Instituto Tecnológico y de Estudios Superiores de Monterrey
Escuela de ingeniería

Materia:

Herramientas Computacionales: el arte de la programación

TC1001S.101

Actividad:

Semana Tec

Profesor designado:

Mauricio Paletta Nanarone

Fecha:

20 de Marzo de 2025

Elaborado por:

Zeus Joel Contreras Molina | A01770422

Índice

Actividad a realizar para la Semana Tec

El profesor nos dio instrucciones para programar un **robot previamente ensamblado** de la marca **Freenove**, modelo **Hexapod FNK0031**. Para ello, seguimos un proceso estructurado:

1. **Investigación completa sobre el robot**
2. **Preparación del ambiente de desarrollo**
3. **Diseño de un paquete en Python**
4. **Integración de Python con Arduino o Raspberry Pi**
5. **Cierre y documentación del proyecto**

Investigación sobre el robot

El **FNK0031 de Freenove** es un **robot hexápodo ensamblable**, con un **cuerpo caminante de seis patas** y una **cabeza giratoria** equipada con **una cámara y un sensor de distancia**. Requiere montaje y es compatible con **Arduino y Raspberry Pi**.

Características principales

- Cuerpo caminante con seis patas y cabeza giratoria.
- Cámara y sensor de distancia integrados.
- No incluye Raspberry Pi ni batería (se deben adquirir por separado).

Modelos compatibles con Raspberry Pi

Es compatible con las siguientes versiones de **Raspberry Pi**:

- Raspberry Pi 5, 4B, 3B+, 3B, 3A+
- Raspberry Pi 2B, B+, A+, Zero 1.3, Zero W *(requieren piezas adicionales)*
- Nota: El módulo LED no funciona con Raspberry Pi 5.

Métodos de control

El robot puede ser controlado de forma inalámbrica mediante:

- Dispositivos Android
- iPhone
- Computadora con Windows, macOS o Raspberry Pi OS

Programación y tutoriales

El robot se programa mediante el **IDE de Arduino** y es compatible con **Python**. Se proporciona un **tutorial detallado paso a paso**, con una guía de montaje y el código completo en Python.
El tutorial no está disponible en papel; el enlace de descarga se encuentra en la caja del producto.

Batería

No se incluye batería. Se recomienda consultar el **tutorial descargado** para conocer el tipo de batería compatible y cómo adquirirla.

Aquí tienes la documentación corregida y con los pasos de instalación traducidos para preparar el ambiente de desarrollo del robot:


ITC BOT 002

Preparación del Ambiente

Para comenzar con la preparación del ambiente, es necesario instalar el **Arduino IDE**, tal como se indica en el tutorial que se encuentra en el archivo ZIP disponible en Freenove. El archivo ZIP se llama **Freenove_Hexapod_Robot-Kit-master**, y dentro de él se encuentra el documento **Tutorial_for_V2.pdf**, que contiene las instrucciones detalladas y son las siguientes:

Instalación de Arduino IDE

El **Arduino IDE** es el software utilizado para programar el robot. Usa los lenguajes **C/C++** y es de código abierto.

Pasos para instalar Arduino IDE:

1. **Descargar Arduino IDE:**
    - Visita la página oficial de Arduino: https://www.arduino.cc/en/software.
    - Busca la sección **Legacy IDE (1.8.X)** y descarga la versión correspondiente a tu sistema operativo.
    - **Para usuarios de Windows:** Selecciona la opción **Windows Installer**.
    - **Importante:** No descargues la versión **Arduino IDE 2.x.x**, a menos que estés familiarizado con ella, ya que puede presentar problemas de compatibilidad.
    - **No se recomienda la versión "Windows App"**, ya que en ocasiones no funciona correctamente.
2. **Instalar Arduino IDE:**
    - Ejecuta el instalador y sigue las instrucciones en pantalla.
    - En Windows, durante la instalación, puede aparecer un cuadro de diálogo solicitando permisos para instalar controladores. Acepta la instalación.
    - Al finalizar, se creará un acceso directo en el escritorio.
3. **Configurar Arduino IDE:**
    - Abre el Arduino IDE.
    - En el menú **Herramientas**, selecciona:
        - **Placa:** "Arduino Mega o Mega 2560".
        - **Procesador:** "ATmega2560 (Mega 2560)".

        - **Puerto:** Selecciona el puerto correspondiente (puede ser COM4, COM5, etc., en Windows o `/dev/ttyUSB0`, `/dev/ttyACM0` en Linux/Mac).
    - Si no sabes qué puerto elegir, desconecta el USB, revisa los puertos disponibles, vuelve a conectar el USB y selecciona el puerto nuevo que aparezca.
4. **Subir un programa de prueba a la placa:**

    - Abre el ejemplo "Blink" desde **Archivo > Ejemplos > 01.Basics > Blink**.
    - Haz clic en el botón **Verificar** (✓) para compilar el código.
    - Luego, haz clic en **Subir** (→) para cargar el código en el robot.
    - Si todo está correcto, el LED marcado como **"L"** en la placa comenzará a parpadear, indicando que el código se está ejecutando.

Este proceso garantiza que el entorno de desarrollo esté listo para programar y controlar el robot Freenove Hexapod.

