

TRABAJO PRÁCTICO N° 4 **RTOS**

Se desea implementar el control de acceso a un edificio de oficinas mediante una tarjeta magnética y un pin (clave de acceso). Para lograr el objetivo se dispone de encoder rotativo, un *display* de siete segmentos y un lector de tarjetas magnéticas. El control de acceso tendrá conectividad con la nube, para poder visualizar la ocupación de los distintos pisos del edificio.

Requerimientos obligatorios

El control de acceso deberá estar basado en código propio desarrollado para el TP1.

La conectividad con la nube se usará para transmitir el conteo de personas que ingresaron al edificio, segregadas por el piso al que se dirigen. Para eso, se deberán pre-grabar al menos 6 tarjetas:

- Los usuarios con las tarjetas 1 y 2 se dirigen al piso 1.
- Los usuarios con las tarjetas 3 y 4 se dirigen al piso 2.
- Los usuarios con las tarjetas 5 y 6 se dirigen al piso 3.

Al pasar una tarjeta y entrar al edificio, el control de acceso incrementará la cuenta de personas que entraron y enviará la información a la nube. La conectividad será con la plataforma ThingSpeak, donde se mostrarán tres gráficos (uno por piso) que indiquen la cantidad de personas que se dirigieron a cada piso.

Se debe mantener la fluidez de la interfaz de usuario aún si hay delays de comunicación con la nube.

Requerimientos deseados

Además de los requerimientos del TP1, se podrá usar un LED que indique si el sistema está conectado a la nube o no (keepalive).

Implementación

A diferencia del TP1, se deberá usar un RTOS, para separar el programa en (al menos) dos threads:

1. Thread principal: la misma funcionalidad del TP1.
2. Thread logging: implementará la conectividad con la nube.

Se utilizará el puerto serie del microcontrolador para comunicarse con un programa (en la PC) que se encargará de hablar con la plataforma ThingSpeak, haciendo una especie de gateway UART<->ThingSpeak. Ver más detalles abajo.

En **todo** el código, se deberá eliminar loops de polling, reemplazándolos por mecanismos de sincronización del sistema operativo. Para eso:

1. Los drivers deberán ser adaptados para usar semáforos o eventos en lugar de flags.
2. Los loops podrán usar OSSemPend() u OSPendMulti() en lugar de polls infinitos.

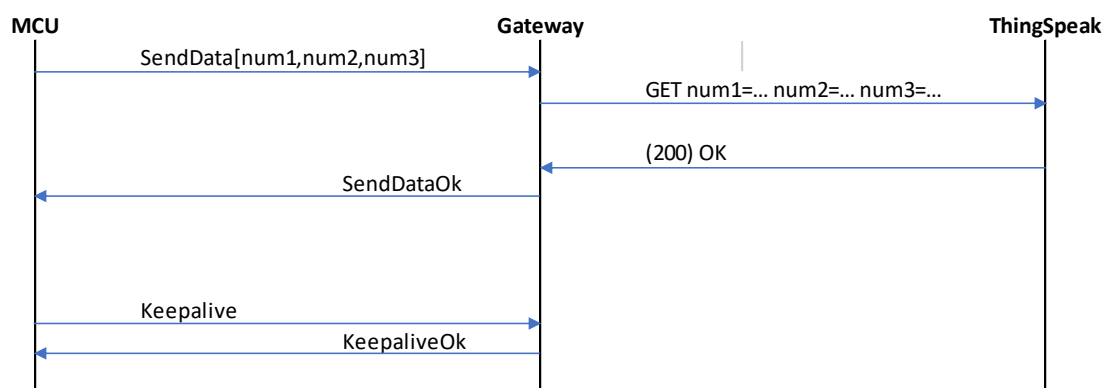
El Thread principal deberá comunicarse con el Thread de logging con un Message Queue u algún otro mecanismo de sincronización.

Gateway

Para instalar y ejecutar el Gateway:

- Instalar nodejs.
- En la consola, navegar al directorio "gateway"
- Ejecutar
npm install serialport
- Editar gateway.js, y configurar la API key y el puerto serie.
- Ejecutar
node gateway.js

El gateway recibirá comandos a través del puerto serie, a 1200bps, 8n1. Los comandos son **síncronicos** entre el microcontrolador y ThingSpeak. El gateway no permitirá encolar comandos. Una vez que se inicia SendData, el gateway no recibirá más comandos hasta emitir una respuesta (SendDataOk o SendDataFail).



