IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 27, 2019, accepted April 24, 2019, date of publication April 29, 2019, date of current version May 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913916

Development of an Easy-to-Use Multi-Agent Platform for Teaching Mobile Robotics

GONZALO FARIAS[1], ERNESTO FABREGAS[2], EMMANUEL PERALTA[1], HÉCTOR VARGAS[1], SEBASTIÁN DORMIDO-CANTO[2], AND SEBASTIÁN DORMIDO[2]

[1]Escuela de Ingenieria Electrica, Pontificia Universidad Católica de Valparaíso, Valparaiso 2147, Chile
[2]Departamento de Informática y Automática, Universidad Nacional de Educación a Distancia, 28040 Madrid, Spain

Corresponding author: Gonzalo Farias (gonzalo.farias@pucv.cl)

This work has been funded by the Chilean Ministry of Education under the Project FONDECYT 1161584, and the Spanish Ministry of Economy and Competitiveness under the Project No. ENE2015-64914-C3-2-R.

ABSTRACT The use of mobile robots for teaching automatic control is becoming more popular in engineering curricula. Currently, many robot simulators with high-graphical capabilities can be easily used by instructors to teach control engineering. However, the use of real robots is not as straightforward as simulations. There are many hardware and software details that must be considered before applying control. This paper presents the development of an easy-to-use platform for teaching control of mobile robots. The laboratory has been carefully designed to conceal all technical issues, such as communications or the localization that do not address the fundamental concepts of control engineering. To this end, a position sensor based on computer vision has been developed to provide the positions of the robots on the platform in real time. The Khepera IV robot has been selected for this platform because of its flexibility and advanced built-in sensors but the laboratory could be easily adapted for similar robots. The platform offers the opportunity to perform laboratory practices to test many different control strategies within a real experimental multi-agent environment. A methodology for using the platform in the lab is also provided.

INDEX TERMS Robotics education, mobile robot laboratory, vision-based indoor positioning sensor.

I. INTRODUCTION

Robotics is a multidisciplinary field that has been introduced in education at all levels in recent years. Mathematics, physics, mechanics, electronics, computer science, and automatic control are some of the fields involved in robotics, which makes it very versatile from a pedagogical point of view. Robotics can help students to understand abstract and complex concepts in an interesting environment.

Nowadays it is common to find robots in many courses and degree programs [1], [2]. Robotics is used into primary and secondary educational levels, where simple experiments with sensors can be done. For example, in [3], [4], the authors present approaches based on well-known platforms (e.g. LEGO NXT Mindstorms, V-REP and LabVIEW) for primary and secondary school students. Robotics is also used for engineering students in higher education, where much more challenging experiments can be performed. For example, in [5], [6] the authors present approaches based on simulations and competitions between robots programmed by the students.

In this context, the development of laboratory practices with robots is a great challenge for instructors. There are several questions that must be answered beforehand to make the laboratory available for students. For example, the types of robots, the scenarios, and the student tasks protocol are the main issues to be solved when designing the laboratory. But, it is also necessary to put under control many technical details that are irrelevant (and maybe undesired) from a teaching point of view. Otherwise, students will dedicate a great effort and time for preparing and configuring several technical issues of the laboratory before actually beginning the practice.

In the literature we can find some research that puts the focus on this kind of issues. For example, in [7] the authors address the design and implementation of robotics remote laboratories employing web technologies (WebLabs). They describe how to incorporate security requirements and quality of service to this kind of applications. Further, they discuss the most appropriate web technologies to fulfill

The associate editor coordinating the review of this manuscript and approving it for publication was Jiahu Qin.

such requirements. In the platform CPE (Control and Programming Environment) presented in [8], the authors propose a remote laboratory for programming and control a mobile robot with simple commands. Which means that students do not have to deal with hardware details or use C/C++ programming language to develop complex programs. In [9] the authors present RoboGen, an open-source and low-cost platform to study a variety of topics including computer programming, artificial intelligence, evolutionary algorithms, etc. This solution is used in the master's level course at the Ecole Polytechnique Federale de Lausanne (EPFL).

Some of the aforementioned approaches may present interesting experiments for engineering students with one robot. However, with these platforms the instructor can not introduce more advanced multi-robots experiments like multi-agents systems (cooperation and collaboration of a group of robots to solve complex problems) [10]. Therefore, some of the following interesting experiments could not be implemented: simultaneous localization and mapping [11], navigation with computer vision, formation control [12], event-based control [13], among others.

This paper presents the development of a platform to design and test control experiments with mobile robots in a multi-agent scenario. The platform is based on the Khepera IV robot [14], which is the most recent mobile robot of the Swiss company K-TEAM. The architecture of the platform is divided into two main parts: a virtual environment (simulation of the system) and an experimental environment (robots in a real scenario). The virtual environment is based on [15], which is a library developed by the authors, to include the Khepera IV robot into V-REP simulator [16]. This simulator is a versatile and scalable framework for creating 3D simulations in a relatively short period of time. It allows to designing and test experiments before the implementation with the real robots.

On the other hand, the experimental environment is composed by four Khepera IV robots that can communicate between them by using a WI-FI network. The rest of the elements of the experimental environment are the following: 1) a PC running the Ubuntu operating system; 2) a local WI-FI network with a router for wireless communication between the robots and the PC; 3) a USB camera (top view of the scenario); 4) the software tool SwisTrack [17], to process images captured from the USB camera in order to obtain the position of the robots; and 5) an IP camera that shows a live streaming video of the experiment. Note that the USB camera, the SwisTrack software, and the WI-FI router work as an Indoor Positioning Sensor (IPS) similar to [18].

The main contributions of this work are the following: 1) the platform is an easy-to-use tool to design and test control experiments with mobile robots in attractive and interactive scenarios; 2) the communication between the robots allows to students the implementation of multi-agent system experiments (decentralized and collaborative approach); 3) the experiments can be designed with a high quality 3D simulator and tested with advanced robots for pedagogical purposes; 4) many technical details such as network communications and the IPS system issues are transparent for the students; 5) with this platform, engineering students can have hands-on experiences in a real-world scenario; and 6) a tasks protocol is provided. This document contains detailed steps for performing each experiment of the laboratory practice with the platform.

The remainder of the paper is organized as follows: Section II describes the platform in detail, considering all its components and their interactions; Section III shows the results of some test experiments developed with the platform; Section IV describes how the platform can be used to teach control engineering; and finally, Section V presents the main conclusions and future work.

II. PLATFORM ARCHITECTURE AND STRUCTURE

A. BACKGROUND

The platform presented in this paper has been designed based on previous works developed by the authors. For example, in [19] we designed, implemented and tested a platform for experimentation with LEGO Mindstorms NXT robots. In this approach the wireless communication between the robots is performed by using Bluetooth technology. This kind of communication consumes a lot of battery, which only allows to send very small data packets (16 bytes), and it is not possible the communication between all the robots at the same time (due to the master-slaves topology). These limitations introduce important drawbacks in the real experimentation, especially for a multi-agent scenario.

In [18] we presented a platform to develop and test position and formation control experiments with Moway robots. In this approach the limitations are similar to the previous platform: the wireless communication between robots is performed by using Radio Frequency (RF), which only allows to send data packets of 8 bytes length. In addition, the processing capacity of these robots is very low, which limits the complexity of the experiments. In both platforms, these issues are mainly due to the low cost of the robots. These two platforms are composed of two main parts: 1) a virtual environment; and 2) an experimental environment. The virtual environment of both platforms is the Simulator RFCSIM presented in [20] that was developed in Java by using the free software tool Easy Java Simulations [21]. This virtual laboratory presents also some limitations: 1) the view of the simulation is only available in 2D; 2) the model of the robots does not consider physical principles (physical interaction with the environment); and 3) the scenarios can contain only one type of robots. Table 1 shows the main specifications of the robots LEGO NXT, Moway and Khepera IV.

The new platform has been designed also based on previous experiences of the authors with remote laboratories [22], [23]. In particular, the platform developed here is a natural extension to complement the virtual environment developed by the authors for performing mobile robot simulations [24]. The main purpose of this new platform is to

TABLE 1. Main specifications between above mentioned robots.

