IoT con PYNQ

Repositorio: `github.com/ayman-krd/pynq-iot-riskengine`

Nadir Mohamed Tanjaba Ayman Karroud Abdeselam

20 de enero de 2026

Sistemas Electrónicos Integrados
Máster Universitario en Ingeniería de Telecomunicación
Universidad de Granada

Índice

Introducción y motivación

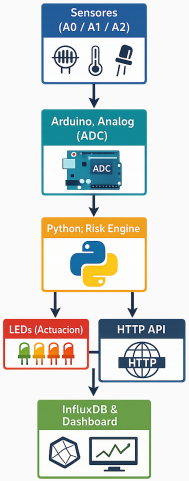
Contexto, propuesta y objetivo

- Contexto: Crecimiento de sistemas IoT para monitorizar entornos y enviar datos a la nube.
- Propuesta: Motor de riesgo sencillo a partir de varios sensores.
- Objetivo: Diseñar e implementar un sistema IoT en la PYNQ-Z2 que lea sensores, calcule un nivel de riesgo y lo muestre de manera local como remota.
- Visualización del riesgo:
  - Local: LEDs on-board.
  - Remota: InfluxDB.

Arquitectura del sistema

Arquitectura general

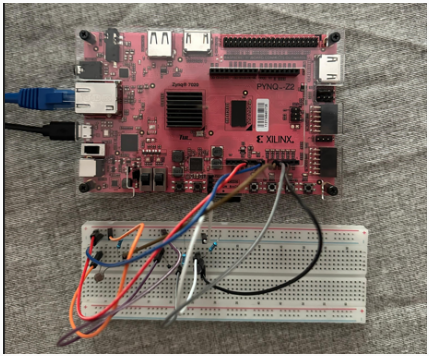
- **Capa física**:
  - PYNQ-Z2.
  - Sensores (Arduino): A0 (LDR), A1 (NTC), A2 (tilt/botón).
  - Actuadores: LEDs on-board.
- **Capa de proceso**:
  - Python en ARM: lectura y cálculo de riesgo.
- **Capa IoT**:
  - Envío a InfluxDB 2.x en el PC.
  - Dashboards y reglas de alerta.



Diseño hardware

Montaje de sensores

- Conexión al conector Arduino (3.3 V lógicos).
- **A0 – Luz (LDR)**: divisor LDR + 10 kΩ.
- **A1 – Temperatura (NTC)**: divisor NTC + 10 kΩ.
- **A2 – Evento (tilt/botón)**: pull-up 10 kΩ y switch a GND.
- LEDs on-board: `base.leds[0..3]`.



Uso del overlay base de PYNQ

- Carga del overlay:

```
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")
```

- Lectura analógica:

```
from pynq.lib.arduino import Arduino_Analog
analog = Arduino_Analog(base.ARDUINO, [0, 1, 2])
```

- Control de LEDs:

```
for led in base.leds:
    led.off()
base.leds[0].on()
```

Diseño software

Lógica de riesgo en Python

- Bucle principal (`main.py`):
  1. Leer A0, A1, A2 con `analog.read_raw()`.
  2. Evaluar alertas: luz baja, temperatura alta, evento activo.
  3. Calcular:
  $$risk\_level = alert\_light + alert\_temp + alert\_event$$
  4. Actualizar LEDs y enviar muestra a InfluxDB.
- Umbrales ajustados experimentalmente.

Integración con InfluxDB

- Envío con `requests` usando *line protocol*:

```
pynq_sensors,source=pynq_z2 light_raw=...,temp_raw=...,event_raw=...,risk_level
    =...,alert_light=...
```

- Configuración: URL, organización, bucket y token.
- En el PC: dashboards para `light_raw`, `temp_raw`, `event_raw` y `risk_level`.

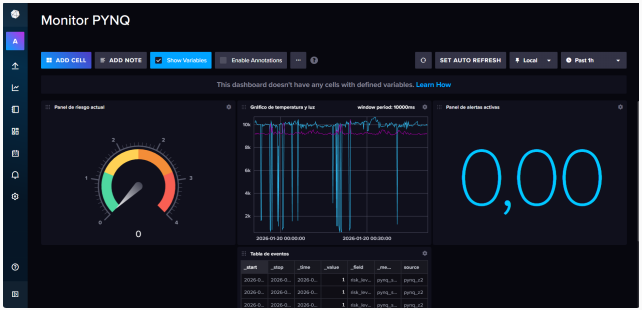
Resultados y demostración

Resultados experimentales

- En reposo: lecturas estables, `risk_level = 0`, LEDs apagados.
- Pruebas:
  - Tapar LDR $\Rightarrow$ riesgo 1 (LED0).
  - Calentar NTC $\Rightarrow$ riesgo 1–2.
  - Activar tilt/botón $\Rightarrow$ riesgo adicional.
  - Combinación $\Rightarrow$ riesgo 3 (LED0–LED2).
- En InfluxDB: trazas sincronizadas y checks/alertas por umbral.

Demostración

1. Lanzar `main.py` en la PYNQ-Z2.
2. Mostrar valores y riesgo por consola.
3. Demostración física (LDR/NTC/tilt) y LEDs.
4. Visualizar gráficas en InfluxDB y comentar alertas.



Conclusiones

Conclusiones y futuro

- Conclusiones:
  - PYNQ-Z2 acelera prototipado IoT (Python + HW reconfigurable).
  - Flujo completo: sensores $\rightarrow$ riesgo $\rightarrow$ actuadores $\rightarrow$ BD temporal.
- Futuro:
  - Migrar riesgo a IP (HLS/Vitis).
  - Añadir sensores (PIR, humedad, etc.).
  - Web de visualización en la propia PYNQ.

Bibliografía

Bibliografía

- PYNQ Project (Xilinx/AMD).
  PYNQ Documentation.
  https://pynq.readthedocs.io/
- TUL Corporation.
  PYNQ-Z2 Board (producto y documentación).
  https://www.tul.com.tw/ProductsPYNQ-Z2.html
- InfluxData.
  InfluxDB 2.x Documentation.
  https://docs.influxdata.com/influxdb/v2/
- InfluxData.
  Write Data: HTTP API (InfluxDB 2.x).
  https://docs.influxdata.com/influxdb/v2/api/#operation/PostWrite

¿Preguntas?