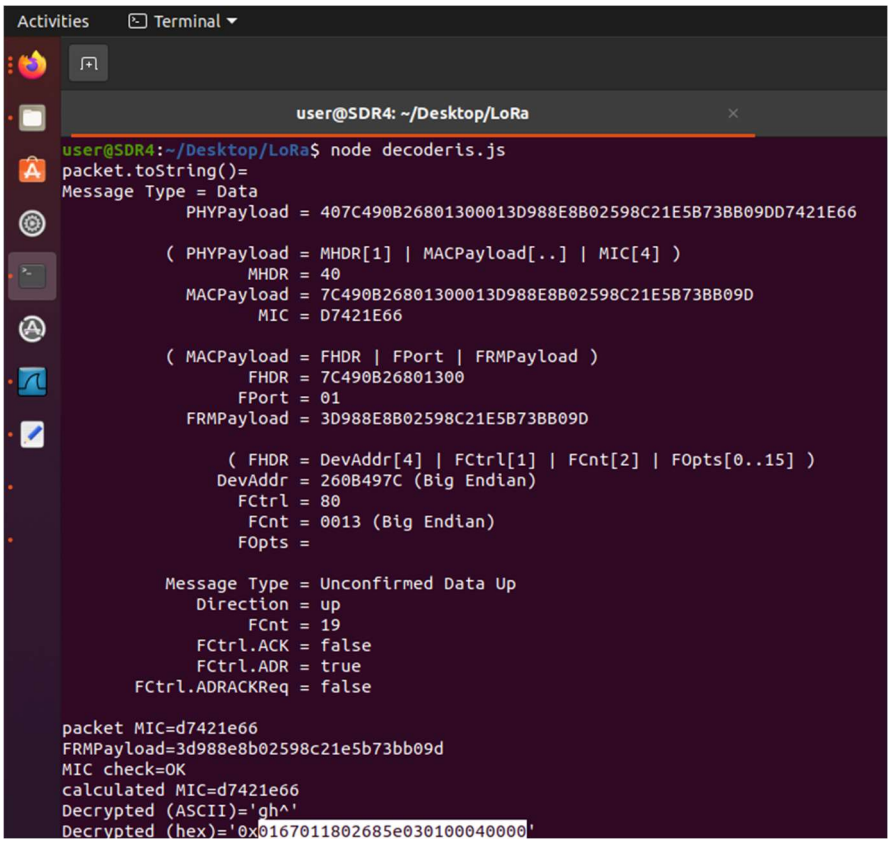
V nasej práci používame nodejs verzie 10. To ovplyvňuje príkaz na inštaláciu

```
npm install lora-packet
```

Ďalej je nutne využiť ukážkový skript z github lora-packet knižnice. Tento skript je v prílohe H. Dešifrovanie paketov je možné aj online na stránke https://lorawan-packet-decoder-0ta6puiniaut.runkit.sh/



Obr.č. 28 – Kód na dešifrovanie paketu v operačnom systéme Linux



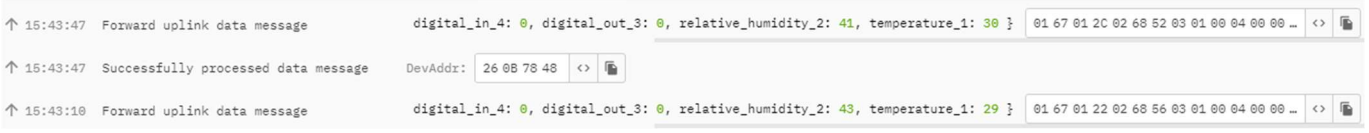
Obr.č.29 – Dešifrovania celej správy pre daný paket v operačnom systéme Linux

Po dešifrovaní nášho paketu sme získali naše výsledne pakety, ktoré boli použité ako výstup práce. Tieto pakety ako je možné vidieť na obrázku č. XX sa zhodujú aj s informáciou, ktorú sme dostali na stránke TTN pod prijatými novými „forward uplink data message“:

Dešifrovaný paket (hex)='0x0167012c026852030100040000'

Dešifrovaný paket (hex)='0x0167012202685603010004000'

Vo zvýraznených hodnotách čiernym „Bold“ písmom dešifrovaných paketov nastala zmena, ktorá sa javí aj ako reálne zmena vlhkosti a teploty zariadenia.



Obr.č.30 – správy obsahujúce reálne dáta

Ak by sme chceli prípadne sami získať z týchto dešifrovaných paketov reálne hodnoty, je nutné tento dešifrovaný paket vložiť do stránky TTN a využiť dekódovanie pomocou Cayenne LPP.

11. CayenneLPP

Na formátovanie našich dát s teplotou a vlhkosťou používame CayenneLPP. CayenneLPP (Low Power Payload) je formát údajov, ktorý sa používa na efektívne kódovanie a dekódovanie senzorových dát v nízkoenergetických sieťach, ako je napríklad nami používaný LoRaWAN. Tento formát bol vyvinutý spoločnosťou myDevices, aby umožnil jednoduché a efektívne posielanie údajov z rôznych senzorov a zariadení v rámci IoT (Internet of Things) aplikácií. [46]

CayenneLPP je často využívaný v sieťach typu LoRaWAN z dôvodu jeho efektívnosti pri formátovaní dát vďaka použitiu binárneho kódovania, ktoré pomáha redukovať veľkosť správy, čím šetrí šírku pásma a znižuje spotrebu energie. Cayenne LPP je taktiež jednoduchý na implementáciu a podporuje rôzne typy senzorových dát, vrátane teplotných senzorov, senzorov vlhkosti, tlaku, svetla atď. V rámci jednej sieti je možné pripojiť a spravovať veľký počet senzorov. [46]

CayenneLPP definuje každú správu ako sériu dátových bodov, kde každý dátový bod obsahuje identifikátor kanálu, typ dát a samotnú hodnotu dát. Tento formát umožňuje rýchle a jednoduché dekódovanie na strane prijímateľa. [46]

V dokumentácii ku CayenneLPP je možné zistiť všetky potrebné informácie ku dekódovaniu, vrátane hexadecimálnych hodnôt reprezentujúcich jednotlivé typy senzorov alebo metriku používanú pri kódovaní samotných dát, ako sú napr. teplota a vlhkosť. [46]

To znamená, že v našom prípade, ak sme po dekódovaní dostali dátový tok s nasledujúcou štruktúrou: `01 67 01 18 02 68 5E 03 01 00 04 00 00` -> vieme presne určiť, čo reprezentujú dané dátové body:

`01` – **Kanál 1**

`67` – Typ senzoru – Teplotný senzor

`01 18` – Hodnota teploty v hexadecimálnom formáte (280 v desiatkovej sústave, čo predstavuje 28°C, keďže hodnota 280 je v desatinách stupňa)

`02` – **Kanál 2**

`68` – Typ senzoru – Senzor vlhkosti

`5E` – Hodnota vlhkosti v hexadecimálnom formáte (94 v desiatkovej sústave, čo predstavuje 47%, keďže hodnota 94 je v pol percentách)

`03` – **Kanál 3**

`01` – Typ senzoru – Digitálny výstup (v našom prípade dymový senzor)

`00` – Hodnota digitálneho výstupu v hexadecimálnom formáte (0 -> nebol detegovaný dym)

`04` – **Kanál 4**

`00` – Typ senzoru – Digitálny vstup

`00` – Hodnota digitálneho vstupu v hexadecimálnom formáte (0)

Po preložení analyzovaného dátového toku v testovacom CayenneLPP dekóderi v prostredí TheThingsNetwork (na karte End devices -> Payload formatters) zistíme, že payload bol dekódovaný správne. [47]

Obr.č.31 - Dekódovanie Cayenne LPP payload-u v prostredí TTN

Záver

V našom projekte sme sa zamerali na odchytávanie packetov vyslaných Arduinom, ku ktorému sú pripojené tri senzory: tepelný, vlhkostný a dymový senzor. Na tento účel sme použili The Things Network (TTN) a Dragino Gateway s RTL anténou. Cieľom projektu je vytvoriť efektívny systém na monitorovanie a analýzu dát zo senzorov pomocou bezdrôtovej komunikácie. Prostredníctvom TTN a Dragino Gateway zabezpečujeme spoľahlivý prenos dát zo senzorov pripojených k Arduinu. Následne tieto dáta zhromažďujeme, spracovávame a analyzujeme, čo nám umožňuje sledovať teplotu, vlhkosť a prítomnosť dymu v monitorovanom prostredí. Tento projekt demonštruje praktické využitie IoT technológií na získavanie a spracovanie dát, ktoré môžu byť následne použité na zlepšenie kvality života a efektivity rôznych systémov.

