An Open Solution for a Low-Cost Educational Toy

Pavel Petrovič[1(✉)] and Jozef Vaško[2]

[1] Comenius University, Bratislava, Slovakia
ppetrovic@acm.com
[2] Fablab, Bratislava, Slovakia
jozef.vasko@fablab.sk

Abstract. In the summer of 2018 we organized two 5-day summer camps each for 20 children aged 11-15 in Fablab Bratislava. In addition to learning about 2D and 3D modelling and Arduino programming, every child has built and experimented with a humanoid toy robot. For this purpose, we have developed a flexible and extensible software solution that easily transitions and scales up to any other toy or even advanced robot controlled by Arduino. In this article, we describe our experiences from the camp as well as some other makers events and efforts and give details on the respective framework and discuss the role of makers movement in the educational process.

Keywords: Dtdt · Otto · Fablab · Makers · Arduino

1 Introduction and Related Work

Young people growing up in this decade benefit from an easy access to and availability of information and technologies of all kinds thanks to the development of Internet. Youngsters who have a deep interest in a particular hobby, subject, a challenge or a specific project, and who have the time, enthusiasm, energy, and a goal mindedness are in a much better starting position to enter and advance along the learning trajectory in a fast pace than ever before. Yet, paradoxically, the amount of information available works counter-productive when the children are facing a difficult question of where to start, how to digest information that is too complex to start with and how to find in the sea of the options – tasks, challenges, projects, and ideas – those that are suitable for their actual knowledge and skill levels. Parents and other adults who face their own challenges in a demanding world are seldom able to follow up on the developments and fail to provide the youngsters the shared time and the opportunities to let them grow from experiences while working with an appropriate learning material.

One possibility – often used by the parents and sometimes schools – is to rely on good quality solutions, example of which are LEGO robotics educational sets and programs. Their price, however, makes it inaccessible for many. Furthermore, the marketing strategies of such producers are often somewhat limiting in terms of

M. Merdan et al. (Eds.): RiE 2019, AISC 1023, pp. 196–208, 2020.
https://doi.org/10.1007/978-3-030-26945-6_18

variability, good modularity, interoperability, portability, open-hardware and software, and the software stability consequently. We feel that the development cycle of 6-8 years[1] is too slow to reflect on the fast technology developments and educators' needs. Thus meanwhile they are seconded by copyright breaching, but technologically superior sets such as KAZI EV5 sold for a quarter price (although possibly not in the same production quality). There is never enough of the credit to give to the robotics set producers for the valuable work they do on these useful educational tools, it is hard to believe that in the highly flexible production factory processes of today it is so difficult to provide a broader selection of different options that would better fit various needs given by the educational goals of the customers. Many alternate solutions are fragmenting the user base meanwhile, an interesting example of such being [9].

Educational programs such as FIRST LEGO League or RoboCup Junior provide an excellent escape from the lack of opportunities. However, peeking at the results from the regional tournaments, we see too many teams achieving too low score. For example, in 2018, from 98 teams participating in 7 regional tournaments of FLL in Slovakia only 30 achieved more than 100 points and 29 achieved less than 50 points suggesting that many teams do not allocate enough time or efforts needed for the learning transfer to be effective. Team performance often depends on the presence of a necessary catalyst, in this case a skilled team-leader providing the technical, motivational, pedagogical guidance. In addition, relatively short meetings duration combined with low meeting frequency further prevents efficient progress. The same material, skills and ideas have to be re-acquired multiple times after having been lost. Sometimes it takes tens of minutes to be able to follow-upon on the previous meeting. Especially when the meeting club must tidy up the room after every single meeting and stow the equipment and the work in progress somewhere in a storage cabinet.

Somewhat more effective regime with faster, yet longer-lasting learning progress can be achieved in summer schools and summer camps. The positive impact of summer schools on the future skill level and performance of participants is well known [8].

