Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-97046

EMBEDDED RIOTS - MODEL PREDICTIVE CONTROL TOWARDS EDGE

Jairo Viola
MESA Lab
School of Engineering
University of California, Merced
Merced, California, USA
Email: jviola@ucmerced.edu

Sina Dehghan
MESA Lab
School of Engineering
University of California, Merced
Merced, California, USA
Email: sdehghan@ucmerced.edu

YangQuan Chen
MESA Lab
School of Engineering
University of California, Merced
Merced, California, USA
Email: ychen53@ucmerced.edu

ABSTRACT

RIOTS is a general purpose optimal problem solver written as a MATLAB toolbox with mixed-language programming (C, Fortran, Matlab, Simulink). This first paper introduces how to make RIOTS run under an embedded platform RP3B (Raspberry Pi 3 B) with Windows 10. We presented the system architecture and a complete demo on running RIOTS as the inner kernel for MPC, using a house made thermal control system based on Peltier modules.

INTRODUCTION

Model predictive control (MPC) is a control strategy employed on different applications as oil refining [1], chemical process [2], medical applications [3], robotics [4] among others. Its main feature is allowing to obtain an optimal control action into a prediction horizon based on a model of the system that could be determined by experimental data with a certain number of constraints. Other features of the MPC control are that it allows handling large-scale multivariable, constrained, and non-minimum phase systems. An important step during the MPC control is the solution to the optimization problem to obtain an optimal control action. For this reason, different optimization tools can be employed for solving the MPC control problem as the Matlab MPC control toolbox [5] or the fast MPC library [6]. One of these tools is the Recursive Integration Optimal Trajectory Solver (RIOTS) [7], which solve linear and nonlinear optimization problems based on Matlab and C++ libraries. So that, the RIOTS optimization toolbox has been employed to perform MPC control on different systems. For example, the RIOTS MPC control is employed to perform the position control of a DC motor [8], an HVAC thermal system [9], and even it has been employed to solve the MPC problem for fractional order systems [10], but the complexity of the MPC optimization and control problem difficult performing an embedded version of this technique.

On the other hand, the rising of edge computing, bring the possibility of employing low-cost small computer devices like the Raspberry Pi to add more computing power directly on a control loop. Thus, it is possible running advanced algorithms as Artificial Intelligence, Machine Learning, Data Analytics or optimization toolbox running on real-time based on the process data to improve its performance.

In this paper, an embedded implementation of RIOTS toolbox is performed employing a Raspberry PI 3B low-cost computer as edge device that runs Windows 10 as a native operating system. This implementation is used to perform a SISO MPC temperature control of a Peltier module using a thermal infrared camera as a feedback sensor. The RIOTS MPC control runs on Matlab-Simulink setup as hardware in the loop (HIL) configuration with an Arduino board to receive the infrared thermal camera measures and send the control action to the Peltier module. For the RIOTS MPC control, a state space model of the Peltier system is identified, which works as a reference model for the optimization algorithm. The control strategy is validated on the house made thermal system.

The main contribution of this paper is the embedded imple-

mentation of the RIOTS toolbox optimal problem solver using a low-cost edge device computer like the Raspberry Pi 3B to perform a SISO temperature MPC control of a thermal system using visual feedback.

The paper is structured as follows. Initially, The MPC problem and the RIOTS toolbox are presented. Then, the embedded system architecture for the RIOTS edge implementation and the house-made thermal system are described. After that, the RIOTS MPC controller design and the HIL implementation are shown for the thermal system. Finally, conclusions are presented.

MODEL PREDICTIVE CONTROL

MPC is an advanced method for process control that allows controlling a system satisfying a set of constrains. This technique is also known as receding horizon control (RHC), which looks to obtain a control signal that minimize a cost function for some system constraints based on a future prediction of the system behavior. Thus, the MPC control has three components, the predictive behavior based on a system model, the optimization based on a cost function, and a receding horizon where the control input is updated at each sample time step. For MPC control, state-space models are widely employed, which is given by (1)

$$\begin{aligned} x(i+1) &= Ax(i) + Bu(i) + w(i) \\ y(i) &= Cx(i) + Du(i) + v(i) \end{aligned} \tag{1}$$

where $x(i)$ is the states of the system, $u(i)$ is the input, $y(i)$ is the system output, A, B, C, D are the linear state-space model matrices, $w(i)$ and $v(i)$ are the state and feedback noises. From the state-space model (1), the future of the system behavior moves through the prediction horizon, and is estimated using only the current state values $x(i)$. Therefore, for the next predicted state $\hat{x}(i+1)$, the system output $\hat{y}(i+k|i)$ can be estimated by all the k steps of the prediction horizon N_p as given by (2).

