



LOW COST HARDWARE IN THE LOOP CONTROL SYSTEM USING MICROCONTROLLERS

Ausama H. Ahmed^{1,*}, Ali O. Alsharif², Othman E. Aburas³

¹Department of Electrical & Computer Engineering, Faculty of Engineering, Elmergib University, Alkhums, Libya, ahahmed@elmergib.edu.ly

²Department of Electrical & Computer Engineering, Faculty of Engineering, Elmergib University, Alkhums, Libya, alsharifali1981@gmail.com

³Department of Electrical & Computer Engineering, Faculty of Engineering, Elmergib University, Alkhums, Libya, oeaburas@elmergib.edu.ly

ABSTRACT

Control systems engineering is a very interesting engineering disciplines. It requires proper tools to teach it effectively, one of the tools is experimental setups. They are needed to understand and apply different control strategies and to connect theory with application. However, such setups might not be easy to reach for a number of reasons such as their relatively high prices. Hardware in the Loop (HIL) is a way to replace an experimental setup to be studied with an equivalent hardware, such as a PC or a microcontroller. In this paper a microcontroller is used to replace two different physical systems. In addition to simulating the systems, another microcontroller is also used to replace the control loop, i.e. the summing node and the controller. The simulation of the systems and the whole control loop on the microcontroller was carried out using difference equation method and Ordinary Differential Equations method whenever it is applicable. The interface between the two microcontrollers was established using serial communication protocol and ESP NOW. The performance of the control system is compared with MATLAB and showed very high accuracy. The simulation was run in real time, with a sample time of 10ms.

Keywords:

Arduino.

ESP32.

ODE difference equation.

Dynamic systems simulation.

Hardware in the Loop.

*Corresponding Author Email: ahahmed@elmergib.edu.ly

1 INTRODUCTION

Modelling and simulation are useful tools to study, and learn about dynamic systems as well as testing different controllers on them. Beside the standard well known experimental setups used to teach and test different control strategies, what is called Hardware in the Loop (HIL) can also be used [1]. It is a technique for testing a controller on a virtual environment representing the real system. HIL is desirable as the hardware might be expensive to build or run into error, or

changes in the hardware are difficult so it can be modelled and implemented on the HIL system [2]. Such a system offers real time testing of the system to be mimicked.

HIL has been implemented on personal computers PC for validating software [3], and electric drive systems [4] for automotive industry, and also for robot manipulators [5]. One of the most widely used Digital Signal Processors are those produced by dSPACETM, they are used to simulate a variety of systems such as power electronics [6] in which the sampling time was $10\mu s$.

A platform called RTSDS is proposed to be used for embedded control algorithms. It consists of Intel DN2800MT Mini-ITX platform as a main control architecture with Arduino Mega and Arduino Uno as data acquisition modules and a PSOC CY8CKIT-001 kit as a signal generation module. It was used to simulate a water tank system and a vehicle control system [7].

A HIL for a chemical system simulation based on Ordinary Differential Equations ODEs solved using Runge-kutta method on 8 bits and 32 bits microcontrollers [8]. The authors studied the speed and error truncation performance for the microcontrollers. The serial communication was used to output the data which resulted in slowing the performance by 16 times and 42 times than without the serial communication. System dynamics simulation on microcontrollers using ODEs can be carried out using Euler method. In which the authors emulated a spring mass damper system, and an RLC circuit. The work was carried out on a 16-bit PIC microcontroller running at 16MHz clock [9].

The mathematical model of a stirred tank heater is simulated on MATLAB/Simulink and an Arduino Uno is used as a controller. Sliding mode controller is implemented and used to control the simulated system [10]. An Arduino Uno board is used as an interface between a host PC and another target PC programmed using MATLAB/Simulink/xPC Target. The target PC is used to simulate an active suspension system [11]. A mobile robot with wheels and an inverted pendulum is simulated using MATLAB/Simulink environment and the controller was implemented on an Arduino Yun board [12]. The communication between them was established using TCP/IP protocol. Hardware in the loop to simulate an autonomous under water vehicle using beagle bone and programmed

using MATLAB/Simulink [13]. Two microcontroller boards, namely STM32F446RE Nucleo, were used to simulate a control system of an inverted pendulum. Where, one is used as a controller and the other is used as the system, in which, everything was coded on MATLAB/Simulink and then deployed on the boards [14].

In this paper, two microcontrollers are used to simulate closed loop control systems. Where, two well-known controllers, namely the Lead and the PID, are used to control two well-known models. The first model is a single input single output system which is a DC-Motor position system, and the second is a single input multi output system which is the inverted pendulum.

This paper is organized as follows; the microcontrollers used for the simulation are presented in section 2, while the methods used for simulation are presented in section 3; the two models with their simulation on MATLAB and microcontrollers are presented in section 4, while the two controllers are presented in section 5; finally the results and the discussion and the conclusion are in sections 5 and 6 respectively.

2 SIMULATION ON MICROCONTROLLERS

In control systems, it is desired for a plant to follow a reference input, usually a step function, with specific characteristics, such as rise time, and settling time. The error between the reference input and the output of the plant is fed to a controller which is designed to shape the response of the system to get the desired response of the plant. This is called a control loop, see Figure 1.

