Clase:

Herramientas computacionales: el arte de la programación

Actividad colaborativa:

Reporte

Profesores:

Doc. Mauricio Paletta

Santiago Sanabria Basurto - A01773072

Grupo: 101

Fecha de entrega: Marzo 21, 2025

Objetivo del Proyecto

El objetivo del proyecto era desarrollar una librería que permitiera conectar Arduino IDE con Python para controlar el robot FNK0031, enviando comandos desde Python y modificando su comportamiento. Además, se buscaba traducir las funciones del robot de inglés a español para facilitar su uso.

Avance del Proyecto

Conexión Establecida:

- La conexión entre Arduino y Python fue lograda de manera exitosa utilizando la librería pyserial, esta nos sirve para enviar y recibir comandos a través del puerto serial.

Comunicación Funcional:

- Las funciones principales del robot, fueron implementadas correctamente.
- Los comandos fueron enviados desde Python y ejecutados por el robot vía puerto de manera exitosa.

Traducción de Funciones:

- Las funciones de control del robot fueron traducidas al español para facilitar la comprensión y modificación desde Python.

Funcionamiento de la Conexión

1. **Código en Python:**
    - Abre la conexión serial utilizando pyserial para comunicarse con el puerto COM del Arduino.
    - Envía comandos al robot mediante la función send_command().
    - Lee las respuestas del robot usando leer_respuesta() para verificar la ejecución de los comandos.
2. **Código en Arduino:**
    - Recibe comandos desde el puerto serial.
    - Guarda los comandos en la **EEPROM** para ejecutarlos incluso después de reiniciar el robot.
    - Ejecuta los comandos mediante la función executeCommand() para controlar las acciones del robot.

Manual de Uso del Programa en Python

Instalación de Dependencias

Ejecutar el siguiente comando para instalar `pyserial`:

pip install pyserial

Configuración del Puerto Serial

Cambiar el valor de `puerto` en el script de Python para que coincida con el puerto COM donde está conectado el Arduino:

```
puerto = 'COM3'  # Cambiar a COM correcto
velocidad_baudios = 9600
```

Ejecución del Código Python

Ejecutar el script para iniciar la comunicación con Arduino:

```
python control_robot.py
```

Usar las funciones para controlar el robot:

```
crawl_forward()  # Avanza
turn_left()      # Gira a la izquierda
leg_move_to_relatively(1, 10, 5, 0)  # Mueve la pata
```

Código de Python

```
import serial
import time

# Configuración del puerto serial
puerto = 'COM3'
velocidad_baudios = 9600

try:
    print(f"Conectando al Arduino en el puerto {puerto}...")
    arduino = serial.Serial(puerto, velocidad_baudios, timeout=1)
    time.sleep(2)
    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."""
    try:
        print(f"Enviando comando: {comando.strip()}")
        arduino.write(comando.encode())
    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"Respuesta del Arduino: {respuesta}")
    except Exception as e:
        print(f"Error al leer la respuesta: {e}")

# Funciones principales
def empezar():
    enviar_comando("INICIAR\n")
    leer_respuesta()

def actualizar():
    enviar_comando("ACTUALIZAR\n")
    leer_respuesta()

def modo_activo():
    enviar_comando("MODO_ACTIVO\n")
    leer_respuesta()

def modo_sueno():
    enviar_comando("MODO_SUENO\n")
    leer_respuesta()

def cambiar_modo(modo):
    enviar_comando(f"CAMBIAR_MODO {modo}\n")
    leer_respuesta()

def bailar_twist(xMove, yMove, zMove, xRotate, yRotate, zRotate):
    enviar_comando(f"TWIST_BODY {xMove} {yMove} {zMove} {xRotate} {yRotate} {zRotate}\n")
    leer_respuesta()

def mover_cuerpo(x, y, z):
    enviar_comando(f"MOVER_CUERPO {x} {y} {z}\n")
    leer_respuesta()
```

```
def rotar_cuerpo(x, y, z):
    enviar_comando(f"ROTAR_CUERPO {x} {y} {z}\n")
    leer_respuesta()

def mover_adelante():
    enviar_comando("MOVER_ADELANTE 0 21 0\n")
    leer_respuesta()

def mover_atras():
    enviar_comando("MOVER_ATRAS 0 -42 0\n")
    leer_respuesta()

def mover_izquierda():
    enviar_comando("MOVER_IZQUIERDA 21 0 0\n")
    leer_respuesta()

def mover_derecha():
    enviar_comando("MOVER_DERECHA -42 0 0\n")
    leer_respuesta()

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

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

def cambiar_altura(altura):
    enviar_comando(f"CAMBIAR_ALTURA {altura}\n")
    leer_respuesta()

def mover_pierna(pata, x, y, z):
    enviar_comando(f"MOVER_PIERNA {pata} {x} {y} {z}\n")
    leer_respuesta()

def establecer_velocidad(velocidad):
    enviar_comando(f"ESTABLECER_VELOCIDAD {velocidad}\n")
    leer_respuesta()

def establecer_grupo(grupo):
    enviar_comando(f"ESTABLECER_GRUPO {grupo}\n")
    leer_respuesta()
```

```
# Cierra la conexión
arduino.close()
```

