2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

A Lego Mindstorms NXT experiment for Model Predictive Control education

Massimo Canale
Dipartimento di Automatica e Informatica
Politecnico di Torino
Italy
massimo.canale@polito.it

Simone Casale Brunet
SCI-STI-MM
École Polytechnique Fédérale de Lausanne
Switzerland
simone.casalebrunet@epfl.ch

***Abstract*— This paper presents an educational framework based on the Lego Mindstorms NXT robotic platform used to outline both the theoretical and practical aspects of the Model Predictive Control theory. The case of a two-wheeled inverted pendulum is considered as at-size scenario. For such a system, starting from its mathematical modeling, an established design methodology is presented aiming to outline step-by-step the predictive controller implementation on a low-power architecture. The effectiveness of this multidisciplinary approach is illustrated along this presentation and demonstrated with experimental results.**

I. Introduction

Model Predictive Control (MPC) [1] has been established as an effective methodology for a wide range of applications in different engineering fields (see e.g. [2]) due to its efficiency in constraints handling and performance achievement. As a matter of fact, since in MPC the control action is computed, for each sampling time, through the solution to an optimization problem, its use has been restricted, for several years, to plants with slow dynamics (e.g. petrochemical, biomedical). Indeed, thanks to the efficient implementation strategies recently developed, the successful employment of MPC has been extended to applications whose sampling times prevented online solution to the optimization problem. In such a context, several methodologies have been proposed in order to compute the control move at a given sampling time (see e.g. [3]–[8] and the references therein). Among these solutions, the use of an approximate control law obtained through the off-line computation of a finite number of values of the actual controller (see e.g. [3], [5], [9]) has shown to be quite effective in the control of nonlinear complex plants as introduced, e.g., in [10], [11]. It is worth noting that, when an approximate MPC controller is used, the actual performance of the controlled system relies on an effective development of several design steps. In particular, a complete design procedure needs, besides a suitable choice of the problem formulation (i.e. plant modeling, cost function, constraints, ...), the computation of the values of the nominal controller in order to perform the control law approximation, realistic simulations of the controlled plant performed through the MatLab/Simulink environment to effectively test the system in the presence of the approximate controller and, finally, an effective software implementation on the available hardware platform. Therefore, a successful realization of an MPC controller requires a certain expertise in all the above described steps of the design procedure. On the basis of the considerations above, improvements in the practice of MPC can be achieved by introducing in the related education, suitable didactic at-size experimental scenarios which take care of all the aspects involved in the design ranging from modeling to real time implementation of the controller. In this contribution, we describe the deployment of an experimental framework well suited for educational purposes in which all the design steps previously introduced are considered. In particular, the problem of design and implementation of an MPC controller for an inverted pendulum built with the didactic environment Lego Mindstorm NXT will be described. Such a platform has been chosen, since, due to its low cost and versatility in building challenging laboratory experiments, it has been widely employed for control education purposes (see e.g. [12], [13]). Another interesting feature of the Lego Mindstorm NXT is that it supports, among others, the open source environment nxtOSEK which includes a real time operating system (i.e. TOPPER/JSP) and it can be easily interfaced with a desktop computer running MatLab allowing, in such a way, the deployment of the source code for the implementation of the MPC controller. The paper is organized as follows: in Section II the main features of the employed Lego Mindstorm NXT are described; the modeling and the MPC control design procedures are introduced in Sections III and IV respectively, while, in Section V controller verification and validation procedures as well as the obtained experimental results are presented. Some concluding remarks end the paper.

II. The Lego Mindstorm Platform

A two wheeled inverted pendulum has been used as at-size scenario, since practical use case studies have been proved to be more productively in the long term [14], [15]. This has been implemented with the low cost and widely used *Lego Mindstorms NXT* robotic kit. This is an educational platform representing the evolution of six decades of modular construction techniques from Lego and with years of computer-based education methods [16]. Using this robotic platform several different at-size scenarios can be easily arranged [12]–[14]. Indeed, the choice of a two wheeled inverted pendulum has been oriented by the challenging goal of controlling a plant with fast dynamics around an

unstable equilibrium point. In the following, the main Lego Mindstorms NXT environment features are outlined.

