INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

SEMANA TEC: “ HERRAMIENTAS COMPUTACIONALES : EL ARTE DE LA PROGRAMACIÓN”

PROFESOR: Dr. Mauricio Paletta

Hannah Nahid Becerril Guadarrama |A01769561

21 de Marzo 2025

Objetivo del Proyecto:

En esta semana TEC hacemos uso de todo lo que hemos aprendido en casi 2 años de nuestra carrera, hablando no sólo de conocimiento de programación sino análisis y habilidad en resolución de problemas que identifican a cualquier ingeniero. El reto fue simple, facilitar la programación de un robot para estudiantes de preparatoria interesados en la carrera ITC, para lograrlo era clave crear una librería importando funciones establecidas en un archivo arduino (.ino) para ser utilizado en el lenguaje de programación Python por los usuarios, evitando el contacto directo con lenguajes más complejos y en general haciendo más fácil su uso.

Códigos SRC verificados :

Como equipo lo primero que hicimos fue realizar una investigación acerca del funcionamiento y la correcta instalación del ambiente requerido para poder mandar los comandos al robot, esto también con el objetivo de poder decidir más adelante que aproche a la solución queríamos tomar dependiendo de su facilidad y su beneficio.
Al término de dicha investigación, concluimos que en base al sistema operativo y al funcionamiento general del robot lo mejor para este proyecto era utilizar la librería de python llamada pyserial. Este enlace sucede gracias a la comunicación serial, método para transmitir datos entre dos dispositivos utilizando un cable (o conexión inalámbrica) y un protocolo específico.
En este caso:

-Arduino: Actúa como el dispositivo que recibe comandos y ejecuta acciones.

-Python: Actúa como el dispositivo que envía comandos y recibe respuestas.

Para lograr una comunicación efectiva se debe tomar en cuenta los siguientes aspectos, así como el siguiente flujo de trabajo:

-Ambos dispositivos deben de estar configurados con los mismos parámetros puerto (COM3 o COM4 en os Windows), velocidad de baudios, y timeout(tiempo máximo que python esperará la respuesta de arduino)

-El envío de comandos se hace desde Python, donde se envían como cadenas de texo a través del puerto serial, haciendo uso de diferentes comandos como write() y command. encode()

-La recepción de los comandos se hace desde el código de arduino usando la función Serial.readStringUntil('\n'), que lee los datos recibidos hasta un salto de línea lo que indica el fin del comando.

-Arduino ejecuta la o las acciones correspondientes.

Cabe mencionar que en ambos dispositivos hicimos uso de conocimientos básicos como la creación y manejo de funciones, creación y manejo de variables, ciclos , condicionales , manejo de excepciones entre otros.

Los códigos anteriores son:

CÓDIGO ARDUINO:

#ifndef ARDUINO_AVR_MEGA2560

```
#error Wrong board. Please choose "Arduino/Genuino Mega or Mega 2560"
#endif

#include <FNHR.h>

FNHR robot;

void setup() {
  Serial.begin(9600); // Inicia la comunicación serial
  robot.Start();      // Inicia el robot
  Serial.println("Robot listo. Esperando comandos...");
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n'); // Lee el comando desde Python
    command.trim(); // Elimina espacios y saltos de línea
    Serial.print("Comando recibido: ");
    Serial.println(command);
    executeCommand(command); // Ejecuta el comando
  }
}

void executeCommand(String command) {
  if (command == "CRAWL_FORWARD") {
    Serial.println("Ejecutando CrawlForward()");
    robot.CrawlForward();
  } else if (command == "CRAWL_BACKWARD") {
    Serial.println("Ejecutando CrawlBackward()");
    robot.CrawlBackward();
  } else if (command == "TURN_LEFT") {
    Serial.println("Ejecutando TurnLeft()");
    robot.TurnLeft();
  } else if (command == "TURN_RIGHT") {
    Serial.println("Ejecutando TurnRight()");
    robot.TurnRight();
  } else if (command.startsWith("LEG_MOVE")) {
    int leg = command.substring(9, 10).toInt();
    float x = command.substring(11, 20).toFloat();
    float y = command.substring(21, 30).toFloat();
    float z = command.substring(31, 40).toFloat();
    Serial.println("Ejecutando LegMoveToRelatively()");
    robot.LegMoveToRelatively(leg, x, y, z);
  } else if (command.startsWith("CHANGE_HEIGHT")) {
    float height = command.substring(14).toFloat();
    Serial.println("Ejecutando ChangeBodyHeight()");
    robot.ChangeBodyHeight(height);
  } else if (command.startsWith("ROTATE_BODY")) {
```

```
    float x = command.substring(12, 21).toFloat();
    float y = command.substring(22, 31).toFloat();
    float z = command.substring(32, 41).toFloat();
    Serial.println("Ejecutando RotateBody()");
    robot.RotateBody(x, y, z);
  } else if (command.startsWith("TWIST_BODY")) {
    float xMove = command.substring(11, 20).toFloat();
    float yMove = command.substring(21, 30).toFloat();
    float zMove = command.substring(31, 40).toFloat();
    float xRotate = command.substring(41, 50).toFloat();
    float yRotate = command.substring(51, 60).toFloat();
    float zRotate = command.substring(61, 70).toFloat();
    Serial.println("Ejecutando TwistBody()");
    robot.TwistBody(xMove, yMove, zMove, xRotate, yRotate, zRotate);
  } else if (command.startsWith("SET_SPEED")) {
    float speed = command.substring(10).toFloat();
    Serial.println("Ejecutando SetActionSpeed()");
    robot.SetActionSpeed(speed);
  }
}
```

CÓDIGO PYTHON :

```
import serial
import time

# Configura el puerto serial (ajusta el puerto COM según tu sistema)
puerto = 'COM3'  # Cambia esto al puerto correcto
velocidad_baudios = 9600

try:
    # Intenta abrir la conexión serial
    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()

def send_command(command):
    """
    Envía un comando al Arduino y muestra un mensaje de depuración.
    """
    try:
        print(f"Enviando comando: {command.strip()}")  # Muestra el comando enviado
```

```
        arduino.write(command.encode())  # Envía el comando al Arduino
    except Exception as e:
        print(f"Error al enviar el comando: {e}")

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 de control del robot
def crawl_forward():
    send_command("CRAWL_FORWARD\n")
    leer_respuesta()

def crawl_backward():
    send_command("CRAWL_BACKWARD\n")
    leer_respuesta()

def turn_left():
    send_command("TURN_LEFT\n")
    leer_respuesta()

def turn_right():
    send_command("TURN_RIGHT\n")
    leer_respuesta()

def leg_move_to_relatively(leg, x, y, z):
    send_command(f"LEG_MOVE {leg} {x} {y} {z}\n")
    leer_respuesta()

def change_body_height(height):
    send_command(f"CHANGE_HEIGHT {height}\n")
    leer_respuesta()

# Ejemplo de uso
print("Iniciando prueba del robot...")
crawl_forward()  # El robot avanza
time.sleep(2)
turn_left()      # El robot gira a la izquierda
time.sleep(2)
crawl_backward() # El robot retrocede
time.sleep(2)
```

```
turn_right()    # El robot gira a la derecha
time.sleep(2)
print("Prueba completada.")

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

Mejoras por hacer:

Debido a la pérdida de conexión con el robot algunas mejoras al código no se realizaron por esta falla, como lo son:

1. Traducción de nombre de las funciones: En el script original los nombres de las funciones que describen los movimientos del robot están en el idioma inglés, por lo que su uso por un alumno que desconoce total o parcialmente del lenguaje se puede ver muy afectado. Para hacerlo
2. Agregar una cola de comandos: El implementar una cola de comandos en arduino para almacenar y ejecutar los comandos en orden sin necesidad de una conexión seria entre el robot y nuestro código.
3. Predeterminación de Parámetros: Existen algunas funciones que hacen uso de parámetros proporcionados por los usuarios desde Python para que después arduino ejecute los comandos correspondientes, sin embargo algunas de ellas tienen parámetros que pueden resultar algo difíciles de entender sin o con poc experiencia en ámbitos como la programación e incluso matemáticas, por lo que la predeterminación de parámetros evita que los usuarios manejen este tipo de conceptos y simplemente manden llamar las funciones.
4. Manejo de errores para evitar fallos en la comunicación de arduino.

Siendo los códigos finales:

Código de Arduino Mejorado:

```
#ifndef ARDUINO_AVR_MEGA2560
#error Placa incorrecta. Por favor, elija "Arduino/Genuino Mega o Mega 2560"
#endif

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

FNHR robot; // Crear objeto robot

// Cola de comandos
String colaComandos[10]; // Arreglo para almacenar comandos (hasta 10)
int inicio = 0, fin = 0; // Índices para manejar la cola

void setup() {
  Serial.begin(9600); // Inicia la comunicación serial
  robot.Start();      // Inicia el robot
  Serial.println("Robot listo. Esperando comandos...");
```

```
}

void loop() {
  // Recibe comandos desde la computadora
  if (Serial.available() > 0) {
    String comando = Serial.readStringUntil('\n'); // Lee el comando
    comando.trim(); // Elimina espacios y saltos de línea
    Serial.print("Comando recibido: ");
    Serial.println(comando);

    // Agregar comando a la cola
    if ((fin + 1) % 10 != inicio) { // Evita desbordamiento de la cola
      colaComandos[fin] = comando;
      fin = (fin + 1) % 10;
    } else {
      Serial.println("Cola de comandos llena, comando descartado.");
    }
  }

  // Ejecuta comandos de la cola
  if (inicio != fin) {
    ejecutarComando(colaComandos[inicio]);
    inicio = (inicio + 1) % 10;
  }
}

void ejecutarComando(String comando) {
  if (comando == "AVANZAR") {
    Serial.println("Ejecutando Avanzar()");
    robot.CrawlForward();
  } else if (comando == "RETROCEDER") {
    Serial.println("Ejecutando Retroceder()");
    robot.CrawlBackward();
  } else if (comando == "GIRAR_IZQUIERDA") {
    Serial.println("Ejecutando GirarIzquierda()");
    robot.TurnLeft();
  } else if (comando == "GIRAR_DERECHA") {
    Serial.println("Ejecutando GirarDerecha()");
    robot.TurnRight();
  } else if (comando.startsWith("MOVER_PATA")) {
    int pata = comando.substring(11, 12).toInt();
    float x = comando.substring(13, 22).toFloat();
    float y = comando.substring(23, 32).toFloat();
    float z = comando.substring(33, 42).toFloat();
    Serial.println("Ejecutando MoverPata()");
    robot.LegMoveToRelatively(pata, x, y, z);
  } else if (comando.startsWith("AJUSTAR_ALTURA")) {
    float altura = comando.substring(15).toFloat();
```

```
    Serial.println("Ejecutando AjustarAltura()");
    robot.ChangeBodyHeight(altura);
  } else {
    Serial.println("Comando no reconocido.");
  }
}
```

CÓDIGO PYTHON MEJORADO:

```
import serial
import time

# Configuración del puerto serial
puerto = 'COM3'  # Cambia esto según tu sistema operativo
velocidad_baudios = 9600

try:
    print(f"Conectando al Arduino en {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()

def enviar_comando(comando):
    """
    Envía un comando al Arduino y espera su respuesta.
    """
    try:
        print(f"Enviando: {comando.strip()}")  # Muestra el comando enviado
        arduino.write(comando.encode())  # Envía el comando
        time.sleep(0.1)  # Pequeña pausa para estabilidad
    except Exception as e:
        print(f"Error al enviar el comando: {e}")

def leer_respuesta():
    """
    Lee la respuesta del Arduino.
    """
    try:
        while arduino.in_waiting > 0:
            respuesta = arduino.readline().decode().strip()
            print(f"Arduino: {respuesta}")
    except Exception as e:
        print(f"Error al leer la respuesta: {e}")
```

