UNIVERSIDAD DEL BÍO-BÍO

21 Comunicación serial Arduino

Tutorial arduino

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Introducción: Creación de una Interfaz Gráfica para Controlar Arduino y Visualizar una Cámara en Tiempo Real con Tkinter y OpenCV

En este tutorial, exploraremos cómo crear una interfaz gráfica de usuario (GUI) en Python utilizando la biblioteca Tkinter. La interfaz permitirá la comunicación con un dispositivo Arduino a través de un puerto serial y mostrará en tiempo real la transmisión de una cámara utilizando OpenCV.

Esta aplicación ofrece una solución práctica para controlar dispositivos Arduino y monitorear información visual mediante una interfaz intuitiva y fácil de usar. Desde el encendido de luces LED hasta la visualización de datos de sensores, esta GUI proporcionará un entorno interactivo para la interacción con dispositivos físicos.

Durante el tutorial, abordaremos conceptos clave de Tkinter para el diseño de la interfaz y utilizaremos OpenCV para la captura y procesamiento de imágenes de la cámara. Además, exploraremos la integración de funcionalidades como el control de luces LED, el envío de comandos al Arduino y la programación de solicitudes automáticas mediante un temporizador.

Sin más preámbulos, comencemos con la creación de esta aplicación versátil y práctica que combina lo mejor de la programación de interfaz gráfica y la interacción con hardware físico.

Este tutorial consta de 3 partes, primero empezaremos por explicar las partes del arduinos que serán utilizadas durante la ejecución, luego el funcionamiento del código en Python y finalmente el código realizado en arduino ide

Parte 1: En esta parte explicaremos el funcionamiento del código en Python encargado de realizar la comunicación serial con el arduino a través de un puerto especifico

Librerías Importadas

```python
import tkinter as tk
from tkinter import ttk, messagebox
import serial
import cv2
import imutils
from PIL import Image, ImageTk
```

Se importan las bibliotecas necesarias para la creación de la interfaz gráfica (Tkinter), manejo de puertos seriales (serial), procesamiento de imágenes (cv2, imutils), y manipulación de imágenes (PIL).

Funciones Relacionadas con la Cámara

```python
def camara():
    global capture
    capture = cv2.VideoCapture(0)
    iniciar()

def iniciar():
    global capture
    if capture is not None:
        ret, frame = capture.read()
        if ret == True:
            frame = imutils.resize(frame, width=440)
            frame = imutils.resize(frame, height=360)
            ImagenCamara = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            im = Image.fromarray(ImagenCamara)
            img = ImageTk.PhotoImage(image=im)
            LImagen.configure(image=img)
            LImagen.image = img
            LImagen.after(1, iniciar)
        else:
            LImagen.image = ""
            capture.release()
```

Nos permitirán iniciar la cámara con el único objetivo de ver y interactuar el Arduino de forma remota

