USING MICROCONTROLLERS FOR EDUCATIONAL ROBOTICS IN UNDERGRADUATE UNIVERSITY STUDY PROGRAM

Pavel Petrovič

Comenius University Bratislava (SLOVAKIA)

Abstract

Microcontrollers are at the core of every robotics application, whether educational, industrial, entertaining, or artistic. Sometimes they are hidden from the user who is enjoying a higher-level interface and programming language, but often the users are being exposed to the internals of the low-level programming in a direct way. Some would argue that the low-level is too difficult, misses the required abstraction, and contains too many technical details and thus it is too challenging, others would correctly point-out that by disclosing this layer to the students, we are creating a much better understanding of the system they are studying, while leaving the abstractions to their imagination. At the same time, we are opening for the possibility for the excellent students to take on their learning experience and develop new products as they advance further on. We believe that both approaches are useful and depend on the target group, situation, kind of students, and learning goals. We have applied programming microcontrollers in several undergraduate courses, summer schools, robot clubs, and workshops. In this article, we explain our experiences, learning situations our students experienced, and solutions we have developed in various courses in the past years, including running a large introductory course for tens of students with microcontrollers in two different ways, and the feedback from the students - both quantitative and qualitative.

Keywords: microcontrollers, undergraduate studies, arduino, stm32.

1 INTRODUCTION

During more than past 10 years, we have been providing several different courses related to robotics as part of Applied Informatics study programs at Comenius University. First, one ECTS credit course invites students to a set of couple of hours long hands-on workshops with various platforms. Typically, it is one of the first meetings of students with robot programming, control, sensing, locomotion and navigation. Students usually work in groups in a constructionist scenario. After a guided exploration of each platform capabilities, they are given a set of many times verified challenges with a possibility to work on their own creative designs and programs. They are exposed to various generations of LEGO robotics sets, Arduino and/or other microcontroller-based robots as we will discuss later. Second, three-credit course includes the above, but expects the students to focus on a single larger project they work on throughout the second part of the semester individually or in groups. They have to produce a simple report with description of the task and their solution, images, code and a video, i.e. their project has to reach a successful implementation phase with a demonstration of their working solution. Sometimes, the students would work on preparing a robot for a competition. Finally, in addition to these two very practical courses, we also provide a course with a balanced theory and practical work, where students learn about various AI algorithms that could be useful for robotic systems, but they also have to realize a practical project in the second part of the semester, and that is a topic this article aims to address.

2 EDUCATIONAL ROBOTICS PLATFORMS

When talking about educational robotics in a broader sense, we usually mean some physical programmable artefacts that can interact with its environment – acquiring information from the environment using sensors and performing actions using mechanical actuators, sound, and visual outputs. Such artefacts are typically built from sets of standardized components that form a certain platform, or alternately designed with help of some digital technology, such as 3D printing or laser cutting. Let us provide a classification of such platforms from different viewpoints.

The first dimension addresses the **age of the learners**. This starts from the children in the kindergarten – as is the case of for instance BeeBot and similar driving robots, which are typically programmed to travel simple trajectories by entering a sequence of elementary moves. They are often supported by stories and driving mats that enable the interaction of the child and robot and the focus is to obtain first

experience with some external entity that could be controlled and programmed by a human. Elementary school pupils (age under 10, sometimes under 12 years) usually work with toy-like systems where the focus is on stories, and typically the programming languages they use contain only loops and no conditionals. However, waiting for events (such as coming to an obstacle) is a supported language feature. An example of such system could be LEGO Education WEDO, or now replaced by LEGO Education Spike Essentials. In the lower secondary or upper primary school level (ages 10-14), students typically work with Scratch-like programming languages which are often extendable with extra functionality or plugins, but they are benefiting from high-level abstraction of electronics layer, which is typically completely transparent: the lowest level being the numerical values obtained from sensors, such as reflected light measurement expressed as a percentage, measured force in N, or rotation angle. This level of programming typically requires abstract and conceptual thinking, which pupils of younger age are not capable of. On one hand, it is not very different from what they need for programming simple computer programs and games for example in Scratch without any robotics hardware involved. The difference lies in their designs being real artefacts that perform in our real world as contrasted to a computer virtual world. This has a positive effect on their creativity, spatial reasoning and imagination, and experiences of concepts from physics, mechanics, construction, dealing with uncertainty, and others. We consider such scenarios clearly superior to plain computer-programming scenarios as they provide superset of the skills and knowledge and higher level of interactions. Secondary-school students are ready for usual text-based programming languages, such as the unfortunately wide-spread Python, or various dialects of C-like languages. Here the skills gap opens at large: some students remaining on a very introductory level, while the most advanced being able to design their own electronics circuits, hardware and software. At higher education level, robotics is being used in the full range of complexity – from simple LEGO robots for first hands-on experiences to designing professional robotic applications and performing research studies.

The second axis that we have already touched is the **purpose** of the platform. The most common purpose is a general motivational introduction to technology and informatics. Those cases typically utilize environments with high-level abstractions. We see however two other very important classes: 1) using educational robotics platforms to teach specific subjects, such as physics, mathematics, art, biology, computer science, or other and 2) learning about embedded computing, low-level hardware platforms, signal processing, electronics, and similar – and that is the area where microcontrollers are at focus.