$$\begin{aligned} \hat{x}(i+k+1|i) &= A\hat{x}(i+k|i) + Bu(i+k|i) \\ \hat{y}(i+k|i) &= C\hat{x}(i+k|i) + Du(i+k|i). \end{aligned} \tag{2}$$

Finally, the optimization problem can be defined by the cost function J given by (3),

$$J = \sum_{k=1}^{N_p} [\hat{y}(i+k|i) - r(i+k)]^T W_y [\hat{y}(i+k|i) - r(i+k)] \tag{3}$$

where $r(i+k)$ is the reference signal, W_y is the positive definite weight matrix, and the system constrains are given by (4):

$$\begin{aligned} u_min < u(i) < u_max \\ x_min < x(i) < x_max \\ y_min < y(i) < y_max. \end{aligned} \tag{4}$$

The control action is calculated for each step of the horizon prediction based on (3) and (4), taking as control action to be applied the first one calculated among the prediction horizon.

RIOTS TOOLBOX

RIOTS is a Matlab toolbox developed by [7], that solve different linear and nonlinear optimization problems. The optimization problem solved by RIOTS is given by:

$$\min_{(\mu,\eta)\in L_\infty^m X R^n} f(\mu,\eta) = g_0(\eta, x(b)) + \int_a^b l_0(t,x,u)dt. \tag{5}$$

subject to the constrains:

$$\begin{aligned} \hat{x} = h(t,x,u), x(a) = \xi, t \in [ab] \\ u_min^j(t) < u^j(t) < u_max^j(t) \\ \xi_min^j(t) < \xi^j(t) < \xi_max^j(t) \\ l_t i^v(t, x(t), u(t)) \leq 0, v \in Q_t i \\ g_e i^v(\xi, x(b)) \leq 0, v \in Q_e i \\ g_e e^v(\xi, x(b)) = 0, v \in Q_e e \end{aligned} \tag{6}$$

where $x(t) \in R_x^n$, $u(t) \in R_u^n$, $g : R_x^n \, X \, R_x^n \to R$, $l : R \, X \, R_x^n \, X \, R_u^n \to R$, $h : R \; X \; R_x^n \; X \; R_u^n \to R_x^n$. The functions $g(\cdot,\cdot)$ and $l(\cdot,\cdot,\cdot)$ are subscripted with o, ti, ei, and ee, each of which stands for objective function, trajectory constraint, endpoint inequality constraint, and endpoint equality constraint respectively. Depending on the nature of the optimal control problem, it can be solved for both the optimal control u and one or more optimal initial state ξ. For the MPC control problem, the RIOTS toolbox can be integrated for solving the optimization problem. Thus, it is possible to integrate into a closed loop for controlling real systems, as can be observed on [8]- [10].

EDGE COMPUTING

Edge computing is defined as a computing paradigm, which decentralizes the information management and processing, putting more memory and processing capabilities close to the source of information in order to reduce the load of the central processing units [11]. This concept is applicable on process

control, where more computing power can be installed directly over the process to be controlled in order to perform data analytics, machine learning and artificial intelligence (AI) tasks that improve the system performance.

It has some essential features that make very valuable for the industry. One of them is reducing the intermittency in connectivity as well as the bandwidtth usagge for cloud transfering data. Thus, its possible to offering real time data about the system behavior without depending on a central server in the cloud. A meaningful use of edge computing is performing real-time predictive maintenance over the process monitored, reducing costs associated with unexpected system stops, improving the system reliability and power consumption.

In general, any device like a smartphone or laptop computer that perform the monitoring and processing tasks in the site of the process can be considered as an edge computing device. This is possible because it has more processing power than embedded devices to perform data analytics and run more complex algorithms. However, the concept of the edge computing device can be applied to the low-cost general purpose computers like the Raspberry Pi, which runs under Linux OS, and has enough computer power to perform AI and data analytics tasks. Moreover, considering the size and power consumption of these devices, make it more suitable for integration directly on a process in order to improve its performance based on the real-time behavior analysis of the system.

