3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)

Proceedings of Engineering & Technology (PET)
pp. 100-106
Copyright IPCO-2016

Design and control of Two-Wheeled Inverted Pendulum Mobile Robot

Sabri-M Ben Mansour
LTSIRS Department
National Engineering School of Tunis
Tunis, Tunisia
sabri.benmansour@gmail.com

Jawhar Ghommam
CEM-Laboratory
National Engineering School of Sfax
Sfax, Tunisia

Saber-M Naceur
LTSIRS Department
National Engineering School of Tunis
Tunis, Tunisia

Abstract: **This paper addresses design and path following control problem of a nonholonomic Two-Wheeled Inverted Pendulum Mobile Robot. We propose control architecture based on two control layers. A speed inner loop control scheme is first designed based on state feedback technique to ensure stability of the inverted structure of the robot. A second outer loop control scheme is proposed to help the robot navigate along a desired path formed by a set of way points. It is designed inspiring the model predictive control technique. The elements of the predictive control, which are the cost function, controls and constraints, must be defined and specified: the use of different trajectories group in the control can adapt the behavior of the robot to different displacement phases.**

Keywords: Mobile robot; navigation; stability; Predictive control

I. INTRODUCTION

The problem of nonholonomic systems control has attracted numerous investigations in the past. A thorough studied case, with great practical significance, is the wheeled mobile robot with a kinematic model similar to a unicycle [1] [2]. The differentially driven mobile robots that are very common in practical applications also have the same kinematic model. Although many researchers coped with the more difficult problem of stabilizing dynamic models for different types of mobile robots [3] [4] [5] [6], the basic limitations of mobile robot control still come from their kinematic model. Kinematic control laws are also very important from the practical point of view, since the wheel-velocity control is often implemented locally on simple micro-controller based hardware.

Traditionally, the problem of mobile robot control has been approached by stabilization point or by redefining the problem as a tracking control one. There are also some approaches that tackle both problems simultaneously. We believe that the tracking control approach is somewhat more appropriate, since the nonholonomic constraints and other control goals (obstacle avoidance, minimum travel time, and minimum fuel consumption) are implicitly included in the path-planning procedure [7] [8].

We were looking for to develop certain tools, as the paths optimization methods, and adapt them to our problem navigation. The aim was to get an under constraints optimization algorithm which can be implemented in a real-time application.

The navigation method that we propose follows several steps: to enable the robot to join the path provided by the modeling environment, a catch curve is generated with the authors approach in [9]. Based on the knowledge of the model of the robot and its constraints, feasible trajectories can be generated on a given time horizon. A constraint optimization algorithm will then determine the closest feasible path by the robot to the reference trajectory under constraints that we have developed. Finally, the generated control law is provided to the robot actuators, and applied according to a chosen sampling time before reconsidering all the navigation process reinitiating.

The problem statement is given in Section II. The kinematic model and the Control design Approach are described in Section III. The speed and predictive control design are developed in Section IV. In Section V, implementation is discussed. The conclusions are stated in Section VI.

II. PROBLEM STATEMENT

Assume a two-wheeled, differentially driven, mobile robot like the one depicted in Fig. 1, where (x, y) is the wheel-axis-centre position and θ is the robot orientation. The kinematic motion equations of such a mobile robot are equivalent to those of an unicycle. Robots with such architecture have a nonholonomic constraint of the form:

$$\dot{\gamma}\,.\cos\theta(t)\ -\dot{\varkappa}.\sin\theta(t)=0 \qquad (1)$$

ISSN: 2356-5608

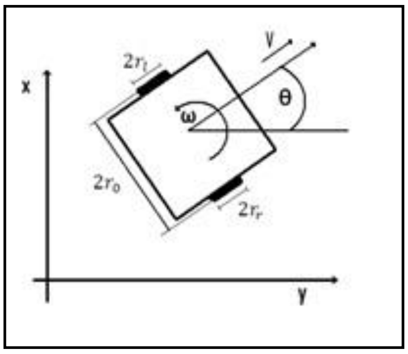

Fig. 1. Mobile inverted pendulum

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}\frac{r_r}{2}\cos(\theta) & \frac{r_l}{2}\cos(\theta)\\ \frac{r_r}{2}\sin(\theta) & \frac{r_l}{2}\sin(\theta)\\ \frac{r_r}{2r_0} & \frac{r_r}{2r_0}\end{pmatrix} . \begin{pmatrix}\dot{\theta}_r\\ \dot{\theta}_l\end{pmatrix} \quad (2)$$

Where $\dot{\theta}_r$ (t) and $\dot{\theta}_l$ (t) are the angular velocity of the right wheel and the angular velocity of the left wheel.