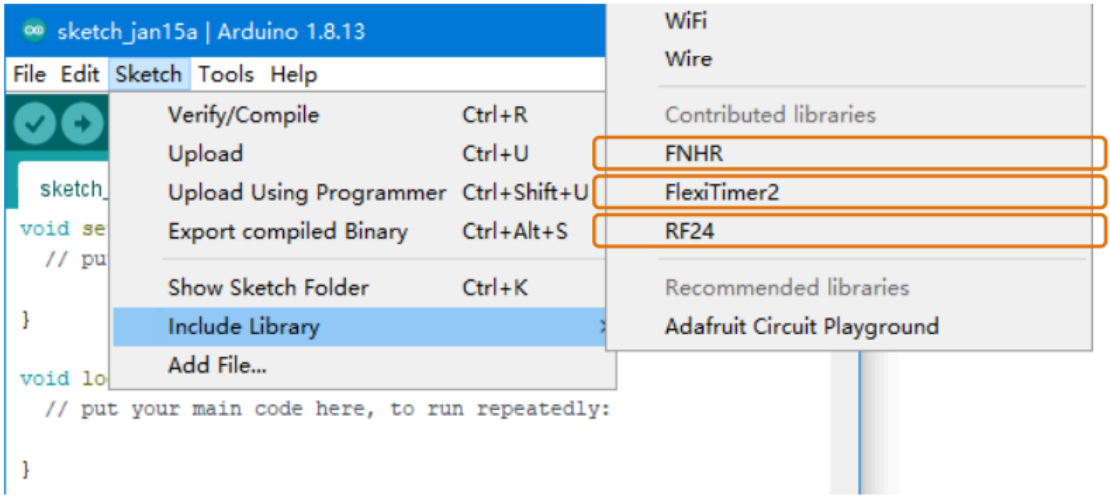
Este proceso garantiza que el entorno de desarrollo esté correctamente configurado para programar y controlar el **Freenove Hexapod**.

Una vez instalado el **Arduino IDE**, es necesario abrir un **Sketch** para verificar el funcionamiento de todas las piezas del robot. Para ello, se deben instalar las librerías **FNHR**, que incluyen cuatro paquetes esenciales para ejecutar los ejemplos proporcionados.

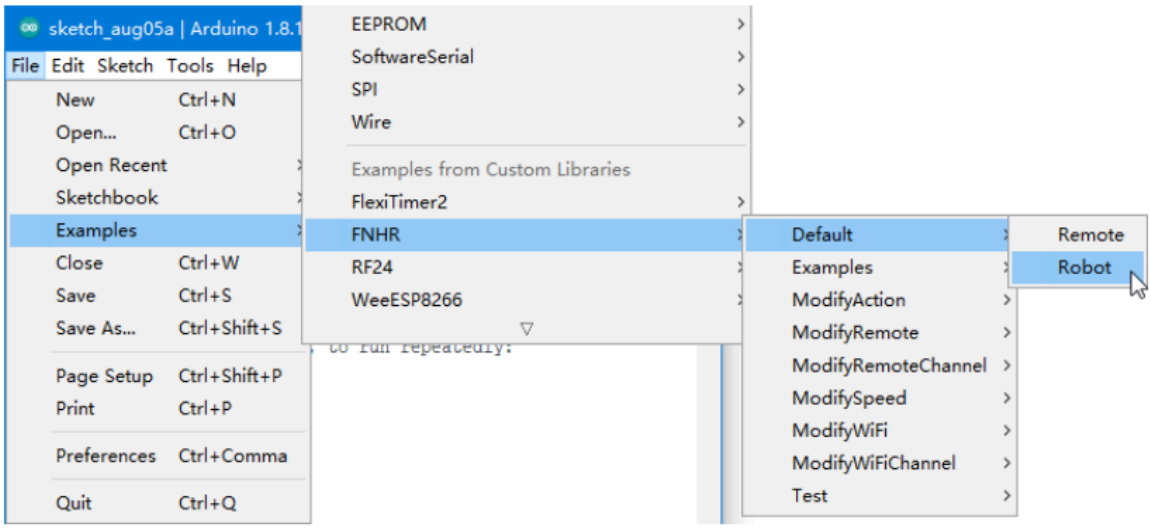
Instalación de las librerías FNHR en Arduino IDE

1. Abrir **Arduino IDE**.
2. Ir a **Sketch** > **Incluir biblioteca** > **Añadir biblioteca .ZIP**.
3. Seleccionar los archivos de las librerías **FNHR**, disponibles en: https://freenove.com/fnk0031.
4. Confirmar que las librerías se hayan agregado correctamente en **Sketch** > **Incluir biblioteca**.

