Raspberry Pi + Arduino UNO + Fast API

Descripción

Este proyecto fue desarrollado con el objetivo de demostrar el potencial del hardware de bajo costo, la integración entre sistemas embebidos y software moderno, y servir como punto de partida para que estudiantes, makers y desarrolladores se animen a experimentar, aprender e inventar soluciones reales.

La solución integra un Arduino UNO, una Raspberry Pi 3 Model B y una API REST desarrollada con FastAPI, combinando electrónica, programación embebida y desarrollo backend profesional en una arquitectura clara, modular y extensible.

El enfoque del proyecto no es únicamente funcional, sino también didáctico y arquitectónico, siguiendo buenas prácticas de desarrollo, separación de responsabilidades y escalabilidad.

Arquitectura y concepto

La arquitectura se basa en una separación clara de roles:

- **Arduino UNO**
  Se encarga exclusivamente del control de hardware y sensores:
  - Lectura de sensores
  - Control de actuadores
  - Comunicación serial estructurada
- **Raspberry Pi**
  Funciona como nodo de procesamiento e integración:
  - Ejecuta la API REST
  - Gestiona la comunicación con Arduino
  - Expone servicios al exterior
- **FastAPI (Python)**
  Actúa como capa de acceso:
  - Endpoints RESTful
  - Manejo de errores centralizado
  - CORS habilitado para frontend
  - Arquitectura basada en routers y services

Este diseño refleja una arquitectura similar a sistemas IoT reales, donde los dispositivos de bajo nivel no se exponen directamente a la red, sino que se integran a través de una capa intermedia segura y controlada.

Sensores y actuadores utilizados

LED (Actuador)

Un LED controlado por GPIO permite verificar:

- Control remoto de hardware
- Respuesta inmediata desde la API
- Funcionamiento end-to-end (API → Serial → Arduino → Hardware)

Sensor DHT11 (Temperatura y Humedad)

El DHT11 permite medir:

- Temperatura ambiente
- Humedad relativa

Este sensor es ampliamente utilizado en proyectos IoT y domótica, y demuestra cómo exponer métricas físicas reales mediante una API HTTP.

Sensor ultrasónico HC-SR04

El HC-SR04 permite medir distancia mediante ultrasonido, útil en:

- Detección de objetos
- Sistemas de proximidad
- Automatización y robótica básica

La lectura de distancia es procesada en Arduino y expuesta por la API.

Importancia de la integración Arduino – Raspberry Pi – FastAPI

Este proyecto pone en evidencia un punto clave:
la integración es tan importante como el hardware o el software por separado.

- Arduino es excelente para tiempo real y control directo de sensores.
- Raspberry Pi permite ejecutar software complejo, redes y servicios.
- FastAPI aporta estándares modernos de integración, escalabilidad y consumo por terceros.

La combinación de estos tres elementos abre la puerta a:

- Sistemas IoT reales
- Prototipos industriales
- Automatización
- Integración con frontends web o móviles
- Exposición de datos a la nube

Backend y APIs: una habilidad esencial hoy

En la actualidad, todo desarrollador backend debería ser capaz de diseñar e implementar APIs, incluso cuando el proyecto involucra hardware.

Exponer dispositivos físicos a través de una API:

- Permite desacoplar sistemas
- Facilita la escalabilidad
- Habilita múltiples consumidores (web, móvil, otros servicios)
- Alinea proyectos embebidos con arquitecturas modernas

Este proyecto demuestra cómo un sistema embebido puede convertirse en un servicio accesible, seguro y reutilizable.

Proyección hacia el mundo IoT

La arquitectura presentada es directamente extrapolable a escenarios IoT reales:

- Sensores distribuidos
- Gateways
- APIs centralizadas
- Integración con dashboards, alertas o machine learning

A partir de esta base, es posible evolucionar hacia:

- Autenticación y autorización
- Persistencia de datos
- Comunicación con la nube
- Automatización avanzada
- Control remoto en tiempo real

Objetivo

El objetivo principal de este proyecto es motivar:

- A aprender
- A experimentar
- A romper la barrera entre hardware y software
- A construir soluciones propias

No se trata solo de encender un LED o leer un sensor, **sino de entender cómo todo se conecta,** cómo se diseña una solución completa y cómo pequeñas ideas pueden escalar a sistemas reales.

