Master-Slave Synchronization of Robotic Arm using PID Controller

Yin Hnin Thet Htun[1], May Su Hlaing[2], Tin Tin Hla[3]
[1,2,3]Department of Electronic Engineering, Mandalay Technological University, Patheingyi Township, Mandalay Region, Republic of the Union of Myanmar

Article Info

Article history:

Received Sep 20, 2022
Revised Dec 23, 2022
Accepted Jan 13, 2023

Keyword:

Master-slave synchronization
Angular position control
DC motor
PID controller
Arduino Uno

ABSTRACT

This paper analyzes the position control of a master-slave synchronization robotic arm driven by a D.C. motor using a PID (Proportional, Integral, and Derivative) controller with software and hardware design. This controller works to achieve the exact desired position simultaneously for the master and slave robot arm with minimal defects. The transfer function of the D.C. motor for the robotic arms used in this research is calculated with black box modelling. MATLAB Simulink block is used to test the software result. The MATLAB built-in auto-tuning method obtains Kp, Ki and Kd gain. These gains are adjusted with manual tuning to get precise angular positions for two robotic arms. This research uses Arduino Uno to act as a controller in the experiment. First, the position control of one robotic arm is tested with the same PID gain in MATLAB Simulink at different input degrees. Then, the hardware experiment of position control in one robotic arm is operated with only one PID gain at various reference degrees. Finally, the I2C communication protocol connects the master and slave robot arms. The main work that the PID controller hardware experiment with controls different level angular positions of two robotic arms.

Corresponding Author:

Yin Hnin Thet Htun,
Department of Electronic Engineering,
Mandalay Technological University,
Patheingyi Township, Mandalay Region, Republic of Union of Myanmar.
Email: yinhninthettun@gmail.com

1. INTRODUCTION

Robotic technology is developing remarkably and adapting to broad fields such as welfare, medicine, agriculture, etc. Robot arms are programmed to perform specified tasks like drilling, welding, soldering, etc. The synchronization of the master-slave robotic system reduces the number of employees in the industrial site and becomes safety in the working area. Several methods of the control algorithm are used to control the position of the master-slave robotic arm driven by a D.C. motor. This research uses the PID (Proportional, Integral, Derivative) control algorithm to control the angular position of the master-slave robotic arms.

To solve the critical problem of estimating the relative camera pose, a multiple master-slave FPGA architecture for a SIFT-based stereo V.O. is proposed to get high efficiency, low power consumption, and low cost [1]. A modular teleoperation software environment based on the robot operating system (ROS) for different master and slave devices is controlled with a hybrid workspace mapping algorithm [2]. A bilateral controller is implemented using a linear motor in the master-slave robotic system for needle indentation and insertion [3]. One research is to control the position of master-slave synchronization of robot manipulators driven by induction motors using the feedback control law based on Lyapunov analysis [4]. For position tracking control of surgical robots, a model-free controller that combines the simplicity of P.D. control and robustness of sliding mode control is used [5]. Synchronization control of multiple robot manipulators is presented using high-order Sliding Mode Control (SMC), which is associated with the cross-coupling synchronization concept in software simulation [6]. One researcher uses force control technology to describe the position and force control of a novel two-finger gripper. A master-slave force control strategy is developed for making the force track the desired force quickly with lower overshoot by using the PID controller to get the desired position for the master finger and by using force control to the slave finger with grey G.M. (1,1) model [7]. A robotic control system based on the synchronization of master-slave remote operation of the mechanical arm controls the slave arm to finish the same action by operating the master automatically. This research adopts the open-source library Xvidcore to code, decode and put forwards the novel motion estimation algorithm ARPS to optimize the library [8]. In the master-slave rehabilitation system, a human upper limb-like robot which can mimic the spasticity of a stroke patient is proposed as the controller device to simulate the severe spasticity of patients [9]. For position synchronization of multiple robot manipulators based on emergent consensus algorithms, the control strategy is to stabilize the position tracking of each manipulator while synchronizing its motion with other manipulator's position motion, so that differential position errors amongst robots converge to zero [10]. In a master-slave control of a virtual reality-based teleoperation construction robot, a fuzzy P.D. controller and conventional PID controller are used to controlling the velocity of hydraulic actuators for C.R. (construction robot) [11]. A newly developed anthropomorphic robot hand, called K.H. Hand type S, has a high potential for dexterous manipulation and displaying hand shape. Its master-slave system uses the bilateral controller for the five-finger robot hand [12]. A new type of master-slave control methodology, which has both unilateral and bilateral ones, is based on the judgment of the contact/non-contact condition of the slave robot followed by the switching of the objectives of unilateral feedback control between position and force [13]. One researcher presented that compared the master/slave control and hybrid control to control the position error and force error [14].

In this paper, the master-slave robot system is that the position angle of the master robot arm is sent to the slave robot system as a control command. The Slave robot arm is moving to follow the master robot arm movement. PID controller controls position control systems. This reduces the number of employees in the industrial site and becomes safety in the working area. The problem statements in this study are (1) when the position data is sent from the master robot arm to the slave robot arm, this work faces position tracking error and time delay, (2) Although the set point reaches the target very fast, there is a slight overshoot and steady-state error where the position control of master-slave synchronization process is tested with the same PID gain, and (3) For position control, with only one PID gain, it is challenging to get precisely every desired degree. To solve the position tracking error and time delay, this experiment is tested again with the tuning of PID gain according to base the background of PID theory to get better performance, remove overshoot, and reduce the steady-state error. Finally, the work is checked with more and more experiment tests. The main targets of this work are (1) to control the required position of the robot arm by using the PID control algorithm and (2) to analyze the position control, accuracy, time delay, stability, and error detection of two robot arms.
Although there are many advanced applications, classical method such as feed forward and cascaded feedback are much preferred to increasing the complexity of the core control algorithm. It is difficult to maintain continuous supports of complex control loops when new support people are not familiar with the control algorithm. Therefore, PID remains a tool of choice for most of the control issues that encounters. The secret to the success of PID based control architectures (such as ratio control, cascade control, and feedforward control), are the low cost/high benefit ratio they provide.