A. Hardware capability and limitation

According to [17], [18], the Lego Mindstorms NXT system architecture is summarized in Fig. 1. The *NXT Intelligent Brick* is the heart of the system and contains the main computational units and communication interfaces used by sensors and actuators. Two processors are used: a 32-bit ARM7TDMI (48 MHz, 256 KB flash memory, 64 KB RAM) used as main processor and a 8-bit ATmega48 micro-controller (4 MHz, 4 KB flash memory, 512 Bytes RAM) used as servo driver. Moreover, the Intelligent Brick has three motor driver ports with encoder inputs and four sensor inputs I2C-capable with two digital I/O lines and one analog input. Both wired (i.e. I2C, USB 1.1 slave mode and R485) and wireless (Bluetooth) communication interfaces are supported for external devices communication. Different kind of manufactured sensors are provided by Lego (e.g. color sensor, touch sensor, light sensor, ultrasonic sensors) and by third party commercial suppliers as well (e.g. accelerometer sensor, gyrosensor, compass sensor).

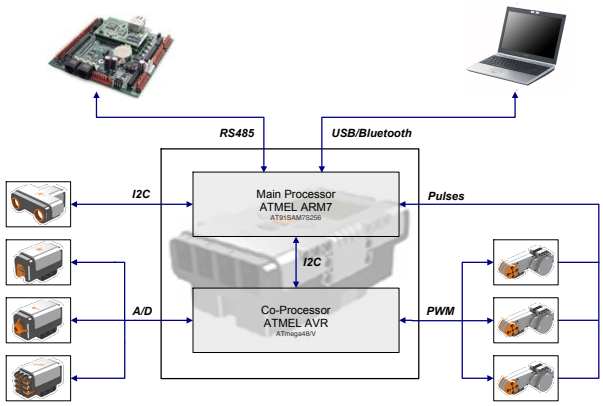


Fig. 1: The Lego Mindstorms NXT platform architecture

B. Software capability and limitation

The Intelligent Brick configuration supports both NXT-G and LabVIEW programming languages, although a variety of similar unofficial languages exist such as NXC, NBC, leJOS NXJ, and RobotC. On the other hand, from the authors' experience, the nxtOSEK [19] seems to be a suitable candidate for a first programming approach towards real time systems due to the characteristics described hereafter. This is an open-source environment that consists of device driver of leJOS NXJ C/Assembly source code, the TOPPERS/ATK (Automotive Kernel, formerly known as TOPPERS/OSEK) and the TOPPERS/JSP Real-Time Operating System. Moreover, it fulfills the OSEK/VDX standards, widely used in the automotive domains, it provides an ANSI C/C++ programming environment and programmable real-time multi tasking features. Furthermore, nxtOSEK also supports MatLab Simunlink as programming and code-synthesis environment.

C. Hardware and Software improvements

Lego provides a free available hardware and software development kit with all schematics and an open source firmware enabling users to build their own sensors or actuators. If the NXT is used to teach embedded developing, even JTAG debugging can be performed. However, its system architecture definitely influences the software run-time environment and its internal electronics cannot be easily altered and improved. In order to implement the MPC controller as illustrated in Section V a memory extension has been required. We think that the easiest way to overcome this issue is to connect an external memory controller (e.g. Rabbit, Arduino embedded systems) through the RS485 interface. In this direction, we developed a high performance driver that we released for the nxtOSEK 2.13. Possible future improvements should focus on integrating additional memory to the NXT Intelligent Brick. However, this will require the soldering of additional component and several modifications on the NXT firmware.