	LEGO NXT	Moway	Khepera IV
Microcontroller	ARM(48 MHz)	PIC(4 MHz)	ARM(800 MHz)
Internal Mem	256 KB	50 MB	4 GB
RAM	64 KB	368 Bytes	512 MB
Wireless	Bluetooth	RF	WI-FI
Data Packet	16 Bytes	8 Bytes	32 Bytes

perform more interesting and complex control experiments based on the high quality of the sensors and actuators, and the computational capacity of the robots including: 1) position control; 2) trajectory tracking and path following; 3) obstacles avoidance; and 4) multi-agent formation control with obstacles avoidance.

B. EXPERIMENTAL ENVIRONMENT

The two main components of the experimental environment are: a) the robots (see Fig. 1), which are the most important component of the platform; and b) the arena (which is the surface where the experiments occur).

FIGURE 1. Khepera IV robot.

Khepera IV is a wheeled mobile robot that has been designed for indoor pedagogical purposes and it brings numerous features, for example, a colour camera, WI-FI and Bluetooth communications, an accelerometer, a gyroscope, improved odometry and precision, an array of 8 infrared sensors for obstacle detection, 5 ultrasonic sensors for long range object detection, 2 very high quality DC motors, a battery (running time of approximately 5 hours) and a Linux operating system core [14].

In this implementation, the robots are connected to a WI-FI router that allows wireless communication between them and with other components of the platform: the IPS and the Monitor Module. The Linux core provides a well-known standard C/C++ environment for application development. Almost any existing library can be easily ported on the Khepera IV, allowing the development of portable embedded algorithms and applications [14].

The arena is the physical space where the experiments with the robot occur. To build it, two tables are joined and a wooden board is screwed to them. Then, denim fabric is glued to the board. The fabric must be perfectly stretched to allow the robot to move smoothly. In addition, 4 rectangular aluminium tubes are fixed to the edge of the arena to prevent

FIGURE 2. Platform in the laboratory.

robots from leaving the workspace. Two cameras are attached to a square arc of aluminium tubes to get an overhead view of the workspace. Figure 2 shows a view of the laboratory and platform.

To perform position control experiments, the absolute position of the robot is required because the robot needs to "know" its current position during the experiment to perform a control action. In physical indoor environments, it can be difficult to obtain the position because the GPS does not work under such conditions. That is why this kind of platforms needs an IPS to locate the robots. The platform is divided in three main parts: 1) The experimental environment (the robot and the arena); 2) the positioning system; and 3) the Monitor Module to address communications with the robots and user interaction. The following sections describe these three main parts of the platform in detail.

C. INDOOR POSITIONING SENSOR (IPS)

Figure 3 shows the architecture of the platform with all software and hardware components and their interactions. Note that the red line encloses the components of the IPS and the green line encloses the Monitor Module.

The hardware and software components of the IPS are the following: a) A personal computer (PC) with the Ubuntu Linux operating system that executes the software tool related to the IPS: SwisTrack; b) a PlayStation 3 (PS3) USB camera (fixed to the aluminium arc described before in the experimental environment) is connected via USB to the PC. The images of this camera are processed to obtain the position of the robot. The PC also executes the Monitor Module. This is a software component that it is explained in the next subsection.

SwisTrack is an open source software tool developed at the EPFL for tracking robots, animals and objects. SwisTrack computes the absolute position and orientation of the robot by image processing of the PS3 camera and detecting a marker

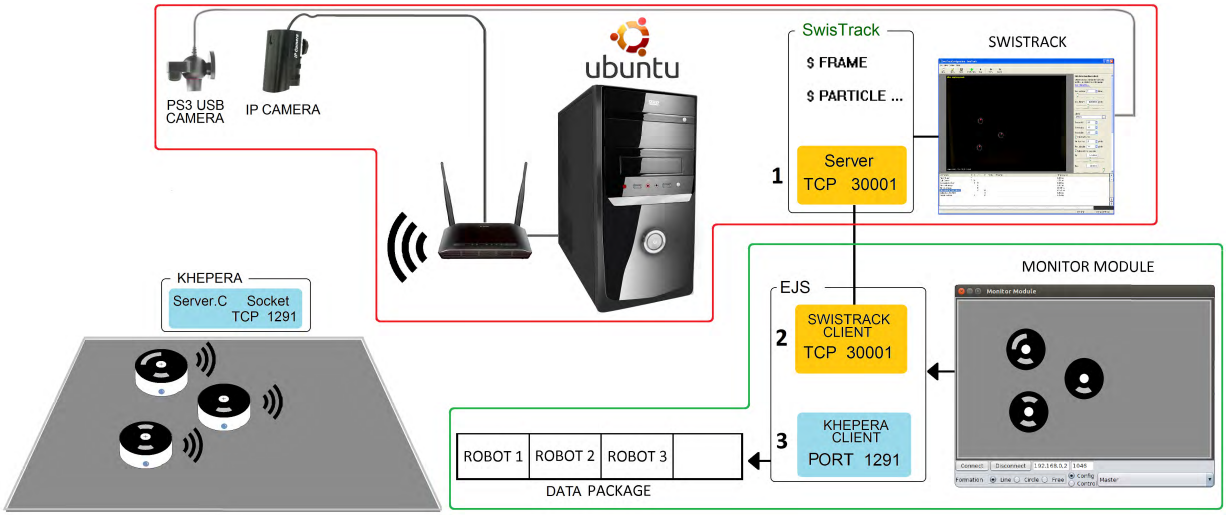

FIGURE 3. Hardware and software components of the platform.

with a circular "barcode" on the top of the robot. Swistrack requires a previous calibration stage, where the coordinates of the image in pixels are converted to coordinates of the real world in centimeters. This calibration is stored in an xml file that Swistrack uses in running time to provide both, coordinates in pixels of the image and coordinates in centimeters of the real world. More details can be found in [17].

The calibration process is considered as previous work, and it is performed by the professor and explained to students before using the platform.

SwisTrack performs steps to process the input images. The steps and their corresponding tasks are the following (see Figure 4):

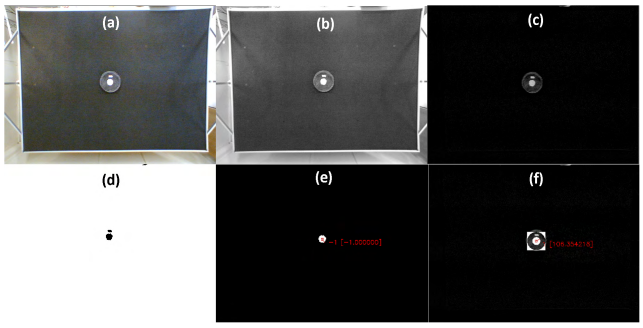

FIGURE 4. Swistrack steps to detect the robot.

a) Input from USB camera: It enables grabbing images from any camera supported by OpenCV library [25].
b) Conversion to Grayscale: It converts the input frame into a grayscale image.
c) Adaptive Background Subtraction: Before starting an experience, Swistrack takes an image of the scenario where the experiments take place. This image is set as background. In this step, this background is subtracted from the image obtained by the camera. This means that all these objects that appear in the background image (obstacles, etc..) are ignored at run-time, and only the new objects added to the scenario are considered.
d) Threshold (grayscale): It intends to separate the objects to track from the background. The background will become white and the objects to track will appear in black.
e) Blob Detection: It detects the blob resulting from a threshold step, and keeps the area, orientation and compactness of the blob.
f) Read Circular Bar-code: It detects the ID and rotation of a marker by reading a circular barcode around previously detected particles. The barcodes are created with MATLAB code provided by the authors of SwisTrack. The position (in pixels) and the uncertainty (in percentage) of the detection of the robot are shown in red. Note that changing light conditions or camera noise can produce a false detection of the robot. In this case, lights installed on the ceiling of the laboratory are sufficient to maintain stable conditions for the detection of the robot.

Once the position of each robot (x_c, y_c, θ) is obtained, then SwisTrack sends it to Monitor Module as a TCP/IP server using the port 30001 by default (labelled as 1 in Fig. 3). To do that, SwisTrack writes the robots positions to the NMEA (National Marine Electronics Association) stream message to send it to other programs. Figure 5 shows an example of these messages for three robots.

