



المعهد الوطني للعلوم التطبيقية والتكنولوجيا  
National Institute of Applied Sciences and Technology

## Personal Professional Project

Degree : Industrial Computing and Automation

First-year Engineering

---

# Remote Control System

---

*Prepared by :*

M. BELHAJ AMOR Amine

M. BOUAFIF Rayen

M. BOUCHRIHA Issmail

*Supervised by :*

M. HADDAD Mohamed

*Defended on the 3rd of June, 2024*

Academic year : 2023/2024

# ACKNOWLEDGEMENT

---

Before any development on this professional experience, we would like to express our deepest gratitude and respect to all those who contributed to the successful completion of this work. First of all, we extend our thanks to our supervisor, Mr. **HADDAD Mohamed**, for granting us the opportunity to undertake this project, and for his guidance throughout the entire process. Additionally, we are also grateful to Mme **AYEB Amani** for her sincere help and advice.

We also wish to extend our appreciation to our colleagues and especially the community of **Aerobotix INSAT Club** , whose support helped us overcome the material challenges that we've encountered. Finally, we would like to express our respect and gratitude to the members of the jury for agreeing to evaluate this modest work. Their insights and feedback will undoubtedly contribute to our growth and the refinement of our skills.

# Contents

ACKNOWLEDGEMENT . . . . .	i
List of Figures . . . . .	iv
List of Acronyms . . . . .	v
<b>1 Introduction</b>	<b>1</b>
<b>2 Solution Overview</b>	<b>2</b>
2.1 System Overview . . . . .	2
2.1.1 Study case . . . . .	2
2.1.2 System modeling and regulation . . . . .	3
2.2 Proposed Solution (Graphical User Interface) . . . . .	4
2.3 Conclusion . . . . .	5
<b>3 User Interface</b>	<b>6</b>
3.1 Control Panel . . . . .	6
3.1.1 Tuning section . . . . .	7
3.1.2 "Go" section . . . . .	8
3.1.3 "Robot locate at" Section . . . . .	8
3.2 Plot panel . . . . .	9
3.3 Conclusion . . . . .	9
<b>4 Communication and Connectivity</b>	<b>10</b>
4.1 Implemented Protocol . . . . .	10
4.1.1 MQTT Protocol . . . . .	10
4.1.2 How does MQTT work ? . . . . .	10
4.1.3 MQTT in Local Area Network (LAN) . . . . .	11
4.1.4 MQTT in Wide Area Network (WAN) . . . . .	12

4.2	Communication Protocol Implementation . . . . .	12
4.2.1	Implementing MQTT on LAN . . . . .	13
4.2.2	Implementing MQTT on WAN . . . . .	14
<b>5</b>	<b>Experimental Results</b>	<b>16</b>
5.1	Launching . . . . .	16
5.2	Control And Tuning : . . . . .	17
5.3	Conclusion . . . . .	19
<b>6</b>	<b>General Conclusion</b>	<b>20</b>
	<b>References</b>	<b>20</b>

# List of Figures

2.1	Study Case : Differential wheeled robot . . . . .	2
2.2	DC Motor Modeling . . . . .	3
2.3	Nested loop controller for 2 DC Motors . . . . .	4
2.4	GUI overview . . . . .	4
3.1	GUI Main sections . . . . .	7
3.2	GUI "Go" section . . . . .	8
3.3	GUI "Robot locate at" section . . . . .	8
3.4	Plot Panel . . . . .	9
4.1	MQTT Architecture . . . . .	11
4.2	Mosquito Download QR code . . . . .	13
4.3	Port forwarding . . . . .	14
4.4	Dynamic Domain Name System . . . . .	15
5.1	Launching the program . . . . .	16
5.2	Setting initial coefficients . . . . .	17
5.3	Applying Go command . . . . .	17
5.4	Before tuning . . . . .	18
5.5	Adjusting coefficients . . . . .	18
5.6	After tuning . . . . .	19

# List of Acronyms

- **GUI** : Graphical User Interface
- **MQTT** : Message Queuing Telemetry Transport
- **LAN** : Local Area Network
- **WAN** : Wide Area Network
- **IP** : Internet Protocol
- **ISP** : Internet Service Provider
- **DDNS** : Dynamic Domain Name System
- **PD** : Proportional-Derivative
- **PI** : Proportional-Integral
- **Wi-Fi**: Wireless fidelity
- **AWS** : Amazon Web Services
- **IBM** : International Business Machines Corporation
- **IoT** : Internet Of Things

# Chapter 1

## Introduction

---

In today's industrial environment, the integration of remote control systems has become increasingly crucial for enhancing operational flexibility. As we transition into the era of Industry 4.0, the demand for remote monitoring in manufacturing is growing significantly. its global market size is anticipated to grow **from USD 24.6 billion in 2022 to USD 32.3 billion by 2027** at a compounded annual growth rate of **5.6%** According to the organization MarketSandMarkets.