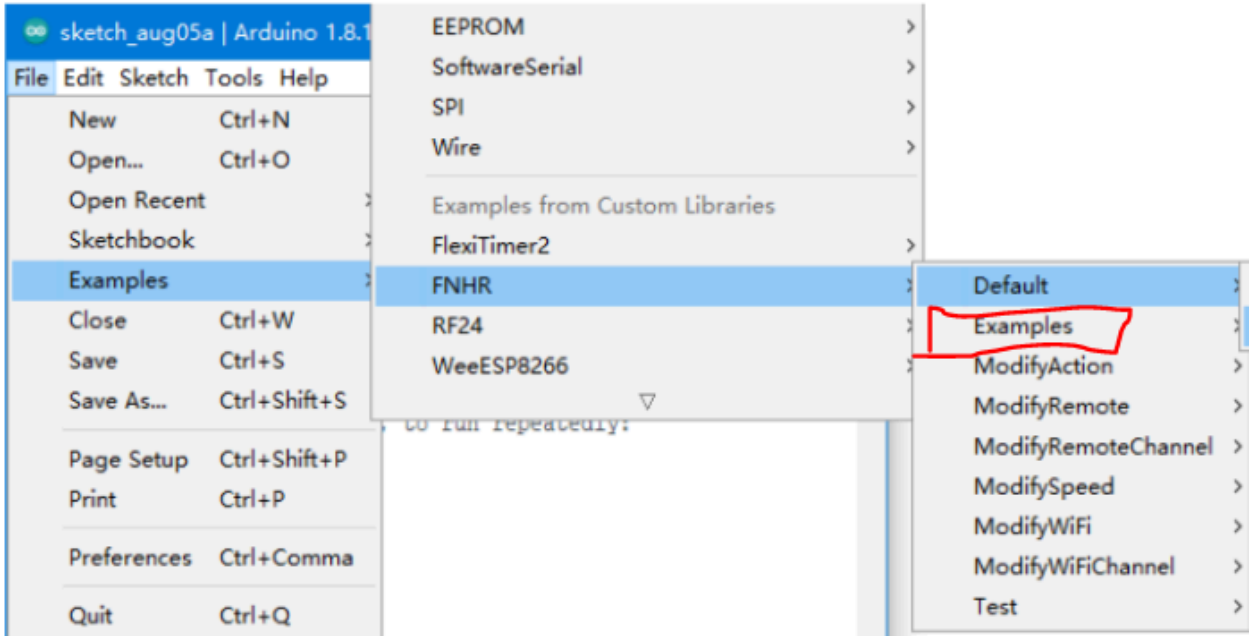
Una vez instaladas las librerías, podemos ejecutar los **ejemplos** incluidos para comprobar que el robot funciona correctamente.



Now open "File" > "Examples" > "FNHR" > "Default" > "Robot".



Después de realizar estos paso podemos empezar a correr los samples que se encuentran en el mismo lugar:



En esta sección, podemos encontrar los **examples** predefinidos por **Freenove** para cargar los códigos al robot y comprobar su correcto funcionamiento. Estos ejemplos permiten ejecutar diversas acciones, como:

- **Crawl (Gatear):** Hace que el robot avance unos pasos.
- **Twist (Meneo):** Controla los motores para inclinar su cuerpo y realizar un movimiento oscilatorio.
- **Otros examples:** Incluyen diversas funciones que permiten al robot ejecutar diferentes movimientos y acciones.

Instalación de Processing

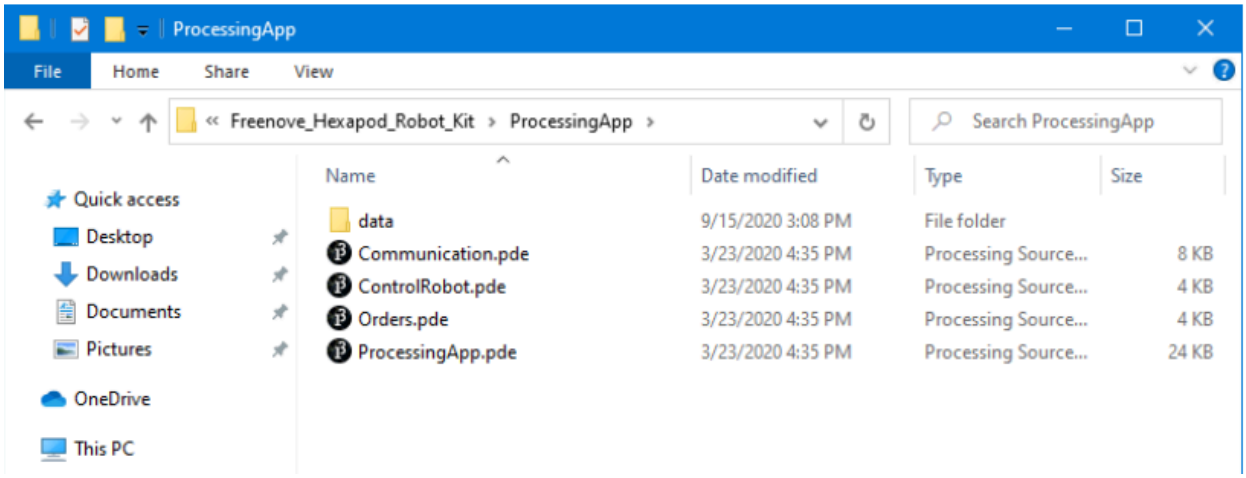
Para completar la instalación del software necesario, es fundamental instalar **Processing**, siguiendo las instrucciones detalladas en el manual del robot.

“Processing IDE utiliza el lenguaje de programación Java por defecto.
Visite https://processing.org/ y haga clic en "Documentación" > "Referencia" para obtener más información.
No se preocupe si no conoce Java, ya que le proporcionamos el código completo.
Primero, instale Processing IDE. Visite https://processing.org/ y haga clic en "Descargar" para acceder a la página de descarga.

