Interdisciplinary Education in Optics and Photonics Based on Microcontrollers

Paul Dreßler[1*], Heinz-Hermann Wielage[1], Ulrich Haiss[1], Oliver Vauderwange[1], Dan Curticapean[1]

[1] University of Applied Sciences Offenburg, Germany, paul.dressler@hs-offenburg.de

ABSTRACT

Not only is the number of new devices constantly increasing, but so is their application complexity and power. Most of their applications are in optics, photonics, acoustic and mobile devices.

Working speed and functionality is achieved in most of media devices by strategic use of digital signal processors and microcontrollers of the new generation.

Considering all these premises of media development dynamics, the authors present how to integrate microcontrollers and digital signal processors in the curricula of media technology lectures by using adequate content. This also includes interdisciplinary content that consists of using the acquired knowledge in media software. These entries offer a deeper understanding of photonics, acoustics and media engineering.

Keywords: 000.2060 Educations, Education in Optics and Photonics, Microcontroller

1. INTRODUCTION

As our university offers several courses of studies with a wide range of subjects – like mathematics, information technology, natural science, electrical engineering and media design – it is always a big topic how to combine these different faculties. The aim is not to teach the subjects separate from each other, but to bring them together in interdisciplinary lectures, projects and applications. Two major components of our media technology lectures and laboratories are issues of optics and photonics.

Inspired by the educational program „Hands-On Optics“ by SPIE, OSA and NOAO [1] two years ago a special idea was conceived at our laboratory: Programming microcontrollers in a hands-on seminar. This seminar contains theoretical contents alternating with practical tasks, whereat focus is on the practical issues (“learning by doing”).

Figure 1: Microcontroller “ATMEGA 2561” [2]

Microcontrollers can be applied in various ways to teach content from the fields of optics and photonics. They can be used to control LEDs, displays, light detectors and infrared sensors, which makes it possible to build measuring instruments like e.g. a lux meter, a light barrier or an optical distance meter.

Our microcontrollers are programmed in the programming language *C*, which is a good choice, for all students have learned the basics of this language in their first semester. Equipped with this previous knowledge the students are able to start their own programming projects after only a short time. The student’s certificate of performance at the end of the semester is to gather in small groups and to develop a microcontroller program, which meets certain demands. Thereby they are encouraged to contribute their own ideas and interests.

* Corresponding author: paul.dressler@hs-offenburg.de

12th Education and Training in Optics and Photonics Conference, edited by Manuel F. P. C. Martins Costa, Mourad Zghal, Proc. of SPIE Vol. 9289, 92891P
 · doi: 10.1117/12.2070556

The learning goals are to stimulate the student's interest in the multiplicity of subjects related to this course and to support a deeper understanding of the close connections between them.

2. THEORETICAL BACKGROUND

2.1 Structure and function of a microcontroller

Generally a microcontroller is a single chip that contains an entire small computer. Hence it is also called "computer on a chip". Usually they are constructed for a particular purpose, e.g. for controlling a traffic light system, a digital clock etc.

The microcontroller's centerpiece is the *central processing unit* (CPU), which is also called *microprocessor* and which primarily consists of the *arithmetic and logic unit* (ALU) and the *control unit* (CU). Its task is to perform all operations on data by means of a program:

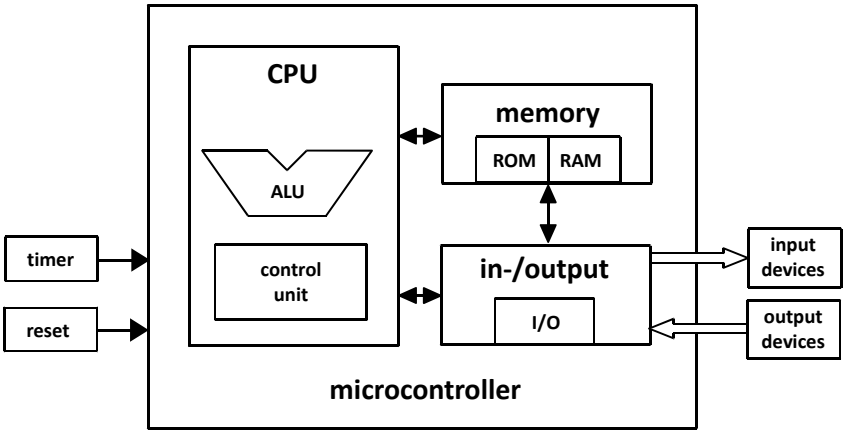


Figure 2: Microcontroller system, drawn by author

The *ALU* loads its data from *memory* and also stores it there afterwards. The *control unit* decodes the given instructions of the program, which can be:

- arithmetic and logic instructions
- transport instructions
- jump instructions
- bitwise operations.

