Implementación y evaluación de PyFirmata para lectura de señales analógicas en Raspberry Pi

RESUMEN: El sistema embebido Raspberry Pi ha demostrado ser un dispositivo practico para múltiples aplicaciones en el área de la ingeniería. Su capacidad de lectura con sensores digitales la convierte en una herramienta muy eficaz para proyectos de automatización, control y monitoreo. Sin embargo, la lectura de sensores analógicos es un desafío común al trabajar en este dispositivo ya que no lo puede hacer directamente. En este artículo, se ha desarrollado un modelo metodológico para interpretar las señales analógicas en la Raspberry Pi utilizando la interfaz PyFirmata, que se basa en el protocolo de comunicación Firmata. Esta permite una interpretación eficiente de las señales analógicas en la Raspberry Pi, estableciendo una conexión efectiva con dispositivos analógicos de diferente tipo. Realizando una comparativa técnica con otros protocolos de comunicación como WiFi, Zigbee, LoRaWAN y Ethernet, demostrando que PyFirmata es una opción confiable y eficiente, sin requerir shields adicionales. Esta metodología fomenta el desarrollo de aplicaciones en Raspberry Pi que necesiten adquirir y procesar datos analógicos, esperando que la adopción de PyFirmata crezca como opción para la comunicación con dispositivos analógicos, brindando a los usuarios una solución a múltiples proyectos que requieran lecturas analógicas.

PALABRAS CLAVE: Analógico, Arduino, Firmata, PyFirmata, Python, Raspberry pi.

Colaboración
Giovani Zamitiz Córdoba; Adriana Pérez López, TecNM/CampusTeziutlán;LuisAlbertoEspejoPonce, TecNM / Campus Zacapoaxtla

Fecha de recepción: 29 de julio del 2023
Fecha de aceptación: 13 de noviembre del 2023

ABSTRACT: The embedded system Raspberry Pi has proven to be a practical device for various engineering applications. Its capability to read digital sensors makes it a highly effective tool for automation, control, and monitoring projects. However, reading analog sensors is a common challenge when working with this device since it cannot perform this task directly. In this article, a methodological model has been developed to interpret analog signals in the Raspberry Pi using the PyFirmata interface, based on the Firmata communication protocol. This allows for efficient interpretation of analog signals in the Raspberry Pi, establishing an effective connection with various types of analog devices. A technical comparison was conducted with other communication protocols such as WiFi, Zigbee, LoRaWAN, and Ethernet, demonstrating that PyFirmata is a reliable and efficient option without requiring additional shields. This methodology encourages the development of applications in Raspberry Pi that require acquisition and processing of analog data. It is expected that the adoption of PyFirmata will grow as an option for communication with analog devices, providing users with a solution for numerous projects that require analog readings.

KEYWORDS: Analog, Arduino, PyFirmata, Python, Raspberry pi, Sensor.

INTRODUCCIÓN

En este artículo, se presenta una metodología para realizar lecturas analógicas con la tarjeta Raspberry Pi utilizando la interfaz PyFirmata basada en el protocolo Firmata. Esta metodología ha sido probada en un invernadero tipo túnel con 4 plantas seleccionadas, permitiendo una comunicación confiable entre la Raspberry Pi, Arduino y los sensores analógicos. Se realizaron comparaciones técnicas con otros protocolos de comunicación, como WiFi, Zigbee, LoRaWAN y Ethernet, demostrando la efectividad de PyFirmata sin necesidad de utilizar hardware adicional. Esta metodología

ofrece una solución práctica y eficaz para proyectos que requieran la adquisición de datos analógicos con Raspberry Pi.

En el estado del arte, se analizaron 5 trabajos que abordan la transformación de señales analógicas utilizando Raspberry Pi con diferentes técnicas y enfoques. Cada documento presentó resultados precisos y confiables en el análisis y procesamiento de datos recolectados por sensores externos. La información relevante de cada trabajo se encuentra resumida en la Tabla 1.

Tabla 1. Comparación de diversas investigaciones en recolección de datos con IoT.

No°	Equipo de medición	Método empleado	Resultado
Lozano García *[1]*	DHT22, ESP8266	Conversión analógica a digital	Buena comunicación con poca latencia de Wi-Fi
Rebeca Cintra *[2]*	Sensor TTC103, DHT11, sensor de Ph, conversor ADC0804	Conversión analógica a digital	Buena conexión con una complejidad de implementación
María Bedolla *[3]*	Amplificadores Lock-in (LIA)	Detección síncrona	Interpretación de datos analógicos correctamente
Arturo Jiménez *[4]*	Cámara web	Captura de imágenes en tiempo real	Lectura de la cámara correcta
Yuliana Andrade *[5]*	Sensor BMP280 de altitud, presión y temperatura	Conversión analógica a digital	Visualización correcta de los datos medidos en tiempo real

Fuente: Elaboración propia.

