POLITECNICO DI TORINO

Master of Science in Mechatronics Engineering

Master of Science thesis

Energy-Aware Adaptive Communication in ROS2-Based Robotic Applications

Supervisor
Prof. Mihai Teodor Lazarescu
Co-supervisor
PhD. Navvab Kashiri

Candidate
Giorgio Vivoli

Academic Year 2024/2025

Summary

In robotic systems, wireless communication is essential for transmitting sensor data, commands, and status information. However, maintaining high communication performance often comes at the cost of increased energy consumption, particularly critical in mobile robots operating on limited power budgets. This thesis presents a modular and flexible adaptive communication framework for robotic applications, based on ROS2 messages, designed to minimise energy consumption while preserving communication reliability. The algorithm uses RSSI (Received Signal Strength Indicator) as a feedback metric to estimate link quality. Based on this estimation, the framework dynamically computes the optimal combination of transmission power and data rate, minimising energy consumption according to the cost associated with each transmission power level, message size, and estimated link quality. In addition, the framework integrates a message prioritisation system that selectively reduces the frequency of lower-priority messages when link quality degrades. This mechanism helps to decrease the effective data rate, thus improving transmission robustness while maintaining essential communication. Designed to be used with ROS2 messages, the framework is suitable for a wide range of robotic applications. Simulations and experimental results demonstrate that the proposed system can reduce communication energy consumption, contributing to more sustainable and intelligent robotic systems.

Contents

Chapter 1

State of the Art

1.1 Adaptive and Feedback-Driven Communication Strategies

1.1.1 Reactive and Predictive Adaptation Approaches

Modern mobile robotic systems require communication strategies that not only respond to changes in connection conditions, but are also able to anticipate them. There are two main types of adaptive communication: reactive and predictive. Both use link quality indicators (LQIs) such as RSSI, SNR and LQI or other types of metrics such as PRR(Packet Received Ratio) or PLR(Packet Loss Ratio) as inputs of the control algorithm.

- Reactive approaches respond after link degradation.
- Predictive approaches anticipate link degradation using models.

Reactive Approach

Reactive adaptation is the most direct form of communication adjustment. Systems monitor metrics that directly measure the link degradation, such as PRR or PLR, in real-time and adjusts the parameters following some rules. For example, in [12] Atsushi Onoe Algorithm and Adaptive Multi-Rate Retry (AMRR) algorithm perform a data rate adaption using Packet loss ratio as input, Automatic Rate Fall back (ARF) algorithm just increase to the next higher data rate after 10 consecutive successful transmission, while decrease to the next lower data rate after 2 consecutive losses. However, the reactive approach implies that degradation must occur before the control adjusts the parameters, potentially leading to brief interruptions.

Predictive Approach

Predictive strategies aim to prevent communication issues by anticipating degradation before it happens. This is typically achieved through statistical models, machine learning, or mobility prediction. Predictive systems use LQIs and models to adjust communication parameters before the link's degradation occurs. In [3] a predictive techniques that use historical RSSI time-series to predict connectivity loss is presented. By integrating prediction with control loops, robots can increase transmission power before the signal fading. While predictive strategies offer smoother performance and fewer disruptions, they are model-dependent or require sufficient training or historical data. In fast-changing scenarios or previously unseen environments, prediction models may not perform well.

Summary

- Reactive methods are simple, fast, and suitable for unknown or rapidly changing conditions but may suffer from delayed response.

- Predictive strategies improve stability and efficiency, especially in patterned environments, but rely on accurate models.

1.1.2 Transmission Power and Data Rate Control in Wireless Sensor Networks

In mobile robots, the management of wireless communication is a significant factor in improving energy efficiency, the reliability of signals, and the entire network performance. To address this, researchers have begun to explore adaptive algorithms that adapt transmission power and data rates, in order to optimize energy consumption and signal quality. A large number of these techniques were initially investigated in the field of Wireless Sensor Networks (WSNs), where static nodes were expected to communicate with each other in an efficient manner.

Transmission Power Control (TPC)

In [20] they address the problem of energy optimization in the field of wireless body area networks (WBANs) design. These devices, due to their small size and limited battery capacity, need to efficiently use energy, without waste, making power control a key design concern, especially in dynamic environments.

The paper introduce a novel **adaptive power control (APC) algorithm** that dynamically adjust the transmission power **(TP)** of sensor nodes based on real-time feedback from a central base station (BS), using RSSI values as the primary channel quality metric. Unlike conventional transmission power control (TPC) approaches, which often rely on static or slowly adaptive strategies and fixed thresholds, the proposed APC scheme is fully adaptive to short-term fluctuations in the wireless channel that arise from changes in body posture or sensor displacement during movement.

The APC algorithm incorporates eight core parameters, including the latest and lowest observed RSSI samples, the weighted average $\overline{R}$ of latest and lowest RSSI samples (which accounts for both good and bad channel conditions using separate weights α_1 and α_2), and a global value of fixed lower threshold of RSSI.

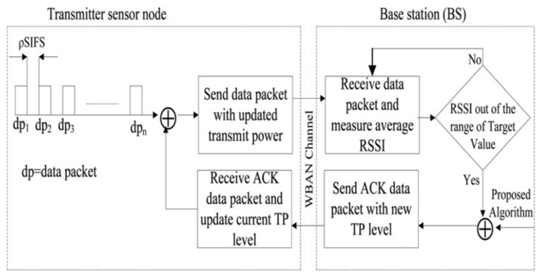

Transmitter sensor node
Base station (BS)
ρSIFS
dp₁ dp₂ dp₃
dpₙ
dp=data packet
Send data packet with updated transmit power
Receive ACK data packet and update current TP level
WBAN Channel
Receive data packet and measure average RSSI
RSSI out of the range of Target Value
No
Yes
Proposed Algorithm
Send ACK data packet with new TP level

Figure 1.1: APC algorithm scheme

The base station computes the $\overline{R}$ of the RSSI samples with exponential weighted averaging before making any power control decisions. The transmitter sensor node receive the RSSI value for each transmitted packet via feedback from the base station and chooses an appropriate transmission power level for subsequent packet transmission. The transmission power and the corresponding RSSI, both in dBm for the latest and lowest RSSI samples, are P_t, R_{latest} and $P_t - \Delta P_i$, $R_{\text{latest}} - 1$, respectively, where $i = 1, 2, \ldots, N$ shows the i_{th} transmission power level. After receiving the RSSI sample, the base station updates the $\overline{R}$, according to 1.1 and 1.2 for good and bad channels, respectively

$$\overline{R} = R_{\text{latest}} + (1 - \alpha_1) \times R_{\text{lowest}} \tag{1.1}$$

$$\overline{R} = R_{\text{latest}} + (1 - \alpha_2) \times R_{\text{lowest}} \tag{1.2}$$

The base station compares the value of $\overline{R}$ with the known RSSI target, and then decides the transmission power levels by using 1.3

$$\begin{aligned} &\Delta P = \arg\left\{\Delta P_1,\ \Delta P_2^{\min},\ \ldots,\ \Delta P_N\left(\sqrt{\left(R_{\text{target}} - \overline{R} - \Delta P_i\right)^2}\right)\right\} \\ &s.t.\quad \Delta P_i > R_{\text{target}} - \overline{R} \end{aligned} \tag{1.3}$$

where N is the number of TP levels. The proposed algorithm increases or reduces TP by comparing $\overline{R}$ with TRH$_{\text{var}}$ and TRL. If $\overline{R}$ exceeds TRH$_{\text{var}}$, indicates that channel state is remarkably good, so the TP is reduced by on-demand to save energy. If on the other hand R falls below TRL, shows the bad channel state, so the TP is quickly increased by on the request to avoid packet loss. If $\overline{R}$ satisfies RSSI threshold range [TRL, TRH$_{\text{var}}$], by adapting TP level in an appropriate manner then TP will not change

$$\text{TRH}_{\text{var}} = \text{TRL} + \sigma \tag{1.4}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R_i - \overline{R})}, \quad i = 1, 2, ..., n \tag{1.5}$$

where σ is the standard deviation (in dBm) of n RSSI samples as in 1.5, R shows RSSI samples, $\overline{R}$ is the RSSI average and TRL is the fixed lower threshold.

Two experimental scenarios were simulated using real world walking motion datasets, “right wrist to right hip” and “chest to right hip”, representing typical use cases in health monitoring. The **Monte Carlo simulations in MATLAB**, based on realistic WBAN channel models, demonstrated that the proposed APC algorithm significantly outperform benchmark methods in terms of energy efficiency. Specifically, the algorithm achieved **energy savings of up to 40.9%** compared to conventional schemes such as Gao's and Xiao's TPC algorithms. Although the packet loss ratio (PLR) was slightly higher (4.47–4.50%) compared to constant TPC (2.55–2.75%).

In comparison, Gao's method employs a threshold-based adjustment strategy that lacks flexibility under dynamic channel conditions, while Xiao's multiplicative-increase/additive-decrease policy is computationally more complex and consumes more energy. The proposed APC method, in contrast, is **computationally lightweight**, incurs **minimal control packet overhead**, and achieves a better balance between **energy savings and communication reliability**.

Furthermore, the algorithm shows **improved RSSI stability**, reflected by a lower normalized standard deviation compared to all baseline methods (5.57 vs. 8.80 for constant TPC), indicating more consistent link quality. The authors also explored the trade-offs between energy consumption and PLR under different parameter settings (e.g. $\alpha_1 = 1.0$, $\alpha_2 = 0.4$), showing that careful tuning allows further optimization depending on application requirements.

In [8] the authors propose ATPC (Adaptive Transmission Power Control), a feedback-driven and lightweight algorithm that enables sensor nodes to dynamically adjust their transmission power to maintain reliable communication while minimizing energy consumption. ATPC aims to ensure reliable wireless communication (high Packet Reception Ratio), between nodes, while reducing transmission energy to the minimum necessary level. This is achieved by allowing each node to adapt its transmission power independently for each neighbour, based on local link quality feedback collected at runtime. Each node constructs a **predictive linear model** for every neighbour, approximating the relationship between **transmission power** and Received Signal Strength Indicator (RSSI) or Link Quality Indicator (LQI). This is based on empirical measurements.

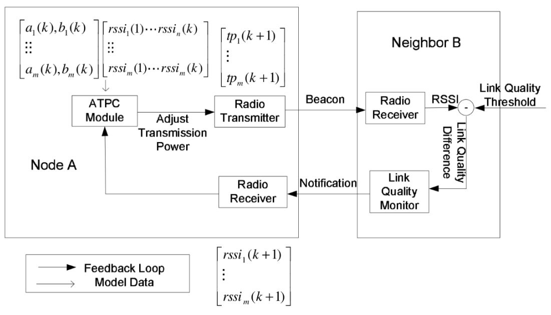


Figure 1.2: ATPC scheme

The ATPC algorithm operates in two phases: During the initialization phase, each node builds a predictive model that estimates the correlation between transmission power and received signal quality for each of its neighbours. This is done by having each node broadcast beacon packets at various transmission power levels. The nodes record the Received Signal Strength Indicator (RSSI) or Link Quality Indicator (LQI) for each received packet and return these measurements to the sender. The sender then uses these data points to compute a least-squares linear approximation of the form $RSSI(tp) = a \cdot tp \cdot b$, where tp is the transmission power level, and a and b are regression coefficients specific to that pair of nodes.

This linear model allow the node to estimate the minimum power level required to achieve a target link quality (e.g. an RSSI threshold of -90 dBm). While the initial model provides a good estimate under stable conditions, environmental dynamics need periodic updates. Therefore, the runtime phase of ATPC introduces a closed loop feedback mechanism to refine transmission power decisions over time.

While transmitting data the receiver keeps monitoring the received packets from its neighbours, which will be used to calculate the current link quality (RSSI or PRR). If the measured quality fall below a lower bound or rise above an upper bound the receiver sends the notify packet back to the sender, then the sender needs to do adjustments. The sender recalculates the new power level based on the updated RSSI and its prediction model.

The predictive model itself is adapted at runtime, the offset (b) is continuously recalculated using recent RSSI feedback to compensate environmental changes. This adjustment makes ATPC robust to long-term variations such as weather or foliage changes, without requiring constant retraining.

The ATPC algorithm was implemented and tested on MICAz motes across different environments (grass field, parking lot, indoor corridor) and conditions (fair and rainy weather). **ATPC reduce average transmission energy consumption to 53.6%** of that used in the maximum power configuration and **78.8%** compared to node-level static configurations, maintaining **PRR $>$ 98%** over 72 hours in a 43-node deployment despite environmental fluctuations.

Data Rate Adaption

In [12], a comparative simulation study of different data rate adaptation algorithms designed for IEEE 802.11 wireless networks is presented.

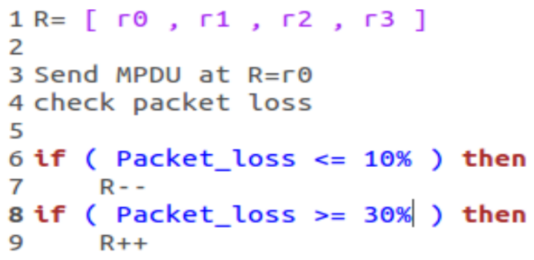
Automatic Rate Fall back (ARF)

The behaviour of ARF is very simple, the sender increases the data rate to the next higher value if a pre-defined number (10) of consecutive successful transmission is reached. In the same way, after a pre-defined number (2) of consecutive losses, it decrease the data rate to the next lower value.

Adaptive ARF (AARF)

AARF differs from ARF by using history of the channel, increasing the number of consecutive successful transmissions, required to increase the data rate, if the connection is good and the transmission works well with a fixed data rate. AARF is similar to ARF, but it increases the number of consecutive successful transmissions depending on the number of failed attempts to probe the the channel at a higher data rate. Every time the probe transmission fails, the number of consecutive successful transmission target is multiplied by two, with a maximum limit of 50. If a packet fails two consecutive times, it set the data rate to the next lower value and the consecutive successful transmission counter is reset to 10 like ARF.

Adaptive Multi-Rate Retry (AMRR)
In AMRR algorithm r3 is always set as the lowest data rate, while r1 is set to the next lower value and r2 to the second next lower value. If the packet loss is below 10%, and at least 10 packed were transmitted, the algorithm increase the data rate; otherwise, if the packet loss is higher then 33%, the algorithm decrease the data rate.



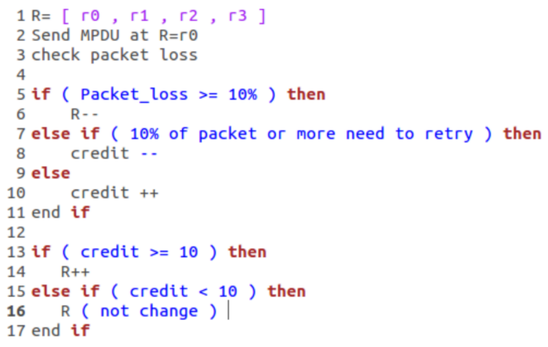
```
R= [ r0 , r1 , r2 , r3 ]

Send MPDU at R=r0
check packet loss

if ( Packet_loss <= 10% ) then
    R--
if ( Packet_loss >= 30% ) then
    R++
```

Figure 1.3: AMRR algorithm

Atsushi Onoe Algorithm
Onoe algorithm represents a variation of the AMRR (Adaptive Multi Rate Retry) strategy. Although it maintains the same rate fallback structure as AMRR for r1,r2,r3, ONOE introduces a credit-based mechanism. The algorithm starts with R=r0, credits are incremented when less than 10% of the packets, transmitted at the current data rate, require retransmission. Once the credit reaches a threshold of 10, the algorithm attempts to increase the transmission rate to the next higher available level and resets the credit to zero. In contrast, if more than 10% of packets need to be retransmitted during a given evaluation period, the rate is reduced to the next lower level, and the credit count is reset again. This credit accumulation is computed over relatively long time intervals, making ONOE less reactive to short-term fluctuations compared to algorithms like ARF and AMRR, which rely on more immediate feedback.



```
R= [ r0 , r1 , r2 , r3 ]
Send MPDU at R=r0
check packet loss

if ( Packet_loss >= 10% ) then
    R--
else if ( 10% of packet or more need to retry ) then
    credit --
else
    credit ++
end if

if ( credit >= 10 ) then
   R++
else if ( credit < 10 ) then
   R ( not change )
end if
```

Figure 1.4: Onoe algorithm

Data Frequency Adaptation

For example, in [16], they introduced an **adaptive data transmission algorithm** designed to improve the performance and efficiency of **wearable inertial sensor systems** used for hand movement acquisition. Their system includes **16 MEMS inertial sensors**, each combining an accelerometer, gyroscope, and magnetometer, placed to capture finger and hand motion.

