Hardware and control design of a ball balancing robot

Ioana Lal, Marius Nicoară, Alexandru Codrean, Lucian Buşoniu

***Abstract*— This paper presents the construction of a new ball balancing robot (ballbot), together with the design of a controller to balance it vertically around a given position in the plane. Requirements on physical size and agility lead to the choice of ball, motors, gears, omnidirectional wheels, and body frame. The electronic hardware architecture is presented in detail, together with timing results showing that real-time control can be achieved. Finally, we design a linear quadratic regulator for balancing, starting from a 2D model of the robot. Experimental balancing results are satisfactory, maintaining the robot in a disc 0.3 m in diameter.**

I. Introduction

A ball balancing robot (ballbot, for short) is a vertical robotic platform mounted on top of a sphere that rolls on the ground, thereby moving the robot. Actuation is usually realized by a set of omnidirectional wheels that sit on the ball and are connected to the rest of the robot body. Compared to the usual, wheeled ground robots, which are inherently stable, a ballbot is inherently unstable and it must always be actively controlled to stabilize it and prevent it from falling. On the other hand, the same instability confers it much more agility (e.g., it can move fast by tilting, and it can achieve any trajectory on the horizontal surface) [5].

Quite a number of ballbot designs have already been proposed in the literature. For instance, the ballbot designed in [5] is a very agile one, reaching a maximum speed of 3.5 m/s and is also able to tilt up to a maximum angle of 17°. Additionally, in the balancing phase, it deviates from the initial point with a maximum of 1 mm. Another example is the ballbot developed in [10]. This robot is unique for being very large, and it is able to transport a person. It reaches a body weight of 68 kg, which is unusually large compared to previously developed ballbots. Ballbots with four rollers instead of three omniwheels have be designed in [8] and [6]. Other ballbots have been developed in e.g. [7], [4].

Our design is inspired by these versions, indeed mostly by [5]. However, our objectives are different. In particular, the main goal is to use the robot to demonstrate certain technologies such as smart office assistance, autonomous guidance and negotiating right-of-way with humans, etc. Additionally, we plan to use the robot as a benchmark for advanced nonlinear control methodologies developed in our research. Finally, the robot may be used as a teaching tool in robotics, system-theory, and control engineering courses.

I. Lal and M. Nicoară are with the Engineering Center Cluj, Robert Bosch, Romania. A. Codrean and L. Buşoniu are with the Automation Department, Technical University of Cluj-Napoca, Romania. Email addresses: ioanalal04@gmail.com, marius.nicoara@ro.bosch.com, {lucian.busoniu, alexandru.codrean}@aut.utcluj.ro.
This work was supported by a grant of the Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P1-1.1-TE-2016-0670, within PNCDI III.

These objectives lead to some specific challenges and unique features of the robot. First of all, we will need to add different functionalities in the future, so it is important to be able to easily make changes in the robot's structure. Therefore, we design it to be modular. Secondly, since one of its subsequent purposes is to be an office assistant, we need a robot which has an appropriate height for interacting with humans in an office environment. Thus, we chose to build a medium-sized system, as tall as an office desk. Furthermore, in the future, we may need to increase the weight and height of the ballbot, as well as to modify some of its parameters. Control must therefore be robust to small parameter variations, and should be easy to adjust when larger variations require redesigning the control law.

Next, Section II presents the design of the ballbot system. Section III explains our control design and provides real-time results for balancing the robot. Section IV concludes the paper and provides some pointers for future work.

II. System design

As is typical in robotics, the physical ballbot represents a fusion between mechanical and electrical subsystems. Moreover, the software that is implemented on top of these two base layers, along with the control algorithms of the robot, are in a close inter-dependency with the entire system. In the next sections, the mechanical, electrical and software subsystems will be detailed in turn.

A. Mechanical design

The first step of the mechanical design was to establish the dimensions of the main assemblies that are integrated in the robot. Figure 1 presents the structure of the ballbot with its main units: body, base, and ball.

The most important characteristics of the mechanical structure of the robot are to achieve a sufficient stiffness and precision, and to be modular. In order to satisfy stiffness and precision, we used aluminum alloy as base structure material (AlMg4,5Mn0,7). In order to satisfy the modularity of the system, the structure of the robot (Figure 1) was designed from adjustable parts (adjustable positioning supports for the electrical components, modular aluminum base structure, pressure adjustable gripper, etc.) that make it easy to reconfigure the structure of the robot: changing the ball diameter, the angle or dimension of the omniwheels, the center of gravity of the robot, etc. Our target was to ensure that the minimum requirements of the system are reached even if the functionality of the robot may vary in the future.