Another aspect is the **computing power** of the platform, ranging from tiny 8-bit platforms with very limited memory and other resources (typically available for 2-5 Eur/piece), followed by low-end 32-bit single-chip computers that already have some advanced capabilities, operate at higher CPU frequencies, but still having very limited memory, and connectivity. On the other hand, both of these typically run the user program as is without any operating system, which allows a full control over the hardware, which is often a requirement for the elements responsible for low-level control of motors, sensors, and interface signals. Therefore, higher-level platforms typically contain also a low-level microcontroller for the bottom layers liberating the main board from the low-level tasks. Higher-level platforms of various power are then comparable to regular computers, running some standard operating system such as Linux and providing large range of connectivity such as wifi, wired ethernet, USB, sata, hdmi, and some specialized connections, such as cameras and displays. Such platforms allow for tasks where computer vision or collecting and processing large amounts of data is required.

On another axis we have a **programming language or interface**. In addition to those described above, worth mentioning are also fully iconographic systems, where icons represent the various commands. One such very advanced programming language is LabView, which has a very strong and distinguishable feature that it is a data-flow language as contrasted to most other languages which are control-flow languages. Data flow has the advantage of natural parallelization of the computation without the need of human engineer designing the individual parallel threads of execution and their interaction, which very quickly becomes very complex and prone to introducing bugs and conceptual mistakes. However, unfortunately, also most of the iconographic programming languages are control-flow including the versions of LEGO MINDSTORMS (both NXT and EV3) that was based on LabView. Currently though the trend is very uniform – to use Scratch-like programming languages, which is a text-based language, where textual commands are available in a palette of commands, thus programming in this language does not require learning the syntax by heart. At the same time, when all programming environments for children use the same style/syntax/semantics, it is easy and very flexible for young learners to select a different appropriate tool in each different learning situation. Yet, we feel that reducing the wider selection of different programming languages to uniform Scratch-like environments is counter-productive in terms of creativity, diversity of ideas, and thinking styles, and such unification is steeling certain richness from the world of young learners.

A very important classification axis is whether the **platform is open or closed** regarding its use and development. Typical examples of closed system are LEGO robotics sets, which do not follow any open standards, and do not allow access to any low level unless reverse engineered. And this typically happens both at the level of hardware – using non-standard connectors, protocols, inaccessible and undocumented components and protocols, and at level of software – with very little or no integration possibilities, no support for writing own plugins and extensions and connectability. That leads to low **flexibility of hardware interfaces**. This is typically for the reasons of for profit-driven management and production and leads into more waste (as the support for a particular product expires, it can hardly be used for anything), less interoperability, less reuse, and such strategy cannot be defended by any means in contemporary world where **reusability**, recycling and sustainable economy are very high priorities. And this is especially true for products that are sold in hundreds of thousand pieces in countries with large markets, and millions of students. A typical example of an open system is Arduino or Raspberry Pi, which use the most standard hardware interfacing. Somewhere in between is micro:bit system, which is extensible only at an extra additional cost using an extension board.

This is closely related to the axis of **industrial compatibility**, which represents how much the skills obtained when learning the platform are directly transferrable to industrial applications. For example, learning the C++ programming of Arduino has a very high compatibility with the skills required for programming industrial applications, whereas Scratch-like languages are useful only in terms of introducing the general programming concepts, lacking the understanding of the whole story and whole processing pipeline. The most disturbing example is the popularization and wide-spread use of Python language in all the learning environments, now including the robotics platforms – even though interpreted languages which often exhibit poorer performance of 5 degrees of magnitude are very unsuitable to program low-end (and thus low-cost) microcontrollers, where the precise control of time and signals and high performance is a critical quality. The most unfortunate part of this Python popularity is that the learners become so bound to that low-performing language that they then tend to use it also for professional applications as grown-up developers, where it really is not advisable. Consequently, we have reached a point where everything is programmed in Python – an unsuitable language for production deployment.

The last classification axis we would like to point out represents the **level of modularity**. At one end there are monolithic educational platforms – a complete robot that cannot be divided into replaceable and serviceable parts and cannot be extended with additional modules. At the other end we again have Arduino with wonderful hardware and software modularity – a system of automatically installable libraries, and thousands of hardware modules that are compatible with Arduino. Some of the modules even follow the pinout form and can be used as a “shield” that transparently adds new functionality.

There are other axes that we omitted in the above description worth attention, such as the *cost*, size of the *user community*, *maturity* of the product, quality of *documentation*, availability of *didactic materials*, *learning curve* slope, *time* requirements, and other.

3 RELATED WORK

Using microcontrollers in all kinds of higher-education scenarios is not uncommon theme in the literature. It has high potential in various branches of technology, such as chemistry [1] or systems engineering [2]. Most authors advocate for hands-on approach, which is today a common way to teach with microcontrollers [3], although it requires organizational, human, material and maintenance resources. Some scholars even proposed using microcontrollers as the first programming course [4]. In some cases, teachers are willing to spend time and efforts to design their own customized kits that support their teaching goals using microcontrollers [5]. Currently, some teachers who found poor results of their students in object-oriented programming are trying to use microcontrollers to improve computational thinking skills of the students in order to have better background when taking the OO course [6]. As we will discuss in the forthcoming sections, using microcontrollers can be useful both in a course with pre-designed and streamlined tutorials and tasks as well in completely open-ended seminar courses.