To better understand the reasons behind this growth, let's delve into these scenarios. Manufacturers operating in hazardous environments often struggle with traditional on-site supervision, making remote control solutions essential. Or four years ago, when the COVID pandemic highlighted the importance of flexible and remote systems. We all remember that period, when the industrial sector in Tunisia experienced **nearly 9% decrease in global production** according to the National Institute of Statistics.

In this report, we will first provide an overview of our proposed solution: an interface that enables remote control while allowing the operator to obtain real-time feedback. We'll be applying it to a chosen system. Next, we will delve into the details of our interface, examining the communication technology used behind the scenes and finally, testing the interface and observe the results.

# Chapter 2

## Solution Overview

---

### 2.1 System Overview

#### 2.1.1 Study case

In this section, we will be representing the system that we are going to control remotely. As a case study, we used a differential wheeled robot equipped with two 5V DC motors, each with an integrated encoder as it's represented in (Figure 2.1)



Figure 2.1: Study Case : Differential wheeled robot

The robot is controlled by an **ESP-32 microcontroller** with an integrated Wi-Fi module. This module is essential for enabling wireless communication. Our interface is designed to control the velocity and position of the robot without any geographical constraints.

It is important to note that the interface is highly customizable and can be adapted to various other industrial systems, whether they are fast or slow. For instance, it can be used for applications such as level regulation, temperature control, and more.



### 2.1.2 System modeling and regulation

To have control over a specific system, we must first understand its dynamics and behavior. This involves several key steps, including system modeling and controller design.

**Modeling :** The first step in controlling a system is to develop a mathematical model that accurately represents its dynamics. This model can be derived from first principles, such as the laws of physics.

For our system, we will be using a simplified DC motor model, which can be represented by the following block diagram: (Figure 2.2)

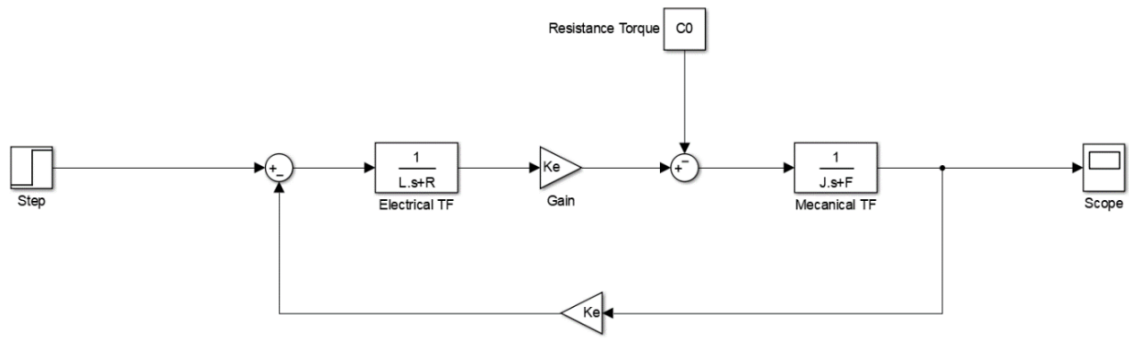


Figure 2.2: DC Motor Modeling

$$\text{Transfert Function of DC Motor : } \frac{\Omega}{U} = \frac{K_e}{L.j.s^2 + (R + j).s + (R.f + K_e^2)}$$

**Controller design :** With an accurate model in hand, the next step is to design a controller that can regulate the system's behavior to achieve desired performance objectives. In our case, we will be implementing a cascade controller (also known as a nested loop controller). we will be feeding the position error to the position regulator (The Master). The output of the position regulator represents a speed command to the velocity controller (The Slave).

As shown below in (Figure 2.3), this is the architecture of the implemented nested loop controller for the 2 DC Motors.