One of the most critical requirements was to ensure a high enough torque at the end of the shaft of each actuator. The first estimation regarding the needed torque was around the

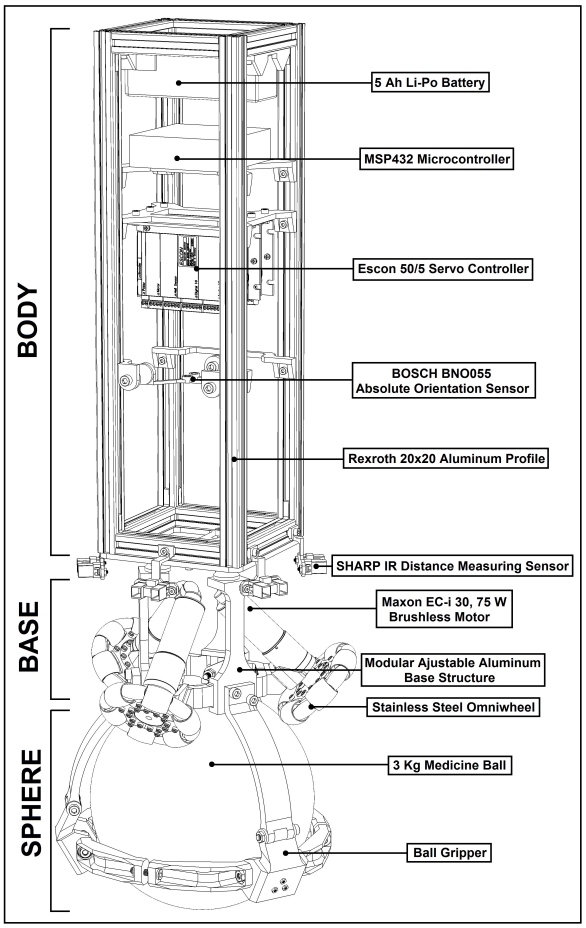

Fig. 1. Structure of the ballbot

value of 6 Nm, a spike value and not a continuous one in the running mode of the robot. Therefore, the electrical motor that we have chosen is a brushless DC motor that is capable to generate a continuous torque of 0.11 Nm (Maxon EC-i 30). A gearbox is required to increase the output shaft torque, reducing the speed. The solution we chose was a planetary gearbox with a 21:1 reduction (Maxon Gearhead GP 32 C). Thus, the gear-motor combination will deliver a constant torque around 2.3 Nm at 5 A and a maximum torque of 6.7 Nm at 15 A.

The omnidirectional wheels (omniwheels, for short) that are attached to the motor shaft need to have a proper diameter so that the gear-motor-encoder assemble can be mounted in a correct position and at a correct angle. Initially, we tried a double-sided plastic omniwheel with rubber rollers as shown in Figure 2. This solution was not suitable for our application: due to the double-point contact between the omniwheels and the surface of the ball, considerable vibrations occurred throughout the structure, omniwheels often lost contact with the ball. Also, because of the smooth surface of the rubber rollers, slipping appeared. A better solution was to use single-sided stainless steel omniwheels, 100 mm in diameter and with a load capacity around 10 kg. Their rollers are also manufactured from stainless steel, with a striated surface that ensures a nearly continuous single-point contact with more grip.

