Deliverable **3.1**

SYSTEM GENERAL SPECIFICATION REPORT

Lead Partner:	UNIPR
Authors:	UNIPR, METEDA, VSRO
Contributors:	
Date:	
Revision:	V1.02
Dissemination Level	PUBLIC

Project Acronym: HELICOPTER

Project full title: HEalthy LIfe support through COmPrehensive Tracking of individual and Environmental Behaviors

AAL project number: AAL-2012-5-150

Table of Contents

1. Introduction

This document describes the overall HELICOPTER technical infrastructure, aimed at supporting HELICOPTER services. With reference to Fig. 1, the HELICOPTER system structure view can be summarized as follow:

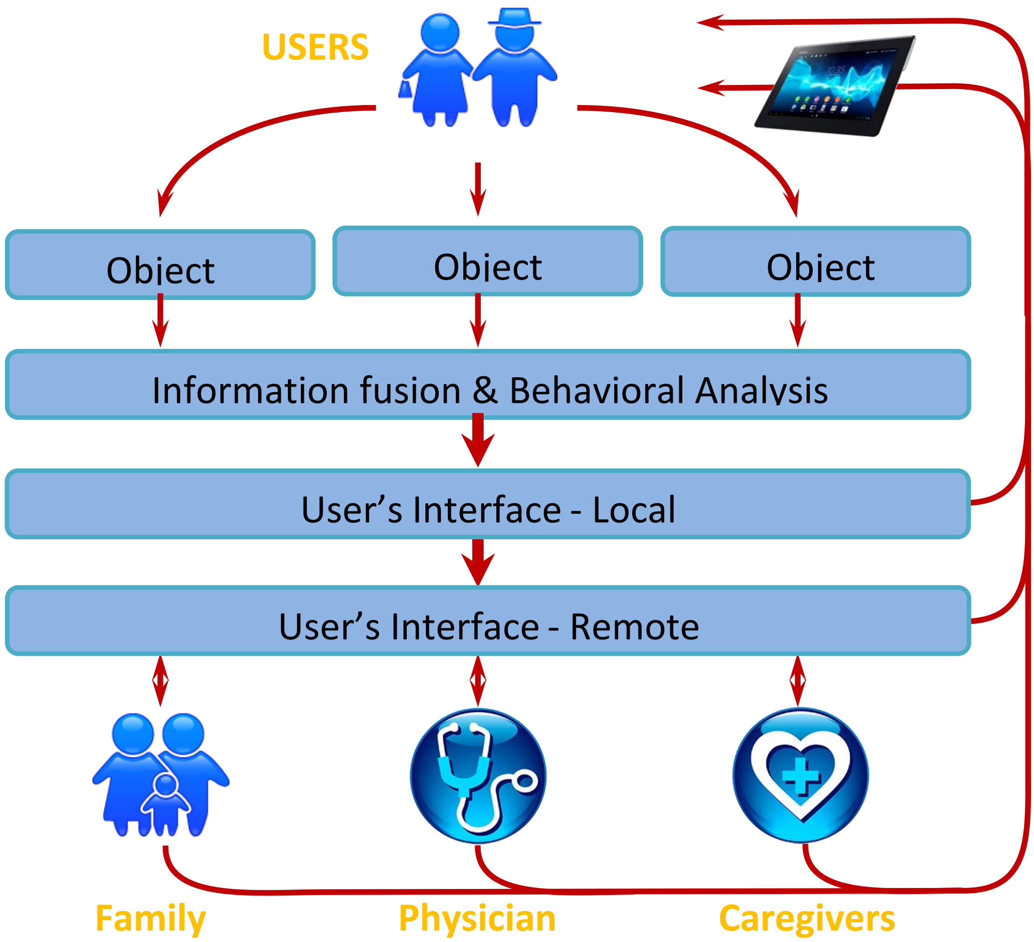


Fig. 1: HELICOPTER System functional view

- the user interacts with several different **sensors**:
    - **clinical** sensors provide the system with accurate data about physiological parameters; their management implies user awareness and action;
    - **environmental** sensors provide data related to the user interaction with the home environment, possibly linked to behavioural meaningful patterns; no user awareness or activation is required; if multiple users are sharing the same environment, criteria for identification of the actual interacting user are needed;
    - **wearable** devices provide information about individual activity, also inherently carrying identification information.
- Sensor data are gathered and fed to a **processing layer,** which implements data fusion and infer behavioural clues

- Based on processing results, user **feedback** is implemented, either at the local (i.e., home) level, or at the “cloud” level, connecting to remote relatives, caregivers, health services.

Such a conceptual vision maps on the more detailed view given in Fig. 2. Most sensors communicate through wireless protocols: for openness and interoperability’s sake, various standards are actually accounted for. Most notably, IEEE 802.15.4-ZigBee and Bluetooth protocols are supported. All wireless sensors communicate with a home gateway device, consisting of a standard PC (or an embedded one, not necessarily exclusively dedicated to HELICOPTER support), equipped with suitable radio transceivers. The PC runs a supervision process, which takes care of several functions, besides managing the actual sensor communication. Data coming from the periphery are suitably abstracted, making them independent of actual physical feature of the given sensor, and stored in a system database. The Helicopter database enable communication among different system modules: in particular, behavioural analysis and anomaly detection is carried out by dedicated modules which acts as “virtual sensors” producing “events” (e.g., anomaly alerts od diagnostic suspicions) which are stored back to the database, in the very same fashion data coming from physical sensors are managed. Similarly, a variety of interfaces can be implemented (aimed at end-users or caregivers) which query the database for system status. I.e., the database is at the crossroads among different subsystems (sensing, processing, interfaces) and thus supports system modularity; a suitable data structure has been devised and implemented, exploiting a MySQL open-source architecture. Details of such structure are given in Appendix 1.