Each line starts with $ symbol and ends with the check-sum (two-digit HEX number obtained by xor'ing all bytes between $ and *). The first line of the message represents a marker of the beginning of the data frame (STEP_START). The second line indicates the number of the frame (FRAMENUMBER, 4210). Lines 3 to 5 contain the positions and the orientations of the three detected robots in this case. The values of each line are comma separated and the

```
$STEP_START*0D
$FRAMENUMBER,4210*75
$PARTICLE,5,518,379,6.12,0.899,-0.518*2C
$PARTICLE,7,720,482,1.71,39.99,19.989*08
$PARTICLE,1,285,281,2.35,-43.83,-20.08*08
$STEP_STOP*55
```

FIGURE 5. SwisTrack NMEA message.

order is the following, regarding to line three: 1) PARTICLE to define the line as robot detected; 2) 5 is the identification of the detected robot; 3) 518 is the value of its X coordinate in pixels of the image; 4) 379 is the value of its Y coordinate in pixels of the image; 5) 6.12 is the value of its orientation (theta) in radians; 6) 0.899 is the value of its X coordinate of the real world in centimeters; 7) -0.518 is the value of its Y coordinate of the real world in centimeters; 8) 2C is the previous mentioned check-sum of this line. Finally, the sixth line represents the marker of the end of the data frame (STEP_STOP).

D. MONITOR MODULE

Monitor Module is a software component developed in Easy Java Simulations (EJS) [26] that allows the interaction between students and the robot. The graphical user interface shows the streaming of the above-mentioned IP camera. By clicking on the image the target point (T_p) for the position control of the master robot can be set. Monitor Module is also in charge of two other tasks: a) the communication with SwisTrack via TCP/IP client at port 30001 (labelled as 2 in Fig. 3); and b) the communication with the robots using a wireless router via TCP/IP client through port 1291 (labelled as 3 in Fig. 3).

The function processSwisTrack allows the communication with SwisTrack. It obtains the position of the robot through TCP port 30001 and builds the data packet that will be sent to the robot. The following source code fragment shows the implementation of this function in EJS.

```
1  public void processSwisTrack(String line){
2   if(line.startsWith(''$PARTICLE,'')){
3   String[] datos=line.split('','');//Splitting by comma
4   robotcode=Integer.parseInt(datos[1]);// Robot ID
5   if(datos[5].indexOf(''.'')>0){//Obtaining X
6    xsh=Integer.parseInt(
7      datos[5].substring(0,datos[5].indexOf(''.'')));
8   }else{xsh=Integer.parseInt(datos[5]);}
9   if(datos[6].indexOf(''.'')>0){//Obtaining Y
10   ysh=Integer.parseInt(
11     datos[6].substring(0,datos[6].indexOf(''.'')));
12  }else{ysh=Integer.parseInt(datos[6]);}
13  double thdou=Double.parseDouble(datos[4]);
14  thsh=(int)(thdou*100.0); //Obtaining th in radians
15  ...  //Converting the data into bytes
16  robotPacket[0]=robot_code;//Array (obtained data)
17  ...
18  robotPacket[8]=xtemp;
19  robotPacket[9]=ytemp;
20  }
```

The code processes the NMEA message sent by processSwisTrack. The function checks if the message starts with "PARTICLE", which means that the message contains information about a detected robots (line 2). If so, the data is separated by comma and stored in the array datos (line 3). The identification code (given by the barcode) of the robot is stored in the variable robot_code (line 4). After that, the position and orientation of the robots are converted and stored in *x_sh*, *y_sh* and *th_sh* respectively (lines 5 to 12). This process is repeated for each line of the frame that start with "PARTICLE". With all these data a packet is built and sent to the robots using a TCP/IP communication by broadcast. All robots receive the package with the positions and orientations of all robots present in the arena.

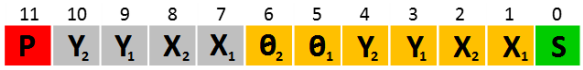

FIGURE 6. Data packet structure.

Figure 6 shows the packet structure, and the sub-frame of each robot which is as follows:

a) Byte 0 (S): Robot ID.
b) Bytes 1 and 2 (XX): 2 bytes corresponding to the X coordinate of the robot position (in centimeters).
c) Bytes 3 and 4 (YY): 2 bytes corresponding to the Y coordinate of the robot position (in centimeters).
d) Bytes 5 and 6 ($\theta\theta$): 2 bytes corresponding to the robot orientation (θ) (in degrees).
e) Bytes 7 and 8 (XX): 2 bytes corresponding to the X coordinate of the target point (in centimeters).
f) Bytes 9 and 10 (YY): 2 bytes corresponding to the Y coordinate of the target point (in centimeters).
g) Byte 11 (P): Marker of the end of the packet.

III. CONTROL EXPERIENCES WITH THE PLATFORM

In this section, the results for some experiments carried out to test the platform are presented and discussed. The experiments are the following: Position Control, Trajectory Tracking, Path Following and Obstacles Avoidance.

A. ROBOT POSITION CONTROL

A simple control example has been implemented to test the platform. This experiment is entitled position control or point stabilization of a differential wheeled mobile robot. The objective of the experiment is to drive the robot from the current position $C(x_c, y_c)$ and orientation of the robot with respect to the plane XY (θ), to the target point $T_p(x_p, y_p)$, as shown in Figure 7. Note that $\theta = 90°$ at point C and $\theta = 0°$ at point T_p.

This problem has been widely studied mainly due to the design of the control law. Under the nonholonomic constraints, this experiment introduces challenging control problems from an academic point of view [27]. To achieve the control objective, the distance (d) and the angle (α) between the points C and T_p are calculated with Equations (1) and (2).

$$d = \sqrt{(y_p - y_c)^2 + (x_p - x_c)^2} \tag{1}$$

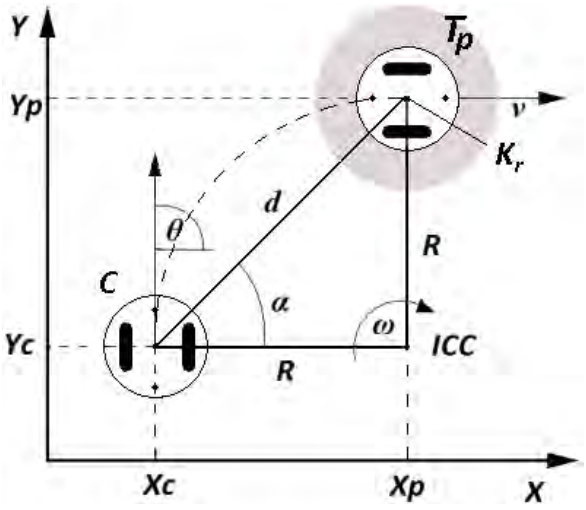

FIGURE 7. Position control problem.

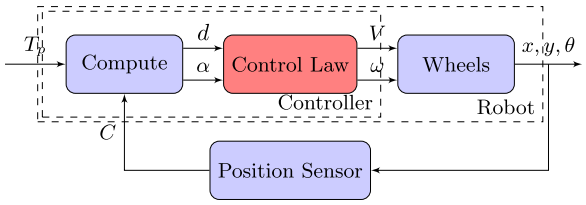

FIGURE 8. Position control block diagram.

$$\alpha = \tan^{-1}\left(\frac{y_p - y_c}{x_p - x_c}\right) \tag{2}$$

Figure 8 shows the control block diagram of this problem. As seen, the Controller is composed of two blocks: 1) block Compare which implements Equations (1) and (2) by using the target point (T_p) as reference and the current position of the robot (C) and 2) Control Law which tries to minimize the orientation error, $\theta_e = \alpha - \theta$, and at the same-time, to reduce the distance to the target point ($d = 0$) by manipulating the control signals (linear velocity (v) and angular velocity (ω) of the robot).

$$v = \begin{cases} v_{max} & if\ |d| > Kr \\ d\left(\dfrac{v_{max}}{K_r}\right) & if\ |d| \leq K_r \end{cases} \tag{3}$$

$$\omega = K_p sin\left(\theta_e(t)\right) + K_i \int \theta_e(t)dt \tag{4}$$

Equations (3) and (4) represent the implementation of the Control Law block based in [28], [29]. Where v_{max} is the maximum linear velocity, K_r is the radius of a docking area (around the target point) and the maximum angular velocity of the robot that will be achieved at $\theta_e = \pm 90°$. Note that this block can be implemented in a different way, the designer only needs to maintain the inputs and the outputs of the block. The following source code segment shows the implementation of the position control of the robot.