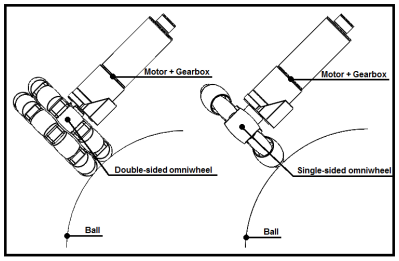

Fig. 2. Double-sided omniwheel vs. single-sided omniwheel

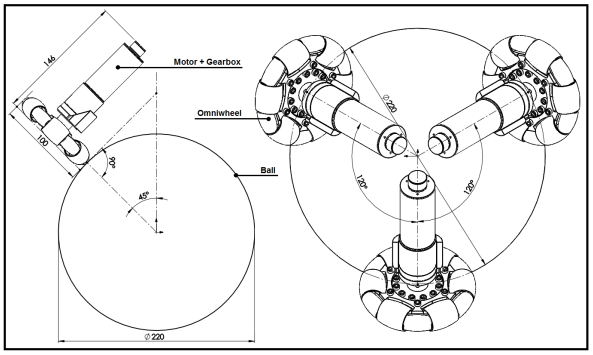

Fig. 3. Contact angle and the distribution of the omniwheels

In order to achieve a good controllability of the robot, a compromise is needed between sufficient support and the robot's agility. This is determined by the Zenith angle, which is the angle between the contact point of the omniwheels and the ball, and the vertical axis of the ball. A higher angle (close to $90°$) makes the movement on the floor very difficult, as the robot cannot easily rotate around the X and Y axes. A smaller angle (close to $0°$) makes the support triangle of the robot very narrow, so it becomes more complicated to balance the robot and to rotate around the Z axis [4]. The Zenith angle is thus chosen to be $45°$, as shown in Figure 3. Figure 3 also highlights the circular distribution of the omniwheels around the ball.

In some situations, due to high acceleration of the wheels or large tilt of the robot, even the new omniwheels lose contact with the ball surface if they are left to freely sit on it. Therefore, an adjustable ABS plastic 3D printed gripper with 3 plastic ball casters was designed and mounted on the bottom of the ball in order to ensure a permanent contact between the ball and omniwheel. The gripper pressure can be adjusted using three screws.

The sphere consists of a 3 kg rubber covered medicine ball with a diameter of 220 mm, which should guarantee a sufficient inertia moment considering the entire weight of the system (6 kg). We chose a medicine ball due to its rough texture, which helps in avoiding slippage between the ball on the one hand, and the omniwheels or the floor on the other.

However, because of this rough texture and of the irregular horizontal surface on which the robot moves, significant vibrations can occur. Vibration rubber dampers were mounted between the base and the body of the robot (not visible in the Figure 1), to reduce the vibration propagation throughout the structure, in order not to disturb the orientation sensor and other fragile electronic components.

B. Electronics

Figure 4 presents the hardware architecture of the ballbot.

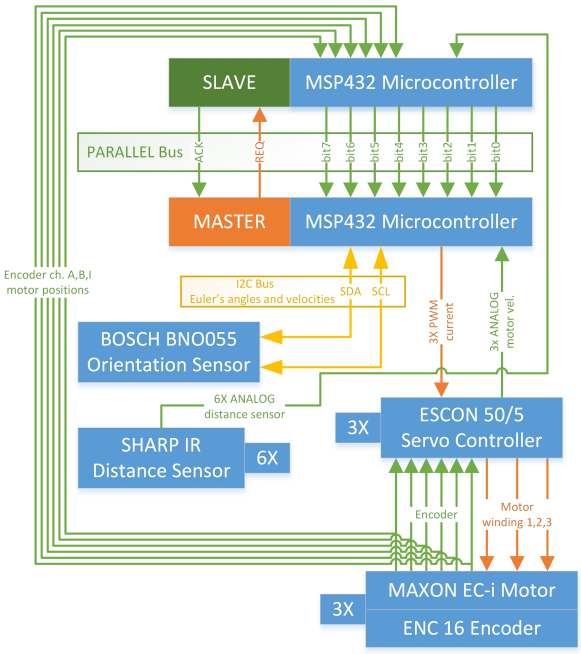

Fig. 4. Hardware architecture

As previously explained, the electrical actuators were chosen as a result of the required torque at the end of the shaft. For the type of brushless motors we selected (Maxon EC-i 30, 75 W, 0.11 Nm continuous torque [3]), there is a near-linear dependence between the output torque and the input current, described by the motor torque constant, with a value of 0.0214 Nm/A. The stabilizing controller for the robot gives a torque command. This command is equivalent to a computed current. The output currents which emerge after the execution of the control algorithm are applied individually to each of the three motors, via the motor servo controllers (ESCON 50/5). Thus, we have an internal control loop that ensures that the required current is the actual current in the motors.