III. THE KINEMATIC MODEL AND THE CONTROL DESIGN

It is possible to model the system described above by the Newton-Euler method.

A. *TWIPMR modeling*

Modeling assumptions:

- The motor correctly filter the PWM.
- The inertia of the motor itself is neglected compared to the inertia of the wheel.
- Motor inductance is neglected.
- Both motors have identical parameters.

1) Mechanical equation of the motor: Consider the general characteristics of the engine we find:

$$U_i(t) = R_i . i_i(t) \quad (3)$$
$$\ddot{\theta}_i(t).I_i = \sum_{i=1}^{n} M_i^j\ (t) \quad (4)$$
$$\sum_{i=1}^{n} M_i^j\ (t) = M_i^1(t) + M_i^2(t) + M_i^3(t) \quad (5)$$
$$M_i^1(t) = k_i.n_i.i_i(t) \quad (6)$$
$$M_i^2(t) = -F_i(t).r_i \quad (7)$$
$$M_i^3(t) = -f_i.\dot{\theta}_i(t) \quad (8)$$

2) Motion equations of the robo: The moment of inertia is on the axis center of the wheels

$$I_0\ .\ddot{\theta}_0(t) = r_0.(F_r(t) - F_l(t)) \quad (9)$$
$$m_0\ .\ddot{x}(t) = \sum_{n=0}^{N} F = F_r(t) - F_l(t) \quad (10)$$
$$\ddot{\theta}_0\ (t) = \frac{\ddot{\theta}_r\ (t).r_r - \ddot{\theta}_l\ (t).r_l}{2.r_0} \quad (11)$$
$$\ddot{x}_0\ (t) = \frac{\ddot{\theta}_r\ (t).r_r + \ddot{\theta}_l\ (t).r_l}{2} \quad (12)$$

The state model can then be discretized to get the form below with system sampling period chosen to 10 ms.

$$X(k+1) = \Phi.\ X(k) + \Gamma\ .\ U\ (k)$$
$$Y(k) = C\ .\ X(k) + D.\ U(k) \quad (13)$$

Φ , Γ, C, D: matrices giving by Matlab calculation

B. *Control design*

In the literature, the control law for the tracking path is based on the assumption that the robot speed limits are directly transmitted [10] [11]. This hypothesis corresponds to a control in perfect speed.

This separation between low-level (robot speed control) and High-level (tracking path) in Fig. 2 appears to be a good solution for many reasons:

- Appropriate segregation of duties for readability
- Best potential for reusability
- Debugging easier when implementing

IV. THE SPEED AND TRAJECTORY CONTROL

A. *Control speed*

Based on the discrete state model, it is possible to synthesize a controller state. The choice fell on the optimal control [12] that responds to the needs perfectly as shown in bloc diagram Fig 3.

The cost function to be minimized is:

$$J = \frac{1}{2}\sum_{k=0}^{N} [x^T(k)Q_1x(k) + u^T(k)Q_2u(k)] \quad (14)$$

We must therefore choose the matrices Q1 and Q2. They are selected positive diagonals. The weight of the position is zero. The weight of the speeds is much higher than the voltage of the engine.

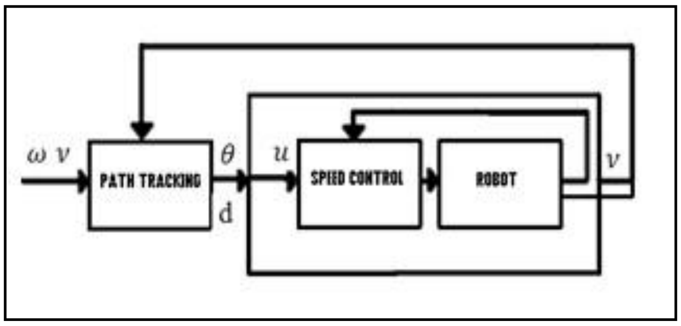

Fig. 2. The two controllers in cascade

Matrices:

$$Q_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 100'000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100'000 \end{pmatrix} \quad (15)$$

$$Q_2 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \quad (16)$$

The dlqr control Matlab gives the optimum gain. Thus, the voltage applied to closed loop is:

$$U(k) = K_G . X(k) \quad (17)$$

Where K_G is a Gain obtained by Matlab calculation

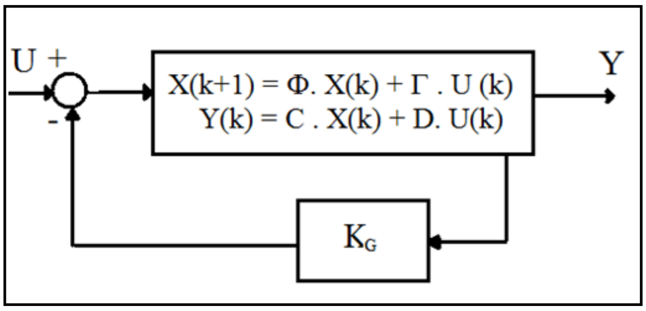

Fig. 3. Block diagram of the control speed

B. Path Tracking

From the information provided by the planer, a remedial curve, connecting the current position of the robot and the desired position can be determined. This curve is a purely geometric, which is not related to time, but that can take into account certain geometric constraints robot (size, turning radius). Angular velocity ω = θ of the robot is used as a control variable as shown in fig. 4.

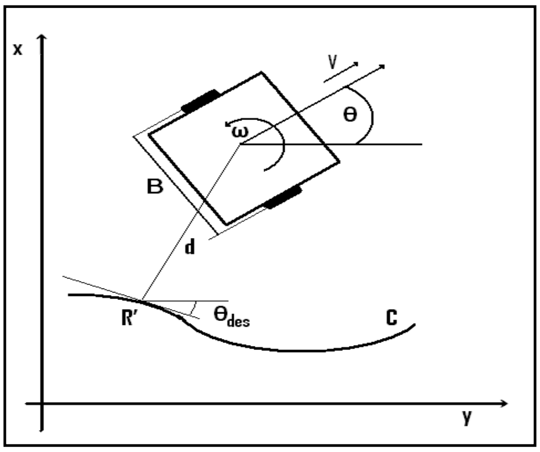

Fig. 4. Path Followed by robot

The purpose of the proposed control law is to reduce both the error in distance and orientation:

$$\theta_e = \theta - \theta_{des} \quad (18)$$

It is given in its simplest form by:

$$\omega = \upsilon.(p(sR') - k_1.d - k_2.\theta_e) \quad (19)$$

Where sR' is the curvilinear abscissa of the point R_0. k_1 and k_2 are two positive constants. Constant values for limiting the oscillations can be obtained in the case of a straight path:

$$k_1 = \xi^2 \\ k_2 = 2\zeta\xi \quad (20)$$

C. Our approach of the Predictive Control

1) The cost function: This cost function calculates the square error between the reference trajectory and the robot path, by weighting differently the various components of the state of the robot and also by weighting differently the terminal error and the tracking error of the course [13] [14] [15].

$$J(k) = \sum_{n=0}^{N} [\hat{z}(k+n) - r(k+n)]^T Q[\hat{z}(k+n) - r(k+n)] + \lambda\omega^2(k+n) \quad (21)$$

2) The control functions and the generated paths: To have good possibilities to avoid obstacles, we used linear type control functions u (v, ξ) in Figs. 5 (a)-(b). The parameters p1 and p3 denote the speed and turning respectively to achieve in $T_0 + \frac{T_P}{2}$ and p2, p4 the parameters of the speed and the turning to reach by $T_0 + T_P$. TP is the horizon of predictions.