Literatúra

[1] Qihao Zhou; Kan Zheng; Lu Hou; Jinyu Xing; Rongtao Xu: Design and Implementation of Open LoRa for IoT [online]. Dostupné na internete: https://ieeexplore.ieee.org/document/8768288

[2] LoRa Alliance, Inc. 32 3855 SW 153rd Drive 33 Beaverton, OR 97007 [online]. Dostupné na internete: https://lora-alliance.org/wpcontent/uploads/2020/11/lorawan1.0.3.pdf

[3] Tapadyuti Baral: LoRaWAN Packets Format [online]. Dostupné na internete: https://medium.com/@arghyabaral/lorawan-packets-analysis-618fe9f56838

[4] LoRaWAN Documentation [online] Dostupné na internete: https://lora-developers.semtech.com/documentation/tech-papers-andguides/sendingand-receiving-messages-with-lorawan/sending-and-receivingmessages/

[5] Tapparel Joachim: Complete Reverse Engineering of LoRa PHY [online]. Dostupné na internete: https://www.epfl.ch/labs/tcl/wp-content/uploads/2020/02/Reverse_Eng_Report.pdf

[6] Device classes [online]. Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/classes/

[7] RTL SDR [online]. Dostupné na internete: https://www.rtl-sdr.com/

[8] BLYTHE,Rick: How to Install and Configure RTL.SDR [online]. Dostupné na internete: https://radiohobbyist.com/rtl-sdr-install-config/

[9] Software Defined Radio: Past, Present, and Future, 2023 [online]. Dostupné na internete: https://www.ni.com/en/perspectives/software-defined-radio-past-presentfuture.html

[10] Redakcia tímu BARRETT a Motorola solutions Company, Understanding the difference between SDR and HDR [online]. Dostupné na internete: https://www.barrettcommunications.com.au/news/understanding-the-differencebetween-sdr-and-hdr/

[11] HATAR Ľuboš,2015, SOFTVÉROVO DEFINOVANÉ RÁDIO RTL-SDR Radio RTL-SDR defined by Software [online]. Dostupné na internete: https://spu.fem.uniag.sk/sit2015/zbornik/sit2015_hatar.pdf

[12] LoRaWAN Documentation [online] Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/addressing/

[13] LoRaWAN Documentation [online] Dostupné na internete: https://lora-alliance.org/sites/default/files/2018-04/lorawantm-backendinterfacesv1.0.pdf

[14] LoRaWAN Documentation [online] Dostupné na internete: https://lora-developers.semtech.com/documentation/tech-papersandguides/semtech-network-server-user-guide/find-the-gateway-id

[15] Dragino LG01-N User Manual [online]. Dostupné na internete: https://www.dragino.com/downloads/downloads/LoRa_Gateway/LG01N/LG01N_LoRa_ Gateway_User_Manual_v1.4.0.pdf

[16] Dragino Wiki: How to recover gateway if I can't access it [online]. Dostupné na internete: http://wiki.dragino.com/xwiki/bin/view/Main/How%20to%20Recover%20Gateway%20if %20can%27t%20access%20it/

[17] Single Channel LoRa IoT Kit v2 User Manual [online]. Dostupné na internete: https://www.dragino.com/downloads/downloads/LoRa_IoT_Kit/v2-Kit/Single%20Channel%20LoRa%20IoT%20Kit%20v2%20User%20Manual_v1.0.7.pdf

[18] LoRaWAN specification version 1.1 [online]. Dostupné na internete: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1

[19] The Things Network [online]. Dostupné na internete: https://www.thethingsnetwork.org

[20] The Things Network Sign Up [online]. Dostupné na internete: https://www.thethingsnetwork.org/get-started

[21] The Things Stack: Adding Gateways [online]. Dostupné na internete:

https://www.thethingsindustries.com/docs/gateways/concepts/adding-gateways/

[22] The Things Network: Message Types [online]. Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/message-types/

[23] The Things Stack: Data Formats [online]. Dostupné na internete: https://www.thethingsindustries.com/docs/the-things-stack/concepts/data-formats/

[24] Linuxinstall (no date) LinuxInstall - GNU Radio LoRaWAN Documentation [online] Dostupné na internete: https://wiki.gnuradio.org/index.php/LinuxInstall

[25] Tapparelj (no date) Tapparelj/gr-LORA_SDR: This is the fully-functional GNU radio software-defined radio (SDR) implementation of a Lora transceiver, EPFL., GitHub. Available at: https://github.com/tapparelj/gr-lora_sdr.

[26] Osmocom (no date) OSMOCOM/GR-OSMOSDR: Mirrored from https://gitea.osmocom.org/sdr/gr-osmosdr, GitHub. Available at: https://github.com/osmocom/gr-osmosdr

[27] GNURadio 3.10 on ubuntu 22 in Virtualbox (2023) YouTube. Available at: https://www.youtube.com/watch?v=r9xwjeqRgDw

[28] Gnuradio¶ (no date) gnuradio. Available at: https://www.gnuradio.org/doc/sphinx3.7.0/index.html#

[29] SDR (Software Defined Radio) " GR-OSMOSDR (no date) GrOsmoSDR - gr-osmosdr - Open Source Mobile Communications. Available at: https://osmocom.org/projects/grosmosdr/wiki

[30] About GNU radio · Gnu Radio (no date) GNU Radio. Available at: https://www.gnuradio.org/about/

[31] Software-defined radio (2023) Wikipedia. Available at: https://en.wikipedia.org/wiki/Software-defined_radio

[32] Greatscottgadgets (no date) OSMOCOM source not working with Gnuradio 3.10.1.1 greatscottgadgets/hackrf, GitHub. Available at: https://github.com/greatscottgadgets/hackrf/issues/1044

[33] GNU radio GR-OSMOSDR on ubuntu installation error (No date) Available at: https://lists.gnu.org/archive/html/discuss-gnuradio/2021-05/msg00058.html

[34] ArduinoIDE. Available at: https://www.arduino.cc/

[35] Knižnica Arduino LMIC. Available at: https://github.com/dragino/arduino-lmic

[36] Knižnica DHTLib pre Arduino. Available at: https://github.com/goodcheney/Lora/blob/patch-1/Lora%20Shield/Examples/DHTlib.zip

[37] Knižnica LoRa-RAW pre Arduino. Available at: https://github.com/sandeepmistry/arduino-LoRa

[38] Github repozitár s použitým zdrojovým kódom pre spojazdnenie komunikácie Arduina a TheThingsNetwork. Available at: https://github.com/dragino/Arduino-ProfileExamples/blob/master/libraries/Dragino/examples/IoTServer/ThingSpeak/LG01N_ ThingSpeak%20example/MQTT_Client_to_ThingSpeak/MQTT_Client_to_ThingSpeak.ino

[39] Nemec, Dávid: Návrh komunikácie a zber dát pomocou Long Range (LoRa) bezdrôtových modulov [online]. Dostupné na internete: https://opac.crzp.sk/?fn=detailBiblioFormChildI1Q2T&sid=A8C5898CD77493B3E2F039491516&seo=CRZP-detail-kniha F039491516&seo=CRZP-detail-kniha

