Título del Documento en Formato IEEE

Jhoan Paul Castañeda Cáceres
FUNDACIÓN UNIVERSITARIA AGRARIA DE COLOMBIA
castaneda.jhoan@uniagraria.edu.co

***Abstract*—The integration of Python and microcontrollers plays a crucial role in the development of intelligent systems and automation. This report presents a series of activities designed to enhance proficiency in Python programming, microcontroller communication, and computer vision using OpenCV. The first section explores Python-based data acquisition and bidirectional communication with Arduino, including real-time sensor reading, data transmission, and hardware control. The second section focuses on image processing and computer vision, covering color space transformations, geometric transformations, edge detection, and contour analysis. These activities aim to provide a comprehensive understanding of key concepts in intelligent systems, preparing students for practical applications in automation, robotics, and artificial intelligence.**

***Index Terms*—Python, Arduino, microcontrollers, serial communication, image processing.**

I. Introducción

El desarrollo de sistemas inteligentes ha cobrado gran relevancia en la automatización, la robótica y el procesamiento de información en tiempo real. Python, con su versatilidad y amplia gama de bibliotecas, se ha convertido en un lenguaje fundamental para la implementación de estos sistemas, especialmente en conjunto con microcontroladores como Arduino. La capacidad de Python para interactuar con hardware a través de comunicación serial permite el control de sensores y actuadores, facilitando la creación de sistemas de monitoreo y respuesta automatizada.

Por otro lado, la visión artificial es un área clave dentro de la inteligencia artificial que permite a las máquinas interpretar y analizar imágenes o videos en tiempo real. OpenCV, una de las bibliotecas más utilizadas en este campo, ofrece herramientas para el procesamiento y análisis de imágenes, incluyendo detección de bordes, filtrado y reconocimiento de patrones. Estas técnicas son esenciales para la automatización de tareas en la industria, la seguridad y la investigación científica.

Este informe documenta el desarrollo de diversas actividades relacionadas con la integración de Python y Arduino, así como la aplicación de algoritmos de visión artificial para el procesamiento de imágenes. Se abordarán temas como la comunicación bidireccional entre Python y microcontroladores, la manipulación de imágenes mediante OpenCV y la implementación de técnicas avanzadas de procesamiento visual. A través de estos ejercicios, se busca fortalecer la comprensión y aplicación de sistemas inteligentes en escenarios prácticos.

***Index Terms*—Python, Arduino, microcontroladores, comunicacion serial, procesamiento de imagenes.**

II. Justificación

El avance de la inteligencia artificial y los sistemas embebidos ha impulsado la necesidad de integrar herramientas de software y hardware para desarrollar soluciones eficientes en diversas áreas, como la automatización industrial, la robótica y la visión artificial. Python, debido a su sintaxis intuitiva y su amplia compatibilidad con bibliotecas de control de hardware y procesamiento de datos, se ha convertido en un lenguaje clave para la implementación de sistemas inteligentes.

La conexión entre Python y microcontroladores como Arduino permite la adquisición, procesamiento y respuesta a datos en tiempo real, facilitando el desarrollo de sistemas de monitoreo, automatización y control de dispositivos. Por otro lado, OpenCV proporciona herramientas avanzadas para la manipulación y análisis de imágenes, esenciales en aplicaciones de visión artificial, desde la detección de objetos hasta el reconocimiento de patrones.

Este informe fue fundamental para comprender la sinergia entre Python, microcontroladores y visión artificial, proporcionando una base teórica y práctica para el diseño de sistemas inteligentes. Su desarrollo reforzó habilidades en el manejo de sensores, procesamiento de señales e interpretación de imágenes, aspectos esenciales en la formación de ingenieros en mecatrónica e inteligencia artificial.

A. Objetivo general

Desarrollar habilidades en la integración de Python con microcontroladores y visión artificial, abordando la adquisición y procesamiento de datos en tiempo real, la comunicación bidireccional entre dispositivos y el análisis de imágenes con OpenCV, con el fin de fortalecer la comprensión y aplicación de estas tecnologías en entornos de automatización y control.