This function receives a char array titled buffer, which is the data packet represented in Figure 6. In lines 7 to 10,

```
1  PositionControl(char buffer[]){
2   float Kpos=3.5,V,alpha,Oc,vl,vr,w,Wmax=2.5/2,th,d,time;
3   double L=10.54,dx,dy;
4   int thi,x,y,aux,Vmax=10,Kr=10;
5   ... //Data conversion from bytes array to float
6   th=thi/100.0; //th in radians
7   d=(pow(pow(xp-x,2)+pow(yp-y,2),0.5)); //Target distance
8   alpha=-(float)atan2(yp-y,-(xp-x)); //Angle to target
9   Oc=alpha-th; //Angle error
10  w=Kp*sin(Oe) + Ki*Ie;
11  if (d>Kr){V=Vmax;} //Control law
12  else {V=d*(Vmax/Kr);}
13  vr=(2*V+w*L)/2; //Velocity right wheel
14  vl=(2*V-w*L)/2; //Velocity left wheel
15  right_speed=(float)(vr/(KH4_SPEED_TO_MM_S/10));
16  left_speed=(float)(vl/(KH4_SPEED_TO_MM_S/10));
17 }
```

the code implements Equations 1, 2, 3 and 4 to compute the control law. Finally, the left and right velocities of the differential model are computed (line 13 to 14) and sent to the motors (line 15 to 16).

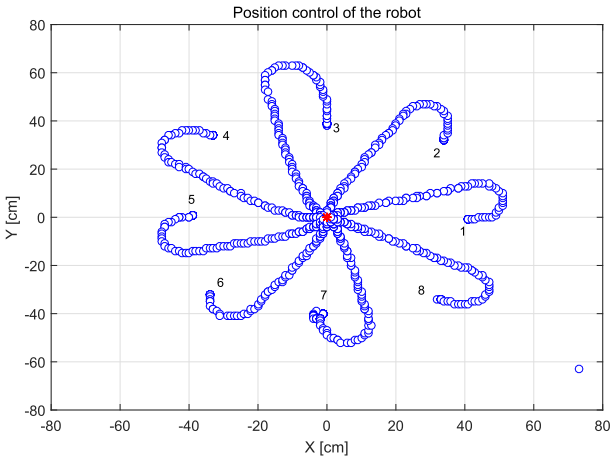

FIGURE 9. Results of the position control experiment.

Figure 9 shows a plot of the control position experiment for eight different initial positions and orientations (this example was taken from [30]). Blue points describe the trajectory of the robot during the experiment. Note that the red point represents the target point at the coordinate (0;0). As can be seen, due to the initial orientation and position of the robot, each experiment describes a different trajectory. Figure 10 shows a sequence of images for this experiment.

B. TRAJECTORY TRACKING AND PATH FOLLOWING EXPERIMENTS

Another two interesting experiments with mobile robots are: a) Path Following, where the robot must follow point by point, a predefined geometrical trajectory that can be dynamically generated; and b) Trajectory tracking, where the robot must follow a trajectory but also considering the time to reach each point of the trajectory [31].

To start with this kind of experiments we show to students the algorithm presented in [32]. This method is a simple approach to track trajectories by assuming that the evolution of the system can be approximated by a linear interpolation in each sampling time.

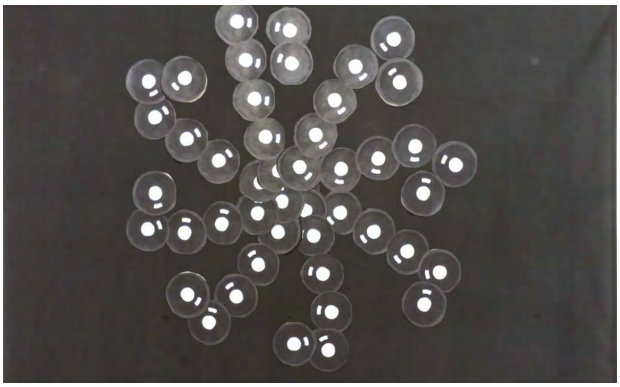

FIGURE 10. Image sequence of the position control experiments.

Each iteration, the robot executes the position control problem (it goes from the ''current position'' to the ''current target'', but taking into account future posture of the robot. To do that, it calculates auxiliary variables ($\triangle x$, $\triangle y$, $\triangle \theta$) [33], using the current posture of the robot (x_n, y_n, θ_n), the current target ($x_{ref(n)}$, $y_{ref(n)}$, $\theta_{ref(n)}$), and the next target ($x_{ref(n+1)}$, $y_{ref(n+1)}$, $\theta_{ref(n+1)}$) as it can be seen in equation 5, where $0 \leq K_x, K_y, K_\theta \leq 1$.

$$\begin{cases} \triangle x = x_{ref(n+1)} - K_x(x_{ref(n)} - x_n) - x_n \\ \triangle y = y_{ref(n+1)} - K_y(y_{ref(n)} - y_n) - y_n \\ \triangle \theta = \theta_{ref(n+1)} - K_\theta(\theta_{ref(n)} - \theta_n) - \theta_n \end{cases} \tag{5}$$

With these values the velocities v_n and ω_n of the robot are calculated as is shown in Eq. 6.

$$\begin{cases} v_n = \dfrac{\theta_{ref(n+1)} - \theta_n}{a^2 + b^2} \left(\dfrac{\triangle x}{T_0} a + \dfrac{\triangle y}{T_0} b \right) \\ \omega_n = \dfrac{\triangle \theta}{T_0} \end{cases} \tag{6}$$

where T_0 is the sampling time, $a = sin(\theta_{ref(n+1)}) - sin(\theta_n)$ and $b = cos(\theta_n) - cos(\theta_{ref(n+1)})$. More details can be found in [33]. Note that by manipulating some of this parameters like T_0, path following experiment can be implemented.

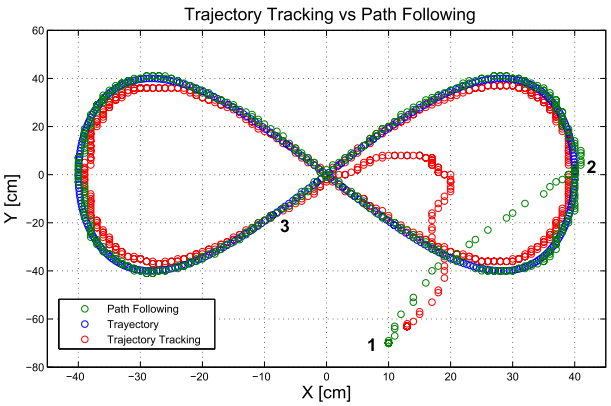

FIGURE 11. Results of the trajectory tracking and path following experiments.

Figure 11 shows the results of these experiments with the Platform. The blue points represent the Lissajous trajectory (dynamically generated), the red points represent the position of the robot for the trajectory tracking experiment, and the green points represent the position of the robot for the path following experiment. Points 1, 2 and 3 have been marked to explain the behavior of each experiment.

In both experiments, the initial position of the robot is point 1, and the trajectory starts at point 2. As it can be seen, in the path following experiment (green points) the robot reaches point 2 in order to follow the desired trajectory from its first point. After that, the robot follows the trajectory with high accuracy. In the case of the trajectory tracking experiment (red points), the robot starts following the trajectory only after point 3. This is because, in this situation, the time to reach each point is considered in the control law as it was said before. Other advanced control algorithms (such as predictive or fuzzy control) for trajectory tracking and path following can be implemented by modifying equation 6.

