Event-Based PI Control Applied to Laboratory Plant

Wilson Kepa[1], Rogelio Mazaeda Echevarría[2] and Raj Kumar[1*]

[1]School of Electrical and Communication Engineering, Papua New Guinea University of Technology
[2]School of Industrial Engineering, University of Valladolid, Spain

*Correspondence Author email: raj.kumar@pnguot.ac.pg

Abstract: In this research, event-based PI controllers is implemented to control a liquid flow-rate in a laboratory plant. With the emerging technologies like Industry 4.0 and IoT, the Networked Control Systems (NCSs) are said to be wirelessly connected in the distributed process industries, thereby reduction in signal transmissions, CPU utilization and power consumption. The flow-rate valve actuator transfer function is identified as a first-order plus dead time (FOPDT) using the open-loop reaction curve of the plant and it is tuned with the SIMC-PI tuning method. A time-based PI controller is designed and simulated on the plant with event-based PI controller techniques. This is further improved as an advanced type of event-based PI strategy (PIDPLUS controller) **to** improve the performance. The**se** techniques are implemented and simulated in the MATLAB/SIMULINK workspace aided by OPC toolbox that provides the interface for laboratory plant data reading and control. The emerging technologies used in this research can be employed by the industries in PNG to enhance their productivity and performance.

Keywords: NCSs, event-based PI, SIMC-PI, sticking effect, oscillatory behaviour, PIDPLUS.

1. INTRODUCTION

The proportional-integral-derivative (PID) controller is the most widely used closed-loop control mechanism in industry, particularly in process and manufacturing industries. There are many variants of PIDs: ideal, parallel, series, one degree of freedom (1DoF), two degree of freedom (2DoF), etc. Most of such controllers are implemented as proportional-integral (PI) controllers, since the derivative action is frequently not used in industrial process control applications due to its noise amplification effect. Traditionally, the control systems used in implementing PI and PID controllers have been based on periodic sampling (time-based), for which data transmission between controller agents like sensors to the controllers and actuators are done periodically.

However, nowadays in response to global competition and the fast adoption of production to the ever-changing market request, radical advances in non-centralized and distributed industrial process technologies have introduced networked control systems (NCSs). In the likes of emerging technologies like Industry 4.0 (Y. Lu., 2017), Internet of Things (IoT) and 5G evolutions where the closed-loop is wirelessly connected; one should consider the trade-off in scarce resources like bandwidth, energy consumption and CPU utilization of embedded processors and industrial computers that perform control actions. Additionally, the wireless network may induce network congestion which could result in large communication delays and loss of information, which greatly affect the process controller performance. Hence, there is a significant increase in research in the last decades that proposed several event-based controller structures and PI tuning methods (Heemels, 2013; Poveda, 2017; Borgers, 2017). Figure 1 shows a typical wirelessly connected control agents of a distributed industry.

Event-based control, referred to as aperiodic or asynchronous where sampling is event-triggered rather than time-triggered (Arzen, 1999). The event-based control technique is a new control method that closes the feedback loop only if the error signal exceeds a predefined threshold. In this method, data transmission between the controller agents are asynchronously triggered, thus reducing the number of signal transmissions. Also, when not actively engaged in transmitting or processing signals, sensor nodes and controllers can be put in sleep mode to save energy. The use of event-based controllers (including event-based PID) have to renounce to a very sizeable part of the linear control system theory, to make an appeal to

Event-Based PI Control Applied to Laboratory Plant
Wilson Kepa, Rogelio Mazaeda Echevarría and Raj Kumar | 2 (2) : 120 - 133 | Received: 27/ 05/2025 | Reviewed: 23/07/2025 | Accepted: 24/08/2025 |
DOI: https://doi.org/10.63900/4g8k4c66

ad hoc solutions or less developed notions such as one provided by the theory of hybrid systems, efforts in trying out and comparing different alternatives are very much welcome.

The event-based PI (EPI) is implemented in the present research what constitutes an ongoing very active effort in academia and industry. The two EPI controllers are compared with a traditional timed-based PI (TPI). The comparisons are made initially in simulation and then they are applied to the control of flow-rate on a real laboratory plant. In order to produce a fair comparison both EPIs are compared with a correctly tuned TPI as per tuning rules (Dwyer, 2009). Here, the methodology (Grimholt, 2018) as applicable to the control of FOPDT processes is used. Many tuning rules, offers a single set of values for the parameters of the PI(D). The SIMC tuning rule offers a simple and intuitive single parameter tuning that help the designer to make the trade-off between the required degree of aggressiveness and robustness.

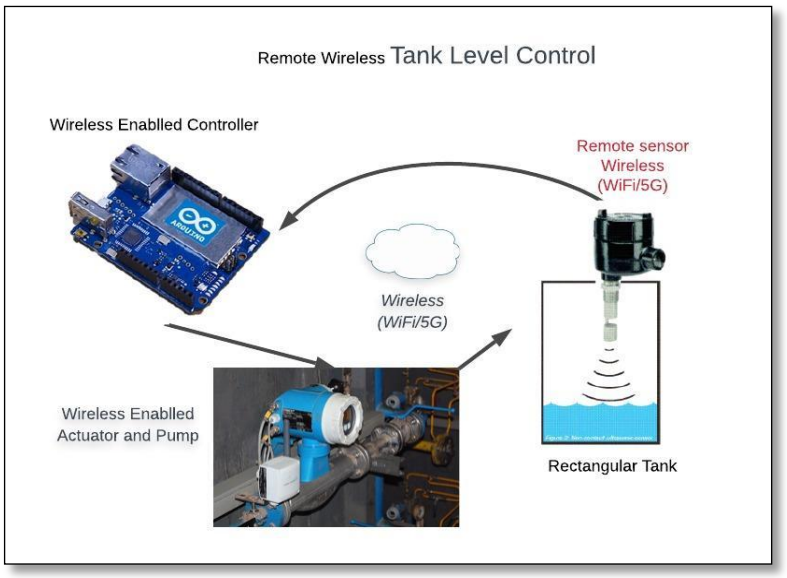

Figure 1: Wireless control of a tank level

The results produced in this research signify a promising solution for distributed industrial settings employing NCSs by attaining similar system performance as in TPI controllers by reducing signal transmissions with the proposed EPI controller algorithms. The benefits of reducing signal transmissions from controller agents are reduced network congestion that may induce network delays and loss of viable data transmissions, improved CPU performance of controllers and efficient use of energy for battery powered controller agents. The MATLAB/SIMULINK (Natick, 2019) environment is used in development. It is used for gathering sensor data from the real plant, to identify dynamics of the flow-rate valve, to create the model of reference TPI, and of the two EPIs, to perform simulations and finally to control the real plant flow-rate actuator with the coded controllers. The interface between MATLAB/SIMULINK and laboratory plant has been achieved using the OPC toolbox, an implementation of the known interoperability protocol (Zamarreño, 2014).

