



Sistema monitor de personas

César López Bermúdez

M.U. Ingeniería de telecomunicaciones





Índice

- Descripción del problema
- Trabajo previo (PoC)
- Productos alternativos
- Propuesta de solución
- Recursos empleados
- Descripción del NCG
- Descripción del DMP
- Planificación temporal
- Trabajos futuros





Descripción del problema(I)

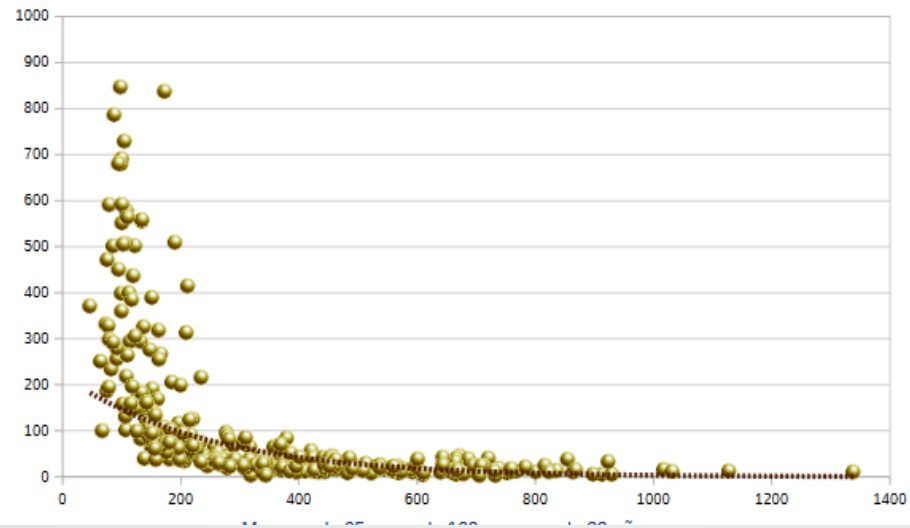
Problema identificado
en el rural gallego

- Envejecimiento de la población
- Éxodo masivo hacia capitales
 - Mejores oportunidades laborales
 - Aislamiento de los mayores
- Características de zonas rurales
 - Pueblos pequeños muy dispersos
 - Vías de comunicación antiguas
 - Zonas montañosas
 - Clima adverso
 - Servicios de asistencia públicos en progresiva reducción

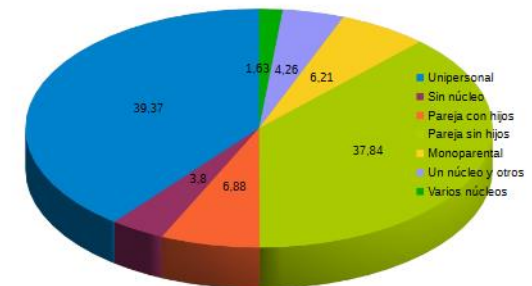


Descripción del problema(II)

