# BOBShield: An Open-Source Miniature "Ball and Beam" Device for Control Engineering Education

Gergely Takács*, Erik Mikuláš, Anna Vargová, Tibor Konkoly, Patrik Šíma,
Lukáš Vadovič, Matúš Bíro, Marko Michal, Matej Šimovec and Martin Gulan
*Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering*
*Slovak University of Technology in Bratislava,* Bratislava, Slovakia
*gergely.takacs@stuba.sk

2021 IEEE Global Engineering Education Conference (EDUCON) | 978-1-7281-8478-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/EDUCON46332.2021.9454013

***Abstract*—This article presents a reference design for the well-known ball-on-beam laboratory experiment, where a spherical ball without direct actuation is only balanced by the inclination of a supporting structure, such as beam, rail or tube. The design introduced here is completely open-source and utilizes only a handful of off-the-shelf components and 3D printing; resulting in an exceptionally low hardware cost. Moreover, the resulting apparatus fits on a standard expansion module format, known as a Shield, which is compatible with a range of microcontroller prototyping boards from the Arduino ecosystem. This affordable, small, reproducible and open design is thus intended to aid control systems or mechatronics education via hands-on student experiments or even conducting research on a budget. In addition to the hardware design with downloadable project files, we also present an application programming interface and the results of a demonstration example here.**

***Index Terms*—open educational resources, control engineering education, microcontrollers, student experiments, mechatronics, educational technology**

## I. Introduction

The ball and beam experimental device is a well-known staple of most laboratories teaching control engineering or mechatronics around the world. The apparatus consists of a surface with a dominant dimension, such as a beam, guiding rails or a tube that is actively inclined by an actuator. The goal of the feedback control system is to adjust the inclination such that a ball tracks a pre-set reference trajectory. This under-actuated, fast, nonlinear system is an excellent tool to test and teach concepts of control theory, modeling, identification, microcontroller technology and signal processing.

There is a range of vendors offering the ball and beam device in a commercial package: some examples include Quanser BB01 Ball and Beam, AMIRA ball and beam, Ball and Beam Control System Trainer Kit by ACROME, TecQuipment CE106 Ball and Beam Apparatus and others. Their commercial and standardized nature means that, in addition to education, they are suitable for research as well (c.f. [1], [2] and many others). Commercial ball and beam systems are high-quality precision products with excellent documentation, however, they usually take up extended bench space and are quite expensive and delicate.

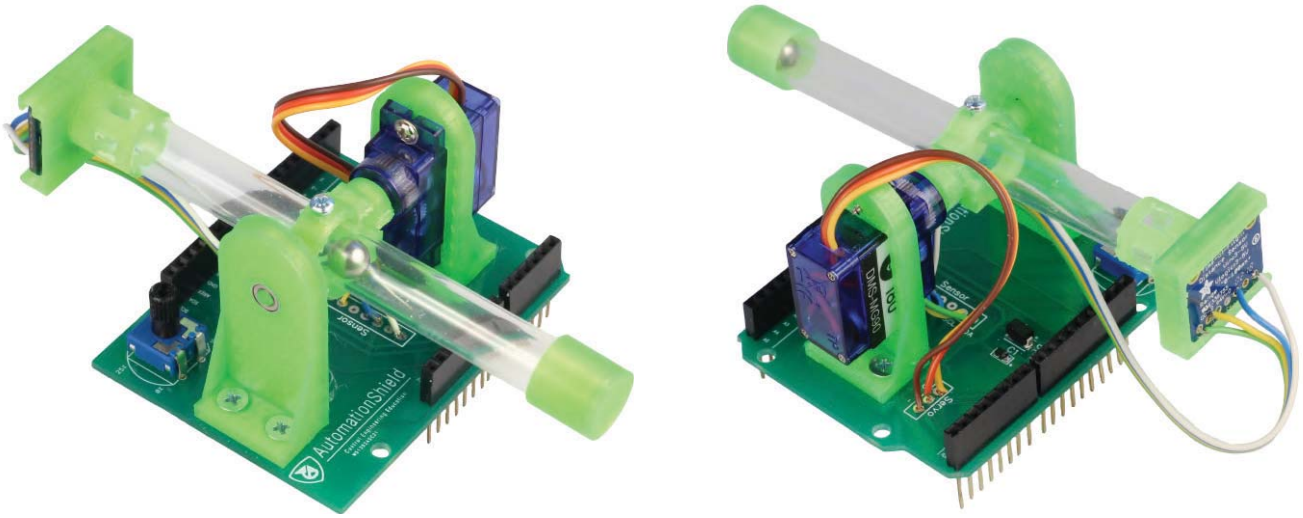
There is a plethora of research articles using ball and beam (also called ball on beam, ball-on-a-beam) systems for control algorithm development and verification, including neuro-fuzzy [1], model predictive [3], sliding mode [4], fractional order control [5] and others.