Este proyecto demuestra que, con herramientas accesibles, buenas prácticas y curiosidad, es posible construir sistemas potentes, profesionales y listos para el mundo real.

Componentes involucrados

- Arduino UNO
- Protoboard
- LED estándar
- Resistencia **220 Ω**
- Módulo con Sensor **DHT11** (temperatura y humedad)
- Sensor ultrasónico **HC-SR04**
- Cables Dupont macho-macho

Conexión del LED con resistencia (la resistencia limita la corriente ~15–20 mA)

Componentes

- LED tiene:
    - **Ánodo** (patita larga)
    - **Cátodo** (patita corta)

Conexión

1. Pin digital 8 del Arduino → Resistencia 220 Ω
2. Resistencia → Ánodo del LED
3. Cátodo del LED → GND de la protoboard

Conexión del módulo con sensor DHT11 (módulo chino, más común)

El DHT11 módulo más común trae:

- El sensor DHT11
- Una resistencia pull-up SMD (muy pequeña)
- A veces un LED indicador
- Tres pines claramente rotulados en la serigrafía de la placa

Pines del módulo mirándolo de frente: letra S a la izquierda, nada en el medio, signo – a la derecha.

Pin	Significado	Función eléctrica
S	Signal	Datos digitales
(sin marcar)	VCC	+5 V
-	Ground	GND

Conectar así:

Pin módulo DHT11	Conectar a
S (izquierda)	Pin digital D7 del Arduino
Centro (VCC)	5V del Arduino
– (derecha)	GND del Arduino