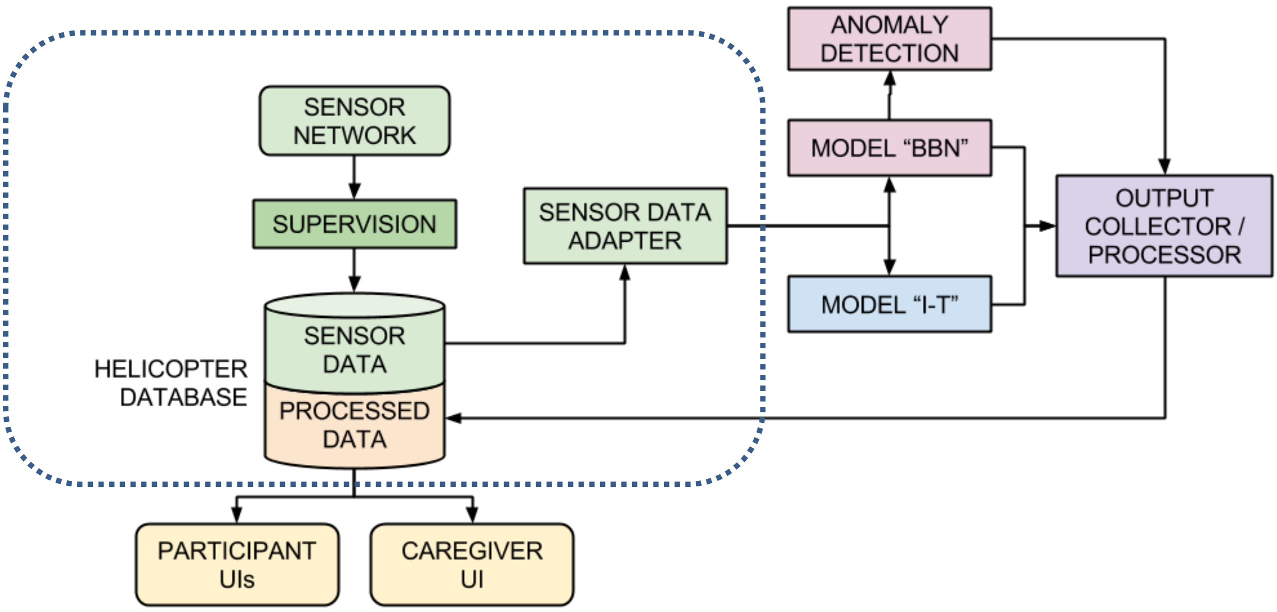


Fig. 2: HELICOPTER System architectural block diagram

This document deals with lowers hierarchical layers only, and more specifically with sensors and with network (dashed area in Fig. 2) and with protocols implementing the whole support infrastructure. Details about behavioural models, service conception and user feedback strategies will be given elsewhere. Sect. 2, in the following, deals with clinical sensors, which have been selected and introduced in the above vision. Sect. 3, instead, deals with wearable sensors, suitable for continuous monitoring of user’s activity and carrying identification information as well. Sect. 4 discusses environmental sensors, while in Sect. 5 the identification/localization issue in a multi-user environment is specifically addressed. Sect. 5 eventually summarizes and draws some conclusions.

2. Clinical sensors

Clinical sensors are exploited for the (self) assessment of physiological parameters: all involved sensors need to be easy to use, suitable for being used by the user himself (or by untrained relatives). The Helicopter system will provide the user with help and guidance in handling clinical sensors and will automatically manage data logging and transmission over the system infrastructure. I.e., no additional burden (with respect to customary device use) is placed on the user for dealing with system communication. It is worth stressing that commercial, off-the-shelf devices have been selected and that the overall system design is making no specific assumption on the specific kind or brand of sensors adopted; i.e., the system is open to communicate with a much wider variety of sensors than that actually planned for pilot trials and will perspectively be able to deal with sensor subsequently made available to the market by third parties, at the expense of properly introducing suitable descriptors in the database structure described later in this document.

The list of currently considered sensors includes:

- **Body Weight scale:**
  A&D Medical UC-321PBT device has been selected. The device features Bluetooth communication capabilities and will be coupled to identification devices (see below) to properly attribute each measure to specific family members.

UC-321PBT					
Information	**Units**	**Range**	**Accuracy**	**Data type**	**Connectivity**
Body Weight	Kg	0 -200 Kg	100 g	Floating	Bluetooth

- **Blood Pressure Monitor:**
  A&D Medical UA-767PBT upper arm blood pressure monitor has been selected, featuring heart-rate measurement, irregular heart beat detection and Bluetooth communication.

UA-767BT					
Information	**Units**	**Range**	**Accuracy**	**Data type**	**Connectivity**
Blood pressure	mmHg	20-280 mmHg	± 3 mmHg	Floating	Bluetooth
Heart rate	bpm	40 – 200 bpm	± 5%	Floating	

- **Pulsoxymeter:**
  The SAT 300 BT fingertip device, manufactured by Contec Medical Devices Co. Ltd, has been selected for measuring the blood oxygen concentration. Heart rate is measured as well, and Bluetooth communication is exploited.

SAT 300 BT					
Information	**Units**	**Range**	**Accuracy**	**Data type**	**Connectivity**
SpO2	%	35% – 100%	+/- 2%	Floating	Bluetooth
Heart rate	bpm	30 – 240 bpm	± 2 bpm	Floating	

- Glucometer

The FORA G31b device, manufactured by Fora Care Inc., has been selected. It allows instant checking of glucose concentration in capillary whole blood, by means of a fingertip puncture. It features Bluetooth communication.

G31b					
Information	**Units**	**Range**	**Accuracy**	**Data type**	**Connectivity**
Glucose level	mg/dL	20-600 mg/dL	-	Floating	Bluetooth

- Portable ECG

  The TD-2202B, manufactured by TaiDoc Technology Corporation, has been selected. It is a handheld electrocardiography devices, suitable for home use. It is Bluetooth-connected, and allows for recording and transmitting EEG waveforms.

TD-2202B					
Information	**Units**	**Range**	**Accuracy**	**Data type**	**Connectivity**
Heart rate	bpm	30 – 250 bpm	± 2 %	Floating	Bluetooth
ECG waveform		Bandwidth 0.05 -100 Hz CMMR > 90 dB Sampling rate 1K s^{-1}			

Bluetooth connectivity toward all devices listed above will be supported by a BT dongle connected to the Helicopter home gateway device. Concerning pilots, not necessarily the whole set of sensors will be made available to each pilot user: some devices have specific clinical scope (e.g., glucometer is needed for diabetes treatment, whereas blood oxygen concentration is relevant in heart failure medical conditions) so the actual set of devices to be deployed at each user's home will be decided on a case-by-case basis, relying on the user's medical profile. Also, some device could be substituted, at pilot implementation time, with different equivalent models.

3. Wearable sensors

Wearable sensors are a key component in the HELICOPTER scenario: we start from previously available work, exploiting the wireless sensor platform MuSA [1], already developed by UniPR and shown in Fig. 3.

