Sensor Proccessing and Behaviour Control of a Small AUV

Chapter · October 2007

DOI: 10.1007/978-3-540-74764-2_51 · Source: DBLP

4 authors, including:

Jan Christian Albiez
Kraken Robotik GmbH, Bremen, Germany
107 PUBLICATIONS **1,096** CITATIONS
SEE PROFILE

Jochen Kerdels
FernUniversität in Hagen
36 PUBLICATIONS **131** CITATIONS
SEE PROFILE

Frank Kirchner
Universität Bremen
313 PUBLICATIONS **2,696** CITATIONS
SEE PROFILE

Sensor proccessing and behaviour control of a small AUV

Jan Albiez, Jochen Kerdels, Sascha Fechner, and Frank Kirchner

DFKI Robotics Lab, Robert-Hooke-Strasse 5, 28259 Bremen, Germany
jan.albiez@dfki.de

Abstract. This paper presents the design of the electronics, the sensor data processing scheme and the implementation of the behaviour based control for a small autonomous underwater vehicle (AUV). The robot has a volume of 12dm^3 and is used as a demonstration vehicle for underwater ai-technologies. The thruster configuration allows the μAUV to hover and its active light sensors enable obstacle detection abilities. The control features several behaviours like obstacle detection and depth sensor calibration and is completely autonomous. An ATMega128 functions as onboard CPU and due to the limited computing power we use a special scheduling algorithm which also acts as the behaviour arbiter.

1 Introduction

To present the possible applications of artificial intelligence in underwater robotics and offshore technology on the 2007 CeBIT fair in Hannover/Germany, we used an aquarium (lwh 3x1x0.8m) with a mockup of a future underwater production facility. To enhance the attraction of this demontration to the fairs audience we wanted to use two completly autonomous underwater vehicles (AUV), which are small enough to navigate inside the aquarium. Because of the required small size of these vehicles we named them μAUV, and they are at the moment the smalles full autonomous AUVs worldwide.

For the μAUV the traditional approaches for controlling small scale AUVs like the Serafina [2] can't be used. The pressured space in the μAUV is to small for the hardware components needed to apply higher behaviours and sensor systems. Therefore we had to develop a high integrated sensor-actor-proccessing scheme to control the μAUV.

During the mechanical construction of this vehicles we encountered several problems which only arrise when building small scale underewater vehicles. First of all is the surface tension. For bigger vehicles (e.g. [1]) surface tension is not a problem, but on the size of the μAUV the buoyancy of several gramms, generated by the surface tension, is a serious problemfor the actuators. We had to keep the area exposed to the surface as small as possible to prohibit the μAUV from "glueing" at the surface. A second problem we encountered is the accumulation of gas bubles at the surface of the vehicle. This normal physical phnomena causes the same problem as the surface tension be increasing the buoyancy.

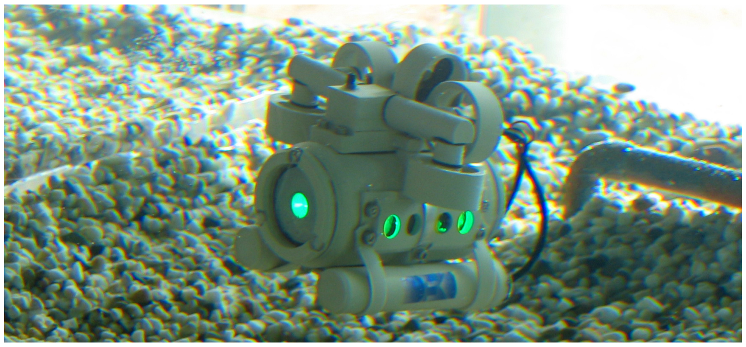

Fig. 1. The vehicle has a cylindrical body with 55 mm in diameter, 125 mm long and has five thruster to hover.

In the following we present the electronics, the sensor system and the internal programming of the μAUV (s. fig. 1) as well as the resolts from the tests on this years CeBIT fair and the conclusions we made from these results.

2 Electronics

The design of the μAUV electronics was a challenging task. In a cylindrical space of just 40mm in diameter and 67mm in height a microcontroller, the power electronics, a pressure sensor, 12 light sensors, 8 ultra bright LEDs and the interconnections of all these components had to be placed (s. fig. 2). Two detachable tubes on either side of the vehicle's main hull contain the electricity supply. Inside these tubes are four AA batteries with 2.7Ah providing electricity for up to 2.5h.