Los trabajos mencionados [1], [2], [3], [4] y [5], demuestran el potencial de la tarjeta Raspberry Pi en combinación con convertidores analógicos a digitales y shields para leer señales analógicas en diferentes contextos. Sin embargo, es importante destacar que es este artículo se evaluó la utilización de la interfaz PyFirmata para valorar su efectividad en esta aplicación específica de lecturas analógicas en colaboración con Arduino. Esta metodología de investigación tiene como objetivo obtener mejor rendimiento de comunicación sin necesidad de adquirir Shields adicionales para la conversión de señales analógicas.

MÉTODOLOGIA

En este estudio, llevamos a cabo la recolección de variables de humedad del suelo utilizando 4 sensores HD-38 y el sensor MQ-135 para evaluar la calidad del aire en un invernadero, contando con 4 plantas seleccionadas. La configuración espacial de este invernadero tipo túnel, al ser de tamaño reducido, permitió ubicar tanto la Raspberry como el Arduino en el mismo lugar, lo que resultó una metodología muy viable y práctica.

La recolección de datos comenzó desde los sensores, los cuales se conectaron al Arduino mediante sus pines analógicos del "A0 al A4". Los primeros 4 pines se utilizaron para los sensores HD-38, colocando uno en cada planta para obtener mediciones individuales de humedad del suelo, mientras que el último pin se destinó al sensor MQ-135 para obtener una medida general de la calidad del aire dentro del invernadero.

Estos sensores se comunican con el protocolo Firmata dentro del Arduino, permitiendo la comunicación con la Raspberry a través del puerto USB utilizando la interfaz PyFirmata con el lenguaje de programación Python, como se observa en la Figura 1.

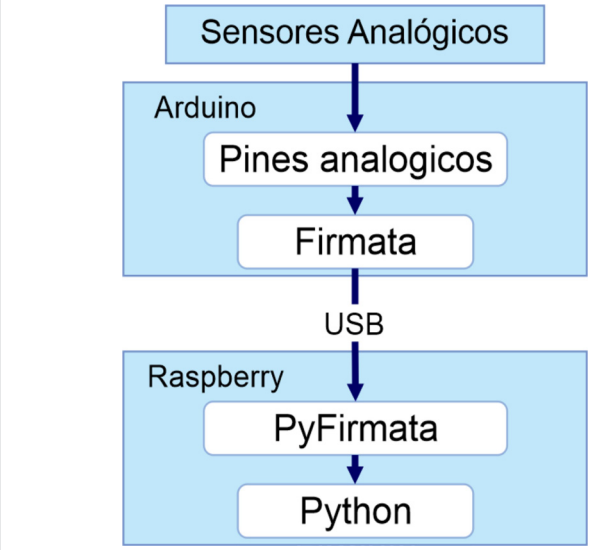

Figura 1. Diagrama de comunicación.
Fuente: Elaboración propia.

Con esta comunicación establecida, se realizaron pruebas directamente en el lugar, verificando el funcionamiento de la metodología. Para ello, se empleó un Arduino con un sketch sencillo que leía los mismos sensores, lo que permitió evaluar la precisión al utilizar PyFirmata con un Arduino esclavo conectado a la Raspberry. La metodología se encuentra explicada de manera más sencilla en la Figura 2, mediante un diagrama de bloques que ilustra los pasos llevados a cabo.

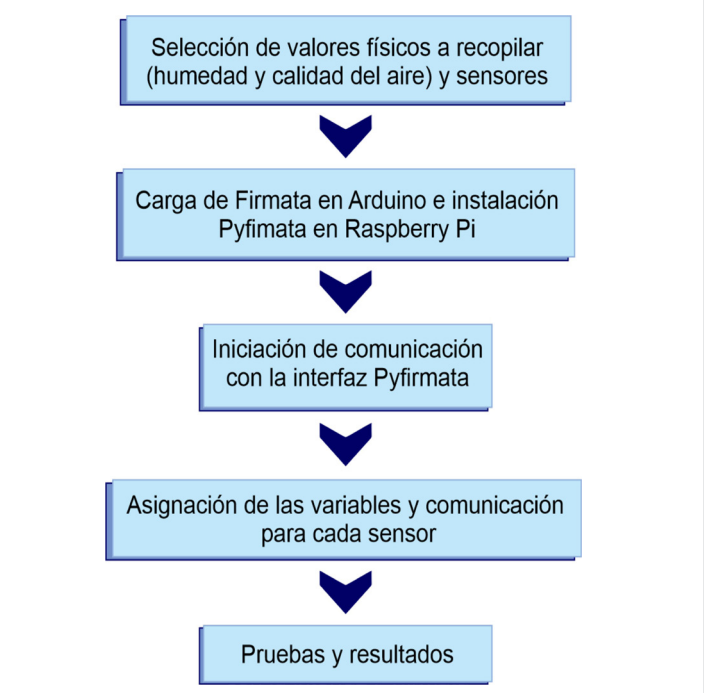

Figura 2. Diagrama de bloques de la metodología.
Fuente: Elaboración propia.