The paper is formulated into the following sections. Section 2 presents the synchronous master-slave control architecture with a robot arm system. Section 3 mentions the implementation of the system design for optimized conditions. Finally, section 4 concludes the research on synchronizing the master-slave robot arm control system with conventional hardware design.

2. SYNCHRONIZATION OF MASTER-SLAVE CONTROL SYSTEM

A master-slave system is raised to reveal the efficacy of acquisitive and working matters. An operator and robots are masters and enslaved persons, correspondingly. The operator controls the robot by employing a control angle, movement position and orientation of the robot arms. A time delay in communications and percent overshoot must be considered in a master-slave scheme

2.1. Master-Slave Control Law

In the master-slave two arms control, the operator (variable resistor) sends the required position data to the Master arm, and this arm is sent to the slave arm the reference position data with I2C (Inter-Integrated Circuit) communication to synchronize the two robot arms (see Figure 1).

The transfer function of DC motor for master and slave arm is calculated with black box modeling (System Identification tool box) using the open loop input/output data of position control system. The obtained transfer function put in the MATLAB Simulink block and tried in simulation with the PID controller. Then, the Kp, Ki, and Kd gain are obtained by adjusting MATLAB auto-tuning techniques. And then, these gains are adjusted again with the Trial-and-Error method to get a better accurate required set point. In hardware testing, the obtained PID gain are readjusted by manual tuning to get more precise position.

Before the master-slave synchronization position control is tested, we analyze the position control of one arm. For the master-slave to control the effort, the enslaved person must retort to variations in the position of the master and try to preserve a steady force. The various desired reference angular positions are tested in only one PID gain to operate simultaneously, both the master and the slave arm. If the master is stirring in the direction of the enslaved person, the slave cannot retort fast enough, and a significant overshoot is a consequence. Cause of the wire communication, the desired data reach late the slave arm, so master-slave synchronization has a time delay.

In this system, we emphasized the master-slave control approaches based on the master-slave control laws in Figure.1. We could focus on the control system design of the master-slave scheme for low delay time and less overshoot communication between two arms. That is our main target for implementing the synchronous techniques.

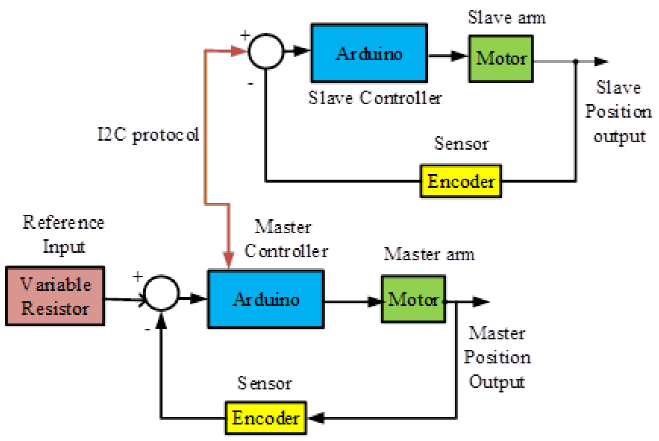


Figure 1. Overall Block Diagram of Master-Slave Synchronization robotic arms

3. IMPLEMENTATION OF SYSTEM

This section could be divided into two main parts. First, to initialize the control system design, the D.C. motor for the robot arm system shall have to be modelled for analysis. Therefore, the first portion only emphasized absolute time position control. And the second portion is the heart of the system, called synchronous master-slave control of robot arms with actual time conditions.

3.1. D.C. Motor Modeling for Robot Arm Control

Two ways to control an excited D.C. motor are (1) Armature Control and (2) Field Control. D.C. motor consists of two parts are a stator and a rotor. The stator consists of the field winding, while the rotor (also called the armature or rotor) consists of the rotor winding. When both rotor and field are excited by D.C. supply, current flows through windings and magnetic flux proportional to the current is generated. When the field's fluctuation interacts with the armature's instability, it results in the motion of the rotor. Rotor control is the most common control technique for D.C. motors. Figure 2 is the circuit diagram of the D.C. motor [15].