For more than 30-years a group of enthusiastic students, professionals, and training staff members organizes a two-week electronics summer camp for talented children (LSTME – Letné sústredenie talentovanej mládeže v elektronike, lstme.sk), where both authors have participated several times as leaders or instructors. From our personal experience, working both as a leader in an afternoon robot-club and as an instructor in this summer camp, we claim with confidence that skills of a typical strongly motivated child advance further in the camp than during a full year of active and periodic participation in an afterschool robot club. Intensity of the learning in the camp is extremely accelerated by the presence of about 15 leaders and instructors - experienced technicians who have answers to all the questions of the curious young mind. Adding up to that they also bring various hardware and workshop equipment to the camp making it possible to demonstrate the ideas in practical realizations: well-defined or even open-ended projects. Disposition of the camp typically includes a full electronics workshop for producing PCBs, LEGO robotics workshop, 3D printing workshop,

[1] MINDSTORMS releases: RCX: 1998, NXT: 2006, EV3: 2013.

computer room and audio-video studio. It takes place in the heart of nature and thus interleaving laboratory work with a relaxing time and sports in the beautiful environment.

One of the authors is a leader and manager of Fablab Bratislava – a fabulous laboratory where everybody can come to realize his or her dreams using workshop equipment such as 3D printers, laser cutters, vinyl cutter, automatic sewing machine, miller machine, and more. Inspired by our experience from LSTME, we decided to organize a day-camp in the space of Fablab Bratislava. We named it Denný tábor digitálnych technológií $(DT)^2$ – a day camp of digital technologies. We had several goals in mind when preparing the camp: (1) to let the children have a hands-on learning experience with digital technologies so that they will understand their principles, purpose and use and that they will be capable of fabricating various designs, potentially coming back later and extending the Fablab users family, (2) show them the whole process of completing a full project resulting in a real product that they take home and can continue using and tinkering with it later on, (3) give them sufficient background on 2D and 3D modelling and Arduino programming so that they understand the complete process of a design and development of a novel prototype.

Further sections of this article describe some related work, the individual activities of children in the camp, the robot that the children built and the framework that we have developed for the robot and finally summarizes our thoughts on the role of such activities in the educational process.

2 Organization

Children were divided into two groups of 10 based on their age and skills. These two groups alternated between two workshops: (1) 2D and 3D modelling and (2) Electronics and programming. The day was started with a social warm-up game while we waited for everybody to arrive, followed by the morning session lasting about 4 h. After the lunch, we spent some time outside, playing games and easy sport such as discgolf. In particular, we have used the activities from the Systems thinking playbook [1] that helps people develop a systems thinking perspective when observing the world or solving problems in a funny and gentle way. In the afternoon, the groups have exchanged the rooms, and continued in another workshop session of about 4 h. In the middle of the camp, we visited the laboratories of the National Centre for Robotics with live demos of manipulators, 3D scanners, large mobile robots and industrial applications. At the end of the camp, all children presented their results and products to the whole group, see Fig. 1. We feel the sharing, and enjoying the sharing is among the most important principles in educational activities.

Fig. 1. A happy participant in a final presentation and children playing with Ottos.

3 2D and 3D Modelling

The first group of the participants learned about 2D modelling by creating various designs in Inkscape open-source software. Every participant drew a picture that was printed on vinyl cutter and ironed on a T-shirt, and they also produced fabric bags and tiny items such as jewelry and souvenirs. They were introduced into the world of 3D design using the TinkerCAD software. Every participant has designed a fairy-tale 3D scene (Fig. 2).

Fig. 2. Example designs from the 2D workshop.

4 Introduction to Arduino Platform

One of our primary goals was to show the children a technology they would be able to use at home or their clubs after they will have returned from the camp. Arduino platform is available at very reasonable prices, has a huge user-base and a community with solutions to almost any challenge that an electronic hobbyist typically could encounter. Every participant received the following equipment for experimentation:

Arduino Nano board, Arduino Nano Expansion Board, USB mini cable, and a set of sensors and servo-motors. We have prepared a set of challenges – little projects on which we demonstrated the various elementary features of the Arduino platform – such as digital and analog inputs and outputs, PWM control as well as the language features – expressions, variables, arrays, statements, conditions and loops.