B. Objetivos específicos

1) Implementar la comunicación entre Python y microcontroladores mediante protocolos de conexión serial e I2C, permitiendo el envío y recepción de datos para el control de sensores y actuadores.
2) Desarrollar aplicaciones prácticas de adquisición y procesamiento de datos utilizando sensores conectados a microcontroladores, facilitando la visualización y análisis de la información en Python.
3) Explorar técnicas de visión artificial con OpenCV, incluyendo la conversión de imágenes a distintos espacios de color, transformaciones geométricas y filtrado para el análisis de imágenes.

4) Aplicar algoritmos de detección de bordes y contornos en imágenes y video, optimizando la identificación de objetos en escenas capturadas en tiempo real.
5) Diseñar un sistema de control basado en datos de sensores, donde la información adquirida se utilice para la toma de decisiones en tiempo real, integrando visión artificial y microcontroladores
6) Fomentar el trabajo en equipo y el desarrollo de soluciones colaborativas en la implementación de sistemas inteligentes, fortaleciendo habilidades en programación, electrónica y procesamiento de datos.

III. Actividades para realizar

El taller está estructurado en dos unidades principales: conexión de microcontroladores con Python y visión artificial con OpenCV. Cada unidad contiene actividades diseñadas para aplicar conceptos teóricos y desarrollar habilidades prácticas en el uso de sistemas inteligentes el cual se puede acceder atraves de los anexos del documento.

A. Unidad 1: Repaso de Python y Conexión con Microcontroladores

Esta unidad se centra en la comunicación entre Python y microcontroladores como Arduino, permitiendo la adquisición y transmisión de datos entre ambos.

B. Unidad 2: Visión Artificial con Python-OpenCV

Esta unidad se enfoca en el procesamiento de imágenes y la aplicación de técnicas de visión artificial para la detección y análisis de objetos.

IV. Desarrollo

El desarrollo del taller se divide en dos unidades principales: Programación en Python y Conexión con Microcontroladores, y Visión Artificial con OpenCV. En cada una, se aplican conceptos clave para la implementación de sistemas inteligentes, combinando teoría y práctica.

A. Unidad 1: Programación en Python y Conexión con Microcontroladores

1) **Configuración del entorno de desarrollo**: Se instalaron y configuraron las herramientas necesarias como Python, Arduino IDE y bibliotecas específicas (pyserial, OpenCV). Esto permite la comunicación entre el software y el hardware.
2) **Establecimiento de la comunicación serial**: Se utilizó la comunicación serial para la transmisión de datos entre la computadora y el microcontrolador, permitiendo la recepción y envío de información en tiempo real.
3) **Implementación de sensores y actuadores**: Se conectaron sensores como LDR y sensores de temperatura para medir variables del entorno, y se emplearon actuadores como ventiladores y pantallas LCD para la visualización y control.
4) **Uso de protocolos de comunicación I2C y Serial**: Se configuró la comunicación mediante I2C para la transmisión de datos con dispositivos como pantallas LCD, facilitando la transmisión de información en paralelo con otros procesos.
5) **Interrupciones y eventos en Arduino**: Se implementaron interrupciones que permiten que el sistema responda inmediatamente a eventos sin necesidad de un monitoreo constante, mejorando la eficiencia del procesamiento.

B. Unidad 2: Visión Artificial con OpenCV

1) **Instalación y configuración de OpenCV**: Se instaló la biblioteca OpenCV en Python para permitir el procesamiento de imágenes y videos en tiempo real.
2) **Carga y manipulación de imágenes**: Se exploraron funciones de OpenCV para leer imágenes, cambiar su formato y ajustar características como brillo y contraste.
3) **Aplicación de transformaciones geométricas**: Se implementaron operaciones como rotación, escalado y traslación para comprender cómo afectan la estructura de las imágenes.
4) **Implementación de filtrado y detección de bordes**: Se aplicaron filtros como el suavizado Gaussiano y detectores de bordes como Sobel y Canny para mejorar la calidad de la imagen y facilitar la identificación de objetos.
5) **Análisis de contornos y segmentación de imágenes**: Se utilizaron herramientas como findContours para identificar bordes y contornos en imágenes y videos, facilitando el reconocimiento de formas.

V. Preguntas de investigación

A. ¿Cuál es la diferencia entre un lenguaje interpretado y un lenguaje compilado?

- **Interprete**: Un intérprete es un traductor que toma un programa fuente, lo traduce y, a continuación lo ejecuta, estos programas ya no se usan más, a excepción de circunstancias especiales. Su funcionamiento consiste en traducir la primera sentencia, detener la traducción y ejecutar el código, y así sucesivamente hasta terminar el programa.
- **Copilador**: Es un traductor que traduce lenguaje de alto nivel a lenguaje máquina. La traducción se realiza en una sola operación denominada compilación, es decir se traducen todas las instrucciones del programa en un solo bloque.

