Imperial College London

UNDERGRADUATE RESEARCH OPPORTUNITIES PROGRAMME (UROP)

IMPERIAL COLLEGE LONDON

DEPARTMENT OF EEE

Remote Control of the Robotic Arms

Author:
Yuxuan Gu

Supervisor:
Prof. Thomas Parisini

Co-supervisor:
Dr. Kaiwen Chen

December 26, 2023

Contents

1 Introduction **3**

2 System Framework **4**

2.1 Hardware 4

2.1.1 Camera 4

2.1.2 Bus servo motor 5

2.2 Control Block Diagram 6

2.3 Inverse Kinematics 7

3 Implementation **9**

3.1 Raspberry Pi 9

3.2 Robot Operating System 2 (ROS2) 10

3.2.1 Introduction of ROS2 Data Distribution Service (DDS) middleware 10

3.2.2 Node Graphs 11

3.2.3 Record and play back 12

3.2.4 Communication between nodes 13

4 Remote PID Tracking Control **14**

4.1 Stability Analysis 14

4.2 Advantages of Remote Controller 15

4.3 Result Analysis 15

5 Conclusion and Future Work **17**

A Videos of experiments **18**

Chapter 1

Introduction

A teleoperation system consists of at least one master and one slave [1]. The master is controlled by human using operator and the communication between master and slave needs to be established so that the slave can be controlled remotely by the master.

[1],[2] have pointed out that teleoperation can extend human expertise to remote or inaccessible sites and scale the power. This improves efficiency and reduce the requirement for manpower.

Additionally, increasing range of applications of teleoperation nowadays makes it more significant in our daily life. Telesurgery is one of the state-of-the-art techniques. This means surgeons can examine patients in real-time from remote positions via electronic communication. They do not need to be physically present in the operating theatre, which is known as telepresence or telemedicine [3]. In this paper, 5G-enabled Tactile Internet are used to ensure ultra-low latency, ultra-high reliability, security and privacy.

Furthermore, teleoperated robots could replace human in the hazardous environment. For instance, the authors in [4] proposed an admittance controller based teleoperation system aiming to replace human in the deposited iron lump removal task in the steel mill.

The main challenge of teleoperation is latency, which will lead to closed-loop instability. There are unavoidable delays between master and slave. It takes time to transmit information from one node to the other. Therefore, many control techniques aiming to minimise the negative effect of delays emerged. For example, predictive feedback [5] is exploited to predict the evolution of the state variable for the period of the delay, so that the slave side could change in advance accordingly. In addition, passivity-based control was proposed to cope with time delay issues [6].

There is a substantial amount of published research focusing on controlling robotic arms using Robot Operating System (ROS). Megalingam *et al.* demonstrated in [7] a ROS-based approach for controlling a 6-DOF robotic arm using the Controller Area Network (CAN). This system comprises six actuator nodes, each representing a degree of freedom, interconnected through the CAN bus to establish a communication network. These nodes receive instructions from a central controller (Raspberry Pi). Notably, joystick commands are transmitted via WiFi. The robotic arm's behavior is simulated within the Gazebo environment, and its visualization is facilitated through Rviz in ROS.

In [8], Zhu *et al.* introduced a built-from-scatch 6-DOF robotic arm. They employed a PID controller for each joint incorporating disturbance compensation control to address the influence of gravity on the arm's dynamics. The team tackled both the forward and inverse kinematics challenges by applying the Denavit–Hartenberg conventions. Notably, their software architecture is based on the ROS framework.

In this project, the remote control of a robotic arm to track a coloured object is studied. The classical transfer function is employed to model the servo motors and PID controllers are utilised to keep track of the object using camera feedback. Addressing the challenge of latency during message transmission, controller parameters are carefully tuned to minimize its detrimental effects. Moreover, inverse kinematics are determined geometrically, with ROS2 linking network nodes. As of my best knowledge, few literature exists regarding the combination of ROS2 and remote control of robotic arm.

Chapter 2

System Framework

This project involves remote control of the robotic arm to track the coloured object and the following sections will introduce the system framework in terms of hardware and software components.

2.1 Hardware

As illustrated in Fig. 2.1, xArm 1S is a 6-degree-of-freedom (6-DOF) robotic arm.

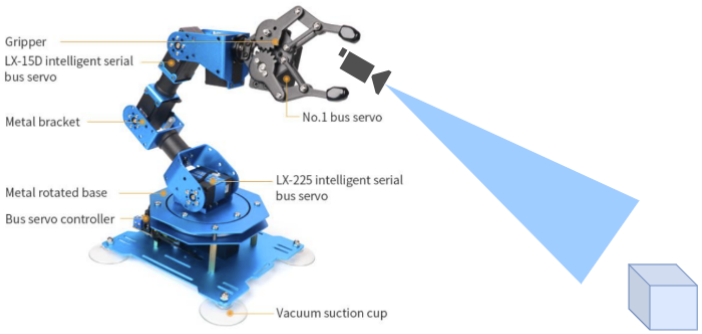


Figure 2.1: xArm 1S Robotic Arm

In this project, only 4 DOFs are used to track a coloured object using camera feedback. The remaining two DOFs, including rotation of the front wrist and the expansion of the grip, are not used.

2.1.1 Camera

The camera used in this project is 1080p UVC-Compliant USB Camera Module with Metal Case (shown in Fig. 2.2).