Selección de valores físicos y sensores

En esta metodología, parte de la necesidad de obtener lecturas de los valores de humedad del suelo y la calidad del aire. Sabiendo que estos valores se recopilan comúnmente a través de sensores analógicos, se realizó una búsqueda exhaustiva para seleccionar los sensores más adecuado para este caso de estudio, tomando en cuenta tanto el precio como la precisión como factores fundamentales. Se llevaron a cabo comparaciones técnicas entre diferentes sensores para la humedad del suelo, con el objetivo de identificar aquel que mejor se adaptara a estudio como se observa en la Tabla 1. La elección del sensor adecuado siendo el HD-38 [6], resultó crucial para asegurar que las mediciones fueran precisas con un precio más accesible y resistencia a condiciones climáticas adversas.

Tabla 2. Comparación de sensores para la medición de humedad del suelo.

Sensor	medición	Protocolo de comunicación	Costo estimado (MXN)
HD-38	0% a 100%	Analógico	$250.00
TRH200	0% a 100%	RS485	$350.00
VH400	0% a 100%	Analógico	$300.00
EC-5	0% a 100%	Digital	$400.00
SM200	0% a 100%	SM2C/dHH2	$450.00

Fuente: Elaboración propia.

En el análisis de la variable de calidad del aire, se evaluaron diversos tipos de sensores con distintos protocolos de comunicación. Se optó principalmente por el sensor analógico con el propósito de utilizar la misma interfaz para todos los sensores y poder obtener lecturas de componentes orgánicos. Específicamente, se seleccionó el sensor MQ-135 [7] debido a su accesibilidad y precio asequible.

Tabla 3. Comparación de sensores para la medición de humedad del suelo.

Sensor	medición	Protocolo de comunicación	Costo estimado (MXN)
MQ-135	CO2, gases orgánicos	Analógico	$150.00
CCS811	TVOCs y CO2.	I2C	$200.00
PMS5003	Partículas PM2.5 y PM10	UART	$250.00
SGP30	TVOCs y CO2 equivalentes	I2C	$450.00

Fuente: Elaboración propia.

Contando con los sensores adecuados para la recolección de las variables ambientales en este caso, se carga el protocolo en Arduino y la interfaz en Raspberry.

Carga del protocolo Firmata e interfaz PyFirmata

Para este caso de estudio, se empleó un Arduino NANO y se utilizó el IDE de Arduino para cargar el sketch de Firmata. Este sketch se encuentra disponible por defecto en los programas de ejemplo con el nombre de "StandardFirmata", siguiendo la dirección para localizar el programa como se muestra en la Figura 3. Posteriormente, se procedió a cargar el sketch en la placa correspondiente (Arduino NANO) a través del puerto COM al que se encontraba conectado, verificando el puerto en el administrador de dispositivos. Este paso aseguró la carga exitosa del protocolo Firmata en el microcontrolador.

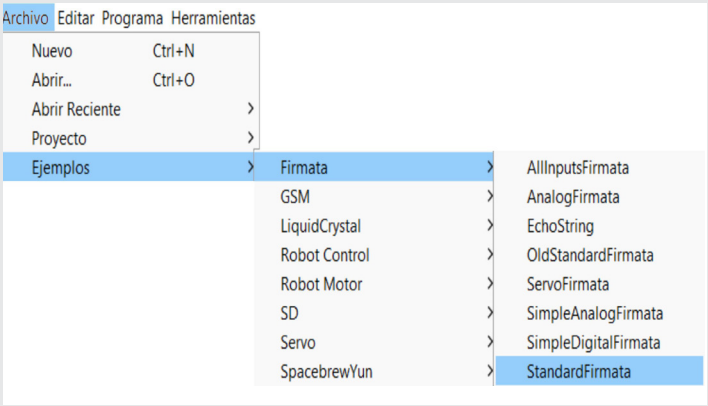

Figura 3. Ruta de instalación de StandardFirmata
Fuente: Elaboración propia.