Subsequently the *ALU* executes these instructions and manages the data in- and output. To get in contact with the outside world you need several peripheral units in addition to the microprocessor, such as digital input/output registers (often called "ports"), data memory, program memory and timers. The in/out registers are there to control and read the peripheral devices. They are the ports to all external devices.

All these peripheral units are built in a microcontroller, for the aim is to integrate as much (programmable) functionality as possible into one single chassis. The microcontrollers we use are reprogrammable, which is absolutely essential for our educational purpose: With this condition our students can freely test their written programs or program fragments by trial and error.

2.2 Architecture of the microcontroller „ATmega2561"

For our development board we use the microcontroller *ATmega2561* from the 8-bit AVR family of *Atmel* Corporation. Building our microcontroller systems based on this device provides us a reasonable and useful hardware development environment.

In Figure 3 you find the system architecture of the *Atmega2561* in detail. The microcontroller is equipped with ten input/output ports, each with eight pins that are switched as digital I/O ports by default. To enable further functions the configuration can be changed with the help of controller-specific register entries.

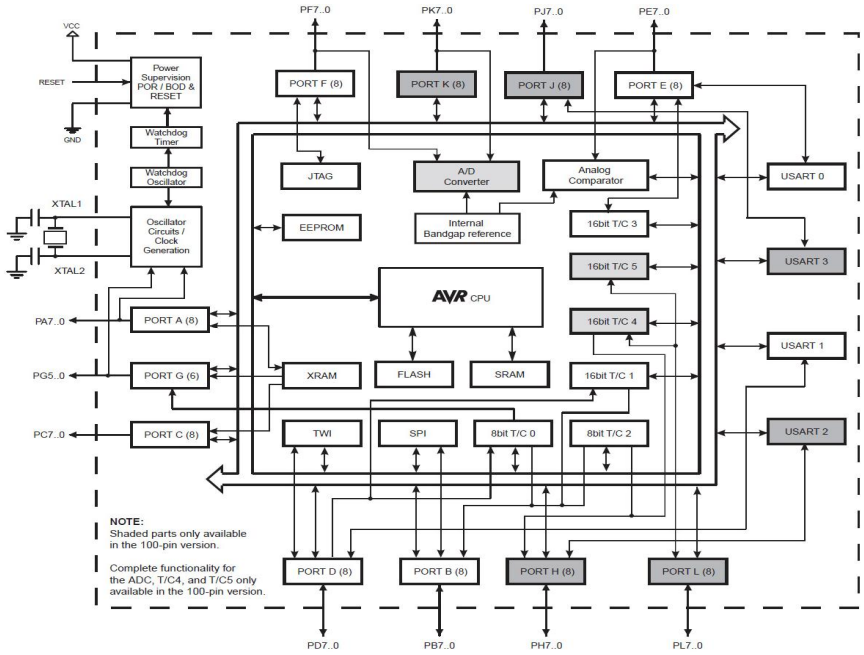
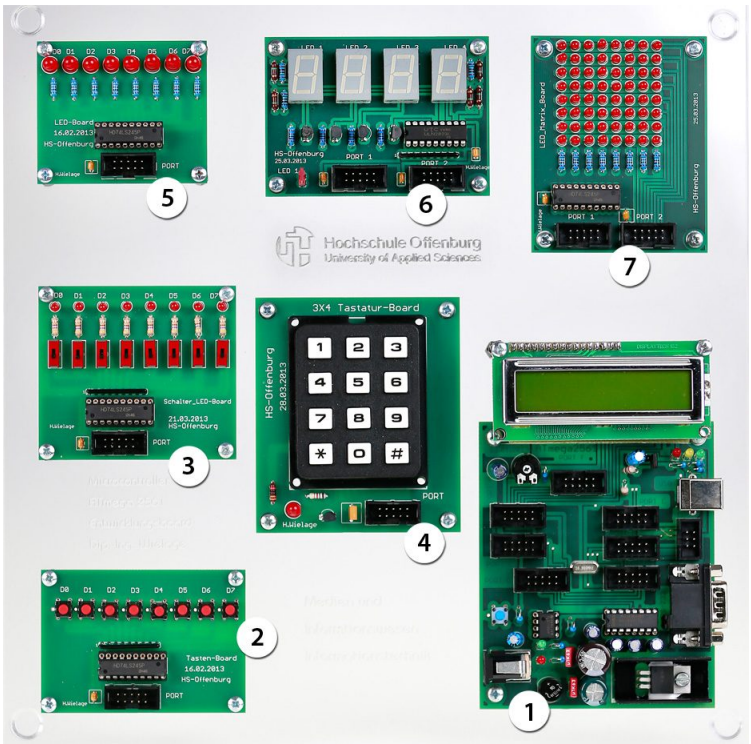