The used microcontroller is an 8-bit ATMega128 with 128KB flash memory and 4KB SRAM (s. [3] for technical specification and datasheet). With the chip running at 14.7456MHz it has to generate the PWM signals for motor control, measure the 12 light sensors and the pressure sensor and control the movements of the vehicle via a behaviour based approach. As all these different tasks have particular requirements with respect to CPU time and periodicity, a very lightweight non-preemptive scheduler was implemented.

The main sensor used is the TCS230 on the sensor-boards. It is a RGB light-to-frequency sensor which generates an output frequency between 1Hz and 600kHz proportional to the measured light intensity of the selected channel (red, green, blue or clear). The frequency is measured by the microcontroller via an externally triggered counter unit. Between one and four light sensors are placed on each sensor board together with an ultrabright green LED (6000mcd). There are two sensor boards on the left, right and bottom side of the vehicle and one sensor board in the front and back respectively.

In addition to the light sensors a pressure sensor is integrated into the hull of the vehicle. The MPX5100DP is used to meter the depth of the vehicle by measuring the differential pressure between the inside and outside of the μAUV. The sensor has a measuring range of $0 - 100$kPa, thus covering depths between 0cm and 1000cm. Besides determining the depth the sensor can also be used to detect leaks since the differential pressure drops in such a case and the vehicle will dive to a higher depth than expected.

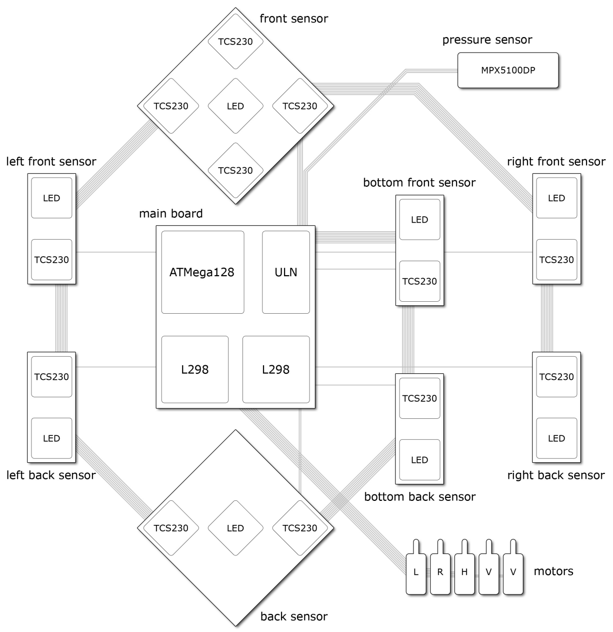

Fig. 2. Schematical overview of the electronic components and their interconnection being packed into a cylindrical space of 40mm in diameter and 67mm in height.

3 Software

The software running on the microcontroller of the μAUV can be divded into three major parts. First, the generation of low level signals, e.g. PWM, to control the power electronics of the vehicle. Second, the measurement of the light and pressure sensors and third, the high level control of the vehicle. Each of these tasks has quite different requirements with respect to computational time and periodicity. Thus we developed a small lightweight non-preemptive scheduler dealing with those requirements.

3.1 Scheduling

To implement a scheduler on a small system with limited capabilities, such as the ATMega128, a number of interdependent aspects have to be balanced. Most of the tasks running on a microcontroller have certain time constrains. Some tasks need an absolute precise timing, where others tolerate some variance. As an example, the generation of a control signal for a common servo motor consists of a pulse with a duration between 1 and 2 milliseconds. This pulse has to be sent every 20 milliseconds. As the pulse width controls the position of the servo motor, the timing determining the pulse width should be very precise. On the other hand, it is not so important to send the pulses *precisly* every 20 millisecond. Most common servo motors can handle any frequency between 50Hz and 100Hz.

In addition to the previously described timing requirements, the different tasks vary in their computational needs. The generation of low level signals, e.g. pulse width modulation, can be characterized as computational cheap but with a high frequency. In contrast, the high level behaviour control is computational more complex but can be executed in a lower frequency. The characteristics of the sensor processing lies in between those two extremes.

Although it would be possible to implement a preemptive scheduler on the ATMega128, this scheduler would need additional information about the timing and computation requirements of the different tasks, as some of these requirements are *hard requirements* which have to be met. Furthermore, the implementation of the single tasks would grow in complexity, as shared resources between the tasks have to be protected by mechanisms like locks or semaphores.