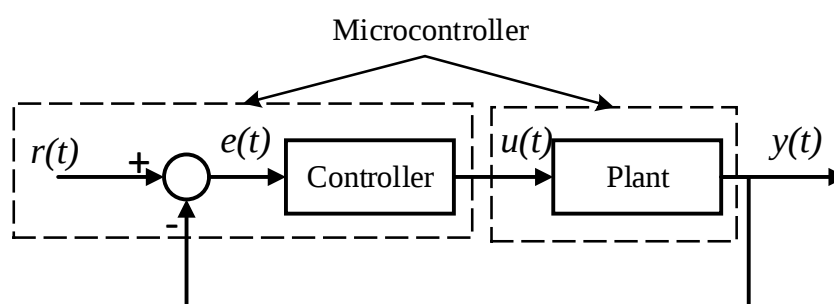


Figure 1. Control loop of a typical control system

Because of the cost of the laboratory experimental setups, it is proposed to use a widely used micro-controller to replace the plant, see Figure 1. It is also

proposed to use another micro-controller to replace the control loop, which includes the reference input, the summing node, and the controller. Designing controllers and implementing them on micro-controllers is a good practice for students and to get an insight on how to use them, as well as for researchers to build, and test their work.

Connecting the two controllers, one for the plant and one for the controller, can be carried out in different methods, depending on the micro-controller. For example two Arduino UNO boards can be connected using serial communication protocol, but cannot be connected using analogue signals as the Arduino UNO does not output a real analogue signal rather than Pulse Width Modulation PWM [15]. However, other boards, such as Arduino Due boards, can be connected using analogue signals as they have true analogue outputs. In this work, two Arduino UNO boards are connected together to simulate the plant and the controller through serial communication. The hardware serial pins, i.e. pins 0 and 1, are not used to connect the two boards together as they are used to send the output signal for visualization. Rather, a soft serial library is used to connect the two Arduino UNO boards at a baud rate of 56700bps using pins 9 and 10, see Figure 2.

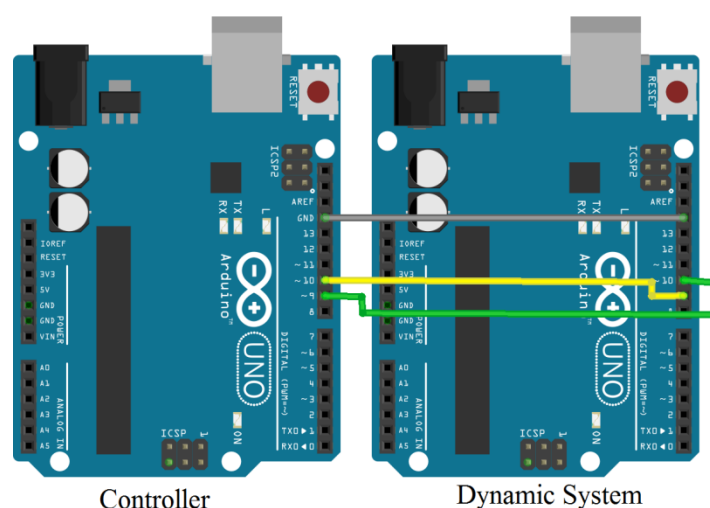


Figure 2. Two Arduino UNO boards connected using serial communication on pins 9 and 10.

Also, two ESP32 microcontrollers are connected together using ESP-NOW protocol. It is a protocol by Espressif used to easily exchange data not larger than 250Bytes at a time between two or more ESP32 boards [16]. The

communication protocol is Wi-Fi based, and a board needs only the mac address of the board to be sent the information to. The ESP32 boards have more than one hardware serial port, and hence can be connected together without using software serial library.

In order to simulate a system, whether it is a plant or a controller, the following steps are proposed:

1. Enter the initial conditions of the system
2. Get the input value
3. Calculate the next value of the simulation
4. Send the output value
5. Wait until a specific time is passed
6. Go to step 2

Note that the simulation must be carried in an accurately timed manner, real time, as making it faster will make the system faster than it should be and vice versa.

3 DYNAMIC SYSTEMS SIMULATION

Dynamic systems can be modelled using ordinary differential equations or transfer functions. The differential equations are simulated on the micro-controllers' using Runge-Kutta method [17], while the transfer function is simulated by discretizing it.

3.1 Differential Equations

Solving systems represented by differential equations, is equivalent to simulating them. They are assumed to be represented in state-space form, where the state and output equations are given as follows:

$$\dot{x} = Ax + Bu \quad (1)$$

$$y = Cx + Du \quad (2)$$

Where, A is the state matrix, B is the input matrix, C is the output matrix, and D is the feed forward matrix.

This system of differential equations is solved (simulated) numerically using the fourth order Runge-Kutta method, sometimes referred to as RK4 [18], [19]. For a differential equation given by:

$$\frac{dx}{dt} = f(t, x), \quad x(t_0) = x_0 \quad (3)$$

Where f is a function of the variables x , t_0 the initial time, and x_0 is the initial condition.

Given the initial condition (value), of the equation and the variable(s) x at the initial time, the solution iteratively approximates the next value of the solution at time $t + h$, where h is the step size of the simulation, by following the following procedure:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (4)$$

$$t_{n+1} = t_n + h \quad (5)$$

$$k_1 = f(t_n, x_n) \quad (6)$$

$$k_2 = f\left(t_n + h, x_n + \frac{k_1}{2}\right) \quad (7)$$

$$k_3 = f\left(t_n + h, x_n + \frac{k_2}{2}\right) \quad (8)$$

$$k_4 = f(t_n + h, x_n + k_3) \quad (9)$$

Where, x_{n+1} means $x(t + h)$, and k_1 is the slope at the beginning, k_2 and k_3 are the slopes at mid-point, and k_4 is the slope at the end of the interval.