Fig. 3 MuSA wearable device

MuSA is a wearable multisensor platform, specifically designed with assistive purposes. It is compliant with ZigBee 2007 PRO standard protocol, and with the ZigBee "Home automation" standard profile thanks to a CC2531 SoC [2]. MuSA is designed to be worn at belt or at chest: it is quite small (78x48x20 mm), and lightweight (about 70 grams, Li-Ion battery included). Different functions can be implemented on the same platform: basic configuration of MuSA includes a call button and automatic fall detection. All of the signal acquisition and processing is carried out by MuSA on-board circuitry. Radio communication is hence kept at a bare minimum (alarm messages and network management), saving battery energy. Two basic building blocks can be identified: a IEEE 802.15.4 radio transceiver, and a microcontroller taking care of ZigBee stack management. The same microcontroller is exploited for digital signal processing.

In its original version, MuSA embeds a tri-axial MEMS accelerometer (LIS331DLH [3], exploited to evaluate human body position and orientation information needed by fall detection algorithms. Within the HELICOPTER project, fall detection features (although still available) are not in the main focus, and wearable sensor MuSA is involved with two basic aims: *i)* providing behavioural information and *ii)* supporting user identification and localization. More specifically, MUSA will contribute to the overall behavioural picture with information about user motion. Both quantitative indicators (walking speed, for instance) and qualitative ones (concerning gait balance, for instance) will be made available to behavioural models implemented at higher hierarchical levels. Also, MuSA will be exploited for user "tagging" and for approximate localization within the home environment, as explained in sect. 4 below. We therefore developed a new version of the MUSA device, featuring a more comprehensive view of the user motion: the 3D accelerometer has been substituted by a full Inertial Measurement Unit (IMU), featuring a 3D digital linear acceleration sensor, and a 3D digital angular rate sensor and a 3D digital magnetic sensor within the same System-on-chip. ST device LSM9DS0-iNEMO [4] has been adopted, taking advantage of backward compatibility with previously installed 3D accelerometer. By suitably combining 9 degrees-of-freedom information, a more accurate and reliable information about user movement can be inferred: at the time of writing the present document, hardware design and implementation of the upgraded MuSA has been completed. Testing is under way and new firmware taking full advantage of new features is being developed. Detailed results will therefore be presented in future documents.

4. Environmental sensors

The use of sensors of many different kinds and functions is planned, in order to feed the behavioural model; in this case too, we rely either on commercial, off-the-shelf devices or (whenever a more specific function is needed) on purposely designed devices. In both cases, we adopt the IEEE 802.15.4/ZigBee wireless transmission protocol, which allows for ample choice of commercial devices and lends itself to efficient power management. Sensors have been selected based also on user-related features, i.e., the need of actual user awareness in sensor management, installation requirements and intrusivity were evaluated.

Environmental sensors available for exploitation in the HELICOPTER framework include:

- **Presence sensor:**
    - Passive InfraRed (PIR) technology
    - Commercial device
    - Small, battery-operated device
    - Placed on a wall
    - ZigBee communication
    - Information: presence of moving persons

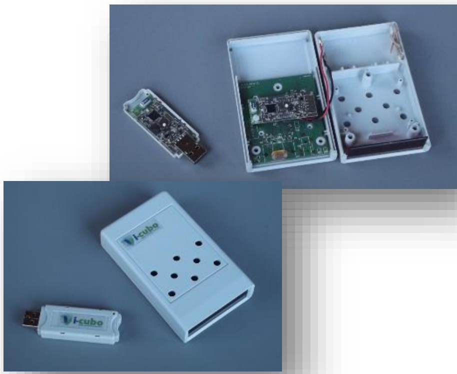
  HELICOPTER relevance:
    - (bath)room access
    - behavioral patterns
    - User involvement/skill: none
    - Installation: requires some skill
    - Sometimes perceived as intrusive

- **Door/drawer sensor**
    - Exploits magnetic contact
    - Commercial device (a custom version has also been developed)
    - Small, battery-operated device
    - Placed at food storage door/drawer
    - ZigBee communication
    - Information: access to the storage

  HELICOPTER relevance:
    - food intake frequency/time
    - User involvement/skill: none
    - Installation: easy

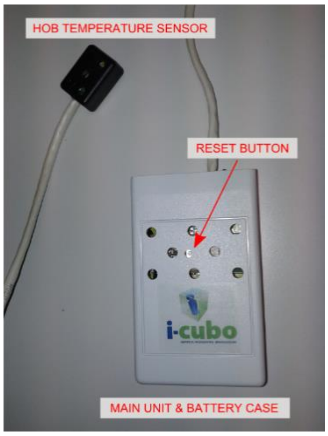
- **Fridge sensor**
    - Custom sensor (designed in the framework of AAL-JP “FOOD” project and further developed here)
    - Small, battery-operated device
    - Placed inside (any) fridge
    - ZigBee communication
    - Information: door opening, temperature, humidity

  HELICOPTER relevance:
    - food intake frequency/time
    - User involvement/skill: low
    - Installation: none



- **Hob sensor**
    - Custom sensor (designed in the framework of AAL-JP “FOOD” project and further developed here)
    - Small, battery-operated device:
    - differential reading (2 coupled temperature sensors)
    - Placed close to (any) hob
    - Measures heat, not actual hob ignition
    - ZigBee communication
    - Information: cooking activity, indirectly food intake

  HELICOPTER relevance:
    - food intake frequency/time
    - User involvement/skill: none
    - Installation: moderately complex (position sensitive)



- **Bed/chair occupancy sensors**
  - Custom sensor
  - Small, battery-operated controller
  - Soft pad (replaceable, many different sizes)
  - ZigBee communication
  - Information: bed/chair occupancy

  HELICOPTER relevance:
  - wake/sleep assessment
  - activity monitoring
  - User involvement/skill: low
  - Installation: easy
  - Sometimes perceived as intrusive

- **Water flow sensor**
  - Custom controller, commercial sensor
  - Small, battery-operated controller (not shown)
  - Placed along the water pipes
  - ZigBee communication
  - Information: water flow metering

  HELICOPTER relevance:
  - toilet flush count
  - (drinking) water consumption
  - User involvement/skill: none
  - Installation: complex/invasive (needs plumber work, accessing piping)

    (Alternatively, mechanic/magnetic switches can be exploited for toilet flush count)

- **Power meter**
  - Commercial device
  - AC-mains operated
  - Smart-plug
  - ZigBee communication
  - Information: electrical power instant consumption

  HELICOPTER relevance:
  - Appliance activity monitoring
  - User involvement/skill: low
  - Installation: easy

  (can be used also to monitor cooking activity, if electrical/induction hobs are used)

5. Identification and localization

All sensors listed in Sect. 2 and 4 are considered as "environmental" sensors, since they are located at given positions into the home environment and are related to home activity and actions; in order to contribute to the behavioural picture onto which the HELICOPTER vision is grounded (and assuming that, in the general case, more than one single person is living in the monitored environment), information coming from such sensors, however, need to be correlated to a given user. I.e., in a multi-user environment, we need to attribute data coming from environmental sensors (e.g., opening of the fridge door) to a specific user. This calls for identification/localization features: interaction between wearable (i.e., personal) sensors and environmental sensors has been exploited to this purpose. Design, implementation and test have been carried out in the framework of the HELICOPTER project: therefore, a more detailed account of this is given below.