Unlike a preemptive approach our non-preemptive scheduler can handle the aforesaid requirements with considerable less time and effort. However, as the scheduler is non-preemptive the single tasks have to be designed to execute as a series of small steps regularly returning the control to the scheduler. In contrast to common non-preemptive schedulers the single tasks do not just yield the processor for another task to be executed. Instead, after each small step the tasks report their desired relative starting time and the address of their next step to the scheduler. This information induces a time-driven scheduling scheme which specifies the execution sequence of the different tasks implicitly (s. Fig. 3). The scheduler itself has just to keep the list of next steps in order and has to execute these steps at their desired execution times. If a collision with respect to the execution times of two or more steps occurs, a "‘first come first serve”’ approach is choosen. For this reason the timing between two steps of a task can diverge from the desired timing. If absolute presice timing must be guaranteed it has to be generated inside a single step of a task as each step is a atomic function which is never interrupted.

Furthermore, due to the fact that the tasks do not only report the desired relative starting time of their next step but also the address of their next step, the internal structure of a task consisting of a interconnected set of steps resembles the structure of a state machine. As many applications, e.g. behaviour control or the generation of control signals, can be modelled as state machines their implementation easily fits in this programming scheme.

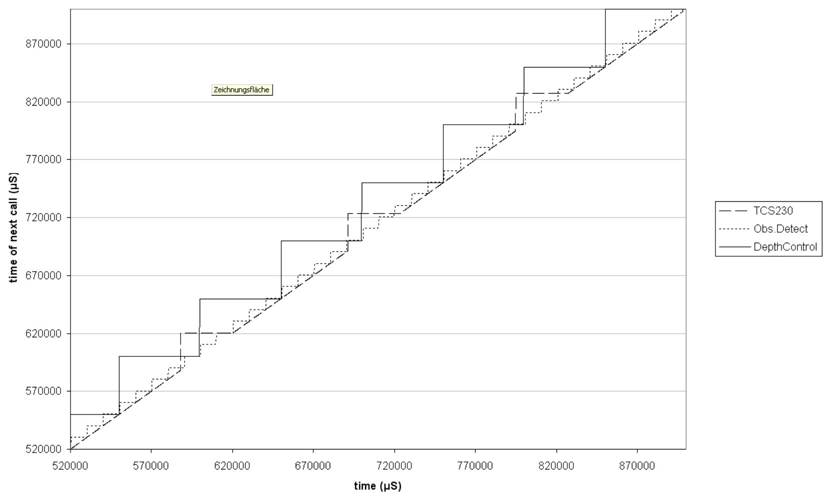

Fig. 3. Example for the automatic interleaving of the single steps of 3 tasks running in parallel on the ATMega128.

3.2 Sensor processing

The μAUV relies on two types of sensors. A pressure sensor which measures the differential pressure between the inside and outside of the vehicle and eight RGB light-to-frequency sensors which measure the intensity of red, green and blue light at different locations on the vehicle (s. Fig. 2).

The differential pressure between inside and outside of the vehicle is used to estimate the depth of the vehicle. For every centimeter in depth the pressure rises by 1 hectopascal. The pressure outside of the vehicle is not only affected by the pressure originated by the water column above the vehicle, it is also affected by the atmospheric pressure at water level. As the pressure inside the vehicle is constant after it is sealed, pressure changes in the atmosphere influence the estimation of the vehicle's depth. The difference between a high-pressure area and a low-pressure area is around 30 hectopascal or about 30 centimeter in terms of the depth estimation. For this reason the μAUV periodically recalibrates the minimum and maximum sensor values by surfacing to water level for the minimum pressure value and descending to the ground for the maximum pressure value.