D. System Building

The two wheeled inverted pendulum used for our purposes has been inspired by the *NXTwayGS* [19] and it is shown in Fig. 2. The main components are a NXT Intelligent Brick and two servo-motors provided by Lego and a gyro-sensor provided by HiTechnic [20]. As illustrated in Fig. 3a, servo-motors includes a DC motor, a rotary encoder and a gearbox. Servos can be powered at $9V$ (alkaline batteries) or at $7.2V$ (NiMH batteries). Each motor is controlled by an H-bridge circuit and the speed of the motor is controlled by a Pulse Width Modulation (PWM) signal whose value is an integer which lies within the interval $[-100, 100]$ with a frequency of 7.8 kHz. In the rest of this paper, this control signal is defined as u. The encoder generates signals in quadrature, allowing the angular position and speed of the motor to be determined. The gyro-sensor is depicted in Fig. 3b and it contains a single axis gyroscopic sensor that detects rotation and returns a proportional value of the angular speed of the body of the pendulum. It is able to measure up to 360 deg/s with a sampling frequency of 300 Hz. It should be noted that measures based on this sensor are affected by a *gyro-offset* error (i.e. the output obtained with null sensor rotations) whose presence needs to be considered during the control design in order to mitigate its effects on control performance. A Rabbit BL2600 [21] has been used as external device for storing the approximate MPC controller. The Lego NXT and the Rabbit BL2600 are connected through a RS485 interface.

III. System Modelling

In this Section, a physical non-linear model of the Lego Mindstorm two wheeled pendulum described in the previous Section is introduced. In this paper, for simplicity, it will be assumed that the two servomotors are driven by the same PWM signal. As a consequence, such a system can be suitably modelled through the inverted pendulum configuration shown in Fig. 4 where the dynamical equations for the torque

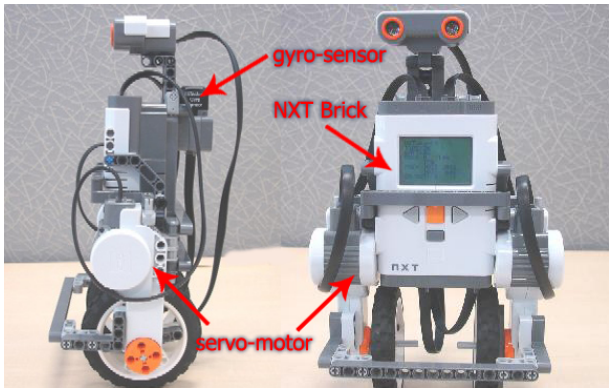


Fig. 2: The NXTwayGS

(a) The servo-motor (b) The gyro-sensor

Fig. 3: Servo-motor and gyro-sensor used for the scenario

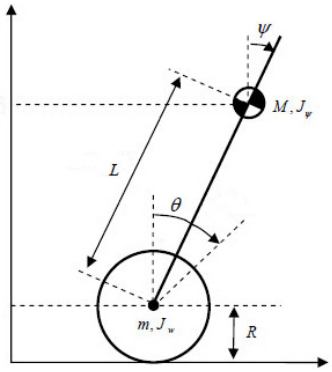


Fig. 4: Two-wheeled inverted pendulum mechanical model

equilibrium are the following:

$$\begin{aligned} T_\theta(t) =&\big((2m+M)R^2 + 2n^2 J_m\big)\ddot{\theta}(t) - MLR\dot{\psi}^2(t)\sin\psi(t) \\ &+ \big(MLR\cos\psi(t) - 2n^2 J_m\big)\ddot{\psi}(t) \\ T_\psi(t) =&\big(MLR\cos\psi(t) - 2n^2 J_m\big)\ddot{\theta}(t) - MgL\sin\psi(t) \\ &+ \big(ML^2 + J_\psi + 2n^2 J_m\big)\ddot{\psi}(t) \end{aligned} \tag{1}$$

where $\theta(t)$ and $\psi(t)$ represent the wheel and the mass angular positions respectively as depicted in Fig. 4. As described in [18], the torques $T_\theta(t)$ and $T_\psi(t)$ depend on the electric motors input voltages $v_l(t)$ and $v_r(t)$ through the following equations:

$$\begin{aligned} T_\theta(t) =&\alpha\big(v_l(t) + v_r(t)\big) - 2\big(\beta + f_w\big)\dot{\theta}(t) + 2\beta\dot{\psi}(t) \\ T_\psi(t) =&-\alpha\big(v_l(t) + v_r(t)\big) + 2\beta\dot{\theta}(t) - 2\beta\dot{\psi}(t) \end{aligned} \tag{2}$$

where:

$$\begin{aligned} \alpha =& \frac{nK_t}{R_m} \\ \beta =& \frac{nK_tK_b}{R_m} + f_m \\ v_l(t) = v_r(t) =& G_u(\mu V_b(t) - V_o)\, u(t - t_a) \end{aligned} \tag{3}$$