Una vez contando con Firmata, se procedió a instalar PyFirmata en la tarjeta Raspberry. Dado que esta tarjeta embebida utiliza un sistema operativo (Raspbian) basado en el kernel Debian, resultó esencial buscar actualizaciones del sistema mediante la terminal [8], como se indica en la Figura 4, con el fin de evitar errores durante la instalación.

```
rasp@raspberrypi:~ $ sudo apt-get update
rasp@raspberrypi:~ $ sudo apt-get upgrade
```

Figura 4. Comandos de actualización
Fuente: Elaboración propia.

Teniendo las actualizaciones más recientes del sistema Raspbian hasta la fecha, se procedió a instalar el lenguaje de programación Python 3, necesario para utilizar PyFirmata, así como su administrador de paquetes correspondiente "pip" [9], como se muestra en la Figura 5 mediante comandos en la terminal. Esta instalación resultó de gran importancia, ya que sienta las bases para el uso de la interfaz.

```
rasp@raspberrypi:~ $ install python3
rasp@raspberrypi:~ $ install python3-pip
```

Figura 5. Comandos de instalación de Python3
Fuente: Elaboración propia.

Realizadas las actualizaciones necesarias y con Python ya instalado, se procedió a completar la instalación de PyFirmata utilizando el comando en la consola [10], tal como se muestra en la Figura 6. Este orden específico de comandos garantizó que la interfaz se instalara correctamente y evitó posibles errores en su uso.

```
rasp@raspberrypi:~ $ sudo pip3 install pyfirmata
```

Figura 6. Comando de instalación de PyFirmata
Fuente: Elaboración propia.

Teniendo la instalación completa en ambos dispositivos se pasa a realizar la comunicación entre ambas tarjetas para obtener las lecturas analógicas del Arduino NANO con PyFirmata y Python directamente desde Raspberry pi.

Conexión entre Raspberry y Arduino

Para que el microcontrolador Arduino pueda enviar la información se hace uso del puerto serial en ambos equipos. Esta conexión se enlaza por medio del puerto micro-USB del Arduino nano al puerto USB de la Raspberry como se muestra en la Figura 7.

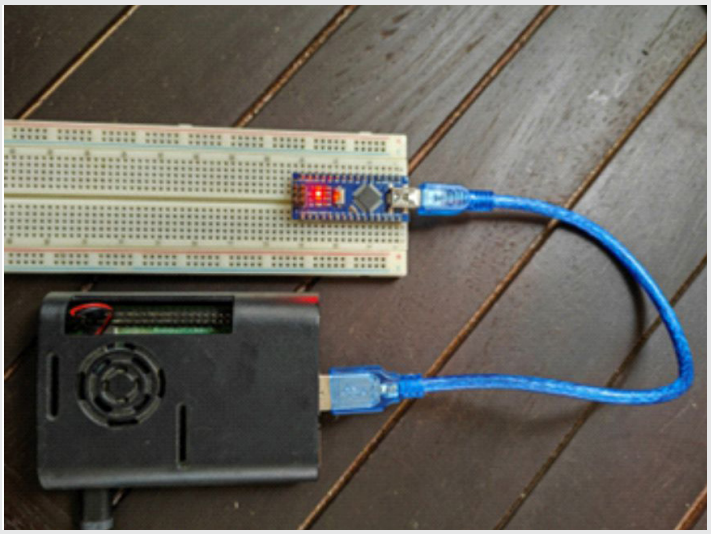

Figura 7. Conexión serial entre dispositivos
Fuente: Elaboración propia.

Se requirió programar en Python para manipular el enlace, facilitando la interpretación directa de los pines analógicos en la Raspberry como si fueran propios, extendiendo sus puertos. Se creó un archivo ".py" desde la terminal de Raspbian, donde se declararon las librerías PyFirmata, util y time. Estas librerías facilitaron la conexión con Arduino, que estaba en la dirección generada al conectarlo al puerto USB (detectada como "/dev/ttyUSB0"). Se estableció un iterador mediante la función útil para iterar continuamente sobre los valores recibidos desde la placa Arduino a través de la conexión serial, como se muestra en la Figura 8.

```
from pyfirmata import Arduino, util
import time

nano=Arduino("/dev/ttyUSB0")
it=util.Iterador(nano)
it.start()
```

Figura 8. Importación de librerías con Arduino
Fuente: Elaboración propia.