Numerous researchers choose an alternate route to equipping laboratories by designing and implementing their own version of the experiment (e.g. [6], [7]). In this case, the hardware is usually poorly documented, utilizes custom-made and improvised mechanical components and is specific to a course or research group. Hence, mainly due to their size and cost, neither of the aforementioned device categories is suitable for take-home student experiments or projects. We cannot fail to mention the importance of experimental laboratory devices that may be taken home in the ongoing COVID-19 pandemic caused by the SARS-CoV-2 virus [8] or aid online education and remote teaching at more ordinary times.

There are several examples of ball and beam devices in the literature (c.f. [9], [10]), some even use the popular and widely accepted Arduino microcontroller unit (MCU) prototyping platform instead of more expensive proprietary real-time computers or laboratory measurement cards. For example, Ahmad and Hussain introduce a relatively small and cheap ball on beam system based on the Arduino Uno, albeit using a rather improvised design [11]. Osinski et al. hint a more robust system using laser-cut mechanical parts in [7], however, do not offer a replicable documentation. An Arduino is employed by Kumar and Pandian as a measurement card without real-time control, utilizing yet again an improvised mechanical design [12]. Gao et al. have used Simulink to transcribe a PID block scheme to machine code for an Arduino Uno and a makeshift wooden ball and beam hardware [13]. There is even a ball on beam device design based on a cardboard construction available online [14]. A notable improvement in documentation quality and parts availability is presented in [15], where the Arduino-based ball and beam system is assembled from commercially available MakeBlock components.

In earlier work, we have proposed a concept of miniaturized, low cost, open-source laboratory devices that utilize the popular Arduino microcontroller prototyping platform for real-

The authors gratefully acknowledge the contribution of the Slovak Research and Development Agency (APVV) under the contracts APVV-18-0023, APVV-17-0214 and APVV-14-0399. The authors appreciate the financial support provided by the Cultural and Educational Grant Agency (KEGA) of the Ministry of Education of Slovak Republic under the contracts 005STU-4/2018 and 012STU-4/2021.




(a) Front view. (b) Back view.

Fig. 1. Isometric view of the assembled BOBShield device.

time feedback control. The electronic and mechanical layout known as the Arduino R3 pinout allows the design of hardware expansion modules, known in this ecosystem as "Shields". Formerly we have described a magnetic levitation apparatus [16], air flotation device [17], thermal experiment [18] and an optical system [19]. The key concept of our approach was low price, small footprint, universal and easy reproducibility.

Here we will introduce an open-source reference design of a low-cost miniaturized ball-on-beam (BOB) experimental device (Fig. 1). We will refer to this as the "BOBShield" for the remainder of the article.

## II. Hardware

The following passages propose the reference design for the BOBShield device. The main aim of this section is to give as much information as possible, so that the device can be robustly replicated or even improved upon by others. Both mechanical and electronic components are marked alphabetically in the text and the reader may follow along the commentary by inspecting Figs. 2–5 and Table I, containing the same type of markings.

### A. Mechanical

First, let us introduce the mechanical construction of the BOBShield. The entire device is built on a printed circuit board (PCB) copying the outline of the Arduino Uno (a). The two-layer FR4 board with 1.6 mm thickness and standard manufacturing tolerances fits into the customary 100 mm × 100 mm size limit, thus allows one to utilize the several small batch PCB manufacturers offering extremely competitive pricing. The BOBShield may be connected and mechanically fixed to any R3 pin layout microcontroller prototyping board by single row stacking header pins ($b_1$-$b_3$). The pair of 8- ($b_1$), 6- ($b_2$) and 10-pin ($b_3$) headers provide easy access to the

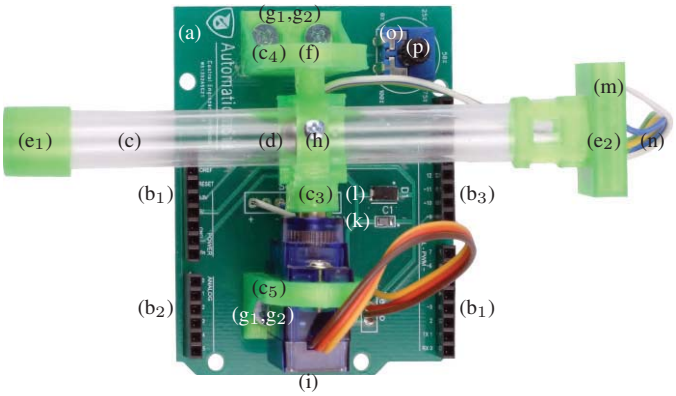



Fig. 2. Top view of the assembled device showing component marking.

electrical connections for external monitoring or data logging by e.g. an oscilloscope or a laboratory measurement card.

