VOL. 19, NO. 3, 2020, 1-6
www.elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Real-time Digital Control of a Coupled-Tank Plant with a Cyber-Physical System Node

Rosbi bin Mamat*

School of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.
*Corresponding author: rosbi@fke.utm.my, Tel: 607-5535239, Fax: 607-5566262

Abstract: A real-time implementation of a digital proportional-integral controller on a Cyber-Physical Systems (CPS) node for a coupled-tank plant to provide a testbed for verification of control algorithm and communication is described. The CPS node embedded computer is based on the ADuC841 8-bit microcontroller and the embedded real-time multitasking software was written in Pascal language. The node is interfaced to the Quanser's coupled-tank plant via the on-chip 12-bit analog-to-digital and digital-to-analog converters, and a RS-485 communication network connects it to other nodes to form a CPS. Experimental results on the closed-loop control performances of the digital proportional-integral controller with three different tuning sets on the coupled-tank plant, verified the functionality of the digital control algorithm.

Keywords: Real-time digital control, Cyber-physical systems, Proportional-integral control, Coupled-tank plant.

Article History: received 25 October 2020; accepted 5 December 2020; published 26 December 2020.

1. INTRODUCTION

Cyber-Physical Systems (CPS) are integrations of networked embedded computers which interact dynamically in real-time with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa [1]. Applications of CPS includes automotive systems, industrial control systems, power grid, avionic systems, medical devices etc.

CPS consist of three main technology components: control, computation and communication – which is commonly referred as 3C technologies. The computation and communication components form the cyber-part of CPS, while the physical process is the physical part of CPS.

The CPS nodes located at the lowest layer of CPS typically are autonomous nodes contain the 3C technologies components. A CPS node consists of embedded computer interfaced to process or plant, and a communication network that connect it to other nodes or a server.

CPS node reads the plant states given by sensors through analog-to-digital converters (ADC), computes the control signals using some digital control algorithms and sends the control signals to actuators through digital-to-analog converters (DAC) to affect the plant. Information are shared between other nodes and with higher levels of CPS, and collaboratively they form global information synthesis to achieve common objectives. This is a main difference to the traditional embedded systems which were isolated and mainly concentrating on predefined functionalities within the overall system context [2].

One of the major reason for many badly tuned proportional-integral-derivative (PID) controllers in process industry is the lack of time to peform the controller tuning [3]. Consequently, many PID controllers operated in proportional-integral (PI) mode only, to ease the tuning process. Therefore, it is reasonable to regard PI controller as the main workhorse in process industry.

Due to the interaction with physical world, the dynamicity and the open-endedness of CPS, software testing and verification post a difficult challenge [4].

This paper describes a real-time implementation of a digital PI controller on a CPS node for a coupled-tank plant to provide a testbed for verification of control algorithm and communication at CPS node level and server-nodes level. The coupled-tank plant is a popular lab-scale plant frequently used in research and education [5, 6, 7].

The paper is organized as follows. In Section 2, the difference equation describing the digital PI controller is derived and the algorithm is presented. In Section 3 the hardware of the CPS node and the embedded software implementation aspects of the digital PI controller is described and discussed. Section 4 describes experimental setup with the coupled-tank plant. Section 5 presents some experimental results on real-time control of coupled-tank plant. Finally, the conclusion is drawn in Section 6.

2. DIGITAL PI CONTROLLER

In order to implement a digital PI controller, the standard continuous-time PI controller need to be discretized or digitized with the aim to obtain an algorithm in a difference equation form suitable for implementation in software.

2.1 Discretization of Continuous-time PI Controller

Consider a standard feedback control structure as shown in Figure 1, where $G_c(s)$ is the PI controller transfer function

given by Equation (1), $U(s)$ is the control signal, $E(s)$ is the error signal, $Y(s)$ is the plant/process output and $R(s)$ is the set point. The proportional gain K_c and the integral time-constant T_i are tuning constants obtained through a design process or from tuning rules.

$$G_c(s) = \frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i s}\right) \tag{1}$$