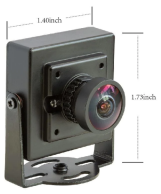


Figure 2.2: USB camera

It is based on the 1/2.8" Sony IMX291 image sensor and compatible with Raspberry Pi 4B and the video frames are not distorted. Its frame rate is 30fps at 1920×1080 for H.264 and MJPG format. The focusing range is from 6.56ft (2M) to infinity. It is connected to Raspberry Pi via USB cable without extra drivers to be installed.

2.1.2 Bus servo motor

Since the robotic arm has 6 DOFs, six bus servo motors are required to control each joint and the spatial arrangement of them are depicted in Fig. 2.3. The link lengths involved in this projects span between motor 6 and 5, 5 and 4, 4 and 3, 3 and 2, and 2 and the end-effector, which are around 6.1, 10.2, 9.6 and 16.6 cm respectively.

Figure 2.3: Six servo motors

Servo motor 1 is used to rotate the gripper. Servo motor 2,3,4,6 have the type LX-15D (shown in Fig. 2.4(a)), offering a torque of 17Kg·cm and servo motor 5 has type LX-225, providing an enhanced torque of 25Kg·cm (shown in Fig. 2.4(b)). Both of these motors have rotation range 0°-240° and are controlled by UART serial command packages with baud rates of 115200.

(a) LX-15D (b) LX-225

Figure 2.4: Two types of bus servo motors

The only command package used in this project is moving the motor to a specified position in a given duration. Table 2.1 shows the command package of CMD_SERVO_MOVE. For individual motor control, Num. of motors is replaced by one and the data length is constant 8. In the Parameter section, Prm1 is the number of motors which is one in this scenario. Prm2 is the most significant 8 bits of duration and Prm3 is the least significant bit of duration. Prm4 is the servo motor ID. Prm5 is the most significant 8 bits of the pulse width (angle) and Prm 6 is the least significant 8 bits of the pulse width (angle). For instance, if motor 3 is expected to move to position 800 in a duration of 1000ms, the command package should be [0x55 0x55 8 3 0x01 0xE8 0x03 0x03 0x20 0x03].

Table 2.1: Command Package of CMD_SERVO_MOVE

Frame Header	Data Length	Command	Parameter
0x55 0x55	Num. of motors×3+5	3	Prm1 · · · PrmN

2.2 Control Block Diagram

The color tracking task involves guiding the robotic arm to accurately trace the movements of a target object. To successfully accomplish this task, two critical pieces of information are required: the present position of the object and its expected position.

The robot obtains the current position data through a camera feedback, while the objective is for the object to be ideally positioned at the center of the frame both horizontally and vertically. Furthermore, it's essential to maintain a consistent distance between the camera and the object. This results in the implementation of three distinct controllers: one for horizontal positioning, another for vertical alignment, and a third for controlling the camera-to-object distance.

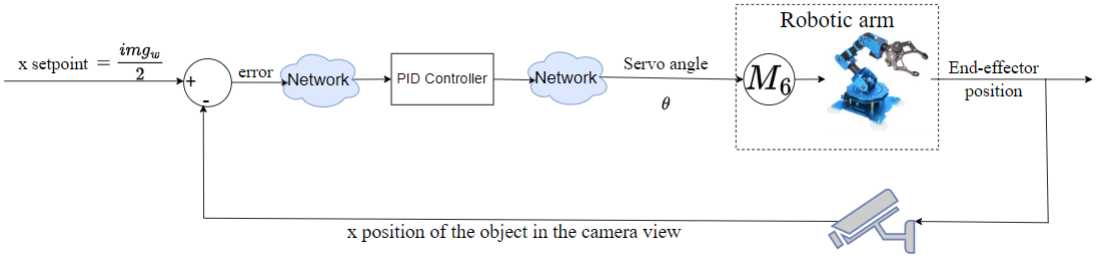


Figure 2.5: x PID Controller

Illustrated in Fig. 2.5, the x PID controller is designed to align the object at the horizontal center of the frame window. Given that solely servo motor 6 can manipulate the object's horizontal positioning within the window, the controller's output directly control the angle of servo motor 6.

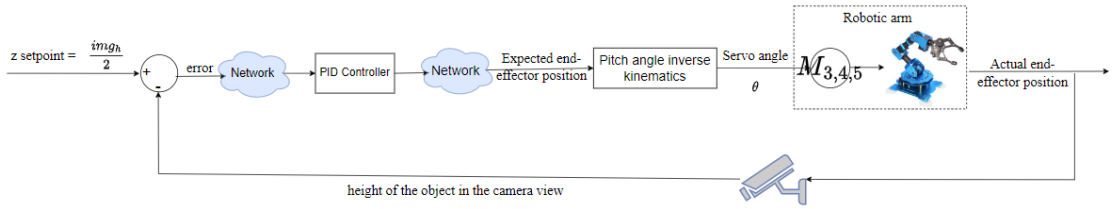


Figure 2.6: z PID Controller

Depicted in Fig. 2.6, the z controller is employed to ensure the object's alignment with the vertical center of the frame window. Given that motors 3, 4, and 5 jointly govern the z coordinate of the end-effector, the controller's output pertains to the expected end-effector position. Subsequently, through the inverse kinematics of the pitch angle, the angles for servo motors 3, 4, and 5 are computed and then transmitted to the robot for implementation.

Similarly, as indicated in Fig. 2.7, the y controller aims to maintain a constant separation between the camera and the object. The direct acquisition of distance from the camera isn't feasible, prompting the utilization of the object's area within the frame window as a substitute for distance. This approach is intuitive, as proximity between the object and the camera results in a larger object area within the frame window, and conversely, greater distance yields a smaller object area.