B. ¿Cuáles son los principales tipos de datos en programación y en qué situaciones se utilizan?

1) Existen dos tipos de datos: básicos, incorporados o integrados (estándar) que se incluyen en los lenguajes de programación; definidos por el programador o por el usuario.
    - **Numéricos**: Conjunto de valores numéricos, enteros y reales. Los enteros son números completos, no tienen componentes fraccionarios o decimales. Los reales siempre tienen un punto decimal y pueden

ser positivos o negativos. Se usan para operaciones numéricas, conteo de datos y variables.
- **Lógicos**: Es aquellos datos que solo puede tomar uno de dos valores, se usan en contextos binarios como en la elección de acciones o protocolos.
- **Cáracter**: Conjunto finito y ordenado de caracteres que la computadora reconoce. Se usa para operar, separa, interpretar.

C. Explica la diferencia entre operadores aritméticos, relacionales y lógicos con ejemplos.

1) La diferencia que existe entre los diversos operadores recae en la utilidad que tiene cada uno de ellos, pues su uso esta limitado al contexto en el que se aplique, ya sea matematico, logico o relacional.
   - **Operadores aritméticos**: Los operadores aritméticos se utilizan igual que en matemáticas.
   - **Operadores Relacionales**: Los operadores relacionales o de relación permiten realizar comparación de valores de tipo numérico o de carácter. Sirven para expresar las condiciones en los algoritmos.

D. ¿Cómo funcionan las estructuras de decisión?

1) Estructura selectiva - Las estructuras selectivas se usan para tomar decisiones lógicas. La representación de una estructura selectiva se hace con palabras en pseudocodigo. (if, then, else). En las estructuras selectivas se evalua una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas.
2) Estructura Repetitivas - Las estructuras que repiten instrucciones un numero determinado de veces se denominan bucles y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones.
   - **Estructura While**: La estructura While es aquella en que el cuerpo del bucle se repite mientras se cumple una determinada condición.
   - **Estructura For**: Se usa para un numero de iteraciones fijo, ejecuta las acciones dentro del cuerpo del bucle un numero especificada de veces, y de modo automático controla el numero de iteraciones atravez del cuerpo del bucle.

E. ¿Qué es una función con retorno y una sin retorno? Proporciona un ejemplo en un lenguaje de programación.

1) En programación, una función con retorno es aquella que, al ser ejecutada, devuelve un valor al punto donde fue llamada. Por otro lado, una función sin retorno realiza una tarea específica pero no devuelve ningún valor.

F. ¿Por qué es importante el manejo de excepciones en un programa? Da un ejemplo práctico.

1) El manejo de excepciones es crucial en la programación porque permite a los desarrolladores gestionar errores o situaciones inesperadas de manera controlada, evitando que el programa se detenga abruptamente y mejorando la experiencia del usuario.

G. ¿Para qué se utilizan los diagramas de secuencia y los diagramas de actividades en el desarrollo de software?

1) Los diagramas de secuencia y los diagramas de actividades son herramientas visuales utilizadas en el desarrollo de software para modelar diferentes aspectos del sistema.
   - **Diagramas de secuencia**: Representan la interacción entre objetos en un sistema en orden cronológico, mostrando cómo se envían mensajes entre ellos. Son útiles para detallar el flujo de eventos en casos de uso específicos.
   - **Diagramas de actividades**: Ilustran el flujo de trabajo o las actividades dentro de un proceso, destacando las decisiones, bifurcaciones y concurrencias. Ayudan a entender la lógica de los procesos y a identificar posibles mejoras.

H. ¿En qué se diferencia un paradigma de programación imperativa de uno declarativo?

- **Programación imperativa**: El programador indica cómo se debe realizar una tarea, especificando los pasos secuenciales que el sistema debe seguir. Ejemplo: lenguajes como C, Java.
- **Programación declarativa**: El programador describe qué resultado desea obtener sin detallar los pasos para lograrlo. El enfoque se centra en el qué y no en el cómo. Ejemplo: lenguajes como SQL, HTML.