The above procedure is repeated until the simulation time is satisfied. Note that this is applied to the case when there is only one differential equation. For a system of higher number of equations, four slopes for each equation are needed to be calculated. For a system of n equations, the procedure is as follows:

$$x_{i,n+1} = x_{i,n} + \frac{h}{6}(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}) \quad (10)$$

$$t_{n+1} = t_n + h \quad (11)$$

$$k_{1i} = f(t_n, x_{i,n}) \quad (12)$$

$$k_{2i} = f\left(t_n + h, x_{i,n} + \frac{k_{1i}}{2}\right) \quad (13)$$

$$k_{3i} = f\left(t_n + h, x_{i,n} + \frac{k_{2i}}{2}\right) \quad (14)$$

$$k_{4i} = f(t_n + h, x_{i,n} + k_{3i}) \quad (15)$$

Where, $i = 1, 2, \dots, n$

3.2 Transfer Function

This form of representation method, the relationship between the input and the output is discretized at a sample time of T_s can be given in the following form:

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_n z^{-n}} \quad (16)$$

Where, z^{-1} represents one time delay unit, $a_1, a_2, \dots, a_n$ and $b_0, b_1, \dots, b_n$ are the coefficients of the denominator and the numerator respectively.

The discretized transfer function is then transformed into difference equation in the following form

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \dots - a_n y(k-n) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2) + \dots + b_n u(k-n) \quad (17)$$

Where, $y(k)$ and $u(k)$ are the output of the system and the input of the system respectively at sample k .

Such system can be simulated by simply plugging in the values of the previous values of the input u and the output y in the difference equation (17) [20]. Note that to ensure an accurate simulation, the result of the equation should be used every time interval of T_s . This method and the previous one are used to implement the mathematical model of a single input single output model, namely DC-motor position system, and a single input multi output model, namely inverted pendulum.

4 PLANT MODEL

4.1 DC Motor

DC motor model is considered one of the most widely used example in control books to learn modelling and control [20-22]. It is used to demonstrate modelling of systems with both electrical and mechanical parts. It is also used as a part of case studies in control.

A. Modelling

The system considered is shown in Figure 3.

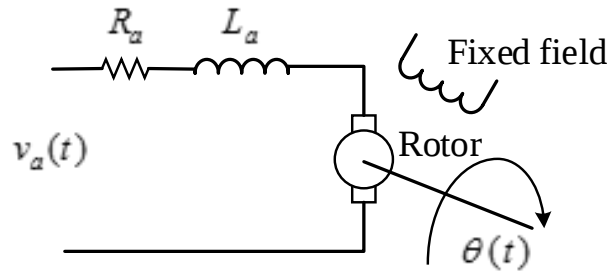


Figure 3. DC motor model

The mathematical model of the DC motor, assuming the output is position, can be given as follows:

$$\frac{\theta(s)}{V_a(s)} = \frac{K_m}{s[(R_a + L_a s)(J s + b) + K_b K_m]} \quad (18)$$

Where, K_m is the motor's constant, K_b is the back electromotive force constant, R_a and L_a are the resistance and the inductance of the armature respectively, J is the inertia, and b is the friction.

Let, $K_m = 0.1$, $K_b = 0.1$, $R_a = 2 \Omega$, $L_a = 0.5H$, $J = 0.02 \text{Kg.m}^2/\text{s}^2$, and $b = 0.2 \text{Nms}$ [23]. The transfer function is evaluated to be

$$\frac{\theta(s)}{V_a(s)} = \frac{10}{s^3 + 12s^2 + 41s} \quad (19)$$

B. Simulation

To simulate the system given by equation (19), it is transformed in state-space form which can be represented as following:

$$\dot{x} = \begin{bmatrix} -14 & -41 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u \quad (20)$$

$$y = [10 \quad 0 \quad 0] x \quad (21)$$

The differential equations, given by (20), are evaluated first using Runge-Kutta method, when implementing this system on a micro-controller; then the output of the system is the first state times 10, derived from the C matrix in equation (21).

The difference equation of the system after discretizing it using Tustin method [20] with a sample time of 0.01sec is:

$$y(k) = 2.865 y(k-1) - 2.735 y(k-2) + 0.869 y(k-3) + 1.167 \times 10^{-6} u(k) + 3.501 \times 10^{-6} u(k-1) + 3.501 \times 10^{-6} u(k-2) + 1.167 \times 10^{-6} u(k-3) \quad (22)$$

Note that, in order to get an accurate enough response, the digits considered, excluding the exponent part, must be at least 6 fraction digits after the decimal point.

The response of the model to the standard step input, carried out using the step command in MATLAB environment, is shown in Figure 4, in solid line. The implementation on micro-controller is carried out, Runge-Kutta and difference equation, and the output is shown in Figure 4, in dotted and dashed lines respectively. Note that in case of Runge-Kutta, the order of evaluation of the constants is K_{11}, K_{12}, K_{13} were computed first, then K_{21}, K_{22}, K_{23} and finally K_{31}, K_{32}, K_{33} . Also the output of the system was the first state times 10, from the C matrix. The error between the Runge-Kutta implementation on the microcontroller and the MATLAB's step command is around 0.1% of MATLAB's response; while the error in case of the difference equation on the microcontroller is around 0.8%, the error might be due to the precision of the microcontroller.