2. LABORATORY PLANT AND MATLAB/SIMULINK OPC TOOLBOX

2.1 Laboratory Plant

A heat exchanger is a mechanical device used particularly for heating and cooling applications where internal thermal energy is transferred between two or more fluids available at different temperatures. Heat exchangers are widely used in industries like process and manufacturing, power, petroleum, transportation, air-conditioning, refrigeration, and other industries. Figure 2 shows the heat exchanger laboratory plant used to implement the proposed event-based PI controller. The flow-rate valve actuator is considered the system of interest where a

FOPDT model is identified, tuned with SIMC method (Grimholt, 2018) and the proposed event-based PI control algorithm is tested and implemented.

2.2 MATLAB/SIMULINK OPC Toolbox

The MATLAB/SIMULINK Open Process Control (OPC) (Drahoš, 2018) toolbox, also known as OLE for process control, is a series of seven specifications defined by the OPC Foundation for supporting open connectivity in industrial automation. The Microsoft DCOM technology is used in OPC to provide a communication link between OPC servers and clients that enhance reliable communication of information in an industrial process plant. In this research, an OPC custom server was configured and interfaced through a USB-1408FS-Plus data acquisition card that communicates to the OPC clients. This setup allows for the reading of sensors values to the controller and for writing actuator values. The OPC interface allows the connection of the MATLAB/SIMULINK coded controller to the sensors and actuators for the closed loop feedback control. The OPC Simulink model used is shown in Figure 3 where an open-loop step test is performed for FOPDT model parameter identification for derivation of system transfer function.

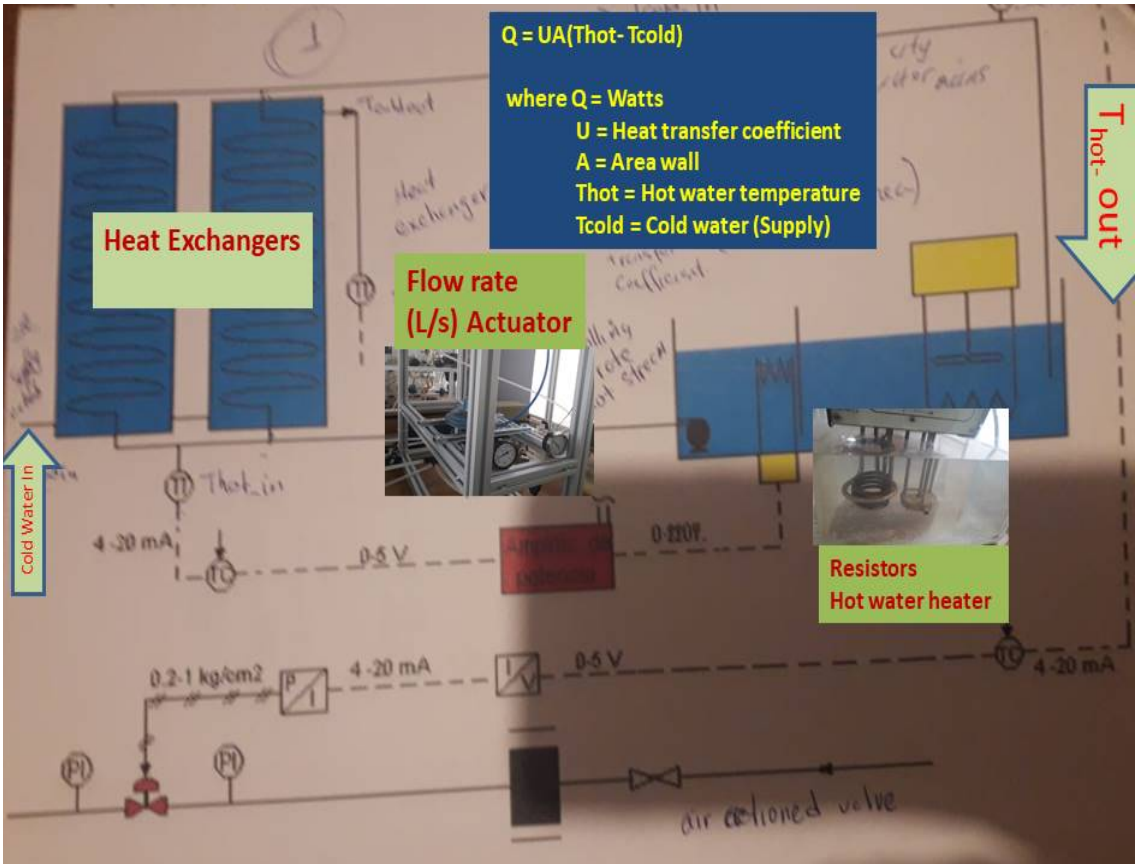

Figure 2: Schematic of the heat exchanger laboratory plant

In the Figure 3, OPC Read block (1) reads the value provided by the flow-rate sensor and made available in the input signal line (in_Volts03) of the OPC server. Depending on the changes in position of the step input block (2), the OPC Write block (3) writes the valve actuator signal via output/control signal line (out_%01) which corresponds to the opening of the control valve. The gain block (2.09) and the negative constant block (-1.82) are the slope and intercepts respectively for the linear calibration of the input sensor signal (y(t)) into its physical quantity (litres/seconds). The simulation output is shown in Figure 4 that shows the plots of step input (u) and system output (y) with respect to time (t) which is used for the identification of the model of the plant.

