Guía

Análisis de movimiento usando un sistema embebido que integra comunicaciones alámbricas e inalámbricas

Grado sugerido: Décimo

Harvey David Rojas Cubides

Publicado en el Banco Virtual de Recursos de Colombia Programa en el año 2025.

Para contactar al autor/a de este recurso, escriba a:
harvey.rojas076@educacionbogota.edu.co

GUIA DE APRENDIZAJE
Análisis de movimiento usando un sistema embebido que integra comunicaciones alámbricas e inalámbricas
Grado: Décimo
Profesor. Harvey David Rojas Cubides, PhD.

Aprendizajes esperados	*Con esta guía podrás alcanzar los siguientes aprendizajes:* - Integra hardware y software para diseñar un dispositivo que interactúa con el entorno físico, respondiendo a necesidades específicas mediante soluciones prácticas y funcionales. - Analiza un problema tecnológico y lo organiza en partes más simples, reconociendo patrones funcionales y aplicando estrategias de diseño para estructurar la solución de forma eficiente. - Desarrolla algoritmos que emplean estructuras condicionales y repetitivas, asegurando el funcionamiento correcto del sistema a través de pruebas y ajustes del código. - Recolecta, interpreta y usa datos físicos del entorno, tales como como aceleración y orientación magnética para tomar decisiones programadas dentro de un sistema embebido.
Duración	*120 minutos*
Materiales Requeridos	*1. Dos (2) Micro:bit V2.0 o superior* *2. Dos (2) Cables USB* *3. Dos (2) Adaptador de pilas para Micro:bit y 2 pilas AAA* *4. Un (1) Computador con acceso a internet y Mu editor.* 5. Un (1) Cuaderno de apuntes o bitácora. 6. Cinta o sujeción para fijar el micro:bit a el brazo o pierna.
Actividades para desarrollar	Estas son las actividades necesarias para alcanzar los aprendizajes esperados: 1. Conformen equipos de 4 estudiantes asegurando los materiales para el desarrollo 2. Ingrese a https://python.Micro:bit.org/v/3 3. Cree un proyecto nuevo de programación en Python. 4. Aborde el siguiente reto de diseño Desarrolle un sistema embebido basado en Micro:bit, es decir un Programmable System-on-Chip (PSoC), para analizar el movimiento durante una caminata, empleando los sensores de **acelerómetro (x.y.z) y brújula**. Se capturarán datos de aceleración y orientación en el PSoC transmisor. Posteriormente, dichos datos se transmitirán vía

radiofrecuencia al PSoC receptor y desde este último se enviarán al **computador mediante USB** para ser graficados en tiempo real y analizados posteriormente.

A continuación, se presenta un diagrama que ilustra el reto

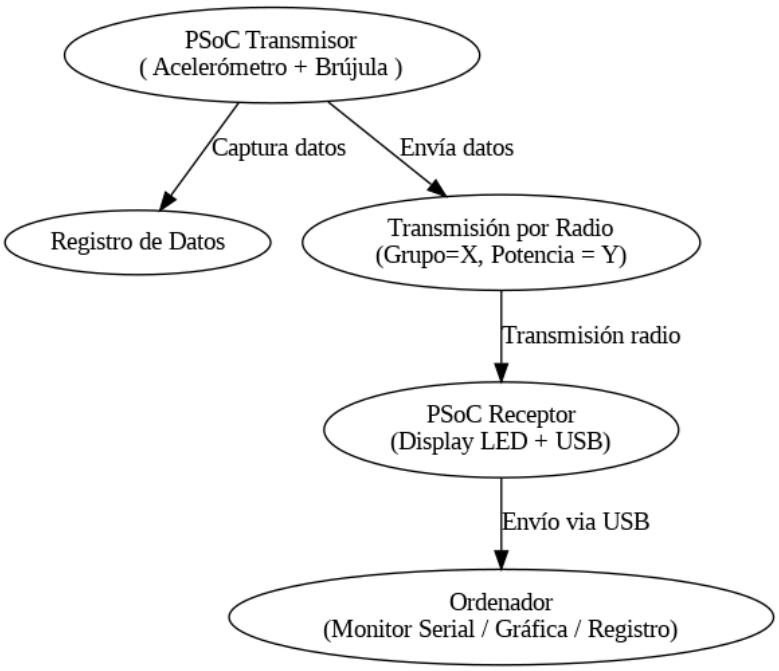

Figura 1. Esquema general de la aplicación

Algunas consideraciones finales:

- La comunicación con el computador será del tipo USB usando el trazador de curvas disponible en Mu Editor. Este es un editor de Python gratuito que se descarga en https://codewith.mu/en/download. La imagen general del editor es la siguiente:

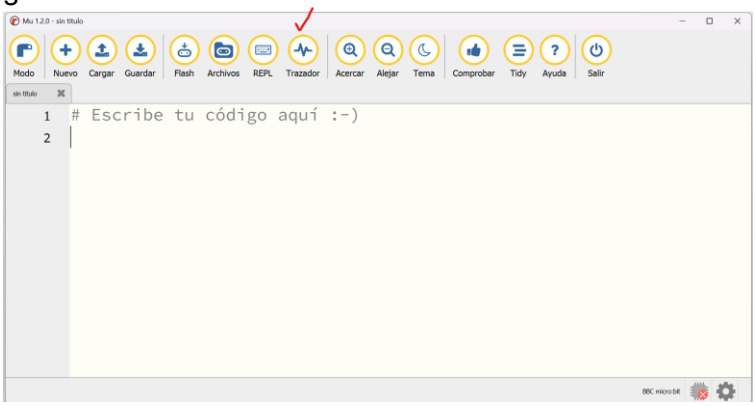

Figura 2. Entorno visual de Mu editor

- El registro de datos en el PSoC transmisor se debe hacer en la memoria interna una vez por segundo. El almacenamiento debe

incluir el número de muestra y el tiempo. Ajuste el valor del tiempo de almacenamiento para resultados óptimos.
- Los datos obtenidos deben ser exportados en formato Excel para su posterior análisis.
- El producto puede ser usado en la pierna o el brazo.

VALOR AGREGADO

Además del funcionamiento básico el producto debe tener un valor agregado personal que mejore su posicionamiento de mercado.

CRITERIOS BÁSICOS DE EVALUACIÓN

Criterio	Ponderación
Funcionamiento de la comunicación	30%
Interacción con el usuario	20%
Calidad de los datos obenidos	10%
Valor agregado y creatividad	20%
Documentación del código	10%
Presentación final	10%

SUGERENCIAS PRÁCTICAS PARA LOS EQUIPOS

A continuación, se plantean sugerencias para el desarrollo del trabajo autónomo de los equipos.

a) Empezar por lo más simple. Separen el problema en partes simples como la comunicación via RF, via USB y el uso de los sensores. Así sabrán si cada parte funciona como debe.
b) Documentar todo realizado. A veces un cambio pequeño en el código puede arreglar o dañar algo. Si llevan registro, les será más fácil volver atrás o explicar lo que hicieron.
c) Incluir comentarios claros en el código. Piensen que otros van a leer su código. Ayuda mucho escribir frases que expliquen qué hace cada parte.
d) Enfocarse primero en lo funcional y después en lo estético. No pierdan tiempo decorando si el programa todavía no hace lo que debe. Hagan que funcione, y luego lo presentan mejor.
e) Trabajar en grupo. A veces una persona entiende una parte mejor que otra. Dividir tareas hace el trabajo más fácil y rápido.
f) Realizar pruebas en distintos lugares. Cambia la superficie de caminata y la orientación para obtener múltiples datos.