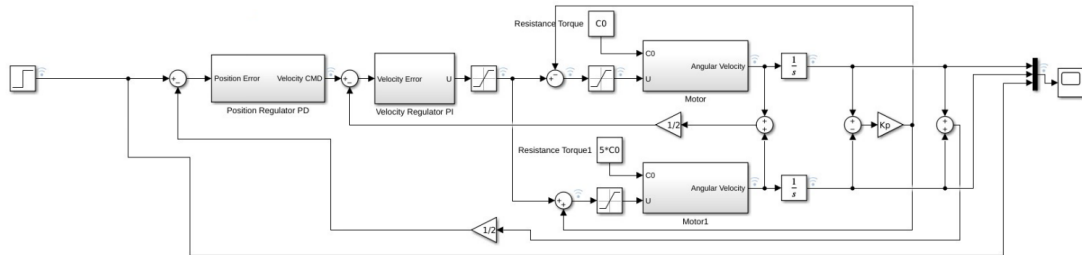


Figure 2.3: Nested loop controller for 2 DC Motors

After designing the controller, the next step is to implement it on our microcontroller as a **C code**, the ESP32 Microcontrollers are powerful devices that serve as both communication bridges and digital regulators in our system.

## 2.2 Proposed Solution (Graphical User Interface)

Our solution is a developed Graphical User Interface (GUI) that comprises two distinct windows designed for control and monitoring (As indicated in Figure 2.4). The first window, the Control Panel, provides an intuitive interface to control the system. The second window serves as a real-time plotting interface. This dual-window approach enhances user interaction by allowing seamless parameter adjustments while simultaneously observing the system's performance.



Figure 2.4: GUI overview

## 2.3 Conclusion

This section provided an overview of our proposed solution: a user interface designed to adapt to our differential wheeled robot (or any controlled system, such as a level or temperature control system) **remotely**. We are using a nested loop controller for both DC motors to regulate both position and velocity. .

**Note that the interface is highly customizable, allowing it to be tailored to control any system we want.**

# Chapter 3

## User Interface

---

As outlined in the solution overview, we have developed a user interface to enhance communication between the operator and the controlled machine. This interface represents the optimal solution for the challenge we introduced. Moving forward, we will provide a detailed exploration of the GUI and integration of this interface into our control system.

The Interface is composed of 2 Panels : **Control Panel and Plot Panel**

### 3.1 Control Panel

For our control panel , We opted for PyQt, a Python framework for GUIs, to develop a user-friendly interface. PyQt's robust features and wide documentations streamlined our design process.

Our control panel offers intuitive control and precise tuning capabilities for our system, featuring three distinct sections: **'Go', 'Robot Locate', and 'Tune'**. (As shown in Figure 3.1). Each section is designed to streamline a different operation and enhance user experience.

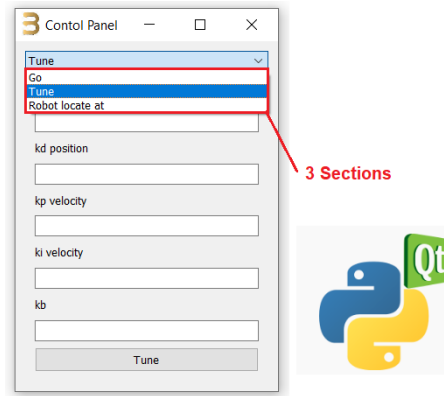
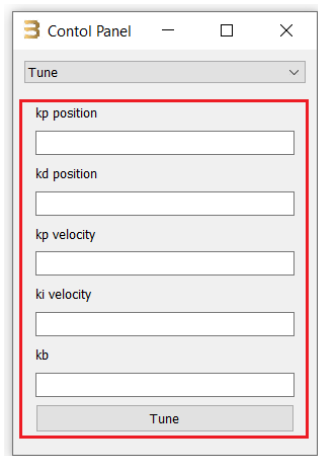


Figure 3.1: GUI Main sections

### 3.1.1 Tuning section

This section is dedicated to tuning our system, where you can adjust various regulation coefficients to fine-tune and optimize the system's performance, ensuring precise control and efficient operations. For our study case, we have 2 Controllers (Master as PD controller and Slave as PI Controller). So in total we can adjust 5 parameters : (We have incorporated a balancing parameter to overcome any asymmetrical movements of the robot)



- **Kp position** : Proportional Position coefficient
- **Kd position** : Derivative Position coefficient
- **Kp velocity** : Proportional velocity coefficient
- **Kd velocity** : Derivative velocity coefficient
- **Kb** : Balance coefficient

### 3.1.2 "Go" section

the 'Go' section of the GUI (Shown in Figure 3.2) is designed to allow the user to control the movement of the robot and its rotation by specifying the distance that the robot should move or its angle.

