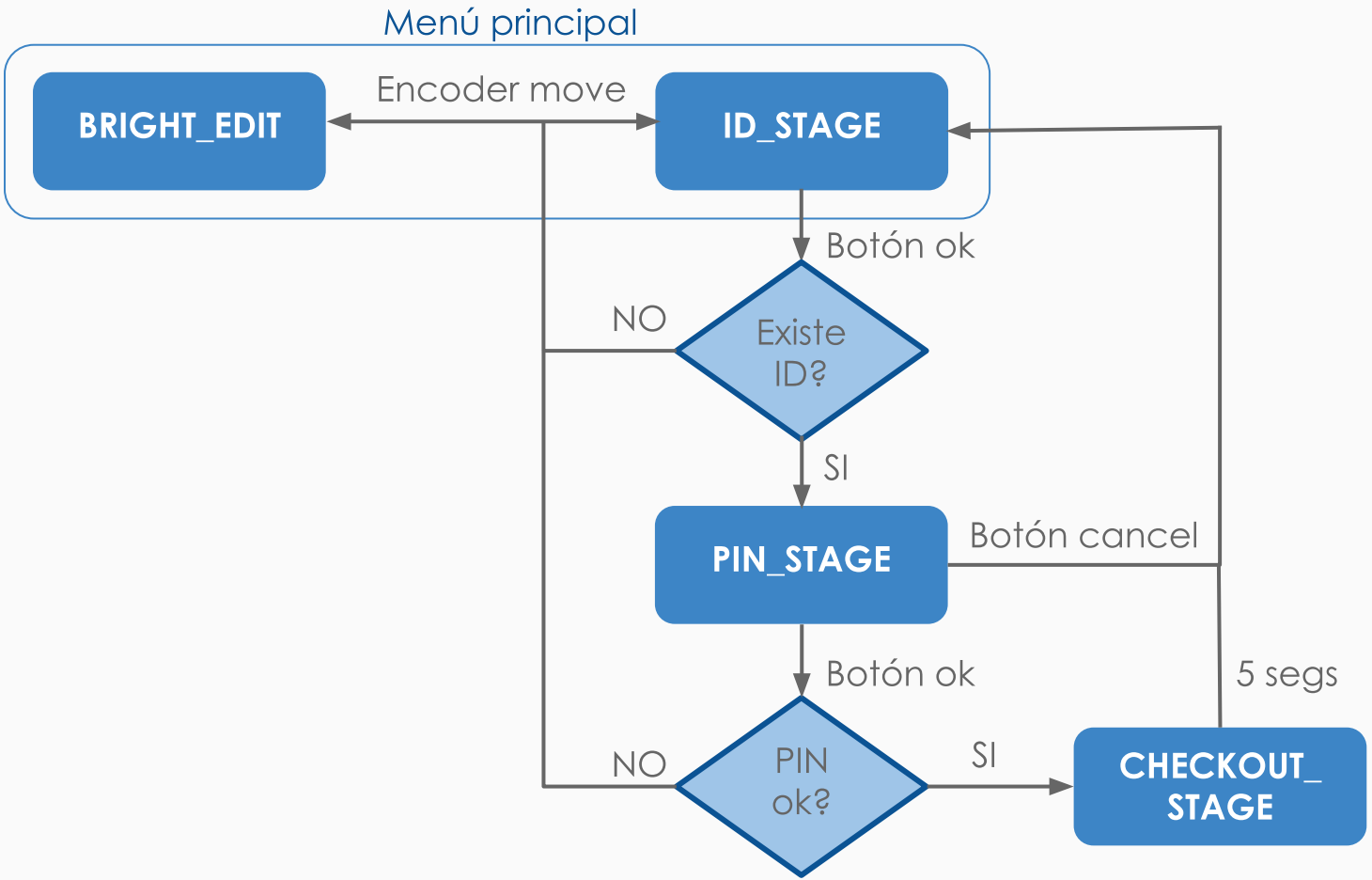
Laboratorio de Microprocesadores

Trabajo Práctico №4

ITBA

Máspero Martina
Mestanza Joaquín
Nowik Ariel
Regueira Marcelo



Máquina de estados

Encoder

Utiliza internamente lectura de las entradas por GPIO, y una ventana de tiempo periódica para filtrar posibles rebotes. Solamente avisa a la App cuando ocurrió algún evento que requiere su atención.

