Implementation of the Hamming code for the detection and correction of errors in a telerobotic system using an industrial communication protocol

Hugo Marcelo Torres Salamea
Electronics Engineering School
Universidad del Azuay
Cuenca - Ecuador
htorres@uazuay.edu.ec

Dalton Demetrio Toledo Torres
Electronics Engineering School
Universidad del Azuay
Cuenca - Ecuador
dtole90@gmail.com

Pablo Dario Urgilés Cardenas
Electronics Engineering School
Universidad del Azuay
Cuenca - Ecuador
dario_urgiles@hotmail.com

Claudio Urrea Oñate
Depart of Electrical Engineering
Universidad de Santiago de Chile
Santiago de Chile - Chile
claudio.urrea@usach.cl

***Abstract*— Data transmission at the industry level encounters problems due to different factors such as noise and electromagnetic interference. This project presents an application for the control of a 3-DOF fault-tolerant robot through the implementation of an encoder and a Hamming decoder (7.4). Data transmission is conducted using a wireless system with XBee modules, while the Modbus RTU Master-Slave industrial communication protocol is employed for communication with the manipulator robot.**

To implement the master, a PC where the Hamming encoder responsible for placing the redundancy bits is installed, and an Xbee module for sending the information through the Modbus protocol are used. For the slave, a second Xbee module and a pcDuino board are used, where the Hamming decoder is located —which is responsible for receiving the encoder's information and perform the detection and correction of errors of the data sent by the master— to then be sent to the Expander Pi board, where the respective PWMs that manage the movement of the manipulator robot are located.

Keywords*— *Hamming, Xbee, Modbus, Manipulator, Wireless Communication

I. Introduction

There is a high index of automation in the industry. This has led technology and industrial communications to advance exponentially, facilitating the remote control and supervision of the production processes [1]. Robotics is a field within industrial automation, which for decades has had a strong impact not only on the manufacturing sector but also on society. Robotics substitutes human labor in dangerous tasks, some of them impossible for people.

Data transmission in industrial areas is subject to different interferences from heat, temperature, or electromagnetic fields, which interrupt or disturb the sending of information. Especially in the case of binary data, these changes can modify the transmission of information.

In the industrial field and particularly in robot manipulation, the concepts related to fault tolerance have greater application and importance because, currently, all robotic systems are manipulated by informatic systems where the transmission of information must be reliable and effective. Therefore, it is crucial that data frames in industrial communications have redundant information for error detection and correction.

Nowadays, wireless communications such as Zigbee, WiFi, Bluetooth, and the XBee protocol are widely used in the field of robotics, specifically in the mining sector, where wireless remote control of manipulator robots is necessary due to the dangerous and difficult access to certain terrains by humans. These communications are also applied to mobile robots for terrain exploration [2].

The objective of this work is to analyze and implement the Hamming code in the Modbus industrial protocol as a method for detecting and correcting errors in a 3-DOF manipulator robot, by means of XBee-based wireless communication to access difficult areas and, above all, eliminate the cable transmission.

II. Research Framework

A. Hamming code

The Hamming code is a parity check code that allows for the correction of individual errors and the detection of double errors [3]. A linear Hamming code (n, k) with 3 or more bits of redundancy satisfies the relation: 2m-1> n and is expressed by the following equations:

$$n = 2^q - 1 \tag{1}$$

$$k = n - q \tag{2}$$

where:
q: Redundancy bits (error checking and correction).
n: Total transmission bits (information bits + redundancy bits for error detection and correction).
k: Information bits

The 3 bits of redundancy for a Hamming code (7.4), are obtained according to the following equations [4]:

$$q_3 = k_1 \text{ XOR } k_2 \text{ XOR } k_3 \tag{3}$$
$$q_2 = k_1 \text{ XOR } k_2 \text{ XOR } k_4 \tag{4}$$
$$q_1 = k_1 \text{ XOR } k_3 \text{ XOR } k_4 \tag{5}$$

The Hamming coding (7,4) can be seen in Figure 1

Information Bits				Redundancy Bits		
k_4	k_3	k_2	k_1	q_3	q_2	q_1

Figure. 1: Hamming encoder (7,4)

B. *XBee*

The XBee technology consists of modules that provide a wireless medium for communication between devices. The XBee module uses the IEEE 802.15.4 communication protocol. With these devices peer-to-peer and multipoint networks can be designed based on the Zigbee protocol [4].