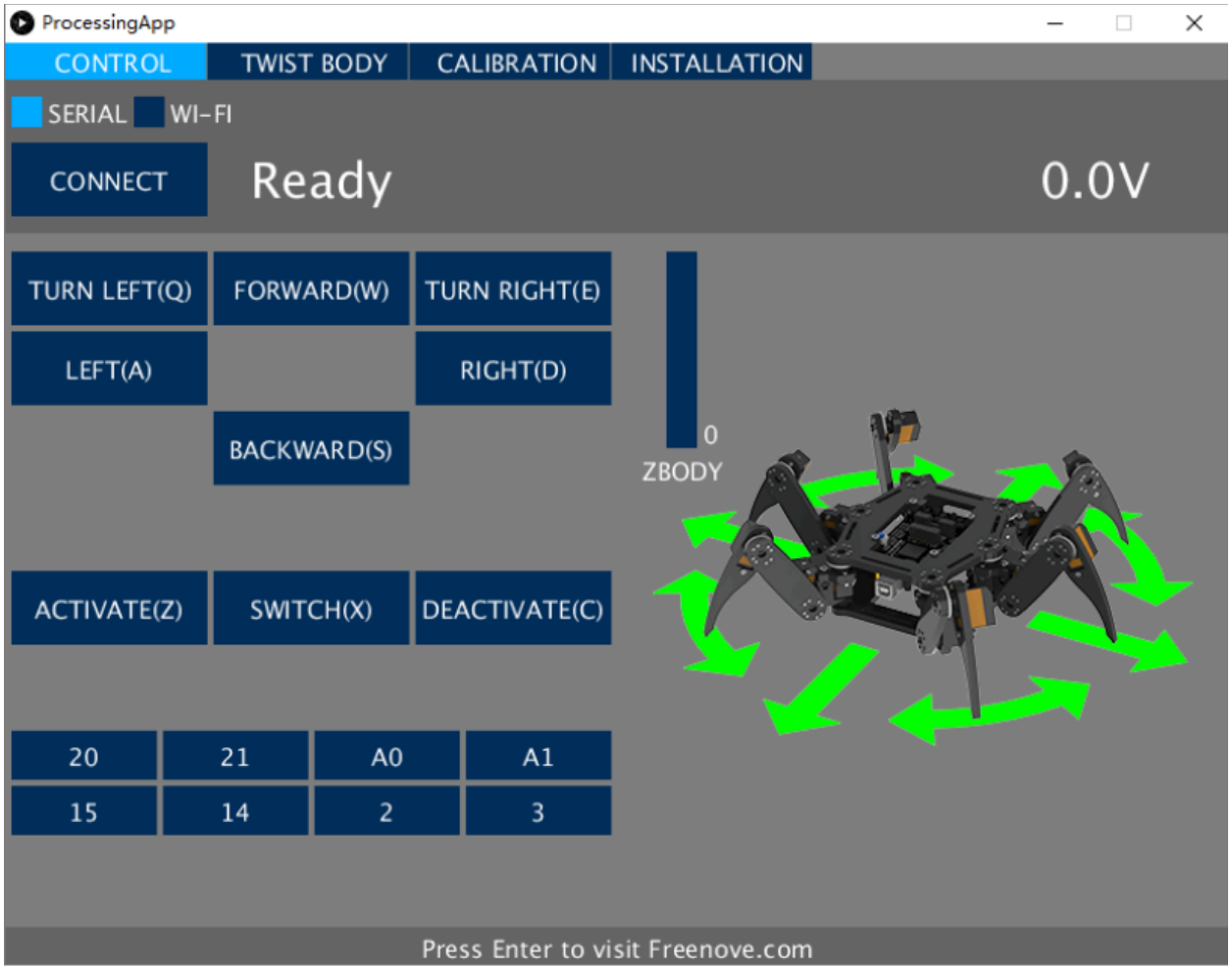
en paso simplificados después de la instalación del software processing abrimos un nuevo sketch, tomamos del archivo.zip del robot llamado PROCESSING y realizamos las conexiones que se nos muestran en el manual:

Note:

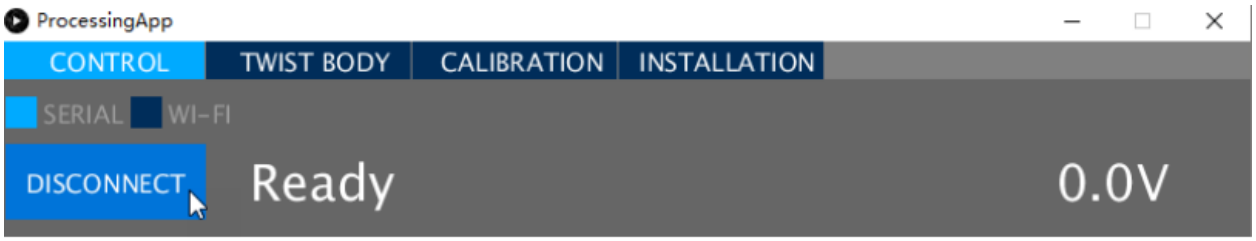
1. Make sure you have unzipped the downloaded file.
2. Keep the files in same folder. Do not seprate them.



y después de correr el script obtendremos la siguiente interfaz para realizar las conexiones antes mencionadas



Con el robot sin encender conectamos mediante un cable a la computadora donde deseamos que funcione y realizamos mediante la opción serial la instalación en el apartado de Instaltion y una vez que se descargue obtendremos esta pantalla:



Ahora si le ponemos baterias y encendemos el robot y mediante la consola nos damos cuenta si todos estos pasos se realizarón correctamente, con esto finalizamos la instalación de todo lo necesario.