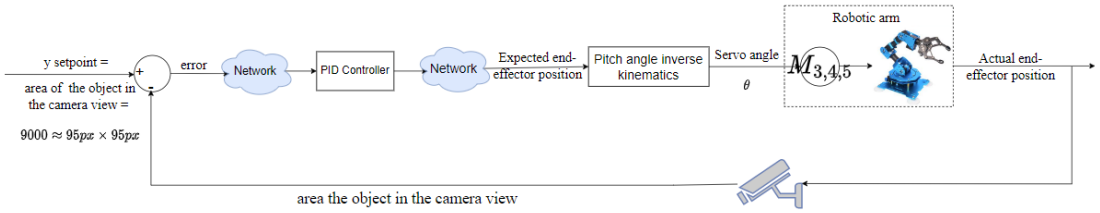


Figure 2.7: y PID Controller

2.3 Inverse Kinematics

Inverse Kinematics entails determining the joint variables from the provided end-effector position and orientation [9]. When dealing with a robotic arm, each servo motor angle is computed by utilizing inverse kinematics, which takes into account the desired end-effector position and the pitch angle.

To simplify the calculation and fulfill the color tracking task, the reduced set of 4 Degrees of Freedom (DOF) is considered, excluding the rotation and grasp extension aspects within this project's scope. The following shows the detailed analysis of two configurations of the model using geometric method.

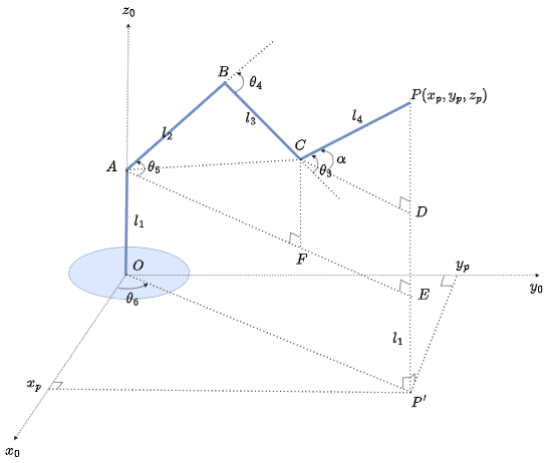


Figure 2.8: Configuration 1

Fig. 2.8 is a specific configuration of a 4-DOF robotic arm, where the blue circle designates the robot's base. The Cartesian coordinate system's origin, denoted as O, is situated at the base's center. The base rotation is managed by servo motor 6, with its rotation angle θ_6 measured relative to the x_0 axis. Joint A hosts servo motor 5, governing the angle θ_5. Additionally, servos 4 and 3 are positioned at joints B and C, controlling angles θ_4 and θ_3 respectively.

Furthermore, P' is the projection of P onto the xy plane and CD is parallel to both AE and OP' while CF and PP' are perpendicular to OP'. Notably, the respective link lengths are l_3, l_4, l_5, and l_6, as indicated in the illustration.

The position of the end-effector is represented as $P(x_p, y_p, z_p)$, and the pitch angle with respect to the horizontal plane is denoted as α. These variables are given, and the task is to compute the joint angles θ_6, θ_5, θ_4, and θ_3.

The following equations are discernible from Fig. 2.8.

$$\alpha = \theta_3 - \theta_4 + \theta_5,$$

$$PD = l_4 \sin(\alpha),$$

$$CD = l_4 \cos(\alpha),$$

$$OP' = \sqrt{x_p^2 + y_p^2},$$
$$AF = AE - FE = OP' - CD,$$
$$CF = DE = z_p - l_1 - PD,$$
$$AC = \sqrt{AF^2 + CF^2}.$$

To determine θ_4, begin by calculating $\angle ABC$, and its complementary angle will yield the desired result. $\angle ABC$ can be solved using the cosine rule

$$\cos\angle ABC = \frac{l_2^2 + l_3^2 - AC^2}{2l_2l_3}.$$

Therefore,

$$\theta_4 = 180^\circ - \angle ABC.$$

Similarly, θ_5 can be derived using cosine rule.

$$\cos\angle CAB = \frac{AC^2 + l_2^2 - l_3^2}{2l_2AC},$$
$$\angle CAF = \arctan(\frac{CF}{AF}).$$

Therefore,

$$\theta_5 = \angle CAB + \angle CAF.$$

Finally, θ_3 can be calculated from α, θ_4 and θ_5.

$$\theta_3 = \alpha - \theta_5 + \theta_4.$$

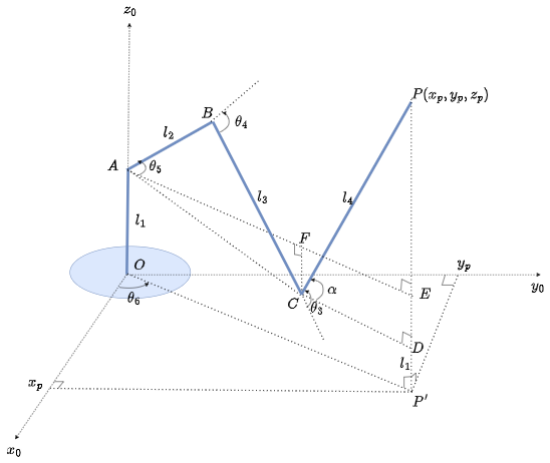


Figure 2.9: Configuration 2

Fig. 2.9 illustrates a different configuration, i.e., C is located below F, indicating $CF < 0$. The only difference with the first configuration is the calculation of θ_5.

