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

A Flexible Low Cost Embedded System for Model Predictive Control of Industrial Processes*

Daniel M. Lima, Marcus V. Americano da Costa and Julio E. Normey-Rico
Department of Automation and Systems
Federal University of Santa Catarina
Florianópolis, Brazil
{daniellm,adacosta,julio}@das.ufsc.br

Abstract— **Model Predictive Control is an algorithm commonly used in the petrochemical and chemical sectors to control large processes. Its success can be traced to the fact that it is MIMO control algorithm with intrinsic dead-time compensation, capacity to handle the process constraints and with tuning parameters that are easily understandable in the time domain. Nevertheless, its usage is limited to large processes where the investment in the currently expensive advanced control systems are economically viable. Thus, this work proposes a low cost embedded MPC controller targeted at companies with medium and small processes that could have production improvements with the use of MPC. This paper details the current embedded system prototype and the results obtained with a hardware-in-the-loop experiment of an ethanol distillation process simulator.**

I. INTRODUCTION

Model Predictive Control (MPC) is largely used in the petrochemical and chemical sectors of the process industry, specially in refineries. Several researches have shown that the use of this type of controller allows the improvement of industrial production quality [1].

The term MPC does not designate a specific control strategy but a very ample range of control methods which make an explicit use of a model of the process to obtain the control signal by minimizing an objective function [1]. This strategy takes into account the future error prediction and the constraints that the process can have on manipulated and controlled variables.

However, the use of these controllers is targeted at large processes in which the investment in Distributed Control Systems (DCS) is economically viable. In these systems, MPC is usually used in cascade with classical PID controllers that manipulate the fundamental variables of the process [2]: material flows, temperature, level, etc., that is, the control signals generated by the MPC controller are the set-points of these local classical controllers.

Although MPC controllers have been successfully applied in some sectors of the industry, it is not well disseminated in medium and small plants, where it could bring significant improvement in production. One of the reasons for this is the high cost of commercial MPC packages currently being sold.

*The authors would like to thank the Brazilian National Counsel of Technological and Scientific Development (CNPq) for the financial support.

There are many works aiming to disseminate the use of MPC through different types of solution. One line of research is to optimize a Field Programmable Gate Array (FPGA) to work with MPC algorithms [3], [4]. FPGA is a type of integrated circuit that can be configured according to the user's application and these works explore some peculiarities of MPC to increase the speed of the optimization procedure. Unfortunately, this kind of solution requires a greater research period and funding. Another type of solution is proposed in [5], where the MPC algorithm is embedded in an industrial programmable logic controller (PLC) using the IEC 61131-3 programming standard.

These projects propose solutions that try to disseminate the use of MPC with low level equipments, e.g. actuators, thus, they manage to obtain sampling times in the order of milliseconds even while handling constraints. Nonetheless, the number of constraints and control variables are very restricted due to the limitations of the hardware being used.

Thus, this project offer a different solution. An embedded MPC controller implemented in an off-the-shelf hardware with greater resources (memory, processor speed, etc.), compared to an FPGA or PLC, that could be used to control small and medium processes with slow dynamics in the same way that the commercial MPC packages are used, i.e. providing set-points to the low level controllers.

The solution proposed offer an embedded system with the following characteristics: (i) Easy integration with numerous industrial communication protocols (FIELDBUS, MODBUS, RS485, RS232); (ii) Fast configuration of the controller and of the hardware; (iii) Multiple MIMO MPC algorithms with different characteristics.

This paper will discuss some of the features of this solution and the latest results regarding a hardware-in-the-loop experiment with the prototype of the embedded MPC controller, which was used to analyze and test the proposed system.

The rest of the paper is organized as follows: the mathematics behind the MPC algorithm will be introduced in section II, and more details about the embedded device being used will be in section III. In section IV a hardware-in-the-loop experiment using the proposed MPC and a simulated distillation process is presented. Finally, the conclusions are provided in section VI.