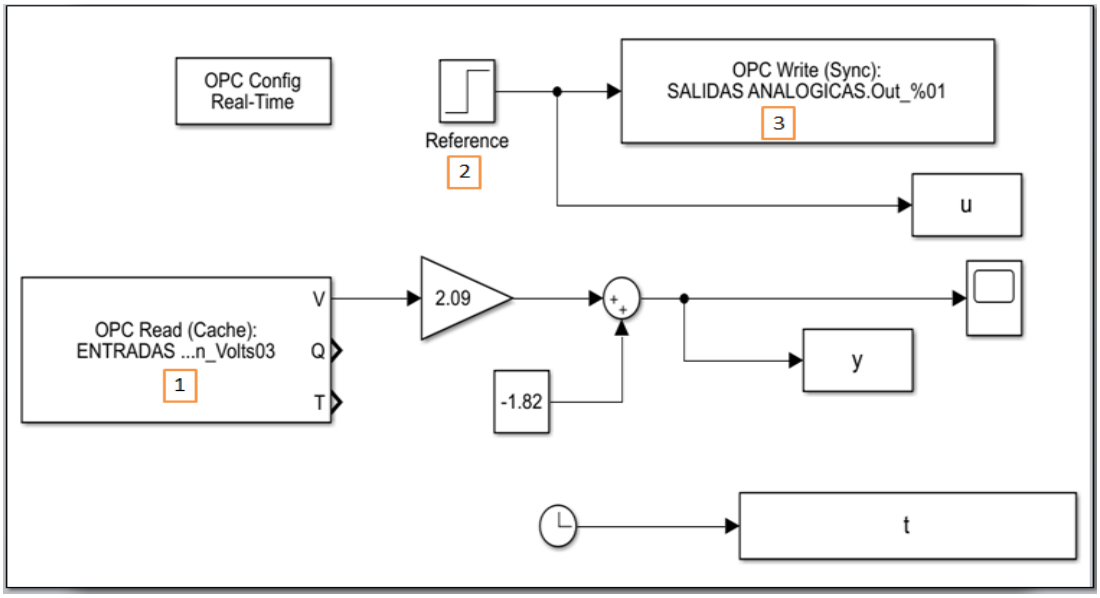

Figure 3: OPC MATLAB/SIMULINK model setup for obtaining reaction curve

3. IDENTIFICATION OF TRANSFER FUNCTION PARAMETERS AND TUNING WITH SIMC AND TPI

3.1 Identification of Actuator Transfer Function Parameters

Identification of transfer function parameters of flow-rate is a necessary requirement for implementation of PI/PID controller tuning. The transfer function of flow-rate actuator valve was derived from the data labelled in Figure 3, given a step input signal assuming the valve was initially closed or at rest. This identification process is known as step test, generally performed in terms of an open-loop control structure. Figure 4 shows the reaction curve of dynamic system response of the flow-rate actuator valve.

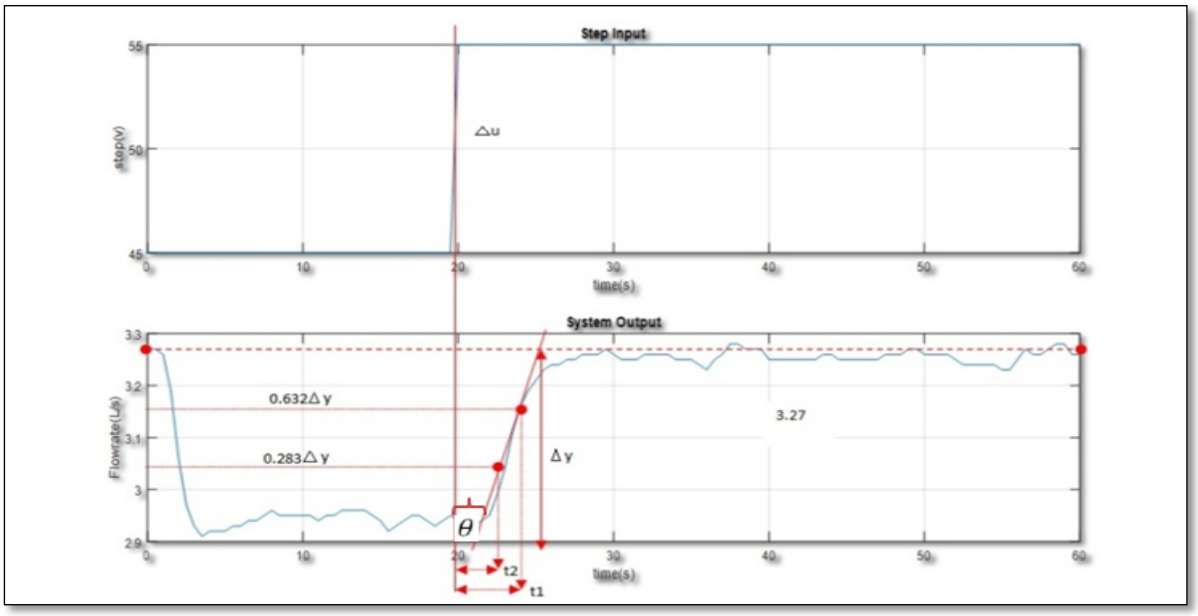

Figure 4: Open loop reaction curve of the flow-rate valve actuator.

A step test at simulation time of 60 seconds yields Figure 4 in which a linear FOPDT model was considered mainly for parameter identification. Generally, the FOPDT model is of the form

$$G(s) = \frac{K*e^{-\theta s}}{\tau s+1} \quad (1)$$

where K depicts process gain, θ process time delay in seconds, and τ process time constant. The calculations for system transfer function parameters using figure 4 give

The Process Gain

$$K = \frac{\Delta y}{\Delta u} = \frac{0.37}{1o} = 0.037 \quad (2)$$

Process Time Delay

$$\theta = t1 - \tau = 3.134\text{-}0.194 = 2.941$$

Process Time Constant

$$\tau = 1.5(t1 - t2) = 1.5\ (3.134 - 3.01) = 0.194$$

The equation (2) is the transfer function of the flow-rate actuator valve of the heat-exchanger plant, where the proposed event-based PI controller is tuned and implemented.

3.2 Tuning of Time-Based PI with SIMC Tuning Method

The PI controller is widely adopted in process industries, while the D-term (derivative) is, in many occasions, neglected to reduce the propagation of amplified random measurement noise via the controller. Given the flow-rate valve actuator transfer function in (2), the challenge now is to apply a proper tuning technique to control the flow-rate valve actuator and to remain in the linear region of the actuator saturation. The transfer function of an ideal PI controller is given in (3),

$$Gc(s) = Kp(1 + \frac{1}{Tis}) \quad (3)$$

where *Kp* is the proportional gain and *Ti* is the integral time constant. The PI controller in its digital equivalent is given in (4),

$$xc(t_{k+1}) = xc(t_k) + \beta(t_k)\,(r - y(t_k)),$$
$$u(t_k) = Kp(r - y(t_k)) + \frac{1}{Ti}xc(t_k) \quad (4)$$

where xc is integrator state, Kp is proportional gain, Ti is integration time, $\beta(t_k)$ is integrator update rate, r is reference signal usually constant, $y(t_k)$ is system output and t_k is time instant in at which a new control input is computed.

A better tuning is achieved by obtaining optimal PI settings, which by definition is the minimum integrated absolute error (IAE) to disturbances for a given robustness level with less response time (Grimholt, 2018). A very simple yet effective methodology for tuning PI(D) controller for different type of plants are the SIMC rule discussed in (Grimholt, 2018). An important plus for adopting SIMC is that it provides an easy way for helping the designer to choose a particular tuning in continuous spectrum. The spectrum encompasses a range of equally effective tunings but that result in different closed loop dynamic behaviors which go, from a very aggressive, rapid and oscillatory response at one extreme, to a smooth, slow, very robust dynamics at the other. Thus, SIMC rule is the systematic approach adopted for this research.