In other systems, this section can be used to choose the system input, such as the desired temperature or desired level, depending on the system's function.

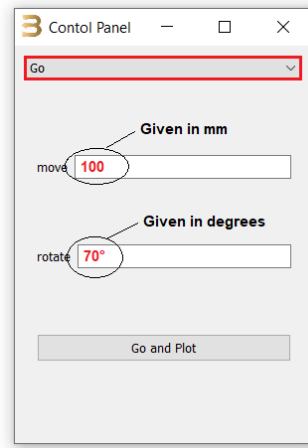


Figure 3.2: GUI "Go" section

### 3.1.3 "Robot locate at" Section

instead of providing the robot with a distance, the operator can specify coordinates for the robot to reach through the **Robot locate at** Section (Figure 3.3). The coordinates are relative to the robot's initial position. We implemented this section to demonstrate our ability to incorporate various control methods into the GUI.

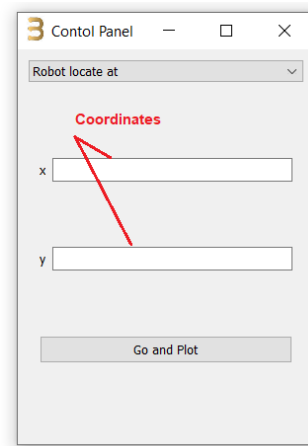


Figure 3.3: GUI "Robot locate at" section

## 3.2 Plot panel

The Plot Panel (See Figure 2.3 below) is developed using **Matplotlib**. The panel visually displays the commands issued to the system alongside the system's real-time responses, providing an immediate and clear understanding of system behavior.

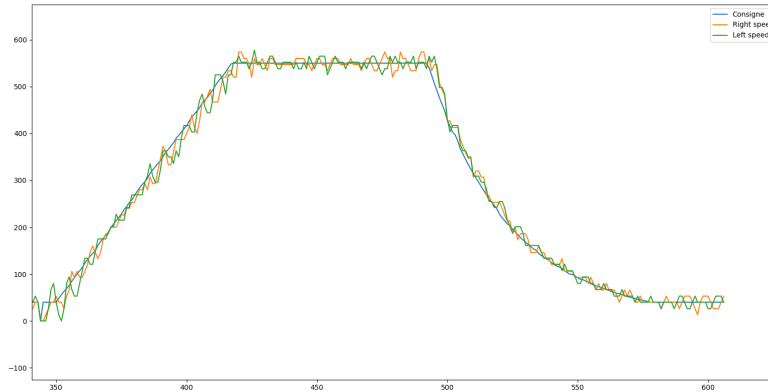


Figure 3.4: Plot Panel

## 3.3 Conclusion

We have successfully developed a user-friendly interface that facilitates the control of our system composed of a **Control Panel** that contains 3 main sections, and a **Plot Panel** where we can see the behavior of our system in real time. However, what we haven't discussed is the underlying mechanism. When the 'Go and Plot' button is pressed, how do we establish communication between our system and our GUI? Let's delve into the details of this interaction in the next section.

# Chapter 4

## Communication and Connectivity

---

### 4.1 Implemented Protocol

This section provides an overview of the communication framework employed in our remote system control interface, focusing on the **MQTT protocol**.

#### 4.1.1 MQTT Protocol

**MQTT (Message Queuing Telemetry Transport)** is a lightweight messaging protocol designed for connecting IoT devices. MQTT is known for its efficiency, low bandwidth usage, and support for unreliable networks.

It's widely used in Industry4.0 applications for its simplicity and reliability.

#### 4.1.2 How does MQTT work ?

The protocol follows a **publish-subscribe model**, where clients (devices) communicate through a central broker (Server). Devices can publish messages to topics, and other devices subscribe to those topics to receive messages.