Figure 3: Atmega2561 system architecture [3]

3. SETUP

3.1 The development board (hardware)



The development board is structured modularly. At the bottom right you find the microcontroller board with its LCD (1).

Figure 4: Realized application setup with Atmega2561 system architecture.

3.2 On-board in- and output devices

The development board has been designed to provide an easy entry into the field of microcontroller technology and programming. The devices are structured in a way which makes their functions easy to understand and allows a wide range of applications. With this approach it is possible to cover the demands of a good entry into the world of microcontrollers.

Our board contains three input devices:

- 8 push buttons (2)
- 8 switches (3)
- a 3x4 numeric keypad (4)

… and three output devices:

- 8 LEDs (5)
- four 7 segment displays (6)
- an 8x8 LED matrix (7)

Out of these boards we will take a closer look at the *LED matrix*, as this board is particularly interesting for optical and mathematical uses:

When designing the LED matrix board (Figure 5) we intentionally avoided the use of a multiplex control, to make it easier for the students to understand the close correlation between the 8 data lines for the columns of the matrix (port 1) and the eight data lines for the rows (port 2).

Example: A logical "1" at row 1 and a logical "0" at column 1 will enlighten the LED11.

On our 8x8 LED matrix we can display the complete ASCII character set, as well as simple graphics.

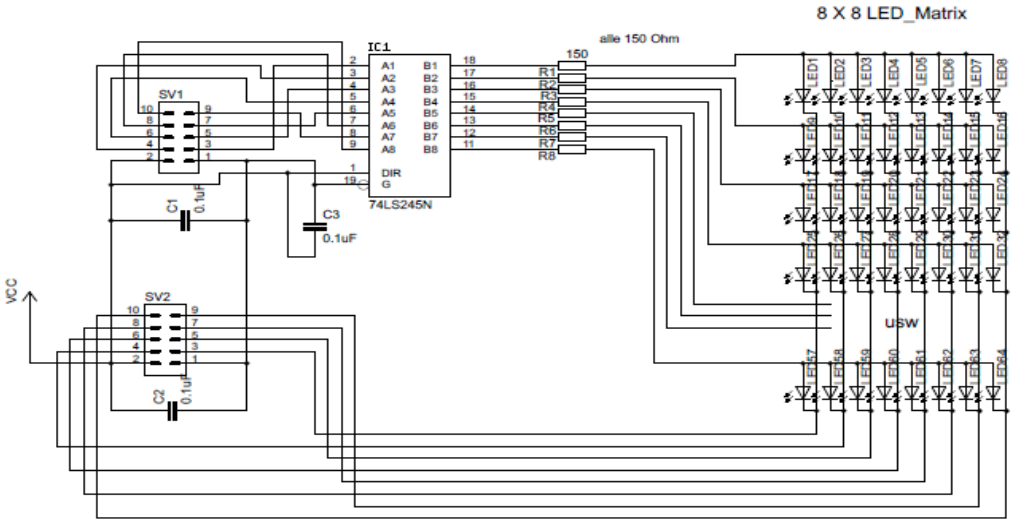


Figure 5: Circuit diagram of the LED matrix, drawn by author

3.3 External devices: sensors

To support our students in their understanding of optics we are additionally using several sensors. For technical measurement reasons they are not part of the development board, but are connected as external input devices to a port of our microcontroller. We use the following sensors:

- A remote controlled temperature sensor: LM334
- A light sensor for the visible spectrum: BPW42 (silicon NPN epitaxial planar phototransistor)
- A light sensor for infrared light: TSOP 1730 (photo modules for PCM remote control systems)

…of which the last two sensors are particular interesting for uses in optics:

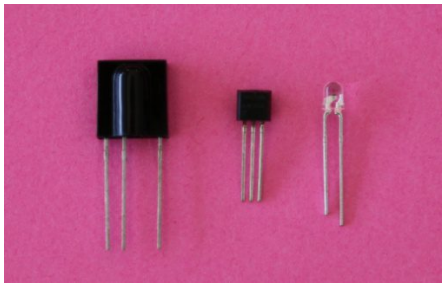


Figure 6: External sensors

Function of the light sensor BPW42

The light sensor BPW42 shows a linear relationship between the irradiance and the phototransistor current. When using the displayed circuit (Figure 7) the sensor delivers an output voltage of $U_L = I_C (R_V+R_P)$ which is proportional to the irradiance E_e (in W/m^2).