In this configuration,

$$\theta_5 = \angle CAB - \angle CAF.$$

Other calculations remain identical as the first one.

Chapter 3

Implementation

This section introduces the pivotal role played by the Raspberry Pi within the system framework. It delves into the intricate node graphs within this project's context. Furthermore, it provides an illustration showcasing inter-node communication across multiple devices sharing a common Local Area Network (LAN). Lastly, an in-depth exploration is conducted into the ROS2's bag recording functions, with an insightful examination of its seamless integration with the MATLAB and Simulink.

3.1 Raspberry Pi

The Raspberry Pi is a series of single-board computers (SBCs) [10], with the present project employing the Raspberry Pi 4 Model B version. Raspberry Pi OS (64-bit) is installed on a 32GB micro-SD card.

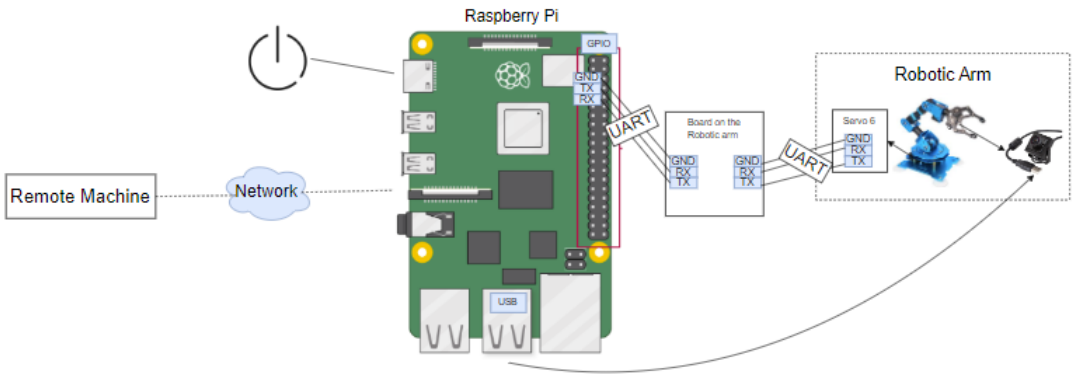


Figure 3.1: Role of Raspberry Pi

As shown in Fig. 3.1, raspberry pi plays a pivotal role: dispatching of command packages to the bus servo motors of the robotic arm via UART protocol, executing image processing to track a designated object, and seamlessly facilitating bidirectional communication between the actuator and the controller on a remote machine through the ROS2 network framework.

The UART protocol mentioned above stands for Universal Asynchronous Receiver/Transmitter, which is used for data transmission between devices.

As illustrated in Fig. 3.2, UART employs two wires – one for data transmission and one for reception – employing asynchronous data transfer, with a start and stop bit framing each data byte [11].

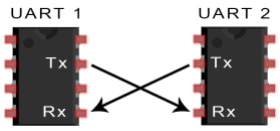


Figure 3.2: UART protocol

The Raspberry Pi sends command packages to the robotic arm's board, which then relays these messages to the interconnected bus servo motors. As all six servos are linked via a singular wire, designated as bus servo motors, the sixth servo can efficiently distribute commands to others using the specific ID number outlined within the package. All these communication between motors, board, and raspberry pi are achieved via UART.

According to the official ROS2 on Raspberry Pi document, Raspberry Pi OS is based on Debian which receives Tier 3 support, but it can run Ubuntu docker containers for Tier 1 support. Therefore, a docker container is built in the Raspberry Pi OS.

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. The docker-compose.yml and dockerfile used in this project can be found in this GitHub repo and more technical details of raspberry pi setup can be found in this GitHub repo.

3.2 Robot Operating System 2 (ROS2)

ROS2, the successor of ROS1, is a middleware connecting software tools and the operating system. The role of ROS2 in this project is to link and communicate between different nodes, such as camera, servo motor, etc.

In this section, Data Distribution Service (DDS) middleware will be introduced, the communication across multiple devices based on ROS2 network will be explored, the node graphs including all the nodes and topics and their relationships will be discussed, and finally the record and play back tools will be examined.

3.2.1 Introduction of ROS2 Data Distribution Service (DDS) middleware

Maruyama *et al* in [12] have pointed out that ROS2 is more suitable for the real-time system. In this project, the remote controller must respond in real time to track the coloured object, and therefore ROS2 is chosen.

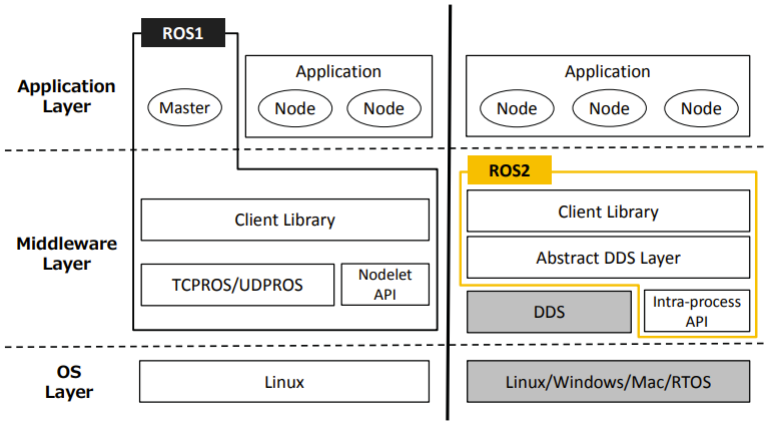


Figure 3.3: ROS1/ROS2 architecture

As depicted in Fig. 3.3, for the communication infrastructure, Data Distribution Service (DDS)

is used in ROS2 instead of the traditional TCP/IP protocol used in ROS1.

