Scientific Research Publishing

Engineering, 2020, 12, 623-639
https://www.scirp.org/journal/eng
ISSN Online: 1947-394X
ISSN Print: 1947-3931

Implementation of Control Algorithms in Small Embedded Systems

Lars Bengtsson

Department of Physics, University of Gothenburg, Gothenburg, Sweden
Email: lars.bengtsson@physics.gu.se

How to cite this paper: Bengtsson, L. (2020) Implementation of Control Algorithms in Small Embedded Systems. *Engineering*, **12**, 623-639.
https://doi.org/10.4236/eng.2020.129044

Received: August 18, 2020
Accepted: September 11, 2020
Published: September 14, 2020

Abstract

This work describes how a control algorithm can be implemented in a small (8-bit) microcontroller for the main purpose of merging embedded systems and control theory in electrical engineering undergraduate classes. Two different methods for discretizing the control expression are compared: Euler transformation and bilinear transformation. The sampling rate's impact on the algorithm is discussed and theoretical results are verified by an application to a temperature control system in a heating plant. Four control algorithms are compared: PID and PI algorithms discretized with Euler and bilinear transformation, respectively. It is shown that for the heating plant used in this work, a bilinear PI algorithm implemented in a small 8-bit microcontroller outperforms a commercial controller from Panasonic. It is also demonstrated that all the derived algorithms can be implemented using integer calculations only, obviating the need for expensive and time-consuming floating-point calculations. This work bridges the gap between control theory equations and the implementation of control systems in small embedded systems with no inherent floating-point processing power.

Keywords

Bilinear Transformation, Euler Transformation, Microcontroller, Control System, PID Algorithm, Temperature Sensor, Set Value, Process Value, Plant

1. Introduction

Control theory and embedded systems are typically treated separately in electrical engineering programs and the implementation of control algorithms in digital computing targets is only "indicated" on a block diagram level. This work presents the details of the implementation of control algorithms in a small microcontroller which at the same time offers a challenging application for an em-

bedded systems class.

Control systems are one of the most frequently occurring electro-mechanical systems in process industry [1]. Temperature, gas and liquid flow, liquid levels and rotation speeds (revolutions) are only some of the physical quantities that are frequently controlled. "Controlled" refers to the fact that the current *process value* is measured (by a *sensor*) and fed back and compared with the intended value (the *set value*) and the control system's task is to eliminate the difference between the set value and the process value. This is illustrated in **Figure 1**.

From **Figure 1**, we can see that the input to the control algorithm is the *error signale* (t), *i.e.* the difference between the set value and the process value. The control algorithm is typically a *PI*- or a *PID*-system. *P* is the *proportional band*; the smaller *P* is, the greater is the amplification K_P (of $e(t)$). *I* represents the integrating part of the control algorithm. This part is necessary in order to eliminate "remaining errors" (otherwise the set and process values may differ even in steady state). Finally, *D* represents the differential part of the algorithm. This part has two objectives; it makes the system more agile to abrupt changes in the set value or perturbations, but most of all, it helps to stabilize the system (all feedback systems run the risk of being instable).

The transfer functions of a PI and a PID system are represented by Equation (1) and Equation (2), respectively.

$$G(s) = K_P \times \left(1 + \frac{1}{T_I s}\right) \tag{1}$$

$$G(s) = K_P \times \left(1 + \frac{1}{T_I s} + T_D s\right) \tag{2}$$

In Equation (1) and Equation (2), s is the Laplace variable ($s = \alpha + j\omega$), K_P is the amplification, T_I is the integration time and T_D is the derivation time. The $\{K_P, T_I, T_D\}$ set is referred to as the *PID parameters* and they need to be determined first; they depend on the physical properties of the process.

The "process" can be very simple or extremely complex, but most systems are approximated with either a first or a second order system and these models are necessary in order to determine the control parameters. This work is not primarily concerned with finding the PID parameters; the primary concern of this work is how to implement expressions such as Equation (1) and Equation (2) into a small embedded system. Having said that, control parameters are typically derived either by rule of thumb [2], by examining the step response [3] [4] or by more advanced *system identification* methods [5].

In this work, Ziegler-Nichols' step response method was applied in order to derive the PID parameters [6]. First, the open-loop step response of the process is registered, see **Figure 2**.

In a digital solution, y is the output from an n_y-bit ADC (Analog-to-Digital Converter), *i.e.* an integer in the range 0 and $2^{n_y} - 1$. Correspondingly, the input step signal u is the output from an n_u-bit DAC (Digital-to-Analog Converter):

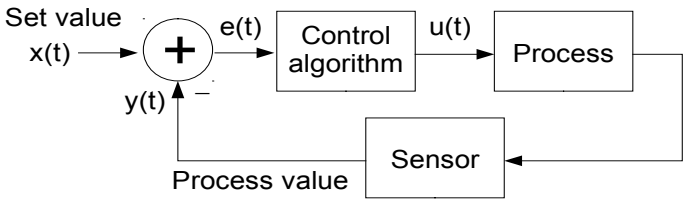

Figure 1. Basic control system.

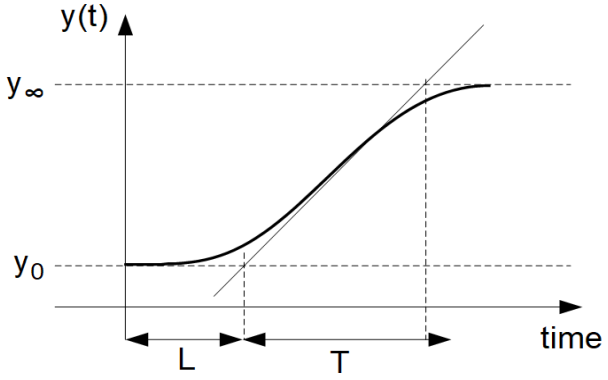

Figure 2. Open loop step response of second order system.

an integer in the range $0 \cdots 2^{n_u} - 1$. If a stepwise input change from u_0 to u_∞ results in an output change from y_0 to y_∞, the open-loop gain is

$$K = \frac{y_\infty - y_0}{u_\infty - u_0} \tag{3}$$

L and T in **Figure 2** are the inherent deadtime and the time constant of the process, respectively. These parameters must be measured in a step response experiment and the parameter set {K, L, T} is used to find the appropriate PID parameters according to **Table 1** [6].

