iHorse – A WSN-based Equine Monitoring System

André Rodrigues, Jorge Sá Silva, Fernando Boavida
Centre for Informatics and Systems
University of Coimbra
Coimbra, Portugal
{arod, sasilva, boavida}@dei.uc.pt

Abstract— Horses played a key role in the history of mankind, having been instrumental in the rise and fall of great empires. In modern days, they are still very important assets, from economical, operational or even sentimental points of view. Their importance is, thus, the main motivation for developing the iHorse equine monitoring system presented in this paper. iHorse is a deployed, fully-operational, commercially-available, wireless-sensor-network-based system and application, developed and tested over a three-year period. The aim of this paper is to provide information on the main features of the system, including its architecture, implementation details and performance data. The paper also discusses the lessons learnt throughout the system development and deployment phases.

Index Terms - wireless sensor networks; sensor-based applications; equine monitoring; practical experience

I. INTRODUCTION

Horse monitoring is essential for a number of reasons, including the fact that horses are prone to sudden illness, such as colic [1], which may cause death. Being able to remotely and constantly monitor horses can, thus, be very important, in order to be aware of their condition and act as quickly as possible in case of need. In this task, wireless sensor networks (WSNs) can play a decisive role, allowing owners and/or breeders to easily monitor horses’ vital signs and anxiety, or even the stables environmental conditions influencing the horses’ comfort.

The utilization of WSNs technology applied to farm animal monitoring has received some attention from the research community. In [2] the focus is on monitoring the behavioural preferences of cattle, by using a combination of GPS, satellite imagery and WSNs. Monitoring cows presence and pasture time on a strip of new grass using the Zigbee technology was addressed in [3]. In the case of horses, wireless sensor technology has been used to analyse gait (in order to identify lameness [4]) and in detecting foaling time [5], [6].

This paper describes iHorse, a WSN-based system that provides a solution for equine online continuous monitoring, allowing users to access and evaluate vital signs, as well as behavioural and environment parameters, anytime and from anywhere, as long as there is an Internet connection. In addition to monitoring, the system comprises registering and alerting functionality, as well as remote video streaming.

The aim of the paper is to provide an insight on the various issues involved in the development of a finished product WSN-based system and application. These comprise traditional engineering aspects, such as requirements, functionality, hardware architecture, software architecture, implementation options and performance, but also a discussion on the lessons learnt during the whole process.

In order to achieve the stated goal, the paper is organised as follows. Section 2 provides a functional system overview, focusing on its main, distinguishing features. Section 3 describes the system architecture from the hardware and software perspectives. The most relevant implementation details are presented in section 4. The considerably long development time – typical of new technologies – allowed us to gather detailed performance data, presented in section 5, and to learn several lessons that we discuss in section 6. Section 7 provides the conclusions and guidelines for further work.

II. iHORSE OVERVIEW

iHorse is organized around the concept of stable, which contains several individual boxes or stalls where horses are kept indoors. The system provides functions to monitor individual horses and stalls, encompassing data collection and storage, alarm management, remote access, several user profiles, and auto-configuration. These are briefly presented below.

A. Data Collection and Storage

All the information gathered by iHorse can be visualized in real-time or saved for later analysis.

Having the capability to view an animal as well as measuring heartbeat and movement levels is important to evaluate the animal’s condition and to help in screening any anomalous situation. iHorse real-time functionality displays the information gathered from the horse (e.g., video, movement index and heartbeat), from the environment (e.g., temperature and humidity), and from the system (e.g., battery level of each monitoring equipment) through a user-friendly interface.

In addition to real-time monitoring, the system has the ability to store the gathered data and video in the system database. This is a key feature, allowing future analysis of monitored data in order to identify patterns of animal behaviour or to assist veterinarian diagnosis.

B. Alarm Management

Alarm management allows one or more parameters to be
under observation. Each alarm is configured by setting thresholds for one or more parameters. If an alarm condition occurs, the user is alerted via an SMS message sent to a predefined mobile phone, or by email. The system allows the user to concurrently set up alarms for all of the deployed sensors.

