Orientarsi nel progetto

Premessa: il software è stato sviluppato con visual studio code con le estensioni di microsoft per eseguire/debuggare codice c/c++.

Attenzione:

- nel codice ho lasciato delle ambiguità che toglierò: dire RTK è uguale a dire GPS e dire PPP è uguale a dire GALILEO (anche se effettivamente non è così, ma nel codice tenete conto che ho commesso qualche errore del genere)

I file “cuore” del software:

- **semor.c** - Starting point of SEMOR. It executes RTKLIB instances from which we take inputs, reads the configuration file and call the function that starts the main loop.

- **src/semor.h** - Header file for global variables

- **src/client.c** - In here we create sockets (to read data from RTKLIB), we find out the output of SEMOR (best position, so the mean of the 3 positions RTK, PPP, IMU)

- **loose-gnss-imu/imu_read.c** - It reads IMU data (data from accelerometer and gyroscope). This is utilized by the file *Loosely.cpp* (see next line).

- **loose-gnss-imu/Loosely.cpp** - Code containing IMU mechanization (calculation of the IMU position)

- **src/ctocpp.cpp** - Bridge between c (pure SEMOR code) and c++ code (imu mechanization library code)

Il resto dei file si può anche non guardare.

Descrizione delle funzioni

semor.c:

- close_io(): chiude input e output dei processi rtklib eseguiti con exec (per non interferire con i/o principale di semor.
- read_conf_file(): legge una linea e modifica le variabili di SEMOR secondo quanto letto.
- main()

src/client.c:

- close_semor(): terminazione SEMOR “morbida”
- str2gnss(): String to gnss_sol_t structure
- gnss2str(): gnss_sol_t structure to string
- gnsscopy(): Copia di una struttura gnss_sol_t in un’altra medesima struttura

- output(): Stampa l'output di SEMOR nel file puntato dalla variabile "file" (default: output.txt)
- print_solution(): utilizzato per scrivere log di posizioni individuali RTK, PPP, IMU
- similar_pos(): Controlla che due posizioni siano simili (ovvero sufficientemente vicine)
- gnss_avg_2(): Imposta la variabile "best" come la media tra due posizioni
- gnss_avg_3(): Imposta la variabile "best" come la media tra tre posizioni
- get_best_sol2(): controlla che due posizioni siano simili (similar_pos()) e se così fosse chiama gnss_avg_2() e ritorna 1, altrimenti ritorna 0.
- get_best_sol_3(): controlla che tutte e tre le posizioni siano simili, se così fosse chiama gnss_avg_3() e ritorna 1, altrimenti controlla se ce ne sono due simili e se così fosse chiama gnss_avg_2 con quelle e ritorna 1, altrimenti ritorna 0.
- process_solutions(): prende le tre posizioni (attenzione, non tutte potrebbero esserci in un certo momento a causa di "buchi" nelle registrazioni, per esempio al PPP succede qualche volta). Con queste posizioni, chiama le funzioni precedenti per trovare l'output. Alla fine fornisce la posizione "best" trovata alla parte di codice responsabile alla IMU mechanization per trovare la prossima posizione IMU.
- check_termination(): se l'utente invia 'q' o 'Q', SEMOR termina
- setup_tcp_socket(): inizializza un socket TCP non bloccante. Prima di ritornare aspetta che il socket sia effettivamente connesso.
- read_gnss(): legge input gnss da socket (o file se in modalità debug)
- handle_connection(): inizializza i socket (chiamando setup_tcp_socket()), la struttura per poll() ed avvia il main loop nel quale:
    - legge l'input gnss (chiamando read_gnss())
    - chiama process_solutions()
- start_processing(): apre i file necessari (per log e output) e poi chiama handle_connection().

loose-gnss-imu/Loosely.cpp:
- init_imu(): inizializza qualche variabile necessaria per l'inizializzazione dell'imu
- get_imu_sol():
    - se variabili imu non ancora inizializzate: esegue un altro step di inizializzazione
    - se variabili imu sono inizializzate: inizializza l'oggetto necessario per l'imu mechanization, chiama SolutionIMU() che esegue l'imu mechanization (tramite MechECEF.MechanizerECEF()) e chiama calculateSTD() che calcola la deviazione standard della posizione IMU appena trovata con l'imu mechanization.
- read_imu(): legge l'input grezzo dell'imu chiamando get_imu_data del file imu_read.c
- il resto delle funzioni contiene algebra lineare che calcola la deviazione standard dell'imu

src/ctocpp.cpp:
- imu_sol(): consente l'accesso del metodo get_imu_sol() di Loosely.cpp a codice c
- init_imu(): consente l'accesso del metodo init_imu() di Loosely.cpp a codice c
- close_ctocpp: utilizzato in close_semor() per terminare SEMOR in modo "morbido"