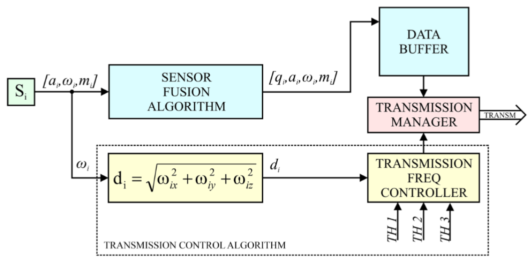

S_i
$[a_i, \omega_i, m_i]$
SENSOR FUSION ALGORITHM
$[q_i, a_i, \omega_i, m_i]$
DATA BUFFER
TRANSMISSION MANAGER
TRANSM
ω_i
$d_i = \sqrt{\omega_{ix}^2 + \omega_{iy}^2 + \omega_{iz}^2}$
d_i
TRANSMISSION FREQ CONTROLLER
TH 1
TH 2
TH 3
TRANSMISSION CONTROL ALGORITHM

Figure 1.5: Data transmission frequency control scheme

The algorithm in Figure 1.5 operates by computing the magnitude of the angular velocity vector. Based on its value, relative to three predefined thresholds, the system dynamically adjusts the **data transmission frequency** for each sensor individually across four levels: 5 Hz, 15 Hz, 30 Hz, and 60 Hz. This allows high-resolution data to be transmitted only during active motion, while lower transmission rates are used during periods of minimal movement.

Their results showed that the proposed adaptive algorithm reduced the data transmission up to 91.6% and current consumption up to 19.9%; with no significant difference ($p > 0.05$) between reconstructed data and full data. These results demonstrate the feasibility of adaptive transmission control for reducing bandwidth and power demands in multi-sensor wearable systems.

Data Rate and Transmission Power Adaptation

In [15] authors introduce AdaptaBLE, an adaptive control scheme for BLE communication, designed to reduce energy consumption while satisfying latency constraints. Unlike previous methods, which individually tune parameters, AdaptaBLE simultaneously adjusts the connection interval, transmission power, and data rate.

AdaptaBLE consists of three main components:

- Latency Estimator: Continuously monitor transmission delays and estimates the expected number of connection intervals (ECI) required for successful delivery of a BLE message.
- Connection Interval Calculator: At the beginning of every adaptation round (occurring every second), the system selects the worst case ECI observed over the last M rounds and uses it to calculate a new connection interval that satisfies the latency requirement t_{th}, while accounting for the connection event duration t_{CE}.
- Data Rate and Transmission Power Manager: Updated less frequently (every M rounds), this component calculates the retransmission ratio ($\mathrm{RT}_{\mathrm{ratio}}$) the fraction of rounds in which retransmissions were required. Based on two threshold values ($\mathrm{RT}_{\mathrm{th,lo}}$ and $\mathrm{RT}_{\mathrm{th,hi}}$), it categorises link quality as good, fair, or bad.

If the link quality is poor (i.e., $\mathrm{RT}_{\mathrm{ratio}} > \mathrm{RT}_{\mathrm{th,hi}}$), AdaptaBLE increases transmission power or decrease data rate to improve reliability. If the link is excellent ($\mathrm{RT}_{\mathrm{ratio}} < \mathrm{RT}_{\mathrm{th,lo}}$), it does the opposite to conserve energy. The algorithm selects between these options using a pre measured energy consumption lookup table, ensuring that each change results in a net energy benefit.

The authors conducted extensive experiments in indoor environments, including both line-of-sight (LoS) and non line-of-sight conditions. The evaluation shows that AdaptaBLE **reduces QoS failure rate by up to 5 times** compared to the prior state of the art scheme that adapts only the connection interval.

1.1.3 Trade-Offs in Adaptive Communication

Adaptive communication strategies in mobile robotics always consist of trading-off three fundamental and often conflicting objectives: energy efficiency, communication reliability, and latency. These trade-offs are central to system design and real-time decision making in robotic applications, especially in dynamic environments, large scale, or resource constrained. Effective adaptive communication scheme must understand and adjust these trade-offs dynamically.

Energy-Reliability

One of the most significant trade-offs is between saving energy and maintaining reliable links. Lower transmission power reduce energy consumption but increases the risk of packet loss, particularly in cluttered or dynamic environments. On the other hand, increasing transmission power improves signal strength and robustness but also increases power consumption.

Reliability-Latency

Maintaining high communication reliability often comes at the cost of increased latency. Retransmissions, for example, can improve delivery success but introduce delays that can affect coordination.

In [4] authors show how retransmission strategies in supervisory control systems can delay feedback to the operator, impacting task completion in remote operations. In critical missions, excessive latency can be a problem. Reliable communication, in this case, must be complemented by prioritisation mechanisms that dynamically adjust message importance and urgency.

Latency-Energy

Low latency communication typically requires higher frequency of data transmission, that consume more energy. Methods to address this trade-off include adjusting packet sizes and limiting transmission frequencies during less critical periods, as in [16].

1.2 Identified Gaps and Research Opportunities

In summary, existing adaptive communication schemes are typically designed for static wireless sensor networks, and they do not consider the operational state of mobile robots, the priority-based of data stream, or integration within a ROS2-based robotic architecture.

Original Contribution

This thesis proposes a modular and flexible ROS2-integrated framework for adaptive communication. The core of the approach is a control strategy based on the minimization of a consumption-model cost function, which determines the configuration of data rate and transmission power that minimizes current absorption. At runtime, the system continuously evaluates this cost function using real-time measurements and computes the optimal combination of communication parameters that minimise current absorption. In contrast to the solutions above, this framework does not rely on offline training (as in reinforcement learning), or adapts a single parameter. Instead, it provides a flexible, runtime-adaptive, and ROS2-native architecture suitable for real-world robotic systems, where channel conditions and mission requirements vary unpredictably. The framework is designed with flexibility as a central feature:

- It allows the user to configure the radio parameters, including available power levels, sensitivity thresholds, supported data rates, and associated power consumption.
- It enables dynamic message control by linking the frequency of each ROS2 topic to a set of user-defined state variables, allowing the system to modulate the publication rate based on robot dynamics.
- It introduces a message prioritisation mechanism, which ensures that, when bandwidth is limited (e.g., due to low data rate selected to compensate to a weak link),

low-priority messages are downsampled, preserving the integrity of high-priority communication.

Final Considerations and Comparative Evaluation

Finally, the proposed framework aims to fill a critical gap at the intersection of wireless communication and mobile robotics by offering a, flexible, and energy-aware communication control system fully integrated within ROS2. By addressing the three core research objectives, joint adaptation of communication parameters, messages rate controlled by robot-state variables, and priority-aware message scheduling, this work contributes both a practical and extensible tool for robotic applications operating under dynamic conditions. The framework's performance is evaluated in the Results chapter, where it is compared against representative state-of-the-art strategies and fixed-parameter configurations tested in a simulation environment.

Chapter 2

Background

2.1 Wireless Communication in Mobile Robotics

Wireless communication plays a main role in modern mobile robotics, enabling autonomy, team coordination, and remote supervision. As robots get more advanced and take on bigger roles in important tasks, there's a growing need for communication systems that are robust, flexible and able to adapt quickly.

2.1.1 The Role of Wireless Communication in Autonomy and Coordination

Connected robots can coordinate tasks and distribute workload more effectively than isolated units. For example, in warehouse operations or agricultural monitoring, a group of robots can cover more ground and handle more complex tasks by sharing sensor data. Sharing information also improves robustness, since multiple robots that share localization or perception data can tolerate individual failures and achieve higher overall accuracy [3]. In short, wireless links make possible distributed decision-making and collective behavior that are impossible for isolated robots.

Autonomous Decision-Making and Coordination

Wireless links let robots share knowledge and negotiate plans, increasing autonomy. Instead of relying only on its own sensors, a robot can access data collected by others. This is useful for tasks such as mapping or localization, where combining information from different units leads to more precise results. This shared information enables distributed decision making, robots can assign tasks among themselves, avoid conflicts, and optimizing the job. Multirobot systems gain robustness thanks to information redundancy and efficiency through cooperative algorithms [3]. For example, in [1] the authors present an online multi-robot SLAM system for 3D LiDAR-equipped robots that cooperatively build a unified global map and estimate relative trajectories. Thanks to real-time data exchange, the system enables robust mapping even in challenging environments. Such collaborative approaches are key enablers for adaptive communication strategies, where the transmission of critical information such as localization and mapping data can be prioritized and optimized to conserve energy while ensuring mission reliability. With high-bandwidth connections, they can even share machine learning models or distribute computing tasks across the network, or offload them to the cloud, making the entire system more efficient. [3].

Remote Supervision

In addition to increasing autonomy, wireless communication is essential when humans need to supervise or control robots from a distance. This is especially important in places that are dangerous or difficult to reach. Operators can send commands and receive sensor data, such as camera output or LiDAR scan, over the network in real-time. For example, in [4], supervisory control schemes have been studied where a human sends high-level commands over delayed links, and the robot autonomously executes them while reporting status back to the human. This control scheme split fast local control loops (on the robot) from slower high-level control (on the operator side), mitigating delay in the network. In short, wireless networking extends the operational range of robots by allowing remote supervision and intervention, which is especially important in critical applications.

2.2 Foundations of radio communication

Quick introduction

Radio communication is a method of transmitting information wirelessly using electromagnetic waves. It is a fundamental technology in numerous modern systems, including mobile networks, sensor networks, and robotic communication. The transmission occurs over the air through modulated radio frequency (RF) signals, allowing devices to exchange data without physical connections. Understanding the principles of radio communication is essential when designing or optimizing wireless systems, particularly in environments with energy constraints or variable channel conditions.

2.2.1 Overview of a Radio Module

Architecture of a Radio Module

A typical radio module is composed of several functional blocks that together enable wireless communication. These include the antenna, radio frequency (RF) front-end, baseband processor, and control interface. The RF front-end handles signal transmission and reception, including amplification, filtering, and frequency conversion. The baseband processor manages signal modulation/demodulation, encoding/decoding, and sometimes implements error correction. Finally, the control interface facilitates configuration and communication with a host processor, typically via UART, SPI, or I^2C protocols. The modular design allows integration into various platforms while maintaining a standardized interface for upper-layer protocols [2].

Physical Layer (PHY)

The physical layer (PHY) is responsible for the actual transmission and reception of raw bits over the air. It defines parameters such as modulation schemes, transmission power, frequency bands, symbol rate, and error correction mechanisms. The PHY layer directly influences the communication range, data rate, energy consumption, and robustness to interference. In practical applications, careful tuning of PHY parameters can significantly enhance performance, particularly in dynamic environments like mobile robotics or industrial wireless networks [2].

MAC Layer (Medium Access Control)

The Medium Access Control (MAC) layer operates above the PHY layer and governs how multiple devices share the communication channel. It implements protocols for addressing, channel access, collision avoidance, and frame handling. Common MAC strategies include Time Division Multiple Access (TDMA), Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), and Frequency Hopping. The MAC layer is crucial in ensuring reliable, fair, and efficient communication in networks where multiple nodes contend for the same medium, such as wireless sensor networks or mesh topologies. Its design has a direct impact on latency, throughput, and overall network scalability [2].

2.2.2 Link Quality Indicators

Wireless links in robot systems can change rapidly due to distance, obstacles, interference, and mobility. To maintain robust connectivity, robots rely on link-quality metrics, as Received Signal Strength Indicator (RSSI), Signal-to-Noise Ratio (SNR), and Link Quality Indicator (LQI), and feed them back into the communication control loop. These metrics allow robots to adapt transmission parameters (such as transmission power and data rate), allowing robots to save energy, manage bandwidth, and maintain connectivity in dynamic environments. [5][17].

Received Signal Strength Indicator (RSSI)

RSSI measures the power of a received wireless signal (often in dBm). In robotic applications, RSSI is commonly used for data rate and transmission power control and connectivity maintenance. For example, a robot can periodically measure the RSSI of a link to another robot or base station and adjust its transmit power to keep RSSI within a target range: if RSSI falls below a threshold, the robot increases power to maintain the link; if RSSI is very high, it can reduce power to save energy[5][17]. Such control loops have been demonstrated in wireless sensor and robot networks, often using simple controllers such as fuzzy logic to trade off power versus link reliability [5].

Because RSSI directly reflects received power, it is easy to measure and can be obtained in real time from most radio hardware. However, it has important limitations. RSSI is volatile, it fluctuates with multipath fading and interference and, by itself, does not distinguish noise from signal. High values of RSSI do not guarantee a good link, strong interference or noise can make the RSSI rise without improving packet delivery, as experimentally confirmed in sensor networks [5][17]. RSSI-driven schemes can oscillate, needing filtering or hysteresis in control loops to avoid instability. Moreover, RSSI thresholds can be hardware and environment specific. Although these challenges, RSSI is widely used for link estimation in robots because, unlike SNR and LQI, it is provided by almost all radios[5][17].

Signal-to-Noise Ratio (SNR)

SNR measures the ratio of signal power to noise power, typically expressed in dB. Unlike RSSI, SNR explicitly accounts for noise and is often a stronger predictor of link quality. In robotics communications, SNR is used for adaptive bit rate selection and robust modulation control. For example, in [7], it was demonstrated a prototype radio (called DRACER) that used the SNR of the last packet to choose the next packet's rate. With this scheme, links with excess SNR could safely use higher rates, saving power and latency. SNR can also be used to adapt transmit power itself. In a control loop, a robot might aim to maintain a target SNR at the receiver if measured SNR falls, it increases power (or reduces rate) to restore it [5].

SNR is closely related to the physical capacity of the channel. Because SNR directly affects bit error rate, adapting data rates based on SNR can maximize throughput under good links and avoid futile high-rate transmissions when the link is weak [7]. Accurate SNR measurement requires knowledge of the noise floor, which is not always directly available. Some radios only report RSSI or LQI value. Estimating SNR from RSSI involves subtracting a noise estimate, which can fluctuate. Also, in very dynamic links, SNR can change from packet to packet, requiring fast adaptation algorithms. Moreover, using an higher data rate requires an higher SNR margin. In a multi-robot system, where they communicate between themself on the same radio frequency, raising SNR by boosting power can increase interference to others (this is solved by frequency hopping). In summary, SNR-driven control is powerful but often implemented in combination with regulatory rules or learning algorithms (e.g. reinforcement learning strategies as in [17]).

Link Quality Indicator (LQI)

LQI is a metric defined in IEEE 802.15.4 that summarizes the quality of a received packet. Its exact formula is radio-specific, but it often reflects the chip error rate or correlator value [7]. Like RSSI, LQI is an instantaneous measure, in fact, LQI has been used similarly to RSSI for link adaptation. For example, a node might reduce power if LQI falls below a threshold to maintain efficiency [14]. In a multi-robot system, one can monitor both LQI and packet error rate together; when LQI is high and errors are low, robots can safely increase their transmission range or rate; instead, when LQI degrades, they reduce the datarate to preserve link quality [14]. Decentralized multi-agent frameworks used LQI even for localization and connectivity checks in wireless sensor infrastructures, for example, in [6] authors proposed a system where RSSI/LQI from static beacons helped locate mobile robots in a coverage task.

However, LQI has some problems. It is not standardized, different radios compute it differently, and its scale often compresses [7]. LQI is less predictive than a measured packet-delivery ratio and so it's not the best candidate for long-term selection of transmission power. Moreover, not every radio can provide this metric.

Benefits and Challenges

Using link-quality feedback enables real-time adaptation that can improve network performance, robots can save power by reducing transmission power when close to each other (or to the base station) or save bandwidth and increase the range by lowering data rates under interference [7]. Indicators like RSSI/SNR/LQI let robots exploit good links with high data rates or maximize reliability when links degrade. Centralized systems can aggregate these metrics to inform global scheduling, while decentralized ones let each robot adapt autonomously.

However, each parameter has its trade-offs. RSSI is noisy and unreliable in the presence of interference while the computation of the SNR requires noise estimation. LQI provides useful information, but it's not standardized and not all radios provide it. Additionally, these control techniques can induce oscillations, so control algorithms must carefully filter. For this reason, adaptive schemes often incorporate hysteresis, thresholds, or learning to avoid thrashing.

In summary, RSSI, SNR, and LQI enable communication controllers to adjust transmission power and data rate in real time, improving throughput, saving resources, and maintaining connectivity in presence of dynamic environments [5] [17].

2.3 Propagation effects and channel degradation

In mobile robotics scenarios, radio signal propagation is affected by multiple phenomena related to robot movement, such as physical obstacles, distance variations, multipath fading effects and weather conditions, as well as interference from other transmissions, which can degrade the quality of the received signal.

2.3.1 Attenuation Due to Distance and Earth Curvature

The effects of distance and Earth curvature on the attenuation of radio waves is a basic issue to be considered in designing and evaluating wireless communication systems. Both deterministic physical processes and empirical factors play a role in this. Theoretical models such as *free-space path loss (FSPL)* offer fundamental insights, while empirical models such as Okumura-Hata or COST-231 provide empirical estimation techniques calibrated by actual measurements in urban environments. In addition, the diffraction effects caused by the Earth curvature restrict line-of-sight (LoS) range and require more complicated modeling, especially for links with a long distance. A full analysis also includes Fresnel zones, which define the spatial volumes in which obstacles can lead to deep diffraction losses.