Diseño de un paquete en Python e Integración de Python con Arduino o Raspberry Pi

Ahora, si colocamos las baterías y encendemos el robot, podemos verificar mediante la consola si todos los pasos previos se realizaron correctamente. Con esto, finalizamos la instalación de todo lo necesario para comenzar a programar el robot de manera más sencilla. El objetivo es que los estudiantes de preparatoria no tengan que escribir códigos tan largos y específicos, como los ejemplos actuales, que requieren personas con experiencia en el campo para desarrollarlos. Así, hemos desarrollado propuestas para crear nuestro 'facilitador'.

Mi propuesta para realizar esta comunicación entre nuestro entorno arduino y python es la libreria llamada Pyserial que funciona en python de la siguiente manera:

¿Cómo funciona pyserial con Arduino? Arduino ejecuta su propio código (cargado desde el IDE de Arduino).

El código de Python usa pyserial para enviar datos al Arduino (por ejemplo, encender un LED con un comando).

Arduino responde enviando datos de vuelta a Python si así está programado.

Un ejemplo de su uso sería:

(Arduino)

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    if (Serial.available() > 0) {
        char received = Serial.read();
        if (received == 'H') {
            Serial.println("Hola desde Arduino");
        }
    }
}
```

(Python)

```
import serial

arduino = serial.Serial("COM3", 9600)  # Cambia "COM3" por el puerto correcto

arduino.write(b"H")  # Envía 'H' a Arduino
respuesta = arduino.readline().decode().strip()
print(respuesta)  # Debería imprimir: "Hola desde Arduino"

arduino.close()
```

Ahora lo que buscamos es realizar esta misma conexión pero con movimientos ya específicos y programados en arduino para que solamente se mande a llamar a python como una variable más:

el código de arduino que le sería integrado al robot sería el siguiente:

(Arudino)

```
#ifndef ARDUINO_AVR_MEGA2560
#error "Por favor selecciona 'Arduino/Genuino Mega o Mega 2560' como placa."
#endif

#include <FNHR.h>  // Librería del robot

FNHR robot;  // Crear una instancia del robot

void setup() {
  Serial.begin(9600);  // Inicia la comunicación serial a 9600 baudios
  robot.Initialize();   // Inicializa el robot
  Serial.println("Robot activado. A la espera de instrucciones...");
}

void loop() {
  if (Serial.available() > 0) {
```

```
    String inputCommand = Serial.readStringUntil('\n');  // Lee el comando que se envía desde la computadora
    inputCommand.trim();  // Elimina espacios innecesarios
    Serial.print("Comando recibido: ");
    Serial.println(inputCommand);
    procesarComando(inputCommand);  // Llama a la función para procesar el comando
  }
}

void procesarComando(String inputCommand) {
  if (inputCommand == "AVANZAR") {
    Serial.println("Ejecutando movimiento hacia adelante...");
    robot.CrawlForward();  // Hace que el robot avance
  } else if (inputCommand == "RETROCEDER") {
    Serial.println("Ejecutando movimiento hacia atrás...");
    robot.CrawlBackward();  // Hace que el robot retroceda
  } else if (inputCommand == "GIRO_IZQUIERDO") {
    Serial.println("Ejecutando giro a la izquierda...");
    robot.TurnLeft();  // Hace que el robot gire a la izquierda
  } else if (inputCommand == "GIRO_DERECHO") {
    Serial.println("Ejecutando giro a la derecha...");
    robot.TurnRight();  // Hace que el robot gire a la derecha
  } else if (inputCommand == "BAILAR") {
    Serial.println("Ejecutando baile...");
```

```
    bailar();  // Llama a la función que hará que el robot baile
  } else {
    Serial.println("Comando no reconocido.");
  }
}