Review of Previous/Related Works

Implementation of control algorithms in single-chip microcontrollers has been reported ever since they were introduced on the market. All ranges and brands have been used. For example, 8-bit microcontrollers were used in [7]-[12], Aslam *et al.* used a 16-bit PIC controller [13] and Krivic *et al.* [14] and Arzak *et al.* [15] used 32-bit ARM controllers. A lot of PID implementations have been reported in FPGAs, either using soft CPUs like Xlinix's Pico Blaze [16] [17] or Altera's Nios processor [18]. Direct implementations in embedded hardware, such as FPGAs [18] [19] and FPAAs [20] have also been reported.

As expected, the most common control implementations are concerned with temperature [7] [18] [21] and DC engines [8] [9] [10] [13] [16] [17], but water level [14], DC-DC output voltage [12] and position control of magnetically levitated balls [11] have been reported. One of the most salient works is the implementation of a microcontroller PID system for controlling the movement of a wheelchair by EEG waves (for quadriplegic patients) [15].

Table 1. PID parameters from step response.

Controller	PID parameters		
	K_P	T_I	T_D
PI	$0.9\,T/KL$	$3L$	
PID	$1.2\,T/KL$	$2L$	$L/2$

In all reported works, about half of them implemented a straightforward Euler-transformed PID (Equation (9) below) [7] [10] [13] [14] [18] [19] [21] and some used Fuzzy logic algorithms [8] [14]. Kheriji suggested a Model Predicted Control algorithm which is an extension to the Euler-transformed PID [22].

In spite of an extensive data base search, no works have been found that utilize bilinear transformation of control algorithms; Euler transformation is apparently prevalent. This work remediates this gap and demonstrates the advantages of bilinear transformation. There is also a paucity of implementation details in previous works; exactly how are control equations translated into C code (without floating-point support) and exactly how is the hardware details designed. For example, how are the algorithm details affected by analogue-to-digital and digital-to-analogue converters? This work will cover all these implementation details.

2. Method and Material

2.1. Theory

The problem addressed in this work is how control equations, such as those represented by Equation (1) and Equation (2), are best implemented in a small, 8-bit microcontroller (non-digital signal-processing chip). Two methods are proposed and compared: *Euler transformation* and *bilinear transformation*.

2.1.1. Euler Transformation

In Equation (1) and Equation (2), $G(s)$ represents the transfer function from $e(t)$ to $u(t)$ in **Figure 1** (*i.e.* $U(s)/E(s)$). Taking the inverse Laplace transform of expressions (1) and (2) gives us the time domain signals:

$$u(t) = K_P \times \left(e(t) + \frac{1}{T_I} \int e(t)\,\mathrm{d}t \right) \tag{4}$$

$$u(t) = K_P \times \left(e(t) + \frac{1}{T_I} \int e(t)\,\mathrm{d}t + T_D \times e'(t) \right) \tag{5}$$

In **Figure 3**, the error signal $e(t)$ has been sampled with a sampling time T_S.

From **Figure 3**, it is obvious that the integral of $e(t)$ can be approximated by the sum of the indicated rectangles, *i.e.*

$$\int e(t)\,\mathrm{d}t \approx \sum\nolimits_i e(i) \times T_S = T_S \sum\nolimits_i e(i). \tag{6}$$

Correspondingly, in **Figure 4**, we can see that the derivative of $e(t)$ at some point may be approximated by the gradient of the straight line between two adjacent samples, *i.e.*

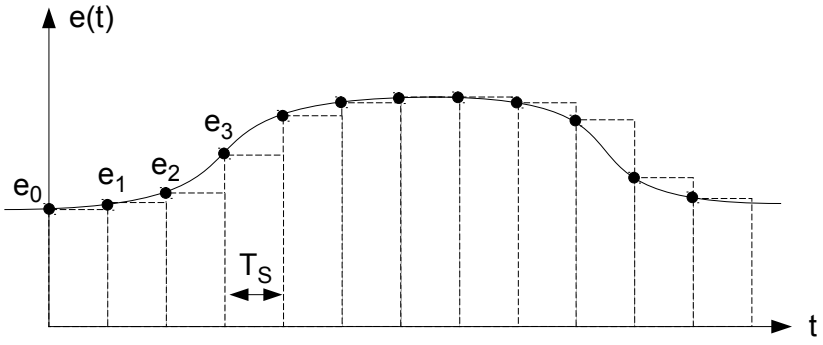

Figure 3. The integral is approximated with the sum of rectangles.

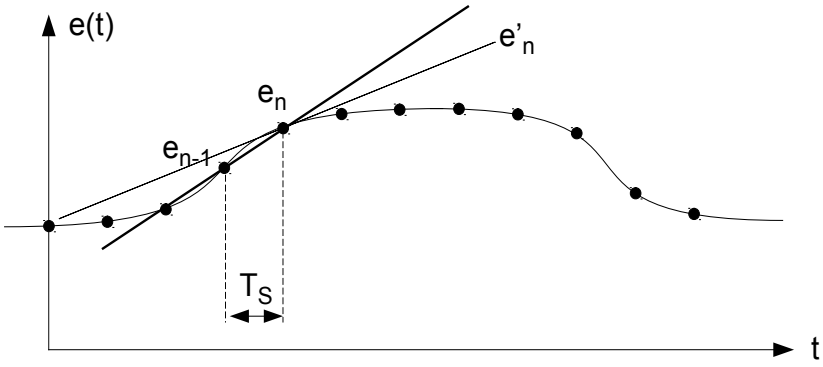

Figure 4. The derivative is approximated by the straight line between two adjacent samples.

$$e'(t) \approx e'(n) = \frac{e(n) - e(n-1)}{T_S}. \tag{7}$$

Hence, when we discretize the control algorithms (4) and (5) using Euler transformations, we get the following expressions:

$$u(n) = K_P \times \left(e(n) + \frac{T_S}{T_I} \sum_{i=0}^{n} e(i) \right) \tag{8}$$

$$u(n) = K_P \times \left(e(n) + \frac{T_S}{T_I} \sum_{i=0}^{n} e(i) + \frac{T_D}{T_S} \left(e(n) - e(n-1) \right) \right) \tag{9}$$

Expressions (8) and (9) are "computer-friendly" expressions that can be readily implemented in an embedded system. Notice how the sampling time becomes a crucial design parameter.