Baja densidad población en sector rural



Elevado número de hogares unipersonales siendo mayor de 65 años





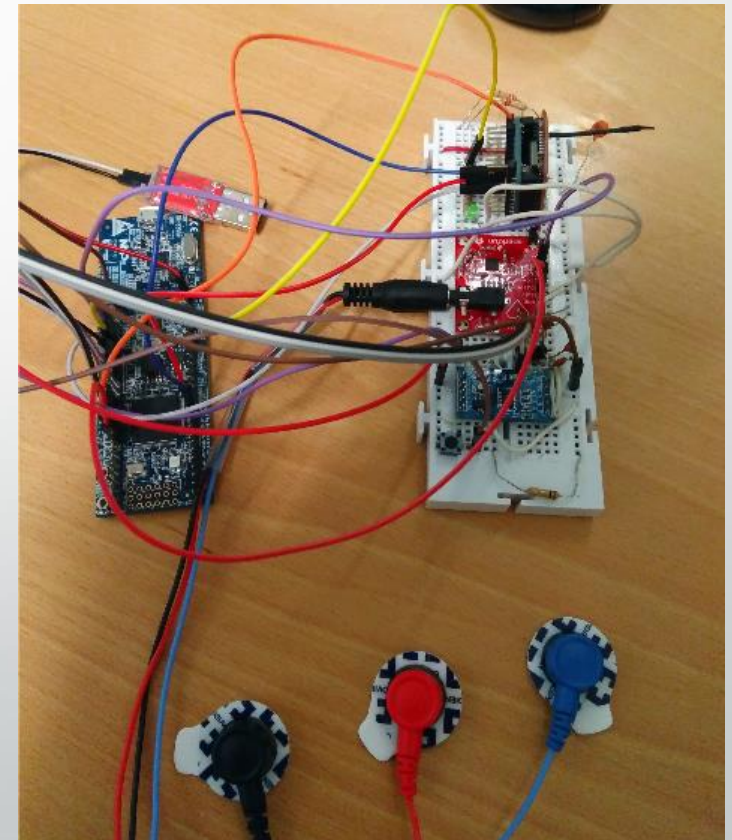


Trabajo previo (PoC)

Prueba de concepto trabajo de fin de grado

Basado en LPCxpreso de NXP

Lenguaje C y FreeRTOS





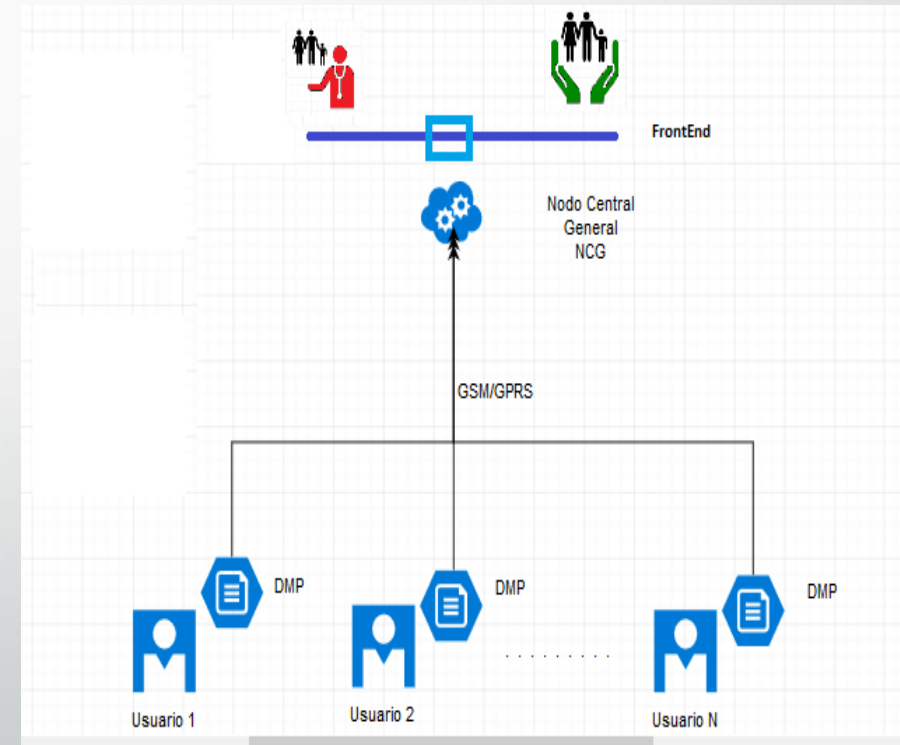
Propuesta de solución (I)

Red de DMPs

- Monitoriza sensores
- Conexión GSM/GPRS
- Envío a Cloud
- Recibe comandos desde NCG

Nodo Central General

- Almacén de datos de DMP
- Envío comandos a cada DMP
- Reconfiguración de DMP
- Procesamiento de datos