Asignación de variables y comunicación con los sensores

Teniendo el enlace establecido, se hace uso de todos los pines tanto digitales como analógicos, declarándolos de múltiples maneras utilizando en este artículo el siguiente método:

$$x = nano.get\_pin("a/d:0:i/o") \qquad \text{Ec. (1)}$$

Donde "nano" hace referencia al modelo de Arduino donde se declaró al inicio, el método ".get_pin" especifica cual es el tipo de pin analógico o digital, su número de identificación, especificando si será utilizado como de entrada o de salida, separada cada atributo con dos puntos, para esta metodología se usaron 5 entradas analógicas, 4 con sensores HD-38 y un sensor MQ-135, declarándolas como se muestra en la Figura 9.

```
sensor1=nano.get_pin("a:0:i")
sensor2=nano.get_pin("a:1:i")
sensor3=nano.get_pin("a:2:i")
sensor4=nano.get_pin("a:3:i")
sensor5=nano.get_pin("a:4:i")
```

Figura 9. Declaración de entradas analógicas
Fuente: Elaboración propia.

Completando la asignación de las variables de entrada, se estableció un bucle "while True" que se ejecuta continuamente hasta que se interrumpe manualmente. Dentro del bucle, se utilizó la función "time.sleep()" para pausar la ejecución del programa durante 5 seg. antes de continuar. Luego, se llamó a la función "read()" en cada uno de los objetos para leer el valor analógico actual de cada uno de los pines que se configuro anteriormente (Figura 10).

```
while True:
    time.sleep(5)
    print (sensor1.read())
    print (sensor2.read())
    print (sensor3.read())
    print (sensor4.read())
    print (sensor5.read())
```

Figura 10. Lectura de pines por ciclo while
Fuente: Fuente: Elaboración propia.

El valor de retorno de la función "read()" es un número entre 0 y 1 que representa la lectura del valor analógico en el pin correspondiente. Finalmente, se imprimen los valores en la consola utilizando la función "print()". Con la comunicación de Arduino y Raspberry Pi exitosamente completada implementando la interfaz PyFirmata para obtener los valores de los sensores analógicos, se realizan las pruebas de comunicación para verificar el funcionamiento correcto de la metodología.

Pruebas

Una vez que se desarrolló la metodología, se procedió a ponerla a prueba en un invernadero tipo túnel. Cada planta estaba equipada con un sensor HD-38, como se mencionó anteriormente, mientras que el sensor MQ-135 se colocó en el interior del espacio para realizar mediciones de calidad del aire, tal como se muestra en la Figura 11.

Figura 11. Colocación de sensores en las plantas.
Fuente: Elaboración propia.

Una vez aclarada la configuración con los sensores y las plantas, se llevaron a cabo pruebas para evaluar la eficiencia de las lecturas utilizando la interfaz PyFirmata, en comparación con un sketch normal de Arduino y la lectura directa del mismo conectado a la PC, sin utilizar Shields adicionales ni protocolos externos. Los resultados obtenidos se presentan en la Figura 12.

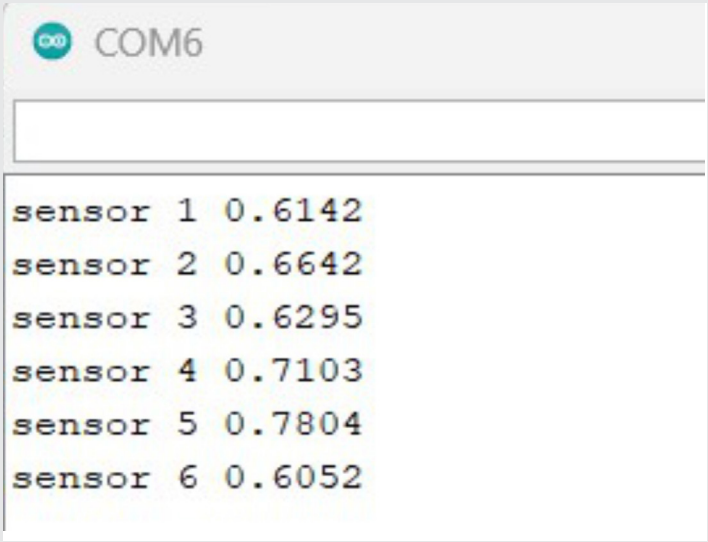

Figura 12. Lecturas del Arduino
Fuente: Elaboración propia.

Durante las pruebas de recolección de datos con la interfaz PyFirmata, se pudo observar que los valores obtenidos son muy similares con las pruebas del Arduino. Esto se debe a que, de cierta manera, PyFirmata actúa utilizando al Arduino como esclavo de sus pines, lo que permite una lectura más natural y resultados bastante coherentes y consistentes como se observa en la Figura 13.

```
rasp@raspberrypi:~/pf $ python3 pyfirmataTest.py
sensor1 0.6140
sensor2 0.6645
sensor3 0.6301
sensor4 0.7098
sensor5 0.7808
sensor6 0.6056
```

Figura 13. Lecturas de Raspberry con PyFirmata
Fuente: Elaboración propia.

