PROJEKT SDR

Praktikum komunikacijskih sustava

Viktor Horvat

17. prosinac 2024.

Zagreb

Zadatak

Zadatak je koristeći se ADALM-PLUTO modulom razviti softversku podršku za primanje i obradu QPSK moduliranog signala na osnovnoj frekvenciji pojasa od 2.2 GHz. Uz poznatu modulacijsku shemu primljenih podataka, potrebno je fotografiju skrivenu u primljenim podacima demodulirati i prikazati u računalno prihvatljivom obliku.

Eksperimentalni primljeni podaci dani su u obliku CSV (Comma Separated Values) datoteke i kompleksnog zapisa. Poznata je frekvencija osnovnog pojasa, širina pojasa, modulacijska shema, preambula koja označava početak i preambula koja označava kraj korisnih podataka tj. bitove koji služe za rekonstrukciju slike.

U zadataku je potrebno provesti sljedeće korake:

1. primjenu rrcos filtera
2. grubu frekvencijsku sinkronizaciju
3. vremensku sinkronizaciju simbola
4. finu frekvencijsku (faznu) sinkronizaciju
5. demodulaciju simbola u bitove i pretvorbu slike

Za obradu informacija preporuča se koristiti Python skriptnim jezikom.

Rješenje

Python biblioteke

Korišteno je nekoliko dodatnih biblioteka za obradu signala, vizualizaciju i rad sa slikama:

1. **numpy**: za rad s vektorima i matricama
2. **pandas**: za učitavanje podataka u CSV formatu
3. **scikit-commpy**: za filtriranje i demodulaciju
4. **matplotlib**: za crtanje grafova I/Q podataka
5. **Pillow**: za manipulaciju i prikaz slike

Imena navedenih biblioteka spremljene su u requirements.txt datoteku kako bi se olakšala instalacija svih predzahtjeva za reprodukciju rješenja prikazanog u ovom dokumentu. Koristeći pip sustav za upravljanje paketima potrebno je izvršiti sljedeću naredbu u terminalu kako bi se sve biblioteke uspješno instalirale u virtualno Python okruženje:

pip install -r requirements.txt

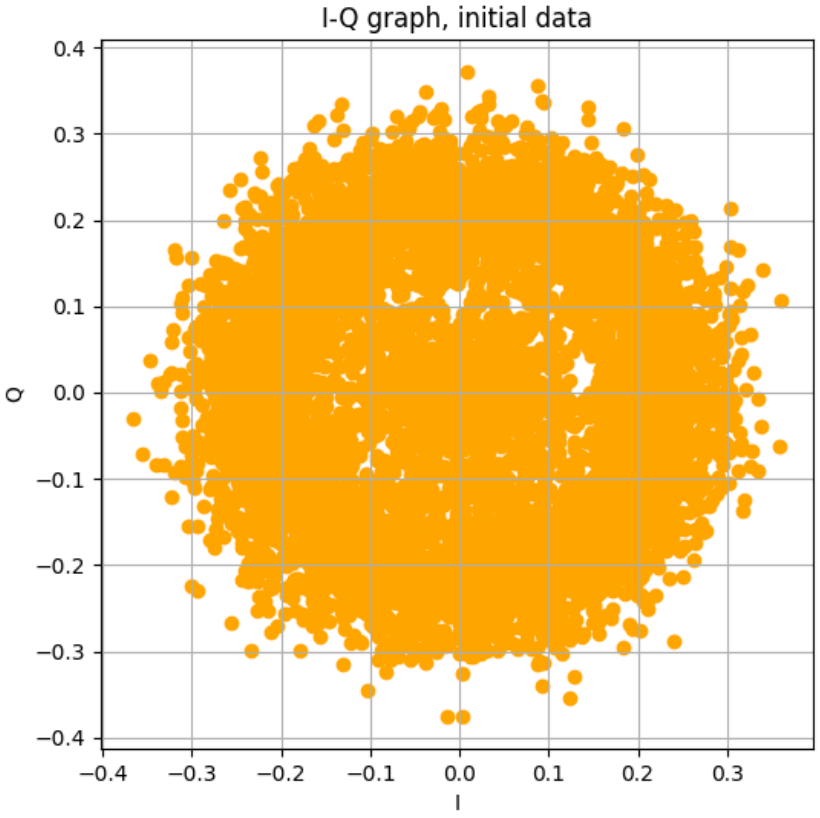
Dodatni kod neće biti dio ovog izvještaja već je u izvornom formatu priložen izvještaju.