II. MPC ALGORITHM

The family of Model Predictive Controllers have different algorithms that deal with different type of representations of a process but, basically, MPC is an optimal control that minimizes a cost function that depends on the predicted output of the process, the future set-points and on the future inputs. The main differences between the MPC algorithms lie in the type of plant and disturbance models used to calculate the future predictions and the cost function to be minimized [1].

To calculate the control action all of the algorithms execute the following steps:

- Prediction: based on a model of the process and disturbances, the predictions of the future behavior of the plant are obtained considering the current state of the process;
- Control action computation: the current control action is obtained minimizing the cost function taking into account the process' constraints;
- Actuation: the control action is sent to the actuator, and the algorithm sequence is repeated.

Most of practical applications of MPC in the process industry are based on linear MPC strategies, thus, in the current version of the proposed platform three linear MPC algorithms were implemented: (a) Generalized Predictive Control (GPC) [1] to allow the operation with transfer function based process models; (b) State Space MPC (SSMPC) to use state space representation of the process dynamics and (c) Dead-Time Compensator GPC (DTCGPC) [6], which is a modified version of GPC that allows an easier robust tuning of the controller. However, in future versions, a nonlinear MPC strategy will be included.

In all cases the algorithm is implemented using the idea of free and forced responses, both of them obtained using the process and disturbance models. The former shows the response of the system if the future control action is not changed and the latter indicates the response in case the control action is modified. Thus, for a $m \times n$ multivariable plant, the future process output prediction vector $\hat{\boldsymbol{Y}}$ can be represented as:

$$\hat{\boldsymbol{Y}} = \boldsymbol{G}\boldsymbol{U} + \boldsymbol{F}, \tag{1}$$

where $\boldsymbol{F}$ is the free response vector, $\boldsymbol{U}$ is the vector of future increments in the inputs that will be calculated and $\boldsymbol{G}$ is a matrix obtained from the model equations and known in literature as step response matrix of the system [1]. In this equation $\hat{\boldsymbol{Y}}$ and $\boldsymbol{F}$ have dimension $n_y \times 1$, $\boldsymbol{U}$ has dimension $n_u \times 1$ and $\boldsymbol{G}$ is $n_y \times n_u$ where $n_y = \sum_{i=1}^{m} N_{y_i}$, $n_u = \sum_{i=1}^{n} N_{u_i}$, N_{y_i} is the prediction horizon of output y_i and N_{u_i} is the control horizon of input u_i.

The quadratic cost function used in the implemented MPCs is:

$$J = [\boldsymbol{W} - \hat{\boldsymbol{Y}}]^T \boldsymbol{Q_y} [\boldsymbol{W} - \hat{\boldsymbol{Y}}] + \boldsymbol{U}^T \boldsymbol{Q_u} \boldsymbol{U}, \tag{2}$$

where $\boldsymbol{W}$ is the vector of future references and $\boldsymbol{Q_y}$, $\boldsymbol{Q_u}$ are, respectively, the weights of the errors and control effort. In this problem, the horizons and weighting matrices are the tuning parameters of the controller.

Substituting (1) in (2), and after some rearrangements it is possible to obtain (3), where the vector $\boldsymbol{U}$ that minimizes this equation contains the increments of the control action to be applied at the current time.

$$J(\boldsymbol{U}) = \frac{1}{2}\boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U} + \boldsymbol{b}^T \boldsymbol{U} + \boldsymbol{f}_0, \tag{3}$$

where $\boldsymbol{H} = 2(\boldsymbol{G}^T \boldsymbol{Q_y} \boldsymbol{G} + \boldsymbol{Q_u})$, $\boldsymbol{b}^T = 2(\boldsymbol{F} - \boldsymbol{W})^T \boldsymbol{Q_y} \boldsymbol{G}$ and $\boldsymbol{f}_0 = (\boldsymbol{F} - \boldsymbol{W})^T \boldsymbol{Q_y} (\boldsymbol{F} - \boldsymbol{W})$.