Encoder.h

void encoderInit(**void** (*funcallback)(**void**));

enc_flag_t **encoderMotionGetEvent**(**void**);

Display

Se maneja periódicamente desde la App, que debe refrescarlo cada 1 ms.
La estructura general del mensaje es:

Palabra de Menú

+

Número Código

Permite modo interactivo o modo scroll para mensajes.

Display.h

void DispBoard_Init(**void**);

void DispShowMsj(disp_msj_t msj);

void DispShiftMsj(**void**);

disp_cursor_t **DispGetCursor**(**void**);

disp_task_t **DispModType**(**void**);

disp_bright_t **DispChangeBright**(**int** move_dir);

void DispClear(**void**);

void LedStatus_Write(**int** code);

int LedStatus_GetState(**void**);

Lector

Usa interrupciones dedicadas para hacer la lectura de la tarjeta. Al completar una lectura se llama al callback que la App registra en el inicializador.

Lector.h

```
int * lector_get_PAN (void);

void lectorInit (void (*funcallback)(void));
```

Users

Inicializa los usuarios por defecto.
Valida los accesos pedidos y bloquea un ID si supera 3 intentos fallidos.

Lleva la cuenta de la cantidad de usuarios que ingresan por piso.

Users.h

```
void initUser(void);

bool validateID(int * id);

bool validateUser(int * id, int * pin);

uint8_t getUsers1Floor (void);

uint8_t getUsers2Floor (void);

uint8_t getUsers3Floor (void);
```

Internal Control

Inicializa utilidades de la placa Freedom (pulsadores SW2 y SW3 para OK y CANCELAR, y el led RGB). El led RGB se utiliza como indicador:

- BLUE: funcionamiento normal
- GREEN: se concede acceso por 5 segundos
- RED: intento de acceso fallido (+ retardo en vuelta al menú)

InternalControl.h

```
void internalControlInit(void (*funcallback)(void));

int internalControlGetEvent(void);

void RGBIndicator(int led_color);
```

UART

Para poder comunicarse desde la placa kinetis hacia el gateway utilizamos protocolo UART.
Con la función sendWord se puede enviar un string.

UART.h

```
void uartInit(uart_cfg_t config);

void sendWord(const char *word);
```

UART

Se agregó la recepción. Se utiliza mediante dos funciones, updateWord() que procesa los nuevos caracteres recibidos y setOnNewCharListener para establecer qué debe suceder cuando llega un nuevo caracter.

UART.h

```
void uartInit(uart_cfg_t config);

void sendWord(const char *word);

void updateWord();
void setOnNewCharListener(void *recv(char));
```

División por Tareas

TASK START

```
OSMutexPend(&mutex, 0, OS_OPT_PEND_BLOCKING,
(CPU_TS*)0, &os_err);
        usersNum[0] = getUsers1Floor();
        usersNum[1] = getUsers2Floor();
        usersNum[2] = getUsers3Floor();
        new_info = true;
        OSMutexPost(&mutex,
OS_OPT_PEND_BLOCKING, &os_err);

OSSemPend(&ValidUserSem,0,OS_OPT_PEND_BLOCKI
NG,(CPU_TS*)0,&os_err);
```

TASK 2

```
OSTimeDlyHMSM(0, 0, 3, 0,
OS_OPT_TIME_HMSM_STRICT, &os_err);
sendKeepAlive();
                OSSemPend(&KeepAliveSem,0,OS_OPT_
PEND_BLOCKING,(CPU_TS*)0,&os_err); // espero
KeepAliveOk
  if(i==14){
        if(new_info){
        OSMutexPend(&mutex, 0,
OS_OPT_PEND_BLOCKING, (CPU_TS*)0, &os_err);
        senddata2thingspeak();
        OSMutexPost(&mutex,
OS_OPT_PEND_BLOCKING, &os_err);
                                        OSSemPend(&uartSe
        m,0,OS_OPT_PEND_BLCKING,(CPU_TS*)0,&os
        _err); // espero sendDataOk
```

ThingSpeak

1. Corremos el script gateway.js
2. Mediante UART mandamos los paquetes al gateway y recibimos sus respuestas.
3. Si no se recibe una respuesta en un tiempo dado, se despierta al thread mediante un timeout.