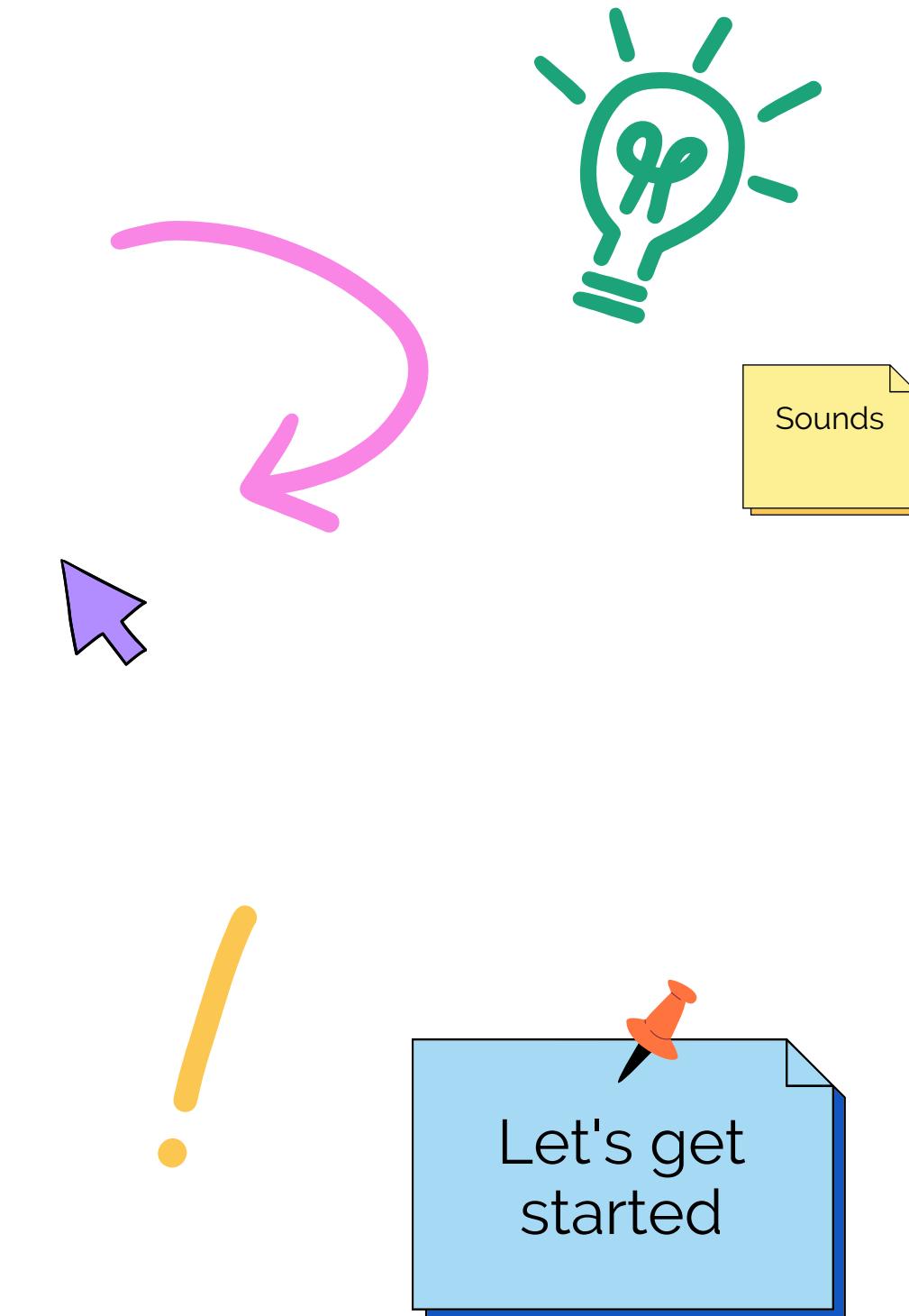
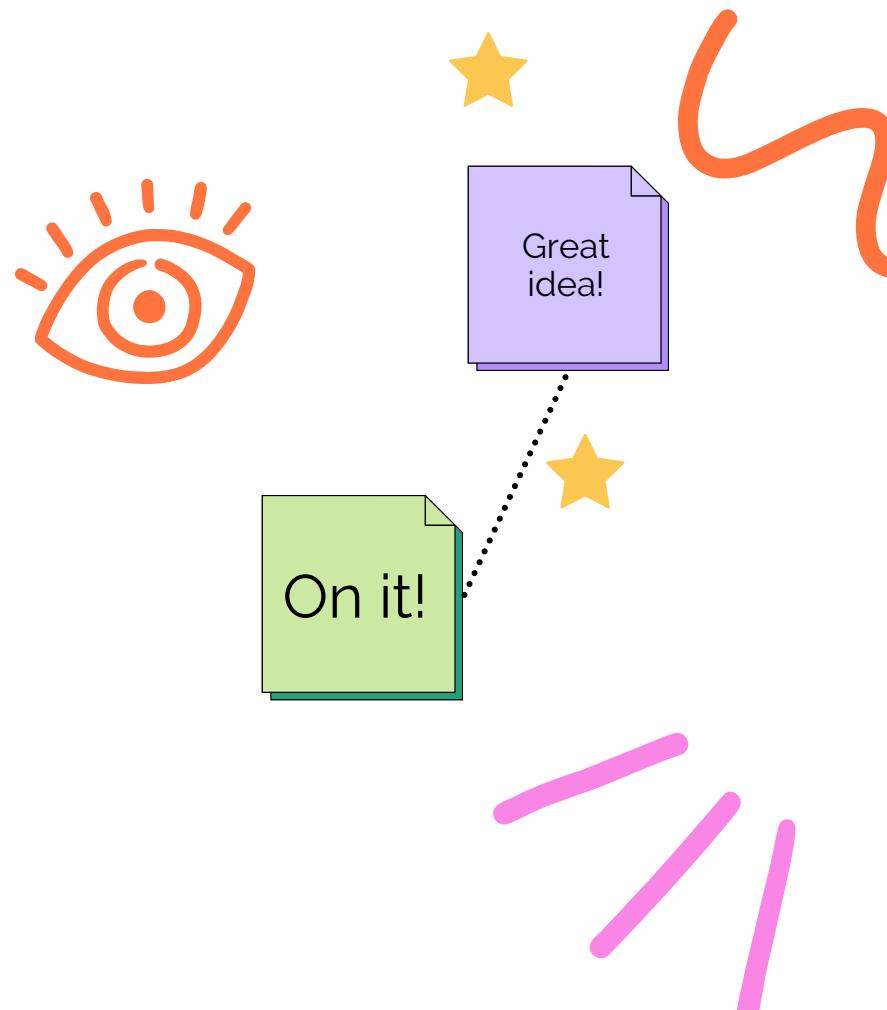
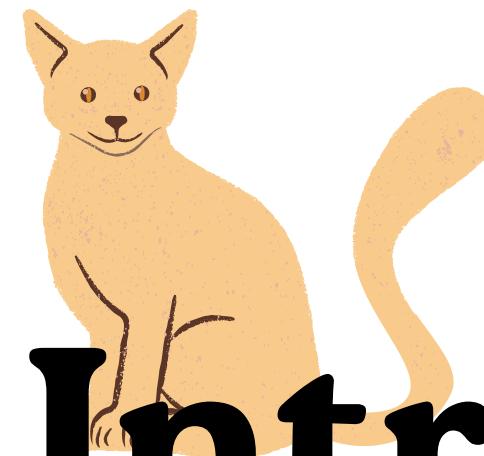


