Towards an Experimental Autonomous Blimp Platform

Axel Rottmann* Matthias Sippel‡ Thorsten Zitterell* Wolfram Burgard* Leonard Reindl‡ Christoph Scholl*

*Albert-Ludwigs-University
Department of Computer Science
Freiburg, Germany
{rottmann, tzittere, burgard, scholl}@informatik.uni-freiburg.de

‡Albert-Ludwigs-University
Department of Microsystem Technology
Freiburg, Germany
{sippel, reindl}@imtek.uni-freiburg.de

***Abstract*— In this paper, we present the design of an autonomous indoor blimp. We describe the individual low-weight components of an embedded system including actors and sensors which achieve a total weight of less than 200 grams. Both, the modular hardware components and the use of generic hardware interfaces facilitate the adaption to different actor and sensor systems. Similar to the flexibility of the hardware components, the user benefits from a generic software framework. Here, an on-board Linux operating system and device driver interface ease the implementation of robotic applications and control tasks. The main challenge of designing a small blimp in regard to autonomous operation is the efficient interplay of the individual software and hardware components. In this work, we show approaches of how to build up such a system which can easily be applied to other robotic systems with constraints in weight and available space. We evaluate the performance of the components and demonstrate their integration in a reinforcement learning setting.**

***Index Terms*— reinforcement learning and navigation**

I. Introduction

The development of small size autonomous flying vehicles represents one of the current frontiers of research in mobile robotics. In this context, aerial blimps have the advantage that they operate at low speed, do not spend energy to keep their position, and are not overly sensitive to control errors compared to other flying vehicles. On the other hand, they are sensible to outside influences like air flow and are subject to a three dimensional motion model with translations and rotations. Therefore, they are a common platform to evaluate robotic algorithms for autonomous flight and navigation.

In this paper, we describe how such a blimp system including an embedded microsystem and software framework can be build up. In this regard, we aim to keep the system as small and agile as possible in order to operate indoors.

This size constraint also limits the possible weight of the blimp. With a maximum possible payload of 200 grams in our configuration the system must include the gondola, actors, sensors, engine control unit, system unit, and battery. Due to these characteristics, blimps are not only interesting for robotics but also for microsystem technology as the attached devices should be both small and efficient. Such performance aspects are evaluated in the experiments.

In order to demonstrate the operational reliability of our blimp system (see Figure 1) we describe a reinforcement learning approach to control the altitude. Here, the blimp autonomously learns the optimal policy from scratch to stabilize its height after it has been switched on.

Fig. 1. This images shows the completely build blimp with a length of 1.8 m and a diameter of 0.8 m.

This paper is organized as follows. After discussing related work in the following section, we will describe in detail our blimp system including the hardware components and the software framework in Section III. Afterwards, we will illustrate a controlling approach based on reinforcement learning in Section IV. In Section V, we will present our first experiences with our blimp system and a practical application which integrates the components in a reinforcement learning setting. Finally, Section VI concludes the paper.

II. Related Work

The scientific research on autonomous blimps has constantly been increasing over the last few years. Most of these blimps are large-scale systems with a payload of several kilograms. Hygounenc *et al.* [4] provided a good overview of how to construct such an outdoor blimp and how to apply it to some fundamental robotic tasks. They illustrated how to control several flight phases from takeoff to landing based on the non-linear system dynamics. Jung and Lacroix [5] presented an approach to simultaneous localization and mapping using low altitude stereo-vision. Also Kim *et al.* [6] illustrated an approach to point to point navigation of an outdoor blimp. Compared to these papers, the blimp described in this paper has been designed for indoor scenarios. It is much smaller than the outdoor systems and has a total weight including the envelope of only 420 grams. Therefore, the challenge is to integrate low-weight sensors in an embedded microsystem

Fig. 2. This image shows the gondola with two rotors for pitch and thrust control. They can be rotated by 180 degrees.

and to provide a software framework for complex tasks like autonomous navigation and localization.