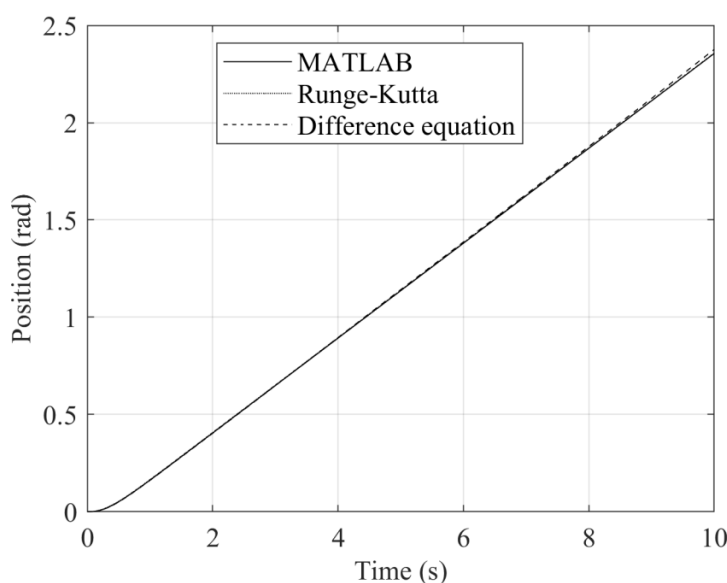


Figure 4. Step response of the DC motor using MATLAB on PC, and Runge-Kutta and difference equation on micro-controller

4.2 Inverted Pendulum Model

In addition to the DC motor, the inverted pendulum system is also one of the widely used models in learning and applying control strategies. This system has a pendulum on a moving base, see Figure 5. It is an example of single input, multi-output system, where the input is the force applied to the base, and the outputs are the inclination angle and the distance travelled by the wheels [24].

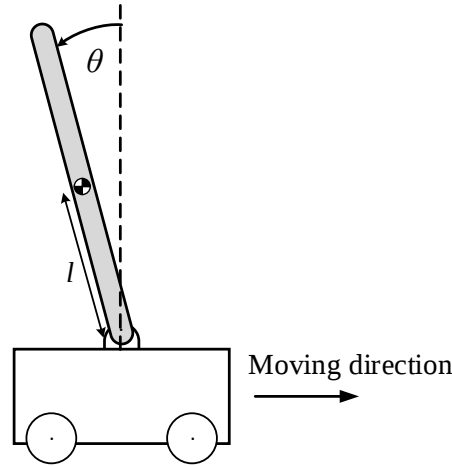


Figure 5. Inverted pendulum model

A. Modelling

The system can be represented by the following state-space mathematical model, assuming there is no friction, and the input is the force applied to the base [24]:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-b(I+ml^2)}{\beta} & \frac{m^2gl^2}{\beta} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-m l b}{\beta} & \frac{m g l(M+m)}{\beta} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{\beta} \\ 0 \\ \frac{m l}{\beta} \end{bmatrix} u \quad (23)$$

Where, $\beta = I(M + m) + Mml^2$

The value of the terms in equation (23) is given in Table 1.

Substituting the values in the table into equation (23), yields the following

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.181818 & 2.672727 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.454545 & 31.181818 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1.818181 \\ 0 \\ 4.545454 \end{bmatrix} u \quad (24)$$

The output is desired to be the angle, i.e. the third state, and the distance, i.e. the first state, then the output equation can be written as

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} \quad (25)$$

Table 1. Symbols, value and units of the terms in equation (23).

Symbol	Unit	Value	Description
m	Kg	0.5	Mass of the pendulum.
l	m	0.3	Distance between the pendulum's center of gravity and the base.
M	Kg	0.05	Mass of the cart
I	Kgm^2	0.006	Moment of inertia of the pendulum
b	N/m/s	0.1	Coefficient of friction of the cart
g	m/s^2	9.81	Gravity.

B. Simulation

The response is evaluated on the microcontroller using Runge-Kutta method, once and the difference equation method. The system is represented by 4 differential equations, i.e. equation (24), which requires the evaluation of 16 equations for the slopes. It is noted that the fractional parts of the numbers in equation (24) are all repeating and can be represented as fractions. That can be useful when implementing on microcontrollers as they can be more precise than using float numbers. Equation (24) can be rewritten using only fractions as follows:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -2/11 & 294/110 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -5/11 & 343/11 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 20/11 \\ 0 \\ 50/11 \end{bmatrix} u \quad (26)$$

Since the system has two outputs, then two discretized transfer functions are needed to represent the system, one for the inclination angle and one for the distance. The discretization uses Tustin method [20] with a sample time of

0.01sec. The difference equations of the system, one for each transfer function, can be given by:

$$y_1(k) = 4.001 y_1(k-1) - 6.001 y_1(k-2) + 3.998 y_1(k-3) - 0.9982 y_1(k-4) + 1.136 \times 10^{-4} u(k) - 2.272 \times 10^{-4} u(k-2) + 1.136 \times 10^{-4} u(k-4) \quad (27)$$

$$y_2(k) = 4.001 y_2(k-1) - 6.001 y_2(k-2) + 3.998 y_2(k-3) - 0.9982 y_2(k-4) + 4.542 \times 10^{-5} u(k) - 1.113 \times 10^{-7} u(k-1) - 9.106 \times 10^{-5} u(k-2) - 1.113 \times 10^{-7} u(k-3) + 4.542 \times 10^{-5} u(k-4) \quad (28)$$

The response of the system, given by equations (26) and (25), is simulated using MATLAB's step command is shown in Figure 6, in solid line. The response of the system implemented on the Arduino UNO and ESP32 micro-controller, in dotted and dot-dashed lines respectively, using Runge-Kutta and difference equation using equations (27) and (28), is shown in Figure 6. The error of using the Runge-Kutta implementation on the microcontroller is around $1 \times 10^{-4}\%$, and 23.0581% for using difference equation for the angle output. And the error of the distance using Runge-Kutta is around $1 \times 10^{-4}\%$, and 12.185% using difference equation. The error in case of using the difference equation is high, this is due to the resolution of the double and float numbers on the Arduino Uno boards. To decrease this error, a better microcontroller, ESP32, in terms of computation power is used. The error is improved by around 10 folds, to be 2.42%, for the inclination angle and 5 folds, to be 2.43%, for the distance output.