4 PRINCIPLES OF COMPUTERS – HARDWARE: TATRABOT

In the Applied Informatics bachelor study program at our faculty, students need to complete three obligatory courses in a series called Principles of Computers: Hardware (H), System Programming (SP), and Operating Systems (OS). In H, they learn about logic circuits fundamentals, while in SP, they learn

basics of computer architecture, about CPUs, and assembly language. In order to prepare the students for the low-level assembly (the main programming courses are in a high-level Python) and at the same time to make their entry to the study program more fun, we added a module to the H course, where they programmed robots in C. Robots are equipped with simple STM32 microcontroller, different sensors, LEDs, two motors with encoders and a software library running on top of ChibiOS/ChibiStudio that provides high-level API for controlling sensors and actuators. The robots are of our own design, built from individual parts around a cheap robot chassis that we modified for our need [7]. At the same time, they receive a short (about 4 weeks, 4 hours/week) introduction to C programming with a lot of exercises to solve. Students work individually in pairs on solving about 25 robot tasks and whenever they reach a solution, they demonstrate it to one of the instructors to have the points assigned. They can also ask the instructors at any time for help when they are struggling with finding the solution. Thus the role of the instructor was observation and guiding whenever required. There were typically two instructors in groups of size 20. Tasks are not very difficult, the most difficult were about recognizing different patterns of black lines taped on the floor or navigating a line maze. Since the microcontroller we have used only had very few ground and power pins, we had to build and solder a separate circuit board for distributing power and ground connections to all the connected elements. To build one piece of robot took us several hours, it required cutting some supporting elements from plastic frames and the cost of parts was about 60 Eur. Most students solved most tasks successfully, although some of them required several trial-error iterations, or a prior discussion with an instructor. Students were already experienced in C-language syntax and thus they could write the code relatively fluently. Work was organized in longer sessions up to 4x45 minutes, sometimes shorter when it was combined with the C-exercises. With a several years break, we have collected answers to a questionnaire, Figure 1 right shows the distribution of their answers.

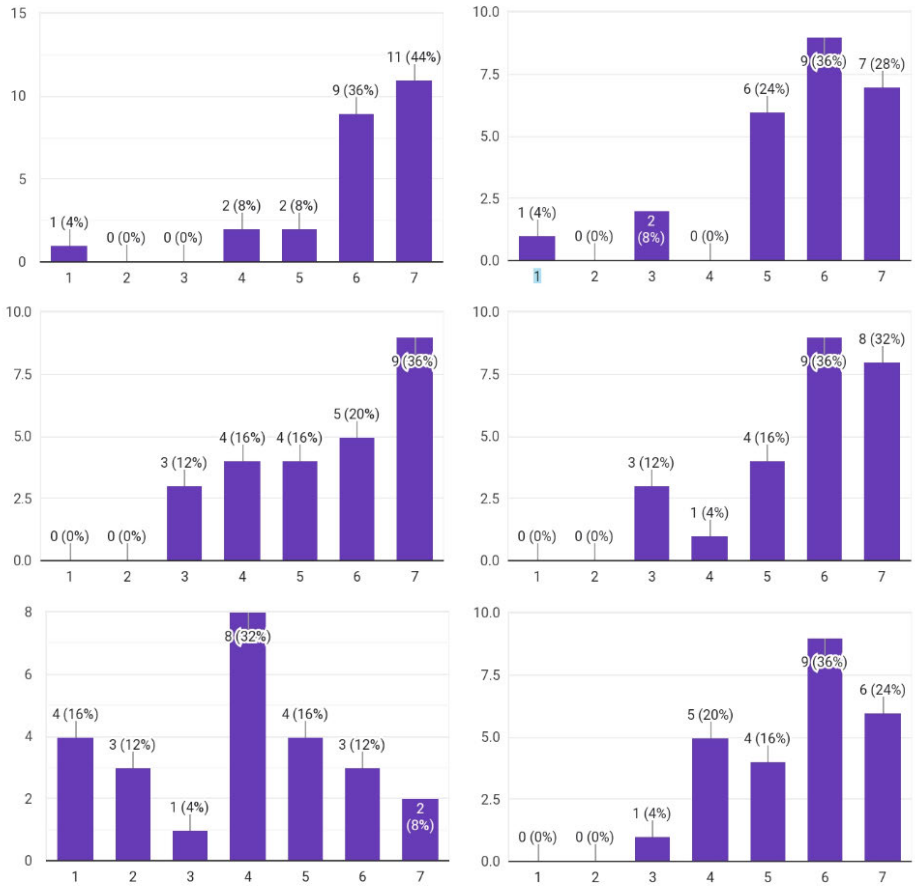

Figure 1. Survey results: course with Tatrabot – left, course with Arduino – right.

5 PRINCIPLES OF COMPUTERS – SYSTEM PROGRAMMING WITH HYBRID LEARNING

After two years, the module was moved to the second semester for the reasons of interference with learning Python. For many students in the first semester, computer programming was completely new, and learning two languages at the same time (Python and C) was causing confusion and syntax mixtures. It was also an advantage for us, because some students always realize in the first semester that the study program is beyond their possibilities, so we did not have to prepare hardware for those. Finally, in the second semester, students were more familiar with fundamental programming principles, so we could have spent more time with robots.

During the pandemic in spring semester of 2021, the school went on-line and working in the lab was not allowed. We had changed the concept and ordered sets with Arduino, and various electronic parts for all approximately 70 students. List of parts: Arduino nano, passive buzzer, 3x LED and resistor, tiny breadboard, microphone, IR distance sensor, servo motor, capacitive touch sensor, joystick with a button, jumper cables of different types, USB programming cable. We prepared video-tutorials explaining how to connect and program each of the parts. Students worked at home individually and the instructor was available for a remote consultation. Students submitted short videos demonstrating they solved the tasks together with source-code. In both setups – with Tatrabot and with Arduino, students learned about C language, and about controlling sensors and actuators, however, in the setup with Arduino, they did not have that library of high-level operations to control the robot as they had for Tatrabot. In consequence, they had to understand the communication at the level of signals, which was not the case with Tatrabot. On the other hand, they only worked with individual parts, not the whole robot, which was more fun, but we had to keep the cost low (one set was less than 10 Eur). And working with hardware is usually fun for most computer students anyway. Since the students had more time at home to solve the tasks, we have included more advanced tasks to better differentiate and give challenges also to the best students. That can partially explain slight differences in the questionnaire results. Examples of more advanced tasks: measuring noise level, measuring tone frequency or recording and visualizing wave shape using microphone – this typically required calibration and signal processing/filtering. See Figure 2 for an example of most successful solution.

