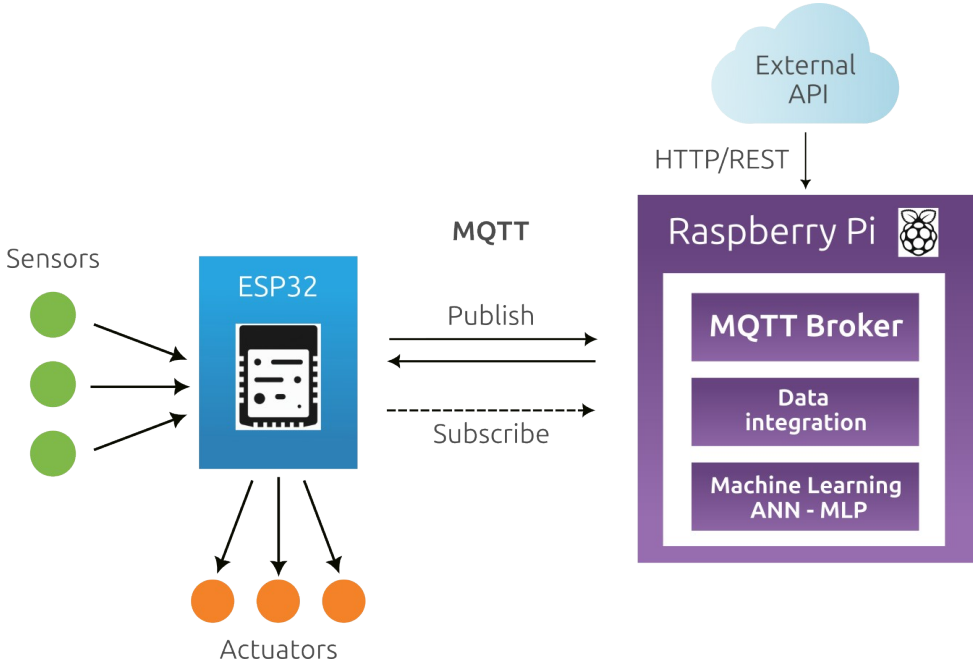
Integración del Agente de IA mediante Conexión ESP32 - Raspberry Pi

Entrega Segundo Proyecto

P. Sistemas Embebidos

La entrega del segundo proyecto consiste en la implementación funcional de la arquitectura completa del sistema.



Cada equipo deberá integrar la comunicación bidireccional entre el ESP32 y la Raspberry Pi mediante el protocolo MQTT, incorporando el flujo de datos desde los sensores hasta la Raspberry Pi, y el envío de decisiones hacia los actuadores. Durante la presentación, se espera que los equipos demuestren:

- El envío y recepción de datos entre el ESP32 y la RPi usando MQTT.
- La lectura de información complementaria desde una API externa.
- El procesamiento básico de los datos en la RPi mediante una red neuronal.
- La ejecución de decisiones en los actuadores, generadas a partir del procesamiento en la Raspberry Pi.

Además, cada grupo deberá explicar claramente el flujo de información dentro del sistema distribuido, destacando el papel de cada componente (sensores, ESP32, RPi, API y actuadores), y cómo estos se integran para materializar el comportamiento inteligente del agente.

Presentación

- Cada equipo contará con 15 minutos máximo para exponer.
- Todos los integrantes deben participar activamente en la presentación.

Requisitos de la entrega

Esta entrega corresponde a la continuación del primer proyecto. Por tanto, todo lo que haya quedado pendiente o no funcional en la primera versión deberá estar completamente operativo como condición de entrada para esta etapa.

Integración funcional ESP32 - RPi

- El ESP32 debe mantener el control directo de los sensores y actuadores de la maqueta.
- La comunicación bidireccional entre el ESP32 y la Raspberry Pi debe implementarse mediante el protocolo MQTT, evidenciando el envío de datos desde los sensores y la recepción de decisiones o comandos hacia los actuadores.
- El sistema debe mostrar indicadores visuales (LED) que reflejen el estado de conexión y transmisión de datos.

Procesamiento y conexión con API externa

- La Raspberry Pi debe recibir y procesar los datos provenientes del ESP32, e integrar la consulta a una API externa que proporcione información complementaria.
- La lógica de integración de datos y el módulo de Machine Learning (ANN-MLP) deben estar implementados en la RPi y evidenciar un proceso básico de inferencia o toma de decisiones.