where $u(t)$ is the PWM signal which, as already said, has been assumed to be equal for both the electric motors and $t_a = 4$ ms is the actuator delay. The description and the nominal values of each model parameter are reported on Table I. By introducing the following state variable $x(t) \in \mathbb{R}^4$:

$$x(t) = \begin{bmatrix} \theta(t) & \psi(t) & \dot{\theta}(t) & \dot{\psi}(t) \end{bmatrix}^T \tag{4}$$

and the input variable $u(t) \in \mathbb{R}$, equations (1), (2) and (3) can be rearranged in order to obtain a nonlinear state-space description of the system dynamics of the form

$$\dot{x}(t) = f(x(t), u(t)) \tag{5}$$

According to the employed sensor configuration described in Section II, the measured outputs are the wheel angular position θ and speed $\dot{\theta}$ and the pendulum angular speed $\dot{\psi}$:

$$y(t) = [\theta,\ \dot{\theta},\ \dot{\psi}]^T$$

TABLE I: Inverted pendulum parameters description and nominal value

Variable	Value	Unit	Description
g	9.81	m s^{-2}	Gravity acceleration
m	0.03	kg	Wheel weight
R	0.04	m	Wheel radius
M	0.6	kg	Body weight
L	0.072	m	Wheel axle center of mass distance
J_ψ	0.001	kg m^2	Body pitch inertia moment
J_m	10^{-5}	kg m^2	DC motor inertia moment
R_m	6.69	Ω	DC motor resistance
K_b	0.468	V s rad^{-1}	DC motor back EMF constant
K_t	0.317	Nm A^{-1}	DC motor torque constant
f_m	0.0022	Nm rad s^{-1}	DC motor friction coefficient
V_b	8.00	V	Power Supply voltage
V_o	0.625	V	Power Supply offset
μ	1.089		Power Supply gain factor
G_u	10^{-2}		PWM gain factor
n	1		Gearbox ratio

As a matter of fact, the state space model (5) can not be directly used for the MPC control design since a discrete time model of the form:

$$x(k+1) = \bar{f}(x(k), u(k)) \tag{6}$$

is required. Such a model has been obtained through a forward difference discretization procedure using the sampling time $T_s = 4$ ms. In (6), for simplicity of notation, the time variable $k \in \mathbb{Z}^+$ is used to indicate the generic sampling instant $k\,T_s$.

IV. Controller Design

In this section, performance requirements are introduced as well as how to take into account of them through a suitable control design problem formulation within the MPC framework. The control objective is the stabilization of the

pendulum around its vertical position corresponding to $\psi = 0$ using the action exerted by the servomotors. In particular, it is desired that the pendulum reaches a "standing" position while keeping the wheels are at rest (i.e. $\dot{\theta} = 0$). Moreover, some physical constraints have to be considered. First of all, the PWM signal amplitude $u(k)$ and its rate of variation $\Delta u(k)$ are physically limited as follows:

$$\begin{cases} |u(k)| \leq 100 \\ |\Delta u(k)| \leq 30 \end{cases}, \forall k \geq 0 \tag{7}$$

Furthermore, the behavior of the pendulum angular position ψ has been constrained as

$$|\psi(k)| \leq 6°, \forall k \geq 0 \tag{8}$$

in order to limit possible oscillation amplitude of the pendulum body during the transient phase.

In order to take into account such requirements, the following performance index can be considered

$$J(k) = \sum_{i=1}^{H_p} (x(k+i|k)^T Q x(k+i|k) + R\Delta u^2(k+i-1|k)) \tag{9}$$