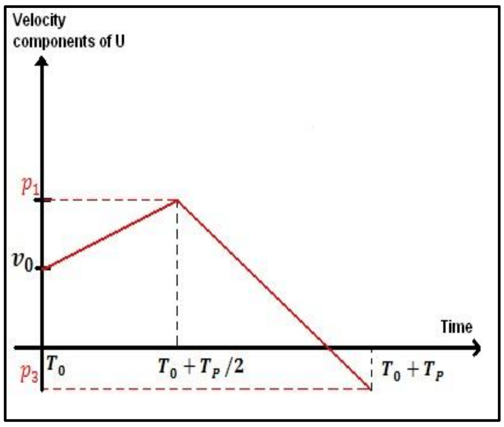

Fig. 5 (a).Input functions for the speed

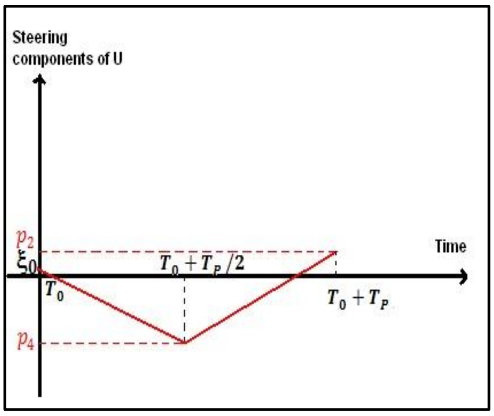

Fig. 5 (b).Input functions for the steering

3) Constraints: These parameters must meet a number of constraints with respect to maximum and minimum speed **($\boldsymbol{\nu}_{\mathbf{min}}$, $\boldsymbol{\nu}_{\mathbf{max}}$)** and steering **($\boldsymbol{\xi}_{\mathbf{min}}$, $\boldsymbol{\xi}_{\mathbf{max}}$):**

$$\begin{aligned} \nu_{\min} &\le p_1 \le \nu_{\max} \\ \nu_{\min} &\le p_3 \le \nu_{\max} \\ \xi_{\min} &\le p_2 \le \xi_{\max} \\ \xi_{\min} &\le p_4 \le \xi_{\max} \end{aligned} \quad (22)$$

From this group of control functions, a group of trajectories is generated as shown in fig. 6. By simply varying each parameter values, we obtain a wide variety of movement possibilities. The Predictive control requires a linear kinematic model giving in [16]. It is based on the following model, consisting for monitoring a straight line:

$$\dot{s} = \vartheta.\cos(\theta - \psi) \quad (23)$$
$$\dot{d} = \vartheta.\sin(\theta - \psi) \quad (24)$$
$$\dot{\theta} = \omega \quad (25)$$

Where **s** is the distance to the next intersection. This model can be linearized around the operating point ($\mathbf{d} = \mathbf{0}$,$\boldsymbol{\theta} - \boldsymbol{\psi} = \boldsymbol{\theta}_{\mathbf{e}} = \mathbf{0}$), this gives:

$$\begin{aligned} \dot{\mathrm{d}} &= \upsilon.\theta_{\mathrm{e}} \\ \dot{\theta}_{\mathrm{e}} &= \omega \end{aligned} \quad (26)$$

And can be discretized with a sampling period h:

$$d(k+1) = d(k) + h.\upsilon.\left[\theta(k) - \psi(k) + \frac{h}{2}\right]$$
$$\theta(k+1) = \theta(k) + h.\omega(k) \quad (27)$$

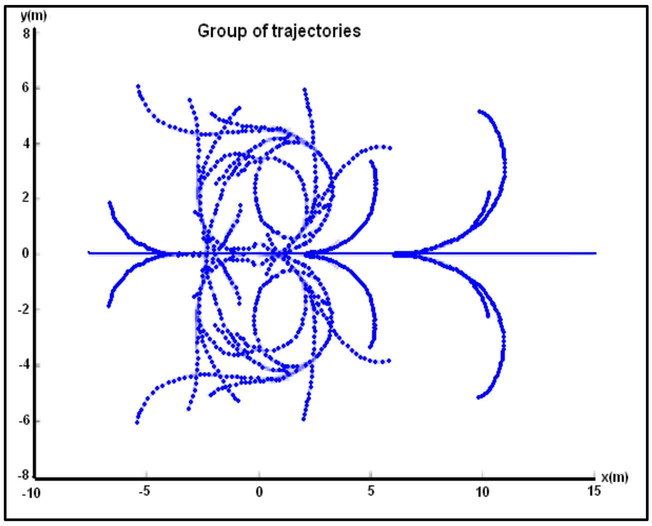

Fig. 6. Setting the parameters of the predictive control

The equations can be writing in state spaces representation:

$$\begin{pmatrix} d(k+1) \\ \theta(k+1) \end{pmatrix} = \begin{pmatrix} 1 \mathrm{h}\upsilon \\ 01 \end{pmatrix} \begin{pmatrix} d(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2}\upsilon \\ h \end{pmatrix} \omega(k) + \begin{pmatrix} 0 & -\mathrm{h}\upsilon \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ \psi(k) \end{pmatrix}$$