The 100 mm long transparent plexiglas tube (c) has an inner diameter of 10 mm and holds the 8 mm steel ball (d) that moves freely along its longitudinal axis, if inclined. One end of the tube is closed by a 3D printed cap ($e_1$), while the other end is terminated by another 3D printed part ($e_2$) designed to hold the distance measurement unit. The tube is fixed at its middle by a 3D printed circular clamp ($e_3$) that permits the inclination of the tube along its major axis. The clamp is held in place at both sides by L-shaped 3D printed brackets, one ($e_4$) containing a miniature ball bearing (f) to facilitate smooth movement and the other ($e_5$) fixed to the clamp through the shaft of the actuator. Figure 3 shows all of the 3D printed components ($e_1$–$e_5$), for which we provide downloadable CAD files [20]. We printed these components on an original Prusa i3 MK3/S 3D 3D printer using a total of 18 g filament under 3.5 h time. The L-shaped brackets are mounted to the PCB by M3 machine bolts ($g_1$) and fixed at the bottom by corresponding

nuts ($g_2$). The clamp is tightened to hold the tube steadily by a single self-cutting screw (h). The actuator is fixed to its mounting bracket by a pair of self-cutting screws that are bundled with the device, along with the usual horns which are not utilized in our design.

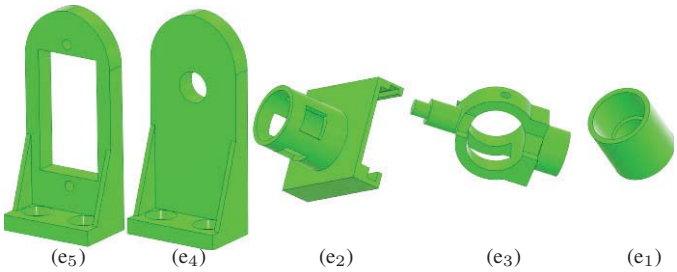



Fig. 3. 3D printed components.

### B. Electronic

The electronic design of the BOBShield is kept as simple as possible, in order to reduce the complexity and price. The resulting electronic schematic drawing is shown in Fig. 4.

The inclination of the tube and, ultimately, the position of the ball, is manipulated by a standard micro RC Servo motor (i) with analog feedback. The total rotation range of the motor utilized in this application is only $\pm 30°$, thus any identically sized servo actuator will be suitable. The motor is connected to the `D9` pin of the standard Arduino R3 layout (j), since this equivalent MCU pin handles interrupts and timing on the Uno as well as other prototyping devices. The current consumption of the servo motor is low enough to be directly powered from the `5V` supply of the Arduino board without the need of an external wall-plug adapter. We added a capacitor (k) parallel to the motor to smooth out possible transients affecting the board supply and a diode (l) to prevent possible back-electromagnetic force damaging the circuitry.

The absolute distance of the ball from the reference located at one end of the tube is measured by the ST Microelectronics VL6180X time-of-flight (TOF) sensor (m). We found that this state-of-the art component compensates well for lightning conditions and surface types and we have not identified any significant problems with the reflective nature of the steel ball or the tube itself. Unfortunately, it comes in a physical package

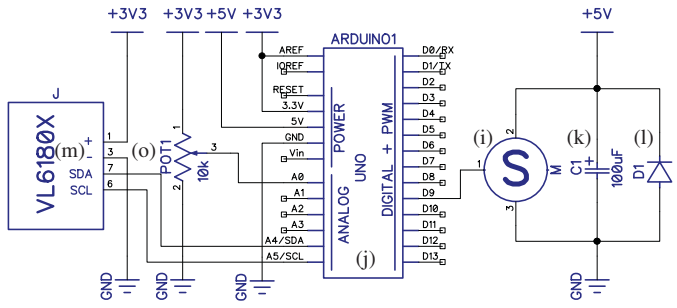



Fig. 4. Electronic schematics of the BoBShield device.

that is designed for industrial reflow soldering, therefore, its custom mounting is outside the reach of hobbyists and students. Hence, we have decided to utilize a widely available breakout board implementing the VL6180X for hobbyist use. This breakout may be purchased as an end-product and implements the sensor on a small-outline PCB with standard 0.1" pitch. The sensor communicates through the I2C protocol and we have utilized the standard `SCA` and `SCL` pins of the layout. The breakout connects to the base board via a flexible 4-wire ribbon cable (n). A flat flexible cable (FFC) would have been a better choice, unfortunately the sensor module would need to be custom-made for this.