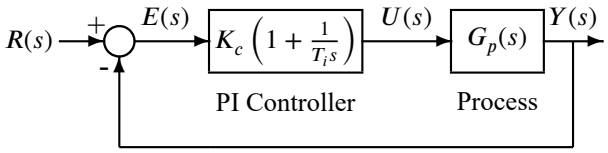

Figure 1. Block diagram of a standard feedback control system with PI controller

To obtain the digital PI controller algorithm, the PI transfer function in Equation (1) was discretized by approximating the derivative (s) in Equation (1) using the Trapezoidal method given by Equation (2), where h is the sampling period and z^{-1} is the back-shift operator. This approximation is also called Tustin's approximation or the bilinear transformation [8].

$$s \approx \frac{2(1 - z^{-1})}{h(1 + z^{-1})} \tag{2}$$

Substituting Equation (2) into Equation (1), Equation (3) is obtained, where $u_k = u(kh)$ and $e_k = e(kh)$ denote the discrete control and error signals respectively, at time $t = kh$.

$$\frac{u_k}{e_k} = K_c\left[\frac{2T_i(1 - z^{-1}) + h(1 + z^{-1})}{2T_i(1 - z^{-1})}\right] \tag{3}$$

By noting the operation of the back-shift operator such that $e_k z^{-1} = e_{k-1}$ and $u_k z^{-1} = u_{k-1}$, Equation (3) can be simplified to:

$$u_k = u_{k-1} + \gamma_1 e_k + \gamma_2 e_{k-1} \tag{4}$$

where,

$$\gamma_1 = K_c\left(1 + \frac{h}{2T_i}\right) \quad \text{and} \quad \gamma_2 = K_c\left(1 - \frac{h}{2T_i}\right)$$

Equation (4) is the difference equation that gives the digital PI control algorithm in a velocity form. To have a good approximation with continuous-time PI controller behaviour, in the discretization process the sampling period h was chosen to be 100 ms, as commonly used in many industrial controllers.

```
Require: Kc, Ti, h and current set-point rk
Initialized: γ1, γ1, uk−1 = 0, ek−1 = 0
1: loop
2:    yk = readADC      // read current process output
3:    ek = rk − yk      // calc error signal
4:    compute uk using Equation (4)
5:    if uk > MAX_VALUE  then
6:       uk = MAX_VALUE
7:    else if uk < MIN_VALUE  then
8:       uk = MIN_VALUE
9:    end if
10:    writeDAC(uk)     // write control signal
11:    uk−1 = uk        // update signals for next sampling
12:    ek−1 = ek        //  period
13:    Wait for next sampling period
14: end loop
```

Figure 2. Digital PI control algorithm for implementation

2.2 Digital PI Controller Algorithm

Figure 2 shows the digital PI controller algorithm for software implementation which will be executed every sampling period. The algorithm requires that the tuning constants K_c and T_i, and the set-point r_k are available, possibly transmitted over the network from the master node. The sampling period h is fixed at 100 ms.

The process output is read from the sensor using the `readADC` function and the control signal is sent to the actuator through the `writeDAC` function. Note that, γ_1 and γ_2 are precomputed outside the control loop to speed-up the control loop execution. Note also, lines 5 – 9 implement control signal clamping so that the control signal is within the predefined minimum and maximum values.

3. CPS NODE IMPLEMENTATION

In this section, the hardware and software aspects of the CPS node implementation are briefly described by highlighting the 3C technologies involve in the node. The node hardware consists of embedded computer, plant and network interfaces, while the software involves real-time multitasking embedded software for control and RS-485 communication.

3.1 CPS Node Hardware Prototype

The embedded computer is based on the ADuC841 8-bit microcontroller manufactured by Analog Devices Inc. [9]. This is an optimized 8052 single-cycle core with peak performance of up to 20 MIPS, integrated with 8-channel high accuracy 12-bit ADC and two 12-bit DAC, suitable for digital control. There are other on-chip peripherals such as Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI), timer and General Purpose Input/Output (GPIO) which were utilised in the node implementation.

The node hardware prototype was built on the ADuC841 Evaluation Board from Analog Devices as shown at the bottom of the circuit board in Figure 3. The RS-485 [10] interface is industry's most widely used balanced transmission-line standard, enables the node to be networked. The 4-bit binary switch can be used to set the node address on this network, and this address is displayed on the 7-segments LED display in hexadecimal. Note that, even though the RS-485 standard allows up to 64 nodes, in this prototype it is limited to only 15 nodes on the network.