In general, indoor location is a complex and multi-faceted issue. A large number of system have been proposed, based on various methods or technologies, ranging from RSSI [5] or time of flight [6] to geo-magnetic field [7] and Mutually Coupled Resonating Circuits [8]. It is worth underlining, however, that Helicopter aimed application poses quite different constraints (in terms of required accuracy, system intrusiveness and affordable costs) with respect to main fields localization technologies have been developed for. Reliable, handy and low-cost solution are needed for daily living in home environment: we therefore developed a low-cost gateway system named CARDEAGate, capable of detecting a person crossing a door or any given gateway, and, if he is wearing a MuSA device, to identify him.

CARDEAGate operating principle exploits the absorption of a radio signal power caused by the body of the person crossing the radio link propagation path [9] and is based on the consequent modulation of the Received Signal Strength Index (RSSI, [10]). Such an approach lend itself to fairly simple implementation and integration within a ZigBee sensor network, and well fits a wide range of AAL-oriented services

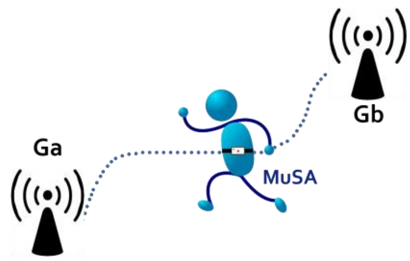


Fig. 4. The CARDEAGate structure

The CARDEAgate system operating principle is illustrated by Fig. 4 and include two basic steps: *i)* **detection** of a person crossing the gate line and, *ii)* **identification** of such a person (provided he's wearing a MuSA device).

CARDEAgate consists of a couple of ZigBee transceiver (referred to as G_a and G_b in this paper) each having the size of a standard USB flash drive (Fig. 5). They can be mounted, for instance, on the two edges of a door or elsewhere, the line between G_a and G_b being the monitored region. Unlike optical-based sensors, CARDEAGate does not need line-of-sight visibility, so it can easily be embedded into doorframes, home furniture or stand behind curtains and thin (non-metallic) walls. This makes the system also less intrusive, and allows for smooth integration into most home environments.. Detection exploits the perturbation in EM waves propagating between the two transceivers caused by the crossing user's body.

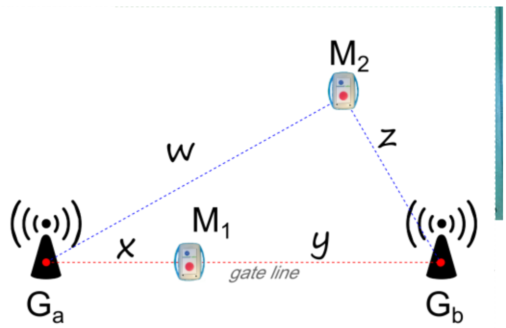


Fig. 6. Identification strategy

To this purpose, G_a and G_b send each other a message every 200ms and monitor the RSSI: if a sudden loss is observed (i.e., the user's "shadow"), a person crossing the gateway is detected.

If the crossing person actually wears a MuSA device, the identification procedure is started: G_a and G_b, in fact, transmit an identification request to the MuSA devices in the network and send the RSSI of received replies to the ZigBee network coordinator. At the supervision level (managed by the HELICOPTER home gateway) a decision strategy is implemented, which finds out which MuSA most likely crossed the gateway.

To this purpose, we start from the simple consideration the device crossing the gateway is the one which features the lowest sum of distances from either gate transceiver: in Fig. 6, for instance, M_1 is the device crossing the gate line, whereas M_2 lies nearby. Elementary geometrical reasoning yields:

$$x + y < w + z \quad (1)$$

where x, y, w and z are the distance between mobile and fixed nodes, as indicated in figure.
According, for instance, to [10], RSSI can be correlated to the distance d between the transmitter and the receiver:

$$RSSI = -(10\, n\, \log_{10} d + \alpha) \quad (2)$$

where n is the signal propagation constant, and α is the reference signal strength (evaluated at a 1m distance). For a given transceiver couple (i, j), after some manipulation, the distance $d_{i,j}$ can be made explicit:

$$d_{i,j} = k * 10^{-RSSI_{i,j}} \quad (3)$$

In Eq. 3, k is a constant involving signal propagation features (n, α) and is therefore related to the actual signal path. Thus:

$$\begin{aligned} x &= k_{1,A} * 10^{-RSSI_{1,A}} \\ y &= k_{1,B} * 10^{-RSSI_{1,B}} \\ w &= k_{2,A} * 10^{-RSSI_{2,A}} \\ z &= k_{2,B} * 10^{-RSSI_{2,B}} \end{aligned} \quad (4)$$

In principle, of course, the propagation constant k is not necessarily uniform along different triangulation legs. Nevertheless, for the sake of simplicity, we can assume a constant value as a worst case scenario: by supposing that propagation along the gate line is better of propagation along longer path (which does not hold rigorously true in general, but makes sense for a sensible gate placement) considering a uniform propagation constant k (i.e., $k_{1,A} = k_{1,B} = k_{2,A} = k_{2,B} = k$) may result in relative underestimation of “outer” device distances, thus not harming the selection criteria below.
Hence, from Eq. 1, the decision test yields:

$$10^{-RSSI_{1,A}} + 10^{-RSSI_{1,B}} < 10^{-RSSI_{2,A}} + 10^{-RSSI_{2,B}} \quad (5)$$

If the inequality (5) holds true, crossing of M_1 is assessed, and M_2 otherwise. More generally speaking, if a multiplicity of MuSA devices is considered, the one crossing the gate line is selected by looking for the minimum value of distance sum:

$$S_j = d_{j,A} + d_{j,B} = k\ (10^{-RSSI_{1,A}} + 10^{-RSSI_{1,B}}). \quad (6)$$

The detection performance of the system was tested, and some preliminary evaluations were made, aimed at assessing practicality and reliability of the proposed approach. The gateway was placed in the middle of an empty room in order to minimize interferences caused by furniture or other objects that could interact with the wave propagation, at a height of 1m. Testing patterns are illustrated by Fig. 7. Among investigated features were:

a) **Sensitivity on the gate width**, (i.e., distance between G_a and G_b). Gate widths ranging from 0.5 to 3 m were accounted for, with intermediate steps of 0.5 m. Central crossing (Test 1 in Fig. 7) and lateral crossing (Test 2) were performed. By repeating each measure 50 times for each width and position, no false negative was actually incurred in (i.e., all 600 passages were correctly detected). In both tests, 100% of actual passages was correctly detected, independently of gate width changes in the given range.

