[RKHS08] *Andreas Reinhardt, Matthias Kropff, Matthias Hollick, Ralf Steinmetz;* **Designing a Sensor Network Testbed for Smart Heterogeneous Applications.** In: Proceedings of the Third IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2008), Montreal, Canada, October 2008. Seite

Designing a Sensor Network Testbed for Smart Heterogeneous Applications

Andreas Reinhardt, Matthias Kropff, Matthias Hollick, Ralf Steinmetz
Multimedia Communications Lab
Technische Universität Darmstadt
Merckstr. 25, 64283 Darmstadt, Germany
Email: {andreas.reinhardt, matthias.kropff, matthias.hollick, ralf.steinmetz}@kom.tu-darmstadt.de

***Abstract*—Future buildings and environments are envisioned to provide ambient intelligence, adapting to a user's preferences based on information about his context and status. Smart heterogeneous sensor networks are well suited data sources for such environments, because they allow for dynamic adaptation to newly added sensor types and novel tasks. Realistic verification of protocols and algorithms for smart networks poses special constraints on testbeds, necessitating support for heterogeneous platforms and mobility in the network. These distinct requirements limit the usability of many known testbeds of purely homogeneous nature.**

In this paper, we determine a minimum set of premises for smart heterogeneous sensor network testbeds and evaluate existing architectures with respect to these requirements. We then present our *tubicle* node platform, an integrated sensor network node providing inherent support for heterogeneity and fulfilling the determined set of requirements in their entirety. A set of twenty tubicles forms the basis for our *TWiNS.KOM* testbed. Specifically designed for heterogeneity, the architecture allows rapid validation of smart sensor network algorithms and quick experimental setup.

I. Introduction

Enriching environments with sensors and mechanisms to infer contextual data is mandatory in ambient intelligence application scenarios. These sensors can be implemented as pure software suppliers, integrating with a user's devices and tracking his preferences and behavior. If physical events are to be monitored, sensors can also be present in terms of dedicated hardware devices. An especially well-suited approach for such scenarios is the use of wireless sensor networks (*WSNs*), comprising devices (*motes*) that combine sensing, computing, and radio transmission capabilities [1]. WSN applications generally exploit the distributed characteristics of a sensor network, and thus implicitly make use of distributed sensing, processing, and multi-hop communications. In many applications, external sinks are targets for collected data, acting as receivers for sensor readings, and offering the received packets to middleware layers or directly to requesters over different communication protocols.

In general, context-aware applications require a high diversity of supplied information, strongly implying the need for heterogeneous sensors. In contrast to this heterogeneity on the sensor type level, applications often need to extract relevant data from a set of samples, creating a demand for higher computational capabilities [2], [3]. Sensor networks used as suppliers to context-aware applications are often required to be heterogeneous in many dimensions, with the most prominent ones being computational power, radio interfaces, and the types of employed sensors.

WSNs still pose interesting research topics, such as efficient time synchronization, localization, or mobility support [4]. While many algorithms were evaluated in theory and simulation, experiences on real WSN hardware did not always confirm the results [5], as realistic radio and sensor behavior often mismatched the idealized models utilized in the simulator. To address this problem, a multitude of testbed architectures and real-world network deployments have been proposed and implemented. Differing in their used radio transmission protocols, platform types and device count, the setups provide the capability to perform real test runs. However, many of them are limited by their purely homogeneous nature, thus incapable of capturing the effects of heterogeneity on applications.

The determined need for heterogeneity in various dimensions is an important requirement imposed on sensor network testbeds. This especially applies to the evaluation of smart sensor network applications, dynamically integrating new and unknown sensor types and capabilities. In this paper, we analyze a set of desirable objectives (see Section II) for a sensor network testbed for smart applications. After comparing the assets and drawbacks of different existing sensor network deployments with the determined premises in Section III, we present our *tubicle* sensor platform in Section IV, a heterogeneous node architecture for sensor network experiments. Implementation details are given in Section V, discussing the capabilities in depth, followed by the conclusion of this paper in Section VI. Further details on the exact hardware implementation and the intended deployment of the described *TWiNS.KOM* testbed are presented in the Appendix.

II. Testbed Objectives

To allow performing a large variety of sensor network experiments on a testbed, we have determined several objectives for the nodes to fulfill. These were taken from real experiment deployments, such as the existing ones to be discussed in Section III, as well as the requirements for research of our own lab students and staff. The following set of desired capabilities has been determined:

Heterogeneity – While many homogeneous deployments make use of a unified platform with a given set of sensors, our testbed architecture targets to be extensible by many kinds of sensing devices. This includes, but is not limited to, devices with different computational capabilities, radio interfaces, and sensors.