[40] LoRaWAN message types [online]. Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/message-types/-types/

[41] Joan Miquel Solé, Roger Pueyo Centelles, Felix Freitag, Roc Meseguer: Implementation of a LoRa Mesh Library [online]. Dostupné na internete: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9930341

[42] The Things Stack: Data Formats [online]. Dostupné na internete: https://www.thethingsindustries.com/docs/the-things-stack/concepts/data-formats/

[43] LoRaWAN Security Whitepaper [online]. Dostupné na internete: https://lora-lliance.org/wpcontent/uploads/2020/11/lorawan_security_whitepaper.pdf

[44] The Things Network: Security [online]. Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/security/

[45] The Things Network: End Device Activation [online]. Dostupné na internete: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

[46] Cayenne Low Power Payload: [online]. Dostupné na internete: https://docs.mydevices.com/docs/lorawan/cayenne-lpp

[47] The Things Network: Cayenne LPP [online]. Dostupné na internete: https://www.thethingsindustries.com/docs/integrations/payload-formatters/cayenne/

[48] Hogan, B. (2021) How to install and use Docker on ubuntu 20.04, DigitalOcean. Available at: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04 (Accessed: 13 May 2024).

[49] rpp0 (no date) Rpp0/GR-Lora: GNU radio blocks for receiving LORA modulated radio messages using SDR, GitHub. Available at: https://github.com/rpp0/gr-lora?tab=readme-ov-file (Accessed: 13 May 2024).

[50] Anthonykirby (no date) Anthonykirby/Lora-Packet: Lora Radio packet decoder, GitHub. Available at: https://github.com/anthonykirby/lora-packet (Accessed: 13 May 2024).

Prílohy

Príloha A: Správa Receive Gateway Status


```
{
  "name": "gs.status.receive",
  "time": "2023-12-04T14:08:10.356758071Z",
  "identifiers": [
    {
      "gateway_ids": {
        "gateway_id": "eui-a840411d0bd04150new",
        "eui": "A840411D0BD04150"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.GatewayStatus",
    "time": "2023-12-04T14:08:10Z",
    "versions": {
      "ttn-lw-gateway-server": "3.28.1-rc1-SNAPSHOT-dcfb7b6d5"     },
    "antenna_locations": [
      {
        "latitude": 22.7,
        "longitude": 114.24,
        "source": "SOURCE_GPS"
      }
    ],
    "ip": [
      "147.175.103.116"
    ],
    "metrics": {
      "rxfw": 0,
"ackr": 0,
      "txin": 0,
      "txok": 0,
      "rxin": 0,
      "rxok": 0
    }
  },
  "correlation_ids": [
    "gs:status:01HGTJGTKMXQP7WYH343QVHZRH"
  ],
  "origin": "ip-10-100-12-105.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_GATEWAY_STATUS_READ"
    ]
  },
  "unique_id": "01HGTJGTKMDKQ7T6F71GQWRXXW" }
```

Príloha B: Správa Receive Uplink Message


```
{
  "name": "gs.up.receive",
  "time": "2023-12-04T14:27:00.909303249Z",
  "identifiers": [
    {
      "gateway_ids": {
        "gateway_id": "eui-a840411d0bd04150new",
        "eui": "A840411D0BD04150"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.GatewayUplinkMessage",
"message": {
      "raw_payload": "AFBB0AsdQUCoaCIG0H7Vs3A9qUfAIdk=",
      "payload": {
        "m_hdr": {},
        "mic": "R8Ah2Q==",
        "join_request_payload": {
          "join_eui": "A840411D0BD04150",
          "dev_eui": "70B3D57ED0062268",
          "dev_nonce": "A93D"
        }
      },
      "settings": {
        "data_rate": {
          "lora": {
            "bandwidth": 125000,
            "spreading_factor": 7,
            "coding_rate": "4/5"
          }
        },
        "frequency": "868100000",
        "timestamp": 3913444359,
        "time": "2023-12-04T14:27:00.890571Z"
      },
      "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "eui-a840411d0bd04150new",
            "eui": "A840411D0BD04150"
          },
          "time": "2023-12-04T14:27:00.890571Z",
          "timestamp": 3913444359,
          "rssi": -85,
          "channel_rssi":     -85,
"snr": 9,
          "uplink_token":
"CiUKIwoXZXVpLWE4NDA0MTFkMGJkMDQxNTBuZXcSCKhAQR0L0EFQEIfYicoOGgwItMO3qwYQwNvB
sQMg2PbX2/LoAw==",
          "received_at": "2023-12-04T14:27:00.909143488Z"          }
      ],
      "received_at": "2023-12-04T14:27:00.909143488Z",
```

```
      "correlation_ids": [
        "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1"       ],
      "crc_status": true
    },
    "band_id": "EU_863_870"
  },
  "correlation_ids": [
    "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1"
  ],
  "origin": "ip-10-100-12-105.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_GATEWAY_TRAFFIC_READ"
    ]
  },
  "unique_id": "01HGTKKANDSDCEXDE4X7TM1Q8Z" }
```

Príloha C: Správa Send Downlink Message


```
{
  "name": "gs.down.send",
  "time": "2023-12-04T14:27:02.719088358Z",
  "identifiers": [
    {
      "gateway_ids": {
        "gateway_id": "eui-a840411d0bd04150new",
        "eui": "A840411D0BD04150"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.DownlinkMessage",
"raw_payload": "IPJnagRd0TiHMWKFKx9SWdFk4vZ8kaGLdoZdaxEiRb6W",
    "scheduled": {
      "data_rate": {
        "lora": {
          "bandwidth": 125000,
          "spreading_factor": 7,
          "coding_rate": "4/5"
        }
      },
      "frequency": "868100000",
      "timestamp": 3918444359,
      "downlink": {
        "tx_power": 16.15,
        "invert_polarization": true
      },
      "concentrator_timestamp": "16803346247000"
    },
    "correlation_ids": [
      "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1",
      "ns:downlink:01HGTKKCDYGEAERC5MA8CQDZRH",
      "ns:transmission:01HGTKKCDY8BJXQ2J0JWK8NRV5"
]
  },
  "correlation_ids": [
    "gs:uplink:01HGTKKAND341PDRYMT6DD3JQ1",
    "ns:downlink:01HGTKKCDYGEAERC5MA8CQDZRH",
    "ns:transmission:01HGTKKCDY8BJXQ2J0JWK8NRV5"
],
  "origin": "ip-10-100-12-105.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_GATEWAY_TRAFFIC_READ"
    ]
  },
  "unique_id": "01HGTKKCDZ2MANPYW85ECQJ9DQ"
}
```

