Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-27 (2019) 277–282

Upgrade of the Ball and Plate Laboratory Model

Adam Březina, Milan Tkáčik, Tomáš Tkáčik, Slávka Jadlovská *

* *Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, Košice 042 00, Slovakia (e-mail: adam.brezina@tuke.sk, milan.tkacik@tuke.sk, tomas.tkacik.3@student.tuke.sk, slavka.jadlovska@tuke.sk)*

Abstract: This paper deals with the overall hardware and software upgrade of the Ball&Plate laboratory model used at the Center of Modern Control Techniques and Industrial Informatics (CMCT&II) at Department of Cybernetics and Artificial Intelligence (DCAI), FEEI, TUKE. Reasons for upgrading the model are described, along with the proposed solution. Design and construction of new control hardware for the model is presented. The paper next describes the implementation of program modules for controlling the position of the ball on the plate at level of the MicroController Unit (MCU) and the supervisor computer. Several approaches of image recognition with the purpose to obtain ball position are tested and compared. Finally, the paper presents the model-based design and verification of a PSD control algorithm for the Ball&Plate model. After the upgrade, the model can be used at the CMCT&II for both research and teaching purposes, with the emphasis on the implementation of applications with high demands on hardware performance, where higher sampling rates of the control algorithms are required.

Keywords: Cyber-physical systems, Electronic design, Embedded systems, Image processing, Unstable system, Computer vision

1. INTRODUCTION

These days, a lot of devices are controlled or monitored by computer-based systems. The mechanical and the software parts of devices are deeply intertwined, which creates cyber-physical systems (Lee (2015)). These devices are tightly connected within the same network, which enables creating scalable autonomous systems with shared data between devices.

The Ball&Plate laboratory model represents a cyber-physical nonlinear MIMO system with two inputs and two outputs, where the control objective is to stabilize the ball in the desired position on the plate or to track the reference trajectory with the ball via plate movement. It is a popular benchmark system used in research and education alike, suitable for use in areas such as analytical and experimental identification of nonlinear dynamic systems, as well as verification of classical control algorithms, i.e. PID/PSD controllers, or modern control methods e.g. state-feedback optimal or predictive control (Bay and Rasmussen (2016)). (Debono and Bugeja (2015)).

Several different Ball&Plate models have been used in the laboratories of the Center of Modern Control Techniques and Industrial Informatics (CMCT&II) at Department of Cybernetics and Artificial Intelligence (DCAI), FEEI, TUKE. The first Ball&Plate model used at CMCT&II was the Humusoft's CE151 which was typically used for the verification of PID/PSD control design (Jadlovská et al. (2009)). Later, a custom Ball&Plate model manufactured by Kybernetika was obtained. MPC control algorithm was deployed here (Oravec and Jadlovská (2014), Oravec and Jadlovská (2015)). The decision to upgrade this model was made because of its insufficient properties from the hardware and software point of view, which prevented testing control algorithms with higher sampling rates.

This paper deals with overall upgrade of the Ball&Plate model at the CMCT&II and is divided into three main chapters. First of these deals with the main reasons for upgrading the existing Ball&Plate model. The second one describes the design of the new electronic and mechanical components of the model. The last chapter describes the novel software modules for the Raspberry Pi and Arduino, including the experimental identification of model's parameters and the implementation of proposed control algorithm. Integration of the Ball&Plate model into the distributed control system infrastructure at the DCAI, FEEI, TUKE is also described.

2. REASONS FOR MODEL'S UPGRADE

The original Ball&Plate model used analog servomotors with a plastic gearbox, whose gears wore out after short time and this caused the servomotors to slip when a higher input was applied. At the same time, the analog servomotors were unable to adjust and maintain the desired plate's inclination, which caused spontaneous oscillations of the plate. The control board itself with a MicroController Unit (MCU) was already obsolete, mainly due to failing linear regulators powering individual servomotors. The need to use a desktop computer and a large external power supply also complicated the transfer of the system in case it

10.1016/j.ifacol.2019.12.673

was necessary to demonstrate its functionality outside the CMCT&II laboratory (Oravec and Jadlovská (2017)).