C. OBSTACLES AVOIDANCE

This subsection presents another example designed to test the platform: Obstacles Avoidance. This kind of experiment is very popular and it can be implemented using different strategies. The experiment consists of a scenario by which the robot must navigate, avoiding the obstacles that it finds in its path. For this example the navigation is based on the previous example of position control of the robot.

In this case, due to the type of proximity sensors of the robot, the Braitenberg algorithm has been selected [34], [35]. This algorithm creates a weighted matrix that converts the sensor inputs into motor speeds. This matrix is a two-dimensional array with the number of columns corresponding to the number of obstacle sensors (8) and the number of rows corresponding to the number of motors (2). The weights of the matrix are determined empirically depending on the location of the sensors in the robot. The 8 sensors of the Khepera IV robot were numbered clockwise beginning with the front sensor [15].

Figure 12 shows a configuration of this experiment with two types of obstacles: walls and cylinders. The obstacles are represented in black and the positions of the robot are represented in blue. The experiment begins with the robot situated at the red star marked as START point.

As can be seen, the robot tries to reach the destination point (marked as GOAL). The behaviour of the robot is based on two states: 1) movement straight ahead with constant speed (when no obstacles are detected); and 2) obstacle avoidance using the Braitenberg algorithm (when obstacles are detected). As can be seen, the robot avoids the walls in an acceptable way. Figure 13 shows a sequence of images for this experiment.

D. MULTI-AGENT FORMATION CONTROL WITH OBSTACLES AVOIDANCE

This experiment is based on [36] and it consists of making a formation in a cooperative and decentralized way (see figure 14). In this case one robot acts as master (labeled as 1)

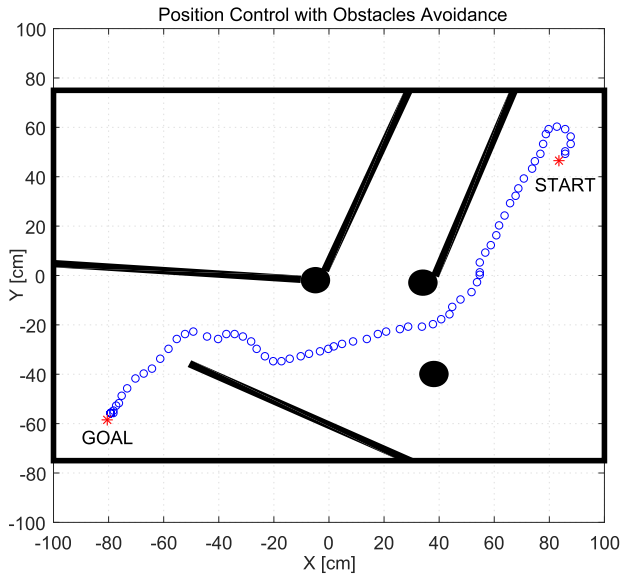

FIGURE 12. Position control with obstacles avoidance experiment.

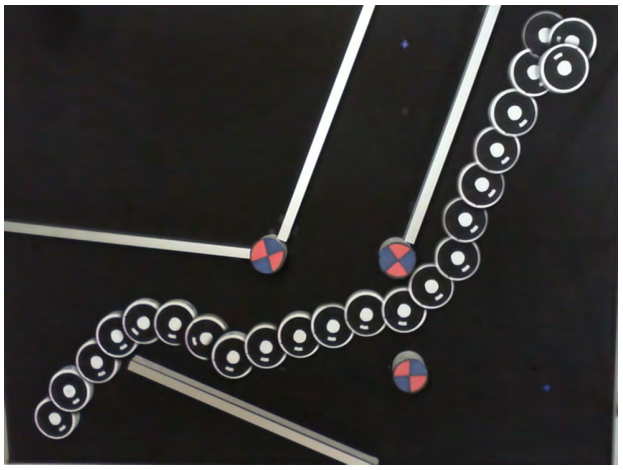

FIGURE 13. Position control with obstacles avoidance experiment.

and the rest as slaves (labeled as 2 and 3). The positions of the slaves robots are controlled as in the previous experiment. To make a formation, slave robots have to reach a position around the master robot. Equation 7 shows how the velocity of the master robot is calculated in function of its own position error (E_{pm}) and the slaves errors in the formation (E_f).

$$v_m(t) = K_p E_{pm}(t) - K_f E_f(t) \tag{7}$$

The values of K_p and K_f are manually adjusted to control the influence of each error in the velocity of the master robot. If $K_f = 0$ the errors of the slaves in the formation are not taken into account, which means that the control is made in a non-cooperative way. Because the master robot does not consider the errors of the slaves. Equation 8 shows how the formation error is calculated.

$$E_f(t) = \sum_{i=1}^{N} E_{pi}(t) \tag{8}$$

For the master robot, the reference is the target point (at the left of the image) and for the slaves the target points are their positions in the formation. They use the position of the master to make a triangular formation around it. Both slaves are situated at 30 centimeters from the master and 135° and −135° behind it, respectively.

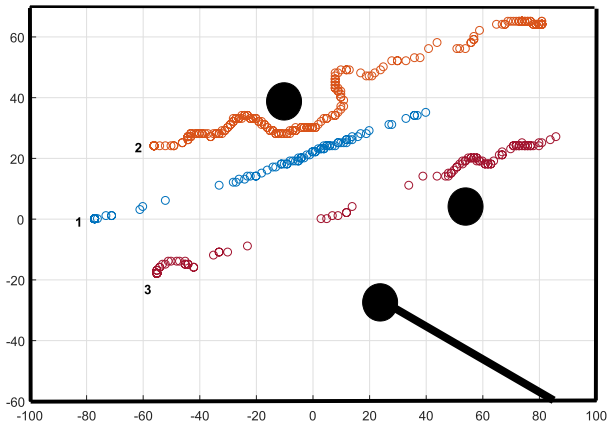

FIGURE 14. Formation control with obstacles avoidance.

Figure 14 shows the results of the implementation of this experiment with obstacles avoidance. In this case, the position of the master robot is represented with blue and robots 2 and 3 with orange and brown colors. Black circles and lines represent the obstacles and walls. The experiment begins at the right side of the image. Once the formation is reached, the master robot begins to move to its target point. The formation is maintained during the movement because the master robot takes into account the positions of the slaves in the formation. During the maneuver, robots avoid the obstacles that appear in their way. Figure 15 shows an images sequence of this experiment.

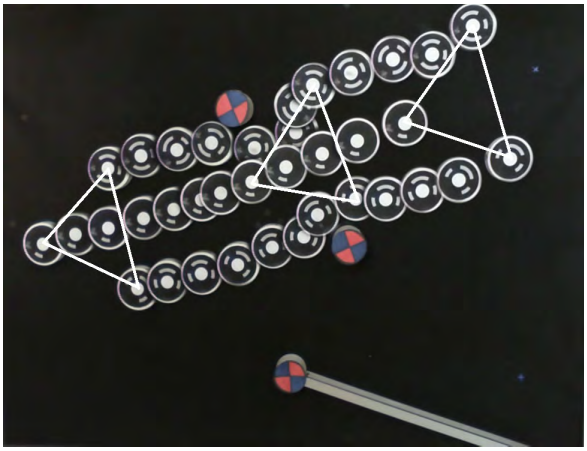

FIGURE 15. Images sequence of the formation control experiment.

E. ADVANCED PROPOSED EXPERIMENTS

The platform is a ready to use tool that allows the implementation of different experiments to study some interesting

problems of mobile robots. These experiments can include sensing and movement tasks for a single robot or cooperation in a multi-robot scenario. Some of the experiments that could be implemented with the developed platform are the following:

- Obstacle avoidance: Using the proximity sensors and their location on the robot, other algorithms of obstacle detection and avoidance can be implemented [37]–[39] and [40].
- Occupancy Grid Mapping: Using the proximity sensors of the robot, an occupancy grid of the environment can be built and stored in memory. This experiment can be implemented using the position control experiment as a basis or with odometry using the encoders of the robot to know its position during the navigation.
- Pursuer-Evader Games: In this type of problem, the robots compete with each other to pursue their own objectives like in [41].
- SLAM (Simultaneous Localization and Mapping): This type of experiment also can be implemented in the platform like in [42].