Thus, this paper employs the Raspberry Pi 3B as edge computing device to perform the MPC control problem using the RIOTS optimal problem solver, which is a complex optimization algorithm to perform the temperature control of a SISO Peltier system.

EMBEDDED RIOTS IMPLEMENTATION

In this paper, an implementation of the RIOTS optimal problem solver is performed employing a Raspberry Pi 3B working with Windows 10 as a native operating system, which runs Matlab-Simulink set in hardware in the loop configuration with an Arduino board to perform the MPC control of the temperature on a Peltier module using the RIOTS toolbox. A house-made system was built for this purpose, which is presented on Fig.1. As can be observed, the system is composed by the Peltier module (P1) as heating element, the infrared thermal camera (P2), the Raspberry Pi running Windows 10 (P3) with Matlab-Simulink configured on HIL with an Arduino and a power driver to manage the power applied to the Peltier module (P4).

Windows 10 running on Raspberry Pi

One of the main challenges for the embedded implementation of the RIOTS toolbox for MPC control is the fact that RIOTS runs only on Matlab, which requires a computer running on Windows or Linux Operating systems (OS) compatible with the x86 architecture. Although most embedded devices support Linux OS, these devices have an ARM-based microprocessor, which is not compatible with x86 architecture programs as Matlab. An alternative to solve this situation is installing a virtual machine that runs a Windows OS on the embedded device, but this virtual machine does not have a good performance because it is a host of the Linux OS and should share the limited embedded system resources with the host OS. Another alternative is employing Windows 10 IoT core for embedded devices, but this is a limited windows version focused on IoT only without the possibility of running windows programs.

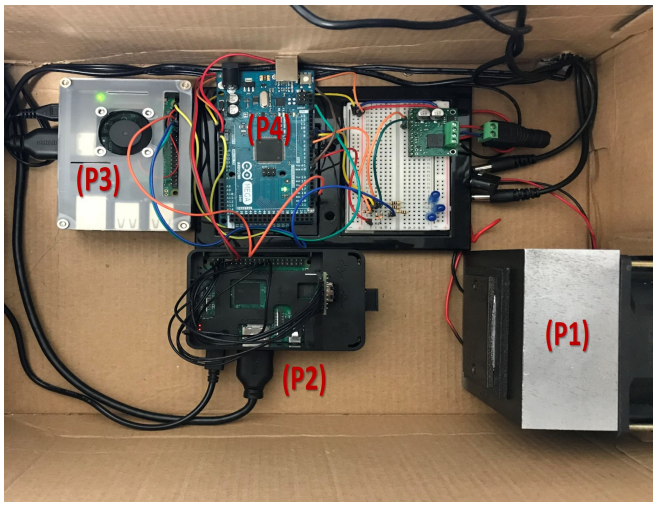


FIGURE 1: HOUSE MADE TEMPERATURE SYSTEM

Recently, the team of Windows on Raspberry Pi project (WORProject), release a Windows 10 version that runs as a native OS on the Raspberry Pi 3B and 3B+ [12]. This is a full 64 bits Windows 10 version based on the Windows 10 ARM developed for the Windows Surface, making possible running Windows programs as MS-Office or Matlab on the Raspberry Pi. Notice that because this Windows 10 version runs as a native OS on the Raspberry Pi, all the resources of this embedded device can be employed, and are not shared with the host Linux OS as happen with a virtual machine. Therefore, there is the possibility of running Matlab-Simulink configured as HIL device for running RIOTS MPC. On the other hand, due to the WORProject was released on May 2018, there are still many drivers and functionalities under development, so that, the connectivity of the Raspberry PI is limited by Serial communication with HIL devices as Arduino. In this implementation, the serial Communication is employed to share the information between the RIOTS MPC running on the Windows 10 Raspberry Pi, the thermal feedback camera and the power driver that controls the Peltier module. Also, for this embedded implementation, the Matlab-Simulink version employed is the 2007b 64 bits because the total amount

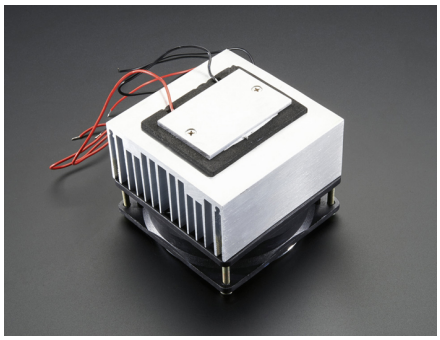


FIGURE 2: PELTIER MODULE

of RAM on the Raspberry Pi is 948 MB, and for most recent versions of Matlab the minimum amount of RAM required is at least 1024 MB. However, this version has all the essential toolboxes and interface resources for running RIOTS MPC on HIL configuration.