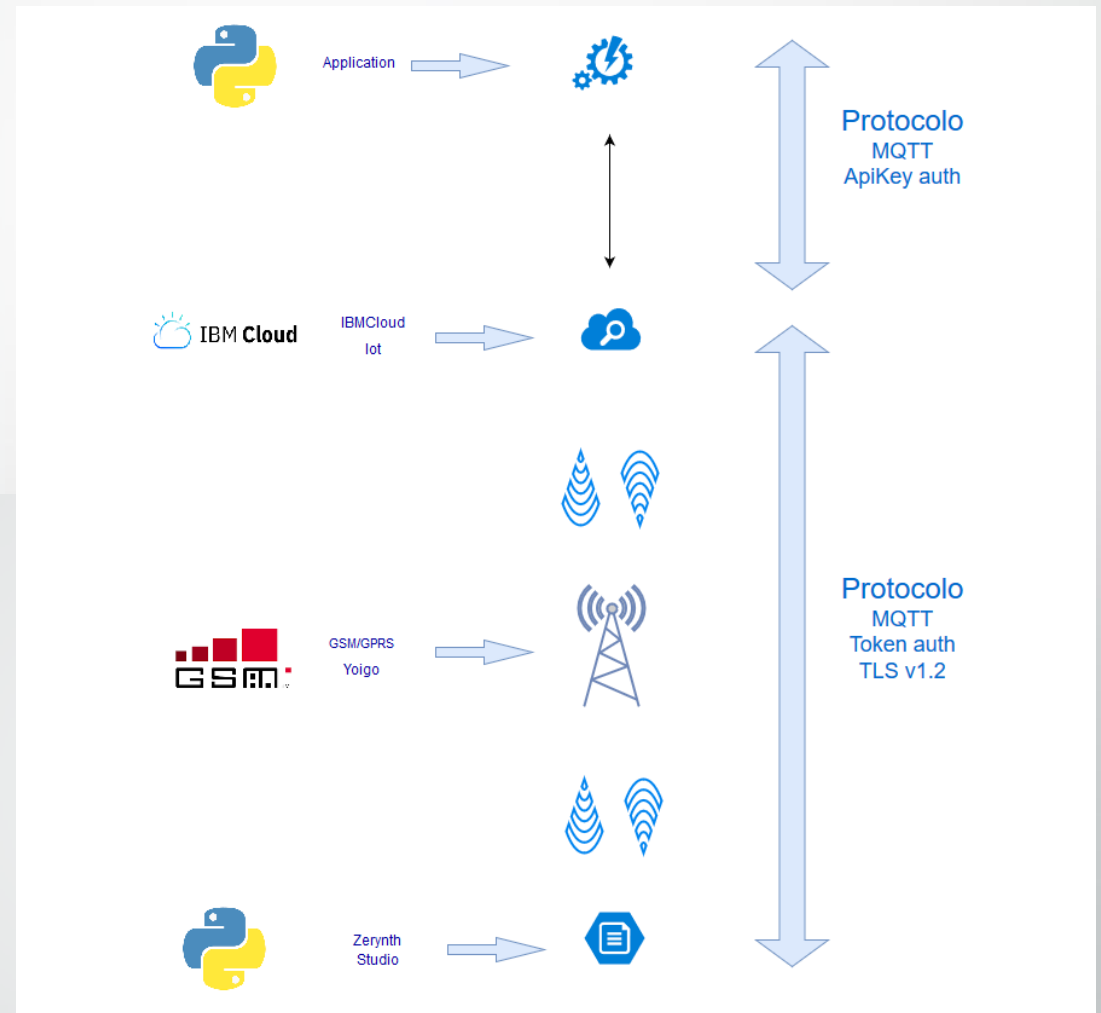




Propuesta de solución (II)