Alarms can be configured via a user-friendly interface, which allows the user to specify thresholds for each parameter. In order to meet requirements from equine professionals, it was necessary to implement an alarm configuration tool (Fig. 1) oriented to narrowing down specific problems related to the horses’ behaviour. Examples of such requirements are given below.

- For a resting horse, heartbeat should not be too low (i.e., less than 20 bpm) or too high (i.e., higher than 80 bpm). Also, values such as 60 bpm if sustained for more than 5 min can indicate some kind of pain. These cases justify informing the owner and/or responsible person.

- There are situations in which an increase in heartbeat does not represent a problem; an example occurs at the beginning of the day when the stables are open and the horses are fed, which leads to high excitation levels; careful selection of the alarm schedule can cope with this.

- Low temperatures can affect the animals’ health; high temperatures can indicate something wrong like a starting or on-going fire.

- When data from a horse’s monitoring equipment fails to reach the monitoring station, this may indicate a problem needing some attention. For example, the horse may have been moved out of range. We enable this as an alarm source and call it Horse Presence.

- Equipment battery condition is critical for correct system operation. Because of this, the system always displays the battery levels in an easy-to-understand way, using green, yellow, and red codes, giving the user the option to be notified on colour transitions.

C. Remote Access

The iHorse system is accessible through a portal that performs user authentication and implements the calls to the user profiles module. The system can be accessed any time, from any location where an Internet connection is available, and using a variety of platforms, such as a personal computer or a smartphone. Interfaces optimised for each type of user device have been developed.

D. User Profiles

The system supports different user profiles, such as stable owner, horse owner, system administrator, and visitor. The first two types of users are allowed to access the information on horses and/or respective boxes, as well as to interact with the data and alarm management functionality. The system administrator has access to all the information and is responsible for system configuration. Visitors can only access public information, such as general information on the facilities, publicity, contacts and location.

E. Auto-configuration

One of the main design options of iHorse was to make the system as easy to use as possible. In this line, auto-configuration capabilities were developed, allowing the automatic addition of new devices to the system. This feature is supported on a discovery process, within the sensor network, which detects new devices and informs the central unit. The new device will then need to register itself, announcing its own service capabilities. Communication between the devices and the central unit is done using a star topology.

III. System Architecture

iHorse was built using a scalable and modular architecture that takes into account the hardware constraints of WSNs, such as radio range / bandwidth, processing capability, available
memory, and energy budget. The main hardware and software architectural aspects are presented below.

A. Hardware Architecture

iHorse relies on a set of sensors that communicate with a central point, responsible for data gathering and consolidation.

Taking into account the typical physical configuration of a horse stable, a star topology was chosen for the WSN. Fig. 2 presents a high-level view of the iHorse hardware, in which a central unit, named Monitoring Unit (MU), works as a gateway between the sensors and the users.

![Figure 2. iHorse hardware architecture](image)

In the stable, each stall is equipped with a customized sensor set that can incorporate, among others, an infrared video camera, environment sensors aggregated in a sensor node named Ambient Station, and specific equine sensors attached to the Horse Station. The Ambient Station typically includes temperature and humidity sensors. The Horse Station is normally equipped with heartbeat and movement sensors. Each sensor node, capable of monitoring several parameters, directly communicates with the central point, a Base Station physically plugged to the MU.

The decision to use a star topology, as opposed to a mesh topology, was taken due to the characteristics of stables: small locations with few RF interfering sources. These characteristics allow sensor nodes to be able to communicate with the MU by using a single hop in the majority of cases. In cases where a one-hop communication is not possible due to range limitations, bridge devices are used for range extension.

The Horse Station (i.e., the device actually deployed on the horse) is based on a customized sensor board. Due to its modularity features, the sensor board can integrate a variety of sensors, such as accelerometers, heartbeat (via a receiver that captures the heartbeat signal sent from OEM heartbeat belts) or blood pressure. The only limitation on incorporating more types of sensors in this device is size, as big devices can impact the horses’ well being.

The Ambient Station is a more robust device, also based on the TMoteSky architecture, with an embedded Sensirion sht11 sensor capable of monitoring temperature and humidity, and is normally attached to the box wall. Similar to the Horse Station, this device can also be upgraded to incorporate additional sensing capability, such as infrared movement detection, box door violation, water consumption, floor moisture, and ammonia, among other. The constraints on this sensor node are quite less restrictive when compared to the Horse Station, mainly because size requirements are more relaxed.