DDS (data distribution service) is a middleware protocol and API standard for data transferring using a publisher-subscriber model [13]. TCP/IP protocol introduces variable latency due to factors like network congestion, packet drops, and priority conflicts. DDS solves this latency problem by introducing Data-Centric Publish-Subscribe (DCPS) model [14], meaning that publishers can send data without needing to know who the subscribers are, and vice versa. This reduces overhead associated with discovering, connecting, and managing communication partners, resulting in lower latency.

Additionally, in ROS2, the master node is eliminated to prevent a single point of failure—a weakness present in ROS1's design. The master node in ROS1 serves to coordinate communication among nodes by maintaining a node registry, managing topic interactions, and assisting node discovery. However, if this master node fails in ROS1, it leads to a complete system failure. By removing the master node in ROS2, the network becomes more robust and resilient, as nodes communicate directly, mitigating the risk of a central failure affecting the entire system.

Within the scope of this project, the communication method used between nodes in ROS2 is the publish/subscribe messaging model as illustrated in Fig. 3.4.

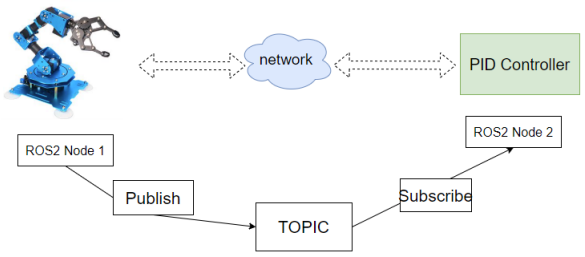


Figure 3.4: Publish/Subscribe messaging model

The robotic arm functions as a ROS2 node situated on a local machine, while the PID controller operates as a separate node on a remote machine. These two components establish a connection within the ROS2 network through the exchange of messages via topic publication and subscription mechanisms.

3.2.2 Node Graphs

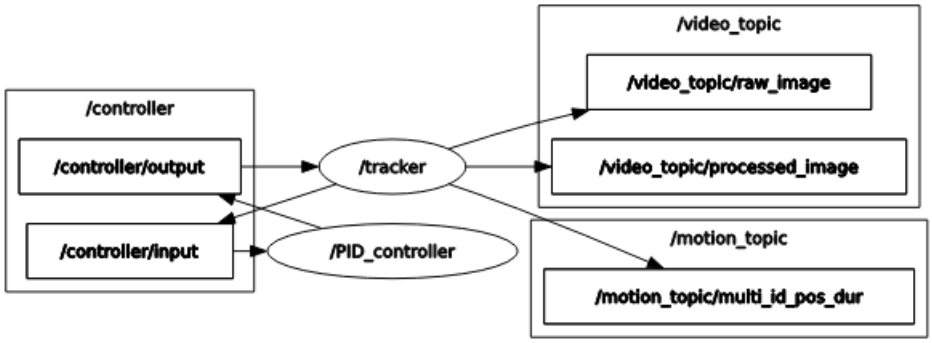


Figure 3.5: Node graph

Fig. 3.5 shows the node graph generated by `rqt` in ROS2. The system comprises two distinct nodes: `tracker` and `PID_controller`. The `tracker` functions as core, responsible for publishing video frames to `/video_topic`, pulse width (angle) of each servo motor to `/motion_topic`, feedback value to the `/controller/input` and subscribes to the `/controller/output`.

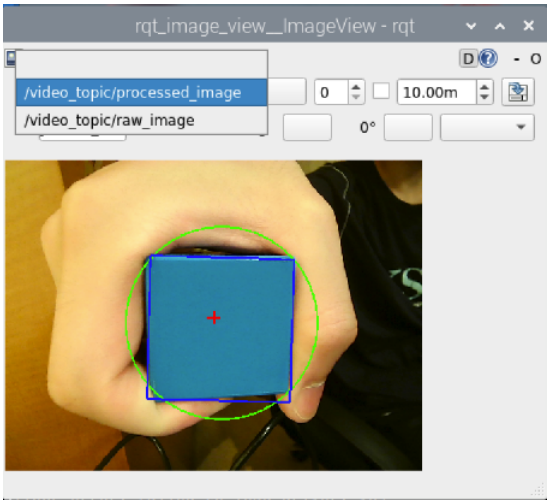


Figure 3.6: Video frames shown in rqt_image_viewer

Additionally, `/PID_controller` is a node that subscribes to the `/controller/output` and apply the PID algorithm, and after which the output is published to the `/controller/output`. Furthermore, the video frames can be viewed in real time using `rqt_image_viewer` (as shown in Fig. 3.6)

For more technical details, please refer to this GitHub repo.

3.2.3 Record and play back

ROS2 provides tools to record the messages published to a topic and store them in .db3 and metadata.yaml file as shown in Fig. 3.7. .db3 is a database file created by the SQLite software, which contains all the data recorded and .yaml is a human-readable language often used to write configuration files, containing information of the data, such as topic name, message type, etc.