3.2 Real-time Embedded Software Implementation

The embedded software for the node was written in Pascal language and compiled with Turbo51 free Pascal compiler [11].

The important software modules and their dependencies on other modules and on the on-chip peripherals are shown in Figure 4. On the top layer, the CPS node embedded software consists of communication (`COMM`),

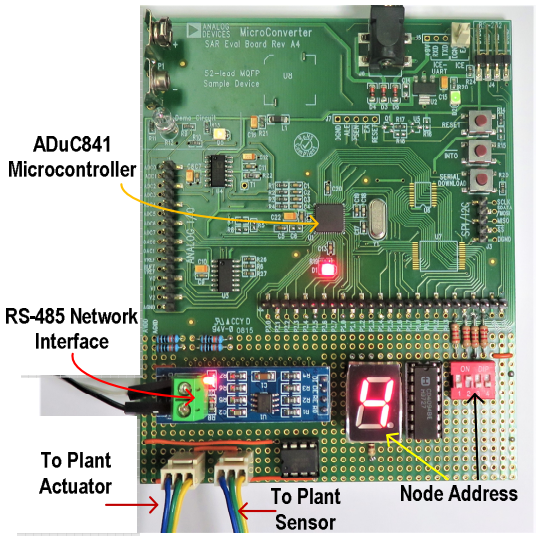

Figure 3. Prototype of the CPS node on the ADuC841 Evaluation Board

control (`CONTROL`), and user interface (`USER INTF`) tasks which are run in a mutitasking environment provided by the operating system module (`OS`).

The OS module basically is a small cooperative real-time kernel that provides the facilities for creating, scheduling and dispatching the software tasks with the help of timer in the ADuC841 chip.

Figure 5 shows an excerpt from the main program of the CPS node. Lines 10 – 12 show how the procedures `loop0` in `CONTROL` module, `task` in `COMM` module, and `heartBeat` are created to be multitasked by the OS module in a periodic time-triggered manner. The period for execution of those modules are defined in lines 5 – 6. In this way, `CONTROL` module is executed every 100 ms, and the `COMM` module is executed every 250 ms. The `heartBeat` task is executed every two seconds used as the CPS software health indicator. These three tasks are scheduled and executed accordingly by `OsSchedule` in line 16.

The `COMM` module depends on reception (`RXPACK`), processing (`PROCPACK`) and transmission (`TXPACK`) of data packets, which eventually depend on UART serial interface to communicate on the RS-485 network. Data received from UART are buffered by interrupt service routine so that no received data is lost. The packet for communication consist of seven bytes data including a checksum byte which provides a rudimentary way for data integrity and error checking. The `PROCPACK` module also responsible to perform other commands in received packets such as setting and updating the PI parameters, change set points, read plant output etc.

The `CONTROL` module consists of procedure `loop0` which executes the PI control algorithm given in Figure 2. However, due to presence of noise in the level sensor readings of the coupled-tank plant, the PI algorithm was modified to include a first-order digital filter to reduce the noise in the sensor readings.

Equation (5) give the difference equation for implementing the noise filter where, f_k and f_{k-1} are current and previous filtered output, and y_k is the current plant output. α ranges 0 – 1 is used to select the filtering effect; $\alpha = 0$ ignores the unfiltered data, while $\alpha = 1$ turn-off filtering. In this implementation $\alpha = 0.9$ was used. With the inclusion of this noise filter, the error calculation in line 2 of Figure 2 is modified to $e_k = r_k - f_k$.