Funciones Relacionadas con Arduino y la Interfaz

```
# Funciones
def click_rojo():
    enviar_datos("rojo")

def click_verde():
    enviar_datos("azul")

def click_amarillo():
    enviar_datos("amarillo")

def click_temperatura():
    limpiar_datos()
    programar_envio("temperatura")

def click_humedad():
    limpiar_datos()
    programar_envio("humedad")

def click_conectar():
    if not SerialPort1.isOpen():
        SerialPort1.baudrate = 9600
        SerialPort1.bytesize = 8
        SerialPort1.parity = "N"
        SerialPort1.stopbits = serial.STOPBITS_ONE
        SerialPort1.port = comboBox1.get()
        SerialPort1.open()
        actualizar_estado("Conectado", "LIME")
        # Iniciar la función para mostrar la cámara

def click_desconectar():
    detener_camara()
    if SerialPort1.isOpen():
        SerialPort1.close()
        actualizar_estado("Desconectado", "red")

def click_enviar():
    msj = TextEnviar.get(1.0, tk.END).strip()
    if msj.lower() == "temperatura":
        # Enviar la solicitud de temperatura inmediatamente
        enviar_datos("temperatura")
    else:
        programar_envio(msj)
```

```python
def enviar_rgb():
    r = str(int(r_spinbox.get()))
    g = str(int(g_spinbox.get()))
    b = str(int(b_spinbox.get()))
    mensaje = f"{r},{g},{b}"
    SerialPort1.write(mensaje.encode() + b"\r")


def detener_camara():
    global carrera_camara
    if carrera_camara is not None:
        ventana.after_cancel(carrera_camara)
        carrera_camara = None


def enviar_datos(mensaje):
    SerialPort1.write(mensaje.encode() + b"\r")
    datos = SerialPort1.read_all().decode()
    TextRecibidos.insert(1.0, datos)

def limpiar_datos():
    TextRecibidos.delete(1.0, tk.END)

def actualizar_estado(mensaje, color):
    TextoEstado["state"] = "normal"
    TextoEstado.delete(1.0, tk.END)
    TextoEstado.insert(1.0, mensaje)
    TextoEstado.configure(background=color)
    TextoEstado["state"] = "disabled"
    messagebox.showinfo(message=f"Puerto {mensaje}")

def actualizar_tiempo_espera(event):
    global tiempo_espera
    tiempo_espera = int(ComboBoxT.get())

def programar_envio(mensaje):
    ventana.after(tiempo_espera * 1000, lambda: enviar_datos(mensaje))
```

Estas funciones se asocian a eventos de la interfaz y realizan acciones como enviar comandos a Arduino, conectar o desconectar el puerto serial, y enviar datos ingresados desde la interfaz, a continuación se detallan un poco más:

1) **click_rojo():**
   - Envia el comando "rojo" a través del puerto serial a Arduino, indicando que se encienda un LED rojo.

2) **click_verde():**
   - Envia el comando "azul" a través del puerto serial a Arduino, indicando que se encienda un LED azul.

3) **click_amarillo():**
   - Envia el comando "amarillo" a través del puerto serial a Arduino, indicando que se realice una acción específica relacionada con un LED amarillo.

4) **click_temperatura():**
   - Limpia el área de visualización de datos (TextRecibidos).
   - Programa el envío automático del comando "temperatura" a Arduino después de un tiempo determinado.

5) **click_humedad():**
   - Limpia el área de visualización de datos (TextRecibidos).
   - Programa el envío automático del comando "humedad" a Arduino después de un tiempo determinado.

6) **click_conectar():**
   - Si el puerto serial no está abierto, configura los parámetros del puerto serial y lo abre para establecer la conexión con Arduino.
   - Actualiza el estado de la conexión en la interfaz.

7) **click_desconectar():**
   - Detiene la captura de la cámara.
   - Si el puerto serial está abierto, lo cierra para desconectar la comunicación con Arduino.
   - Actualiza el estado de la conexión en la interfaz.

8) **click_enviar():**
   - Obtiene el mensaje ingresado en el área de texto (TextEnviar).
   - Si el mensaje es "temperatura", envía inmediatamente el comando "temperatura" a Arduino. De lo contrario, programa el envío del mensaje después de un tiempo determinado

9) **enviar_rgb():**
   - Obtiene los valores de rojo (r), verde (g) y azul (b) desde los spinboxes.
   - Construye un mensaje con el formato "r,g,b" y lo envía a Arduino a través del puerto serial.

10) **detener_camara():**
   - Detiene la función iniciar() que muestra las imágenes de la cámara en tiempo real.

11) **enviar_datos(mensaje):**
   - Envía un mensaje a Arduino a través del puerto serial y lee la respuesta, luego muestra los datos recibidos en el área de visualización (TextRecibidos).

12) **Limpiar_datos():**
- Borra todo el contenido en el área de visualización de datos (TextRecibidos).

13) **actualizar_estado(mensaje, color):**
- Actualiza el estado de la conexión en la interfaz con un mensaje y un color específicos.
- Muestra un cuadro de información emergente con el estado del puerto.

14) **actualizar_tiempo_espera(event):**
- Actualiza el valor de tiempo_espera según la opción seleccionada en el combobox (ComboBoxT).

