Actuador - Proyecto Final Ingeniería Electrónica

Generated by Doxygen 1.15.0

Chapter 1

Documentaci-n_Actuador

Aquí se podrá encontrar la documentación correspondiente al actuador del Proyecto Final de Ingeniería Electrónica.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Control del Proceso y Actuador . . . . . **??**
Comunicación WiFi/MQTT . . . . . **??**
Manejo de Tareas FreeRTOS . . . . . **??**

Chapter 3

Directory Hierarchy

3.1 Directories

src . . . . . ??
Actuador.ino . . . . . ??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/Actuador.ino . . . . . ??

Chapter 5

Topic Documentation

5.1 Control del Proceso y Actuador

Implementa el controlador PI discreto y el actuador físico (servo).

Functions

- float scaleToDegrees (float u)

  Escala el mando lógico (0..100 %) al rango de grados del servo.
- void recalculateCoefficients ()

  Recalcula los coeficientes del PI discreto (Kpos, Kneg) usando Tustin.
- void setServoDegrees (float deg)

  Aplica un ángulo al servo, saturando al rango permitido e invirtiendo el sentido.
- void applyControl ()

  Aplica un paso del controlador PI discreto y actualiza el servo.

Variables

- Servo servo
- volatile float Kp = 5.0f

  Ganancia proporcional del controlador PI (Kp de MATLAB).
- volatile float Ti = 250.0f

  Tiempo integral del controlador PI (Ti de MATLAB) en segundos.
- const uint32_t CONTROL_INTERVAL_MS = (uint32_t)(Tm $*$ 1000.0f)

  Intervalo de ejecución del lazo de control en milisegundos.
- volatile float Kpos

  Coeficiente b0 del PI discreto en la forma de Tustin.
- volatile float Kneg

  Coeficiente b1 del PI discreto en la forma de Tustin.
- volatile float u_logic = 25.0f

  Salida lógica actual del controlador PI, en % (0..100).
- volatile float pos_deg = 0.0f

  Posición actual del servo en grados físicos.
- volatile float temp_setpoint = 25.0f

Temperatura de consigna (SP) recibida por MQTT.

- volatile float temp_actual = 25.0f

  Temperatura de proceso (PV) recibida por MQTT.

- volatile float error_k_1 = 0.0f

  Error de control en el instante k-1 (e[k-1]).

- volatile unsigned long sampleIndex = 0

  Índice de muestra del lazo de control (k).

- bool initial_state_reset = false

  Indica si ya se realizó el reseteo de estado inicial del controlador PI.

5.1.1 Detailed Description

Implementa el controlador PI discreto y el actuador físico (servo).

Este módulo:

- Mantiene los parámetros y estados del controlador PI discretizado por Tustin.
- Recibe SP y PV desde el módulo de comunicaciones (Comunicación WiFi/MQTT).
- Entrega un mando lógico (0..100 %) que se mapea a grados de servo.

5.1.2 Function Documentation

5.1.2.1 applyControl()

```
void applyControl ()
```

Aplica un paso del controlador PI discreto y actualiza el servo.

Implementa el algoritmo recursivo:

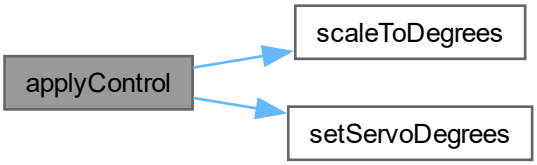
$$u[k] = u[k-1] + Kpos \cdot e[k] + Kneg \cdot e[k-1]$$

donde $e[k] = SP - PV$.

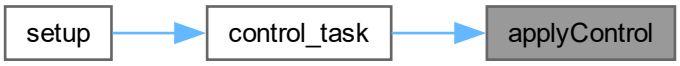
Luego:

- Se satura $u[k]$ al rango 0..100 %.
- Se actualizan error_k_1 y u_logic.
- Se convierte $u[k]$ a grados de servo y se llama a setServoDegrees().

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.2 recalculateCoefficients()

```
void recalculateCoefficients ()
```

Recalcula los coeficientes del PI discreto (Kpos, Kneg) usando Tustin.

