10. frame detector sensor if detect Buzzer Passive on and show in OLED

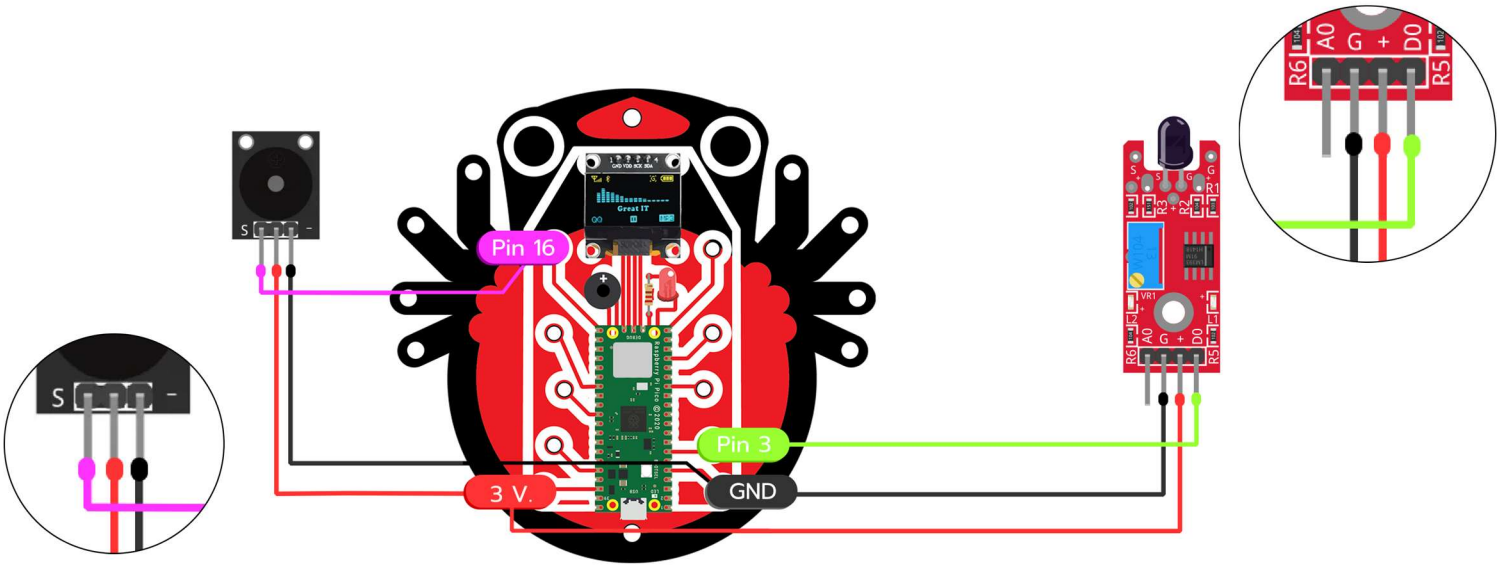
เตรียมอุปกรณ์

- Spidey
- OLED (บนบอร์ด Spidey)
- frame detector sensor
- Buzzer Passive

การเชื่อมต่อ

-ตัวเซนเซอร์จะคล้ายกับ เซนเซอร์วัดฝน แต่เซนเซอร์วัดไฟ ตัวนี้เราจะใช้ Digital และปรับค่าความเข้มข้นของแสงที่ตัวเซ้นเซอร์แทน เพื่อให้ง่ายต่อการใช้งาน โดยให้ต่อเซนเซอร์เข้ากับพินที่ 3 ของสไปดี้

-โปรเจกต์นี้เพิ่ม Buzzer แบบ Passive เข้ามา โดยต่อขาที่ 16 ของสไปดี้



การโค้ด

```
from machine import Pin
from machine import I2C
import ssd1306
```

```
from time import sleep
from machine import PWM
import time

pIn3=Pin(3, Pin.IN, Pin.PULL_UP)


while True:
  i2c=I2C(0, scl=Pin(1), sda=Pin(4))
  oled_width = 128
  oled_height = 64
  oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
  oled.text(('').join([str(x) for x in ['frame', ' : ',
pIn3.value()]])), 40, 47)
  oled.show()
  if (pIn3.value()) == 0:
    pwm15 = PWM(Pin(15))
    pwm15.freq(440)
    pwm15.duty_u16(10000)
    time.sleep(1)
  else:
    pwm15 = PWM(Pin(15))
    pwm15.freq(1000)
    pwm15.duty_u16(0)
  oled.fill(0)
  time.sleep(1)

  time.sleep(1)
```