A MU must be deployed in each stable. The MU is the central unit in the system, implementing all the user and administration functionality, supporting the communication with the sensor nodes, processing the video streams from the cameras, and interfacing the system with the Internet and cellular networks.

B. Software Architecture

In what concerns software, iHorse functionality is supported on a set of modules executing at the MU and on the sensor nodes. This section starts by briefly describing the MU modules functionality and how they collaborate to provide the iHorse functionality. The section ends with a presentation of the sensor nodes software workflow.

The MU software architecture (Fig. 3) is composed of several modules, namely, portal, video server, kernel, WSN access, and alarm processing. Each of these is briefly described below.

**Portal** – this module implements the user interface, providing authentication and access to the iHorse functionality. The user interface is dynamically generated, according to each user’s profile and to the system state. The adaptation of the user interface to the user’s profile and device is extremely important in order to optimise performance, resources and user experience, even when low bandwidth connections are being used.

![Figure 3. Monitoring Unit software architecture](image)
The Portal module accesses the system database in order to:

- retrieve user interface configuration data (e.g. names of horses / boxes);
- obtain historic data to be displayed or exported;
- get sensor configuration data (e.g. sensor names, ranges, sampling periods);
- store information required for configuration of the monitoring schedule and alarm notification functionality.

Real-time visualization requires the Portal module to access the Video Server module, as well as to access parameter values, the latter by querying the Kernel module.

**Video Server** – the system supports diverse types of cameras, analogue CCTV or digital (either Ethernet or WIFI). Each video stream is processed by a Video Server module and is made available to the users in MMS, MPEG, or MJPEG format, depending on the user device. The Portal module automatically chooses and configures the appropriate video plug-in, according to the user platform and the available video streams.

**Kernel** – this is the central module of the MU, being the coordinator of all the modules. The Kernel module:

- receives sensor queries from the Portal module and forwards them to the WSN via the WSN Access module;
- supports the monitor scheduling functionality, by starting/stopping each monitor and storing the received sensor data in the system database;
- supports the alarm notification mechanism, by sending the respective configuration (received from the Portal) to the sensor network, and by receiving raw alarm data and forwarding it to the Alarm Processing module.

**WSN Access** – this module comprises the TinyOS serial forwarder, used to interface with the sensor network, as well as a watchdog component developed to check and report any communication problem between the MU and the Base Station.

**Alarm processing** – this module implements the algorithms that process the raw data received from the WSN, in order to detect conditions that trigger the alarms. It also implements robustness mechanisms to cope with temporary unavailability of the cellular network, e.g. by queueing the messages for sending at a later point in time, and by emailing the administrator in case of non-transient failures. In addition to sending messages via a cellular network, the system can be configured to send SMSs messages via an Internet gateway.

Horse Stations and Ambient Stations share the same architecture, whose workflow is illustrated in Fig. 4. Basically the software is organized around the three iHorse main functions: alarm notification, real-time reporting, and scheduled monitoring.

![Figure 4. Simplified sensor node workflow](image)

**IV. IMPLEMENTATION**

This section presents the main decisions concerning the platform and the application implementation aspects, with the objective of detailing the system construction and justifying the main choices.

**A. Platform**

The MU hardware is based on a compact embedded PC with several hardware interfaces, either native or attached via USB, that enable communications using the following technologies: 3G, WIFI, Cable/ADSL, and 802.14.5. The system supports several video camera options, namely analogue and digital cameras, either wired or wireless. Analogue cameras are supported via the PC embedded video processing module, which provides 4 video inputs. The system runs on Linux, making use of additional software packages: Lighttpd, PHP, MySQL (also supporting PostgreSQL), VLC server, JRE6, Gnokii (optional support for sending SMSs messages from an attached mobile phone [7]), and Nagios, the latter for supporting remote monitoring of key system aspects.

