The Particle Computer System

Christian Decker, Albert Krohn, Michael Beigl, Tobias Zimmer
Telecooperation Office (TecO)
University of Karlsruhe
Karlsruhe, Germany
{cdecker, krohn, beigl, zimmer}@teco.edu

***Abstract*—This paper presents a sensor-based, networked embedded system, referred to as the Particle computer system. It is comprised of tiny wireless sensor nodes, capable of communication with each other, as well as connectivity with backend, PC-based systems, thereby facilitating software development and data analysis in an integrated systems package. The core design principles of the sensor nodes enable operation in very mobile settings and truly ad-hoc, peer-to-peer interoperation without the intervention of a master or explicit middleware layer. The two main system properties highlighted in this paper are: 1) information distribution to all components within the system and 2) the usage of a common communication language in all system components. This language has been proprietarily developed for the Particle system and is known as ConCom. As a result of these system properties, we have found the Particle system to be very extensible and applicable in many everyday scenarios. The paper presents insights to the implementation of the Particle computer system, including software development and data analysis capabilities, and the overall system integration.**

Keywords – Particle Computer, AwareCon, ConCom, Middleware-free Architecture, Sensor Network, File System

I. INTRODUCTION

The Particle computer system (details found at http://particle.teco.edu) is the result of intensive, long-term research. The current system has its roots in the EC funded Smart-its (http://www.smart-its.org) project, which started in 2000 and ended in 2003. Within the Smart-Its project a concept for a generic platform appropriate for embedding computation in the real world was developed and implemented as three concrete prototype platforms. The resultant devices, originally called Smart-Its, were small, embedded devices, allowing attachment to everyday objects, augmenting them with sensing, computation and communication capabilities. Subsequent to the project's conclusion, the authors of this paper resolved to refine and enhance the prototypes, leading to the Particle platform described in this paper. The paper continues by motivating the design principles of the Particle computer and then proceeds to discuss its core computing, communications and sensing technology in section 3. Section 4 reverts to a broader picture of the system, with a focus on integration, while section 5 explains the development and data analysis issues that arose. The paper concludes with a short experience report in section 6 and a future work in section 7.

II. MOTIVATION

Networked sensor systems are becoming very popular as base technology for many military, industrial and home applications, such that generalized platforms for supporting the prototyping, analysis and ensuing development of these applications is in high demand. One of the more popular platforms to have emerged from collaboration between academia and industry is the Berkeley Motes [8], whose developers can claim a rich research history, including the area of distributed data processing within sensor networks. Communication is organized using the concept of active messages [9], which facilitate automated network organization based on the building and maintenance of a routing tree between all nodes. The backend system, such as a PC, has a special role as the controller of the tree's root node, imposing a hierarchy on the overall system.

The Particle system shares some similarities with the Motes, yet there are fundamental differences stemming from the development goals. In particular, the Particle computer system targets *highly mobile settings*, where many wireless sensor nodes encounter each other and exchange data over a relatively short period. An example of such a setting is in an office, where lots of people interact with each other and with their environment in order to make appointments, organize and schedule meetings, or to support collaboration during meetings. Consequently many everyday items and objects, including tables, chairs, pencils, notepads, office machinery and others, are interacted with and carried by people in efforts to collaborate and complete their everyday tasks. We identified more than 450 such items in a regular office, both purposely and inadvertently transported by office workers, for usage in planning, meeting, developing ideas, or even in more casual situations. As these items are already part of the everyday, productive office interactions, they provide a valuable source of analysis information regarding the dynamics of tasks within the office environment and subsequent development of solutions for enhancing the interactive experience with these items. This however requires unobtrusive sensing or task, activity and environment properties, as well as a means of exchanging and interpreting this potentially explosive set of data. We therefore designed the Particles nodes with these requirements in mind, such that attachment to various objects like chairs, windows, doors, pens, video projectors and other devices, items and even people, various control tasks and systems in the office environment can make more informed decisions. Secondly, by collaborating with each other, the nodes report environmental conditions or intrinsic states and may decide appropriate actions based on rules defined for the management of the environment. Consider automating the decision to close the jalousie if the video projector is switched on, or take a picture of the whiteboard when the pen is laid down. The deployment properties and rules will vary in different environments, such that the feature of *ad-hoc collaboration* between sensor nodes is desirable. Ideally no master should be required or previously selected, as the network should be organized in a peer-to-peer fashion, enabling the mutability of collaboration possibilities for highly dynamic tasks. However, in cases where information from backend systems in a LAN infrastructure is needed or applications in this infrastructure should be supported, the Particle nodes should be *seamlessly integrated in this infrastructure*. Consider a calendar application that needs to be updated when a spontaneous meeting is detected in a previously unreserved room. In order to support these goals a very pragmatic design principle has been applied, namely, "there must exist a direct means of

