

ZEBRABIDE MULTAK

Sandra, Peru, Edurne, Naroa Z, Naroa M

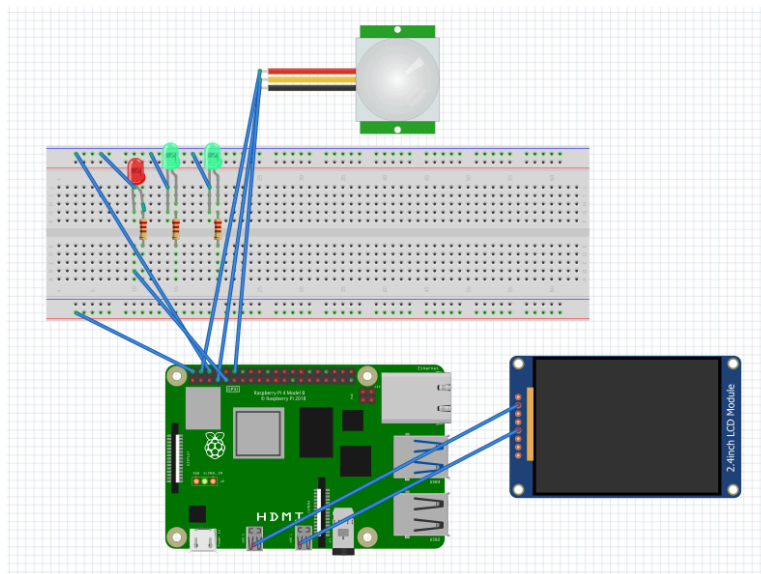
Zertan datza proiektua

Proiektua zebrabideak Raspberry Pi 4 batera konektatutako sentsoreen bidez gainbegiratzeko sistema adimendun bat garatze du helburu. Helburu nagusia bide-segurtasuna hobetzea eta oinezkoek trafiko-arauak bete ditzaten sustatzea da. Horretarako, sistema gai da automatikoki detektatzeko pertsona batek zebrabide bat noiz gurutzatzen duen eta modu egokian gurutzatzen duen ala ez, hau da, argi-seinaleztapena errespetatzen duen (oinezkoentzako semaforo berdea) edo gorritz gurutzatzen duen, eta hori arau-haustea da.

Sistemaren nukleoa Raspberry Pi 4 batera konektatuta dagoen mugimendu-sentsore batean oinarritzen da. Sentsore horrek aukera ematen du zebra-bidearen eremuan pertsona bat dagoela identifikatzeko eta gurutzaketa gertatzen den une zehatza detektatzeko. Raspberry Pi-k prozesatzeko unitate zentral gisa jarduten du, sentsorearen informazioa bilduz eta semaforoaren egungo egoerarekin kontrastatuz (gorritz edo berdez dagoen oinezkoentzat).

Pertsona batek zebrabidea gurutzatzen duenean semaforoa gorri dagoela, sistemak arau-hauste gisa erregistratzen du. Kasu horretan, gertaerari lotutako isun-erregistro bat sortzen da automatikoki. Datu horiek guztiak (gurutzatzeko unea, semaforoaren egoera, eta arau-hausterik egon den ala ez) datu-base batean gordetzen dira, ondoren aztertu, kontsultatu edo zehapenak aplikatzeko.

Proiektuaren kableen irudi digitalizatua:



Zebrabide kodigoa

Python-ko prograna sortu dugu, gorriz gurutzatzen dituzten oinezkoen guztira kalkulatzeko eta datu horiek nodeRed-ra bidaltzeko.

```
import requests
from gpiozero import LED, MotionSensor
from time import sleep, time

pir = MotionSensor(18) # PIR en GPIO 18
led_verde = LED(17)    # LED verde en GPIO 17
led_rojo = LED(27)     # LED rojo en GPIO 27
counter = 0

tiempo_rojo = 5
tiempo_verde = 5

url = "http://formacioniot2025.devlon.es/grupo4"

while True:

    led_rojo.on()
    led_verde.off()
    print("LED rojo encendido - Esperando detección de movimiento")

    movimiento_detectado = False
    tiempo_inicio = time()

    while time() - tiempo_inicio < tiempo_rojo:
        if pir.motion_detected:
            print("¡Movimiento detectado!")
            movimiento_detectado = True
            counter += 1
            print(f"Detección número: {counter}")

            data = {
                "detected": 1,
                "detections": counter,
                "time": time()
            }

            try:
                response = requests.post(url, json=data)
                if response.status_code == 200:
                    print("Datos enviados correctamente a Node-RED")
                else:
                    print(f"Error al enviar los datos. Código de estado: {response.status_code}")
            except Exception as e:
```

```

    print(f"Error al hacer la solicitud POST: {e}")

    sleep(0.5)

    led_rojo.off()
    led_verde.on()
    print("LED verde encendido - No se detecta movimiento")

    tiempo_inicio_verde = time()

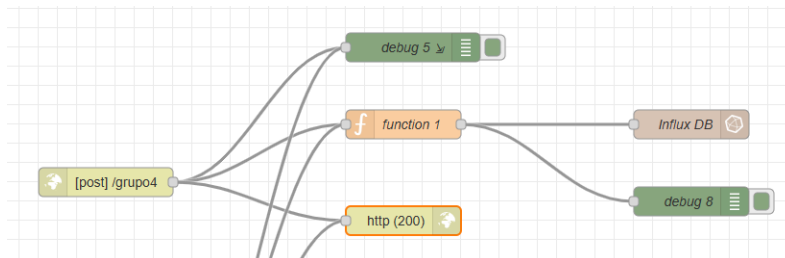
    while time() - tiempo_inicio_verde < tiempo_verde:
        sleep(0.5)

    print("Ciclo completo, reiniciando...")