As a result of the above-mentioned reasons, the decision to significantly upgrade the entire Ball&Plate model was made. The upgrade included more precise and powerful servomotors with a steel gearbox in order to eliminate the spontaneous vibrations of the plate. Apart from that, all electronics including a control board with a MCU, the supervisor computer and a new power supply mainframe was inserted into a single box attached to the model itself. These hardware changes ensure the more precise positioning of the plate, removal of unwanted vibration phenomena, and allow for reaching higher sampling rates of control algorithms.

The hardware upgrade of the Ball&Plate model also requires a new software implementation of the individual program modules. Instead of a desktop computer with the MS Windows operating system, a single-board Raspberry Pi computer with the OS Linux Raspbian was used. Moreover, the implementation of the image recognition algorithm has been moved from the MATLAB/Simulink simulation environment to a native application implemented in the C++ programming language, resulting in a significant impact on the performance of the implemented algorithm.

The Ball&Plate model is built around the distributed control systems architecture. The lowest *Level of Models, Sensors and Actuators* contains the Ball&Plate sensors and actuators: the camera for image acquisition, gyroscope for determining plate's inclination and servomotors for moving the plate. The *Technological level of control and regulation* comprises the Arduino for controlling the servomotors and the gyroscope, and partly the Raspberry Pi computer that also serves as computation base for the control algorithm. The highest level included in implementation of this model, the *SCADA/HMI level*, is represented by the Raspberry Pi computer, since it provides the generation of ball's reference trajectory, adjustment of the control parameters and remote visualization.

3. HARDWARE DESIGN OF THE NEW MODEL

The upgraded hardware implementation brings several fundamental changes to the entire system. The original analog servomotors have been replaced with digital ones, allowing more precise adjustment of the inclination of the plate. For obtaining the angle and angular velocity values, the MPU6050 triaxial gyroscope was placed on the bottom of the tiltable plate. The illumination of

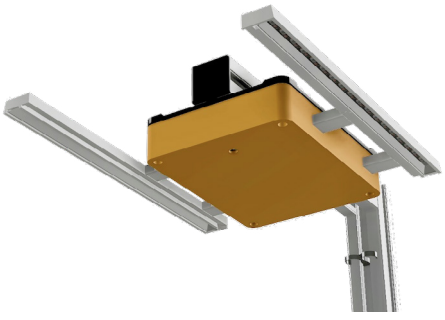


Fig. 1. 3D render of the box for electronics components

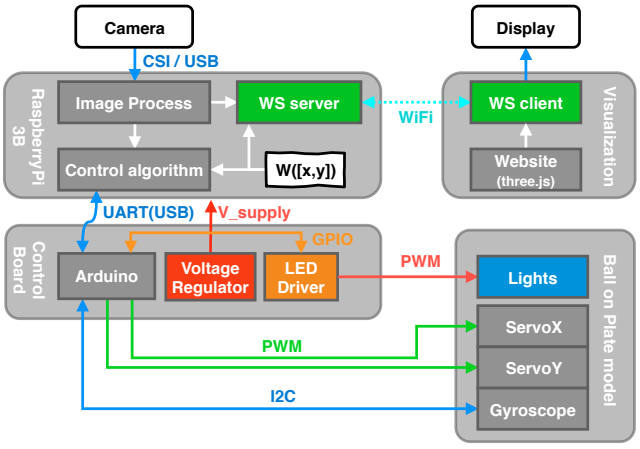


Fig. 2. Concept scheme of the Ball&Plate model

the plate has also been changed: the new model uses two LED strips with PWM control, which significantly improved the lifespan of light compared to previously used neon tubes, see Fig. 1. The external computer used for image recognition and control algorithm implementation has been replaced with a Raspberry Pi 3 single-board computer (Richardson and Wallace (2012)). Instead of the previously used basic web camera, the new implementation uses the 8-megapixels Raspberry Pi camera V2, which is connected directly to the Raspberry Pi using the CSI bus. The logical diagram which depicts the interconnection of the individual Ball&Plate model components can be seen in Fig. 2.

The new hardware design has also allowed to move the implementation of control algorithms to the lower level of the architecture, thus providing faster and more accurate control. While in the original system, the control algorithm was implemented in the MATLAB simulation environment (Oravec and Jadlovská (2014), Oravec and Jadlovská (2017)), in the new system it is implemented in a C++ application.