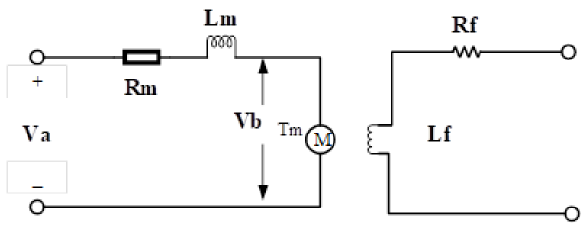


Figure 2. Equivalent Circuit Diagram of DC Motor

As in armature control, field flux is constant; equations governing the operation of the motor become linear, that is

Apply KVL in the armature winding,

$$V_a - R_a I_a - L_a \frac{d\,I_a}{dt} - V_b = 0$$

$$V_a - V_b = R_a I_a + L_a \frac{d\,I_a}{dt}$$

Laplace Transform,

$$V_a(s) - V_b(s) = R_a I_a(s) + L_a .sI_a(s)$$

$$= I_a(s)\,[R_a + L_a.\,s]$$

$$I_a(s) = \frac{V_a(s)-V_b(s)}{R_a + L_a\,.s} \quad (1)$$

Torque, $Tm(s) = (K_1 K_f I_f) I_a(s)$

$$T_m(s) = K_m I_a(s) \quad (2)$$

Angular displacement and angular velocity,

$$T_m = J\,\frac{d^2\,\theta}{dt^2} + b\,\frac{d\theta}{dt} = T_d$$

Taking Laplace transform,

$T_m(s) = J\,.s\,\omega(s) + b\,\omega(s)$

$\omega(s) = T_m(s)\,.\,\frac{1}{J\,s+b}$(3)

(Where $T_d(s) = 0$)

$\theta(s) = \frac{1}{s}\,.\omega(s)$ (3)

Speed and back emf,

$V_b = K_b\,\omega$

Taking LT,

$V_b(s) = K_b\,\omega(s)$ (4)

From eq 1,2,3, and 4

Transfer Function of D.C. motor,

$$G(s) = \frac{\theta(s)}{V_a(s)} = \frac{K_m}{s(R_a + L_a s)(Js+b) + K_b K_m} \quad (5)$$

Where,

K_f, K_m, K_1= motor constant

I_a	= Amature current(A)
I_f	= Field current (A)
$T_m(s)$	= motor torque (Nm)
$T_L(s)$	= load torque (Nm)
$T_d(s)$	= disturbance torque (Nm)
$\omega(s)$	= angular velocity (rads^{-1})
$\theta(s)$	= angular position (degree)
V_a	= field voltage(V)
V_b	= back electromotive-force voltage(V)
R_f	= field resistance(Ω)
L_f	= field inductance (H)
b	= motor friction coefficient (N)
J	= momentof inertia (kgm^2)

The above transfer function of the D.C. motor is obtained using armature control modelling. This research's transfer function model is obtained using MATLAB's S.I. (black box modelling) toolbox. At first, the input-output real-time data are put in the MATLAB code and then the estimation process operates. When the transfer function model output is validated with the I/O data, the estimation data is 84.5% of accuracy. So, this model is appropriate for the system because the accuracy percentage is higher than 80%. The transfer function (tf1) is obtained according to the black-box modelling estimation. The transfer function model of the D.C. motor is

$$\text{TF} = \frac{6.384e^5}{s(s^2+1.06e^5+5.617e^5)} \tag{6}$$

3.2. Real-Time Position Control and their Results

This portion presents the D.C. motor angular position control applying the PID (Proportional, Integral, and Derivative) controller. This control algorithm drives the D.C. motor position at least error at the desired value of the set point. Firstly, the D.C. motor position control based on software simulation uses MATLAB/SIMULINK block for testing the precise angular position with the PID controller. Then, the Kp, Ki, and Kd gain are obtained by adjusting MATLAB auto-tuning techniques. And then, these gains are adjusted again with the Trial-and-Error method to get a better accurate required set point. In this study, Arduino Uno acts as the PID controller. After that, a hardware experiment of D.C. motor position control is tested using a PID controller to control the various degree. At last, the results described that the PID controller had good performance characteristics where the required set point of the dc motor was maintained. This study confirms that the D.C. motor position control tests the various degree setpoint with minimum overshoot and fast rising time.

In this simulation, the D.C. motor position is controlled with the same PID gain to maintain the desired different function, to less overshoot, to less steady-state error and to get fast rising time.

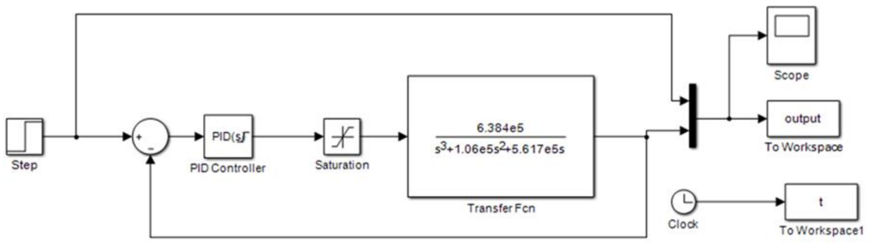


Figure 3. MATLAB Simulink Model

Firstly, K_p, K_i, and K_d gains are tuned by the MATLAB auto-tuning method. Then, the tuning results are adjusted with the Trial-and-Error method. The last obtaining results are Kp =1.3, Ki = 0.003, Kd = 0.005. These gains are put in the MATLAB Simulink model to test the position control system for different positions. Figure 3 is the MATLAB Simulink model which the research used. Figure 4 shows the step response of D.C. motor position control at the desired set point (90°, 180°, 270° and 360°). Figure 5 shows the error results for the position control of the D.C. motor.