2.1.2. Bilinear Transformation

In the Euler transform, we first apply the inverse (Laplace) transform and then discretize the time. In the bilinear transformation, we do exactly the opposite; we first transform the continuous-frequency expressions (1) and (2) to the discrete-frequency domain and then apply the inverse transformation to get the time-expressions for the computer algorithm.

The discrete-time correspondence to the Laplace transform is the *z* transform

[23]. In the z transform, the frequency variable is the *normalized* frequency variable Ω (normalized to the sampling rate):

$$\Omega = \frac{\omega}{f_S} = \omega \times T_S \tag{10}$$

Since ω is limited to frequencies $< 2\pi \times f_s/2$, due to the sampling theorem [24], Ω is limited to $\pm\pi$ (z transforms are periodic with period 2π to represent the periodicity of sampled signals in frequency space).

Hence, a transformation from continuous-time to discrete-time becomes cumbersome, since only a limited number of frequencies can be carried over to a discrete-time representation. To overcome this, we have to "squeeze" in *all* frequencies in the continuous time-domain, from $-\infty$ to $+\infty$, into the $\pm\pi$ interval in the z domain. The following expression accomplishes this:

$$\Omega = 2 \times \arctan\left(k\omega\right) \tag{11}$$

Equation (11) maps ω to Ω as:

$\omega = -\infty \to \Omega = -\pi$;

$\omega = 0 \to \Omega = 0$;

$\omega = +\infty \to \Omega = +\pi$.

This is illustrated in **Figure 5**.

In Equation (11), we solve for ω (and multiply both sides by the imaginary number j):

$$\begin{aligned} \mathrm{j}\omega &= \mathrm{j}\frac{1}{k}\tan\left(\frac{\Omega}{2}\right) = \frac{1}{k} \times \frac{\mathrm{j} \times \sin\frac{\Omega}{2}}{\cos\frac{\Omega}{2}} = \frac{1}{k} \times \frac{\frac{1}{2}\left(\mathrm{e}^{\mathrm{j}\Omega/2} - \mathrm{e}^{-\mathrm{j}\Omega/2}\right)}{\frac{1}{2}\left(\mathrm{e}^{\mathrm{j}\Omega/2} + \mathrm{e}^{-\mathrm{j}\Omega/2}\right)} \\ &= \frac{1}{k} \times \frac{\mathrm{e}^{-\mathrm{j}\Omega/2}\left(\mathrm{e}^{\mathrm{j}\Omega} - 1\right)}{\mathrm{e}^{-\mathrm{j}\Omega/2}\left(\mathrm{e}^{\mathrm{j}\Omega} + 1\right)} = \frac{1}{k} \times \frac{\mathrm{e}^{\mathrm{j}\Omega} - 1}{\mathrm{e}^{\mathrm{j}\Omega} + 1} \end{aligned} \tag{12}$$

For $\alpha = 0$, $s = \mathrm{j}\omega$ and we can write Equation (12) as

$$s = \frac{1}{k} \times \frac{z-1}{z+1} \tag{13}$$

where $z = \mathrm{e}^{sT_s} = \mathrm{e}^{(\alpha+\mathrm{j}\omega)T_s} = \mathrm{e}^{\mathrm{j}\Omega}$ if $\alpha = 0$. Equation (13) represents the classic bilinear transformation that transfers a Laplace transform expression into the corresponding z transform expression.

However, we are not quite done yet; we need to determine the constant k in Equation (13). We determine it so that we get a good mapping for the lowest frequencies; if Ω is "small" then, $\tan(\Omega/2) \approx \Omega/2$ and expression (12) is now

$$\omega = \frac{1}{k} \times \frac{\Omega}{2} \tag{14}$$

Solving for k, and remembering that Ω is the normalized frequency variable, we get:

$$k = \frac{\Omega}{2\omega} = \frac{\omega T_S}{2\omega} = \frac{T_S}{2} \tag{15}$$

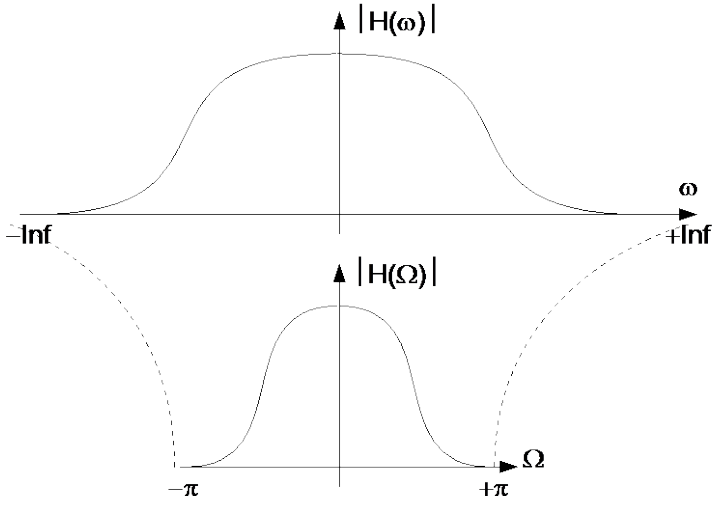

Figure 5. Bilinear transformation compresses the frequency axis.