Príloha D: Správa Forward Uplink Data Message


```
{
  "name": "as.up.data.forward",
  "time": "2024-03-25T14:38:57.947669274Z",
  "identifiers": [
    {
      "device_ids": {
        "device_id": "eui-70b3d57ed0062268",
        "application_ids": {
          "application_id": "fei-tp-projekt"
        },
        "dev_eui": "70B3D57ED0062268",
        "join_eui": "A840411D0BD04150",
        "dev_addr": "260B1AAC"
      }
    }
  ],
  "data": {
    "@type": "type.googleapis.com/ttn.lorawan.v3.ApplicationUp",
    "end_device_ids": {
      "device_id": "eui-70b3d57ed0062268",
      "application_ids": {
        "application_id": "fei-tp-projekt"
      },
      "dev_eui": "70B3D57ED0062268",
      "join_eui": "A840411D0BD04150",
      "dev_addr": "260B1AAC"
    },
    "correlation_ids": [
      "gs:uplink:01HSV0SPPAA6RS406P0FZFM0M1"
    ],
    "received_at": "2024-03-25T14:38:57.944275711Z",
    "uplink_message": {
      "session_key_id": "AY518ZCYf/0hhkb7SR/Ykw==",
      "f_port": 1,
      "f_cnt": 174,
      "frm_payload": "AWcA5gJoWAMBAAQAAA==",
      "decoded_payload": {
        "digital_in_4": 0,
        "digital_out_3": 0,
        "relative_humidity_2": 44,
        "temperature_1": 23
      },
      "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "eui-a840411d0bd04150new",
            "eui": "A840411D0BD04150"
          },
          "time": "2024-03-25T14:38:57.710297Z",
          "timestamp": 573978898,
          "rssi": -49,
          "channel_rssi": -49,
          "snr": 9,
```

```
          "uplink_token":
"CiUKIwoXZXVpLWE4NDA0MTFkMGJkMDQxNTBuZXcSCKhAQR0L0EFQEJLy2JECGgwIgZmGsAYQt+Hs3wIg0
Ny7ntqNAQ==",
          "received_at": "2024-03-25T14:38:57.715756666Z"
        }
      ],
      "settings": {
        "data_rate": {
          "lora": {
            "bandwidth": 125000,
            "spreading_factor": 7,
            "coding_rate": "4/5"
          }
        },
        "frequency": "868100000",
        "timestamp": 573978898,
        "time": "2024-03-25T14:38:57.710297Z"
      },
      "received_at": "2024-03-25T14:38:57.738627122Z",
      "consumed_airtime": "0.061696s",
      "locations": {
        "user": {
          "latitude": 48.15241306020923,
          "longitude": 17.073821724010376,
          "altitude": 30,
          "source": "SOURCE_REGISTRY"
        }
      },
      "network_ids": {
        "net_id": "000013",
        "ns_id": "EC656E0000000181",
        "tenant_id": "ttn",
        "cluster_id": "eu1",
        "cluster_address": "eu1.cloud.thethings.network"
      }
    }
  },
  "correlation_ids": [
    "gs:uplink:01HSV0SPPAA6RS406P0FZFM0M1"
  ],
  "origin": "ip-10-100-6-111.eu-west-1.compute.internal",
  "context": {
    "tenant-id": "CgN0dG4="
  },
  "visibility": {
    "rights": [
      "RIGHT_APPLICATION_TRAFFIC_READ"
    ]
  },
  "unique_id": "01HSV0SPWVC9FSQB499A51R7VP"
}
```

Príloha E: Záznam z inštalácie RTL na Ubuntu distribúciu

```
tp@Filip:~$ sudo apt-get update
[sudo] password for tp:
Hit:1 http://sk.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://sk.archive.ubuntu.com/ubuntu jammy-updates InR lease [119 kB]
Hit:3 http://sk.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 229 kB in 1s
 (209 kB/s)
Reading package lists... Done
tp@Filip:~$ sudo apt-get install rtl-sdr
Reading package lists... Done
Building dependency tree... Done Reading state
information... Done rtl-sdr is already the
newest version (0.6.0-4).
0 to upgrade, 0 to newly install, 0 to remove and 131 not to
upgrade. tp@Filip:~$ sudo apt-get install gqrx Reading package
lists... Done
Building dependency tree... Done
Reading state information... Done
Package gqrx is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or is only available from another
source
However the following packages replace it:
gqrx-sdr tp@Filip:~$ sudo apt-get install
gqrx  Reading  package  lists...  Done
Building dependency tree... Done Reading
state information... Done
tp@Filip:~$ gqrx
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland
anyway. gr-osmosdr 0.2.0.0 (0.2.0) gnuradio 3.10.1.1
built-in source types: file fcd rtl rtl_tcp uhd hackrf bladerf rfspace airspy airspyhf soapy
redpitaya freesrp
file_source :warning: file size is not a multiple of item size
Resampling audio 96000 -> 48000
BandPlanFile is /home/tp/.config/gqrx/bandplan.csv
BookmarksFile is /home/tp/.config/gqrx/bookmarks.csv
[INFO] [UHD] linux; GNU C++ version 11.2.0; Boost_107400; UHD_4.1.0.5-
3 libusb: warning [libusb_exit] device 2.1 still referenced libusb:
warning [libusb_exit] device 1.2 still referenced libusb: warning
[libusb_exit] device 1.1 still referenced gr-osmosdr 0.2.0.0 (0.2.0)
gnuradio 3.10.1.1
built-in source types: file fcd rtl rtl_tcp uhd hackrf bladerf rfspace airspy airspyhf soapy
redpitaya freesrp [INFO] Using format CF32.
tp@Filip:~$ dpkg -l | grep gqrx ii  gqrx-
sdr
2.15.8-1build1 amd64        Software defined radio receiver
```