A typical project consisted of a challenge that we solved together, while explaining the principles. Next, the participants were asked to make modifications, improvements, and solve similar challenges. These included – flashing LEDs on LED panel, alarms, reacting to sound intensity, remembering and replaying a clapping rhythm. In some activities, we combined LEGO parts with Arduino and servo. For example, designing a ticking clock, or making the FLL mission models to move on their own.

In addition, we presented a set of additional optional projects to the participants that demonstrated further features and that were meant for those who are confident with the acquired or previous experience and were seeking more information. We have used some of these ideas in our framework as well. This included: using interrupts to respond to ultrasonic sensor, harmless software serial communication (see below), playing more complex melodies described in easy sequence, playing tones and melodies in the background (Fig. 3).

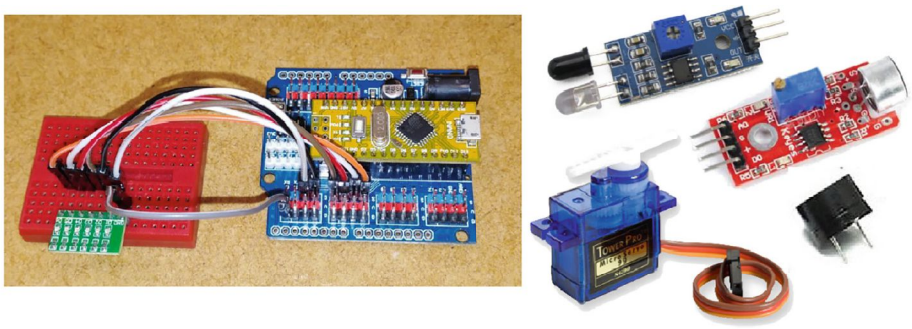

Fig. 3. Parts used in the Arduino challenges.

5 Building Otto

We used a design that was inspired by [2], but we wished to make it more human-like. We added the arms of the same shape as the legs, except of making the arms a little bit wider in order to increase their distance from the body. We have produced both a version for the 3D printer and a version for laser cutter. The main difference from our point of view was the time required for producing the parts – about 12 h for 3D printing compared to about 30 min of plywood laser cutting. The difference increases if the production process fails for any unspecified reason. Since we really wanted to prepare parts for all the 40 participants, we chose the laser-cut version to save the time. Another difference between the two designs is the way the joints are attached to the servo-motors. In the 3D printed version, the motor is attached by a single screw,

whereas in the plywood version, servo horn is attached to the motor, and the horn fits inside a wooden part to smoothly control the turning. The original author of Otto has later improved the 3D version too. The plywood version contains more parts and requires more advanced manual skills. We have therefore used the 3D printed version in the group with younger and less experienced students (Fig. 4).

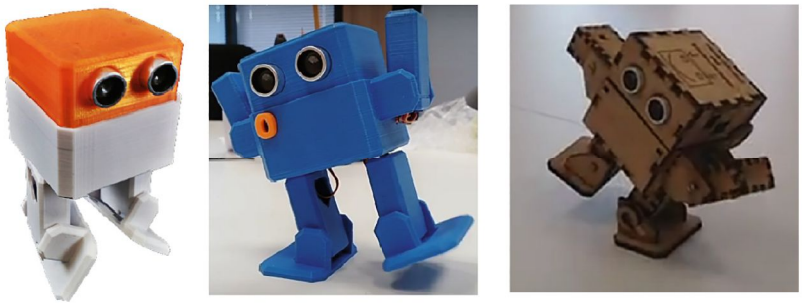

Fig. 4. The original [2] and our two versions of robot Otto.