Korak 0: Učitavanje podataka

Inicijalni korak u obradi signala jest učitavanje podataka iz CSV datoteke, gdje su podaci spremljeni kao niz kompleksnih brojeva. Ovi kompleksni brojevi predstavljaju I (*in-phase*) i Q (*quadrature*) komponente, koje zajedno čine osnovu za obradu faznih modulacija kao što je QPSK. Kompleksni signal omogućuje prijenos i obradu informacija pomoću promjena amplitude i faze, što je ključna karakteristika modernih digitalnih komunikacijskih sustava. Na ovaj način možemo istovremeno prenositi više informacija koristeći ograničeni frekvencijski spektar, što povećava efikasnost sustava.

Vizualizacija I-Q signala (I-Q graf) omogućuje bolje razumijevanje oblika signala i uočavanje mogućih anomalija u prijenosu. Učitani podaci su temelj za sve daljnje korake obrade, uključujući filtriranje, sinkronizaciju i demodulaciju. Na početnom prikazu

podataka, reduciranom na svaku 25. vrijednost iz početnog seta podataka, nazire se kružnica koja opisuje nesinkroniziranu QPSK modulaciju.



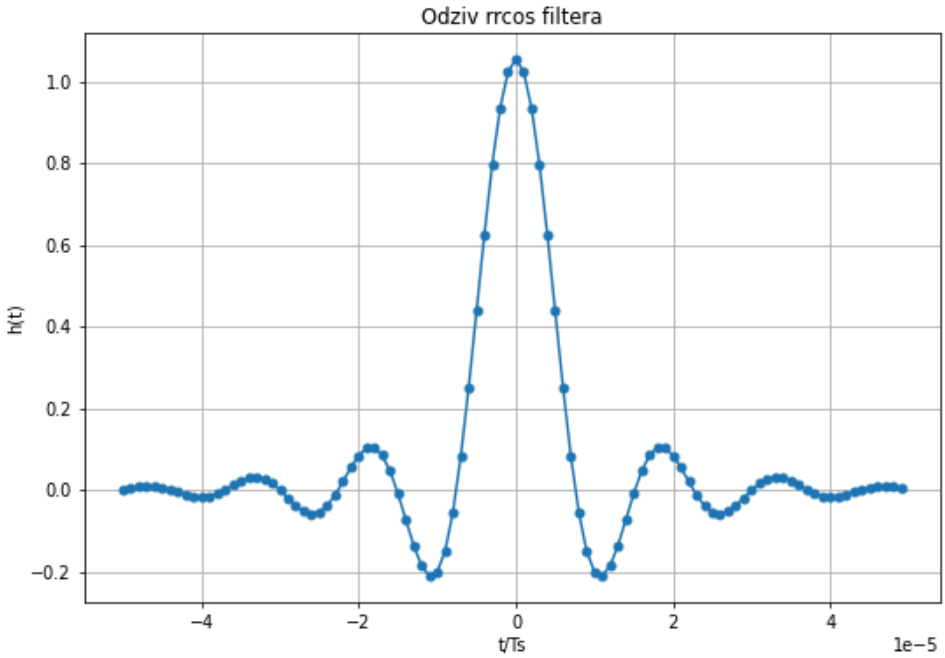
Korak 1: Filtriranje pomoću RRCOS filtera

U ovom koraku primijenili smo *Raised Cosine* filtar s ograničenim impulsnim odzivom (RRCOS) kako bismo smanjili intersimbolnu interferenciju (ISI). ISI nastaje zbog preklapanja susjednih simbola, što može otežati njihovo pravilno dekodiranje i povećati pogreške u komunikacijskom sustavu. RRCOS filtar je posebno dizajniran za minimiziranje ISI-a jer oblikuje impulse na način koji ograničava njihovo širenje izvan predviđenog vremenskog intervala.