On the sensor node side it was decided to use TinyOS since, at the date the development took place, it represented the most common operating system for WSNs, followed by Contiki. At that time, various versions of TinyOS were available: TinyOS 1.x, TinyOS 2 (T2), and the Sentilla (former MotelV) version called Boomerang [8]. The decision to use T2 was motivated by the conviction that soon it would be the most widely supported version, which eventually happened.
In the first version, the selected hardware was the TMoteSky platform due to the fact that it was one of the TinyOS standard platforms, it was FCC pre-certified, it had an acceptable cost and it had the ability to meet the core system requirements.

Because the TMoteSky hardware did not support movement sensing and heartbeat detection, an add-on board containing a 3-axis accelerometer (Freescale MMA7260) and a heartbeat receiver (Polar RTCM01 operating at 5.5 KHz) was developed. This was installed in a water-resistant box to be further attached to a specifically developed horse belt. The horse belt also included a heartbeat transmitter responsible for collecting and analysing heart signals, and transmitting them to the paired heartbeat receiver in the add-on board.

The TMoteSky platform became commercially unavailable during the iHorse production and, as a result, a solution based on the IRIS mote was designed. Concurrently, in order to avoid further surprises, a decision was made to develop our own mote, which we named Pegasus, with the aim of reducing the dependence from third parties in the fast moving area of WSN node hardware.

The IRIS mote leads to an increase in link budget of 9dB, which roughly triples the range of TMoteSky in free space propagation. Table 1 shows that 6dB come from an increase in receiver sensitivity and 3dB come from extra TX power.

<table>
<thead>
<tr>
<th>Transceiver vendor</th>
<th>TMoteSky</th>
<th>IRIS</th>
<th>Pegasus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chipcon</td>
<td>CC2420</td>
<td>RF230</td>
<td>CC1101</td>
</tr>
<tr>
<td>CC2420</td>
<td>2.4GHz</td>
<td>2.4GHz</td>
<td>315/433/868/915</td>
</tr>
<tr>
<td>Atmel</td>
<td>2003</td>
<td>2006</td>
<td>500</td>
</tr>
<tr>
<td>Operating freq. (MHz)</td>
<td>2.4GHz</td>
<td>2.4GHz</td>
<td>315/433/868/915</td>
</tr>
<tr>
<td>Max data rate (kbps)</td>
<td>250</td>
<td>250</td>
<td>500</td>
</tr>
<tr>
<td>RX power (mA)</td>
<td>18.8</td>
<td>15.5</td>
<td>16.9 (868MHz)</td>
</tr>
<tr>
<td>TX power (mA/Dbm)</td>
<td>17.4 / 0</td>
<td>16.5 / 3</td>
<td>34.2 / 12</td>
</tr>
<tr>
<td>Rec. sensitivity (dBm)</td>
<td>-95</td>
<td>-101</td>
<td>-96 (at 250kbps)</td>
</tr>
<tr>
<td>Power down. (µA)</td>
<td>1</td>
<td>0.02</td>
<td>0.2</td>
</tr>
<tr>
<td>Turn on time (ms)</td>
<td>0.58</td>
<td>&lt;1</td>
<td>0.24</td>
</tr>
<tr>
<td>Modulation</td>
<td>DSSS-O-</td>
<td>DSSS-O-</td>
<td>OOK/FSK/GFSK/MSK/ASK</td>
</tr>
<tr>
<td></td>
<td>QPSK</td>
<td>QPSK</td>
<td></td>
</tr>
</tbody>
</table>

For the new platform (Pegasus) we established the following requirements:

- to be easily ported to T2;
- to have a more recent processor;
- to increase the communication range;
- to support an SD card;
- to include all the components on one PCB;
- to support a rechargeable LiPoly battery;
- to be compact and light.

The first requirement had the objective of reducing the system development time, by reusing the available T2 source code as much as possible.

Regarding the processor, MSP430F2618 was used, as it is considered an interesting substitute for the MSP430F1611 [9], having more program memory and two USCI modules, each one supporting two independent serial communication channels.

The communication range is an important factor, since it is necessary to cover as much as possible of the stables area without deploying bridge nodes. By selecting the CC1101 transceiver, it is possible to improve the link budget by 13dB when compared to the CC2420 transceiver (thus increasing the range by a factor of 4), and because the nodes operate in the 868MHz frequency we can expect another 2 times increase in the range due to the lower communication frequency factor. Curiously, the same CC1101 transceiver, integrated in the TI CC430 SoC is used in the OSIAN [11] WSN technology, for use in commercial and residential places, in order to maximise range.

