
**Sistemas Embebidos Distribuidos**

Maestría en sistemas embebidos | 8va Cohorte

# Practica 2

Autor:

**Marcos Dominguez**

mrds0690@gmail.com

31 de mayo del 2024

# Introducción

El objetivo de esta actividad es implementar un sistema que permita relevar el brillo sensado por el LDR en función del tiempo, utilizando un ESP32. Además, el sistema debe permitir controlar en tiempo real el brillo de un LED externo agregado al ESP32 mediante la App IoT MQTT Panel.

## Requerimientos

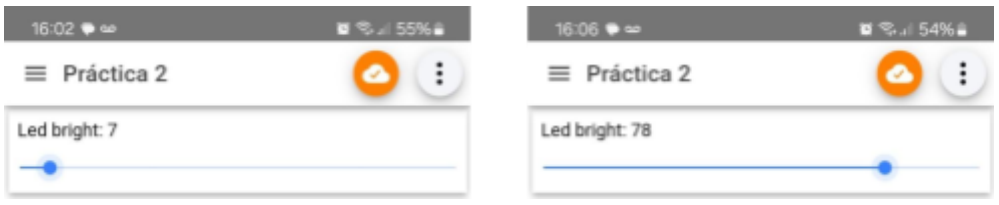
El sistema debe cumplir con los siguientes requisitos:

1. Desde la App, mediante un panel Slider, se debe poder controlar el brillo del LED conectado al nodo.
2. Desde la App, mediante un panel de tipo Button, se debe poder controlar el inicio y finalización del muestreo del LDR.
3. La tasa de muestreo del nodo debe ser de 10 Hz y debe finalizar luego de 20 segundos de medición o cuando el usuario cancele el muestreo desde la App.
4. En la App debe agregarse un panel de tipo LED que indique si la medición está activa o no.
5. En la App debe agregarse un panel Line Graph que muestre en tiempo real el valor de las lecturas del LDR enviadas por el ESP32.
6. En una PC conectada a un broker MQTT local, debe ejecutarse un script que escuche un topic MQTT en el que se reciban las lecturas del LDR y almacene los datos en un archivo CSV.

# Implementación

## Control del Brillo del LED

En la App IoT MQTT Panel, se configuró un panel Slider que permite controlar el brillo del LED conectado al ESP32. La posición del Slider se traduce en un valor PWM que ajusta la intensidad del LED.



## Control del Muestreo del LDR

Debido a que no disponía de un LDR se reemplaza este por un potenciómetro para simular su comportamiento.

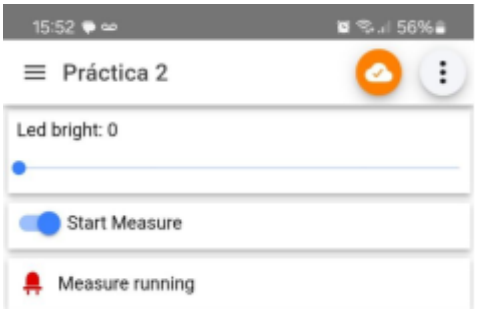
Se configuró un panel de tipo Button en la App para iniciar y finalizar el muestreo del ADC. Al presionar el botón, el ESP32 comienza a muestrear la señal a una tasa de 10 Hz. El muestreo se detiene automáticamente después de 20 segundos o cuando el usuario presiona nuevamente el botón para finalizar.

