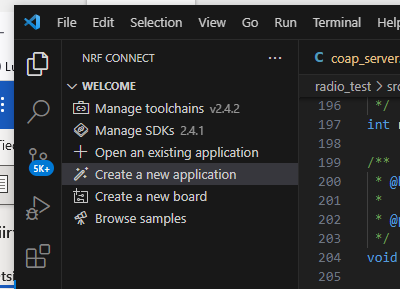
# Thread demo nRF5340dk alustalle

Demossa käytetään kolmea nRF5340dk kehitysalustaa, joista kaksi kommunikoi keskenään Threadin avulla ja yksi kuuntelee viestintää sivusta. Ohjelmina viestivillä alustoilla on Nordicin “OpenThread CoAP server” yhdellä ja “OpenThread CoAP client” toisella. OpenThread on Googlen toteuttama vapaan lähdekoodin Thread ohjelmisto, jonka kehittämisessä Nordic on ollut aktiivisesti yhteistyössä. Ohjelmat pitää kääntää itse valmiista esimerkistä Visual Studio Codessa käyttäen nRF Connect for Desktop lisäosaa. Viestintää kuuntelevalle alustalle pitää asentaa [nRF Sniffer for 802.15.4](https://infocenter.nordicsemi.com/topic/ug_sniffer_802154/UG/sniffer_802154/installing_sniffer_802154.html) jota käytetään Wiresharkilla viestinnän seurantaan.

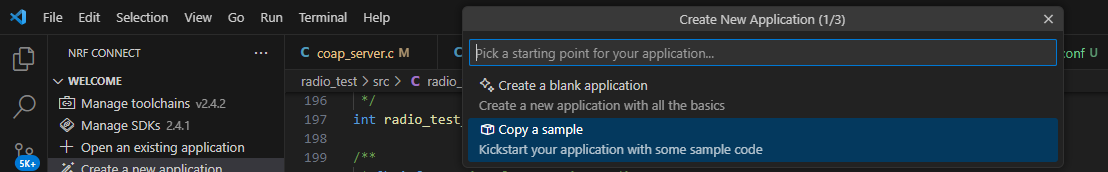
# OpenThread CoAP server asennus

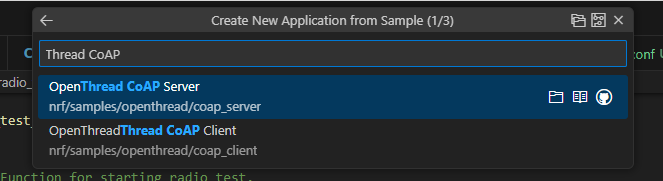
VS Coden nRF Connect for desktop lisäosassa valitse “Create new application”



Kuva1 *Uuden ohjelman luonnin aloitus.*

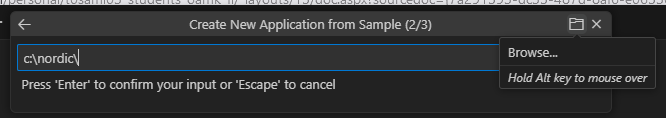
Kun alat luomaan ohjelmaa, VS coden yläreunaan avautuu valikko, jossa ohjataan, minkä pohjalle uusi ohjelma tehdään (kuva2). Tätä esimerkkiä varten tehdään valmis esimerkki, jota ei tarvitse muokata. Valitse “Copy a sample”, jonka jälkeen kirjoita esimerkin hakupalkkiin “Thread CoAP” ja valitse “OpenThread CoAP Server” (kuva3).

Kuva2 *Uusi ohjelma tehdään valmiin esimerkin pohjalle, valitse ”Copy a sample”.*



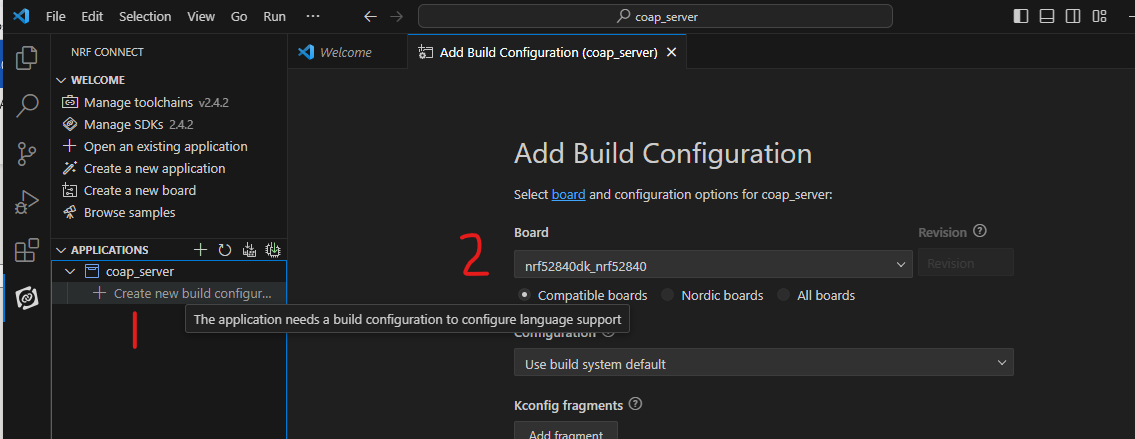
Kuva3 *Ohjelman teko esimerkin pohjalta.*

Valinnan jälkeen valitse kansio, johon haluat ohjelman luoda. Voit kirjoittaa polun suoraan, tai valita sopivan kansion omasta järjestelmästä (Kuva3). Suositus on, että ohjelma olisi lähellä aseman juurta, eikä polku sisällä erikoismerkkejä. Liian pitkä ohjelmapolku voi aiheuttaa ongelmia. Lopuksi kirjoita ohjelmalle nimi.



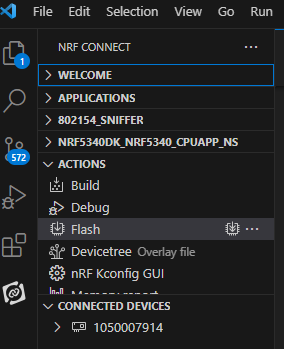
Kuva4 *Asennus kansion valinta*

Kun ohjelman pohja on luotu, sille pitää valita kääntöasetukset (Kuva 5. Kohta 1). Tämä avaa uuden valikko välilehden, josta valitaan mille alustalle ohjelma tehdään (Kuva 5 kohta 2), valitse listalta alustaksi nrf5340dk\_nrf5340\_cpuapp. Muita asetuksia ei tarvitse muuttaa. Paina ikkunan pohjalla olevaa “Build configuration” ja ohjelma käännetään. Kääntämisessä voi mennä useita minuutteja.



Kuva5 *Kääntö asetukset ja alustan valinta.*

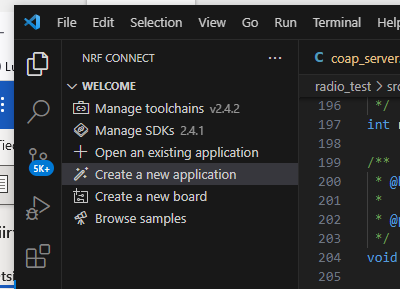
Kun ohjelma on valmis, sen voi ohjelmoida tietokoneeseen kytketylle kehitysalustalle nRF Connect for Desktop lisäosan Actions valikosta Flash komennolla (Kuva 6).



Kuva6 *Ohjelman asennus ohjelmointialustalle*

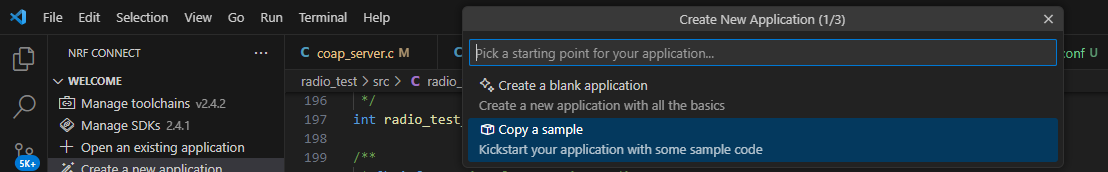
# OpenThread CoAP client asennus

VS Coden nRF Connect for desktop lisäosassa valitse “Create a new application” (kuva6)

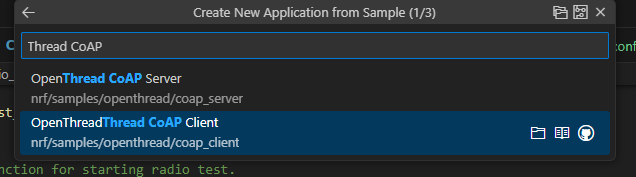


Kuva7 *Uuden ohjelman luonnin aloitus.*

Kun alat luomaan ohjelmaa, VS Coden yläreunaan avautuu valikko, jossa ohjataan, miten uusi ohjelma tehdään (kuva7). Tätä esimerkkiä varten tehdään valmis esimerkki, jota ei tarvitse muokata. Valitse “Copy a sample”, jonka jälkeen kirjoita esimerkin hakupalkkiin “Thread CoAP” ja valitse “OpenThread CoAP Client” (kuva8).

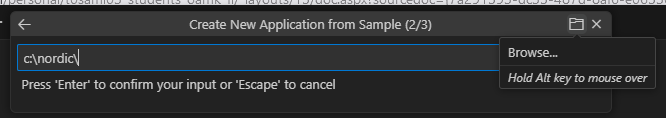
Kuva8 *Uusi ohjelma tehdään valmiin esimerkin pohjalle, valitse ”Copy a sample”*.