(28)

Or in compact form it is writing:

$$z(k+1) = B.z(k) + B_\omega.\omega(k) + B_r.r(k) \quad (29)$$

The criterion chosen to minimize the horizon is as following:

$$J(k) = \lambda\omega^2(k+n) + \sum_{n=0}^{N}[\hat{z}(k+n) - r(k+n)]^T Q[\hat{z}(k+n) - r(k+n)]$$

(30)

Where $\hat{z}$ is the predicted output, Q is the Weighting matrix, λ is a scalar weight and N is the horizon. λ : A scalar weight Using (29) in (30), we have the following:

$$J(k) = \left[\hat{Z}(k) - R(k)\right]^T IQ\left[\hat{Z}(k) - R(k)\right] + \lambda\Omega^T(k)\Omega(k)$$

(31)

Where

$$\hat{Z}(k) = \begin{pmatrix} \hat{z}(k\backslash k) \\ \vdots \\ \hat{z}(k+N\backslash k) \end{pmatrix} = Fz(k) + G_\omega\Omega(k) + G_r r(k)$$

(32)

$$\Omega(k) = \left(\omega(k) \cdots \omega(k+N)\right)^T \quad (33)$$

$$R(k) = \left(r(k) \cdots r(k+N)\right)^T$$

$$= \begin{pmatrix} 0 \\ \Psi(k) \end{pmatrix}^T = \begin{pmatrix} 0 & \cdots 0 \\ \psi(k) \cdots \psi(k+N) \end{pmatrix}^T \quad (34)$$

4) The predictive algorithm: The goal is to find the control sequence ω(k); $1 \leq K \leq n$ that minimizes J (k).

$$\Omega(k) = -L_z z(k) - L_r R(k) \quad (35)$$

Where

$$L_z = \left(\lambda + G_\omega^T I_Q G_\omega\right)^{-1} G_\omega^T I_Q G_\omega F$$
$$L_r = \left(\lambda + G_\omega^T I_Q G_\omega\right)^{-1} G_\omega^T I_Q (G_r - 1) \quad (36)$$

In fact only the next order ω(k) is applied to the system. Therefore only the first line of L_z and the two first line of L_r. That gives the following:

$$\omega(k) = -k_\theta \theta(k) - k_d d(k) - K_\Psi \Psi(k) \quad (37)$$

Where k_d and k_θ are scalar and vector K_ψ as $[1 \times (N+1)]$ ψ multiplying the reference vector ψ(k)
In the case of a turn (sequence of two lines), the vector reference has the following form:

$$\psi(k) = ((\psi_1 \cdots \psi_1, \psi_2 \cdots \psi_2)^T \quad (38)$$

V. Implementation and Robot Construction

A. *Architecture*

A Raspberry Pi module will contain data processing. An application will play the role of a planner. Indeed it's the path provider, at first it will provide several consecutive points that the robot must cross to reach its final goal. The architecture is shown in Fig 7.

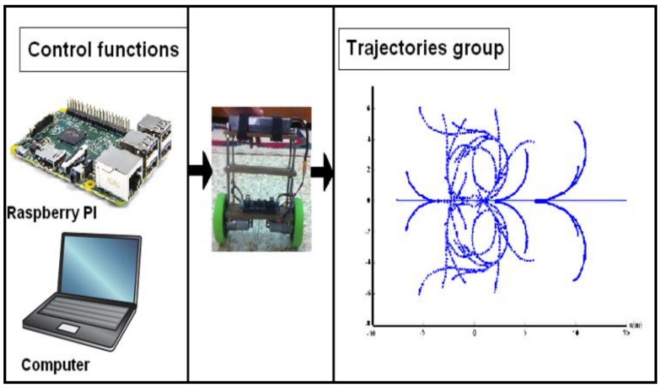

Fig. 7. System Architecture

The synthesis of the two regulators in cascade was made and discussed. We must now put it in common and do all the required adaptations. The block diagram in Fig. 8. shows the general principle of the regulation.