Theoretical Models: Free-Space Path Loss (FSPL)

In idealized free-space conditions, such as propagation through a vacuum or a free open environment, the received signal strength is only affected by the spreading of the wavefront as it travels away from the transmitter. The *Free-space path loss (FSPL)* model provides a theoretical baseline for signal attenuation in the absence of obstacles, reflections, or interference. Mathematically, it is expressed as:

$$L_{\text{fs}}(\text{dB}) = 32.45 + 20\log_{10}(d) + 20\log_{10}(f) \tag{2.1}$$

where d is the distance in Km and f is the frequency in MHz. This model serves as a reference baseline but fails to capture the complexities introduced by ground reflections, obstacles, and diffraction in real environments.

Empirical Models

Empirical Models are a category of propagation models that are fundamentally based on observation and measurement data, rather than purely theoretical explanations of a system [13].

- **Hata-Okumura Model**
  Developed to simplify Okumura's graphical path-loss data for computer implementation, the Hata model provides analytical expressions. It is an empirical formulation, though its application is limited to certain input parameter values. For **urban areas**, the median path loss is given by:

$$L(\text{urban}) = 69.55 + 26.16\log f - 13.82\log h_{te} - a(h_{re}) + (44.9 - 6.55\log h_{te})\log d \tag{2.2}$$

  In this formula, f is the frequency in MHz (ranging from 150-1500 MHz), h_{te} and h_{re} are the effective heights of the base station and mobile antennas (meters) respectively, and d is the distance from the base station to the mobile antenna. The correction factor for the effective mobile antenna height, $a(h_{re})$, depends on the city size:

  - For small to medium-sized cities:

$$a(h_{re}) = (1.1\log f - 0.7)h_{re} - (1.56\log f - 0.8) \tag{2.3}$$

  - For a large city:

$$a(h_{re}) = 8.29(\log(1.54h_{re}))^2 - 1.1, \text{ for } f < 300\text{MHz} \tag{2.4}$$

$$a(h_{re}) = 3.2(\log(11.75h_{re}))^2 - 4.97, \text{ for } f \geq 300\text{MHz} \tag{2.5}$$

  To calculate path loss in a **suburban area**, the standard Hata formula is modified:

$$L_{50}(\text{dB}) = L_{50}(\text{urban }) - 2[\log(f/28)]^2 - 5.4 \tag{2.6}$$

  And for **open rural areas**:

$$L_{50}(\text{dB}) = L_{50}(\text{urban }) - 4.78(\log f)^2 - 18.33\log f - 40.98 \tag{2.7}$$

- **Ericsson Model**
  This model, often used by network planning engineers, is a modification of the Okumura-Hata model, allowing for adjustments to parameters based on the specific propagation environment. The path loss, L, according to the Ericsson model, is given by:

  $$L = a_0 + a_1 \log(d) + a_2 \log(h_b) + a_3 \log(h_b) \log(d) - 3.2(\log(11.75h_r)^2) + g(t) \tag{2.8}$$

  where $g(t)$ is defined by:

  $$g(f) = 44.49 \log(f) - 4.78(\log(f))^2 \tag{2.9}$$

  Here, f is the Frequency in MHz, h_b is the transmission antenna height in meters, and h_r is the Receiver antenna height in meters. The coefficients a_0, a_1, a_2, and a_3 have default values that vary for urban, suburban, and rural environments.

- **COST-231 Hata Model**
  This model is a widely used extension of the Hata-Okumura model, designed for the frequency band from **500 MHz to 2000 MHz** and includes corrections for urban, suburban, and rural environments. The basic equation for path loss in dB is:

  $$L_{50}(\text{dB}) = 46.3 + 33.9 \log f - 13.82 \log h_b + c_m - ah_m + (44.9 - 6.55 \log h_b) \log d \tag{2.10}$$

  Here, f is the frequency (MHz), d is the distance (km), and h_b is the base station antenna height above ground level (meters). The parameter c_m is 0 dB for suburban or open environments and 3 dB for urban environments. The parameter a_hm is defined as:

  - For urban environments (f > 400 MHz):

    $$ah_m = 3.2(\log(11.75h_r))^2 - 4.97 \tag{2.11}$$

  - For suburban or rural (flat) environments:

    $$ah_m = (1.1 \log f - 0.7)h_r - (1.56 \log f - 0.8) \tag{2.12}$$

  h_r is the mobile antenna height above ground level. This model requires the base station antenna to be higher than all adjacent rooftops.

- **Stanford University Interim (SUI) Model**
  Proposed by the IEEE 802.16 Broadband Wireless Access working group, the SUI model is an extension of the Hata model, intended for frequencies below 11 GHz, with correction parameters for up to 3.5 GHz. It is defined for MMDS in the USA (2.5 GHz to 2.7 GHz band). The base station antenna height can range from 10 m to 80 m, and the receiver antenna height from 2 m to 10 m, with a cell radius from 0.1 km to 8 km. The SUI model categorizes terrain into three types, A, B, and C, representing increasing levels of path loss:

- **Terrain A**: Hilly areas with moderate or very dense vegetation, exhibiting the highest path loss, and considered a dense populated urban area.
- **Terrain B**: Hilly terrains with rare vegetation, or flat terrains with moderate or heavy tree densities, representing intermediate path loss and considered a suburban environment.
- **Terrain C**: Flat terrains or rural areas with light vegetation, where path loss is minimal.

The basic path loss expression of the SUI model with correction factors for distances $d > d_0$ (where $d_0 = 100$m) is:

$$L = A + 10\gamma \log\left(\frac{d}{d_0}\right) + X_f + X_h + s \tag{2.13}$$

Here, d is the distance between base station and mobile antenna in meters, λ_e is the wavelength in meters, X_f is the correction for frequency above 2 GHz, X_h is the correction for receiving antenna height, and S is the correction for shadowing in dB (a log-normally distributed factor, values between 8.2 dB and 10.6 dB). γ (gamma) is the path loss exponent. The parameter **A** is defined as:

$$A = 20 \log\left(\frac{4\pi d_0}{\lambda_e}\right) \tag{2.14}$$

The path loss exponent γ is given by:

$$\gamma = a - bh_b + \left(\frac{c}{h_b}\right) \tag{2.15}$$

where h_b is the base station antenna height in meters (between 10 m and 80 m), and **a**, b, and c are constants that depend on the terrain type. The frequency correction factor X_f and the correction for receiver antenna height X_h are expressed as:

$$X_f = 6.0 \log\left(\frac{f}{2000}\right) \tag{2.16}$$

$$X_h = -10.8 \log\left(\frac{h_r}{2000}\right), \text{ for terrain type A and B} \tag{2.17}$$

$$X_h = -20.0 \log\left(\frac{h_r}{2000}\right), \text{ for terrain type C} \tag{2.18}$$

where f is the operating frequency in MHz, and h_r is the receiver antenna height in meters.

- **ECC-33 Model**
  The ECC-33 model extrapolates original Okumura measurements and modifies assumptions to better represent a wireless system. It is considered more appropriate for "medium city" environments in Europe than highly built-up areas like Tokyo. The path loss is defined as:

$$L = A_{fs} + A_{bm} - G_b - G_r \tag{2.19}$$

The components are individually defined as:

- A_{fs} (free space attenuation):

$$A_{fs} = 92.4 + 20\log d + 20\log f \tag{2.20}$$

- A_{bm} (basic median path loss):

$$A_{bm} = 20.41 + 9.83\log d + 7.89\log f + 9.56[\log f]^2 \tag{2.21}$$

- G_b (Base station height gain factor for medium city environments):

$$G_b = \log\left(\frac{h_b}{200}\right) \cdot (13.958 + 5.8\log(d))^2 \tag{2.22}$$

- G_r (receiver height gain factor):
  * For medium city environments:

$$G_r = [42.57 + 13.7\log f][\log(h_r) - 0.585] \tag{2.23}$$

  * For large city environments:

$$G_r = 0.759h_r - 1.862 \tag{2.24}$$

Here, f is the frequency in GHz, d is the distance in km, h_b is the base station antenna height in meters, and h_r is the mobile antenna height in meters. A notable characteristic is that the predictions from the ECC-33 model do not form straight lines when plotted against distance on a log scale.

- **ITU-R P.1238-12 Model**
The ITU-R P.1238-12 Recommendation [18] "Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 450 GHz" presents an empirical formulation for the median basic transmission loss, applicable when both the transmitter and receiver are located on the same floor. The formula of the this model is given by:

$$L_b(d, f) = 10\alpha\log_{10}(d) + \beta + 10\gamma\log_{10}(f) \tag{2.25}$$

where:

- **f:** frequency in GHz
- **d:** distance in meters from the transmitter (base station)
- α**:** coefficient associated with the increase of the basic transmission loss with distance
- β**:** coefficient associated with the offset value of the basic transmission loss
- γ**:** coefficient associated with the increase of the basic transmission loss with frequency

Recommended coefficient values for α, β, γ, and σ(standard deviation of the zero-mean Gaussian distribution to simulate the Shadowing effect) are provided for various indoor environments like Office, Corridor, Industrial, and Conference/lecture rooms, covering both Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) conditions across a frequency range from 0.3 GHz to 83.5 GHz and distance ranges from 2 m to 160 m; and are provided in the following table:

Table 2.1: Model parameters for different indoor environments (from ITU-R P.1238)

Environment	**LoS/NLoS**	**Frequency range (GHz)**	**Distance range (m)**	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$	$\boldsymbol{\sigma}$
Office	LoS	0.3–83.5	2–27	1.46	34.62	2.03	3.76
	NLoS	0.3–82.0	4–30	2.46	29.53	2.38	5.04
Corridor	LoS	0.3–83.5	2–160	1.63	28.12	2.25	4.07
	NLoS	0.625–83.5	4–94	2.77	29.27	2.48	7.63
Industrial	LoS	0.625–70.28	2–102	2.34	24.26	2.06	2.67
	NLoS	0.625–70.28	5–110	3.66	22.42	1.94	9.00
Conference room	LoS	0.625–82.0	2–21	1.61	28.82	2.37	3.28
	NLoS	7.075–82.0	4–25	2.07	28.13	2.67	3.67

Earth Curvature

As radio signals travel over longer distances, the curvature of the Earth has a significant influence on line-of-sight (LoS) communication. In typical scenarios, the radio horizon, defined as the maximum distance at which the Earth's curvature still allows direct propagation, sets the practical limit for LoS transmission. Beyond this point, the direct path is blocked by the Earth's surface. In long-range communication systems where LoS is no longer available, maintaining connectivity requires alternative propagation mechanisms. Two physical phenomena that enable signals to extend beyond the radio horizon are diffraction and tropospheric scattering (troposcatter).

Diffraction occurs when radio waves bend around obstacles such as terrain features (e.g., hills or mountains) or even the curvature of the Earth. When the line-of-sight (LoS) path is partially or completely obstructed, diffraction enables the signal to "bend" around the obstacle to reach the receiver. This bending of the wavefront is more pronounced at lower frequencies, which have longer wavelengths. As a result, frequencies in the VHF range (30MHz – 300MHz) and the lower portion of the UHF band (300MHz – 3GHz) show stronger diffraction effects compared to higher frequencies [19, 9].

In the context of Earth curvature, when a signal reaches the geometric limit of the radio horizon, diffraction allows it to continue propagating by bending over the curved surface. However, this comes at a cost: the signal strength drops significantly compared to direct LoS conditions. The degree of attenuation depends on factors such as antenna height, frequency, distance, and the shape of the terrain. Smooth, rounded obstructions like hills or the Earth itself produce more predictable diffraction losses, whereas irregular terrain introduces complex and variable attenuation patterns [19, 9]. Diffraction is especially important in medium-range communications where the link just exceeds line-of-sight but remains within a few tens of kilometers beyond the horizon. Accurate modeling of this effect is critical for predicting coverage in non line-of-sight (NLoS) conditions and ensuring reliable service in systems like mobile networks, rural broadband, and broadcasting [19, 9].

For communication distances significantly beyond the radio horizon, typically 100 to 500 kilometers, **troposcatter** becomes a dominant mechanism. Troposcatter, or tropospheric scatter, exploits the fact that the troposphere (the lowest layer of the atmosphere, up to around 10–15 km in altitude) is not perfectly uniform. It contains small-scale variations in temperature, pressure, humidity, and refractive index that create localized regions of inhomogeneity [19, 9]. When a radio signal is transmitted toward the horizon (or slightly above it), a small portion of the wave interacts with these irregularities and is **scattered** in many directions. A very small fraction of the scattered energy continues toward the distant receiver, even though the two endpoints are well beyond each other's visual or geometric horizon. This allows for communication without the need for satellites or repeaters. However, troposcatter signals are extremely weak compared to line-of-sight or even diffracted signals. This means that high transmit power, large antenna gains, and sensitive receivers are typically required. Moreover, the received signal is affected by fading and variability due to changing atmospheric conditions [19, 9].

Fresnel Zones and Radio Horizon

Beyond terrain and curvature, the concept of *Fresnel zones* plays a main role in evaluating the degree of obstruction along the path. These are ellipsoidal regions around the direct path between the transmitter and receiver, with the *first Fresnel zone* being the most critical [19]. The radius of an ellipsoid (R_n) at a point between the transmitter and the receiver can be approximated in practical units by:

$$R_n = 550\sqrt{\frac{n d_1 d_2}{f(d_1 + d_2)}} \quad \text{meters}$$

where f is frequency (MHz), and d_1 and d_2 are distances (km) between the transmitter and receiver to the point where the ellipsoid radius is calculated [19]. Propagation is assumed to occur in line-of-sight (LoS) with negligible diffraction phenomena if there is no obstacle within the first Fresnel ellipsoid. The diffraction zone begins where the path intersects 60% of the radius of the first Fresnel zone (R_1) and extends to a region well beyond the transmitter's horizon, where troposcatter becomes the dominant propagation mechanism [**reference18**]. Additionally, the *radio horizon distance* due to Earth curvature is given by:

$$d_{\text{los}} = \sqrt{2a_e}(\sqrt{h_1} + \sqrt{h_2}) \tag{2.26}$$

where a_e is the effective Earth radius and h_1, h_2 are antenna heights. Beyond this horizon, as path length increases, mechanisms like troposcatter become predominant [19].

Combined Effects in Practical Contexts

In practical scenarios, especially in urban and hilly terrain, attenuation due to distance and Earth curvature is not simply additive. Multipath fading, building shadowing, tree absorption, and diffraction all contribute interactively. Measurement programs reported in [9] found that actual path loss can exceed free-space values by 15–40 dB depending on the frequency, clutter density, and antenna heights. Furthermore, field strength variability due to multipath fading and shadowing can reach standard deviations up to 12 dB in dense environments.

In summary, estimating attenuation over distance and Earth curvature demands a multifaceted approach: theoretical models describe ideal behavior, empirical models fit real world environments, and diffraction theory accounts for geometric obstructions and non-line-of-sight propagation. Consideration of Fresnel distance and radio horizon constraints is essential to ensure reliable wireless communications over both short and long ranges.

2.3.2 Signal Fading

Signal fading refers to the fluctuations in received signal strength over time, frequency, or space, caused by variations in the propagation environment. In radio systems, fading is typically categorized into two main types: *slow fading* (or shadowing), which occurs over longer distances or timescales due to large obstacles, and *fast fading*, which occurs over short distances and is caused by multipath interference. Both types of fading were extensively observed and analyzed in the empirical measurements reported in [9].

Shadowing

Shadowing, often referred to as path-to-path or location variability, describes the gradual changes in received signal strength over larger geographical areas. This form of slow fading is fundamentally caused by the shadowing of radio energy as it encounters terrain irregularities, buildings, large obstacles and walls. The degree of signal attenuation increases with greater terrain irregularity or a higher density of obstacles, with these effects becoming more pronounced at higher frequencies [9].

In urban settings, the additional power loss due to shadowing can vary significantly, ranging from negligible amounts to approximately 40 dB. For instance, measurements in Manhattan indicated signal levels 20 to 40 dB below calculated smooth-earth values. This "urban factor" tends to increase with frequency but decrease with distance from the transmitter. A crucial aspect influencing this attenuation is the angle of approach at the receiving antenna; lower angles result in greater attenuation because the radio path through intervening obstacles is effectively longer.

Statistically, the measured field strength values, are generally log-normally distributed. So the shadowing component X_σ is modeled as a zero-mean Gaussian random variable (in dB) with a standard deviation σ:

$$X_\sigma \sim N(0, \sigma^2) \tag{2.27}$$

The **standard deviation** (σ_L) typically ranges from about 5 to 20 dB, influenced by the radio frequency and terrain type. For non-urban areas with randomly located receiving antennas, a relationship for σ_L can be expressed as:

$$\sigma_L = 6 + 0.55 \left(\frac{\Delta h}{\lambda}\right)^{1/2} - 0.004 \left(\frac{\Delta h}{\lambda}\right) \text{ dB}$$

where Δh represents terrain irregularity and λ is the radio wavelength. For smooth to slightly hilly terrain at frequencies 10 MHz and above, the frequency dependence of σ_L is approximately:

$$\sigma_L \cong 5 \log f - 1 \text{ dB}$$

where f is frequency in MHz [9].