Sampling of the movement sensors is done at 100 Hz. A total of 12 bytes of data are generated per sample, comprising accelerometer and a gyroscope data. With these values, it takes around 15 minutes to fill the TMoteSky 1024KB flash. Taking into account that there may be relatively long periods of time during which transmission may not be possible (for instance, when the horse is lying down and the node is pressed against the ground) it is desirable to have enough memory to store readings. To cope with this, an SD memory card was included in the platform, with room for large amounts of raw accelerometer / gyroscope data without the risk of missing samples.

Having all the components on one PCB and using a compact battery was key to having small and light devices, thus minimising the negative impact on the horses’ well being.

As a result of the use of TinyOS and of the multi-platform strategy, the current iHorse system is highly flexible. The application source code is platform-independent because hardware sensor differences are hidden by the drivers. On the other hand, the system can run on any of the three developed platforms.

B. Application

1) Traffic optimisation and information displaying

The system allows the user to view the horses in real time, side-by-side with information on the horses’ parameters updated every 2 seconds and displayed as instantaneous values and/or graphically over the last 60 seconds. The first implementation approach required a query to the sensor network for each parameter update. Such approach led to high energy and bandwidth consumption, and did not scale. To address these problems the following improvements were made:

- **traffic aggregation** – one query for all node parameters instead of querying the parameters individually;
- **query number reduction** – instead of querying a node each time the parameters need to be updated, the Kernel module asks the node to send the parameters during a time window (e.g., 4 minutes). At a
value caching – parameter values from the sensor network are saved with a timestamp in a cache inside the Kernel module. If a request for a parameter value is received from a user, the cached information is returned without querying the sensor network if the timestamp information assures it is still valid.

All these optimizations enable to have most of the traffic from the nodes to the sink. Also, this traffic is not dependent on the number of system users accessing the parameters of a specific horse, being approximately 0.5 packets per second for each visualized horse. This represents an important improvement in relation to the initial version, in which the traffic per box was 2n packets per second, with n being the number of users visualizing the same horse.

2) Alarm management

Initial alarm management relied almost entirely on sensor node processing. After the specification of the valid range by the user, this information was sent to the sensor, which had the responsibility of generating an alarm message to the MU whenever the parameter values were outside the specified values for a specific period of time. After discussions with horse professionals, a more flexible and powerful alarm management system was designed (illustrated in Fig. 1) which required more processing. In order to limit sensor code modifications, allow greater flexibility in rule definition, and not to put too much processing load on sensors, only simple alarm rules are set in the sensors (e.g., in Fig. 1, for the heartbeat parameter, a range of [20;60]), and all the remaining processing (e.g., time of the day the alarm occurred, duration of a rule violation) is done at the MU. This approach greatly simplifies processing at sensor nodes, without wasting too much energy in communications, since alarms represent low traffic.

3) Energy management

Each iHorse monitoring node executes a TinyOS based application (see Fig. 4 for details). This application makes use of TinyOS communication capabilities for enabling high link reliability, via the software ACK mechanisms offered by the Packet Link Layer module, and energy saving through radio duty cycling, using Low Power Listening. To save energy, sensors are kept in a low power, sleep mode between readings. Note that, for instance, an accelerometer in active mode consumes 500uA, and in sleep mode just 3uA. On the other hand, heartbeat receivers do not support sleep mode, but only consume 60uA.

4) Horse Station parameters

The Horse Station is responsible for acquiring horse heartbeat and movement index data. In the following, some information is provided on how each of these parameters is calculated.

• Movement index – the accelerometer is configured for measuring accelerations in the -2g to 2g range. This range enables capturing small acceleration values. Each accelerometer axis is sampled at 100Hz and band-pass filtered (in the 1Hz – 20Hz band) in order to remove noise and to discount for the static acceleration components. This frequency band contains the acceleration components generated by usual movements while horses are resting. The sum of the absolute values of the 3-axis is calculated for the samples contained in a non-overlapping 2 seconds window. The value is then normalized to fit a 0 to 10 scale. This parameter was validated by observing horse movement and comparing it to the indicator response.