	g) Realiza muchas pruebas. Imaginen que el producto será usado por usar otra persona. ¿Esa persona sabría cómo usarlo sin preguntarles? Eso ayuda a que el diseño sea más claro. h) Equivocarse es aprender. Programar es como experimentar: algo falla, se corrige, y al final se aprende. A veces, los errores enseñan más que los aciertos. i) Comparar y evaluar resultados. Miren qué han hecho otros estudiantes o creadores. No para copiar, sino para inspirarse. Siempre hay ideas que uno no se le habrían ocurrido solo. 4. Reconstruya en Excel las gráficas de aceleración y orientación. Analice cualitativa y cuantitativamente los datos de caminata de mínimo 2 minutos. 5. Socializa el desarrollo del producto y los datos obtenidos permitiendo que algunos usuarios interactúen con él. 6. Compare el producto y los datos con los de otros grupos de trabajo 7. Redacte un documento de evaluación de resultados planteando posibles mejoras a futuro.
Adaptaciones	*1. Adaptación para zonas sin acceso a internet constante.* - *Como alternativa a la programación en* https://python.Micro:bit.org/v/3 se puede descargar el editor gratuito Mu Editor disponible en https://codewith.mu/en/download *2. Adaptaciones para estudiantes con baja visión* - *Usar una pantalla conectada al computador con zoom o ampliación del texto.* - *Aplicar ajustes de pantalla que permitan la programación en entorno con alto contraste (modo oscuro) y letras grandes en los comentarios y código.* *3. Adaptación para estudiantes con discapacidad auditiva* - *Emplear instrucciones escritas claras. Videos con subtítulos o interpretación en lengua de señas.* - *Cambiar los elementos dependientes del sonido en el reto por secuencias de imágenes en la matriz de leds o via USB-serial.* *4. Adaptación para instituciones sin Hardware (Micro:bit)* - *En caso de no contar con los dispositivos físicos se pueden realizar simulaciones 100% funcionales en* https://python.Micro:bit.org/v/3

Referencias	- Hancock, M., & Smith, A. (2021). Exploring computational thinking in primary classrooms using the BBC micro:bit. Journal of Computer Assisted Learning, 37(2), 430–442. https://doi.org/10.1111/jcal.12509 - Halverson, E. R., & Sheridan, K. M. (2014). The Maker Movement in education. Harvard Educational Review, 84(4), 495–504. https://doi.org/10.17763/haer.84.4.34j1g68140382063 - Chambers, D., & Carbonaro, M. (2020). BBC micro:bit and computational thinking in middle school: A case study. Education and Information Technologies, 25(4), 3283–3299. https://doi.org/10.1007/s10639-020-10122-z - George, D. (2020). MicroPython: Programming and usage for microcontrollers. MicroPython.org. https://micropython.org - Microsoft Research. (2018). Micro:bit Educational Foundation documentation. https://Micro:bit.org/

ANEXO 1.
SOLUCIÓN BÁSICA DEL RETO

A continuación, encontrarán una solución al reto planteado que puede simularse en https://python.Micro:bit.org/v/3 y probarse en el hardware real. El código en formato. .hex puede descargarse en los siguientes enlaces: TRANSMISOR y RECEPTOR

#TRANSMISOR

```
from microbit import *
import radio
import log

radio.on()
radio.config(group=1)
compass.calibrate()

muestra = 0
tiempo_inicial = running_time()

while True:
    muestra += 1
    tiempo_actual = running_time() - tiempo_inicial  # Tiempo transcurrido en milisegundos

    # Sensores
    temperatura = temperature()
    sonido = microphone.sound_level()
    luz = display.read_light_level()
    rumbo = compass.heading()

    # Registro interno en el log
    log.add({
        'sample': muestra,
        'time_ms': tiempo_actual,
        'temperature': temperatura,
        'sound': sonido,
        'light': luz,
        'heading': rumbo
    })

    # Envío por radio al receptor
    mensaje = "{},{},{},{},{},{}".format(
        muestra, tiempo_actual, temperatura, sonido, luz, rumbo
    )
    radio.send(mensaje)

    sleep(1000)  # Esperar 1 segundo
```

```
#RECEPTOR
from microbit import *
import radio

radio.on()
radio.config(group=1)

while True:
    mensaje = radio.receive()
    if mensaje:
        try:
            print(mensaje)
        except:
            pass
    sleep(50)
```

ANEXO 2.

MATERIAL DE APOYO

A continuación, encontrará algunos enlaces disponibles en el sitio web oficial de Micro:bit https://Microbit.org/ que serán útiles para iniciar el desarrollo del dispositivo.

- Uso de la brújula
  https://microbit.org/projects/make-it-code-it/compass-north
- Uso del display
  https://microbit.org/projects/make-it-code-it/animated-animals/?editor=python
- Uso de los botones a y b
  https://microbit.org/projects/make-it-code-it/get-silly/?editor=python
- Uso del acelerómetro
  https://microbit.org/projects/make-it-code-it/sensitive-step-counter/?editor=python
- Uso de la comunicación via RF
  https://microbit.org/projects/make-it-code-it/teleporting-duck/?editor=python