Combining Equation (13) and Equation (15), we finally get our transformation expression:

$$s = \frac{2}{T_S} \times \frac{z-1}{z+1} \tag{16}$$

This is the substitution we need to do in expressions (1) and (2) to translate the control frequency functions to discrete-time (sampled) space. We do Equation (1) first:

$$\begin{aligned} G(z) &= K_P \times \left(1 + \frac{1}{\frac{2T_I}{T_S}\frac{z-1}{z+1}}\right) = K_P\left(1 + \frac{T_S}{2T_I}\frac{z+1}{z-1}\right) \\ &= K_P\left(\frac{2T_I}{2T_I}\frac{z-1}{z-1} + \frac{T_S}{2T_I}\frac{z+1}{z-1}\right) \\ &= \frac{K_P}{2T_I}\left(2T_I(z-1) + T_S(z+1)\right) \times \frac{1}{z-1} \\ &= \frac{K_P}{2T_I}\left(z(2T_I + T_S) - (2T_I - T_S)\right) \times \frac{1}{z-1} \\ &= \frac{K_P}{2T_I}\left((2T_I + T_S) - (2T_I - T_S)z^{-1}\right) \times \frac{1}{1-z^{-1}} \end{aligned} \tag{17}$$

Equation (1) represents the transfer function from $e(t)$ to $u(t)$ in **Figure 1**. Hence,

$$G(z) = \frac{U(z)}{E(z)} \tag{18}$$

$$\frac{K_P}{2T_I}E(z)\left(2T_I + T_S - (2T_I - T_S)z^{-1}\right) = U(z)\left(1 - z^{-1}\right) \tag{19}$$

Taking the inverse z transform of Equation (19), and remembering that in the z transform, z^{-m} represents a time delay of m sample times ([25], p. 345), we get

the difference equation we are looking for:

$$u_n = u_{n-1} + \frac{K_P}{2T_I}\left(\left(2T_I + T_S\right)e_n - \left(2T_I - T_S\right)e_{n-1}\right) \tag{20}$$

Notice an important difference between the Euler expression (8) and the bilinear expression (20); in Equation (20) we need old samples of the control signal (u_{n-1}). This implies poles in the z plane, *i.e.* the system could become instable (if the poles are outside the unit circle in the z plane). With the Euler transform, we always get inherently stable systems (no poles).

Next, we need to do the same substitution in Equation (2). This will give us the following expression:

$$\begin{aligned} u_n = u_{n-2} + \frac{K_P}{2T_I T_S}\Big\{&\left(2T_I T_S + T_S^2 + 4T_I T_D\right)e_n + \left(2T_S^2 - 8T_I T_D\right)e_{n-1} \\ &+ \left(4T_I T_D + T_S^2 - 2T_I T_S\right)e_{n-2}\Big\} \end{aligned} \tag{21}$$

2.2. Hardware

The hardware setup is illustrated in Figure 6. The heart of the control system is a microcontroller from Microchip; the 20-pin, 8-bit PIC16F1769 mid-range controller [26] (list price is $1.92). The process to control (the "plant") is the temperature of a power resistor. Its temperature is measured by a temperature sensor, LM35 [27], and fed back to the control algorithm. The sensor's sensitivity is 10 mV/˚C and its signal is amplified 10 times by a non-inverting amplifier (taking advantage of an op amp embedded into the microcontroller). The output

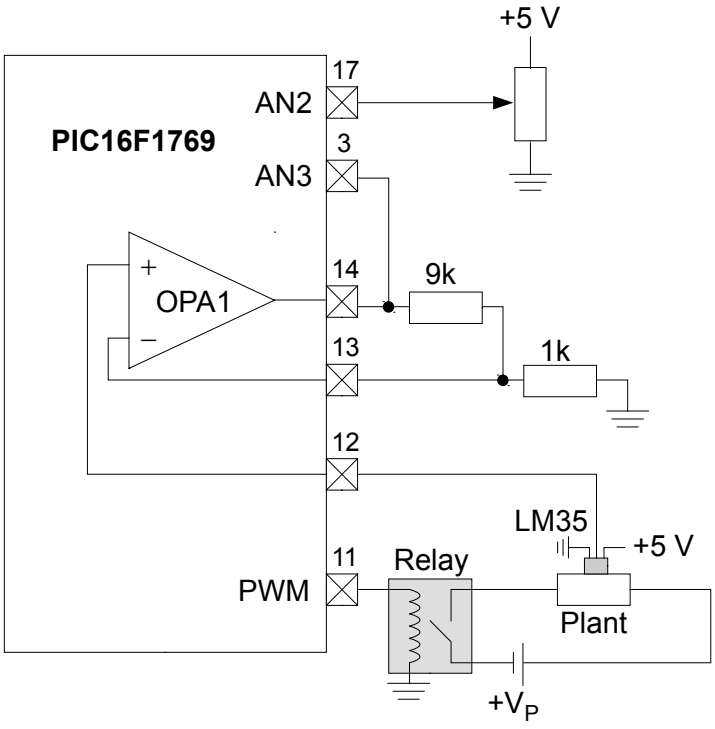

Figure 6. Microcontroller-based control system.

is connected to one of the analog input channels; this represents the process value and since the maximum allowed input is +5 volts, the temperature range is 0..50˚C.

The output from the control algorithm is sent to a pulse width modulated (PWM) output (period = 4.4 seconds). This output is connected to a semiconductor relay [28] that controls the dc power source (V_p) that heats the plant resistor. The V_P voltage was adjusted so that a PWM duty cycle of 100% would correspond to a plant temperature of +50˚C; V_P = 8.0 volts. The set value is adjusted by a 10-turn potentiometer connected to one of the analog inputs (AN2) and the plant temperature is registered on analog channel AN3.

2.3. Software

The PWM duty cycle value can only be updated at the end of each period and hence it makes no sense to read the set and plant temperatures more often than the PWM period (*i.e.* the sampling time equals the PWM period). The software was designed to generate an interrupt at the end of each PWM period and the control algorithms were implemented in the ISR (interrupt service routine; the software is background/foreground oriented ([29], pp. 83-84)) and the ISR is illustrated in **Figure 7**.

The heart of the algorithm is the calculation of the new output value and the update of the old samples. This part of the C code is illustrated below.

```
u_temp = u2 + 5892*e–7154*e1 + 2172*e2;     //calculate new
    if(u_temp > 65535)                      //limit the output to
    u_temp = 65535;                         //a 16-bit range
else if(u_temp < 0)
    u_temp = 0;
u = (unsigned int)u_temp;                   //type conversion
PWM6_DutyCycleSet(u);
PWM6_LoadBufferSet();                       //Update output
e2 = e1;                                    //Update old samples
e1 = e;
u2 = u1;
u1 = u;
```

The program was developed in Microchip's MPLABX® IDE using the XC8 C compiler. The microcontroller ran at 16 MHz and the first C code line above (calculating the new output value), was timed in the simulator to take 52 μs.