Peltier module

Peltier is a particular case of the thermoelectric effect that produces a temperature change at an electrified junction of two different conductors. If an electric current across the junction, it generates a temperature change, which according to the direction of the current could produce a heating or cooling effect. The Peltier module shown in Fig.2 is employed as a heating element because it is a solid state device with low maintenance requirements and long service lifetime. The temperature range for this device comes from 15^o to 50^oC, with maximum heating of $60W$ and a power requirement of $12v$ and $5A$. In this application, the power on the Peltier system is controlled using pulse width modulation (PWM) with the Arduino board working on the range of -255to255. Besides, the Peltier module includes a heat sink and an external fan to release the extra power produced on the back side of the Peltier plate to extend its lifetime.

Thermal infrared camera

Fig.3 shows the thermal infrared camera employed as a temperature sensor for this system. Manufactured by FLIR [13], it is a long wave infrared camera that measures the temperature over a surface through its emitted infrared radiation. The wavelength range for this camera comes from $8\mu m$ to $14\mu m$ with a maximum frame rate of 9 FPS. The camera has a resolution of 80x60 pixels with an accuracy of $\pm 0.5^oC$, and its size is less than a quarter coin. Besides, the camera has an SPI interface allowing its connection with many embedded devices using the LeptonThread software development kit, which is available for the camera data acquisition and runs in Python and C++. Notice that employing a thermal camera, each pixel acts as a temperature sensor, giving a big picture of the dynamical behavior of the temperature on the Peltier module surface. In this house-made thermal system,

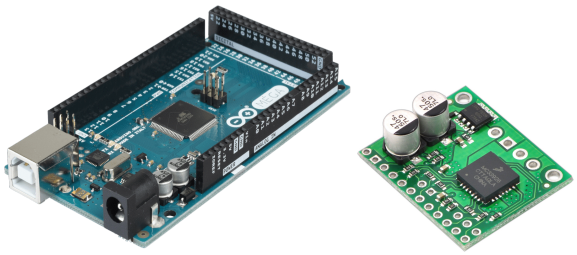
FIGURE 3: FLIR LEPTON THERMAL CAMERA

FIGURE 4: THERMAL IMAGE OF THE PELTIER MODULE WITH TEMPERATURE VALUES

the infrared camera is connected to a Raspberry Pi 3B+, which reads the infrared camera through the SPI interface and send the temperature data to the Raspberry Pi running Matlab-Simulink employing serial communication. Fig.4 shows a thermal image of the Peltier module at 25^oC acquired with the infrared camera running on raspberry PI 3B. As can be observed, the thermal image on the left is colored according to the temperature changes, and the console in the right shows the temperature values of some selected points on the image.

Arduino as HIL device

An Arduino MEGA board shown on Fig.5a is used in this platform to exchange the information between the thermal camera, the Matlab-Simulink control algorithm, and the Peltier module. This board acts as HIL device, receiving the temperature feedback of the camera, send it to the Raspberry Pi working on Windows 10 with Matlab-Simulink, and applying the control action generated by the RIOTS MPC controller to the Peltier module given as a PWM signal from Matlab-Simulink through serial communication. The use of Arduino as interface element results from the real driver restrictions of the Raspberry Pi running Windows 10, which is limited only to serial port communication. On the other hand, an MC33926 DC motor driver shown in Fig.4b is employed as a power interface to control the Peltier module using the Arduino. The driver works in a rage of $5V$to$28V$ with a maximum current supply of $3A$.



