



UNIVERSIDAD  
**COMPLUTENSE**  
MADRID

# ESP-IDF. Comunicación y Sincronización

ARQUITECTURA INTERNA DEL NODO IoT

# Comunicación entre tareas FreeRTOS

- Si todas las tareas comparten el espacio de direcciones, la forma más sencilla de comunicación será... a través del uso de variables *normales*
  - Pero debemos declararlas de modo que la visibilidad permita su acceso

```
...
xTaskCreate(&ejemploTask, "Ejemplo", 3072, NULL, 5, NULL);
...
xTaskCreate(&ejemploTask, "EjemploPinned", 3072, NULL, 5, NULL, 0);
....
```

```
→ int32_t varEjemplo = 0;
void ejemploTask( void *pvParameters )
{
    for (i=0;i<10;i++) {
        varEjemplo++;
    }
    vTaskDelete( NULL );
}
```

¿Valor final de  
**varEjemplo**?  
→ Si no hay cerreras, → **20**

lectura  
escritura{

# Comunicación entre tareas FreeRTOS

- Si todas las tareas comparten el espacio de direcciones, la forma más sencilla de comunicación será... a través del uso de variables *normales*
  - Pero debemos declararlas de modo que la visibilidad permita su acceso

```
...
xTaskCreate(&ejemploTask, "Ejemplo", 3072, NULL, 5, NULL);
...
xTaskCreate(&ejemploTask, "EjemploPinned", 3072, NULL, 5, NULL, 0);
...

void ejemploTask( void *pvParameters )
{
    int32_t varEjemplo = 0;
    for (i=0;i<10;i++) {
        varEjemplo++;
    }
    vTaskDelete( NULL );
}
```

¿Valor final de  
**varEjemplo**?

2 variables varEjemplo  
↳ **10**

# Comunicación entre tareas FreeRTOS

- Si todas las tareas comparten el espacio de direcciones, la forma más sencilla de comunicación será... a través del uso de variables *normales*
  - Pero debemos declararlas de modo que la visibilidad permita su acceso

```
...
xTaskCreate(&ejemploTask, "Ejemplo", 3072, NULL, 5, NULL);
...
xTaskCreate(&ejemploTask, "EjemploPinned", 3072, NULL, 5, NULL, 0);
...

void ejemploTask( void *pvParameters )
{
    static int32_t varEjemplo = 0;
    for (i=0;i<10;i++) {
        varEjemplo++;
    }
    vTaskDelete( NULL );
}
```



¿Valor final de  
**varEjemplo**?

(carreras) → 20

# Comunicación y sincronización

- ❑ Pero sólo con variables compartidas no tenemos control sobre el **orden relativo** en el que ocurren los accesos

```
...
xTaskCreate(&ejemploTask, "Ejemplo", 3072, NULL, 5, NULL);
...
xTaskCreate(&ejemploTask2, "EjemploPinned", 3072, NULL, 5, NULL, 0);
...

int32_t varEjemplo = 0;
void ejemploTask( void *pvParameters )
{
    for (i=0;i<10;i++) {
        varEjemplo++;
    }
    vTaskDelete( NULL );
}

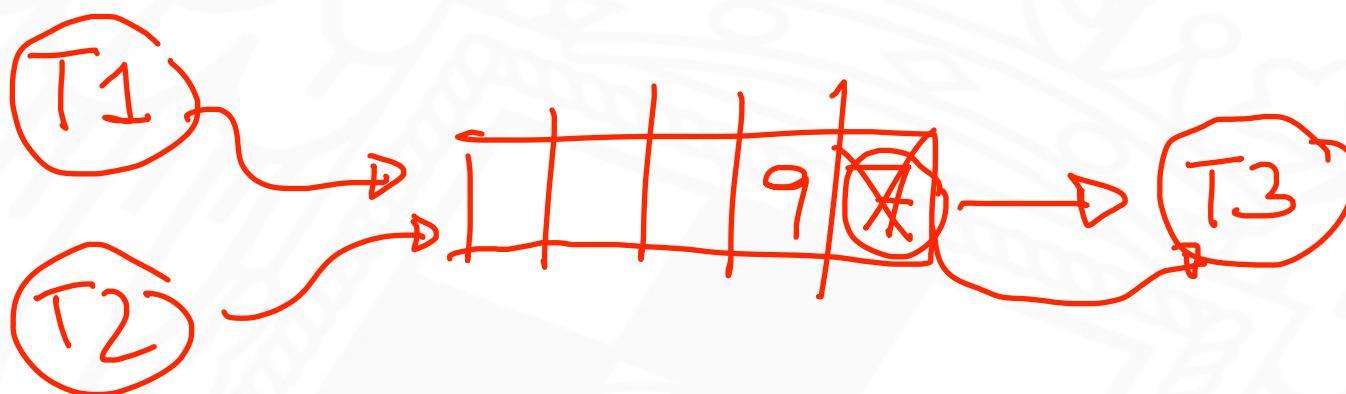
void ejemploTask2( void *pvParameters )
{
    for (i=0;i<10;i++) {
        varEjemplo+=2;
    }
    vTaskDelete( NULL );
}
```