Furthermore, blimps have been studied in various contexts. Varella Gomes and Ramos [14] described the physical principles of airship operations often used to design a controller. Zhang and Ostrowski [16], for instance, used a vision-guided blimp in combination with a PID controller. Rao *et al.* [9] proposed a fuzzy logic controller which was based on the dynamics of the vehicle. Whereas Wyeth and Barron [15] presented a low-level reactive controller, Fukao *et al.* [2] illustrated an image-based tracking control for an indoor blimp. Moreover, reinforcement learning has successfully been applied to learn the control policy of an indoor blimp. Ko *et al.* [7] used Gaussian Processes to estimate the residuals of the dynamics and learn to control the yaw and the yaw rate. Other applications such as indoor navigation with one camera were presented by Green *et al.* [3] and a surveillance system by Kukao *et al.* [1].

III. Blimp System

The characteristics of an aerial robot result in various restrictions considering the assembled hardware. For a blimp system, the higher the volume of the envelope and therefore the ascending force, the higher the possible payload. However, a bigger envelope makes it less applicable for indoor navigation and results in more inertial flight characteristics. For our blimp system, the goal was to minimize both the size of the blimp and consequently the weight of the needed hardware equipment. The challenges under such constraints are to find a trade-off between agility, range, and energy consumption, and to develop appropriate algorithms for small aerial robots which are highly sensitive to outside influences (e.g., air flow).

In addition, high flexibility and maintainability of the software components have been a matter of particular interest for us. The embedded system has to be powerful enough to process on-board calculations for autonomous flights while also allowing to transparently perform these computations on external computers. Basically, the same piece of control software must be executable on-board and also on an external development system without code modifications.

A commercial 1.8 m blimp envelope [8] builds the basis for our blimp as depicted in Figure 1. The blimp is equipped with three motors. One motor is mounted in the tail fin to control the yaw. The other two are mounted on each side of the gondola to control the pitch and thrust. The latter two are attached to a shaft which can be rotated up to 180 degree by a servo (see Figure 2). The gondola includes all hardware components used to control the speed of the motors using PWMs, to change the position of the servo, and to process the sensor data. The weights of all components are listed in Table I.

TABLE I

Weights of the individual components.

Component	*Weight [g]*
Envelope	230
Gondola	15
Fins, Propellers, & Cables	75
Gumstix	8
Gumstix Peripheral Interface Card	21
Blimp Engine Control Board	7
Inertial Measurement Unit	10
USB hub & WLAN	16
Battery	33
Sum	415

A. Hardware Equipment

One of the main aspects in the design phase of our system was to keep the flexibility of the individual system components to a maximum. Therefore, we separated the system into several main parts, such as: the system core unit, the peripheral interface card, and the motor control and sensor board. To obtain this flexibility all parts can be exchanged and adapted to other systems and tasks. For instance, we also plan to use this hardware on other flying robots like a small-scaled helicopter. The main components are shown in Figure 3 and described in the following sections.

1) System Core Unit and Communication: The core unit of the blimp is an Intel XScale PXA270 based system-on-a-chip (SoC), or more precisely a *Gumstix verdex XL6P* board with 600 MHz and 128 MB RAM. An on-board 32 MB flash memory serves as storage for the Linux operating system. The SoC provides various low-level communication buses and protocols (e.g., I^2C, four independent UARTs, SPI) as well as General Purpose Inputs/Outputs (GPIOs) and a USB host port to access peripheral devices (e.g. WLAN stick). The Gumstix board provides these circuit points over separate connectors (Figure 3). The 60-pin connector is used to plug a self-assembled peripheral interface card described in the next section.

2) Gumstix Peripheral Interface Card (GPIC): The Gumstix Peripheral Interface Card (GPIC) was designed for an easy exchange of the motor control and sensors. This provides the opportunity to use our hardware system to several other robots. The GPIC forms the interface between the system core unit and the actors and sensors. Referring to this point of view the GPIC needs various connectors to handle different external modules. Therefore, the GPIC provides:

- 8 ports for generic devices communicating via UART or SPI,
- 4 connectors for I^2C devices,
- charge control and power supply for the complete system,
- sensor to monitor battery state and power consumption,
- 2 additional UARTs for further purposes, and
- one USB connector for Gumstix' system maintenance and a second one to attach common USB devices.