3.2.1 SIMC Tuning Method

The SIMC turning rule (Skogestad, 2004) works well for both pure time delay and integrating processes and for both set-point tracking and load disturbances rejections. The original SIMC rule is of the form:

$$kc = \frac{1}{k}\frac{\tau}{(\tau c+\theta)}, \ \tau i = min\ \{\tau, 4(\tau c + \theta)\} \quad (5)$$

where *kc* is gain of the PI controller to tune, τi is integral time of the PI controller to tune, the *k, τ, θ* are the identified FOPDT parameters of valve actuator in (2) respectively. The SIMC procedure is provided with a single tuning parameter τc which helps the designer to adjust desired close loop dynamics behavior of the PI settings which is much simpler with meaningful interpretation than adjusting *kc* and τi directly. The recommended choice for parameter tuning for optimal value of τc is determined by the trade-off between (i) quick system response and good disturbance rejection (small value of τc*)* and (ii) *s*tability, robustness and small input variation (favored by a large value of τc). Therefore, a good trade-off is recommended by choosing $\tau c = \theta$, which is for FOPDT models gives a reasonably fast response with moderate input usage and good robustness margins. Calculated in (2), letting the $\tau c = \theta = 2.941s$, *kc* and τi are now calculated as shown,

$$kc = \frac{1}{k}\frac{\tau}{(\tau c+\theta)} \approx 0.89 \qquad \tau i = min\ \{\tau, 4(\tau c + \theta)\} \approx 0.194.$$

Therefore, the $kc \approx 0.89$ and the $\tau i \approx 0.194$ are the tuned PI parameters of which a time-based FOPDT PI controller is designed, the base for fair comparison with proposed event-based counterparts. The tuned kc and τi parameters are used in event-based simulations for control of flow-rate valve actuator of heat exchanger laboratory plant.

3.2.2 Time-based PI Controller of FOPDT Model

Given the FOPDT transfer function of flow-rate valve actuator in (2) to be controlled by tuned PI controller, a time-based closed-loop PI controller with unity feedback is implemented. Figure 5 shows the implementation of time-based PI controller**.** The code in *Listing 1* is implemented in the user-defined MATLAB function block on the SIMULINK block diagram. The code includes an anti-windup mechanism. When the controller's output enters into valve saturation region (above 100% or below 0%) the summation represented by the integral term is stopped. With this simple measure in place, the very harmful windup phenomenon, that results in an unacceptable behavior is avoided.

For the time-base case, it is evident that the sampling is done periodically, since the number of samples/events of the system output shown in Figure 5(b) when implemented in the plant yields N = 10,000 samples. This is so because the simulation time of 200 seconds is evenly divided by the nominal sampling time. As a rule of thumb in control engineering for proper sampling of analog signals, the nominal sampling time is 10 times faster than the process time constant, that is $\tau/10$ ($hnom = 0.02\ s$). This increase in events due to periodic sampling is said to be greatly reduced as proposed with event-based techniques.

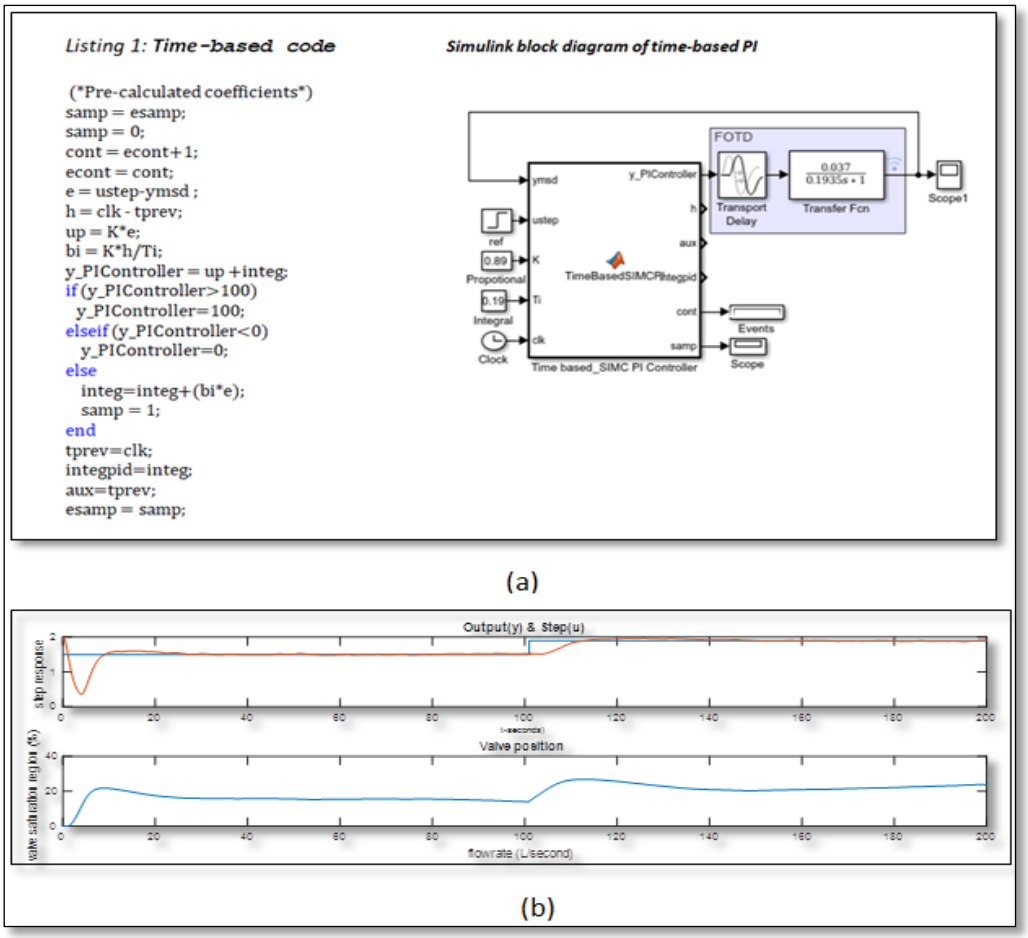

Figure 5: Time-based SIMC PI: (a) Listing 1: MATLAB code; Simulink model
(b) Output(y) and step (u); Valve position

4. SIMPLE EVENT-BASED PI CONTROL

As shown in time-based PI control Figure 5, even when the process output is around set-point, the signals are periodically sampled where no more transmissions would be required. This presents a challenge for distributed wireless NCSs where the control agents involved in the control loop are connected by Wi-Fi. The reduction in signal transmission is a relevant issue accounting for the scarce information bandwidth and network congestion that may lead to data loss and communication delays. Moreover, reduction in information flow will improve the CPU utilization (Arzen, 1999) for controllers, and saves energy for battery powered control agents.