**In our study case**, the industrial system subscribes to "**systemCommand**" topic of the MQTT to obtain regulation coefficients and commands. Conversely, it publishes the commands it executes and the corresponding system responses in real-time through the "**dataReadings**" topic. This bidirectional communication flow ensures prompt action and feedback. (Refer to Figure 4.1 for clarification)



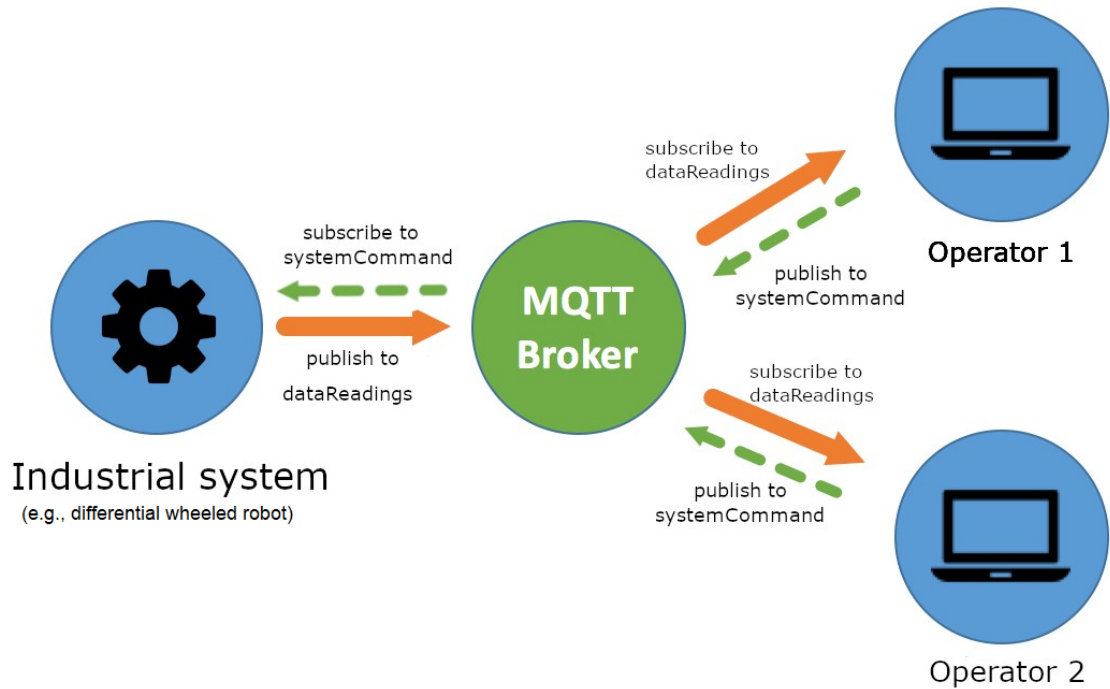


Figure 4.1: MQTT Architecture

**MQTT Broker:** it's the server that receives all messages from the clients and then routes those messages to the appropriate destination clients. The broker plays a crucial role in the publish/subscribe messaging model by managing the connections and routing the informations..

**MQTT Client:** An MQTT client is any device (such as a sensor, mobile device, or computer) that interacts with the MQTT broker. As mentioned before, they can publish or subscribe to topics.

### 4.1.3 MQTT in Local Area Network (LAN)

In this case, The MQTT broker is typically installed on a server or a computer within the LAN, making it accessible to all devices connected to the same local network (a server can be a simple computer).

The LAN offers low latency and a quick response , but it has a limited range , meaning that it can't reply to the needs of our project.

#### 4.1.4 MQTT in Wide Area Network (WAN)

To address the limitations of the LAN, we can leverage a wide area network (WAN) setup. This approach allows devices to communicate over larger distances.

To enable MQTT communication over a WAN, we have two primary options: hosting the MQTT broker on a cloud or using the global IP address of the server with port forwarding.

- **MQTT Cloud Broker:** as a cloud-hosted service, it offers scalable and reliable infrastructure for facilitating MQTT communication between devices. By leveraging cloud resources, these brokers can easily handle large-scale deployments and ensure high availability. They provide built-in security features and global reach, allowing devices to connect to the nearest broker for optimal performance. However, it's also not free, which may make it impossible for our projects. We can cite examples such as: HiveMQ , Amazon Web Services (AWS) , IBM Watson IoT Platform
- **Using global IP address with port forwarding:** In this setup, we configure the existing local server to make it accessible over the internet using its global IP address. Port forwarding is necessary to route incoming MQTT traffic from the global IP address to the local server hosting the MQTT broker. For our project, we have adopted this method because it's cost-effective.

## 4.2 Communication Protocol Implementation

As discussed in the previous section, the MQTT protocol can be implemented using two types of networks, depending on the operator's requirements and the desired scope. In this section, we will explore the implementation process and how to achieve reliable communication.