Once the participants acquired the elementary programming skills and began to be fluent in using Arduino platform and programming in its C++ language, they received a bag containing laser-cut plywood or 3D printed parts, screws, battery pack, wires, and electronic parts and they began to build their own robot Otto. The building process took the whole 4-h session, but every participant managed to build it. The process is documented at the website of the day camp [3]. For the wooden version, participants used hammer to align the parts properly, and a glue just for a very few connections with otherwise loose contact (Fig. 5).

Fig. 5. Building robot Otto from parts.

Otto has been prepared for use of the Arduino Nano with its expansion board, which fit nicely into its head. Apart from that, we chose to design the electronics in our own custom way and write all the software completely on our own.

The following parts have been used (in total including the material worth about 20 Eur): 6 pcs. SG-90 micro-servo, Arduino Nano with ATmega328 (and mini/micro USB cable), Arduino Nano Shield I/O Extension Board, 4 AA batteries holder, passive buzzer, ultrasonic sensor HC-SR04P, at least 10pcs female-female 10 cm jumper cables, HC-05 Bluetooth module, SB550A Schottky diode – or similar with about 0.5–0.7 voltage drop, minimum voltage 10 V, and current 3A, 1000 uF capacitor – or similar to filter out high current demands, KCD11 power switch, 4pcs screws M3/5 mm, optionally: DFPlayer mini mp3 player, 8 Ohm 1 W speaker.

5.1 Powering and Wiring the System

A very nice feature of the Arduino Nano Expansion Board is that it provides multiple pins with GND and 5 V connections, allowing easy connection of many sensors, servos, and other devices. Unfortunately, all these 5 V pins rely on a single power regulator, which is capable of delivering 1.5 A current. The power consumption of 6 servo-motors, Bluetooth module, ultrasonic sensor, and other devices summed up indicates higher demands. Therefore, we ought to skip the DC power input of the expansion board, its power regulator and built-in capacitor and connect the batteries directly to the 5 V pins, ensuring the highest possible current supply. Unfortunately, that would breach another limitation – the absolute maximum ratings of the Arduino board (and in fact most TTL electronics) voltage, which is stated as 6 V in the data-sheet. A pack of 4 AA alkaline batteries when fresh (each producing 1.65 V) gives 6.6 V in total, which is unacceptable. We have therefore inserted a Schottky diode, with about 0.5–0.7 voltage drop to clamp the maximum within the limits. Since the original capacitor was also circumvented, we added another high-capacity 1000 uF electrolytic capacitor. This part of the process as well as installation of the ultrasonic sensor in a more space-efficient way required a little bit of soldering. Most of the participants had no soldering experience, and this was a nice opportunity for them to see it and try it for the first time, while we observed the process and made sure the resulting connections are fine.

The connections to the buzzer, Bluetooth module, ultrasonic sensor and optional devices were made with usual jumper wires.

6 Open Software Framework

To satisfy one of the goals stated in the introduction, we intended to create a more sophisticated control framework for the Otto robot so that the participants do not have to invent their own complete code for the Otto robot. The latter option was not possible due to a short time remaining in the stage of the camp after they have built their robots. Yet, with the background they have acquired in the first part of the week, they should be able to make modifications in the code and tune it to their needs and desires. We

wanted to give the children the ability to create their own choreographies in a simple manner so that they could also be easily shared with others.

6.1 Calibration

When building the robot, servos are screwed in at some particular angular position while the leg or arm is turned to a particular configuration. The builder has to observe that the full range of movement can be achieved for each degree of freedom. Yet, it is very difficult to mount the motors at the same exact position on every robot and the servo motors themselves can exhibit somewhat different behavior as such. To compensate for that and to produce the very same results when dancing according to some shared choreography, each robot should be calibrated. The semi-automatic calibration procedure involves manually tuning the position of each leg to a standardized position and then storing the calibrated values into EEPROM memory so that the robot will remember the calibration even when powered off and on again. In addition, some children have made a mistake when mounting the servo motor on the leg, turning it around – getting a fully functional robot, but with one (or more) servos reversed. To deal with the situation, the calibration also stores the orientation of each servo. When dancing according to some choreography, the actual value is recomputed to match both the servo shift and the servo reversal. The calibration also stores the allowed range for each degree of freedom, which is enforced when controlling the robot movement manually.