NCG

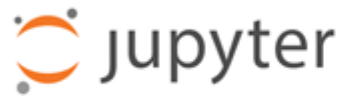
DMP





Recursos empleados

Software



Hardware





Descripción del NCG

- IBMCloud + Jupyter Notebook(Python)
- Conectividad MQTT vía Api Key

- Detección de eventos
- Envío de comandos



1.4 Suscripción a publicaciones DMPs

```
In [ ]:
# Inicializar aplicación cliente
try:
    appOptions = {"org": organization, "id": appId, "auth-method": appMethod, "auth-key": authKey, "auth-token": apptoken}
    appCli = ibmiotf.application.Client(appOptions)
except Exception as e:
    print(str(e))
    sys.exit()

appCli.connect()

# Suscripción a status (para registro del connect y disconnect)
appCli.deviceStatusCallback = myStatusCallback
appCli.subscribeToDeviceStatus()

# appCli.subscribeToDeviceEvents(deviceType, deviceId, "Initialized")
# appCli.deviceEventCallback = myAppEventCallback
```

1.5 Envío de notificaciones a DMP via MQTT

Se simula el envío de notificaciones al dispositivo DMP. Para ello se presuponen conocido el identificador del dispositivo cliente. Este será proporcionado por un agente externo a través de un FrontEnd determinado. A modo de ejemplo se simulan envíos de notificaciones médicas, expedidas por facultativos a través de un posíble aplicativo. Mediante una base de datos se hallará el Cliente del DMP objetivo a partir de los datos del paciente/usuario. Este paso se omite, considerando el Cliente fijado al definido para pruebas.

1.5.1 Envío de Aviso de toma de medicinas

```
In [ ]: cmdData1={"Medicina": "Adolonta"}
appCli.publishCommand(devTip, deviceId, "medi", "json", cmdData1)
```

1.5.2 Envío de Cita con médico

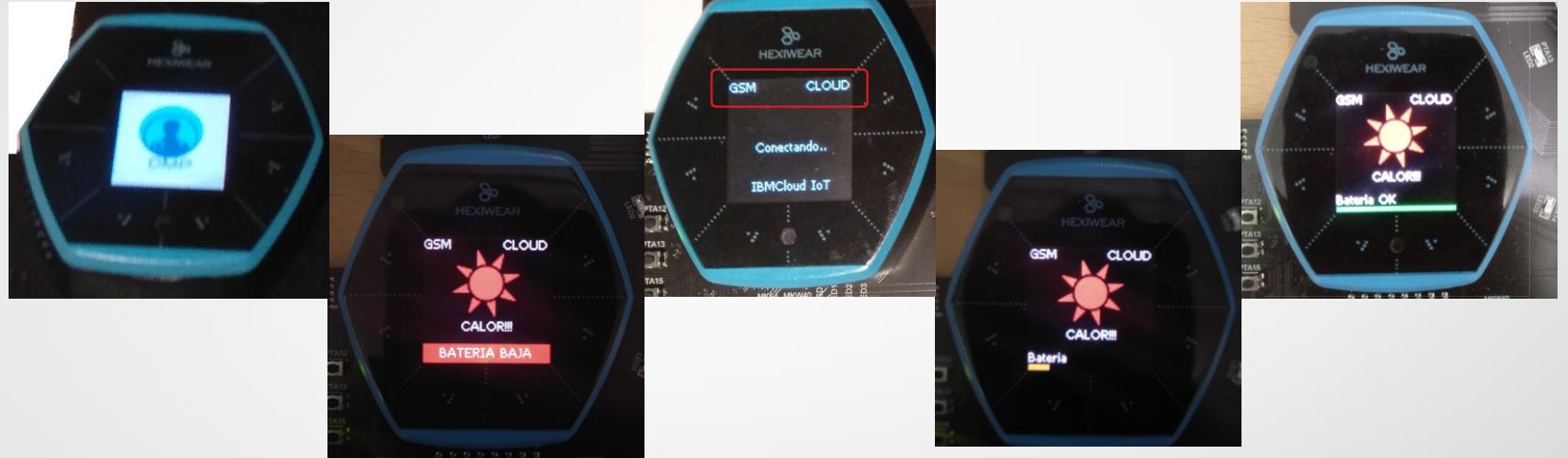
```
In [ ]: cmdData2={"Fecha": "12/10 15:00"}
appCli.publishCommand(devTip, deviceId, "cita", "json", cmdData2)
```