Comandos y Respuestas

SendData	AA	55	C3	3C	07	01	pp	pp	qq	qq	rr	rr
SendDataOk	AA	55	C3	3C	01	81						
SendDataFail	AA	55	C3	3C	01	C1						

KeepAlive	AA	55	C3	3C	01	02						
KeepAliveOk	AA	55	C3	3C	01	82						

Descripción

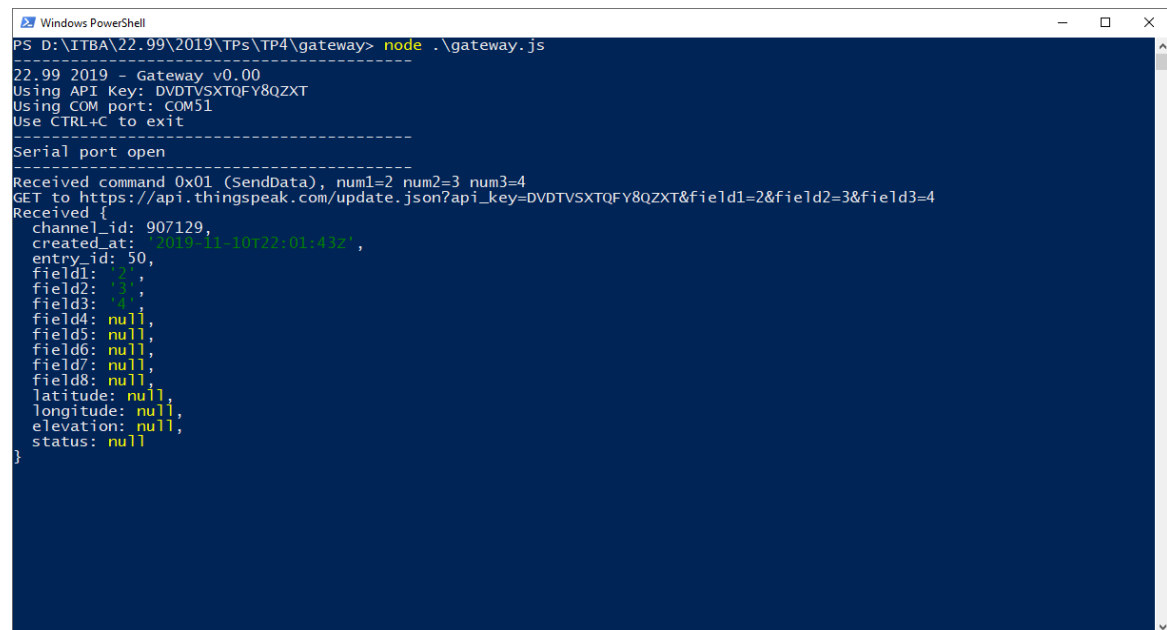
Header												
Length												
Command												
Answer												
Data:	pp	pp										
	qq	qq										
	rr	rr										

Cant. Piso 1, UINT16LE
 Cant. Piso 2, UINT16LE
 Cant. Piso 3, UINT16LE

Notas

- ThingSpeak acepta GETs separados por (al menos) 15 segundos. Este delay debe estar del lado del microcontrolador. Recordar que el gateway no puede encolar comandos.
- Se puede implementar un timeout para SendData (opcional).

- El keepalive (opcional) debería enviarse con un periodo razonable para detectar desconexiones.



```
PS D:\ITBA\22.99\2019\TPs\TP4\gateway> node .\gateway.js
22.99 2019 - Gateway v0.00
Using API Key: DVDTVSXTQFY8QZXT
Using COM port: COM51
Use CTRL+C to exit
-----
Serial port open
-----
Received command 0x01 (SendData), num1=2 num2=3 num3=4
GET to https://api.thingspeak.com/update.json?api_key=DVDTVSXTQFY8QZXT&field1=2&field2=3&field3=4
Received {
  channel_id: 907129,
  created_at: '2019-11-10T22:01:43Z',
  entry_id: 50,
  field1: '2',
  field2: '3',
  field3: '4',
  field4: null,
  field5: null,
  field6: null,
  field7: null,
  field8: null,
  latitude: null,
  longitude: null,
  elevation: null,
  status: null
}
```

Evaluación

Para la nota del trabajo contemplará en orden los siguientes puntos:

1. Funcionamiento obligatorio: Cumplimiento de los requerimientos obligatorios.
2. Implementación: Correcta arquitectura, uso correcto de los mecanismos de sincronización, y adecuada protección de recursos compartidos entre Threads y/o Thread-ISR.
3. Funcionamiento deseado: Robustez, uso intuitivo y versatilidad.
4. Presentación: Explicación de la estructura del programa, claridad en la explicación, etc.