communication between all system components". An appropriate radio protocol for the sensor nodes should incorporate this principle in order to support high mobility and ad-hoc collaboration. Similarly, the backend integration requires that the interaction between nodes and the PC-based systems is based on a consistent, common communications interface. This is achieved by two mechanisms: 1) distribution of all information within the system and 2) a common communication language. Both mechanisms combined enable communication that is understood by all components without any conversion or mediation through a third party such as a middleware. As a result, the system architecture can be defined as being "flat", yet portrays loose coupling and high cohesion with its distributed approach, which removes the conceptual border between sensor nodes and backend systems. This implies that every component in the system can be conceptually considered as another Particle node. Application Programmers would appreciate the systems uniformity in this respect, as they could flexibly shift between using the Particle nodes as pure sensor value producers and situations where the nodes operate collaboratively. Even in the case of hybrid application design where functionality is implemented on the nodes and also within the backend, a programmer will always have direct control over the communication. This similarity of software components and sensor nodes enables a straightforward, but still very flexible approach for developing distributed applications with the Particle system. This is needed for the expected diversity of applications in such agile domains like the office domain.

III. PARTICLE TECHNOLOGY

A. *Particle Computer Hardware*

The Particle hardware follows the engineering concept of separation of concerns. In doing so, a Particle node consists of two boards - one for containing the communications functionality and the other for other utilities of the node such as sensing. The communications board is shown in Fig. 1, and implements the Particle's wireless networking functionality. The board's dimensions are 15x48 millimeter and it includes a PIC18F6720 microcontroller running at 20 MHz (5 MIPS), a TR1001 transceiver enabling a data rate of 125kbit/s on 868 MHz, a 512KB flash memory, a real-time clock (RTC) and a power circuit, which employs a power supply compatible with a single, regular AAA battery. Additionally, the board comprises two LEDs for visual indication, e.g. communication status, and a speaker for audio notification. Running on a single 1.2V AAA rechargeable battery the board consumes on average 40mA with the communication and the LEDs active.

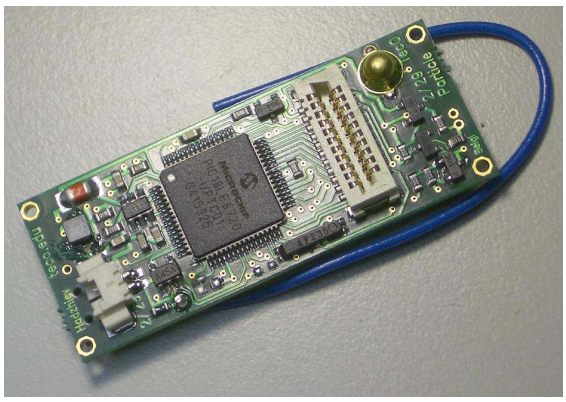

Figure 1. Particle communication board
(transceiver, external flash memory, speaker on the backside)

Particle communication boards can be connected to additional boards via the onboard connector, such that the core Particle hardware functionalities can be extended. The onboard connector provides a 21-pin interface to the microcontroller's I²C and serial communication capabilities, several digital I/O pins and pins for analog measurements through the microcontroller's analog-to-digital converter. The connector further provides the supply voltage for these boards. In [1] an analysis of various applications for sensor nodes identified and classified the selection of sensors commonly used for measurement and deriving information from the environment. This motivated the selection of an acceleration sensor, a temperature sensor, a light sensor and a microphone on a typical sensor board of 17x22 millimeter (Fig. 2a).