Código de Arduino

```
#include <FNHR.h>
#include <EEPROM.h>

FNHR robot;
#define EEPROM_ADDRESS 0

void setup() {
  Serial.begin(9600);
  robot.Start();

  String storedCommand = leerComandoEEPROM();
  if (storedCommand.length() > 0) {
    ejecutarComando(storedCommand);
  } else {
    Serial.println("No hay comandos almacenados.");
  }
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    guardarComandoEEPROM(command);
    ejecutarComando(command);
  }
}

void guardarComandoEEPROM(String command) {
  for (int i = 0; i < command.length(); i++) {
    EEPROM.write(EEPROM_ADDRESS + i, command[i]);
  }
  EEPROM.write(EEPROM_ADDRESS + command.length(), '\0');
}

String leerComandoEEPROM() {
  String command = "";
  char ch;
```

```
  int i = 0;
  while ((ch = EEPROM.read(EEPROM_ADDRESS + i)) != '\0' && i < 100) {
    command += ch;
    i++;
  }
  return command;
}

void ejecutarComando(String command) {
  if (command == "INICIAR") {
    robot.Start();
  } else if (command == "ACTUALIZAR") {
    robot.Update();
  } else if (command == "MODO_ACTIVO") {
    robot.ActiveMode();
  } else if (command == "MODO_SUENO") {
    robot.SleepMode();
  } else if (command.startsWith("CAMBIAR_MODO")) {
    int modo;
    sscanf(command.c_str(), "CAMBIAR_MODO %d", &modo);
    robot.ChangeMode(modo);
  } else if (command.startsWith("TWIST_BODY")) {
    float xMove, yMove, zMove, xRotate, yRotate, zRotate;
    sscanf(command.c_str(), "TWIST_BODY %f %f %f %f %f %f", &xMove, &yMove, &zMove,
&xRotate, &yRotate, &zRotate);
    robot.TwistBody(xMove, yMove, zMove, xRotate, yRotate, zRotate);
  } else if (command.startsWith("MOVER_CUERPO")) {
    float x, y, z;
    sscanf(command.c_str(), "MOVER_CUERPO %f %f %f", &x, &y, &z);
    robot.MoveBody(x, y, z);
  } else if (command.startsWith("ROTAR_CUERPO")) {
    float x, y, z;
    sscanf(command.c_str(), "ROTAR_CUERPO %f %f %f", &x, &y, &z);
    robot.RotateBody(x, y, z);
  } else if (command == "MOVER_ADELANTE 0 21 0") {
    robot.CrawlForward();
  } else if (command == "MOVER_ATRAS 0 -42 0") {
    robot.CrawlBackward();
  } else if (command == "MOVER_IZQUIERDA 21 0 0") {
    robot.MoveLeft();
  } else if (command == "MOVER_DERECHA -42 0 0") {
    robot.MoveRight();
  } else if (command == "GIRAR_IZQUIERDA") {
    robot.TurnLeft();
```

```
  } else if (command == "GIRAR_DERECHA") {
    robot.TurnRight();
  } else if (command.startsWith("CAMBIAR_ALTURA")) {
    float altura;
    sscanf(command.c_str(), "CAMBIAR_ALTURA %f", &altura);
    robot.ChangeBodyHeight(altura);
  } else if (command.startsWith("MOVER_PIERNA")) {
    int pata, x, y, z;
    sscanf(command.c_str(), "MOVER_PIERNA %d %d %d %d", &pata, &x, &y, &z);
    robot.LegMoveToRelatively(pata, x, y, z);
  } else if (command.startsWith("ESTABLECER_VELOCIDAD")) {
    float velocidad;
    sscanf(command.c_str(), "ESTABLECER_VELOCIDAD %f", &velocidad);
    robot.SetActionSpeed(velocidad);
  } else if (command.startsWith("ESTABLECER_GRUPO")) {
    int grupo;
    sscanf(command.c_str(), "ESTABLECER_GRUPO %d", &grupo);
    robot.SetGroup(grupo);
  } else {
    Serial.println("Comando no reconocido.");
  }
}
```

Pendientes por Hacer

1. **Corrección del Almacenamiento en EEPROM:**
    - Verificar si la dirección de la memoria EEPROM está actualizando correctamente los comandos.
    - Probar el uso de EEPROM.update() para mejorar la eficiencia del almacenamiento.
2. **Validación de Comandos Almacenados:**
    - Mejorar la lectura de la EEPROM para validar la integridad de los comandos antes de ejecutarlos.
3. **Implementar Verificación de Datos:**
    - Incluir un mecanismo que valide si el comando leído desde la EEPROM es correcto antes de ejecutarlo.