Equivalente a la discretización realizada en MATLAB con:

```
controladorZ = c2d(controlador, Tm, 'tustin');
```

Note

Se llama:

- En setup(), para inicializar el controlador.
- Cuando se actualizan Kp o Ti vía MQTT.

Here is the caller graph for this function:

5.1.2.3 scaleToDegrees()

```
float scaleToDegrees (
            float u)
```

Escala el mando lógico (0..100 %) al rango de grados del servo.

Parameters

in	*u*	Mando lógico en porcentaje (0..100).

Returns

Ángulo en grados dentro del rango [SERVO_DEG_MIN, SERVO_DEG_MAX].

Here is the caller graph for this function:

5.1.2.4 setServoDegrees()

```
void setServoDegrees (
            float deg)
```

Aplica un ángulo al servo, saturando al rango permitido e invirtiendo el sentido.

Parameters

in	*deg*	Ángulo deseado en grados.

La función:

- Satura `deg` al rango [SERVO_DEG_MIN, SERVO_DEG_MAX].
- Actualiza la variable global pos_deg.
- Llama a servo.write() con el ángulo invertido (180 - deg) según el montaje físico.

Here is the caller graph for this function:

5.1.3 Variable Documentation

5.1.3.1 CONTROL_INTERVAL_MS

```
const uint32_t CONTROL_INTERVAL_MS = (uint32_t)(Tm * 1000.0f)
```

Intervalo de ejecución del lazo de control en milisegundos.

5.1.3.2 error_k_1

```
volatile float error_k_1 = 0.0f
```

Error de control en el instante k-1 (e[k-1]).

Se utiliza por el algoritmo recursivo del PI discreto:

$$u[k] = u[k-1] + Kpos \cdot e[k] + Kneg \cdot e[k-1]$$

5.1.3.3 initial_state_reset

```
bool initial_state_reset = false
```

Indica si ya se realizó el reseteo de estado inicial del controlador PI.

El reseteo inicial fuerza:

- u_logic = 50 %
- error_k_1 = 0
- temp_actual = temp_setpoint

Esto permite comenzar desde una condición de estado estacionario.

5.1.3.4 Kneg

```
volatile float Kneg
```

Coeficiente b1 del PI discreto en la forma de Tustin.

5.1.3.5 Kp

```
volatile float Kp = 5.0f
```

Ganancia proporcional del controlador PI (Kp de MATLAB).

Note

Se puede actualizar en tiempo de ejecución vía MQTT en el tópico `emu/kp_set`.

5.1.3.6 Kpos

```
volatile float Kpos
```

Coeficiente b0 del PI discreto en la forma de Tustin.

Los coeficientes se recalculan en recalculateCoefficients().

5.1.3.7 pos_deg

```
volatile float pos_deg = 0.0f
```

Posición actual del servo en grados físicos.

5.1.3.8 sampleIndex

```
volatile unsigned long sampleIndex = 0
```

Índice de muestra del lazo de control (k).

Se incrementa en control_task y se utiliza para:

- Logging en forma de CSV para MATLAB.
- Implementar la lógica de reset inicial del PI cuando `sampleIndex` == 5.

5.1.3.9 servo

```
Servo servo
```

Objeto servo utilizado como actuador principal.

5.1.3.10 temp_actual

```
volatile float temp_actual = 25.0f
```

Temperatura de proceso (PV) recibida por MQTT.

5.1.3.11 temp_setpoint

```
volatile float temp_setpoint = 25.0f
```

Temperatura de consigna (SP) recibida por MQTT.

5.1.3.12 Ti

```
volatile float Ti = 250.0f
```

Tiempo integral del controlador PI (Ti de MATLAB) en segundos.

Note

Se puede actualizar vía MQTT en el tópico `emu/ti_set`.

5.1.3.13 u_logic

```
volatile float u_logic = 25.0f
```

Salida lógica actual del controlador PI, en % (0..100).

Esta es la variable de mando en unidades lógicas. Luego se mapea a grados de servo mediante scaleToDegrees() y setServoDegrees().

Note

Se inicializa en 25 % y luego, tras el reset inicial (a los 5 s), se fuerza a 50 % como condición de estado estacionario.