One of the advantages of XBee technology over other wireless communication media is its capacity to work with multiple devices intercommunicated and in different mesh topologies, where high data traffic, low latencies low and minimum energy consumption are required. Furthermore, XBee features a sleep mode, which saves a big amount of energy when the device is not being used [5]

The XBee S2 modules are the most common due to their data rate characteristics (250 Kbps), operate in a 2.4 GHz band and have a range of 40 meters indoors and up to 120 meters in the open field, enabling a very simple communication between different embedded systems to manipulate different robots or any industrial device [2].

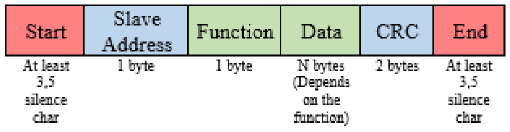
C. *Modbus*

Modbus is an industrial communication protocol used to establish Master-Slave connections. It transmits digital and analog signals between them. The protocol uses the RS-232 standard, which defines the physical characteristics of the connection. In addition, communication can be point to point with a single slave or a general message (broadcast) [6].

The Modbus network is based on a control field bus, which has 1, 2 and 7 OSI model levels that correspond to the physical, link and application layers, respectively. Their topology is linear, with a master that controls access to the medium and monitors network operation, and one or more devices that act as slaves.

Communication is performed in an asynchronous serial way, under the RS-232 or RS-485 standards for semi-duplex links and RS-422 for duplex links. This protocol has two ASCII and RTU transmission modes for the intercommunication of messages between different devices in the network. These messages, known as frames, contain the data necessary to recognize the origin and destiny of each message placed on the bus by any of the devices. Their length depends on the transmission mode, and is bounded by a maximum of 256 characters [6].

According to the MODBUS standard, for speeds up to 19,200 bps, the time between frames must be at least 3.5 times the duration of a character, and a fixed time of 1.75ms is recommended for higher speeds Figure 2.



Start	Slave Address	Function	Data	CRC	End
At least 3,5 silence char	1 byte	1 byte	N bytes (Depends on the function)	2 bytes	At least 3,5 silence char

Figure 2: Structure of the frame in RTU mode

A silence character has the duration of a data byte sent by the medium, but does not carry data, and its duration (T) depends on the speed (Vt) and the number of bits used for its coding (N) according to the following equation:

$$T = N/Vt \tag{6}$$

III. Implementation of the System

A. *Hardware Implementation*

The hardware implementation for the control of a 3-DOF industrial manipulator robot through wireless communication based on the Modbus industrial protocol was conducted employing a PC operating as the master, a pcDuino 3 card working as a slave and an Expander Pi card where the different PWMs were located to control the 3 servo motors of the robot.

The pcDuino 3 is a mini PC of high performance and low cost. It works with operating systems such as Ubuntu, Linux and Android Additionally, it has Analog Digital Converters (ADC) and Pulse Width Modulation (PWM), as well as an HDMI output for TV or Monitor

For wireless communication between the master and the slave, two XBee S2 wireless modules that work in the ISM bands (Industrial, Scientific and Medical) 902-928 MHz, 2.4-2.4835 GHz, 5.725-5.85 GHz was employed as wireless networks. These bands are internationally reserved for non-commercial and unlicensed use in industrial, scientific, and medical areas [7].

To connect the XBee wireless transmitter to the PC, a SparkFun XBee Explorer USB module was used (Figure 3a), and the connection of the XBee receiver to the mini PC was performed using the SparkFun XBee Explorer Regulated module (Figure 3b).

Figure. 3a: SparkFun XBee Explorer USB [8]

Figure. 3b: SparkFun XBee Explorer Regulated [8]

The block diagram of the control system for the 3-DOF robotic manipulator built with the XBee technology and Modbus industrial protocol is presented in Figure 4.