The main processing unit (master board) is a MSP432 Texas Instruments microcontroller [2] capable of running at a frequency up to 48 MHz. It is responsible with absolute orientation data acquisition from the BOSCH BNO055 [1] inertial measurement unit (Euler's angles and their velocities). These two components communicate on a 400 kHz I2C bus. Also, this microcontroller is connected with the three servo controllers, sending via PWM signals the requested current/torque of each motor and receiving via analog signals the angular velocity measurements from the motors.

While this main processing unit is powerful enough for the needed computations, it is insufficient to acquire the large amount of data required, due to its low number of input-output pins and limited ability to handle a large number of interrupts. Therefore, a second data acquisition microcontroller (slave board) was added to the system in order to deal with multiple interrupts given by the input signals: three quadrature encoder signals (128 counts/turn) and six analog signals from the IR distance sensors.

The information acquired by this second microcontroller has to be transferred to the main one (master-slave data transfer). In order to do that, we have created a high-speed custom designed parallel protocol between the master and slave microcontrollers that can transfer large amounts of data in a short period of time (300 kB/sec). A suitable solution for our application was to use an entire 8 bits I/O port on both boards as data bus and other two pins as request bit from master and acknowledgment bit from slave. This solution was chosen based on our previous positive experience with it. In the future, an additional board will be added for higher level tasks, such as image processing, trajectory planning or human interaction. An alternate architecture is proposed in [10], built around a mini PC that deals with the data acquisition and computing the control signals. It also communicates using an RS232 protocol with a second board, which commands the auxiliary 4 motors that act on control moment gyroscopes.

Our main power supply is an 18.5 V, 5 Ah Li-Po battery, directly connected to the servo controllers, maintaining the demanding power required by the motors. This battery grants more than 8 h autonomy (a full working day) in balancing mode. A smaller, single cell Li-Po battery was also introduced to supply the logic system and some sensors. At the moment, we do not handle the situation when the robot's battery dies. However, in the future, a "skirt" system will be added that prevents the robot from falling for any reason, including low battery.

C. Software and timing

The software implemented on the MSP432 microcontroller has a sequential design in which the software functions are called one after the other, in a prescribed order. We have used a register-level C programming language, software being divided in multiple .c and .h files. We have created initialisation functions at system startup that test the functionality of the sensors and wait for the user to press the start button in order to confirm that the system is ready to be engaged in movement. The main loop represents a series of function calls, in which the sensors are read, the high-level controller algorithm is processed and its output signals are sent to the servo controllers of the motors.

Figure 5 highlights the timing of the main loop in the context of the required sampling period by the controller. In our case, we adopted a 10 ms sampling period, since the timing of the main loop allowed it, and a smaller sampling time usually leads to better control performances.

We recorded both main buses with a logic analyzer: SDA and SCL are the data and clock lines of the I2C bus and the REQ and ACK represent the request and acknowledgement lines of the parallel bus. The 10 ms sampling time can be divided in three main time intervals: signal acquisition time (t_a), control algorithm processing time (t_p) and sampling time delay (t_d). During the sampling time delay, the loop waits for the 10 ms to pass, in order to start a new sampling

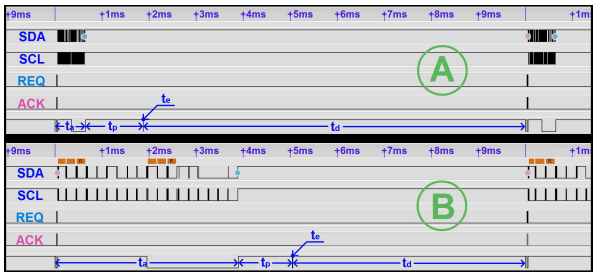

Fig. 5. Software cycle timing: without (A) and with (B) clock stretching

period. Figure 5 also shows the execution time t_e, needed to send the control signals from the microcontroller to the servo controller, but this time interval is very short, on the order of nanoseconds. The time needed for the currents to stabilize is under 0.5 ms and is included in t_d. Defining the sampling time as t_s, we can state: $t_s = t_a + t_p + t_e + t_d$.