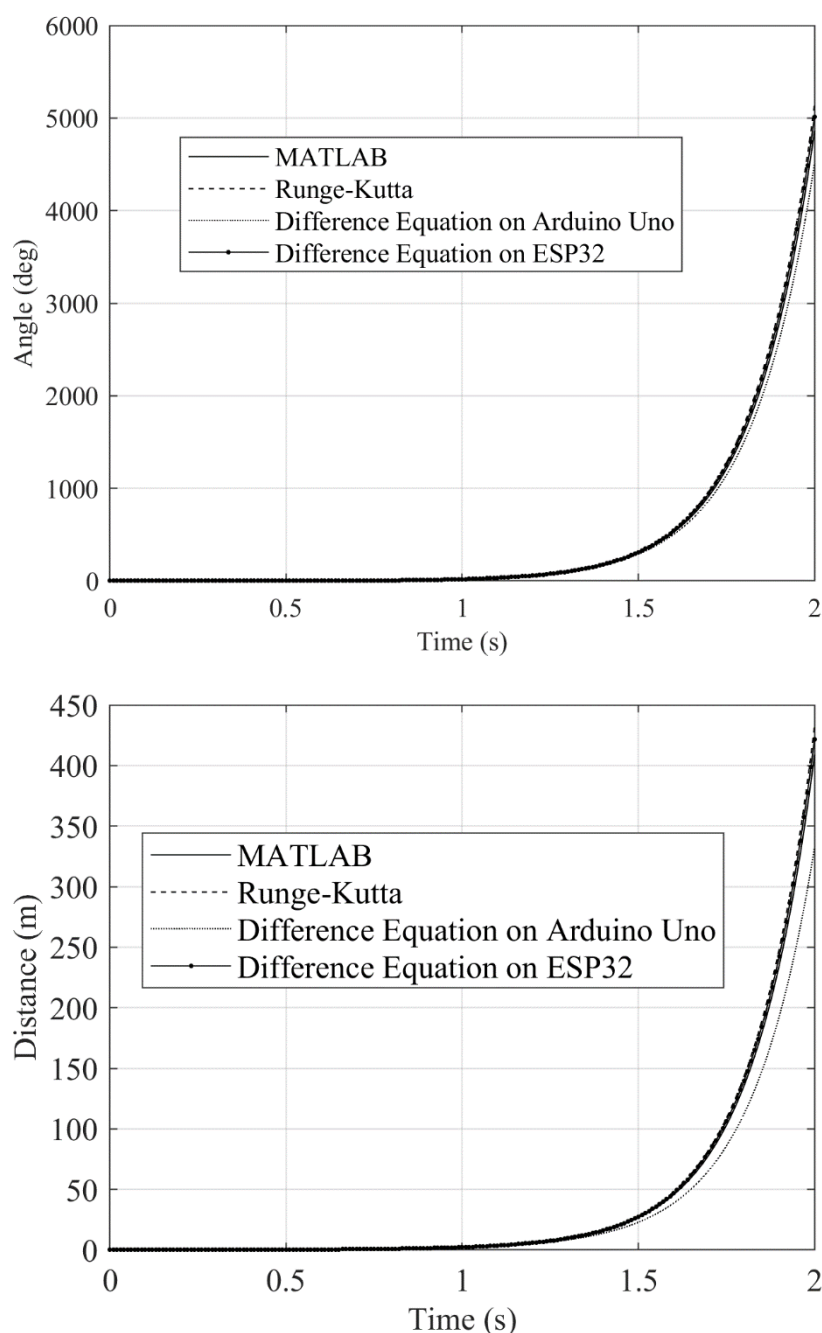


Figure 6. Step response of the inverted pendulum using MATLAB, and Runge-Kutta and difference equation on two microcontrollers, the inclination angle is above and the distance is below

5 CONTROLLERS

Similar to the dynamic system's simulation, two controllers are implemented using the two methods previously mentioned, i.e. Runge-Kutta and the difference equation. Note that the Rung-Kutta method is not applicable to

transfer functions with numerator order larger than denominator order such as the well-known PID controller. The implementation is carried out for two widely studied controllers, namely the Lead and the PID controllers [21, 22, 25].

5.1 Lead controller

The phase Lead controller has two zeros and two poles, it can be given in the following form:

$$G_L(s) = \frac{(s+z)}{(s+p)} \quad (29)$$

Where, z and p represent the zero and pole locations of the lead controller respectively.

To illustrate the accuracy of implementing the lead controller on microcontrollers, this controller is designed to control the DC-motor given by equation (19). The response is desired to have an overshoot of 6.58%, and a settling time of 1s. The designed lead controller's transfer function is

$$G_L(s) = \frac{58.287(s+3.564)}{(s+14.3)} \quad (30)$$

The equation can be simulated using Runge-Kutta method by solving the state space form given as following

$$\dot{x} = -14.3x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \quad (31)$$

$$y = -625.7692x + 58.287u \quad (32)$$

Note that the symbols u and y here refer to the input and the output of the controller.

The system is also simulated by transforming equation (30) into difference equation, which can be presented in the following form

$$y(k) = 0.8665y(k-1) + 55.3669 u(k) - 53.4282 u(k-1) \quad (33)$$

The DC-motor's equations are implemented on one microcontroller and the controller is implemented on another as previously explained in section 2. The response of the whole system, plant with the controller carried on MATLAB in solid line, and on the microcontrollers, using Runge-Kutta, in dashed line, and difference equation using dashed line are shown in Figure 7.