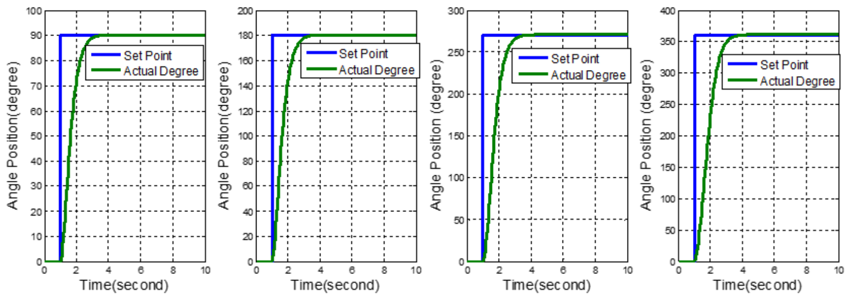


Figure 4. Step Response of Position Control DC motor (90°, 180°, 270° and 360°)
(Kp =1.3, Ki = 0.003, Kd = 0.005)

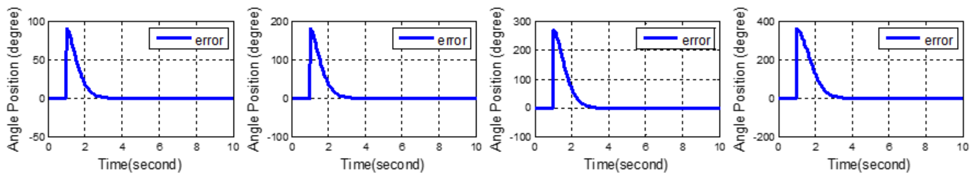


Figure 5. Error Result for Position Control DC motor (90°, 180°, 270° and 360°)

For the desired 90̊ angle position, the rising time T_r is 1.43 sec, the settling time T_s is 2.5 sec and the percent overshoot is 0.93%. For the 180̊ set point, the rising time T_r is 1.47 sec, the settling time T_s is about 2.2sec, and the percent overshoot is 0.92%. For 270̊ reference input, rising time T_r is 1.71 sec, settling time T_s is about 2.2 sec, and percent overshoot is 0.74%. For the 360̊ angle position, the rising time T_r is 1 sec, the settling time T_s is 2.2 sec and the percent overshoot is 0.56%. According to figure 5, the error of the position control system reaches almost zero for all reference input levels. The results show a slight overshoot of under 1% and steady-state error at every desired position level. So, the PID controller is good performance characteristics for the position control system of the D.C. motor.
Percent overshoot is

$$\% \text{ Overshoot } = \frac{M_{pt}-F_v}{F_v}. \quad 100\% \tag{7}$$

M_{pt} = the peak parameter of the time response
F_v = The magnitude of the input

And then, the hardware experiment is tested for position control of the D.C. motor tend to one robot arm. The closed-loop control system is implemented with an Arduino Uno board, L298N motor driver, and JGA25-370 12VDC motor, shown in Figure 6. Arduino Uno operates as the central controller in the position control system. Firstly, the output step responses are tested by obtaining PID gains from the simulation results. When these gains are returned with the Trail and Error method to get better stability responses, the final tuning gains are acquired, such as K_p=1, K_i =0.18, and K_d =0.05. Finally, Arduino code is written in the Arduino IDE software to control the D.C. motor position.

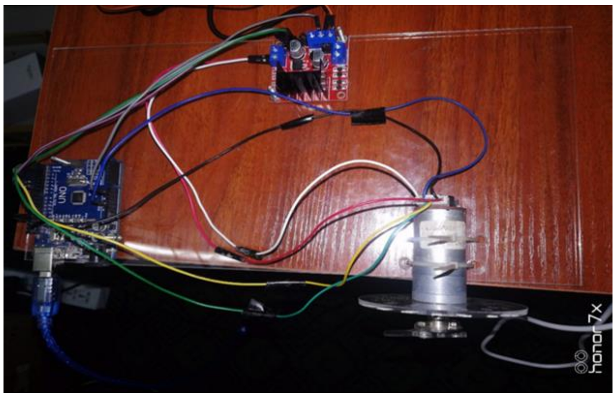


Figure 6. Real-Time Position Control System

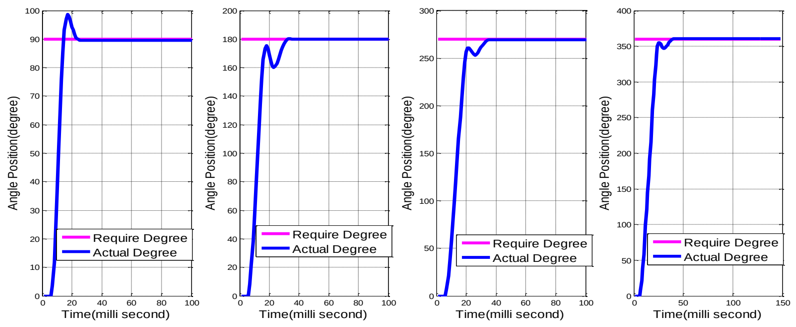


Figure 7. Experimental Result of Position Control System
(90̊, 180̊, 270̊ and 360̊)(Kp = 1, Ki = 0.18, Kd = 0.05)

Figure 7 shows the experimental results of various step responses in different input levels. The steady-state error is within +1 and -1 of the set point for all degree levels (90̊, 180̊, 270̊ and 360̊). The rising time is about 14.6 milliseconds, and the settling time is 24millisecond for a 90-degree angle. And then, the increasing time is 32.5 milliseconds, and the settling time is 34.5millisecond for 180 degrees set point. At a 270-degree setpoint, the rising time is 36 milliseconds, and the settling time is 36.5 milliseconds. Lastly, in a 360-degree position, the rising time is 39.1millisecond and the settling time is 40 milliseconds. According to all experiment results, there is no overshoot for the 180-degree 270 degrees and 360 degrees, and then, a little steady-state error and rise time is short. So, the performance of the PID controller is good for the D.C. motor position control system.