Mobility – Integrating mobile sensing devices into given experiments allows to regard the impact of node mobility. This is especially helpful in analyzing changes of the link properties during runtime. Moving nodes also pose a challenge to many sensor network routing algorithms, and are thus essential in the process of their validation.

Deployment Support – Once a set of testbed nodes are deployed, the efforts to monitor and individually control them increases with their number. Centrally coordinating node control operations, such as reset, halt and resume, leads to significant reductions of the efforts required. Remote reprogramming functionality additionally simplifies deploying new application images.

Portability – A compact testbed architecture allowing to relocate the nodes easily is a prerequisite for experiments in different environments. Battery-powered operation of the nodes even provides the capability of relocating the nodes during experiment runtime to change the underlying environment easily.

Debugging Capabilities – It proves very helpful to output textual status messages as well as the contents of registers, variables and objects to the user, as especially WSN nodes exhibit extremely limited user interfaces (some platforms do not provide a user interface at all). Converting the status data to a human-readable form and forwarding it to the user is hence strongly desired.

This list of capabilities defines the minimum requirements for our testbed architecture. The decision to set up a real-life testbed was mainly motivated by the fact that simulations often employ statistical models of radio and sensors, while real deployments inherently exhibit the native characteristics of real sensor, radio, and hardware devices. By setting up a testbed, we aim to create a generic platform that can be used in various research domains, including support for context-aware communications, object tracking, validation of ad hoc routing protocols, ambient intelligence, building maintenance, and emulation of medical and emergency scenarios.

III. Related Work

In general, two kinds of sensor network deployments can be distinguished. As *indoor testbed* deployments typically feature a set of nodes with debugging and deployment support interconnections, the node behavior can be easily traced, and software development and modifications performed quickly. On the other hand, many *outdoor experiments* are well suited to conduct long-term experiments to gain information on the performance and efficiency of existing algorithms that have passed their verification on an indoor testbed. However, debugging support is often limited in outdoor deployments.

As of today, many testbeds provide support for computational heterogeneity through a two-tier architecture with small sensing and more powerful computing nodes. Other architectures introduce heterogeneous extensibility by allowing to attach different sensor platforms that use the same radio channels and protocols. However, to the best of our knowledge, only the Kansei testbed [6] exhibits heterogeneity in terms of communication protocols, sensor hardware, and computational capabilities. Still, although capturing real-world characteristics, none of the indoor testbeds has been set up to take real user behavior into account; instead, they rather rely on physical readings. A selected set of sensor network deployments of in- and outdoor kinds have been collected in this section, and are compared in Table I.

A. Indoor Testbeds

Indoor testbed setups are a common step on the way to outdoor deployments, as they provide a useful basis to run practical experiments with the capability of exchanging messages with the connected nodes. A selection of existing indoor testbed structures are presented and briefly discussed.

Werner-Allen et al. initially connected 30 Mica motes in the Motelab testbed at Harvard University [7]. By today it has been upgraded to connect a set of 190 deployed tmote sky sensor nodes with a central server over their USB connections and Linksys NSLU2 devices acting as gateways between the USB and Ethernet protocols. All motes being connected to the backbone network, this structure allows to transmit control commands to, and debugging information from the nodes over the secondary connection.

In a similar fashion, Handziski et al. set up the TWIST sensor network testbed [8], initially consisting of 57 eyesIFXv2 nodes, interconnected in the same manner. It has since been upgraded to 102 eyesIFX nodes and 102 tmote sky devices. Spare USB hubs and Linksys NSLU2 devices were deployed to allow relocating the sensor nodes to different pre-defined positions.

Beutel et al. [9] proposed the JAWS backbone network where each mote in the sensor network is connected to a BTnode over its serial port. The support network communicates over a secondary radio channel, and is well-suited to introduce both deployment support and mobility in the network, as it is not dependent on wired connections. However, by using the backbone network for deployment support only, it does not contribute to heterogeneity in terms of available radio protocols.

In the Kansei testbed, 210 heterogeneous nodes – combining an Extreme Scale Mote and a Stargate each, with 150 nodes carrying tmote sky devices as well – have been deployed in a grid structure by researchers of Ohio State University [6]. In addition, a portable array of 50 Trio motes and five mobile robots fitted with a heterogeneous integration of mote platforms are available.