# Project Volume Control System & Game

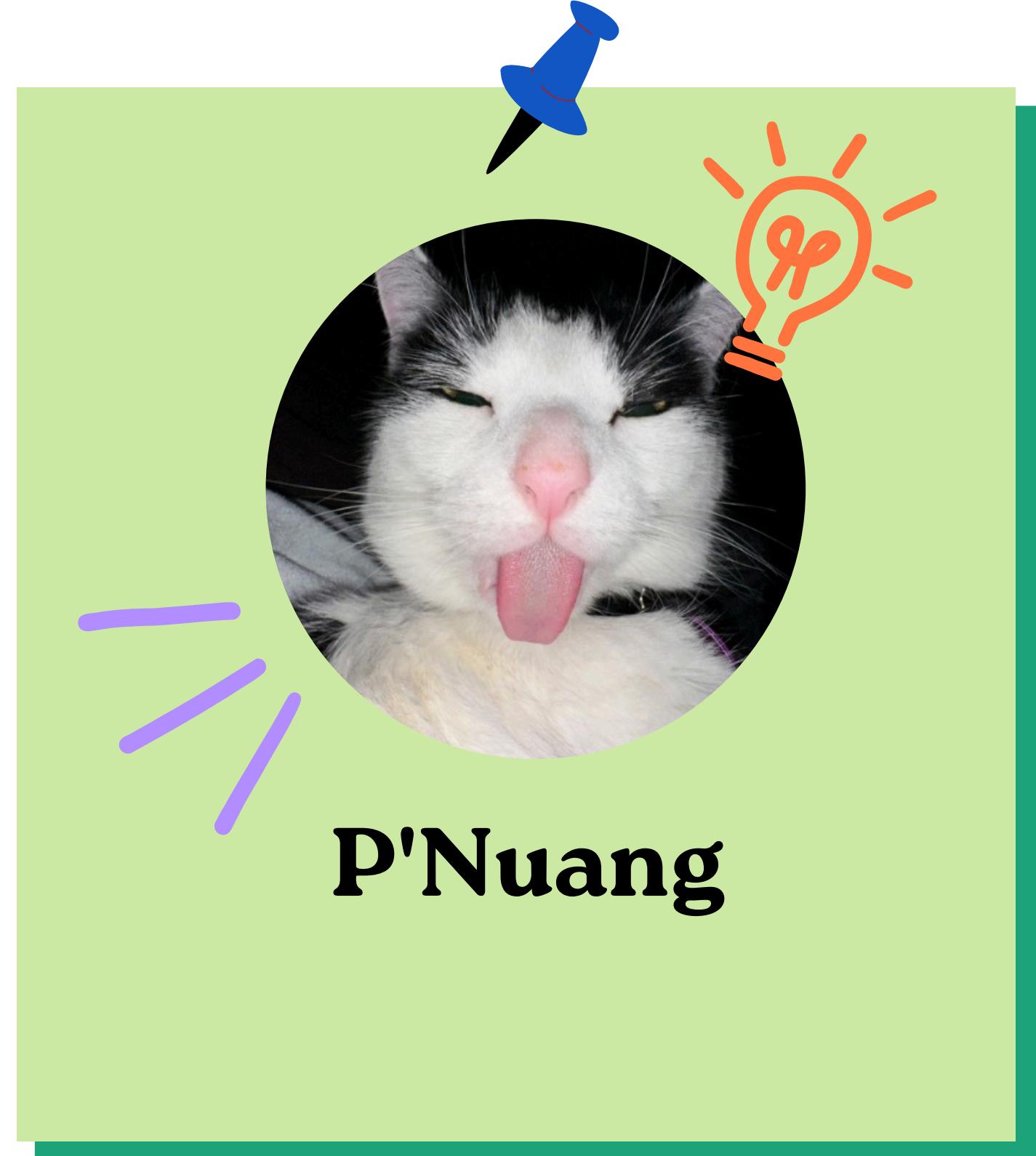




# Introduction

## Group member

- ✓ 2211310764 บัตร์มน จันทร์สีดา
- ✓ 2211311465 ภูมิพัฒน์ สุดใจ
- ✓ 2211311549 พิชญุตม์ ไก铍ร
- ✓ 2211312141 พิรประภา พฤกษ์จะมาศ



# Group member

✓ 2211310764 นักรมน จันทร์สีดา



✓ 2211311465 ภูมิพัฒน์ สุดใจ



✓ 2211312141 พีระพง พฤกษาวงศ์



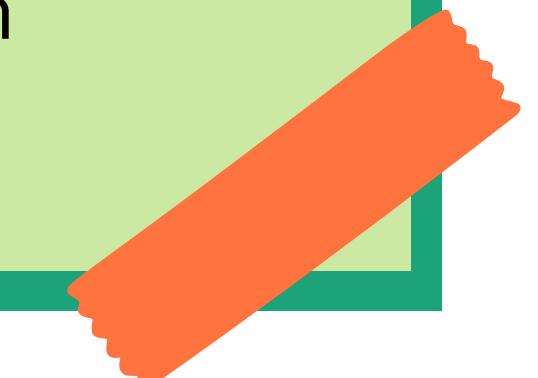
✓ 2211311549 พิชณุตม์ ไก่ธ์



# Project Overview

เป็นโปรเจกต์ที่มุ่งเน้นการควบคุมตัวโน้มต่อ Potentiometer และ โหมดการเล่นเกม

และแสดงผลระดับเสียงนั้นบนจอ OLED Display ซึ่งสามารถนำไปประยุกต์ใช้ในการควบคุมระดับเสียงของลำโพงหรืออุปกรณ์เครื่องเสียงอื่น ๆ ได้



# System components

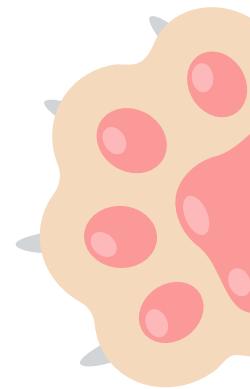


External  
Interrupt

Speaker

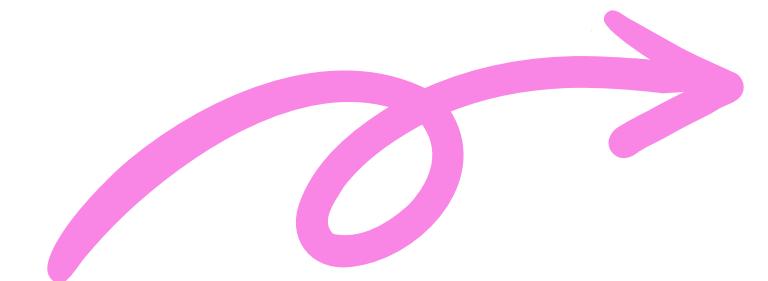


- **ESP32** : อ่านค่าจาก Potentiometer และควบคุมการแสดงผล
- **Potentiometer** : ใช้ปรับระดับเสียง และปรับความสว่างของ LED
- **OLED Display** : แสดงเป็นข้อมูลสถานะต่างๆ
- **LED** : จะสว่างตามระดับเสียง โดยใช้ PWM ในการควบคุมความสว่างของ LED ตามค่าที่อ่านจาก Potentiometer ยิ่งระดับเสียงสูง LED ก็ยิ่งสว่างมากขึ้น
- **External Interrupt** : สำหรับกดเปลี่ยน Mode และ อื่นๆ
- **Speaker** : ใช้เปลี่ยนตัวโน้ตตามการหมุน Potentiometer