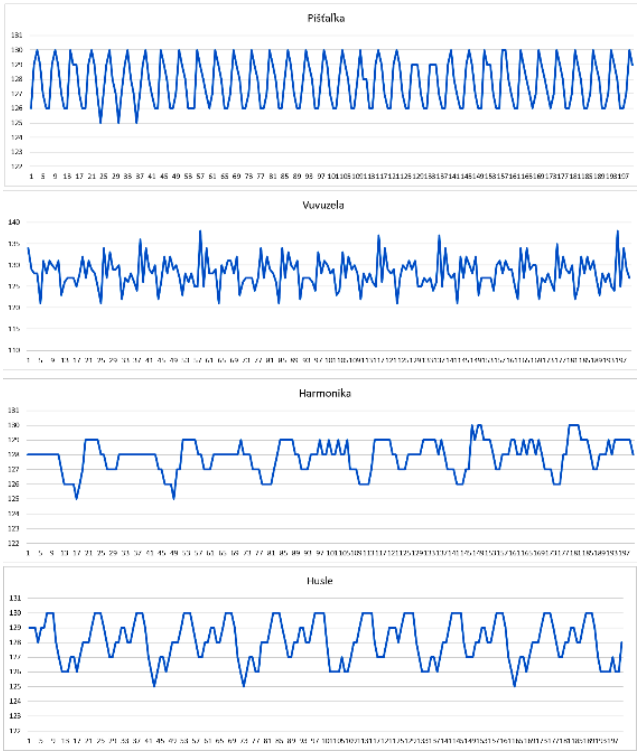

Figure 2. Visualizing sound wave recorded with analog microphone and Arduino, 4 different instruments: whistle, vuvuzela, accordion, violin.

We have collected answers to a questionnaire of both groups early 2022 (i.e. after half a year break for Arduino group and after a year or two break for Tatrabot group), and the results are shown in Figure 1, the questions in the three rows in the figure were as follows: 1) “Inclusion of Tatrabot/Arduino to this course has been an attractive and enlivened“; 2) “If I were to return to those tasks today, I could solve most of them, especially if I had some time to recall some details“; 3) “I can imagine solving similar tasks without Tatrabots, but with microcontrollers at home with a remote support of an instructor” (left); and “It was not a significant obstacle for me that I was solving these tasks at home with a remote support of an instructor”. A scale 1-7 in all cases 7=completely agree, 1=completely disagree. As we can see, students generally welcomed the module, real robots were a little bit more attractive than microcontrollers, and they feel confident about what they have learned. About a quarter of the students answered that they had not meet microcontrollers before this course (76%, 72%), and majority answered that they would recommend microcontrollers to their friend or relative (84%, 88%). From the few students who did not give high agreement score, they wanted even more support and did not feel so confident working on their own.

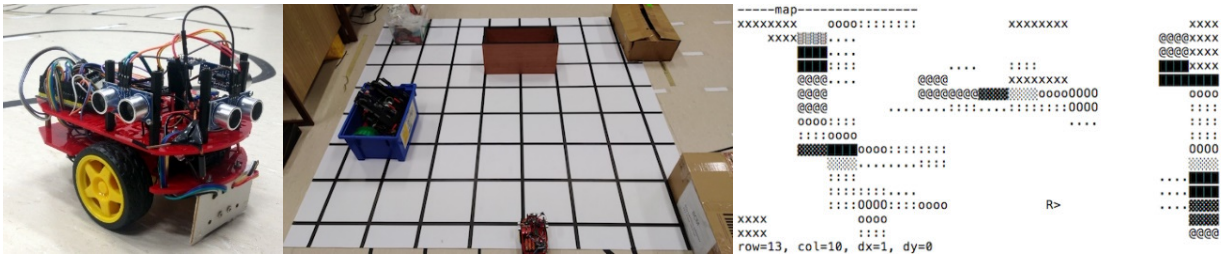

Figure 3. Mapping a grid world with ultrasonic sensors, a modified Tatrabot robot.

They also struggled with the third round of the tasks. The tasks were arranged in three groups in increasing level of difficulty. The requirement was to complete at least 70% of tasks, however including various programming exercises in C. Some of the tasks counted as extra bonus points, i.e. it was possible to reach more than 100%. About two thirds of students who passed also had 70% for microcontroller part of the grading and around 15% of students earned more than “100%” of points for microcontroller tasks. After this course has been implemented, we have observed slight increase of topics of bachelor and semester student projects where microcontroller was involved. In the next year (2022), we faced the question whether to return back to Tatrabot platform or repeat the on-line scenario from 2021 with Arduino. Since the situation with the development of the pandemic was not clear and since we believed the Arduino platform allows the students to learn and create more connections about “principles of computers” – as the title of the course asks for, we chose to repeat the microcontroller setup for another year. However, the prices of Arduino modules went up by about 100%, so we chose a different platform MH ET Attiny88 modules that can also be programmed from Arduino IDE. For the kind of tasks performed, the lower resources: 8KB of program flash (instead of 32KB of Arduino) and 0.5KB of data RAM (instead of 2KB of Arduino), one missing timer, and missing serial port (that we replaced with low-cost USB-serial converter) did not pose any significant difference, while pressing the price back under 2.50 Eur for a module including the external serial converter. We had to fix a bug in the analogRead() function of the attiny88 software provided for these modules, and the modules had a tendency to loose the bootloader when students clicked RST button at wrong moment, but both issues did not mean a significant obstacle.