Valitse ohjelman pohjaksi OpenThread Coap Client.



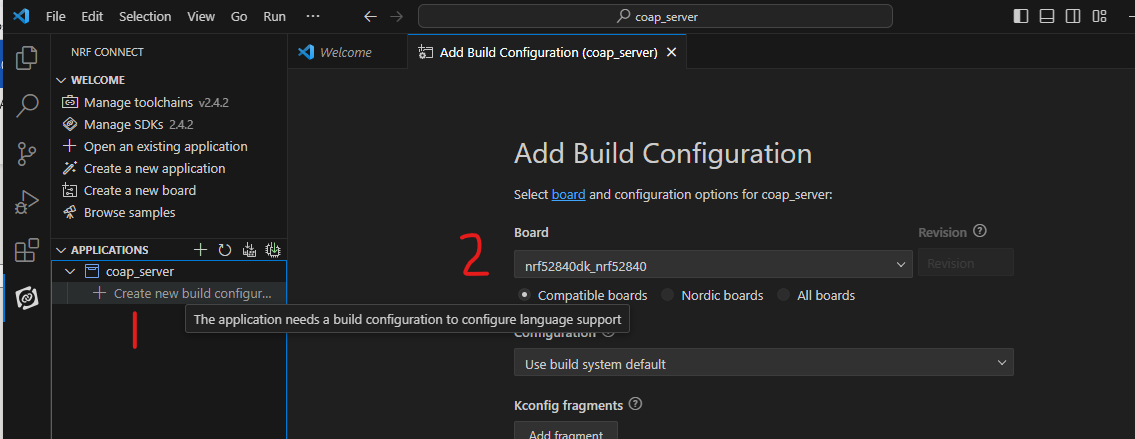
Kuva9 *OpenThread Coap Client esimerkkiohjelman valinta*.

Valinnan jälkeen valitse kansio, johon haluat ohjelman luoda. Voit kirjoittaa polun suoraan, tai valita sopivan kansion omasta järjestelmästä (Kuva9). Suositus on, että ohjelma olisi lähellä aseman juurta, eikä polku sisällä erikoismerkkejä. Liian pitkä ohjelmapolku voi aiheuttaa ongelmia. Kansion valinnan jälkeen ohjelmalle nimi.



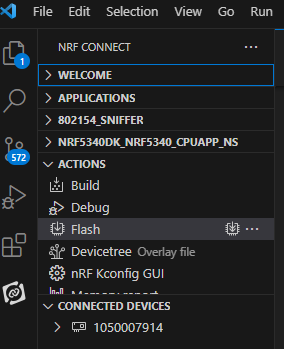
Kuva10 *Ohjelman kansion valinta*

Kun ohjelman pohja on luotu, sille pitää valita kääntöasetukset (Kuva 5. Kohta 1). Tämä avaa uuden valikko välilehden, josta valitaan mille alustalle ohjelma tehdään (Kuva 5 2), valitse listalta nrf5340dk\_nrf5340\_cpuapp. Muihin asetuksiin ei tarvitse koskea, valitse pohjalta “Build configuration” ja ohjelma käännetään. Kääntämisessä voi mennä useitakin minuutteja.



Kuva11 *Kääntö asetukset ja alustan valinta.*

Kun ohjelma on valmis, sen voi ohjelmoida tietokoneeseen kytketylle kehitysalustalle nRF Connect for Desktop lisäosan Actions valikosta Flash komennolla (kuva 12).



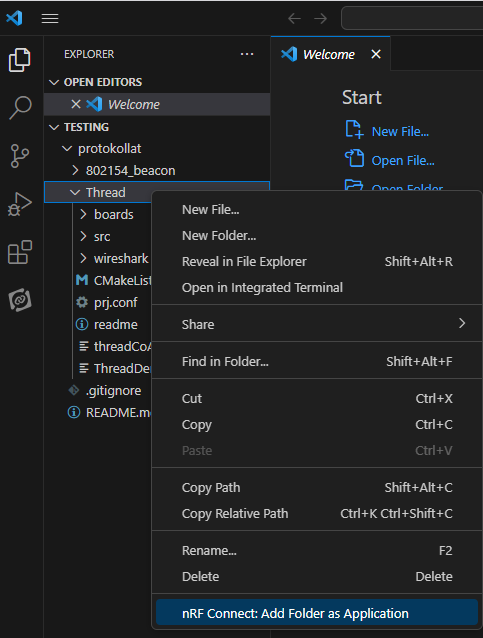
Kuva12 *Ohjelmointi alustan ohjelmointi*

# Sniffer for 802.15.4

Kehitysalustan käyttäminen viestinnän seurantaan luettavassa muodossa, vaatii pakettien kaappauksen lisäksi niiden purkamista. Viestien kaappaamiseen käytetään Nordicin tekemää kuuntelijaa “Sniffer for 802.15.4” josta on toimiva versio julkaistu tällä hetkellä vain nRF52840dk alustalle tai siihen pohjautuvaan USB-mokkulalle tarkoitettu. Ohjelman julkaistu koodi toimii pienellä muutoksella myös nRF5340dk alustalla, joka on tehty valmiiksi githubissa mukana olevaan versioon. Viestinnän purkamiseen käytetään Wireshark-ohjelmaa, joka on verkkoliikenteen seurantaan tehty ja sisältää viestinnän purkumetodit myös 802.15.4 protokollan yli menevälle viestinnälle.

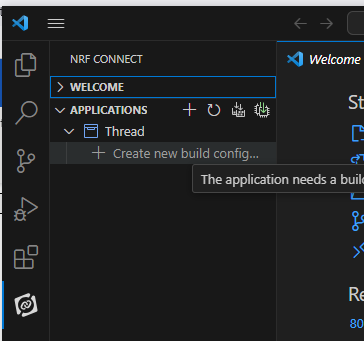
## Sniffer for 802.15.4 ohjelmointi

Hae ohjelman kansio githubista <https://github.com/oamk-yritysprojekti-jlink/protokollat>. Avaa kansio “Thread” VS Codessa ja siellä lisää kansio sovelluksena nRF:Connectiin(kuva 13).

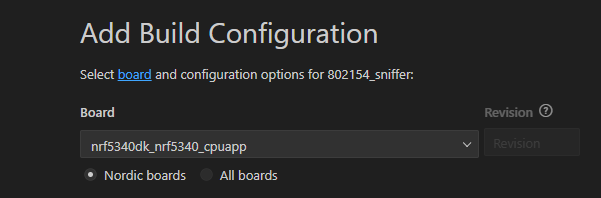


Kuva13 *Kansion lisääminen ohjelmaksi nRF Connect lisäosalle*

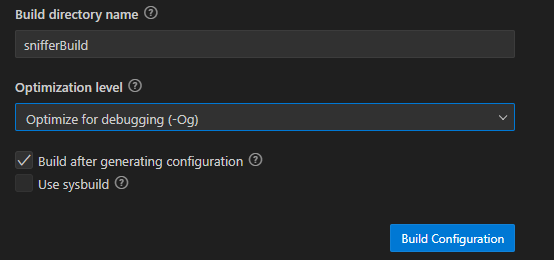
Tee ohjelmalle uudet käännösasetukset (Kuva 14). Valitse ohjelmointialustaksi nrf5340dk\_nrf5340cpuapp (Kuva 15) ja muuta käännöksestä koodin optimointi tasoksi “Optimize for debugging (-Og)” (Kuva 16). Muita asetuksia ei tarvitse muuttaa, mutta kansion nimi kannattaa vaihtaa kuvaavaksi, jos tekee käännöksiä eri asetuksilla.