A key physical mechanism contributing to shadowing is **diffraction**, where radio waves bend around obstacles . The [19] Recommendation provides comprehensive models for calculating **diffraction loss** over various obstacle types, including spherical Earth, isolated knife-edges, and rounded obstacles. For a single knife-edge obstacle, the diffraction loss, $J(\nu)$, is a function of a dimensionless parameter ν, which combines the geometrical parameters of the path and obstacle. For ν greater than -0.78, an approximate value for the loss can be obtained from:

$$J(\nu) = 6.9 + 20 \log \left(\sqrt{(\nu - 0.1)^2 + 1} + \nu - 0.1 \right) \text{ dB}$$

For diffraction over a spherical Earth, the diffraction field strength E relative to the free-space field strength E_0 can be calculated using a formula involving a distance term, $F(X)$, and height gain terms, $G(Y_1)$ and $G(Y_2)$:

$$20 \log \left(\frac{E}{E_0}\right) = F(X) + G(Y_1) + G(Y_2) \text{ dB}$$

where X is the normalized length of the path between the antennas at normalized heights Y_1 and Y_2. This highlights how terrain and specific obstacles contribute to the overall shadowing effect.

Multipath Fading

In contrast to the shadowing, **multipath fading**, due to constructive and destructive interference of multiple signal components arriving at the receiver via different paths, is characterized by rapid and deep fluctuations in the received signal strength. These multiple paths are generated by reflections, scattering, and diffraction from surrounding terrain, buildings, and trees.

The severity of multipath fading can be considerable, with fading depths of 30 dB being common and observed amplitudes reaching 40 dB. Variations of 20 to 25 dB have been detected over short distances, for example 60 to 120 meters on city streets. The fading rate is directly proportional to both the radio frequency and the speed of the mobile unit, with signal strength minima often occurring at approximately half wavelength spacing. This interference can also introduce **echoes and delays** in the received signal, with delays ranging from -3 to +12 in dense urban environments like lower Manhattan. Such echoes can significantly limit the performance of wide-band radio systems. Different paths result in different Doppler shifts, causing beats between signals [9].

The amplitude variation of the signal envelope due to multipath can be statistically modelled based on the propagation environment:

- When **no significant direct (line-of-sight) component exists**, and the received fields are entirely scattered, the signal amplitude for a short run is typically **Rayleigh distributed** [11]. This is characteristic of indoor environments or dense urban areas where receivers are "immersed in a sea of clutter" and the field is severely distorted [9].
- Conversely, when a **significant direct component wave is present** alongside diffuse multipath, the signal envelope may follow a **Rician distribution** [11]. This is observed in less cluttered environments such as "towns and woodlands" or suburban areas, particularly when a partial line-of-sight path is available, for instance, with a well elevated transmitting antenna or a receiving antenna above local roof levels .

The received signal envelope has a distribution given by:

$$p(r) = \frac{2(K+1)r}{\Omega} \exp\left[-K - \frac{(K+1)r^2}{\Omega}\right] I_0\left(2\sqrt{\frac{K(K+1)}{\Omega}}r\right), \quad r \geq 0,\ K \geq 0,\ \Omega \geq 0 \tag{2.28}$$

where $I_n(\cdot)$ is the nth order modified Bessel function of the first kind, $\Omega = E[r^2]$ is the mean value of received signal strenght and K (the K-factor) is the ratio of the power received from the LOS component to the power contribution of the NLOS components. For $K = 0$ we have Rayleigh fading, whereas for $K = \infty$ we have no fading.

In conclusion, both shadowing and multipath fading are critical factors in urban radio wave propagation. Shadowing causes signal variability over broader areas due to large scale obstructions and terrain, statistically described by a log-normal distribution. Multipath fading, characterised by rapid fluctuations, results from the intricate interference of multiple signal paths and is often modelled by Rayleigh or Rician distributions depending on the presence of a direct signal component.

2.3.3 Interferences

When two transmitters broadcast on overlapping frequencies, their signals superimpose and act as mutual noise. This co-channel interference degrades every receiver's signal-to-noise ratio. In practice, colliding packets are corrupted or lost, forcing retransmissions. In other words, interference causes performance degradation (lower throughput), higher error rates, and effectively shorter range. Interference appears to a receiver as additional noise. This raises the bit-error rate: packets arrive corrupted or must be repeated, reducing net data throughput. In severe cases, a receiver may be unable to detect the desired signal at all, effectively reducing its reliable range. Devices may compensate by lowering data rates or increasing transmission power.

Frequency Hopping Spread Spectrum (FHSS) Frequency hopping spread spectrum (FHSS) is a transmission technique that rapidly switches the carrier among many sub-channels. Instead of using one fixed frequency, an FHSS transmitter and receiver pair agree on a pseudorandom "hop" sequence over a set of narrowband channels. Each short burst of data is sent on one channel, then the radio hops to another channel in the next burst. In general, FHSS systems transmit fast, narrowband bursts on a sequence of frequencies known only to the transmitter/receiver. The effect is that any narrowband interference will only disrupt a small fraction of the packets, and the receiver simply reassembles the message from all the hops. In practice, an FHSS radio has a fixed bandwidth subdivided into many arrow channels. During a connection, the radio transmits each symbol or block on a different channel, following a pre-agreed pseudo-random sequence. Both sides synchronize on the hop timing. Because the pattern is known only to the pair, the sequence appears random to outsiders. Because each transmitter hops independently, two FHSS radios are unlikely to stay on the same channel for long. Thus, simultaneous transmitters only collide briefly. In crowded or hostile RF environments, FHSS helps maintaining high reliability.

2.4 Mathematical models on the consumption of a radio module

The energy consumption associated with packet transmission in a wireless node can be modelled as the sum of two distinct components. The first concerns the energy consumed by the transmitter's processing circuit, while the second relates to the energy required by the power amplifier to generate the output power necessary for signal transmission. On the other hand, the energy consumption in receiving a packet is only attributable to the activity of the receiving circuit.

Let (u, v) be a wireless link between the transmitting node u and the receiving node v. Let r be the transmission speed from u to v, P_t the power required for the transmitter processing circuit, P_r the power required by the receiving circuit, $P_{u,v}$, the transmission power from u to v, and k the efficiency of the power amplifier. The energy required by u to transmit a packet of length x bits is therefore given by:

$$\epsilon_{u,v}\left(x,r\right)=\left(P_t+\frac{P_{u,v}}{k}\right)\times\frac{x}{r} \tag{2.29}$$

The energy consumed by v to receive a packet from u is:

$$\omega_{u,v}(x,r)=\frac{P_r}{r}\times x \tag{2.30}$$

However, if the transmitter does not receive an acknowledge (ACK), it retransmits the packet. This can happen due to the loss of the packet or the ACK. The transmitting node continues to retransmit until it receives the ACK or until a maximum number of attempts M is reached. In this way, both the packet and the ACK can be transmitted up to $m \leq M$ times respectively. Consequently, the actual energy consumption for packet delivery must also include the energy used during retransmissions.

Let $X \in \{1,2,...,M\}$ be the total number of packet transmissions and $Y \in \{0,1,...,M\}$ be the number of ACKs received. The total energy consumed by node u to deliver a packet to v, including ACKs, is expressed by:

$$e_t(u,v) = X \times \epsilon_{u,v}(L_d, R_d) + Y \times \omega_{u,v}(L_a, R_a) \tag{2.31}$$

where L_d and L_a represent the size (in bits) of the data packet and the ACK, respectively, while R_d and R_a are the respective data rates. The values of X and Y depend on the reliability of the link, thus showing how link reliability directly affects the energy consumption required to exchange a packet between two nodes. [21]

In modern wireless communication contexts, *Link Quality Estimation* (LQE) is crucial for ensuring communication reliability, routing efficiency and dynamic network topology management. Accurate estimation of link quality allows for improved resource allocation, reduced energy consumption, and increased system robustness, especially in dynamic or high-density node scenarios. Traditionally, connection quality is estimated using metrics such as Received *Signal Strength Indicator* (RSSI), *Signal-to-Noise Ratio* (SNR) and *Link Quality Indicator* (LQI), all of which can be obtained directly from radio modules and are often available at low computational cost.

Among the available metrics, the *Packet Reception Rate* (PRR) is widely considered one of the most reliable measures for assessing the actual quality of communication. The PRR is defined as the ratio between the number of packets received correctly and the total number of packets transmitted (considering the retransmissions), and is a direct indicator of the reliability of the connection over time. In fact, the average number of transmissions of a packet (considering both the first and any retransmissions) can be calculated using the PRR:

$$n_T = \frac{1}{PRR} \tag{2.32}$$

However, calculating the PRR requires observing transmission sequences and is therefore not always available in real time. For this reason, it is essential to use models that allow the PRR to be estimated based on other immediately observable quantities, such as RSSI, SNR or LQI.

There are well-established theoretical and semi-empirical models in the literature. One of these is the theoretical model based on SNR, referred to as the TH-SNR Model, for the 2.4 GHz physical layer of the IEEE 802.15.4 standard, typically used in Low-power and Lossy Networks (LLN), in which the PRR is expressed by the following relationship:

$$PRR = \left(1 - Q\left(\sqrt{2 \cdot \frac{B_N}{R} \cdot 10^{\frac{SNR}{10}}}\right)\right)^{Nbit} \tag{2.33}$$

where $Q()$ represents the Q function, B_N is the noise bandwidth, R is the data rate, and Nbit is the number of bits per packet.

Alongside this theoretical approach, empirical models based on logistic regression are also proposed. For example, the LR-SNR Model establishes a link between SNR and PRR through the function:

$$PRR = \frac{1}{1 + e^{b_1 \cdot SNR + b_2}} \tag{2.34}$$

where b_1 and b_2 are parameters determined by fitting experimental data. A similar approach is also used for RSSI, giving rise to the LR-RSSI Model:

$$PRR = 1 - \frac{1}{1 + c_1 \cdot e^{c_2 \cdot RSSI + c_3}} \tag{2.35}$$

In this model, the constants c_1, c_2 and c_3 must also be determined experimentally. The logistic form of the function allows for a continuous and differentiable representation of the PRR, making it suitable for dynamic systems and adaptive control.

Among the metrics available at the output of the radio module used, only RSSI is actually provided. Other metrics commonly used to evaluate radio link quality, such as SNR and LQI, are not supported by the module in use, or are not accessible via software. For this reason, system modelling and control choices were based solely on RSSI.

In addition to these models, there is also a simplified theoretical model that links RSSI and PRR, again for the IEEE 802.15.4 standard at 2.4 GHz, which is shown below:

$$PRR = \left(\frac{1}{2} + \frac{1}{2} \cdot \sqrt{1 - e^{-1.78944 \cdot (10^{\frac{RSSI - P_n + 1}{10}} - 1)}}\right)^{Nbit} \tag{2.36}$$

where P_n is the power in dBm of the background noise. [10]

The current absorbed by the wireless module is calculated by distinguishing between two operating intervals:

- The time during which the module is actively transmitting, whose consumption depends on the selected power.
- The time during which the module is in reception mode (idle), characterised by a specific consumption.

These intervals are directly related to the amount of data to be transmitted, the transmission frequency defined in the previous phase, the selected data rate and any retransmissions caused by low link quality:

$$T_{\text{req}} = \sum (f_k \cdot (nbits_k \cdot n_T)) \tag{2.37}$$

$$D = T_{\text{req}} / T_{\text{available}}(datarate) \tag{2.38}$$

$$I = I_{rx} \cdot (1 - D) + I_{tx} \cdot D \qquad D = \frac{t_{on}}{T} \tag{2.39}$$

Where T_{req} is the total throughput required to send all messages at their computed frequencies, $T_{\text{available}}(datarate)$ is the real throughput of the selected data rate, f_k is the frequency calculated in the previous step of the ROS k message, $nbits_k$ is the size in bits of message k, and n_T is the average number of transmissions calculated from the PRR.

Chapter 3

Methods

3.1 Hardware Setup and Radio Module Specification

The wireless communication system used in this project is built around the DIGI XBee SX RF Modem, based on a XBee-PRO SX RF Module, an high-performance sub-GHz transceiver operating in the 902–928 MHz ISM band and designed for long-range communication.

3.1.1 Radio Module Technical Specifications

Communication with the module can be established through the serial port (UART). The technical specifications of the radio module are the following:

- **Operating Frequency:** 902–928 MHz (ISM band)
- **Modulation:** Frequency Shift Keying (FSK)
- **Channel Access:** Frequency Hopping Spread Spectrum (FHSS)
- **Transmission Power:** Configurable up to +30 dBm (20dBm, 27dBm 30dBm)
- **Receiver Sensitivity:** Down to –113 dBm, depending on configuration
- **RF Data Rate:** Configurable, 10kbps, 110kbps and 250kbps
- **Interface:** SPI, UART, supporting baud rates up to 921600 bps
- **Range:**
  - **Urban:** Up to 18 km (11 mi) at low data rate
  - **Outdoor(line-of-sight):** Up to 105 km (65 mi) at low data rate
- **Power Consumption:**
  - Transmit (max power): 900 mA

  - Receive: 40 mA
  - Sleep: 2.5 μA

- **Antenna Interface:** RP-SMA connector

Serial communication modes

The XBee-PRO SX RF Module supports two different modes for the serial communication with the host: **Transparent mode** and **API mode**.

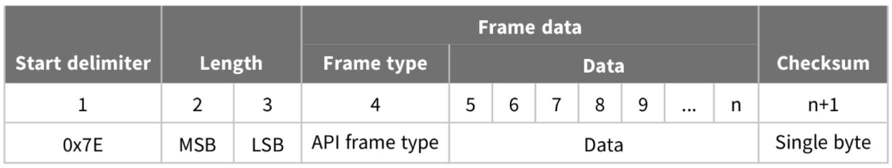
- **Transparent Mode:** This is the default mode for the devices. In this mode, the module works as a serial line replacement: it queues all UART data received via the serial port for RF transmission, and forwards any received RF data out through the serial port. The destination address for RF transmissions is determined by the DH (Destination Address High) and DL (Destination Address Low) parameters. This mode offers a simple interface where all received serial data is transmitted on the RF channel. When the radio works in Transparent mode, to change a parameter it should enter in Command mode. Command mode is a state where the firmware interprets incoming characters as AT commands to modify the device's configuration parameters. To enter Command mode the sequence "+++" must be issued within one second, with a minimum of one second of silence before and after the sequence. If no valid AT commands are received within the Command Mode Timeout (CT) (default 10 seconds), the device exits Command mode. It can also be forced to exit by sending the CN (Exit Command Mode) command. Changes made to configuration command registers using AT commands do not take effect until they are applied with AC (Apply Changes) or saved permanently with WR (Write) commands.

- **API Mode:** API mode is an alternative to Transparent mode, providing a frame-based protocol that allows for packet-based data direction. The device communicates UART or SPI data in packets, also known as API frames. This mode is ideal for structured communications with computers and microcontrollers. Moreover, when working in API mode, it's easier to change a parameter, since it's only need to send the right API frame with the new parameter value. Key advantages of API mode include:

  - Easier transmission to multiple destinations, which is beneficial for complex robotic systems sending data to various components.
  - The host receives the source address for each received data frame, providing critical information about the origin of data.
  - Parameters can be changed without entering Command mode, offering more dynamic configuration flexibility
  - Configuration parameters can be queried or set while a pending command is in progress, a feature not possible in Command mode..

  An API frame (Figure 3.1) consists of the following fields:

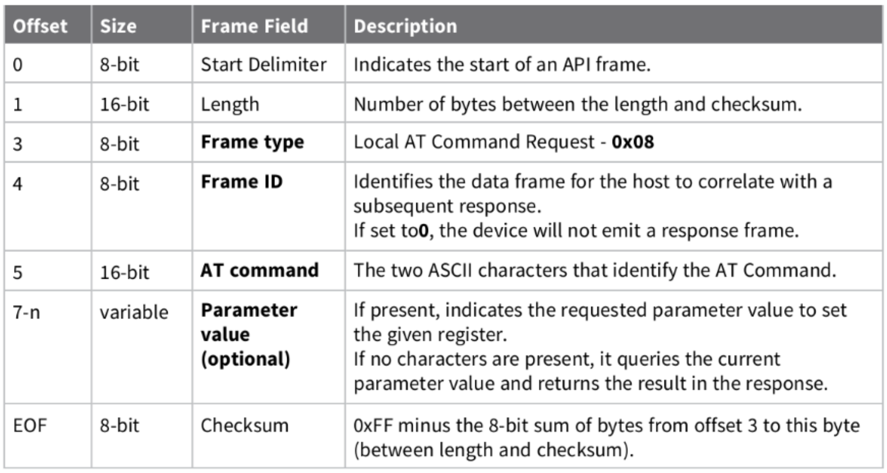
- **Start Delimiter (1 byte):** This is always 0x7E. Any data received before this byte is silently discarded.
- **Length (2 bytes):** This field specifies the number of bytes in the Frame Data field. It does not include the checksum field.
- **Frame Data (variable bytes):** This is the core of the API frame and contains the data being transmitted:
  1. **Frame Type (1 byte):** This single byte identifies the specific type of API frame (e.g., Transmit Request, Local AT Command Request). It dictates how the subsequent "Data" field is organized.
  2. **Data (variable bytes):** This field holds the actual payload or command parameters, and its structure depends on the "Frame Type". Multi-byte values within this field are sent in big-endian.
- **Checksum (1 byte):** This byte is used for error detection. If the frame is not received correctly or the checksum fails, the XBee module will indicate the failure.



Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Data							Single byte

Figure 3.1: Structure of an API frame.

For example, to configure a parameter on the local device, an AT command request must be sent. This is done using a specific API frame type identified by the code **0x08**, which corresponds to the Local AT Command Request. The structure of this frame type is illustrated in Figure 3.2.



Offset	Size	Frame Field	Description
0	8-bit	Start Delimiter	Indicates the start of an API frame.
1	16-bit	Length	Number of bytes between the length and checksum.
3	8-bit	**Frame type**	Local AT Command Request - **0x08**
4	8-bit	**Frame ID**	Identifies the data frame for the host to correlate with a subsequent response. If set to**0**, the device will not emit a response frame.
5	16-bit	**AT command**	The two ASCII characters that identify the AT Command.
7-n	variable	**Parameter value (optional)**	If present, indicates the requested parameter value to set the given register. If no characters are present, it queries the current parameter value and returns the result in the response.
EOF	8-bit	Checksum	0xFF minus the 8-bit sum of bytes from offset 3 to this byte (between length and checksum).

Figure 3.2: Structure of a Local AT Command Request API frame.

RF communication modes

Moreover, the XBee-PRO SX RF Module offers several RF communication modes, also referred to as delivery methods, which define how RF data is transmitted across a network. These methods can be configured using the TO (Transmit Options) command when the device is in Transparent mode, or through the TxOptions field within API mode frames.

One fundamental mode is **Point-to-Point/Point-to-Multipoint**, where data is transmitted directly from the sender to the destination. In this mode, intermediate nodes do not repeat the packet, so the receiving device must be within the direct RF range of the transmitting device for unicast delivery.

Another communication mode is **Repeater mode (directed broadcast)**, which enables messages to cross a network by being broadcast by intermediate nodes. If a node receives a message and is not the final destination, it will broadcast it, extending the reach of the transmission. The maximum number of hops for these broadcasts is governed by the **NH (Network Hops)** and **BH (Broadcast Hops)** parameters, and non-routing nodes (configured with **CE = 2**) will not repeat directed broadcasts.

The last RF communication mode is **DigiMesh**. This forms a peer-to-peer topology where all nodes cooperatively transmit information. The advantages of DigiMesh include automatic routing, where messages hop along dynamically discovered paths and automatically network creation. DigiMesh also incorporates a quiet protocol to minimize routing overhead, performs route discovery only when needed, uses selective acknowledgments for route requests, ensures reliable data delivery through acknowledgments (ACKs), and supports low-power synchronized sleep modes.

The differents between Repeater mode and DigiMesh are that Repeater mode provides a simpler form of broadcasting across a network with defined hop limits. While DigiMesh offers a much more robust, dynamic, and self-organizing mesh network, complete with advanced routing algorithms, reliable unicast delivery with acknowledgments, and synchronized sleep capabilities, making it ideal for power-efficient distributed sensor networks.

Key Communication Parameters and Mechanisms

Frequency Hopping Spread Spectrum (FHSS): The XBee-PRO SX RF Module uses FHSS as its spreading technology, contributing to its robustness in challenging RF environments. This technique mitigates interference by making the radio "hop" to different channels within the 902–928 MHz band for each packet transmission. By default, only 50 channels are enabled, but up to 101 can be selected, significantly reducing the probability of persistent interference on a single channel and improving link reliability. This is particularly important for mobile robots operating in dynamic environments with various interference sources.

RF Data Rate (BR): The device supports software-selectable RF data rates of 10 kb/s, 110 kb/s, and 250 kb/s. A higher data rate allows for reduced transmission time, which contributes to lower energy consumption during the active transmission phase. However, it also decreases receiver sensitivity, meaning that a stronger signal (and so higher transmit power) is needed to maintain stable communication at the same distance and environmental conditions. This presents a direct trade-off between data rate and power that the project's optimization algorithm must dynamically manage to balance energy efficiency and communication quality.

Transmit Power Level (PL): the module supports three selectable transmit power levels (20dBm, 27dBm and 30dBm). Reducing the power level can significantly decrease current consumption but comes at the expense of reduced radio range.

Hardware Interface and Physical Specifications

Serial Communication Interfaces: UART (Universal Asynchronous Receiver/Transmitter). The interface baud rate is software selectable via the BD (Interface Baud Rate) command, ranging from 1200 to 921600 baud. For reliable data transfer, RTS (Request to Send) and CTS (Clear to Send) can be used for UART flow control.

Power Requirements: The XBee-PRO SX RF Module operates within a supply voltage range of 2.6V to 3.6V with a typical supply voltage of 3.3 V, while the Modem has a supply voltage range of 7V to 30V with a typical supply voltage of 12V. Transmit current varies with the selected power output, ranging from 330 mA for 20 dBm up to 900 mA for 30 dBm (at 3.3V) for the module.

3.1.2 System Configuration

The hardware setup consists of two DIGI XBee SX Modems operating in point-to-point mode. One module acts as the transmitter, physically connected to a Raspberry Pi 5, while the other acts as the receiver, connected to a laptop workstation. Both modules were powered using regulated 12V power supplies, and communication between each XBee Modem and its respective host was established via a USB-to-serial converter cable. The serial interface was used to transmit data using the API mode of the XBee protocol, which provides structured frames.

On the Raspberry Pi 5, the operating system used was Raspberry Pi OS (64-bit). A Docker container was used to run Ubuntu with ROS2, enabling modular deployment and environment reproducibility. On the laptop side, the receiver was also interfaced via USB-serial, and custom software was used to capture, decode, and reconstruct the received ROS2 messages.

The optimization algorithm was executed only on the transmitter side (Raspberry Pi 5B), since with the Xbee Sx module it's possible to remotely configure radio parameters of another device over the air. So, communication parameters such as the data rate, which must match between transmitter and receiver, was dynamically updated on receiver by the transmitter.

For both the transmitter and receiver modules, Abracon AEACAC195013-S915 antennas were used. These are high-efficiency monopole antennas optimized for the 900 MHz ISM band. Each antenna provides a nominal gain of +3 dBi, with an omnidirectional radiation pattern in the horizontal plane.

3.2 Wireless Channel Characterization and Performance Measurement

3.2.1 Real Throughput Measurement

To accurately characterize the wireless communication performance of the radio module, a series of tests were conducted to measure the real achievable throughput at different nominal data rates.

Test Setup

The test was performed using two XBee SX Modems configured in transparent mode (so without the API frame protocol overhead) and connected to two embedded devices (Raspberry Pi and Laptop) via UART. To simulate ideal conditions:

- The two nodes were placed in line-of-sight (LOS), close proximity (about 4 meters)
- The RF output power was set to its maximum value
- No other RF sources were active in the frequency band to eliminate interference

The test was performed by continuously sending as much data as possible from the transmitter to the receiver over a fixed period of 30 seconds, while logging the total amount of data successfully received on the other side. The throughput was then calculated by dividing the received payload size (in bits) by the total transmission time (30):

$$\text{Throughput} = \frac{\text{Total received bits}}{\text{Total transmission time}} \tag{3.1}$$

The test was repeated 10 times for each supported data rate of the radio module (250 kbps, 110 kbps, and 10 kbps), and then the the average value (for each data rate) was computed. This approach provides a practical estimation of the real usable throughput under conditions of continuous transmission.

Results

The measured throughput for each nominal data rate is summarized in the table below:

Table 3.1: Measured Throughput for Each Nominal Data Rate

Nominal Data Rate (bps)	**Measured Throughput (bps)**
250,000	∼90,000
110,000	∼65,000
10,000	∼7,000

3.2.2 RSSI–PRR Modeling

In the context of adaptive wireless communication, accurately estimating link quality is essential to make right changes on transmission power and data rate. Among the various link quality indicators discussed in literature, such as SNR (Signal-to-Noise Ratio), LQI (Link Quality Indicator), and RSSI (Received Signal Strength Indicator), only RSSI is provided by the radio module used in this work. Consequently, the entire modeling and control logic was built around RSSI.

To construct an empirical model linking **RSSI to Packet Reception Rate (PRR)**, a series of experimental measurements were conducted. The goal was to derive a function $PRR = f(RSSI)$ suitable for predicting link reliability in real-time, enabling energy-efficient and robust communication adaptation.

Environment and Experimental Setup

The measurement tests were performed in an indoor environment, although outdoor testing would have increased the generalizability of the results, it was not possible to perform such experiments due to company policies. One XBee SX Modem (the receiver) was mounted in a fixed position, connected via USB–UART to a Raspberry Pi 5B. The other radio (the transmitter) was connected to a laptop and was physically carried through the environment to simulate realistic mobility scenarios. The experiment consisted of gradually moving the transmitter away from the receiver, crossing physical obstacles such as walls and doors. This introduced varying channel conditions including line-of-sight, non-line-of-sight, and multipath fading.

Metrics Collection and Data Processing

To characterize the link behavior, three tests, with the three different data rate values, were performed. During each test, the transmitter sent a total of **1800 packets** at a constant rate of **10 Hz**, resulting in a test duration of 3 minutes. This time window was sufficient for the operator to move away from the receiver, cross obstacles, and return close to the receiver, covering a range of signal conditions from strong line-of-sight (LOS) to weak NLOS links.

Each packet included a **32 bytes payload**. Together with the 18-byte protocol overhead, each packet had a **total size of 50 bytes**, resulting in a total throughput requirement of:

$$Throughput = 50 \times 8 \times 10 = 4kbps \tag{3.2}$$

This value is well below the maximum throughput of approximately 7 kbps achievable with the radio modules operating with the lowest data rate value (10 kbps). Operating below the throughput limit ensured that the network was not saturated and that observed packet losses were primarily due to link degradation rather than buffer overflows or congestion.

During the test, diagnostic and performance metrics were collected on both the transmitter and receiver sides, including:

- **RSSI (Received Signal Strength Indicator):** Measured at the receiver, RSSI represents the strength of the received signal in dBm. It is a direct physical layer indicator of signal quality. Because of his highly noisy nature, a moving average filter with a five length window was applied.

- **GD (Good Packets Received):** This counter increments when the receiver successfully receives a data frame with a valid MAC header over the RF interface.
- **UA (Unicasts Attempted Count):** This metric represents the number of unicast transmissions that required an acknowledgment. It reflects the total number of transmission attempts initiated by the transmitter for unicast communication.

The **Packet Reception Rate (PRR)** was calculated as the ratio of Good Packets Received (GD) to Unicast Attempts (UA), as expressed by the formula:

$$PRR = \frac{GD}{UA} \tag{3.3}$$

The final dataset consisted of a series of timestamped pairs ($RSSI_i$,PRR_i), each representing the signal strength and corresponding reception quality at a given point in the experiment.

To ensure the accuracy and consistency of the collected data, since *RSSI* and *GD* were collected on the receiver side while *UA* on the transmitter side, it was essential to synchronize the clocks of the transmitter and receiver devices. Even small variations in time could result in misaligned packet logs, leading to incorrect association between transmitted packets, received packets, and their corresponding *RSSI* measurements.

For this purpose, Chrony was used. Chrony is a versatile and precise network time synchronization tool, designed to maintain accurate system clocks, even on devices with unstable or drifting hardware clocks. It operates using the Network Time Protocol (NTP) but offers superior performance in terms of convergence speed and resilience to clock frequency changes, making it particularly suitable for experimental environments.

In this setup, both the transmitter and receiver devices were synchronized to the same NTP server using Chrony. The observed average clock offset between the devices was within one millisecond, ensuring sufficient precision for aligning packet logs and performance metrics.

Using Chrony allowed precise correlation between transmission and reception events, ensuring all metrics were accurately matched across devices. This step was critical to guarantee the reliability of the collected dataset and the validity of the resulting RSSI–PRR model.

Model Fitting

To model the relationship between RSSI and PRR, a logistic function fit was adopted. This choice is motivated by prior literature and theoretical considerations, where the logistic function is commonly used to approximate the transition region from high-reliability to low-reliability regimes in wireless communication.

The selected model is 2.35, where c_1, c_2, and c_3 are parameters determined through non-linear curve fitting of the collected data. To find the optimal parameters values, we used the *scipy.optimize.curve_fit* Python function on all three RSSI-PRR datasets, obtaining one fitting function for each data rate value.

The fitted models for the three data rates are plotted in Figures 3.3, 3.4, and 3.5. Each figure includes:

- The empirical data points collected during the experiments.

- The corresponding fitted logistic curve.

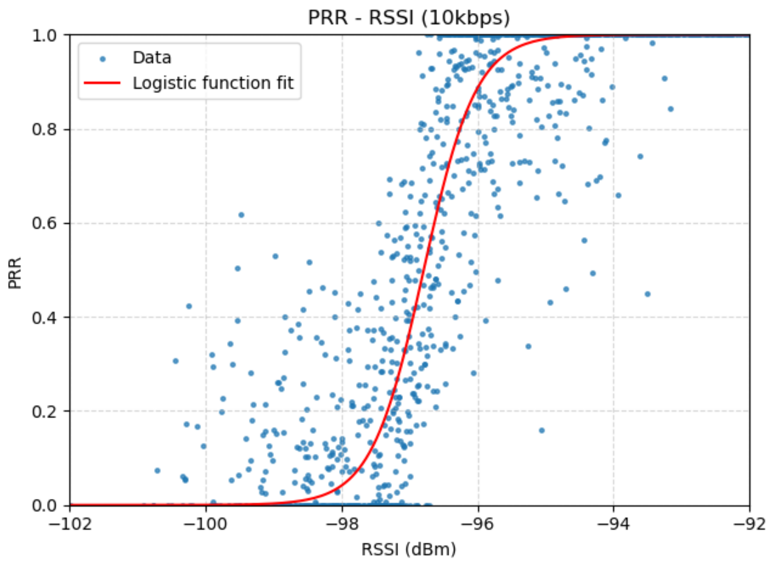


Figure 3.3: Logistic model fitting for 10 kbps data rate.

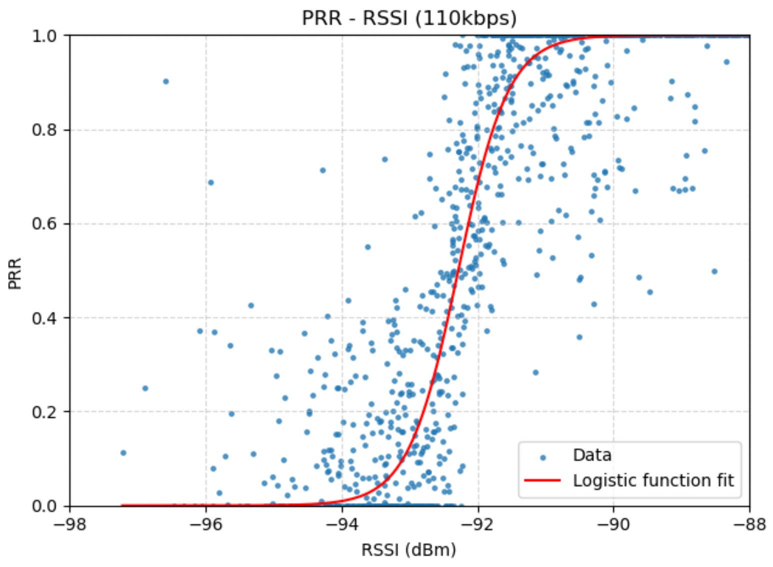


Figure 3.4: Logistic model fitting for 110 kbps data rate.