IV. USING THE PLATFORM TO TEACH CONTROL ENGINEERING

In this section, the methodology for using the platform in the lab is described in detail. Learning experiences of the students with this platform and other similar laboratories are also provided.

A. METHODOLOGY FOR PERFORMING LABORATORY PRACTICES

Laboratory practices have been used for several years in automatic control courses at our University. The educational methodology of current laboratories can be defined as follows:

a) Homework tasks: In this stage, students must do individual study of the theoretical aspects related to the experiments that they will perform in the laboratory. The study is guided by three main elements: 1) documentation of the practice using tree documents to support the study; 2) a simulation of the system; and 3) a bibliography related to the experiments.

b) Laboratory Session: After working at home with the documentation, the bibliography and the simulation, students must attend to the laboratory to perform the practice with the platform.

At the end of the process, students make and submit the report containing the results and conclusions. This report is evaluated by the professor. The results of the laboratory practice and the final test score determine whether students pass the course.

Figure 16 shows the time-line of the process with the Platform. At the beginning of the first week, the professor explains in class the tasks to the students and give them the instructions and the library to carry out the simulation of the system. During the first week, students make the

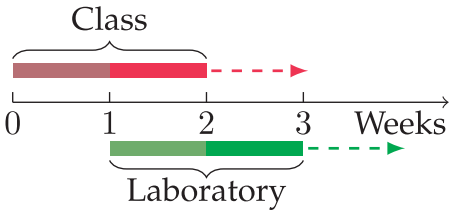

FIGURE 16. Time-line of the process.

home-work tasks and they can consult all the doubts to the professor. At the beginning of the second week, students show to the professor their home-works. Then, students that obtained good results (G1) can start to use the Platform at the laboratory. In contrast, students who did not get good results (G2) have another week to finish the tasks and deliver the laboratory report. At the end of the second week, students of G1 can deliver their results to the professor. If some students of G1 do not finish the laboratory work during that week, they can use the platform during the next week until finish the work. Students of G2 start to use the Platform at the end of the second week. The entire process for both groups must finish at the end of the third week. The teacher can manage the times according to the progress of the students.

B. HOME-WORK TASKS

The individual study of the system is based on the bibliography and the provided material, which consists of three documents:

a) Practice Guide: This document is a guide that contains the theoretical aspects of the laboratory. This document is structured as follows: Introduction; Objective of the practice; Description of the system; Mathematical model of the robot (differential wheeled mobile robot); Experiments (Position control, Formation control and Obstacle avoidance); Tasks to carry out; and References.

b) Tasks Protocol: This document contains detailed steps for performing each experiment of the practice.

c) Applications Interfaces: This document contains a detailed explanation of the interfaces of the software applications involved in the practice.

To perform the experiments in simulation time, students use the library of the robot developed for V-REP software and presented in [15], [24], [43]. Figure 17 shows the library running an experiment of comparison between two obstacles avoidance algorithms (Braitenberg and VFH).

To use this example, the students load a predefined simulation in V-REP (a **.ttt* file) where the robot and some other elements (such as targets and obstacles) are included. This file has everything needed to run the simulation, except the code segment in which the control algorithm is implemented. Thus, the students should complete this part to execute the position control experiment successfully. Note that to ease this task the target point $P(X_p; Y_p)$ and the position of the robot $(X_r; Y_r)$ (the inputs of the algorithm) should be set in

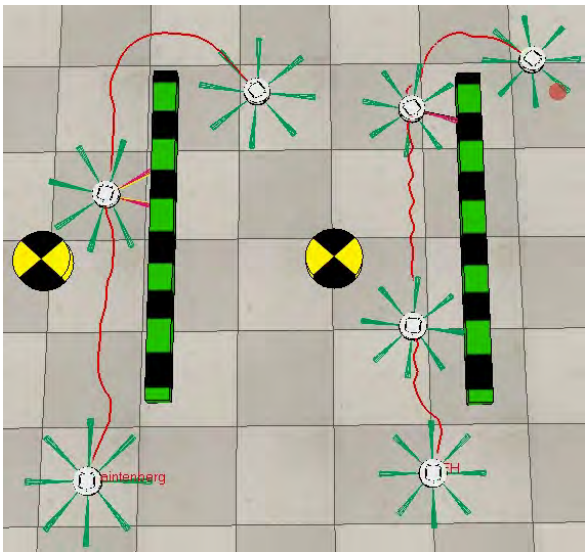

FIGURE 17. Khepera IV library in V-REP. Images sequence of using the library with two different obstacle avoidance algorithms.

the code beforehand to compute the control law. Note also that the algorithm should provide the control law through the linear (v) and angular (ω) velocities, which are used to calculate the differential velocities of each wheel. Finally, note that the students can access to the complete functionality of the simulation (over 500 V-REP API methods) to generate a final report with analyses and plots of all variables from the sensors and actuators of the Khepera IV robot.

C. LABORATORY SESSION

With the previous work finished, the students come to the laboratory to perform the practice. During the practical session in the laboratory the following steps are carried out:

a) Professor checks the home-work tasks of each student (because this is an essential prerequisite to perform the practice at the real laboratory).
b) At the beginning of the practice, the professor explains the technical details of the platform (communications and the Indoor Positioning Sensor). The professor then shows to the students how the robot is programmed (using a basic example for position control).
c) Under the supervision of the professor, students must implement their own algorithm in the robot.
d) Students perform the experiment with the designed controller.
e) Students evaluate the quality of their results.
f) If the results are not satisfactory, students can improve the control algorithm under the supervision of the professor. This process can be repeated until they obtain the expected results, which is why it is very important for students to successfully complete the home-work before facing the real laboratory.
g) If the results are as expected, they collect the data for the final report.

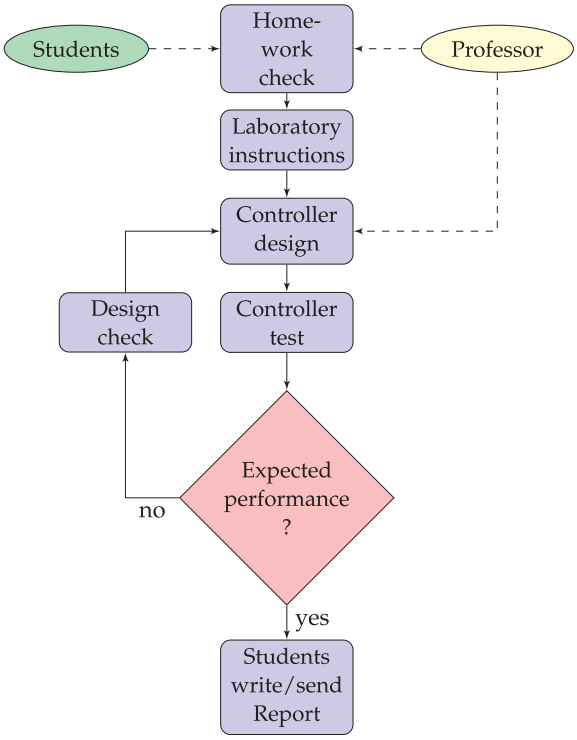

FIGURE 18. Flow diagram of a laboratory session.

h) Students do the report with the results from the simulation and the platform. Students must analyze the results and provide some conclusions.
i) The professor evaluates the student reports.

Figure 18 shows the flow diagram of the process that is performed in a laboratory session.

D. LABORATORY REPORT

The laboratory report must be written by students based on a provided template. The template described in [44], [45] and [46] is organized as follows:

a) Title: Laboratory practice of mobile robots control.
b) Abstract: The abstract presents a synopsis of the Laboratory session: previous work, experiments and results.
c) Introduction: The introduction must present the objectives of the laboratory, the importance of the experiments and the background of the theoretical aspects related to the experiences developed using the simulation and the laboratory session. A hint for the students can be based on the following question: How did you study the problem?
d) Procedures or Methods: This section must describe the steps performed to develop the experiments. Additionally, the practical problems faced with real robots and their solutions. This part of the report can be based on, for example, the following questions: How did you use the platform? How did you proceed with the platform?
e) Results and Discussion: The presentation and the discussion of the results are the most important parts of the report. Students should emphasize the quality of

their results. For example, if the robot reaches the target point with an appropriate speed.

f) Conclusions: In the conclusion, students must do a global analysis of the experiment with the expected and obtained results. Additionally, a description of the reached and unreached objectives based on the results.

g) Appendices: In the appendix, students can present some figures and tables to support their results.