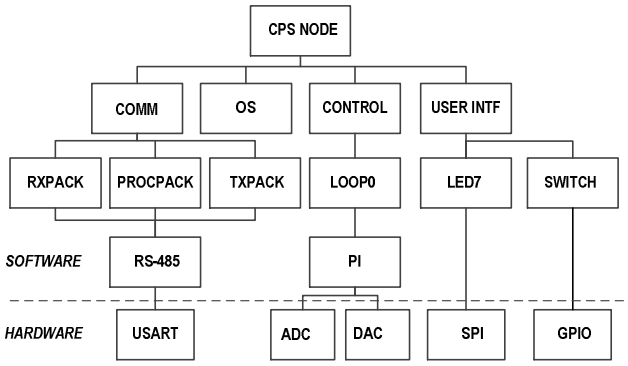

Figure 4. Software modules and on-chip peripherals dependencies in the CPS node

```
 1: program main;
 2: uses ...;                // include all the modules
 3:
 4: const
 5:   CTRL_period  = 100;
 6:   COMM_period  = 250;
 7:   OSTick_mS    =   5;  // OS tick in milliseconds
 8: begin
 9:   OSInit;
10:   OSCreateTask(Ctrl.loop0, CTRL_period div OSTick_mS, 0);
11:   OSCreateTask(Comm.task,  COMM_period div OSTick_mS, 1);
12:   OSCreateTask(heartBeat,  2000 div OSTick_mS, 2);
13:   OsRun;
14:   while True  do
15:     begin
16:       OsSchedule;
17:     end;
18: end.
```

Figure 5. Main program of the CPS embedded software

$$f_k = \alpha y_k + (1 - \alpha) f_{k-1} \tag{5}$$

4. EXPERIMENTAL SETUP

To test the functionality of the CPS node embedded software and for verification of digital control algorithm, the CPS node was used to control the water level of a coupled-tank plant manufactured by Quanser Inc. [12]. The experimental setup is shown in Figure 6.

4.1 Coupled-tank Plant Configuration

The Quanser's coupled-tank plant consists of a pump, two cylindrical tanks and a water basin which acts as a reservoir. The two tanks are vertically mounted on top of each other such that water from the top tank (Tank1), which is filled by the pump from the water basin, can flow to the bottom tank (Tank2) which eventually will flow into the water basin under gravity.

In this experiment, the plant is configured as a state-coupled SISO system (configuration 2) [12]. The controller is designed to regulate or track the water level in Tank2 by manipulating the pump, while Tank1 acts as buffer.

In each tank, water flow from the bottom through an outlet orifice. Outflow from the tanks, hence the plant dynamics, can be varied by using the different outlet inserts

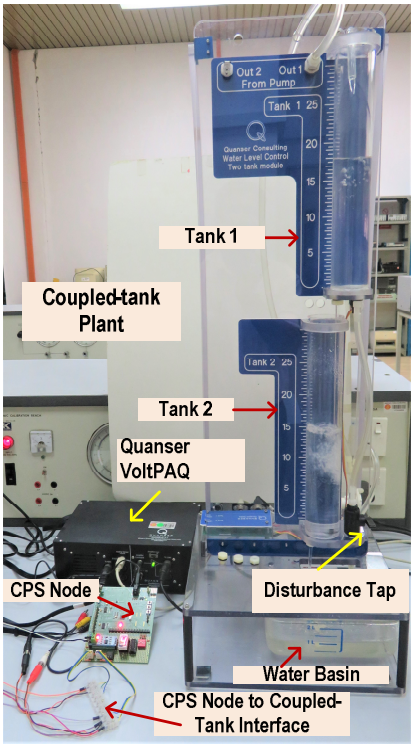

Figure 6. Coupled-tank plant and CPS node setup

which can be screwed into the tapped holes at the bottom of each tank. In this experiment, inserts with diameter of 0.476 cm (medium insert) were fitted in both tanks.

A disturbance tap, when opened, cause water from Tank1 flows directly to the water basin, is used to introduce disturbance flow into Tank2. The water level in each tank is measured by a pressure sensor located at the bottom of each tank which was calibrated to give a sensitivity of 6.1 cm/V. This sensitivity was used to convert the level readings from voltage to centimeters.

The Quanser VoltPAQ module provides the necessary instrumentations for the sensors, power supply and amplifier. The signals from the sensors and to the pump are 0 – 5 V compatible. The pump amplifier gain was set to 3 in this this experiment. The interfacing between the CPS node and the coupled-tank plant sensor and actuator is shown by the block-diagram in Figure 7.

