DOCUMENTACIÓN DE UN SISTEMA DE CONTROL DE ACCESO

PRIMER PARCIAL DE ESTRUCTURAS COMPUTACIONALES

SAMUEL MARULANDA SALAZAR

PROFESOR:

SAMUEL ANDRES CIFUENTES MUÑOZ

MANIZALES

2024

Este README describe un sistema de control de acceso simplificado implementado en una placa STM32 Nucleo. El sistema emula un mecanismo de puerta segura con indicaciones básicas, eventos de entrada y capacidades de control/monitoreo remoto vía UART.

El siguiente ejercicio debe contener los siguientes parámetros:

Recursos de Hardware

- **LED de Latido (Heartbeat):** Un LED dedicado que parpadea periódicamente (cada 500 ms) para indicar que el sistema está funcionando correctamente.

- **LED de Estado de la Puerta:** Indica el estado actual de la puerta:
    - Encendido (fijo): La puerta está desbloqueada (temporal o permanentemente).
    - Apagado: La puerta está bloqueada.

- **Botón de Abrir/Cerrar Puerta:**
    - **Pulsación Sencilla:** Desbloquea temporalmente la puerta (por un tiempo predefinido, ej. 5 segundos).
    - **Doble Pulsación:** Desbloquea la puerta de forma permanente hasta recibir un comando de cierre (por UART u botón de control).

- **UART:**
    - Recibe comandos como **O (OPEN)** para abrir y **C (CLOSE)** para cerrar desde un sistema externo (PC u otro controlador).
    - Envía eventos de registro como **DOOR_UNLOCKED** y **DOOR_LOCKED.**

- **Opcional:**
    - **Botón de Timbre (Ring Bell):** Pulsar este botón podría generar un evento de “timbre” (**BELL_RING**) enviado por UART u otra acción definida por la aplicación. Este botón es opcional y no afecta la lógica principal del control de la puerta.

Operación a Alto Nivel

1. **Siempre:**

- El LED de heartbeat parpadea periódicamente.

2. **Inactivo (Bloqueado):**
   - El LED de Estado de la Puerta está apagado.
   - El sistema espera eventos:
     - Pulsación sencilla del botón de abrir/cerrar: Desbloqueo temporal.
     - Doble pulsación del botón de abrir/cerrar: Desbloqueo permanente.
     - Comando UART:
       - **O (OPEN):** Desbloquear la puerta.
       - **C (CLOSE):** Bloquear la puerta (si está desbloqueada).

3. **Desbloqueado Temporalmente:**
   - El LED de Estado de la Puerta se enciende (fijo).
   - Inicia un temporizador (ej. 5 segundos). Al expirar, la puerta se vuelve a bloquear (LED apagado) y se envía **DOOR_LOCKED**.
   - Una pulsación adicional del botón antes de que el tiempo expire puede cambiar a modo desbloqueo permanente.
   - Un comando **C** vía UART bloquea inmediatamente la puerta.

4. **Desbloqueado Permanentemente:**
   - El LED de Estado de la Puerta se enciende (fijo).
   - La puerta permanece desbloqueada indefinidamente hasta que:
     - Se reciba un comando **C (CLOSE)** por UART.
     - Una pulsación sencilla del botón de abrir/cerrar indique volver a bloquear.

Se hace un diagrama para entender mejor el código:

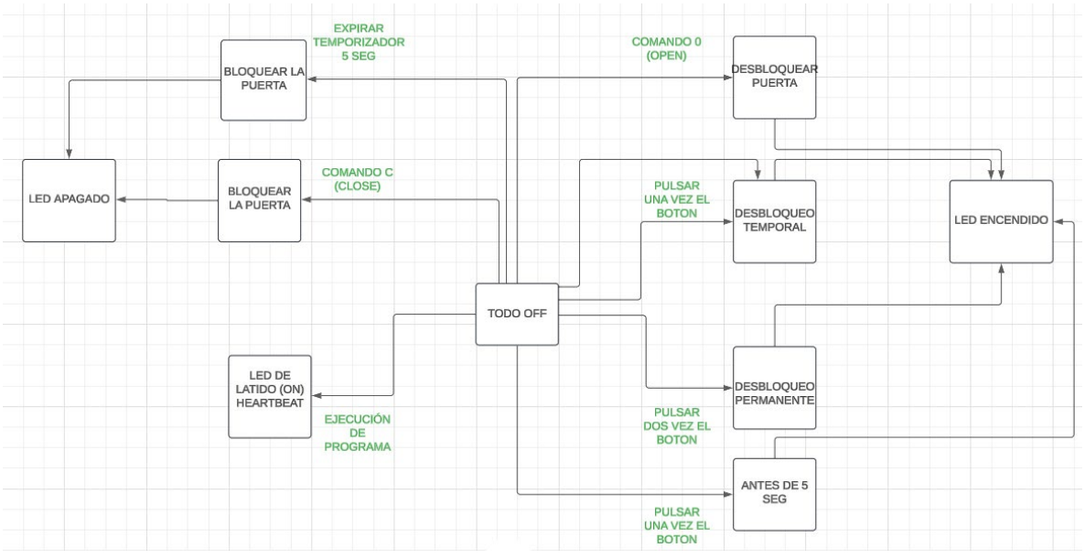

Ilustración 1: DIAGRAMA

A continuación, se explicará la solución del parcial:

Bibliotecas Incluidas y Definiciones de Registros

```
#include "gpio.h"     // Librería para manejar los pines GPIO
#include "rcc.h"      // Librería para configuración del reloj
#include "systick.h"  // Librería para manejar retardos y temporizadores
```

Estas bibliotecas permiten acceder a funcionalidades específicas del hardware.