A combination of motes and more powerful microservers is present as well in the EmStar architecture by Girod

TABLE I
COMPARISON OF EXISTING DEPLOYMENTS AND TESTBEDS

Deployment Name	Heterogeneity			Mobility	Deployment Support	Portability	Debugging
	Computing	Sensors	Radio				
Motelab [7]	—	—	—	—	yes	—	yes
TWIST [8]	yes	—	—	pre-defined locations	yes	—	yes
JAWS [9]	—	—	—	yes	yes	yes	yes
Kansei [6]	yes	yes	yes	yes	yes	50 nodes	yes
EmStar [2]	yes	—	—	yes	yes	16 nodes	yes
Cane Toad Monitoring [3]	yes	—	—	—	—	yes	—
Volcano Observation [10]	—	yes	—	—	—	yes	—
Semiconductor Plant / Oil Tanker [11]	yes	yes	yes	—	—	—	packet traces
ZebraNet [12]	—	yes	yes	yes	—	—	—
Great Duck Island [13]	—	yes	—	—	—	yes	battery level
Health Monitoring [14]	—	yes	—	yes	—	yes	—
Potato Field Observation [5]	—	—	—	—	—	—	—
Glacier Recession Tracking [15]	yes	yes	yes	—	—	yes	—

et al. [2]. In addition to its simulation, emulation and evaluation capabilities, 55 Mica motes were deployed in a ceiling array to report seismic activity, and 16 portable motes are available for mobility support or field experiments.

B. Outdoor Deployments

A number of sensor networks have been deployed in real-world application scenarios. Exemplarily, a selection is listed here, giving a small insight about their experiment objectives and the underlying topologies used.

- The Australian cane toad population was monitored by Hu et al. [3], employing microphone-fitted sensor nodes to sample surrounding frog croaks, and determining the species of frogs that was originating the sound. The deployed nodes were separated in a two-tier architecture, with high-power Stargate devices performing the analysis, and low-power Mica motes that sampled the data.
- Volcanic eruptions have been monitored by Werner-Allen et al. [10] in 2005, deploying a set of sensor nodes around Volcán Tungurahua. Sensor readings were gathered in a distributed manner and forwarded to an external sink for processing and evaluation.
- A semiconductor plant and an oil tanker have been fitted with sensors by Krishnamurthy et al. [11], performing vibration analysis, infrared thermography and ultrasonic detection. Sensor data sets taken by Mica motes were forwarded to deployed Intel Motes, from where they were relayed to a Stargate, acting as central data sink.
- The wildlife of zebras was tracked in the african desert by Juang et al. [12] in the ZebraNet project. Social relations between zebras as well as GPS-based information about their current whereabouts were gathered and stored. Aggregated data sets were subsequently forwarded to the base station, freeing new space on the devices.
- In the Great Duck Island project, the habitat of duck populations were monitored by Mainwaring et al. [13], analyzing their nesting behavior and environmental parameters. Readings were taken by a distributed set of sensor nodes, collected at several data sinks within the network, and subsequently forwarded to a central server for evaluation.
- Health monitoring was done by Shnayder et al. [14] in the CodeBlue project, attaching Body Area Networks to people and performing electrocardiograph, pulse oximeter, and motion sensor readings. The heterogeneous set of sensors hereby allowed to gain and aggregate information from different sources.
- A set of sensors has been deployed on a potato field by Langendoen et al. [5], and were set up to monitor the microclimate. In a similar fashion to the aforementioned networks, sensor readings were collected in a distributed manner, and forwarded to a sink at one corner of the field, from where they were transmitted to a server.
- The recession of glaciers was tracked in Norway by Martinez et al. [15], dropping sensors into holes drilled into the ice. The sensors were set up to monitor their positions, temperature, and a set of further environmental parameters. The results were subsequently forwarded to a base station on the ice, from where they were relayed to a server storing the measured data sets.

In summary, these outdoor deployments were mainly used to take environmental measurements from a given set of sensors, process and aggregate the results, and forward them to a central device for analysis. The set of sensors was assumed to remain static in most cases, disallowing for smart sensor network applications with dynamic changes in the availability of sensor types (sensor node *churn*). Still, many practical issues with sensor networks were found in the evolution of the used algorithms and shared in the mentioned publications, providing helpful information to other application developers.