In order to apply our system on different robots eight ports for generic devices are provided. These ports communicate

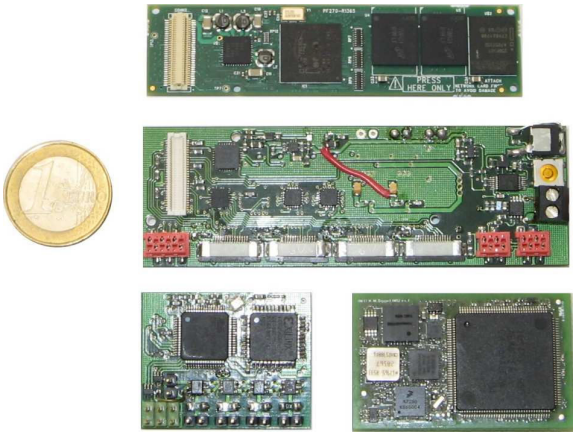

Fig. 3. The main components of the hardware top down: Gumstix board, Gumstix Peripheral Interface Card (GPIC), and right at the bottom from left to right the Blimp Engine Control Board (BECB) and the Inertial Measurement Unit (IMU). The GPIC is attached via the 60-pin connector to the Gumstix board.

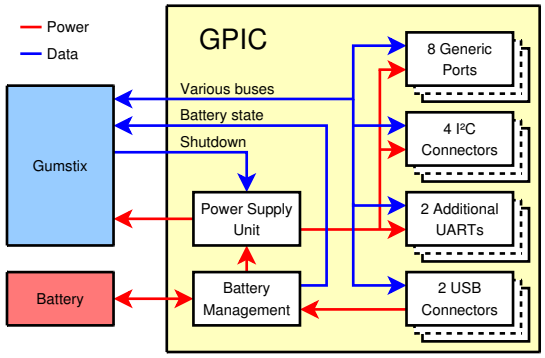

Fig. 4. The Gumstix Peripheral Interface Card (GPIC) consists of a power supply and battery management unit. Several ports are provided to attach actors and sensors.

via UART or SPI and hence can be used for miscellaneous applications. In our system, we attached a motor control board and an inertial measurement unit. In general, this provides the opportunity to easily extend the system with sensors or actors or transfer it to other robots.

Our system is powered by one 3.7 V lithium polymer battery. The power supply unit for the complete system is placed on the GPIC and provides multiple voltages (3.3 V, 5 V, and 6 V). A power management unit offers several advantages. First, it prevents total discharge of the battery (battery management) and provides different stabilized voltages for other modules (power supply unit). Second, the system can be extended by additional boards and sensors without taking the power requirements into account. Finally, the electric current, the current over time, and the actual battery voltages are monitored. A systematic layout of the GPIC including the various connections and ports is shown in Figure 4. In our current system, we use a 3.7 V/1500 mAh battery which offers an operating time-span of 50 minutes with an engine utilization of 80%.

3) Blimp Engine Control Board (BECB): To have a wide range of use we also constructed an external Blimp Engine Control Board (BECB). In case the GPIC should be applied to other robots it can easily be replaced with a new actor-specific control board attached to one port for the generic devices. The core unit of the BECB is a microcontroller which handles simple string commands from the system unit and controls the percental power of the motors and the setting of the servo.

Considering the engines, their power depends on the supply voltage. As the supply voltage decreases during operation, the thrust of the motors would vary for identical high-level command. Therefore, we determine a power correction factor based on the current battery voltage and the characteristics of the motor to guarantee a constant thrust of the engine. Consequently, the user can expect the same physical effects of an action independent of the system state.

4) Inertial Measurement Unit (IMU): A fundamental requirement for autonomous navigation of robots is the ability to localize itself. In the case of flying robots, information about its flight attitudes, namely the Tait-Bryan angles roll, pitch, and yaw, are required. These flight attitudes can be obtained by an inertial measurement unit (IMU). In addition, we use the data of the IMU as input for predictive filters to estimate the current position based on the prior position and to increase the accuracy of the absolute position measurement. In this regard, GPS-based localization is not applicable as our blimp system is targeted on indoor environments.