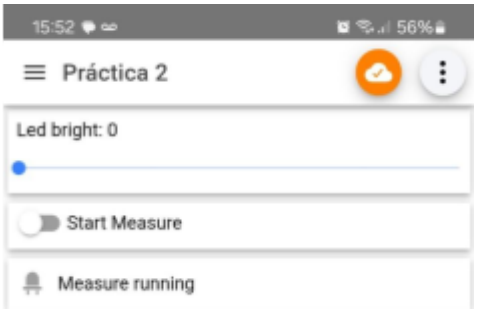
```
static void CRONO_timerCallback(void *arg) {      You, 5 hours ago • feat: ac
    if (n < SAMPLES_SIZE) {
        samples[n] = IO_readAdc();
        IO_monitorStem(samples[n]);
        static char str[20];
        sprintf(str, "%d", samples[n] * 100 / 4096);
        MQTT_publish("marcos_practica2/measurement", str);
        n++;
    }
    else {
        REPORT_MEASUREMENTReportEnable(DISABLE_MEASUREMENT_MEASUREMENT);
    }
}
```

El tiempo del timer está seteado a 100 ms y SAMPLES_SIZE es 200.

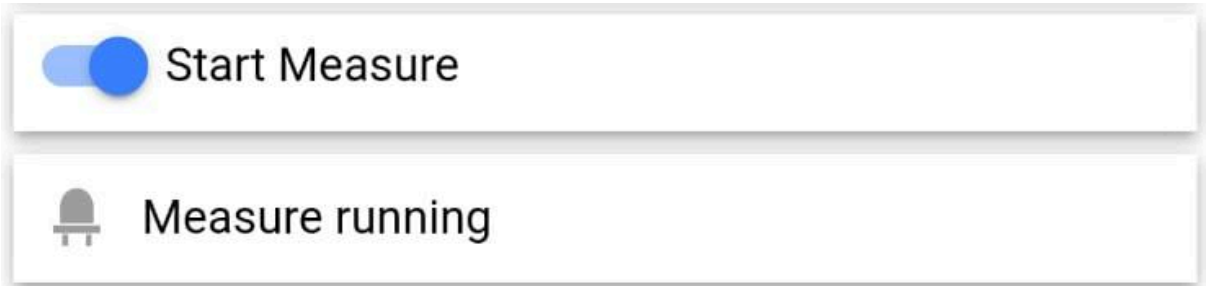
### Indicador de Estado del Muestreo

Se agregó un panel de tipo LED en la App para indicar si la medición está activa o no. Este LED se enciende cuando el muestreo está en curso y se apaga cuando el muestreo se detiene.




En esta imagen se observa cómo se envía a grabar y el sistema reporta que está grabando con el indicador led. También puede estar el led apagado con el Start Measure activado si este se mantuvo encendido más de 20 segundos.

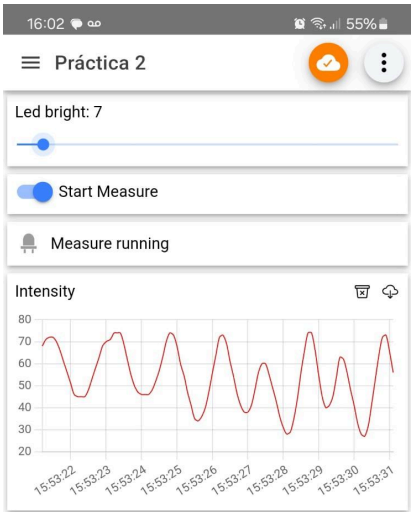



Esto se maneja en la siguiente sección del código:

```
while (1) {
    // IO_toggleLed();
    if (REPORT_MEASUREMENTReportCheck() != 0) {
        if (IO_getLed() == 0) {
            IO_setLed(1);       You, 3 hours ago • Uncommitted ch
            MQTT_publish("marcos_practica2/led", "ON");
            CRONO_timerStart(100);
        }
    }
    else {
        if (IO_getLed() != 0) {
            CRONO_timerStop();
            IO_setLed(0);
            n = 0;
            MQTT_publish("marcos_practica2/led", "OFF");
        }
    }
    CRONO_delayMs(100);
}
```

## Visualización en Tiempo Real

Un panel de tipo Line Graph en la App muestra en tiempo real los valores de las lecturas enviadas por el ESP32. El parámetro Max Persistence se configuró en 100 para obtener una visualización adecuada.