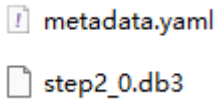


Figure 3.7: Recorded messages files

ROS Toolbox in MatLab is used to extract the recorded message and analyse it. As indicated in Fig. 3.8, ROS Toolbox is a software interface to connect ROS2 nodes running in MatLab and Simulink to other nodes in the ROS2 network. According to the ROS Toolbox System Requirements, MatLab 2023a, Visual Studio (C++ Compiler) 2022 and Python 3.8 are chosen to fulfil the constraints. More technical details of ROS Toolbox usage can be found in this GitHub repo.

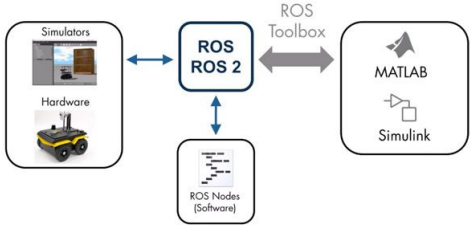


Figure 3.8: ROS Toolbox in MatLab

3.2.4 Communication between nodes

As discussed in 3.2.1, the communication between nodes are established by the DDS middleware in ROS2. In this project, two devices are involved: a Raspberry Pi and a virtual machine. They are in the same Local Area Network (LAN). Therefore, by specifying the IP address on each device, a route is established for seamless topic publication and subscription, ensuring the visibility of topics and nodes in the range of ROS2 network.

Technically, for two nodes to be recognisable from each other in the Local Area Network (LAN), each IP address needs to be provided in the .bashrc script file which will be run automatically whenever a new terminal is open. The codes example is provided in Fig. 3.9. More technical details can be seen in this GitHub link.

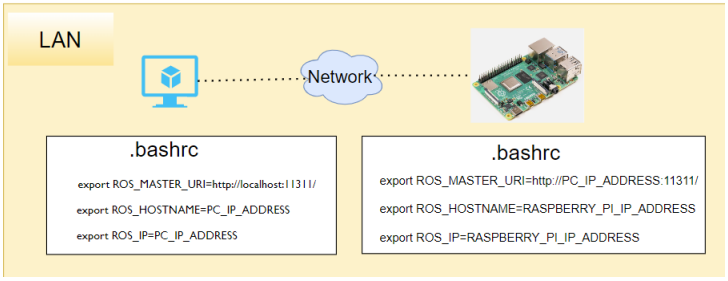


Figure 3.9: Communication between PC and Raspberry Pi

Chapter 4

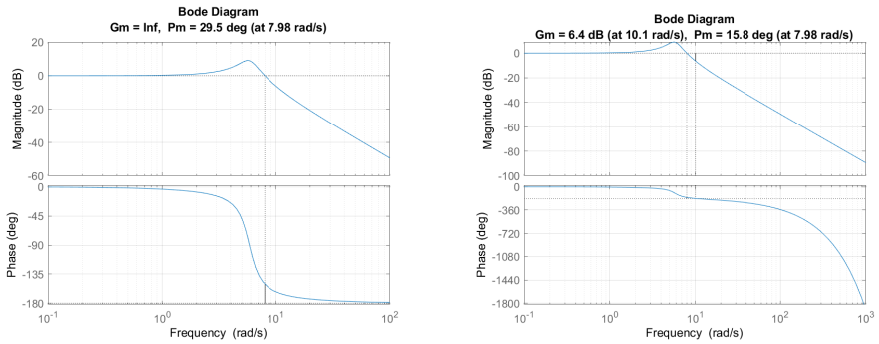
Remote PID Tracking Control

This chapter investigates the effect of separating the controller to a remote machine, discusses the benefit of remote control, plot a 3D trajectory of end-effector and build a approximated 2^{nd} order model by analysing the step response.

4.1 Stability Analysis

Within this project, a deliberate decision has been made to separate the controller from the main tracker function, relocating it from the local Raspberry Pi to a remote machine. This strategic shift is primarily motivated by the pursuit of streamlined future maintenance and development processes.

When the controller is moved to remote machine, unstable behaviour is observed (as demonstrated in this YoutTube video) due to time delay in the network communication. Take servo motor 6 as an example. Assume it is a Linear Time Invariant (LTI) system and a constant time delay is introduced. The time delay can be represented as $e^{-\tau s}$, meaning that it will decrease the phase without affecting the magnitude on the bode plot.



(a) Bode plot of servo motor 6 (b) Bode plot of servo motor 6 with 30ms delay

Figure 4.1: Bode plot of transfer function of servo motor 6

As shown in Fig. 4.1, the normal transfer function of motor 6 (derived in 3.2.3) has phase margin of 29.5 degrees. When 30 ms delay is added to the system, the phase margin diminishes to 15.8 degrees. This decreases the stability margin, rendering the system more susceptible to instability. Therefore, to enhance the stability of the system, one solution is provided: decrease the Proportional term in the PID controller. This will shift the magnitude plot downwards, decreasing the cross-over frequency ω_c and hence increase the phase margin and stability margin.

4.2 Advantages of Remote Controller

There are several advantages of splitting the controller to a remote machine. The remote machine can serve as a robust server, housing specialized hardware such as GPUs tailored for deep learning tasks. Embedding these hardware components directly into a local machine can be extremely expensive. Conversely, by sharing resources and distributing data to multiple end users effectively mitigates the cost while maintaining efficient performance [15].

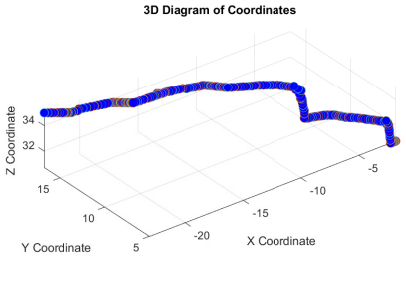
Furthermore, this specialization significantly boost overall productivity [16]. For example, a control engineer can wholly concentrate on refining control algorithms on the remote machine, extricating concerns about other system aspects.