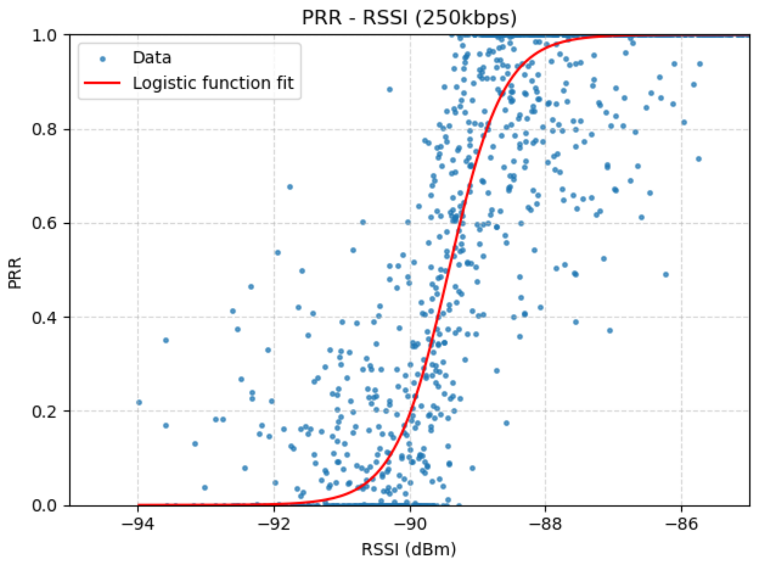


Figure 3.5: Logistic model fitting for 250 kbps data rate.

Below the final logistic equations for the three tested data rates:

$$\begin{aligned} PRR_{10} &= 1 - \frac{1}{1 + 0.59 \cdot e^{2.60 \cdot RSSI + 251.90}} \\ PRR_{110} &= 1 - \frac{1}{1 + 1.50 \cdot e^{2.71 \cdot RSSI + 249.87}} \\ PRR_{250} &= 1 - \frac{1}{1 + 1.78 \cdot e^{2.47 \cdot RSSI + 220.44}} \end{aligned} \tag{3.4}$$

To assess the quality of the fitted models, we computed the Root Mean Squared Error (RMSE) between the predicted PRR values and the empirical measurements. The RMSE values obtained were:

Table 3.2: RMSE of RSSI-PRR model

DataRate	**RMSE**
10kbps	0.164
110kbps	0.166
250kbps	0.175

Despite RMSE values ranging from 0.164 to 0.175, which are not negligible, this level of error is considered acceptable in an indoor environment where fading effects are significantly. Indoor multipath fading is typically more severe than outdoor since indoor spaces enforce multiple strong reflections. Outdoors, the dominant LOS path helps reducing the multipath fading effects. Moreover, the inflection points identified through fitting:

- **10 kbps:** -96.79 dBm
- **110 kbps:** -92.28 dBm
- **250 kbps:** -89.42 dBm

are significantly higher than the theoretical receiver sensitivity values reported in the XBee SX datasheet:

- **10 kbps:** -113 dBm
- **110 kbps:** -106 dBm
- **250 kbps:** -103 dBm

Furthermore, the empirical link degradation starts well before these theoretical thresholds, with a clear drop in PRR already observable around:

- **10 kbps:** $\sim$ -94 dBm
- **110 kbps:** $\sim$ -90 dBm
- **250 kbps:** $\sim$ -86 dBm

This discrepancy is probably due to the indoor experimental environment, where multipath fading and non-line-of-sight conditions degrade communication performance earlier than in the idealized free-space conditions assumed in datasheet testing.

3.3 ROS2 Communication Implementation

This project relies on **ROS2 (Robot Operating System 2)** as the middleware framework for managing data exchange between different components of the robotic system. In ROS2, software components, called *nodes*, communicate by publishing and subscribing to *topics*, which are streams of typed `messages`. For example, a sensor node might publish a `sensor_msgs/msg/PointCloud2` message to the `/lidar_points` topic, while another node subscribes to that topic to process or transmit the data. This abstraction allows modular design and decouples computation from communication.

While ROS2 is typically used in systems with reliable and high-bandwidth connections, wireless communication introduces new challenges. In particular, low data rates, increased latency, and the risk of packet loss require special handling of large or frequent messages.

To address these limitations, a custom communication control module was developed and integrated into the ROS2 architecture. This module acts as an intermediary between ROS2 topics and the low-level wireless interface provided by the DIGI XBee SX modules. It implements several key features to ensure efficient and robust communication over a constrained wireless link:

- **Message fragmentation and reassembly** to allow large ROS2 messages to be split into smaller, transmittable frames.
- **Protocol overhead management**, which accounts for the additional bytes added by the serial communication protocol and custom headers.
- **Partial message reconstruction**, enabling incomplete messages reconstruction instead of being discarded entirely.
- **Filtering and compression of PointCloud data**, to reduce bandwidth usage without compromising essential information.

The following subsections describe the implementation details of these components and how they interact with the ROS2 messaging system to maintain reliable communication in a low-bandwidth wireless environment.

3.3.1 Message Fragmentation and Protocol Overhead

In ROS2, messages such as `sensor_msgs/msg/PointCloud2` or `sensor_msgs/msg/Laserscan` are often large in size and can easily exceed the maximum payload allowed by the XBee SX radio modules. These radios, operating over a serial interface with API framing, impose strict limits on the number of bytes that can be transmitted in a single frame, for our radio the maximum payload is 256 bytes. To enable reliable transmission of such messages over the low-bandwidth wireless channel, a message fragmentation system was implemented.

The sender node, written in Python and integrated with the **rclpy** ROS2 client library, serialises each ROS2 message using the `rclpy.serialization.serialize_message` function. The resulting byte stream is then divided into smaller chunks, each of which fits within a transmission frame. The size of each chunk is determined based on the maximum payload size (256 bytes). Each chunk starts with a **custom header** of 5 bytes to allow proper reassembly on the receiver side, so the maximum payload is reduced to 251 bytes. The header includes the following fields:

- **Message ID:** A unique identifier assigned to the message instance. (2 bytes)
- **Chunk Index:** The sequence number of the current chunk within the message. (1 byte)
- **Total Chunks:** The total number of chunks that the message has been divided into. (1 byte)
- **Message Type ID:** An identifier representing the ROS2 message type (e.g., PointCloud2, Odometry). (1 byte)

This header design ensures that the receiver can verify the integrity and completeness of the message and reassemble the chunks in the correct order.

In addition to chunking the message payload, the implementation explicitly accounts for protocol overhead introduced by the XBee API mode. This overhead includes frame delimiters, length fields, checksums, and other control bytes (18 bytes in total), which reduce the usable bandwidth. It's important to consider this protocol overhead in order to precisely compute the actual size of each ROS2 message, which is a main parameter for the optimisation algorithm.

The fragmentation system is essential to enable the transmission of large ROS2 messages over a constrained serial wireless link. Without this mechanism, communication would be limited to small messages only, significantly restricting the functionality of the robotic system.

3.3.2 Message Reconstruction

On the receiving end of the communication link, fragmented messages are handled by a custom receiver node, implemented in the **receiver.py** script. As each packet arrives via the serial interface, the node first parses the custom header to extract the **Message ID**, **Chunk Index**, **Total Chunks**, and **Message Type ID**. This information is used to associate the received chunk with the correct message and to place it in the appropriate position within a reconstruction buffer.

The system maintains a buffer for each in-progress message, storing received chunks until either all expected chunks have arrived or a timeout is reached. Under ideal conditions, the full message is reassembled by concatenating all the chunks in order. However, in real-world wireless communication, particularly over low-bandwidth or lossy links, it is common for some chunks to be lost due to interference or signal degradation.

To ensure system resilience under such conditions, the receiver supports a **partial message reconstruction** strategy. When a timeout occurs or it is determined that certain chunks are missing, the system fills the missing positions with predefined filler data (e.g., zero bytes). This allows the receiver to proceed with message reassembly and publication, even if the reconstructed message is incomplete. While such messages are marked as corrupted, they may still contain sufficient information for downstream processing, particularly in the case of sensor data such as PointClouds, where partial spatial coverage is often preferable to a complete data loss.

Once the reconstruction process is complete, the resulting byte stream is deserialised using the `rclpy.serialization.deserialize_message` function. The message is then published on the appropriate ROS2 topic based on the decoded message type, allowing other nodes in the system to consume the data transparently.

This mechanism significantly improves the robustness of the communication pipeline by reducing the impact of packet loss on high-frequency or high-volume message streams, and contributes to maintaining system continuity in degraded network conditions.

3.3.3 PointCloud Filtering and Compression

PointCloud messages, typically published using the `sensor_msgs/msg/PointCloud2` type in ROS2, represent a dense collection of 3D spatial points acquired by LIDAR sensors. These messages are often large in size—frequently exceeding several hundred kilobytes per frame—and present a significant challenge for transmission over low-bandwidth wireless links. To address this, a dedicated pre-processing step was introduced to filter and compress PointCloud data before transmission. The filtering and compression are handled by a ROS2 node implemented in C++ (`PointCloudFilterNode`). This node subscribes to raw LIDAR data published on `/sensors/lidar3d_0/velodyne_points` and performs two sequential operations:

1. **Distance-based Filtering:** The incoming PointCloud is first filtered to retain only the points within a configurable radius from the robot, defined by the `max_range` parameter (2m arbitrarily chosen). This spatial filtering step removes distant points that are unlikely to be relevant for immediate navigation or obstacle detection, significantly reducing the overall data size.

2. **Voxel Grid Downsampling:** The filtered cloud is then passed through a Voxel Grid filter (`pcl::VoxelGrid`) to reduce point density. This filter partitions the 3D space into voxels (cubic cells of a given `leaf_size`, 0.1m arbitrarily chosen), and represents each voxel with a single representative point. This compression method preserves the geometric structure of nearby objects while reducing the total number of points.

The result of this process is a significantly smaller PointCloud, which is then published on the topic `/output/pointcloud`. This topic is subscribed to by the Python-based sender node (`sender.py`), which serialises and transmits the message over the wireless link.

The filtering based on distance (max_range: 2.0 m) and voxel grid downsampling (leaf_size: 0.1 m) was selected with the specific goal of ensuring a representation of obstacles that is sufficiently accurate for reactive navigation, while minimising the amount of data to be processed or transmitted. In this context, preserving the exact geometric shape of obstacles is not necessary, as the robot does not require this kind of precision in its path planning, but only needs to reliably detect and avoid collisions. Any loss of fine detail does not compromise the safety or functionality of the system: the critical factor is accurately identifying obstacles in the navigable environment, not reproducing every surface irregularity of the detected objects.

In addition to the filtered cloud, the node computes the average distance of the detected points within the filtered region and publishes this value on a separate topic `/output/filtered_object_distance`. This distance metric is used downstream by the communication control module to dynamically adjust the message priority or transmission frequency.

On the transmission side, the sender node includes logic to calculate the actual message size, including protocol overhead, using the `compute_dim` function in `utils.py`. This function determines the number of bytes required to transmit the full message once fragmented and framed for the XBee protocol. The calculated size is then passed to the control algorithm, which uses it to optimise the transmission rate based on available throughput and current channel conditions.

Together, these filtering and compression strategies allow the system to efficiently transmit high-density LIDAR data in constrained environments, prioritising essential information while minimising bandwidth usage.

3.4 Control Logic and Optimisation

The adaptive communication control algorithm developed in this thesis is designed to minimise the energy consumption of a wireless communication module while maintaining reliable message transmission in dynamic environments. It operates by jointly optimising the transmission power, data rate, and message publication frequency of multiple ROS2 data streams. This approach ensures that essential information continues to flow reliably even in dynamic environments in conditions of channel degradation.

The control is modular and generic, making it applicable to a wide range of robotic platforms and radio modules, as long as basic feedback such as RSSI is available. Its design allows for real-time adaptation, enabling autonomous systems to respond to changes in mobility, obstacle proximity, and channel quality.

Objectives

The algorithm is designed to minimise the cost function. To achieve this, the system dynamically selects:

- **Transmission power** $P_{\text{tx}} \in \{20,27,30\}$ dBm
- **Data Rate** $R \in \{10,110,250\}$ kbps
- **Transmission frequency** f_k of each ROS2 message k, bounded between $f_{\text{min,k}}$ and $f_{\text{max,k}}$.

The central cost function minimised by the control logic is the average current consumption of the module, computed as 2.39.

Algorithmic Flow

The control logic operates in periodic cycles, executing the following steps:

- **Environmental Monitoring:** The RSSI value is continuously monitored and used to:
  - estimate the probable number of transmissions for each combination of transmission power and data rate using the logistic models previously fitted (3.4).
  - use it as feedback of the control to filter data rate and transmission power configurations that allow to stay above the sensitivity thresholds.
- **Frequencies Computing:** each data stream is associated with a state variable (e.g., robot speed, distance to obstacle, angular velocity) that reflects its urgency or relevance. These variables are mapped to desired publication frequencies using configurable scaling functions (e.g., linear, quadratic, exponential). This results in a proposed set of frequencies $f_k \in [f_{\text{min,k,}}, f_{\text{max,k,}}]$ for each message k.
- **Throughput Budgeting and Feasibility Check:** the algorithm computes the total required throughput as in 2.37. Then it is compared to the available throughput that's depends on the current data rate (2.38), if $T_{\text{req}} > T_{\text{available}}(datarate)$ the message frequencies must be reduced. Two alternative strategies are employed:
  - **Proportional reduction:** all frequencies are scaled down uniformly.
  - **Priority-based reduction:** frequencies of low-priority streams are reduced first, preserving high-priority data.
- **Energy Estimation for Each Configuration:** after being filtered, for each remaining pair (P_{tx}, R), the algorithm estimates the average current consumption I_{avg} using the model described above. This step accounts for:
  - Radio power consumption tables (from datasheet)
  - Message sizes
  - Current frequency allocation
  - PRR-based retransmission overhead

Priority Management

In bandwidth constrained situations, for example when it's necessary to select the lowest data rate due to weak link, a frequencies reduction is necessary. To maintain the integrity of essential information, the proposed framework includes a dynamic priority management module. Each ROS2 topic is assigned a priority class:

- **High priority:** for example LiDAR near obstacles, robot state diagnostics
- **Medium priority:** for example odometry, IMU data
- **Low priority:** debug messages, static sensor readings, telemetry

When the throughput demand exceeds the capacity of the selected data rate, the algorithm selectively reduces the publication frequency of lower-priority streams.

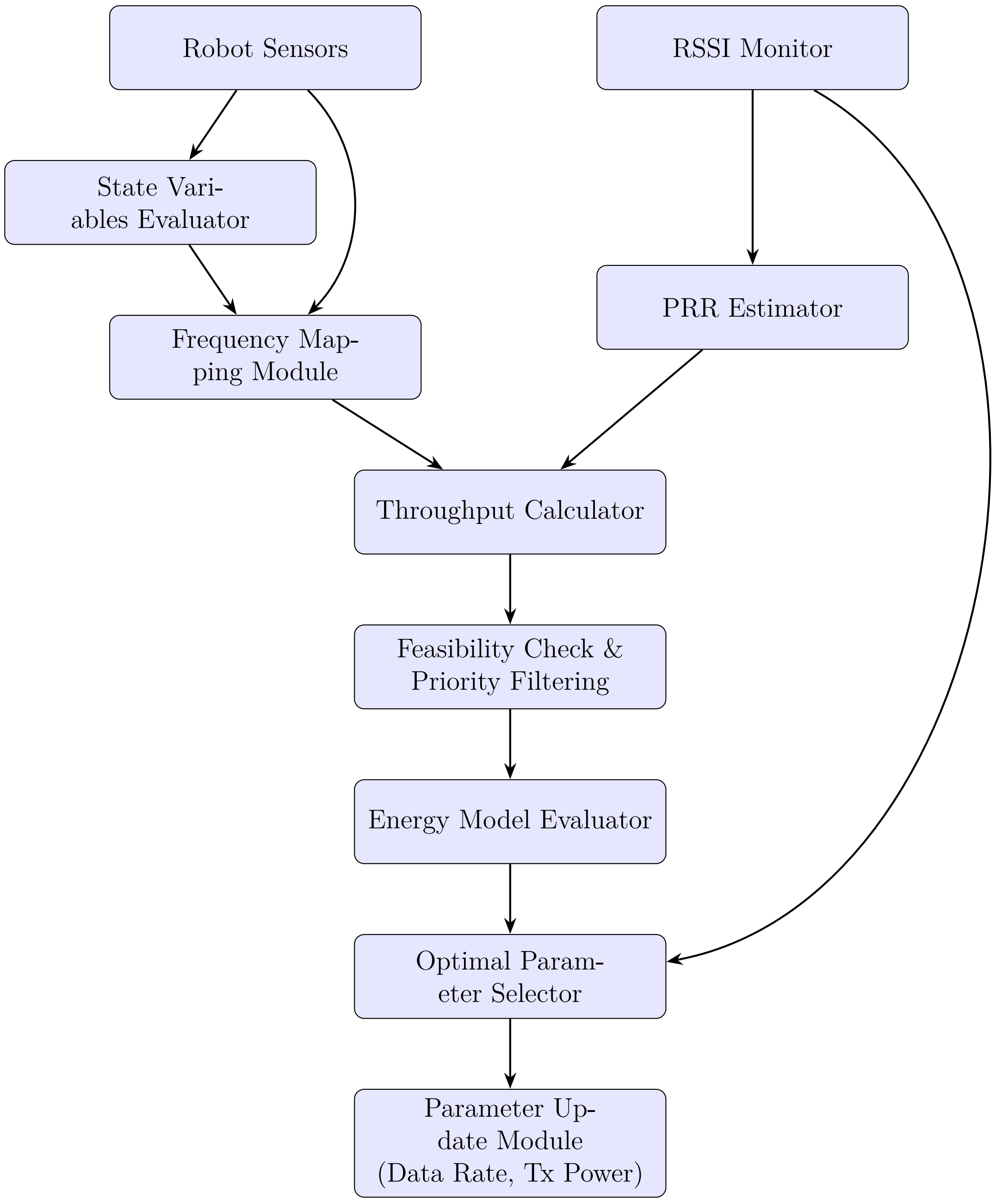

Robot Sensors
RSSI Monitor
State Variables Evaluator
Frequency Mapping Module
PRR Estimator
Throughput Calculator
Feasibility Check & Priority Filtering
Energy Model Evaluator
Optimal Parameter Selector
Parameter Update Module (Data Rate, Tx Power)

Figure 3.6: Adaptive Communication Control Loop.