15) **programar_envio(mensaje):**
- Programa el envío de un mensaje a Arduino después de un tiempo determinado, utilizando el valor de tiempo_espera.

Componentes de la interfaz gráfica inicialización del objeto Serial para comunicación con arduino

```
# Componentes de la interfaz gráfica
BotonConectar = tk.Button(ventana, text="Conectar", command=click_conectar)
BotonDesconectar = tk.Button(ventana, text="Desconectar", command=click_desconectar)
BotonRojo = tk.Button(ventana, text="Rojo", command=click_rojo)
BotonVerde = tk.Button(ventana, text="Azul", command=click_verde)
BotonAmarillo = tk.Button(ventana, text="Amarillo", command=click_amarillo)
BotonTemperatura = tk.Button(ventana, text="Temperatura", command=click_temperatura)
BotonHumedad = tk.Button(ventana, text="Humedad", command=click_humedad)

BotonConectar.place(x=70, y=40, width=75, height=23)
BotonDesconectar.place(x=310, y=40, width=75, height=23)
BotonRojo.place(x=40, y=100, width=75, height=23)
BotonVerde.place(x=40, y=130, width=75, height=23)
BotonAmarillo.place(x=40, y=160, width=75, height=23)
BotonTemperatura.place(x=40, y=190, width=75, height=23)
BotonHumedad.place(x=40, y=220, width=75, height=23)

# Marco para agrupar botones y etiqueta
MarcoBotones = tk.Frame(ventana, bd=2, relief=tk.GROOVE)
MarcoBotones.place(x=120, y=330, width=180, height=110)

SerialPort1 = serial.Serial()
ventana.mainloop()
```

Se crean los componentes de la interfaz, como botones, etiquetas, cuadros de texto, etc. Además, se inicializa un objeto Serial (SerialPort1) para la comunicación con Arduino.

Parte 2: Explicación del código creado en el Arduino IDE que recibe comunicación serial con el código programado en python.

Este código es un programa para un Arduino que utiliza un sensor DHT11 para medir la temperatura y la humedad. Además, controla varios LEDs según los comandos recibidos a través del puerto serial. Aquí está la explicación de las partes principales del código:

1. Configuración de Pines y Sensores:

- Se definen los pines a los que están conectados los LEDs (led, led2, led3, led4, led5) y el pin al que está conectado el sensor de temperatura DHT11 (dhtPin).

```
#include <DHT.h>

const int led = 12;
const int led2 = 13;
const int led3 = 9;
const int led4 = 10;
const int led5 = 11;

const int dhtPin = 4;  // Pin al que está conectado el sensor de temperatura
```

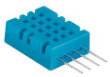
2. Inclusión de Librerías

Se incluye la librería DHT que proporciona funciones para trabajar con el sensor de temperatura/humedad DHT11.

```
#include <DHT.h>
```

3. Creación de un Objeto DHT

```
DHT dht(dhtPin, DHT11);
```



Se crea un objeto de la clase DHT para interactuar con el sensor de temperatura y humedad.

4. Configuración Inicial

```
void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);

  dht.begin();
}
```

- Se inicia la comunicación serial con una velocidad de 9600 baudios.
- Se configuran los pines de los LEDs como salidas.
- Se inicializa el sensor DHT11.

5. Bucle Principal (loop)

```
void loop() {
  if (Serial.available() > 0) {
    String option = Serial.readStringUntil('\r');
    String rgb = Serial.readStringUntil('\r');
```

Este fragmento de código está ubicado dentro del bucle principal loop() y se encarga de manejar la comunicación serial entrante. Aquí hay una explicación detallada:

if (Serial.available() > 0): Esta condición verifica si hay datos disponibles para leer en el puerto serial. Serial.available() devuelve el número de bytes disponibles para la lectura.

String option = Serial.readStringUntil('\r');: Lee una cadena de caracteres desde el puerto serial hasta que encuentra el carácter de retorno de carro ('\r'). Este carácter generalmente indica el final de un comando o mensaje.
String rgb = Serial.readStringUntil('\r');: Realiza lo mismo que la línea anterior, pero esta vez lee una cadena que se espera que represente los valores RGB para el control de color de los LEDs.