Kuva14 *Uusien käännösasetusten luonti*

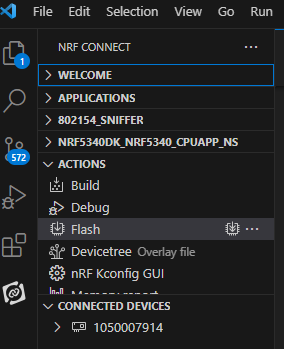


Kuva15 *Ohjelmointialustan valinta*



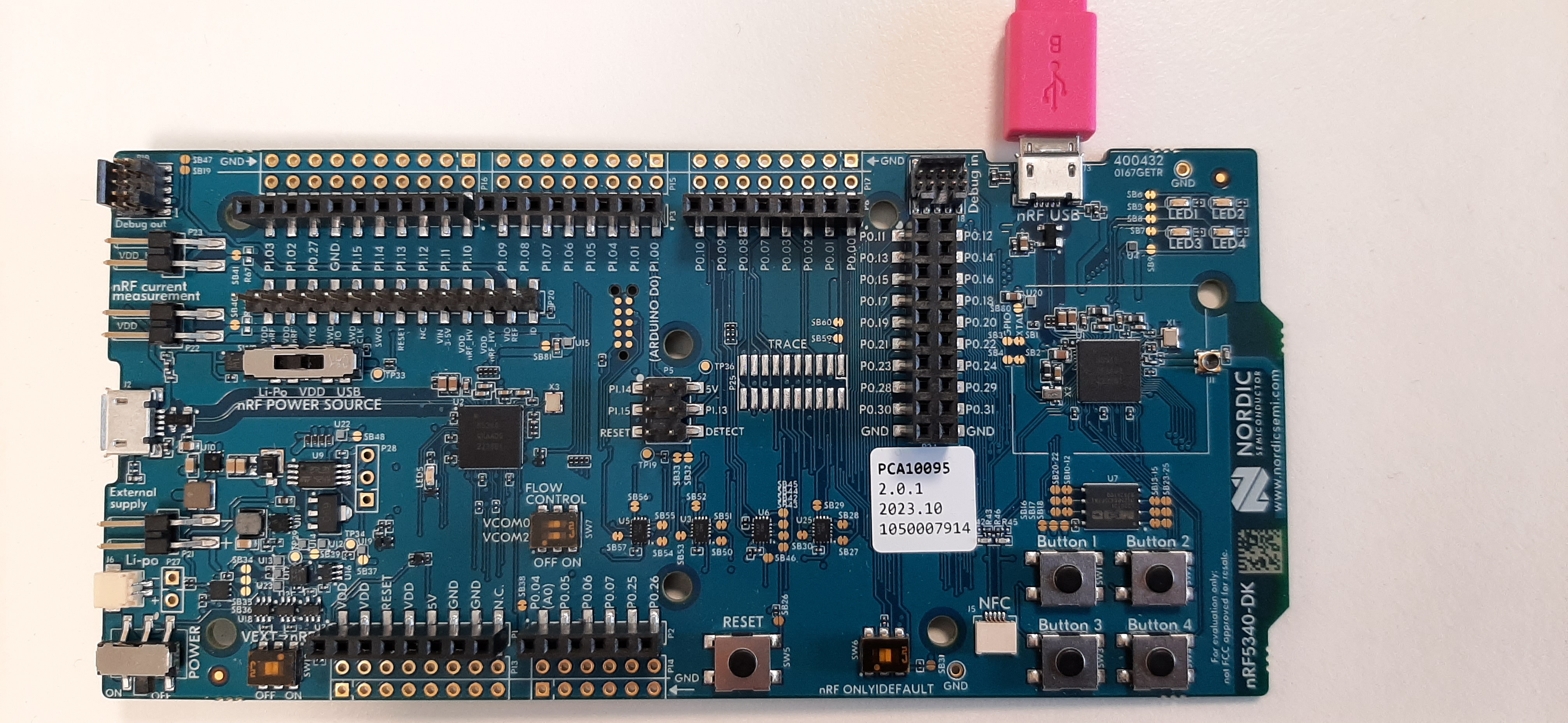
Kuva16 *Optimointi tason määritys virheiden etsintään sopivaksi*

Kun ohjelma on rakennettu, sen voi ohjelmoida ohjelmointi alustalle Flash komennolla (Kuva 17)



Kuva17 *nRF5340dk alustan ohjelmointi luodulla ohjelmalla*

Kun ohjelma on siirtynyt ja käynnistynyt se viestii vain “nRF USB”-portin kautta (kuva 18). Voit kokeilla ohjelmaa ottamalla sarjaporttiin yhteyden esim PuTTY:llä seuraten kappaleen “PuTTY yhteys” ohjeita. Yhteyden oton jälkeen “receive” komento käynnistää radion ja “sleep” sammuttaa radion.

Kuva18 nRF5340dk, jossa mikroUSB johto nRF USB portissa.

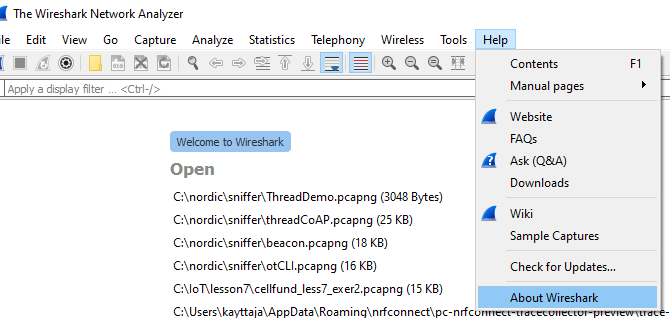
## WireShark käyttöönotto

Asenna Wireshark, jos sitä ei vielä ole asennettu. Se löytyy osoitteesta https://www.wireshark.org/download.html

Wireshark tarvitsee lisäosan, jotta sillä voi lukea suoraan USB-väylästä snifferin kaappaamaa tietoa. Lisäosan käyttöä varten tarvitaan Python 3.7 tai uudempi ja siihen pySerial moduuli. Python asenuspaketti löytyy osoitteesta https://www.python.org/downloads/, jos se puuttuu. Kun python on asennettuna, pySerialin voi asentaa komennolla:

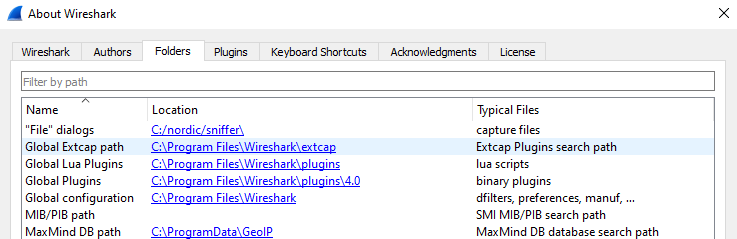
pip install pyserial

Kun pySerial on asennettuna, kopioi lisäosan tiedostot Wiresharkin extcap kansioon. Kansion polun löytää wiresharkin sisällä Help->About Wireshark (Kuva 19) valikon alta.



Kuva19 *About Wireshark valikon avaus*

Avaa Folders välilehti ja etsi listalta “Global Extcap path” (kuva 20), josta Wireshark hakee lisäosia ulkoisia kaappauslaitteita varten. Kansion voi avata suoraan kaksoisklikkaamalla alleviivattua polkua.

Kuva20 *Lisäosakansion polku ulkoisia kaappauslaitteita varten*

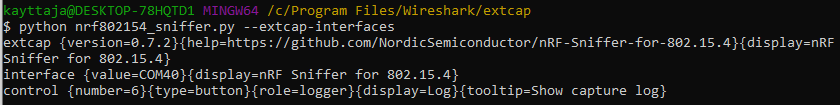
Kopioi tähän kansioon sniffer ohjelmiston wireshark kansiosta nrf802154\_sniffer.py tiedosto. Voit varmistaa, että lisäosa toimii oikein ajamalla tässä kansiossa Windows koneella komennon:

python nrf802154\_sniffer.py --extcap-interfaces

Mac ja Linux koneilla toimivuuden voi varmistaa komennolla:

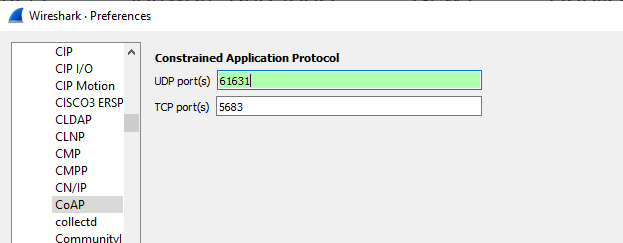
nrf802154\_sniffer.py --extcap-interfaces

Jos kaikki on kunnossa, vastaus komentoon on kuvan 21 kaltainen. Extcap versio kertoo python koodi version, interface listaus sisältää sarjaportin, jossa laite on, ja laitteen ohjelmiston nimen.

Kuva21 *Lisäosan tiedot wiresharkia varten*

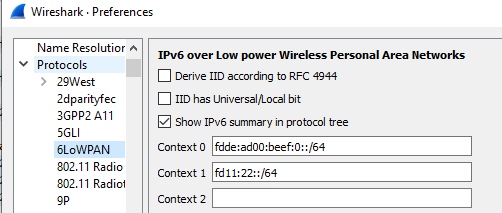
Wiresharkin asetuksia täytyy muuttaa, että se tulkitsee kaapattua tietoa oikein ja purkaa salauksen. Avaa Edit valikon alta Preferences. Protokolla listasta avaa IEEE 802.15.4. Avaa Decryption keys vieressä olevasta Edit(kuva 22, 1) painikkeesta tunnetut salausavaimet. Lisää avaimiin “+” painikkeesta (kuva 22, 2) esimerkin käyttämä avain “00112233445566778899aabbccddeeff”. Varmista että avaimen “Decryption key index” on 0 ja “Key hash” on tyyppiä Thread hash. Tallenna muutokset painamalla ikkunasta OK. Kuva22 IEEE802.15.4 salauksenpurku avaimen lisäys.

Koska halutaan nähdä myös verkon hallinta, täytyy asettaa Wiresharkille asetukset Threadin käyttämää CoAP porttia varten. Valitse protokolla listasta CoAP. Muuta UDP portiksi 61631, tai lisää 61631 pilkulla erotettuna listalle.

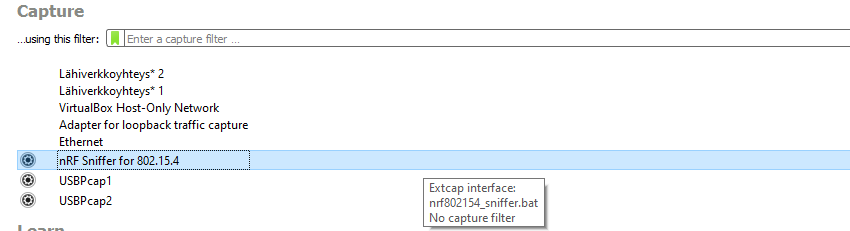


Kuva23 *Coap portin muuttaminen.*

6LoWPAN käyttää lyhennettyjä IPv6 osoitteita, joiden tulkitsemiseen tarvitaan konteksti, jotta osoitteet tulkitaan varmasti oikein. Avaa protokolla listasta 6LoWPAN ja lisää sinne tarvittavat rivit. Context 0 riville lisää “fdde:ad00:beef:0::/64” ja Context 1 riville lisää “fd11:22::/64”.