Taking the constraints into account, the problem to be solved by the MPC algorithm can be represented as the minimization of the following Quadratic Problem (QP):

$$\begin{array}{ll} \text{Minimize} & J(\boldsymbol{U}) \\ \text{Subject to:} & A\boldsymbol{U} \leq B \end{array} \tag{4}$$

where $A\boldsymbol{U} \leq B$ represents the constraints conditions on the inputs and outputs of the process.

Summarizing, the MPC algorithm steps are: (a) obtain the reference $\boldsymbol{W}$; (b) compute the free response $\boldsymbol{F}$; (c) compute the constraints matrixes A and B; (d) obtain the solution of the QP (4); (e) apply the computed control signal; (f) update the variables and repeat.

III. EMBEDDED SYSTEM ARCHITECTURE

As explained before, the goal of this project is to develop a low cost embedded MPC controller that can be easily integrated and configured with the existing equipments of industrial processes. To do that, it was opted to use off-the-shelf components instead of building the embedded hardware from scratch. The chosen hardware is described bellow.

A. Hardware

The embedded system used in this project is a TS-7500 Development Kit (Fig. 1) which is composed of a 250 MHz Cavium ARM9 CPU, 64 Mb of DDR-RAM, micro-SD slot and a bootable 4 Mb of on-board flash. Its hardware also allows the use of the following communication protocols: Ethernet, RS232, RS485 and CANBUS. This development kit met all the requirements to be used in this project, that is, it is readily available, relatively inexpensive and offers a range of different communication hardware.

B. Software

To make the final product even less expensive, it is necessary not to use third-party proprietary software libraries and programs. Fortunately, the open-source community offers innumerable free software alternatives that can be used to accelerate the project development. For instance, the prototype being develop runs the Debian Linux (kernel version 2.6) operating system (OP). The OP manages the embedded system hardware, providing easy to use programming interfaces that enables quick time to market for end-users' applications. Another advantage of these interfaces is that they help separate hardware specific configurations from the

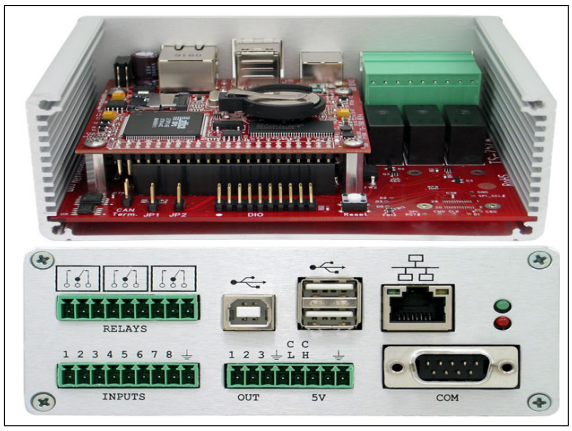

Fig. 1. Front and inside views of the embedded system used in the project as a MPC controller

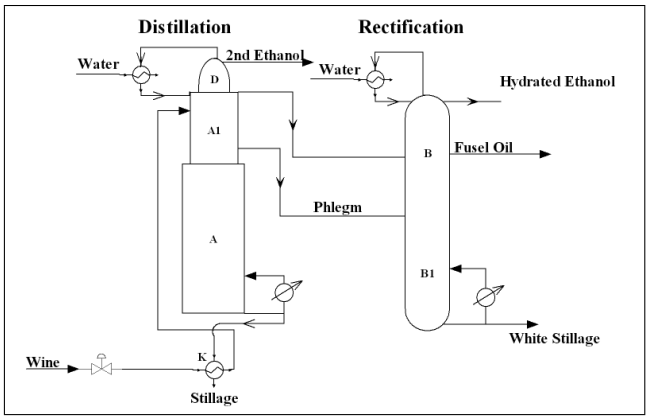

Fig. 2. Ethanol distillation unit

software being developed thus making it more portable, that is, the software is hardware-independent.

The software of the MPC controller was developed in Python, an interpreted programming language that offers many free open-source libraries. One of great importance is the Convex Optimization package (CVXOPT), that is responsible for the minimization of the cost function of the MPC algorithm whose result is the control action to be applied.