6.2 Bluetooth Communication

Arduino boards are typically programmed in a very convenient way through built-in USB to serial converter and using a built-in bootloader program, and we like to keep up with this standard. However, unfortunately, the ATmega328 microcontroller only has a single USART hardware port. Connecting the Bluetooth module to its Tx, Rx pins interferes with the programming and the module must be disconnected each time before the programming takes places. That is a no-option, since the robot would have to be opened each time for the program download. In addition, we would like to keep the Bluetooth communication in a separate channel so that the wireless communication link does not have to be re-established every time we want to update the program with a new version. A possible solution is provided by the SoftwareSerial library for Arduino. Unfortunately, this library is implemented in a poor way: during the communication on the serial line, the global interrupt flag is disabled, meaning the timer-generated PWM signals by the Servo library for the movements of the legs and arms get extremely distorted, resulting in chaotic and unpredictable movements. We have therefore implemented our own "software serial" communication that relies only on the pin-change interrupt, which is available on most Arduino pins. The challenge is that every byte transmitted may have a different number of pin change interrupts and often even after the last pin-change interrupt occurs, we do not yet know, which byte is being transmitted. Consider, for instance, two different situations: receiving byte $191 = (10111111)_2$ vs. receiving byte $175 = (10101111)_2$, see Fig. 6.

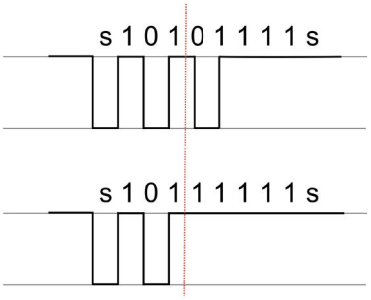

Fig. 6. Ambiguity within the pin-change interrupt routine.

After reading bits 0-2 (101), we will see no more pin change interrupts in the case of 175. We must leave the buffer in a "quantum" undecided state and infer that the received byte is 175 only after we will have received a request to read the next byte from the port. If that request, however, comes before the expected time of the stop bit, we may not be sure, and we must report that no new bytes are available yet. Otherwise, we can conclude that the received byte must be 175. The full implementation of this efficient and non-disturbing software serial algorithm is demarked in the source code [4].

6.3 Playing Melodies in the Background

A simple passive buzzer is a very low-cost solution to add sound to an Arduino project, and even though the sound is a bit loud, it can play nice melodies. Arduino has the built-in *tone*() function for producing sounds at specified frequencies of a specified duration in the background, using timer 2. Unfortunately, this plays only a single tone, and we need to play the full melodies. And since all Arduino timers are already occupied (0 – system time, 1 – servos, 2 – tone), there is no remedy. We chose to phase-out the standard *tone*() function and write our own that allows playing full melodies. A melody is a sequence of bytes, where each typically byte represents a single tone – its duration (full, half, quarter, eighth, sixteenth), and octave (1–5). It allows also rests, dotted notes, playing sounds of arbitrary melody, and in the latest version also a repetition sign. The tricky part for this implementation was that the timer pre-scaler has to change depending of the note frequency since humans have quite a large audible frequency range, see again the source code for details.