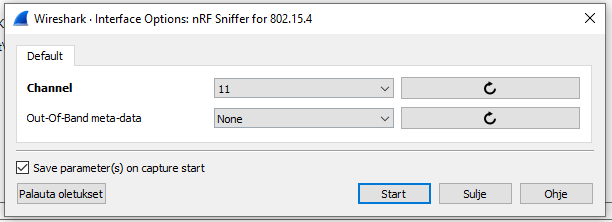


Kuva24 *6LoWPAN konteksti IPv6 osoitteiden lyhentämiseen.*

Ohjelmointialusta pitää kytkeä tietokoneeseen siinä olevasta nRF USB portista (kuva 18), jolloin tietoliikenteen ei tarvitse kulkea ohjelmointipiirin läpi. Kun laite on ohjelmoitu ja Wireshark lisäosa on oikein asennettu, Wiresharkin alkunäkymässä “Capture” listauksessa näkyy nRFSniffer for 802.15.4. jota kaksoisnapauttamalla kaappaus alkaa (kuva 25). 

Kuva25 *Wireshark ohjelmalla kuuntelun aloitus.*

Wireshark voi kysyä kuuntelun aloituksessa käyttöliittymä asetuksia, joissa pitäisi olla alla olevat arvot (kuva 26). Paina Save tai Start ja kuuntelu alkaa. Wiresharkilla voi mennä pitkään ennen kuin se vaihtaa näkymää ja vastaa ohjelmana, jos nRF5340dk levyn LED1 vilkkuu nopeasti, kuuntelu on alkanut ja tapahtumat näkyvät wiresharkissa. Lepotilassa LED:n tila muuttuu sekunnin välein, kuuntelutilassa LED:n tila muuttuu neljä kertaa sekunnissa.



Kuva26 Wireshark kuunteluasetuksien muuttaminen.

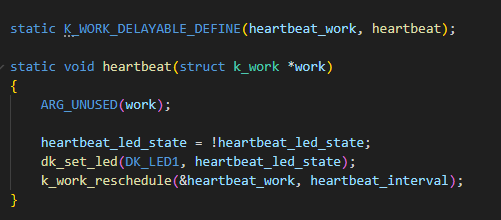
## Ohjelman toiminta

Sniffer ohjelmisto koostuu kolmesta osasta, alustan laitetiedoista, alustan ohjelmasta ja wireshark lisäosasta. Laitetiedot tulevat zephyr kirjaston kautta, kun ohjelma käännetään, osittain prj.conf tiedoston ohjaamina ja osaksi käännösasennuksessa valitun alustan perusteella. Alustan varsinainen ohjelma on main.c -tiedosto. Se sisältää laitteelle ohjelmoitavan pääohjelman, joka ohjaa LED vilkuntaa, käsittelee verkkoliikenne paketit sekä radion ohjauskomennot ja säätää radion asetukset. Python-tiedosto nrf802154\_sniffer.py luo wiresharkiin käyttöliittymän sniffer ohjelmistolle, kirjoittaa kehitysalustan antamat tiedot oikeaan muotoon wiresharkille ja antaa laitteelle ohjauskomentoja radion käynnistämiseen, sammuttamiseen ja kanavan valitsemiseen.

### Main.c

Kohta 1. sisältää

dk\_leds\_init() valmistelee LED käytön. uart\_shelliksi haetaan pointteri, joka ohjaa kirjoitukset oikeaan porttiin. Heartbeat\_interval asetetaan lepotilassa käytettyyn sekunttiin ja K\_work\_reschedule funktio uudelleen ajastaa heartbeat\_workin tapahtumaan sekunnin päästä, joka vain suorittaa heartbeat funktion.



Heartbeat funktio vaihtaa ledin tilaa ja ajastaa funktion uudestaan heartbeat\_interval ajan päähän.

Seuraavaksi kuvassa kohta 2. main luo ajuri asetukset 802.15.4 käyttöliittymälle promiscuous asetuksella, jolloin kaikki laitteen näkemä viestintä ohjataan korkeammalle tasolle, tyypistä tai kohteesta huolimatta. Normaali tilassa viestit, joilla on tuntemattoman tyyppi, ACK eli vahvistus vastaanotosta, sekä viestit, joiden kohde osoite ei ole tällä laitteella jätetään ohjaamatta.

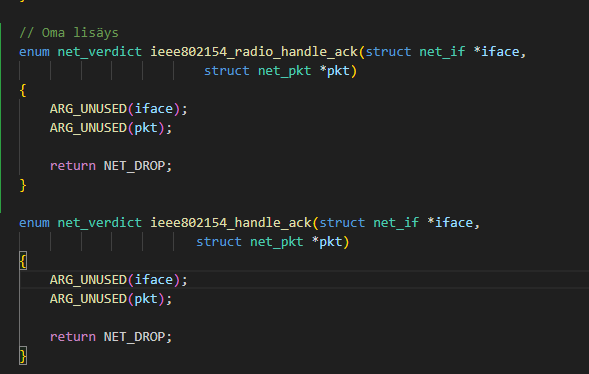
Kolmanneksi main ohjelmassa radio\_api asetetaan muistiosoite 802.15.4 radion apiksi käytetyllä laitteella. Neljäs vaihe ajaa tehdyt asetukset radiolle.

Yleisen verkkoliikenteen käsittele net\_recv\_data. Ei tyhjistä paketeista jäsennellään tiedot kuva kohta 1:

* Data sisältö
* Paketin koko
* Linkin laatu indikaattori, LQI (Link Quality Indication)
* Signaalin voimakkuus, RSSI (Received Signal Strength)
* Paketin aikaleima, joka muutetaan sopivaan muotoon

Jäsentelyn jälkeen vaihdetaan LED4 tilaa, merkiksi että paketti on vastaan otettu. Kolmas kohta kuvassa, paketin data ja koko muutetaan heksastringiksi ja tulostetaan UART-väylään luotu heksa, RSSI, LQI ja aikaleima. Lopuksi vapautetaan paketti.

handle\_ack funktiot käsittelevät laitteen vastaan ottamat kuittausviestit, jotka normaalisti eivät tulisi ohjelma tasolle.



Viimeinen osa ohjelmaa on laitteen tilan hallintaan luodut komento funktiot: cmd\_channel, cmd\_sleep ja cmd\_receive. Jokaiselle näistä luodaan SHELL\_CMD\_ARG\_REGISTER makrolla oma root tason komento, joita käyttämällä Wireshark ohjaa laitetta. Komennolla cmd\_channel luetaan nykyinen kanava, jos ei ole ylimääräistä argumenttia tai vaihdetaan kanava, jos uusi kanava numero on komennon argumenttina annettu. Molemmat cmd\_sleep ja cmd\_receive ovat radiota ohjaavia, sleep sulkee radion ja receive käynnistää sen.

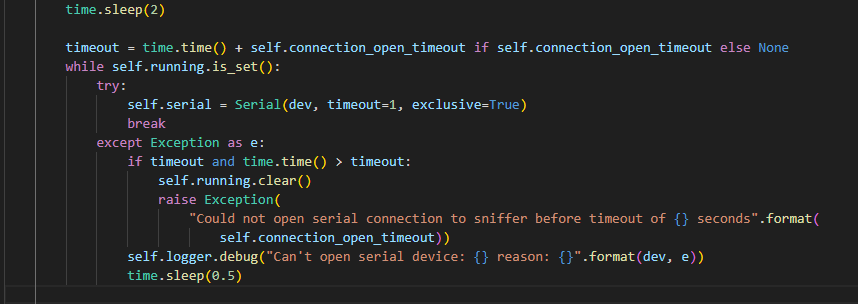
### nrf802154\_sniffer.py

Python koodi nrf802154\_sniffer.py on Wiresharkille komentoja kehitysalustan ohjaukseen, mutta sillä voi myös erillisenä ohjelmana tarkistaa onko laitteessa toimiva sniffer ohjelmisto. Ohjelma toimii oliona ja kommunikoi kehitysalustan kanssa UART shellin läpi.

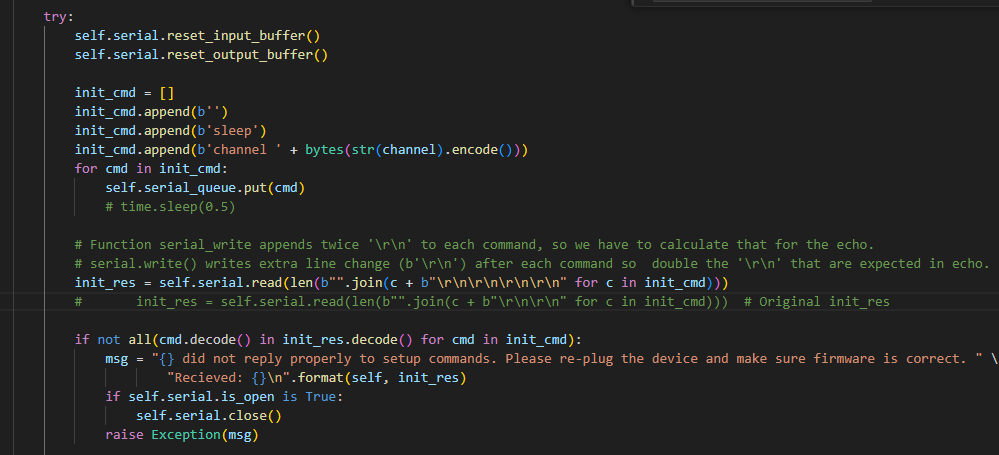
Tiedoston alussa alkaa luokan määritelmä Nrf802154Sniffer luokalle. Luokan init funktio luo luokasta tehtävälle oliolle tarvittavat muuttujat nollatuilla arvoilla.