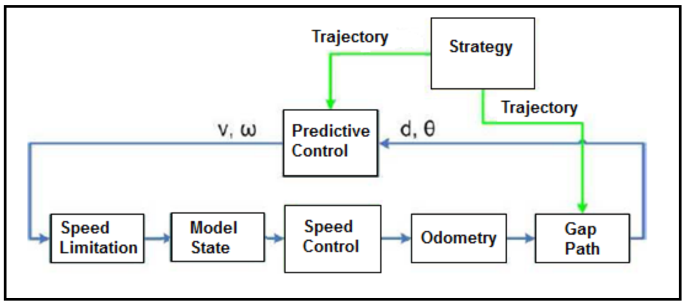

Fig. 8. General functional regulation

The block "strategy" is the artificial intelligence of the robot. It dictates the way and the desired velocity. A simulation of this control principle has been made on the software Matlab Simulink simulation. This optimizes some parameters to save time on the actual implementation.
The best results of simulations are shown in Fig 9. We see the reference path dotted line and how far the robot follows the line. It is noted that even with an initial position outside path (the angle is also set to a divergent path), the robot hunt although the way.

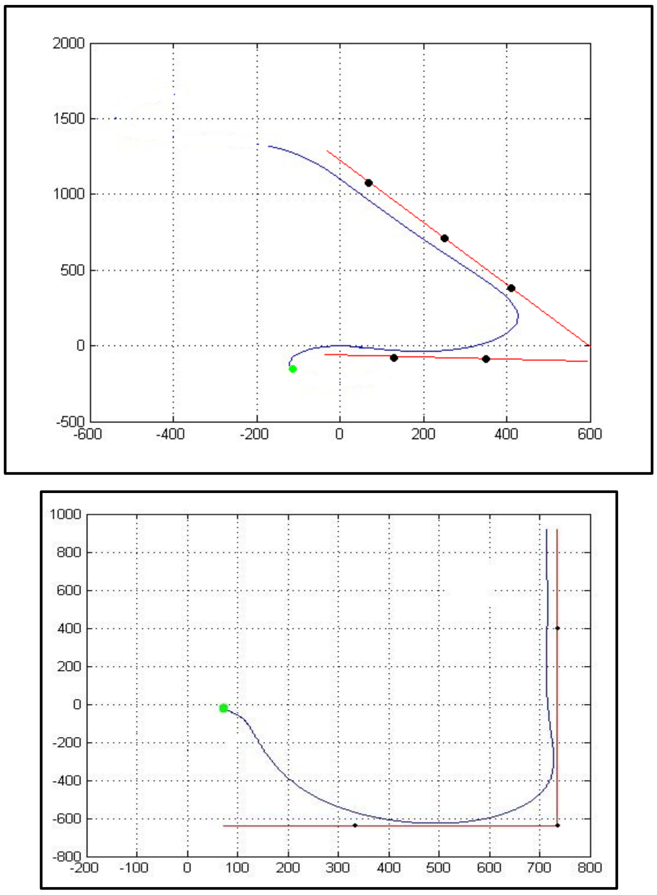

Fig. 9 - Simulation Results

The MEMS (microelectromechanical systems) chip of the 6-axis MPU-6050-gyroscope-accelerometer is very accurate with an analog-digital conversion of 16 bits simultaneously on each connection and an I2C interface. The reading of this sensor measurement is easy.

The sensor contains a FIFO register of 1024 bytes that the Arduino microcontroller can read, being informed by an interrupt signal. The module operates as a slave on the I2C bus with respect to the Arduino (SDA pins, SLC) but can also control another downstream device with AUX and AUX-DA-CL. Its consumption is low with 6 activated sensors.

The sensor has a Digital Motion Processor able to do quick calculations directly on the chip from the raw sensor measurements but it is very slow. It is therefore easier to process raw measurements on the Arduino board. The assembled robot is shown in Fig. 11.

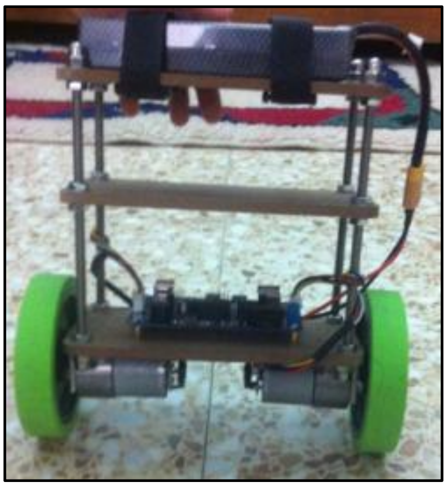

Fig. 11. Assembled parts of the Robot

Here are the experimental results on the Robot in Fig 12.