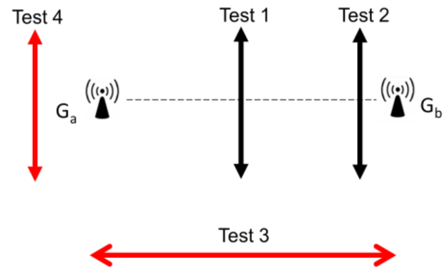


Fig. 7. Detection test patterns

b) **robustness to false positives**: to this purpose, walking paths close to the gateway but not actually crossing it were tested, both in the parallel (Test 3) and transverse (Test 4) directions. Again, test were repeated 50 times for each configuration. The aim of the Test 3 was to find the lower distance from the gateway line that could give false positives. An accurate measurement of this turned out to be unpractical, due to the complex shape of the human body and to the inherent uncertainty in controlling human gait. Nevertheless, it was found that such a figure slightly depends on the gate span, with the "safe" walking distance (i.e., minimum distance from the gate line not causing false positive) ranging from a few centimeters for the narrowest gate span, up to some 60 cm for the widest span tested. This is of course due to the non-selective radiation patterns of used antennas (as mentioned, standard ZigBee transceiver were used, and no critical alignment/calibration procedure was required) and to radio waves reflection/scattering, which are unavoidable in real-world environment. Nevertheless, obtained figures are more than suitable for the aimed purposes. Test 4, eventually, allows for evaluating selectivity of the gate when placed in an open space, discriminating passaged within the gate opening and outside it. In this case too, no false positive was detected, regardless of the actual distance from gateway side and of the gate span.

After that, a gateway was installed on an actual door (110cm wide), thus accounting for more realistic boundary conditions: in this case too, 100% of actual passages were correctly detected, with sensitivity fading to 0 at a 30cm distance from the door threshold. Despite a more accurate and quantitative characterization is under way, such performance is quite promising, with respect to both the drift error reset in inertial navigation and for behavioral pattern inference purposes.

Next, identification performance was tested for. As mentioned before, identification comes from pairing RSSI information coming from the communication between the MuSAs in the network and the fixed gateway devices. Of course, RSSI is meaningful only when a 1-hop messaging path is exploited (i.e., direct communication between the gateway and MuSA device occurs). Since MuSA is a battery-operated device, it relies on a sleep-wake cycle to reduce power consumption (as of ZigBee protocol [11]). Obviously, it can receive messages only when awake, with the ZigBee routing node storing undelivered message until destination node awakens. This makes it impossible to communicate with a MuSA device using a 1-hop message at any time: to cope with this, once a passage is detected by a gateway, G_a and G_b send to each MuSA in the network a message, to which they will reply in 1-hop mode once awaken. RSSIs associated to such replies are then forwarded to the supervisor, which takes care of the decision about identification, according to the strategy depicted above.

A first test has been carried out, in which a person wearing a MuSA (M1) walked through the gateway (installed on an actual door) and another person, with a second MuSA (M2), was standing elsewhere. 40 tries were carried out and the results were evaluated.

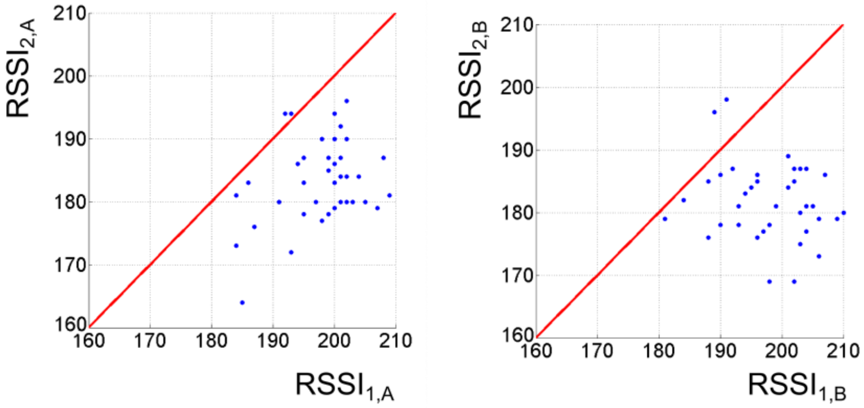


Fig. 8. RSSI scatter plots, as evaluated at gate edge G_a (left plot) and at gate edge G_b (right plot)

Fig. 8 shows the plot of the RSSI retrieved by G_a (upper plot) and G_b (lower plot). Indices related to M_1 are assumed as the plot abscissa (*x* axis), whereas the ordinate refer to M_2 (*y* axis): each dot refers to the same test, as sensed by either transceiver. Therefore, if the dot lies below the diagonal line (*y*=*x*), a greater RSSI was associated to M_1 than to M_2 (i.e., $RSSI_{1,i} > RSSI_{2,i}$), implying (under the aforementioned simplifying assumptions) that M_1 is closer to the given gate edge; if the dot lies above the diagonal, of course, the opposite condition occurs (i.e., $RSSI_{1,i} < RSSI_{2,i}$) indicating M_2 is actually closer.

The test configuration, exploiting a standard door width (so that just a single user is allowed to pass at a time), makes quite likely that the device crossing the gate line is actually closer to both gate edges than other devices. Nevertheless, it is shown that a simple proximity test is not accurate enough: due to both the random device placement and the wave propagation features in a real environment, a few points happen to lay above the diagonal in either plot. This shows that pairing transceiver responses is inherently necessary: accounting for the geometrical algorithm introduced above, ambiguities are solved, as shown in Fig. 9.

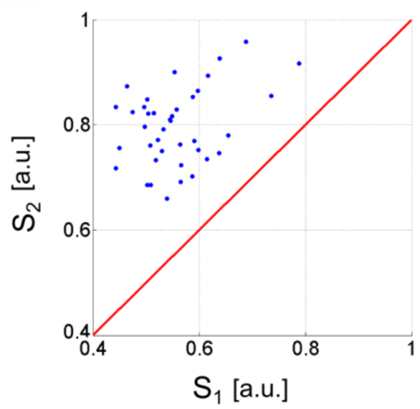


Fig. 9.Scatter plot of distance sums

Here, the scatter plot refers to the estimated total distances defined by Eq. 6, and show that all of the events are correctly interpreted: every point lies above the diagonal indeed, yielding $S_2 > S_1$. Also, fair clearance from the diagonal line can be assumed as a confidence indicator for the inferred information. At the time of writing this document, a more thorough testing of the proposed approach is under way, and more results will be provided in subsequent project deliverables.

