A μCONTROLLER LAB FOR DISTANCE LEARNING

Günther Gridling, Bettina Weiss
Embedded Computing Systems Group E182/2, Technische Universität Wien, Treitlstrasse 3, A-1040 Wien.

1. Introduction

This paper[1] presents the hardware for an introductory lab course on microcontrollers suitable for distance learning. The course mainly targets computer engineering students (about 120–150) who have not worked with microcontrollers before, and should teach them microcontroller programming in C and assembly, working with datasheets, and a basic understanding of the hardware used in the course. Although the course is introductory, we tried to make it also appeal to more advanced students who already have some familiarity with microcontrollers.

Since microcontroller programming is an important aspect of embedded systems education, there are naturally a great many courses with a wide variety of hardware, e.g. [7, 5, 2, 1, 6]. Most of them, however, were not designed for distance learning and use fixed hardware.
For our course, we developed modular hardware and an extensive set of mostly voluntary exercises, which allows beginners to develop basic skills on an inexpensive take-home lab kit, whereas advanced students may work on more challenging exercises with additional hardware modules. Our take-home lab kit mainly consists of a controller board with an easy-to-program microcontroller and an I/O board with some basic I/O components. Further boards offer the students several different communication interfaces and motor drivers. Each student can borrow a lab kit for the whole term (or buy it), whereas the more advanced boards are only available in a small number and can be borrowed just for a limited time.
The remainder of the paper briefly describes our hardware. For an extensive description and schematics, refer to [3].

2. Basic Boards

The Microcontroller course was first set up in 2003, and we conducted an extensive search for suitable off-the-shelf hardware at that time. The hardware needed to be low-cost for us to acquire a sufficient supply and loan it to students. It also had to be modular and support a variety of exercises. A further crucial feature was robustness, since beginners tend to do things wrong and produce shorts.

[1]This work is part of the SCDL "Seamless Campus: Distance Labs" project, which received support from the Austrian "FIT-IT Embedded Systems" initiative, funded by the Austrian Ministry for Transport, Innovation and Technology (BMVIT) and managed by the Austrian Research Promotion Agency (FFG) under grant 808210. See http://www.ecs.tuwien.ac.at/Projects/SCDL/

Although there are a lot of quite advanced simulators available, the idea of a course based on the use of a simulator was dismissed early on, mostly for two reasons: One is that beginners tend to misuse the comfortable debugging environment of a simulator. Instead of using it as the ideal tool to help them understand the inner workings of the controller, they end up following a pure trial-and-error approach for debugging. A second reason not to use a simulator is that our students are often unfamiliar with typical embedded systems hardware and we wanted them to gain some hands-on experience. For these reasons, we set out to acquire suitable boards. Unfortunately, back in 2003, we found no off-the-shelf hardware which fulfilled all our requirements and was still affordable. Therefore, we decided to design dedicated hardware for our course.

Controller Board. We decided that a suitable controller for teaching introductory microcontroller programming should have at least four bidirectional I/O ports with optional pull-ups, 8KB flash, 1KB SRAM, two 8-bit timers/counters, PWM, external interrupts, an ADC, preferably also an analog comparator, UART, a watchdog, and various sleep modes. On the other hand, the microcontroller should have a simple architecture and instruction set. Therefore, we chose an Atmel AVR 8-bit RISC microcontroller, the ATmega16 in particular, for our controller board, which fulfills these conditions.
The board contains all logic required by the microcontroller, see Figure 1 (left). All the I/O pins of the controller are accessible through sockets, so that they can be connected to I/O boards, and are protected with series resistors against accidental shorts. Next to the standard Atmel ISP-Header, our board also features a USB controller to allow programming via USB, which was widely requested by students with laptops.

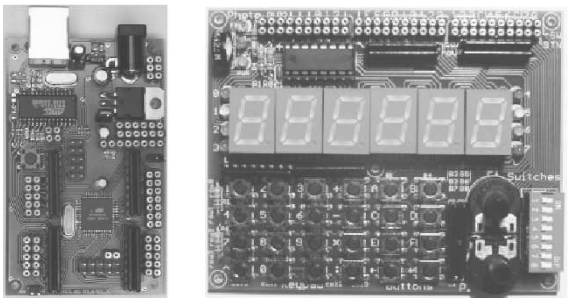

Figure 1: Controller Board (left) and Simple I/O Board (right)

Simple I/O Board. Our basic I/O board contains most of the hardware covered by our compulsory exercises, namely a six digit multiplexed numeric LED display with darlington drivers, a matrix keypad of 4x4 buttons, 8 single buttons (one of them debounced), 8 single LEDs, 8 switches, 2 potentiometers, and 1 photo transistor, see Figure 1 (right). Again, all components are accessible through sockets. To prevent damage due to improper connections, and to allow simple component testing of the board, all sockets are protected by series resistors. Exercises with this board range from reading the state of a switch or turning on a LED up to writing drivers for the keypad and the display.