Because of the interpreted nature of the Python language, Python programs usually take more time to execute than their implementations in languages like C or C++. In spite of this, it will be shown that the current version of the software is capable to handle small and medium processes with slow dynamics.

C. Integration with Processes

In the process industry nowadays, actuators, sensors and a range of other equipments necessary for the correct operation of a process are interconnected via one or more industrial networks (MODBUS, Foundation Fieldbus, etc.) [7]. These equipments are monitored and controlled by supervisory software, or equipments, so that the production obeys certain requirements made by the production management. So, to become one of the equipments of the process, the embedded MPC controller must be inserted in one of these networks, then it will have access to the set-points, manipulated and process variables to control the process. This integration is only possible if the controller is compatible with the type of industrial network being used. That is why the hardware chosen for the embedded controller must have various communications interfaces, so it can be compatible with different type of networks without modifications to the hardware, thus making the integration easier.

In the next section, the proposed hardware platform will be integrated with a commercial simulator that represents an ethanol distillery plant.

IV. EXPERIMENT

This section presents a hardware-in-the-loop experiment using the proposed low cost MPC controller and the simulator Aspen HYSYS. It is important to note that this simulator is largely used in the industry, making the test experiment closer to real situations. The main goal of this experiment is to ascertain the stability and the capacity of the current version of the embedded controller. The results will provide information about the current hardware computational capabilities.

Before the results, it will be explained the process that the MPC was implemented in.

A. Distillation Unit

In the Brazilian ethanol industry, the distillation is commonly configured according to Fig. 2, in which it is shown the presence of two stages: distillation and rectification. The products generated at this stages are 2nd and hydrated ethanols. The sub-products are stillage, white stillage and fusel oil.

The distillation unit has a significant role in the ethanol industry, since this stage consumes most of the energy used in this process production. In spite of that, unfortunately, it is very common an inadequate operation of the control systems in these columns which, in most cases, causes a higher consumption of energy and a degradation of product quality. To study this problem, a simulator of a complete ethanol distillery was developed using the software Aspen HYSYS®[8]. This program is a market-leading process modeling system, used by the world's leading oil and gas organizations. Although the control schemes used in the simulation were implemented according to structures presented in the related literature, it is possible to test numerous configurations thanks to the flexibility of the software.

Despite a great number of alternative strategies to improve the operation of distillation columns, the use of the MPC algorithm is constantly growing and could be a viable option to increase the energy efficiency in the ethanol production [8]. Furthermore, the fact that the simulation software provides a very realistic model of the distillation unit, it was chosen as the first experiment to test the embedded system developed in this project.

B. Description of the Process

The distillation process is based on the difference of volatility of its components, characterized by a double change of physical state. Initially, a substance in liquid state is heated until the boiling point is reached and at this moment the substance begins to vaporize. The liquid and the vapor phases generally contain the same components but in different relative quantities. The liquid, in its bubble point[1], and the vapor, in its dew point[2], are both at an equilibrium where there is a simultaneous mass exchange of liquid due to vaporization and vapor due to condensation. The final effect is an increased quantity of the most volatile component in the vapor and of the least volatile component in the liquid. Later on, the vapor is cooled until most of its mass returns to liquid [8], thus obtaining a separate liquid substance where the main component is different.

In this type of process, the main controlled variables are: concentration of the top and bottom products of the column (the compositions are frequently inferred from the temperature and boiling points in the current measured pressure); liquid levels at the bottom of the column and in the reboiler and condenser; temperature and column pressure. It is important to note that all the mass inputs and outputs of the column must be measured and local controllers should be used in the vapor inputs that can be manipulated [8]. The manipulated variables are the output of the products, heating and cooling fluids and the reflux. The latter makes the existence of the liquid phase at the top of the column possible by cooling it. It is usually not possible to manipulate the feeding input because this is dependant of other processes.