If required, the irradiance has to be converted into the illuminance E_v (in *lux*), while having to pay attention to the correct wavelength. Afterwards the calculated voltage U_L has to be adapted to the 10-bit A/D converter. [4]

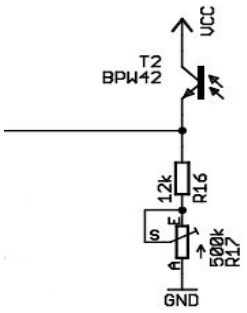


Figure 7: BPW42, drawn by author

Function of the infrared light sensor TSOP 1730

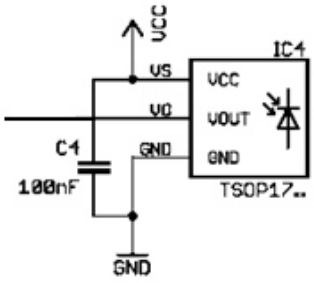


Figure 8: TSOP 1730, drawn by author

The light sensor TSOP 1730 is an infrared receiver with a digital output signal, whose spectral maximum sensitivity peaks at a wavelength of $\lambda = 950$ nm. It works with a carrier frequency of 30 kHz and a maximum data rate of 2400 bps.

An infrared transmitter sends a carrier-keyed signal, i.e. the infrared signal is being switched on and off with a frequency of 30 kHz. Such signal sequence ("burst") has a transmission period of circa 600 µs. By sending this burst at specific times it becomes possible to transmit data. The receiver's output signal S_{OUT} can be passed directly to the microcontroller.

Figure 9 illustrates the relationship between the optical signal and the output signal S_{OUT} at the receiver. [5]

For the development of appropriate algorithms for measuring illuminance or temperature the students have to possess a basic knowledge of optics and electrical engineering.

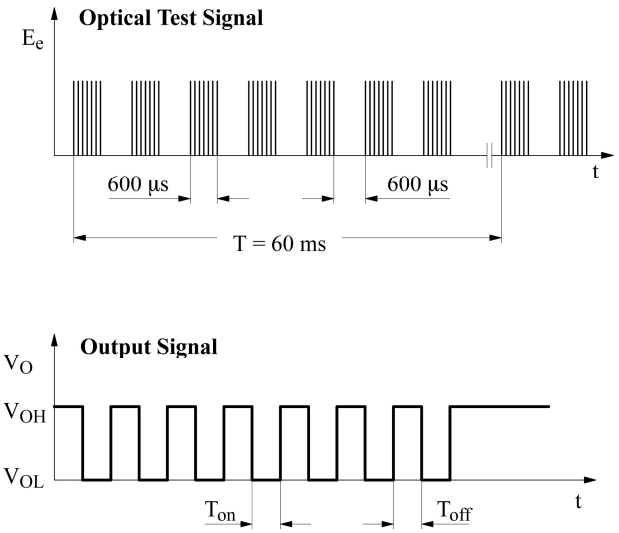


Figure 9: Optical signal and digital output signal [5]

3.4 Software

When it comes to the software, an important part of our microcontroller project – along with electrical engineering, optics and mathematics – is the subject of information technology. This includes knowledge about the syntax of the programming language *C*, the ability to structure the own source code and the skill of operating the IDE.

The IDE (integrated development environment) is an application package for the PC combining editor, assembler, compiler, linker and simulator applications under one uniform surface, which can be operated more easily. For programming microcontrollers there is a variety of development environments for all different types of microcontroller hardware. All of them have in common that they are useful tools, which offer fast access to important functionalities, liberating the developer from recurring and formal tasks and helping him to manage his work results. This has the advantage for the developer that he can focus completely on his real task: the development of microcontroller applications [6].

To transfer the written source code to the microcontroller, the code first is being compiled into machine code in *hex* format, then the controller has to be connected to the computer via a programmer to load the code to the controller's memory.