Due to the obsolete control board of the original system, it was necessary to design and manufacture a new version that would be compatible with the Raspberry Pi computer. For the ease of implementation, the Arduino Nano platform with the ATMega328 MCU was chosen as the main control unit on the board. Power supply and communication with a supervisor computer is provided by a USB interface that connects the Arduino with the Raspberry Pi (Tkáčik (2019a)).

The control board mainly serves to maintain the required position of both servomotors and to provide a communication interface with the attached gyroscope located at the bottom of the tiltable plate of the model. The signal inputs of the individual servomotors are connected to the PWM outputs of the Arduino and their power supply is provided by a 6 V linear regulator 7806T. The gyroscope is connected to the Arduino using the I2C bus with the Fastwire communication extension and is powered directly from the Arduino platform.

The control board also provides power supply for the Raspberry Pi single-board computer using a 5 V linear regulator 7805T (Tkáčik (2019a)). The last part of the

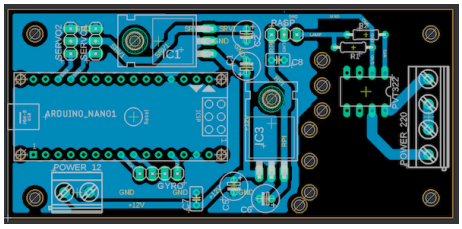


Fig. 3. PCB design for model's control board

control board is the lighting control section, which enables the dimming LED strips via Raspberry Pi's software. Fig. 3 depicts the final version of the proposed printed circuit board for the model's control unit.

To power the control board and the Raspberry Pi computer, a 12 V switching power supply is used. All the electronic components - the control board, the Raspberry Pi with the camera and the power supply are placed in a box attached directly to the model's skeleton.

The 3D render of the placement of individual model components in the box can be seen in Fig. 4. The control board is located on the left, with the camera below it. The Raspberry Pi supervisor computer is located in the middle and the power supply is located on the right.

4. SOFTWARE DEVELOPMENT

The software components of the Ball&Plate model fall into two categories. First of these is the firmware for Arduino Nano that ensures the control of the individual servomotors and readout of gyroscope values. The second one is a set of program modules within the Raspberry Pi supervisor computer that ensures localization of the ball on the plate, calculation of control algorithms and visualization of ball's position.

4.1 Firmware for the Arduino Nano

The program for the Arduino is written in C++ and provides two main functionalities: control of the individual servomotors and readout of the rotation values from the gyroscope. Communication with Raspberry Pi takes place by means of the UART via the USB 2.0 serial bus. The UART transfer rate is set to 1,000,000 Mbps, using 8-bit word length, no parity, and one stop bit (Tkáčik (2019a)).

The readout of the data from UART is performed in an infinite loop of the main program function. The program checks the amount of data received in each cycle and if three bytes are received, the data is evaluated. The first byte specifies which servomotor the command is intended for, and the other two bytes represent a 16-bit integer containing the desired servomotor angle with the precision of one-tenth of a degree. This number is then recalculated to the PWM pulse duration in the range of one to two milliseconds. Based on the calculated value, the desired duty cycle is set on the specific channel of the timer serving the given servomotor. Reading of the rotation values from the gyroscope is performed using I2C interface for gyroscope communication with the aid of MPU6050 library. The data reading itself is implemented in the timer interrupt handler set at 25 ms interval. The obtained values are subsequently sent to the Raspberry Pi using the UART. Rotation data of each of the three axes are sent to the Raspberry Pi using a three-byte message format, where the first byte represents the gyroscope axis and the other two bytes contain the 16-bit value read from the gyroscope.

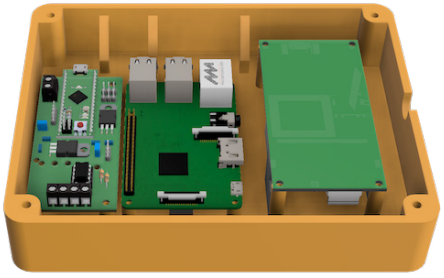
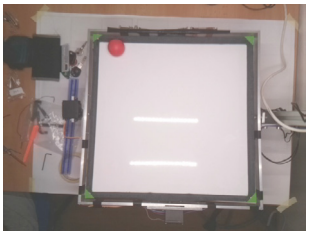