Pääfunktio extcap\_capture jonka Wireshark kutsuu lisäosan käynnistyksessä ottaa vastaan laitetiedot ja kuunneltavan kanava. Se myös käynnistää muut tarvittavat säikeet, jotka hallitsevat viestien vastaanottamista alustalta, alustalle komentojen lähetystä ja saatujen tietojen muokkaus pcap-muotoon. Jos ohjelma ajetaan itsenäisenä ohjelmana, jää pyörimään lopuksi silmukkaan, kunnes näppäimistökeskeytys tai laite sammutetaan. Jos laite on etäohjauksessa, käynnistetään myös säie control\_reader joka tulkitsee ohjauskomentoja etälähteestä. Tämä toiminto ei vielä ole loppuun asti tehty Wiresharkissa.

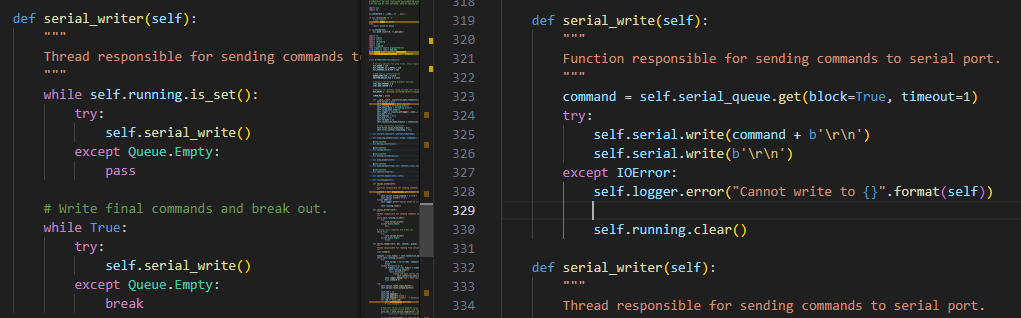
Säie ja funktio portin lukemiseen on serial\_reader. Funktion ensimmäinen osa kuva XX yrittää ottaa yhteyden sarjaporttiin. Sekunnin välein, jos yhteyttä ei saada muodostettua, nostaa poikkeuksen ja yrittää puolen sekunnin päästä uudestaan. Kun yhteys onnistutaan luomaan, siirrytään seuraavaan osaan.

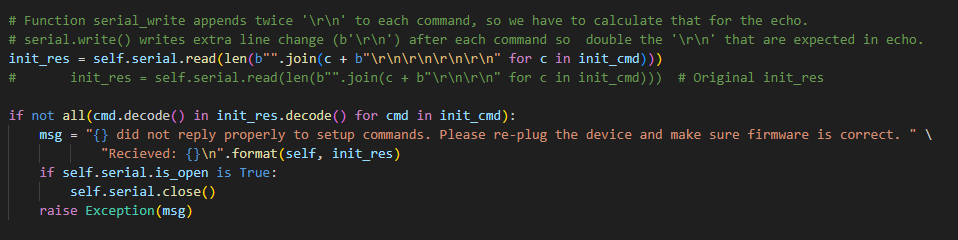


Seuraava vaihe kuva XX on bufferien tyhjennys, että viesti ei sisällä ylimääräisiä osia. Ja komentojonon luonti, jossa init\_cmd lista täytetään komennoilla, jotka halutaan ajaa. Komennot asetetaan serial\_queueen jotta serial\_writer thread lukee.



Serial\_writer säie on silmukassa, kunnes ohjelma lopetetaan ajamassa serial\_write funktiota, jolla lähetetään viestit serial\_queuesta. serial\_write funktio ottaa jonosta komennon ja yrittää kirjoittaa komennon ja rivinvaihdon sarjaporttiin. Jos kirjoitus ei onnistu, logger tulostaa virheilmoituksen.

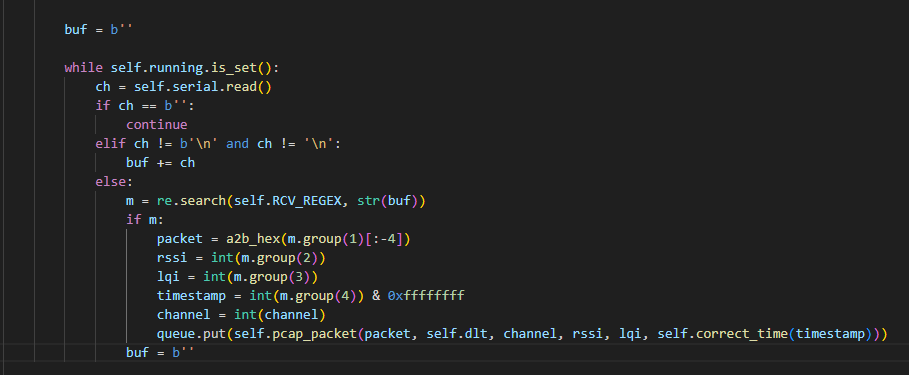


Kun viestit on kirjoitettu sarjaporttiin, luodaan vertailukappale odotetuista vastauksista. Ja verrataan saatua vastausta siihen. Vastaus ei ollut oikea, suljetaan sarjaportti ja tehdään virheilmoitus sekä tulostetaan ongelman kuvaus. 

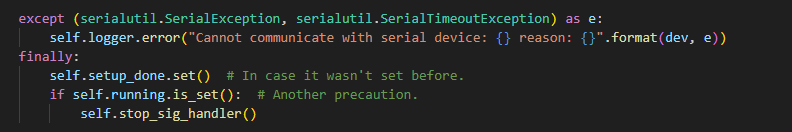
Jos vastaus oli oikea, annetaan ohjelmointi alustalle kuuntelun aloitus käsky “receive” ja asetetaan setup\_done ominaisuus, joka on yksi määrittelyistä, joilla katsotaan, onko ohjelma toiminnassa.



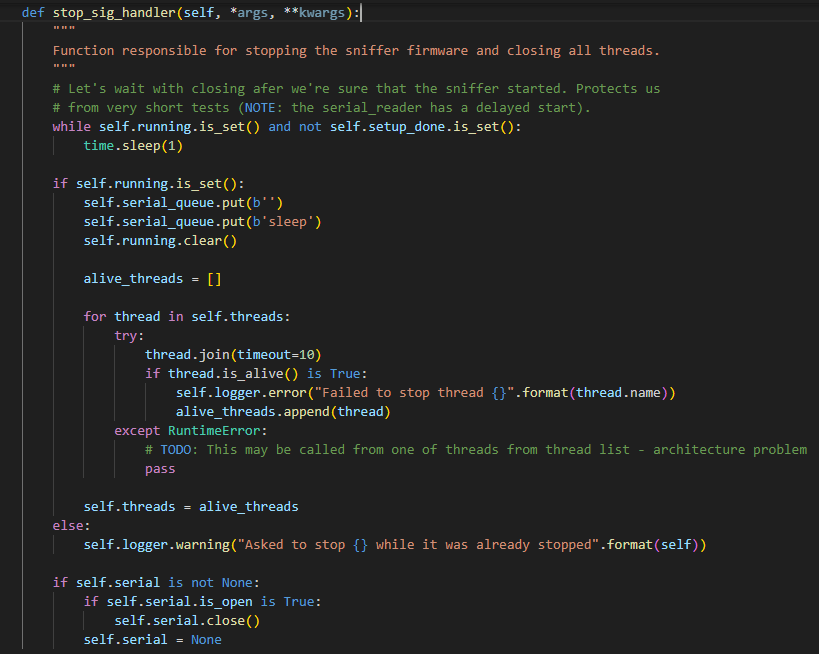
Seuraavassa osiossa luodaan puskuri ja silmukassa kuunnellaan nRF5340dk:n viestejä niin kauan kun ohjelma on toiminnassa.



1. Jos portissa ei ole mitään, mennään silmukan alkuun.
2. Jos portissa on muuta kuin rivinvaihto, se lisätään puskuriin, mennään silmukan alkuun.
3. Nyt luettiin rivinvaihto, joten on vastaanotettu kokonainen viesti ja data jäsennellään.
4. Käytetään re.search funktiota, että löydetään puskurista oikein muotoiltu viesti, eikä muuta.
5. Jos puskurissa oli viesti, jaetaan siitä paketti binäärinä, RSSI, LQI, aikaleima ja viestintäkanava.
6. Tieto asetetaan jonoon odottamaan tulostusta pcap tiedostoksi Wiresharkille käyttäen jäsentely funktiota pcap\_packet
7. Puskuri asetetaan tyhjäksi ja palataan silmukan alkuun



Jos sarjaporttikommunikaatiossa on virhe, siitä tehdään virhelogiin ilmoitus ja siirrytän “finally:” osaan. finally ehto suoritetaan aina ennen kuin try except rakenne loppuu. Siellä kutsutaan stop\_sig\_handler funktio sulkemaan sarjaportti yhteys ja sniffer ohjelma oikein.