La lógica aquí es que el programa espera recibir comandos y datos a través de la comunicación serial. Por ejemplo, si desde tu programa en Python envías la cadena "rojo\r",

el Arduino interpretará esto como la opción "rojo" y ejecutará el código correspondiente para encender y apagar el LED asociado.

Para comunicarte con este código desde Python, necesitarás un programa que se ejecute en tu computadora y que utilice el puerto serial para enviar comandos y recibir datos desde Arduino, el código que esta en la parte 1 realiza esta acción utilizando la biblioteca pyserial.

6. Control de LEDs según la Opción Recibida

```
if (option == "rojo") {
  digitalWrite(led, HIGH);
  delay(200);
  digitalWrite(led, LOW);
}

if (option == "azul") {
  digitalWrite(led2, HIGH);
  delay(200);
  digitalWrite(led2, LOW);
}

if (option == "amarillo") {
  // Establecer un color amarillo personalizado
  analogWrite(led3, 255);    // Rojo al máximo brillo
  analogWrite(led4, 233);   // Azul con una intensidad personalizada
  analogWrite(led5, 0);
  delay(200);

  // Apagar el LED después de un tiempo
  analogWrite(led3, 0);
  analogWrite(led4, 0);
}
```

Se controlan los LEDs según la opción recibida a través de la comunicación serial.

7. Lectura de Temperatura y Humedad

```
if (option == "temperatura") {
  // Leer la temperatura del sensor
  float temperatura = dht.readTemperature();

  // Mostrar la temperatura en el Monitor Serie
  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" °C");
}

if (option == "humedad") {
  // Leer la humedad del sensor
  float humedad = dht.readHumidity();

  // Mostrar la humedad en el Monitor Serie
  Serial.print("Humedad: ");
  Serial.print(humedad);
  Serial.println(" %");
}
```

Se leen los datos de temperatura o humedad del sensor en el pin 4 y se envían a través de la comunicación serial.

8. Control de Color RGB Personalizado

```
int r, g, b;
if (sscanf(rgb.c_str(), "%d,%d,%d", &r, &g, &b) == 3) {
  // Ajustar los pines LED con los valores RGB
  analogWrite(led3, r);
  analogWrite(led4, g);
  analogWrite(led5, b);
```

- Se lee y descompone la cadena RGB recibida a través de la comunicación serial.
- Se ajustan los pines LED con los valores RGB.

9. Espera y Apagado de LEDs

```
delay(200);
analogWrite(led3, 0);
analogWrite(led4, 0);
analogWrite(led5, 0);
```

- Se añaden pausas y se apagan los LEDs RGB después de un tiempo.

Este código permite controlar LEDs y obtener datos de temperatura/humedad a través de comandos enviados por la comunicación serial desde un dispositivo externo (como un ordenador). El sensor DHT11 proporciona información de temperatura y humedad, y los LEDs responden a comandos para cambiar de color o parpadear.

Instalacion y ejecucion del programa

Paso 1: Instalar el Arduino IDE

Asegúrate de tener instalado el Arduino IDE en tu computadora. Puedes descargar la última versión desde el sitio web oficial de Arduino: Arduino Software.

Paso 2: Conectar la Placa Arduino

Conecta tu placa Arduino a tu computadora utilizando un cable USB.

Paso 3: Abrir el Arduino IDE

Abre el Arduino IDE en tu computadora.

