# Experiences and Failures from Two Decades in Embedded System Design

**Micheal Beigl**
Karlsruhe Institute of Technology
TECO / Pervasive Computing
Karlsruhe, Germany
michael@teco.edu

**Matthias Berning**
Karlsruhe Institute of Technology
TECO / Pervasive Computing
Karlsruhe, Germany
berning@teco.edu

**Matthias Budde**
Karlsruhe Institute of Technology
TECO / Pervasive Computing
Karlsruhe, Germany
budde@teco.edu


*UbiComp/ISWC'17 Adjunct* , September 11–15, 2017, Maui, HI, USA
ACM 978-1-4503-5190-4/17/09.
https://doi.org/10.1145/3123024.3124393

## Abstract
Embedded hardware platforms that carry sensors and actuators play a major role in many Ubicomp applications. With the advent of DIY platforms as Arduino, hardware development for embedded systems is in the reach of everybody today. These projects often build on low-fidelity systems or even prototypes and are often carried out by engineers with no strong background in electrical engineering. On the other end of the spectrum lie specialized or highly integrated platforms with much higher complexity. This paper summarizes some of the pitfalls one may encounter in both kinds of such multi-disciplinary projects and presents some general lessons that can be drawn from past experience. Lastly, we discuss the tradeoffs that come with different levels of fidelity and how user and environment may (and should) affect design decisions.

## Author Keywords
Failure and Errors; Systems Design; Embedded Systems; Hardware; Prototyping; Fidelity

## ACM Classification Keywords
B.m [Hardware]: Miscellaneous

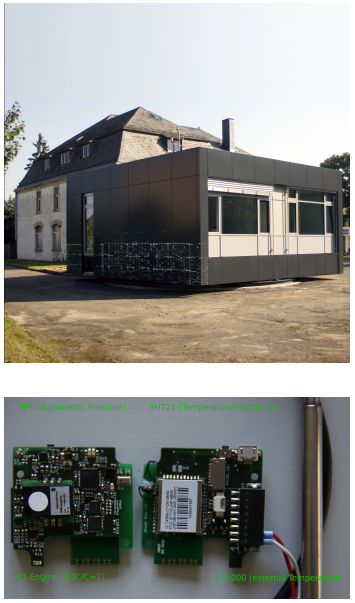

**Figure 1:** The *LOBSTER* (top), a modular, rotatable office space equipped with a variety of actuators e.g. to control heating and cooling, as well as sensors like the *jNode-Env* (below).

## Introduction

Projects that include hardware development are generally different from those projects that focus on software only. The physicality of the hardware makes it much slower to add, change or remove functionality in the system compared to software. In Ubicomp systems, things tend to become even more complex, as these systems are usually tightly connected to their deployment environment. In addition to the physical context, this also includes the users, as well as their interactions and social relationships. Usually these systems include a power supply, some form of (wireless) communication, as well as sensors and actuators to interface with the environment. Each of these has its pitfalls, as we describe later.

Hardware components are often well characterized and described down to their internal detail. Tools for various specific tasks in the process of developing hardware – from simulation, emulation, to design and testing – are available and comparable in function and use between vendors. This enables engineers to focus more on the solution of the task than dealing with possible incompatibilities of the system. In this respect, hardware design and development seems to be much less error prone than software. Still, hardware design is also more challenging, simply because it is bound to interaction with physicality – which can change drastically during the lifetime of a system. On the other hand, a software system uses standardized interfaces to the world, which dramatically reduces possible problems and errors.

One established method to deal with complex systems is iterative development with small steps and interleaved testing. One of the more prominent examples of this approach is User Centered Design (UCD), which incorporates the user as early as possible in the process [7]. The key component are prototypes that can demonstrate important features for testing, based on the current state of development. Usually, their complexity and scope increases from low-fidelity mockups, to high-fidelity systems that are almost fully functional [4].