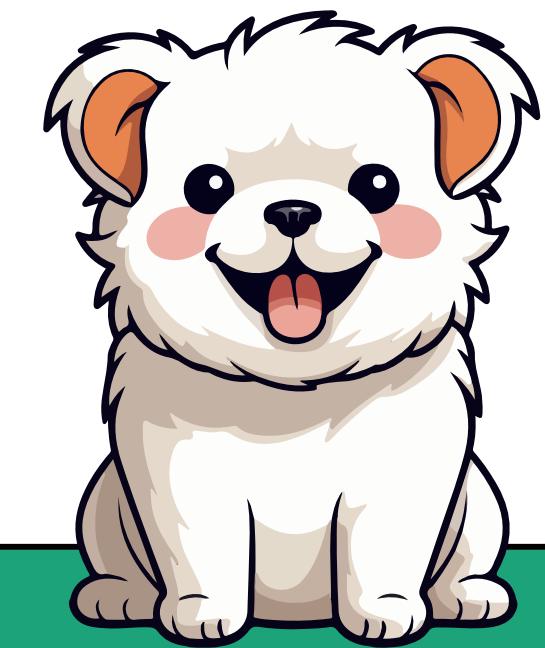




# การใช้งาน

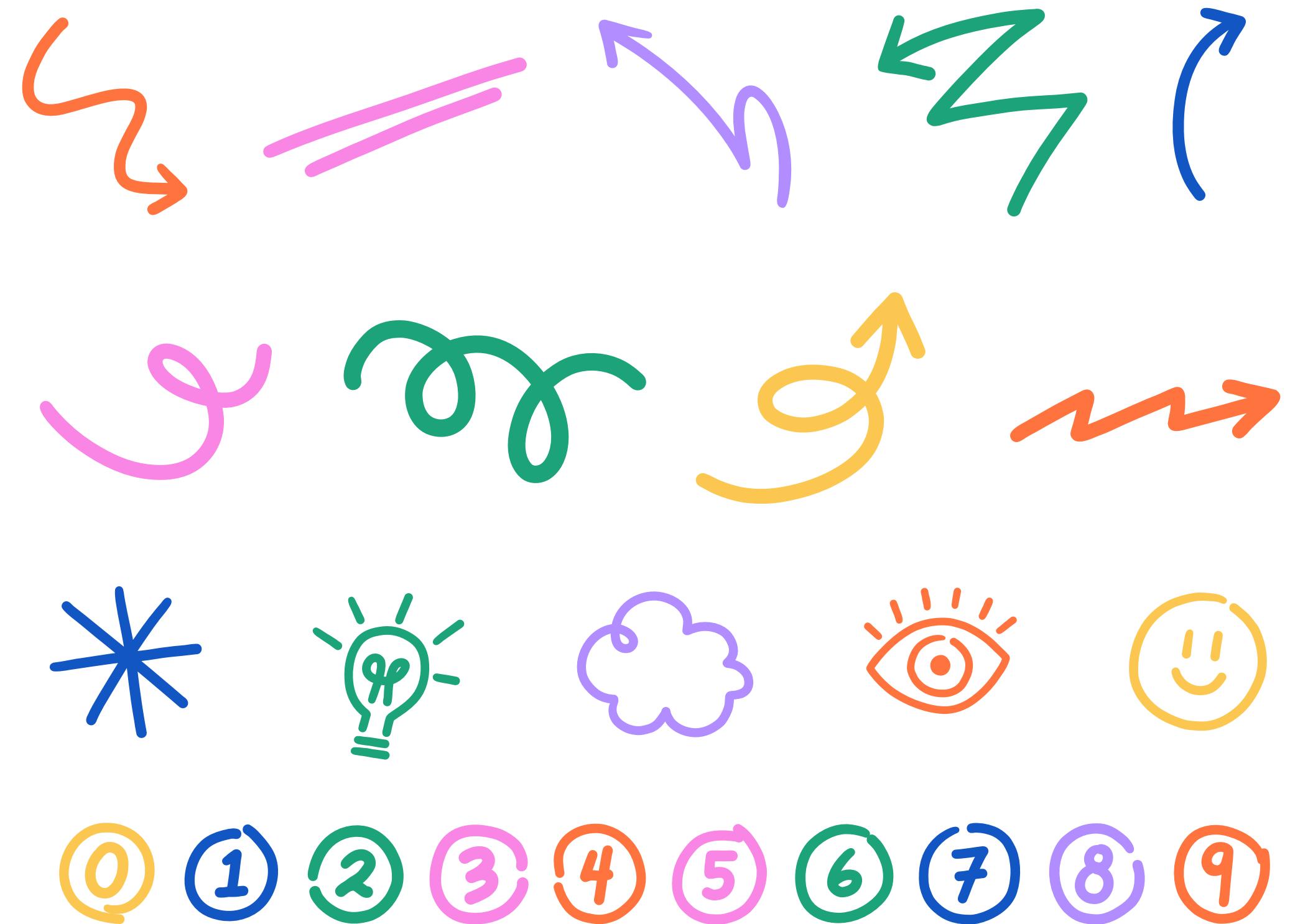
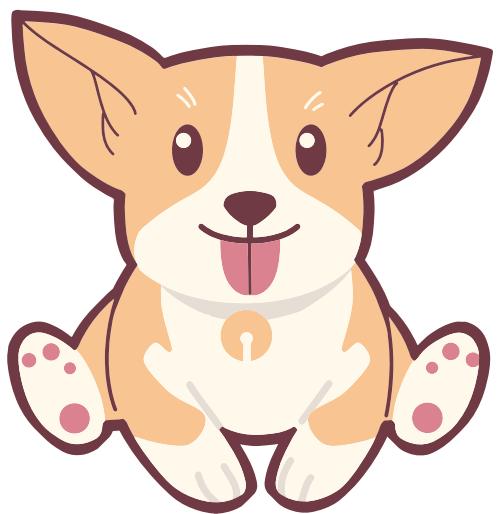


- ผู้ใช้หมุน Potentiometer เพื่อปรับระดับเสียง
- ESP32 อ่านค่าและแสดงระดับเสียงบนจอ OLED จะแสดงเป็น percent ตัวเลข
- LED จะแสดงความสว่างตามระดับของเสียงที่เพิ่มขึ้น(0-100%)
- กดปุ่มให้ได้ 3 ครั้งใน 7 วินาทีเพื่อชบะเกม



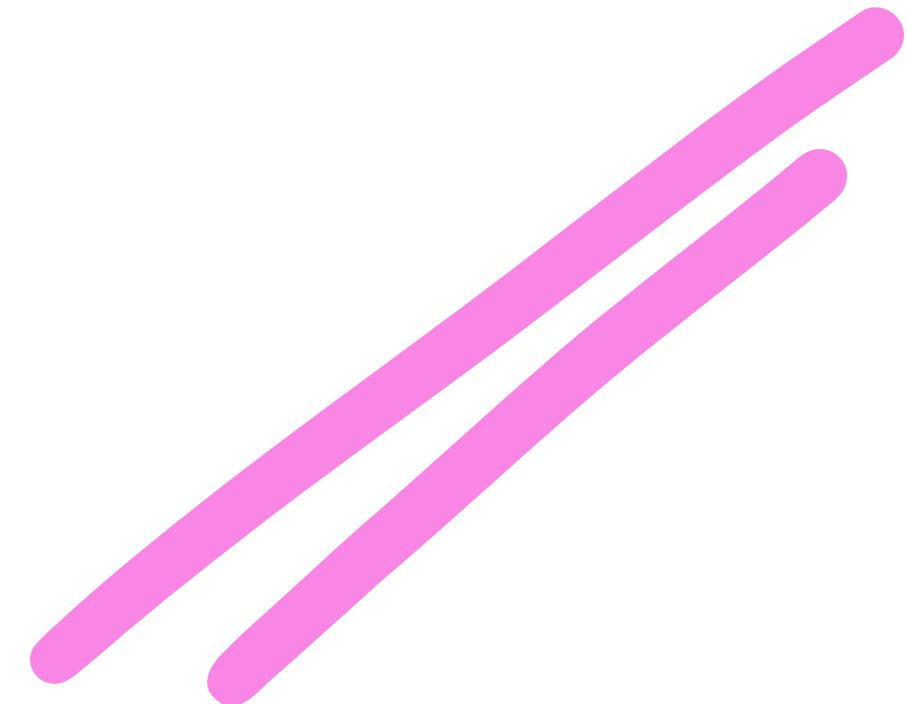
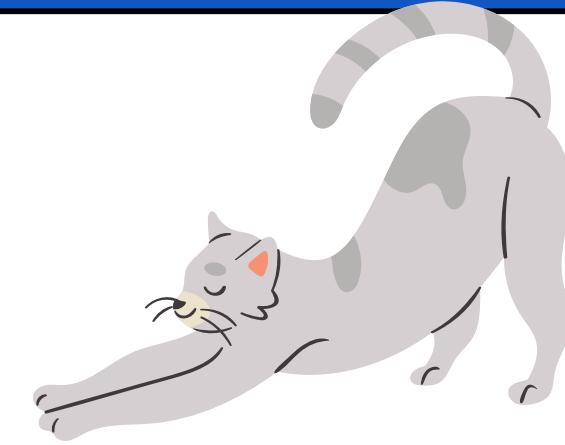
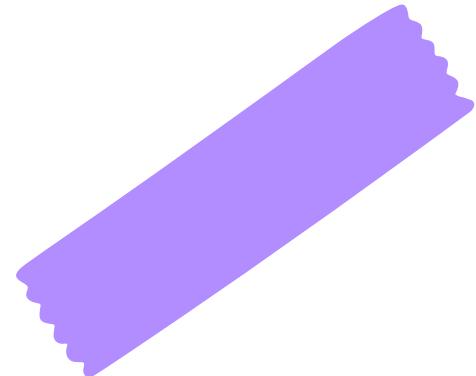
# V-Model