4.2 Data Collection

The plant output $y(t)$, and the control signal $u(t)$ were measured directly with a data acquisition card equipped with 12-bit ADC and DAC on a PC. The sampling rate for data acquisition is two seconds. The PC also served as the master node for sending PI controller tuning parameters and set points to the CPS node via the RS-485 network.

5. EXPERIMENTAL RESULTS

Two sets of experiments were performed. In the first set, the coupled-tank plant model is estimated experimentally in the form of a first-order plus dead-time (FOPDT) model shown in Equation (6), where K is the process gain, θ is the delay and τ is the time-constant. This FOPDT model is used to calculate tuning settings for the PI controller using three different tuning rules. These three different sets of

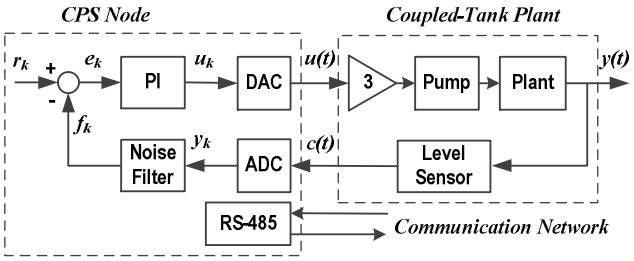

Figure 7. Interfacing the CPS node to coupled-tank plant

tunings serve as ways of obtaining three different closed-loop dynamics which can be used to access or verify the digital PI control algorithm.

In the second set of experiments, the closed-loop performances of the digital PI controller on level tracking and load disturbance rejection were recorded for three different sets of tunings. The load disturbance is introduced by turning the disturbance tap to half-open, which will cause some of the water in Tank1 flows directly to the water basin, consequently, produced a level drop in Tank2.

$$G_p(s) = \frac{Ke^{-\theta s}}{(1+\tau s)} \tag{6}$$

5.1 Plant Model Estimation and PI Tuning

Due to the nonlinear dynamics of coupled-tank plant, the plant model was estimated around an operating level of 15 cm to obtain a linearized model around 15 cm level. A step input voltage of 0.2 V ($u(t)$) was given to the pump amplifier and the level output voltage given by the level sensor ($c(t)$) was recorded in open-loop. From this open-loop response, the estimated FOPDT model for the coupled-tank plant is given by Equation (7). Where, the delay and time-constant are in seconds, and the unit for the process gain is Volt/Volt.

$$G_p(s) = \frac{C(s)}{U(s)} = \frac{2.3e^{-9s}}{(1+35s)} \tag{7}$$

This model includes the dynamics of the amplifier, pump, level sensor and water tanks. It will be used to calculate the PI controller tuning settings using three well-known PI tuning rules. Two tuning rules which were proposed in the third millennium, the Amigo [3] and the SIMC [13], and the classical Ziegler-Nichols [14] were selected. For a FOPDT process model in the form of Equation (6), the tuning rules for Amigo is given by Equation (8), SIMC by Equation (9) and Ziegler-Nichols by Equation (10).

$$\begin{aligned} K_c &= \frac{1}{K}\left(0.15 + 0.35\frac{\tau}{\theta} - \frac{\tau^2}{(\theta+\tau)^2}\right) \\ T_i &= 0.35\theta + \frac{13\theta\tau^2}{\tau^2 + 12\theta\tau + 7\theta^2} \end{aligned} \tag{8}$$

$$K_c = \frac{0.5\tau}{K\theta} \qquad T_i = \min(\tau, 8\theta) \tag{9}$$

$$K_c = \frac{0.9\tau}{K\theta} \qquad T_i = 3.3\theta \tag{10}$$

From the estimated FOPDT model in Equation (7) and tuning rules given by Equations (8) – (10), the PI settings

were calculated as tabulated in Table 1. These settings were used in evaluating the closed-loop performances of the digital PI controller.

Table 1. PI tuning settings calculated from three different tuning rules

Tuning Rules	K_c	T_i
Amigo	0.38	28.9
SIMC	0.85	35.0
Ziegler-Nichols	1.52	29.7

5.2 Closed-loop Responses Under Digital PI Control

The two performances observed in the closed-loop responses are set-point tracking and load disturbance rejection around the 15 cm level where the FOPDT model was estimated. For each PI tuning (K_c, T_i) in Table 1, set-point changes were given after the levels have reached their steady-states. Set-points were changed from 15 cm to 18.3 cm levels and then back to 15 cm. After this the load disturbance was introduced. The point at which the disturbance was introduced is indicated by an arrow in Figures 8 – 10.