Our self-made IMU is a highly integrated, planar, strapdown system based on sensors in MEMS technology. To achieve our complete state space with adequate accuracy we measured the following physical units: acceleration, angular rate, magnetic field, and pressure. The acceleration is measured in three axis with four different selectable sensitivities. With these sensitivities one can determine the static sensor offset at zero gravitation to perform an auto-calibration. The pressure sensor is used to obtain changes in the altitude.

5) Additional Sensors: In addition to the sensors of the IMU we equipped our system with an ultrasonic sensor and a miniature USB camera. Both are downward-facing mounted at the bottom of the gondola to measure the altitude of the blimp and to prepare the application for a visual navigation system. The sonar sensor module SRF10 is attached to the I^2C bus and has a weight of 8 grams. The measurements are integrated by means of a Kalman filter which sequentially estimates the altitude and the vertical velocity of the blimp. The attached USB camera has a weight of 9 grams and provides JPEG pictures with a resolution of up to 640×480 pixels.

B. Software Architecture

This section considers the software environment of our blimp system. Similar to the hardware framework, our aim was to apply standard software which provides easy maintainability, configuration, and flexibility with regard to the operating system, hardware access, and communication. Figure 5 shows the interplay of the software framework, i.e., of the operating system, drivers, and interprocess communication.

1) Operating system: Despite its small dimensions the Gumstix board runs a full-fledged Linux operating system which provides interfaces to access the connected hardware and eases the configuration of the blimp as a wireless client

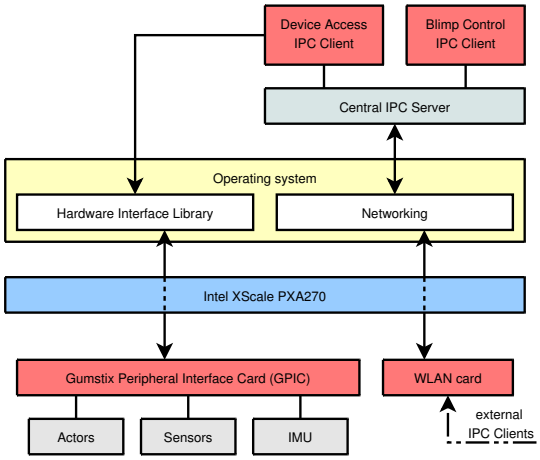

Fig. 5. The Figure shows the individual core components of the blimp system. The operating system including the interface library provides higher-level access for the IPC framework and application processes.

and of the TCP/IP network. The latter allows easy exchange of application software during operation. Although the operating system provides generic access to our hardware, i.e., I^2C, UART, and SPI, we implemented a hardware interface library to provide higher-level access to the GPIC and the connected sensors and actors.

2) Hardware Interface Library: The hardware interface library is responsible for correctly controlling the sensors and actors attached to the GPIC and providing high-level routines to access them. The library provides a generic device type and routines to send and receive data once the programmer has configured the communication port and protocol. Upon this generic device communication layer we implemented more specific routines to access actors and sensors. New sensors and actors can be easily attached to the GPIC and configured during run-time if needed. The generality of the library makes it easily applicable in other robotics systems.

3) Interprocess Communication: As mentioned above the Linux operating system provides all necessary support for TCP/IP networking. In order to control the blimp we use the interprocess communication (IPC) framework of [12] based on TCP/IP. The IPC framework allows to separate the processes which accesses the actors and sensors of the blimp and which actually control the blimp. In our configuration, a central server process runs on the blimp and manages the communication between several IPC client processes. One IPC client process accesses the devices by the hardware interface library and has to be executed on the blimp. However, other controlling client processes can either be executed on the local system for autonomous flights or externally. Another reason for us to use the IPC framework is its platform independence, i.e., it can be compiled for different architectures (e.g. ARM on the blimp) and takes care of marshaling and unmarshaling data.