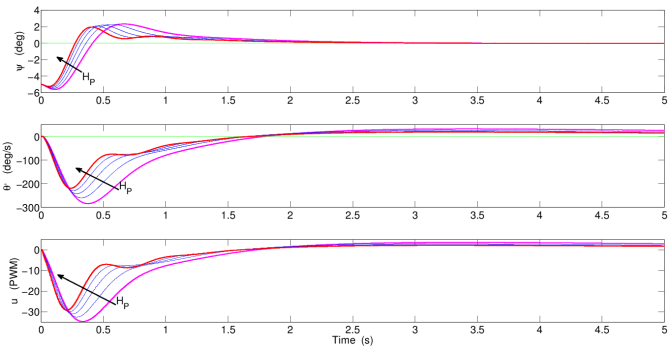
where H_p is the prediction horizon. An MPC controller is designed considering the following optimization problem:

$$\min_{\Delta U} J(k)$$
$$\text{subject to} \begin{cases} (7), (8) \\ x(k+H_p|k) = 0 \end{cases} \tag{10}$$

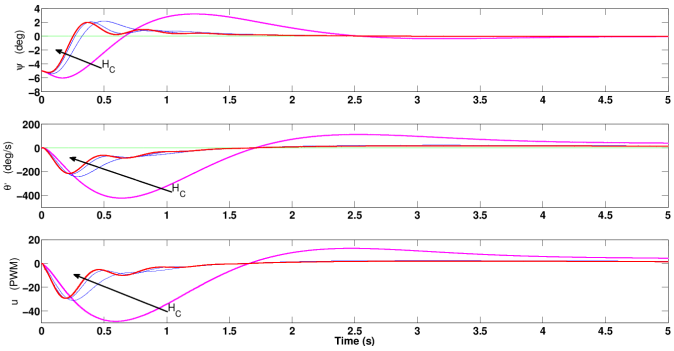
In (10), $x(k+i|k)$ is the i^{th} step ahead prediction of the system state obtained using the model (6) and $\Delta u(k) = u(k) - u(k-1)$ is the control input increment, the vector $U = [\Delta u(k|k), \ldots, \Delta u(k+H_c-1|k)]^T$ is the sequence of the control rates to be optimized, $H_c \leq H_p$ is the control horizon. The remaining optimization variables $[\Delta u(k+H_c|k), \ldots, \Delta u(k+H_p-1|k)]^T$ are set to 0, i.e. $u(k+j|k) = u(k+H_c-1|k)$, for all $j = H_c, \ldots, H_p-1$. The terminal state constraint $x(k+H_p|k) = 0$ (see e.g. [1]) has been included too in order to guarantee closed loop stability. Moreover, $Q \in \mathbb{R}^{4,4}$ is a suitable positive semi-definite matrix and $R \in \mathbb{R}$ is a positive scalar. In particular, given the state vector definition in (4) and the control requirements introduced above, matrix Q is assumed of the form:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & q_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{44} \end{bmatrix}$$

where the entries q_{22}, q_{44} and R have to be determined to trade-off among the different requirements. In particular, the following values have been chosen $q_{22} = 4.5\ 10^7$, $q_{44} = 10^6$, $R = 3.5\ 10^5$. It also must be noted that, since the optimization is performed w.r.t. ΔU, an integrator, which can be helpful in mitigating the effects of the gyro-sensor offset,



(a) Tuning of $H_p \in [150, 250]$ with $H_c = 4$.



(b) Tuning of $H_c \in [2, 10]$ with $H_p = 200$.

Fig. 5: Time behavior of ψ, $\dot{\theta}$ and u for different choices of: (a) the prediction horizon H_p, (b) the control horizon H_c.

is added in the loop. The control action is then computed according to the receding horizon (RH) strategy described below:

1) At time instant k get $x(k)$ and $u(k-1)$
2) Solve the optimization problem (10) and get the minimizer $\Delta U^* = [\Delta u^*(k|k), \ldots, \Delta u^*(k+H_c-1|k)]^T$
3) Apply $u(k) = u(k-1) + \Delta u^*(k|k)$
4) $k \leftarrow k+1$

Note that, since the performance index $J(k)$ in (9) depends on $x(k)$ and $u(k-1)$, then $\Delta u^*(k|k)$ can be implicitly express as $\Delta u^*(k|k) = \kappa(x(k), u(k-1)) = \kappa(w(k))$, where $w(k) = [x(k);\ u(k-1)] \in \mathbb{R}^5$. Therefore, the application of the RH procedure gives rise to a feedback controller of the form $u(k) = u(k-1) + \kappa(w(k))$.