At this project we are using the development platform *AVR Studio 4* by *Atmel* in combination with the compiler *WinAVR*. This IDE offers:

- Source code editor with syntax highlighting
- Project administration for single projects
- Integrated project administration (automatic makefile generation or use of external makefiles)
- AVR-GCC (WinAVR) and AVR assembler support
- Debugger support for AVR simulator and emulator
- AVR programmer (flasher) support for several controllers. (However, to program the ATmega 2561 we do not use the integrated AVR programmer, but the tool *USBasp* by Thomas Fischl [7]).

4. APPLICATIONS

The modular structure of the development board with arbitrary wiring possibilities between in- and output ports offers a big variety of thinkable applications.

To getting started after a short theoretical instruction at the beginning of the course the students are confronted with several small tasks, which are comparatively simple to solve. This can be for example:

Triggering an LED

Task: The LED connected to Pin 4 of Port A shall be switched on and off periodically every second. Source code (solution):

```
//Switching on and off an LED

#include <avr/io.h>            // adding library (input/output etc.)
#include <util/delay.h>        // adding the "delay" functionality

int main(void){

  DDRA |= (1<<PA4);            // declare Pin 4 of Port A as output
  PORTA |= (1<<PA4);           // switch off Pin 4

  for(;;) {                    // repeat forever
    PORTA &= ~(1<<PA4);        // switch it on
    _delay_ms(500);            // wait half a second
    PORTA |= (1<<PA4);         // switch it off
    _delay_ms(500);            // wait half a second again
  }

  return 0;
}
```

A short and clear code snippet like this has proven to be most suitable for getting used to the syntax of the programming language *C* and of the control commands for our microcontroller. After a couple of similar finger exercises with a constantly increasing level of difficulty the students are finally able to manage programs with more complex structures and functionalities.

5. RESULTS

We started the microcontroller project two years ago at our laboratory as an optional alternative to the media technology lab. In the following phase of consolidation it became more and more popular among the students of our faculty, notably also among many female students.

The results of the participant's big programming tasks (which are their certificate of performance) at the end of each semester are very encouraging: Not only that they are able to get along with their tasks – in addition many of them are highly motivated to go further into the subject by coming up with their own ideas how to enhance their projects.

Considering the taxonomy of learning goals proposed by Bloom [8] this achievement of supporting the students in realizing their own ideas reaches level K5 to K6:

- Knowledge (K1): Precise (if not literal) reproduction of information;
- Understanding (K2): Ability of applying sense-preserving transformations to knowledge, ability of paraphrasing knowledge in own words and finding examples;
- Application (K3): Ability of applying abstractions, rules, methods, algorithms etc. in concrete situations;
- Analysis (K4): Ability of decomposing ideas and problem settings into its elements and comparing problems, spotting differences;
- Synthesis (K5): Composition of individual elements to a whole;
- Evaluation (K6). [9]

The feedback sheets our students returned were very positive and had one common tenor: "Finally a really 'hands-on' course".

REFERENCES

[1] The Optical Society (OSA): "Hands-On Optics", Washington DC, USA, <http://www.osa.org/en-us/membership/youth_education/hands-on_optics> (10 June 2013).

[2] reichelt elektronik GmbH & Co. KG: „ATmega2561", Sande, Germany, <http://www.reichelt.de/bilder/web/artikel_ws/A300/TQFP100.jpg> (03 June 2013).

[3] Atmel Corporation: "Datasheet of Atmega640/1280/1281/2560/2561 Complete", pg. 5, San Jose, CA, <http://www.atmel.com/Images/doc2549.pdf> (10 June 2013).

[4] Telefunken: "BPW 42", datasheet, Frankfurt am Main, Germany, <http://www.datasheet4u.net/download.php?id=555151> (27 May 2013).

[5] Vishay Semiconductor GmbH: "TSOP17..," datasheet, pg. 5, Heilbronn, Germany, <http://pdf.datasheetcatalog.com/datasheet/vishay/82030.pdf> (21 May 2013).

[6] Wikipedia, Die freie Enzyklopädie: „Integrierte Entwicklungsumgebung", version: 8. June 2013, 11.15 am, <https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung> (18 June 2013).

[7] Fischl, Thomas: „USBasp - USB programmer for Atmel AVR controllers", Obernzell, Germany, <http://www.fischl.de/usbasp> (27 May 2013).

[8] Bloom, Benjamin S.: [Taxonomie von Lernzielen im kognitiven Bereich], 4. Auflage, Beltz Verlag, Weinheim und Basel, Germany/Switzerland (1972).

[9] Anderson, Lorin W. & Krathwohl, David R. (Hrsg.): [A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives], Addison-Wesley, New York (2001).