IV. TUBICLE AND TWINS.KOM

Based on the constraints set in Section II, an architecture has been developed, incorporating all set design requirements, and additionally offering a set of distinct further features. All hardware has been integrated within a transparent 40cm (16inch) acrylic glass cylinder to reduce radio attenuation and allow for a closer look at the sensor node status LEDs. Three different sensor network platforms have been integrated in our

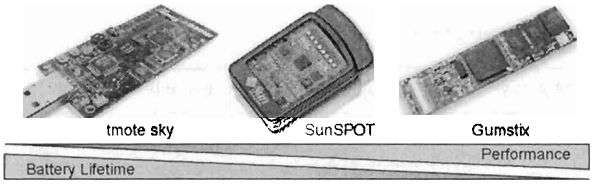

Fig. 1. Comparison of the Employed Platforms

tubicle, and are compared schematically in Fig. 1. In contrast to sensor nodes capable of emulating arbitrary node platforms, such as the FPGA-based nodes presented in [16], our design relies on the integration of existing platforms well known and supported by the sensor network community. This selection of components allows existing applications designed for any of the integrated platforms to be run our nodes.

The approach to heterogeneously combine three platforms is similar to the architecture of Kansei [6], although the basic deployment of TWiNS.KOM is targeted to take place in an office environment to support context-aware applications, provide object and person tracking capabilities, and thus allow to realize ambient intelligence.

A detailed system blueprint is given in the Appendix, where complete diagrams and software configurations allow the gentle reader to self-construct a tubicle. A photography of the integrated sensor node platform is shown in Fig. 2, and detailed descriptions about its components are presented in the following subsections. An overview of the interconnections between the integrated devices is depicted in Fig. 3.

A. Gumstix Verdex

The Gumstix embedded Linux platform is based on an PXA270 XScale CPU operating at a clock frequency of 600MHz and featuring 128MB of RAM [17]. It runs a Linux operating system, and dedicated hardware provides support for WiFi and Ethernet connectivity, USB host functionality, a microSD memory card slot, and a sound chip. Additionally, the platform offers a number of General Purpose Input/Output (*GPIO*) signals, which can be used to connect external sensors or hardware. The provided USB host function and multiple serial ports allow to attach peripheral devices easily – in the standard version of our tubicle, a Bluetooth dongle is connected to provide connectivity with sensing devices using this radio protocol.

B. SunSPOT

The SunSPOT is a wireless sensor node platform based on an ARM920T CPU running at a clock frequency of 180MHz and offering 512kB of RAM [18]. It runs a Java Virtual Machine and can hence execute Java applications, providing type-safety to developers and allowing for rapid application prototyping. It additionally offers a sensor board with temperature, humidity, and light sensors, a three-axis accelerometer, and a set of GPIO pins. Due to the implemented Java Virtual Machine and the comparably small amount of RAM, it has less computational performance than the aforementioned