Definiciones de registros

```
#define EXTI_BASE 0x40010400
#define EXTI ((EXTI_t *)EXTI_BASE)
```

Se define la dirección base y se crea un puntero para acceder a los registros EXTI (Control de interrupciones externas).

```
#define EXTI15_10_IRQn 40
#define NVIC_ISER1 ((uint32_t *)(0xE000E104))
```

Se define el número de interrupción para pines del 10 al 15 y el registro para habilitar dicha interrupción en el NVIC.

```
#define SYSCFG_BASE 0x40010000
#define SYSCFG ((SYSCFG_t *)SYSCFG_BASE)
```

Se configura el sistema para mapear la interrupción al pin correspondiente.

Configuración del GPIO para UART

```
void configure_gpio_for_usart(void)
```

Esta función configura los pines **PA2** (TX) y **PA3** (RX) para funcionar en modo UART:

- **Habilita el reloj** para GPIOA.
- **Configura los pines como función alternativa**.
- **Asigna la función alternativa AF7**, necesaria para USART.
- **Ajusta velocidad y resistencias internas** para garantizar la transmisión de datos.

Configuración del GPIO General

```
void configure_gpio(void)
```

Esta función configura el LED (PA5) como salida y el botón (PC13) como entrada:

- Habilita el reloj para los puertos GPIOA y GPIOC.
- Configura el LED como salida push-pull.
- Configura el botón como entrada con resistencia pull-up o pull-down.
- Configura el EXTI para generar interrupciones por flanco descendente en el pin del botón.

Funciones de Control del Sistema

void gpio_set_door_led_state(uint8_t state)

Controla el estado del LED de la puerta. Enciende o apaga el LED según el estado solicitado.

void gpio_toggle_heartbeat_led(void)

Alterna el estado del LED de latido para indicar que el sistema está activo.

Detección de Eventos en el Botón

uint8_t button_driver_get_event(void)

Devuelve el estado del botón (presionado o no) y lo reinicia.

void detect_boton_presionado(void)

Detecta pulsaciones simples o dobles en el botón. Verifica los tiempos entre eventos para ignorar rebotes y diferenciar entre pulsaciones simples y dobles.

Interrupción del Botón

void EXTI15_10_IRQHandler(void)

Este manejador de interrupciones se activa cuando se presiona el botón.

- **Limpia la bandera de interrupción** para evitar activaciones repetidas.
- **Llama a la función para detectar pulsaciones**, lo que permite procesar el evento correspondiente.

Máquina de Estados (main.c)

La máquina de estados controla el comportamiento del sistema dependiendo de las entradas (botón o UART).

Estados del sistema

```
typedef enum {
  LOCKED,
  TEMP_UNLOCK,
  PERM_UNLOCK
} DoorState_t;
```

- **LOCKED**: La puerta está bloqueada.
- **TEMP_UNLOCK**: La puerta está desbloqueada temporalmente por un tiempo definido.
- **PERM_UNLOCK**: La puerta está desbloqueada permanentemente hasta recibir otra instrucción.

Lógica de Estados

```
void run_state_machine(void)
```

Gestiona las transiciones de estados:

1. En estado **LOCKED**, no realiza ninguna acción periódica.
2. En estado **TEMP_UNLOCK**, regresa al estado bloqueado después de que haya transcurrido un tiempo determinado.
3. En estado **PERM_UNLOCK**, mantiene la puerta desbloqueada indefinidamente.

Manejo de Eventos

```
void handle_event(uint8_t event)
```

Procesa los eventos provenientes del botón o UART:

- Pulsación única del botón -> Desbloqueo temporal.
- Pulsación doble del botón -> Desbloqueo permanente.

- Comando UART 'O' -> Desbloqueo temporal.
- Comando UART 'C' -> Bloqueo inmediato.

Bucle Principal

int main(void)

1. Inicializa el sistema (reloj, GPIO, UART).
2. Enciende el LED de latido cada 500 ms para indicar actividad.
3. Verifica los eventos del botón o comandos UART.
4. Actualiza el estado de la máquina de estados.

Módulo UART

Inicialización de USART2

void usart2_init(void)

1. Configura los pines para transmisión y recepción (TX, RX).
2. Habilita el reloj para USART2.
3. Configura la velocidad de transmisión (9600 baudios).
4. Habilita transmisor y receptor.
5. Activar interrupciones para recepción de datos.

Enviar Datos

void usart2_send_string(const char *str)

Envía cadenas de texto mediante UART, esperando que el registro de transmisión esté disponible antes de enviar cada carácter.

Recepción de Comandos

command_t usart2_get_command(void)

Devuelve el último comando recibido ('O' para abrir, 'C' para cerrar) y lo reinicia.

Manejador de Interrupciones

void USART2_IRQHandler(void)

1. Verifica si hay datos recibidos.
2. Lee el dato recibido y actualiza el comando.