5.2 Comunicación WiFi/MQTT

Gestión de la conectividad WiFi y del enlace con el broker MQTT.

Functions

- PubSubClient mqtt (wifiClient)
- void mqtt_task (void *parameter)

  Tarea de comunicaciones (gestión de WiFi y MQTT).

Variables

- WiFiClient wifiClient

5.2.1 Detailed Description

Gestión de la conectividad WiFi y del enlace con el broker MQTT.

Este módulo:

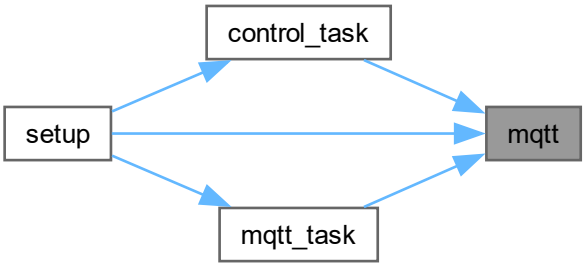
- Establece la conexión WiFi.
- Gestiona la conexión al broker MQTT y la suscripción a tópicos (SP, PV, Kp, Ti).
- Publica el estado del controlador (U lógica) y la posición del servo.
- Interpreta los mensajes de entrada para actualizar variables del módulo Control del Proceso y Actuador.

5.2.2 Function Documentation

5.2.2.1 mqtt()

```
PubSubClient mqtt (
            wifiClient )
```

Cliente MQTT basado en PubSubClient. Here is the caller graph for this function:



5.2.2.2 mqtt_task()

```
void mqtt_task (
            void * parameter)
```

Tarea de comunicaciones (gestión de WiFi y MQTT).

Parameters

in	*parameter*	Puntero genérico no utilizado (debe ser NULL).

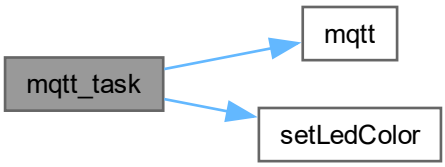
La tarea:

- Supervisa permanentemente el estado de la conexión WiFi.
- En caso de desconexión:
  - Muestra el LED en rojo.
  - Registra mensajes de diagnóstico por serie.
  - Reintenta la conexión WiFi con un período definido.
- Cuando hay conexión WiFi:
  - Muestra el LED en verde.
  - Verifica y restablece la conexión MQTT si es necesario.
  - Llama periódicamente a mqtt.loop().

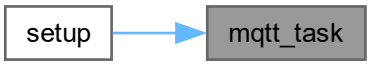
Note

Si se pierde la comunicación con el broker, el lazo de control se congela en el último estado conocido, ya que los datos de SP, PV y sintonía se encuentran centralizados en el broker.

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3 Variable Documentation

5.2.3.1 wifiClient

```
WiFiClient wifiClient
```

Cliente TCP/IP para la pila WiFi.

5.3 Manejo de Tareas FreeRTOS

Organización de la ejecución en tiempo real mediante tareas FreeRTOS.

Functions

- void control_task (void ∗parameter)

  Tarea de control discreto (PI + servo + logging).
- void setup ()

  Función de inicialización del firmware.

5.3.1 Detailed Description

Organización de la ejecución en tiempo real mediante tareas FreeRTOS.

Este módulo:

- Define la tarea de control discreto (control_task).
- Define la tarea de comunicaciones (mqtt_task).
- Crea y configura las tareas desde setup().

5.3.2 Function Documentation

5.3.2.1 control_task()

```
void control_task (
            void * parameter)
```

Tarea de control discreto (PI + servo + logging).

Parameters

in	*parameter*	Puntero genérico no utilizado (debe ser NULL).