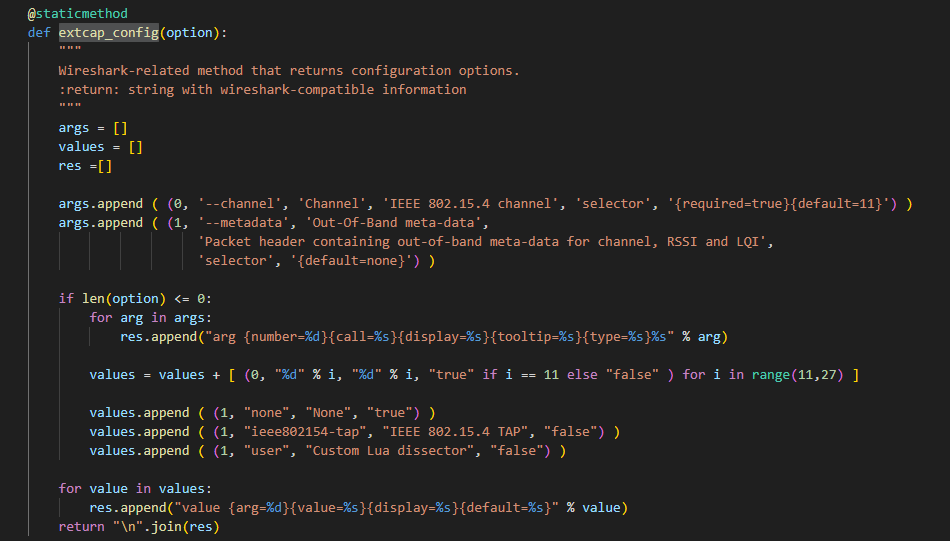


Funktio aluksi odottaa sekunti kerrallaan, että käynnistys tehdään loppuun asti, eli setup\_done asetettu tai ohjelma ei ole toiminnassa eli self\_running ei ole asetettu. “if self.running.is\_set()” tarkistaa onko ohjelma toiminnassa. Serial komennot lähettävät käskyn ohjelmointi alustalle sammuttaa radio. Self.running.clear() asettaa ettei ohjelma ole toiminnassa, jotta seuraavat komennot lopettaa toiminta eivät turhaan toista sulkemista, ellei ohjelma käynnistettä uudelleen. Jokaiselle säikeelle ajetaan thread.join(timeout=10) joka yrittää sulkea säikeen antaen sulkemiselle aikaa 10 sekuntia. Tämän jälkeen, jos säie ei sulkeutunut lokitietoihin lisätään virhe ja se lisätään elossa oleviin säikeisiin. Lopuksi jos sarjaporttiyhteys on käynnissä, se suljetaan.

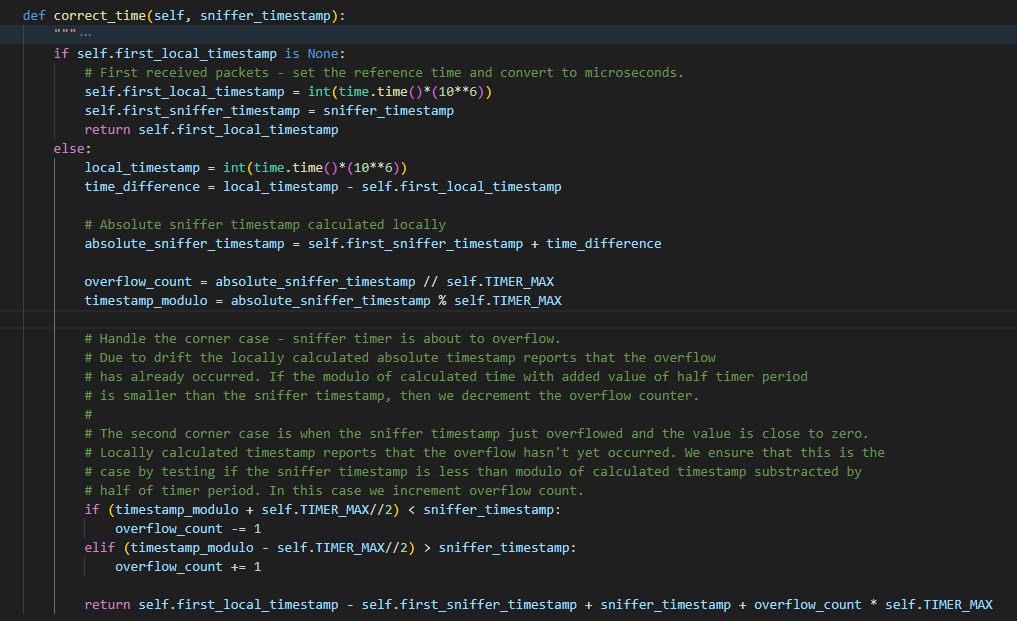
Viimeinen säie on fifo\_writer, joka käsittelee tiedot jonosta pcap-tiedostoksi jota Wireshark lukee. Säikeen luonnissa “fifo” on Wiresharkin pcap-lokitiedosto ja queue pakettijono johon serial\_reader säie lisää kehitysalustan lähettämät verkkoliikennetiedot. Komennolla open avataan tiedosto bitti kirjoitusta varten. Aluksi kirjoitetaan header jotta tiedetään millaisessa formaatissa tiedot ovat.



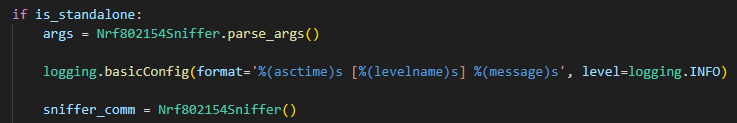
Wiresharkin käyttöön tehty extcap\_config funktio antaa ulos asetukset, joita laitteella voi käyttää ja mikä on oletus asetuksena.



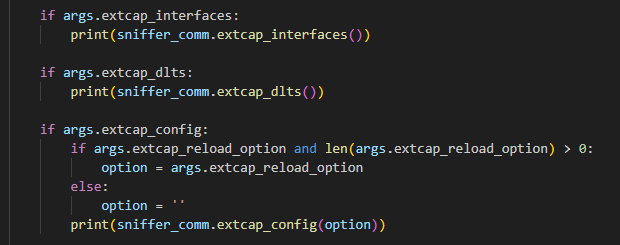
Oikean ajan saamiseen tehty funktio correct\_time korjaa annetun aikaleiman, jos sniffer on ollut käynnissä tarpeeksi kauan aikalaskurin ympärimenemiseksi. Funktio hakee oikean ajan tietokoneelta ja vertaa sitä oikeaan laitteen aikaan. Ensimmäisen kerran funktio asettaa first\_local\_timestamp ja first\_sniffer\_timestamp arvot myöhempää vertailua varten. Muilla kerroilla funktio hakee tietokoneen ajan ja laskee eron ensimmäiseen. Eroa käytetään laskemaan absoluuttinen snifferin aikaleima. Ylivuotojen määrä lasketaan jakamalla absoluuttinen snifferin aikaleima aikarekisterin koolla, saadaan kokonaiset ylivuodot. Modulo aikaleimasta lasketaan poikkeustapauksia varten, joita sattuu, kun oikea aika ajautuu erilleen snifferin ajasta pienten epätarkkuuksien myötä. Ensimmäinen poikkeus, kun oikean ajan mukaan on jo ylivuoto laskettu, mutta modulo ja puolet maksimi ajasta on pienempi kuin snifferin oma aikaleima, eli sniffer ei ole vielä ylivuotanut, vähennetään yksi ylivuoto. Toinen poikkeustapaus on, kun snifferin aikaleima on juuri ylivuotanut ja arvo on nollan lähellä, mutta tietokoneen ajasta laskemalla ylivuotoa ei ole tullut. Varmistetaan laskemalla, jos modulo aikaleimasta miinus puolet ajastimen maksimi arvosta on suurempi kuin snifferin aikaleima, vähennetään yksi ylivuoto.



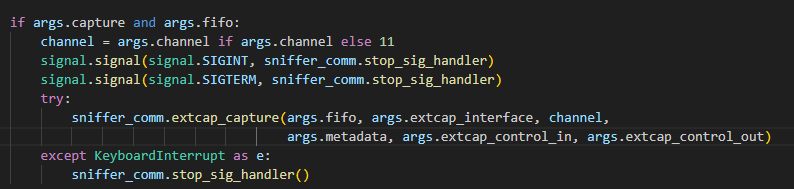
Tiedoston lopussa on osa, joka suoritetaan, kun python ohjelmaa ajetaan sellaisenaan mahdollisesti jollain komentorivi argumenteilla. Ensin se käsittelee annetut argumentit, asettaa lokin kirjoitustason ja luo olion, jolla ohjataan alustaa.



Argumenttien käsittelyssä ensin tarkistetaan, onko tulostuksen vaativia argumentteja extcap-interfaces, extcap-dlts tai extcap-config.