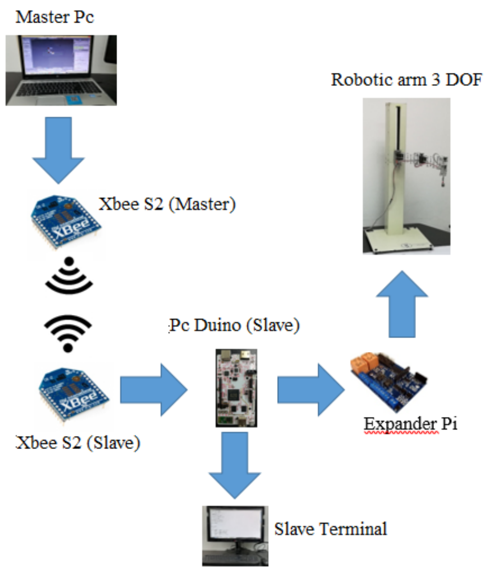


Figure. 4: System block diagram

B. Implementation of the Hamming encoder and decoder in the Modbus protocol for the control of the 3-DOF robot

The implementation of the Hamming encoder and decoder for the communication between the master and the slave through the Modbus protocol was conducted using different open source programs such as:

Python

Python is an easy to learn and use programming language. In addition, it has a wide expression range, which means that as a general rule, it makes it possible to write less code lines than other languages such as C ++ and Java.

Modbus-tk

Modbus Test Kit or Modbus-tk is an implementation of the Modbus protocol in Python that enables work in Modbus TCP and Modbus RTU modes; it can be used for testing both master and slave.

Blender

Blender is a software for creating content in 2D and 3D. This software offers a wide variety of functions for texturing, modeling, lighting, animating, and post-processing video. Additionally, its open architecture offers extensibility with other software such as Python [9].

B1. Implementation of the Hamming encoder (7,4) with the Modbus protocol for the control of the 3-DOF robot (Master)

For the implementation of the master, the Blender software was used, which contained the 3D animation of the 3-DOF manipulator robot, as shown in Figure 5.

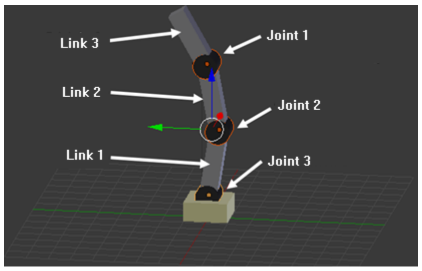


Figure. 5: 3D robotic arm with 3 degrees of freedom

In addition to the animation, the Python language was necessary to program the communication between master and slave using the Modbus protocol, for which the APIs of Modbus tk, the same that can work as a master, were employed, as shown in Figure 6.

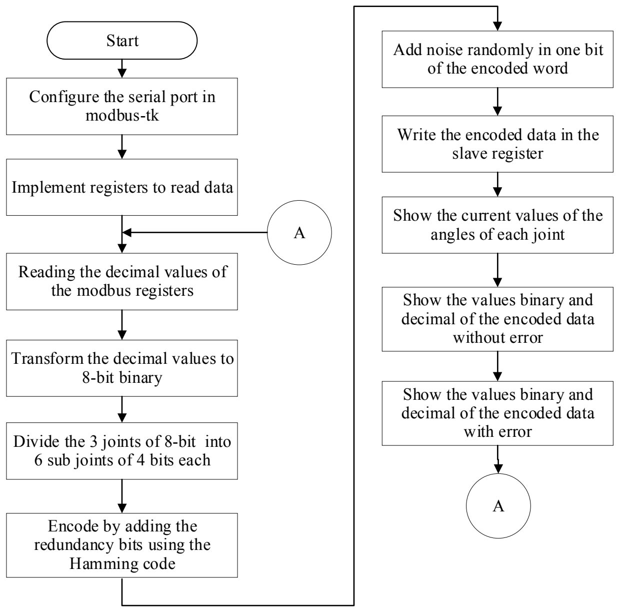

Start
Configure the serial port in modbus-tk
Implement registers to read data
A
Reading the decimal values of the modbus registers
Transform the decimal values to 8-bit binary
Divide the 3 joints of 8-bit into 6 sub joints of 4 bits each
Encode by adding the redundancy bits using the Hamming code
Add noise randomly in one bit of the encoded word
Write the encoded data in the slave register
Show the current values of the angles of each joint
Show the values binary and decimal of the encoded data without error
Show the values binary and decimal of the encoded data with error
A

Figure.6: Diagram of the Hamming encoder (7,4) with Modbus protocol for the control of the 3-DOF robot (Master)

B2. Implementation of the Hamming decoder (7,4) with the Modbus protocol for the control of the 3-DOF robot (slave)

For the implementation of the Hamming decoder in the slave, Python language, the same in the PcDuino 3 was used. Python is also in charge of leading the PWMs of the Expander Pi card for each servomotor in the manipulator robot. Figure 7 shows the flowchart for receiving data sent from the master as well as the block to be manipulated from the PcDuino, the robotic arm that acts as a Modbus slave.