IV. Reinforcement Learning as Online Application

In this section, we introduce how the blimp described so far can be used as an autonomous platform. A fundamental requirement for autonomous navigation in indoor environments are efficient controllers. Therefore, we consider the task of stabilizing the blimp at a given goal altitude without knowing the specific dynamics of the system or any parameter of the environment. In practice, height control is already a complex task as blimps are very sensitive and the behavior highly depends on outer influences like payload, battery level, temperature, and air flow. In the past, blimp controllers are mainly based on standard controllers, e.g., PID or fuzzy logic, or on the dynamics of the system. The disadvantages of such approaches are the direct dependency on the parameter of the dynamics. The optimal behavior cannot be guaranteed if the conditions of the environment are not constant. In general, it is hard to determine a globally suitable policy applicable to several conditions. Therefore, we seek to learn the best policy from scratch for the current conditions while the blimp is in operation. If the real system learns online without requiring any pre-defined controller or parameter, it is robust against outside influences and independent of the current conditions. For this task, we suggest the Monte Carlo learning approach using Gaussian Processes to reinforcement learning as proposed in [11].

Reinforcement learning [13] is based on the idea that an agent interacts with a potentially unknown environment and gets rewarded or penalized according to the actions it performs. In this setting, the agent receives rewards for actions that are beneficial in certain states for achieving a long-term goal. The agent thus seeks to behave in a way that maximizes its numerical reward. The goal is to determine a policy function π, which maps each state to an action, so that the overall reward is maximized.

The used learning approach apply the Monte Carlo method to reinforcement learning. This has the advantage to learn directly from experience while interacting online with a completely unknown environment. This enables us to learn without prior knowledge and also in situations in which no simulation environment is available. Formally, we seek to learn the state-action function $Q(s,a) : S \times A \rightarrow \mathrm{R}$, representing the expected future reward when selecting action a in state s. In our case, the states S consist of the altitude and the vertical velocity of the blimp and the actions A represent the vertical thrust of the propellers. To learn this function we generate episodes $e_1, \ldots, e_N$ from a sequence of measurements $(s_1, a_1), \ldots, (s_T, a_T)$ obtained while the blimp is moving through the environment. For each state-action pair (s_t, a_t) in the sequence an episode $e_t = \big((s_t, a_t), \ldots, (s_{t+p}, a_{t+p})\big)$ is generated consisting of the p successor states. The length p of an episode is defined by a factor γ. An episode ends if the factor γ^p is smaller than a given threshold of 0.1. The expected long time reward for a state-action pair is finally calculated by

$$R(s_t, a_t) \quad = \quad \sum_{i=0}^{p} \gamma^i r_{t+i}\,, \tag{1}$$

where $\gamma \in [0, 1)$ is a discount factor and r the immediate reward received when executing action a in state s. Finally, the best policy is given by the maximum over the Q-value

function

$$\pi(s) = \arg\max_a Q(s,a)\,. \tag{2}$$

Another advantage of this learning approach is the usage of Gaussian Processes [10] to approximate the Q-function. With this framework, we are able to retain our observed data during online learning and to predict the Q-values for previously unseen states-action pairs. Therefore, not all states have to be explored during the learning task and good policies can be estimated even after a few learning steps. Furthermore, in contrast to common Monte Carlo learning approaches, no discretization of the state and action space have to be made. This allows it to learn in continuous spaces which leads to no discretization errors and better policies are expected. For details about this learning approach, we refer to the work of Rottmann *et al.* [11].

V. Experiments and Evaluation

The following experiments describe our first experiences with our blimp system. In order to get an overview of the hardware performance we measured achievable throughput rates to communicate with sensors and actors. Moreover, we show that the assembled hardware and software framework are applicable for complex tasks and evaluate the reinforcement learning approach proposed in Section IV.

A. Performance Measurements

The progress of learning the behavior of an autonomous robot in an unknown environment highly depends – apart from the implemented models and algorithms – on the characteristics of the attached actors and sensors and the latency of actions and their effects. In the same way, the performance of filter algorithms applied in dynamic systems is influenced by the sampling rate of measurements. In order to get an overview of the characteristics of the hardware equipment we performed experiments considering throughput, sample rates, and response times of the assembled devices.