Nowadays, agile software development is already characterized by small iterations, short sprints and countless “code-compile-test” cycles. In hardware projects, this is only possible for small systems on the breadboard level, early in the development process. Luckily, Arduino and co. provide an easy entry and hardware is getting cheaper, but scaling up still requires significant investment in testing equipment, time and especially components for each instance in each iteration. In general, this increases the pressure to integrate more changes in each step and subsequently, prolongs the duration of the cycles. In addition, early stage hardware prototypes are not robust enough, are not optimized for power consumption or lack sufficient integration for real world deployment and testing.

This paper reports selected errors made and some pitfalls of such projects in the Ubicomp domain, which we experienced in the past in an anecdotal way. Subsequently, we pick up on these examples to discuss tradeoffs that come with different levels of fidelity and deployment environments.

## 1) Unnoticed change in power supply

In the *LOBSTER* project [6], we equipped a novel type of building (see Figure 1) for human comfort studies with a high number of high-precision sensors. Sensors are integrated into walls, floor and ceiling and also worn on the body of the inhabitants to measure a variety of environmental conditions and how the inhabitants perceive them. The system was based on the jNode sensor node [5], which was initially developed for battery powered operation. After

being tested successfully in our lab, we installed them in the building. While in the lab the data was of high quality, in the building we got erroneous sensor readings and system crashes. What had happened? As the conditions were (almost) the same this was very odd. In the end we found that one minimal factor had changed: The USB power supply. In contrast to our lab supply, it exhibited a small noise in the voltage that lead to the described effects. The external supply was added for stationary sensors to lower the maintenance effort for the researchers. In the end, it made their experiments harder until we could fix the bug.

*Take away message:* Even minor changes in the electrical operating conditions can influence the performance drastically.

## 2) Change in environmental conditions

In another project we had designed and developed a new family of low power wireless sensor system. The system had been tested in the lab and some outdoor conditions with very good measurement and transmission rates. In the field, the performance of the wireless data transmission of was not convincing. As the complete system, including power supply, had been tested beforehand, an explanation was difficult. It could have been the case that the transmission characteristics were rougher than in previous tests. But this could not explain the full effect and we searched for alternative explanations. In the end we found the decisive reason: The oscillator which determined the frequency for wireless communication varied too much because the spread of operating temperatures was higher than in the test settings.

*Take away message:* Any change in any physical condition can influence any piece of hardware in an unexpected way. Due to the complex interplay, testing is the only way to find such problems.

## 3) Uncertainty reduction causes uncertainty

In time and resource constrained hardware projects, the engineer tries to reduce the overall uncertainty of the system. This is especially important in the early phase, because changes in the design are getting more and more costly with increasing runtime of the project and especially with maturity of the system. In addition, the requirements from customers and partners are a moving target. One often-used way to mitigate this challenge is to modularize the system, so that later changes will only influence a single module, but not the whole system. Using hardware modules we were able to lower the uncertainty we had for the overall system design in several projects. We were able to proceed working despite of changes in parts of the system later on. Generally, this also facilitates testing of early prototypes, because of a low entry barrier and replaceable parts in the field. The approach comes at a price. First, it adds overhead on development time and thus cost, but this might be acceptable due to later savings. Second, the system is usually not so well integrated and larger than necessary. And third, the modules need to be connected. Any wire and connector is a primary source of error and issues, especially during longer operation time. Partially, this also holds for soldered wires which can break under mechanical stress. One of our designs was deployed on heavy machinery with a lot of vibration. This lead to several failures in early prototypes, resulting in several unnecessary iterations. Ultimately, the project was prolonged because of the modularization.

*Take away message:* Consider if the method to reduce uncertainty is really having the expected effect.

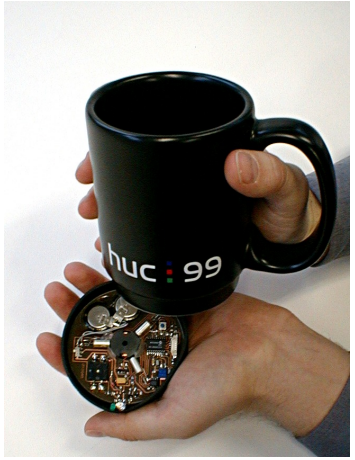


**Figure 2:** The *Mediacup*, an early IoT device with sensors to detect movement, temperature and liquid level of the cup [1]. The picture shows the final version from 1999 with wireless recharging.