(a) (b)

FIGURE 5: ARDUINO MEGA AND MC33926 MOTOR DRIVER

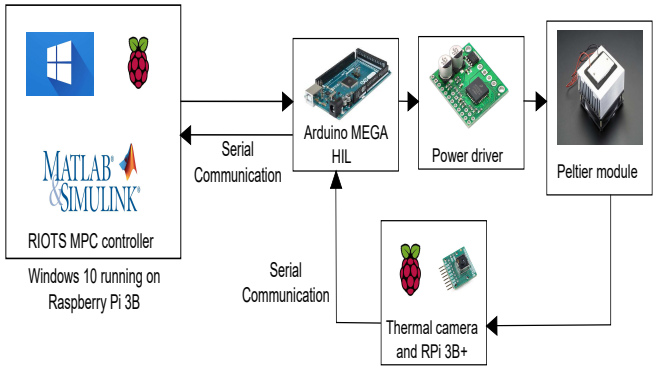


FIGURE 6: TEMPERATURE SYSTEM CLOSED LOOP CONFIGURATION

Closed-loop System configuration

Figure 6 shows the block diagram of the closed-loop system configuration used for the temperature control of the Peltier module employing the RIOTS MPC algorithm. As can be observed, the system temperature feedback is captured using the thermal camera and sent to the Arduino using serial communication. Then, the data is sent to the Raspberry Pi running windows 10 with Matlab-Simulink, and the MPC RIOTS control running on HIL to produce the control action. Once it is calculated, the control action is sent to the Arduino using serial communication, to be applied as PWM to the Peltier module.

System identification

The RIOTS MPC controller requires a state space model to made the system behavior prediction. So that, the identification signal shown in Fig.7 is applied to the system, which applies the maximum power to the Peltier module by 250s, and then power off the module to analyze its natural cooling response with an environment temperature of 22^0C. The data acquisition of the system uses the HIL closed-loop architecture described above with the following Matlab-Simulink interface shown Fig.7. As can be observed, the interface receives the visual feedback data from the thermal camera through the serial port. This data corresponds to the temperature in the center of the Peltier module

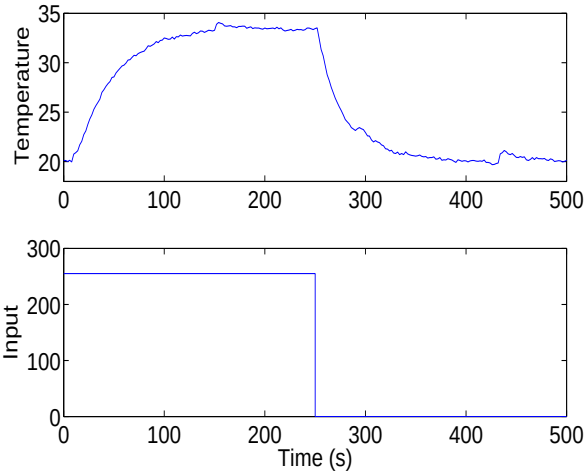
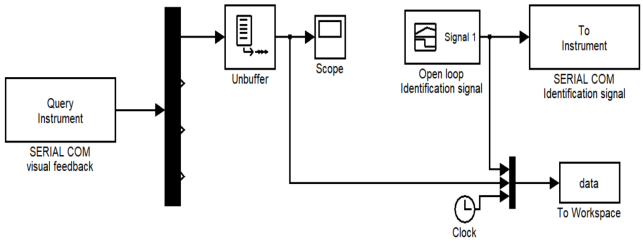


FIGURE 7: TEMPERATURE SYSTEM CLOSED LOOP CONFIGURATION



(a)

FIGURE 8: MATLAB-SIMULINK HIL DATA ACQUISITION INTERFACE

plate. Simultaneously, the identification signal is transmitted by the serial port. The sampling time employed for this system is two seconds. With the acquired data and employing the Matlab Identification toolbox, a second order linear state-space model is determined with a fit of 70% given by (7).

$$\dot{x} = \begin{bmatrix} 0.0017 & -0.0009 \\ 0.18 & -0.0299 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 36.2 \\ 896.6 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 148.99 & -5.95 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0u(t). \quad (7)$$

RIOTS MPC controller design

Considering the system is modeled using a state-space model given by (7), the full state data of the system should be available. So, the states should be directly measured or estimated using a state observer. In this case, the system has a second order state space model with only one feedback signal. For this reason, a state observer is required. According to [7], the Luenberger observer can be employed for the state estimation $\hat{x}$, which is given

by (8).