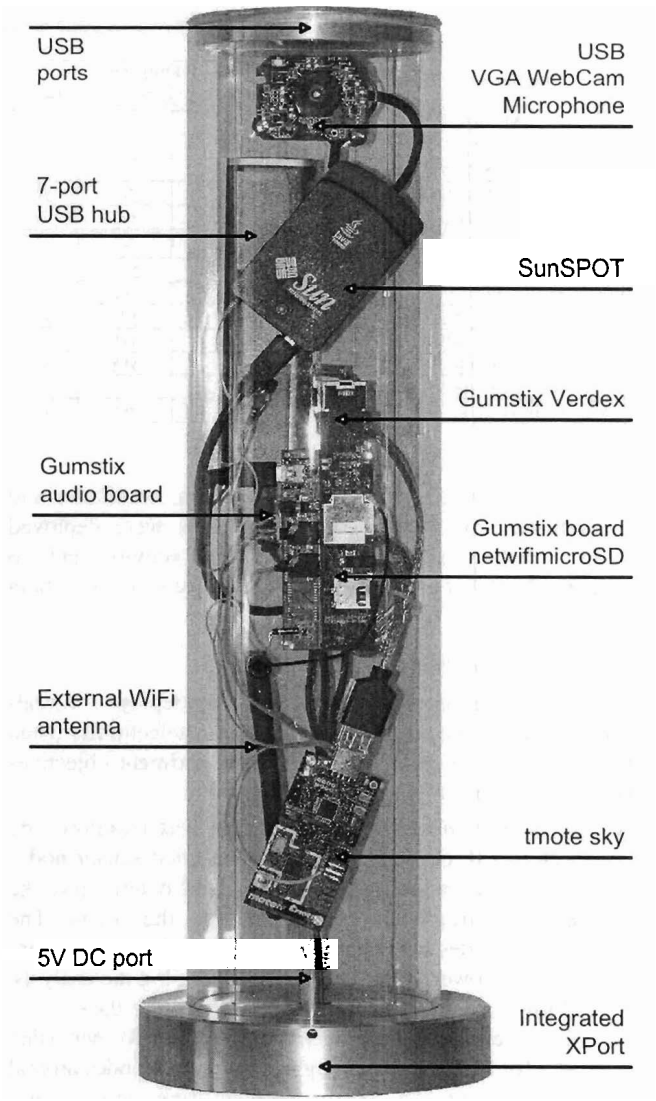

Fig. 2. The TWiNS.KOM Sensor Tubicle

Gumstix platform. Making use of the Chipcon CC2420 radio transceiver, it is additionally limited to radio communications over the IEEE 802.15.4 protocol stack.

C. tmote sky

The tmote sky (identical to the TelosB mote) sensor node platform is based on a low-power 16-bit Texas Instruments MSP430 microcontroller, running at clock frequency of 8MHz only, and thus allowing for significant energy savings. However, it is limited to 10kB of RAM and 48kB of Flash memory [19], which necessitates lightweight operating systems, such as TinyOS [20]. The tmote sky platform features temperature, humidity, and light sensors, and communicates over IEEE 802.15.4 as well. An on-board flash chip allows to store up to 8 MBits of data.

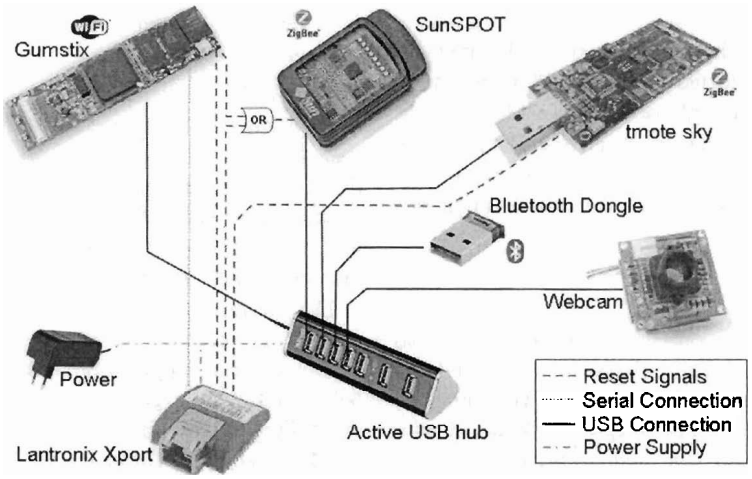

Fig. 3. Interconnections Between the Employed Platforms

D. Additional Features

An additional Lantronix XPort [21] is integrated within the tubicle socket and acts as an Ethernet-to-Serial converter, effectively allowing to attach the node to the Ethernet and thus remotely log in over a serial console connection. It features three additional GPIO lines, which are connected to the reset inputs of all integrated platforms, allowing to remotely reset the devices and thus recover from unexpected errors. The serial console even allows to intervene with the boot loader operation and update the flashed Linux image on the Gumstix. In conjunction with a centralized control operations server, this serial interface drastically increases usability of the testbed by allowing to broadcast kernel images and thus centrally update all attached nodes.

The Labtec Webcam Pro attached to the USB hub can be configured to deliver video streams or still pictures. Combined with the computational power of the Gumstix platform, new high-level sensing capabilities, such as object recognition or sophisticated activity detection, render possible.

V. Implementation Details

The addressed objectives that were determined in Section II have been incorporated by the system set up in the tube. A detailed discussion on how they were addressed, resolved, and implemented is given in this section.

A. Heterogeneity

Different flavors of heterogeneity are supported by our sensor network testbed. The common interpretation of having nodes with different computational power is realized by integrating tmote sky, SunSPOT and Gumstix sensor devices. While the tmote sky platform is limited in computational power, memory and program size, these nodes present a rather inexpensive base for wireless sensors. The more powerful SunSPOTs are capable of performing more demanding algorithms, although still being limited to the IEEE 802.15.4 radio transceiver and the corresponding low data rates. The Gumstix platform bridges the gap to other network interfaces, and offers connectivity to Bluetooth and WiFi.

However, heterogeneity is also supported in terms of different sensor modules attached to the sensor nodes. A variety of these sensors has been integrated on the sensor node platforms for demonstration purposes, including a heartbeat sensor, passive infrared (PIR) detectors, RFID readers and Nike shoe tags based on the ANT radio protocol [22].

