UNIVERSITÀ DI PISA

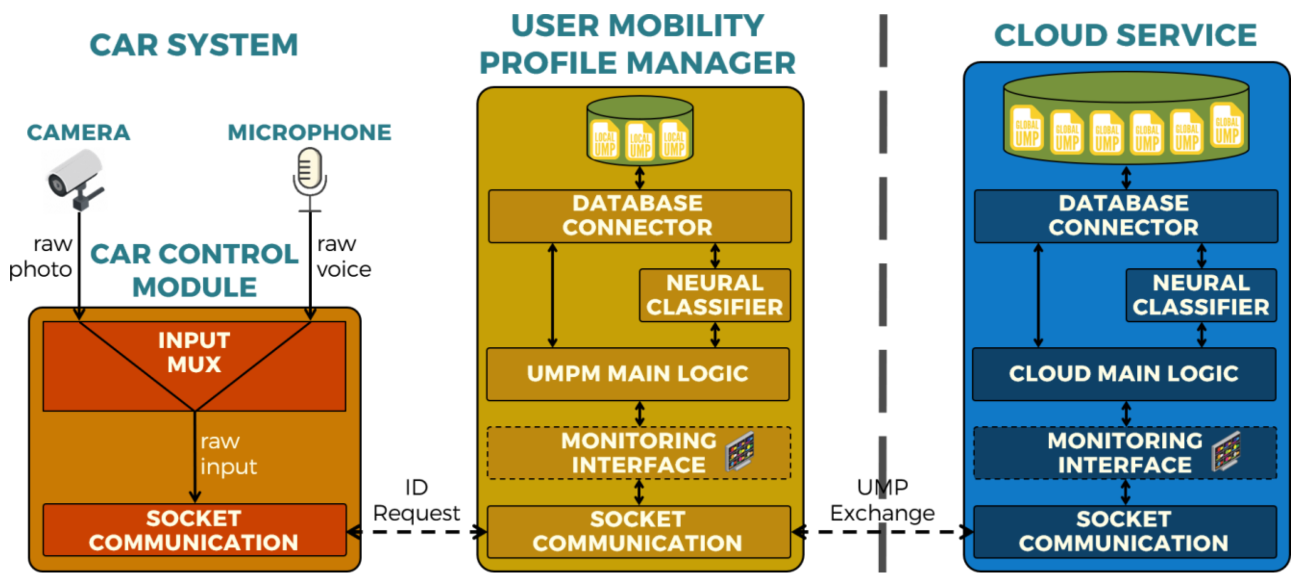
Industrial Applications

The User Mobility Profile Prototype Report

Table of Contents

Prototype Architecture

The following represents the architecture of the User Mobility Profile (UMP) system prototype that was developed for demonstration purposes, where a more detailed description of the functionalities offered by the overall system and the individual components can be found in the related project specification.



The system is composed of three main modules, interchanging data through socket communication

- A Car Control module, representing a stub of the car operating system and control logic, whose task in the context of the prototype is limited to continuously collecting video and voice samples through a camera and microphone and forwarding them to the local User Mobility Profile Manager (UMPM) module to identify the users associated with such inputs.