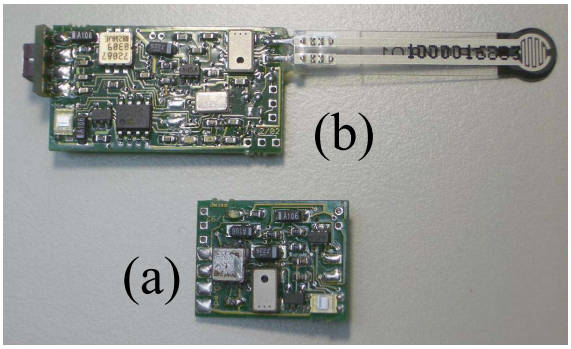

Figure 2. (a) Sensor board with light, temperature, acceleration and microphone sensor, (b) Sensor board with additional microcontroller (backside) and force sensor

The sensors on the board in Fig. 2a are directly connected via both boards' connectors to the microcontroller of the communication board. In addition to the networking tasks, the microcontroller samples and aggregates sensor information, e.g. computing of the average temperature over a period of time. In this case, one could argue that this violates the separation of concerns principle, as proposed at the beginning of this section. However, it is shown that by interconnecting another sensor board (Fig. 2b), which includes its own microcontroller, here a PIC18F452, the sensing and communication functions are easily separated. Important to note is the role of the connector on the boards. It enables the successful separation and is therefore required as a constant part in the hardware design. The implementation of the separation also leads to an inherent distribution of functionality and contributes to defining the flexibility of the Particle computer hardware. This is one of the success criteria for the Particle system, since the hardware can then be utilized in various scenarios covering a huge domain for sensor network applications. There are also other boards that exploit this functional separation, including the so-called "breakout boards" for easy connection of peripheral sensors and hardware, serial boards and USB boards for communication with other systems via RS232 and USB protocol, and power boards for extending the available power supply. An important additional board is the "bridge board" (Fig. 3), which enables communication between Particle nodes and other computing systems over a UDP network.

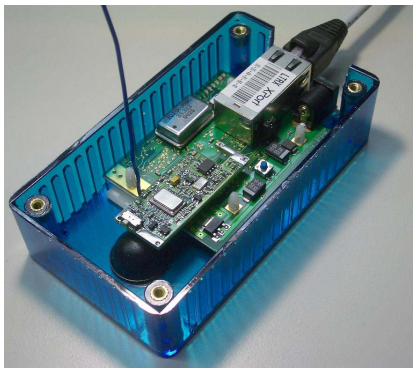

Figure 3. Particle bridge

B. *Particle Communication*

The communication board runs a customized communication protocol AwareCon [2] especially designed for ad-hoc communication, with a design that follows the fundamentals of the established OSI/ISO layered approach. The layers include physical radio layer, (RF) Link Layer (LL) and the Application Convergence Layer (ACL). Several features have been implemented in order to obtain a specialized protocol for distributed networked sensor system and the properties of the Particle's target application environments. Mobility was an important aspect for the protocol design and as a result AwareCon is able to handle high mobility of nodes. In Fig. 4 the delays until a single node was synchronized with the network or with another single node were measured and shown as a distribution. As a result, we found that the synchronization with a new network in range takes typically around 12ms. The mean delay for the synchronization with another single partner is around 40ms.

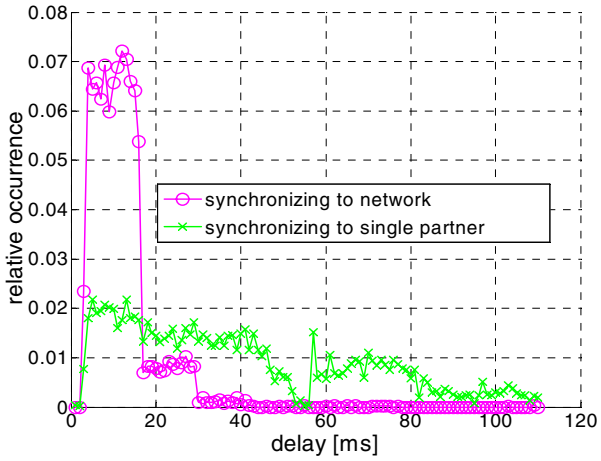

Figure 4. Distribution of synchronization times

Once synchronized, nodes exchange synchronization signals in a random and distributed manner and establish a common time slot scheme. With a common time slot established they can immediately exchange data. With this fast and self-organized synchronization AwareCon is suitable for highly mobile environments.