Problema Detectado

Aunque los comandos desde Python eran enviados correctamente y ejecutados por el robot, **los comandos no se guardaban permanentemente en la memoria del robot FNK0031**. Al reiniciar el robot, los comandos almacenados no eran reconocidos o se perdían.

Causa del Problema

El problema radica en que la función que guarda los comandos en la **EEPROM** no estaba funcionando correctamente. Es probable que:

- La dirección de memoria utilizada para almacenar el comando en la EEPROM no esté siendo actualizada correctamente.

Propuestas para Resolver el Problema

Revisión del Almacenamiento en EEPROM:

- Validar que `saveCommandToEEPROM()` guarde el comando correctamente en la dirección correspondiente.
- Utilizar `EEPROM.update()` en lugar de `EEPROM.write()` para reducir el desgaste de la memoria.

Validación de la Lectura:

- Mejorar `readCommandFromEEPROM()` para comprobar si los datos leídos son válidos antes de ejecutarlos.

Depuración y Verificación:

- Imprimir los valores almacenados y leídos desde la EEPROM para comprobar la integridad de los datos.

Nuevos códigos con las propuestas adaptadas

Código de Arduino Actualizado

```
#include <FNHR.h>
#include <EEPROM.h>

FNHR robot;
#define EEPROM_ADDRESS 0  // Dirección inicial de la EEPROM
#define MAX_COMMAND_LENGTH 50  // Longitud máxima del comando

void setup() {
  Serial.begin(9600);
  robot.Start();

  String storedCommand = leerComandoEEPROM();
  if (storedCommand.length() > 0) {
```

```
    Serial.print("Comando almacenado: ");
    Serial.println(storedCommand);
    ejecutarComando(storedCommand);  // Ejecuta el comando almacenado
  } else {
    Serial.println("No hay comandos almacenados.");
  }
}

void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();  // Elimina espacios en blanco
    if (command.length() > 0) {
      guardarComandoEEPROM(command);  // Guarda el nuevo comando
      ejecutarComando(command);       // Ejecuta el comando recibido
    }
  }
}

// Guarda el comando en la EEPROM
void guardarComandoEEPROM(String command) {
  int length = min(command.length(), MAX_COMMAND_LENGTH - 1);  // Limita la longitud
  for (int i = 0; i < length; i++) {
    EEPROM.update(EEPROM_ADDRESS + i, command[i]);
  }
  EEPROM.update(EEPROM_ADDRESS + length, '\0');  // Agrega terminador de cadena
  Serial.println("Comando guardado correctamente en EEPROM.");
}

// Lee el comando almacenado en la EEPROM
String leerComandoEEPROM() {
  String command = "";
  char ch;
  int i = 0;
  while ((ch = EEPROM.read(EEPROM_ADDRESS + i)) != '\0' && i <
MAX_COMMAND_LENGTH - 1) {
    command += ch;
    i++;
  }
  return command;
}

// Ejecuta el comando recibido o almacenado
void ejecutarComando(String command) {
```

```
  if (command == "INICIAR") {
    robot.Start();
  } else if (command == "ACTUALIZAR") {
    robot.Update();
  } else if (command == "MODO_ACTIVO") {
    robot.ActiveMode();
  } else if (command == "MODO_SUENO") {
    robot.SleepMode();
  } else if (command.startsWith("CAMBIAR_MODO")) {
    int modo;
    sscanf(command.c_str(), "CAMBIAR_MODO %d", &modo);
    robot.ChangeMode(modo);
  } else if (command.startsWith("BAILAR_TWIST")) {
    float xMove, yMove, zMove, xRotate, yRotate, zRotate;
    sscanf(command.c_str(), "BAILAR_TWIST %f %f %f %f %f %f", &xMove, &yMove, &zMove,
&xRotate, &yRotate, &zRotate);
    robot.TwistBody(xMove, yMove, zMove, xRotate, yRotate, zRotate);
  } else if (command.startsWith("MOVER_CUERPO")) {
    float x, y, z;
    sscanf(command.c_str(), "MOVER_CUERPO %f %f %f", &x, &y, &z);
    robot.MoveBody(x, y, z);
  } else if (command.startsWith("ROTAR_CUERPO")) {
    float x, y, z;
    sscanf(command.c_str(), "ROTAR_CUERPO %f %f %f", &x, &y, &z);
    robot.RotateBody(x, y, z);
  } else if (command == "MOVER_ADELANTE 0 21 0") {
    robot.CrawlForward();
  } else if (command == "MOVER_ATRAS 0 -42 0") {
    robot.CrawlBackward();
  } else if (command == "MOVER_IZQUIERDA 21 0 0") {
    robot.MoveLeft();
  } else if (command == "MOVER_DERECHA -42 0 0") {
    robot.MoveRight();
  } else if (command == "GIRAR_IZQUIERDA") {
    robot.TurnLeft();
  } else if (command == "GIRAR_DERECHA") {
    robot.TurnRight();
  } else if (command.startsWith("CAMBIAR_ALTURA")) {
    float altura;
    sscanf(command.c_str(), "CAMBIAR_ALTURA %f", &altura);
    robot.ChangeBodyHeight(altura);
  } else if (command.startsWith("MOVER_PIERNA")) {
    int pata, x, y, z;
    sscanf(command.c_str(), "MOVER_PIERNA %d %d %d %d", &pata, &x, &y, &z);
```

```
    robot.LegMoveToRelatively(pata, x, y, z);
  } else if (command.startsWith("ESTABLECER_VELOCIDAD")) {
    float velocidad;
    sscanf(command.c_str(), "ESTABLECER_VELOCIDAD %f", &velocidad);
    robot.SetActionSpeed(velocidad);
  } else if (command.startsWith("ESTABLECER_GRUPO")) {
    int grupo;
    sscanf(command.c_str(), "ESTABLECER_GRUPO %d", &grupo);
    robot.SetGroup(grupo);
  } else {
    Serial.println("Comando no reconocido.");
  }
}
```