4. SYNCHRONOUS MASTER-SLAVE CONTROL OF ROBOTIC ARM

Figure 8 illustrates the overall circuit diagram for the synchronous master-slave control system. There are two microcontrollers for the master-slave robot arm control system. In the hardware experiment, two Arduino Uno, two L298N motor drivers, two D.C. motors with encoders, and a computer are used to test the angle position of the master-slave robotic arm. This work is that the master arm and slave arm are synchronized simultaneously when the variable resistor gives an input value—the master side is connected to the slave side with the I2C bus. For the experiment, we test 120 degrees, 160 degrees, 180 degree, 230 degrees, 280 degrees and 360 degrees. According to all results, there is a slight overshoot and delay time, and slave action follows the master action similarly.

And also used two D.C. motors for master control and slave control of the two robot arms. The computer is the power source for the two controllers for synchronous conditions. Therefore, the master and slave controller coding with PID controller gain could be achieved with the real-time data from the encoder for the synchronous master-slave control system.

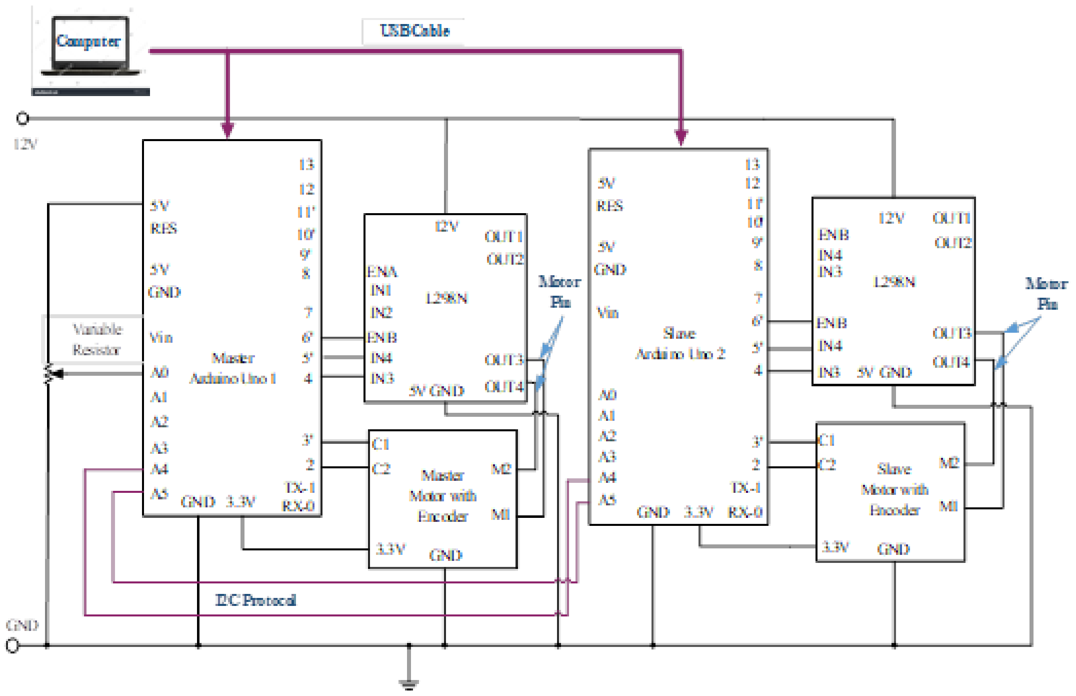


Figure 8. Overall Circuit Diagram of Synchronous Master-Slave Robotic Arm

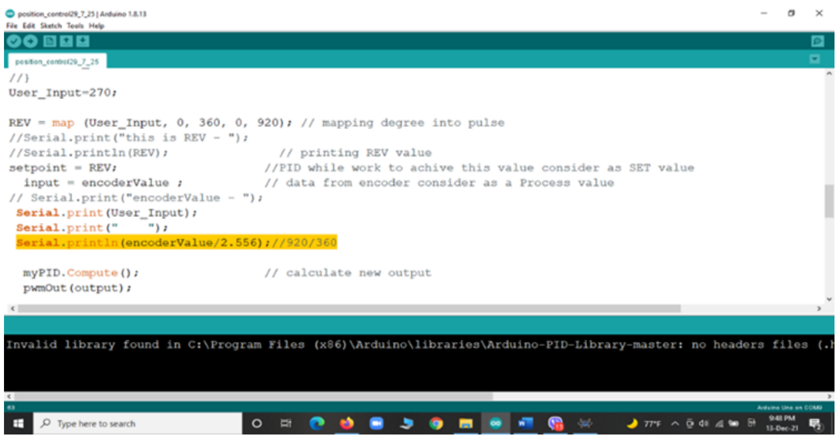
We shall have to collect the pulse data from the encoder when the D.C. motor rotates one complete cycle with the signal change from the two pins of the encoder. The actual data of the pulse from the encoder when the D.C. motor rotates one entire process is 920 or (230×4). By calculating the output degree, we could use the following equation.

$$\text{Pulses from encoder during 20 m} = 360 \times \frac{\text{Pulses in 20ms}}{920} \quad (8)$$

$$\text{Pulses from encoder during 20 m} = \text{Pulses in 20ms} \times \frac{360}{920} \quad (9)$$

$$\text{Pulses from encoder during 20 m} = \frac{\text{Pulses in 20 ms}}{2.556} \quad (10)$$