In this particular case, the input (wine), which is the product of the previous step of the ethanol plant, the fermentation process, is heated and feeds column A1. The wine composition is approximately 90 % of water, ethanol (7 to 10 ^oGL) and other substances found in smaller quantities. The wine starts to boil and to loose the most volatile components that rise to column D. From the base of column A1 the wine passes to column A and descends, loosing alcohol until it arrives at the bottom of the column, generating stillage. At the top of column A there is an output of phlegm in vapor state that is taken to column B1 for rectification. At the top of column D, vapor is condensed and the product is obtained, the 2nd ethanol at a concentration of 92 ^oGL, while the product at the bottom of this column, which is rich in ethanol, is taken to column B for rectification. The base of column B is fed with phlegm (approximately 50 ^oGL), which will rise through the column, becoming enriched with ethanol, until it reaches the top where it will be condensed and withdrawn as hydrated alcohol (approximately 96 ^oGL). In the lower levels, the solution will form white stillage, a byproduct very poor in ethanol.

[1]Temperature at which vaporization begins.
[2]Temperature at which condensation begins.

C. Control of the Distillation Column

As this process works on a continuous setup stabilized at a certatin operation point, the main problem in control is the rejection of the disturbances. Thus, the goals of the MPC controller in this experiment are: (i) maintain the 2nd ethanol concentration at 92 ^{o}GL; (ii) maintain the hydrated ethanol concentration at 96 ^{o}GL; (iii) maintain the stillage concentration below 0.04 ^{o}GL.

For that, the MPC control system at the top layer in cascade with the local PID controllers will manipulate three variables: pressure at the bottom of column A, temperature at the top of column D, and the level of the tank of the condenser at column B. The disturbances considered in this problem are: (i) wine feed, produced by the previous process, (ii) phlegm rate flow, and (iii) hydrated ethanol rate flow. The last two are altered to adjust the process output according to the production directives.

Although the columns have highly nonlinear dynamics, the MPC algorithm being used calculates the control signals using linear equations. In this way, it is first necessary to identify the linearized models of the process in a desired operation point to tune the control system. The operation points chosen are, for the outputs, $\bar{y}_1 = 92$ ^{o}GL for the 2nd ethanol, $\bar{y}_2 = 0.03$ ^{o}GL for the stillage, and $\bar{y}_3 = 96$ ^{o}GL for the hydrated ethanol. For the inputs, $\bar{u}_1 = 119.1$ kPa for bottom pressure, $\bar{u}_2 = 72.38$ ^{o}C for top temperature and $\bar{u}_3 = 50\%$ of the maximum tank level of the condenser of column B. Finally, for the disturbances, $\bar{d}_1 = 295000\ kg/h$ for the wine feed, $\bar{d}_2 = 39550\ kg/h$ for the phlegm output and $\bar{d}_3 = 18130\ kg/h$ for the hydrated alcohol output.

After identification, the linearized input-to-output, $P_u(s)$ in (5), and disturbance-to-output, $P_d(s)$ in (6), normalized models were obtained in the Laplace domain. The equation is $\boldsymbol{\Delta y}(s) = P_u \boldsymbol{\Delta u}(s) + P_d \boldsymbol{\Delta d}(s)$ where $\boldsymbol{\Delta y} = [\Delta y_1, \Delta y_2, \Delta y_3]^T$, $\boldsymbol{\Delta u} = [\Delta u_1, \Delta u_2, \Delta u_3]^T$, and $\boldsymbol{\Delta d} = [\Delta d_1, \Delta d_2, \Delta d_3]^T$.

$$P_u = \begin{bmatrix} \frac{-2.6443}{1971.2s+1} & \frac{6.9829}{1706.3s+1} & 0 \\ \frac{-4.4526 \cdot 10^{-2}}{1610s+1} & \frac{1.6649 \cdot 10^{-2}}{2392.8s+1} & 0 \\ 0 & 0 & \frac{-9.674 \cdot 10^{-3}(508.86s+1)}{(222.35s+1)(15.074s+1)} \end{bmatrix} \quad (5)$$

$$P_d = \begin{bmatrix} \frac{0.121}{1445.7s+1} & \frac{0.201}{1880.93s+1} & 0 \\ \frac{7.28 \cdot 10^{-3}}{815.89s+1} & \frac{-20.66 \cdot 10^{-3}}{204.26s+1} & 0 \\ 0 & \frac{0.205}{s} & \frac{0.05}{s} \end{bmatrix} \quad (6)$$

