First steps towards implementing the project:

Circuit

LED: Connect the anode (longer side) of the LED to GPIO 21 and the
cathode (shorter side) to a 220-330Ω resistor, which then goes to GND (ground) on the Raspberry Pi.
then goes to the Raspberry Pi's GND (ground).
Button: Connect one side of the button to GPIO 17 and the other side to GND.
Use an internal pull-up resistor in your code to
avoid false readings.

Python Code:

```
import RPi.GPIO as GPIO
import time
from pythonosc import udp_client

# Defines the GPIO pins for the ultrasonic sensor, LED and button
GPIO.setmode(GPIO.BCM)
TRIG = 23
ECHO = 24
LED = 21
BUTTON = 17

# Configures the ultrasonic sensor, LED and button pins
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)
GPIO.setup(BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Initialises LED off
GPIO.output(LED, GPIO.LOW)
# Initial button status
buttonState = False

# Configures the OSC client to send data to Pure Data
client = udp_client.SimpleUDPClient('localhost', 8000)

def measure_distance():
"""Measures distance using the ultrasonic sensor."""
    GPIO.output(TRIG, False)
    time.sleep(0.1)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)
    return distance

def button_callback(channel):
"""Handles button presses by switching the LED status."""
    global buttonState
    buttonState = not buttonState
    GPIO.output(LED, buttonState)  # Alterna o estado do LED
    client.send_message("/button", int(buttonState))  # Envia o estado do botão via OSC
    print(f"Estado do botão: {int(buttonState)}")

# Sets up edge detection for the button with 300ms debounce
GPIO.add_event_detect(BUTTON, GPIO.FALLING, callback=button_callback,
bouncetime=300)

try:
    while True:
        distance = measure_distance()
    # Sends the distance value via OSC
        client.send_message("/distance", distance)
# Prints the distance and status of the button on the console
        print(f"Distância: {distance} cm, Estado do botão: {int(buttonState)}")
        time.sleep(0.5)

except KeyboardInterrupt:
    print("Programa terminado pelo usuário")
    GPIO.cleanup()  # Clears the GPIO pin configuration
```

Steps to make the file executable from the terminal:

1.Create a file from the terminal

```
nano geradorsonoro.py
```

Paste the python code into the file generatorsonoro.py

NOTE: insert the line at the beginning of the code:

```
#!/usr/bin/env python3
```

Save: Ctrl+O
Close: Ctrl+X

2. activate the virtual environment

```
source venv/bin/activate
```

3. get permission

```
chmod +x geradorsonoro.py
```

4. execute the file

```
./geradorsonoro.py
```

In Pure Data

message: listen 8000
object: netreceive -u -b
object: oscparse
object: route list
object: route button object:route distance
object: number object: number

this patch reads the distance values and the button status (0 and 1).