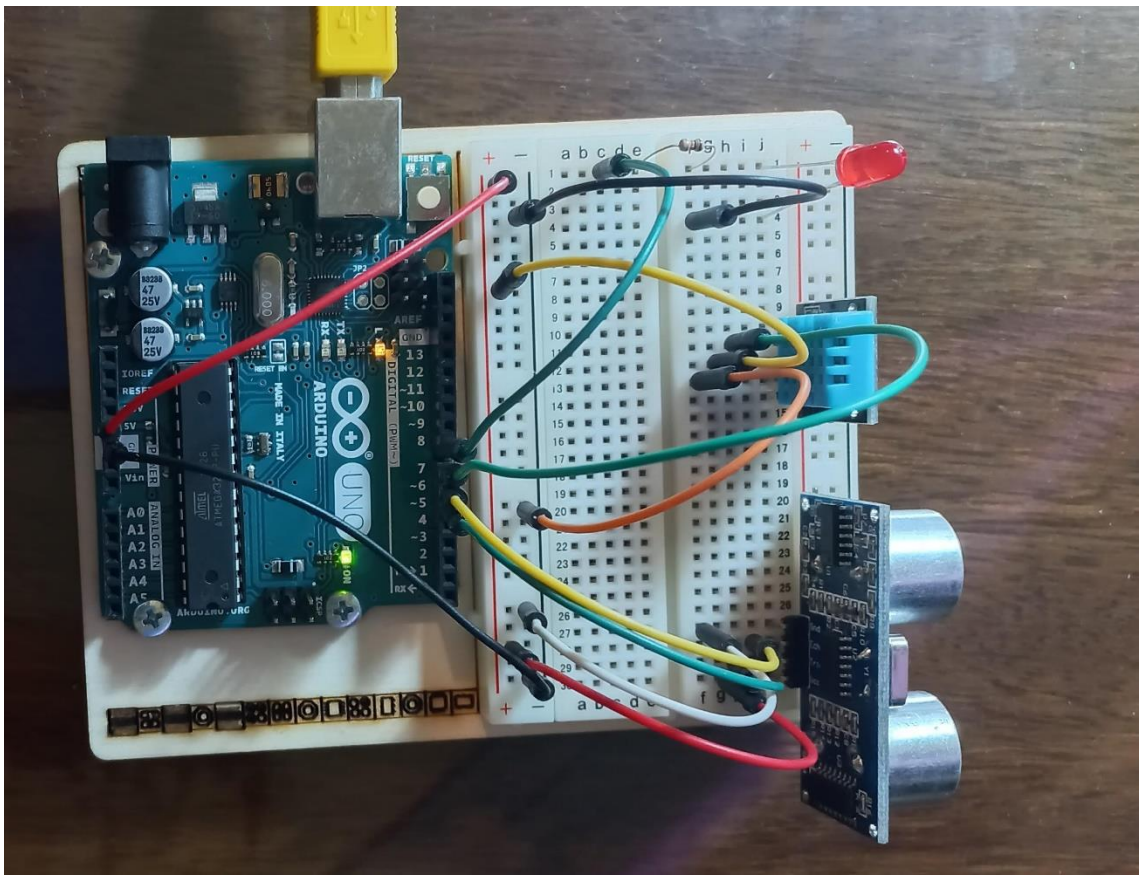
Conexión del sensor ultrasónico HC-SR04

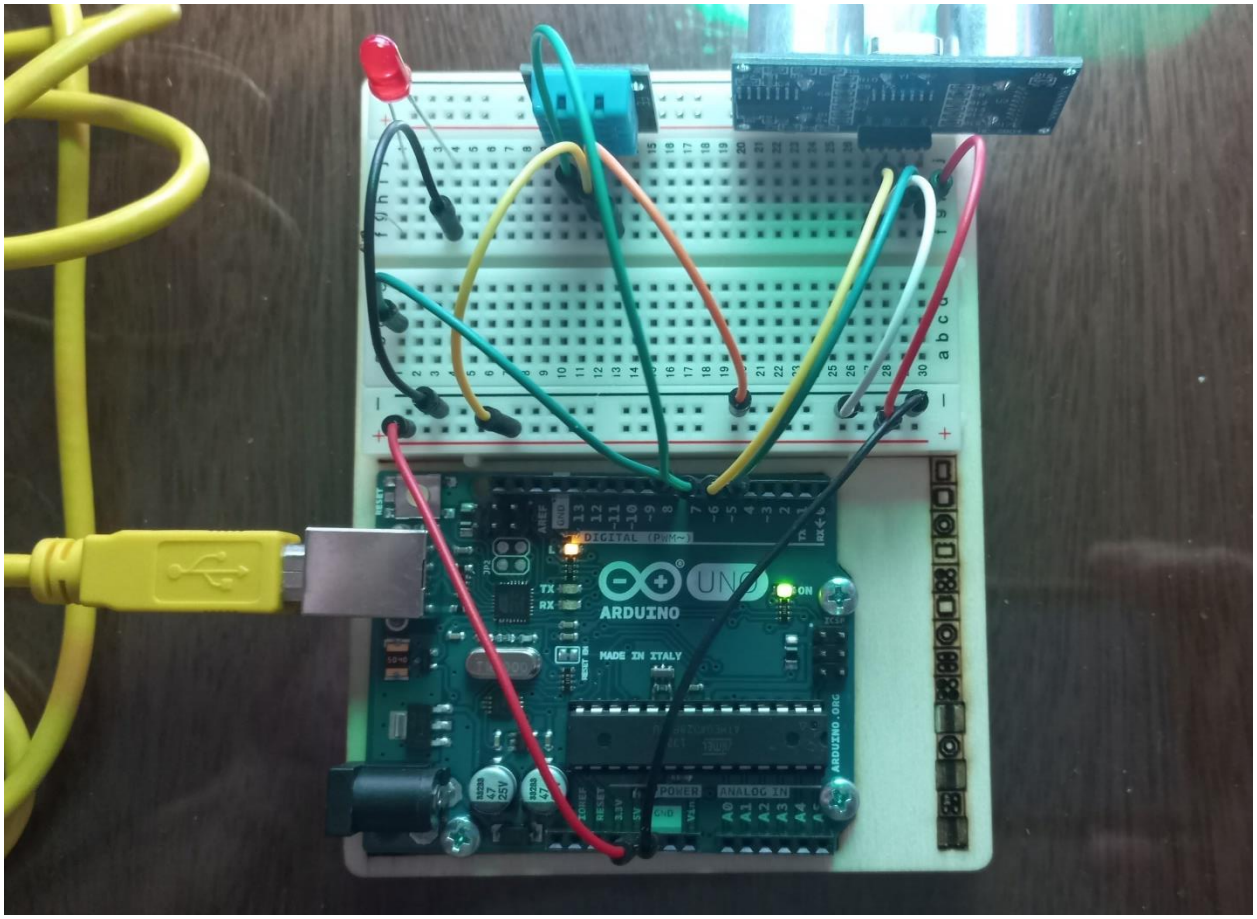
Pines del HC-SR04: VCC ─ TRIG ─ ECHO ─ GND (identificados en la placa)