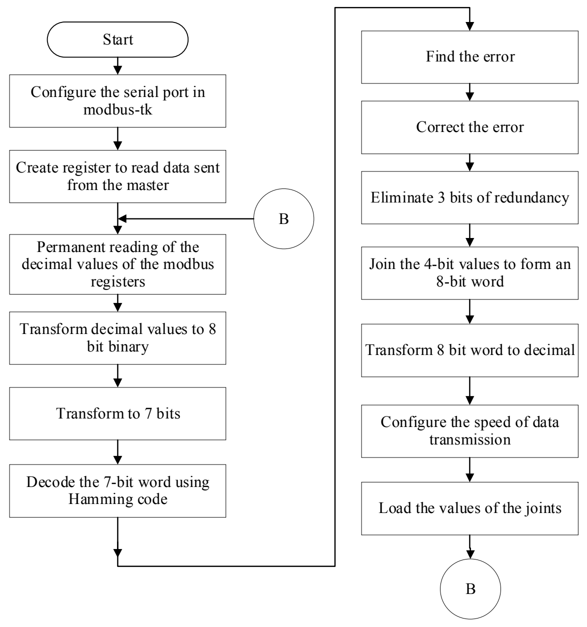

Start
Configure the serial port in modbus-tk
Create register to read data sent from the master
B
Permanent reading of the decimal values of the modbus registers
Transform decimal values to 8 bit binary
Transform to 7 bits
Decode the 7-bit word using Hamming code
Find the error
Correct the error
Eliminate 3 bits of redundancy
Join the 4-bit values to form an 8-bit word
Transform 8 bit word to decimal
Configure the speed of data transmission
Load the values of the joints
B

Figure. 7: Diagram of the Hamming decoder (7,4) with Modbus protocol for the control of the 3-DOF robot (Slave)

C. *System implementation*

The implementation of the fault tolerant system using Hamming coding for the communication between the master and the slave using the Modbus protocol and with wireless communication based on XBee technology can be seen in Figure 8.

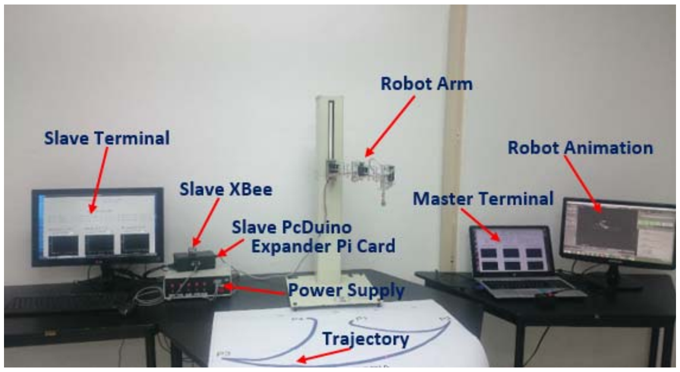


Figure. 8: Fault tolerant XBee communication system between Modbus master and slave

The role of master is performed by the Blender software installed on a PC and an XBee S2 module. The software used for programming the master is Python and runs on Ubuntu, which is a Linux based operating system distributed as a free software.

The slave function is executed by a PC and a pcDuino 3 card where the Modbus tk software is located. This card is also responsible for controlling the PWMs of the robot servo motors using the Expander Pi card.

The link between the master and the slave is made by means of the Modbus protocol, using the Python Modbus tk APIs and a PcDuino 3 card.

IV. Tests and Results

A. *Visualization of the Modbus frame times*

To obtain the respective transmission and response frames of a Modbus communication between a PC (Master) and a Modbus slave, function 01 was used, which is responsible for reading the state of the coils. The waveforms that were obtained from the communication using XBee technology are within the times that characterize the Modbus protocol. The Modbus sending and response frames are shown in Figure 9 and 10.