V-shape model



# Functional Requirements

- ปรับตัวโน้ตโดย  
Potentiometer
- แสดงระดับเสียงบน  
OLED
- Control Output
- ไฟ LED แสดงสถานะ
- แสดงผลเกมแพ้ชนะบน  
OLED



# ปรับโน๊ตเสียงโดย Potentiometer

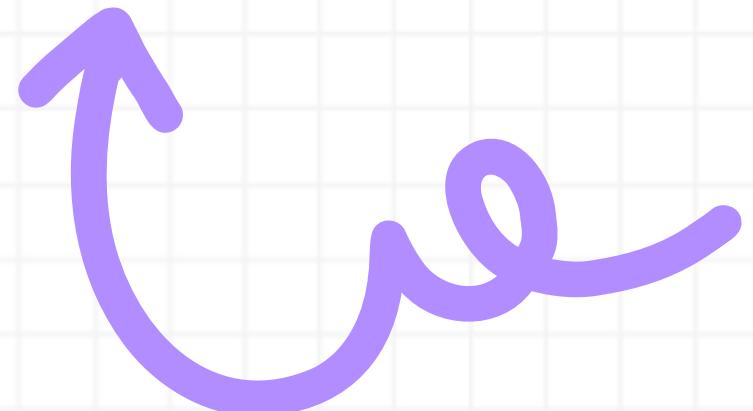
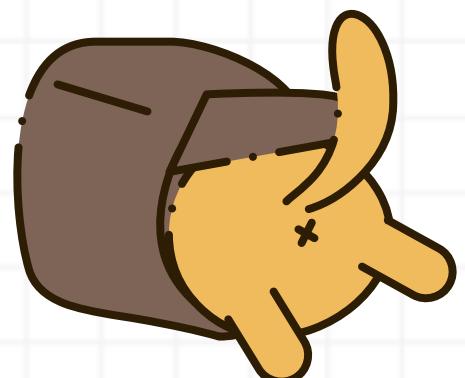


- ระบบจะต้องอนุญาตให้ผู้ใช้ปรับระดับเสียงโดยใช้ Potentiometer
- Potentiometer ต้องปรับระดับเสียงตั้งแต่ 0% ถึง 100% ได้อย่างราบรื่น



## แสดงระดับเสียงบน OLED

- ระบบจะต้องแสดงระดับเสียงปัจจุบันเป็นเปอร์เซ็นต์ (0-100%) บนหน้าจอ OLED แบบเรียลไทม์เมื่อมีการปรับ Potentiometer
- หน้าจอ OLED ควรอัปเดตทันทีเมื่อเปิด Potentiometer
- ระบบจะต้องแสดงการบับคอยหลังในโหมดเกม จำนวนการกด และผลแพ้ชนะ



# Control Output

ระบบควรใช้ PWM เพื่อส่งสัญญาณเอาต์พุตที่สอดคล้องกันเพื่อควบคุมระดับเสียงของลำโพงหรืออุปกรณ์เสียงอื่นที่เชื่อมต่ออยู่

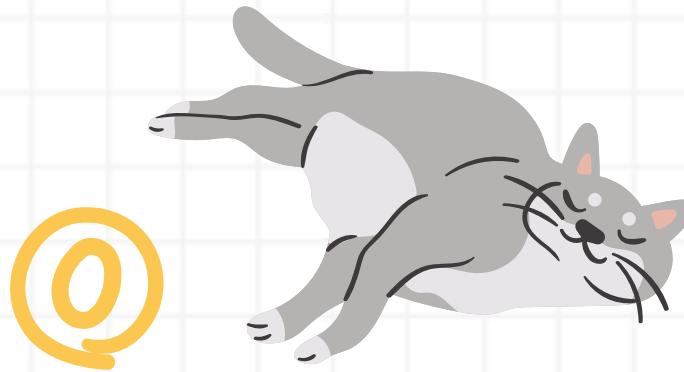


## ไฟ LED แสดงสถานะ

- ระบบควรมีไฟ LED ที่แสดงระดับเสียงปัจจุบันโดยการปรับความสว่างตามเปอร์เซ็นต์ระดับเสียง

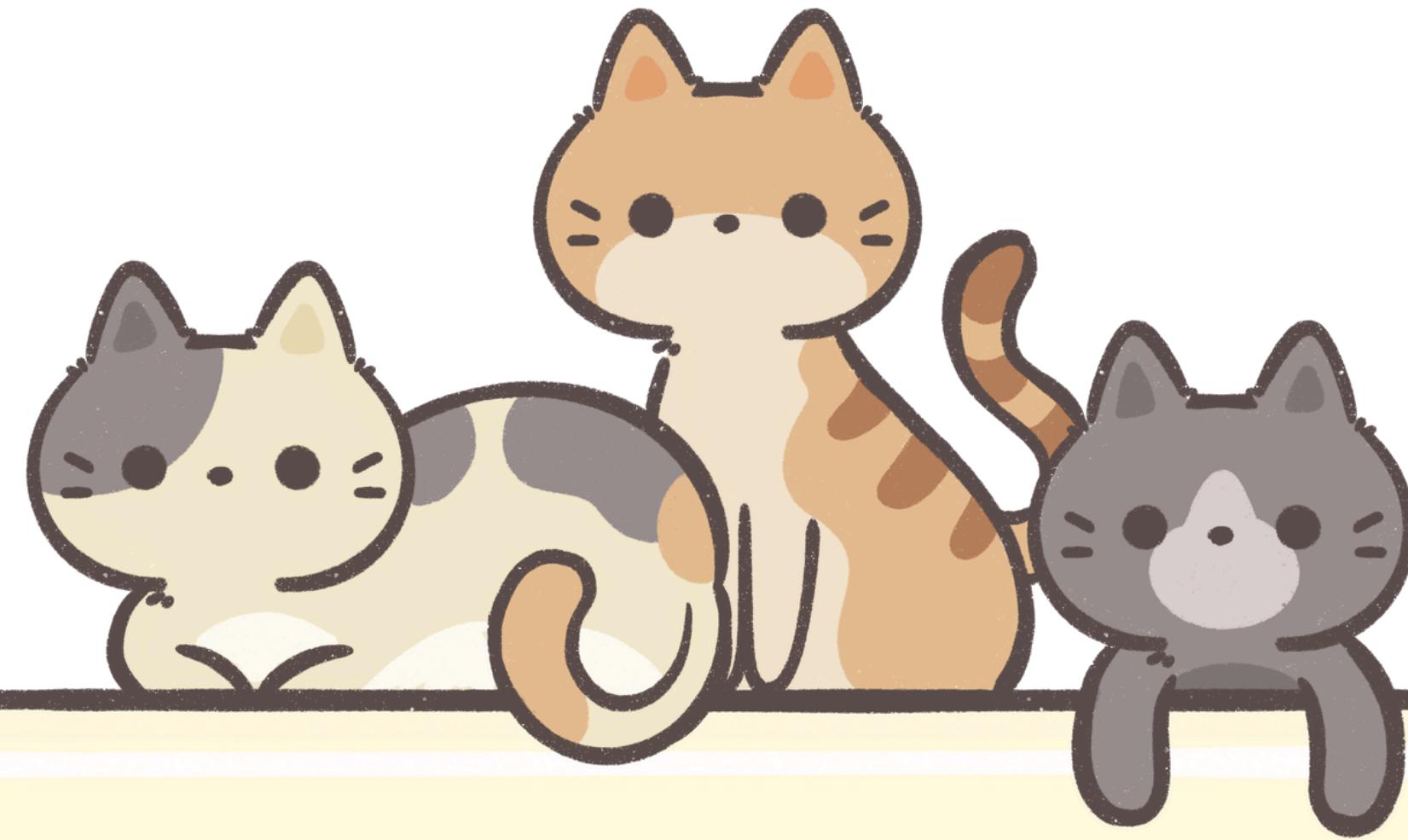


# Requirements <-> Acceptance testing



- ① Potentiometer สามารถปรับระดับเสียงได้อย่างแม่นยำ
- ② OLED แสดงระดับเสียงอย่างแม่นยำและเป็นเรียลไทม์
- ③ เสียงที่ออกจาก Speaker เปลี่ยนแปลงตามระดับเสียง
- ④ LED สว่างตามระดับเสียง
- ⑤ ระบบตอบสนองต่อการปรับ Potentiometer ได้อย่างรวดเร็ว
- ⑥ เกมกดรู้ผลแพ้ชนะ
- ⑦
- ⑧
- ⑨