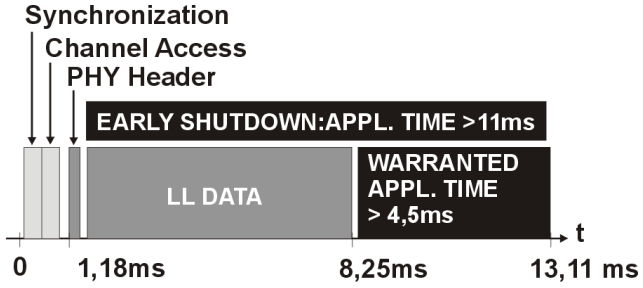

Figure 5. AwareCon time slot

Fig. 5 shows the slotted TDMA structure of AwareCon with its 13ms time slot. The design of AwareCon also reflects the concept of a fully distributed system. Nodes all have equal responsibilities to establish time slots, exchange synchronization signals and keep an established timing scheme alive. There is no access point or master devices like in W-LAN, Bluetooth or many other known protocols. The channel access uses a *nondestructive bit wise arbitration* known from wired networks such as the CAN field bus. This access method achieves outstanding low collision rates especially for high number of concurrent nodes. It is also known for its good capabilities to handle priorities. Since only bits need to be signaled, the arbitration slot can be very short. However, the scheme imposes hard requirements on the hardware, since the Particle node's TR1001 RF front-end has to be constantly switched between sending and receiving mode. We compared the probability of no collisions of the AwareCon arbitration with the traditional CSMA scheme of W-LAN in order to illustrate the performance of AwareCon during the arbitration. The following figure depicts the probability of no collisions for an increasing number of nodes that have concurrent send requests. Thereby, AwareCon uses 10 arbitration slots and the W-LAN arbitration 50 slots.

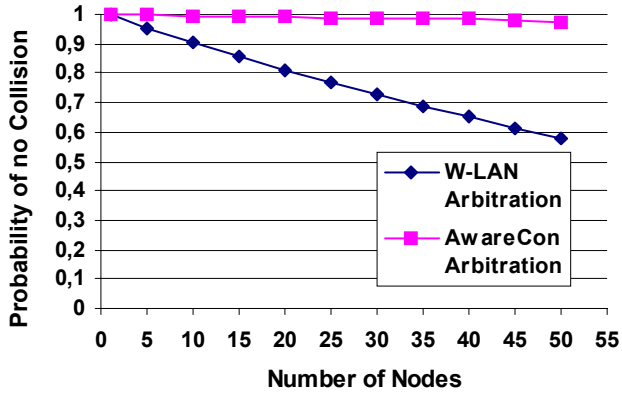

Figure 6. Probability of no Collisions for W-LAN and AwareCon arbitration

The data traffic is organized in packets of 64Bytes data payload. Each time slot of AwareCon can carry one packet of data. Several energy saving mechanisms have been introduced to AwareCon of which the most important one is the so-called early shutdown. Semantic data filtering [10] enables nodes to interrupt and cancel a running packet reception at an early state before the transmission is completed. Thereby, an application on top of AwareCon can subscribe to the data that concerns its logic. During reception, incoming data is analyzed and matched according to these local subscriptions. In case of a mismatch the nodes can turn off their radio front-ends or even go into sleep mode until the next timeslot and thereby save energy. According to Fig.5 this cross-layer approach would save up to 91% if the node is set to sleep until the next slot. As the protocol is mainly implemented in software and runs on the same processor like the application, AwareCon foresees a certain time in each timeslot with no protocol activity to always guarantee a percentage of >33% of the CPU time for the application even during high data traffic times.

C. *Particle Computer Software*

Further capabilities of the communication board are encapsulated in the system library. It provides system functions for a basic configuration of the microcontroller's general I/O pins and I²C and serial communication subsystem, basic access methods for the analog-to-digital converter and the internal EEPROM of 1KB as well as the external flash memory. It further provides methods for reading and setting the onboard RTC. In case the sensors are connected to the communication board, an additional sensor library is used on top of the system library utilizing these system functions. The sensor library includes drivers for sampling previously mentioned typical sensors, such as an acceleration sensor, an I²C temperature sensor, a light sensor and a microphone. The time synchronization of the communication stack works virtually like a scheduler, leaving 4.5 milliseconds for sensor sampling before the next communication phase starts again. The drivers have to be aware of this constraint, as it will affect for instance a series of consecutive samplings. Recently, the Particle file system [5] was implemented on top of all libraries. Representing all

resources, such as communication, sensors and memory as files enables a uniform access via only two functions `read(..)` and `write(..)`. The Particle file system is a hierarchical access structure, which also allows the file-based representation of the API of the system library. All files can be shared among Particle nodes, establishing a concept of distributed software among them.