There are two different cases captured in Figure 5. In the first case (top plot, A) the 10 ms sampling time is shown with a typical short time data acquisition ($t_a = 0.64$ ms). The second case (bottom plot, B) highlights a much greater data acquisition time ($t_a = 3.89$ ms), six times longer than the normal case. This phenomenon is known as I2C clock stretching. The BNO055 IMU delivers fused absolute orientation data that are internally calculated from accelerometer, gyroscope and the magnetometer, and, if the calculations are not ready when the master controller requests them, clock stretching appears. Fortunately, for our 10 ms sampling time, we are sure that a minimum 5 ms time interval is available for processing time ($t_p = 1.22$ ms, nearly constant value for all cases) and for execution ($t_e = 0.0038$ ms, nearly constant value for all cases). The remaining time (t_d) varies therefore from 5 ms (plot B) up to 8 ms (plot A) allowing us to increase the processing time by implementing other more complex controllers.

III. Model-based control design

A ballbot is inherently unstable, thus a proper controller is important in stabilizing it, prior to any other task such as moving the robot in the horizontal plane. We will apply a model based controller design; as such, a model of the system is needed. The following subsections introduce, in turn, the mathematical model of the ballbot, the controller design, and some experimental results.

A. Dynamic model

1) Nonlinear model: In this section, the model of the ball-balancing robot is presented, as derived in [10]. The model is developed using the Euler-Lagrange equations [11], an approach for modeling a robotic system based on the energy balance. First, some assumptions are made: there is no slip between the ball and the floor or the ball and the omniwheels, the ball is a perfect sphere, the floor is flat and the dynamics of the motors are fast enough to be disregarded. Moreover, we neglect the friction, because it would add more complexity to the model and its parameters are difficult to estimate. Here, the 3D model is considered to be a combination of three 2D models, by separating the

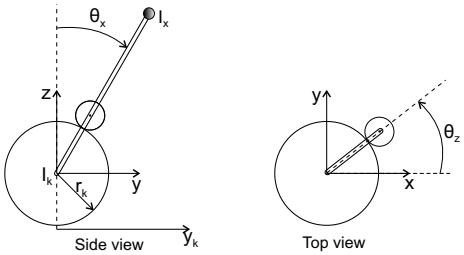

Fig. 6. The coordinates for planes YZ and XY

planes YZ, XZ and XY. These planes can be seen in Figure 6. It should be noted that the X axis is aligned with motor 1. The minimal coordinates for the planes are:

$$q_x = \begin{bmatrix} y_k \\ \theta_x \end{bmatrix}, q_y = \begin{bmatrix} x_k \\ \theta_y \end{bmatrix}, q_z = [\theta_z] \tag{1}$$

where θ_x, θ_y and θ_z are the Tait-Bryan angles of the body, and x_k and y_k give the ball's position on the floor. The robot dynamics in the YZ plane are:

$$M(q_x)\ddot{q}_x + C(q_x, \dot{q}_x)\dot{q}_x + G(q_x) = Q_x \tag{2}$$

where:

$$\begin{aligned} M(q_x) &= \\ &\begin{bmatrix} m_k + \frac{I_k}{r_k^2} + m_a + \frac{3I_w \cos^2\alpha}{2r_w^2} & \frac{3I_w \cos^2\alpha}{2r_w^2} r_k - m_a l \cos\theta_x \\ \frac{3I_w \cos^2\alpha}{2r_w^2} r_k - m_a l \cos\theta_x & m_a l^2 + \frac{3I_w r_k^2 \cos^2\alpha}{2r_w^2} + I_x \end{bmatrix} \\ G(q_x) &= \begin{bmatrix} 0 & -m_a g l \sin\theta_x \end{bmatrix}^T \\ Q_x &= \begin{bmatrix} \frac{1}{r_w}\tau_x & \frac{r_k}{r_w}\tau_x \end{bmatrix}^T \\ C(q_x, \dot{q}_x) &= \begin{bmatrix} 0 & m_a l \dot{\theta}_x \sin\theta_x \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{3}$$

with the parameters described and given in Table I. The equations for the XZ plane are analogous to the ones for the YZ plane. There is only one equation for the XY plane:

$$\ddot{\theta}_z = \frac{I_k}{I_k I_z + 3(I_k + I_z) I_w \frac{r_k^2}{r_w^2} \sin^2\alpha} \frac{r_k}{r_w} \tau_z \tag{4}$$

The inputs for the models τ_x, τ_y, τ_z are the equivalent torques on each plane, but a conversion is needed, since we need to express the model in terms of the motor torques. This conversion is represented by the transformation:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{3\cos\alpha} & 0 & \frac{1}{3\sin\alpha} \\ -\frac{1}{3\cos\alpha} & \frac{\sqrt{3}}{3\cos\alpha} & \frac{1}{3\sin\alpha} \\ -\frac{1}{3\cos\alpha} & -\frac{\sqrt{3}}{3\cos\alpha} & \frac{1}{3\sin\alpha} \end{bmatrix} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{5}$$

where τ_1, τ_2, τ_3 are the torques of each motor, respectively.