To address these issues, event-based sampling and control technique has been used. This allows an efficient utilization of the network resources, while ensuring a desired behaviour of the closed-loop control system. This is achievable by transmitting information only when a certain threshold around the set-point is exceeded, where only then a control action is triggered. To achieve this, we introduce a new parameter namely a threshold around the set-point, $elim$. However, there are issues as pointed out by (Tiberi, 2012) that adoption of the event-based techniques may introduce *sticking* effect. With event-triggered controllers, it is in general impossible to reduce to zero the control error at steady state. This happens even in the presence of integral action in the controller. So, it is said that the error gets stuck on a finite, different from zero value, provided that this value is less than the pre-defined threshold. In many applications, like in Figure 1, the presence of a steady state error does not represent a major problem. In other cases, however, it is necessary to enforce a zero error. In this latter case, one would think that the problem could be solved by adding a time out (h_{max}) to the sampling rule, so

that the sensor is enforced to send a new measurement to the controller whenever the closed-loop system gets stuck, even when the threshold value is not surpassed. Thus, the new sampling rule is of the form:

$$\tau(t) = (|(e - eprev)| \geq elim)\text{OR}(hactual \geq hmax) \quad (6)$$

However, introducing the time-out barely solves the sticking effect but introduces another problem, which is the possible presence of sustained *oscillations* around the set-point. The increase in h_{max} increases oscillations around the set-point which then increases the number of events. Since the integrator update rate $\beta(t_k)$ is equal to the inter-event times h_k, if h_{max} is too large so it is $\beta(t_k)$. This is true because control input applied at time $t_{k+1} = t_k + h_{max}$ may be too aggressive and it potentially triggers an oscillatory behaviour of the output. Because of the sticking effect and the oscillatory behaviour of this simple event-based scheme, it is thus named a naive EPI controller. Figure 6 shows the MATLAB code and the Simulink model block diagram of the simple EPI controller. In order to realize the drawbacks mentioned, three test conditions (viz. Sticking Effect Simulation, Oscillations Around Set-Point Simulation, and Less Aggressive Tuning for Robust Response Simulation) are simulated to showcase the presence of the sticking effect, oscillatory behavior, and finally with a less aggressive tuning and selecting a reasonable time-out that at least cancels both the drawbacks.

Listing 2: ***Naive Event-based code***

```
(*Pre-calculated coefficients*)
e=ysp - ymsd;
h=clk-tprev;
bi=K/Ti;
hactual=hactualprev+h;
cont=econt;
samp=0;
if (abs(e-eprev)>elim || (hactual>=hmax))
  up=K*(ysp-ymsd);
  samp=1;
  eprev=e;
  y_controller=up+uiprev;
  ui=uiprev+bi*hactual*(ysp-ymsd);
  cont=cont+1;
  econt=cont;
  hactual=0;
  uiprev=ui;
  uprev=y_controller;
else
  y_controller=uprev;
end
esamp=samp;
tprev=clk;
hactualprev=hactual;
```

Simulink block diagram of naive event-based PI

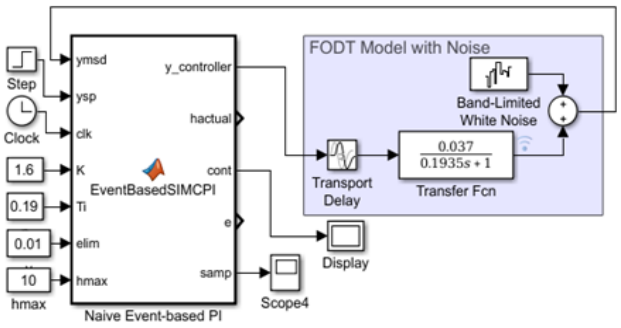

Figure 6: Simple event-based PI controller; Listing 2: MATLAB code, Simulink model block

4.1 Simulations of Simple EPI Applied on the Laboratory Plant

To validate the simulations, Simulink model in Figure 6 is implemented on the flow-rate actuator of the laboratory plant as in the time-based simulations for a fair comparison. We consider the sampling time $hactual = 0.02\ s$, as that of the nominal sampling time $hnom$ of the TPI, and following the same steps as in the TPI simulation, three separate simulations with varying $elim$ and $hmax$ are shown in Figure 7. The simulations are performed with a less aggressive tuning by setting Kp = 0.89, Ti = 0.19. Note that the noise signal and the FOPDT models blocks in the Figure 6 are replaced with the plant, via the OPC toolbox in MATLAB/SIMULINK interface.

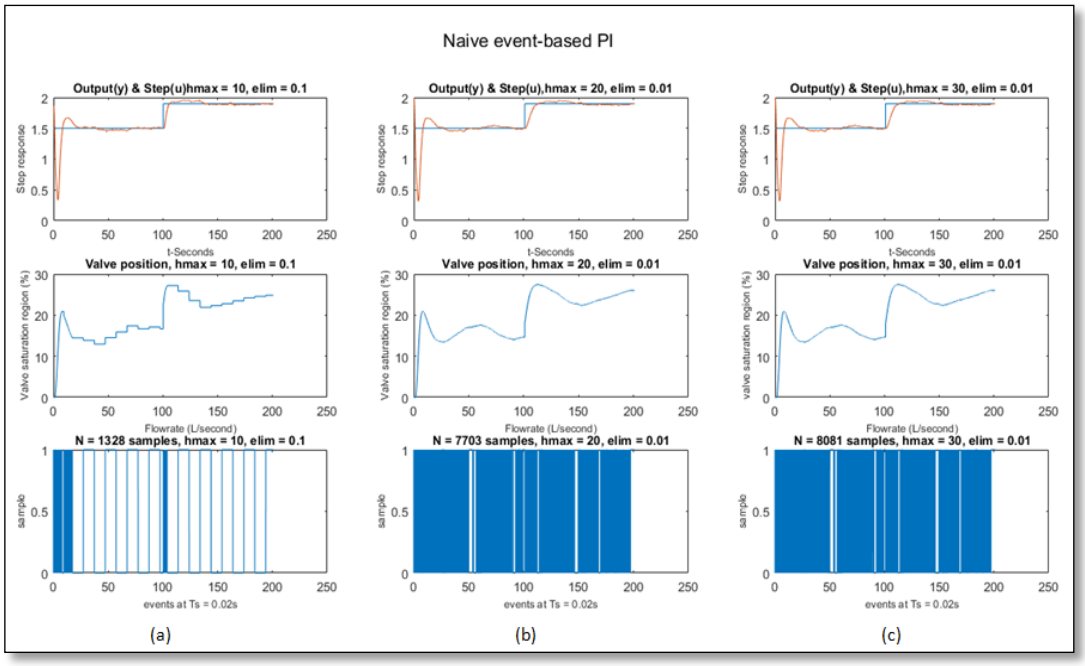

Figure 7: Simple event-based PI control: Laboratory plant simulations (a) elim = 0.1, hmax = 10 s yields N = 1328 events; (b) elim = 0.01, hmax = 20 s yields N = 7703 events; (c) elim = 0.01, hmax = 30 s, N = 8081 events.

From the simulations, it is seen that in the simple EPI the sticking effect introduces a non-zero steady state error, in spite of the presence of the integral action. The increase in the size of the $hmax$ introduces sustained oscillations around the set-point, resulting in a bad steady state behavior, and provoking the increase of the number transmissions, as can be seen in Figure 7 (b) and (c) having the same $elim = 0.01$. On the other hand, by choosing small values for $hmax$ to reduce the oscillation, we may lose the benefits of using the proposed EPI scheme since more transmissions are required. The simple event-based PI controller is further improved with the advanced event-based PIDPLUS at the cost of highest complexity.

5. ADVANCED EVENT-BASED PIDPLUS

The sticking effect and the oscillatory behaviour that presents in the earlier EPI controller are further improved with the PIDPLUS technique at the cost of a higher complexity. The complexity in the PIDPLUS version of the event-based controller presents then two degrees of freedom: one degree of freedom is represented by the selection of an appropriate design of the integrator update rate $\beta(t_k)$ implemented on the controller, while the other concerns the choice of a suitable event-based rule $\tau(t) \leq elim$ implemented on the sensor.