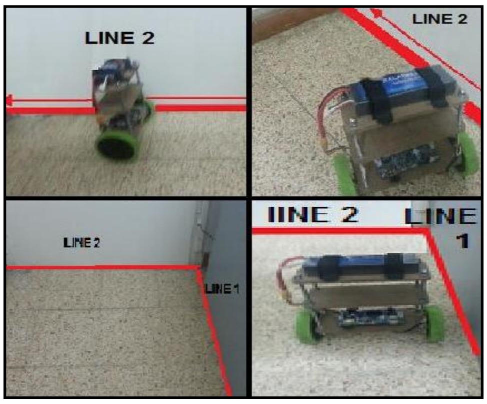

Fig. 12. The robot following the path and responding to constraints

VI. Conclusion

The purpose of this work was to control a robot on a path. The robot was controlled through state method and on a path according an improved control law. These two controllers in cascade made it possible to follow a path. A simulation allowed to find the correct settings required for implementation on the real system.

References

[1] F. Grasser, A. D'arrigo, S. Colombi and A. Rufer, "Joe: A Mobile, Inverted Pendulum", IEEE Transaction on Industrial Electronics,2002.

[2] R. C. Ooi, "Balancing a Two-Wheeled Autonomous Robot", Final Year Thesis, The University of Western Australia School of Mechanical Engineering, Faculty of Engineering and Mathematical Sciences University of Western Australia, Australia 2003.

[3] K. C. R. Ho, "Balancing Wheeled Robot", Research Project, University of Southern Queensland, Australia. 2005.

[4] R. Grepl, "Balancing Wheeled Robot: Effective Modelling, Sensory Processing And Simplified Control", Engineering Mechanics, 16 (2), pp.141–154,2009.

[5] Y. Takita, H. Date and H. Shimazu, "Competition of Two-wheel Inverted Pendulum Type Robot Vehicle on MCR Course", The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems,5578-5584, 2009.

[6] Y. O. Chee and M. S. Z. Abidin, "Design and Development of Two Wheeled Autonomous Balancing Robot", Student Conference on Research and Development, 2006.

[7] S. W. Nawawi, M. N. Ahmad and J. H. S. Osman, "Real-Time Control of a Two-Wheeled Inverted Pendulum Mobile Robot", World Academy of Science, Engineering and Technology,214-220, 2008.

[8] C. C. Tsai, C. K. Chan and Y. H. Fan, "Planned Navigation of a Self-balancing Autonomous Service Robot", IEEE International Conference on Advanced Robotics and Its Social Impacts, vol 6764, 2008.

[9] A. Micaelli, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots", Sophia-Anitpolis, France: Institut National de Recherche en Automatique er en Automatique, 1993.

[10] J. S. Hu, M. C. Tsai, F. R. Hu and Y. Hori, "Robust Control For Coaxıal Two-Wheeled Electrıc Vehıcle", Journal of Marine Science and Technology, pp 172-180,2010.

[11] G. Chi, J. Hausbach and B Hunter, "Segbot", Senior Design Project, University of Illinois at Urbana-Champaign, USA , 2005.

[12] Bock, H.G., & Plitt, K.J, "A multiple shooting algorithm for direct solution of optimal control problems", 9th IFAC world congress Budapest. Pergamon Press, 1984.

[13] G. Klancar, Skrjanc, I, "Tracking-error model-based predictive control for mobile robots in real time", Robotics and Autonomous Systems, 2007.

[14] E. Courtial,"Commande predictive et estimation d'état de systèmes non linéaires », rapport de thèse université Claude Bernard – Lyon.,1996.

[15] T. Vallius and J. Röning, "Embedded Object Concept: Case Balancing Two-Wheeled Robot", Proceedings of the SPIE, Vol. 6764, 2007.

[16] M.Bak. N.K.Poulsen, O.Ravn, "Path Following Mobile Robot in the Presence of Velocity Constraints", Kongens Lyngby, Denmark : Technical University of Denmark, 2000.