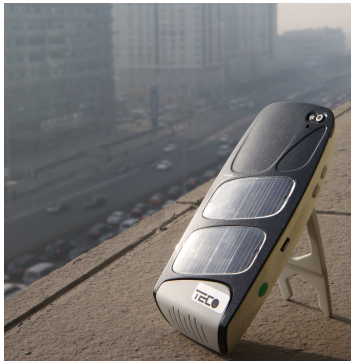

**Figure 3:** The *Envboard*, a polished external design which limited the internal sensors. (Image taken from [3]).

## 4) Lack of experience in specific detail

Hardware systems usually require embedding the electronics in housings. Those housings are specific for an application case. In one of our projects, a partner that was quite experienced with the application domain, designed such a housing for a rough outside use. In this case, an IEC IP69k housing was selected that can survive dust, submersion and even close-range high-pressure water. After deployment more and more of the devices failed, all of them due to moisture induced short circuits. Still, the housing itself was water-proof. What had happened? The device was not only exposed to outside water, it was also exposed to extreme temperature changes, which we did not test. These temperature changes lead to changes in the internal air pressure. Because the housing was not air-proof (which is very difficult to implement), water vapors enters slowly. With sudden temperature change from high to low, it cannot escape and condenses on the inside of the housing and creates unintended connections in the electronics.

*Take away message:* Try to get as much input from as many experts as possible, not to overlook some important detail. One expert might not be enough.

## 5) Long term user behavior may lead to fundamental changes in system design

One of the first everyday objects where we integrated electronics was the MediaCup in 1998 [1] (see Figure 2). It was a research prototype to carry out living lab studies and collect experience regarding technology and application potential of computation in everyday objects. The design was guided by user interviews and some initial studies to retrieve general requirements. The cup was able to collect data about the frequency of coffee intake, temperature of the liquid and, location of the cup. The applications were the warning of too hot coffee, drinking behavior statistics, but also meeting room scheduling – due to the correlation between cup movements and meeting room occupation. The first version of the cup contained movement sensors, a weight and a temperature sensor, a wireless communication module and a battery. In our study everything went fine – for the first three months. Then, more and more cups dropped out.

What had happened? After three months, the batteries were empty. Despite the fact that every user found it a nice gadget, nobody wanted to take the burden to change batteries – although it was quite simple to do. With the second version we addressed this topic by revising the design and implementing a wireless recharging unit and infrastructure in the lab to ensure continuous operation. The runtime on a single charge was much shorter, but recharging was much easier. Still, this redesign took the most effort in the overall system implementation.

*Take away message:* Long term effects may influence major design decisions.

## 6) Unclear requirements should be approached cautiously

In the *Envboard* project [2] we were confronted with a rather vague description of requirements, along with the use case of mobile environmental monitoring. Despite of our recommendation for an iterative approach we were expected to deliver a high-fidelity product-like platform. This started with the challenge of integrating the electronics and sensors into already finished and fixed housing design (see Figure 3). This included aspects that did not make a lot of sense, as e.g. the solar cells in the surface. While they were too small to make a contribution to the energy budget, they significantly prolonged the manufacturing process: The delicate cells had to be soldered by hand, many breaking during

**Take-away Messages**

1. Even minor changes in the electrical operating conditions can influence the performance drastically.
2. Any change in any physical condition can influence any piece of hardware in an unexpected way. Due to the complex interplay, testing is the only way to find such problems.
3. Consider if the method to reduce uncertainty is really having the expected effect.
4. Try to get as much input from as many experts as possible, not to oversee some important detail. One expert might not be enough.
5. Long term effects may influence major design decisions.
6. Even if you are only responsible for the electronics, challenge the requirements and clarify the goal for each iteration

assembly or installation. Other problems stemming from the unclear scope, were late additions to the list of components, the need to substitute a sensor late in the project and a long list of change requests concerning the interaction design. As a result, the project exploded in time, scope and cost and yielded some undesirable outcomes, such as limited battery lifetime or a rather complex firmware. In the end, we created an interesting platform for environmental research. On the one hand, it satisfied the requirements we were given. On the other hand, it could have been more suitable for the original application case if it had been reduced in functionality and tested iteratively with the involvement of target users.