2.4. Step Response

The open-loop step response was measured using two analog channels on a Tektronix MSO2012B oscilloscope. Since the relay that controls the plant is controlled by a PWM signal, an input step corresponds to a step change in the PWM signal's duty cycle. The open-loop step response was recorded (on pin 3/14 in **Figure 6**) by the oscilloscope and data was transferred to MATLAB via a USB

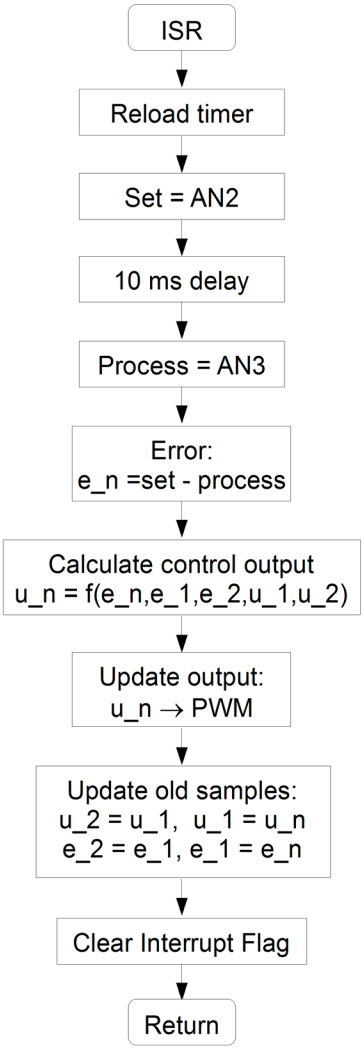

Figure 7. Software flowchart of control system.

memory stick. Graphs were plotted in MATLAB to facilitate a convenient estimation of the K, T and L parameters in Figure 2 and Table 1. The input step change was generated by changing the PWM duty cycle from 20% to 60%.

2.5. Performance/Verification

The performance of the derived control algorithms was evaluated by registering the control system's response to 1) a positive step change in the set value (from 30˚C to 40˚C), 2) a negative step change in the set value (from 40˚C to 30˚C) and finally, 3) a perturbation test where the plant was subjected to an instant cooling perturbation (using cold spray). All responses were observed for 17 minutes.

Also, in order to benchmark our control algorithms, their performance was compared to that of a commercial temperature controller from Panasonic (KT4) [30]. The Panasonic controller controlled the temperature of the same power resistor. The only difference in the setup was that the Panasonic controller only

accepts thermocouples or Pt-100 temperature sensors as input (a type K thermocouple was used); the thermocouple was attached to the power resistor (but the LM35 resistor was not removed, it was used to register the temperature on the oscilloscope). Other than that, the exact same setup was used except that the relay in Figure 6 was removed; the Panasonic controller also has a PWM output with an internal relay. The Panasonic controller was first set to "autotuning mode" where it evaluates the connected systems and finds its own PID parameters (by some built-in rules of thumb).

3. Results

3.1. Control Parameters

Figure 8 illustrates the open-loop step response when the input PWM duty cycle ($u(t)$) changes instantly from 20% to 60%. (In Figure 8, the step signal is the output from an auxiliary I/O pin that only marks the start of the step change in the PWM signal; its amplitude has been manipulated in MATLAB to fit the graph).

From Figure 8, we can see that the output changes from 3.03 volts to 3.95 volts. For a 10-bit ADC, with a reference voltage of 5.0 volts, that corresponds to integers

$$3.03\times\frac{2^{10}}{5.0}=621 \quad \text{and} \quad 3.95\times\frac{2^{10}}{5.0}=809$$

The PWM input signal has a resolution of 16 bits and hence a duty cycle change from 20% to 60% corresponds to integers

$$0.2\times 2^{16}=13107 \quad \text{and} \quad 0.6\times 2^{16}=39322$$

From Equation (3), we get the open-loop amplification:

$$K=\frac{809-621}{39322-13107}=0.0071$$

Also, from close inspection of Figure 8, we get L = 9 seconds and T = 100 seconds. Finally, combining these parameters with Table 1 and expressions (8), (9), (20) and (21), we get four different control algorithms according to Table 2. (Notice that Table 2 indicates that only integer operations are required).

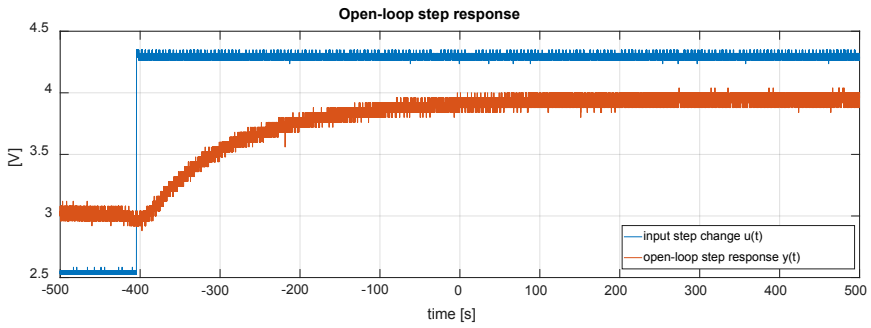

Figure 8. Open-loop step response.

Table 2. Suggested control algorithms.

Controller	u_n
Euler, PI	$1395e_n + 277 \times \sum_{i=0}^{n} e_i$
Euler, PID	$1860e_n + 455 \times \sum_{i=0}^{n} e_i + 1902 \times (e_n - e_{n-1})$
Bilinear, PI	$u_{n-1} + 1509e_n - 1281e_{n-1}$
Bilinear, PID	$u_{n-2} + 5892e_n - 7154e_{n-1} + 2172e_{n-2}$

3.2. Performance, Microcontroller

Figures 9(a)-(d) illustrate the (positive) step responses for each one of the four controllers presented in **Table 2**.