อธิบายโค้ด

โค้ดนี้เป็นตัวอย่างการใช้เซ็นเซอร์ตรวจจับเฟรม (frame detector sensor) ร่วมกับหน้าจอ OLED และบัซเซอร์ (buzzer) ดังนี้:

1. นำเข้าโมดูลและตั้งค่าตัวแปร:
    - **from machine import Pin**: นำเข้าโมดูล Pin เพื่อใช้ในการควบคุมขา GPIO.
    - **from machine import I2C**: นำเข้าโมดูล I2C เพื่อใช้ในการสื่อสารผ่าน I2C.
    - **import ssd1306**: นำเข้าโมดูล ssd1306 เพื่อใช้ในการควบคุมหน้าจอ OLED.
    - **from time import sleep**: นำเข้าฟังก์ชัน sleep เพื่อใช้ในการหน่วงเวลา.
    - **from machine import PWM**: นำเข้าโมดูล PWM เพื่อใช้ในการควบคุมสัญญาณ PWM.

- **import time**: นำเข้าโมดูล time เพื่อใช้ในการหน่วงเวลา.

2. กำหนดขา GPIO ที่เชื่อมต่อกับเซ็นเซอร์ตรวจจับเฟรม:
    - **pIn3=Pin(3, Pin.IN, Pin.PULL_UP)**: กำหนดขา GPIO 3 ให้เป็นขาป้องกันแบบ Pull-up.
3. เริ่มลูปการทำงานอย่างต่อเนื่อง:
    - **while True:**: เริ่มลูปการทำงานอย่างต่อเนื่อง.
4. กำหนดการเชื่อมต่อและตั้งค่าหน้าจอ OLED:
    - **i2c=I2C(0, scl=Pin(1), sda=Pin(4))**: สร้างอ็อบเจ็กต์ I2C เพื่อเชื่อมต่อกับหน้าจอ OLED ที่ขา SCL (GPIO 1) และ SDA (GPIO 4).
    - **oled_width = 128** และ **oled_height = 64**: กำหนดขนาดความกว้างและความสูงของหน้าจอ OLED.
    - **oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)**: สร้างอ็อบเจ็กต์ OLED ด้วยขนาดความกว้างและความสูงที่กำหนด.
5. แสดงผลสถานะของเฟรมบนหน้าจอ OLED:
    - **oled.text((''.join([str(x) for x in ['frame', ' : ', pIn3.value()]])), 40, 47)**: สร้างข้อความที่แสดงสถานะของเฟรมโดยใช้ค่าจากการอ่านขา GPIO 3 และแสดงผลที่พิกัด (40, 47) บนหน้าจอ OLED.
    - **oled.show()**: แสดงผลที่ออกแบบไว้บนหน้าจอ OLED.
6. ควบคุมเสียงด้วยสัญญาณ PWM ตามสถานะของเฟรม:
    - **if (pIn3.value()) == 0:**: ตรวจสอบว่าสถานะของเฟรมเป็น 0 หรือไม่ (ไม่มีการตรวจจับเฟรม).
        - **pwm15 = PWM(Pin(15))**: สร้างอ็อบเจ็กต์ PWM ที่ขา GPIO 15.
        - **pwm15.freq(440)**: กำหนดความถี่ของสัญญาณ PWM เป็น 440 Hz.
        - **pwm15.duty_u16(10000)**: กำหนดระดับความสว่างสูงสุดของสัญญาณ PWM เป็น 10000.
        - **time.sleep(1)**: หน่วงเวลา 1 วินาที.
    - **else:**: กรณีสถานะของเฟรมไม่เป็น 0 (มีการตรวจจับเฟรม).
        - **pwm15 = PWM(Pin(15))**: สร้างอ็อบเจ็กต์ PWM ที่ขา GPIO 15.
        - **pwm15.freq(1000)**: กำหนดความถี่ของสัญญาณ PWM เป็น 1000 Hz.
        - **pwm15.duty_u16(0)**: กำหนดระดับความสว่างของสัญญาณ PWM เป็น 0.
7. เคลียร์หน้าจอ OLED และหน่วงเวลา:
    - **oled.fill(0)**: เคลียร์หน้าจอ OLED เป็นสีดำ.

- **time.sleep(1)**: หน่วงเวลา 1 วินาที.

โค้ดดังกล่าวจะทำงานอย่างต่อเนื่อง โดยตรวจจับสถานะของเฟรมผ่านเซ็นเซอร์ตรวจจับเฟรมที่เชื่อมต่อกับขา GPIO 3 และแสดงผลสถานะบนหน้าจอ OLED พร้อมกับส่งสัญญาณ PWM เพื่อควบคุมเสียงบัซเซอร์ที่ขา GPIO 15 ตามสถานะของเฟรมที่ตรวจจับได้