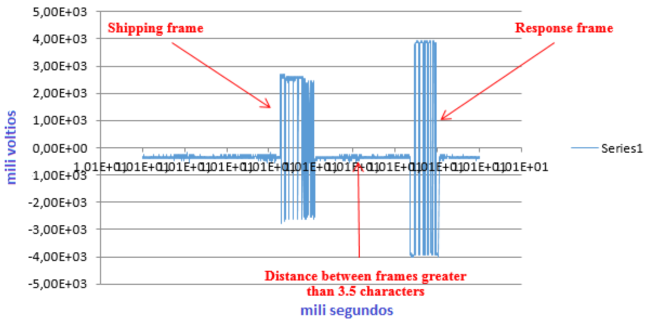


Figure. 9: Display of the Modbus sending and response frames of function 01

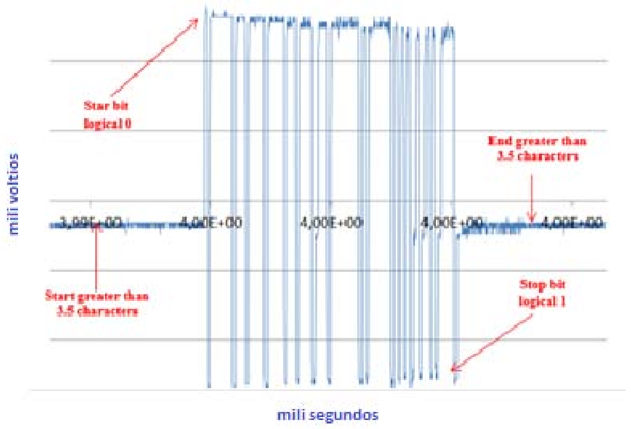
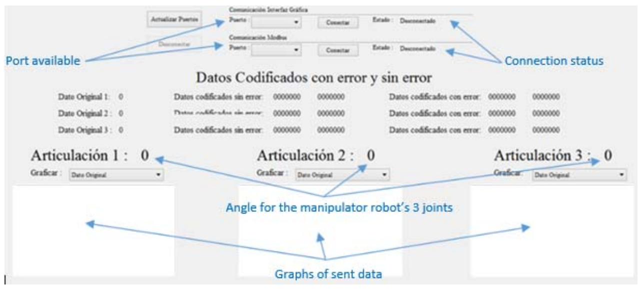


Figure. 10: Characteristics of the frames from Shipping

B. *Graphical interface for obtaining results between the encoder (master) and Hamming decoder (slave)*

To verify the results of the communication a graphical interface was developed in Python using the Qt Designer tool for the master and the slave as can be seen in the Figures 11 and 12.



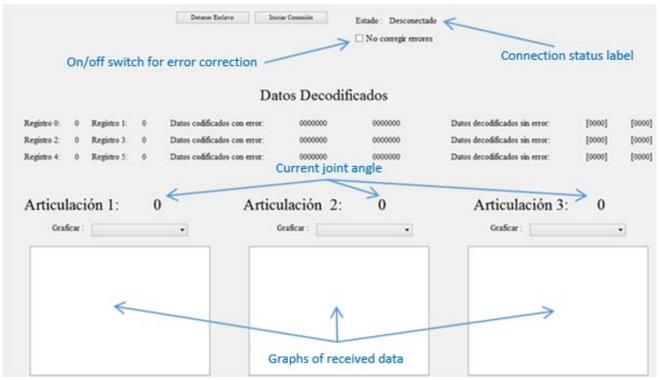
Estado : Desconectado
Estado : Desconectado

Datos Codificados con error y sin error

Dato Original 1:	0	Datos codificados sin error:	0000000	0000000	Datos codificados con error:	0000000	0000000
Dato Original 2:	0	Datos codificados sin error:	0000000	0000000	Datos codificados con error:	0000000	0000000
Dato Original 3:	0	Datos codificados sin error:	0000000	0000000	Datos codificados con error:	0000000	0000000

Articulación 1 : 0
Articulación 2 : 0
Articulación 3 : 0

Figure. 11: Graphic interface of the master



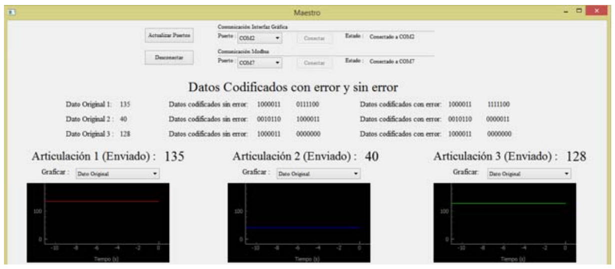
```
Estado :  Desconectado
☐ No corregir errores

Datos Decodificados
Registro 0:  0  Registro 1:  0  Datos codificados con error:  0000000  0000000  Datos decodificados sin error:  [0000]  [0000]
Registro 2:  0  Registro 3:  0  Datos codificados con error:  0000000  0000000  Datos decodificados sin error:  [0000]  [0000]
Registro 4:  0  Registro 5:  0  Datos codificados con error:  0000000  0000000  Datos decodificados sin error:  [0000]  [0000]

Articulación 1:  0    Articulación 2:  0    Articulación 3:  0
```