I. Explica las principales características de la programación orientada a objetos.

1) La programación orientada a objetos es un paradigma que organiza el software en ”objetos” que encapsulan datos y comportamientos. Las principales características de la programación orientada a objetos de pueden definir como:
   - **Encapsulamiento**: Agrupa datos y métodos que operan sobre esos datos dentro de una misma entidad u objeto, protegiendo el estado interno del objeto.
   - **Herencia**: Permite crear nuevas clases basadas en clases existentes, facilitando la reutilización y extensión del código.
   - **Polimorfismo**: Habilidad de diferentes objetos para ser tratados como instancias de una misma clase base, permitiendo que un mismo método funcione de manera diferente según el objeto que lo invoque.
   - **Abstracción**: Simplifica la complejidad al permitir que los programadores se enfoquen en la interfaz de un objeto, ocultando los detalles de implementación.

J. ¿Cuál es la diferencia entre la programación estructurada y la programación funcional?

- **Programación estructurada**: Se basa en la descomposición de un programa en subrutinas o funciones,

utilizando estructuras de control como bucles y condicionales. Se centra en el cómo se realizan las tareas.

- **Programación funcional**: Se basa en la composición de funciones puras, evitando el uso de estados mutables y efectos secundarios. Se centra en el qué se desea lograr, describiendo las transformaciones de datos.

VI. Conclusiones

1) El desarrollo del taller permitió reforzar los fundamentos de la programación y explorar los paradigmas más utilizados en el desarrollo de software.
2) A través de ejercicios prácticos en Csharp y Python, se evidenció la importancia de comprender estructuras de control, manejo de excepciones y estructuras de datos para escribir código más eficiente y robusto.
3) La comparación entre distintos paradigmas mostró cómo cada enfoque tiene ventajas según el problema a resolver. La programación estructurada ayudó a organizar el código en funciones, la programación orientada a objetos facilitó la modelación de entidades mediante clases y métodos, y la programación funcional permitió soluciones más concisas y expresivas.