Employing three different platforms in our sensor network tubicle allows for both experimenting with a single platform type, i.e. deactivating all other integrated devices to avoid interference, as well as evaluating the assets and drawbacks of heterogeneity by providing nodes with different computational performances. The GPIO extension pins of all employed platforms and the integrated USB hub allow to attach new sensors easily, and spare areas on the carrier board offer suitable locations for prospective extensions.

B. Portability

The sensor tubes are powered by an external 5 volts wall plug, providing power to all integrated platforms. Replacing this wall plug by a battery enclosure and connecting a battery instead allows for unbound movements of the tubes, and thus to perform experiments at any place. The static operating current with all devices being active has been measured as 1100 milliamperes, equalling a total power consumption of 5.5 watts. Power consumption can however be reduced by deactivating the illumination LEDs and the Ethernet-to-Serial converter, if the node is not connected to the Ethernet support network. Eventually, selectively deactivating the integrated platforms, radio interfaces and USB peripherals can lead to even further savings, making the tubicles operable on batteries for up to several hours. A sample application run with WiFi and IEEE 802.15.4 communications originating from tmote sky and Gumstix could successfully be run at a current of 500 milliamperes only.

The robust enclosure allows a stable stand in most places, and protects the nodes from physical influences. The transparent tube material has been selected to reduce radio attenuation to a minimum and allow clear sight on the embedded devices. The main target of tubicle placement is distributing them evenly in office-like buildings, however their portable design makes them suited to be deployed in most network topologies and environments.

C. Mobility

The support for mobility in our testbed is present in two different degrees, as both tubicles affixed to mobile robots, and single node platforms with different sensor types attached to moving objects or persons can be integrated. At the current stage, only the latter solution has been implemented, as it provides more information about the current location and context of objects, being an important data source for our targeted context-aware applications.

Moving nodes are supported in our testbed by providing additional tmote sky and SunSPOT devices, which can be mounted on objects desired to be tracked, or provided to persons to trace human movement schemes. Gained information can be used to improve existing movement models in office environments. Users carrying mobile devices at their person for the day or the runtime of the experiment also allow for emerging application scenarios like reality mining [23].

When attaching tubicles to mobile robots, the powerful Gumstix platform can be configured to additionally perform the task of coordinating node movements. The resulting mobile devices are full-featured, supporting all determined functionalities, and can thus seamlessly integrate with TWiNS.KOM. The platforms based on mobile robots offer a variety of capabilities, such as performing cooperative sensing tasks or taking the role of mobile agents in sinkless networks. The supported battery supply mode allows for unbound operation for an extended duration.

D. Centralized Control

A control software has been developed, allowing for sophisticated node management operations, similar to the Web interfaces of Motelab [7] or TWIST [8]. Apart from offering capabilities to deploy new firmware images to the platforms, it manages the connection to the Gumstix over its serial port (tunneled over the Ethernet connection), and allows to exert node control operations, such as to halt, resume or reset the nodes. Data logged from the connected motes can be retrieved after completion of an experiment via the node management tool to have a synoptical view on the occurred events.

Node control operations are mainly targeted to use the WiFi channel, as it provides higher throughput than the serial connection tunneled over the Ethernet. However, in case the WiFi adapter is configured to perform different tasks, such as ad hoc routing, it cannot be employed to manage a backbone network. In this case, the Ethernet-based deployment support network provides similar functionality.

E. Debugging Support

While the centralized control interface focuses on transmitting control commands and evaluating the application output after running experiments, real-time debugging support offers the capability of forwarding all transfers taking place over the serial port of SunSPOT and/or tmote sky devices to the node management application. Messages can be investigated in realtime this way, allowing the developer to immediately intervene when unexpected events occur.

The platforms can be controlled both in terms of sending as well as receiving messages over their serial ports. This allows to trace application behavior and deduce erroneous behavior from the gathered data easily. Real-time debugging of applications running on the Gumstix that occupy the WiFi interface can be performed as well, making use of the Ethernet connection. If no wired access to the tubicle is possible either, data logging capabilities are present to evaluate collected data when the experiment has finished.

VI. Conclusion

In this work we have presented the tubicle, an integrated sensor node tube comprising three embedded systems with different characteristics, thus creating a truly heterogeneous sensing platform. In the context of our TWiNS.KOM testbed, we target to deploy twenty tubicles, controlled and administered by our testbed management application. The tubicles have been designed to provide extensive support to application developers, both in terms of node control and debugging capabilities. Wired and wireless access is possible, allowing to perform experiments at virtually any location.