Figure. 12: Graphic interface of the slave

C. *Communication tests between Hamming encoder and decoder using Modbus protocol with error correction*

C1.Encoder

The encoder tests are performed by obtaining the different data of the joints of the robot animation made in Blender. Figure 13 shows the data encoded of the of the 3 robot joints without error and with error.

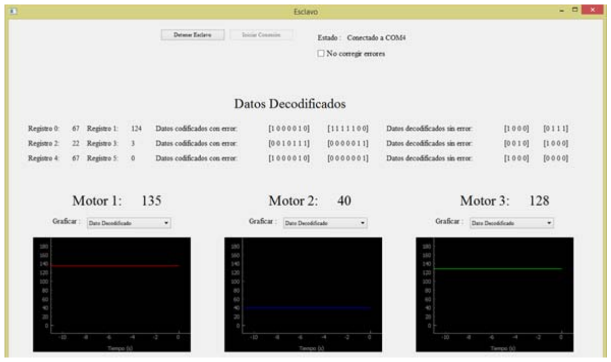


```
Comunicación Interfaz Gráfica
Puerto : COM2    Estado :  Conectado a COM2
Comunicación Modbus
Puerto : COM7    Estado :  Conectado a COM7

Datos Codificados con error y sin error
Dato Original 1:  135   Datos codificados sin error:  1000011  0111100   Datos codificados con error:  1000011  1111100
Dato Original 2 :  40   Datos codificados sin error:  0010110  1000011   Datos codificados con error:  0010110  0000011
Dato Original 3 :  128  Datos codificados sin error:  1000011  0000000   Datos codificados con error:  1000011  0000000

Articulación 1 (Enviado) :  135   Articulación 2 (Enviado) :  40   Articulación 3 (Enviado) :  128
```

Figure. 13: Original data encoded and sent with errors.

C2. Decoder

It is responsible for ensuring that the received angles are equal to the original data sent by the encoder. Figure 14 shows the graphs and the data encoded with error and decoded without error in the registers.

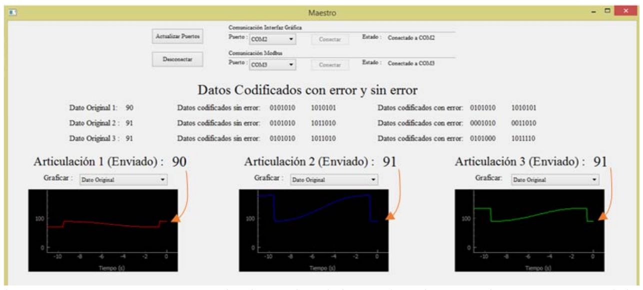


```
Estado :  Conectado a COM4
☐ No corregir errores

Datos Decodificados
Registro 0:  67  Registro 1:  124  Datos codificados con error:  [1 0 0 0 0 1 0]  [1 1 1 1 1 0 0]  Datos decodificados sin error:  [1 0 0 0]  [0 1 1 1]
Registro 2:  22  Registro 3:  3    Datos codificados con error:  [0 0 1 0 1 1 1]  [0 0 0 0 0 1 1]  Datos decodificados sin error:  [0 0 1 0]  [1 0 0 0]
Registro 4:  67  Registro 5:  0    Datos codificados con error:  [1 0 0 0 0 1 0]  [0 0 0 0 0 0 1]  Datos decodificados sin error:  [1 0 0 0]  [0 0 0 0]

Motor 1:  135    Motor 2:  40    Motor 3:  128
```