# Specification

# system info

esp32

OLED

speaker

potentiometer

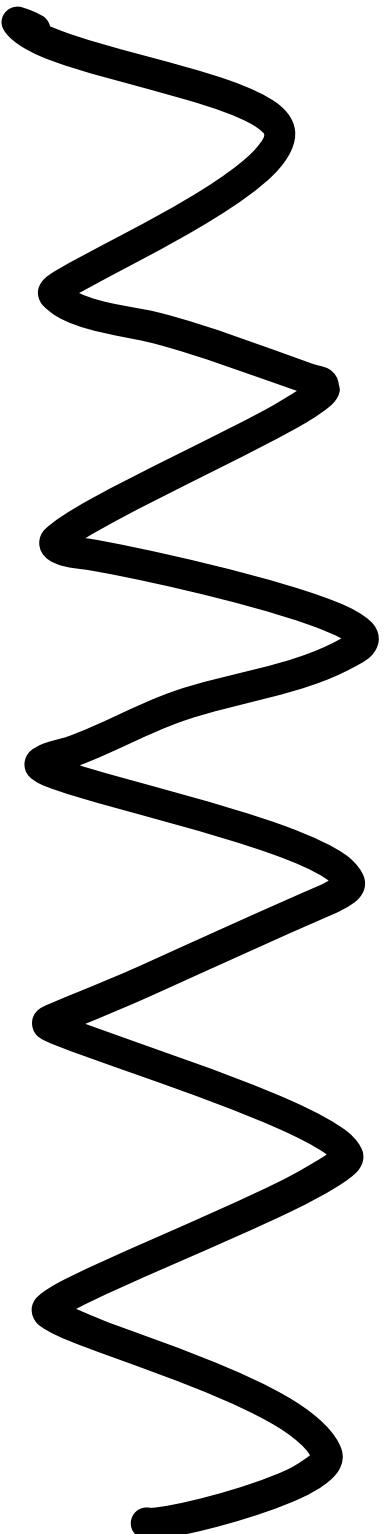
led

button

# function

change volume

play game



# behavior

Volume song

adjust volume 0% - 100%

LED brightness

Show display

show volume 0% - 100%

Play game

push button 3 times in 7 seconds



# specification

**เปลี่ยนโหมด ( เล่นเกม, พังเพว ):**

mode selection, mode song control, mode play game

**How to use system:**

เริ่มต้นที่ mode selection

if กดปุ่ม 1 ครั้ง

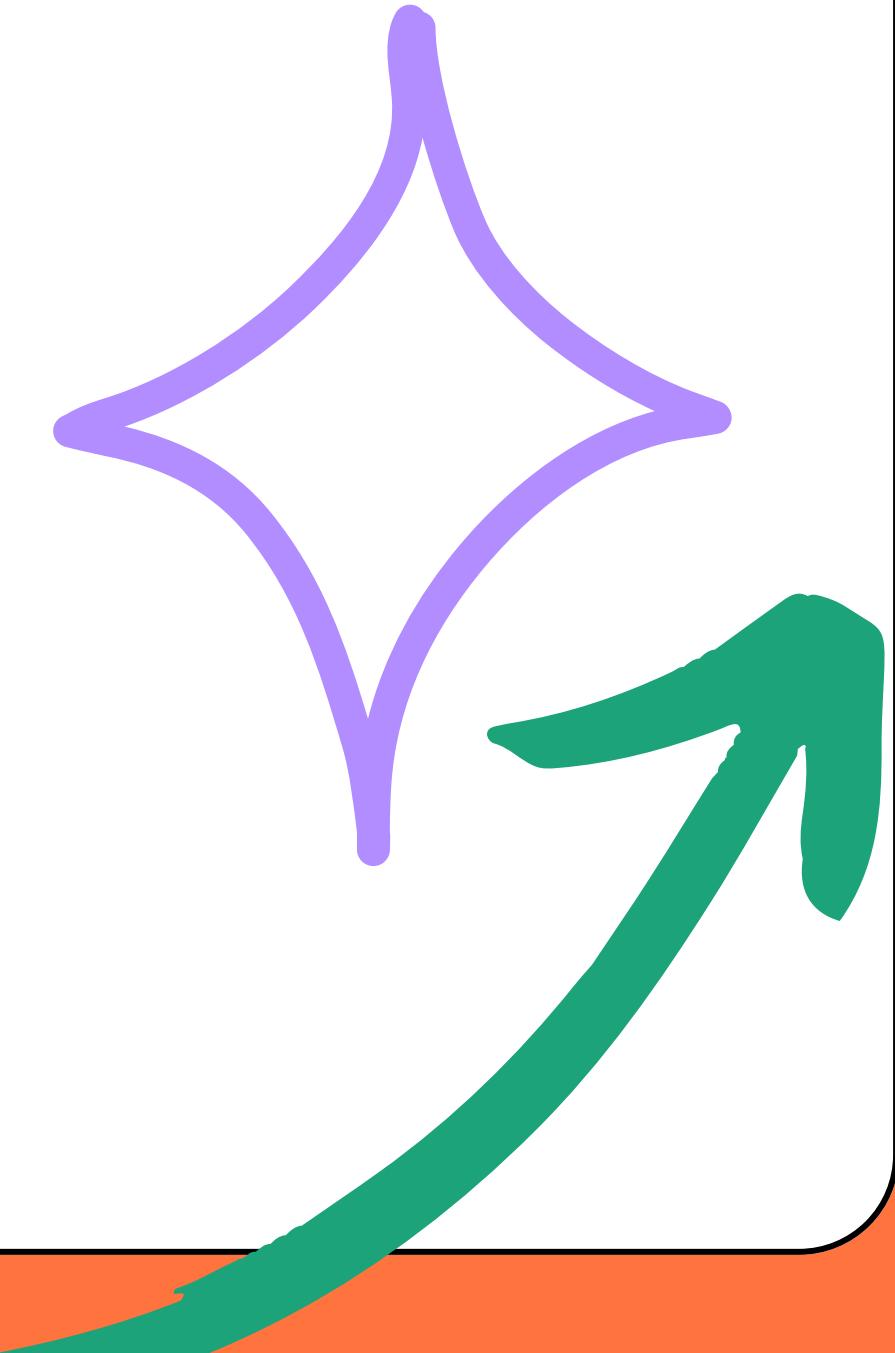
ไปที่ mode song control

In mode song control กดปุ่ม 1 ครั้ง

ไปที่ mode play game

In mode play game for 5 seconds

กลับไปที่ mode selection



# specification

## mode play game:

if push\_button >= 3 in 7 seconds

ໃນ oled ຈະແສດງວ່າ You Win!

ຄ້າໄມ່ໄດ້ກີເລັນໃໝ່ຈົນກວ່າຈະກຳປັບປຸງໂທມດ



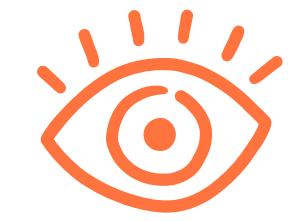
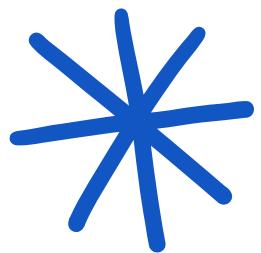
## mode play song:

volume = (pot\_read/1023)\*100

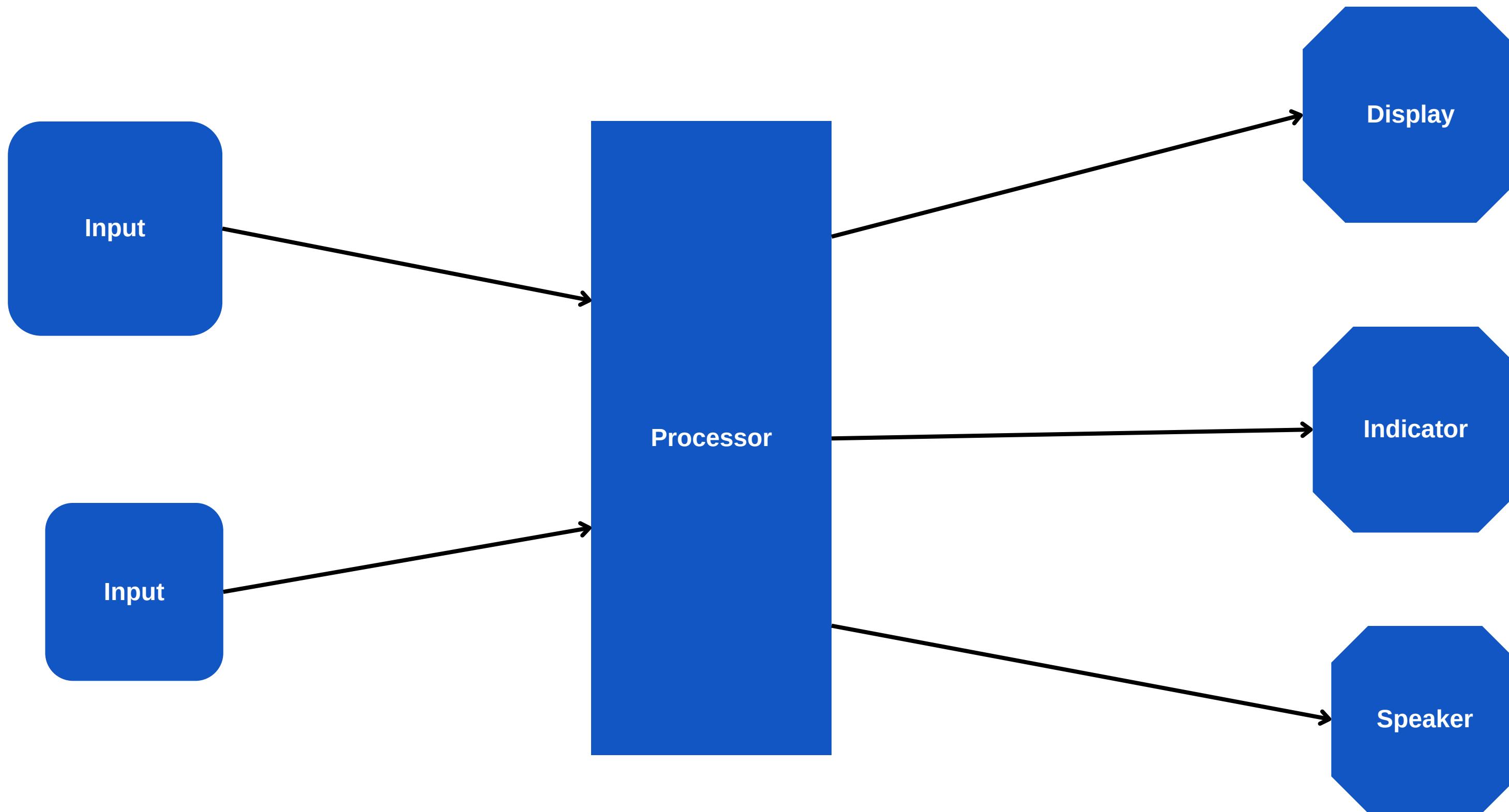
ຮະດັບຂອງເສີຍງຈະປັບຕາມຄ່າກີວັດໄດ້

oled ກຳການແສດງຮະດັບເສີຍງ 0 - 100%

# Architectural design



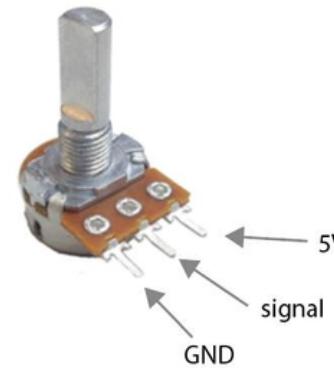
# Block Diagram



# Select Hardware : Input



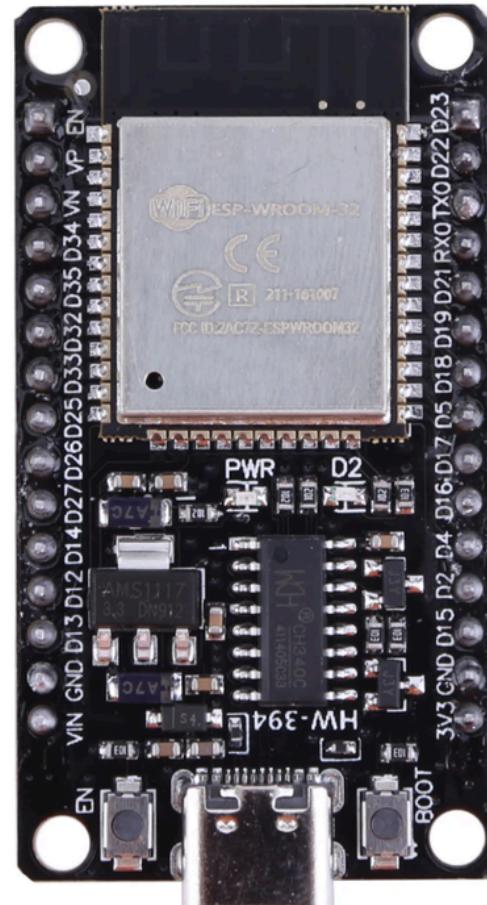
Generic Arduino Potentiometer



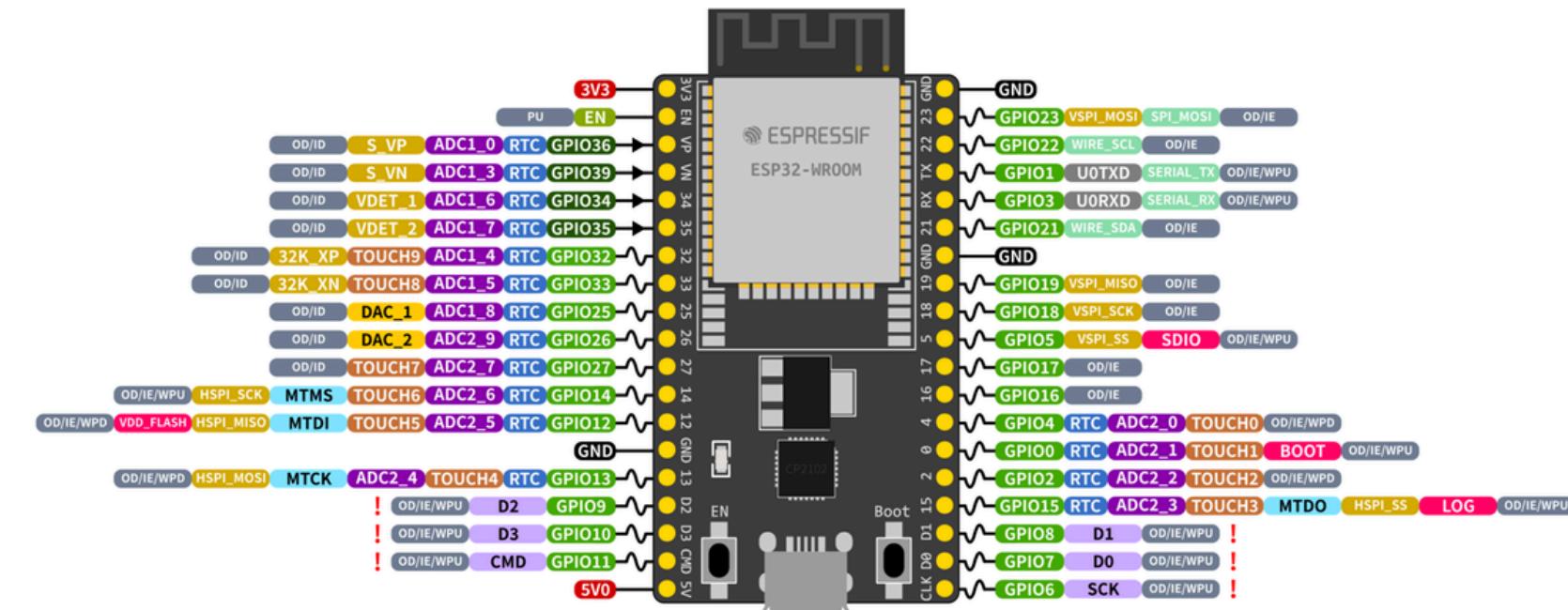
2 Legs tactile switch

# Select Hardware : Processor

## ESP32 WROOM



ESP32-DevKitC



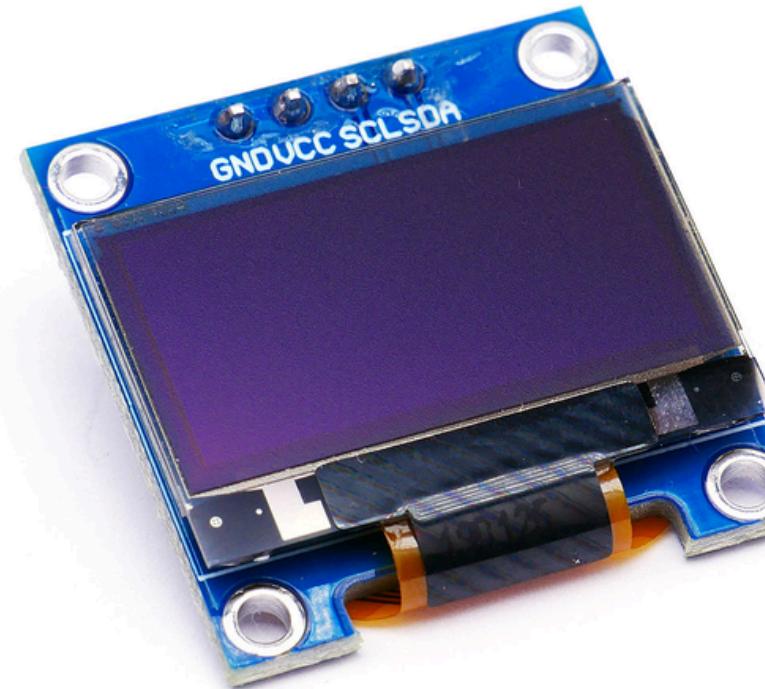
### ESP32 Specs

32-bit Xtensa® dual-core @240MHz  
Wi-Fi IEEE 802.11 b/g/n 2.4GHz  
Bluetooth 4.2 BR/EDR and BLE  
520 KB SRAM (16 KB for cache)  
448 KB ROM  
34 GPIOs, 4x SPI, 3x UART, 2x I2C,  
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,  