Ovaj filtar se često koristi u digitalnim komunikacijama zbog svojih optimalnih svojstava kada se koriste višesimbolički sustavi. Konvolucija primljenog signala s impulsnim odzivom filtra poboljšava oblik signala, uklanjajući neželjene smetnje i omogućujući jasniju detekciju simbola. Filtriranje je neophodan korak jer pomaže u pripremi signala za

demodulaciju i kasniju obradu, osiguravajući da će prijemnik točno rekonstruirati originalne podatke.

Korak 1.1: RRCOS filter



RRC filter karakteriziran je s dvije ključne vrijednosti: β, faktorom roll-off-a, i Ts tj. recipročnom vrijednošću simbolne brzine:

1. β (roll-off faktor) definira širenje spektra izvan osnovnog pojasa signala. Vrijednosti β variraju od 0 do 1, gdje niže vrijednosti (npr. β = 0) predstavljaju idealan filtar bez ikakvog širenja spektra, dok veće vrijednosti dopuštaju veće širenje, smanjujući međusimbolnu interferenciju, ali zauzimajući širi frekvencijski spektar. U praksi se često koristi kompromisna vrijednost β, koja omogućuje ravnotežu između spektralne učinkovitosti i tolerancije na šum.

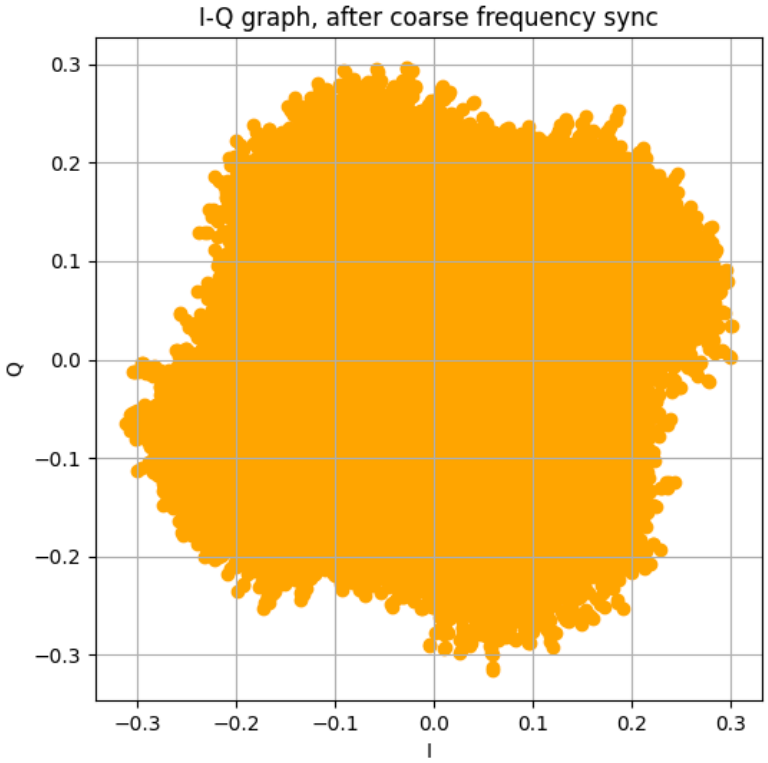
2. Ts (recipročna vrijednost simbolne brzine) odnosi se na vremenski interval između dva uzastopna simbola. Simbolna brzina (također poznata kao Baudova brzina) određuje koliko simbola se šalje po jedinici vremena, a Ts je njen recipročni ekvivalent, odnosno vrijeme potrebno za prijenos jednog simbola. Ova vrijednost izravno utječe na širenje signala i oblikovanje impulsa kroz RRC filter. Veći interval Ts (tj. manja simbolna brzina) omogućuje veće razdvajanje između simbola, što može smanjiti međusobne smetnje, ali također može smanjiti

ukupnu brzinu prijenosa podataka.U našem slučaju korištene su vrijednosti od 0.2 za roll-off faktor te 8 mikrosekundi za Ts.