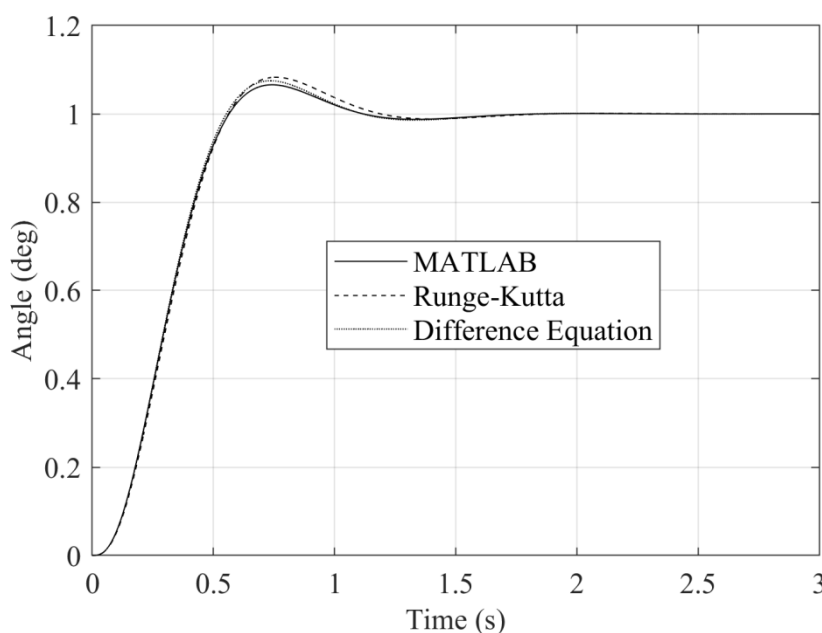


Figure 7. Step response of the DC motor with lead controller using MATLAB, and Runge-Kutta and difference equation on micro-controller

The figure showed a very close response between the MATLAB and the microcontroller's simulations. The percentage overshoot using Runge-Kutta and the difference equation methods on the microcontroller are 8.3% and 7.5% respectively compared to 6.58% on MATLAB. While the settling time using the Runge-Kutta and the difference equation methods are 1.07s and 1s respectively compared to 1s on MATLAB.

5.2 PID controller

The PID controller sometimes called the three terms controller, it is considered to be one of the most widely used controllers in industry. It comprises of three parts, namely a proportional gain, an integrator, and a differentiator. It can be given using the following equation:

$$G_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \quad (34)$$

Where, K_P , K_I , and K_D are gains.

Applying Runge-Kutta method to this controller is not possible, because the order of the numerator of the transfer function of the controller is higher than the denominator, and cannot transform it to ODEs. Only the difference equation is used to simulate the system on the microcontroller.

A PID controller was designed, so that the system has an overshoot of 10% and a settling time of 1 sec. The gains of the PID controller are $K_P = 74.5$, $K_I = 400$, $K_D = 17$. Plugging in the gains in equation (34) and discretizing the transfer function, yields the following difference equation:

$$y(k) = y(k-2) + 3476u(k) - 6796u(k-1) + 3327u(k-2) \quad (35)$$

Note that the symbols u and y here refer to the input and the output of the controller.

The response of the whole system, controller with the inverted pendulum, simulated on MATLAB and the microcontroller are shown in Figure 8.

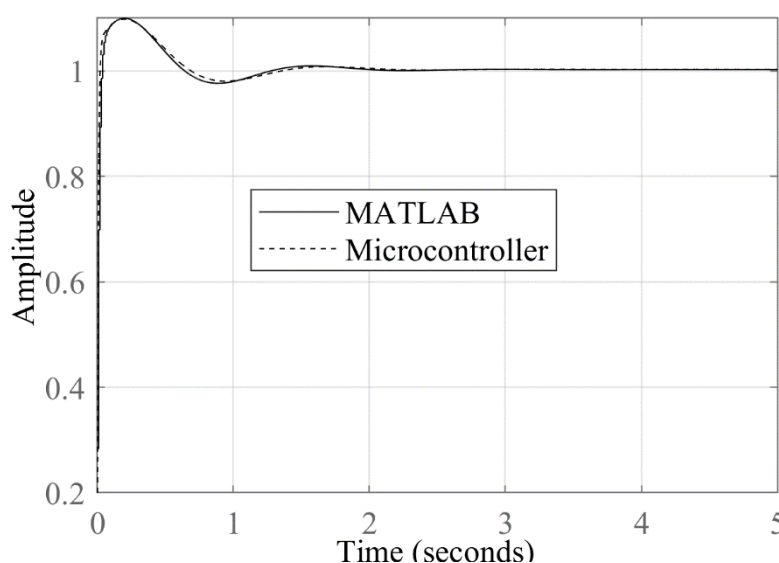


Figure 8. Step response of the closed loop simulated on MATLAB and microcontroller.

6 RESULTS AND DISCUSSION

The results obtained using microcontrollers are very close to that obtained using MATLAB. The Runge-Kutta method of simulating systems on microcontrollers

showed a better results than that obtained using difference equations method. That might be because the Runge-Kutta method has numbers of close logarithmical scale, in the order of 0 and 2 in case of the DC-motor model. On the contrary, the difference equations for the simulated systems with a 10ms sampling time have logarithmically diverse numbers, in the order of 0 to -6 (3.9 and 1×10^{-6}). The difference causes a loss in precision as the microcontrollers have only 6 digits after the decimal point, unlike the MATLAB environment which has 15 digits after the decimal point precision. That was confirmed when the same equations are implemented on an 8-bit microcontroller, (Arduino UNO), and then on a 32-bit microcontroller (ESP32). The results were up to 10 folds better in case of the inverted pendulum case.

