LoRa ESP32’s en Intel NUC





Studenten: Thomas van Egmond

Cursus: TINPRJ0478

Docenten: W.B. Volders

S.M. Hekkelman

Context

In dit document gaan we een blik werpen op de volgende onderdelen.

* LoRa ESP32
* Intel NUC

We gaan kijken waarom en waarvoor we deze onderdelen gebruikt hebben.

LoRa ESP32

Eén van de vereiste van de RDW voor de open category in om een remote kill switch te hebben voor het voertuig in geval van nood. We moesten dus opzoek naar iets wat een signaal kon vervoeren over een afstand van ongeveer 300 meter.



Als beginpunt hebben we deze afbeelding gebruikt een van onze andere cursussen. Zoals je kan zien is bereik on belangrijkste vereiste en aan het uitende van deze as staan mobiele data en LoRa. Mobiele data kan ook gebruikt worden voor deze doeleinde alleen zit je dan dus aan de limitaties van het bestaande mobiele netwerk zoals bereik. Mobiele data kost ook meer om te implementeren, de ESP’s die dit ondersteunen kosten meer en je zult een iets van een abonnement moeten afsluiten.

De volgende methode die we bekeken was LoRa en dit voldeed aan onze eisen. Goedkoop, makkelijk te implementeren, een groot bereik van door ons getest minimaal 300 meter zonder data corruptie. Met LoRa is het mogelijk om je peer-to-peer eigen netwerk op te zetten, hierdoor zijn we ook niet afhankelijk van bijvoorbeeld locatie van zendmasten zoals bij mobiele data wel het geval is.

Ultra-wideband is ook door ons onderzocht maar bleek te kort te komen in het gebied van bereik.

Nu het wat meer technische gedeelte. LoRa zit gelimiteerd aan 256 bytes per packet dat je verstuurd, wat dus niet veel is. Daarnaast heeft LoRa zelf ook header packets wat realistisch het aantal bytes dat je kan versturen verlaagd naar ongeveer 200. LoRa doet er ook langer over om meer bytes te sturen, 150 bytes geeft bijvoorbeeld een rate van data transfer per seconden terwijl 2 bytes onder de rond de 100 transfers per seconden zitten. Voor ons is een lage zendtijd heel belangrijk voor de veiligheid aangezien de kill switch zijn signaal zal sturen over LoRa. Een kill switch waarde is gelukkig maar 1 byte (of zelfs bit) om te versturen dus voor ons is LoRa snel genoeg.

Een andere RDW vereiste was om het voertuig handmatig te kunnen besturen. Wij hebben besloten dit te doen door een DualShock 4 controller aan de ESP32 op het voortuig te koppelen, dit werkte perfect). Toen leek het ons leuk om de controller te verbinden aan de ESP die niet aan het voertuig vastzit maar wij gebruiken voor de remote kil switch. Dit was zeker mogelijk, alleen het versturen van controller data is een stuk meer dan 1 of 2 bytes als je doet op de gebruikelijke manier zoals JSON sturen. In plaats daarvan kunnen we elke boolean knop (bijvoorbeeld het rondje, ingedrukt of niet) uitdrukken als één bit van een byte, zo kunnen we 8 knoppen per byte versturen. Voor elke analoge trigger of stick is een volledige byte nodig, dit zijn namelijk 0-255 waardes. Dit zou samen neerkomen op 16 boolean knoppen wat 2 bytes zijn samen met 6 bytes voor de triggers en sticks, een totaal van 8 extra bytes! Een heel groot stuk minder dan een JSON sturen (90 bytes). Wel is van belang dat beide kanten van het LoRa netwerk exact weten welke bytes ze kunnen verwachten en wat deze bytes dan betekenen. Deze extra 8 bytes hadden een technisch meetbaar effect maar voor het menselijk ook was een geen onderscheid te makken in snelheid en werkte dit in realtime!

Intel NUC

Het door ons getrainde objectherkenningsmodel draaiende op YoloV8 (zie ‘YoloV8 en easyOCR’ in het mapje ‘Modules) zowel als onze voertuig logica (zie ‘main.py’ in het mapje ‘Code’) moet draaien op iets van een computertje wat op het voertuig gemonteerd kan worden.

Voor het selecteren van zo een computer hebben wij rekening gehouden met drie vereiste:

1. Het is compatibel met onze apparatuur.
2. Het is krachtig genoeg om onze programma’s te draaien.
3. Het valt binnen ons budget

Het selectieproces verliep vrij vlot, in eerste instantie kijken wij naar wat de closed category aangeleverd had gekregen, dit bleek een Intel NUC te zijn. Wij hadden toevallig op school nog een Intel NUC liggen die tot onze beschikking gesteld kon worden. Voor vereiste 3 is dit natuurlijk de beste uitkomst, nu de andere twee nog. Vereiste 1 bleek ook geen probleem te zijn, onze apparatuur werkte vlekkeloos met de NUC.

Vereiste 2 ging iets meer werk in zitten. Het verkleinen van onze dataset waar ons model op getraind was bleek genoeg te zijn om het programma soepel te laten draaien op de NUC. Dit komt omdat de NUC geen grafische processor bevat, machine learning algorithms zoals YoloV8 draaien namelijk slecht op een normale processor en profiteren heel erg van de veel grotere aantallen cores die in grafische processoren zitten.

Dus, een computer kopen met een grafische processor bijvoorbeeld een Nvidia Jetson. Helaas kijk je dan al snel naar 300€ +, iets wat wellicht had gekund maar als het niet anders kon liever niet. Gelukkig hebben we door gebruik te maken van een software genaamd OpenVINO ons trainde model kunnen optimaliseren voor de Intel processor in de NUC. Dit maakte het verschil dat we nodig hadden om ons programma soepel te laten draaien! (Zie ‘Onderzoeken/The impact of OpenViNO on performance for CPU based Edge AI devices.’)

Intel NUC & LoRa ESP32

Nu kunnen we de rijinstructies die ons programma berekent doorsturen naar het voertuig. Hiervoor gebruiken we een seriële connectie tussen de NUC en de LoRa ESP32. De standaard manier van seriële data uitlezen op een ESP32 bleek niet goed te snel genoeg te werken. Deze manier is ‘blocking’ wat betekent dat de ESP32 niks anders kan doen dan wachten en uitlezen van seriële data. Dit is overduidelijk niet het idee. Daarnaast loopt het ophalen van data op de ESP32 en het sturen van data op de NUC niet synchroon wat lijdt tot niet correcte of niet volledig uitgelezen data. De oplossing hiervoor is het de data byte voor byte uitlezen en gebruik maken van een start en eind karakter wat het begin en einde van een data transfer aangeeft (zie ‘Code/Production/ESP32/ESP32.ino’, meer weten? Kijk op [Gammon Forum : Electronics : Microprocessors : How to process incoming serial data without blocking](https://www.gammon.com.au/serial).