D. Hardware-in-the-loop Setup

The basic idea of the Hardware-in-the-loop simulation (HIL) technique is very simple. In order to test a piece of hardware, a special simulation model is used to generate test information (in this case the distillation unit simulation), which "fools" the hardware into behaving as though it is operating in a real environment [9]. HIL simulation is usually used when it is possible to model the dynamics of the process with good precision and fidelity and when experiments with the real process would implicate in an unnecessary cost

because of the time required to install and test the hardware. Besides, in many cases, it is a form of avoiding unnecessary risks in projects that can compromise the safety of people.

The HIL simulation can be divided in three basic parts [10]:

- A Controller Interface Device (CID). This device provides the interface from the embedded controller to the computer running a the simulation. In this case, CID is a software as will be explained later on.
- A software interface module to provide the linkage between the CID and the simulation program.
- A simulation engine that is responsible for generating the outputs of the process according to the inputs provided by the controller. In this experiment, this is provided by the software Aspen HYSYS®.

To make the HIL scheme, it is first necessary to be able to communicate with the software running the simulation. Thankfully, Aspen made it very easy by making its program compatible with OLE Automation, which is an inter-process communication mechanism created by Microsoft. OLE Automation makes it possible for one software application to manipulate objects implemented in another application, or to expose objects so they can be manipulated. Therefore, as the embedded system being develop will communicate with the process via industrial networks, it was only necessary to create a software CID to simulate the network between the process and the controller. In this experiment, a basic Ethernet protocol was developed to test the system. The data sent to the embedded controller are the set-points and current process and disturbances values. On the other hand, the controller sends to the CID the calculated values of the manipulated variables to be applied in the process.

E. Experiment's Parameters

The test conditions on this experiment are the following: (a) the process is in steady-state on the chosen operating point; (b) the wine mass flow will be changed to simulate a variation of $\pm 1.7\%$ in its production by the previous process (fermentation), this will be followed by changes in the phlegm and hydrated ethanol mass flows to compensate the variations of the wine input. The MPC controller will have to maintain the concentration of the output products at the values specified in section IV-C.

After some initial tuning and simulations with the linearized models, the parameters of the controller were chosen in such a way that the disturbances are rejected in less than 20 minutes, according to the simulations of the linearized model.

It is well known that one of the limitations of MPC algorithms is the necessity to solve an optimization problem, or the QP, at real-time. This prevents the application of MPC in several contexts, either because the computer technology needed to solve the QP within the sampling time is too expensive or simply infeasible, or because the computer code implementing the numerical solver causes software certification concerns, especially in safety critical applications [11]. So it is important to keep track of the time needed by the system to compute the control signal to ascertain its capabilities and have an estimation of what is the limit it could reach in terms of minimum sampling time, which will indicate what kind of process it can work with.

TABLE I

TESTS PARAMETERS AND TIME TAKEN TO COMPUTE THE CONTROL ACTIONS

	Test 1	Test 2	Test 3
Variables	35	35	50
Constraints	140	70	200
Optimization (s)	14.25/2.60	5.87/0.65	34.0/5.47
Others (s)	0.28/0.25	0.26/0.13	0.32/0.14
Total Time (s)	14.52/2.60	6.12/0.67	34.31/5.46

The time necessary to solve the QP is related to the number of unknown variables, i.e., the sum of the control horizons, and the number of constraints being considered. So, to test the system's computational capabilities, three different tests were made considering different control horizons and constraints.