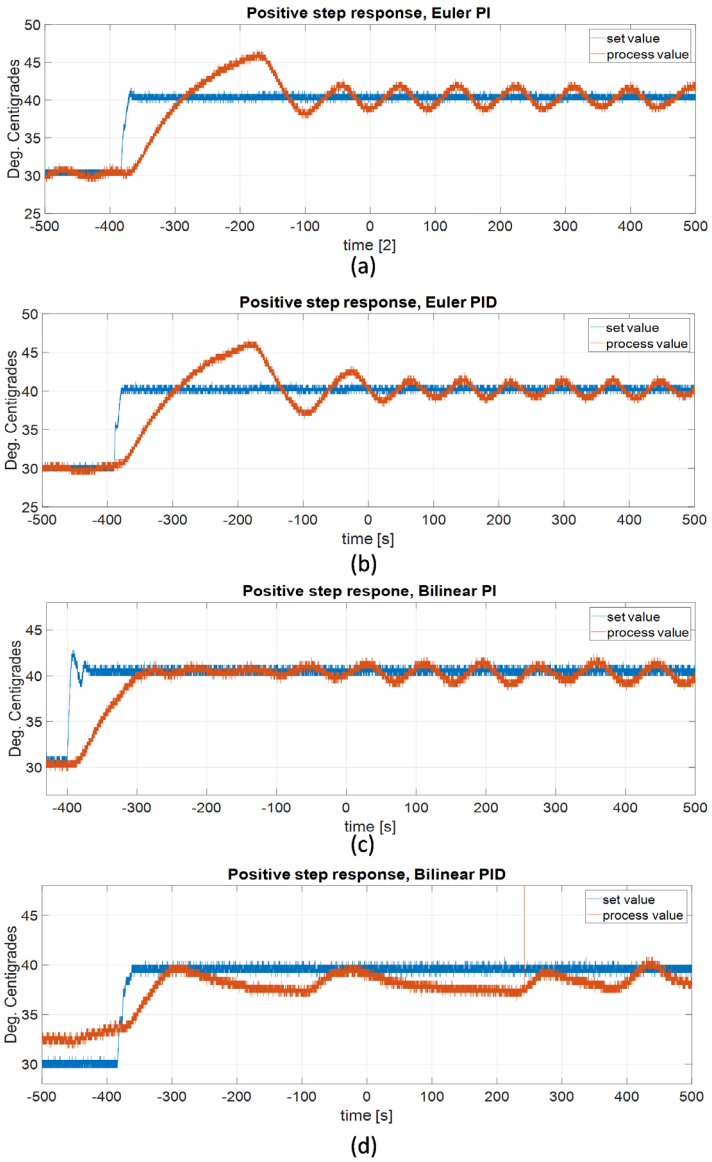

Figure 9. (a) The positive step response of Euler PI algorithm; (b) The positive step response of Euler PID algorithm; (c) The positive step response of bilinear PI algorithm; (d) The positive step response of bilinear PID algorithm.

In the Euler case, the PID algorithm appears to be slightly better than the PI algorithm; both oscillate a little in the steady state, but the oscillation amplitude is slightly less in the PID case (≈1˚C).

For the algorithms based on bilinear transformation, the PID algorithm is outperformed by the PI algorithm. Overall, the bilinear PI algorithm exhibits the best performance; it has the same steady-state oscillation as the Euler PID algorithm, but much smaller initial overshoot.

Figure 10 below illustrates the negative step response of the bilinear PI algorithm and Figure 11 illustrates the perturbation response.

3.3. Performance, Panasonic Controller

After the autotuning was completed, the Panasonic PID parameters were set to P = 22.2˚C, I = 34 seconds and D = 8 seconds. Figures 12-14 below illustrate the positive and negative step responses and the perturbation response of the Panasonic controller.

Figures 12-14 should primarily be compared to Figure 9(c), Figure 10 and Figure 11, which illustrate the corresponding performance of the bilinear PI algorithm.

4. Discussion

From Figures 9-11, it must be concluded that the bilinear PI algorithm exhibits the overall best performance of all the four suggestions in Table 2. From Figures 12-14,

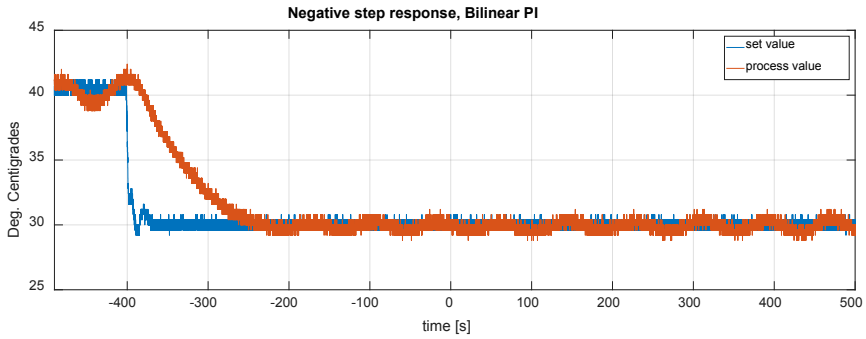

Figure 10. The negative step response of the bilinear PI algorithm.

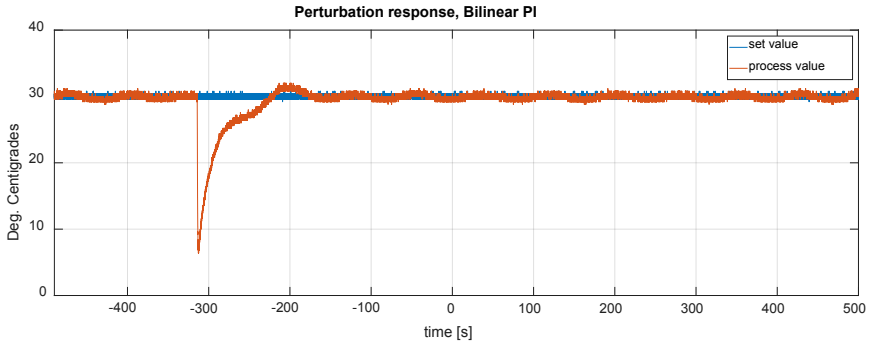

Figure 11. The perturbation response of the bilinear PI algorithm.