```
# Funciones de control del robot con nombres en español y valores predeterminados
def avanzar():
    enviar_comando("AVANZAR\n")
    leer_respuesta()

def retroceder():
    enviar_comando("RETROCEDER\n")
    leer_respuesta()

def girar_izquierda():
    enviar_comando("GIRAR_IZQUIERDA\n")
    leer_respuesta()

def girar_derecha():
    enviar_comando("GIRAR_DERECHA\n")
    leer_respuesta()

def mover_pata(pata=1, x=5.0, y=5.0, z=0.0):
    """
    Mueve una pata del robot a una posición relativa.
    Parámetros predeterminados para evitar valores difíciles de entender.
    """
    enviar_comando(f"MOVER_PATA {pata} {x} {y} {z}\n")
    leer_respuesta()

def ajustar_altura(altura=10.0):
    """
    Ajusta la altura del robot.
    Altura predeterminada de 10.0 para facilitar el uso.
    """
    enviar_comando(f"AJUSTAR_ALTURA {altura}\n")
    leer_respuesta()

# Ejemplo de uso del robot
print("Prueba del robot iniciada...")
avanzar()
time.sleep(2)
girar_izquierda()
time.sleep(2)
retroceder()
time.sleep(2)
girar_derecha()
time.sleep(2)
mover_pata()  # Se mueve con valores predeterminados
time.sleep(2)
ajustar_altura()  # Se ajusta a la altura predeterminada
time.sleep(2)
print("Prueba completada.")
```

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

Ya terminando los pasos anteriores el paso final es transformar los códigos en una librería para que los usuarios lo puedan descargar directamente la carpeta para que puedan hacer uso de la misma, lo principal es organizar la estructura del proyecto para que sea fácil de entender y siga las convenciones de python, en este caso la estructura que propongo es:

```
robot/                  # Carpeta principal del proyecto
│
├── arduino_code/       # Carpeta para el código de Arduino
│   └── robot_control.ino   # Archivo con el código de Arduino
│
├── python_code/        # Carpeta para el código de Python
│   └── robot_control.py    # Archivo con el código de Python
│
└── README.md           # Archivo de instrucciones
```

Los pasos para usar la carpeta y crear un código personalizado de comandos en python son:

1. Los alumnos deben descargar la carpeta completa
2. Abrir el archivo robot_control.ino en el IDE de arduino, conectar el arduino a la computadora, seleccionar la placa correcta (Arduino Mega 2560) y el puerto COM correspondiente
3. Cargar el código de arduino al robot.
4. Abrir el archivo robot_control.py en un editor de código (asegurarse de haber instalado previamente la librería pyserial con el comando “pip install pyserial” en la terminal de comandos de windows)
5. Modificar el puerto serial en el código de Python (puerto = 'COM3') para que coincida con el puerto donde está conectado el Arduino.
6. Ejecutar el código de Python para controlar el robot.
7. Los estudiantes pueden crear un archivo de python en la carpeta python_code, en este archivo pueden importar y usar las funciones del archivo robot_control.py para crear sus propios programas

Ejemplo de archivo personalizado creado por alumnos de prepa:

```
# python_code/mi_codigo.py
from robot_control import Robot
import time

# Crear una instancia del robot
robot = Robot(puerto='COM3')  # Cambiar el puerto si es necesario

# Conectar con el Arduino
robot.conectar()
```

```
# Ejecutar movimientos personalizados
print("Iniciando secuencia personalizada...")
robot.avanzar()
time.sleep(2)
robot.girar_izquierda()
time.sleep(2)https://github.com/TC1001S-2025/Hexapod.git
robot.retroceder()
time.sleep(2)

# Cerrar la conexión
robot.cerrar_conexion()
print("Secuencia completada.")
```

Con estos cambios los usuarios pueden manipular fácilmente el robot sin necesidad de hacer o comprender conceptos más complejos logrando el objetivo inicial.