5.1 PI- based Triggering Rule

The PI-based triggering rule is introduced by (Tiberi, 2012) intuitively considering the dead-band triggering rule (Otanez, 2002) for an appropriate filtered version of the output signal. The filtered control input proposed is of the form:

$$\tilde{u}(t) = \acute{K}p\left(\left(r - y(t)\right) + \frac{1}{\check{T}i}\int_{tk}^{t}(r - y(s)ds + \tilde{}xc(tk)\right) \quad (7)$$

where $\acute{K}p$ and $\check{T}i$ are two shadow sampling parameters, and $\tilde{}xc$ is the shadow state of the integrator implemented on the sensor, thus they proposed the PI-based triggering rule denoted as

$$\tau(t) = \left|\tilde{u}(tk) - \acute{K}p\left(\left(r - y(t)\right) - \frac{1}{\check{T}i}\int_{tk}^{t}(r - y(s))ds - \tilde{}xc(tk)\right)\right| \leq elim. \quad (8)$$

Whenever the sticking effect occurs, the integral term in (8) grows unbounded, forcing the sensor to send a new event to the controller, thus avoiding the sticking effect. Moreover, the time-out in the simple EPI implementation is disregarded such that the controller is no longer updated if and only if $y_{k^*} = r$, that yields $\tilde{u}(t_{k+1}) = \tilde{u}(t_k)$ for all k greater than k^*. This is because when the system gets stuck, the sampling rule imposes a time-out that depends on the distance y_k from the set-point r. However, it should also be noticed that even if the triggering rule cancels the sticking effect and potentially sends a new control input $\tilde{u}(t_{k+1})$ straight to the actuator, the controller would be updated with $\tilde{u}(t_{k+1}) = \tilde{u}(t_k) \pm elim$. This control update rule leads to limit cycles that may generate unacceptable oscillation of the output. To avoid such oscillations, it is further proposed that whenever the sensor verifies that the PI-based triggering rule in (8) is violated, the sensor transmit to the controller the $y_p(t_k)$ instead of the value of $\tilde{u}(t_{k+1})$. Therefore, the controller updates the input signal according to the received measurement signal $y_p(t_k)$ and to the elapsed inter-event times.

5.2 Integrator Update Rate Adoption

The integrator update rate $\beta(t_k)$ is implemented on the controller and is of the from:

$$\beta(t_k) = Ti(\,1 - e^{\frac{-hactual}{Ti}}) \tag{9}$$

where Ti is the controllers integral time constant and the $hactual$ is the current inter-event period of the controller. For consistent results the formulation of the PIDPLUS as (11), with $\beta(t_k)$ as in (9). In (Tiberi, 2012), stability results of the proposed scheme are given, making use of advanced concepts of hybrid discrete-continuous theory that are well beyond the objectives of this project.

The MATLAB code in the *Listing 3* of Figure 8 shows the implementations of the advanced event-based PIDPLUS controller where the PI-based triggering rule as well as the integrator update rate are implemented. As in the simple EPI implementation, to visualize the achievements with the advanced event-based PIDPLUS controller, simulations at different test conditions are presented. Here the $hmax$ is set to infinity or neglected, since the tuning rule at the sensor side is determined by (8).

Listing 3: Advanced PIDPLUS event-based code

```
(*Pre-calculated Coefficients*)
e = ysp-ymsd;
h=clk-tprev;
bi=K/Ti;
hactual=hactualprev+h;
cont=econt;
samp=esamp;
bis=Ks*h/Tis;
xc=xcprev;
uiprevshow=uiprev;
error=abs(uprev-Ks*((ysp-ymsd))-xc-uiprev);
error1=Ks*(ysp-ymsd)+xc+uiprev;
error2=uprev;
if (abs(uprev-Ks*((ysp-ymsd))-xc-uiprev)>elim)
    up=K*(ysp - ymsd);
    if (samp==0)
        samp=1;
    else
        samp=0;
    end
    xcprev=0;
    xc=0;
    eprev=e;
    ui=0;
 y_controller = up+1*uiprev;
    if (y_controller>100)
        y_controller=100;
    elseif (y_controller<0)
        y_controller=0;
    else
         ui=uiprev+bi*Ti*(1-exp(-hactual/Ti))*(ysp - ymsd);
    end
    cont=cont+1;
    econt=cont;
    hactual=0;
    uiprev=ui;
    uprev=y_controller;
else
    ui=uiprev;
    y_controller=uprev;
end
xc=xcprev+bis*(ysp-ymsd);
xcprev=xc;
esamp=samp;
tprev=clk;
hactualprev=hactual;
```

Simulink block diagram of advanced PIDPLUS event-based PI

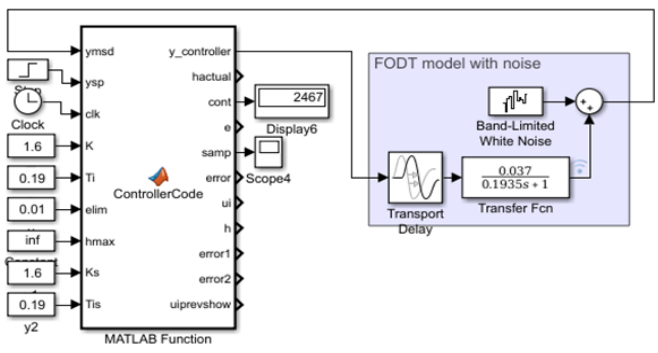

Figure 8: *Event-based PIDPLUS controller; Listing 3: MATLAB code, Simulink block diagram*

5.3 Simulations of Advanced Event-based PIDPLUD on the Laboratory Plant

As in the TPI and the simple EPI simulations, the advanced event-based PIDPLUS controller is implemented on the laboratory plant. Three different test conditions were simulated with varying $elim$ and the number of events transmissions were recorded as shown in Figure 9. It is noted that the computer simulations and with implementation on the laboratory plant, increase in $elim$ decreases the events transmissions and vice versa. Also, selecting different tuning of the parameters $\acute{K}p$ and $\check{T}i$ at the sensor and Kp and Ti at the controller has no restrictions.