6 ROBOTICS SEMINARS

While the introductory course is a lot of work and lot of learning, the most valuable for us are the projects in robotic seminars and more advanced robotics course. These are completely open-ended creative projects with goals that are formulated in the interaction with the students and that involve individual work with the instructor with the student or student groups in robotics laboratory, where we accumulated various experimental robot platforms over the past 14 years.

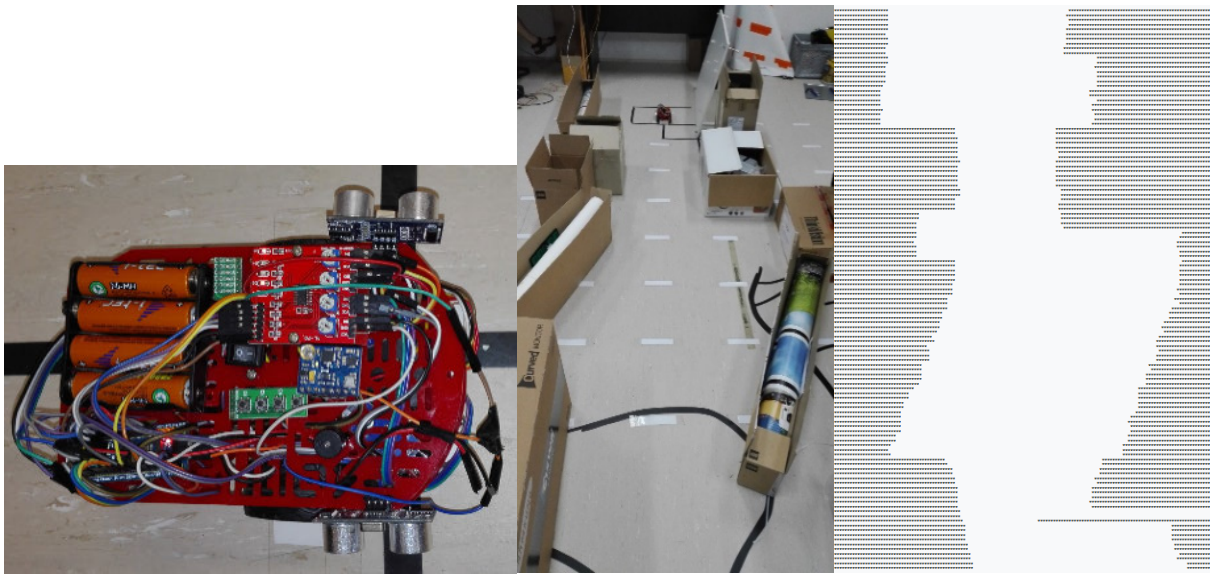

Figure 4. A 1D mapping example with Tatrabot robot.

The participants of these seminars are either cognitive science master students, or applied informatics students at an arbitrary level. The project is always tailored to both the skills and interests of the students. While cognitive students come with various backgrounds – such as psychology, linguistics, and philosophy, they often need detailed guidance and help with practical software implementations. On the other hand, they usually have more interesting ideas and ambitions, and thus achieve more interesting results (with much more efforts required from the instructor). Figure 3 shows an example of probabilistic mapping of a grid world using a modified Tatrabot robot (added ultrasonic sensors). The resulting map is drawn in semigraphic ASCII characters on a BlueTooth console.

Much simpler example is shown in Figure 4, where the robot is mapping a corridor with walls of different shapes using two ultrasonic sensors. The motivation of this task was to demonstrate how does ultrasonic sensor respond at various angles, corners and whether it can provide any useful results in this context.

One of the most advanced projects is shown in Figure 5. A humanoid robot Lilli was constructed by two students from a kit designed by a Norwegian artist. We have later equipped the robot with our own controller, software, and added ZED Mini stereo vision module and Nvidia Jetson TX2 GPU computer. In a follow-up project, group of 2 students made the robot be able recognize the location of red cubes and reach after them with its hand.

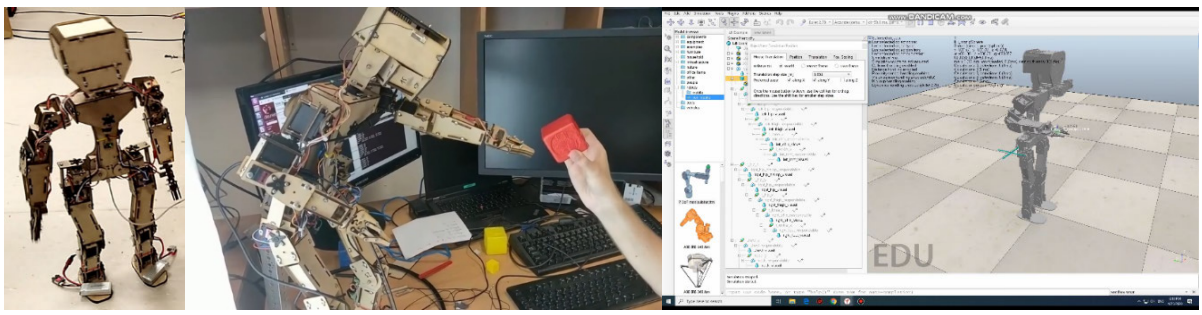

Figure 5. Humanoid robot Lilli with 25 dof (left). Stereo vision and simple inverse-kinematics experiment (middle). Simulator of Lilli (right).