Fig. 4. 3D render of components placements inside the box

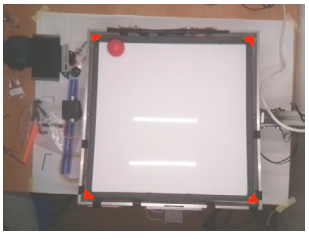
4.2 Software for the Raspberry Pi

The program the implemented in Raspberry Pi consists of four program modules: *Interface*, *Vision*, *PolyReg*, and *Main*. The *Interface* program module provides communication with the Arduino via USB interface and positioning of servomotors to the desired angles. This module can also adjust the offset of each servomotor to refine the zero position setting in both axes. The *Interface* module also allows for reading of the plate inclination values from the gyroscope (Tkáčik (2019a)).

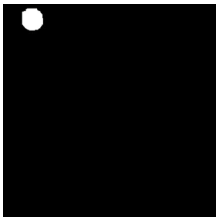
The *Vision* program module reads the camera image and then locates the plate area by segmenting the color labels located in the plate corners (Březina (2019)). The located area is transformed into a square shape by a warp transformation, which is followed by the localization of the ball by segmenting its color and finding the center of gravity pertaining to the obtained points. The resulting data obtained from this module are the coordinates of the ball position in the X and Y axis relative to the center of the plate. The entire process of ball localization is visualized in chronological order in Fig. 5 (a-d).



(a) Original Image



(b) Green corners seg.



(c) 2D transform, red seg.

(d) Ball localization

Fig. 5. Image recognition steps

The *PolyReg* program module provides a versatile implementation of the discrete PSD, as well as the polynomial controller. Implementation of this module allows to change the parameters of the controller while the program is running, and also provides the possibility to set the control output saturation. For this model, two instances of the *PolyReg* program module are used, since the control of the ball's position in the X and Y axes is performed independently.

The *Main* module provides the initialization of all mentioned program modules, setting limits and parameters of controllers and desired ball trajectory. After the successful initialization, it enters the loop that is executed every 25 ms. In this loop the *Vision* module is used to find the position of the ball. After the successful localization of the ball, its position coordinates are passed into instances of the *PolyReg* program module controllers and the calculated action values are sent to the Arduino using the *Interface* module.

4.3 Testing of several approaches of image recognition

Image recognition within the *Vision* module is implemented using the C++ language and the OpenCV library, which offers a number of tools for image processing (Senthilkumar et al. (2014)). When recognizing the area of the plate, green triangles in the corners of the plate are used to detect of the plate edges, see Fig. 5a. Since the plate can be tilted against the camera, it was necessary to use a 2D image transformation to place this image parallel to the camera. The ball is recognized based on the center of gravity of the filtered red color on the image.

First, it was necessary to obtain a camera image that was subsequently converted from BGR (Blue Greed Red) to HSV (Hue Saturation Value) format. Next, the obtained image is searched for the green color, based on the experimental selection of the range of hue, saturation and value parameters, where it was necessary to find the appropriate lower and upper limits of the individual parameters.

After segmenting the green color (see Fig. 5b), it was necessary to group clusters of green pixels into groups belonging to each corner of the plate. The grouping is based on comparing the distance of the newfound pixel from the center of gravity of the already found clusters. A new cluster is created if the processed pixel is far enough from all existing clusters. In the end, clusters with the largest number of points are considered. These clusters has to be assigned according to their position to individual corners of the plate.

After assigning the clusters to particular corners, the 2D transformation of the desired area is applied to get the image of the surface parallel to the camera. After reducing the image size and filtering out the red color (see Fig. 5c) it is possible to locate the ball. Since the illumination is almost uniform across the entire area, determining the center of gravity of the cluster of red points yields accurate information about the position of the ball, which can be seen in Fig. 5d.

Several algorithms were used to determine the position of the ball on the plate, while the performance of the individual algorithms was measured and evaluated (Tkáčik (2019b)). The time needed for the localization of the ball on the plate using different algorithms can be seen in Fig. 6. The first non-optimized algorithm (previous single thread) required up to 65 ms to locate the ball. Since this algorithm was not suitable for use due to the required sampling rate of the control algorithm, it was redesigned using multiple image recognition optimization algorithms, making it able to locate the ball in 30 ms (new single thread).