$$\dot{\hat{x}} = (A - GC)\hat{x} + Gy + Bu \tag{8}$$

where G is a gain matrix designed to place the eigenvalues of $A - GC$ at [-1,-2,-3,-4]. The cost function for the RIOTS MPC controller is presented on (9)

$$J = y((N_p) - r)^T W_y(y(N_p) - r) + \int_0^{N_p} ((y(N_p)) - r)^T W_y(y(N_p) - r)dt \tag{9}$$

where N_p is the prediction horizon, $y(N_p)$ is the system output at the instant N_p, and W_y is the weight matrix for the system output error. For this system, the prediction horizon N_p and the weight matrix W_y are defined as

$$W_y = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} \quad N_p = 20. \tag{10}$$

Matlab-Simulink HIL RIOTS MPC controller implementation

The HIL implementation of the RIOTS MPC controller using Matlab-Simulink running on a Raspberry Pi with Windows 10 as native OS is presented on Fig.9. As can be observed, the implementation is divided into three sections, the HIL data acquisition (red square), the state observer based on the system feedback loop (blue square) and the RIOTS MPC controller with the temperature reference for the system (green square).

Fig.10a and Fig.10b show the time response and the control action of the RIOTS MPC controller for a temperature setpoint of 25^oC. As can be observed, the Peltier module reaches the desired temperature in about 170s without overshoot. Besides, the control action starts giving the maximum power to the Peltier module. After that, the power applied to the Peltier module is reduced up to reach a constant value that maintains the desired temperature.

CONCLUSIONS

This paper presented an embedded implementation of the RIOTS optimal problem solver to perform an MPC control on a Peltier temperature system. The implementation is performed on a raspberry PI 3B running Windows 10 as a native operating system, which runs Matlab-Simulink using the HIL configuration to perform the data acquisition and control tasks with a thermal infrared camera as a temperature sensor. The obtained results show that the RIOTS MPC control embedded implementation on Raspberry Pi performs the control tasks adequately with a good

