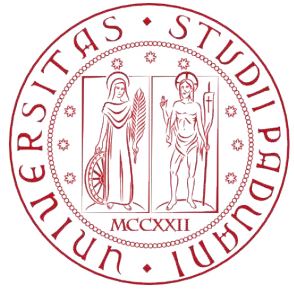
Sistema Multimodale di Sensoristica con scambio dati in formato Vrs

Marco Greco, Marco Scarabello e Cristian Tramarin

20 Marzo 2023



Università degli Studi di Padova

Struttura moduli

Introduzione

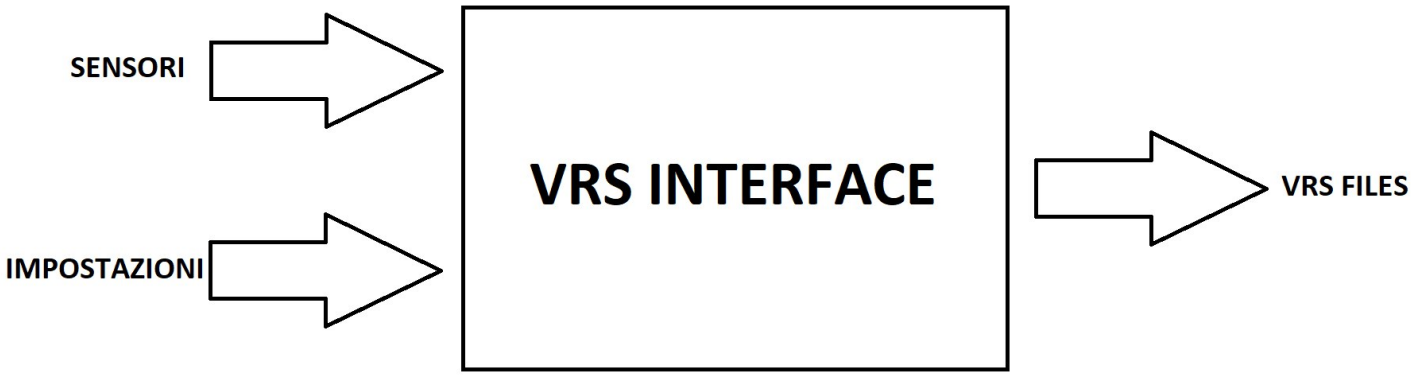
Per lo sviluppo del dispositivo Aria, gli sviluppatori di Meta hanno progettato un nuovo sistema di formattazione, al fine di gestirne la sensoristica, chiamato VRS. Esso è in grado di processare in poco tempo anche quantità considerevoli di dati dai sensori, oltre a permettere di includere in un'unico file le registrazioni di tutti i dispositivi attivi, previa una configurazione iniziale, il cui stato viene poi anch'esso salvato nel file vrs. Per queste sue proprietà, abbiamo deciso di usare la libreria VRS per la progettazione di un'applicazione per dispositivi Android in grado, attraverso una connessione TCP IP, di inviare i dati di sensoristica, quali microfono, telecamera e giroscopio, ad un server.

Introduzione

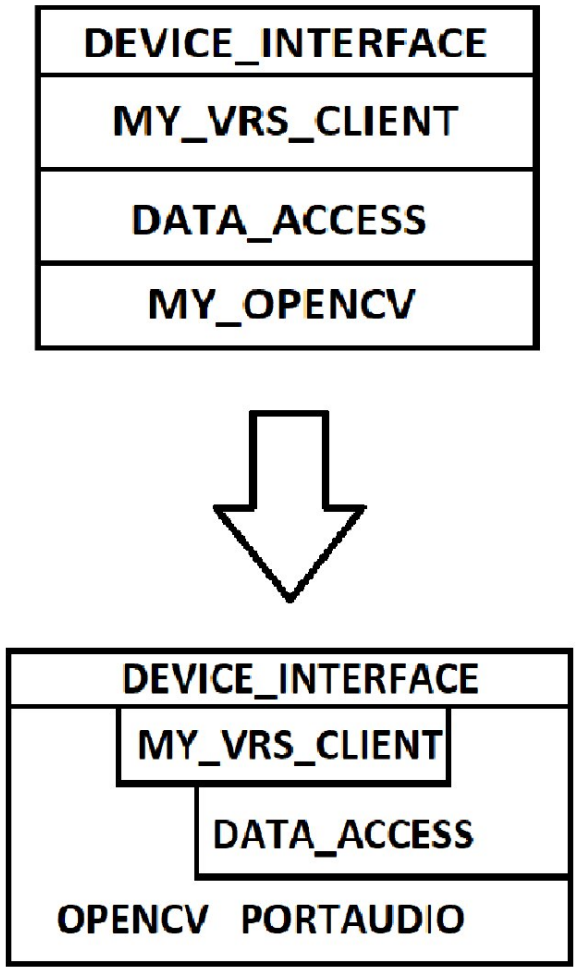
Il progetto quindi verte sullo sviluppo di un'applicazione acquisizione di dati da un sistema multimodale di sensoristica, tramite l'utilizzo di una formattazione file vrs e una connessione Client Server. Abbiamo utilizzato:

- Software Unity
- Codice C++
- Codice C#
- Vrs

Softcoded vrs interface

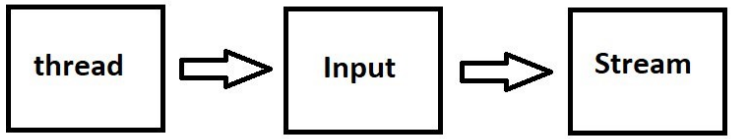


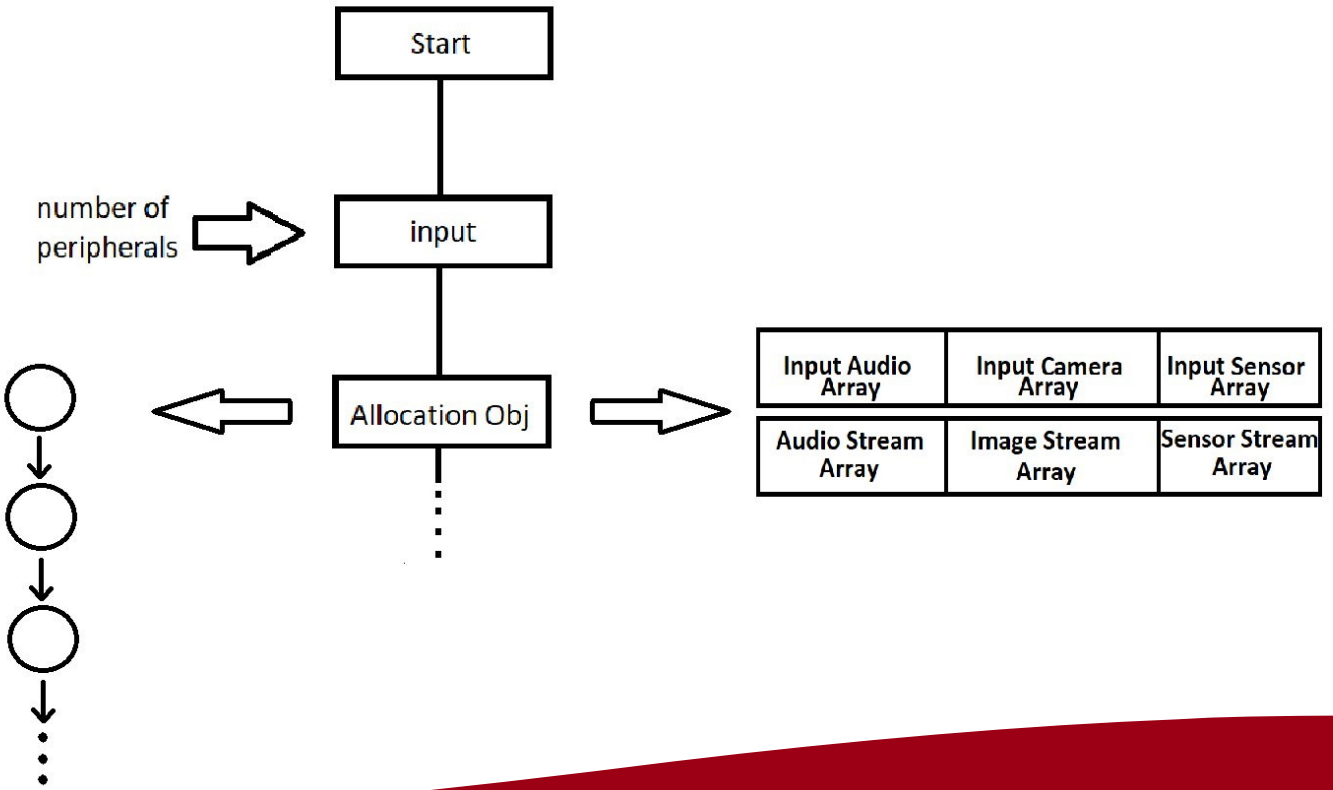
Softcoded vrs interface



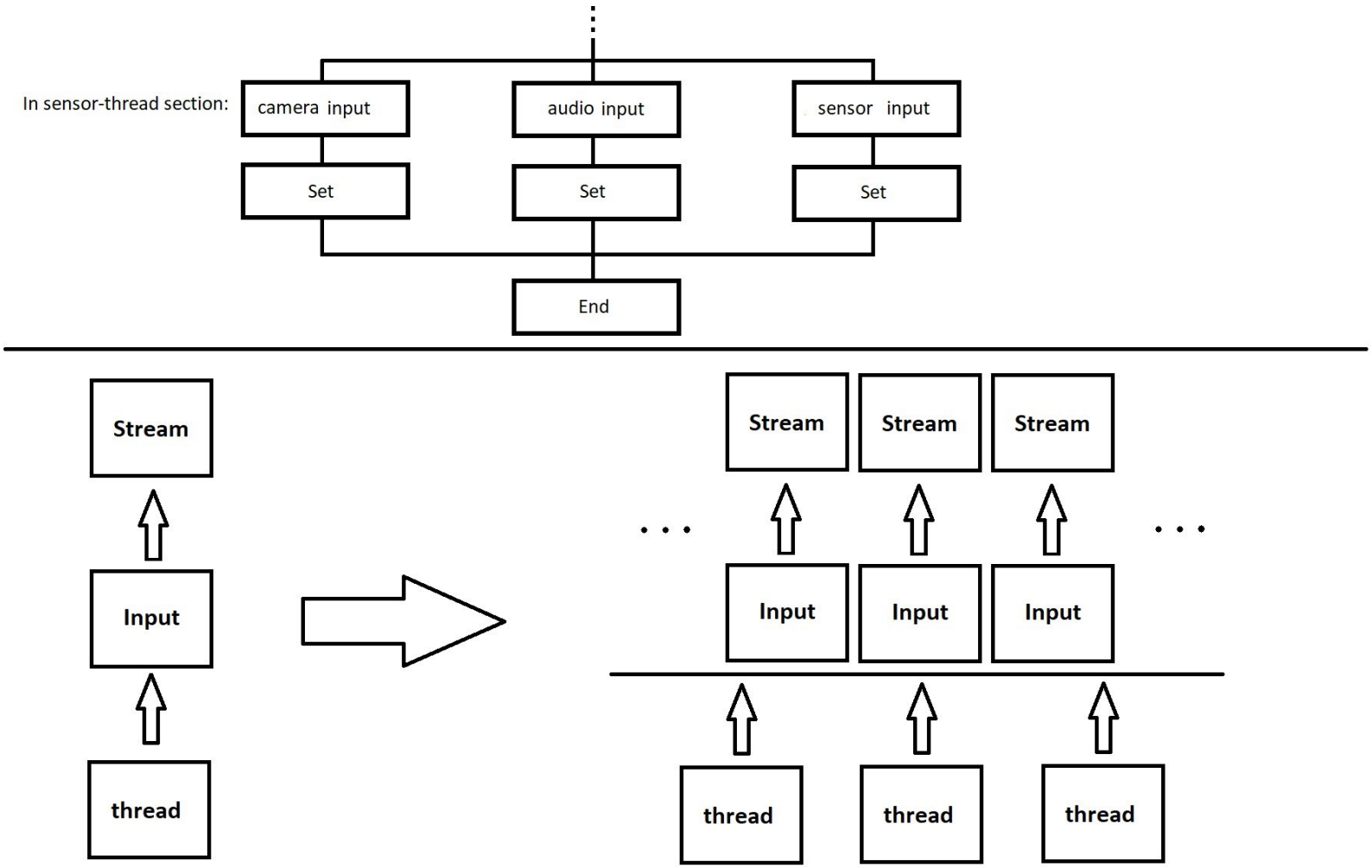
Softcoded vrs interface

DEVICE_INTERFACE:

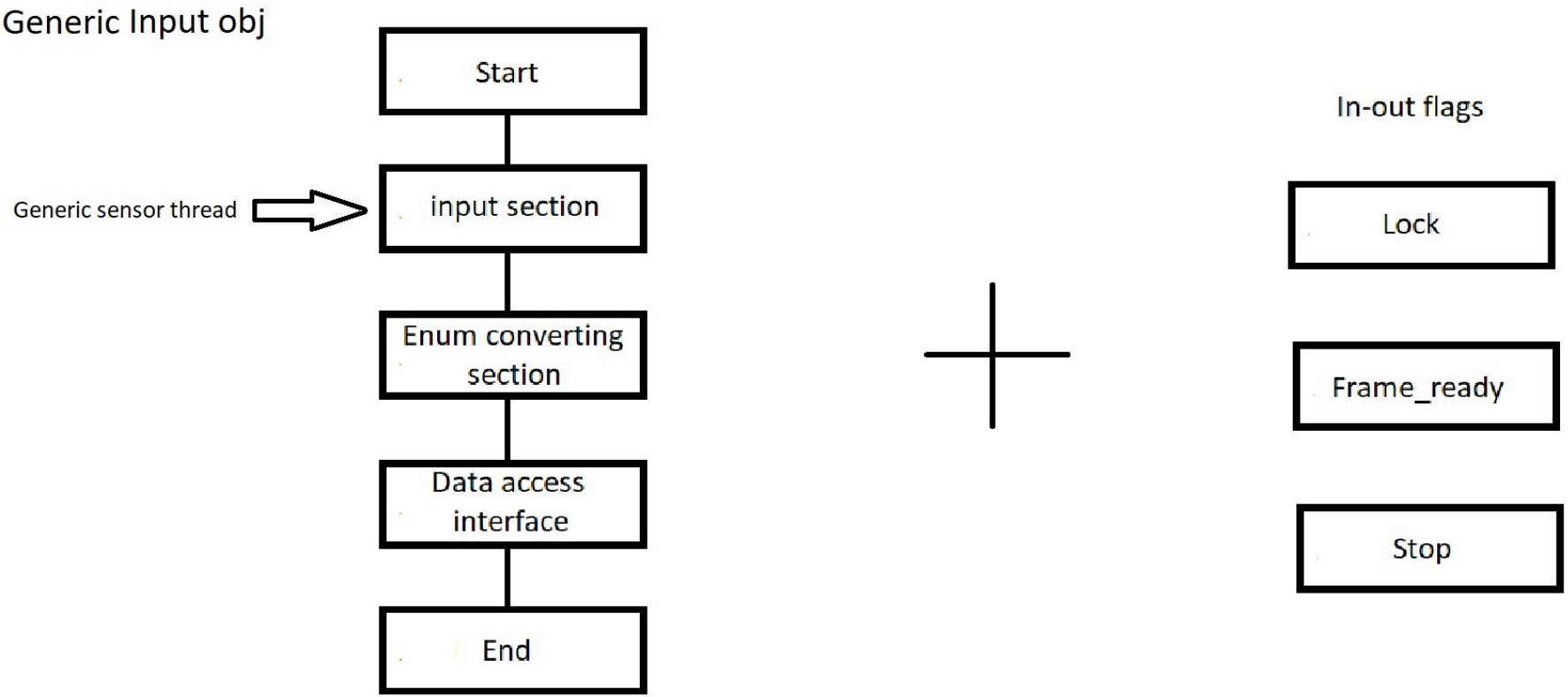




Softcoded vrs interface

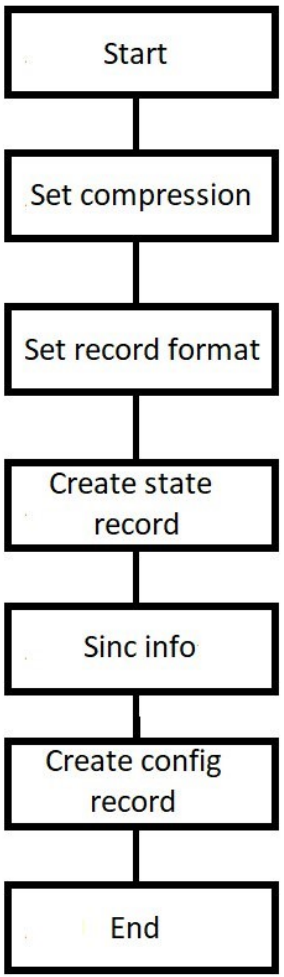


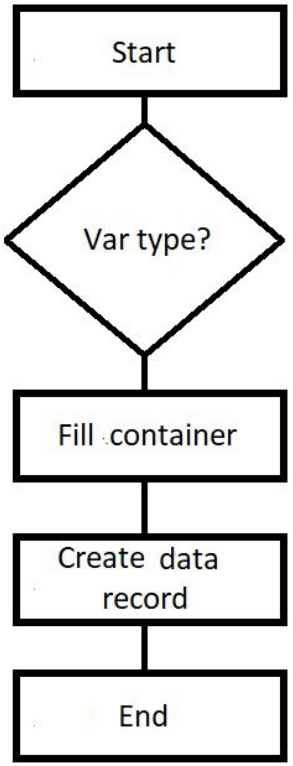