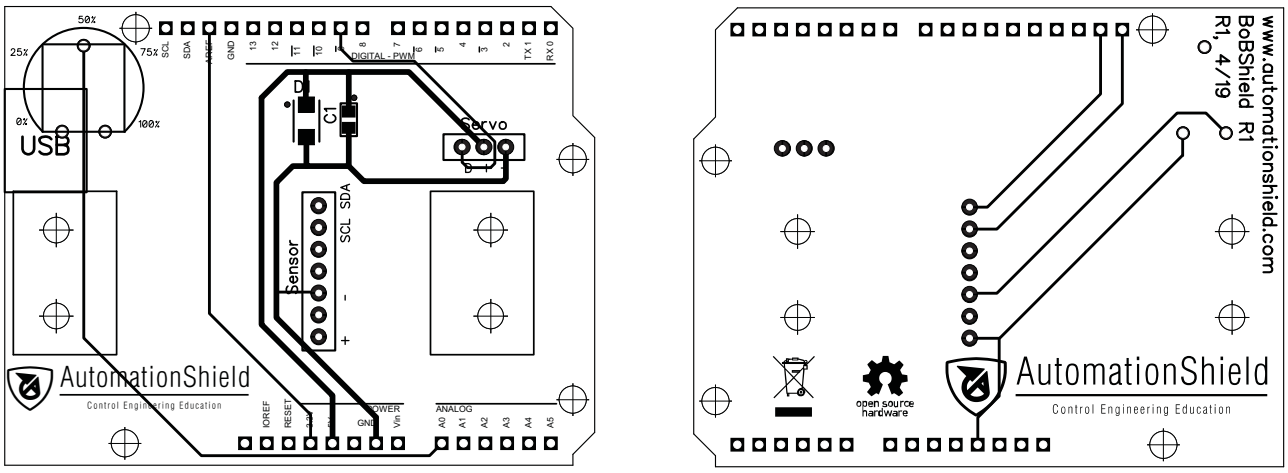
Finally, there is a potentiometer (o) with a small plastic shaft (p) that may be used for any user programmable role; but the main idea is to provide means to manually input the desired reference signal. The potentiometer is fed by the 3.3 V source for its maximal setting, so that the wiper connected to the `A0` analog port does not damage other microcontroller boards—such as the Due or Zero—operating with CMOS logic levels. To maximize analog resolution, we have also connected the `3.3V` source to the `AREF` pin and compensate for this change in code. The sensor board is also powered by the 3.3 V source.

The resulting two-layer printed circuit board is shown in Fig. 5. We have avoided including further header pins and connectors and soldered the wires connecting to the servo and sensor directly to the PCB. The electronic schematics and the PCB were designed in the free version of DIP Trace, making further changes accessible to anyone. We provide the editable schematic files and the editable PCB layout, along with the manufacturing-ready Gerber format PCB [20].

### C. Cost and part availability

A comprehensive parts list with a detailed description of our component choices and pricing is listed in Tab. I. The pricing excludes postage, consumables (solder, flux, etc.) and labor cost. Unit prices have been determined by searching for the cheapest possible alternatives using meta-search engines, through a range of various vendors and for relatively small quantity purchases that have been for <100 pieces in all cases. We have also included auction sites, since these may offer lower-priced small quantity purchases than established electronic component vendors. We believe that this calculation can be realistic even for a larger-scale production, since the unaccounted for items are balanced by deals on parts. The total price will approximately double, if parts are sourced from a single supplier in low quantities and time is an issue. We have decided not to include the price of the Arduino board in our figure, as these devices are useful in other roles as well and the BOBShield is compatible with several alternatives.

The roughly $10 U.S. final material cost makes our proposed design affordable to any institution teaching mechatronics or control engineering concepts, even in low-income countries. The device is well within the reach of individual students too; making it easy to distribute for projects, assignments or for remote teaching.




(a) Front side.

(b) Back side.

Fig. 5. The printed circuit board of the BoBShield device.

We have taken care to only include parts that are easily available and can be globally sourced. Due to its open-source nature, the sensor board is offered by various vendors. Practically any size-compatible servo motor will work, and the rest of the components can be effortlessly exchanged to equivalent alternatives.

## III. Application Programming Interface

This section introduces the application programming interface for the BOBShield device, written in C/C++ for the Arduino IDE. The goal of the API is to abstract basic hardware functionality to input-output functions, thus encouraging the users to quickly create their own control applications. It goes without saying, that letting students to write their own interface holds educational value as well.

The API is contained within the `BOBShield.h` header file of the open-source AutomationShield library [21]. Since every implementation file (`*.cpp`) is compiled in a given Arduino library regardless of actual usage, the servo motor and sampling subsystems would conflict in re-assigning interrupt service vectors. Thus we have sacrificed good programming practice for the sake of didactic value, since the AutomationShield library must handle the API and application examples for several other devices.