Figure 8 shows the responses under PI control with Amigo tunings, where y is the water level and u is the control signal produced. The set-point tracking shows a good tracking with about 6% overshoot. The load disturbance caused the water level to drop by more than 3 cm, and it took about 250 seconds to fully reject the disturbance.

Figure 9 shows the responses under PI control with SIMC tunings. The set-point tracking shows a faster response compared to the Amigo tunings with about 14% overshoot. The load disturbance caused the water level to drop by 3 cm, and it took 200 seconds to fully reject the disturbance, faster than the one with Amigo tunings.

Figure 10 shows the responses under PI control with Ziegler-Nichols tunings. The set-point tracking shows a fast response with high overshoot (about 32%). This is where the Ziegler-Nichols tuning-rules are frequently abused. The tuning rules were meant for good load disturbance rejection, not for set-point tracking. As can be observed in Figure 10, Ziegler-Nichols tunings gave excellent response to load disturbance, where the water level dropped by 2 cm only, and it took 130 seconds to fully reject the disturbance, much quicker than PI controller with Amigo and SIMC tuning rules.

Observation of the responses produced by these tuning rules as shown in Figures 8 – 10, shows that this general behaviour is consistent with results already analyzed and reported elsewhere [3], [15]. These results verified the correctness and functionalty of the implemented digital PI controller.

6. CONCLUSION

An implementation of a digital proportional-integral (PI) controller on a Cyber-Physical Systems (CPS) node to provide a testbed for verification of control algorithm and communication was described. This shows that, it is possible to implement a real-time digital PI controller with