Príloha F: Zdrojový kód použitý pre testovanie prenosu packetov

```
/*******************************************************************************
* Copyright (c) 2019 Thomas Telkamp and Matthijs Kooijman
 *
* Permission is hereby granted, free of charge, to anyone
* obtaining a copy of this document and accompanying files,
* to do whatever they want with them without any restriction,
* including, but not limited to, copying, modification and redistribution.
* NO WARRANTY OF ANY KIND IS PROVIDED.
 *
* This example sends a valid LoRaWAN packet with payload "Hello,
* world!", using frequency and encryption settings matching those of  * the The Things Network.
 *
* This uses OTAA (Over-the-air activation), where where a DevEUI and
* application key is configured, which are used in an over-the-air
* activation procedure where a DevAddr and session keys are  * assigned/generated for use with all
  further communication.
 *
* Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
* g1, 0.1% in g2), but not the TTN fair usage policy (which is probably  * violated by this sketch
  when left running for longer)!


* To use this sketch, first register your application and device with  * the things network, to set
  or generate an AppEUI, DevEUI and AppKey.
* Multiple devices can use the same AppEUI, but each device has its own  * DevEUI and AppKey.
 *
* Do not forget to define the radio type correctly in config.h.
 *
 *******************************************************************************/



#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include
<dht.h>   dht
DHT;
#define DHT11_PIN A0
const int ctl_pin=4; //define the output pin of realy const
int flame_pin=3;  //define the input pin of flame sensor


// This EUI must be in little-endian format, so least-significant-byte  //
first. When copying an EUI from ttnctl output, this means to reverse //
the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,  //
0x70.
static const u1_t PROGMEM APPEUI[8]={ 0x50, 0x41, 0xD0, 0x0B, 0x1D, 0x41, 0x40, 0xA8 }; void
os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}


// This should also be in little endian format, see above. static const u1_t PROGMEM
DEVEUI[8]={ 0x68, 0x22, 0x06, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 }; void os_getDevEui (u1_t*
buf) { memcpy_P(buf, DEVEUI, 8);}
// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In  //
practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key. static const u1_t PROGMEM APPKEY[16] ={ 0xF4,
0x7E, 0xB3, 0x24, 0x9F, 0xFB, 0x39, 0x94, 0x1B, 0xB9,
0x90, 0xD3, 0xF8, 0x3E, 0x59, 0x7C }; void os_getDevKey
(u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
  static float temperature,humidity,tem,hum; static uint8_t LPP_data[13]
=
{0x01,0x67,0x00,0x00,0x02,0x68,0x00,0x03,0x01,0x00,0x04,0x00,0x00};
//0xO1,0x02,0x03,0x04 is Data Channel,0x67,0x68,0x01,0x00 is Data Type static
uint8_t opencml[4]={0x03,0x00,0x64,0xFF},closecml[4]={0x03,0x00,0x00,0xFF}; //the
payload of  the cayenne or ttn downlink   static unsigned int count = 1;
```

```
 static osjob_t
sendjob;


// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations). const
unsigned TX_INTERVAL = 250;


// Pin mapping  const
lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};  void onEvent (ev_t ev) {
Serial.print(os_getTime());
Serial.print(": ");
switch(ev) {         case
EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
break;         case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
break;         case EV_BEACON_MISSED:
Serial.println(F("EV_BEACON_MISSED"));             break;
case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
break;         case EV_JOINING:
Serial.println(F("EV_JOINING"));             break;
case EV_JOINED:
            Serial.println(F("EV_JOINED"));


            // Disable link check validation (automatically enabled
// during join, but not supported by TTN at this time).
LMIC_setLinkCheckMode(0);             break;         case EV_RFU1:
            Serial.println(F("EV_RFU1"));
break;         case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
break;         case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
break;             break;         case
EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
if (LMIC.txrxFlags & TXRX_ACK)               Serial.println(F("Received ack"));
if(LMIC.dataLen>0)
            {               int i,j=0;               uint8_t
received[4]={0x00,0x00,0x00,0x00};
Serial.println("Received :");
for(i=9;i<(9+LMIC.dataLen);i++)   //the received buf
              {
                Serial.print(LMIC.frame[i],HEX);
received[j]=LMIC.frame[i];                j++;
               Serial.print(" ");
              }
             Serial.println();
if
((received[0]==opencml[0])&&(received[1]==opencml[1])&&(received[2]==opencml[2])&&(received[3]==ope
ncml[3])) {
            Serial.println("Set pin to
HIGH.");               digitalWrite(ctl_pin,
HIGH);             }             if
((received[0]==closecml[0])&&(received[1]==closecml[1])&&(received[2]==closecml[2])&&(received[3]==
closecml[3])) {
            Serial.println("Set pin to LOW.");
digitalWrite(ctl_pin, LOW);
            }
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
break;         case EV_LOST_TSYNC:
```

```
            Serial.println(F("EV_LOST_TSYNC"));
break;         case EV_RESET:
            Serial.println(F("EV_RESET"));
break;         case EV_RXCOMPLETE:
            // data received in ping slot
Serial.println(F("EV_RXCOMPLETE"));
break;         case EV_LINK_DEAD:
Serial.println(F("EV_LINK_DEAD"));             break;
case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
break;          default:
            Serial.println(F("Unknown
event"));             break;      }  }
void dhtTem()
{         int16_t
tem_LPP;
       temperature = DHT.read11(DHT11_PIN);    //Read Tmperature data
tem = DHT.temperature*1.0;
       humidity = DHT.read11(DHT11_PIN);      //Read humidity data
       hum = DHT.humidity* 1.0;
Serial.print("###########    ");
       Serial.print("COUNT=");
       Serial.print(count);
       Serial.println("    ###########");
       Serial.println(F("The temperature and humidity:"));
       Serial.print("[");
       Serial.print(tem);
       Serial.print("℃");
       Serial.print(",");
       Serial.print(hum);
       Serial.print("%");
       Serial.print("]");
Serial.println("");
count++;        tem_LPP=tem *
10;            LPP_data[2] =
tem_LPP>>8;
       LPP_data[3] = tem_LPP;
       LPP_data[6] = hum * 2;
}
void pinread() {       int
val,val1;
val=digitalRead(ctl_pin);
val1=digitalRead(flame_pin);
if(val==1)
     {
        LPP_data[9]=0x01;
     }
else
{
        LPP_data[9]=0x00;
    }
if(val1==1)
    {
      LPP_data[12]=0x01;
    }
else
{
      LPP_data[12]=0x00;
    }  }   void
do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
dhtTem();
pinread();
        // Prepare upstream data transmission at the next possible time.
        LMIC_setTxData2(1,LPP_data, sizeof(LPP_data), 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}
```

```
void setup() {
Serial.begin(9600);
while(!Serial);
    Serial.println("Connect to TTN and Send data to mydevice cayenne(Use DHT11 Sensor):");
pinMode(ctl_pin,OUTPUT);     pinMode(flame_pin,INPUT);
//  attachInterrupt(1,fire,LOW);  //no connect Flame sensor should commented this code
    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
pinMode(VCC_ENABLE, OUTPUT);
digitalWrite(VCC_ENABLE, HIGH);     delay(1000);
    #endif


    // LMIC init     os_init();      // Reset the MAC state. Session and pending
data transfers will be discarded.     LMIC_reset();


    // Start job (sending automatically starts OTAA too)
do_send(&sendjob);
}  void
fire()
{
     LPP_data[12]=0x00;
dhtTem();
     LMIC_setTxData2(1,LPP_data, sizeof(LPP_data), 0);
     Serial.println("Have fire,the temperature is send");
}  void loop() {      os_runloop_once();
}
```

Príloha G: Inštalačný manuál pre GNU Radio

Inštalačný manuál funguje na inštalácií priamo zo zdroja keďže zaisťuje stiahnutie najnovšej verzie a zároveň získame program so všetkými balíkmi funkcií ktoré sú ponúknuté. Tento typ inštalácie nie je nutný ale odporúčaný. Programy budeme ukladať do adresáru **/usr/local**.

- UHD sa nachádza v oficiálnej dokumentácií pre inštaláciu GNU radio no je nepovinný. Ponúka drivery pre zariadenia USRP od spoločnosti Ettus Research. Neponúka driver pre dipólovú anténu s RTL-SDR donglom typu R820T V3. Pre prácu s anténou je nutné nainštalovať drivery určené pre RTL-SDR anténu v Linux OS.
- Volk (Vector-Optimized Library of Kernels) slúži pre optimalizovanie kernelov k spracovaniu signálov. Nie je nutný pre všeobecný beh GNU radio no zlepšuje jeho výkon pomocou implementácie pokročilejších operácií spracovania signálov než ktoré sa nachádzajú v samotnom GNU radio. Kernely sú napísané pomocou SIMD kódu. Výhodou je že používatelia nie sú obmedzený pri používaný Volk funkcií tým že ich zariadenia sú postavené na rôznych architektúrach. Volk je v dokumentácií popísaný ako agnostický čo práve opisuje spomenutú vlastnosť. Volk je nutné nainštalovať separátne od GNU Radio a to pred samotným inštalovaním GNU Radio. Inštalácia prebieha pomocou cmake metódy a ukladá sa do adresára /usr/local.
- metóda vybranej inštalácie je pomocou cmake metódy. Existujú aj alternatívne metódy inštalácie overené užívateľmi. Takouto metódou je napríklad CondaInstall, ktorá v základe obsahuje aj modul OsmoSDR ako aj mnoho dalších modulov. Metóda ktorá nie je odporúčaná je BinaryInstall keďže oproti cmake metóde nemusí nainštalovať program so všetkými funkciami. Metódy cmake a CondaInstall toto splňujú.

Manuál pre inštaláciu

Nasledujúce príkazy nainštalujú všetky požadované závislosti pre použitú verziu Linux distribúcie Ubuntu 22.04. Závislosti sú aktuálne pre obdobie tvorenia experimentov a môžu sa v budúcnosti pri nových verziách Ubuntu meniť.

```
sudo apt install git cmake g++ libboost-all-dev libgmp-dev
swig python3-numpy python3-mako python3-sphinx python3-lxml
doxygen libfftw3-dev libsdl1.2-dev libgsl-dev libqwt-qt5-dev
libqt5opengl5-dev python3-pyqt5 liblog4cpp5-dev libzmq3-dev
python3-yaml python3-click python3-click-plugins python3-zmq
python3-scipy python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

```
libcodec2-dev libgsm1-dev libusb-1.0-0 libusb-1.0-0-dev
libudev-dev pybind11-dev python3-matplotlib libsndfile1-dev
libsoapysdr-dev soapysdr-tools python3-pygccxml
python3pyqtgraph libiio-dev libad9361-dev libspdlog-dev
python3packaging python3-jsonschema
```

Inštalácia Volk