Referencias

1) Al-jabery, K. K., Obafemi-Ajayi, T., Olbricht, G. R., & Wunsch II, D. C. (2020). 9—Data analysis and machine learning tools in MATLAB and Python. En K. K. Al-jabery, T. Obafemi-Ajayi, G. R. Olbricht, & D. C. Wunsch II (Eds.), Computational learning approaches to data analytics in biomedical applications (pp. 231-290). Academic Press. https://doi.org/10.1016/B978-0-12-814482-4.00009-7
2) Barillaro, L. (2024). Deep learning platforms: PyTorch. En Reference module in life sciences. Elsevier. https://doi.org/10.1016/B978-0-323-95502-7.00093-2
3) Cortell-Nicolau, A. (2024). Agent-based modeling. En E. Nikita & T. Rehren (Eds.), Encyclopedia of archaeology (second edition) (second edition) (Second Edition, pp. 1090-1098). Academic Press. https://doi.org/10.1016/B978-0-323-90799-6.00094-X
4) Ebrahimi, A., Sefat, H. V., & Amani Rad, J. (2025). 1—Basics of machine learning. En J. Amani Rad, S. Chakraverty, & K. Parand (Eds.), Dimensionality reduction in machine learning (pp. 3-38). Morgan Kaufmann. https://doi.org/10.1016/B978-0-44-332818-3.00009-5
5) Kong, Q., Siauw, T., & Bayen, A. M. (2021). Chapter 1—Python basics. En Q. Kong, T. Siauw, & A. M. Bayen (Eds.), Python programming and numerical methods (pp. 3-25). Academic Press. https://doi.org/10.1016/B978-0-12-819549-9.00010-5
6) Lasisi, M., Kolade, K., & Rotimi, O. (2025). Data science. En D. Baker & L. Ellis (Eds.), Encyclopedia of libraries, librarianship, and information science (first edition) (First Edition, pp. 89-96). Academic Press. https://doi.org/10.1016/B978-0-323-95689-5.00268-6
7) Lenz, S., Farhadyar, K., Hess, M., Binder, H., & Knaus, J. (2021). Encoding medical experiments in jupyter notebooks. En O. Wolkenhauer (Ed.), Systems medicine (pp. 445-452). Academic Press. https://doi.org/10.1016/B978-0-12-801238-3.11687-3
8) Liu, Z., Gu, Z., & Liu, P. (2025). Chapter 2—Data analysis in python. En Z. Liu, Z. Gu, & P. Liu (Eds.), Transportation big data (pp. 13-49). Elsevier. https://doi.org/10.1016/B978-0-443-33891-5.00013-2
9) Low, K. K., & Ling, M. H. (2024). Cell modelling and simulation. En Reference module in life sciences. Elsevier. https://doi.org/10.1016/B978-0-323-95502-7.00105-6
10) Mccaffrey, P. (2020). Chapter 11—A selective introduction to Python and key concepts. En P. Mccaffrey (Ed.), An introduction to healthcare informatics (pp. 145-157). Academic Press. https://doi.org/10.1016/B978-0-12-814915-7.00011-9
11) McClarren, R. G. (2018). Chapter 2—Digging deeper into python. En R. G. McClarren (Ed.), Computational nuclear engineering and radiological science using python (pp. 21-35). Academic Press. https://doi.org/10.1016/B978-0-12-812253-2.00003-0
12) Milano, M. (2019). Computing languages for bioinformatics: R. En S. Ranganathan, M. Gribskov, K. Nakai, & C. Schönbach (Eds.), Encyclopedia of bioinformatics and computational biology (pp. 199-205). Academic Press. https://doi.org/10.1016/B978-0-12-809633-8.20367-1
13) Pineda, S., Morales, J. M., & Wogrin, S. (2023). Mathematical programming for power systems. En J. García (Ed.), Encyclopedia of electrical and electronic power engineering (pp. 722-733). Elsevier. https://doi.org/10.1016/B978-0-12-821204-2.00044-1
14) Squire, J. C., & English, A. E. (2025a). 1—Introduction to python. En J. C. Squire & A. E. English (Eds.), Introduction to python and spice for electrical and computer engineers (pp. 1-48). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-443-19007-0.00001-5
15) Squire, J. C., & English, A. E. (2025b). 2—Python as a calculator. En J. C. Squire & A. E. English (Eds.), Introduction to python and spice for electrical and computer engineers (pp. 49-97). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-443-19007-0.00002-7
16) Squire, J. C., & English, A. E. (2025c). 3—Plotting in python. En J. C. Squire & A. E. English (Eds.), Introduction to python and spice for electrical and computer engineers (pp. 99-158). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-443-19007-0.00003-9
17) Squire, J. C., & English, A. E. (2025d). 4—Python programming. En J. C. Squire & A. E. English (Eds.), Introduction to python and spice for electrical and computer engineers (pp. 159-228). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-443-19007-0.00004-0
18) Voinov, A. (2019). Conceptual diagrams and flow diagrams. En B. Fath (Ed.), Encyclopedia of ecology

(second edition) (Second Edition, pp. 58-64). Elsevier. https://doi.org/10.1016/B978-0-12-409548-9.11143-1

19) Voinov, A. A. (2008). Conceptual diagrams and flow diagrams. En S. E. Jørgensen & B. D. Fath (Eds.), Encyclopedia of ecology (pp. 731-737). Academic Press. https://doi.org/10.1016/B978-008045405-4.00178-6

20) Zohuri, B., Mossavar-Rahmani, F., & Behgounia, F. (2022). Chapter 26—Python programming–driven artificial intelligence. En B. Zohuri, F. Mossavar-Rahmani, & F. Behgounia (Eds.), Knowledge is power in four dimensions: Models to forecast future paradigm (pp. 827-836). Academic Press. https://doi.org/10.1016/B978-0-323-95112-8.00026-X

Jhoan C. Castañeda Nació en Colombia el 18 de noviembre de 1998. Obtuvo su formación académica en distintas áreas, comenzando con estudios de primaria y bachillerato en el Instituo Distrital Laureano Gomez, seguido por un título técnico en desarrollo gráfico del SENA. Posteriormente, completó una tecnología en desarrollo gráfico para proyectos de arquitectura e ingeniería en la misma institución. Actualmente, estudia ingenieria mecatrónica en la institucion Uniagraria.

Ha trabajado en el ámbito de la ingeniería gastronómica, enfocándose en el desarrollo de ductos de extracción e inyección. Sus principales áreas de interés incluyen la mecatrónica, la programación y el desarrollo de planos.

Dayanna

Yesennia

Daniel