The prediction and control horizons, H_p and H_c, can be tuned through iterative simulations using MatLab/Simulink environment. In such a context, the MatLab function `fmincon` has been used to solve the optimization problem (10). Fig. 5a shows the behavior of the variables of interest (i.e. ψ, $\dot{\theta}$ and u) for different values of prediction horizon $H_p \in [150, 250]$ when control horizon is kept fixed at the value $H_c = 4$. Fig. 5b shows the similar results when the prediction horizon is fixed at $H_p = 200$ and $H_c \in [2, 10]$. In both Figures arrows in the plots indicate an increase of H_p and of H_c respectively. According to the simulation results, the values $H_p = 200$ and $H_c = 4$ have been chosen as the final settings in the design.

V. Controller implementation and experimental results

In this Section, implementation issues of the designed MPC controller are described. Note that, since the control action computation performed through the receding horizon strategy can not be realized within the required sampling time $T_s = 4$ ms, a fast implementation procedure has to be used. In particular, the Nearest Point (NP) method introduced in [5], has been employed due to its low computational burden and effectiveness in applications with small computing power such as the Mindstorms NXT (see also, in this regard, [11]). Basically, the NP method performs the implementation by approximating the control function $\kappa(w)$ using a finite number ν of its values obtained via offline simulations of the controlled system. In this regard, the needed control law values are computed over a compact subset $\Omega \subset \mathbb{R}^5$ of the function $\kappa(w)$ domain. Inside Ω, a finite number ν of points $\tilde{w}_\ell = \{\tilde{x}_\ell, \tilde{u}_{-1,\ell}\}$, $\ell = 1, \ldots, \nu < \infty$ is suitably chosen, giving rise to the set $\Omega_\nu = \{\tilde{w}^\ell \in \Omega,\ \ell = 1, \ldots, \nu\}$. For each value $\tilde{w} \in \Omega_\nu$, the corresponding value $\Delta\tilde{u} = \kappa(\tilde{w})$ is computed solving off-line the optimization problem (10). The values of $\tilde{w}$ and $\Delta\tilde{u}$ are stored and successively used for the on-line computation of the approximating function κ^{NP}. The set Ω_ν is obtained by choosing variables $\tilde{w} = [\tilde{x}; \tilde{u}_{-1}]$ within the hyperbox:

$$\Omega \doteq \{\tilde{w} : w^l \preceq \tilde{w} \preceq w^u\} \subset \mathbb{R}^5$$

where the symbol $\preceq$ indicates element-wise inequalities and $w^l \in \mathbb{R}^5$ and $w^u \in \mathbb{R}^5$ represent, respectively, the vectors containing the chosen upper and lower bound of each component $\tilde{w}_\ell, \ell = 1, \ldots, 5$ of $\tilde{w}$. Assuming that the values of $\tilde{w}_\ell$ within the interval $[w^l_\ell, w^u_\ell]$, are chosen using a uniform gridding procedure with a spacing amplitude of Δw_i, the number n^s_ℓ of samples of each component $\tilde{w}_\ell$ is given by:

$$n^s_\ell = int\left(\frac{w^u_\ell - w^l_\ell}{\Delta w_\ell}\right) + 1$$

where $int(\cdot)$ is the nearest integer approximation. The values of n^s_ℓ are chosen in order to obtain a suitable trade-off between accuracy and memory usage. Therefore, in the considered case case of a uniform gridding of Ω, the size ν_{UG} of the set is given by:

$$\nu_{UG} = \prod_{\ell=1}^{5} n^s_\ell$$

Once the elements $\tilde{w}^h$, $h = 1, \ldots, \nu_{UG}$ of the set have been defined, the corresponding values $\Delta\tilde{u}^h$ are computed. The data obtained in such a way (i.e. $\tilde{w}^h$ and $\Delta\tilde{u}^h$) are stored in a unique array $\mathbb{W}_{UG}$. For a given w, the NP approximation $\kappa^{NP}(w)$ of the controller function $\kappa(w)$, is the computed as:

$$\kappa^{NP}(w) = \kappa(\tilde{w}^{NP}) = \Delta\tilde{u}^{NP} \tag{11}$$

where

$$\tilde{w}^{NP} = \arg\min_{\tilde{w} \in \Omega_\nu} \|w - \tilde{w}\|_2 \tag{12}$$