void bailar() {
  for (int i = 0; i < 3; i++) {  // Realiza 3 ciclos de baile
    robot.CrawlForward();  // Avanza
    delay(500);  // Espera 500ms
    robot.TurnRight();  // Gira a la derecha
    delay(500);  // Espera 500ms
    robot.CrawlBackward();  // Retrocede
    delay(500);  // Espera 500ms
    robot.TurnLeft();  // Gira a la izquierda
    delay(500);  // Espera 500ms
  }
  Serial.println("Baile terminado.");
}
```

Explicación

Configuración Inicial:

- Se incluye la librería `FNHR.h`, que contiene las funciones necesarias para controlar el robot.
- Se crea una instancia del objeto `robot`, que representa al robot en el programa.

Función `setup()`:

- Se inicia la comunicación serial a 9600 baudios, lo cual permite la interacción entre la computadora y el Arduino.
- Se inicializa el robot con el comando `robot.Initialize()`.
- Se imprime un mensaje en el monitor serial para indicar que el robot está listo para recibir comandos.

Función `loop()`:

- En un bucle continuo, se verifica si hay datos disponibles en el puerto serial.
- Si hay datos, se lee el comando enviado desde la computadora y se procesa.
- Luego, se llama a la función `procesarComando()` para ejecutar la acción correspondiente.

Función `procesarComando()`:

- Esta función verifica el comando recibido y ejecuta la acción correspondiente:
    - Si el comando es "AVANZAR", el robot se mueve hacia adelante.
    - Si el comando es "RETROCEDER", el robot retrocede.
    - Si el comando es "GIRO_IZQUIERDO", el robot gira a la izquierda.
    - Si el comando es "GIRO_DERECHO", el robot gira a la derecha.
    - Si el comando no es reconocido, se imprime "Comando no reconocido".

¿Qué hace este código?

Este código permite controlar el robot con comandos simples enviados desde una computadora o terminal serial. Los comandos permiten que el robot realice movimientos básicos como avanzar, retroceder, girar a la izquierda o a la derecha. Es un ejemplo básico de cómo controlar un robot utilizando Arduino y comunicación serial.

Código en Python mediante Pyserial:

```
import serial
import time

# Configuración de conexión serial (ajusta el puerto según tu sistema)
puerto = 'COM3'  # Cambia esto al puerto correcto
velocidad_baudios = 9600

# Intenta conectar con el Arduino
try:
    print(f"Conectando al Arduino en el puerto {puerto}...")
    arduino = serial.Serial(puerto, velocidad_baudios, timeout=1)
    time.sleep(2)  # Espera a que se establezca la conexión
    print("Conexión establecida con el Arduino.")
except Exception as e:
    print(f"Error al conectar con el Arduino: {e}")
    exit()

# Función para enviar comandos al Arduino
def enviar_comando(comando):
    """
    Envía un comando al Arduino y muestra un mensaje de depuración.
    """
    try:
        print(f"Enviando comando: {comando.strip()}")  # Muestra el comando enviado
        arduino.write(comando.encode())  # Envía el comando al Arduino
    except Exception as e:
        print(f"Error al enviar el comando: {e}")

# Función para leer la respuesta del Arduino
def leer_respuesta():
    """
    Lee la respuesta del Arduino y la muestra en la consola.
    """
    try:
```

```
        while arduino.in_waiting > 0:  # Verifica si hay datos disponibles
            respuesta = arduino.readline().decode().strip()  # Lee la respuesta
            print(f"Respuesta del Arduino: {respuesta}")
    except Exception as e:
        print(f"Error al leer la respuesta: {e}")

# Funciones que envían comandos al Arduino para controlar el robot
def controlar_robot(accion):
    """
    Función para controlar el robot con base en una acción.
    Las acciones posibles son 'avanzar', 'retroceder', 'girar_izquierda', 'girar_derecha'.
    """
    acciones = {
        'avanzar': "AVANZAR\n",
        'retroceder': "RETROCEDER\n",
        'girar_izquierda': "GIRO_IZQUIERDO\n",
        'girar_derecha': "GIRO_DERECHO\n"
    }

    if accion in acciones:
        enviar_comando(acciones[accion])
        leer_respuesta()
    else:
        print(f"Acción desconocida: {accion}")

def mover_pierna(pierna, x, y, z):
    """
    Mueve una pierna del robot a las coordenadas especificadas.
    """
    enviar_comando(f"MOVER_PIERNA {pierna} {x} {y} {z}\n")
    leer_respuesta()

def cambiar_altura(altura):
    """
    Cambia la altura del cuerpo del robot.
    """
    enviar_comando(f"AJUSTAR_ALTURA {altura}\n")
    leer_respuesta()

# Ejemplo de control del robot con variables sencillas
```

```
if __name__ == "__main__":
    print("Iniciando control del robot...")

    # Variables de control
    accion = 'avanzar'  # Puede ser 'avanzar', 'retroceder', 'girar_izquierda',
'girar_derecha'
    controlar_robot(accion)
    time.sleep(2)

    accion = 'girar_izquierda'
    controlar_robot(accion)
    time.sleep(2)

    accion = 'retroceder'
    controlar_robot(accion)
    time.sleep(2)

    accion = 'girar_derecha'
    controlar_robot(accion)
    time.sleep(2)

    # Control de piernas (puedes modificar las coordenadas)
    mover_pierna(1, 10, 20, 30)  # Mueve la pierna 1 a las coordenadas (10, 20, 30)
    time.sleep(2)

    # Cambiar altura (puedes cambiar el valor de altura)
    cambiar_altura(15)
    time.sleep(2)

    print("Control del robot completado.")