The increasing demand to evaluate algorithms designed for heterogeneous environments has necessitated according testbed infrastructures. Our tubicle platform has been designed to fulfill these requirements, with rich support for both application and hardware developers. Mobile and moving sensors are fundamental entities in object and person tracking scenarios, although not widely included in existing indoor sensor network testbeds. To cover mobility in sensor network experiments on our testbed, we allow to regard mobility of sensors attached to moving objects.

We believe that fitting an entire heterogeneous sensor network node into a single system, adding deployment and debugging support, and keeping the solution as extensible as possible, provides an excellent basis for sensor network experiments and can thus lead to significant improvements of algorithms and applications.

Acknowledgment

This research has been supported by the German Research Foundation (DFG) within the Research Training Group 1362 "Cooperative, adaptive and responsive monitoring in mixed mode environments". We would further like to thank the Adolf Messer Foundation, Frank Jöst and the mechanic's workshop of the department of Electrical Engineering and Information Technology at Technische Universität Darmstadt.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, 2002.

[2] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[3] W. Hu, V. N. Tran, N. Bulusu, C.-T. Chou, S. Jha, and A. Taylor, "The Design and Evaluation of a Hybrid Sensor Network for Cane-toad Monitoring," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[4] T. Locher, P. von Rickenbach, and R. Wattenhofer, "Sensor Networks Continue to Puzzle: Selected Open Problems," in *Proceedings of the 9th International Conference on Distributed Computing and Networking*, 2008.

[5] K. Langendoen, A. Baggio, and O. Visser, "Murphy Loves Potatoes – Experiences from a Pilot Sensor Network Deployment in Precision Agriculture," in *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems*, 2006.

[6] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, and H. Cao, "Kansei: A Testbed for Sensing at Scale," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, 2006.

[7] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[8] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks," in *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, 2006.

[9] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald, "Next-Generation Prototyping of Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004.

[10] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring Volcanic Eruptions with a Wireless Sensor Network," in *Proceedings of the European Workshop on Wireless Sensor Networks*, 2005.

[11] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, "Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005.

[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebranet," in *Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.

[13] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[14] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, "Sensor Networks for Medical Care," Diversion of Engineering and Applied Sciences, Harvard University, Tech. Rep. TR-08-05, 2005.

[15] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A Sensor Network for Hostile Environments," in *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[16] H. Hinkelmann, A. Reinhardt, and M. Glesner, "A Methodology for Wireless Sensor Network Prototyping with Sophisticated Debugging Support," in *Proceedings of the 19th IFIP IEEE International Symposium on Rapid System Prototyping*, 2008.

[17] Gumstix Inc., "Gumstix - Way Small Computing," Online: http://www.gumstix.com, 2008.

[18] Sun Microsystems Inc., "Project SunSPOT - Sun Small Programmable Object Technology," Online: http://www.sunspotworld.com, 2008.

[19] Sentilla Corp., "Tmote Sky Datasheet," Online: http://www.sentilla.com/moteiv-transition.html, 2007.

[20] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.

[21] Lantronix Inc., "Lantronix XPort DirectX+ Embedded Device Server," http://www.lantronix.com, 2007.

[22] Dynastream Innovations Inc., "ANT - The Power of Less," Online: http://www.thisisant.com, 2008.

[23] N. Eagle and A. Pentland, "Reality Mining: Sensing Complex Social Systems," *Journal of Personal and Ubiquitous Computing*, vol. 10, no. 4, 2006.

Appendix

This section provides detailed information on how to construct a tubicle[1]. Starting from the list of required parts, the interconnections are discussed in detail, and the concepts of the node management software explained.

A. Shopping List

The employed components and according configuration options have been compiled in this section to allow constructing a basic tubicle. Emphasis has been put on using parts that are available off the shelf. To build a tubicle, you need:

- A Gumstix Verdex XL6P base board with the following additional modules:
  - The audiostix2 module for USB host and audio capabilities
  - The netwifimicroSD module offering WiFi, microSD and Ethernet connectivity
  - A microSD card with a capacity of at least 1 gigabyte

  A stationary SunSPOT sensor platform to be mounted within the tubicle
- One tmote sky (or TelosB) mote for fixed installation
- A seven-port USB hub, such as the Trust HU-5870V, with its uplink port connected to the Gumstix board and the following devices attached:
  - Labtec Webcam Pro (stripped from its case for eased mounting), or any other web camera well supported by Linux
  - A small Bluetooth USB Dongle, such as the Trust Ultra Small Bluetooth 2.0 USB Adapter
  - Cables connecting to the USB ports of the SunSPOT and tmote sky motes

  An Ethernet-to-Serial converter, consisting of:
  - A Lantronix XPort Direct+ device
  - A corresponding printed circuit board including a voltage regulator
  - A logical OR gate to forward reset command from both Gumstix and XPort to the SunSPOT
- An actively powered loudspeaker that connects to the audiostix2 board
- A 5 volts wall plug, providing a minimum output current of 1.5 amperes
- Some 0.1" connectors and cables to interconnect the devices
- A set of fancy LEDs

[1] Do not hesitate to contact the authors for further information.