In all tests the GPC algorithm was used with sampling time of 60 seconds. The weights were chosen as $Q_{u_1} = 2$, $Q_{u_2} = 4$ and $Q_{u_3} = 0.1$ for the control efforts, and $Q_{y_1} = Q_{y_2} = 0.01$ and $Q_{y_3} = 8$ for the errors. The prediction horizons are $N_{y_1} = N_{y_2} = 60$ for, respectively, the 2nd ethanol and stillage, and $N_{y_3} = 30$ for the hydrated ethanol. The constraints, when considered, are: (a) maximum and minimum values for the manipulated variables that are $\boldsymbol{U_{max}} = [121.1\,\text{kPa}, 75.38\,^o\text{C}, 90\,\%]$ and $\boldsymbol{U_{min}} = [117.1\,\text{kPa}, 69.38\,^o\text{C}, 10\,\%]$; (b) maximum slew rate that are $\Delta u_1 = \pm 1$ kPa/min, $\Delta u_2 = \pm 1\ ^o$C/min, and $\Delta u_3 = \pm 10\,\%$/min. The remainder of the control parameters are:

- Test 1: $N_{u_1} = N_{u_2} = 15$, $N_{u_3} = 5$ and constraints in the inputs and in its slew rates;
- Test 2: same as Test 1, but without the slew rates constraints;
- Test 3: $N_{u_1} = N_{u_2} = 20$, $N_{u_3} = 10$ and constraints in the inputs and in its slew rates.

V. RESULTS

In Fig. 3 and 4, it is possible to see, respectively, the process' outputs and inputs in each test. All configurations tested of the MPC controller manage to reject the disturbances in a similar fashion and according to the specifications. The effects of the disturbances in the 2nd ethanol are rejected after 20 minutes, as required, and the disturbances barely affect the hydrated ethanol. In the case of the stillage, all controllers manage to maintain the concentration below 0.04^oGL.

The most interesting results though, are the ones related to the embedded system time to compute the control signals. In Table I the times, measured in seconds, are separated in three categories: (a) Optimization, which is the time taken to solve the QP; (b) Others, related to the other operations (matrix products, free response calculation, etc.); and (c) Total, which is the sum of the other two. The times in Table

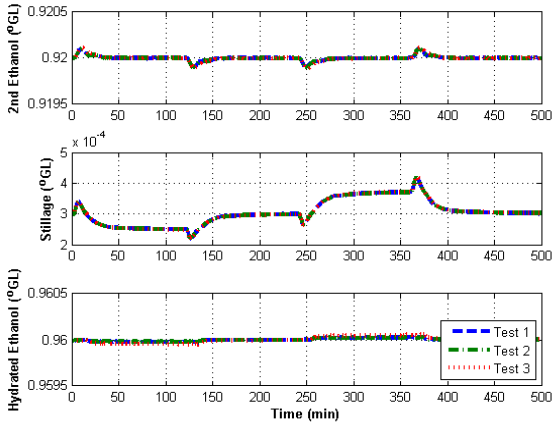

Fig. 3. Dynamics of the controlled variables during the different tests. The set-points for the 2nd ethanol and hydrated ethanol are, respectively, 0.92 and 0.96 ^{o}GL.

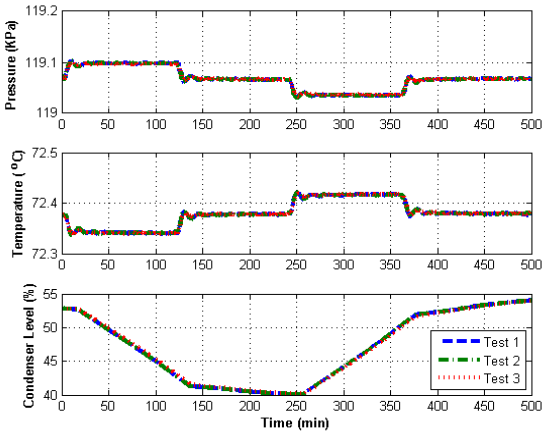

Fig. 4. Response of the manipulated variables during the different tests.