Therefore, we shall have to write the implemented coding in the program by using the above equations with Serial.println(encoderValue/2.556). That is the output degree calculation from the natural and actual degrees of the D.C. motor. The yellow colour is highlighted in the coding for the position calculation. Figure 9 mentions the necessary coding for the synchronous control system.



```
//}
User_Input=270;

REV = map (User_Input, 0, 360, 0, 920); // mapping degree into pulse
//Serial.print("this is REV - ");
//Serial.println(REV);                  // printing REV value
setpoint = REV;                        //PID while work to achive this value consider as SET value
  input = encoderValue ;               // data from encoder consider as a Process value
// Serial.print("encoderValue - ");
 Serial.print(User_Input);
 Serial.print("    ");
 Serial.println(encoderValue/2.556);//920/360

  myPID.Compute();                     // calculate new output
  pwmOut(output);
```

```
Invalid library found in C:\Program Files (x86)\Arduino\libraries\Arduino-PID-Library-master: no headers files (.h
```

Figure 9. Necessary Coding for Master-Slave Synchronous Control System

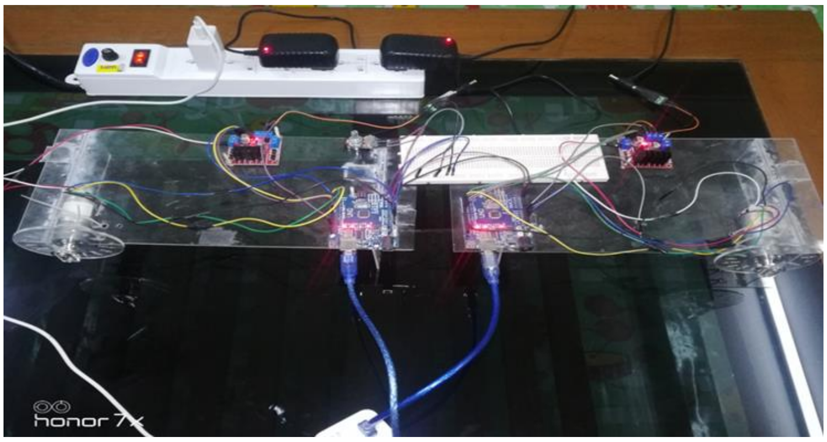
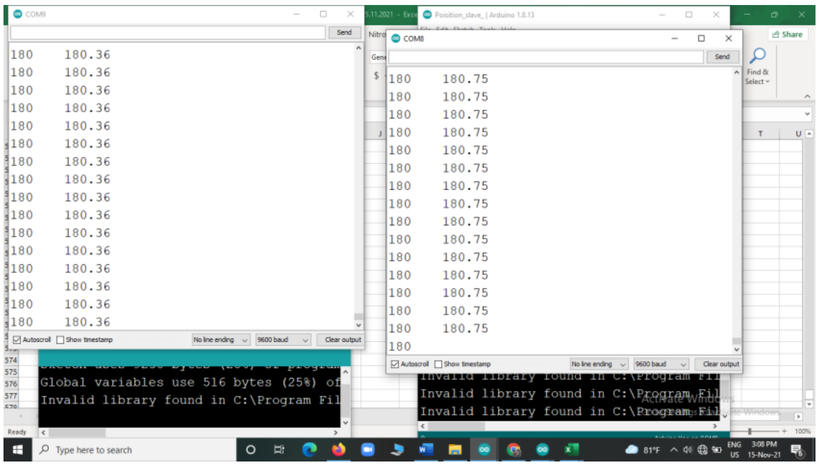


Figure 10. Hardware Experiment of Master-Slave Robotic Arm



```
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
180     180.36
```

```
Global variables use 516 bytes (25%) of
Invalid library found in C:\Program Fil
```

```
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180     180.75
180
```

```
Invalid library found in C:\Program Fil
Invalid library found in C:\Program Fil
```

Figure 11. Collected Data from the Hardware Experiment

The real-time experiment for synchronization of the master-slave robotic arm is seen in Figure 10. Firstly, the input data obtained from the variable resistor control the master-slave robotic arm simultaneously with both the master controller and slave controller. The experimental input/output data is collected from the hardware experiment, as shown in Figure 11. The variable resistor is utilized for input setpoint values. The master robotic arm is the control command according to the input data. A similar action could have occurred in the slave portion. The coding for the master and slave arm is the same. The command from the master arm can be sent as the data to the slave arm. The I2C is the communication medium between the master and slave arms.

Figure 12, 13, 14, 15,16 and 17 show the step response of the master-slave synchronization robotic arm for the desired referenced level (120°, 160°, 180°, 230°, 280°, and 360°). These result curves are obtained from the real-time input/output of the hardware experiment. All desired input degrees are reached to get the target with only one PID gain. Although there is a slight overshoot and delay time for all reference levels, it can see that almost no error for all desired set points. Steady-state error is almost none above and below the reference input level.