The next algorithm (new multithread) adds to the previous algorithm multi-thread image processing, where one thread takes care of the image acquisition from the camera and the other performs the ball's localization itself. This algorithm saved in average 3 ms compared to the previous one. The last algorithm (new - ROI multithread) uses the same processes as the previous one, except that only the region of interest, where the plate is expected to be in, is read from the camera. This algorithm saved in average an additional 7 ms, with the total time of 20 ms in average required to locate the ball. This algorithm achieves more than three times better performance than the original algorithm and is currently used in the *Vision* module. To visualize the output of control and image processing algorithms, a web based visualization is used (ball position and plate inclination visualization using web application), see Fig.7.

4.4 Experimental identification of model parameters

In order to design and implement model-based control algorithms, an accurate mathematical model of the system is necessary. In our case, we used experimental identification to obtain model parameters (Jadlovská et al. (2016)).

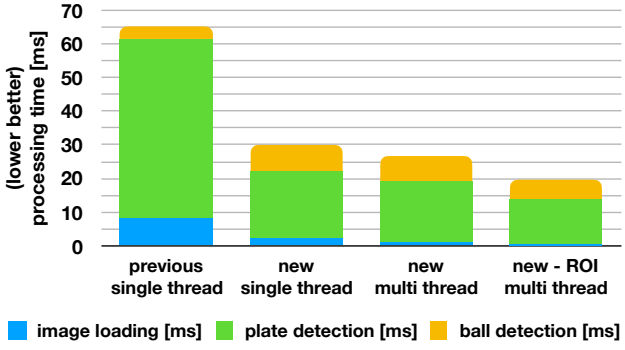


Fig. 6. Ball localization algorithms exec. times comparison

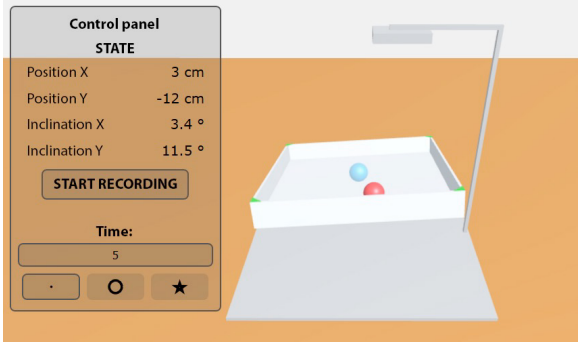


Fig. 7. Visualization of the ball's position on a website

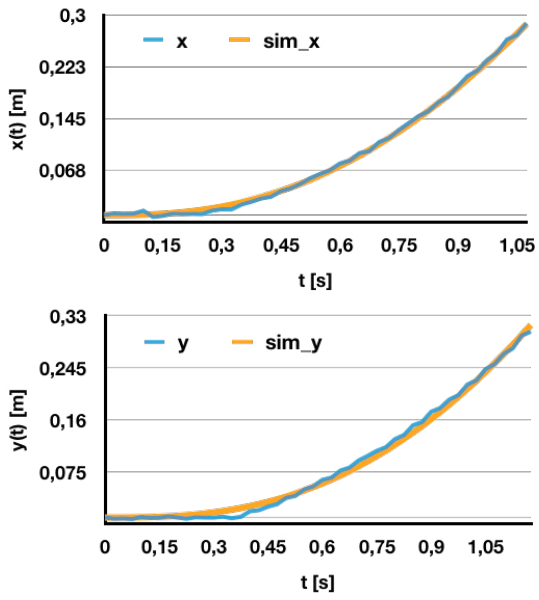


Fig. 8. The step response of the real/simulation model

The Ball&Plate model is considered as two independent SISO systems, with the input of each being the rotation angle of the respective servo motor (the servomotors have the control of angle implemented), while the output is the ball position in the given axis. Since the entire system can be divided into two independent subsystems, model parameter identification has been performed independently and separately for each subsystem (Oravec and Jadlovská (2015)).