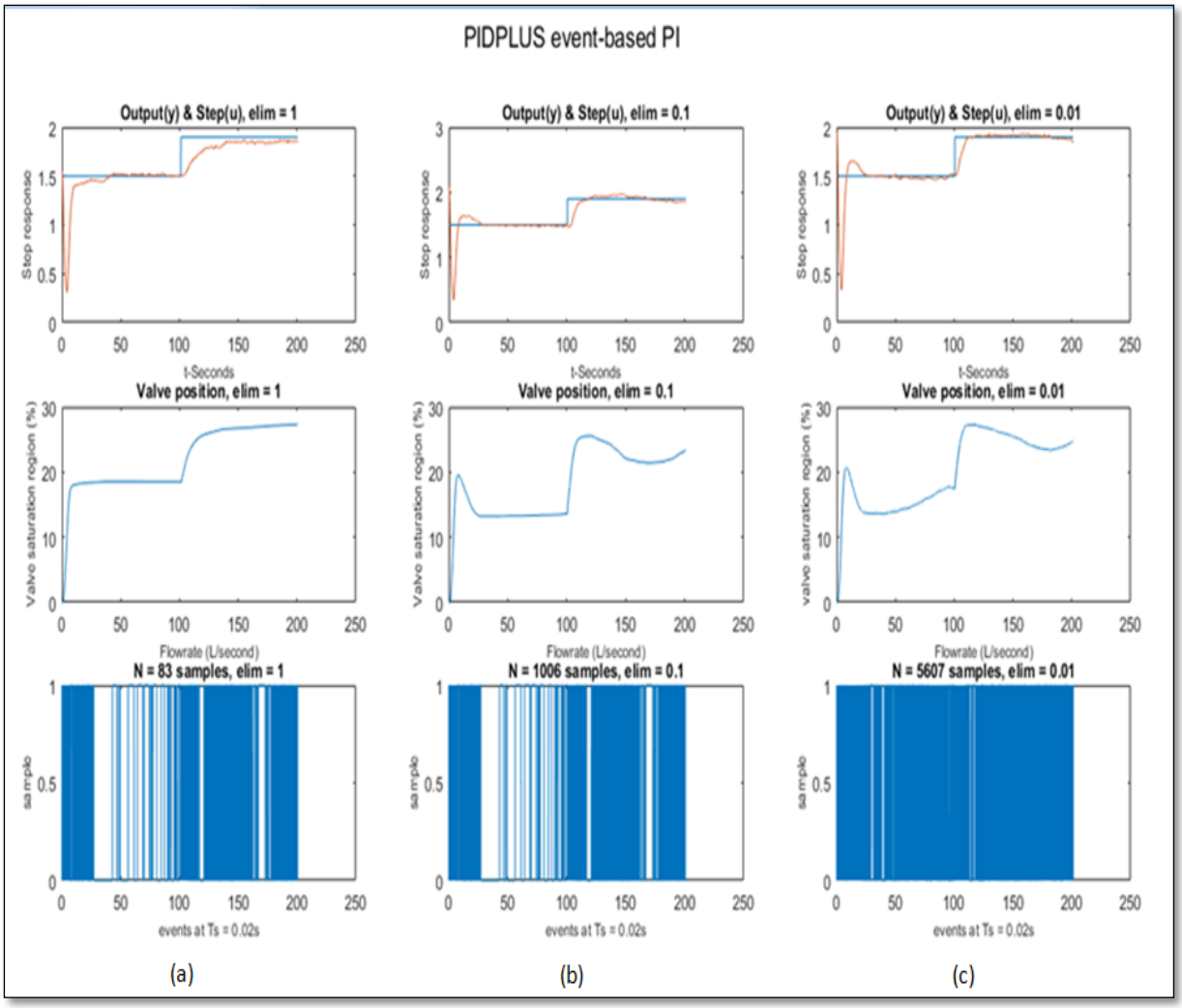

Figure 9: Advanced event-based PIDPLUS: Laboratory plant simulations: (a) elim = 1 yields N = 83 events; (b) elim = 0.1 yields N = 1006 events; (c) elim = 0.01 yields N = 5605 events.

A flowchart to distinguish the sensor and the controller algorithms for two event-based PI controllers is presented at Figure 10. Since the sensors and the controllers are located remotely and connected wirelessly, the algorithms are visualized separately for the simple EPI and the advanced PIDPLUS in the flowcharts shown in Figure 10. The flowchart implies the algorithms separately for simple EPI and advanced PIDPLUS controller at the sensor node and the controller respectively. Thus, the presented MATLAB codes shows that there is no difference in the results presented.

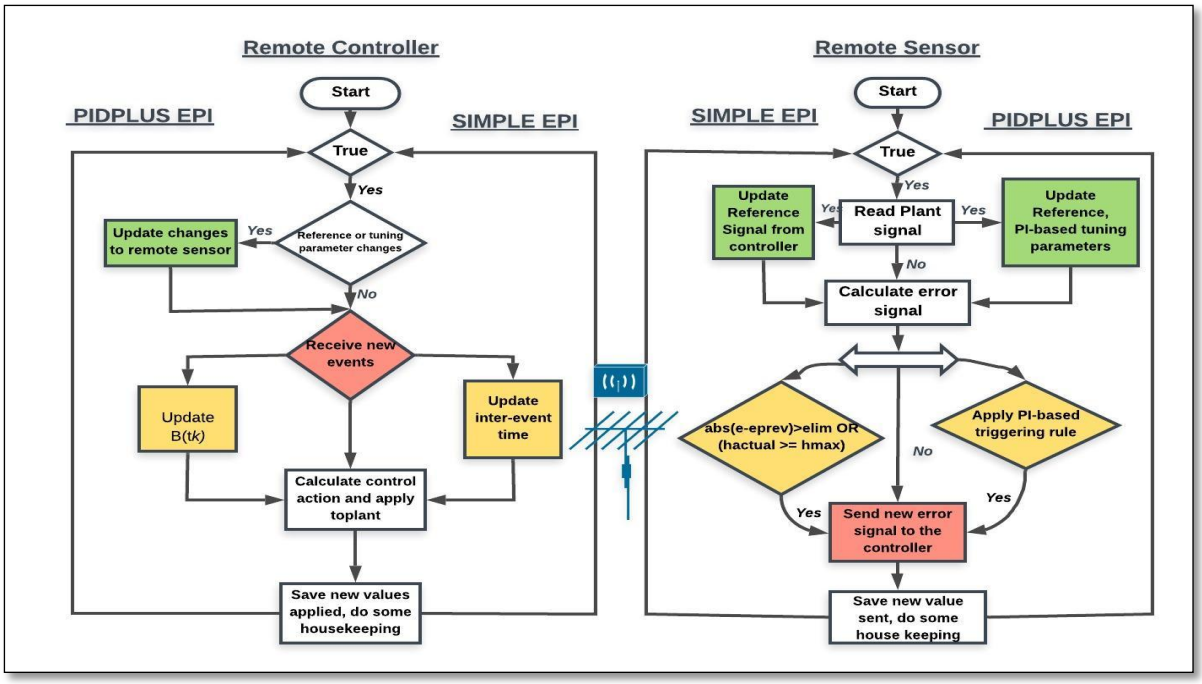

Figure 10: Algorithm flowcharts for Simple EPI and PIDPLUS at controller and sensor

6. RESULTS AND ANALYSIS

In this research, a systematic approach in implementing the proposed event-based PI controller of a linear FOPDT model is presented. Two distinct schemes have been designed and simulated; one being a simple EPI and other an advanced EPI with the PIDPLUS, which both seem promising for wireless NCSs over the traditional TPI methods. To better visualise the improvements in the traditional TPI over the proposed simple EPI and the advanced PIDPLUS, as far as reduction in events transmission is concerned, at several test conditions implemented on the plant, we obtain Table 1.