Later a complete simulator in CoppeliaSim software with more proper inverse kinematics was implemented as part of master thesis of Gabor Halasi [8].

Students who take our seminar course sometimes participate in further activities. For one, those are various robotics contests, where they help us as referees, or participate themselves. Other times they design robots that can be used in other educational settings. Figure 6 shows the MoKraRosA robot [9]

designed in Fusion 360 by one of our students and driven by Arduino Nano. 80 copies of this robot were built in three summer camps that we organized with Fablab Bratislava in the summer of 2019.

Cooperation takes place in opposite direction as well. Figure 7 shows robot 100(1+1) that was designed by a talented member of our robot club in elementary school for a robot competition. It has an array of 8 line sensors, an IR distance sensor and a BlueTooth module so that it can be interactively controlled by a remote computer. We have used it with cognitive science students in two mapping scenarios: producing topological and metric maps.

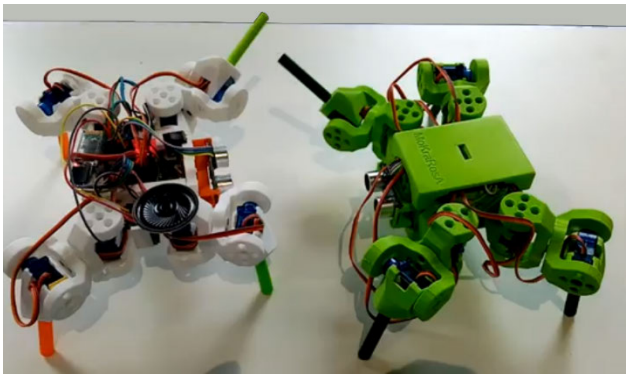

Figure 6. MoKraRosA, version 1 (left) and version 2 (right). Version 2 has been designed after the summer camps and provides a "shelf" for every electronics part included.

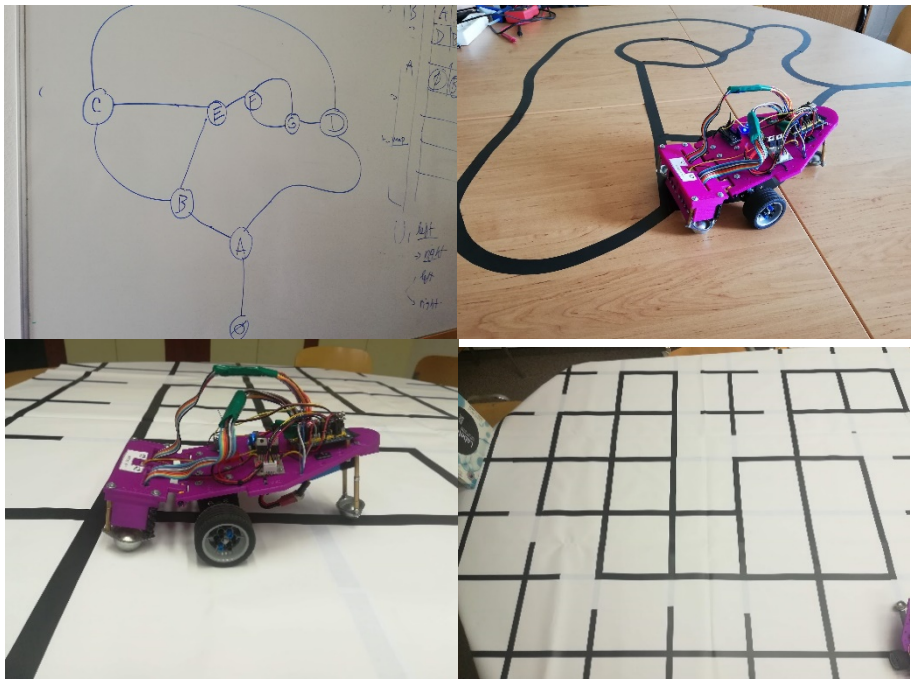

Figure 7. Robot 100(1+1) controlled by Arduino. Experiment with interactive topological mapping (top), user selects next route in each crossing. Fully autonomous metric mapping on a grid world (bottom).

Another fruitful cooperation is based in association Robotika.SK with our colleagues from Slovak Technical University. For example, they have designed educational robot Acrob [10] controlled with Arduino-compatible board, and we have used this robot in many Arduino workshops and robot seminar projects. An example of such student project with metric mapping is shown in Figure 8. Java application written by students connects to the robot over BlueTooth and draws the map in its graphical window as it is being discovered by the robot. The software can also navigate the robot in the constructed map.