E. EDUCATIONAL EVALUATION OF THE PLATFORM

In order to evaluate the Platform based on the feedback from students, we designed a questionnaire following the same methodology of [47] and [48]. Although we have not conducted a systematic evaluation of the Platform, 3 groups of 5 students were asked for the evaluation. The questions were grouped in three main aspects:

- **Learning Value** summarizes their perceptions of how effectively the Platform helps them learn the relevant contents of mobile robots control.
- **Usability** focuses on students' opinions of the easy-to-use of the Platform to carry out experiments with mobile robots.
- **Technology** assesses students' perceptions of how well the Platform functioned technically and about the robustness of the system.

Regarding the Learning Value, the majority of students positively evaluate the fact of facing first to simulation and after that, work with real robots. Because simulation makes it possible to become familiar with the mobile robots and understand some important concepts (model, behavior, control, etc.). While the experimental environment allows to face real life problems that are not present in simulation time (battery charge level, friction of the wheels with the surface, noises in the images of the camera, communications losses, etc. In general, they see this Platform as an attractive environment to learn mobile robots control in a different way, where they can combine the theory with the practice.

With respect to the Usability, most students agree that the Platform is very easy-to-use. Due to the supporting software are easy to learn and use. The programming and manipulation of the robots are not complex. The instructions are adequate and they are easy to follow. The control laws can be easy tested and improved. Some students pointed out that since they were not familiar with the Ubuntu operating system, it cost them a little more. But that this did not mean a limitation in the usability of the Platform.

Regarding Technology, most students appreciated the fact that they did not have to deal with the communications between robots, noises and battery discharges. Nearly all said they found the hardware and software robust enough to carry out the mobile robot control experiences without additional problems.

V. CONCLUSION

This paper presents a pedagogical platform for performing hands-on experiments with advanced mobile robots. The experimental environment uses the Khepera IV robot and a vision-based positioning sensor to perform multi-agent control problems.

The platform is an easy-to-use environment that allows students to quickly perform several control experiments with mobile robots. The main goal of the platform is to help students of control engineering to understand control theory concepts in an attractive and encouraging environment. There are many technical issues that need to be solved before using the laboratory in order to conceal many irrelevant details (communications, programming, etc).

The experimental set up can be easily updated by users to test different control strategies or to modify the parameters of predefined control algorithms. All data of each experiment are recorded to facilitate the drafting of student reports.

The Indoor Positioning Sensor (IPS) captures an overhead image of the workspace to detect the robots. To this end, the system processes the image to look for a circular barcode, which is located on the top of each Khepera, to get the position and orientation of a robot in the workspace. This information is then sent out to the Khepera, which will compute the control action to reach a desired target.

To test the platform some experiments have been designed and developed, for example, position control, tracking control, path following and multi-agent formation control with obstacle avoidance. The first control problem simply requires moving a robot to a position target. The tracking control problem implies that the robot should follow a predefined trajectory. Similar to this latter problem, the path following requires that the robot follows a route and reach each point of the trajectory at a desired time. The multi-agent with obstacle avoidance problem allows a group of robots to evade obstacles distributed on the arena using its ultrasonic and infrared built-in sensors but keeping the formation control during all the trajectory. All these experiments can be used by the students to develop their own control algorithms.

This paper also presents a guide to using the platform in the laboratory to teach control engineering. The guide is divided in two main parts: previous work developed by students at home, and a laboratory session performed by students at the university under the supervision of the instructor.

Future works involve adding new features and experiments to the platform such as, augmented reality, the use of the on-board camera of the Khepera robot to perform obstacle avoidance control strategies, mapping tasks, pursuer-evader games, and SLAM (Simultaneous Localization and Mapping).

REFERENCES

[1] F. B. V. Benitti, and N. Spolaôr, N. "How have robots supported STEM teaching?" in *Robotics in STEM Education*. Cham, Switzerland: Springer, 2017, pp. 103–129.

[2] A. F. Browne and J. M. Conrad, "A versatile approach for teaching autonomous robot control to multi-disciplinary undergraduate and graduate students," *IEEE Access*, vol. 6, pp. 25060–25065, 2018.

[3] K. Mayerové, and M. Veselovská, "How to teach with LEGO WeDo at primary school," in *Robotics in Education*. Cham, Switzerland: Springer, 2017, pp. 55–62.