The hardware functionality is handled by the `BOBClass` class which creates a `BOBShield` object. As per Arduino custom, the initialization functionality is contained in the `begin()` method and calling

```
BOBShield.begin();
```

will start the servo motor functionality by assigning its hardware pin for later use. The sensor readings can be calibrated by using the

```
BOBShield.calibration();
```

method, which tilts the beam towards $\pm 30^{\circ}$ and selects the maximal, respectively, the minimal distance readings. These calibrated readings are accessible to other methods.

All I/O functionality is handled by three methods, that have been named `read()` and `write()` according to Arduino nomenclature. Using

```
y = BOBShield.sensorRead();
```

will output the absolute position of the ball $y$ (mm) as a floating-point number by polling the ToF sensor. Similarly, the percentual position based on calibration may be accessed by the `sensorReadPerc()` method. The manual reference from the potentiometer is read by the

```
r = BOBShield.referenceRead();
```

method, returning the reference $r$ as a floating-point number. Finally, the desired angle $u$ (°) is sent to the servo motor by

```
BOBShield.actuatorWrite(u);
```

where the input parameter is a floating point number.

The rest of the AutomationShield library for the Arduino IDE provides numerous universal functions that are common to feedback control, such as, a comprehensive interrupt-based sampling framework, a PID routine, signal processing, Kalman filtering, etc. The library, including the BOBShield API, is compatible with microcontroller architectures other than the Atmel AVR assumed for the widely used Arduino Uno/Mega2560. For example, the code shall be compatible with various other devices with ARM Cortex M architecture MCU, e.g. Arduino Due (Atmel SAM3X8E), Arduino Zero (Atmel SAMD21), Adafruit Metro M4 Express (Microchip ATSAMD51), etc.

## IV. Examples

In addition to the API, the AutomationShield library contains demonstration examples for each device. Currently, there are four examples offered for the BOBShield Arduino API.

TABLE I
COMPONENT LIST AND COST CALCULATION FOR MATERIAL PRICE IN U.S. DOLLARS.

| Name | Part no., value. | PCB | Mark | Pcs. | Unit | Total |
|---|---|---|---|---|---|---|
| PCB | 2-layer, FR4, 1.6 mm thick (e.g. JCLPCB) | - | (a) | 1 | 0.40 | 0.40 |
| 3D print | 18 g, $\phi = 1.75$ mm PETG filament, bright green, at 240 °C (90 °C bed), 3 h 20 min | - | ($e_{1-5}$) | 1 | 0.64 | 0.64 |
| Trimmer | 10 kΩ, 250 mW, single turn THT trimmer (e.g. ACP CA14NV12,5-10KA2020) | POT1 | (o) | 1 | 0.19 | 0.19 |
| Screw | DIN 7971C 2.2 × 6.5 | - | (h) | 1 | 0.02 | 0.02 |
| Diode | generic diode, DO214AC (e.g. Vishay Semiconductor BYG20J) | D1 | (l) | 1 | 0.38 | 0,38 |
| Screws | M3×8 DIN963A | - | ($g_1$) | 4 | 0.02 | 0.08 |
| Nuts | M3 DIN439B | - | ($g_2$) | 4 | 0.01 | 0.04 |
| Header | 10×1, female, stackable, 0.1“ pitch (e.g. SparkFun 474-PRT-10007) | - | ($b_3$) | 1 | 0.09 | 0.09 |
| Pot shaft | For “POT1”, (e.g. ACP CA9MA9005) | - | (p) | 1 | 0.07 | 0.07 |
| Header | 8×1, female, stackable, 0.1“ pitch (e.g. SparkFun 474-PRT-10007) | - | ($b_1$) | 2 | 0.06 | 0.12 |
| Header | 6×1, female, stackable, 0.1“ pitch (e.g. SparkFun 474-PRT-10007) | - | ($b_2$) | 1 | 0.06 | 0.06 |
| Tube | transparent, plexiglass (PMMA), $L$=100 mm, 12/10 mm | - | (c) | 1 | 0.38 | 0.38 |
| Sensor | VL53L1X breakout, time of flight ranging SNSR | J | (m) | 1 | 4.20 | 4.20 |
| Motor | Metal geared 9 g, 5 V micro servo | M | (i) | 1 | 1.92 | 1.92 |
| Capacitor | 100 uF, 6.3V, 20, 0805, tantalum | C1 | (k) | 1 | 0.53 | 0.53 |
| Ball | Steel ball, diameter 8 mm | - | (d) | 1 | 0.18 | 0.18 |
| Bearing | Miniature ball bearing, 3×6×2 mm, (e.g. MR-63-ZZ) | - | (f) | 1 | 0.26 | 0.26 |
| Cable | Ribbon cable, 4-7 wires, cca. 0.2 m (e.g. EL-2468) | - | (n) | 1 | 0.08 | 0.08 |
| | | | | | **Total:** | **$9.46** |

The file `BOBShield_SelfTest.ino` implements a basic self-test routine, aiding the verification of the hardware functionality. First, the ToF sensor is initialized and, if the MCU cannot connect to the sensor chip, will report the failure. Then, the servo turns to each of its extreme operation points and the routine compares the distance readings with expected values.