IV. THE PARTICLE SYSTEM

The Particle system view (Fig. 7) depicts the Particle technology as a lower layer and the backend system components as a higher-level. The latter communicate with other components via UDP or to Particles nodes via bridges. The communication between Particle nodes in the technology layer is reflected as UDP broadcast communication in the backend layer. Thereby, information from the nodes is distributed to all backend components connected to one LAN.

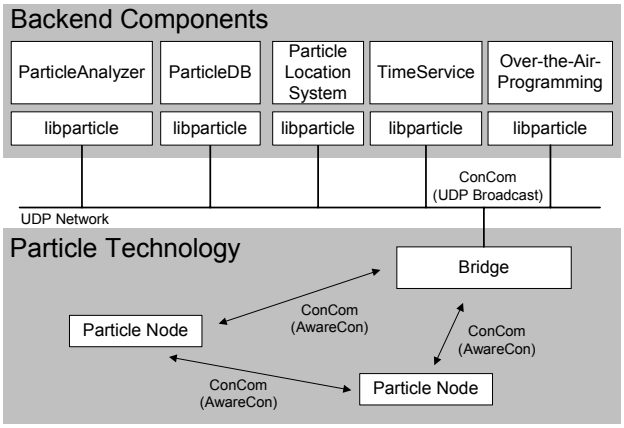

Figure 7. The Particle System

Typical backend system components are permanent services such as the time service (provides the actual time by request of setting the RTC on the core board), the ParticleDB (logs all message communication of Particles nodes), and the Particle Location System [6] (implements a cell-of-origin location system utilizing the bridges). Other components support the development and data analysis processes: the ParticleAnalyzer serves as the real-time monitoring tool, and Over-the-Air-Programming allows an application specific in-situ update of the Particle computer software. Again it is to be noted that the Bridge does not constitute a "mediating middleware component", as there is no semantic translation between the Particles nodes and the backend components when communicating across the bridge. The bridge is just another Particle node with no exceptions. The same is true from the perspective of backend components for the communication with Particle nodes. This design results in a flat system architecture.

A. ConCom

As a consequence of this flat architecture a common communication language between all components is required. In the Particle system the proposed approach is ConCom[10]. ConCom represents data in a strictly typed form of tuples. A tuple starts with a type identifier (3 bytes), followed by a length statement (1 byte) and then a number of data bytes specified by the length. Tuples can be concatenated to sentences. Thereby, the first tuple is referred to as the subject. Fig. 8 illustrates a ConCom sentence. Type identifiers are freely selectable and enable a flexible and expressive way to describe the data. In the example above the sentence is originated with a subject ABC additionally containing version number 1 and continues further with temperature data identified by STE (sensor temperature) of 23.5 degrees Celsius. However, the structure in sentences is rather flat, such that complex statements may prove difficult to describe by a combination of other tuples. In these cases new tuple types have to be introduced describing the complex statement as a block of encoded data, as opposed to a more expressive tuple concatenation. The notion of a subject in ConCom enables application specific processing of data within the system. Each application running on the Particle nodes and on the backend components is identified by its subject. An application subscribes itself to a subject and filters thereafter all received information, while the procedure for sending from the application uses the subject as the prescript of the outgoing message. This enables various applications, distributed across multiple Particle nodes and backend components, to operate in one Particle system at the same time without data interference. As a result, type identifiers differing from the subject, can be reused in other meanings within the context of the respective application. ConCom is in this sense the consistent, underlying coordination model employed throughout the Particle System. The packets communicated between Particle nodes on the lower layer, between backend components and across the both layers use the ConCom format. Without middleware, ConCom tuples enable a type safe way to communicate a diversity of data, ranging from various sensors to data resulting from computing processes on any layer and component in the Particle system.

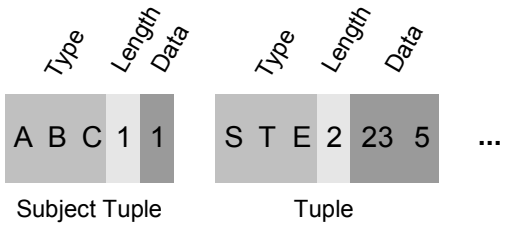

Figure 8. Example of a ConCom Sentence

B. Backend Components

In this section we return to discuss the upper layer backend components of the Particle system, as depicted in Fig. 7. Each component possesses a common interface referred to as libparticle for the communication with Particle nodes or to other components. The libparticle is a cross platform library written in C for Windows and Linux systems. It is a functional part of each backend component and handles the UDP broadcast communication, providing functions for creation, parsing and filtering ConCom sentences. As a consequence, backend components can be distributed throughout the UDP network and operate independently. The backend components therefore follow a similar distributed approach to the Particle nodes.