    # Cerrar la conexión serial
    arduino.close()
    print("Conexión serial cerrada.")



import serial
import time
```

```
# Configuración de conexión serial (ajusta el puerto según tu sistema)
puerto = 'COM3'  # Cambia esto al puerto correcto
velocidad_baudios = 9600

# Intenta conectar con el Arduino
try:
    print(f"Conectando al Arduino en el puerto {puerto}...")
    arduino = serial.Serial(puerto, velocidad_baudios, timeout=1)
    time.sleep(2)  # Espera a que se establezca la conexión
    print("Conexión establecida con el Arduino.")
except Exception as e:
    print(f"Error al conectar con el Arduino: {e}")
    exit()

# Función para enviar comandos al Arduino
def enviar_comando(comando):
    """
    Envía un comando al Arduino y muestra un mensaje de depuración.
    """
    try:
        print(f"Enviando comando: {comando.strip()}")  # Muestra el comando enviado
        arduino.write(comando.encode())  # Envía el comando al Arduino
    except Exception as e:
        print(f"Error al enviar el comando: {e}")

# Función para leer la respuesta del Arduino
def leer_respuesta():
    """
    Lee la respuesta del Arduino y la muestra en la consola.
    """
    try:
        while arduino.in_waiting > 0:  # Verifica si hay datos disponibles
            respuesta = arduino.readline().decode().strip()  # Lee la respuesta
            print(f"Respuesta del Arduino: {respuesta}")
    except Exception as e:
        print(f"Error al leer la respuesta: {e}")

# Funciones que envían comandos al Arduino para controlar el robot
def controlar_robot(accion):
    """
```

```
    Función para controlar el robot con base en una acción.
    Las acciones posibles son 'avanzar', 'retroceder', 'girar_izquierda', 'girar_derecha'.
    """
    acciones = {
        'avanzar': "AVANZAR\n",
        'retroceder': "RETROCEDER\n",
        'girar_izquierda': "GIRO_IZQUIERDO\n",
        'girar_derecha': "GIRO_DERECHO\n"
    }

    if accion in acciones:
        enviar_comando(acciones[accion])
        leer_respuesta()
    else:
        print(f"Acción desconocida: {accion}")

def mover_pierna(pierna, x, y, z):
    """
    Mueve una pierna del robot a las coordenadas especificadas.
    """
    enviar_comando(f"MOVER_PIERNA {pierna} {x} {y} {z}\n")
    leer_respuesta()

def cambiar_altura(altura):
    """
    Cambia la altura del cuerpo del robot.
    """
    enviar_comando(f"AJUSTAR_ALTURA {altura}\n")
    leer_respuesta()

# Ejemplo de control del robot con variables sencillas
if __name__ == "__main__":
    print("Iniciando control del robot...")

    # Variables de control
    accion = 'avanzar'  # Puede ser 'avanzar', 'retroceder', 'girar_izquierda',
'girar_derecha'
    controlar_robot(accion)
    time.sleep(2)
```

```
accion = 'girar_izquierda'
controlar_robot(accion)
time.sleep(2)

accion = 'retroceder'
controlar_robot(accion)
time.sleep(2)

accion = 'girar_derecha'
controlar_robot(accion)
time.sleep(2)

# Control de piernas (puedes modificar las coordenadas)
mover_pierna(1, 10, 20, 30)  # Mueve la pierna 1 a las coordenadas (10, 20, 30)
time.sleep(2)

# Cambiar altura (puedes cambiar el valor de altura)
cambiar_altura(15)
time.sleep(2)

print("Control del robot completado.")

# Cerrar la conexión serial
arduino.close()
print("Conexión serial cerrada.")
```

Explicación:

1. **Configuración de conexión serial**: Se conecta al Arduino utilizando el puerto especificado y la velocidad de baudios (`9600`).

2. **Función `enviar_comando`**: Envía un comando al Arduino y muestra un mensaje de depuración para indicar que el comando ha sido enviado correctamente.

3. **Función `leer_respuesta`**: Lee cualquier respuesta enviada desde el Arduino y la muestra en la consola.

4. **Funciones de control del robot**:
    - **`controlar_robot`**: Envia comandos de control como 'avanzar', 'retroceder', 'girar_izquierda', 'girar_derecha'.
    - **`mover_pierna`**: Mueve una pierna a las coordenadas especificadas.
    - **`n`** Ajusta la altura del robot.
5. **Flujo principal (`if __name__ == "__main__":`)**: Este bloque contiene las acciones que el robot realizará, como avanzar, retroceder, girar, mover piernas y cambiar altura.

6. **Cierre de conexión**: Al final, se cierra la conexión seria.

Este código es muy sencillo y debería ser accesible para alguien sin mucha experiencia en programación, permitiéndoles controlar el robot a través de comandos simples.

un ejemplo de su aplicación sería:

```
import serial

import time


# Configuración de conexión serial (ajusta el puerto según tu sistema)

puerto = 'COM3'  # Cambia esto al puerto correcto

velocidad_baudios = 9600


# Intenta conectar con el Arduino

try:

    print(f"Conectando al Arduino en el puerto {puerto}...")

    arduino = serial.Serial(puerto, velocidad_baudios, timeout=1)

    time.sleep(2)  # Espera a que se establezca la conexión
```

```
    print("Conexión establecida con el Arduino.")
except Exception as e:
    print(f"Error al conectar con el Arduino: {e}")
    exit()

# Función para enviar comandos al Arduino
def enviar_comando(comando):
    """
    Envía un comando al Arduino.
    """
    try:
        arduino.write(comando.encode())  # Envía el comando al Arduino
        print(f"Comando enviado: {comando.strip()}")
    except Exception as e:
        print(f"Error al enviar el comando: {e}")

# Función para realizar una acción en el robot
def ejecutar_accion(accion):
    """
    Ejecuta una acción simple para controlar el robot.
    """
    acciones = {
        'avanzar': "AVANZAR\n",
```