```
cd git clone –recursive
https://github.com/gnuradio/volk.git
cd volk mkdir build cd build
cmake-DCMAKE_BUILD_TYPE=Release-
  DCMAKE_INSTALL_PREFIX=/usr/local-
  DPYTHON_EXECUTABLE=/usr/bin/python3../
make make test sudo make install sudo
ldconfig
```

U príkazu make môžeme použití funkciu -jx pre vyhradenie x počtu jadier pre inštaláciu. Väčšie množstvo jadier urýchli inštaláciu no treba spomenúť že tento počet je odporúčané usmerniť na počet jadier zariadenia, na ktorom je inštalácia uskutočnená. Na konci príkazu "make test" je optimálne aby všetky testy úspešne prešli. Ak nastane situácia že len pár testov neprešlo nemusí to byť hneď problematické podľa spätnej odozvy užívateľov.

Nesplnené testy väčšinou naznačujú že chýbajú niektoré knižnice pre správny chod programu.

Inštalácia GNU radio

```
cd git clone
https://github.com/gnuradio/gnuradio.git
cd gnuradio mkdir build cd build
cmake-DCMAKE_BUILD_TYPE=Release-
  DCMAKE_INSTALL_PREFIX=/usr/local
  DPYTHON_EXECUTABLE=/usr/bin/python3 ../
make -jx make test sudo make install sudo
ldconfig volk_profile
```

Počas tejto inštalácie je nutné na konci použiť príkaz "volk_profile" pre využitie napojenia Volk s GNU radio. Celkovo je to pre účel zvýšenia výkonu aplikácie GNU radio pri matematických funkciách a funkciách spracovania signálu.

Príloha H: JavaScript kód pre dešifrovanie LoRa paketu

```
const lora_packet = require("lora-packet");
const packet =
lora_packet.fromWire(Buffer.from("4048780b2680020001c4e5094f5aec992259a6e6be17369
0cc51", "hex"));
console.log("packet.toString()=\n" + packet);
console.log("packet MIC=" + packet.MIC.toString("hex"));
console.log("FRMPayload=" + packet.FRMPayload.toString("hex"));
const NwkSKey = Buffer.from("D9B11AA8AA97C53DDE4105B63B0DA957", "hex");
console.log("MIC check=" + (lora_packet.verifyMIC(packet, NwkSKey) ? "OK" :
"fail"));
console.log("calculated MIC=" + lora_packet.calculateMIC(packet,
NwkSKey).toString("hex"));
const AppSKey = Buffer.from("6C0257F1635228B9587B1BFD127C9174", "hex");
console.log("Decrypted (ASCII)='" + lora_packet.decrypt(packet, AppSKey,
NwkSKey).toString() + "'");
console.log("Decrypted (hex)='0x" + lora_packet.decrypt(packet, AppSKey,
NwkSKey).toString("hex") + "'");
const constructedPacket = lora_packet.fromFields(
  {
    MType: "Unconfirmed Data Up", // (default)
    DevAddr: Buffer.from("01020304", "hex"), // big-endian
    FCtrl: {
      ADR: false, // default = false
      ACK: true, // default = false
      ADRACKReq: false, // default = false
      FPending: false, // default = false
    },
    FCnt: Buffer.from("0003", "hex"), // can supply a buffer or a number
    payload: "test",
  },
  Buffer.from("ec925802ae430ca77fd3dd73cb2cc588", "hex"), // AppSKey
  Buffer.from("44024241ed4ce9a68c6a8bc055233fd3", "hex") // NwkSKey
);
console.log("constructedPacket.toString()=\n" + constructedPacket);
const wireFormatPacket = constructedPacket.getPHYPayload();
console.log("wireFormatPacket.toString()=\n" + wireFormatPacket.toString("hex"));
```

Príloha J: Príloha pre inštaláciu GnuRadio

Výstupná správa z príkazu `apt-cache policy docker-ce`

```
Output
docker-ce:
       Installed: (none)
       Candidate: 5:19.03.9~3-0~ubuntu-focal
       Version table:
              5:19.03.9~3-0~ubuntu-focal 500
              500 https://download.docker.com/linux/ubuntu focal/stable amd64
Packages
```

Výstupná správa z príkazu `sudo systemctl status docker`

```
Output
● docker.service - Docker Application Container Engine
       Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
enabled)
       Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
       TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com/ Main
       PID: 24321 (dockerd)
       Tasks: 8
       Memory: 46.4M
       CGroup: /system.slice/docker.service
               └─24321         /usr/bin/dockerd         -H         fd://         --
containerd=/run/containerd/containerd.sock
```

Výstupná správa z príkazu `groups`

```
Output
Username sudo docker
```

Výstupná správa z príkazu `docker run hello-world`

```
Output
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest:      sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