3.5 Simulation-Based Validation

The preliminary validation of the adaptive control algorithm was conducted through a simulation phase, designed to emulate the main operating conditions expected in the real context, while maintaining a controlled and repeatable environment. The purpose of this phase was to test the control algorithm in realistic scenarios, considering the noisy behaviour of the RSSI, evaluating the impact of dynamic choices of data rate and transmission power in response to changes in the robotic state and wireless link quality.

3.5.1 Gazebo and ROS2 Simulation Environment

The simulation environment was developed using Gazebo integrated with ROS2. This environment enabled interaction between the robot and a virtual indoor space, allowing to test both LOS and NLOS conditions.

Environment Setup

As Gazebo map, a custom simulation layout, similar to the actual laboratory where real-world experiments are planned to be conducted and where the PRR-RSSI fitting models experiments were done, was designed. The simulated environment included rooms, long corridors, obstacles, and a fixed base station, representing the radio receiver, placed in the first room of the path. The spatial configuration was chosen to reflect typical indoor propagation conditions and to test communication performance under both LOS and NLOS conditions.

Robot Configuration and Sensor Simulation

The simulations employed a Clearpath Husky A200 differential-drive robot model equipped with a 3D LiDAR and an Inertial Measurement Unit (IMU), matching the hardware that will be used in the real-world tests. The robot published the following main ROS2 topics:

- `sensor_msgs/Imu`
- `sensor_msgs/PointCloud2`
- `nav_msgs/Odometry`

The robot was manually teleoperated along a path that covered the entire simulated laboratory, passing close to obstacles at varying distances. This behavior was essential to evaluate the adaptive system's ability to prioritize *PointCloud2* messages based on proximity to obstacles, calculated from LiDAR data.

Data Logging

During the entire the simulation, data was logged at 100 Hz into a .csv file for offline analysis. The following values were recorded:

- **Timestamps** – for time alignment

- **Distance from the base station** – computed in real-time using the robot's pose
- **IMU data** – linear acceleration and angular velocity
- **Distance to the closest obstacle** – extracted from LiDAR data
- **Message sizes** – of both pointcloud and odometry messages
- **Line-of-Sight (LOS) status** – computed using raycasting (RayShape) in Gazebo, to detect whether a direct path existed between the robot and the base station

3.5.2 RSSI Simulation Model

To evaluate the performance of the adaptive communication algorithm under realistic wireless conditions, a simulation of the Received Signal Strength Indicator (RSSI) was implemented, using as input data the distance values from the base station, the frequency of the radio communication, and the LOS/NLOS values. Instead of relying on ideal or simplified models, the model adopted combines:

- Distance-based path loss
- Slow fading (shadowing) effects
- Fast fading (multipath) effects
- A fixed degradation component to simulate areas with bad link quality, necessary to evaluate the controller near the point of link degradation.

Distance-Based Path Loss (ITU-R P.1238-12)

The main component of the RSSI simulation is the ITU-R P.1238-12 median basic transmission loss model 2.25. The model was applied using the recommended parameters for corridor environments, ensuring appropriate attenuation behavior in indoor settings. The path loss was computed for every timestamp using the real-time Euclidean distance between the robot and the base station.

Slow Fading (Shadowing)

To simulate shadowing, signal fluctuations caused by large static obstacles (walls, equipment), the shadowing component X_σ as a zero-mean Gaussian random variable 2.27 with a standard deviation $\sigma = 4.07$ for LoS condition and $\sigma = 7.63$ for NLoS condition, as reported in the *Corridor* pramaters in the Table 2.1.

Fast Fading (Multipath)

In indoor environments, fast fading due to multipath propagation can lead to rapid fluctuations of the signal level even over small distances. This effect was included based on the LOS status determined by the raycasting mechanism in Gazebo. In **LOS** conditions, the fading is modeled as a **Rician distribution** while in NLoS conditions, it is modeled as a **Rayleigh distribution**. The 2.28 distribution was used with $K = 0$ for Rayleigh fading, and $K = 6$ for Rician fading, since for indoor environments typically K values for Rician distribution are greater then 5.

Fixed Link Degradation

To ensure the presence of a significant signal degradation zone in the dataset a fixed degradation offset L_{offset} of 50dBm was manually added to the path loss formula. The value of L_{offset} was tuned experimentally to create a zone where the RSSI dropped close to (or below) the sensitivity threshold of the simulated receiver. This was essential in order to properly test the control algorithm.

3.5.3 PRR Simulation

To simulate the Packet Reception Radio (PRR), the RSSI values computed at each time step were used as input to one of three empirical PRR-RSSI models 3.4. For each time sample, the right model was selected according to the current data rate used by the robot, and the PRR was evaluated from the corresponding RSSI value. The computed PRR was then used to simulate the success or failure of a packet transmission using a Bernoulli trial: for each packet, a uniform random variable $u \in [0, 1]$ was drawn and if $u <$ PRR the packet was considered successfully received, otherwise not. This resulted in a boolean variable `packet_arrival_success`:

$$\texttt{packet\_arrival\_success} = \begin{cases} 1 & \text{if } u < \text{PRR} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Finally, the PRR over the entire simulation was computed as the ratio between the number of successfully received packets and the total number of packets sent:

$$\text{PRR}_{\text{sim}} = \frac{\sum_{i=1}^{N} \texttt{packet\_arrival\_success}_i}{N} \tag{3.6}$$

where N is the total number of packets transmitted during the simulation interval.

3.6 Real-World Experimental Validation

3.6.1 Physical Setup and Test Scenario

To complement the simulation-based validation and assess the actual deployability of the proposed adaptive communication system, a real-world experimental campaign was conducted. The primary goal of these tests was to verify the correct functionality of the complete system pipeline, including hardware interfaces, message serialization and deserialization, remote parameter configuration, and transmission over the real wireless channel, components that cannot be entirely validated in a simulated environment.

The test consisted of the same differential-drive mobile robot used in the simulation, a Clearpath Husky A200, equipped with a Velodyne VLP-16 3D LiDAR and the configuration Raspberry Pi 5 (running ROS2) + the Xbee SX Modem. The robot was configured to publish different topics, including PointCloud2 messages and IMU data which were serialized and transmitted via the XBee radio operating in API mode. On the receiving side, an identical the configuration Xbee SX Modem + laptop was used, which received the byte arrays stream, deserialized the messages, and republished them to its local ROS2 environment.

Due to internal policies, it was not possible to perform outdoor tests, and the robot's movements were confined to a large indoor laboratory room, the same environment normally used for routine robot operations. It was not possible to access the corridors or the other parts of the laboratory. However, in order to simulate realistic degradation of the wireless channel, the receiving laptop (with its XBee module) was gradually moved away from the room, introducing a mix of LOS and NLOS conditions, with an initial LoS condition due to the presence of a long corridor in front of the door of the room, simulating practical variations in signal quality.

To ensure the repeatability and precise analysis of the results, ros2bag recordings were performed simultaneously on both the Raspberry Pi 5 (transmitter side) and the laptop (receiver side). The clocks of both devices were synchronized using Chrony, ensuring sub-millisecond accuracy and allowing precise alignment of the original and reconstructed message streams during post-analysis.

Chapter 4

Results

4.1 Simulations Results

This section presents the results of the simulation-based evaluation of the adaptive communication control algorithm. Multiple variations of the proposed algorithm and fixed-parameters configurations were tested, for each strategy, ten repeated runs were performed on the same input dataset, to ensure statistical significance of the results and reduce the effect of randomness in RSSI noise.

4.1.1 Overview of Evaluation Metrics

The main performance indicators used to evaluate each strategy are:

- **Packet Reception Ratio (PRR):** The ratio between total successfully received packets and total transmitted packets.
- **Average Current Absorption:** The mean of the instantaneous current consumption.

Additional metrics were also recorded and used for correlation and others analysis:

- Timestamp
- Distance from base station
- IMU data (ax, ay, az, gx, gy, gz)
- RSSI
- Internal movement state indicator
- Obstacle distance
- Message sizes (pointcloud and odometry)
- Transmission settings (TxPower, Datarate)

- Effective topic frequencies
- T_{on} (time when the radio is transmitting)
- Original topic frequencies(before the control on the max radio throughput)
- Boolean flag indicating packet arrival success

These secondary metrics support a deeper understanding of the control logic behavior and wireless link dynamics.

4.1.2 Summary of Main Performance Results

The following table reports the average PRR and average current absorption across 10 trials for each tested strategy.

Table 4.1: Summary of PRR and Average Current Absorption for All Tested Strategies

Label	**Strategy**	**RSSI Filter Window**	**RSSI Thresholds Offset** [dBm]	**Avg Current** [mA]	**PRR** [%]
A	Proposed	10 samples	+0dBm	27,71	86,26
B	Proposed	5 samples	+0dBm	28,62	87,31
C	Proposed	10 samples	+3dBm	30,36	87,79
D	Proposed	5 samples	+3dBm	30,72	88,83
E	Proposed	10 samples	+7dBm	36,70	90,39
F	Proposed	5 samples	+7dBm	37,35	90,81
G	Proposed	10 samples	+10dBm	45,71	92,32
H	Proposed	5 samples	+10dBm	52,54	93,25
I	Sodhro's Algo [20] DataRate=250000	-	-	39,13	87,15
J	ARF [12] TxPower=30	-	-	80,23	99,40
K	AARF [12] TxPower=30	-	-	80,45	99,22
L	AMRR [12] TxPower=30	-	-	80,32	99,95
M	Onoe's Algo [12] TxPower=30	-	-	81,17	99,97
N	Fixed Parameters: TxPower=30 DataRate=250000	-	-	80,23	99,91
O	Fixed Parameters: TxPower=27 DataRate=250000	-	-	57,07	99,46
P	Fixed Parameters: TxPower=20 DataRate=250000	-	-	29,74	85,25
Q	Fixed Parameters: TxPower=30 DataRate=110000	-	-	181,32	99,82
R	Fixed Parameters: TxPower=27 DataRate=110000	-	-	128,93	99,18
S	Fixed Parameters: TxPower=20 DataRate=110000	-	-	66,39	92,45
T	Fixed Parameters: TxPower=30 DataRate=10000	-	-	685,38	100
U	Fixed Parameters: TxPower=27 DataRate=10000	-	-	487,3	99,58
V	Fixed Parameters: TxPower=20 DataRate=10000	-	-	251,30	97,58

Table 4.2: Summary of PRR and Average Current Absorption for All Tested Strategies with α multiplied by 1.5

Label	**Strategy**	**RSSI Filter Window**	**RSSI Thresholds Offset [dBm]**	**Avg Current [mA]**	**PRR [%]**
A	Proposed	10 samples	+0dBm	106,41	70,43
B	Proposed	5 samples	+0dBm	112,11	70,94
C	Proposed	10 samples	+3dBm	128,73	78,28
D	Proposed	5 samples	+3dBm	149,78	79,23
E	Proposed	10 samples	+7dBm	241,25	88,91
F	Proposed	5 samples	+7dBm	234,80	87,34
G	Proposed	10 samples	+10dBm	252,32	91,27
H	Proposed	5 samples	+10dBm	259,06	90,76
I	Sodhro's Algo [20] DataRate=250000	-	-	63,48	73,19
J	ARF [12] TxPower=30	-	-	200,73	87,40
K	AARF [12] TxPower=30	-	-	226,12	87,22
L	AMRR [12] TxPower=30	-	-	195,36	88,95
M	Onoe's Algo [12] TxPower=30	-	-	274,32	91,17
N	Fixed Parameters: TxPower=30 DataRate=250000	-	-	80,23	78,50
O	Fixed Parameters: TxPower=27 DataRate=250000	-	-	55,34	60,32
P	Fixed Parameters: TxPower=20 DataRate=250000	-	-	21,44	32,56
Q	Fixed Parameters: TxPower=30 DataRate=110000	-	-	179,90	86,34
R	Fixed Parameters: TxPower=27 DataRate=110000	-	-	124,82	69,27
S	Fixed Parameters: TxPower=20 DataRate=110000	-	-	47,98	35,83
T	Fixed Parameters: TxPower=30 DataRate=10000	-	-	684,85	94,28
U	Fixed Parameters: TxPower=27 DataRate=10000	-	-	484,30	82,65
V	Fixed Parameters: TxPower=20 DataRate=10000	-	-	233,76	42,54

4.1.3 PRR vs Current Trade-off

To show the trade-off between reliability and energy efficiency, the following scatter plots compares the PRR and average current for all strategies.