The first experiment was to evaluate the direct communication performance between our system core unit and the motor control unit. In our current configuration, the following steps are performed to set the speed of a single motor: calling the corresponding function of the hardware interface library, creating a string command, selecting a port by setting the corresponding GPIO address lines, and sending the string command to the motor control unit via UART. The limiting element considering performance in our configuration is the 115200 Baud UART bandwidth and the computation performance of the attached hardware. As we have a maximum length of six bytes for each command and two bytes for an acknowledgment message we could transmit approximately 1440 commands per second to the motor control unit. Due to the computation overhead we achieve a value of approximately 607.5 $\pm$ 9.9 commands per second, in practice. However, we are convinced that this rate can be increased by optimizing the code and the communication protocol.

In another experiment, we additionally considered the IPC performance, i.e., motor control commands are initiated by an external Blimp Control IPC client and the local Device Access IPC client calls the function of the hardware interface library. In this scenario, we get an additional overhead for the TCP/IP communication via the wireless link and can transmit on average 187.8 $\pm$ 7.0 commands per second.

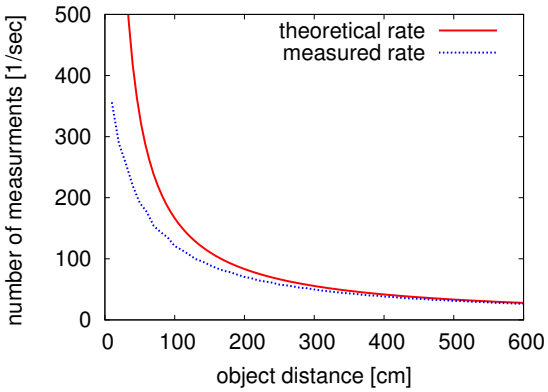

Fig. 6. Sonar sample rate: The curve for theoretical rate is derived from the sonic speed $v = 333\,\mathrm{m/s}$ and does not take the system overhead into account. The measured rate reflects the performance of the system including the hardware and software.

Due to the specification of our ultrasonic sensor, the latency between triggering a measurement and availability of the result is at most 65 ms for a maximum distance range of 11 m. However, this is only valid for the worst case. If the distance s_{obj} to the nearest object in the cone of the sensor is smaller, the measurement result is available earlier and more measurements can be performed in the same time interval. Additionally, the sensor can be reprogrammed to operate in a lower maximum distance range s_{max} which would result in better sampling rates. If we consider a distance $s = \min(s_{\text{obj}}, s_{\text{max}})$ to an object and a sonic speed v, the theoretical number of samples per time interval T is $n(s) = \frac{T \cdot v}{2s}$. Figure 6 illustrates the theoretical and measured sample rate. The smaller the distance s the more the overhead of the operating system and device drivers become visible. For a typical indoor operating range of 6 m we get a minimum sample rate of 27 measurements per second.

We can summarize that the achieved rates should be sufficient for most applications. However, our embedded system and software framework is not limited to a blimp but can also be applied in other aerial robots. Considering an autonomous helicopter, for example, the latency should be minimal between retrieving a sensor value, computation, and performing an action. In this case, the communication between external actors and sensors and the system core unit could be switched to the SPI protocol which achieves much higher data throughput.

B. Online Learning

This experiment demonstrates that our blimp system can learn to control the altitude based on the experience gathered during moving in the environment. We used the reinforcement learning approach extended with Gaussian Processes for approximating the Q-function. It also illustrates that our

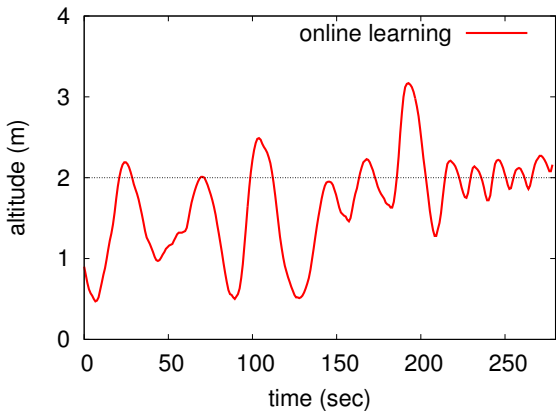

Fig. 7. Evolution of the altitude during the learning progress to stabilize at 2 m.