Tras las pruebas realizadas en el invernadero, la metodología empleada con PyFirmata como interfaz entre la Raspberry Pi y el Arduino ha demostrado ser efectiva y fiable en la recolección de datos ambientales. Los sensores HD-38 y MQ-135 proporcionaron lecturas precisas de la humedad del suelo y la calidad del aire, respectivamente. La comparación entre las lecturas obtenidas mediante PyFirmata y las lecturas directas del Arduino reveló resultados muy similares, evidenciando la consistencia y precisión de la interfaz. PyFirmata facilitó la recolección de datos analógicos sin cargar sketches constantemente, simplificando la ejecución de diversas configuraciones de sensores.

RESULTADOS

Como resultado de la implementación, se llevó a cabo una evaluación de la latencia de recepción al establecer la comunicación entre la tarjeta Raspberry y el envío de datos a los servicios en la nube. Este análisis se realizó utilizando los servicios de Amazon Web Services (AWS). La Figura 14 presenta la gráfica que ilustra la latencia de recepción al emplear IoT Core en conjunto con CloudWatch de AWS, utilizando el protocolo MQTT para la transmisión de datos [11].

Se logró una latencia pico de 120 milisegundos, lo que indica un rendimiento eficiente en el contexto de IoT. Tanto la lectura como el envío de información se ejecutan de manera ágil, destacando no solo su eficacia sino también su capacidad como alternativa viable para la implementación de servicios en la nube en el ámbito del Internet de las cosas.

Los resultados de esta metodología simplifican significativamente la recolección de valores analógicos al agilizar el uso de los pines analógicos en Raspberry Pi. Esto contrasta con la complejidad de otros protocolos que requieren shields adicionales y cambios de sketch en caso de modificaciones, los cuales se evitan al eliminar la carga constante de sketches. En términos de comunicación, los resultados fueron consistentes al utilizar el puerto USB de ambos dispositivos, evitando el ruido, la latencia alta y las pérdidas de comunicación. En cuanto a la mínima diferencia de señales, se encuentra dentro del límite de tolerancia del sensor, ya que, al tratarse de sensores analógicos, son muy sensibles a pequeños cambios en el entorno.

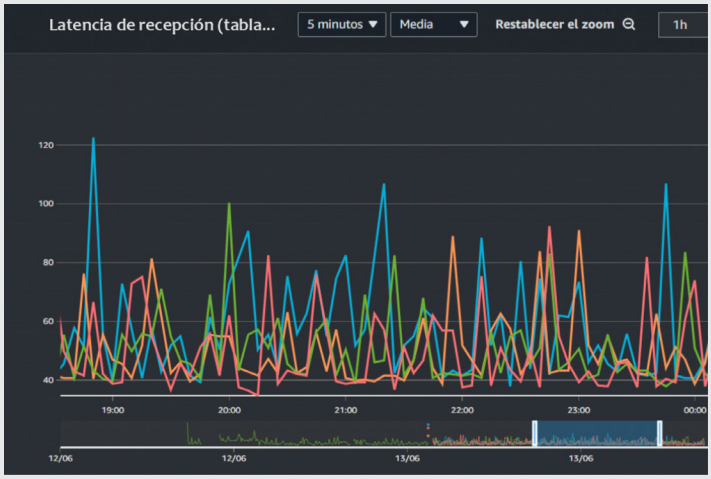

Figura 14. Latencia de recepción de datos en AWS
Fuente: Elaboración propia.

Esto aseguró un rendimiento estable en la transferencia de datos entre la Raspberry Pi y el Arduino. En programación, la elección de utilizar un único lenguaje, Python con PyFirmata, resultó ser favorable, facilitando el llamado al Arduino, el manejo de variables y el procesamiento de datos de manera homogénea y accesible. Esta elección se convirtió en una ventaja significativa, ya que agregar más sensores seguía la misma estructura sin cambiar de lenguaje o configurar diferentes protocolos.

Cabe destacar que, al utilizar el Arduino como dispositivo esclavo, se proporciona protección adicional a la Raspberry Pi en caso de incidentes en los sensores, como cortocircuitos. Dado el menor costo del Arduino, aproximadamente 10 veces más barato que la Raspberry Pi, esta configuración garantiza una protección efectiva y asequible para el equipo.

DISCUSIÓN

En esta sección, se realiza la discusión de comparar la metodología basada en Firmata [12] con otros protocolos de comunicación, como WiFi [13], Zigbee [14], LoRaWAN [15] y Ethernet [16]. Se analizaron las características técnicas de comunicación de cada protocolo, centrándonos en su precisión, consistencia y confiabilidad en la adquisición de datos analógicos. Basándose en el Arduino Nano como el microcontrolador principal en todas las configuraciones, junto con los Shields corres-

pondientes de cada protocolo, lo que nos permitió realizar una evaluación equitativa, como se muestra en la Tabla 4.