Conexión a Arduino:

- VCC → 5V protoboard
- GND → GND protoboard
- TRIG → Pin digital 5 del Arduino
- ECHO → Pin digital 6 del Arduino

Importante:
El ECHO entrega 5V, pero como va al Arduino (5V lógico), NO requiere divisor.


ARDUINO
UNO
MADE IN ITALY


ARDUINO
UNO
MADE IN ITALY

Firmware de prueba (Arduino IDE, ver dentro del directorio Arduino_UNO test.ino)

Este sketch debe permitir confirmar, que:

1. El LED responde a comandos de software
2. El DHT11 entrega temperatura y humedad válidas
3. El HC-SR04 mide distancia correctamente
4. La comunicación Serial USB funciona

Librerías necesarias (Arduino IDE):

- DHT sensor library"
- Adafruit Unified Sensor (dependencia que ya viene instalada)

Pruebas básicas

Abrir el Monitor Serial (9600 baudios) y verificar:

LED

- Se enciende y apaga cada ciclo
  ✓ Confirmación de pin y resistencia correctos

DHT11

- Valores razonables:
  - Temperatura: ~15–35 °C
  - Humedad: ~30–80 %
    ✓ Confirmación de señal, pull-up y alimentación

HC-SR04

- Distancia coherente al mover la mano
  ✓ Confirmación de TRIG/ECHO y temporización

```
----------------------------
LED: ENCENDIENDO
LED: APAGANDO
DHT11: Humedad = 40.00 % | Temperatura = 25.00 C
HC-SR04: Distancia = 15.21 cm
----------------------------
LED: ENCENDIENDO
LED: APAGANDO
DHT11: Humedad = 40.00 % | Temperatura = 26.00 C
HC-SR04: Distancia = 15.21 cm
----------------------------
LED: ENCENDIENDO
LED: APAGANDO
DHT11: Humedad = 40.00 % | Temperatura = 25.00 C
HC-SR04: Distancia = 15.23 cm
----------------------------
```

Validado lo anterior, avanzamos a diseñar protocolo de comandos por Serial, por ejemplo:

LED_ON

LED_OFF

READ_DHT

READ_DIST

Respuestas esperadas:

OK

TEMP:23.5, HUM:54.2

DIST:18.3

ERROR

Luego FastAPI consumirá estos comandos como servicio.