6.4 Choreographies

The most important feature of the framework is the ability to design choreographies with no programming skills. A choreography is a plain text file consisting of a sequence of movements, one per line. Each movement is a triple: the time in milliseconds (from the start of the movement, or the latest time reset command), the degree of freedom to move, and the target position to reach. Each degree of freedom can move using a different speed, which can be changed anywhere within the choreography. Everything on a line behind a # character is treated as a comment. Instead of a

movement command, the line can contain a control command. These include: starting a specified melody, playing a sound effect, "goto" – continuing the choreography from a specified line, resetting the time clock, which is important because the time is specified only using a 16-bit integer, which would prevent specifying longer choreographies, setting a total time of the choreography – even when looped, the dance will be stopped after that time, and in the most recent version, also the possibility to define procedures (i.e. sequences of movements) that can be inserted at any other location by a single command. This involves recalculating the times for the movement commands, meaning the time specification inside of a procedure is relative to the time of the procedure start. In the framework that we have presented to the participants, there was a limitation of one movement at a time (with the exception of full-speed movements – they could be triggered in parallel). In the most recent version, we have implemented the required data structures and procedures that allow simultaneous movements. This, however, changed the semantics – times in the original version meant the time to wait since the last movement command, whereas times in the latest version are times relative to the choreography start, or the most recent clock reset command. Every choreography is terminated by a triple "0 0 0".

Children in the camp used the last day to get their robot working, and they created a couple of little dances as well as the sequence that produced a bipedal walking. To upload the choreography to the robot volatile SRAM memory, it can be simply copy-pasted to the terminal window, preceded by the '@' character. The framework allows to store up to three choreographies to the non-volatile EEPROM memory and recalling them after the robot is powered up again. It is also possible to set the autostart flag, which means the robot will start a selected choreography automatically when turned on.

6.5 Starting Programs Using Ultrasonic Sensor

We wanted to implement a convenient interaction with the robot using the ultrasonic sensor and timing. Placing a hand, or some more hard-surface object (such as mobile phone) triggers interaction. After that, the robot observes carefully the movements in front of its sensor. By repeating the obstacle/no-obstacle sequence of gestures, the robot counts the "program" to be started. When the sequence stops, it will know how much it has counted, and starts the respective program. To make the interaction easier, the whole scenario is accompanied by sounds.

The robot is equipped with a single sensor – a forward pointing low-cost ultrasonic sensor, which occasionally suffers from freezing when it receives no echo. It can be reset by grounding its output ECHO signal for a short time, but we have to take caution to detect this behavior with proper timing. Also, the measured distance in this case would be replaced by a magic constant and it has to be distinguished from the real distance readings. The robot may be operated in a large room, or in a small space, and thus to program the above-described procedure took some noticeable efforts, see again the sources for details.

6.6 Controlling the Robot with Android Devices

The interaction with the robot can take place both at the USB serial communication line and on the Bluetooth serial line. All information is printed to both and inputs from both are being responded to. Most commands are triggered immediately by pressing a single key. This communication interface allows for a direct control of any degree of freedom, changing the speed of the movement, testing all the sound effects, melodies, saving and loading the choreographies and calibration parameters, entering the calibration procedure, but also triggering special choreographies. We have tuned walking behavior in four directions: forward and backward walking, and turning at a spot left/right, and these could be triggered by sending a single character through the communication line. Thus it is possible to easily control the robot from Android after being connected through Bluetooth, using a free BT control application, such as Arduino Bluetooth Controller [5].

7 Example Choreographies

We have implemented two example choreographies, which attracted some attention to this robot and our summer activity at various national fora: a cancan dance based on the melody of Offenbach, and walking accompanied by Jarre's Popcorn melody. Both can be found at the camp's website [3]. Here is one version of infinite walking choreography with music in the background

```
@1 11 1
#lean to the left
100 1 48
1 2 69
#move right hand
1 6 180
100 4 104
1 3 104
#change hands
1 6 90
1 5 0
100 1 111
1 2 146
1 6 0
1 5 90
# second step
100 3 62
1 4 69
# -
100 1 48
1 2 69
#end of steps
1 9 2
0 0 0
```

:

8 Framework Scalability

This framework for the robot Otto that we have developed is a generic framework that can in fact be used with any other toy controlled by Arduino. We have done just that after we have built a larger humanoid robot Lilli with 25 degrees of freedom [7]. Servos are controlled by two external boards connected through I2C bus, but this required a very little work. Furthermore, we have added a very low-cost mp3 player that interacts with Arduino and added commands that start or stop playing a specified sound sample on a connected speaker.