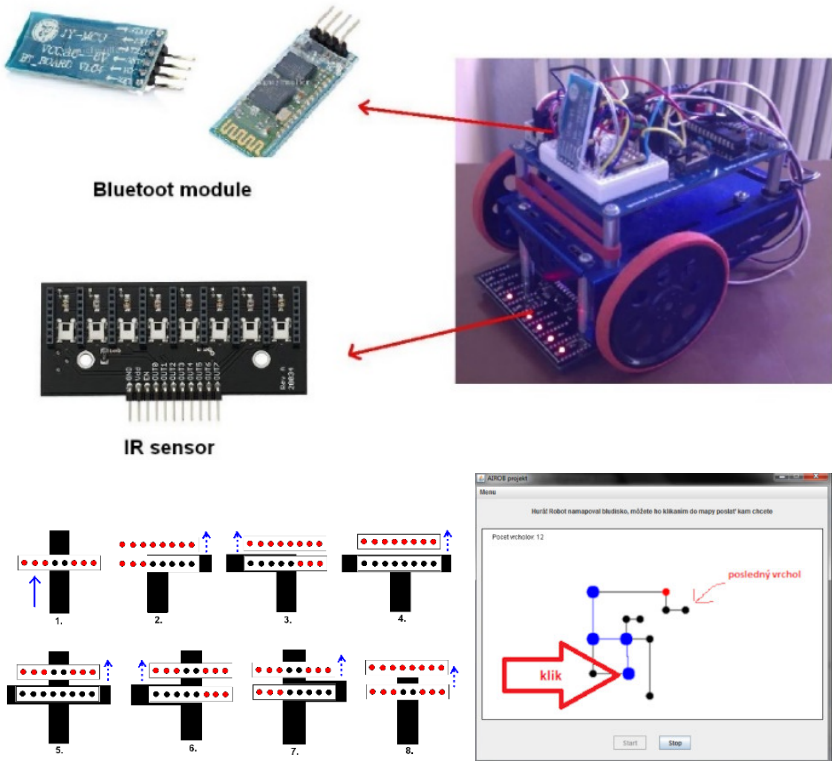

Figure 8. Robot Acrob metric mapping. Analysis of possibilities of sensor readings (bottom left) and the output window with clickable map (bottom right).

Various student projects involve 3D modelling – either from scratch or by modifying/improving designs found in the Internet. One such example from the last year is a cat-like robot Kocúr v Čižmách shown in Figure 9, and it was used in three semesters in three follow-up projects. In the first, it was adopted from previous design and modified, constructed and programmed for simple movements. In the second project, walking pattern for the robot was found, and a more complete software was implemented. In the third project, the robot was further modified so that the front part with the legs can rotate for much better maneuverability. Design of the legs was also improved to make them easier to move but still keeping the segments together strong.

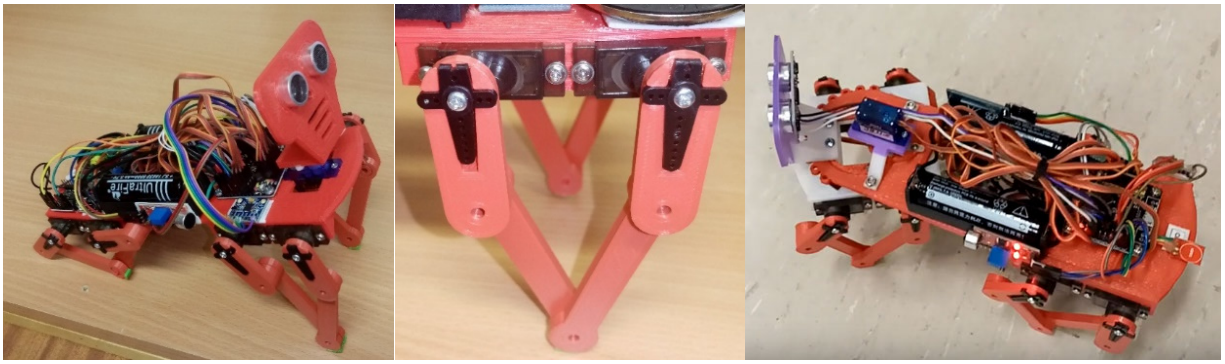

Figure 9. Robot Kocúr v Čižmách with 9 servomotors that walks. Version from the first project is left, version from the third follow-up is right.

One of the most advanced robots in our group is robot Nico [11]. An Erasmus student from Spain came up with an idea of constructing our own virtual reality glove to steer the robot. We have combined the usual bend sensors with 3 IMU sensors containing 3 gyroscopes and 3 accelerometers so that each dof of the robot hand (10 dof in total) can be controlled by movements of human hand and arm. This required providing the student with an advanced solution where three Arduino computers are connected in a communicating chain and transmit the state of all sensors over the BlueTooth to a PC. First, we ran experiments to test the capabilities of the IMU sensor (IMU 6050) with a simple webpage that visualized

3D model rotations according to the movements of the IMU sensor. And then together with the student discussed, designed and tested the overall solution, see Figure 10. The glove was then later used in one bachelor thesis of a different student in selected experiments, and we plan to improve it and use it in our research project.