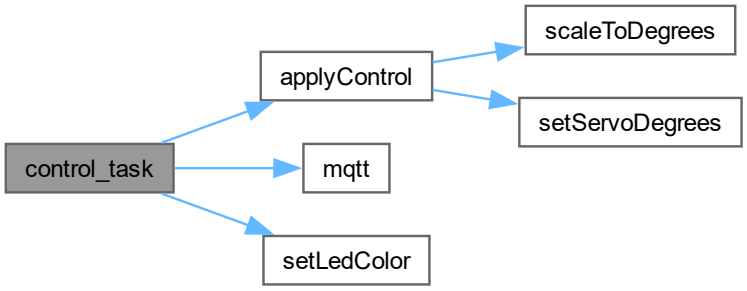
La tarea:

- Se ejecuta periódicamente cada CONTROL_INTERVAL_MS milisegundos mediante vTaskDelayUntil().
- Verifica que exista conexión WiFi y MQTT antes de aplicar el control.
- Aplica la lógica de reseteo inicial del PI cuando sampleIndex == 5:
  - Fuerza u_logic = 50 %.
  - Resetea error_k_1.
  - Igual temp_actual a temp_setpoint.
- Llama a applyControl() y publishState().
- Envía al puerto serie un log en formato CSV:
  `k,t,SP,PV,U_logic,Deg`
- Genera un breve pulso del LED para indicar actividad de control.

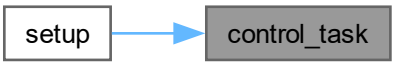
Note

Esta tarea nunca retorna.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 setup()

```
void setup ()
```

Función de inicialización del firmware.

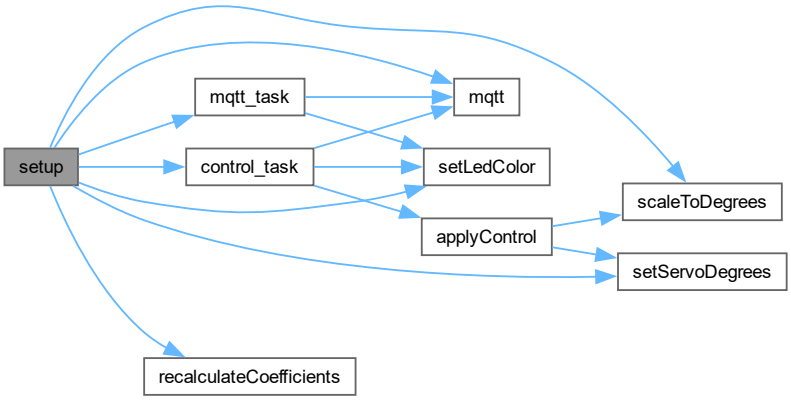
Realiza:

- Inicialización del puerto serie y encabezado de log CSV.
- Inicialización del LED integrado (apagado).
- Cálculo inicial de los coeficientes del PI discreto.
- Configuración y posicionamiento inicial del servo.
- Conexión inicial a la red WiFi con timeout de 10 s.
- Configuración del cliente MQTT y registro del callback.
- Creación de las tareas FreeRTOS:
  - control_task con prioridad 3.
  - mqtt_task con prioridad 2.

Note

La lógica de control y comunicaciones queda delegada por completo a las tareas.

Here is the call graph for this function:



Chapter 6

Directory Documentation

6.1 src Directory Reference

Files

- file Actuador.ino

Chapter 7

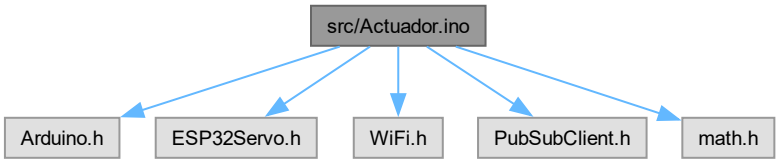
File Documentation

7.1 README.md File Reference

7.2 src/Actuador.ino File Reference

```
#include <Arduino.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <math.h>
```

Include dependency graph for Actuador.ino:



Macros

- #define RGB_BRIGHTNESS 64

  Brillo lógico utilizado para el LED integrado (0..255).

Functions

- void setLedColor (uint8_t r, uint8_t g, uint8_t b)

  Actualiza el color lógico del LED y enciende/apaga el LED integrado.
- PubSubClient mqtt (wifiClient)
- float scaleToDegrees (float u)

  Escala el mando lógico (0..100 %) al rango de grados del servo.