Comunicación y control

- Las decisiones generadas en la Raspberry Pi deben enviarse al ESP32 mediante MQTT, para accionar los actuadores en tiempo real.
- La estructura de comunicación debe estar claramente documentada, incluyendo los tópicos MQTT y el flujo de mensajes (publicación y suscripción).

Presentación al profesor

La presentación debe incluir:

- Descripción general del sistema distribuido y su relación con el agente de IA propuesto.
- Explicación del flujo de información completo, desde la adquisición de datos hasta la acción del actuador.
- Demostración del intercambio de datos en tiempo real entre el ESP32 y la Raspberry Pi.

- Descripción de la organización de archivos y módulos tanto en el ESP32 (MicroPython) como en la Raspberry Pi (Python).

Demostración práctica

Evidenciar en vivo:

- La lectura de sensores, el envío de datos al broker MQTT, y la recepción de decisiones de la RPi.
- La ejecución de acciones en los actuadores basadas en los resultados del módulo de Machine Learning.
- El correcto funcionamiento de todos los componentes del sistema físico y lógico.

Recomendaciones para la red neuronal

La red neuronal debe implementarse en la Raspberry Pi, recibiendo datos del ESP32 y de una API externa. Su función es procesar las entradas, generar inferencias o clasificaciones simples y enviar decisiones al ESP32 para controlar los actuadores.

A continuación, se presentan recomendaciones para guiar el desarrollo y entrenamiento de la red:

Estructura del dataset

- El conjunto de datos debe contener entradas representativas de los sensores (por ejemplo: temperatura, humedad, luz, ruido, etc.) y las salidas esperadas (por ejemplo: nivel de ventilación, iluminación, alerta, etc.).
- Registrar los datos directamente desde la maqueta o usar un conjunto inicial generado por simulación.
- Organizar el dataset en formato CSV para leerlo con NumPy y/o Pandas, con las columnas etiquetadas con los nombres de las entradas y salidas.
- Asegurarse de que las entradas estén normalizadas o escaladas entre 0 y 1 antes del entrenamiento.
- Separar el dataset en dos conjuntos: 70 % entrenamiento / 30 % validación, o 80 % entrenamiento / 20 % validación.

Arquitectura de la red neuronal

- Utilizar una red neuronal feed-forward (Multilayer Perceptron, MLP).
- Configuración básica recomendada:
  - Capa de entrada: igual al número de variables de los sensores.
  - 1 o 2 capas ocultas: entre 8 y 32 neuronas cada una.
  - Función de activación: relu para capas ocultas, sigmoid o softmax para la salida (según sea binaria o multiclase).

- Tasa de aprendizaje inicial: 0.001 a 0.01.
- Épocas de entrenamiento: entre 100 y 300 (ajustar según el error).
- Optimizador: adam.
- Función de pérdida: mse (regresión) o categorical_crossentropy (clasificación).

Pruebas y validación

- Observar la pérdida y exactitud (loss / accuracy) en entrenamiento y validación.
- Verificar que la red no esté sobreajustada (overfitting): si el error de validación aumenta mientras el de entrenamiento baja, detener el entrenamiento.
- Realizar pruebas en tiempo real:
  - Enviar valores de sensores desde el ESP32 a la RPi.
  - Ejecutar el modelo cargado en la RPi con tensorflow lite.
  - Verificar que la salida cambie coherentemente según los valores de entrada.

Consejos prácticos

- Comenzar con una red pequeña y pocos datos, y escalar gradualmente.
- Asegurar que las unidades de medida sean consistentes entre los datos históricos y los valores en tiempo real.
- Visualizar los resultados con gráficos simples (por ejemplo, predicción vs real) para verificar comportamiento.
- Registrar ejemplos de entrada/salida para documentar las pruebas.
- No buscar precisión perfecta.

Aspectos clave de la evaluación

- Funcionamiento real del sistema distribuido (sensores, actuadores, comunicación MQTT y procesamiento en la RPi).
- Claridad técnica en la explicación de la arquitectura, los componentes y el flujo de datos.
- Participación equilibrada de todos los integrantes del equipo durante la presentación.
- Integración coherente entre los datos de los sensores, la información externa (API), el modelo de ML y las acciones de los actuadores.
- Calidad general de la presentación, organización, contenido y demostración práctica.