Figure. 14: Data encoded with error and decoded without errors

D. *Communication tests between Hamming encoder and decoder using Modbus protocol without error correction*

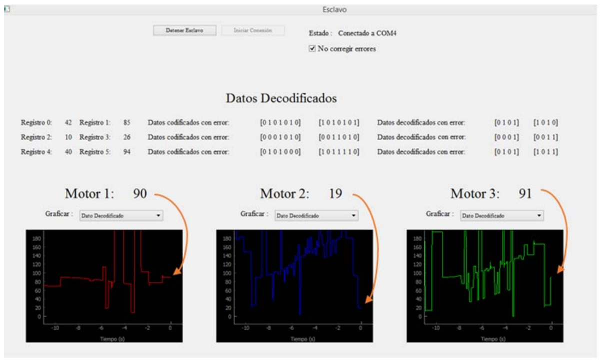
Figure 15 shows the original data, the coded data, data with a single random error added and the graphs of the original values.



```
Comunicación Interfaz Gráfica
Puerto : COM2    Estado :  Conectado a COM2
Comunicación Modbus
Puerto : COM3    Estado :  Conectado a COM3

Datos Codificados con error y sin error
Dato Original 1:  90   Datos codificados sin error:  0101010  1010101   Datos codificados con error:  0101010  1010101
Dato Original 2 :  91  Datos codificados sin error:  0101010  1011010   Datos codificados con error:  0001010  0011010
Dato Original 3 :  91  Datos codificados sin error:  0101010  1011010   Datos codificados con error:  0101000  1011110

Articulación 1 (Enviado) :  90   Articulación 2 (Enviado) :  91   Articulación 3 (Enviado) :  91
```

Figure. 15: Data encoded and with a single random error added

The slave has the option of not correcting the errors from data arriving at the decoder. This option is selected to show the variations that occur in the frames when the received data are not corrected, as can be seen in Figure 16.



```
Estado :  Conectado a COM4
☑ No corregir errores

Datos Decodificados
Registro 0:  42  Registro 1:  85  Datos codificados con error:  [0 1 0 1 0 1 0]  [1 0 1 0 1 0 1]  Datos decodificados con error:  [0 1 0 1]  [1 0 1 0]
Registro 2:  10  Registro 3:  26  Datos codificados con error:  [0 0 0 1 0 1 0]  [0 0 1 1 0 1 0]  Datos decodificados con error:  [0 0 0 1]  [0 0 1 1]
Registro 4:  40  Registro 5:  94  Datos codificados con error:  [0 1 0 1 0 0 0]  [1 0 1 1 1 1 0]  Datos decodificados con error:  [0 1 0 1]  [1 0 1 1]

Motor 1:  90    Motor 2:  19    Motor 3:  91
```

Figure. 16: Data decoded without error correction

E. *Test of the Path to be followed by the manipulator robot*

For testing the operation of the 3-DOF manipulator robot, a trajectory was implemented as shown in Figure 17.

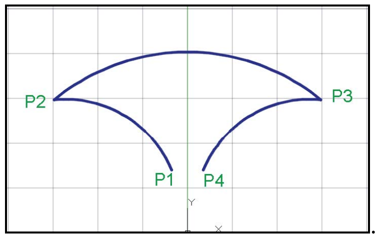


Figure. 17: Path for the movement of the manipulator robot

The trajectory is defined based on 4 different points, which are designed by the angles of each link as shown in Table 1.

TABLE I
TRAJECTORY POINTS IN ANGLES

	P1	P2	P3	P4
Link 1	180°	135°	45°	0°
Link 2	90°	135°	45°	90°
Link 3	0°	135°	45°	180°

To perform the path tracking test, we departed from the joint angles, namely 135, 40 and 128°, which are the same delivered by 3D animation to the encoder to add the respective redundancy bits using the Hamming code. Afterwards, a bit was randomly modified to produce an error at the moment of data transmission.

The communication test with error correction between the encoder (master) and decoder (slave) can be observed in Figures 13 and 14.

Figure 18 shows the position of the robot according to the angles of the servomotors in the 3 joints.

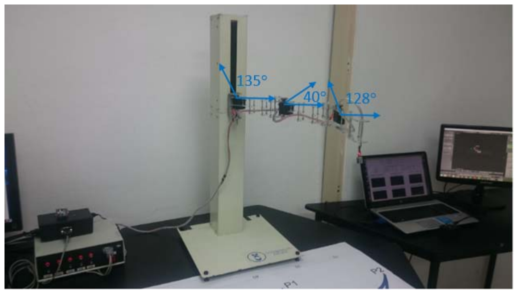


Figure. 18: Angles received in the robot.

Figure 19 shows that the robot tracks the path even though random errors are introduced in the encoder, and the decoder detects and corrects them, thereby retrieving the original data.