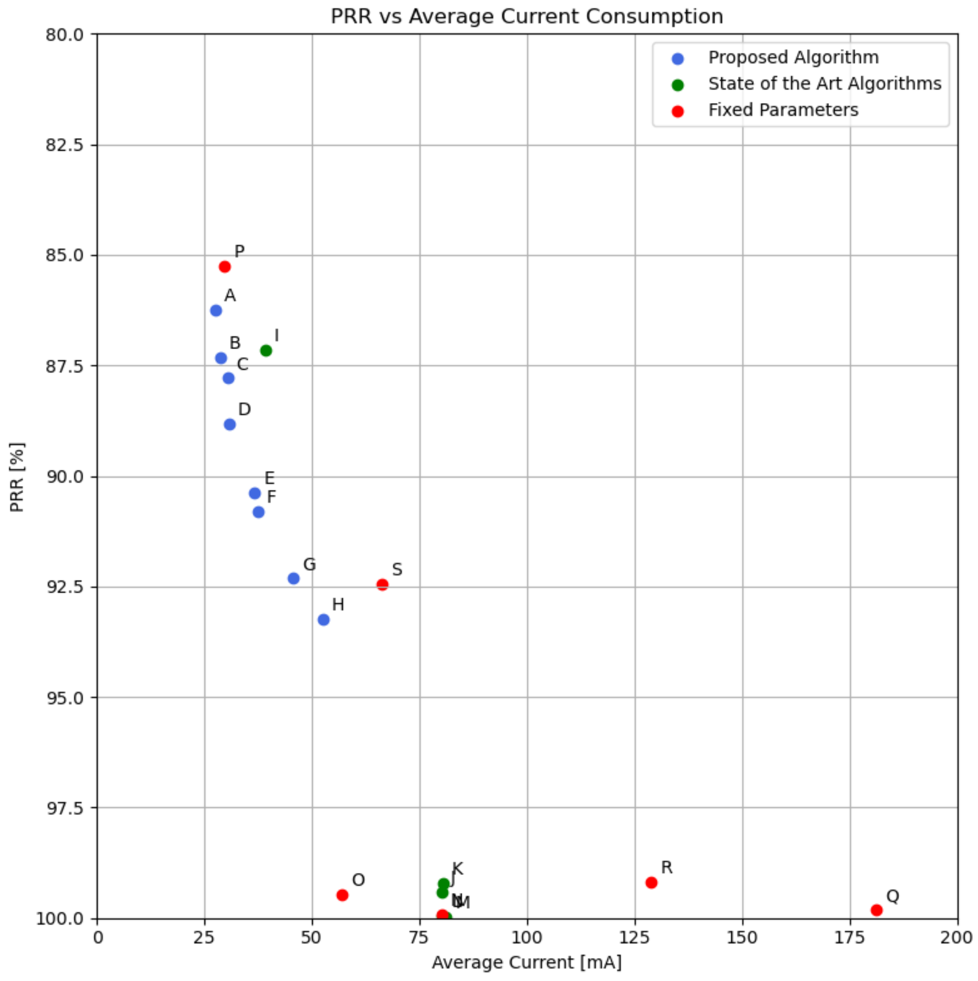


Figure 4.1: PRR vs Average Current Consumption

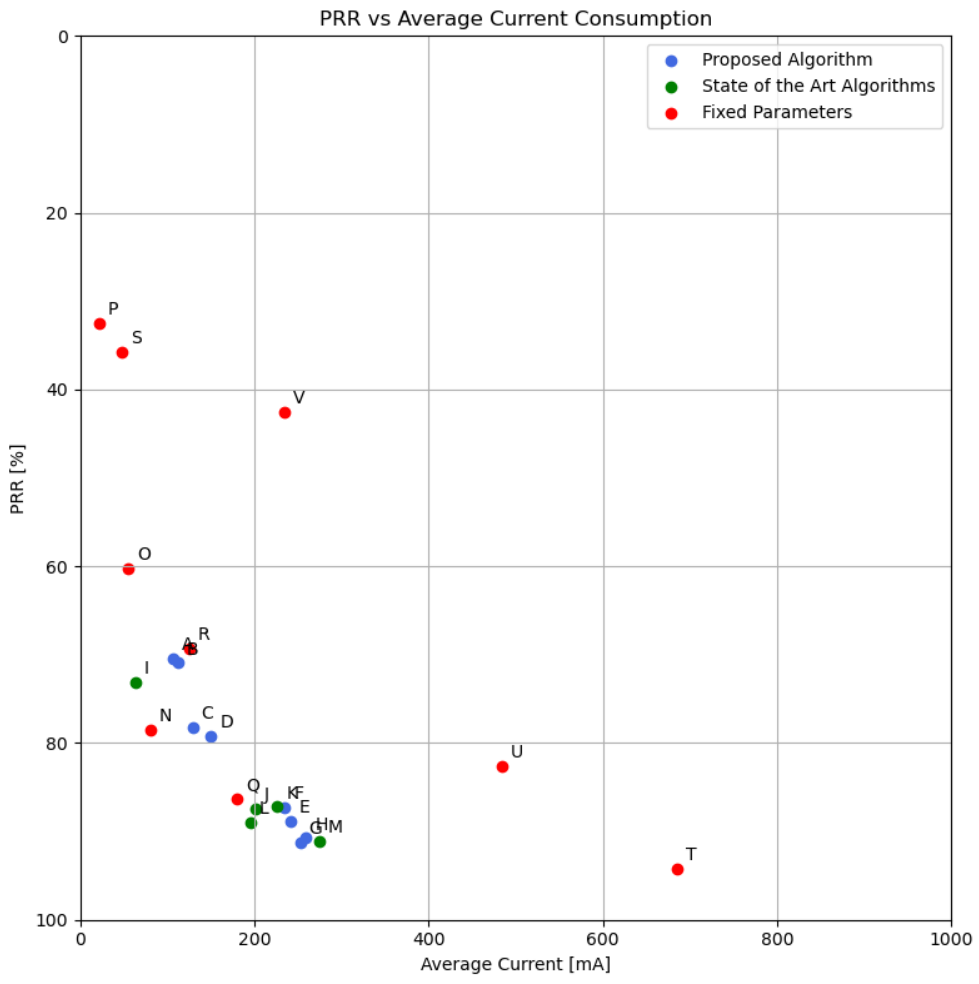


Figure 4.2: PRR vs Average Current Consumption with with α multiplied by 1.5

This plot clearly shows how adaptive strategies tend to balance energy and performance, while fixed parameter approaches achieve high PRR at the cost of high energy usage or vice versa.

4.2 Real-world experiments results

The experimental results confirmed the correct behaviour of all system components in realistic conditions. The core functionality of the communication stack, including ROS2 message serialisation, transmission, reception, and deserialisation, was validated end-to-end. The system successfully reconstructed and visualised PointCloud2 messages in RViz on the receiver laptop, without noticeable degradation in message structure, size, or visualisation fidelity.

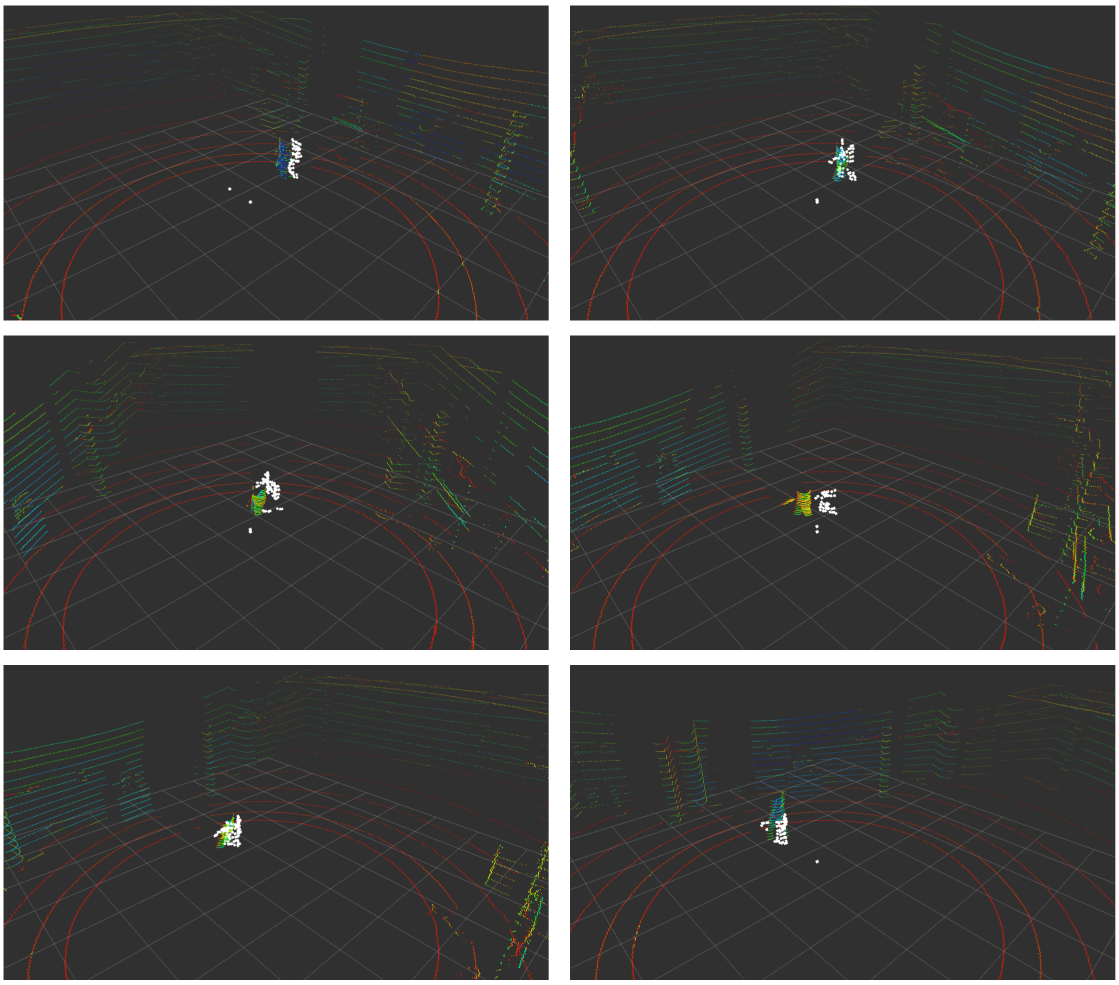


Figure 4.3: Original PointCloud (colored) vs Reconstructed PointCloud (white)

To demonstrate this, Figures 4.3, shows a side-by-side comparison between the original PointCloud2 (recorded on the robot) and the deserialised PointCloud2 (on the receiver), overlaid in the same reference frame. It is noticeable that the reconstructed message experiences a delay, which can be attributed both to the low frequency selected by the control system and to the latency caused by the large message size.

For example, a `sensor_msgs/PointCloud2` message containing 100 points with only the `x`, `y`, and `z` fields occupies approximately **1282 bytes**, computed as follows: each point includes three 32-bit floating-point values (12 bytes per point), so the binary data array contributes $100 \times 12 =$ **1200 bytes**; metadata fields, including header (28 bytes), dimensions (2 fields $\times$ 4 bytes), step sizes (2 $\times$ 4 bytes), flags (2 $\times$ 1 byte), and the `fields[]` array (3 fields $\times$ 12 bytes), account for an additional $28{+}4{+}4{+}4{+}4{+}1{+}1{+}36 =$ **82 bytes**, yielding a total of $1200 + 82 =$ **1282 bytes**.

Given that the maximum payload size per wireless packet is **256 bytes total**, and this already includes a **5 byte custom header**, the effective payload available for actual message data is:

$$256 \text{ bytes} - 5 \text{ bytes} = \textbf{251 bytes per packet} \tag{4.1}$$

To transmit 1282 bytes, the message must be fragmented into:

$$\left\lceil \frac{1282}{251} \right\rceil = \textbf{6 packets} \tag{4.2}$$

Each packet carries an additional **18 bytes** of protocol overhead, so the total number of bytes transmitted over the air is:

$$6 \times (256 + 18) = 6 \times 274 = \textbf{1644 bytes} \tag{4.3}$$

In bits:

$$1644 \times 8 = \textbf{13,152 bits} \tag{4.4}$$

Considering the three radio configurations with effective throughput values of 7 kbps, 60 kbps, and 95 kbps, the resulting transmission latency for the entire message is:

- **At 10 kbps (7 kbps throughput)**:
$$\frac{13{,}152}{7{,}000} \approx \textbf{1.88 seconds} \tag{4.5}$$
- **At 110 kbps (60 kbps throughput)**:
$$\frac{13{,}152}{60{,}000} \approx \textbf{0.22 seconds} \tag{4.6}$$
- **At 250 kbps (95 kbps throughput)**:
$$\frac{13{,}152}{95{,}000} \approx \textbf{0.14 seconds} \tag{4.7}$$

This analysis shows that, due to fragmentation and per-packet overhead, the total transmitted data significantly exceeds the raw message size, causing substantial transmission delays. These delays explain the latency observed in the reconstructed point cloud, especially at lower throughputs.

In addition, the adaptive control algorithm was observed to behave as expected:

- The frequency of non-critical topics was reduced as the communication channel degraded.
- Transmission power was increased and data rate decreased automatically when the RSSI dropped below the selected thresholds.
- In high-quality channel conditions, the system returned to low power and high data rate modes to improve efficiency.

These dynamic behaviours were consistent with the control strategy defined in the simulation environment.

4.3 Discussion

The results obtained from simulation-based tests clearly highlight the trade-offs, between energy saving and reliability in the design of communication strategies for mobile robots. In particular, while fixed-parameter configurations can achieve excellent performance under certain conditions, they are not flexible across different environments and dynamic network conditions. For example, the fixed configurations with highest transmission power and lowest datarate (**T**) achieved the highest Packet Reception Ratio (PRR), but at the cost of significantly higher average current absorption. On the other hand, the fixed configuration with lowest power and highest datarate (**P**) achieved more energy-efficient but suffered from substantial packet loss in areas with increased path loss or non-line-of-sight (NLOS) conditions. Moreover, while in the first test the best performance were achieved by the **(O)** (27-250000), in the second test it didn't perform well. This because in the second test, due to a worse signal degradation environment, a high transmission power and low data rate configuration was required to maintain a good link quality. This demonstrates that fixed strategies are highly environment-dependent: a configuration that performs well in a given environment might completely fail in another with different propagation characteristics, obstacle density, or mobility patterns. In contrast, adaptive communication control algorithm offers a robust and flexible alternative, capable of dynamically adjusting communication parameters based on real-time context. This adaptability enables the system to balance reliability and energy efficiency, maintaining communication quality while minimising unnecessary energy waste. However, single-parameter adaptive algorithms, such as ARF, AARF, AMRR and Onoe's algorithm, that adapts data rate, are environment-dependent. For example, they performed well in the second test, but due to the fixed transmission power set at 30 dBm, they waste energy in situations where it is not needed, such as in the first test, which contains a greater number of areas with a good link. Instead, Sodhro's algorithm performed better in the first test than in the second, because in the second test, with a fixed data rate of 250 kbps, in some cases it was not enough to set the maximum transmission power, but it was also necessary to reduce the data rate to increase the quality of the connection. Experiments with different configurations of the control algorithm revealed how tuning internal parameters, such as the RSSI threshold offset, directly impacts performance. As shown in Figure 4.1 and Figure 4.2, increasing the threshold offset leads to improved PRR due to more aggressive responses to weak links (e.g., higher transmission power or lower datarates), but it also results in higher current absorption. This illustrates the trade-off within the adaptive control approach, which can be tuned according to the application's priorities, favoring either energy savings or communication robustness. Importantly, the adaptive algorithm proposed maintains acceptable performance across all tested environments, this underscores a key strength of the approach: it does not aim to maximize a single metric in a specific scenario but rather to provide consistently good performance across diverse and unpredictable conditions. In conclusion, these findings support the idea that adaptive communication control is not only beneficial but essential for autonomous systems operating in real-world, dynamic environments. The capacity to react to environmental changes, network degradation, and varying task demands makes adaptive communication strategies far more suitable for mobile robotics than fixed-parameter approaches, especially when long-term reliability and energy autonomy are critical.

Bibliography

[1] Renaud Dubé et al. “An online multi-robot SLAM system for 3D LiDARs”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1004–1011.

[2] Roger L Freeman. *Radio system design for telecommunications*. John Wiley & Sons, 2007.

[3] Jennifer Gielis, Ajay Shankar, and Amanda Prorok. “A critical review of communications in multi-robot systems”. In: *Current robotics reports* 3.4 (2022), pp. 213–225.

[4] Kimberly Hambuchen et al. “Towards Supervising Remote Dexterous Robots Across Time Delay”. In: *ICRA06*. 2006.

[5] Chung-Wen Hung et al. “Transmission power control in wireless sensor networks using fuzzy adaptive data rate”. In: *Sensors* 22.24 (2022), p. 9963.

[6] Andrés C Jiménez, Vicente García-Díaz, and Sandro Bolaños. “A decentralized framework for multi-agent robotic systems”. In: *Sensors* 18.2 (2018), p. 417.

[7] Steven Lanzisera, Ankur M Mehta, and Kristofer SJ Pister. “Reducing average power in wireless sensor networks through data rate adaptation”. In: *2009 IEEE International Conference on Communications*. IEEE. 2009, pp. 1–6.

[8] Shan Lin et al. “ATPC: Adaptive transmission power control for wireless sensor networks”. In: *ACM Transactions on Sensor Networks (TOSN)* 12.1 (2016), pp. 1–31.

[9] A. G. Longley. *Radio Propagation in Urban Areas*. Technical Report OT Report 78-144. Also known as OT Report 78-144, Institute for Telecommunication Sciences. Washington, D.C.: U.S. Department of Commerce, 1978. URL: https://its.ntia.gov/publications/download/ot-78-144.pdf.

[10] Jing Mao et al. “Revisiting link quality metrics and models for multichannel low-power lossy networks”. In: *Sensors* 23.3 (2023), p. 1303.

[11] Kristian Mella. “Theory, Simulation and Measurement of Wireless Multipath Fading Channels”. MA thesis. Institutt for elektronikk og telekommunikasjon, 2007.

[12] MA Mohamed, WM Bahget, and SS Mohamed. “A performance evaluation for rate adaptation algorithms in IEEE 802.11 wireless networks”. In: *International Journal of Computer Applications* 99.4 (2014), pp. 54–59.

[13] Michael S Mollel and Michael Kisangiri. "An overview of various propagation model for mobile communication". In: *Proceedings of the 2nd Pan African International Conference on Science, Computing and Telecommunications (PACT 2014)*. IEEE. 2014, pp. 148–153.

[14] R Nagaraj et al. "LQI-Per-Based Range Adaptation For Enhanced Communication Range In Wireless Networks." In: *Journal of Namibian Studies* 33 (2023).

[15] Eunjeong Park et al. "AdaptaBLE: Adaptive control of data rate, transmission power, and connection interval in bluetooth low energy". In: *Computer Networks* 181 (2020), p. 107520.

[16] Michał Pielka et al. "Adaptive Data Transmission Algorithm for the System of Inertial Sensors for Hand Movement Acquisition". In: *Sensors* 22.24 (2022), p. 9866.

[17] Iliar Rabet et al. "Actor: adaptive control of transmission power in RPL". In: *Sensors* 24.7 (2024), p. 2330.

[18] *Recommendation ITU-R P.1238-12: Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 450 GHz*. ITU-R Recommendation. Version 12. International Telecommunication Union Radiocommunication Sector (ITU-R). URL: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1238-12-202308-I!!PDF-E.pdf.

[19] *Recommendation ITU-R P.526-15: Propagation by Diffraction*. ITU-R Recommendation. Version 15. International Telecommunication Union Radiocommunication Sector (ITU-R). URL: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.526-15-201910-I!!PDF-E.pdf.

[20] Ali Hassan Sodhro, Ye Li, and Madad Ali Shah. "Energy-efficient adaptive transmission power control for wireless body area networks". In: *IET communications* 10.1 (2016), pp. 81–90.

[21] Javad Vazifehdan et al. "An analytical energy consumption model for packet transfer over wireless links". In: *IEEE Communications Letters* 16.1 (2011), pp. 30–33.