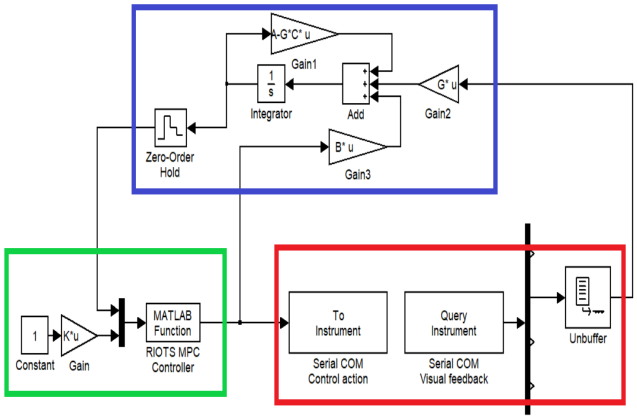


FIGURE 9: RIOTS MPC CONTROLLER ON HIL CONFIGURATION ON MATLAB SIMULINK

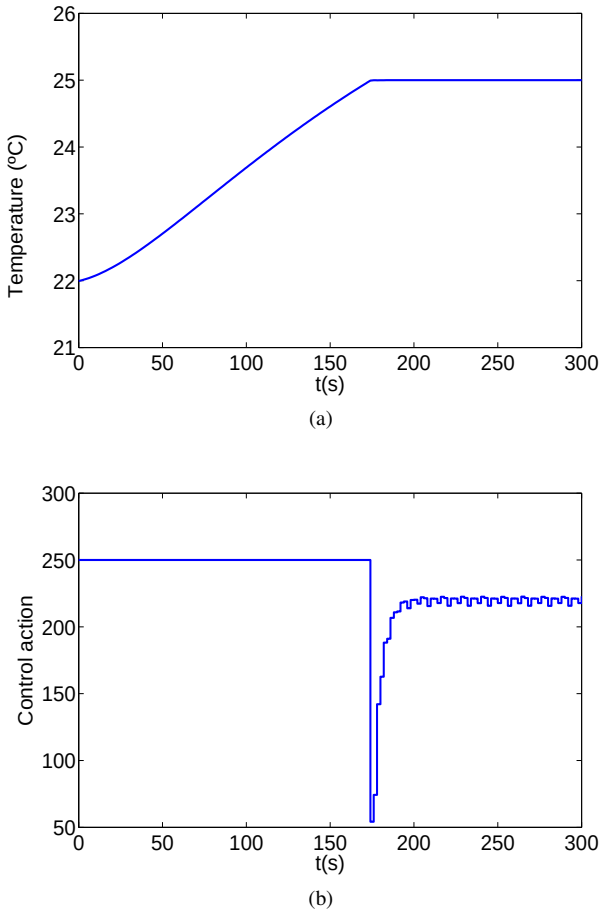


FIGURE 10: RIOTS MPC TEMPERATURE CONTROL A) TIME RESPONSE AND B) CONTROL ACTION

time response and control action. Besides, the possibility of running Windows 10 on a Raspberry PI as native OS brings windows OS to the embedded devices, giving the opportunity for testing and implement advanced control techniques as the RIOTS MPC or even adaptive or nonlinear controllers within an embedded device using standard Matlab-Simulink tools.

ACKNOWLEDGMENT

The authors are thankful to the members of PTUC/CPC (precision temperature uniformity control / cognitive process control) research group at UC Merced MESA Lab: Alberto Radici and Jie Yuan contribution to this work.

REFERENCES

[1] Zhang S, Zhao D, Spurgeon S. K. and X. Yan, "Distributed model predictive control for the atmospheric and vacuum distillation towers in a petroleum refining process" *2016 UKACC Int. Conf. Control. UKACC Control*, pp. 1–6, 2016.

[2] Kamesh R. and Rani K. Y, "Novel Formulation of Adaptive MPC as EKF Using ANN Model: Multiproduct Semibatch Polymerization Reactor Case Study," *IEEE Trans. Neural Networks Learn. Syst*, Vol. 28, no. 12 (2017) pp. 3061–3073.

[3] Czakó, B, Sápi, J, and Kovács L., "Model-based optimal control method for cancer treatment using model predictive control and robust fixed point method," *IEEE 21st Int. Conf. Intell. Eng. Syst. Proc*, pp. 271–276, 2017.

[4] Hafez A. T, Marasco A. J., Givigi S. N., Iskandarani M., Yousefi S., and Rabbath C. A. "Solving Multi-UAV Dynamic Encirclement via Model Predictive Control," *IEEE Trans. Control Syst. Technol*, vol. 23, no. 6 (2015), pp. 2251–2265.

[5] MathWorks inc, 2018, "Model Predictive Control Toolbox,". last modified February 28, 2019, accessed March 19, 2019. https://www.mathworks.com/products/mpc.html.

[6] Wang Y. and Stephen B., 2018 "fast_mpc: code for fast model predictive control." last modified May 23, 2019, accessed May 23, 2019, http://web.stanford.edu/boyd/fast_mpc.html.

[7] Schwartz A., Polak E., and Chen Y. Q., "RIOTS-A Matlab Toolbox for Solving General Optimal Control Problems." last modified May 23, 2019, accessed May 23, 2019 http://mechatronics.ucmerced.edu/RIOTS.

[8] Chen Y.Q.and Tricaud C "Linear and Nonlinear Model Predictive Control Using A General Purpose Optimal Control Problem Solver RIOTS_95" *2008 Chinese Control and Decision Conference (CCDC 2008)* pp. 1552-1557, 2008.

[9] Dehghan S., Zhao T., Zhao Y., Yuan J., Ates A., and Chen Y. Q., "PID2018 Benchmark Challenge: Model Predictive Control With Conditional Integral Control Using A General Purpose Optimal Control Problem Solver – RIOTS.," *IFAC-PapersOnLine*, vol. 51, no. 4 (2018), pp. 882–887.

[10] Zhao T., Li Z., and Chen Y.Q, "Fractional order nonlinear model predictive control using RIOTS-95," *Int. Conf. Fract. Differ. Its Appl. ICFDA 2014*, no. 1, pp. 2–7, 2014.

[11] G.E Digital, "What is Edge Computing?," 2019. last modified January 10, 2019, accessed May 23, 2019 https://www.ge.com/digital/blog/what-edge-computing.

[12] Windows on Raspberry Pi Team, "Windows on Raspberry imager" last modified April 10, 2019, accessed May 23, 2019 Available: https://www.worproject.ml/.

[13] F. systems Inc, "FLIR LEPTON Infrared camera," last modified February 19, 2019, accessed May 24, 2019: https://goo.gl/RnKbKS