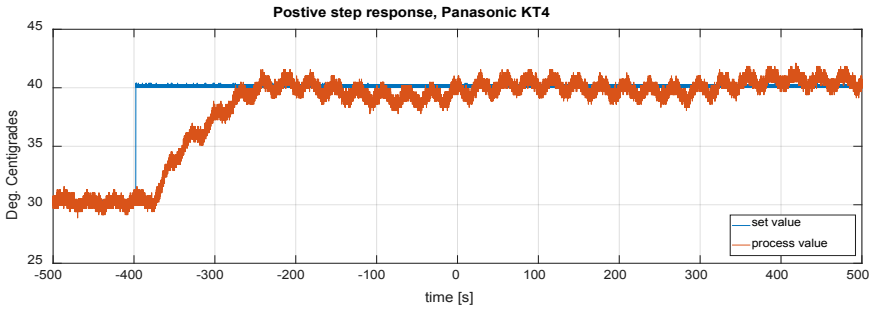

Figure 12. The positive step response of Panasonic controller.

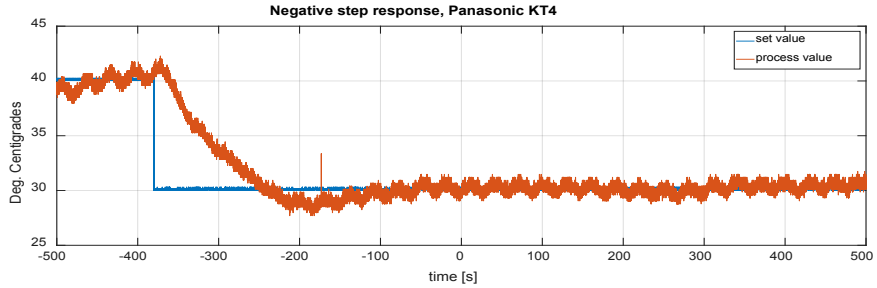

Figure 13. The negative step response of the Panasonic controller.

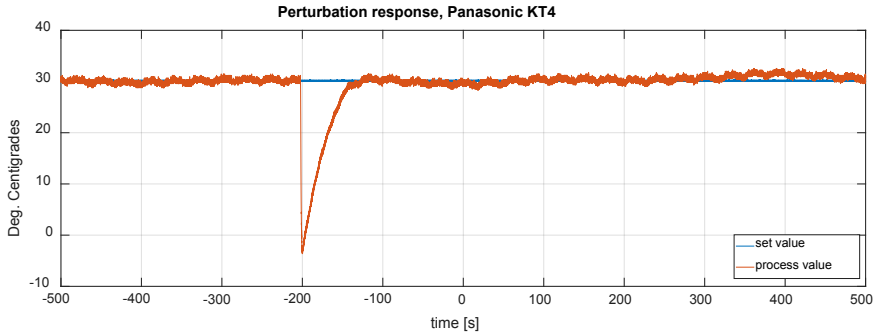

Figure 14. The perturbation response of the Panasonic controller.

it is also clear that it performs even better than a commercial temperature controller from Panasonic.

In all controllers, the plant temperature oscillates a little in steady state. In Figure 9(c), the amplitude of the oscillations seems to increase with time. For that reason, the output of this controller was carefully monitored for two hours and the oscillations petered out almost completely.

The first version of this design did not work as expected; the microcontroller was not able to control the temperature. After extensive debugging, the error was found in the software; the delay between the sampling of AN2 and AN3 was inserted. According to the datasheet ([26], pp. 185-186), there is only one ADC; all channels use the same sample-and-hold circuit and an analog multiplexer forwards the analog channels to the ADC. Hence, the shared sample-and-hold capacitor must be allowed time to charge/de-charge between channel switching

and for that reason a delay needs to be inserted between readings of different channels. (10 ms is an overkill but compared to the system's time constant it is negligible).

5. Conclusion

This work has presented detailed instructions on how to implement PI and PID algorithms into small embedded systems and it has been proven that a bilinear transformation of the control expressions yields a better controller performance than the general modus operandi of Euler transformations. The reason why a bilinear transformation has better performance than the Euler transformation lies probably in the fact that the bilinear transformation produces algorithms with feedback; they utilize previous samples of the output signal. Herein lies also the weakness; control algorithms derived by bilinear transformation may become inherently unstable (which cannot happen with Euler algorithms). This work has also demonstrated that a complete controller can be implemented in an inexpensive, analog/digital hybrid 8-bit microcontroller, costing less than $2. That will make it less expensive than any other implementations in ARM- or FPGA-based solution and in terms of performance, it has been demonstrated that it can outperform a commercial desktop controller. This also makes it readily available for undergraduate classes in electrical engineering. Here, the application was a temperature plant, but the same system has been successfully implemented also on a water level plant and the fact that the arithmetic is based on integer calculations indicate that it should not be a problem to apply it also to DC motor control. The fact that only integer calculations are required (see **Table 2**) eliminates the need for expensive floating-point processors.

Conflicts of Interest

The author has no conflicts of interest to report.

References

[1] Harris, T., Seppala, C. and Desborough, L. (1999) A Review of Performance Monitoring and Assessment Techniques for Univariate and Multivariate Control Systems. *Journal of Process Control*, **9**, 1-17. https://doi.org/10.1016/S0959-1524(98)00031-6

[2] Ziegler, J. and Nichols, N. (1942) Optical Settings for Automatic Controllers. *Transactions of the American Society of Mechanical Engineers*, **64**, 759-768.