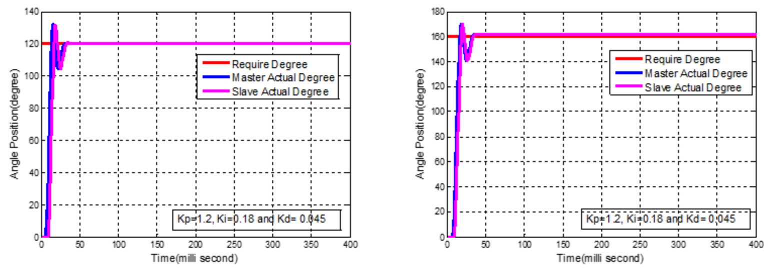


Figure 12. Step Response of Master-Slave Synchronous Robotic Arm (120° and 160°)

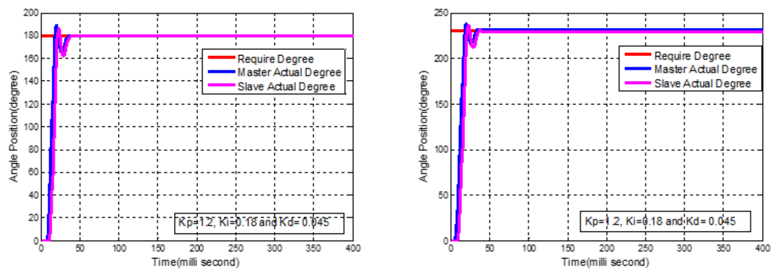


Figure 13. Step Response of Master-Slave Synchronous Robotic Arm (180° and 230°)

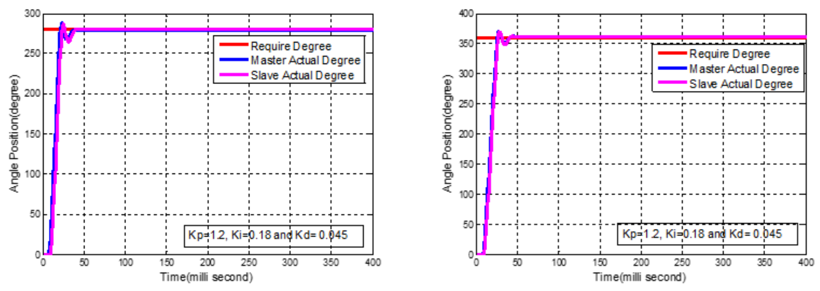


Figure 14. Step Response of Master-Slave Synchronous Robotic Arm (280° and 360°)

For steady-state error, at 120°, it is 0.11 degrees more than the desired set point for both the master arm and slave arm. At 160°, it is 0.58 degrees more than the reference degree on both the master arm and slave arm. At 180°, it reaches 0.42 degrees less than the reference degree for the master arm and 0.03 less than the reference input for the slave arm. At 230°, it has 0.83 degrees more than the reference degree for the master arm and 0.74 less than the required degree for the slave arm. In 280°, this is 0.66 degrees less than the set point for the master arm and 0.91 degrees more than the desired for the slave arm. Finally, at 360°, this is 0.5 degrees more than the set point for both the master and slave arms.

For percent overshoot, this reaches 10.2% for the master arm and 9.55% for the slave arm in 120°. The percent overshoot of the master and slave robotic arm is 6.125% in 160°. For 180°, P.O. is 5% for the master arm and 3.46% for the slave arm. For 230°, it is 3.26% for the master arm and 2.4% for the slave arm. At 280°, it reaches 2.86% for the master and 2.56% for the slave arm. Finally, at 360°, the percent overshoot of the master arm is 2.78%, and that of the slave arm is 2.59%.

For a rising time, at 120°, the master robotic arm's rising time (T_s) is 13.43millisecond, and the slave robotic arm's rising time (T_s) is 16.46millisecond. At 160°, this time is 16.34millisecond for the master side, and this time is 18.34millisecond for the slave side. In 180°, the rising time of the master is 18.15millisecond, and that of the slave is 21.14millisecond. For 230°, it is 18.2millisecond for the master arm, and it is 21.12millisecond for the slave. Finally, at 280°, it reaches 21.35millisecond for the master and 23.2millisecond for the slave. Final testing 360̊ obtains 25.65millisecond for the master arm and 26.56millisecond for the slave arm.

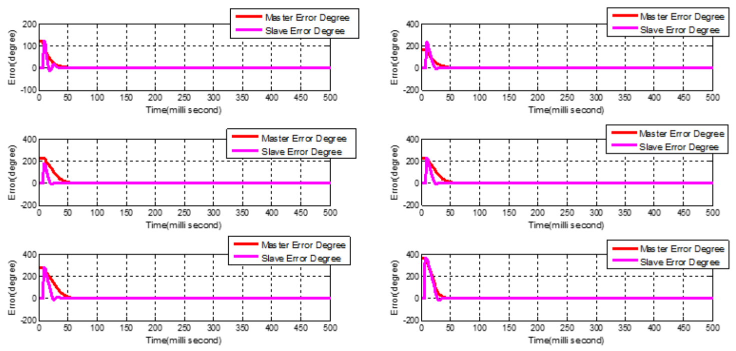


Figure 15. Error Results for the Difference Desired Degree
(120̊, 160̊, 180̊, 230̊, 280̊ and 360̊)

The above figure 18 shows the error results for the various set points (120̊, 160̊, 180̊, 230̊, 280̊ and 360̊). According to the results, we can see that the error degrees for the hardware experiment for the synchronous master-slave robotic arm almost reach zero.

5. STATISTIC TABLE FOR TIME DELAY RESULTS

The performance comparisons of the time delay analysis on the present works and recent works are given in the following Table.1. The time delay in milliseconds is only 3 by comparing with the other works. According to this performance comparison, the present system implementation is acceptable for Synchronous Master-Slave Robot Arm Control System in real world applications.