Código de Python Actualizado

python

Copiar

Editar

```
import serial
import time

# Configuración del puerto serial
puerto = 'COM3'
velocidad_baudios = 9600

try:
    print(f"Conectando al Arduino en el puerto {puerto}...")
    arduino = serial.Serial(puerto, velocidad_baudios, timeout=1)
    time.sleep(2)
    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."""
    try:
        print(f"Enviando comando: {comando.strip()}")
        arduino.write(comando.encode())
        leer_respuesta()
    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()
            if respuesta:
                print(f"Respuesta del Arduino: {respuesta}")
    except Exception as e:
        print(f"Error al leer la respuesta: {e}")

# Funciones principales
def empezar():
    enviar_comando("INICIAR\n")

def actualizar():
    enviar_comando("ACTUALIZAR\n")

def modo_activo():
    enviar_comando("MODO_ACTIVO\n")

def modo_sueno():
    enviar_comando("MODO_SUENO\n")

def cambiar_modo(modo):
    enviar_comando(f"CAMBIAR_MODO {modo}\n")

def bailar_twist(xMove, yMove, zMove, xRotate, yRotate, zRotate):
    enviar_comando(f"TWIST_BODY {xMove} {yMove} {zMove} {xRotate} {yRotate} {zRotate}\n")

def mover_cuerpo(x, y, z):
    enviar_comando(f"MOVER_CUERPO {x} {y} {z}\n")

def rotar_cuerpo(x, y, z):
    enviar_comando(f"ROTAR_CUERPO {x} {y} {z}\n")

def mover_adelante():
    enviar_comando("MOVER_ADELANTE 0 21 0\n")

def mover_atras():
    enviar_comando("MOVER_ATRAS 0 -42 0\n")

def mover_izquierda():
    enviar_comando("MOVER_IZQUIERDA 21 0 0\n")

def mover_derecha():
    enviar_comando("MOVER_DERECHA -42 0 0\n")
```

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

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

def cambiar_altura(altura):
    enviar_comando(f"CAMBIAR_ALTURA {altura}\n")

def mover_pierna(pata, x, y, z):
    enviar_comando(f"MOVER_PIERNA {pata} {x} {y} {z}\n")

def establecer_velocidad(velocidad):
    enviar_comando(f"ESTABLECER_VELOCIDAD {velocidad}\n")

def establecer_grupo(grupo):
    enviar_comando(f"ESTABLECER_GRUPO {grupo}\n")

# Cierra la conexión
arduino.close()
```

Conclusión

El proyecto logró una conexión funcional entre Arduino y Python para controlar el robot FNK0031, pero aún es necesario resolver el problema de almacenamiento en la EEPROM para asegurar que los comandos persistan después del reinicio del sistema. Se recomienda realizar pruebas adicionales para validar el correcto funcionamiento del almacenamiento y lectura de comandos desde la EEPROM.