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

- PWM Capable Pin
- GPIO Input Only
- GPIO Input and Output
- Digital-to-Analog Converter
- JTAG for Debugging
- External Flash Memory (SPI)
- Analog-to-Digital Converter
- Touch Sensor Input Channel
- Other Related Functions
- Serial for Debug/Programming
- Arduino Related Functions
- Strapping Pin Functions

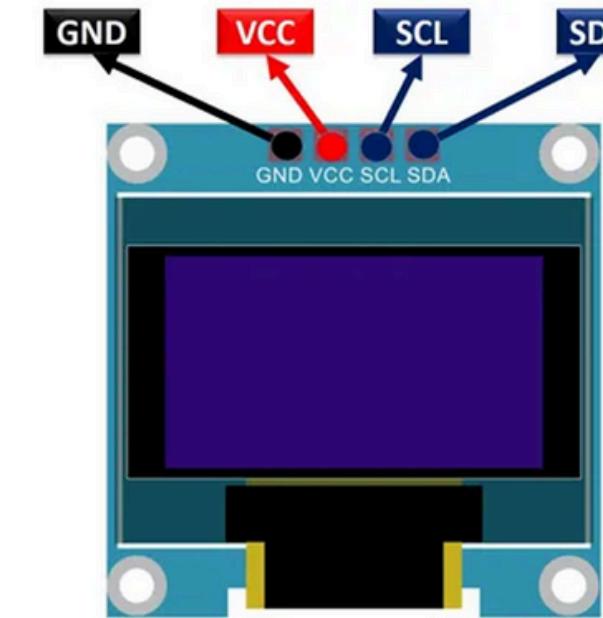
- RTC RTC Power Domain (VDD3P3\_RTC)
- GND Ground
- PWD Power Rails (3V3 and 5V)
- ! Pin Shared with the Flash Memory
- WPU Weak Pull-up (Internal)
- WPD Weak Pull-down (Internal)
- PU Pull-up (External)
- ID Input Enabled (After Reset)
- ID Input Disabled (After Reset)
- OE Output Enabled (After Reset)
- OD Output Disabled (After Reset)