2) Linear model: As it can be seen, the YZ and XZ models are nonlinear, so a linearization around an equilibrium point is needed, to apply linear control design. We consider points when the body is upright (balancing). There is an infinite number of such points, for every ball position on the floor. However, for simplicity, we choose to linearize around the point represented by the origin of the coordinate system, which we take to be the initial position of the robot, around which the balancing should be performed. We linearize the

Parameter	Symbol	Value
Mass of the ball	m_k	3 kg
Radius of the ball	r_k	0.11 m
Inertia moment of the ball	I_k	0.0242 kgm^2
Mass of the body	m_a	6.0874 kg
Body moment of inertia around the x-axis	I_x	1.6839 kgm^2
Body moment of inertia around the y-axis	I_y	1.6839 kgm^2
Body moment of inertia around the z-axis	I_z	0.0228 kgm^2
Radius of an omniwheel	r_w	0.05 m
Omniwheel moment of inertia	I_w	0.0015 kgm^2
Height of the body's center of mass	l	0.3251 m
Gravitational acceleration	g	9.81 m/s^2
Zenith angle	α	45°

TABLE I
BALLBOT PARAMETERS

models for the two planes separately, therefore we have state vectors and inputs corresponding to each of the planes. The state vector is represented by the minimal coordinates and their velocities, and the input is the equivalent torque for that plane, $\mathbf{x}_x = [q_x^T \ \dot{q}_x^T]^T, u_x = \tau_x, \mathbf{x}_y = [q_y^T \ \dot{q}_y^T]^T, u_y = \tau_y$.[1] We can now write the model of the system, considering that $\dot{\mathbf{x}}_x = [\dot{q}_x^T \ddot{q}_x^T]^T$, $\dot{\mathbf{x}}_y = [\dot{q}_y^T \ddot{q}_y^T]^T$ and that $\ddot{q}_x$ can be computed from equation (2), by multiplying to the left with $M^{-1}(q_x)$ and $\ddot{q}_y$ can be found in a similar way. Therefore, the model will be:

$$\begin{aligned} \dot{\mathbf{x}}_x &= \mathbf{f}_x(\mathbf{x}_x, u_x) \\ \dot{\mathbf{x}}_y &= \mathbf{f}_y(\mathbf{x}_y, u_y) \end{aligned} \tag{6}$$

with $\mathbf{f}_x$ and $\mathbf{f}_y$ being non-linear vector functions. Their expressions are complex, so they will not be provided here. The equilibrium point is represented by $\mathbf{x}_{x_0} = [0\,0\,0\,0]^T, u_{x_0} = 0$ and $\mathbf{x}_{y_0} = [0\,0\,0\,0]^T, u_{y_0} = 0$.

After linearizing, we obtain a state space form of the model, as follows:

$$\begin{aligned} \dot{\mathbf{x}}_x &= \mathbf{A}_x\mathbf{x}_x + \mathbf{B}_x u_x \\ \dot{\mathbf{x}}_y &= \mathbf{A}_y\mathbf{x}_y + \mathbf{B}_y u_y \end{aligned} \tag{7}$$

By replacing the parameters with their actual values, we get the following matrices:

$$\begin{aligned} \mathbf{A}_x &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 3.4586 & 0 & 0 \\ 0 & 20.6773 & 0 & 0 \end{bmatrix}, \mathbf{B}_x = \begin{bmatrix} 0 \\ 0 \\ 2.1950 \\ 2.7587 \end{bmatrix} \\ \mathbf{A}_y &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -3.4586 & 0 & 0 \\ 0 & 20.6773 & 0 & 0 \end{bmatrix}, \mathbf{B}_y = \begin{bmatrix} 0 \\ 0 \\ -2.1950 \\ 2.7587 \end{bmatrix} \end{aligned} \tag{8}$$