¿Qué secuencia de valores toma **varEjemplo**?

# Comunicación por colas

## ❑ Estructuras de comunicación con escrituras y lecturas potencialmente bloqueantes

- Escritura se bloquea si la cola está llena; Lectura se bloque si la cola está vacía
- La lectura elimina el elemento de la cola



# API de ESP-IDF (similar a FreeRTOS)

**QueueHandle\_t** xQueueCreate( **UBaseType\_t** uxQueueLength, **UBaseType\_t** uxItemSize );

*nº elementos* *tamaño*

**BaseType\_t** xQueueSendToFront( **QueueHandle\_t** xQueue, **const void \*** pvItemToQueue,  
**TickType\_t** xTicksToWait );

**BaseType\_t** xQueueSendToBack( **QueueHandle\_t** xQueue, **const void \*** pvItemToQueue,  
**TickType\_t** xTicksToWait );

**BaseType\_t** xQueueReceive( **QueueHandle\_t** xQueue, **void \* const** pvBuffer,  
**TickType\_t** xTicksToWait );

API completo en

<https://docs.espressif.com/projects/esp-idf/en/stable/api-reference/system/freertos.html#queue-api>

# Ejemplo de uso

```

static void vSenderTask( void *pvParameters ) {
    int32_t lValueToSend;
    BaseType_t xStatus;
    lValueToSend = ( int32_t ) pvParameters;
    for( ; ; ) {
        xStatus = xQueueSendToBack( xQueue,
&lValueToSend, 0 );
        if( xStatus != pdTRUE ) {
            printf("Error sending...\n");
        }
    }
}

```

~~QueueHandle\_t xQueue;~~

```

void app_main( void ) {
    xQueue = xQueueCreate( 5, sizeof( int32_t ) );
    if( xQueue != NULL ) {
        xTaskCreate( vSenderTask, "Sender1", 1000, ( void * ) 100, 1, NULL );
        xTaskCreate( vSenderTask, "Sender2", 1000, ( void * ) 200, 1, NULL );
        xTaskCreate( vReceiverTask, "Receiver", 1000, NULL, 2, NULL );
    }
    ...
}

```

```

static void vReceiverTask( void *pvParameters )
{
    int32_t lReceivedValue;
    BaseType_t xStatus;
    const TickType_t xTicksToWait = pdMS_TO_TICKS( 100 );
    while (1) {
        xStatus = xQueueReceive( xQueue, &lReceivedValue,
xTicksToWait );
        If ( xStatus == pdTRUE ) {
            vPrintStringAndNumber( "Received = ",lReceivedValue );
        }
    }
}

```

# Tipos de objetos en colas

- ¿Y si queremos enviar información de diferente naturaleza?
  - Uso de structs
- En ESP-IDF los envíos en las colas se hacen por **copia**
  - Uso de punteros
- Total flexibilidad: uso de punteros a structs

```
envue {  
    tipo_1,  
    + po2,  
    :  
    : tipo_t;
```

```
struct {  
    tipo_t claseDato;  
    void * data;  
}
```