Korak 2: Gruba frekvencijska sinkronizacija

Gruba frekvencijska sinkronizacija izuzetno je važna jer osigurava da frekvencija lokalnog oscilatora prijemnika bude usklađena s frekvencijom prijenosnog signala. U ovom koraku signal podižemo na četvrtu potenciju, čime uklanjamo fazne rotacije specifične za QPSK modulaciju (koja koristi četiri različita fazna stanja).

Ova tehnika omogućuje nam da poništimo faznu komponentu modulacije i izdvojimo dominantnu frekvenciju. Nakon toga primjenjujemo Fourierovu transformaciju kako bismo detektirali frekvencijski pomak uzrokovan nesavršenostima u oscilatoru prijemnika ili zbog Dopplerovog efekta. Detektirana frekvencija se koristi za podešavanje signala tako što ga množenjem s eksponencijalnim faktorom prebacujemo na ispravnu frekvenciju. Ovaj korak uklanja veći dio frekvencijskih anomalija i postavlja signal na osnovnu frekvenciju koja je potrebna za precizniju demodulaciju u kasnijim fazama obrade.

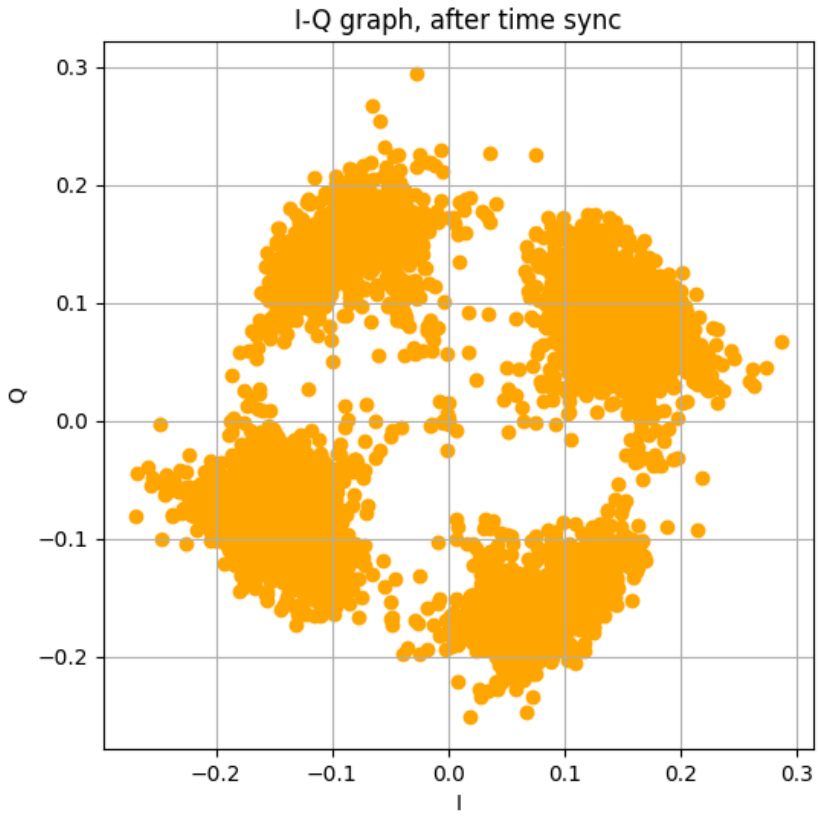


Nakon grube frekvencijske sinkronizacije nazire se poznati nam konstelacijski dijagram korištene QPSK fazne modulacije, no zbog još uvijek prisutne desinkronizacije on je zakrenut.

Korak 3: Vremenska sinkronizacija simbola

Nakon što smo stabilizirali frekvenciju signala, sljedeći korak je vremenska sinkronizacija, koja osigurava da simboli budu uzorkovani u točno odgovarajućem trenutku. Ovdje koristimo Mueller-Müller (MM) algoritam, koji se temelji na analizi razlika između uzoraka simbola kako bi se odredio optimalan trenutak uzorkovanja. Algoritam detektira promjene u stvarnim i imaginarnim dijelovima uzorka te izračunava korekcijski signal koji nas vodi prema točnom vremenu uzorkovanja.