Works	Methods	Delay Time(ms)
[5]	Robust trajectory tracking of Master-Slave surgery robot system based on PD with Integral Sliding Mode Control	10
[12]	Developments of New Anthropomorphic Robot Hand and its Master Slave System	10
This work	Master-Slave Synchronization of Robotic Arm using PID Controller	3

6. CONCLUSION

In the simulation of position control, the desired position is adjusted using the same PID gain. Percent overshoots are within 1 percent of the expected value at all ranges. And then, the rising time is between one second and two seconds at all required degree set points. Similarly, there is a little steady-state error, short rise time and settling time. According to the simulation and real-time results, there is not much difference between the two robotic arms.

In the central system that synchronization of the master-slave robotic arm, the experimental results are tested with the target position (120̊, 160̊, 180̊, 230̊, 280̊ and 360̊) using the same PID gain. We can see that the slave arm follows the master's action, although there is a little time delay. The rising time for both master and slave arms is below 1 second. Except for the 360-degree target point, the percent overshoot is below 7% for all reference levels. The steady-state error of the master and slave arm is below 1 degree for all desired targets. According to the error result curves, the error signal of both master and slave arms almost reaches zero. The time delay is the least compared to other recent research work. Therefore, it is assumed that this research obtains to achieve the desired target. Thus, the PID controller performance is satisfied for this work.

ACKNOWLEDGEMENTS

The author wants to thank many colleagues from the Department of Electronic Engineering at Mandalay Technological University, for their kindness, encouragement, and assistance while writing this paper.

REFERENCES

[1] Chiang-Heng Chien, Chen-Chien James HSU, and Chiang-Ju Chien, “Multiple Master-Slave FPGA Architecture of a Stereo Visual Odometry”, IEEE Access, Volume 9, July 20,2021.

[2] Zheng Chen, Shuifeng Yan, Mingxing Yuan, Bin Yao, and Jinfei Hu, “Modular Development of Master-Slave Asymmetric Teleoperation Systems with a Novel Workspace Mapping Algorithm”, IEEE Access, Volume 6, February 27, 2018.

[3] Jaehyun Shin, Yongmin Zhong, and Chengfan Gu, “Master-slave robotic system for needle indentation and insertion”, Computer Assisted Surgery, Volume. 22, NO. S1, 100–105, ISSN 2469-9322, September 22, 2017.

[4] F. J. Torres, G. V. Guerrero, C. D. García, J. F. Gómez, M. Adam, and R. F. Escobar, “Master-Slave Synchronization of Robot Manipulators Driven by Induction Motors”, IEEE LATIN AMERICA TRANSACTIONS, Volume. 14, NO. 9, SEPTEMBER 2016.

[5] Hansoul Kim, Minho Hwang, Donghoon Baek and Dong-Soo Kwon, “Robust trajectory tracking of Master-Slave surgery robot system based on PD with Integral Sliding Mode Control”, 15th International Conference on Ubiquitous Robots (UR), June 27-30, 2018.

[6] Marwa Fathallah, Fatma Abdelhedi, and Nabil Derbel, “Synchronization of multi-robot manipulators based on high order sliding mode control”, International Conference on Smart, Monitored and Controlled Cities (SM2C), February 17-19, 2017.

[7] Wei Cheng, Xuelin Wang, and Haiyan Ma, “Master-Slave Force Control Based on Grey GM(1,1) Model of Robot Gripper”, Proceeding of the International Conference on Advanced Mechatronic Systems, August 22-24, 2015.

[8] Yan Kai. Chao, Yong Fei. Zhou, and Yu Lin. Xu, “Mobile Robotic Control System Based on Master-Slave Teleoperate Mechanical Arm”, 25th Chinese Control and Decision Conference, May 25-27, 2013.

[9] Shuxiang Guo, Songyuan Zhang, Zhibin Song, and Muye Pang, “Development of a Human Upper Limb-like Robot for Master-slave Rehabilitation”, ICME International Conference on Complex Medical Engineering, May 25-28, 2013.

[10] Yassine BOUTERAA, and Jawhar GHOMMAM, “Synchronization Control of Multiple Robots Manipulators”, 6th International Multi-Conference on systems, Signals and Devices, March 23-26, 2009.

[11] Tang Xinxing, YAMADA Hironao, and Anmad Anas Yusof, “Virtual Reality-Based Master-Slave Control System for Construction Tele-operation Robot”, International Conference on intelligent Computing and Intelligent Systems, November 20-22, 2009.

[12] Tetsuya Mouri, Haruhisa Kawasaki, and Katsuya Umebayashi, “Developments of New Anthropomorphic Robot Hand and its Master Slave System”, IEEE/RSJ International Conference on Intelligent Robots and Systems, August 02-06, 2005.

[13] Ikuo Yamano, Kenjiro Takemura, Ken Endo, and Takashi Maeno, “Method for Controlling Master-Slave Robots using Switching and Elastic Elements”, IEEE International Conference on Robotics and Automation, May 11-15, 2002.

[14] Carl D. Kopf, and Tetsuro Yabuta, “Experimental Comparison of Master/ Slave and Hybrid Two Arm Position/Force Control”, IEEE International Conference on Robotics and Automation, April 24-29, 1988.

[15] Yin Hnin Thet Htun, May Su Hlaing, Tin Tin Hla, “DC Motor Real Time Position Control Using PID Controller,” Proceedings of the Universal Academic Cluster International Online Conference in Bangkok, pp 276-289, Thailand, 2021.

[16] Yin Hnin Thet Htun, May Su Hlaing, Tin Tin Hla, “Comparative Study of PID, PI-D, I-PD Controller for DC Motor Speed Control,” Proceedings of the Universal Academic Cluster International Online Conference in Bangkok, pp 290-302, Thailand, 2021.