Paso 4: Crear un Nuevo Proyecto

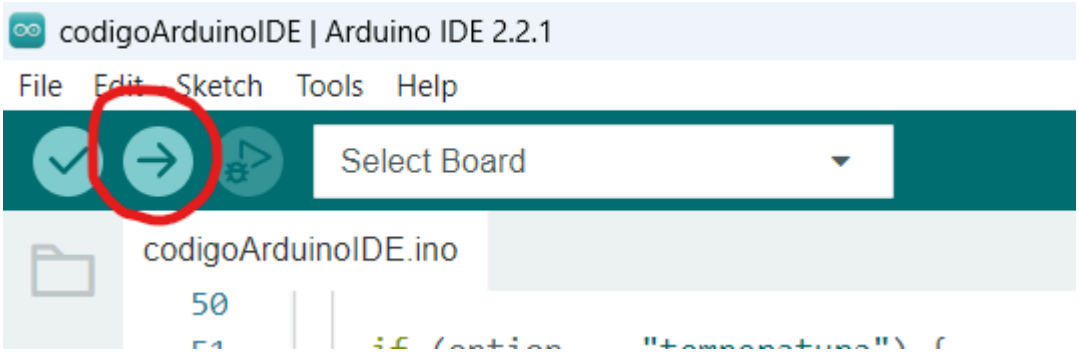
- Haz clic en Archivo > Nuevo para abrir un nuevo sketch.
- Copia y pega el código proporcionado en la ventana del Arduino IDE.

Paso 5: Seleccionar el Tipo de Placa

Ve a Herramientas > Placa y selecciona tu modelo de Arduino. Por ejemplo, "Arduino Uno", asegúrate de que el Puerto también esté seleccionado. Debería ser algo como "COMx" en Windows o "/dev/ttyUSB0" en Linux.

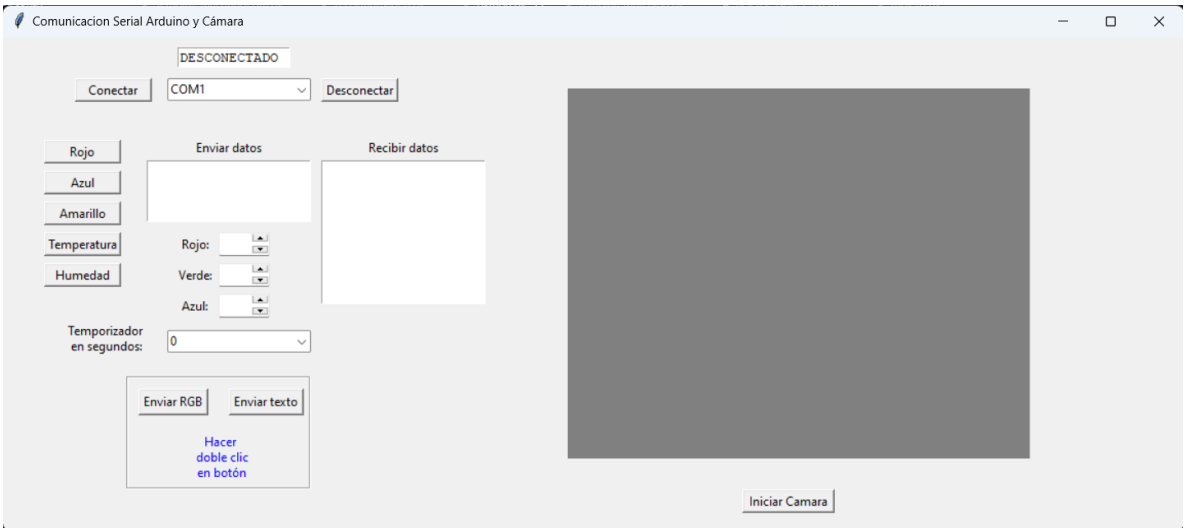
Paso 6: Cargar el Programa en la Placa

Haz clic en el icono de flecha hacia la derecha (➔) en la esquina superior izquierda del Arduino IDE para cargar el programa en la placa.
Observa la barra de estado en la parte inferior del IDE. Si todo está bien, deberías ver "Compilación terminada" y "Carga terminada".



Para establecer una comunicación serial entre el código de Python de la Parte 1 y el Arduino, es esencial garantizar que ambos estén conectados a la misma placa, como se especifica en el Paso 5 del Arduino IDE. Una vez que hayas realizado esta configuración, la conexión estará establecida y podrás enviar comandos al Arduino mediante el puerto serial.

Interfaz gráfica para interactuar con arduino



Hacer doble clic en los botones para que envíen la señal.