Bez ove sinkronizacije, simboli bi mogli biti uzorkovani prerano ili prekasno, što bi dovelo do pogrešnog dekodiranja podataka. MM algoritam prilagođava fazu i vrijeme uzorkovanja kroz nekoliko iteracija kako bi minimalizirao pogrešku između stvarnog i idealnog trenutka uzorkovanja. Nakon vremenske sinkronizacije konstelacijski dijagram prikazuje simbole grupirane oko njihovih očekivanih pozicija, no cijeli dijagram je još uvijek zakrenut.



Korak 3.1: Mueller-Müller algoritam

Mueller-Müller algoritam (MM) koristi povratnu vezu kako bi sinkronizirao vrijeme uzorkovanja signala, osiguravajući da simboli budu detektirani na točnom vremenskom mjestu. Ovaj algoritam uspoređuje trenutne i prethodne uzorke signala kako bi odredio smjer i veličinu pogreške u vremenu uzorkovanja. MM algoritam ne zahtijeva pilotske simbole, što ga čini izuzetno efikasnim u stvarnim prijenosnim sustavima.

Korištenjem iterativne povratne veze, algoritam kontinuirano prilagođava trenutak uzorkovanja kako bi minimizirao razliku između očekivanih i stvarnih pozicija simbola. MM algoritam također koristi karakteristike fazne modulacije kako bi precizno odredio trenutak kada simbol prelazi preko idealne točke uzorkovanja, što omogućava preciznu detekciju simbola. Referiramo li se na korak 3 u izvornom kodu i tamo pristne varijable u petlji:

```
while i_out < len(samples) and i_in+16 < len(samples):
        out[i_out] = samples[i_in]
        out_rail[i_out] = int(np.real(out[i_out]) > 0) + 1j
*int(np.imag(out[i_out]) > 0)
        x = (out_rail[i_out] - out_rail[i_out-2]) * np.conj(out[i_out-1])
        y = (out[i_out] - out[i_out-2]) * np.conj(out_rail[i_out-1])
        mm_val = np.real(y - x)
        mu += samples_per_symbol + 0.7*mm_val
        i_in += int(np.floor(mu))
        mu = mu - np.floor(mu)
        i_out += 1
```

mu: predstavlja fazni pomak između trenutnog trenutka uzorkovanja i idealnog trenutka uzorkovanja. Inicijalno je postavljen na 0, ali se dinamički ažurira tijekom izvršavanja algoritma.

out: niz koji pohranjuje uzorke signala na temelju trenutnog vremena uzorkovanja.

out_rail: sadrži aproksimacije trenutnih uzoraka signala, pojednostavljene na binarne vrijednosti, tj. prema znaku realnog i imaginarnog dijela.

mm_val: ključna vrijednost koju izračunava Mueller-Müller algoritam, i koja se koristi za prilagodbu faze (vrijednosti mu) tijekom iteracija

.

U svakoj iteraciji, algoritam uzima trenutni uzorak iz signala *(out[i_out])*, kao i prethodne uzorke. Razlika između uzorka i aproksimacije *(out_rail)* koristi se za izračunavanje Mueller-Müller vrijednosti *(mm_val)*, što je mjera vremenske pogreške. Na temelju te pogreške, algoritam zatim prilagođava fazu pomoću izraza:

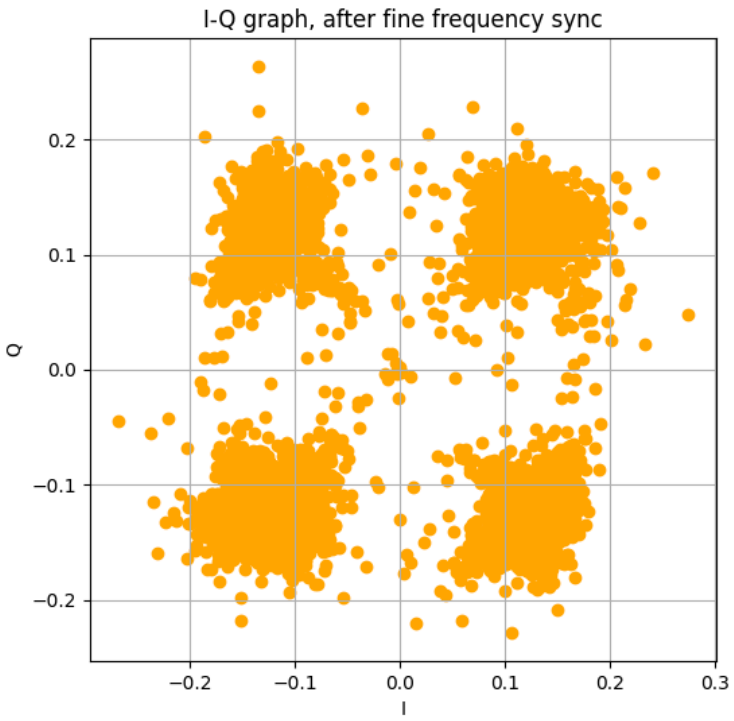
$$mu \mathrel{+}= samples\_per\_symbol + 0.7 * mm\_val$$

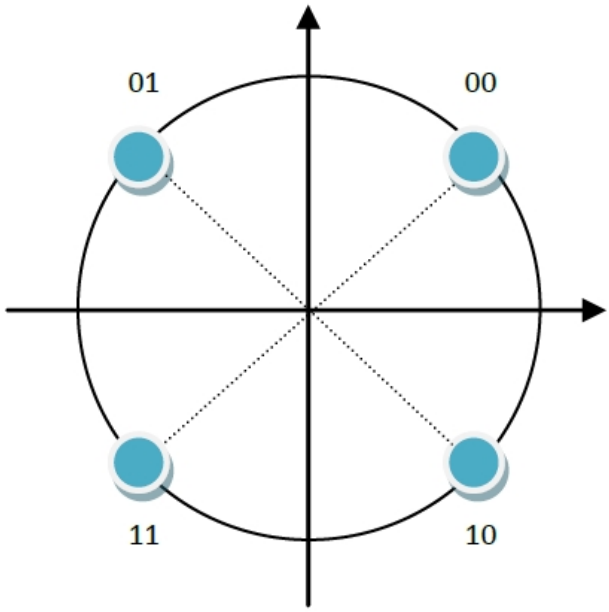
Ovdje je *samples_per_symbol* prosječna udaljenost između dva uzorka, a mm_val predstavlja korekciju na temelju trenutne procjene pogreške. Koeficijent 0.7 određuje koliko brzo se prilagođava trenutak uzorkovanja na temelju pogreške.

Korak 4: Fina frekvencijska sinkronizacija

Nakon što smo izveli grubu frekvencijsku sinkronizaciju, preostali frekvencijski pomaci se korigiraju u ovoj fazi fine sinkronizacije. Fina sinkronizacija koristi algoritam za detekciju faze (Phase Detector) kako bi iterativno ispravila preostale frekvencijske nepravilnosti. U ovom postupku koristi se informacija o trenutnoj fazi i frekvenciji signala kako bi se osiguralo da nema preostalih frekvencijskih devijacija koje bi mogle utjecati na točnost demodulacije. Korekcija se primjenjuje kroz niz malih prilagodbi frekvencije, čime se smanjuju preostale fazne i frekvencijske pogreške. Fina frekvencijska sinkronizacija ključna je za uklanjanje malih, ali značajnih devijacija koje su ostale nakon grubog podešavanja frekvencije.

To omogućuje precizniju demodulaciju i smanjuje šum u konačnom signalu, što osigurava bolju točnost u prijenosu podataka. Nakon fine frekvencijske i vremenske sinkronizacije, prikazujući svaki peti uzorak, jasno vidimo sinkronizirane podatke koji se nalaze na pretpostavljenim mjestima kod uporabe QPSK modulacije. Obzirom da se radi o realnom sustavu, šum u kanalu i manje greške su očekivane, no dobiveni konstelacijski dijagram trebao bi omogućiti donošenje ispravne odluke o simbolu kod provedbe demodulacije.