In order to speed up the computation of κ^{NP}, a suitable structure for $\mathbb{W}_{UG}$ should be employed. In particular, in [11], an effective structure has been introduced which allows the computation of $\tilde{w}^{NP}$ (and of $\Delta\tilde{u}^{NP}$), through the determination of the row n^{NP} of $\mathbb{W}_{UG}$ where the needed value of has been stored (see [11] for details). At the each sampling time, the control move $u(k)$ is obtained by applying the following procedure:

1) Acquire $x(k)$ and $u(k-1)$
2) Compute the index n^{NP}
3) Get $\Delta\tilde{u}^{NP}$ as $\mathbb{W}_{UG}(n^{NP})$
4) Apply $u(k) = u(k-1) + \Delta\tilde{u}^{NP}$

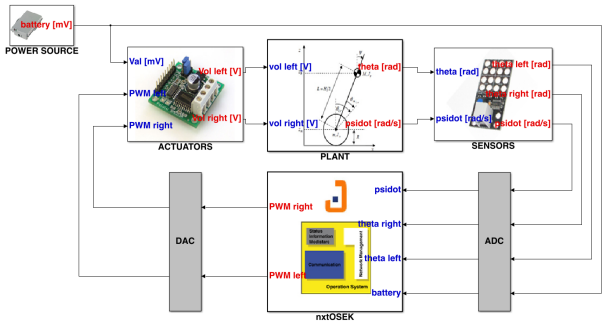


Fig. 6: Simulink realistic system model

In Table II, the gridding configuration chosen for this application is reported. This requires the off-line computation of 1279091 values of vector w. Each single incremental command value Δu is stored in a 8 bit signed integer. The memory requirement for $\mathbb{W}_{UG}$ is approximatively $1.3MB$. This array has been stored in the Rabbit BL2600 memory. As a matter of fact, due to the sensor configuration described in Section II, not all the states are measured. Therefore, a Luenberger observer, obtained using a linearised model of the state equation (6) has been added in order to provide such state values starting from the measured values of $\theta, \dot{\theta}$ and $\dot{\psi}$. In order to test the system performance when the approximate controller κ^{NP} is used, the accurate overall Simulink model reported in Fig. 6 has been realized. The final configuration of κ^{NP} is tested simulating the controller using the realistic Simulink model. In particular, such a model is made up by the following blocks: 1) the *Plant* block contains the mechanical non linear model of the system depicted in Fig. 4; 2) the *Actuators* block models the PWM effects; 3) the *Sensors* block models the generated analog signals according to the sensors non linearities and the respectively noise models; 4) the *AD/DA converters* block models the quantization and time delays effects introduced by the converters; 5) the *Operating System* contains the digital controller algorithm coded with the nxtOSEK MatLab Environment; 6) the *Power Supply* models the power consumption and battery discharging effects on the actuators. In Fig. 7 a comparison between the results obtained with the MPC design described in the previous section (magenta line) and with the NP approximate controller (blue line) are reported. From Fig. 7 a quite good agreement between

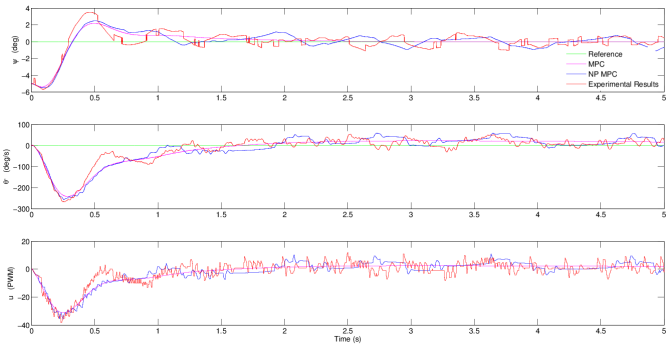


Fig. 7: Simulation and experimental results: (green line) the reference, (magenta line) simulation results for the ideal MPC controller, (blue line) simulation results for the NP MPC controller, (red line) experimental results for the implemented NP MPC controller.