8.1 Multiple Arduino Units

Sometimes a single Arduino Nano is not sufficient for the project needs. One possibility is to leave the platform and move to more advanced solutions such as the family of the STM32 boards, however, nothing can really beat the high availability of solutions and libraries that are available to Arduino. One option is to connect several Arduino computers together. We have done just that in one of our outdoor robots, where we have developed a proprietary fast communication protocol using standard GPIO lines (since the serial lines, I2C and SPI are often used by other connected devices) [6]. We are currently working on extending Otto's framework with the ability to distribute the control over multiple Arduinos by forwarding the communication traffic to all the Arduinos in the chain.

9 Makers Movement in Educational Process

Our efforts are very deeply founded in the makers movement. A prototype of Lilli, the more advanced of the robots controlled by our framework has first been presented at Maker Faire in Vienna in May 2018 by Per Salkowitsch. We have encountered Otto robot for the first in June 2018 at Robotic Day in Prague. Makers movement provides unlimited sources of inspiration. It is in conformance with our very strong belief in the joy of sharing. Fablab's also are very central to the Makers movement – for instance Maker Faire in Vienna is organized by Viennese Fablab. However, what is the role of all this technology – 3D modelling, printing laser, vinyl cutter, automated sewing machine, water ray production, 3D scanning, and other in the educational process?

With the Industry 4.0, there is a clear shift into very high versatility in production, very large degree of automation and customization. Prototyping is a skill that becomes essential for all parties involved in the new organization of the development and production cycle. Some schools (with the help of EU funding) have invested large resources into establishing technical workshops, where all pupils spend several hours per week learning about this revolutionary technology. We believe that significant efforts are needed to make it easier for the educators to select in the sea of available information suitable activities, projects, and technologies to make this transition succeed. Our little contribution has been realized with this idea in mind. We also believe in the change of the organization of school education. Learning 1 lesson of history per week might contribute to a reasonable and regular work habit, but it is not suitable for efficient learning. Instead, spending more hours a week until a particular educational unit is completed gives stronger and longer lasting experience and allows for some courses to take place outside of the schools, in regional centers similar to Fablabs.

10 Future Directions and Conclusions

We have received only a positive feedback from both the participants and the parents. Several participants have attended "Otto service days" in the weeks after the camp. Some of them are now regularly attending a club for children in Fablab

Bratislava, others joined robotics clubs around the city. We are planning a second $(DT)^2$ in the forthcoming summer, considering our own reconfigurable robotics as the target platform to work on. Since the last summer, "our" version of Otto has been built by several groups around the country and we often use it at various public presentations. The framework has been used multiple times in seminars with the students of Applied Informatics at Comenius University. Links to the Github repositories with the software and all the details about the camp and our version of Otto robot can be found at the $(DT)^2$ website [3].

References

1. Sweeney, L.B., Meadows, D. (eds.): The Systems Thinking Playbook: Exercises to Stretch and Build Learning and Systems Thinking Capabilities. Chelsea Green Publishing, White River Junction (2010)
2. Otto DIY (2019). www.ottodiy.com
3. $(DT)^2$ website (2018). dtdt.fablab.sk
4. $(DT)^2$ Otto Github repo. (2019). github.com/Robotics-DAI-FMFI-UK/dtdt-otto/
5. Ioannis Tzanellis: Arduino Bluetooth Controller. play.google.com/store/apps/details?id=eu.jahnestacado.arduinorc
6. Smelý Zajko Github rep. github.com/Robotics-DAI-FMFI-UK/smely-zajko-ros
7. Cu-lilli robot home. kempelen.dai.fmph.uniba.sk/lilli/
8. Markowitz, D.G.: J. Sci. Educ. Technol. **13**, 395 (2004)
9. Klein, M., et al.: Hedgehog: a versatile controller for educational robotics. In: Constructionism 2018 Conference Proceedings (2018)