The project file `BOBShield_Identification.ino` performs an open-loop test by supplying a range of actuator settings, while `BOBShield_Identification_aprbs.ino` does the same but applies an amplitude-modulated pseudo-random signal (Fig. 6). Both examples assume a strict interrupt-based sampling framework and print the input $u$ and corresponding output $y$ to the serial line. This feed can be recorded with an arbitrary serial terminal software and later used as input-output data for system identification and mathematical modelling.

Finally, `BOBShield_PID.ino` demonstrates the proportional-integral-derivative (PID) position control of the steel ball. Besides the engaging visual action of the hardware, results are also printed to the serial line. Figure 7 demonstrates a step change of the reference, where the process variables $r$ (blue), $y$ (red) and $u$ (green) may be followed in the oscilloscope-like rolling window of the Arduino IDE Serial Plotter. Similar to the open-loop case, any other terminal program is suitable for monitoring and data logging.

The PID demonstration example is sampled with a $T = 10$ ms period, and a pre-set reference vector of $[30, 50, 8, 65, 15, 40]^{\mathrm{T}}$ mm is requested for 1000 samples (10 s) providing sufficient time for transients. The PID controller is manually tuned to the $K_{\mathrm{P}} = 0.3$ (mm/°), $T_{\mathrm{I}} = 600$ and $T_{\mathrm{D}} = 0.22$ constants. An integral windup clipping strategy is set to ±10 (°), while the input to the servo motor is saturated at $u = 30$ (°). The results of this experiment are illustrated in Fig. 8, where reference position $r$ is compared to achieved

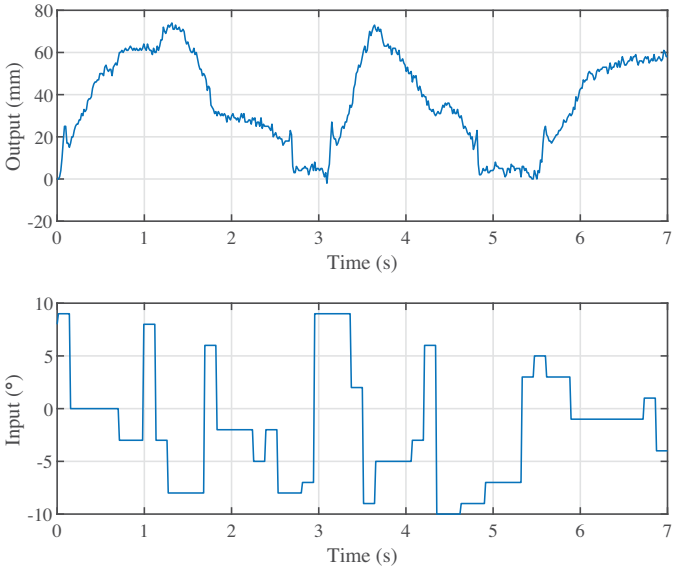



Fig. 6. Open-loop response for system identification.

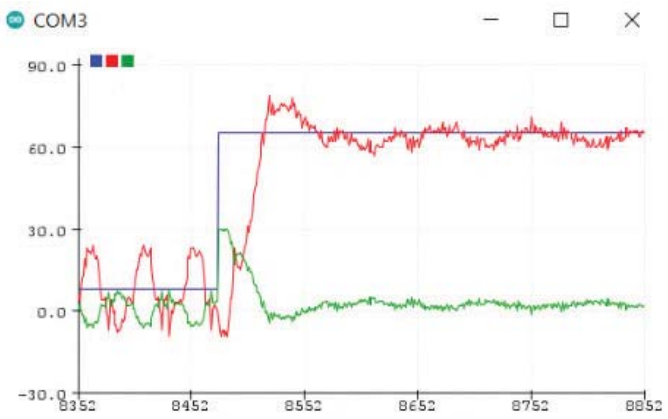



Fig. 7. Screenshot of a PID experiment in the Arduino IDE Serial Plotter.