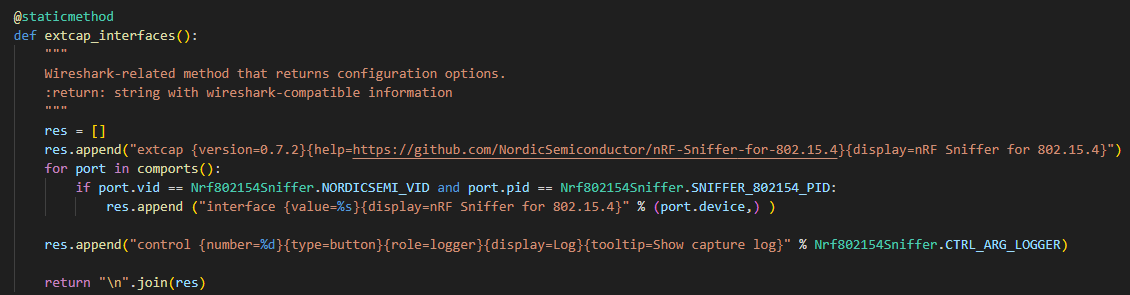


Viimeiseksi tarkistetaan, onko argumentteina kuuntelun käynnistys käsky ja tiedostopolku johon loki kuuntelusta tallennetaan. Tiedosto määrätään “--fifo” komennolla

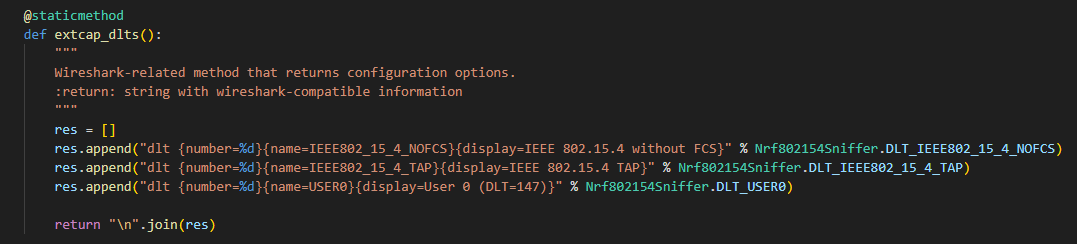


Tallennusaloitetaan kanavalta 11, ellei argumenteissa ollut muuta määrätty. Signal komennot ohjaavat näppäinkeskeytyskomennon ja ohjelman terminointi signaalin stop\_sig\_handler funktioon, jotta laite suljetaan oikein. Sitten yritetään aloittaa kuuntelu.

Muut osat ohjelmasta ovat komentorivi argumenttien käsittelyyn, joilla ohjataan ohjelman toimintaa. Parse\_args käytetään käsittelemään annetut argumentit, samalla se varmistaa, että kuuntelun aloituksessa on valittuna myös käyttöliittymä, eli kuunteleva laite.

Extcap\_interfaces palauttaa tekstinä tietoa ohjelmasta ja lisää porttien tiedot, joissa on sniffer-ohjelmalla varustettu kehitysalusta.

Extcap\_dlts funktio antaa asetukset, joita nordicin alusta tukee. Dlt == Diagnostic Log and Trace

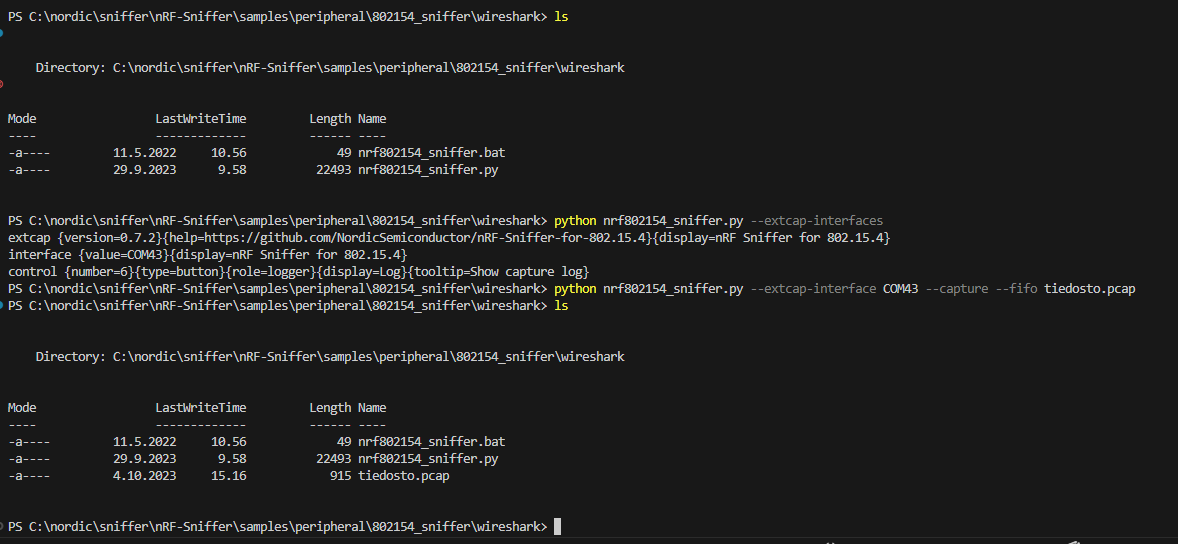


Python ohjelmaa voi käyttää itsessään snifferiksi ohjelmoidun nRF5340dk-alustan ohjaukseen ja loki tiedoston luomiseen myöhempää avausta varten.

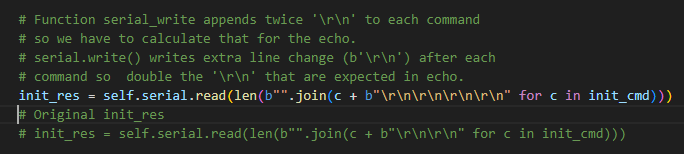
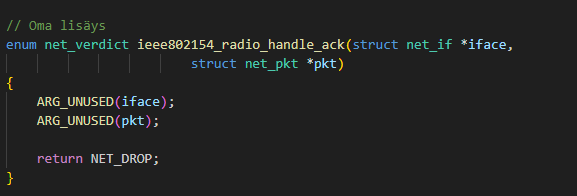
python nrf802154\_sniffer.py --extcap-interfaces

python nrf802154\_sniffer.py --capture --fifo a.pcap --extcap-interface COM43

Ensimmäinen komento tulostaa mahdollisen sniffer laitteen osoitteen. Sieltä valitaan toiseen komentoon laitteen COM-portti käyttöliittymäksi ja lisätään --fifo:lla tiedosto johon loki tallennetaan.



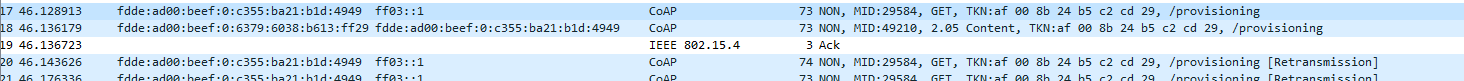
## Sniffer ohjelmistoon tehdyt muutokset

* prj.conf tiedostosta CONFIG\_USB\_DEVICE\_PID = 0x154a (oli 0x154b)
* nrf802154\_sniffer.py tiedoston serial\_reader funktion vastauksen tarkistus
  + 
* Main.c tiedostoon lisättiin käytöstä poistuneelle funktiolle oma määrittely
  + 

# Demon testaus

Kuuntelevassa koneessa käynnistä Wireshark ja valitse lähteeksi nRF Sniffer for 802.15.4. käynnistä ohjelmointialustat, joihin on asennettu OpenThread CoAP ohjelmat. Kun alustoissa syttyy LED1 ne ovat valmiita Threadin yli kommunikointiin. Aloita parin muodostus alustalla, jossa on server-ohjelmisto napista 4, LED4 alkaa vilkkua merkiksi. Kun server hakee paria Client alustalla paina nappia 4, jolloin ne parittuvat ja server lopettaa parin hakemisen. Nyt client nappi 1 lähettää CoAP PUT komennon /light osoitteeseen serverillä ja muuttaa siellä LED2 tilan päälle tai pois.

Wiresharkin Info sarakkeessa näkyy “/provisioning” parinmuodostus vaiheessa ja “/light” kun valojen tilan muutos käsky lähetetään (Kuva 26).



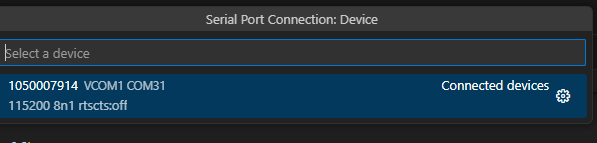
Kuva26 *Wiresharkin info sarakeen tilanne.*

Client alustan nappi 2 pitäisi muuttaa kaikkien kuulolla olevien palvelin alustojen LED2 tilan.

UDP viestin lähetys laitteelta toiselle onnistuu esim PuTTY:llä tai VS Codella. VS Codessa nRF Connect lisäosan Connected devices –osiosta voi ottaa yhteyden ohjelmointialustaan. Avaa laitteen tiedot nuolesta (kuva 27, 1) ja valitse VCOM1 portin yhdista RTT vaihtoehto (Kuva 27, 2)

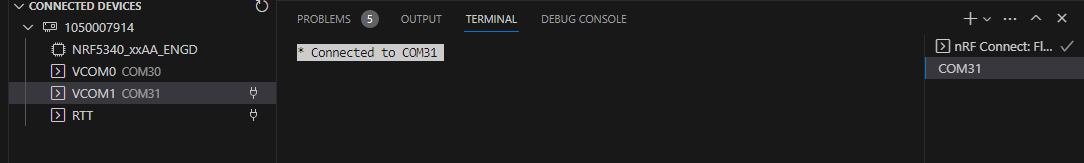
Kuva27 *Konsoli yhteyden muodostus ohjelmointialustaan sarjaportin kautta*

Joudut valitsemaan laitteen vielä uudestaan ikkunan yläreunaan avautuvasta palkista (kuva 28)