The speed of the loop, from reading a value till outputting the response, is tested for a sample time of 10ms. However, by removing the delay restrictions, the maximum sample time obtained was approximately 3.5ms on the Arduino UNO board for the Runge-Kutta method. The fastest sample time that obtained using difference equation was 2.4ms. Note that no techniques to optimize the speed of the loop, such as transforming the division into multiplication, were used. The fastest simulation time obtained was 1.06ms using the difference equations of the DC-Motor on the ESP32 board.

7 CONCLUSIONS

In this work, microcontrollers are used to simulate dynamic systems. In which it can be used as a testing platform instead of experimental setups. Two typical case studies and two controllers were simulated on microcontrollers. The connection between the plants and the controllers were once a serial communication protocol and another time a Wi-Fi based communication protocol. The accuracy of the simulation is highly accurate when compared with MATLAB. The work can serve an excellent setup for researchers and students to apply and learn digital control systems principles, such as the effect of sampling time and aliasing on systems, and implement different controllers on microcontrollers.

The implementation presented is for linear systems, however, nonlinear properties such as saturation, and backlash can be added in programming to

make it behave closer to a real setup. Note that saturation, relatively high values, is already present as the float numbers used on the Arduino UNO is $\pm 3.4028235 \times 10^{38}$. The analogue inputs and outputs of the ESP32 can be used to connect the two systems instead of the Wi-Fi connection. Also other communication protocols such as I2C and SPI can also be used for connecting microcontroller boards. In order to make the system more complete with the possibility of replacing the microcontroller with analogue circuits, such as operational amplifiers and filters, digital to analogue converters must be added.

8 REFERENCES

- [1] "Hardware-in-the-Loop (HIL) Simulation - MATLAB & Simulink." <https://www.mathworks.com/discovery/hardware-in-the-loop-hil.html> (accessed Oct. 15, 2021).
- [2] S. Demers, P. Gopalakrishnan, and L. Kant, "A Generic Solution to Software-in-the-Loop," in *MILCOM 2007 - IEEE Military Communications Conference*, Oct. 2007, pp. 1–6. doi: 10.1109/MILCOM.2007.4455268.
- [3] S. Jeong, Y. Kwak, and W. J. Lee, "Software-in-the-Loop simulation for early-stage testing of AUTOSAR software component," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2016, pp. 59–63. doi: 10.1109/ICUFN.2016.7536980.
- [4] G. M. Casolino, M. AlizadehTir, A. Andreoli, M. Albanesi, and F. Marignetti, "Software-in-the-loop simulation of a test system for automotive electric drives," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct. 2016, pp. 1882–1887. doi: 10.1109/IECON.2016.7794145.
- [5] M. B. Ayed, L. Zouari, and M. Abid, "Software In the Loop Simulation for Robot Manipulators," *Eng. Technol. Appl. Sci. Res.*, vol. 7, no. 5, Art. no. 5, Oct. 2017, doi: 10.48084/etasr.1285.
- [6] R. Crosbie, J. Zenor, R. Bednar, D. Word, N. Hingorani, and T. Ericson, "High-Speed, scalable, real-time simulation using DSP arrays," in *18th Workshop on Parallel and Distributed Simulation, 2004. PADS 2004.*, May 2004, pp. 52–59. doi: 10.1109/PADS.2004.1301285.
- [7] I. D. T. de Souza, S. N. Silva, R. M. Teles, and M. A. C. Fernandes, "Platform for Real-Time Simulation of Dynamic Systems and Hardware-in-the-Loop for Control Algorithms," *Sensors*, vol. 14, no. 10, pp. 19176–19199, Oct. 2014, doi: 10.3390/s141019176.
- [8] R. D. P. Filho, V. B. Furlong, and J. a. V. Costa, "ARDUINO OPEN HARDWARE TEST THROUGH CHEMICAL SYSTEM SIMULATION UNDER LUMPED MODEL DYNAMICS," in *Blucher Chemical Engineering Proceedings*, 2015, vol. 1, no. 2, pp. 11406–11413. Accessed: Oct. 09, 2021. [Online]. Available: <https://www.proceedings.blucher.com.br/article-details/arduino-open-hardware-test-through-chemical-system-simulation-under-lumped-model-dynamics-18067>
- [9] S. Nuratch, "Study and implementation of microcontroller-based system dynamics emulator using numerical method and reduction of order technique," in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Jun. 2016, pp. 1–6. doi: 10.1109/ECTICon.2016.7561446.
- [10] O. Camacho, R. Díaz, and C. Obando, "Hardware in the Loop Simulation for Sliding Mode Control Schemes for Deadtime Systems," in *2019 International Conference on Information*