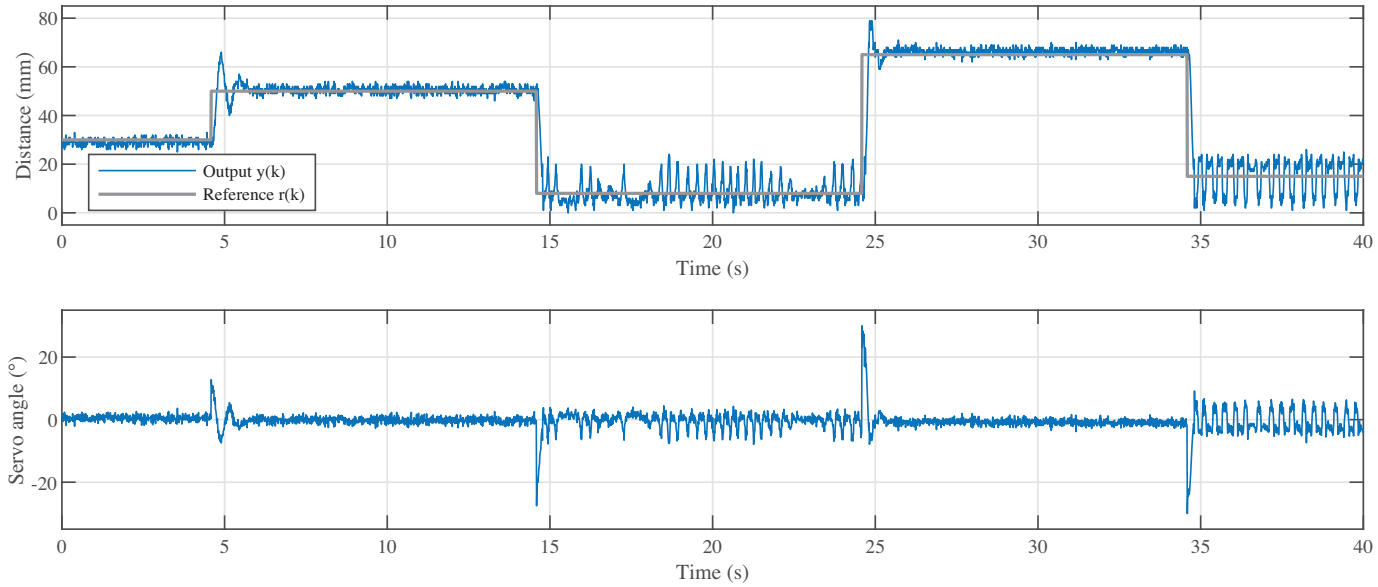



Fig. 8. PID control of the ball position.

output $y$ on the top and the corresponding servo input $u$ is presented at the bottom.

The ball follows the reference position trajectory faithfully across a wide span of the working range. The tracking is worse when the ball is close to the sensor, we believe this is caused by sensor noise originating from unwanted reflections or other physical effects. Since the main point of this article is to present our hardware design, and due to the limited scope we will not introduce other control methods here. However, the reader shall note that it is likely that the response shown in Fig. 8 may be improved by more advanced control and/or appropriate signal processing.

## V. Conclusion

This paper presented a low-cost, open-source "ball-on-beam" didactic device that can be physically attached to a range of Arduino-compatible microcontroller prototyping boards and is suitable for control and mechatronics education. In addition to the hardware, we introduced the input-output interface and demonstrative examples.

Implementing this reference design results in an apparatus with a material cost below $10 and a small physical outline, making it suitable for classroom distribution and take-home student experiments. The open hardware, interface and examples foreshadow the possibility of collaboration between various educators and instructors in bettering the API, examples and even creating open teaching materials.

Future work shall improve the hardware design mainly by making the 3D printed parts more robust and resilient. We are also planning to expand the current API to MATLAB, Simulink, Python and possibly a range of other scientific and engineering software, such as LabVIEW, Octave and Scilab/X-cos. The range of demonstration examples shall be broadened by the addition of other control engineering methods, such as, linear quadratic (LQ) or model predictive control (MPC).

## Acknowledgements

The authors would like to thank the involuntary assistance of Marek Krippel and Rastislav Haška with the original version of the documentation. We are still not really sure what Samuel Mladý was working on, but thanks for sticking around.

## References

[1] N. Eqra, R. Vatankhah, and M. Eghtesad, "A novel adaptive multi-critic based separated-states neuro-fuzzy controller: Architecture and application to chaos control," *ISA Transactions*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019057820304997

[2] I. M. Mehedi, U. M. Al-Saggaf, R. Mansouri, and M. Bettayeb, "Two degrees of freedom fractional controller design: Application to the ball and beam system," *Measurement*, vol. 135, pp. 13–22, 03 2019.

[3] N. Hara, M. Takahashi, and K. Konishi, "Experimental evaluation of model predictive control of ball and beam systems," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 1130–1132.