## Almacenamiento de Datos en CSV

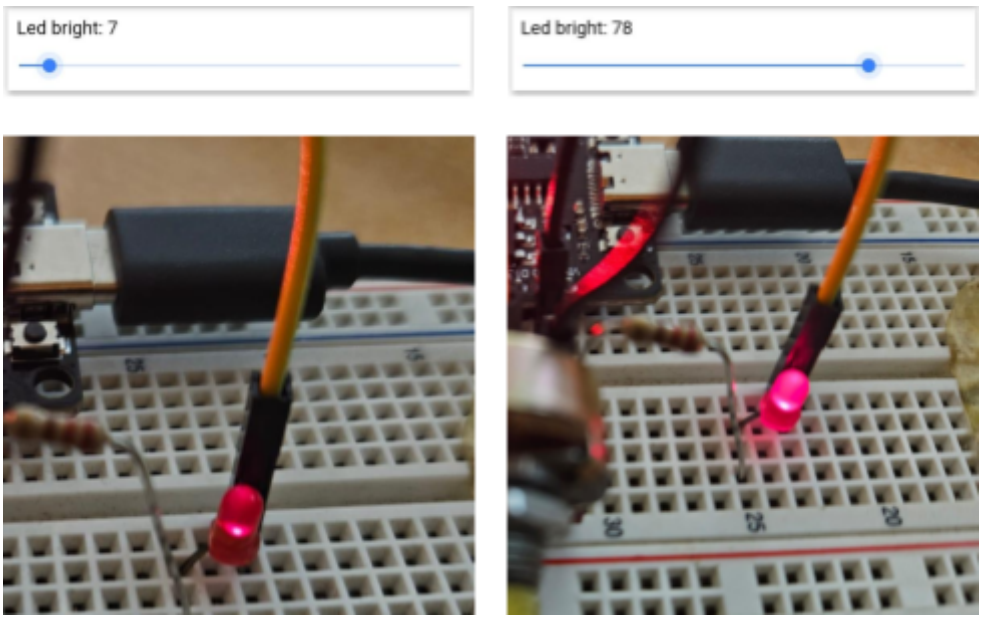
Se ejecutó un script en una PC conectada a un broker MQTT local, que escucha un topic específico donde se publican las lecturas del ADC. Estos datos se almacenaron en un archivo CSV con el formato especificado: la primera columna contiene el instante del muestreo en milisegundos y la segunda columna el valor sensado por el ADC.

```
broker="192.168.0.4"
port="1883"
topic="marcos_practica2/measurement"
archivo="out/recibidos.csv"

#------------------------------------------------

if [ ! -d "out" ]; then
    mkdir out
fi

# Agregar encabezado al archivo CSV si está vacío o no existe
if [ ! -f $archivo ] || [ ! -s $archivo ]; then
    echo "timestamp,value" > $archivo
fi

# Ponemos el cliente de mosquitto a escuchar
mosquitto_sub -t $topic -h $broker -p $port | while read value; do
    ts=$(date "+%s%3N") # Obtenemos la marca de tiempo en milisegundos
    # Guardamos valores en formato CSV:
    echo "$ts,$value" >> $archivo   # guardamos datos en archivo CSV
    echo "$value"                   # mostramos el resultado por consola
done
```