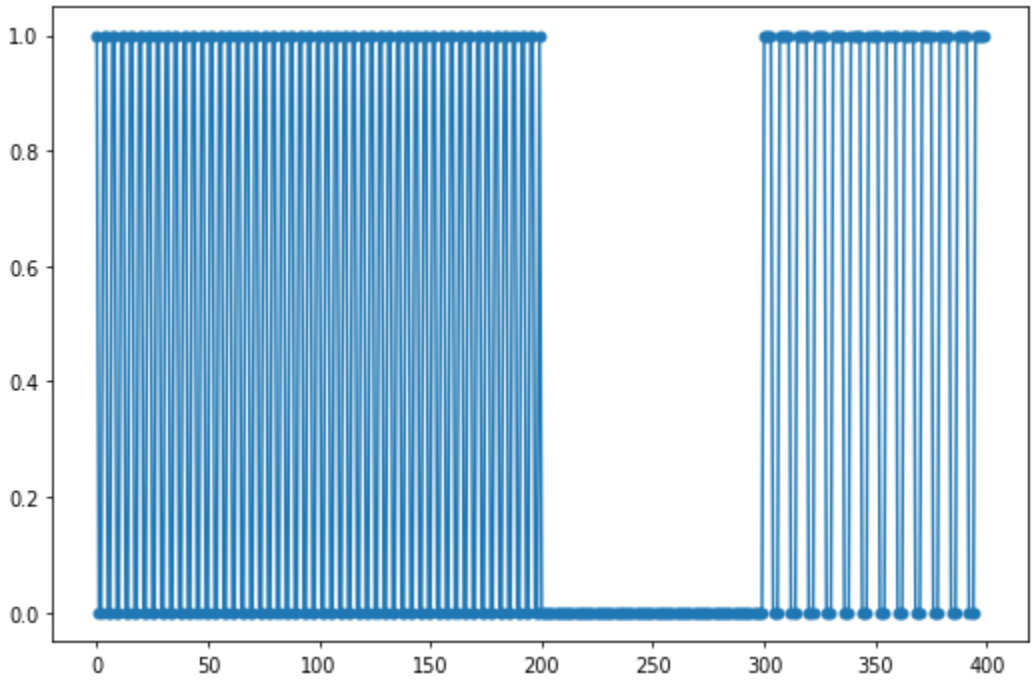
Korak 5: Demodulacija podataka i ekstrakcija slike

Završni korak obrade uključuje demodulaciju primljenog QPSK signala kako bi se pretvorio natrag u niz binarnih podataka. Koristimo hard-decision demodulaciju, pri čemu svaki primljeni simbol biva mapiran na odgovarajući binarni kod. Demodulacijom vraćamo modulirane simbole u niz bitova, što nam omogućuje daljnju obradu i rekonstrukciju originalnih podataka.

Kako bismo pronašli početak slike u nizu primljenih bitova, koristimo poznatu sekvencu bitova (preambulu) za detekciju početne točke slike. Preambula omogućuje precizno lociranje relevantnih podataka unutar primljenog signala, čime osiguravamo da se slika može pravilno rekonstruirati. Nakon što je preambula detektirana, niz bitova koji slijedi interpretira se kao slikovni podaci, a zatim se pretvaraju u piksele pomoću funkcije za konverziju bitova u sliku. Konačno, rekonstruirana slika se prikazuje, čime završava proces prijema i dekodiranja signala.

Prikaz preambule na početku podatkovnog slijeda, i na kraju podatkovnog slijed dan je u nastavku, a sastoji se od:

1. dio niz od dva simbola [10,01] ponavlja se 50 puta
2. dio jedan simbol [00] ponavlja se 50 puta



Dobivena slika nakon provedenih svih prije spomenutih postupaka:

Zaključak

Uspješno su implementirani postupci primjene rrcos filtera, grube frekvencijske sinkronizacije, vremenske sinkronizacije simbola, fine frekvencijske (fazne) sinkronizacije te demodulacije simbola u bitove i pretvorbe primljenih demoduliranih podataka u sliku.

Izvori

M. Lichtman, "Carrier and Timing Recovery (Synchronization)," *PySDR: A Guide to SDR and DSP using Python*, [Online]. Available: https://pysdr.org/content/sync.html. [Pristupljeno: Dec. 17, 2024].