3. Advanced Boards

The advanced boards are mostly for voluntary exercises. The idea here is to provide hardware that covers interesting topics and preferably is fun to work with as well.

Motor Board. In order to show students how to control dc and stepper motors, we developed two boards for motor control. One uses OnSemi's MC3479 motor controller to drive a bipolar stepper motor. The other features an L293D to provide a four-channel push-pull driver, which can be used as two independent H-bridges to drive stepper or dc motors. Figure 2 shows the lab board, which features a dc motor in the middle, controlled by the L293D, a unipolar stepper motor, also controlled by an L293D, and a bipolar stepper motor controlled by the MC3479. Encoder discs and photointerrupters mounted on the motors give the students feedback. Exercises with these boards range from simple motor control to plotting speed curves.

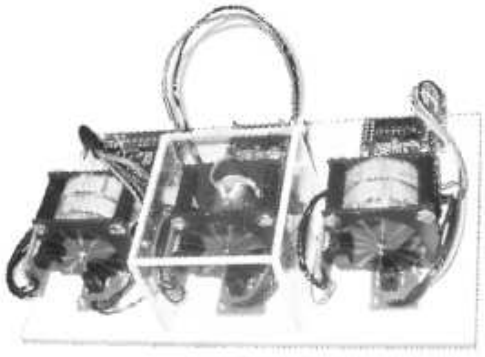

Figure 2: Motor Board with dc motor, unipolar and bipolar stepper motors

Other Boards. We offer several other boards as well. One is an interface board with USB, CAN, RS-232, and RS-485 drivers. Another provides access to a remote-controlled line-following robot. Yet other boards allow programming various LCD displays, interface to a compact flash card reader, access an MP3 decoder, or provide an IDE interface for harddisk access. Due to the modular nature of our hardware, the possibilities here are virtually unlimited, and the more complex hardware can be used in other courses as well.

Other Microcontroller Boards. Students who already have some background are, of course, soon bored with our simple Atmel AVR controller. To allow these students to broaden their horizons as well, we offer several other microcontroller boards which can be substituted for the Atmel AVR board. Currently, we offer boards with a HCS12 (16-bit CISC), an ARM (32-bit RISC), and a MSP430 (16-bit RISC). These boards are available in limited quantities, and only the best students are allowed to switch to one (or more) of these controllers as the course progresses.

4. Distance Learning

Since our course should support distance learning, we offer a lab kit consisting of the Atmel AVR board, the simple I/O board, and the development environment. We stock sufficiently many lab kits so that all students may either buy the lab kit, or borrow it for the duration of the term. To avoid trouble with different operating systems, we offer a Knoppix CD with the lab kit, which contains the complete (Linux) development environment. We solely use freeware/open source software.
Lab kits are tested before they are issued, and then again when they are returned. Testing is semi-automatic (testing the functionality of buttons and switches requires manual interaction); a dedicated test board performs the tests.
To support online submissions of exercises and remotely conducted exams, we are currently developing a test environment that can automatically verify the correctness of submitted programs [4]. This allows exams to be conducted anywhere, without the need for qualified supervisors.

5. Conclusion

We presented the hardware for an introductory microcontroller lab course which supports distance learning and multiple difficulty levels. Our hardware is designed to be modular and inexpensive to enable students to work at home on borrowed lab kits. More advanced students are supported by a variety of different microcontrollers and additional hardware. We believe that our hardware is well suited for beginners, while at the same time also capturing the interests of advanced students.

References

[1] M. Amirijoo, A. Tešanović, and S. Nadjm-Tehrani. Raising motivation in real-time laboratories: The soccer scenario. In *35th SIGCSE Technical Symposium on Computer Science Education*, pages 265–269, Mar. 2004.

[2] R. Bachnak. Teaching microcontrollers with hands-on hardware experiments. *Journal of Computing Sciences in Colleges*, 20(4), Apr. 2005.

[3] G. Gridling. Microcontroller lab hardware. Research Report 15/2006, Vienna University of Technology, Institut für Technische Informatik, 2006.

[4] V. Legourski, C. Trödhandl, and B. Weiss. A system for automatic testing of embedded software in undergraduate study exercises. In *Workshop on Embedded Systems Education (WESE'05)*, pages 44–51, Sept. 22nd, 2005.

[5] J. W. McCormick. We've been working on the railroad: A laboratory for real-time embedded systems. In *36th SIGCSE Technical Symposium on Computer Science Education*, pages 530–534, Feb. 23–27, 2005.

[6] S. Merchant, G. D. Peterson, and D. Bouldin. Improving Embedded Systems education: Laboratory enhancements using programmable Systems on Chip. In *IEEE International Conference on Microelectronic Systems Education (MSE'05)*, pages 5–6, June 12–14, 2005.

[7] C. E. Nunnally. Teaching microcontrollers. In *26th Annual Frontiers in Education Conference (FIE'96)*, volume 1, pages 434–436, Nov. 6–9, 1996.