Table 1: Events transmissions at different test conditions of the laboratory plant

	Kp	*Ti*	*θ*	*hnom(s)*	*elim*	*hmax(s)*	*Events (samples)*
Time-based	*0.89*	*0.194*	*2.941*	*0.02*	–	∞	**10000**
Naive Event-based	*0.89*	*0.194*	*2.941*	*0.02*	1	∞	**91**
	0.89	*0.194*	*2.941*	*0.02*	0.1	10	**1328**
	0.89	*0.194*	*2.941*	*0.02*	0.1	20	**1678**
	0.89	*0.194*	*2.941*	*0.02*	0.1	30	**1943**
	0.89	*0.194*	*2.941*	*0.02*	0.01	10	**6885**
	0.89	*0.194*	*2.941*	*0.02*	0.01	20	**7703**
	0.89	*0.194*	*2.941*	*0.02*	0.01	30	**8081**
PIDPLUS Event-based	*0.89*	*0.194*	*2.941*	*0.02*	1	∞	**83**
	0.89	*0.194*	*2.941*	*0.02*	0.1	∞	**1006**
	0.89	*0.194*	*2.941*	*0.02*	0.01	∞	**5607**

The data presented in Table 1 is with the similar parameter settings and tested for a duration of 200 seconds duration time at sampling rate of 0.02 seconds. The number of events transmissions are reduced with the proposed EPI compared to that of TPI where N = 10,000 samples.

7. CONCLUSIONS

The adoption of wireless NCSs in the emerging technologies like Industry 4.0 and Internet of Things (IoT) reduces information transmission rate from controller agents with event-based techniques as compared to traditional time-based sampling techniques. An advanced event-based PIDPLUS model is designed to control the flow-rate of the heat exchanger laboratory plant using MATLAB/SIMULINK OPC toolbox. A time-based PI controller is designed and simulated for comparisons with the proposed event-based counterparts. These techniques are implemented, simulated and tested on flow-rate of the laboratory plant. Simulations and real data show reduction in signal transmission rate, less bandwidth usage, improved network congestions, improved CPU utilization and extended battery life reducing the need of regular battery replacements. This research uses emerging technologies viz Industry 4.0, Internet of Things (IoT) and 5G evolutions which can be employed by the industries in PNG to enhance their productivity and performance.

ACKNOWLEDGMENT

This research work was supported by ERASMUS+ KA107 master mobility exchange program, managed by the European Commission, at University of Valladolid, Spain.

REFERENCES

Aceto G., Persico V., and Pescape A. (2019) “A survey on information and communication technologies for industry 4.0: state-of-the-art, taxonomies, perspectives, and challenges”. IEEE Communications Surveys & Tutorials, vol. 21(4), pp.3467–3501.

Algirdas Baskys (2024) “A Combined Controller for Closed-Loop Control Systems Affected by Electromagnetic Interference”, MDPI Journal of Sensors, vol.24(5), pp.1466-87.

Aranda Escolástico, et.al. (2020) “Event-based control: A bibliometric analysis of twenty years of research”, IEEE Access, 8:47188–47208.

Arzen K.E. (1999). “A simple event-based PID controller,” 14th World Congress IFAC Proceedings, 32(2), 8687-8692.

Borgers D. P, Dolk V. S. and. Heemels W. P. M. H. (2017) “Riccati-Based Design of Event-Triggered Controllers for Linear Systems with Delays,” IEEE Trans. Automat. Contr., 63(1), 1–19.

Duca, Octavian Gabriel, et.al. (2022) “Event-based PID control in a flexible manufacturing process”, Proc. 26th International Conference on System Theory, Control and Computing (ICSTCC), pp.182-187.

Drahoš P., Ku E., Haffner O., and Klimo I. (2018) “Trends in Industrial Communication and OPC UA,” Cybernetics & Informatics (K&I) conference proceedings, 4–8.

Grimholt C. and Skogestad S. (2018). “Optimal PI and PID control of first-order plus delay processes and evaluation of the original and improved SIMC rules,” J. Process Control, 70, 36–46.

Heemels W. P. M. H. and Donkers M. C. F. (2013) “Model-based periodic event-triggered control for linear systems,” Automatica, 49(3), 698–711.

Lu Y. (2017). “Industry 4.0: A Survey on Technologies, Applications and Open Research Issues,” Journal of Industrial Information Integration, 6, 1-10.

Mahmoud Abdelrahim and Dhafer Almakhles (2023) “Output-Based Dynamic Periodic Event-Triggered Control with Application to the Tunnel Diode System”, MDPI Journal of Sensor and actuator networks, vol. 12(5), pp.66-87.

MATLAB, (2019). Natick, Massachusetts: The MathWorks Inc.

Murugan C. et.al. (2024) “An event triggered control scheme for enhanced production of Escherichia coli and biomass concentration during fed-batch cultivation”, Heliyon, vol.10(12), pp. 1-12.

O’Dwyer. (2009) Handbook of PI and PID controller tuning ruler (3rd ed.), Imperial College Press.

Otanez P. G., Moyne J. R., and Tilbury D. M. (2002) “Using deadbands to reduce communication in networked control systems,” Proc. Am. Control Conf., vol. 4, pp. 3015–3020.

Poveda J. I. and Teel A. R. (2017) “A Robust Event-Triggered Approach for Fast Sampled-Data Extremization and Learning,” IEEE Trans. Automat. Contr., 62(10), 4949–4964.

Sebastián Dormido, Manuel Beschi, José Sánchez, Antonio Visioli (2012) “An Interactive Software Tool for the Study of Event-based PI Controller”, IFAC Proceedings vol. 45(3), pp.164-169.

Skogestad. (2004) “Simple analytic rules for model reduction and PID controller tuning,” Model. Identif. Control, vol. 25, no. 2, pp. 85–120.

Silvano Seva, William Fornaciari, Alberto Leva (2021) “Event-Based Control Enters the Real-Time World: Perspectives and Pitfalls”, Second Workshop on Next Generation Real-Time Embedded Systems (NG-RES 2021), pp.4:1-4:11.

Tiberi U, Araújo J, and Johansson K. H. (2012) “On event-based PI control of first-order processes,” IFAC Proceedings, 2(1), 448–453.

Xiaoyu Li, Xiangye Zeng, Jingyi Wang, Qi Li, Baoshuo Fan and Qi Zeng (2025) “Analysis of Unmanned Surface Vehicles Heading KF-Based PI-(1+PI) Controller Using Improved Spider Wasp Optimizer”, MDPI Journal Drones, vol. 9(5), pp.326-45.

Zamarreño J. M, Mazaeda R, Caminero J. A., Rivero A. J. and Arroyo J. C. (2014) “A new plug-in for the creation of OPC servers based on EcosimPro© simulation software,” Simulation Modelling Practice and Theory, 40, 86–94.