C. Scalability of the Particle System

Although we have motivated the advantages of complete information distribution and language uniformity throughout the Particle system, we also considered some of the disadvantages of this architectural decision. One of the issues we considered is that of scalability, given that we have already shown in Fig.6 that the AwareCon protocol scales very well and is able to handle many concurrent send requests. The problem of scalability mainly arise at the bridges due to the bandwidth differences between the backend and the Particle nodes. Considering a regular bandwidth of 100MBit/s in a LAN and the 125KBit/s of the Particle nodes, we concluded that 800 Particle nodes would be required in order to exceed the LAN's bandwidth. Since the arbitration of AwareCon prohibits 800 concurrently sending Particle nodes in one Particle network, this mass can only be generated by 800 bridges forwarding data concurrently into the LAN. We therefore do not place emphasis on investigating scalability issues in this regard. Nevertheless, in cases where backend components send ConCom sentences at a high rate they may create a situation of overloaded bridges. A solution to this problem is given in [3] where the informa-

tion distribution is restricted to certain, semantically defined spatial regions. The bridges in the system implement a filtering on such region-descriptions in order to approach the scalability issue. Particle nodes can easily be located in such regions through "pings", which triggers them to send a response. Received by a bridge, this message is preceded by the region description of that bridge. Backend components have to precede their ConCom sentences with this region description in order to communicate with the Particle nodes in that region. Particle nodes themselves are not aware of this scheme. Although this addresses the scalability problem, it has an effect on the communication of very mobile Particle nodes, since they may change their region before the backend has located and communicated with them. In order to estimate the effect of region based communication on mobile nodes, we can use the round trip time (RTT) for the communication between backend and Particle nodes. Assuming that the smallest region is around one bridge – this would address the scalability problem at best – we can determine the maximum speed of the node in order to be reachable by the backend in that region. If that speed is exceeded, the backend would send a ConCom sentence to a region which the receiving node has already left. In this situation the system would not be able to handle the mobility anymore. For a RTT of 100ms and circular region around a bridge with a diameter of 10m and placing the node in the middle between region border and the bridge, the maximum speed would be 25m/s. This is sufficient for office environments as the primary application domain of the Particle system.

V. Development and Data Analysis

In this section we illustrate how to use the distributed system approach in order to develop applications with the Particle system. For testing and evaluation of those applications appropriate debugging and analyzing components are presented.

A. Particle Development

As the first step in the development of a new Particle application the developer selects an appropriate ConCom subject. In each communication this subject precedes the sentence and separates the new application from the parallel running ones. The consistency of subjects is ensured by a type file containing all used ConCom types and their meaning in the context of their respective application. This type file is managed by a type manager implemented as a backend component. A developer registers the selected subject and includes the created type file in his application. The decoupling achieved does not require the type manager during the runtime of an application and therefore maintains the completely distributed approach of the Particle system. The actual programming elements comprise both application development on Particle nodes and backend programming. Both are usually done in C since this language is supported on all layers through the system. On Particle nodes the application development preferably utilizes the uniform access to sensors and the communication stack, which is provided by the Particle node's file system. Consider the following sample expression: `write("/dev/awarecon", "/dev/STE")`. This reads the temperature sensor (abbreviated by its ConCom type) and writes the reading in ConCom format on the communication stack, AwareCon, which completes the transmission. Communication with backend components is integrated seamlessly through the use of ConCom, easing the application design for the developer. Once the application for the nodes is compiled, it is transferred to the specific nodes via Over-the-Air-Programming (OtAP). A predefined ConCom subject in conjunction with an identification of the specific node ensures that the code is downloaded to the one the developer selected previously. The system library of each node application supports OtAP as an integral part, i.e. there is no Particle node program without this capability. OtAP is an essential part of the development process because it enables the in-situ reprogramming of nodes while being integrated in an application scenario. Besides the application code OtAP comprises the complete system image resulting in lots of redundant code to be downloaded. Realizing this, Particle nodes provide a virtual machine (VM) as a second way to developing applications. The developer programs on a PC using a Basic-like language, which is then compiled to an intermediate byte-code. The byte code is downloaded utilizing the OtAP semantic to a specific Particle node running the VM. It is possible to store more than one program on a node and select the one which should be executed. Byte code for a simple application is transferred within 5 seconds, while the transfer of a comparable one including the system image takes 85 seconds under best conditions. In particular, if there are several programming processes running concurrently, the VM based programming contributes definitely to scalability of the overall system, since the number of exchanged messages is reduced. For comparable programs this results in a reduction by a factor of 1/258. However, in general it is common for VM approaches to consume more energy in long-term settings annihilating the savings for fast program transfer. Further, applications run slower since the code needs to be interpreted.

