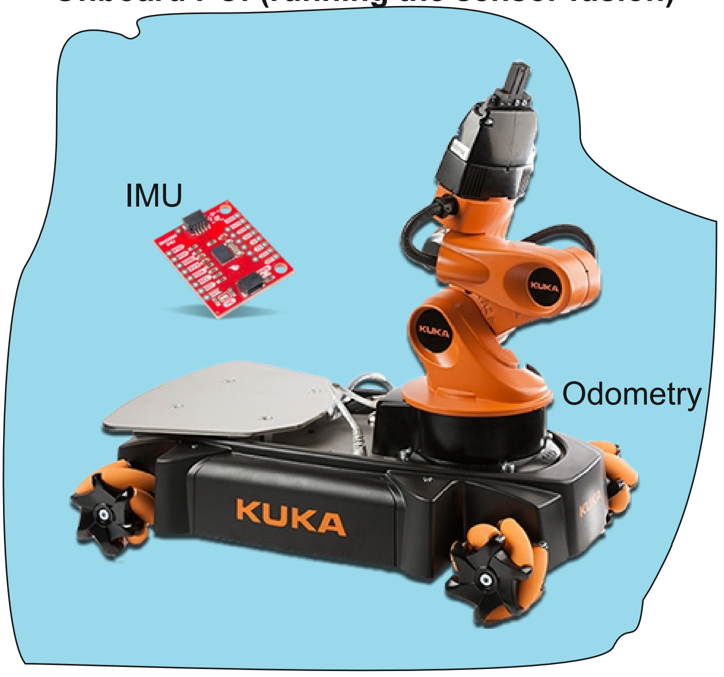
PC on the local network with IP 192.168.0.5

Rapberry PI on the local network with IP 192.168.0.15

Onboard PC: (running the sensor fusion)



network_config_file.json

```
{
        "Remote" : {

                "optical_flow_PC" : {
                        "IP" : "192.168.0.5",
                        "ClockServerPort" : "1234",

                        "Peripheries" : {
                                "optical_flow": {
                                        "Port": "5555"
                                }
                        }
                }.

                "GPS_RP" : {
                        "IP" : "192.168.0.15",
                        "ClockServerPort" : "1234",

                        "Peripheries" : {
                                "GPS": {
                                        "Port": "5555",
                                        „hwm”: 10
                                }
                        }
                }

        },

        "LocalPeripheries" : {
                "IMU" : {
                        "type": "tcp",
                        "address": "5556",
                        "hwm" : 15
                },

                "odometry" : {
                        "type" : "tcp",
                        "address" : "5560"
                }
        }

}
```

main.cpp - a periphery

```
#include "Periphery.h"
#include "ClockSynchronizer.h"
using namespace SF;

int main() {
        try {
                NetworkConfig n("network_config_file.json");
                auto clockServer = InitClockSynchronizerServer(n.GetClockSyncData("optical_flow_PC"));
                Periphery p1(n.GetPeripheryData("optical_flow"));
                while (true) {
                        // Get values (covariance matrix) from the sensor
                        auto v = Eigen::VectorXd::Ones(4) * 5;
                        auto S =Eigen::MatrixXd::Identity(4, 4) * 7;
                        p1.SendValueAndVariance(8, v, S, OUTPUT);
                }
        return 0;
        }
        catch (std::exception e) {
                std::cout << e.what() << std::endl;
                exit(EXIT_FAILURE);
        }
}
```

(similarly)

main.cpp - a local periphery

```
#include "Periphery.h"
using namespace SF;

int main() {
        try {
                Periphery p1(n.GetPeripheryData("IMU"));
                while (true) {
                        // Get values (covariance matrix) from the sensor
                        auto v = Eigen::VectorXd::Ones(4) * 5;
                        auto S =Eigen::MatrixXd::Identity(4, 4) * 7;
                        p1.SendValueAndVariance(8, v, S, OUTPUT);
                }
        return 0;
        }
        catch (std::exception e) {
                std::cout << e.what() << std::endl;
                exit(EXIT_FAILURE);
        }
}
```

(similarly)

main.cpp - the logger

```
#include „Logger.h"
using namespace SF;

int main() {
        try {
                std::string filename = "log_output.log";

                Logger l(filename);
                l.AddPeripheries(NetworkConfig("networkconfig_1.json"));
                l.Start(DTime(2000));

                //Until your job is done...
                std::this_thread::sleep_for(std::chrono::milliseconds(3000));
                return 0;
        }
        catch (std::exception e) {
                std::cout << e.what() << std::endl;
                exit(EXIT_FAILURE);
        }
}
```