...
```

Výstupná správa z príkazu `./lora_receive_file_nogui.py`

```
$ ./lora_receive_file_nogui.py
[?] Download test LoRa signal to decode? [y/N] y
[+]  Downloading  https://research.edm.uhasselt.be/probyns/lora/usrp-868.1-sf7-cr4-
bw125-crc-0.sigmf-data -> ./example-trace.sigmf-data . . . . . .
. .
[+]  Downloading  https://research.edm.uhasselt.be/probyns/lora/usrp-868.1-sf7-cr4-
bw125-crc-0.sigmf-meta -> ./example-trace.sigmf-meta . .
[+] Configuration: 868.1 MHz, SF 7, CR 4/8, BW 125 kHz, prlen 8, crc on, implicit off
[+] Decoding. You should see a header, followed by 'deadbeef' and a CRC 5 times.
Bits (nominal) per symbol: 3.5
Bins per symbol:    128
Samples per symbol: 1024
Decimation:  8
 04 90 40 de ad be ef 70 0d
 04 90 40 de ad be ef 70 0d
 04 90 40 de ad be ef 70 0d
 04 90 40 de ad be ef 70 0d
 04 90 40 de ad be ef 70 0d
[+] Done
```

Zápisnica č. 1 zo stretnutia tímu LoRa

Dátum: 18.9.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Filip Fuňák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tím absolvoval rýchle stretnutie, ktoré obsahovalo základné informácie k predmetu a priradenie témy.

Výstup stretnutia:

- Zadanie témy TP
- Zloženie tímu
- Získanie základných informácií

Zápisnica č. 2 zo stretnutia tímu LoRa

Dátum: 25.9.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Filip Fuňák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tím si študoval podklady k zadanej problematike a svoje nadobudnuté zistenia poznačil do záznamníkov.

Študované podklady:

- Experimenty s IoT sieťami - DP (Kabašta)
- LoRaWAN 1.0.3 Specification
- Design and Implementation of Open LoRa for IoT
- Complete Reverse Engineering of LoRa PHY
- LG01 LoRa Gateway User Manual

Výstup stretnutia:

- Zistenie základných informácie o LoRa

Zápisnica č. 3 zo stretnutia tímu LoRa

Dátum: 2.10.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Filip Fuňák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tim tento týždeň kontroloval stav zariadení Arduino a ich komponentov. Zistené informácie sme zdokumentovali v priebežnom reporte, spolu s fotografiami a poznámkami.

Výstup stretnutia:

- Zdokumentovanie stavu zariadení Arduino
- Čiastočne spojazdnenie Dragino Gateway

Zápisnica č. 4 zo stretnutia tímu LoRa

Dátum: 9.10.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Filip Fuňák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Na začiatku stretnutia boli zadané úlohy na pre členov tímu:

- Dorčák vytvoril a začal vypracovať zápisnicu + inštaláciu SDR antény na Windowse
- Čech a Merva riešili registráciu Dragino Gateway na portáli ThingsNetwork, následne prepojenie Arduina so snímačom a Dragino Gateway
- Fuňák študoval a spojazdňoval GNU Radio -> SDR na Windowse
- Rarogiewicz taktiež študoval a spojazdňoval GNU Radio na Linuxe

Výstup stretnutia:

- Dorčák spísal zápisnicu, rozbehal RTL-SDR anténu na Windowse
- Fuňák Naštudovanie materiálov (inštalácia, drivery, dostupné programy) RTL-SDR antény. Pre Windows
- Štúdium materiálov na stránke www.rtl-sdr.com
- Čech a Merva úspešne nastavili Dragino Gateway na portáli ThingsNetwork a následne sa navyše venovali aplikácií Arduino a vysielača, snímač reagoval na bežné príkazy od Arduina úspešne (svietenie, blikanie).
  Prepojenie snímača a Dragino zlyhalo, tieto systémy nepodporujú novšie verzie Windowsu a Arduina - možný fix v pokračovaní na Virtual Machine alebo inštalácie staršej verzie Arduina - pokračovanie do týždňa č. 5.

Zápisnica č. 5 zo stretnutia tímu LoRa

Dátum: 16.10.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: Filip Fuňák

Priebeh stretnutia:
Na začiatku stretnutia boli zadané úlohy na pre členov tímu:

- Dorčák a Rarogiewicz dostali za úlohu prepojiť RTL-SDR anténu s GNU Radio na rôznych OS platformách
- Čech a Merva riešili Arduino IDE a prepojenie Dragino, ďalšie nové spojazdnenie ThingSpeak a MQTT

Výstup stretnutia:

- Dorčák spísal zápisnicu, odskúšal GNU Radio na Windowse pričom prišiel k záveru že optimálnejším riešením bude pokračovať s GNU Radio-m na Ubuntu platforme. Po úspešnej inštalácií GNU Radio na Ubuntu 22.04.02. a vypracoval návod na inštaláciu GNU Radio
- Rarogiewicz ďalej pracoval na príprave GNU Radio prostredia v Ubuntu 22.04.03 image vo VirtualBoxe pre prácu s RTL-SDR anténou. USB passthrough bol na VirtualBoxe úspešne nastavený a RTL-SDR anténa bola na porte detekovaná. Ďalej do GNU Radio pridal LoRa implementáciu a umožnil použiť LoRa bloky v GNU radio.
- Čech a Merva, pokus virtual machine ubuntu pass through, celkovo pokusy realizované na viacerých počítačoch (3).
  Program arduino ignoruje port Dragino naďalej, vyskúšané aj staršie verzie Arduino, 1.6.8, 1.8.5, taktiež ignorujú port Dragino, pokus podľa draginov manuálov, potom postupy podľa Ing. Kabastu.
  Celkový problém je že je nemožné spustiť Dragino example, link plugin error alebo avr gcc error.
  Spojazdnili bránu MQTT, ale nemôžeme sa v aplikácií na ňu pripojiť cez logins - dôvod disconnected na ThingSpeak. Thingspeak úspešne nastavený.

Zápisnica č. 6 zo stretnutia tímu LoRa

Dátum: 23.10.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz,Filip Fuňák

Neprítomný: -

Priebeh stretnutia:

Rozdelenie úloh na tento týždeň:

- Dorčák sa pridal k Mervovi a Čechovi na rozbehanie Arduino gateway
- Fuňák sa pridal k Rarogiewiczovi a pokračovali na spojazdňovaní RTL SDR

Výstup stretnutia:

- Fuňák s Rarogiewiczom nainštalovali GNUradio s LoRa a RTL modulmi do novo vytvorených virtualnych image Ubuntu 22.04. A teda je možné vytvoriť manuál pre inštaláciu. Tá bude dokončená doma mimo stretnutia.
- Nový manuál pre zariadenie Dragino, brána plne spojazdená a prepojená cez LoRaWan a TTN, Arduino doska plne funkčná meria humidity a temperature, problém posledný mismatch MMC v parametroch medzi bránou a zariadením, snažia sa nadviazať kontakt ale asi je chyba v parametroch, inak sa registrujú a snažia sa poslať si údaje. Knižnica DHTlib v návode je nefunkčná a treba stiahnuť novšiu verziu. V rámci budúceho týždňa sa budeme pokúšať spracovať parametre aby sme zabezpečili plne komunikáciu medzi bránou a doskou.

Zápisnica č. 7 zo stretnutia tímu LoRa

Dátum: 30.10.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva,Filip Fuňák

Neprítomný: Lukáš Rarogiewicz

Priebeh stretnutia:

- Dorčák, Čech, Fuňák, Merva pokračovali na práci s Dragino gateway.

Výstup stretnutia:

- Podarilo sa rozbehať prepojenie medzi Arduino doskou, senzormi a Dragino Gateway-om. Na TheThingsNetwork vidíme komunikáciu medzi zariadeniami (aj uplink aj downlink). Ďalej by sme chceli byť schopný vidieť v packetoch parametre nasnímané zo senzorov (chyba pravdepodobne v kóde).

Zápisnica č. 9 zo stretnutia tímu LoRa

Dátum: 13.11.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva,Filip Fuňák, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

- Dorčák, Čech, Fuňák, Merva pokračovali na práci s Dragino gateway a senzormi.

Výstup stretnutia:

- Na stretnutí sme spojazdnili zadanú úlohu na dnes, konkrétne parsovanie prijatého payloadu od senzorov. Bolo za potreby vymeniť/nastaviť iný typ parseru, zistiť hodnoty payloadu a správne ich vyčítať.

Zápisnica č. 10 zo stretnutia tímu LoRa

Dátum: 20.11.2023

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva,Filip Fuňák, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

- Pracovalo sa na spojazdnení s

Výstup stretnutia:

- Na stretnutí sme spojazdnili zadanú úlohu na dnes, konkrétne parsovanie prijatého payloadu od senzorov. Bolo za potreby vymeniť/nastaviť iný typ parseru, zistiť hodnoty payloadu a správne ich vyčítať.

Zápisnica č. 11 zo stretnutia tímu LoRa

Dátum: 15.2.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tím absolvoval rýchle stretnutie online cez GMeet, kde bolo zadané základné ciele.

Výstup stretnutia:

- Feedback od vedúceho predmetu
- Zmeny rozloženia členov v tímoch
- Zadanie úloh pre jednotlivých členov