```

Eraikitze-prosezua

Oinarri-fluxua hartuta, gure fluxu propioa eraiki dugu datuak jaso eta influxDb(e)ra bidaltzeko.



- **POST-a:** raspberry pi-ak HTTP post web-eskaera eskaintza baten bitartez bidaltzen ditu datuak nodeRed-ra, beraien arteko komunikazioa sortuz.
- **function:** post-ak raspberry pi tik JSON formatuan bidalitako datuak eskuratu eta influx DB-ak datu horiekin lan egiteko behar duen formatura egokitzen ditu.

```

var payload = msg.payload;

var tags = {
  fuente: "sensorPIR"
};

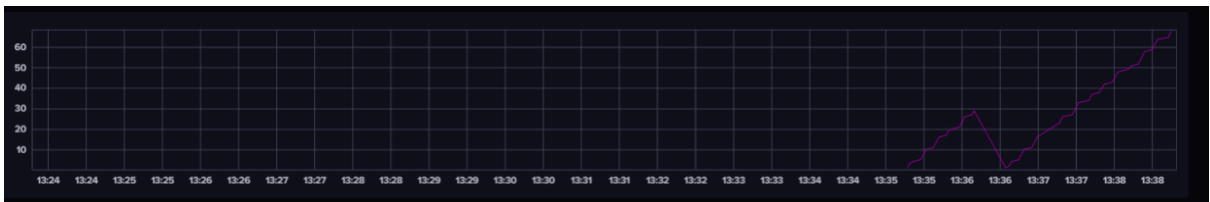
var fields = {
  detections: Number(payload.detections),
  detected: payload.detected ? 1 : 0
};

```

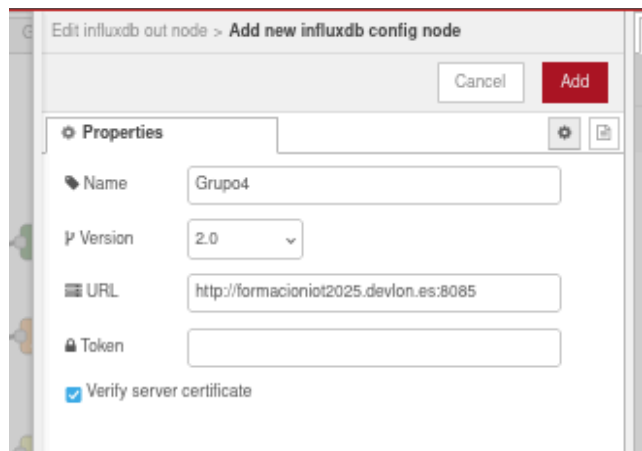
```
let time = new Date().toLocaleString();
```

```
msg.payload = {  
  measurement: "detecciones",  
  detections: Number(payload.detections),  
  detected: payload.detected ? 1 : 0,  
  String : time  
};  
  
return msg;
```

- **influx DB:** Makinak semaforoa gorritz dagoenean datuak bidaltzen ditu eta datu horiek influx db-an gordetzen dira, zebraidea gorritz gurutzatu duten pertsonen kopurua eta ordu zehatzekin.



- **Datubarekin konexioa:** DB bucket izena + URL + token



The screenshot shows a dialog box titled 'Edit influxdb out node > Add new influxdb config node'. It has 'Cancel' and 'Add' buttons. Under the 'Properties' tab, the following fields are visible: 'Name' with the value 'Grupo4', 'Version' with a dropdown set to '2.0', 'URL' with the value 'http://formacioniot2025.devlon.es:8085', and 'Token' which is empty. There is also a checked checkbox for 'Verify server certificate'.

- **debug 8:** Debugaren blokeari esker, funtzioak datuak behar bezala prozesatzen dituela ziurta dezakegu.

```
9/5/2025, 13:45:38  nodo: debug 5
msg: Object
  ▼ object
    _msgid: "04e036e1a8d29fa0"
    ▼ payload: object
      detected: 1
      detections: 7
      time: 1746791138.5397277
    ▶ req: object
    ▶ res: object

9/5/2025, 13:45:38  nodo: debug 8
msg.payload: Object
  ▼ object
    measurement: "detecciones"
    detections: 7
    detected: 1
    String: "9/5/2025, 13:45:38"
```

Graphana

Graphanarekin lan egiteko dashboard berria sortu behar da, dashboard hori influxDB-ko bucket-kin konektatuko dugu hurrengo kodea erabiliz.

```
A      (grupo4)

1  from(bucket:"Grupo4")
2    |> range(start: -1d)
3    |> filter(fn: (r) => r._measurement == "test" and r._field == "detections")
4    )
5
```