For the XY plane, Equation (4) (which was already linear) becomes after parameter substitution:

$$\ddot{q}_z = 50.0343\tau_z \tag{9}$$

Considering the system state as $\mathbf{x}_z = [q_z \ \dot{q}_z]^T$ and the input as τ_z, we can write the system in state-space form:

$$\dot{\mathbf{x}}_z = \mathbf{A}_z\mathbf{x}_z + \mathbf{B}_z u_z \tag{10}$$

[1]We use subscript x to denote the YZ plane, and bold $\mathbf{x}$ to denote state vectors. Similarly, subscript y denotes the XZ plane.

where the matrices $\mathbf{A}_z$ and $\mathbf{B}_z$ are:

$$\mathbf{A}_z = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{B}_z = \begin{bmatrix} 0 \\ 50.0343 \end{bmatrix} \tag{11}$$

B. Control design

Having the linearized model of the system, we can now design a controller for balancing the robot. We used the linear quadratic regulator (LQR) framework [9] to design three separate controllers, one for each of the robot planes. In this section, we present the model used, the design of the three controllers, and the adjustments we had to make for the real-time experiments.

1) Linear Quadratic Regulator: Consider a state space model of the form:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{12}$$

with $\mathbf{x}$ representing the state vector, $\mathbf{u}$ the control vector, matrices $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{n\times m}$. We shall consider a cost function of the form:

$$J = \frac{1}{2}\int_0^\infty (\mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u})dt \tag{13}$$

with the matrix $Q \in \mathbf{R}^{n\times n}$ being positive-definite or positive-semidefinite, and $R \in \mathbf{R}^{m\times m}$ a positive-definite hermitian or real symmetric matrix. The objective is to find a control that minimizes the cost function. For that, it will be sufficient if we apply a control law of the following form:

$$\mathbf{u} = -K\mathbf{x}, \tag{14}$$

where the gain K is:

$$K = R^{-1}B^T P, \tag{15}$$

P being the stabilizing solution to the Ricatti equation [9]:

$$Q + A^T P + PA - PBR^{-1}B^T P = 0 \tag{16}$$

The cost function that needs to be minimized depends on the Q and R matrices. There is no exact algorithm to choose them; in most cases, the tuning of the matrices is done experimentally. Q determines the importance of the states, while R penalizes the inputs. It is therefore necessary to find a balance between the speed of recovery from an unstable position and the magnitude of the input signals. The two matrices are usually diagonal matrices, often identity matrices or scaled versions thereof. In our case, we found that the best results are given by the following expressions of Q and R: $Q = \mathrm{diag}([50\ 50\ 5\ 5]),\ R = 5$. The resulting K_x and K_y matrices are:

$$\begin{aligned} K_x &= [-3.1623 \ \ 32.9537 \ \ -3.6595 \ \ 7.7799] \\ K_y &= [\ \ 3.1623 \ \ 32.9537 \ \ \ \ 3.6595 \ \ 7.7799] \end{aligned} \tag{17}$$

For the XY plane, an LQR controller is also designed, using the following weight matrices: $Q = \mathrm{diag}([30\ \ 5]), R = 10$, which leads to:

$$K_z = [1.7322 \ \ 0.7636] \tag{18}$$

2) Adjusting the gains: Although the simulations give satisfying results with the above gains, for the real-time experiments some adjustments need to be made. We noticed that, with these gains, the robot is moving too violently, so we decrease the gains accordingly. A solution that works for station-keeping is dividing the gains corresponding to the rotations (namely the angles and the angular velocities) by 3, and the gains corresponding to the movement in the plane (namely the x and y positions and their velocities) by 1.5. Therefore, the new gains are:

$$\begin{aligned} K_x &= [-2.1082 \;\; 10.9846 \;\; -2.4397 \;\; 2.5933] \\ K_y &= [\;\; 2.1082 \;\; 10.9846 \;\;\; 2.4397 \;\; 2.5933] \end{aligned} \tag{19}$$

We can understand this in terms of LQR design as increasing the components of the matrix R accordingly.

C. Experimental results

This section provides illustrative real-time results, in experiments where the ballbot must be balanced in an upright position. In the real-time implementation, we filtered some of the signals with a low pass filter. We used first order filters for the motor speeds, angular velocities and for the command signals, with the time constants 30 ms, 10 ms and 10 ms, respectively. Figure 7 presents the evolution of the angles of the robot in time. The controller succeeds in keeping the robot almost upright, as it inclines at most 5° around the y axis and at most 7° around the x axis.