Softcoded vrs interface



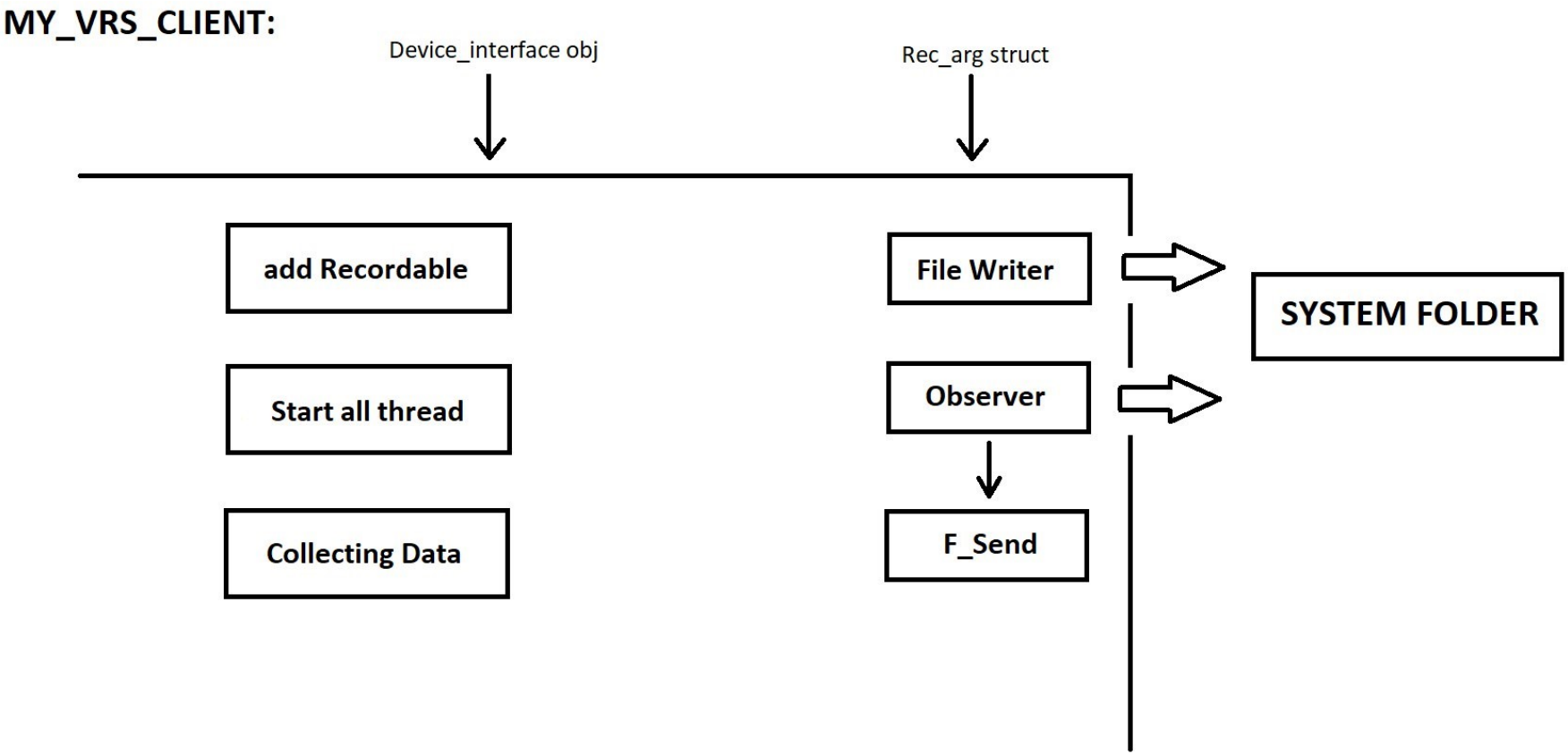
Softcoded vrs interface

Generic stream obj



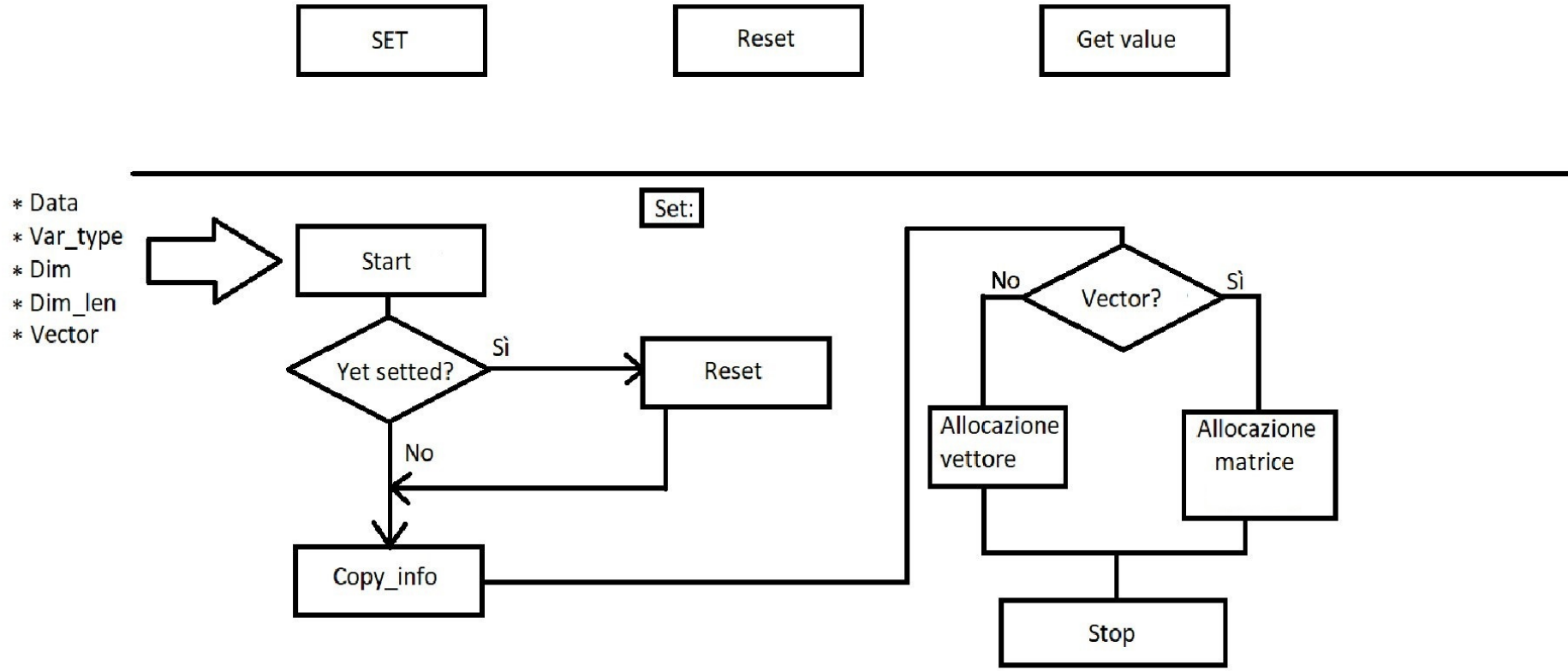