The strategy introduced so far allows to identify a user at specific locations within the home environment (where gates have been placed). CARDEAGate features, in fact, the basic functionality of any “sight-line” sensor (e.g., infrared barriers), detecting any person crossing the gateway line (regardless of him wearing a MuSA device), however posing much less stringent constraints in terms of placement, alignment and maintenance. If the user wears a MuSA, further “active” interaction modes with the passing user are enabled, allowing for user’s identification. CARDEAGate can be exploited to monitor the access to zones of interest (a room or even the fridge, an armchair, etc.). This is not to be considered as a full “localization” system, but provide useful hints in “attributing” actions detected by environmental sensor to actual home guests. Such a strategy can be improved in many ways:

Taking advantage of the embedded inertial unit, inertial navigation tracking can be implemented on MuSA: starting from any known reference location, in fact, relative position can be inferred, based on the numerical (double) integration of acceleration data; stockpiling integration errors, however, yields some drift in the results and progressively affect localization accuracy. CARDEAGate then allows for resetting such a stockpiling error, restoring a reliable reference for inertial tracking [12]. Full integration between MuSA inertial navigation capabilities and CARDEAGate beaconing is currently being investigated. Of course, knowing actual location of the user within the home environment allows for personal tagging of environmental sensor data.

Alternatively, almost all environmental sensors listed above can be coupled to the CARDEAGate beacon feature: once a given sensor (e.g., the fridge door) recognize some activity, an identification request can be triggered. This can be accomplished by placing a gate in the immediate nearby, or even by exploiting the very same sensor transceiver for measuring and comparing RSSI received by different MuSA devices scattered within the room. So doing, a simple “maximum RSSI” search can be implemented with no need of additional hardware: the function can be implemented as an additional firmware module.

Finally, at a higher hierarchical level, raw data can be combined, accounting for some combinational and sequential logic: artificial reasoning may help in solving possible ambiguities and in validating identification data. E.g., by matching sequence of gate crossings and PIR responses, the system may “follow” users along their home walk, and thus have some preliminary information about distribution of users in different rooms; this, in turn can be used to rule out some user-sensor combinations and to simplify the identification/attribution task.

Of course, methods above does not strictly guarantee a reliable identification: it is worth to be stressed, however, that such identification feature does not trigger any “mission-critical” activity and simply support building of behavioural profiles, on a statistical basis. Hence, some (limited) error rate in tricky situations can be tolerated, without jeopardizing the whole picture.

6. Conclusions

In this document, the complete home infrastructure supporting HELICOPTER services is described. More specifically, clinical, wearable and environmental sensors have been selected and implemented. Both commercial and custom devices have been considered. Since behavioural modelling requires individual data discrimination, attributing information coming from environmental sensors to a given user (within the family members set, for instance) is of paramount importance. Hence, specific identification features have been studied, with the aim of finding general solution and of trading off among performance, cost and system intrusivity. The whole system is currently under test, and will result, first, on the HELICOPTER system demonstrator (D3.2) and, subsequently, in pilot kit deployment at participating user homes.

References

[1] V. Bianchi, P. Ciampolini, I. De Munari, F. Grossi, «MuSA: a multisensor wearable device for AAL», proc. of FedCSIS 2011, pp. 375-380, 2011

[2] http://www.ti.com/product/cc2531

[3] http://www.st.com/web/catalog/sense_power/FM89/SC444/PF218132

[4] http://www.st.com/web/en/catalog/sense_power/FM89/SC1448/PF258556

[5] Y. Tian, B. Denby, I. Ahriz, P. Roussel, «Practical Indoor Localization using Ambient RF», 2013 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp 1125-1129, 2013

[6] G. Santinelli, R. Giglietti, A. Moschitta, «Self-Calibrating Indoor Positioning System Based On ZigBee Devices», 2009 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp 1205-1210, 2009

[7] A. Saxena, M. Zawodniok, «Indoor Positioning System Using Geo-Magnetic Field», 2014 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp 572-577, 2014

[8] G. De Angelis, A. De Angelis, M. Dionigi, M. Mongiardo, A. Moschitta, P. Carbone «An Accurate Indoor Positioning-Measurement System Using Mutually Coupled Resonating Circuits», 2014 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp. 844-849, 2014

[9] J. Wilson, N. Patwari, «Radio Tomographic Imaging with Wireless Networks», IEEE Transactions on Mobile Computing, vol. 9,

n. 10, pp. 621-632, 2010
[10] S. J. Parker, J. Hal, W. Kim, «Adaptive Filtering for Indoor Localization using ZIGBEE RSSI and LQI Measurement», Adaptive Filtering Applications, vol. 14, pp. 305-324, 2007
[11] http://www.zigbee.org/Standards/Downloads.aspx
[12] O. J. Woodman, «An introduction to inertial navigation», University of Cambridge technical report, n. 696, 2007

Appendix 1: HELICOPTER database structure

In this appendix, a description of the data structure which has been designed for ruling data management at the infrastructure level is presented.

In order to ease the interaction among different involved partners, data abstraction and standardization is needed: to this purpose, a shared database architecture has been agreed upon by involved partners. In order to allow for (future) interoperability, the overall structure is highly configurable: configuration data are stored in the database itself and need to be defined just once, as the actual system features and local configuration are defined.

For the sake of generality, a highly hierarchical structure has been foreseen, in which system components are grouped at different levels:

Objects are any kind of entity generating information, such as actual “contact points” with user’s home and activities (sensors, etc), or “virtual sensors” (result of data analysis).

Objects communicate with the database through a supervising manager subsystem: a multiplicity of such subsystem is foreseen (for instance, a subsystem may deal with BlueTooth-enabled clinical sensors, while a second one may deal with Zig-Bee environmental ones). Such entities are called “gateways” in the following (or hosts).

A summary view of the database organization is given in Fig. A.1

MySQL has been adopted for implementation, because:

- it is distributed as a free, open-source tool;
- it is available on many software platforms (GNU-Linux and MS-Windows, in particular);
- API (Application Programming Interface) and libraries are available in many programming languages.

Aiming at maximizing interoperability, an “Internet-of-things” oriented approach has been devised: all devices in the Helicopter system will be associated to a unique IPv6 (128 bit integer) identifier, and all interaction with the system will be managed through the database itself.