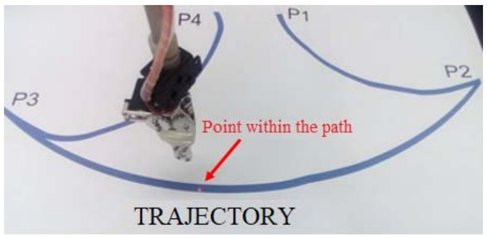


Figure. 19: Trajectory tracking with error correction.

Figure 20 shows that the point is out of the trayectory path as the decoder did not correct the error introduced by the Hamming encoder.

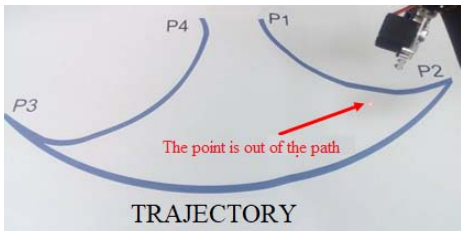


Figure. 20: Trajectory tracking without error correction.

V. Conclusions

The implementation of a communication through the modules XBee for the control of a 3-DOF robotic arm using a master-slave architecture by means of the Modbus protocol allows for fault tolerant communication between the Hamming coder and decoder. This implementation is crucial when data needs to be sent from a place distant from the manipulator robot, especially in places dangerous for humans like the mining industry. It is important to conclude that this study was carried out with a robot in the laboratory, the same that can be implemented in the industrial field with more robust robots.

One of the advantages of ZigBee communication is that it allows for working with mesh topology, which enables more connections of manipulator robots in network. The Modbus protocol shares this feature, as it can employ multiple slaves, which makes it a system applicable at the industrial level.

With the development of this project, a solution has been found to the problem originated from the alteration or introduction of error bits in the communication between the master and the slave at the industrial level. Thanks to the Hamming coding (7,4) redundancy bits can be introduced, and by means of the decoder, the errors can be detected and corrected, so that the movement of the robot follows the proposed trajectory.

The implementation of the Hamming Code in the Modbus protocol a Forward Error Correction (FEC) has been incorporated without the need for an automatic repeat request also known as Backward Error Correction (BEC) used by the Modbus or Modbus TCP / IP protocol through the Cyclic Redundancy Code (CRC)

Acknowledgment

This work was possible thanks to the support of the Vicerrectorate for Research of the Universidad del Azuay, Cuenca - Ecuador and the Electrical Engineering Department of the Universidad de Santiago de Chile, Santiago de Chile - Chile.

References

[1] E. Sanchez Gutiérrez, A. Cervantes Herrera y G. Ramírez Prado, “XBee Distance Measurement Through The Log Normal Shadowing Model”, IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), pp. 1 - 5, 2016.

[2] Teodorescu, R. M., Cioc, I. B., Vochin, B. A., & Lita, A. I. (2016, June). LabVIEW application used for remote control of a mobile robot with Xbee communication. In 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (pp. 1-4). IEEE.

[3] Singh, Anil Kumar. "Error detection and correction by hamming code." International Conference on Global Trends in Signal Processing, Information Computing and Communication, IEEE, 2016.

[4] Tsuruda, Yusuke, and Akio Tsuneda. "Study on Error-Correcting Using Output Level of Correlation Receivers and Hamming Codes in CDMA Communications." 2018 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2018.

[5] Febriani, Rosita, Eka Susanti, and Emilia Hesti. "Xbee pro module application in to organize and monitoring earthquake disaster location with the robot control system." 2018 International Conference on Information and Communications Technology (ICOIACT). IEEE, 2018.

[6] Tamboli, Sadik, et al. "Implementation of Modbus RTU and Modbus TCP communication using Siemens S7-1200 PLC for batch process." 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM). IEEE, 2015.

[7] P. H. To, B. D. Nguyen, V.-S. Tran y K. T. Pham, “Single-feed wideband circularly polarized antenna for 2.4 GHz ISM band applications”, IEEE *International Conference on Advanced Technologies for Communications (ATC),* pp. 686-688, 2013.

[8] Sparkfun, «SparkFun Electronics,» 10 January 2017. [Online]. https://www.sparkfun.com/products/retired/10414.. [Last acceso: 20 Mar 2020].

[9] S. Dere, S. Sahasrabudhe y S. Iyer, “Creating open source repository of 3D models of laboratory equipments using Blender, *IEEE International Conference on Technology for Education,* pp. 149-156, 2010.