- void recalculateCoefficients ()

  Recalcula los coeficientes del PI discreto (Kpos, Kneg) usando Tustin.
- void setServoDegrees (float deg)

  Aplica un ángulo al servo, saturando al rango permitido e invirtiendo el sentido.
- void applyControl ()

  Aplica un paso del controlador PI discreto y actualiza el servo.
- void control_task (void ∗parameter)

  Tarea de control discreto (PI + servo + logging).
- void mqtt_task (void ∗parameter)

  Tarea de comunicaciones (gestión de WiFi y MQTT).
- void setup ()

  Función de inicialización del firmware.
- void loop ()

  Bucle principal de Arduino (no utilizado).

Variables

- Servo servo
- volatile uint32_t current_color = 0

  Color lógico actual del LED, empaquetado como 0xRRGGBB.
- volatile float Kp = 5.0f

  Ganancia proporcional del controlador PI (Kp de MATLAB).
- volatile float Ti = 250.0f

  Tiempo integral del controlador PI (Ti de MATLAB) en segundos.
- const uint32_t CONTROL_INTERVAL_MS = (uint32_t)(Tm ∗ 1000.0f)

  Intervalo de ejecución del lazo de control en milisegundos.
- volatile float Kpos

  Coeficiente b0 del PI discreto en la forma de Tustin.
- volatile float Kneg

  Coeficiente b1 del PI discreto en la forma de Tustin.
- volatile float u_logic = 25.0f

  Salida lógica actual del controlador PI, en % (0..100).
- volatile float pos_deg = 0.0f

  Posición actual del servo en grados físicos.
- volatile float temp_setpoint = 25.0f

  Temperatura de consigna (SP) recibida por MQTT.
- volatile float temp_actual = 25.0f

  Temperatura de proceso (PV) recibida por MQTT.
- volatile float error_k_1 = 0.0f

  Error de control en el instante k-1 (e[k-1]).
- volatile unsigned long sampleIndex = 0

  Índice de muestra del lazo de control (k).
- bool initial_state_reset = false

  Indica si ya se realizó el reseteo de estado inicial del controlador PI.
- WiFiClient wifiClient

Tópicos de suscripción para sintonización del PI

Tópicos de suscripción para señales de control

Tópicos de publicación de estado

7.2.1 Macro Definition Documentation

7.2.1.1 RGB_BRIGHTNESS

```
#define RGB_BRIGHTNESS 64
```

Brillo lógico utilizado para el LED integrado (0..255).

Note

Actualmente solo se usa para decidir encendido/apagado, no para controlar un LED RGB real.

7.2.2 Function Documentation

7.2.2.1 loop()

```
void loop ()
```

Bucle principal de Arduino (no utilizado).

Todas las funcionalidades del sistema están implementadas en las tareas FreeRTOS. Esta función permanece vacía.

7.2.2.2 setLedColor()

```
void setLedColor (
            uint8_t r,
            uint8_t g,
            uint8_t b)
```

Actualiza el color lógico del LED y enciende/apaga el LED integrado.

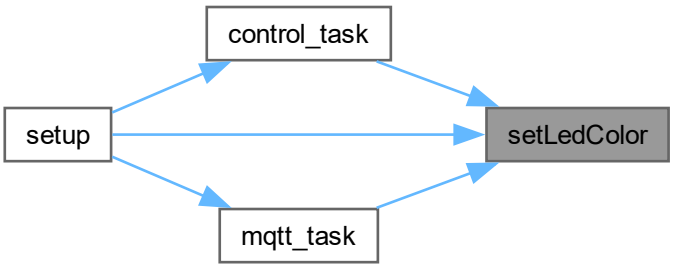
Parameters

in	*r*	Componente rojo (0..255).
in	*g*	Componente verde (0..255).
in	*b*	Componente azul (0..255).

Note

Actualmente el hardware solo permite un LED "on/off", por lo que se enciende el pin `RGB_BUILTIN` si el color es distinto de negro. El valor de current_color se mantiene para futuros usos.

Here is the caller graph for this function:



7.2.3 Variable Documentation

7.2.3.1 current_color

```
volatile uint32_t current_color = 0
```

Color lógico actual del LED, empaquetado como 0xRRGGBB.

Se utiliza para recordar el estado previo del LED y restaurarlo luego de pequeños pulsos de actividad (por ejemplo, en la tarea de control).