Each “object” is then described by an arbitrary number of “variables”, defining specific object features. Variables are described by a 16-bit “variable ID” and may have different types, define in a specific table of the database. Stored data can be in any case reduced to one of these three main data types:

- 64-bit signed integer
- Single-precision floating point
- Arbitrary lenght text string

Such a choice allows for ample generality, at the same time making easy to manage and interpret data. Also, development and debugging phases are made simple by the “readability” of data in their native form.

As shown in Fig. A.1, a number of tables have been defined: valid identifiers in all tables start from 1, with 0 being reserved to NULL value.

All tables marked as "STATIC" are compiled at first system configuration and updated only if some structural changes are implemented in the installation itself.

HOST_TABLE (Gateways) (STATIC)

Lists all "gateways" (i.e., different subsystems accessing the database in independent fashion). Each table record is related to a gateway.

HOST_ID	MANUFACTURER	DESCRIPTION
smallint unsigned	varchar[128]	varchar[128]

Host_id: gateway unique identifier
Manufacturer: ASCII text string, reporting manufacturer information
Description: ASCII text string, reporting gateway information.

OBJECT_TABLE (STATIC)

Lists object identifiers and descriptions.

OBJ_ID	IPv6_ADDRESS	MANUFACTURER	DESCRIPTION
Smallint unsigned	Ipv6 char[32] (nibble ASCII)	varchar [128]	varchar [128]

Obj_id: unique object identifier
IPv6_address: unique object address
Manufacturer: ASCII text string, reporting manufacturer information
Description: ASCII text string, reporting gateway information.

VARIABLE_TABLE (STATIC)

Defines variables.

VAR_ID	NAME	DATATYPE_ID	UNIT_ID	MIN VALUE	MAX VALUE	UNVAILABLE VALUE	DESCRIPTION
smallint unsigned	varchar[64]	tinyint unsigned	tinyint unsigned	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)	varchar [128]

Var_id: variable identifier [2 bytes]
Datatype_id: variable type [1 byte]
Unitcode_id: unit code (see below) [1 byte]
Min_value: minimum allowed value (big-endian)*
Max_value: maximum allowed value (big-endian)*
Unv Value: invalid value (big-endian)*
Description: ASCII string, reporting variable textual description. [128 bytes].

*Actual significant byte number, for a given *Datatype_id* is given in the *Datatype.Length* field of *DataType Table*. NULL if meaningless.

UNIT_TABLE (STATIC)

List of units of measure.

UNIT_ID	SI_NAME	EXPONENT
smallint unsigned	varchar [128]	smallInt

Unit_id: unique ID
SI_name: name of the unit, according to the SI International System of Units
Exponent: scale (10^X) factor.

Examples
[1, Volt, 3]: kV unit
[2, Ampere, -3]: mA unit

HOST_OBJECTS_TABLE (STATIC)

Associates objects to gateways/hosts.

HOST_ID	OBJ_ID

Many objects can be associated to the same host.

OBJECT_VARIABLES_TABLE (STATIC)

Associates variables to objects.

OBJ_ID	VAR_ID

Many variable can be associated to the same object.

BUILDING_TABLE (STATIC)

List of buildings.

BUILDING_ID	BUILDING	FLAT	GEO_COORDINATES
smallint unsigned	varchar[256]	varchar[256]	varchar [128]

Building_id: unique identifier
BUILDING: building ASCII text description
FLAT: flat ASCII text description. NULL if meaningless
GeoCoordinates: building localization (long/lat) ASCII text description.

LOCALIZATION_TABLE (STATIC)

List available localizations within a user's flat.

LOC_ID	BUILDING_ID	ROOM	ZONE	USER_ID
Usmallint	Usmallint	varchar[256]	varchar[256]	Usmallint

Loc_id: unique ID
Building_id: building identifier
ROOM: room ASCII text description. NULL if meaningless
ZONE: zone ASCII text description.
User_id: User ID, if wearable device.

OBJECTS_LOCALIZATION_TABLE (STATIC)

Associates objects to localization.

LOC_ID	OBJ_ID

Many objects can be associated to the same localization.

DATATYPE_TABLE (STATIC)

List available datatypes.

DATA_CODE	NAME	DATALEN	MIN_VALUE	MAX_VALUE
tinyint unsigned	varchar[64]	smallint unsigned	varchar[64] (nibble ASCII)	varchar[64] (nibble ASCII)

Examples:

01	BOOLEAN	1	FALSE	TRUE
02	UCHAR	1	0	255
03	CHAR	1	-128	127
04	UINT2	2	0	65,535
05	INT2	2	−32,768	32,767
06	UINT4	4	0	4,294,967,295
07	INT4	4	−2,147,483,648	2,147,483,647
08	UINT8	8	0	18,446,744,073,709,551,615
09	INT8	8	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
10	FLOAT	4	3.4E - 38	3.4E + 38
11	DOUBLE	8	1.7E - 308	1.7E + 308
12	TIMESTAMP	6	NULL	NULL
13	STRING	255	NULL	NULL
14	BLOB	8	NULL	NULL

RESULT_CODE_TABLE (STATIC)

Lists result codes and their meaning, to be adopted by services interacting with the database..

RESULT_CODE	DESCRIPTION

tinyint unsigned	varchar[128]

Example:

00	NO_ERROR
01	DB_ERROR
02	…

STATUS_CODE_TABLE (STATIC)

Lists status codes exploited in Status_table (see below).

STATUS_CODE	DESCRIPTION
tinyint unsigned	varchar[128]

Examples:

00	OPERATIVE_ONLINE
01	OFFLINE
02	UNCONFIGURED
03	OUT_OF_SERVICE
04	DEBUG_INFO
05	…

FAULT_CODE_TABLE (STATIC)

Lists fault codes exploited in Status_table (see below).

FAULT_CODE_ID	DESCRIPTION
tinyint unsigned	varchar[128]

Examples:

00	NONE
01	MEASURE_WRONG_READING
02	…

HOST_CMD_TABLE (STATIC)

For each host, a list of “command” codes and their description.

HOST_ID	CMD_CODE	PARAMETER_BYTES_LENGTH	DESCRIPTION
smallint unsigned	tinyint unsigned	smallint unsigned	varchar[128]

DATA_TABLE

It is the main system table, collecting information coming from all gateways and related to all objects.