• Heartbeat – the heartbeat receiver generates a signal for each beat detected by the transmitter attached to the horse. We calculate the heartbeat considering the last 8 received beats. There are a few sources of error that should be accounted for in the calculation of the average heartbeat, some examples being communication losses between the sensor and the heartbeat receiver, false beats generated by horse movement, and arrhythmia. The veterinarians that collaborated in the system development validated the readings by comparing them with manual readings.

V. PERFORMANCE

Energy consumption is a key factor for WSNs, although in the current scenario it is not as critical as in deployments where it is difficult to access nodes to change batteries. However, frequent battery changes at Horse Stations and/or Ambient Stations should be avoided. As mentioned before, reduction of energy consumption was one of the major concerns in designing the system, and was achieved by traffic load optimisations and T2 energy saving mechanisms.

In the TmoteSky version of the iHorse node hardware, energy is supplied by a pair of AA alkaline batteries that are changed on a need basis, i.e., at 2.2 V for both cells. This is a value that does not compromise the proper functioning of hardware components, but wastes approximately 10% of the battery capacity.

During the last two years battery duration was registered for the iHorse main demo / test deployment. During this period the system was used for various purposes and by diverse users, for demos at shows, for testing by candidate users, and for running experiments. This diversified use pattern enabled the collection of statistics on battery usage. The average value for the Horse Station battery duration is 34 days with a standard deviation of 10 days. This high standard deviation is justified by the occurrence of periods of light activity (e.g., accelerometer sampling at 100Hz to calculate the movement index, and a node reporting mechanism that sends one packet per hour) and others of heavy system use for test purposes (e.g., previously mentioned mechanisms, plus active monitoring generating 0.2 packets per second, and parameter updates for real-time visualization generating 0.5 packets per second a few hours per day).

Typical use patterns enable the Horse Station batteries to last for 1 month. On the other hand, the available data on the Ambient Station points to an average battery duration of 3 months.
We have also performed communication range measurements. Indoor communication range is impacted by several factors, such as propagation and interference problems [10]. The graph in Fig. 5 shows the communication range measured for the 3 hardware platforms supported by iHorse. The measurements were done outdoor, on an open field with vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m measurements were done outdoor, on an open field with vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height accounts for a standing horse and the 0.25m vegetation less than 0.30m high, and inside a stable. The 1.50m node height applies when the horse is lying down. The following measurement conditions apply:

- communication ranges were measured with an application that queries a remote node one time per second; based on this, several parameters were calculated, namely percentage of queries that were answered, RSSI and LQI for both nodes;
- the distances in the graph are for links in which the percentage of successful queries was more than 95%; lower quality links reduce the battery duration because of packet retransmissions;
- the Base Station was deployed at a height of 2m;
- the distances inside the stable are for the worst locations, i.e., the ones that had the highest number of concrete box walls between the Horse Station and the Base Station;
- nodes were static, in a horizontal position, and used the original antennas (for IRIS and TmoteSky);
- the maximum distance between the Base Station and the Horse Station located in the most distant box was 55m, corresponding to 18 boxes between transmitter and receiver.

![Figure 5. iHorse platforms communication ranges](image)

Considering the presented conditions (selected considering the horse monitoring scenario) only the new Pegasus node was able to communicate in one hop with a node in any of the stable boxes. This is not a major limitation if only a fraction of the horses are being monitored or if bridges are used. Nevertheless, it justifies the use of nodes with as large a range as possible. Another reason for having nodes with increased range, not apparent in Fig. 5, is the fact that the measurements were done with the nodes on a horizontal position and, as we have learned before [12], other orientations may lead to shorter range. Only the Pegasus node was able to assure one hop communication irrespectively of the node orientation as, in fact, the 55m limitation comes from the stable dimension and not from propagation or communication path issues.

VI. LESSONS LEARNT

This section summarises the main lessons learnt during the development and testing of the iHorse system.

A. Real world requirements

Research requirements are clearly different from the requirements of finished products. It is of paramount importance to talk with field professionals to clearly identify real world requirements. This is what happened with the development of all of the system functions, with emphasis on the alarm generation sub-system. In addition, real world requirements nearly always lead to simple architectures.