```
        'retroceder': "RETROCEDER\n",
        'girar_izquierda': "GIRO_IZQUIERDO\n",
        'girar_derecha': "GIRO_DERECHO\n",
        'bailar': "BAILE\n"
    }

    if accion in acciones:
        enviar_comando(acciones[accion])
    else:
        print(f"Acción desconocida: {accion}")

# Función para detener el robot
def detener_robot():
    """
    Detiene el robot de inmediato.
    """
    enviar_comando("DETENER\n")

# Ejemplo de uso del robot
if __name__ == "__main__":
    print("Iniciando control del robot...")

    # 1. El robot avanza hacia enfrente
```

```
ejecutar_accion('avanzar')
time.sleep(2)

# 2. El robot gira a la izquierda
ejecutar_accion('girar_izquierda')
time.sleep(2)

# 3. El robot comienza a bailar
ejecutar_accion('bailar')
time.sleep(5)  # El robot baila durante 5 segundos

# 4. El robot se detiene
detener_robot()
print("El robot ha detenido su acción.")

# Cerrar la conexión serial
arduino.close()
print("Conexión serial cerrada.")
```

1. **Funciones de acción simples**:
    - `ejecutar_accion(accion)`: Esta función recibe el nombre de la acción (como `avanzar`, `retroceder`, `girar_izquierda`, `bailar`) y envía el comando correspondiente. El usuario solo necesita ingresar el nombre de la acción.

    - Las acciones son mapeadas a comandos muy fáciles de entender, como "AVANZAR" o "GIRO_IZQUIERDO".

2. **Detener el robot**: La función `detener_robot()` detiene inmediatamente al robot al enviar el comando `"DETENER\n"`, para que no haya que preocuparse de cómo detenerlo en cada caso.

3. **Comandos simplificados**: Ya no hace falta que el usuario entienda cómo se mueven las piernas o el cuerpo del robot. Solo le dice al robot lo que quiere hacer con un nombre simple, como `bailar`, `avanzar`, o `girar_izquierda`.

Ejemplo de ejecución:

1. El robot avanza (`avanzar`).

2. El robot gira a la izquierda (`girar_izquierda`).

3. El robot baila (`bailar` durante 5 segundos).

Cierre y documentación del proyecto

Implementación a futuro:

Implementación como Librería en Python

A futuro, el sistema funcionará de manera similar a la librería *Tello*, usada para drones. Se desarrollará un paquete de Python que permita controlar el hexápodo con funciones intuitivas como:

- La librería se encargará de la comunicación con Arduino de forma transparente para el usuario.

- Se incluirían modos predefinidos como secuencias de baile y movimientos sincronizados.

- Se integraría un instructivo detallado, explicando el uso sin necesidad de conocimientos previos en programación.

Este enfoque permitirá que cualquier usuario, sin importar su nivel de experiencia, pueda programar y controlar el hexápodo de manera sencilla, transformándolo en una herramienta educativa y de desarrollo avanzada.

Conclusión del Proyecto

Durante la **Semana Tec**, nuestro equipo de cinco integrantes trabajó en la programación y control del **robot hexápodo Freenove FNK0031**. Siguiendo un enfoque estructurado, investigamos el funcionamiento del robot, preparamos el entorno de desarrollo y diseñamos el código necesario para su operación. Nuestra intención era lograr una comunicación fluida entre **Arduino y Python** mediante **Pyserial**, permitiendo el control del robot con comandos intuitivos.

Sin embargo, a pesar de nuestros esfuerzos, nos enfrentamos a un problema crítico: **el robot dejó de funcionar tras la implementación de Pyserial**. A partir de ese momento, **incluso los ejemplos oficiales de Freenove dejaron de ejecutarse correctamente**, lo que nos impidió llevar a la práctica nuestras pruebas y validaciones finales.

Ante esta situación, en lugar de abandonar el proyecto, decidimos documentar de manera teórica cómo **llevaríamos a cabo la implementación de este sistema si el robot estuviera en condiciones óptimas**. Así, desarrollamos una propuesta detallada para estructurar la comunicación entre **Python y Arduino**, diseñando un código modular que permitiría:

1. **Controlar el robot con comandos simples** como avanzar, retroceder y girar.
2. **Implementar una estructura de librería en Python**, similar a las utilizadas en drones, para facilitar su uso sin necesidad de conocimientos avanzados de programación.
3. **Optimizar la comunicación serial**, asegurando una mejor integración entre Arduino y Raspberry Pi.
4. **Desarrollar una interfaz más accesible**, que permitiera controlar el robot de manera intuitiva.

Si hubiéramos podido continuar con la ejecución práctica, nuestra solución habría permitido a cualquier usuario operar el robot sin complicaciones, convirtiéndolo en una excelente herramienta educativa. Aunque no logramos probar nuestro código en hardware real, nuestra documentación y el diseño de la solución pueden servir como **base para futuros proyectos**, asegurando que equipos posteriores puedan retomar y mejorar nuestra propuesta.

A pesar de las dificultades, este proyecto nos permitió reforzar nuestras habilidades en **programación, resolución de problemas y trabajo en equipo**, dejándonos con valiosas lecciones para el futuro.