All of these components have been mounted on a acrylic glass board placed inside a transparent tube, enclosed by socket and top parts made out of aluminum, with the XPort and the LEDs fitted into the aluminum socket.

B. The Interconnections

The interconnections between the listed devices were made according to Fig. 3. An Ethernet-to-Serial converter and the corresponding voltage regulator are mounted in the socket of the pipe, allowing to remotely access the Gumstix platform and set it up in different manners, such as to perform wireless ad hoc network experiments using the employed radio chip. While experiments are conducted, data traces are saved to the integrated memory card, to be forwarded to our experiment server after the experiment. The according setup sequence is managed by the server and can be executed by the user by a single mouse click or within scripts.

Reset signals are wired from the XPort's GPIO pins to each connected platform, and connected accordingly. On the tmote sky 6-pin expansion connector, pin 6 provides an active-low connection to the mote's reset signal and can be directly tied to the XPort. In a similar manner, pin 37 of the 60-pin connector present on the Gumstix can be connected to the XPort to reset the platform. Forwarding the reset signal to the SunSPOT proved slightly more complicated as the same signal must be triggered when programming the mote. A logical OR gate connected to the reset signal input solved the issue by allowing both the XPort (for resetting) and the Gumstix (for programming) to set the pin. Keeping the reset pin set for a longer duration additionally allows to turn the SunSPOT off.

Further interconnections exist between the USB host connection of the Gumstix and the connected peripheral USB devices. Both tmote sky and SunSPOT occupy a USB slot, and so do the Bluetooth dongle and the employed webcam. Two USB ports are present on the top of the tubicle, allowing to program additional connected SunSPOT or tmote devices with application images.

The tubicle is illuminated by a set of LEDs that make the upper rim shimmer in a smooth pink color, and thus allow to identify whether the node is running.

C. The Software

The Gumstix platform runs the *Ångström-2007.1* Linux distribution, that is currently available with a version 2.6.21 kernel. A plethora of pre-compiled software bundles are available via its integrated package manager.

To support native compilation of application software on the platform, a complete build tool chain has been installed. For larger projects that exceed the Gumstix' computational capabilities, cross-compilation on machines with installed XScale compilers is possible.

To support the integrated webcam, the additional *gspca* kernel module has been installed, and both *w3cam* and *xawtv* are available to capture images from the webcam. Java applications can be run by *jamvm*, a lightweight Java Virtual Machine compliant to the JVM specification, with the GNU Classpath libraries installed. Additionally, the libraries for communication with the SunSPOTs (*rxtx*) and tmote sky (*toscomm*) have been installed, providing interfaces to the motes for Java applications and access over the console.

Stable revisions of our management interface, based on a system running an Apache Web server, a SQL database, and a PHP engine, are available for download on our TWiNS.KOM website, located at http://www.kom.tu-darmstadt.de/twins. The management interface comprises functions for easy application deployment, node control, and visualizes status information about participating nodes.

Programming the tmote sky and the SunSPOT nodes is automated by a set of scripts on the Gumstix platform that can be triggered by the Web interface. However, the Linux installation also allows access via the secure shell (*ssh*), which can be used to manually control the platform and the attached USB devices.

D. Intended Deployment

Having successfully mastered its prototype stage, the TWiNS.KOM tubicle is ready for deployment at scale. In a first step, the intended deployment location will be the offices of the Multimedia Communications Lab (KOM) at the Hans-Busch-Insitut building of Technische Universität Darmstadt. Twenty tubicles will be placed in individual locations during the initial deployment phase, capturing user behavior and environmental parameters. By configuring the tubicles accordingly, sensed data can be used for various purposes, such as supporting the process of context determination.

Once this core of the testbed has been set up, additional devices can be added easily, including tubicles mounted on mobile robots. However, any device communicating over one of the supported radio protocols can be connected to the sensor network testbed. When smart applications are executed, these newly added platforms are detected and integrated without any additional efforts required.