[4] R. N. Beyers and L. van der Merwe, "Initiating a pipeline for the computer industry: Using scratch and LEGO robotics," in *Proc. Conf. Inf. Commun. Technol. Soc. (ICTAS)*, Mar. 2017, pp. 1–7.
[5] L. Guyot, N. Heiniger, O. Michel, and F. Rohrer, "Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions," in *Proc. 2nd Int. Conf. Robot. Educ.*, Vienna, Austria, 2011, pp. 1–6.
[6] R. Sadanand, R. P. Joshi, R. G. Chittawadigi, and S. K. Saha, "Virtual experiments for integrated teaching and learning of robot mechanics using roboanalyzer," in *Proc. CAD/CAM, Robot. Factories Future*. New Delhi, India: Springer, 2016, pp. 59–68.
[7] E. G. Guimaraes, E. Cardozo, D. H. Moraes, and P. R. Coelho, "Design and implementation issues for modern remote laboratories," *IEEE Trans. Learn. Technol.*, vol. 4, no. 2, pp. 149–161, Apr./Jun. 2011.
[8] M. S. dos Santos Lopes, I. P. Gomes, R. M. P. Trindade, A. F. da Silva, and A. C. de C. Lima, "Web environment for programming and control of a mobile robot in a remote laboratory," *IEEE Trans. Learn. Technol.*, vol. 10, no. 4, pp. 526–531, Oct./Dec. 2016.
[9] J. E. Auerbach, A. Concordel, P. M. Kornatowski, and D. Floreano, "Inquiry-based learning with robogen: An open-source software and hardware platform for robotics and artificial intelligence," *IEEE Trans. Learn. Technol.*, to be published.
[10] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.
[11] J. Luo and S. Qin, "A fast algorithm of SLAM based on combinatorial interval filters," in *IEEE Access*, vol. 6, pp. 28174–28192, 2018.
[12] C. L. P. Chen, D. Yu, and L. Liu, "Automatic leader-follower persistent formation control for autonomous surface vehicles," *IEEE Access*, vol. 7, pp. 12146–12155, 2018.
[13] R. Socas, S. Dormido, and R. Dormido, "Optimal threshold setting for event-based control strategies," *IEEE Access*, vol. 5, pp. 2880–2893, 2017.
[14] *Khepera IV User Manual*, KTeam Inc., Cham, Switzerland, 2015.
[15] E. Fabregas, G. Farias, E. Peralta, H. Vargas, and S. Dormido, "Teaching control in mobile robotics with V-REP and a Khepera IV library," in *Proc. IEEE Multi-Conf. Syst. Control*, Buenos Aires, Argentina, Sep. 2016, pp. 821–826.
[16] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, Nov. 2013, pp. 1321–1326.
[17] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "SwisTrack—A flexible open source tracking software for multi-agent systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 4004–4010.
[18] E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, and J. Sánchez, S. D. Bencomo, "Platform for teaching mobile robotics," *J. Intell. Robot. Syst.*, vol. 81, no. 1, pp. 131–143, 2016.
[19] M. Guinaldo *et al.*, "A mobile robots experimental environment with event-based wireless communication," *Sensors*, vol. 13, no. 7, pp. 9396–9413, 2013.
[20] E. Fabregas, G. Farias, S. Dormido-Canto, and S. Dormido, "RFCSIM simulador interactivo de robótica móvil para control de formación con evitación de obstáculos," in *Proc. 16th Congreso Latinoamericano de Control Automático*, Cancún, Mexico, 2014, pp. 1392–1397.
[21] W. Christian and F. Esquembre, "Modeling physics with easy Java simulations," *Phys. Teacher*, vol. 45, no. 8, pp. 475–480, 2007.
[22] J. Sáenz, J. Chacón, L. De La Torre, A. Visioli, and S. Dormido, "Open and low-cost virtual and remote labs on control engineering," *IEEE Access*, vol. 3, pp. 805–814, 2015.
[23] L. de la Torre, M. Guinaldo, R. Heradio, and S. Dormido, "The ball and beam system: A case study of virtual and remote lab enhancement with moodle," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 934–945, Aug. 2015.
[24] E. Peralta, E. Fabregas, G. Farias, H. Vargas, and S. Dormido, "Development of a *Khepera* IV library for the *V-REP* simulator," in *Proc. 11th IFAC Symp. Adv. Control Educ.*, Bratislava, Slovakia, Jun. 2016, pp. 81–86.
[25] N. Correll, G. Sempo, Y. L. De Meneses, J. Halloy, J. L. Deneubourg, and A. Martinoli, "SwisTrack: A tracking tool for multi-unit robotic and biological systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 2006, pp. 2185–2191.
[26] F. Esquembre, "Using easy Java simulations to create scientific simulations in Java," in *Proc. IEEE Region 8 EUROCON Comput. Tool.*, Ljubljana, Slovenia, Sep. 2003, pp. 20–23.
[27] F. Kühne, W. F. Lages, and J. M. G. da Silva, Jr., "Point stabilization of mobile robots with nonlinear model predictive control," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Niagara Falls, ON, Canada, vol. 3, Jul. 2005, pp. 1163–1168.
[28] D. Galán *et al.*, "Online virtual control laboratory of mobile robots," in *Proc. 3rd IFAC Conf. Adv. Proportional-Integral-Derivative Control (PID)*, Ghent, Belgium, 2018, pp. 316–321.
[29] V. J. G. Villela, R. Parkin, M. L. Parra, and J. M. D. González, and M. J. G. Liho, "A wheeled mobile robot with obstacle avoidance capability," *Tecnol. Desarrollo*, vol. 1, no. 5, pp. 159–166, 2004.
[30] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2011.
[31] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Cincinnati, OH, USA, vol. 1, May 1990, pp. 384–389.
[32] G. Scaglia, V. Mut, A. Rosales, O. Quintero, "Tracking control of a mobile robot using linear interpolation," in *Proc. 3rd Int. Conf. Integr. Modeling Anal. Appl. Control Autom. (IMAACA)*, Buenos Aires, Argentina, Feb. 2007, pp. 11–15.
[33] G. Scaglia, A. Rosales, L. Quintero, V. Mut, and R. Agarwal, "A linear-interpolation-based controller design for trajectory tracking of mobile robots," *Control Eng. Pract.*, vol. 18, no. 3, pp. 318–329, 2010.
[34] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA, USA: MIT Press, 1986.
[35] X. Yang, R. V. Patel, and M. Moallem, "A fuzzy–braitenberg navigation strategy for differential drive mobile robots," *J. Intell. Robot. Syst.*, vol. 47, no. 2, pp. 101–124, 2006.
[36] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Trans. Robot. Autom.*, vol. 19, no. 6, pp. 933–941, Dec. 2003.
[37] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. Auto. Robot Vehicles*. Cham, Switzerland: Springer, 1986, pp. 396–404.
[38] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991.
[39] I. Ulrich and J. Borenstein, "VFH$^+$: Reliable obstacle avoidance for fast mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Leuven, Belgium, vol. 2, May 1998, pp. 1572–1577.
[40] I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, vol. 3, Apr. 2000, pp. 2505–2511.
[41] M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino, "A remote lab for experiments with a team of mobile robots," *Sensors*, vol. 14, no. 9, pp. 16486–16507, 2014.
[42] A. Marjovi, J. G. Nunes, L. Marques, and A. de Almeida, "Multi-robot fire searching in unknown environment," in *Field and Service Robotics*. Berlin, Germany: Springer, 2010, pp. 341–351.
[43] G. Farias, E. Fabregas, E. Peralta, E. Torres, and S. Dormido, "A *Khepera* IV library for robotic control education using *V-REP*," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9150–9155, 2017.
[44] G. Artus, S. Bienz, K. Venkatesan, and J. Helbing, "Handbook on writing laboratory reports," Dept. Chem., Univ. Zürich, Zürich, Switzerland, 2016.
[45] C. Moskovitz and D. Kellogg, "Inquiry-based writing in the laboratory course," *Sci.*, vol. 332, no. 6032, pp. 919–920, 2011.
[46] K. Walker, "Integrating writing instruction into engineering courses: A writing center model," *J. Eng. Educ.*, vol. 89, no. 3, pp. 369–375, 2000.
[47] R. Dormido *et al.*, "Development of a Web-based control laboratory for automation technicians: The three-tank system," *IEEE Trans. Educ.*, vol. 51, no. 1, pp. 35–44, Feb. 2008.
[48] E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, and F. Esquembre, "Developing a remote laboratory for engineering education," *Comput. Educ.*, vol. 57, no. 2, pp. 1686–1697, 2011.

GONZALO FARIAS received the B.S. degree in computer science from the University of La Frontera, Temuco, Chile, in 2001, the Ph.D. degree in automatic control from the Universidad Nacional de Educación a Distancia (UNED), in 2010, and the Ph.D. degree in computer science from the Complutense University of Madrid, Spain, in 2013. Since 2012, he has been a Professor with the Electrical Engineering School, Pontificia Universidad Católica de Valparaíso (PUCV). His current research interests include machine learning, the simulation and control of dynamic systems, and engineering education.

ERNESTO FABREGAS received the B.S. degree in automation control and the M.S. degree in digital systems from the Polytechnic University Jose Antonio Echeverría (CUJAE), Havana, Cuba, in 2004 and 2008, respectively, and the Ph.D. degree in computer science from the Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain, in 2013. Since 2015, he has been a Postdoctoral Fellow of UNED. His current research interests include the simulation and control of multi-agent systems, machine learning, mobile robot control, remote laboratories, and engineering education.

EMMANUEL PERALTA received the B.S. degree in electrical engineering from the Pontificia Universidad Católica de Valparaíso (PUCV), Chile, in 2016, where he is currently pursuing the M.S. degree in sciences of engineering. His current research interests include virtual and remote laboratories, mobile robots' control, and machine learning.

HÉCTOR VARGAS received the B.S. degree in electrical engineering from the University of La Frontera, Temuco, Chile, in 2001, and the Ph.D. degree in computer science from UNED, Madrid, Spain, in 2010. Since 2010, he has been a Professor with the Electrical Engineering School, Pontificia Universidad Católica de Valparaíso (PUCV). His current research interests include the simulation and control of dynamic systems, multi-agent systems, and engineering education.

SEBASTIÁN DORMIDO-CANTO received the M.S. degree in electronics engineering from the Universidad Pontificia de Comillas (ICAI), Madrid, Spain, in 1994, and the Ph.D. degree in physics from the Universidad Nacional de Educación a Distancia (UNED), Madrid, in 2001. Since 1994, he has been with the Department of Computer Science and Automatic Control, UNED, where he is currently a Full Professor of control engineering. His research and teaching activities include automatic control, machine learning, and parallel processing.

SEBASTIÁN DORMIDO received the B.S. degree in physics from Complutense University Madrid (UCM), Madrid, Spain, in 1968, the Ph.D. degree in sciences from Basque Country University, Bilbao, Spain, in 1971, and the Ph.D. degree from the Universidad de Huelva and the Ph.D. degree from the Universidad de Almería. He was a Professor of control engineering with the National University of Distance Education, Madrid, in 1981. From 2001 to 2006, he was the President of the Spanish Association of Automatic Control, CEAIFAC. He has authored or coauthored over 250 technical papers in international journals. He has supervised over 35 Ph.D. theses. His scientific activities include the computer control of industrial processes, model-based predictive control, hybrid control, and Web-based laboratories for distance education. He has received the National Automatic Control Prize from the IFAC Spanish Automatic Control Committee.