Before the data acquisition started, the servomotor was turned to the limit position considered in control, i.e. 0.17 rad. The ball was placed on the edge of the plate on the lower side, so that it does not move spontaneously. Subsequently, the capture of the ball position was initiated (only one axis was considered for each experiment) and the required inclination was set to the other limit position (i.e. -0.17 rad). Once the ball motion had stabilized, data capture was finished. The goal of the experimental identification was to obtain the gain K and time T constants for the systems corresponding to each axis of the model, so that we can obtain transfer functions in the form of:

$$F(s) = \frac{K}{(Ts+1)s^2} \tag{1}$$

From (1), we get a function of the step response of the system in the time domain using inverse Laplace transform:

$$x(t) = K\left(\frac{t^2}{2} \quad T\left[t + T(e^{-\frac{t}{T}} \quad 1)\right]\right) \tag{2}$$

For the purpose of experiment identification, we need to express (2) in discrete form, since the measured data are represented by discrete values. Therefore we introduce a substitution $t = k\Delta t$, where k is the number of current sample and Δt is the sampling interval:

$$x(k) = K\left(\frac{k^2\Delta t^2}{2} \quad T\left[k\Delta t + T(e^{-\frac{k\Delta t}{T}} \quad 1)\right]\right) \tag{3}$$

To obtain the K and T constants of the system, we use the least squares regression method. We therefore define residuals as $r_k = x_k \quad x(k)$, where x_k is the measured value of the system's output in the sample number k and $x(k)$ is the output of the system calculated using (3). The optimal parameters of the system are found by minimizing the sum of squared residuals S:

$$S = \sum_{k=1}^{n} r_k^2 \tag{4}$$

The square of residuals is minimal when all partial derivations of S are equal to zero, so optimal parameters K and T can be found by solving the set of two equations:

$$\begin{aligned} \frac{\partial S}{\partial K} &= 2\sum_{k=1}^{n} r_k \frac{\partial r_k}{\partial K} = 0 \\ \frac{\partial S}{\partial T} &= 2\sum_{k=1}^{n} r_k \frac{\partial r_k}{\partial T} = 0 \end{aligned} \tag{5}$$

Equations (5) are solved in the MATLAB simulation environment with respect to the obtained data of the ball's position in response to the input signal of the unit step (change of the required servo angle from one value to the other). The transfer functions for both axes with approximated constants K and T differ slightly from each other, even if the same servomotors are used, because the hardware design is not identical in both axes. The step response of the laboratory and simulation model in axis X and Y can be seen in Fig. 8.

After obtaining the transfer functions for both axes, it was possible to continue designing the closed-loop control algorithm, see Fig. 9. The quality of the control algorithm based on PID controller parameters computed via Graham-Lathrop's method was first verified in Simulink using transfer functions obtained via experimental identification (Kladný (2019)).

After the verification of PID controller parameters, these parameters were converted into the discrete form of the PSD controller using the trapezoidal rule with a 25 ms sampling period. The calculated parameters were then applied to instances of the *PolyReg* program module for both system axes. To verify the functionality of the proposed control algorithm, a simple application was developed to ensure that the ball's position is maintained in the center of the plate. In the beginning, the ball is in the corner