Implementación final, ver directorio Arduino_UNO, implementación_final.ino

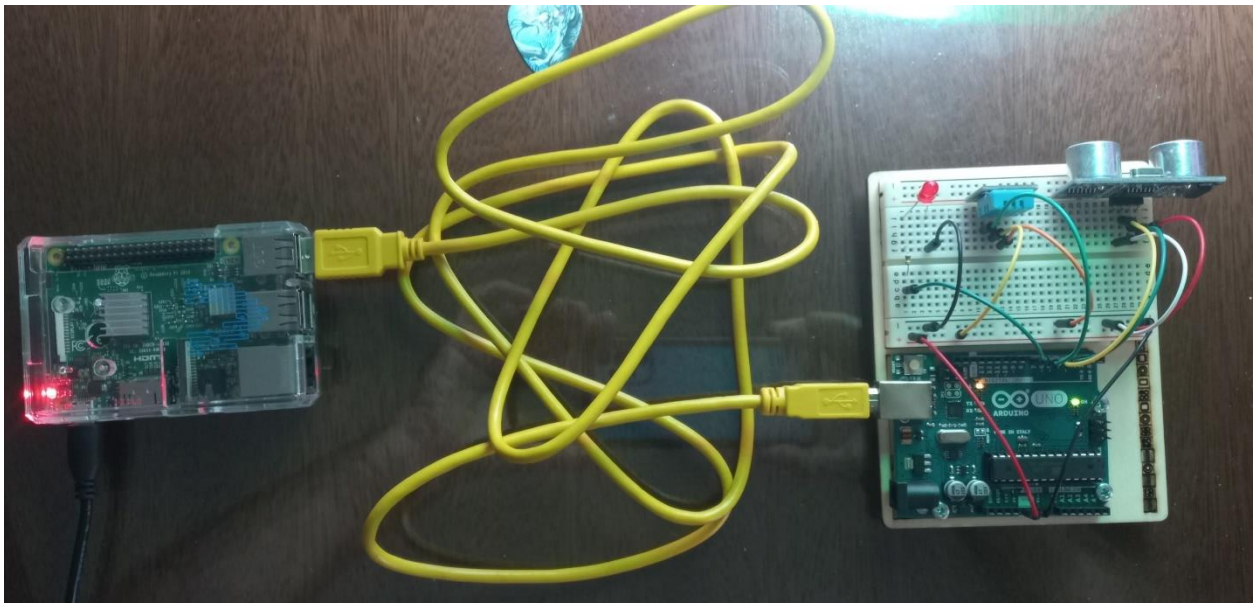
Pruebas para este firmware:

Subir el sketch

- Compila
- Sube
- Abrir Serial Monitor (Baudrate: 9600, Line ending: Newline)

Probar los commandos manualmente: LED_ON, LED_OFF, READ_DHT, READ_DIST

Conectar el Arduino por USB a la Raspberry Pi



Verificación inicial en la Raspberry Pi:

Cuando la Raspberry esté arriba:

ls /dev/tty*

Debemos ver algo como:

/dev/ttyUSB0 o /dev/ttyACM0

(ese será el puerto del Arduino)

Crear el directorio para el proyecto y el entorno virtual:

python -m venv env

Activar el entorno e instalar dependencias y framework Fast API para Python:

pip install pyserial

pip install "fastapi[standard]"

Antes de comenzar con el desarrollo de la API, creamos un script de prueba simple (test_serial.py)

test_serial.py

```python
import serial
import time

ser = serial.Serial('/dev/ttyACM0', 9600, timeout=2)
time.sleep(2)

ser.write(b'LED_ON\n')
print(ser.readline().decode().strip())

ser.write(b'READ_DHT\n')
print(ser.readline().decode().strip())

ser.write(b'READ_DIST\n')
print(ser.readline().decode().strip())

ser.write(b'LED_OFF\n')
print(ser.readline().decode().strip())

ser.close()
```

Ejecutar: python test_serial.py

Si vemos respuestas correctas: comunicación Arduino ↔ Raspberry OK

Si hay error de permisos:

(env) monitor@raspberry:/opt/arduino_api $ ls -l /dev/ttyACM0

crw-rw---- 1 **root dialout** 166, 0 Jan 11 22:27 /dev/ttyACM0

Solución para el usuario que estemos usando: sudo usermod -aG dialout monitor (reiniciar Raspberry Pi)

Prueba nuevamente:
(env) monitor@raspberry:/opt/arduino_api $ python test_serial.py

READY

OK

TEMP:25.00, HUM:46.00

DIST: 7.19

Implementación endpoints de la API

/led/on enciende el LED
/led/off apaga el LED
/sensores/dht <devuelve valores reales>
/sensores/distancia <devuelve distancia>

- Puerto serial abierto una sola vez

Pruebas realizadas en un frontend HTML/CSS/JS

Desarrolla: Matías Santos