[3] Ahmed, S., Huang, B. and Shah, S. (2007) Novel Identification Method from Step Response. *Control Engineering Practice*, **15**, 545-556. https://doi.org/10.1016/j.conengprac.2006.10.005

[4] Bi, Q., Cai, W.-J., Lee, E.-L., Wang, Q.-G., Hang, C.-C. and Zhang, Y. (1999) Robust Identification of First-Order plus Dead-Time Model from Step Response. *Control Engineering Practice*, **7**, 71-77. https://doi.org/10.1016/S0967-0661(98)00166-X

[5] Ljung, L. (1999) System Identification. Prentice-Hell, New Jersey. https://doi.org/10.1002/047134608X.W1046

[6] Basilo, J. and Matos, S. (2002) Design of PI and PID Controllers with Transient

Performance Specification. *IEEE Transactions on Education*, **45**, 364-370. https://doi.org/10.1109/TE.2002.804399

[7] Chesaru, V., Dan, C. and Bodea, M. (2006) PID Algorithm for Controller Processors. 2006 *International Semiconductor Conference*, Sinaia, 27-29 September 2006, 429-432. https://doi.org/10.1109/SMICND.2006.284037

[8] Sarin, S., Hindersah, H. and Prihatmanto, A. (2012) Fuzzy PID Controllers Using 8-Bit Microcontroller for U-Board Speed Control. 2012 *International Conference on System Engineering and Technology*, Bandung, 11-12 September 2012, 1-6. https://doi.org/10.1109/ICSEngT.2012.6339355

[9] Uzunovic, T., Zunic, E., Badnjevic, I., Miokovic, I. and Konjicia, S. (2010) Implementation of Digital PID Controller. *The* 33*rd International Convention MIPRO*, Opatija, 24-28 May 2010, 1357-1361.

[10] Xu, C., Huang, D.G., Huang, Y.P. and Gong, S.G. (2008) Digital PID Controller for Brushless DC Motor Based on AVR Microcontroller. 2008 *IEEE International Conference on Mechatronics and Automation*, Takamatsu, 5-8 August 2008, 247-252.

[11] Kalúz, M., Klauco, M. and Kvasnica, M. (2015) Real-Time Implementation of a Reference Governor on the Arduino Microcontroller. 2015 20*th International Conference on Process Control*, Strbske Pleso, 9-12 June 2015, 350-356. https://doi.org/10.1109/PC.2015.7169988

[12] Boudreaux, R.R., Nelms, R.M. and Hung, J.Y. (1997) Simulation and Modeling of a DC-DC Converter by an 8-Bit Microcontroller. *Proceedings of APEC* 97*—Applied Power Electronics Conference*, Atlanta, 27 February 1997, 963-969.

[13] Aslam, S., Hannan, S., Sajjad, U. and Zafar, W. (2016) Implementation of PID on PIC24F Series Microcontroller for Speed Control of a DC Motor Using MPLAB and Proteus. *Advances in Science and Technology Research Journal*, **10**, 40-50. https://doi.org/10.12913/22998624/64060

[14] Krivic, S., Hujdur, M., Mrzic, A. and Konjicija, S. (2012) Design and Implementation of Fuzzy Controller on Embedded Computer for Water Level Control. 2012 *Proceedings of the* 35*th Convention MIPRO*, Opatija, 21-25 May 2012, 1747-1751.

[15] Arzak, M., Sunarya, U. and Hadiyoso, S. (2016) Design and Implementation of Wheelchair Controller Based on Electroencephalogram Sinal Using Micronctroller. *International Journal of Electrical and Computer Engineering*, **6**, 2878-2886.

[16] Das, A. and Banerjee, K. (2009) Fast Prototyping of a Digital PID Controller on a FPGA Based Soft-Core Microcontroller for Precision Control of a Brushed DC Servo Motor. 2009 35*th Annual Conference on IEEE Industrial Electronics*, Porto, 3-5 November 2009, 2825-2830. https://doi.org/10.1109/IECON.2009.5415406

[17] Youness, H., Moness, M. and Khaled, M. (2014) MPSoCs and Multicore Microcontrollers for Embedded PID Control: A Detailed Study. *IEEE Transactions on Industrial Informatics*, **10**, 2122-2133. https://doi.org/10.1109/TII.2014.2355036

[18] Chan, Y., Moallem, M. and Wang, W. (2007) Design and Implementation of Modular FPGA-Based PID Controllers. *IEEE Transactions on Industrial Electronics*, **54**, 1898-1906. https://doi.org/10.1109/TIE.2007.898283

[19] Gosh, S., Barai, R., Bhattarcharya, S., Bhattarcharya, P., Rudra, S., Dutta, A. and Pyne, R. (2013) An FPGA Based Implementation of a Flexible Digital PID Controller for Motion Control System. 2013 *International Conference on Computer Communication and Informatics*, Coimbatore, 4-6 January 2013, 1-6. https://doi.org/10.1109/ICCCI.2013.6466277

[20] Lita, I., Visan, A. and Cioc, I. (2009) FPAA Based PID Controller with Application

in the Nuclear Domain. 2009 32*nd International Spring Seminar of Electronics Technology*, Brno, 13-17 May 2009, 1-4. https://doi.org/10.1109/ISSE.2009.5206979

[21] Moallem, M. (2004) A Laboratory Testbed for Embedded Computer Control. *IEEE Transactions on Education*, **47**, 340-347. https://doi.org/10.1109/TE.2004.825054

[22] Kheriji, A., Bouani, F., Ksouri, M. and Ahmed, M. (2011) A Microcontroller Implementation of Model Predictive Control. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, **5**, 600-606.

[23] Bohner, M. and Peterson, A. (2002) Laplace Transform and Z-Transform: Unification and Extension. *Methods and Applications of Analysis*, **9**, 151-158. https://doi.org/10.4310/MAA.2002.v9.n1.a6

[24] Shannon, C. (1949) Communication in the Presence of Noise. *Proceedings of the Institute of Radio Engineers*, **37**, 10-21. https://doi.org/10.1109/JRPROC.1949.232969

[25] Proakis, J. and Manoakis, D. (2004) Digital Signal Processing. 2nd Edition, Macmillan Publishing Company, New York.

[26] Microchip Technology Inc. (2018) PIC16(L)F1764/5/8/9. Microchip Tech. Inc., Tucson.

[27] Analog Devices (2015) TMP35/TMP36/TMP37 Low Voltage Temperature Sensors. Rev. H ed., Norwood, Maine.

[28] Panasonic, S. (2017) Relays—Polarized Power Relays. ASCTB207E ed., Osaka.

[29] Laplante, P. (2004) Real-Time Systems Design and Analysis. John Wiley & Sons, Inc., Hoboken. https://doi.org/10.1002/0471648299

[30] Panasonic (2020) Panasonic Industrial Automation. https://www3.panasonic.biz/ac/na_download/fasys/component/temperature_controller/manual/kt4_manual_e.pdf?f_cd=3614