TP Systèmes et Architectures Temps Réel

FreeRTOS, esp-idf, esp32, Queues, UART

2022-11-28

Guilhem Saurel

This presentation

Available at

`https://nim65s.github.io/talks/tp-satr.pdf`

Under License

This presentation (continued)

Source

`https://github.com/nim65s/talks : tp-satr.md`

Discussions

https://matrix.to/#/#conception-orientee-objet:laas.fr

FreeRTOS

https://freertos.org

ESP-IDF

https://github.com/espressif/esp-idf

https://docs.espressif.com/projects/esp-idf/

Hello World

```
cp -r $IDF_PATH/examples/get-started/hello_world .
cd hello_world
idf.py set-target esp32
# idf.py menuconfig

# sdkconfig:
CONFIG_XTAL_FREQ_26=y
CONFIG_XTAL_FREQ=26

idf.py build
idf.py flash
idf.py monitor
```

Generic GPIO

```
cp -r $IDF_PATH/examples/peripherals/gpio/generic_gpi

cd hello_world
idf.py set-target esp32
# idf.py menuconfig

# sdkconfig:
CONFIG_XTAL_FREQ_26=y
CONFIG_XTAL_FREQ=26

idf.py build
idf.py flash
idf.py monitor
```

Banque: Server

```
cp -r $IDF_PATH/examples/peripherals
    /uart/uart_async_rxtxtasks .

cd uart_async_rxtxtasks
idf.py set-target esp32

# sdkconfig:
CONFIG_XTAL_FREQ_26=y
CONFIG_XTAL_FREQ=26

idf.py build
idf.py flash
idf.py monitor
```

Banque: UART

```
# main/uart_async_rxtxtasks_main.c
-#define TXD_PIN (GPIO_NUM_4)
-#define RXD_PIN (GPIO_NUM_5)
+#define TXD_PIN (GPIO_NUM_1)
+#define RXD_PIN (GPIO_NUM_3)

UART_NUM_1 -> UART_NUM_0
```

Messages

`typedef struct { ... } message_t;`, avec:

- MessageType: Entier, 8bits
- Action: Entier, 8bits
- RequestId: Entier, 8bits
- PortId: Entier, 8bits
- Amount: Entier, 32 bits

Header:

```
const uint8_t HEADER[4] = {0xCA, 0xFE, 0xCA, 0xFE};
```

Queues

```
static xQueueHandle rx_queue = NULL;
static xQueueHandle tx_queue = NULL;
```

Et les initialiser dans `init()`

Tasks

- `uart_rx_task`
- `uart_tx_task`
- `bank_task`

Rx Task

- lit les bytes sur `UART_NUM_0` un par un
- quand 4 bytes correspondent à `HEADER`, lit `sizeof(message_t)`
- envoie ce `message_t` sur `rx_queue`

Bank Task

- possède un `uint32_t balance = 100;`
- pour chaque message sur `rx_queue`
  - fait l'opération demandée si elle est possible
  - écrit une réponse, et l'envoie sur `tx_queue`

Tx Task

- pour chaque messages sur `tx_queue`:
  - envoie le `HEADER` sur `UART_NUM_0`
  - envoie le message sur `UART_NUM_0`

Client

- `struct`
- `serial(python -m pip install pyserial)`