Zápisnica č. 12 zo stretnutia tímu LoRa

Dátum: 19.2.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Otestovanie funkčnosti Arduina, Dragino Gateway, spojazdnenie RTL antény a GNU Radio. Po tomto stretnutí sme schopný detekovať signály vysielané medzi Arduinom a Dragino Gateway na frekvencii 868.1MHz. Do ďalšieho stretnutia musíme naštudovať RFTap SW aby sme boli schopný dané pakety dešifrovať.

Výstup stretnutia:

- Rozbehanie Dragino Gateway, Arduina
- Spojazdnenie Linuxu, RTL antény, GNU Radio
- Dosiahnutá detekcia signálu pomocou GNU Radia
- Študovanie RFTap SW pre Linux

Zápisnica č. 13 zo stretnutia tímu LoRa

Dátum: 26.2.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Na stretnutí sa pracovalo na úlohách z minulého týždňa, teda zachytení packetu pomocou niektorého zo softvérov. Na tomto stretnutí sme sa pokúšali hlavne o zachytenie pomocou Wiresharku. Pakety, ktoré by mali byť podľa zdrojov viditeľné sú typu UDP, avšak na LL sme žiadne nevideli. Pokusy sme skúšali na Windowse, kde sme boli schopný vidieť komunikáciu cez RFTap na frekvencii 868,1MHz a rovnako aj na Linuxe. V oboch prípadoch sme však aj napriek snahe o troubleshooting boli neúspešný. Ďalší týždeň bude za úlohu skúsiť iné dostupné riešenia na zachytenie packetov.

Výstup stretnutia:

- Spojazdnenie Wiresharku na Windowse a Linuxe
- Snaha o odchytenie packetov
- Neúspešnosť v oboch prípadoch

Zápisnica č. 14 zo stretnutia tímu LoRa

Dátum: 4.3.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tento týždeň sme sa snažili hľadať alternatívy Wiresharku na odchytenie paketov a ich dešifrovanie. Dešifrovanie paketu je možné pomocou NwkSKey a Appskey na stránke LoraWan paket dekodér, tieto kľúče sú taktiež prenášané v LoRa správe ale inak sú vopred definované na stránke LoraWan, do tejto dešifrácie je taktiež potrebné zadať Base64/Hex-Encoded paket (tieto údaje by mali byť viditeľné pri zachytení paketu pomocou našich nástrojov). Našli sme rôzne detektory na LoRa pakety, ktoré otestujeme keď úspešne odchytíme LoRa paket vo Wireshark-u. Na zachytenie sme skúsili nasledovné alternatívy: tcpdump (linux) a PRTG (Windows). Obe sú voľne dostupné, avšak neboli schopné zachytiť packet vysielaný medzi Arduinom a Draginom. Kolegovia Rarogiewicz a Fuňák zatiaľ pracovali na zostavení diagramov v GNURadiu. Z nájdených informácií sme zistili, že možno bude problém medzi samotnými verziami GNURadio a jednotlivými článkami, ktoré v schéme používame. Niektoré stránky na GitHube už totiž nie sú dostupné (problém pri inštalácii gr-gsm modulu)

Výstup stretnutia:

- Otestovanie ďalšieho softvéru (tcpdump, PRTG)
- Prerobenie schémy na dešifrovanie
- Zisťovanie kompatibility jednotlivých súčastí

Zápisnica č. 15 zo stretnutia tímu LoRa

Dátum: 11.3.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tento týždeň sme pokračovali s riešením problémov spojených s výpisom odchytených paketov v GNUradiu, a popri tom sme sa oboznámili so zariadením USRP B210, ktoré bude slúžiť ako záložné zariadenie pre pokračovanie v projekte.

Výstup stretnutia:

- Oboznámenie sa s USRP B210

Zápisnica č. 16 zo stretnutia tímu LoRa

Dátum: 18.3.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Tento týždeň sme mali dve úlohy. Kolegovia Čech a Dorčák sa venovali inštalácií a spojazdneniu USRP B210 zariadenie s úspešným výsledkom. Merva a Rarogiewicz sa snažili prísť na problém vypisovania odchyteného LoRa paketu cez Lora RX blok na nových verziách GNUradia a LoRa add-on.

Výstup stretnutia:

- úspešné spojazdnenie USRP B210 zariadenia
- vypisovania LoRa bloku je stále problématické

Zápisnica č. 17 zo stretnutia tímu LoRa

Dátum: 25.3.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:

Stretnutie bolo sústredené na riešení nevýpisu LoRa bloku na nových vydaniach GNUradia a LoRa bloku no zatial neúspešne. Kolega Dorčák úspešne nainštaloval GNUradio s LoRa blokom pomocou dockera. Docker balíček obsahuje staršie verzie oboch programov no v plnom balíčku a plne kompatibilné. Vďaka tejto metóde sa nám úspešne podarilo odchytiť hexdump z LoRa komunikácie a následne sa ostatní členovia tímu venovali hľadaním správnej dešifrovacej metódy. Zistili sme že payload je nutné previesť do Base64 aby bolo čitatelný.

Výstup stretnutia:

- Odchytenie LoRa paketov na starej verzií GNUradio a LoRa blocku

Zápisnica č. 18 zo stretnutia tímu LoRa

Dátum: 1.4.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:
Po úspešnom spojazdnení dockeru s funkčnou verziou GNURadia. Potreba prejsť na RTL z USRP keďže nám USRP začalo robiť problémy s viacerými zariadeniami (potrebné otestovať prípadne na ďalších). Docker naštastie obsahoval aj inštaláciu pre RTL-SDR a teda aj potrebný blok v GNURadiu. Spustenie je teda úspešné, na ďalšom stretnutí je potrebné nastaviť všetky parametre správne a spojazdniť prípadný odchyt do WireSharku.

Výstup stretnutia:

- Spojazdnenie GNURadio 3.7 s RTL modulom, úspešné spustenie
- Potreba správnej konfigurácie a možnosť uloženia záchytu

Zápisnica č. 19 zo stretnutia tímu LoRa

Dátum: 8.4.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:
Po pridaní všetkých parametrov na záchyt sa nám podarilo úspešne odchytiť packet. Výpis je možný buď v konzole alebo vo Wiresharku po nakonfigurovaní Dockeru, aby mal prístup k sieti počítaču. USRP sme vzdali po neúspešných pokusoch ho rozbehať na 3 zariadeniach (2x Lenovo, 1x Dell laptop) predpokladáme nejakú chybu s USB portami na Lenovo laptopoch, na Dell laptope spôsoboval BSOD. Ďalšie kroky potrebné na finish sú dešifrovanie a zobrazenie správy packetu.

Výstup stretnutia:
- Úspešný výpis packetu z odchytu
- Ďalšie stretnutie, potrebné začať dešifrovať tento packet a extrahovať z neho správu zaslanú z Arduina

Zápisnica č. 20 zo stretnutia tímu LoRa

Dátum: 15.4.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:
Tím sa pokúšal dešifrovať odchytený packet pomocou viacerých spôsobov, aktuálne máme problém s tým, že sa nám nedarí správne dešifrovať packet kvôli MIC mismatch, ak to správne chápeme tak sa jedná o niečo ako CRC, čo validuje packet. Skúšali sme zatiaľ web verziu ale k ničomu sme sa tento týždeň nedostali, bude potrebné si lepšie pozrieť do ďalšieho týždňa AES a ako toto MIC vzniká.

Výstup stretnutia:
- Štúdium AES, MIC a dešifrovania packetov.
- Na ďalšie stretnutie je potrebné to pochopiť aby sme z packetu vedeli extrahovať správu.

Zápisnica č. 21 zo stretnutia tímu LoRa

Dátum: 22.4.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:
Packet sa nám podarilo úspešne dešifrovať, kolega Merva našiel spôsob akým je treba orezať packet tak aby po vložení do programu bolo MIC dobré a teda aby sme dostali správu akú treba. Program je spravený v JavaScripte a dá sa rozbehať aj lokálne, bez potreby internetového pripojenia. Taktiež sme si potvrdili, že sa kľúče menia pri každo napojení Arduina (nie len pri resete Dragina). Máme teda všetko potrebné na tvorbu OVA a finalizácie TP.

Výstup stretnutia:
- Úspešné dešifrovanie packetu.
- Splnenie všetkých zadaní TP.
- Na poslednom stretnutí začneme dávať dokopy všetky materiály a kompletizovať odovzdanie.

Zápisnica č. 22 zo stretnutia tímu LoRa

Dátum: 29.4.2024

Zápis: Branislav Dorčák

Prítomný: Miroslav Čech, Branislav Dorčák, Nikolas Merva, Lukáš Rarogiewicz

Neprítomný: -

Priebeh stretnutia:
Na tomto stretnutí sme sa snažili zreplikovať všetky kroky na lokálnom školskom PC. Podarilo sa nám úspešne rozbehať všetky prerekvizity, Docker, GNURadio a taktiež aj spraviť odchyt, uloženie do súboru a opätovné načítanie z neho.

Na záver sme si rozdelili časti, kto bude čo vypracovávať do finálneho dokumentu nasledovne:

Nikolas: Dešifrovanie ,popísať packet z čoho sa skladá, čo sa z neho oseká
Branislav: GNURadio, chytanie/čítanie packetu
Miroslav: TTN kľúče + setup + Cayenne
Lukáš: Linux verzia,Docker + dependencies

Výstup stretnutia:

- Finalizácia a rozdelenie úloh na vypracovanie záverečného dokumentu.