Tabla 4. Comparación general de los protocolos.

Protocolo	Velocidad	Shield Arduino Nano	Shield Raspberry Pi
Firmata	57.6 kbps	Ninguno	Ninguno
WiFi	54,000 kbps	Wi-Fi ARD-390	Incluida en la B3+ y B4
Zigbee	250 kbps	XBee Adaptador USB UAR	XBee Adaptador USB UAR
LoRa WAN	300 kbps	ESP32 SX1262 LoRA WAN	ESP32 SX1262 LoRA WAN
Ethernet [16]	100,000 kbps	Módulo Spi Enc28j60	Incorporado

Fuente: Elaboración propia.

La velocidad de transferencia es vital para la cantidad de datos transmitidos en un período dado. Aunque en ocasiones la velocidad puede ser baja, la información por señal suele ser pequeña en señales analógicas. Firmata se utiliza principalmente para leer estas señales representadas como números de punto flotante de 4 bytes. A una velocidad de transferencia de 57.6kbps, es suficiente para enviar múltiples señales analógicas en poco tiempo, logrando una transferencia de datos efectiva. Al comparar Firmata con otros protocolos para Arduino Nano, es importante considerar los costos asociados con las Shields necesarias para la conexión. Protocolos como WiFi, Zigbee, LoRaWAN y Ethernet requieren Shields específicas con costos adicionales, que deben ser contemplados en el presupuesto del proyecto.

Tabla 5. Comparación de precios de Shields.

Protocolo	Shield	Costo estimado (MXN)
WiFi	Wi-Fi ARD-390	$249.00 MXN
Zigbee	XBee Adaptador USB UAR	$403.39 MXN
LoRa WAN	ESP32 SX1262 LoRA WAN	$567.51 MXN
Ethernet	Modulo Enc28j60 Spi	$253.62 MXN

Fuente: Elaboración propia.

Los precios indicados son aproximados y pueden variar entre distintas tiendas en línea, generando posibles costos adicionales al proyecto. No obstante, Firmata simplifica la implementación y disminuye los gastos al no requerir Shields adicionales. En la Tabla 6, se encuentra una comparativa que incluye a Firmata y los cuatro protocolos analizados previamente. En dicha tabla se detallan las ventajas y desventajas de cada protocolo en términos de características clave.

Tabla 6. Comparación de ventajas y desventajas de los protocolos.

Protocolo	Ventajas	Desventajas
Firmata	- Versatilidad para la lectura de señales analógicas en general sin necesidad de adquirir shields adicionales. - Uso como esclavo sin cargar un sketch personalizado.	- Alcance limitado por la longitud del cable de conexión entre Raspberry Pi y Arduino Nano.
WiFi	- Mayor alcance inalámbrico de hasta 100 metros. - Alta velocidad de transferencia de hasta 54,000 kbps. - Amplia disponibilidad y compatibilidad.	- Posible interferencia de otros dispositivos inalámbricos. - Mayor consumo de energía. - Requiere el uso de un Shield específico.
Zigbee	- Alcance inalámbrico de hasta 100 metros. - Consumo de energía eficiente. - Excelente capacidad de conexión en malla.	- Limitado a la comunicación en red de área personal (PAN). - Requiere el uso de un shield específico. - Velocidad de transferencia de 250 Kbps.
LoRa WAN	- Alcance inalámbrico de hasta 10 km. - Bajo consumo de energía ideal para aplicaciones de IoT de largo alcance.	- Restricciones regulatorias de frecuencia en algunos países. - Requiere el uso de un Shield específico.
Ethernet	- Mayor velocidad de transferencia de 100,000 kbps. - Conexión cableada confiable, estable y amplia compatibilidad.	- Requiere el uso de un shield específico. - Necesita cables de conexión física.

Fuente: Elaboración propia.

La tabla indica que PyFirmata con Firmata ofrece ventajas notables gracias a su sencillez de uso. No es necesario cargar un sketch personalizado en el Arduino Nano, lo que facilita la implementación de proyectos que requieren la rápida y fácil adquisición y procesamiento de datos analógicos. Aunque la distancia de transmisión está limitada por la longitud del cable, esto no es un problema significativo en proyectos con dispositivos cercanos. Además, la conexión por cable asegura una comunicación más estable y confiable en comparación con los protocolos inalám-

bricos. A pesar de la velocidad de transferencia baja, Firmata puede enviar varias señales analógicas en un breve intervalo de tiempo, logrando una transferencia de datos efectiva.