### 4.2.1 Implementing MQTT on LAN

We can start an MQTT broker on a server by first downloading the **Mosquitto package**, which is available for free online. (Scan the QR in figure 4.2 to download)



Figure 4.2: Mosquito Download QR code

Let's Suppose that the server (or central PC) has the local IP address: **192.168.1.19**. To start the Mosquitto broker, we use the next bash command:

```
$sudo systemctl start mosquitto
```

Now any terminal or device (Client) connected to the same LAN and with MQTT installed can publish or subscribe to a topic by specifying the local IP address of the server hosting the MQTT broker (in our case 192.168.1.19)

**Publishing:** A terminal can publish a message to a topic using the `mosquitto_pub` command.

```
$mosquitto_pub -h 192.168.1.19 -t systemCommand -m "300,100"
```

Where:

- `-h 192.168.1.19`: Specifies the IP address of the MQTT broker.
- `-t systemCommand`: Specifies the topic to which the message is published.
- `-m "300,0"`: Specifies the message to be published.

**Subscribing:** A terminal can subscribe to a specific topic using the `mosquitto_sub` command.

```
$mosquitto_sub -h 192.168.1.19 -t dataReadings
```

Where:

- `-h 192.168.1.19`: Specifies the IP address of the MQTT broker.
- `-t dataReadings`: Specifies the topic to which we want to subscribe.

**Note:** We could connect all devices via LAN, but since it doesn't fully meet our requirements, we need an wider solution.

## 4.2.2 Implementing MQTT on WAN

We've chosen to implement this method in our project because our primary goal is to achieve complete control from any location worldwide. However, several steps must be completed before we can accomplish this.

- **Setting up Port forwarding:** Port forwarding is a networking technique used to redirect communication requests from one address and port number combination to another. In our case Port forwarding allows routing of commands from the global IP address of the network to the internal IP address of the MQTT broker server, let's suppose that the global IP address of our Network is 41.230.166.88 . Figure 4.3 below demonstrate the principle of the port forwarding functionality.



Figure 4.3: Port forwarding

In this way, any device worldwide with MQTT installed can publish or subscribe to a topic via the MQTT broker by simply providing the global IP address of the MQTT broker server:

```
$mosquitto_pub -h 40.230.166.88 -t systemCommand -m "300,100"
```

A major problem encountered with WAN is the frequent change of the global IP address assigned by the Internet Service Provider (ISP). This dynamic IP address can change periodically, making it difficult for external devices to consistently locate and connect to the MQTT broker.

Two proposed solutions can be implemented: Static IP address or Dynamic DNS.

- **Static IP** : Purchasing a static IP address from the Internet Service Provider (ISP) ensures that the global IP address remains constant. Although this solution is straightforward and effective, it often comes at an additional cost, as ISPs typically charge for static IP addresses.
- **Dynamic DNS** : A more cost-effective approach, DDNS assigns a domain name to the global IP address and automatically updates this mapping whenever the IP address changes. By continuously tracking the IP, the DDNS service ensures that the domain name always points to the current IP address. Let's suppose that we configured DDNS under domain name "mqttbroker.ddns.net" and the current IP address is 40.230.166.88, after certain time the IP address changed to 237.158.18.36 for one reason or another. The domain name always points to the updated IP address (refer to figure 4.4).

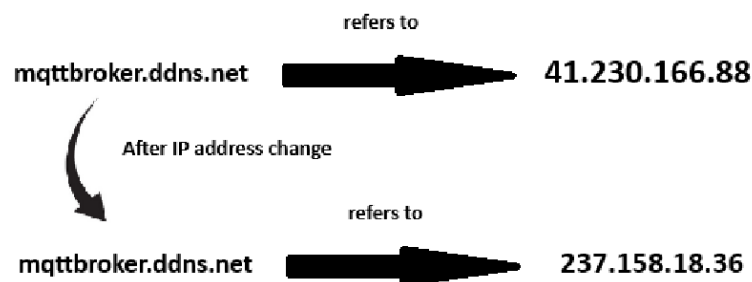


Figure 4.4: Dynamic Domain Name System

Now any device worldwide with MQTT installed can publish or subscribe to a topic via the MQTT broker at **any time** by simply providing the Domain Name of the MQTT broker server:

```
$mosquitto_pub -h mqttbroker.ddns.net -t systemCommand -m "300,100"
```

# Chapter 5

## Experimental Results

---

### 5.1 Launching

In this section, we will apply everything we have covered in the previous chapters. We will go through the entire process, from launching our interface to plotting the results.