*Take away message:* Even if you are only responsible for the electronics, challenge the requirements and clarify the goal for each iteration.

## Discussion and Conclusion

Looking back at the last two decades of embedded system design, the number of applications for Ubicomp are steadily increasing. Our lab has worked on a variety of projects, integrating computing systems in everyday objects, heavy machinery, buildings and novel sensing appliances, only to name a few. Although the electrical engineer aspects were challenging in terms of miniaturization and power consumption, most issues were connected to the application environment. This highlights the close connection of these systems to the deployment context, which is not limited to environmental conditions, but also includes users, customers and other researchers. With such complex dependencies, it is impossible to specify every aspect in advance. Therefore iterative development, deployment and testing is unavoidable. This requires suitable prototypes, which should focus on a specific aspect for each iteration. Still, there are several interdependent design tradeoffs, which must be considered: e.g. power supply, (wireless) communication, integration (size, complexity), robustness, user interface (and UX) and development time and cost. Shortcomings in one aspect might lead to iterations with little results – failure in robustness leads to bad UX; failure in UX leads to bad battery life; failure in power supply leads to broken communication – as we have illustrated in the examples above. This highlights the interdisciplinary nature of embedded system design for Ubicomp applications which cannot be covered by computer scientists or electrical engineers on their own. On the other hand, there is an ever increasing number standard platforms, Software/Hardware Development Kits (S/HDK) and tools with a low entry barrier, which enable domain experts to come up with their own solutions. Overall, there are still challenging projects ahead and we are looking forward to the failures and lessons in the next 20 years.

## REFERENCES


1. Michael Beigl, Hans-W Gellersen, and Albrecht Schmidt. 2001. Mediacups: experience with design and use of computer-augmented everyday artefacts. *Computer Networks* 35, 4 (2001), 401–409.
2. Matthias Budde, Matthias Berning, Mathias Busse, Takashi Miyaki, and Michael Beigl. 2012. The TECO Envboard: A mobile sensor platform for accurate urban sensing – and more. In *Ninth International Conference on Networked Sensing Systems (INSS'12)*. IEEE, 1–2.
3. Matthias Budde, Rayan El Masri, Till Riedel, and Michael Beigl. 2013. Enabling low-cost particulate matter measurement for participatory sensing scenarios. In *12th International Conference on Mobile and Ubiquitous Multimedia (MUM'13)*. ACM, 19.
4. Michael McCurdy, Christopher Connors, Guy Pyrzak, Bob Kanefsky, and Alonso Vera. 2006. Breaking the fidelity barrier: an examination of our current

characterization of prototypes and an example of a mixed-fidelity success. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 1233–1242.

5. Philipp M Scholl, Kristof Van Laerhoven, Dawud Gordon, Markus Scholz, and Matthias Berning. 2012. Jnode: a sensor network platform that supports distributed inertial kinematic monitoring. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*. IEEE, 1–4.
6. Marcel Schweiker, Sabine Brasche, Maren Hawighorst, Wolfgang Bischof, and Andreas Wagner. 2014. Presenting LOBSTER, an innovative climate chamber, and the analysis of the effect of a ceiling fan on the thermal sensation and performance under summer conditions in an office-like setting. In *Windsor Conference: Counting the Cost of Comfort in a Changing World*, Vol. 8.
7. Karel Vredenburg, Ji-Ye Mao, Paul W Smith, and Tom Carey. 2002. A survey of user-centered design practice. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 471–478.