Softcoded vrs interface

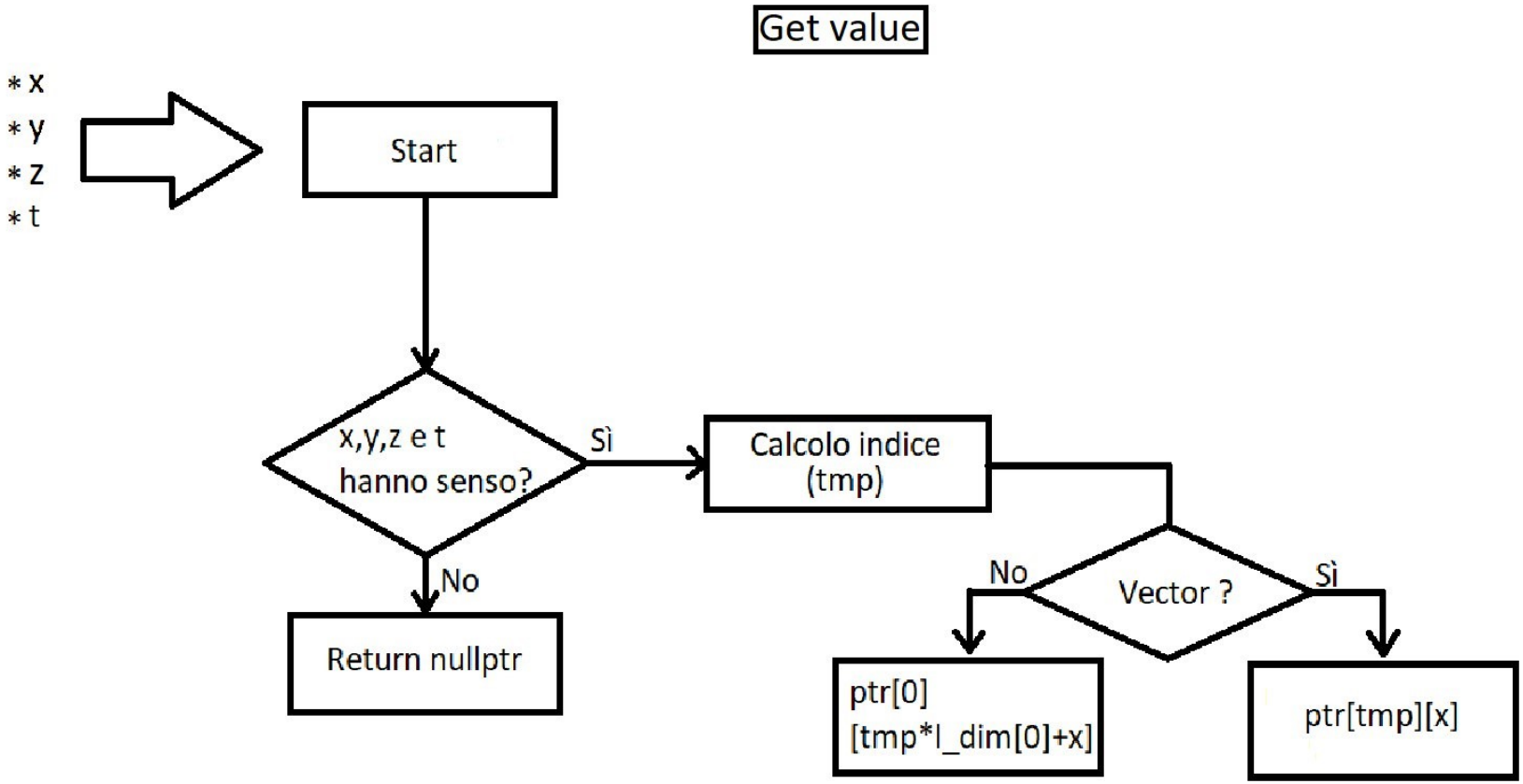


Softcoded vrs interface

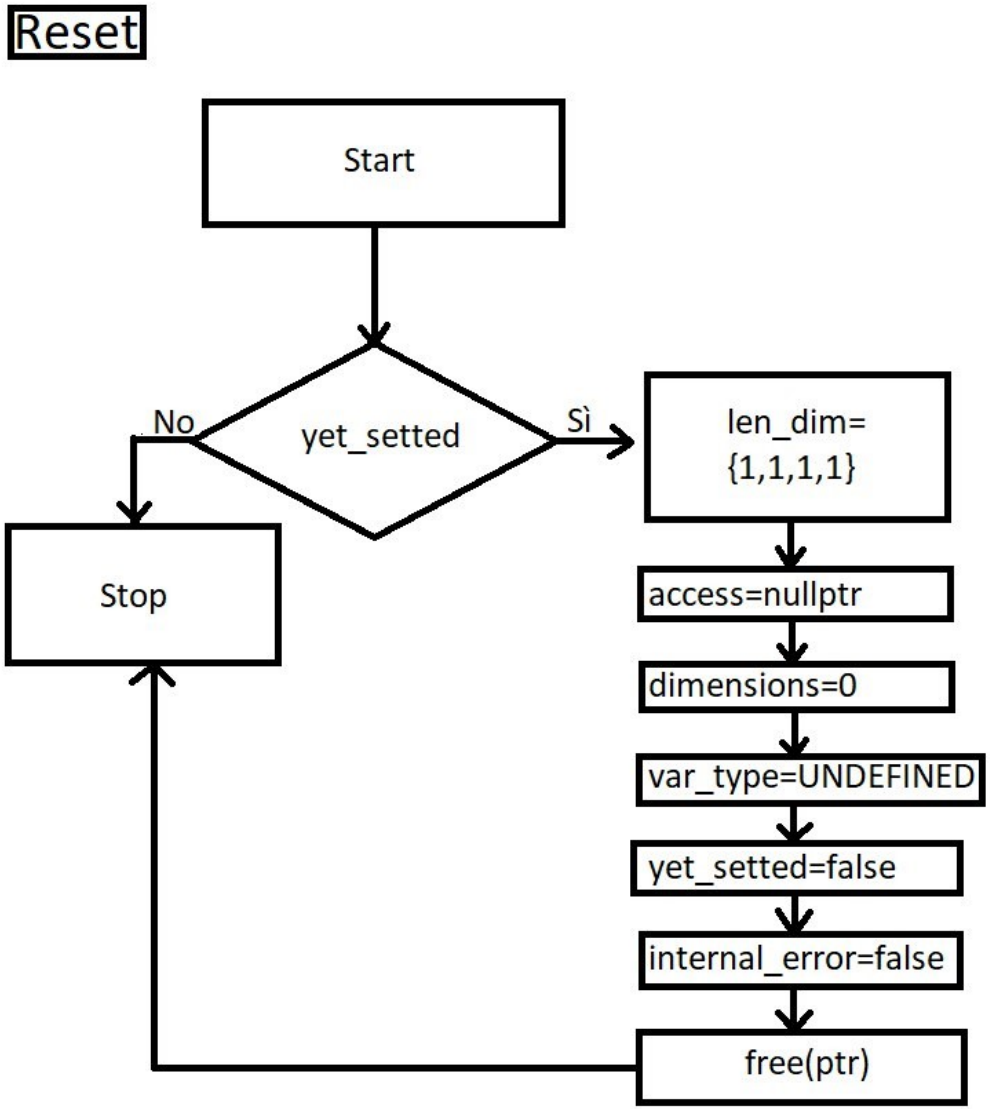
Data_access:



Softcoded vrs interface



Softcoded vrs interface



La Crosscompilazione

Android supporta applicazioni in C# e Java, per cui scelgo Unity. Posso usare codice C++ solo se precompilato e wrappato in una libreria dinamica. Necessaria quindi crosscompilazione per ottenere una libreria dinamica compatibile con il dispositivo bersaglio.
Ho riscontrato i seguenti problemi maggiori:

- Come crosscompilare.
- Come rappresentare i dati da fornire alle funzioni VRS.
- Come scambiare dati in maniera continua tra applicazione e libreria.

La Crosscompilazione

Processo di compilazione attraverso il quale si genera del codice eseguibile da una architettura differente da quella su cui si compila.

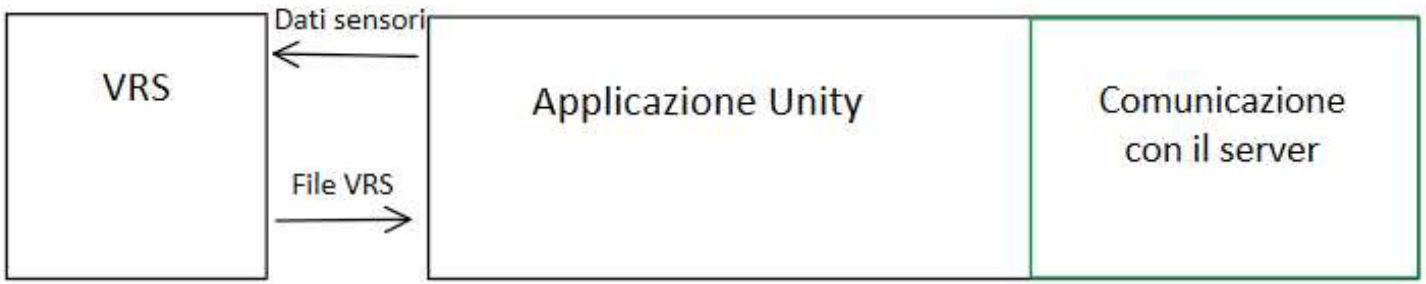
Tool necessari su Linux:

- Android ndk
- Cmake

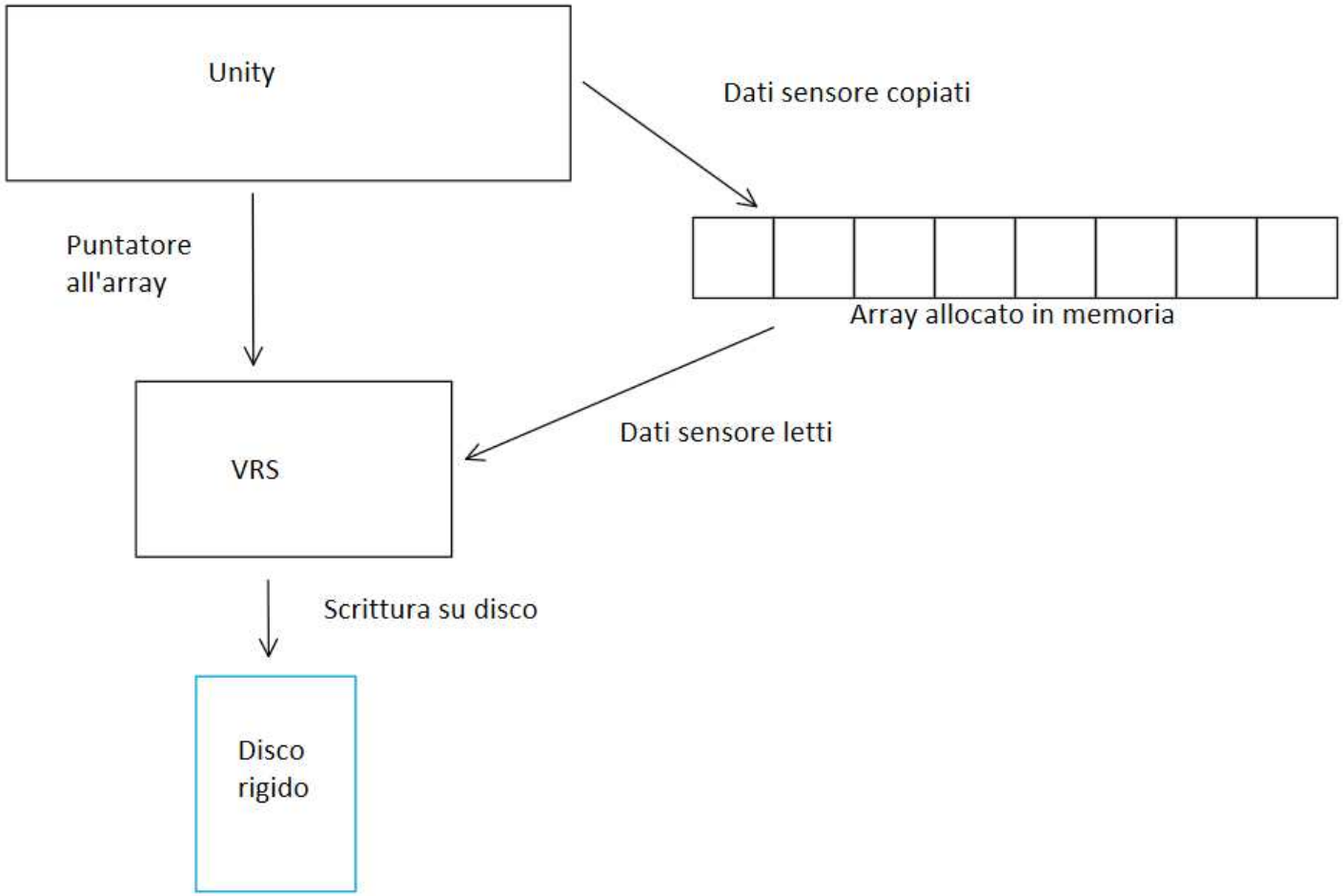
Procedura:

1. Configurazione iniziale
2. Installazione delle dipendenze cross-compilate
3. Cross-compilazione della libreria

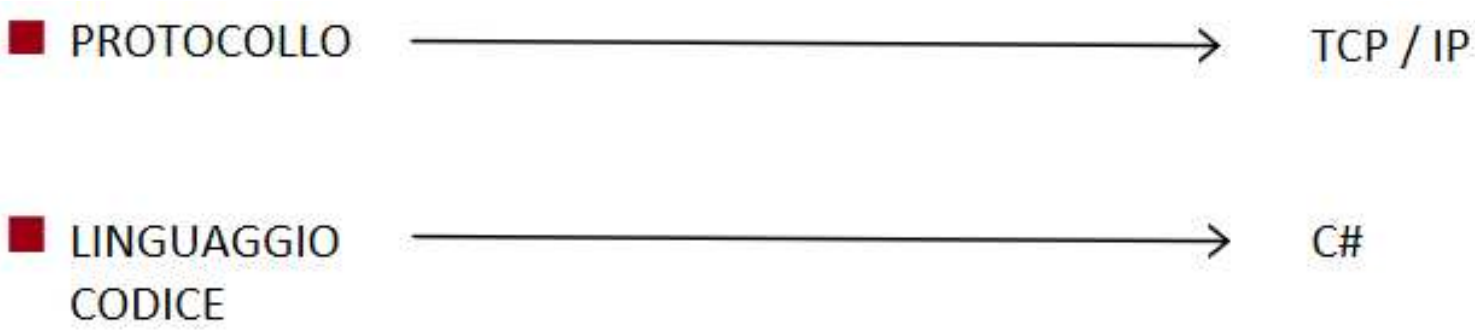
L'applicazione



L'applicazione



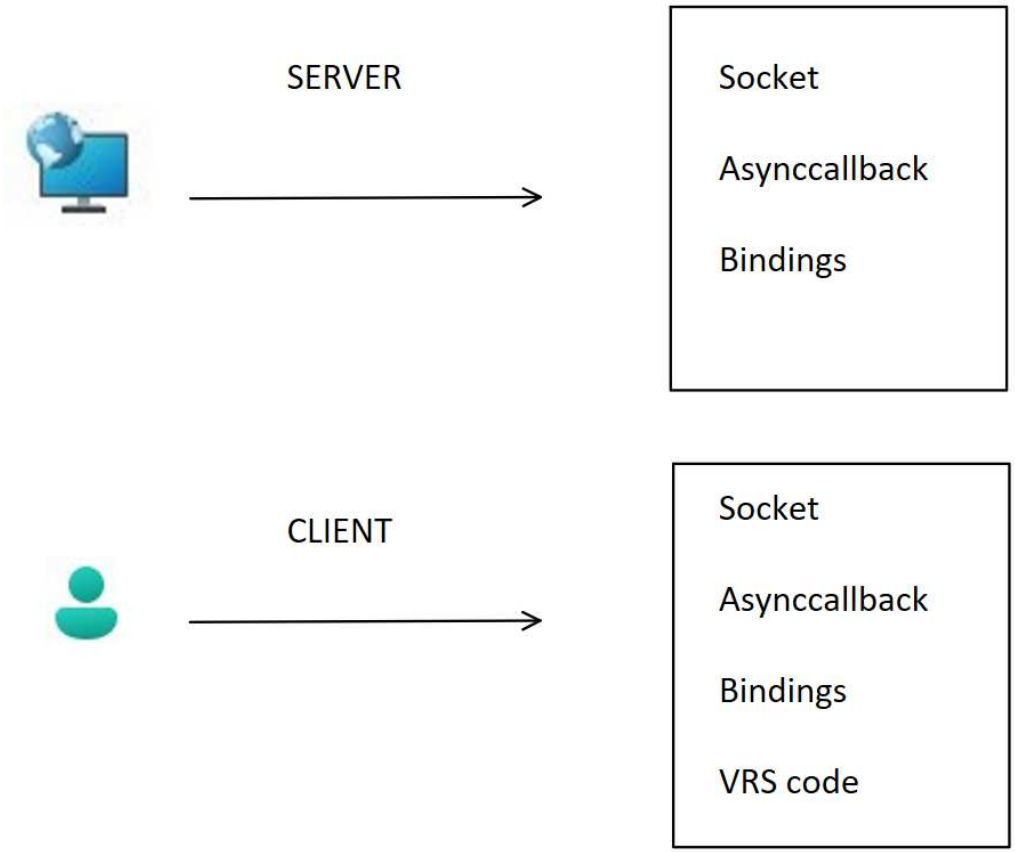
Scambio dati protocollo Client Server



C# si interfaccia con il programma Unity e presenta delle librerie atte alla creazione di un codice Client Server.