I are shown in the format $\bar{x}/\sigma$, where $\bar{x}$ is the mean and σ is the standard deviation, considering a gaussian distribution. In the table are also the number of unknown variables (UV) and constraints. The data collected can be used to verify if the prototype will be able to compute the control actions within the sampling period. Because of the non-deterministic nature of the optimization algorithm being used, it is not possible to know a priori the time needed to solve the QP. Therefore, a statistical approach is needed. By calculating the mean and the standard deviation, it is possible to give an estimation of the reliability of the prototype for a given system and control parameters. For instance, for the distillation process described, considering the worst case, which is test 3, it is possible to affirm that the prototype will compute the control action within 45.23 seconds with a statistical certainty of 95%. Thus, in this case, the prototype could be used.

Another interesting aspect to note is the correlation between the time taken to compute the control signals and the size of the QP, which is related to the number of unknown variables and constraints. For instance, halving the number of constraints from 140 to 70 (tests 1 and 2), yields a decrease of 60% in the optimization time. This characteristic can be used to give a greater certainty that the prototype will compute the control signals in time, i.e. it is possible to reduce the mean time by cutting down the control horizons and/or the constraints. Nevertheless, it is important to note that the reduction of the control horizons can be detrimental to the stability of the closed-loop system, so these type of alterations should be made with care.

VI. CONCLUSIONS

This paper showed the latest results concerning a new MPC product consisting of a low cost embedded system targeted at small and medium sized processes where the use of MPC could improve the quality of production. MPC is usually not adopted in these type of processes because investments in advanced control techniques are not currently economically viable. This new MPC product tries solve this by offering an easily integrated hardware component that supports many different communication protocols, which will make its installation easier and less expensive by using the already installed process' infrastructure.

Though the results obtained in the experiment described in this paper indicate the potential of the prototype being develop, there are some open problems. For example, the software is still incapable of handling the infactibility of the QP, i.e., it is not possible to find a solution considering the given restrictions. Finally, it is also noteworthy that the time needed to compute the control signals could be reduced if the optimization library was implemented in C or C++.

REFERENCES

[1] E. F. Camacho and C. Bordons, Model Predictive Control, Springer London, 1999.
[2] W.S. Levine, The Control Handbook, CRC Press and IEEE Press, 1996.
[3] K.V. Ling, S.P. Yue and J.M. Maciejowski, A FPGA implementation of model predictive control, American Control Conference, pp.6, 14-16 June 2006.
[4] A. Wills, A. Mills and B. Ninness, FPGA Implementation of an Interior-Point Solution for Linear Model Predictive, 18th IFAC World Congress 2011, Milan, Italy, pp 1-9, August 2011.
[5] G. Valencia-Palomo and J.A. Rossiter, Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information, ISA Transactions, Volume 50, Issue 1, pp. 92-100, January 2011.
[6] J.E. Normey-Rico and E.F. Camacho, Control of Dead-Time Processes, Series Advanced Textbooks in Control and Signal Processing, Springer London, 2007.
[7] J. Berge, Fieldbuses for Process Control: Engineering, Operation, and Maintenance, 1st ed., ISA - The Instrumentation, Systems, and Automation Society, 2002.
[8] M. V. Americano da Costa, D. M. Cruz and J. E. Normey-Rico, Modelling, Simulation and Control of a Distillation Unit in an Alcohol production Plant (in Portuguese), XIX Congresso Brasileiro de Automática, Campina Grande-PB-Brazil, 2012.
[9] Zhen Li, M. Kyte and B. Johnson, Hardware-in-the-loop real-time simulation interface software design, Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference, pp. 1012-1017, October 2004.
[10] D. Bullock, B. Johnson, R. B. Wells, M. Kyte and Zhen Li, Hardware-in-the-loop simulation, Transportation Research Part C: Emerging Technologies, Volume 12, Issue 1, pp. 73-89, February 2004.
[11] A. Alessio and A. Bemporad, "A survey on explicit model predictive control", in Nonlinear Model Predictive Control, pp. 345-369, Springer-Verlag, 2009.