1.5.3 Envío de actividad física baja

```
In [ ]: cmdData3={"Movil": "Bajo"}
appCli.publishCommand(devTip, deviceId, "Bajo", "json", cmdData3)
```



Descripción del DMP



Características

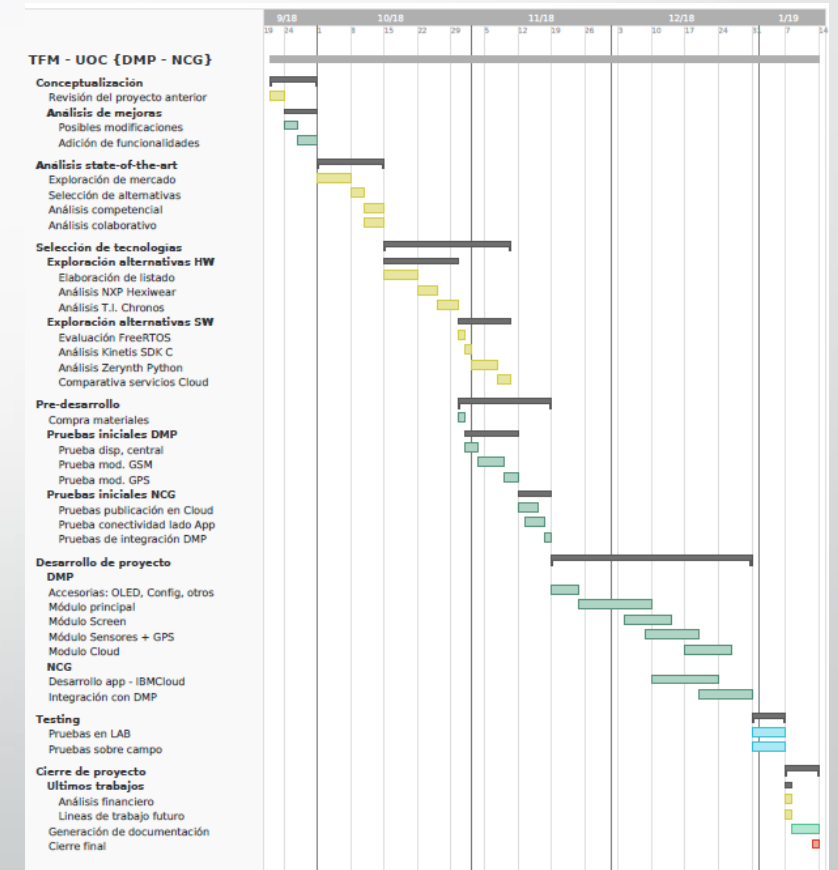
- Hexiwear + click GSM₄ + click NanoHornet GPS
- Alarma temperatura exterior
- Conectividad end-to-end vía MQTT
- Logo, iconos, mensajes sobre OLED





Planificación temporal

- Estructura partiendo de fechas de PECs
- Riesgos potenciales
 - Recepción de material
 - Detección de inviabilidad de componente
- Plazos temporales amplios para protección ante riesgos





Trabajos futuros

- Desarrollo completo del DMP
 - Desarrollo del software completo
 - Diseño PCB
 - Implementación de prototipo final
- Desarrollo de FrontEnd
- Machine-Learning para procesamiento BigData
- Agente conversacional
 - Twilio+SIP Trunk+Watson
 - Cierre de lazo del sistema





Gracias por su atención

César López Bermúdez

clopezberm@uoc.edu