The development of backend components utilizes the libparticle. It shields thereby the details of the communication process and ensures via ConCom the seamless information exchange to other backend components as well as Particle nodes. However, the developer should be aware of the bandwidth differences between backend and Particle nodes. If packets are permanently sent at a high data rate from the backend, the usage of the region-based communication from section IV.C is recommended to maintain the scalability of the system. Nevertheless, short burst can be buffered by the bridge. Once developed and compiled, the backend component can be deployed on any computer system within the UDP network. The seamless integration of backend components without a middleware layer enables a powerful online debugging. Thereby, an application runs on a Particle node and the debugger runs within the backend. In order to allow the debugging of the distributed applications, the debugger needs to be integrated in the environment of the application. Our system architecture exactly enables this behavior as there is no obvious border between the backend and the Particle nodes. This allows the debugger to directly communicate with the nodes and basically hook into the running distributed application. The debugger enables the tracing of function calls, the watching of variables and complex data types like C structs, and the use of assertions. All debugging information is communicated via the AwareCon communication stack during this process to the debugger in the backend. Similar to the OtAP approach a pre-selected ConCom subject was used to identify this information and to separate the debugging process from the application specific communication. Sharing AwareCon for debugging and the application limits the usage of the debugger on code outside the strictly synchronized communication interface. Nevertheless, the approach allows to directly trace the execution of distributed applications just like as another Particle node would see the application. This was firstly applied during the development of the file system and helped to achieve a stable and powerful implementation.

B. Data Analysis

During the application runtime, Particle nodes and backend components communicate with each other. In order to support evaluation of the application, as well as for logging and analyzing purposes of sensor readings, we have developed the ParticleAnalyzer and the ParticleDB. The ParticleAnalyzer (Fig.9) is an all-round tool intended for real-time analysis. ConCom sentence are parsed and sensor information is plotted in real-time graphically as well as on a console for a detailed view.

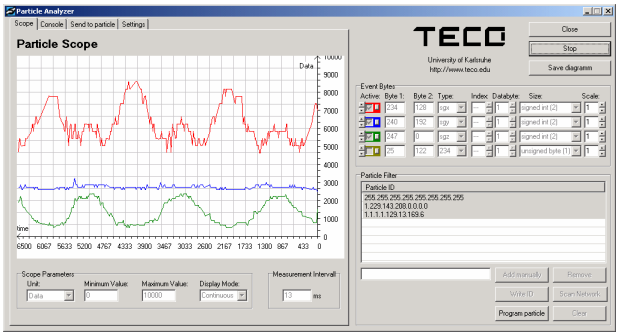

Figure 9. ParticleAnalyzer plotting acceleration data during a rotation of a Particle node

Various filters on ConCom types can be applied immediately in order to separate the data for plotting. The broadcast nature of communication allows the analyzer to passively identify Particles nodes. While the AwareCon synchronizes the data transfer on the radio channel, the bridge and the computing system running the ParticleAnalyzer impose a non-deterministic delay until sensor data is plotted. This can cause distortion of the real-time nature of the data. If the ConCom sentence contains a time stamp tuple from the Particle node's onboard clock, the analyzer can use this information to plot the sensor data correctly. The analyzer links also to the OtAP component, combining in this way the easy selection of Particle nodes and in-situ reprogramming. The ParticleDB (http://www.teco.edu/projects/particledb) is a permanently running backend component which creates a huge archive of all communication going on between Particle nodes. As a result it provides the opportunity for an analysis of long term data. A query template on the WWW enables an comfortable way to query for data using various possibilities to filter them. Returned data can be browsed or optionally can be exported to a comma-separated format for import in other applications.