B. Management Unit hardware

The quality of the MU hardware is crucial for continuous system operation. Stables potentiate several aggression sources to the equipment: temperature (during summer inside temperatures can be high), dust (the system is deployed on a shelf 1 meter below the ceiling), and animals (mice, flies and spiders). Equipment must be protected from these adverse conditions. After a few experiences with equipment that stopped working because of environmental aggression, we choose to use an embedded industrial PC.

C. Video cameras

Night vision (0 lux) camera image quality is a very important issue, so that the system can be fully used 24 hours a day. We found that low cost Ethernet / WIFI cameras do not have the necessary infrared projectors to provide an adequate image quality during nighttime. We also found that in what concerns night vision image quality, analogue CCTV cameras offer decent quality at a fraction of the cost of equivalent digital cameras. Still, digital WIFI cameras are a good option when wireless communication is required, not only due to signal propagation reasons but also due to scaling reasons, as it is possible to support a higher number of digital cameras.

Another aspect that we discovered was premature infrared LED death. In standard cameras, infrared LEDs lifetime is between 10,000 to 20,000 hours. We found out some cameras for which infrared LEDs died before 2 years of daily usage.

Another aspect concerning cameras is the need for periodic cleaning due to insects, without which night image quality quickly degrades.

D. Sensor node hardware

Using OEM node hardware reduces development time, cost and required expertise, as we discovered when using the TmoteSky and IRIS nodes. This is a viable option if the hardware fits the requirements, but creates dependence on the OEM (usually a small company). Our final approach was to...
platform using an available OEM pre-certified transceiver.

Another interesting approach would be to develop a custom platform using an available OEM pre-certified transceiver module, or even a module containing the microcontroller and the transceiver, either as a SoC or as discrete components. The availability of these modules by major suppliers will greatly reduce the risk, time, expertise, and cost in developing new platforms. In [9] the authors describe a set of requirements these modules should support in order to simplify custom platform development.

E. Sensor node software

Development using TinyOS 2 requires a relatively long learning curve, but the supported functionality in terms of energy management and communications, in conjunction with the stability of the programming interfaces, clearly justify the additional work at the beginning. One aspect that could be improved would be to offer a mature integrated development environment.

F. Performance management

The iHorse system is extremely robust, with our oldest deployments already being in continuous operation for many months. However, in the first versions we experienced sporadic problems, normally related to the video subsystem. As a result, in the current version we monitor several aspects of the system operation (CPU consumption, disk space, Internet access state, VLC processor consumption, Base Station behaviour) and generate alerts in case there is any fault. This enables us to immediately take corrective actions.

G. Vertical knowledge

Due to their very large potential applicability, WSNs present a set of characteristics that force system developers to take into account, from an early stage, the vertical knowledge of the specific application domain. This means that it is important to understand the solution not only from the point of view of the technology to be used, but also from the point of view of the application itself. A practical example from iHorse was the development of the horse belt, for which multiple versions were made in order to meet requirements according to different perspectives: horse owners (durability, ease of installation), horse veterinaries (horse stress, horse health), horse (light, adapted), and manufacturer (cost, time to manufacture, impact on the quality of the sensed heartbeat and movement signals).

VII. Conclusion

Although considerable research has been and is being done on WSNs, real world applications are still not common. In this paper we have presented and discussed several aspects of a deployed WSN-based horse monitoring system, in what concerns main functionality, architecture, implementation and evaluation. The presented system was developed and tuned over a three-year period, which allowed us to deal with and overcome many real-world obstacles, the most relevant ones having been addressed in the paper.

Despite the fact that the system is now being commercially explored, development work will continue, with the aim of integrating additional diagnostic features and extending its application to foaling detection.

ACKNOWLEDGMENT

We would like to thank the team that collaborated with us in the iHorse development, by naming the two persons that were most involved: Ricardo Ruivo and Miguel Silva. The iHorse demo / test deployment is located in a stable that belongs to Escola Superior Agrária de Coimbra. We thank their continuous support in providing access to the facilities and in giving advice on system requirements. Finally we also thank Envispot, the startup that continues the iHorse development and commercialisation, as well as PDMFC, which does the same for the Pegasus node, for their support in this work.

REFERENCES