Additionally, networked control systems offer the advantage of flexible distribution of control components across various networked locations, enabling easy modifications or upgrades to accommodate new or existing system components without significant structural changes [17]. This arrangement also facilitates streamlined central management, which becomes particularly crucial when scaling up the system to control multiple robots. While this project focuses on controlling a single robot, the scalability demands of managing multiple robots highlight the need for a strategically positioned centralized controller in the remote machine.

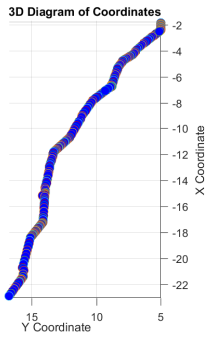
4.3 Result Analysis

In this project, two analyses have been carried out: 3D trajectory of the end-effector and step response of the robotic arm.

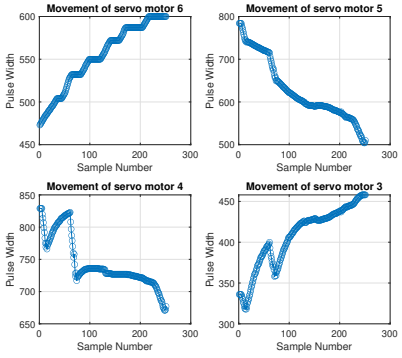
Fig. 4.2 visually presents the 3D trajectory of the end-effector, along with the corresponding pulse width (angle) values of servo motors 3, 4, 5, and 6 throughout the entire movement sequence. A comprehensive view of the experiment is available for observation within this YouTube video.



(a) 3D trajectory



(b) Trajectory in xy plane



(c) Servo motor movement

Figure 4.2: Movement of the robotic arm

Furthermore, a step response analysis of the robotic arm has been conducted. In this context, the term "step response" refers to the system's reaction to an abrupt alteration in the position of the tracked object.

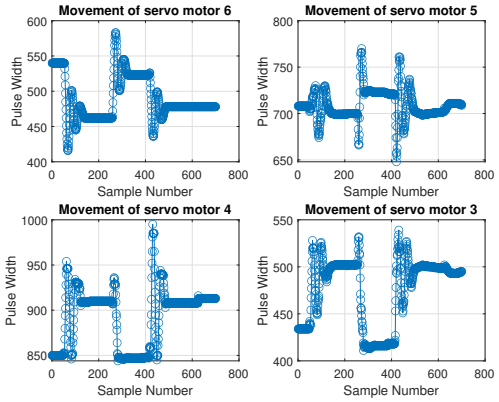


Figure 4.3: Step response of motor 3,4,5,6

Fig. 4.3 illustrates the step response of all motors—motor 3, 4, 5, and 6. Notably, motor 6's step response exhibits a notably regular shape owing to its direct control by a PID controller. In contrast, the remaining motors are collectively governed by two controllers, which imparts distinct characteristics to their step responses. The experiment details are recorded in this YouTube video.

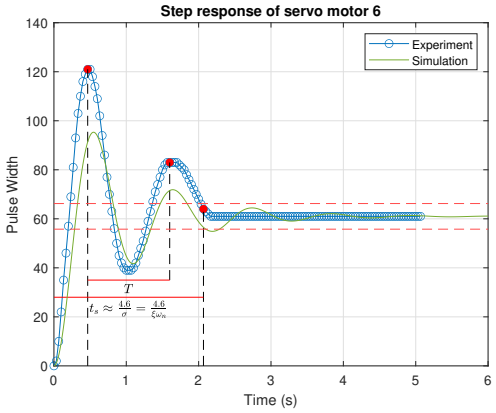


Figure 4.4: Step response experiment and simulation of motor 6

Fig. 4.4 zooms in on a segment of motor 6's response, enabling the calculation of essential parameters such as settling time (2.1s), peak value (583), percentage overshoot (11.47%) and the 'period' of oscillation (1.13s). In this particular scenario, the sampling time is roughly $\frac{1}{30}$ seconds, aligning with the publication rate of video frames to the topic—-30 frames per second. Concurrently, during each publication, the motor's pulse width is published to the */motion_topic*.

Additionally, an approximated 2^{nd} order model is built from damping ratio ξ and natural frequency ω_n calculated from the settling time and percentage overshoot according to the formulae $\Delta\% = 100 \cdot e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}}$ and $t_s = \frac{4.6}{\xi\omega_n}$. The simulated step response is plotted in green line shown in Fig. 4.4. The approximated transfer function of servo motor 6 is:

$$G(s) = \frac{34.03}{s^2 + 2.1s + 34.03},$$

where $\xi = 0.18$ and $\omega_n = 5.83$. The small damping ratio indicates that the system conserves energy during oscillations, implying that the motor's mechanical components, such as gears, exhibit little friction and energy absorption. This make it respond fast to change in position and velocity. However, the fast response is accompanied by drawbacks – overshooting and oscillations.

Chapter 5

Conclusion and Future Work

In this project, the task of remote control of a colour-tracking robotic arm has been achieved. First of all, inverse kinematics of the 4-DOF robotic arm has been derived and the command packages are transmitted to the board on the robot through UART so that individual motor can move to the given position within an interval of time.