VI. Experience

Our permanent test bed for Particle applications is the AwareOffice (http://www.teco.edu/awareoffice), where various objects are augmented with the nodes. The chairs register human beings sitting on them and this information controls an electronic meeting doorplate. A whiteboard pen detects interactions like writing, playing or being laid down and triggers a video annotation system. In [4] and in the according video we demonstrated how to build intelligent office environments with the Particle computer system. Objects like chairs, office items and people were equipped with the Particle nodes. The system's flat architecture enables for such setting a very quick and easy setup. Applications incorporated ad-hoc collaboration between sensor nodes or collaboration with the backend components and could be deployed within several minutes. Since all information were distributed to all components, i.e. sensor nodes and backend, and all of them can communicate directly to each other utilizing ConCom applications start operating without the setup of mediating components usually found in middleware systems. With the DigiClip [6] we investigated the contexts of physical documents and tested new sensors, for instance a capacitive page count sensor, by exploiting separation of concerns, which underpins the hardware design.

Several labs around the world already bought Particles, amounting to several communication boards and supplemental hardware. A successful result was the RELATE project in cooperation with the Lancaster University. Particle nodes utilized ultrasonic transducers as sensors and computed in a distributed approach the relative position to each other. An example application of this system can be found in [11]. Recently, OCE (http://www.oce.com) an office solution provider from the Smart Surroundings project (http://www.smart-surroundings.nl) approached us in order to setup another AwareOffice environment as a test bed for further intelligent office technologies within that project.

VII. Conlcusion and Future Work

We presented the Particle system – a completely distributed networked sensor system that seamlessly integrates sensor node technology and backend components without a middleware layer. The approach is achieved by the distribution of all information and the use of ConCom as a common communication language uniformly throughout the system. As a result, the loose coupling of components makes it appropriate for highly mobile and ad-hoc settings. As next steps, we will broaden the palette of sensors and improve the developer support, by introducing new programming abstractions for managing the system at this level of distribution. The goal is to decrease the complexity for novice developers of the system, but still to facilitate the flexibility and detailed control of the current library based approach for experts. The Particle file system will play a major role in this effort and will be enhanced with adaptive capabilities.

Acknowledgments

The work presented in this paper was partially funded by the European Community through the project CoBIs (Collaborative Business Items) under contract no. 4270 and by the Ministry of Economic Affairs of the Netherlands through the BSIK project Smart Surroundings under contract no. 03060.

References

[1] M. Beigl, A. Krohn, T. Zimmer, C. Decker, "Typical Sensors needed in Ubiquitous and Pervasive Computing". Proceesdings of INSS 2004, Tokyo, Japan, June 22-23. 2004, pp 153-158

[2] M. Beigl, A. Krohn, T. Zimmer, C. Decker, P. Robinson, "AwareCon: Situation Aware Context Communication", Ubicomp 2003, Oct. 12-15, Seattle, USA

[3] M. Beigl, T. Zimmer, C. Decker, "A Location Model for Communicating and Processing of Context", PUC Vol. 6 Issue 5-6, pp. 341-357, ISSN 1617-4909, 2002

[4] M. Beigl, T. Zimmer, A. Krohn, C. Decker P. Robinson. "Creating Ad-hoc Pervasive Computing Environments", Video at Pervasive 2004 in "Advances in Pervasive Computing", ISBN 3-85403-176-9, pp. 377-381, Vienna, Austria.

[5] C. Decker, M. Beigl, A. Krohn, "A File System for System Programming in Ubiquitous Computing", To appear in the proceedings of the ARCS 2005, March 14 –17, 2005, Innsbruck, Austria

[6] C. Decker, M. Beigl, A. Krohn , P. Robinson, T. Zimmer, J. Ma, "A Three-Tier Architecture for Location Presentation", Adjunct Poster Proceedings of Ubicomp 2004, September, 7-11, 2004, Nottingham, UK

[7] C. Decker, M. Beigl, A. Eames, U.Kubach, "DigiClip: Activating physical documents", IWSAWC 2004, Tokyo, Japan, Proceedings of the IEEE ICDCS 2004, pp 388-393

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System architecture directions for network sensors", ASPLOS 2000, Cambridge, November 2000.

[9] J. Hill, P. Bounadonna, D. Culler, "Active message communication for tiny network sensors", In the Proceedings of INFOCOM, 2001.

[10] A. Krohn, M. Beigl, C. Decker, P. Robinson, T. Zimmer, "ConCom – A language and Protocol for Communication of Context", Technical Report ISSN 1432-7864 2004/19

[11] A. Krohn, T. Zimmer, M. Beigl, "Enhancing Tabletop Games with Relative Positioning Technology", Advances in Pervasive Computing, Oesterreichische Computer Gesellschaft, Wien 2004.