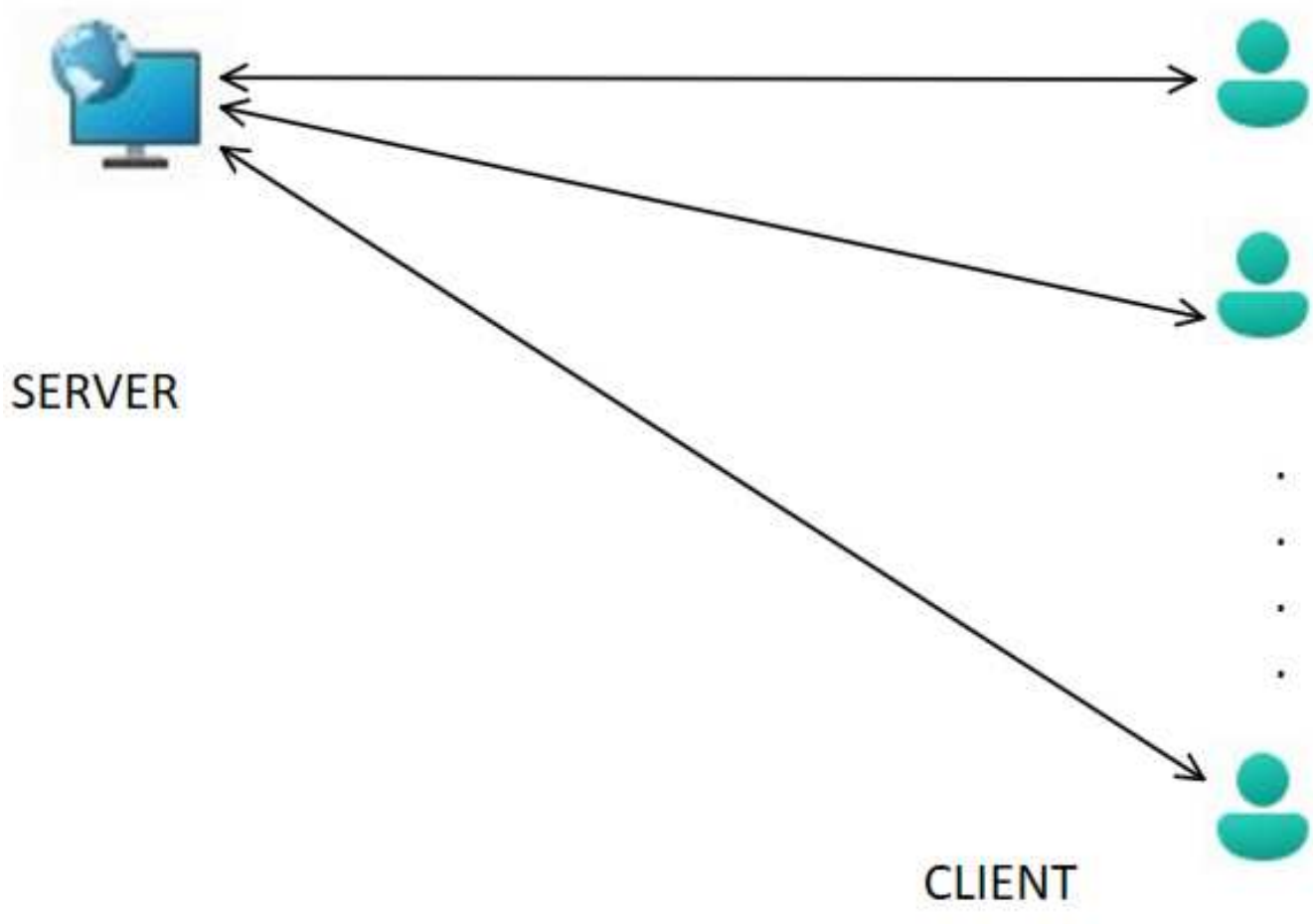
Scambio dati protocollo Client Server

Protocollo Client Server, con connessione AsyncCallback Così costituito:

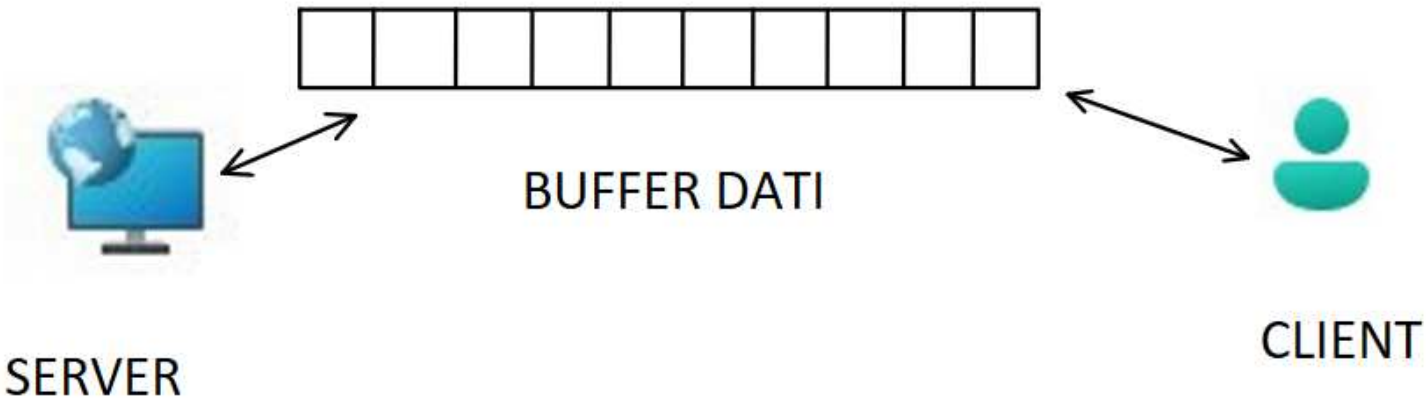


Trasmissione multi client

Server Multi-client così costituito:



Scambio dati Client Server



Implementazioni future, Client Server

Le possibili implementazioni future:

- Introdurre un sistema di gestione tramite database file vrs lato server.
- Implementare un metodo tramite richiesta per l'invio file da server verso client.
- Migliorare la gestione di memoria della connessione Client Server.