UART

iita salta

April 2025

1 Manejo de la libreria UART en micro python

UART() es una función del módulo pyb (PyBoard) de MicroPython que configura un puerto serial UART en la OpenMV. Permite comunicar la cámara con otros dispositivos como Arduinos, Teensy, ESP32, etc., a través de UART. Tenemos a la sentencia

uart = UART(3, 19200, bits=8, parity=None, stop=1, timeout char=100)

En donde los parametros

bits Número de bits de datos (usualmente 8)

parity Paridad (None, 'even', 'odd') El parámetro parity en la configuración de un puerto UART se refiere a un mecanismo de detección de errores en la transmisión de datos. ¿Cómo funciona la paridad? Supón que vas a enviar el byte 10110010:

Paridad par ('even'): El número total de 1s (incluyendo el bit de paridad) debe ser par. → Como hay 4 unos, el bit de paridad es 0.

Paridad impar ('odd'): El número total de 1s debe ser impar. → Hay 4 unos, así que se agrega un 1 como bit de paridad. Usa None (sin paridad) si ambos dispositivos no lo necesitan (lo más común en Arduino, OpenMV, etc.). Usa paridad solo si estás conectado a un sistema que lo requiere (como algunos equipos industriales, PLCs, o sensores antiguos).

stop Bits de parada (normalmente 1)

timeout char Timeout para recepción de caracteres (en ms)

2 Ejemplo de codigo

El código usa la cámara OpenMV para detectar objetos de color amarillo, y luego envía un '1' si lo detecta, o '0' si no, a través de UART.

- **Importa los módulos necesarios**:

- 'sensor': control de la cámara.

- 'image': para trabajar con imágenes (detección de colores, blobs, etc.).

- 'time': para controlar temporizadores.

- 'UART': para comunicación serial con otro microcontrolador (como Arduino o Teensy).

Configuración de la cámara

sensor.reset()

Reinicia el sensor de la cámara para empezar con una configuración limpia.

- Configura el **formato de color** a **RGB565** (16 bits por píxel).

- Este formato es ideal para trabajar con color (mejor que 'GRAYSCALE').

sensor.set framesize(sensor.QVGA)(agrega guion bajo)

Define el tamaño de imagen: QVGA = 320x240 píxeles y Tamaño decente para análisis sin ser tan pesado.

- Espera **2 segundos** para que la cámara se estabilice antes de comenzar a capturar imágenes.

- Es útil para que se ajusten automáticamente exposición/luz si se permite.

sensor.set auto gain(False)

sensor.set auto whitebal(False)

clock = time.clock() Crea un objeto 'clock' para medir **cuántos cuadros por segundo (FPS)** se están procesando.

yellow threshold = (30, 100, -10, 40, 40, 100) Este es el umbral de color en formato LAB Se usa para identificar píxeles que corresponden a "amarillo".

while True: Comienza con el ciclo principal. clock.tick() utilizamos al objeto clock. Luego con img = sensor.snapshot() tomamos la imagen y la guardamos en la variable img. La variable blobs = img.find blobs([yellow threshold], pixels threshold=100, area threshold=100) Busca "blobs" (regiones) en la imagen que coincidan con el color amarillo (usando el umbral definido), pixels threshold=100: ignora regiones muy pequeñas (menos de 100 píxeles). area threshold=100: ignora blobs con área total pequeña (evita ruidos). Ahora con la condicion if blobs: Si encontró al menos una región amarilla: si es verdadero ingresa y uart.write(b'1') Envía el byte '1' a través de UART y print("Amarillo detectado") Muestra en la consola de OpenMV que se detectó amarillo.

for b in blobs:

– img.draw rectangle(b.rect())

– img.draw cross(b.cx(), b.cy()) Este trozo de codigo me indica:

- Para cada blob encontrado: — Dibuja un **rectángulo** alrededor de él.

— Dibuja una **cruz en el centro** del blob.

— Esto es solo visual, para que lo veas en la ventana de la cámara.

else:

– uart.write(b'0')

– print("No se detectó amarillo")

Este codigo es en el caso de no detectar amarillo envia un 0 como string al puerto serial.

time.sleep ms(100) Espera **100 milisegundos** antes de procesar el siguiente cuadro

3 Ejemplo del receptor teensy 4.1

En el archivo platformio.ino agregue:

monitor speed = 9600

lib deps =

—— clamepending / ZirconLib
—— Wire
—— SPI
—— Adafruit BusIO
En el archivo main escribi el codigo siguiente:
include ¡Arduino.h¿
include ¡zirconLib.h¿
void setup()
InitializeZircon();
Serial.begin(9600); // Comunicación con la PC
Serial1.begin(19200); // Comunicación con OpenMV (UART)
void loop()
if (Serial1.available())
char recibido = Serial1.read();
if (recibido == '1')
Serial.println("Se detectó color amarillo.");
motor1(100,0);
else if (recibido == '0')
Serial.println("No se detectó color amarillo.");
motor1(0,0);
else
Serial.print("Dato no esperado: ");
Serial.println(recibido);