TABLE II: Uniform Gridding Configuration

Variable	Num. of Points	Min.	Max.
θ	11	-300 °	300 °
$\dot{\theta}$	31	-360 ° s^{-1}	360 ° s^{-1}
ψ	31	-10 °	10 °
$\dot{\psi}$	11	-40 ° s^{-1}	40 ° s^{-1}
$u(k-1)$	11	-100	100

the ideal MPC closed loop system performance and the one governed by the NP controller has been achieved. After the assessment of such a positive performance the NP controller has been implemented on the NXT-Lego system and the red line on Fig. 7 reports the behaviour of the recorded variables on the real system. Observing Fig. 7, it can be noted that the behaviour of the simulated system and the real one are quite close thus proving the effectiveness of the proposed design framework. A short video showing the achieved capabilities of the designed system is available at `http://staff.polito.it/massimo.canale/mpcNXT.html` .

VI. Conclusion

In this paper, it has been shown how a low-cost Lego Mindstorms NXT platform can be successfully employed in the education of the practical issues of Model Predictive Control. A two wheeled inverted pendulum has been chosen as at-size scenario for the experimental practice. Control design and accurate simulations have been performed using a suitable tool developed in the MatLab/Simulink environment. Practical implementation of the controller has been realized through the Nearest Point approximation technique via basic C programming under the nxtOSEK Real Time operating systems whose characteristics are similar to other platforms employed in industrial applications. Experimental results show the effectiveness of the introduced approach.

References

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[2] S. Qin and T. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[3] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[4] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[5] M. Canale, L. Fagiano, and M. Milanese, "Set membership approximation theory for fast implementation of model predictive control laws," *Automatica*, vol. 45, no. 1, pp. 45–54, 2009.

[6] S. Richter, C. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods." in *Proc. of the 48th IEEE Conference on Decision and Control*, 2009, p. 73877393.

[7] J. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transaction on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[8] A. Grancharova and T. Johansen, *Explicit Nonlinear Model Predictive Control, LNCIS 429*. Berlin, Heidelberg: Springer Verlag, 2012.

[9] T. A. Johansen, "Approximate explicit receding horizon control of constrained nonlinear systems," *Automatica*, vol. 40, pp. 293–300, 2004.

[10] M. Canale, M. Milanese, and C. Novara, "Semi-active suspension control using "fast" model-predictive techniques," *IEEE Transactions on Control System Technology*, vol. 14, no. 6, pp. 1034–1046, November 2006.

[11] M. Canale, L. Fagiano, and V. Razza, "Approximate nmpc for vehicle stability: design, implementation and sil testing," *Control engineering practice*, vol. 18, pp. 630–639, 2010.

[12] S. Wadoo and R. Jain, "A lego based undergraduate control systems laboratory," in *Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island*, may 2012, pp. 1 –6.

[13] Y. Kim, "Control systems lab using a lego mindstorms nxt motor system," in *Control Automation (MED), 2010 18th Mediterranean Conference on*, june 2010, pp. 173 –178.

[14] P. Richmond and D. Chen, "A model predictive control package for undergraduate education," *Education for Chemical Engineers*, vol. 7, no. 2, pp. e43 – e50, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1749772812000024

[15] B.-Y. Shih, C.-Y. Chen, C.-W. Chen, and I. Hsin, "Using lego nxt to explore scientific literacy in disaster prevention and rescue systems," *Natural Hazards*, vol. 64, pp. 153–171, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11069-012-0233-2

[16] S. Papert, *The children's machine: rethinking school in the age of the computer*. New York, NY, USA: Basic Books, Inc., 1993.

[17] Lego. (2012, Oct.) Mindstorms NXT Hardware Developer Kit (HDK). [Online]. Available: http://mindstorms.lego.com/en-us/support/files/default.aspx

[18] M. Gasperi and P. Hurbain, *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level, Second Edition*, ser. Technology in Action Series. Apress, 2009.

[19] T. Chikamasa. (2012, Oct.) nxtOSEK/JSP. [Online]. Available: http://lejos-osek.sf.net/

[20] HiTechnic. (2012, Oct.). [Online]. Available: http://www.hitechnic.com

[21] Digi. (2012, Oct.) RabbitBL 2600. [Online]. Available: http://www.digi.com