- Systems and Computer Science (INCISCOS)*, Nov. 2019, pp. 183–188. doi: 10.1109/INCISCOS49368.2019.00037.
- [11] “[PDF] Development of a Microcontroller-Based Control System with a Hardware-in-the- Loop (HIL) Method for Control Education using MATLAB/Simulink/xPC Target* | Semantic Scholar.” <https://www.semanticscholar.org/paper/Development-of-a-Microcontroller-Based-Control-with-Martono-Salami/d3893ca5bd6d7f6496c7646abf143ff79813594b> (accessed Oct. 15, 2021).
- [12] I. Tejado, J. Serrano, E. Pérez, D. Torres, and B. M. Vinagre, “Low-cost Hardware-in-the-loop Testbed of a Mobile Robot to Support Learning in Automatic Control and Robotics**This work has been supported by the Spanish Ministry of Economy and Competitiveness under the project DPI2012-37062-C02-02 and the Junta de Extremadura under the Ayuda a Grupos with reference GR15178.” *IFAC-Pap.*, vol. 49, no. 6, pp. 242–247, Jan. 2016, doi: 10.1016/j.ifacol.2016.07.184.
- [13] T. W. Oktaviana Putri, M. F. Ginting, B. R. Trilaksono, E. M. Idris Hidayat, and M. F. Sagala, “Hardware in the loop simulation development of guidance system for autonomous underwater glider,” in *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, Nov. 2017, pp. 1–4. doi: 10.1109/ICEEI.2017.8312388.
- [14] D. S. Bowden and J. A. Mynderse, “From Industry to the Classroom: A Low-Cost Hardware-In-Loop Simulator for Classic Controls Experiments,” presented at the 2019 ASEE Annual Conference & Exposition, Jun. 2019. Accessed: Oct. 16, 2021. [Online]. Available: <https://peer.asee.org/from-industry-to-the-classroom-a-low-cost-hardware-in-loop-simulator-for-classic-controls-experiments>
- [15] “UNO R3 | Arduino Documentation | Arduino Documentation.” <https://docs.arduino.cc/hardware/uno-rev3> (accessed Oct. 18, 2021).
- [16] “ESP-NOW - ESP32 - — ESP-IDF Programming Guide latest documentation.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html (accessed Oct. 16, 2021).
- [17] Daniel, “Real-time simulations on an Arduino (double pendulum),” *Homemade multibody dynamics*, Jan. 05, 2017. <https://hmbd.wordpress.com/2017/01/05/real-time-simulations-on-an-arduino-double-pendulum/> (accessed Oct. 09, 2021).
- [18] W. H. Press, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd edition. Cambridge, UK ; New York: Cambridge University Press, 2007.
- [19] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*, 1st edition. Cambridge ; New York: Cambridge University Press, 2003.
- [20] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 8th edition. Ny, NY: Pearson, 2018.
- [21] R. Dorf and R. Bishop, *Modern Control Systems*, 13th edition. Hoboken: Pearson, 2016.
- [22] K. Ogata, *Modern Control Engineering*, 5th edition. Boston: Pearson, 2009.
- [23] “DC Motor Control - MATLAB & Simulink Example.” <https://www.mathworks.com/help/control/ug/dc-motor-control.html> (accessed Sep. 12, 2021).
- [24] “Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling.” <https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum§ion=SystemModeling> (accessed Oct. 16, 2021).
- [25] K. J. Åström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton: Princeton University Press, 2008.

نظام تحكم بأسلوب عتاد محاكاة الواقع قليل التكلفة باستخدام متحكمات دقيقة

أسامة الهادي أحمد^{1*}، علي عمر الشريف²، عثمان أحمد أبوراس³

¹قسم الهندسة الكهربائية والحاسوب، كلية الهندسة، جامعة المرقب، الخمس، ليبيا، ahahmed@elmergib.edu.ly

²قسم الهندسة الكهربائية والحاسوب، كلية الهندسة، جامعة المرقب، الخمس، ليبيا، aosalsharif@elmergib.edu.ly

³قسم الهندسة الكهربائية والحاسوب، كلية الهندسة، جامعة المرقب، الخمس، ليبيا، oeaburas@elmergib.edu.ly

الملخص

أنظمة التحكم الآلي هي تخصصات هندسية غاية في الأهمية. تعلم هذه التخصصات والوصول إلى خبرة عالية يتطلب أدوات مناسبة، إحدى هذه الأدوات هي تجهيزات التجارب المعملية. تلك التجهيزات مطلوبة لفهم استراتيجيات التحكم المختلفة وتطبيقاتها وربط النظريات العلمية بالتطبيق العملي. ولكن في بعض الأحيان قد لا يكون من السهل الحصول على مثل هذه التجهيزات لعدة أسباب من هذه الأسباب التكلفة العالية و عدم توفرها في الأسواق. عتاد محاكاة الواقع (HIL) هي طريقة لاستبدال التجهيز لتجربة مطلوب دراستها بعتاد مكافئ مثل جهاز كمبيوتر أو متحكم دقيق. في هذا البحث تم استخدام متحكم دقيق ليحل محل نظامين فيزيائيين مختلفين. بالإضافة إلى المحاكاة، تم أيضاً استخدام متحكم آخر لاستبدال حلقة التحكم، والتي تشمل عقدة الجمع ووحدة التحكم. تم تنفيذ المحاكاة للأنظمة وحلقة التحكم الكاملة على المتحكمات الدقيقة باستخدام طريقة معادلة الفرق وطريقة المعادلات التفاضلية كلما كان ذلك ممكناً. حيث تم الربط بين المتحكمات الدقيقة باستخدام بروتوكول الاتصال التسلسلي و ESP NOW. وتمت مقارنة أداء نظام التحكم مع MATLAB وأظهر دقة عالية جداً بعد تشغيل المحاكاة بزم من حقيقي بوقت عينة 10 مللي ثانية.

الكلمات الدالة:

Arduino

ESP32

معادلات تفاضلية.

معادلة الفرق.

محاكاة الأنظمة الديناميكية.

عتاد محاكاة الواقع.

*البريد الإلكتروني للباحث المراسل: ahahmed@elmergib.edu.ly