Figure 8 illustrates, in blue continuous line, the trajectory of the robot in the XY plane (on the floor). As it can be seen, the ballbot remains around its initial position, which is the $(0, 0)$ position. We notice oscillations around the initial position. They occur because the robot always tilts in some direction, so the controller has to compensate and move the robot, in order for it to recover. Moreover, there exists of course unmodeled behavior, including friction, mechanical backlash, and other nonlinear dynamics. In addition to that, the 2D model represents a simplification of the 3D one, which would consider also the coupling between the planes [5]. All these factors lead to the oscillations that we observe.

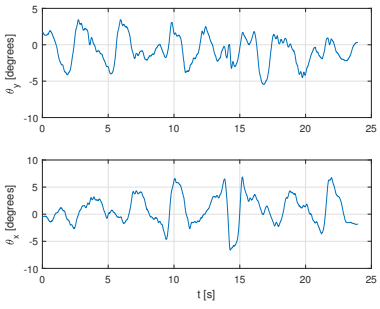

Fig. 7. Robot inclination

In addition, for the purpose of comparison, we present the best obtained result with the first omniwheels we attempted, the ones with 2 rows, see again Figure 2. The position of the robot in the plane can be seen in Figure 8, in black dashed line. As we can see, the oscillations around the origin are much greater in this case. This is because the

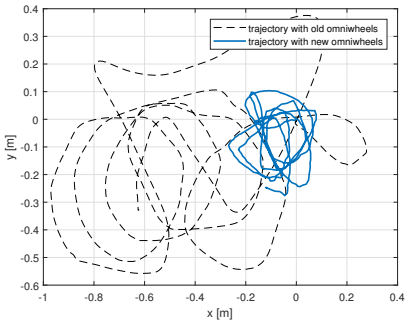

Fig. 8. Ball position in the plane with the old and new omniwheels

the old omniwheels introduced vibrations, due to the passing from one row to another. With the new omniwheels, reduced vibration enabled the use of larger control signals, leading to more accurate control. One objective in future work is to limit the radius of oscillation even more, so as to make it comparable to that of the robots from [10] (10-12 cm), [7] (2-3 cm) or [5] (2-3 mm).

IV. Conclusions

This paper presented the construction of a ball-balancing robot, together with the design of a stabilizing controller to balance the robot around the vertical position. The mechanical and electronic hardware was presented in detail, and the selection of the omnidirectional wheels actuating the ball proved particularly important. Real-time control results showed satisfactory performance, where the robot was maintained in a disc of about 0.3 m in diameter.

An important direction for future work is to consider the coupling between the motions on the three planes in the control design, together with the nonlinear behavior of the robot when significantly tilted from vertical. Tracking control should be developed next, together with higher-level sensing and trajectory generation. To this end, an additional computing board will be added to the platform, together with additional ranging and camera sensors.

References

[1] "BNO055 Datasheet," BST - BNO055 - DS000 - 1, 2014.

[2] "Texas Instruments Technical Reference Manual," SLAU356H, 2017.

[3] "Maxon Motor Datasheet," http://maxon.blaetterkatalog.ch/b9990/catalog, 2018.

[4] J. Blonk, "Modeling and control of a ball-balancing robot," Master's thesis, University of Twente, 2014.

[5] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," B.S.C. thesis, Eidgenössische Technische Hochschule Zürich, 2010.

[6] J. Fong, S. Uppill, and B. Cazzolato, "Design and build a ballbot," *Report, The University of Adelaide, Australia*, 2009.

[7] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *International Conference on Control, Automation and Systems*. IEEE, 2008, pp. 433–438.

[8] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proceedings 2006 IEEE International Conference on Robotics and Automation*. IEEE, 2006, pp. 2884–2889.

[9] K. Ogata and Y. Yang, *Modern control engineering*. Prentice Hall, 2002.

[10] D. B. Pham, H. Kim, J. Kim, and S.-G. Lee, "Balancing and transferring control of a ball segway using a double-loop approach [applications of control]," *IEEE Control Systems*, vol. 38, no. 2, pp. 15–37, 2018.

[11] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. Wiley New York, 2006.