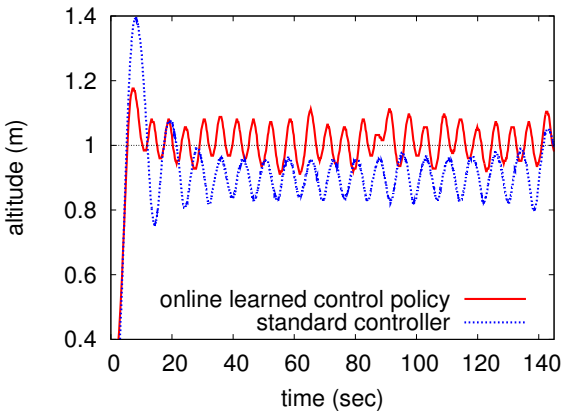

Fig. 8. Progress of applying the online learned control policy and a PID controller while stabilizing at 1 m.

approach efficiently learns on a completely model-free, real system with unknown dynamics. To perform this experiment we run the blimp in a factory building with a vertical exploration space of 5 m. Figure 7 plots the altitude movement of the blimp during the learning task. As can be seen, at the beginning the blimp is exploring the states and in the course of time the blimp is more and more exploiting the current policy and finally stabilizing at the goal altitude of 2 m. In an additional run as depicted in Figure 8 we compare the policy of our learning approach with a standard controller. We used a controller based on both the difference to the goal altitude and the vertical velocity which leads in our setting to a much better performance than a standard PID controller. Anyway, the standard controller behaves suboptimal as the current environmental conditions were unknown while the parameters were established. Otherwise, the policy learned for the current conditions stabilize the blimp much better at the given goal altitude of 1 m.

VI. Conclusions and Future Work

In this paper, we presented a powerful, low-weight, and embedded microsystem which is applicable for autonomous blimps and other systems with size and weight constraints. The modular structure of the hardware components and the use of generic hardware interfaces and common bus protocols facilitate the adoption of other actor and sensor systems. The user benefits from a flexible software framework consisting of a common Linux operating system and peripheral device drivers. Finally, we demonstrated the capabilities of our blimp system in the context of a reinforcement learning task.

Despite these encouraging results, there are several aspects that warrant future research. For example, we plan to evaluate the use of dead reckoning algorithms for our inertial measurement unit and consider further localization techniques for aerial robots. In this context, the challenge is to integrate appropriate algorithms which allow accurate position estimations. Using localization techniques we also plan to extend our reinforcement learning approach for 3D worlds. Additionally, we plan to investigate how the learning system reacts if the environmental parameters change abruptly during online learning.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) within the Research Training Group 1103.

References

[1] T. Fukao, K. Fujitani, and T. Kanade. An autonomous blimp for a surveillance system. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[2] T. Fukao, K. Fujitani, and T. Kanade. Image-based tracking control of a blimp. In *Proc. of the IEEE Conf. on Decision and Control*, 2003.

[3] W. Green, K. Sevcik, and P. Oh. A competition to identify key challenges for unmanned aerial robots in near-earth environments. In *Proc. of the Int. Conf. on Advanced Robotics (ICAR)*, 2005.

[4] E. Hygounenc, I-K. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project at laas/cnrs: Achievements in flight control and terrain mapping. In *International Journal of Robotics Research*, 2003.

[5] I-K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Int. Conf. on Computer Vision*, 2003.

[6] J. Kim, J. Keller, and V. Kumar. Design and verification of controllers for airships. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[7] J. Ko, D. Klein, D. Fox, and D. Hähnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[8] RCGuys Radio Control Models http://www.rcguys.com.

[9] J. Rao, Z. Gong, J. Luo, and S. Xie. A flight control and navigation system of a small size unmanned airship. In *Proc. of the IEEE Int. Conf. Mechatronics and Automation*, 2005.

[10] C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[11] A. Rottmann, C. Plagemann, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[12] R. Simmons and D. James. *Inter-Process Communication: A Reference Manual (for IPC Version 3.6)*, 2001.

[13] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[14] S. Varella Gomes and J. Ramos. Airship dynamic modeling for autonomous operation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[15] G. Wyeth and I. Barron. An autonomous blimp. In *Proc. of the IEEE Int. Conf. on Field and Service Robotics*, 1997.

[16] H. Zhang and J. Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1999.