Kuva28 *Laitteeseen yhdistäminen sarjaportin kautta lisävalinta ikkuna*

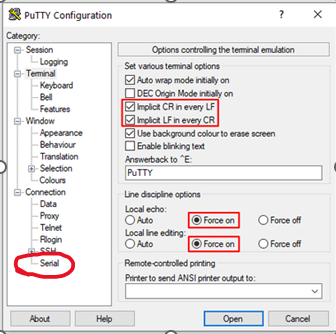
Yhdistyksen jälkeen aukeaa uusi terminaali VS Coden alareunassa, johon voi kirjoittaa komennot laitteen ohjaamiseksi (Kuva 29). Enter painamalla näkyy terminaalissa “uart:~$ “

Kuva29 *Terminaalin ikkunan avautuminen*

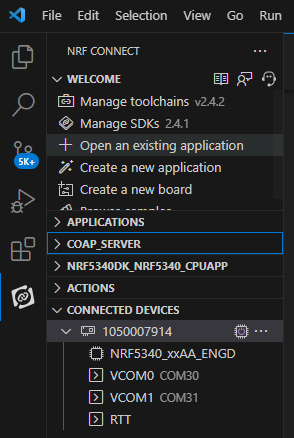
## PuTTY yhteys

Jos haluat käyttää PuTTYä, valitse Connection Type Serial ja valikon Terminal ja Connection -> Serial kohdista muuta asetukset Nordic Semiconductorin ohjeen ja kuvan 30 mukaisiksi:

* Baud rate: 115200 (Speed)
* 8 data bits
* 1 stop bit
* No parity
* HW flow control: None

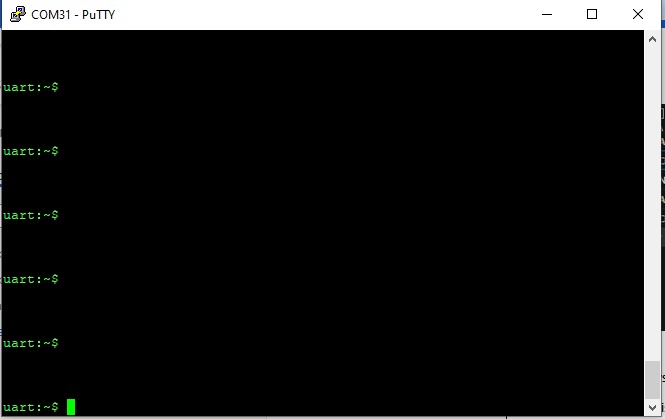


Kuva30 *PuTTY asetuksia sarjaportti yhteyttä varten*

Oikea COM portti löytyy esim VS Codella liitetyistä laitteista (Kuva 31, laitehallinnasta portit valikosta tai komentokehotteessa komennolla ”reg query HKLM\HARDWARE\DEVICEMAP\SERIALCOMM”. VCOM1 on ohjelmaytimen virtuaaliportti johon yhteys pitää muodostaa. Kun asetukset ovat oikein, Open avaa yhteyden. Porttinumero näkyy myös laitehallinnassa Portit valikon sisällä.

Kuva31 *VS Codessa nRF Connecti -lisäosan tunnistamat yhdistetyt laitteet*

Kun yhteys on saatu laitteeseen, avautuu musta ruutu, paina enter ja ruudulla pitäisi lukea “uart:~$”, kuten kuvassa 32



Kuva32 *PuTTY terminaali, kun yhteys on luotu*

## UDP viestin lähetys

Yhdistämisen jälkeen aja komennot UDP:n avaamiseksi ja ip osoitteen selvittämiseksi molemmissa laitteissa

* ot udp open
* ot ipaddr

Viestin vastaanottajassa kuuntele porttia 1234 ajamalla komento

ot udp bind :: 1234

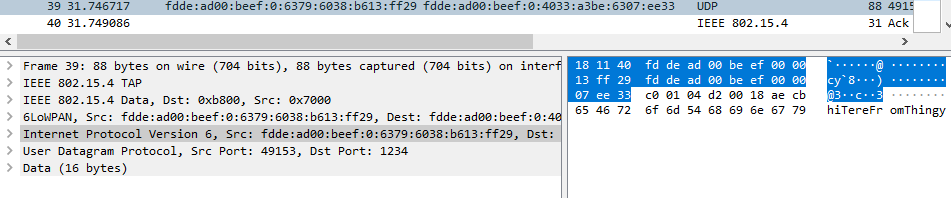
Nyt lähettäjässä voi tehdä viestin, esim

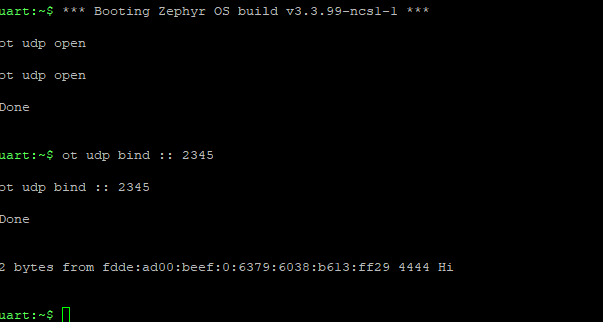
ot udp send fdde:ad00:beef:0:c355:ba21:b1d:4949 1234 Hello

Jossa punainen osa on kohteen ip osoite, 1234 vastaanottava portti ja Hello viesti, joka lähetetään. Jos haluaa lähettää useampia viestejä, voi asettaa automaattisesti vastaanottajan yhdistämällä osoitteeseen ja porttiin. Jonka jälkeen vastaanottajaa ei tarvitse enää määrittää.

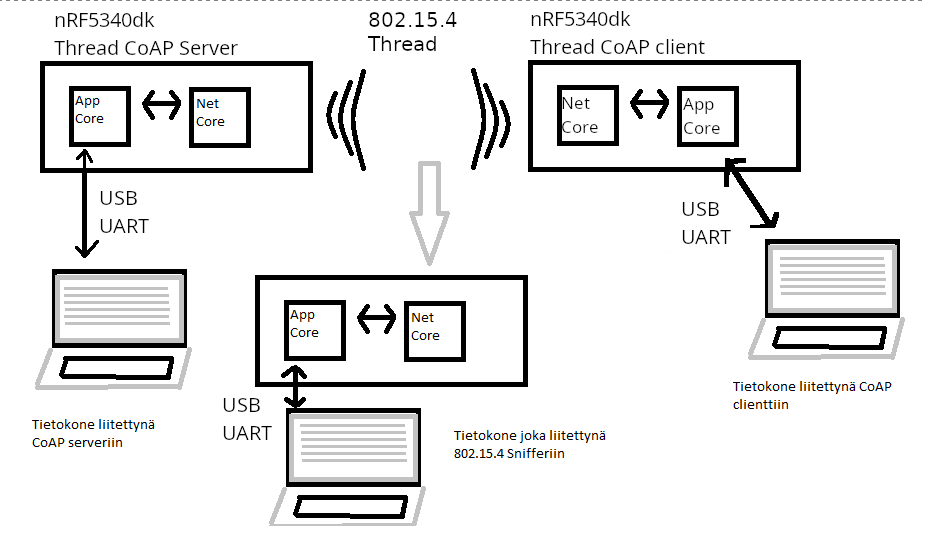
ot udp connect fdde:ad00:beef:0:c355:ba21:b1d:4949 1234

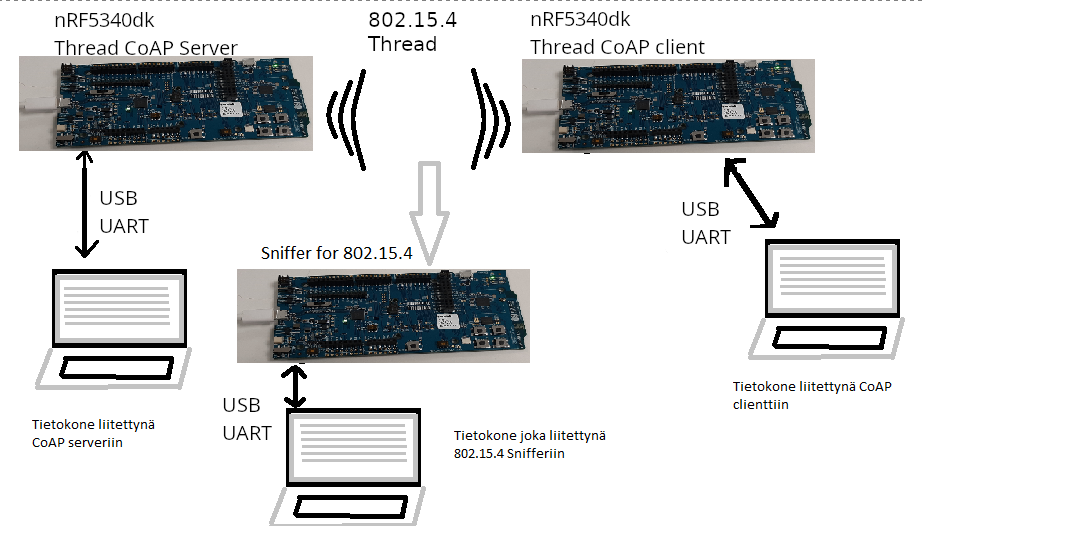
ot udp send GreetingsFromOverHere

Viestin voi lukea wiresharkissa, kun asetukset ovat oikein ja salausavain tunnetaan. Jotta se olisi selkokielinen pitää lukea 6LoWPAN tai alempi rivi. Kuva33 *UDP viesti Wiresharkissa kaapattuna*



Kuva34 *Viesti vastaanottajan näkökulmasta*



Kuva35 *Kaavio laitteiden välisestä viestinnästä*

*Kuva36 Protokolla pino*