# Resultados

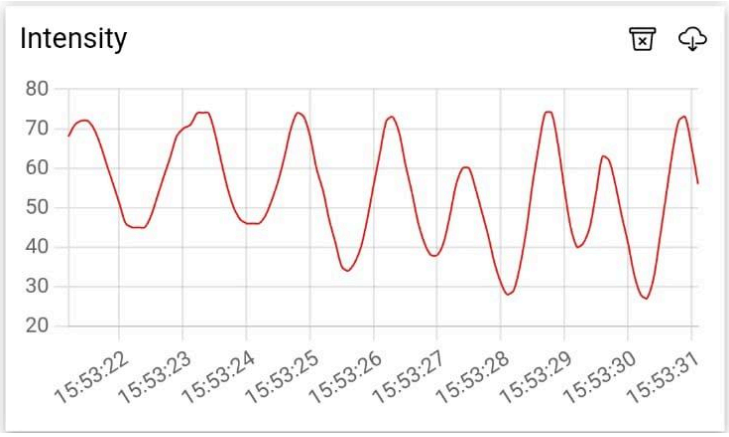
## Control del Brillo del LED

Se realizaron pruebas con distintas posiciones del Slider para controlar el brillo del LED. A continuación, se presentan las imágenes del LED en dos estados diferentes del Slider.



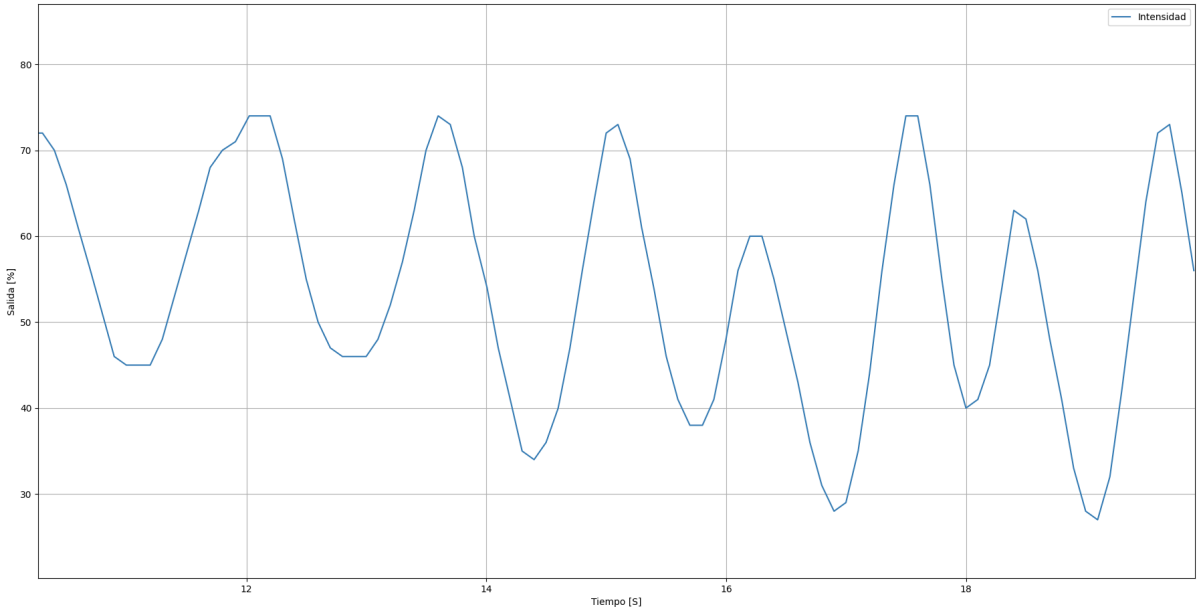

## Curva de Medición del LDR

Se modificó el porenciómentro a distintos niveles durante el periodo de muestreo de 20 segundos y se observó la curva resultante en el panel Line Graph.




## Gráfico de la Serie Temporal

Utilizando Python, se generó un gráfico de la serie temporal registrada en el archivo CSV. A continuación, se presenta el gráfico.




Se observa un gráfico de tiempo vs intensidad (%). El registro tiene un zoom para mostrar el periodo coincidente con la captura mostrada anteriormente el la app. Se muestra la última porción del registro, el cual termina en el segundo 20, donde el sistema deja de medir y reportar valores nuevos.

# Conclusiones

La implementación del sistema permitió cumplir con todos los requerimientos especificados. Se logró controlar el brillo del LED y relevar el valor sensado por el ADC en función del tiempo, visualizando los datos en tiempo real y almacenándolos correctamente en un archivo CSV.