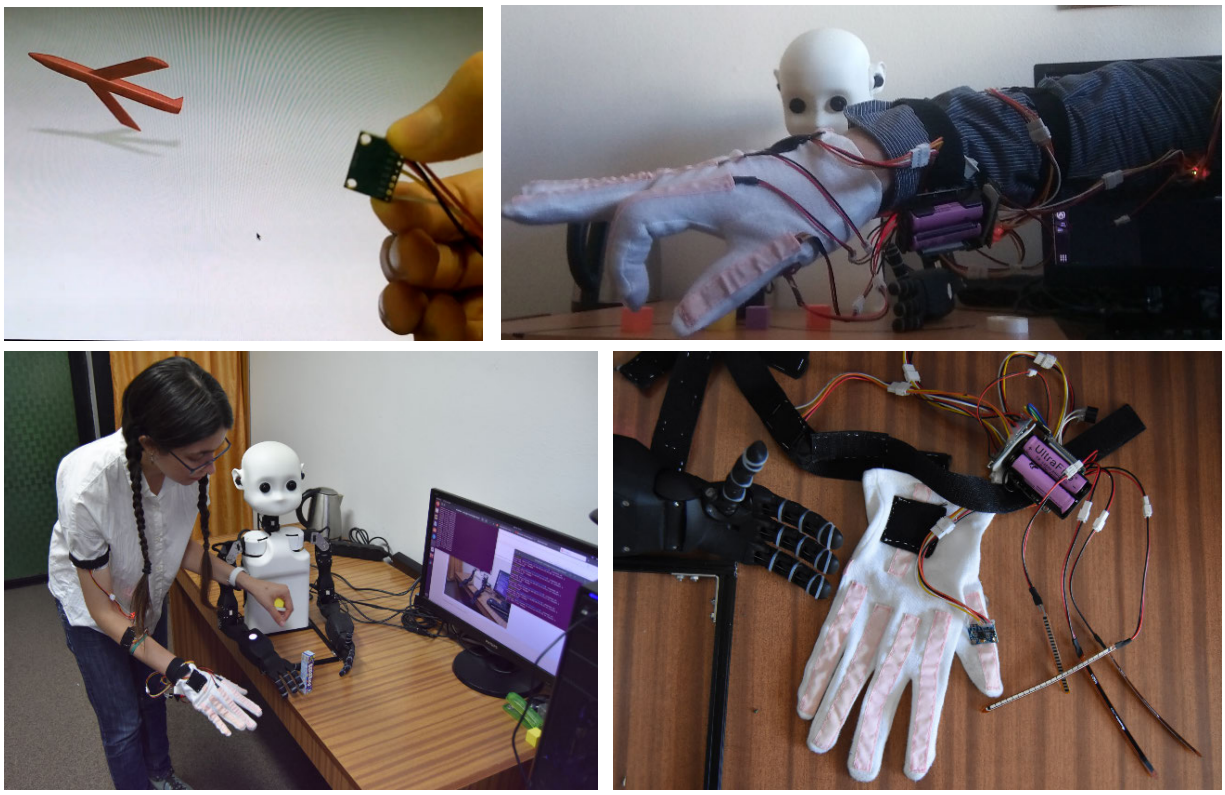

Figure 10. Virtual reality glove with bend sensors and IMU controlling hand and arm of robot Nico. See https://dai.fmph.uniba.sk/projects/nico/

7 CONCLUSIONS

In the courses and student projects we have discussed, we used microcontrollers as the platform for different educational robotics learning scenarios with college level students. Either the usual Arduino Nano or STM32 microcontrollers were used. For the STM32, the ChibiOS/ChibiStudio framework while providing more rich features (such as multi-threading, i/o drivers) is much more difficult to program in for novice users, and thus we provided a library with high-level functions that the students used without having to understand the underlying details of the software. However, they have also lost the possibility to understand the programming of microcontrollers at this low level. In another course trial of the course with Arduino microcontrollers, students were required to learn at that level, but due to the very user-friendly and easy to use Arduino platform, they have completed the tasks successfully. We have used microcontrollers in workshops, after-school clubs, and summer camps as well, and the open-endedness, standardized interfaces, unlimited possibilities of the modularity and extensibility of this platform made it the right choice in all these situations. As we have shown, it also led to natural interchange of resources, and outputs between various groups of learners for the benefit of all of them. In addition, by being exposed to programming the microcontrollers, students experienced this low-level platform in a way that is similar to real industrial applications, and thus they have a much better idea about what is going on inside of various embedded devices that are appearing all around us in greater amount. Links to the projects described in this paper can be found at our group website kempelen.dai.fmph.uniba.sk/rg.

In one of the tasks of the Principles of Computers course, we gave the students a Visual Studio project that connects to the serial port (i.e. over BlueTooth connection), contains a simple communication protocol and allows interaction with the microcontroller from their own application running on the Desktop. Their task was to visualize sensory readings made by the module on the computer screen. In the future, we would like to elaborate in this direction and create a simple framework/library that they can use to create distributed applications that run on both platforms simultaneously, although we may use it in a different course.

REFERENCES

[1] G.A. Mabott, "Teaching Electronics and Laboratory Automation Using Microcontroller Boards", in *Journal of Chemical Education*, vol. 91, no. 9, pp. 1458–1463, 2014.

[2] M. F. Silva, A. P. Dias and M. T. Costa, "Using Robotics to Teach Systems Engineering: A Hands-on learning course example", Human-Centric Robotics, pp. 161-168, 2017.

[3] D. Ibrahim, "A New Approach for Teaching Microcontroller Courses to Undergraduate Students", Procedia - Social and Behavioral Sciences, vol. 131, pp. 411-414, 2014.

[4] W. D. Lubitz, “Rethinking the First Year Programming Course”, in *Proceedings of CDEN/C2E2 Conference*, 2011.

[5] Y. Li, "Teaching embedded systems using a modular-approach microcontroller training kit", *World Transactions on Engineering and Technology Education*, Vol.6, No.1, 2007.

[6] M. T. Fülöp, J. Udvaros, Á. Gubán and Á. Sándor. "Development of Computational Thinking Using Microcontrollers Integrated into OOP (Object-Oriented Programming)", *Sustainability*, vol. 14, no. 12, 2022.

[7] P. Petrovič, "Tatrabot - a mobile robotic platform for teaching programming", in *Constructionism in Action*, 2016.

[8] G. Halasi, *Humanoid Robot Lilli*, Master thesis, Faculty of Mathematics, Physics and Informatics, Comenius University Bratislava, 2020.

[9] P. Petrovič, J. Vaško, "MoKraRoSA: A Constructionist Platform for All Ages and Talents", in *Proceedings of the 2020 Constructionism Conference,* pp. 413-427, 2020.

[10] R. Balogh, "Laboratory exercises with Acrob robot", in: *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, pp. 41-46, 2011.

[11] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "NICO—Neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction", in 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 113-120, IEEE, 2017.