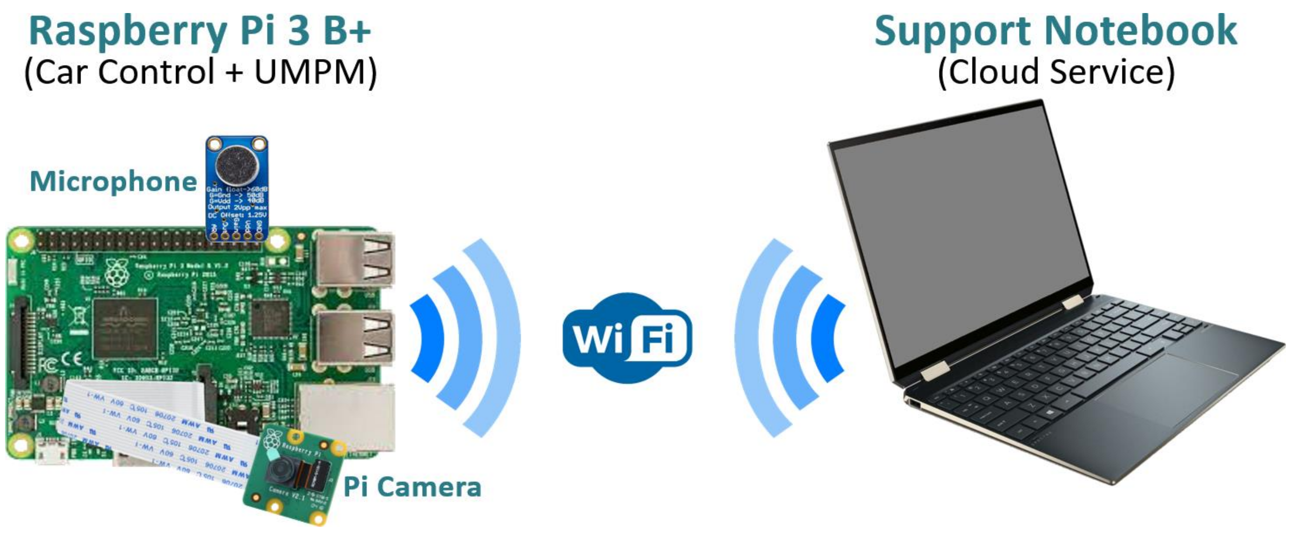
- A User Mobility Profile Manager (UMPM) module, again deployed in the local car environment and which implements the following functionalities:
  - Each input sample received from the Car Control module is matched against the corresponding biometric feature contained in every UMP stored in a local database, where:
    - If a match was found, a local unique identifier associated with such user is returned to the Car Control module.
    - If a match was not found, a temporary profile associated with such user is created and its associated unique identifier is returned to the Car Control module, while at the same time a UMP request containing the unmatched feature is forwarded towards the cloud service to retrieve the actual user profile and then merge it with the temporary profile once it has been returned by the cloud service.
  - An interface allowing other software components to read and update the information contained in the UMPs stored in the local database.
    It should be noted that, while in the broader context of the system such interface would be exploited by the applications running in the car ecosystem, within the scope of this prototype it is instead used by an ad-hoc graphical user interface allowing to test the module and monitor its status.
  - The forwarding of each update of the UMP contents to the cloud service for synchronization purposes.

    - A mechanism implementing the temporary caching of the UMPs of the users that have not recently been matched by the system, where every time a user is matched, if their UMP is found in the local cache their local unique identifier is immediately returned to the Car Control module, while a message asking for possible updated versions of the associated UMP is forwarded towards the cloud service.

- A Cloud Service module, implementing the following functionalities:
    - Every time a UMP request is received from the UMPM module, the biometric input it contains is matched against the corresponding biometric feature held in every UMP stored in a local database, to subsequently return the associated UMP if a match is found.
    - The mirroring of each UMP content update received from the UMPM module to the corresponding UMP in the local database.
    - The verification that a cached UMP received from the UMPM module corresponds to the latest version of such UMP, returning the updated version if that is not the case.
    - A graphical user interface, that similarly to the one integrated within the UMPM makes it possible to monitor the operations being performed by the module as well as read and update the information of the UMPs stored in the local database.

Hardware Deployment

With reference to the architecture previously described, the software modules composing the system were deployed as follows:



- Both the Car Control and the User Mobility Profile Manager modules were deployed in a Raspberry Pi 3 B+ equipped with a microphone and a Pi Camera and running the Raspbian operating system.

- The Cloud Service module was deployed in a support notebook running the Mac OS X operating system.

- Network connectivity between the two devices was obtained through the use of an Ethernet cable.

Implementation Details

Presented below are some details regarding the software implementation of the modules previously described, where more in-depth information can be found in the code source files associated with this report.

- All modules were implemented using the Python language v 3.6, where the latest version (v 3.8) was discarded due to compatibility issues with the Raspbian operating system.

- MongoDB was selected as the Database Management System for storing the User Mobility Profiles, both in the UMPM and in the Cloud Service modules.

- The Cloud Service Database was populated with a collection of photos and voice samples of the developers.

- The speech and face recognition routines were implemented using the AcustID and DLib libraries respectively, while the monitoring interface was implemented with the support of the TkInter library.

Prototype Benchmarks

The following represents a selection of benchmarks relative to the system execution on the chosen platforms, where it should be noted that the Raspberry Pi is equipped with a Cortex-A53 CPU @1.4GHz with 1GB of LPDDR2 memory, while the supporting notebook is powered by an Intel i9-9880H @4.8 GHz with 16GB of DDR4 memory.

	Raspberry Pi 3 B+	**Support Notebook**
Face Recognition Average Matching Time	5,54s	1,25s
Speech Recognition Average Matching Time	8,5s	1,94s
User Identification Average Latency	15s	
Average CPU Utilization during matching	30% (4 cores / 4 threads)	2% (8 cores / 16 threads)
Average Memory Utilization during matching	412MB	637MB