[4] S. Sameera, R.; Arun, "Position tracking of ball and beam system using fractional order controller," *International Journal of Advanced Research Trends in Engineering and Technology*, vol. 4, no. 6, pp. 174–180, 2017.

[5] T. Emami, "Mixed sensitivity design of pid controller-applied to a ball and beam system," in *ASME 2017 International Mechanical Engineering Congress and Exposition*, vol. 4. ASME, 2017.

[6] M. Saad and M. Khalfallah, "Design and implementation of an embedded ball-beam controller using PID algorithm," *Universal Journal of Control and Automation*, vol. 4, pp. 63–70, 2017.

[7] C. Osinski, A. L. R. Silveira, C. Stiegelmaier, M. G. Bergamini, and G. V. Leandro, "Control of ball and beam system using fuzzy PID controller," in *2018 13th IEEE International Conference on Industry Applications (INDUSCON)*, 2018, pp. 875–880.

[8] K. Bangert, J. Bates, S. Beck, Z. Bishop, M. D. Benedetti, J. Fullwood, A. Funnell, A. Garrard, S. Hayes, T. Howard, C. Johnson, M. Jones, P. Lazari, J. Mukherjee, C. Omar, B. Taylor, R. Thorley, G. Williams, and R. Woolley, "Remote practicals in the time of coronavirus, a multi-disciplinary approach," *International Journal of Mechanical Engineering Education*, vol. 0, no. 0, p. 0306419020958100, 0.

[9] M. M. Kopichev, A. A. Kuznetsov, A. R. Muzalevskiy, and T. L. Rusyaeva, "Ball and beam stabilization laboratory test bench with intellectual control," in *2020 XXIII International Conference on Soft Computing and Measurements (SCM)*, 2020, pp. 112–116.

[10] G. Dewantoro, D. Susilo, and D. C. Amanda, "Development of microcontroller-based ball and beam trainer kit," *Indonesian Journal of Electrical Engineering and Informatics*, pp. 45–54, 03 2015.
[11] B. Ahmad and I. Hussain, "Design and hardware implementation of ball beam setup," in *2017 Fifth International Conference on Aerospace Science Engineering (ICASE)*, 2017, pp. 1–6.
[12] S. T. Kumar and B. J. Pandian, "Tuning and implementation of fuzzy logic controller using simulated annealing in a nonlinear real-time system," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, 2014, pp. 916–921.
[13] Z. Gao, S. Wijesinghe, T. Pathinathanpillai, E. Dyer, and I. Singh, "Design and implementation a ball balancing system for control theory course," 2015.
[14] K. Benchikha, "Ball and Beam: PID control on Arduino with LabView to stabilize a ball on a beam," Online, 2019, [cited 11.27.2020]; Available from https://create.arduino.cc/projecthub/karem_benchikha/ball-and-beam-601d7a. Arduino Project Hub.
[15] Wolfram Research, "Microcontroller Kit Sample Projects: Balancing a ball on a beam," Online, 2020, [cited 11.27.2020]; Available from https://reference.wolfram.com/language/MicrocontrollerKit/workflow/BallAndBeamControl.
[16] G. Takács, J. Mihalík, E. Mikuláš, and M. Gulan, "MagnetoShield: Prototype of a low-cost magnetic levitation device for control education," in *Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON)*, Porto, Portugal, April 2020, pp. 1516–1525.
[17] G. Takács, P. Chmurčiak, M. Gulan, E. Mikuláš, J. Kulhánek, G. Penzinger, M. Podbielančík, M. Lučan, P. Šálka, and D. Šroba, "FloatShield: An open source air levitation device for control engineering educationn," in *Proceedings of the 2020 IFAC World Congress*, Berlin, Germany, July 2020, pp. 1–8, (Preprints).
[18] G. Takács, M. Gulan, J. Bavlna, R. Köplinger, M. Kováč, E. Mikuláš, S. Zarghoon, and R. Salíni, "HeatShield: a low-cost didactic device for control education simulating 3D printer heater blocks," in *Proceedings of the 2019 IEEE Global Engineering Education Conference*, Dubai, United Arab Emirates, April 2019, pp. 374–383.
[19] G. Takács, T. Konkoly, and M. Gulan, "Optoshield: A low-cost tool for control and mechatronics education," in *Proceedings of the 12th Asian Control Conference*, Kitakyushu-shi, Japan, Jun 2019, pp. 1001–1006.
[20] G. Takács *et al*., "BOBShield," Online, 2020, [cited 1.12.2020]; GitHub Wiki page documenting the BOBShield device. Available from https://github.com/gergelytakacs/AutomationShield/wiki/BoBShield.
[21] ——, "AutomationShield: Control Systems Engineering Education," Online, 2020, [cited 11.24.2020]; Available from http://www.automationshield.com. 2018–2020.