Additionally, colour tracking has been achieved using OpenCV tools and three PID controllers are designed to align the object in the centre of the frame window both horizontally and vertically while maintaining a constant object-to-camera distance.

Furthermore, ROS2 framework has been used to communicate between different nodes and topics. In particular, the remote controller communicates with the local main tracker node via ROS2 network in the same LAN.

Finally, the step response of the robotic arm has been analysed and the transfer function of motor 6 has been derived using an approximated 2^{nd} order linear model.

A variety of algorithms and experiments have been left for the future due to lack of time. This report focuses on remote control of the robotic arm to track a coloured object. The following ideas will be tested in the future:

1. In terms of control, only PID feedback control is used currently. Predictive feed-forward control together with PID feedback can be used in the future as the main control method to effectively reject disturbances for fast disturbance dynamics or system with input-output delays [18].

2. Within this the context of this report, the classical transfer function is used to represent the system. The three PID controllers might lead to sub-optimal coordination between axes. If one controller reacts faster or slower than the other, it can result in overshooting or lagging in one direction, causing imprecise tracking.

   To solve this problem, modern state-space control can be used instead to more comprehensively represent and handle this multi-input multi-output (MIMO) system [19]. With state-space model, the system's dynamics can be represented as a set of state variables, such as [x_{position}, x_{velocity}, $x_{\text{acceleration}}$,...]. The full state feedback (pole placement) method can consider all of these states simultaneously when generating control outputs. However, more sensors are required to measure velocity and acceleration as state variables on each motor. Unfortunately, the limitations due to funding constraints have led to their exclusion from the scope of this current project.

3. Presently, a singular robotic arm is in operation. However, multiple robots can work together to achieve some well-defined goals, being managed via a singular remote controller. This multi-robot system is still in its infancy and requires several techniques, such as coordination and control, communication, localisation and mapping, etc [20]. The physical existence of all these robots is not imperative; they can be either simulated using tools like ROS Gazebo and Rviz.

Appendix A

Videos of experiments

Some simple experiments during the setup stage:

1. Single servo motor movement by sending commands package via UART
2. Move to different coordinates based on inverse kinematics
3. Same end-effector coordinate, different configuration
4. Color tracking

Experiments in the ROS2 framework:

1. 3D trajectory of the end-effector
2. Step response of the robotic arm
3. Unstable behaviour when the controller is moved to remote machine
4. Remote control of the robotic arm

Bibliography

[1] M. Shahbazi, S. F. Atashzar, and R. V. Patel, "A systematic review of multilateral teleoperation systems," *IEEE Transactions on Haptics*, vol. 11, no. 3, pp. 338–356, 2018.

[2] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109806002871

[3] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "Tactile-internet-based telesurgery system for healthcare 4.0: An architecture, research challenges, and future directions," *IEEE Network*, vol. 33, no. 6, pp. 22–29, 2019.

[4] D. Lee, W. K. Chung, and K. Kim, "Safety-oriented teleoperation framework for contact-rich tasks in hazardous workspaces," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4268–4275.

[5] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019.

[6] M. Risiglione, J.-P. Sleiman, M. V. Minniti, B. Çizmeci, D. Dresscher, and M. Hutter, "Passivity-based control for haptic teleoperation of a legged manipulator in presence of time-delays," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5276–5281.

[7] R. K. Megalingam, A. Sahajan, A. Rajendraprasad, S. K. Manoharan, and C. P. K. Reddy, "Ros based six-dof robotic arm control through can bus interface," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2021, pp. 739–744.

[8] M. Zhu, X. Lv, S. Zuo, and X. Zhang, "The design of robotic arm based on ros," in *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*, 2023, pp. 203–207.

[9] S. M. Spong, M.W.; Hutchinson, "Robot modeling and control," vol. 33, no. 5, 2006. [Online]. Available: https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/How_to_avoid_singular_configurations/attachment/59d6361b79197b807799389a/AS%3A386996594855942%401469278586939/download/Spong+-+Robot+modeling+and+Control.pdf

[10] B. Balon and M. Simić, "Using raspberry pi computers in education," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 671–676.

[11] P. Sharma, A. Kumar, and N. Kumar, "Analysis of uart communication protocol," in *2022 International Conference on Edge Computing and Applications (ICECAA)*, 2022, pp. 323–328.

[12] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ros2," in *2016 International Conference on Embedded Software (EMSOFT)*, 2016, pp. 1–10.

[13] K. Krinkin, A. Filatov, A. Filatov, O. Kurishev, and A. Lyanguzov, "Data distribution services performance evaluation framework," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, 2018, pp. 94–100.

[14] H. Yuefeng, "Study on data transmission of dcps publish-subscribe model," in *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, 2018, pp. 1–2172.

[15] M. S. Mahmoud, "Approaches to remote control systems," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 4607–4612.

[16] M. Jackson, "Specialising in software engineering," in *14th Asia-Pacific Software Engineering Conference (APSEC'07)*, 2007, pp. 3–10.

[17] X.-M. Zhang, Q.-L. Han, and X. Yu, "Survey on recent advances in networked control systems," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1740–1752, 2016.

[18] X. Li, S. Liu, K. K. Tan, Q.-G. Wang, and W.-J. Cai, "Predictive feedforward control," in *2016 12th IEEE International Conference on Control and Automation (ICCA)*, 2016, pp. 804–809.

[19] C. Unsalan, D. E. Barkana, and H. D. Gurhan, *State-space Based Control System Analysis*, 2021, pp. 227–246.

[20] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, 2012, pp. 1–5.