RECORD_ ID	RECORD_ TIMESTAMP	HOST_ID	OBJ_ID	VAR_ID	USER_ID	TIMESTAMP	DATA	INT_ VALUE	REAL_ VALUE
Uint	datetime	smallint unsigned	smallint unsigned	smallint unsigned	smallint unsigned	datetime	varchar [256] (nibble ASCII)	bigint	double

record_id unique ID
record_Timestamp: timestamp (related to database access)
host_id: gateway identifier
obj_id: object identifier
var_id: variable identifier
user_id: user identifier (if any, NULL otherwise)
timestamp: timestamp (associated to data coming from gateway, if any; NULL otherwise)
data: ASCII data body (big-endian)
int_value integer data coding (if applicable)
real_value floating (double) data coding (if applicable)
user_id user_id (if any), referring to USER_TABLE.

USER_TABLE (STATIC)

Lists participating persons.

USER_ID	USER_PERS_DATA_ID	SURNAME	NAME
Uint	smallint unsigned	Varchar(128)	Varchar(128)

User_id: unique ID
USER_PERS_DATA_Id: external reference to User personal data registry (outside this database architecture)
Surname: user's surname
Name: user's name.

BLOB_TABLE

This table is provided for generic data not fitting any of the available data types in DATA_TABLE (e.g., larger than 128 bytes or 256 characters). Semantics of such data is not defined, though, and its management is left to external services.

BLOB_ID	DATA
smallint unsigned	Blob[65535 bytes]

Blob_id: Unique ID
Data: free-format data.

STATUS_TABLE

Keeps an up-to-date list of status information, regarding all system objects.

RECORD TIMESTAMP	HOST_ID	OBJ_ID	STATUS TIMESTAMP	STATUS_CODE	FAULT_CODE	FREETEXT
datetime	Usmallint	Usmallint	datetime	tinyint unsigned	tinyint unsigned	varchar[128]

Record_Timestamp: timestamp (related to database access)
Host_id: gateway identifier
Obj_id: object identifier
Status_ts: timestamp (associated to data coming from gateway, if any; NULL otherwise)
Status_code: status code (see STATUS_CODE_TABLE above)
Fault_code: fault code (see FAULT_CODE_TABLE above)
Freetext: free text description (if any).

CMD_TABLE

This table allows for interaction among different gateways, through the database itself. Whenever a given gateway needs a different gateway to carry out some task, it places a request by adding a new record to the CMD_TABLE.

CMD ID	RECORD TS	HOST_ID SOURCE	HOST_ID DEST	OBJ ID	VAR ID	CMD CODE	PARAMS LEN	PARAMS	DONE FLAG	DONE TS	DONE HOST_ID
int	datetime	smallint unsigned	smallint unsigned	smallint unsigned	smallint unsigned	tinyint unsigned	smallint unsigned	varchar[512] nibble ASCII	tinyint unsigned	datetime	smallint unsigned

Cmd_ID: unique ID
Record_Timestamp: timestamp (related to database access)
Host_id_source: source gateway ID
Host_id_dest: destination gateway ID
Obj_id: object ID
Var_id: variable ID
Cmd_code: command code, as of HOST_CMD_TABLE table
Parameters_len: parameter string actual length
Parameters: command parameters body
Done_flag: flag, set if the command was executed (either with success or failure)
Done_Ts: timestamp (related to command execution)
Done_host_id: gateway “signature” (ID of the gateway in charge of command execution).

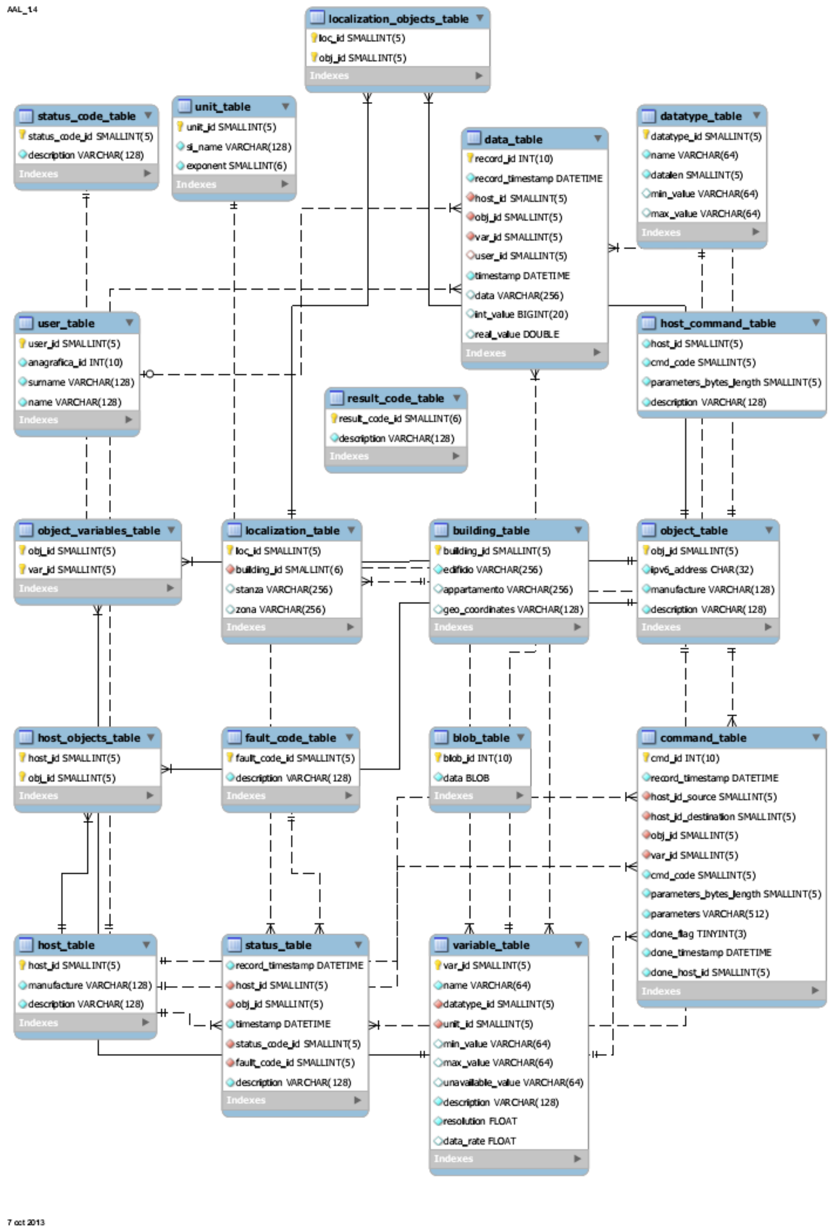


Fig. A.1: summary view of the HELICOPTER database structure