The eight RGB light-to-frequency sensors which are distributed over different locations on the vehicle measure the light intensity of red, green and blue light at their particular positions. The sensors generate a square wave whose frequency is proportional to the intensity of light measured. One of the sensors main purposes is obstacle detection. To detect an obstacle, the green light intensity is measured before and during a green LED pointing outwards is turned on (s. Fig. 1). If an obstacle is near it reflects the green light and a significant increase in the green light intensity can be detected. In contrast to the pressure sensor which generates a very low-noise analog signal, the signal generated by the light sensors is very noisy and the absolute values vary depending on the enviromental light conditions and other enviromental causes, e.g. the condensed water at the windows of the vehicle. To smooth the signal we use a simple sliding average filter. The filter works fine with respect to cancellation of noise, but also smoothes out the peaks we need for obstacle detection. Furthermore we want to avoid the usage of absolute thresholds as these require new calibration each time the enviroment changes. To differentiate between between a common variation in the signal due to noise and a peak caused by an obstacle, we calculate in addition to the default sliding average a *lower* sliding average and an *upper* sliding average. The lower sliding average consists of all values below and the upper sliding average consists of all values above the default sliding average. The difference between upper and lower sliding average gives an approximation of the sliding standard deviation of the signal and is computationally cheaper than directly computing the sliding standard deviation. Thus a peak caused by an obstacle is detected when the gradient of the signal is larger by a factor α then the approximated standard deviation.

3.3 Behaviour control

As described in section 3.1 the structure of each single task running on the μAUV resembles a state machine. Therefor the behaviour is organized as a set of small interacting state machines where each state machine is responsible for a specific task. This means that the complete behaviour arbitration is done directly by the scheduler-behaviour interaction. Basic tasks include depth control, sensor reading and filtering, obstacle detection, motor control and higher behaviour, e.g. responsible for long-time periodic tasks like recalibration of the depth sensor. The different modules communicate simply via shared variables. This can be done without any protective mechanisms, e.g. locks, as the single steps of the tasks are atomic functions which are never interrupted by another task or the scheduler.

As an example, if the obstacle detector module registers an obstacle it reports this by writing an accordant intensity value in an obstacle-detected variable. As long as the intensity of the obstacle stays quite small and the obstacle position is not directly in the way of the vehicle, the obstacle information is only used by the motor control module wich will slightly adjust the motor values to extend the distance to the assumed obstacle. Instead, if the intensity value of the obstacle is high and/or the obstacle is directly in the way of the vehicle, the higher behavior control will trigger an evasive movement.

The whole behaviour system is inspired by the behaviour network architecture presented in [4], scaled down to the resources available on a small microcontroller system.

4 Evaluation

The μAUV have been successfully used as demonstrators over the complete duration of this years CeBIT fair in Hannover/Germany. We used four different systems with two systems in the water and two systems being recharged. The hardware of the system worked reliable and all systems were still running after one week of daily eight hours of demonstration. At the end of the week we noticed a certain amount of corrosion in the motors and around the power connectors as a direct result of electrolysis. The runtime of a battery charge was around two and a half hours.

The clear glass of the aquarium posed a problem for the μAUV's obstacle avoidance algorithm. The amount of light reflected by the glass was neglectable as long as the the front sensor of the μAUV was not positioned rectangular to the glass, which resulted in a lot of collisions or "hanging around the glass"-positions. As negatively as this looks in the first place as much it was approved by the audience, since it was possible to get a reaction from the μAUV by placing a hand directly at the glass, enhancing the reflective properties of the glass and triggering the obstacle avoidance behaviour, giving the whole demonstration an interactive components.

By running the "sleep" and the "sensor callibration" behaviours with a very short frequency (3min and 2min), there was always enough action inside the aquarium to hold the visitors at the booth. This, from an engineering point ridiculous, short frequency is the statistically proofed maximum time of an visitor observing a demonstration at a fair.

5 Conclusion

Using micro-scale underwater vehicles as a demonstrator for new underwater technologies and as attraction for a fair-booth proofed to be a highly successful project. The actual design flaws of the current μAUV are mainly its limited computing capabilities and the major corrosion problem at the motors and the power connectors. We are currently making a redesign of the μAUV. This μAUV^2 incorporates a better computer architecture based on a DSP/FPGA combination, uses a sensor suite enhanced by a small CCD camera, has a better thruster system and will also apply a diving cell to adjust buoyancy. The μAUV^2 will then not only used as a demonstration vehicle but also as basis-system for master and bachelor student projects.

References

1. Clarke, H. et al.: Design and Development of the AUV Harp. 3rd Annual International Autonomous Underwater Vehicle Competition, Orlando, (2000).
2. Navinda Kottege and Uwe R. Zimmer: Relative Localization for AUV swarms. Proceedings of the International Symposium on Underwater Technology, Tokyo, Japan, 2007
3. ATMega128: Technical specification and datasheet. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf
4. Jan Albiez: Verhaltensnetwerke zur adaptiven Steuerung biologisch motivierter Laufmaschinen Dissertationsschrift, GCA Verlag, Februar 2007