CONCLUSIONES

En general, la implementación de PyFirmata como método para la lectura de señales analógicas en la Raspberry Pi ha demostrado como una opción confiable y eficiente. PyFirmata ofrece versatilidad en la lectura de señales analógicas, bajo costo y simplicidad de implementación. No es necesario cargar un sketch personalizado en el Arduino Nano, lo que simplifica el proceso de desarrollo de proyectos. Aunque PyFirmata tiene un alcance limitado por la longitud del cable de conexión entre la Raspberry Pi y el Arduino Nano, esto no es un problema significativo en proyectos donde la distancia entre los dispositivos es relativamente corta. En comparación con otros protocolos de comunicación como WiFi, Zigbee, LoRaWAN y Ethernet, PyFirmata se destaca por no requerir el uso de shields adicionales, lo que reduce los costos asociados y simplifica la implementación.

AGRADECIMIENTOS

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONAHCYT) por la beca otorgada para realizar estudios de posgrado a nivel maestría a través de la convocatoria "BECAS NACIONALES PARA ESTUDIOS DE POSGRADO 2022".

BIBLIOGRAFÍA

[1] L. García, O. Vergara, and V. Cruz, "Integración De Un Sistema Inalámbrico De Muestreo Con Esp8266 Y Raspberry Pi," Pistas Educativas, vol. 41, no. 134, pp. 203–2018, 2019.

[2] R. C. Cintra Hernández, L. Romero Amondaray, M. Hernández Martínez, and O. P. Montero, "Implementación De Una Estación Meteorológica De Bajo Costo Con Raspberry Pi," Revista Telemática, vol. 19, no. 1, pp. 11–21, 2020, [Online]. Available: http://revistatelematica.cujae.edu.cu/index.php/tele.

[3] M. Bedolla, A. Flores, and V. Everardo, "Detector De Fase De Bajo Costo Controlado Con Raspberry Pi," Miscelánea Científica en México, vol. 4, pp. 248–262, 2020.

[4] A. Jiménez, "Digitalización De Información De Un Manómetro Analógico A Través De Visión Artificial Para Generar Un Registro En La Web," 2022.

[5] Y. Andrade, "Implementación de un servidor LAMP con una Raspberry Pi y ESP32 para monitorear la temperatura, presión y humedad de un Laboratorio de Ciencias Básicas," Instituto Tecnológico Superior de Martínez de la Torre, Martínez de la Torre, 2022.

[6] OXDEA, "Sensor de Humedad de Suelo-Higrómetro Anticorrosivo HD-38," https://oxdea.gt/product/sensor-de-humedad-de-suelo-higrometro-anticorrosivo-hd-38/.

[7] Winsen, "MQ135," 2015. [Online]. Available: www.winsen-sensor.com.

[8] Raspberry Pi Foundation, "Updating and upgrading Raspberry Pi OS." Accessed: Apr. 26, 2023. [Online]. Available: https://www.raspberrypi.org/documentation/raspbian/updating.md

[9] Python Software Foundation, "Installing Python." Accessed: Apr. 23, 2023. [Online]. Available: https://www.python.org/downloads/.

[10] PyFirmata, "PyFirmata Documentation." Accessed: Apr. 27, 2023. [Online]. Available: https://pyfirmata.readthedocs.io/en/latest/.

[11] Amazon, "AWS IoT Core: Guía para desarrolladores," 2022. Accessed: Oct. 02, 2022. [Online]. Available: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/iot-dg.pdf#what-is-aws-iot.

[12] C. Cáceres, F. Fuentes, and J. Barrientos, "Sistema de monitoreo de calidad del agua en tiempo real utilizando Raspberry Pi, PyFirmata y AWS," Ingeniería Industrial, vol. 39, no. 2, pp. 141–151, 2020.

[13] Intel, "Diferentes protocolos de Wi-Fi y velocidades de datos," https://www.intel.la/content/www/xl/es/support/articles/000005725/wireless/legacy-intel-wireless-products.html. Accessed: Jun. 08, 2023. [Online]. Available: https://www.intel.la/content/www/xl/es/support/articles/000005725/wireless/legacy-intel-wireless-products.html.

[14] J. M. Moreno, "Informe Técnico: Protocolo ZigBee (IEEE 802.15.4)," 2007.

[15] J. de C. Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities," in 2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech), 2017, pp. 1–6.

[16] IBM, "Configuración de la Agregación de enlaces IEEE 802.3ad." Accessed: Jun. 08, 2023. [Online]. Available: https://www.ibm.com/docs/es/aix/7.2?topic=teaming-ieee-8023ad-link-aggregation-configuration.