We will first launch our program with the following bash command : (Figure 5.1)

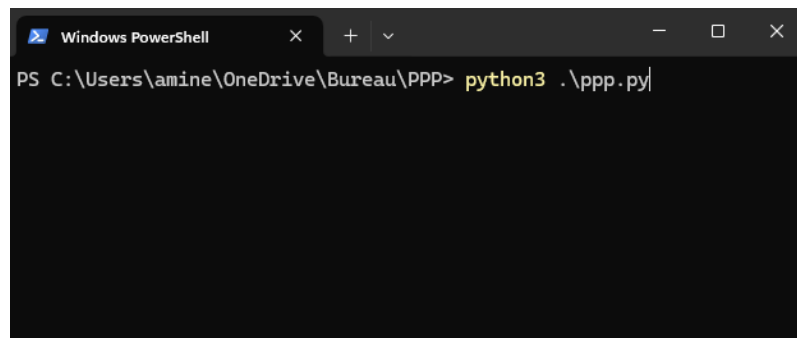
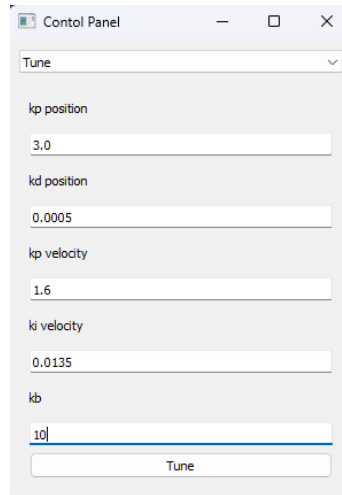


Figure 5.1: Launching the program

Once communication is successfully established, the control panel will appear, allowing users to interact with and control the system, Otherwise, if the communication fails or a timeout occurs, an error message will be displayed, prompting the user to check the connection and try again.

## 5.2 Control And Tuning :

First of all, we will set the coefficients of our system by selecting the **Tune section** as it's shown in Figure 5.2 .We will begin with a theoretical approach to determine the initial values of these coefficients.



Control Panel

Tune

kp position  
3.0

kd position  
0.0005

kp velocity  
1.6

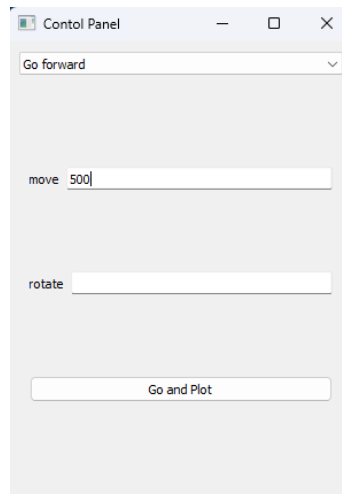
ki velocity  
0.0135

kb  
10

Tune

Figure 5.2: Setting initial coefficients

Secondly, we will command our robot to move forward for 50 cm. By selecting the **Go section** (Figure 5.3) and specifying the desired distance.



Control Panel

Go forward

move 500

rotate

Go and Plot

Figure 5.3: Applying Go command

Once we press **Go and Plot**, the Plot Panel will appear, displaying the real-time response of the robot (Refers to Figure 5.4).

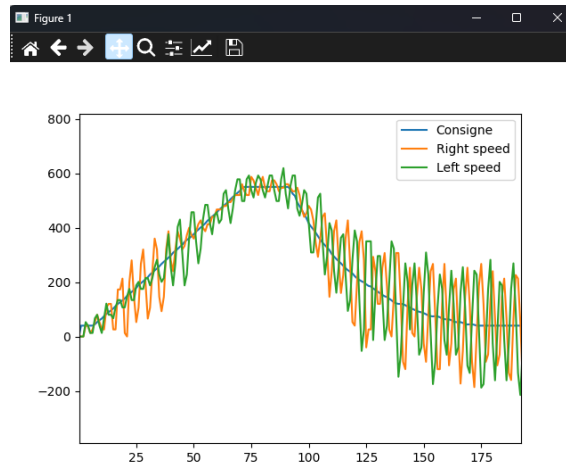


Figure 5.4: Before tuning

We are satisfied with the velocity command given output from the position regulator (The blue plot), as it does not exhibit any instability or static error. However, we have observed some instabilities in the velocity of both motors (Green and orange plots) when following the trajectory given by the previous regulator. We can achieve better results by reducing the velocity proportional coefficient (Figure 5.5).