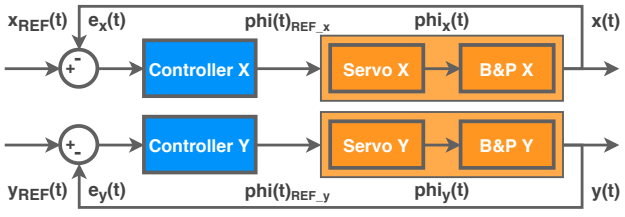


Fig. 9. Closed-loop controllers for X and Y subsystems

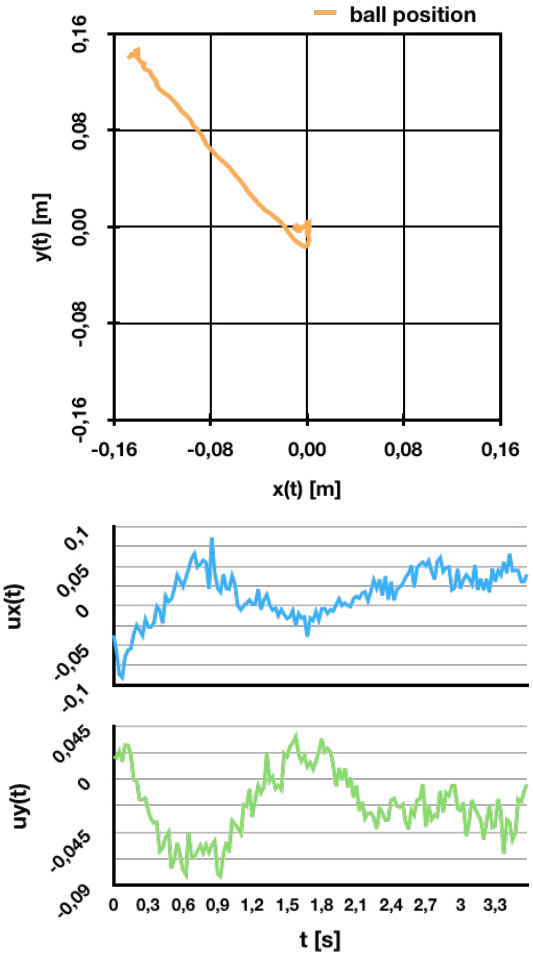


Fig. 10. Ball trajectory to position [0,0]

of the plate and the tilting of the plate must be able to get the ball to its center and then hold it there. With this application, it is possible to verify the stability of the designed control algorithm. The course of controlled variables and control actions during the experiment can be seen in Fig. 10, showing acceptable, accurate control performance.

5. CONCLUSION

This paper presented the upgrade of the Ball&Plate laboratory model. The upgrade included extensive hardware modifications with use of modern embedded system techniques and implementation of new software modules for controlling the position of the ball. During the upgrade, several approaches of image recognition were tested and the final approach provided three times better performance than the initial one. Further modifications will include more powerful power supply regulator for powering the servomotors and achieving even more stable regulation of the plate's inclination.

This model can be used within the CMCT&II for research activities, including the verification of state-feedback optimal or model predictive control algorithms. It can be also used in the educational process to demonstrate the design and implementation of either basic or advanced control algorithms.

ACKNOWLEDGEMENTS

This work has been supported by the grant KEGA - Implementation of research results in the area of modeling and simulation of cyber-physical systems into the teaching process - development of modern university textbooks - 072TUKE-4/2018.

REFERENCES

C. J. Bay and B. P. Rasmussen. Exploring controls education: A re-configurable ball and plate platform kit. In *2016 American Control Conference (ACC)*, pages 6652–6657. IEEE, 2016.

A. Březina. Implementations of embedded systems in robotics systems (in slovak), pages 55-68. Master's thesis, DCAI, FEEI, Technical University of Košice, 2019.

D. Debono and M. Bugeja. Application of sliding mode control to the ball and plate problem. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 412–419. IEEE, 2015.

A. Jadlovská, Š. Jajčišin, and R. Lonščák. Modelling and pid control design of nonlinear educational model ball & plate. In *17th International Conference on Process Control*, volume 2, pages 871–874, 2009.

A. Jadlovská, S. Jadlovská, and D. Vošček. Cyber-physical system implementation into the distributed control system. *IFAC-PapersOnLine*, 49(25):31 – 36, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

L. Kladný. Control and simulation of ball&plate laboratory model using matlab interactive tools (in slovak), pages 40-50. Bachelor's thesis, DCAI, FEEI, Technical University of Košice, 2019.

E. Lee. The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869, 2015.

M. Oravec and A. Jadlovská. Design of virtual models of mechatronics systems with simulink 3d animation toolobox. *Technical Computing, Bratislava*, 2014.

M. Oravec and A. Jadlovská. Model predictive control of a ball and plate laboratory model. In *2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 165–170. IEEE, 2015.

M. Oravec and A. Jadlovská. Intelligent positioning plate predictive control and concept of diagnosis system design. *Manufacturing and Industrial Engineering*, 15 (1-2), 2017.

M. Richardson and S. Wallace. *Getting started with raspberry PI.* " O'Reilly Media, Inc.", 2012.

G. Senthilkumar, K. Gopalakrishnan, and V. S. Kumar. Embedded image capturing system using raspberry pi system. *International Journal of Emerging Trends & Technology in Computer Science*, 3(2):213–215, 2014.

M. Tkáčik. Embedded systems and their implementation in distributed control and monitoring systems (in slovak), pages 59-69. Master's thesis, DCAI, FEEI, Technical University of Košice, 2019a.

T. Tkáčik. Design and implementation of a set of applications in the field of computer vision (in slovak), pages 40-50. Bachelor's thesis, DCAI, FEEI, Technical University of Košice, 2019b.