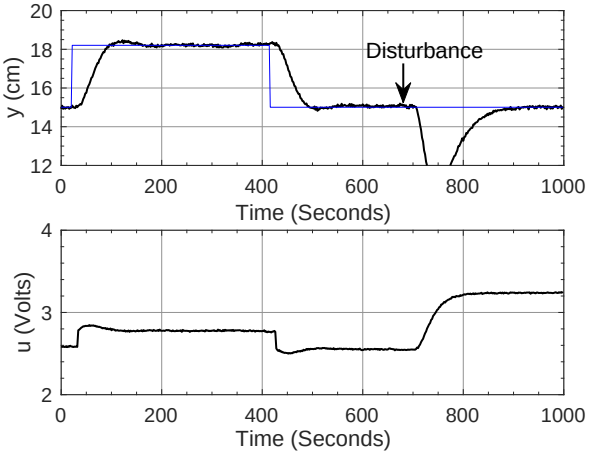

Figure 8. Closed-loop response under digital PI control with AMIGO tuning rules

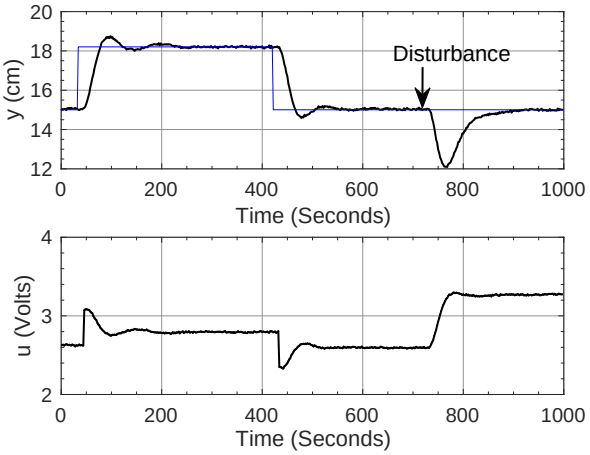

Figure 9. Closed-loop response under digital PI control with SIMC tuning rules

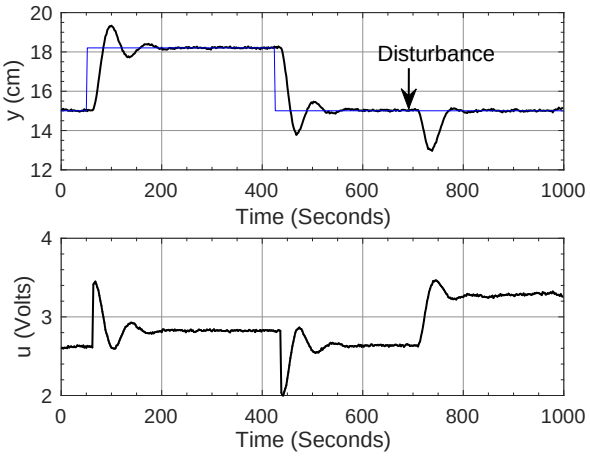

Figure 10. Closed-loop response under PI control with Ziegler-Nichols tuning rules

RS-485 networking facility using an 8-bit microcontroller such as ADuC841 with minimal hardware requirement.

The CPS node was used to control the Quanser's coupled-tank plant and the closed-loop performances were observed when the plant were under the digital PI control, tuned with three different tuning rules. The set-point tracking and load disturbance responses with those tunings show general behaviour which is consistent with results already analyzed and reported in literature. These results verified that: (i) the implemented digital PI controller functionalty is correct; and (ii) the estimated FOPDT process model is accurate enough to be used for calculating the PI tunings.

REFERENCES

[1] E. A. Lee, "Cyber Physical Systems: Design Challenges", *Proc. 11th IEEE Int'l Symp. On Object and Component-Oriented Real-Time Distributed Computing (ISORC 08)*, 2008, pp. 363–369.

[2] B. Rumpe, I.Schaefer, B. Schlingloff and A. Vogelsang, "Special issue on engineering collaborative embedded systems", in *SICS Software-Intensive Cyber-Physical Systems*, vol. 34, pp. 173–175, 2019.

[3] T. Hägglund, "The one-third rule for PI controller tuning", *Computer and Chemical Engineering*, vol. 127, pp. 25–30, 2019.

[4] T. Bures, D. Weyns et. al, "Software Engineering for Smart Cyber-Physical System – Towards a Research Agenda", *ACM SIGSOFT Software Engineering Notes*, vol. 40, pp. 28–32, November 2015.

[5] K. H. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero"', *IEEE Transactions on Control Systems Technology*, vol. 8, pp. 456 – 465, 2000.

[6] J. B. M. Santos, G. A. J'unior, H. C. Barroso and P. R. Barros, "A Flexible Laboratory-Scale Quadruple-Tank Coupled System for Control Education and Research Purposes", *Proc. 10th International Symposium on Process Systems Engineering - PSE2009*, 2009, pp. 2151-2156.

[7] J. Carrasco, W. P. Heath, M. C. R. Liñan, R. Alli-Oke, O. A. R. Abdel Kerim and S. R. Gutierrez, "Controlling a quadruple tanks rig with PLCs as a Masters dissertation project", *10th IFAC Symposium Advances in Control Education*, August 28-30, Sheffield, UK, 2013, pp. 238 – 243.

[8] K.J. Åström and B. Wittenmark, *Computer Controlled Systems, Theory and Design*, 2nd ed., Prentice-Hall, Englewood Cliffs., N.J., 1990.

[9] Analog Devices Inc., *aduc841_842_843 Datasheet*, 2017, http://www.analog.com, Accessed 20 February, 2020.

[10] Telecommunications Industry Association (TIA), *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*, ANSI TIA/EIA-485A, March 1998.

[11] Turbo51, *Turbo Pascal for 8051 microcontrollers*, https://turbo51.com/, Accessed 20 February, 2020.

[12] Quanser Inc., *User Manual Couple Tank Experiment Set Up and Configuration*, Quanser Inc., Canada, 2012.

[13] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning", *Journal of Process Control*, vol. 13, pp. 291 – 309, 2003.

[14] J.G. Ziegler and N.B. Nichols, "Optimum settings for automatic controllers", *Trans. ASME*, vol. 64, pp. 759 – 768, 1942.

[15] K.J. Åström and T. Hägglund, "Revisiting the Ziegler–Nichols step response method for PID control", *Journal of Process Control*, vol. 14, pp. 635 – 650, 2004.