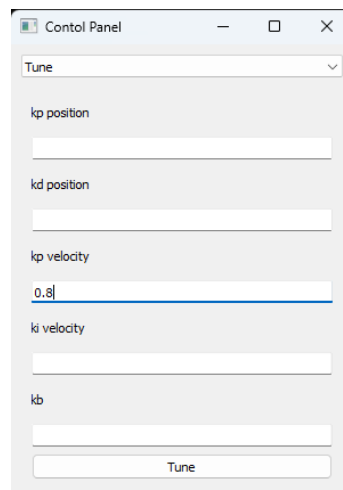


Figure 5.5: Adjusting coefficients



Let's issue the same command as before (Figure 5.3) and observe the Plot Panel once again (Figure 5.6) :

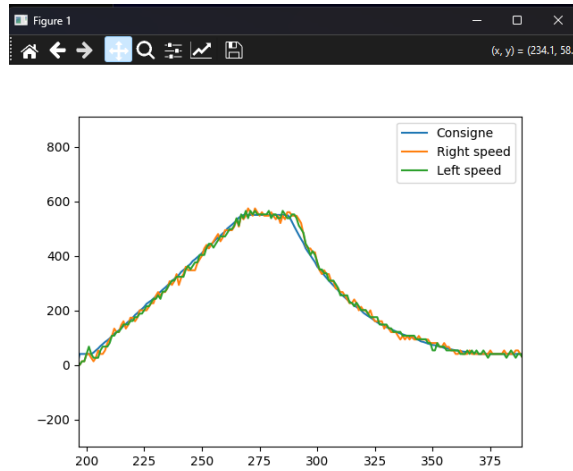


Figure 5.6: After tuning

## 5.3 Conclusion

By adjusting the proportional velocity coefficient, we achieved a significantly more stable response. This adjustment demonstrates the successful establishment of communication between the interface and our system, streamlining the process for easy and effective control.

## Chapter 6

### General Conclusion

---

In a nutshell, our interface provides a significant solution to the challenges that we mentioned in the introduction . With its simple and user-friendly GUI developed by PyQt5, we empower operators to adjust and control their systems effectively **From anywhere !**. As a case study, we focused on the differential wheeled robot with nested loop controller and developed an interface dedicated to its needs. Noting that our solution is highly customizable, making it adaptable to other systems after studying it. Moreover, by using MQTT, our messaging protocol, our interface ensures efficient and reliable communication between both ends (The operator through the interface and the system) enhancing the performance and flexibility of the system.



# Bibliography

- [1] Remote Monitoring and Control Market Industry and Region - Global Forecast to 2027 by **MarketsAndMarkets** platform
- [2] Institut national de la statistique , Indice annuel de la production industrielle (Statistique 138)
- [3] Amazon Web Services, Inc. Article **What is MQTT ?**
- [4] Amazon Web Services, Inc. Article **Qu'est-ce qu'un DDNS ?**
- [5] Domene.si Article **What is Path forwarding ?**
- [6] PhD.Eng Ahmed Braham Regulation Courses
- [7] PyQt5 Documentation
- [8] Matplotlib Documentation
- [9] MQTT Paho C++ and Python Documentation



## Abstract

BBB Solutions provides a robust solution to overcome the geographical constraints often encountered by operators in the industrial field. By developing a user-friendly and reliable interface, we have established fast and secure communication between both ends, enabling operators to maintain full control and visibility of the system from anywhere in the world, at any time. To demonstrate the effectiveness of our project, we chose to control a simple differential wheeled robot.

## Résumé

BBB Solutions offre une solution robuste pour surmonter les contraintes géographiques souvent rencontrées par les opérateurs dans le domaine industriel. En développant une interface conviviale et fiable, nous avons établi une communication rapide et sécurisée entre les deux extrémités, ce qui permet aux opérateurs de garder le contrôle total et l'observabilité du système de n'importe où dans le monde et à n'importe quel moment. Pour démontrer l'efficacité de notre projet, nous avons choisi de contrôler un simple robot différentiel à 2 roues.

## ملخص

نقدم لكم حلاً فعالاً لتجاوز القيود الجغرافية التي يواجهها المشغلون في المجال الصناعي. من خلال تطوير واجهة سهلة الاستخدام وموثوقة، قمنا بإنشاء اتصال سريع وآمن بين الطرفين، مما يتيح للمشغلين الحفاظ على السيطرة الكاملة و متابعة النظام من أي مكان في العالم، و في أي وقت. لتوضيح فعالية مشروعنا، اخترنا التحكم في روبوت بسيط متحرك ذات عجلتين.