# Select Hardware : Display



## SSD1306 128x64 Mono I2C OLED Display

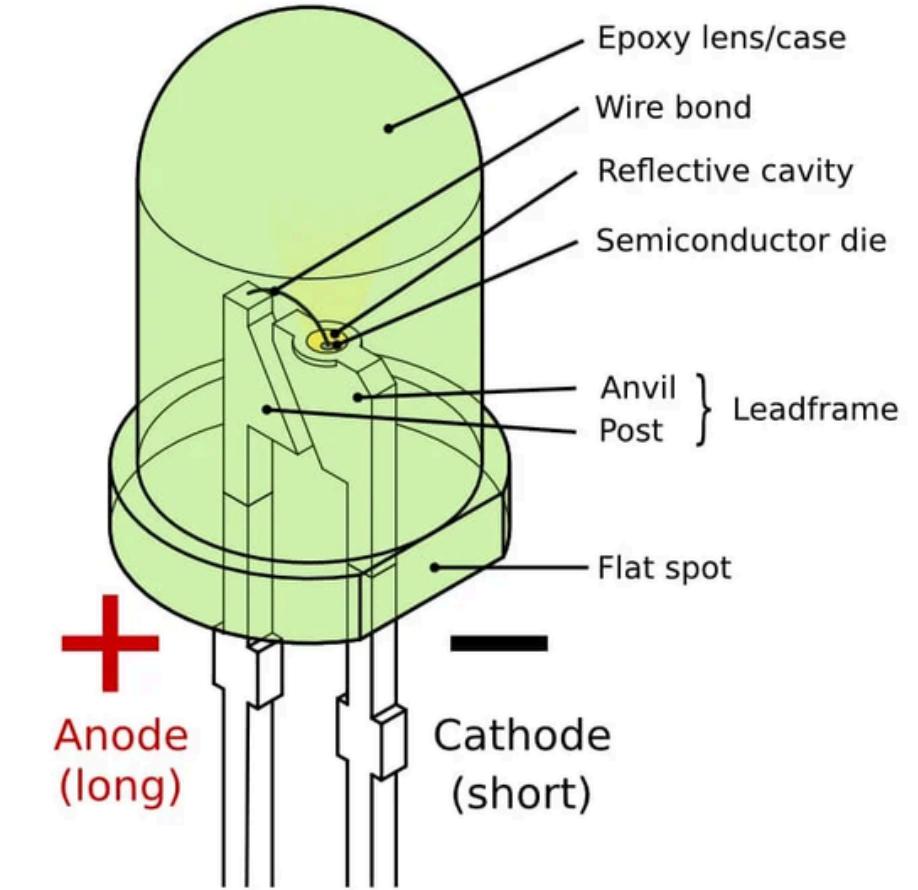
### SSD1306 Pinout



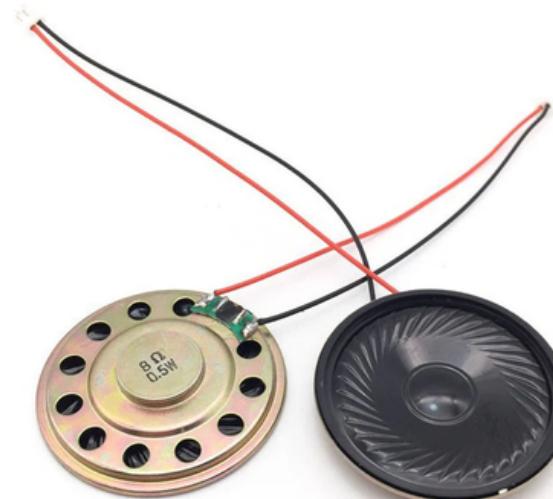
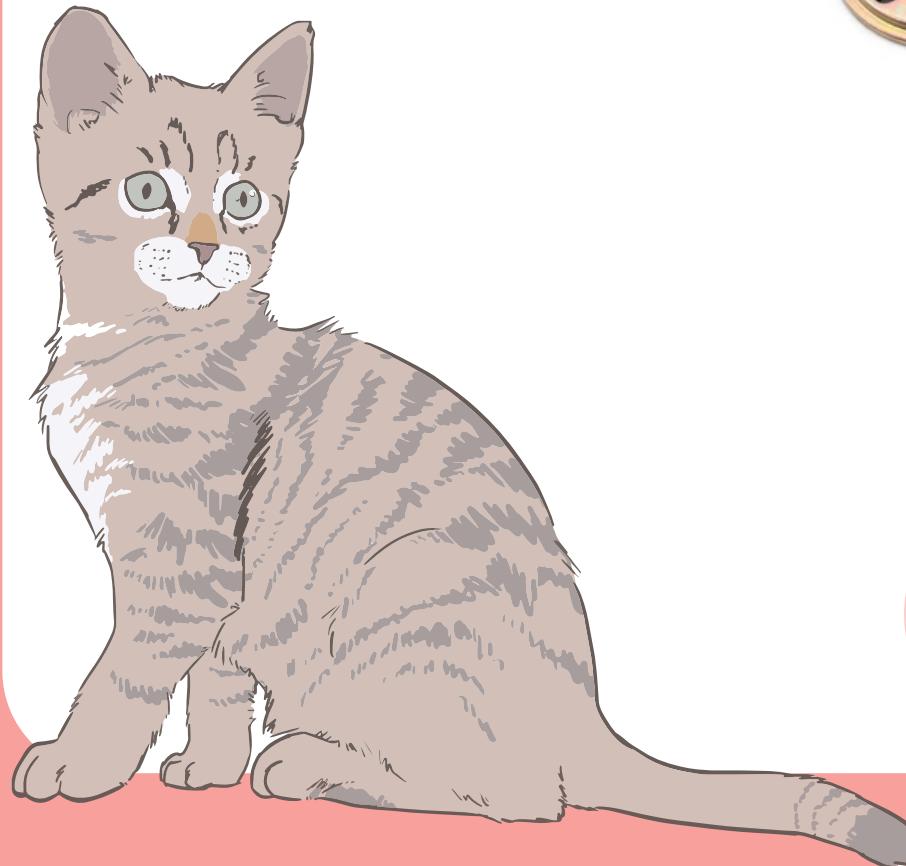
Pin Name	Description
GND	ground pin
Power (VCC, VDD, 5V)	single power pin for all the power inputs.
SCK (CLK, SCL)	performs a common clock signal for both microcontroller/Arduino with OLED.
SDA (MOSI, DI)	receives the data from the controlling device/Arduino.

# Select Hardware : Indicator

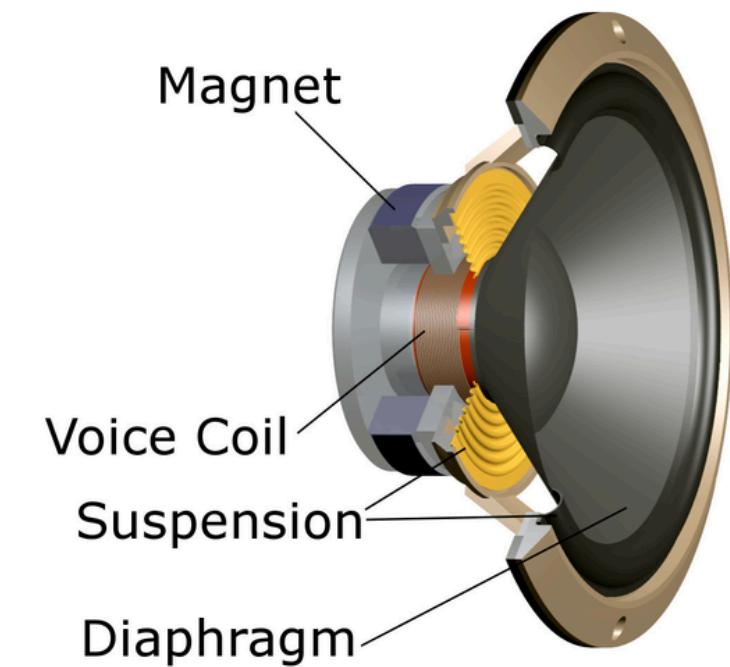
Generic LED



# Select Hardware : Speaker



## Basic Speaker

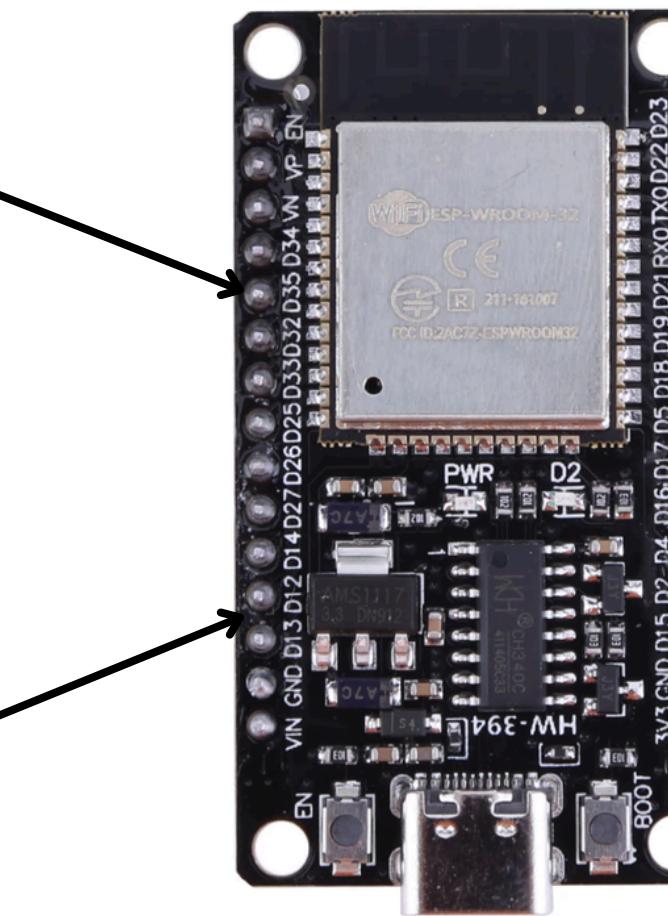




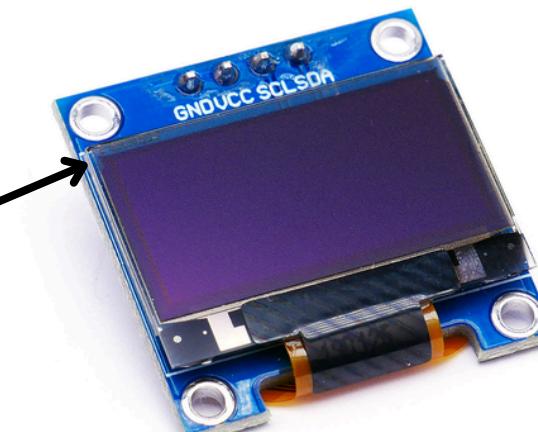
Generic Arduino potentiometer



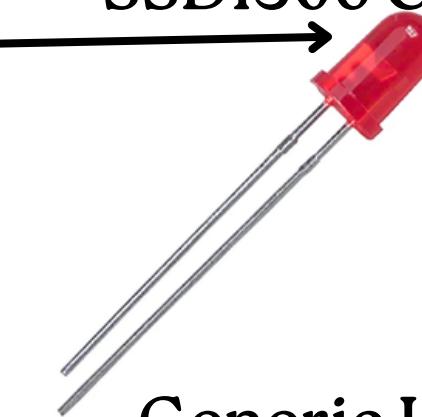
2 Legs tactile switch



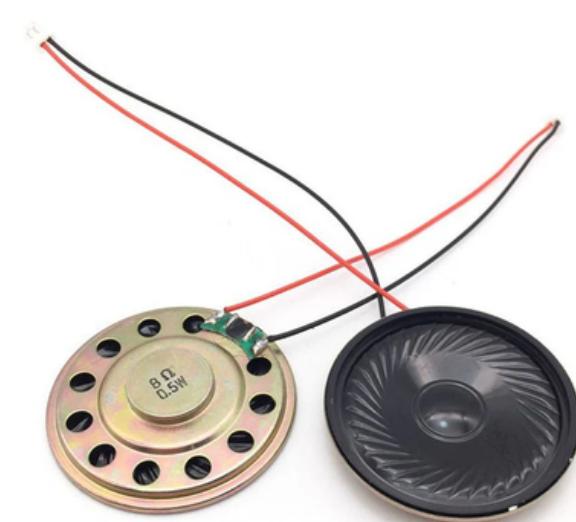
ESP32 WROOM



SSD1306 OLED

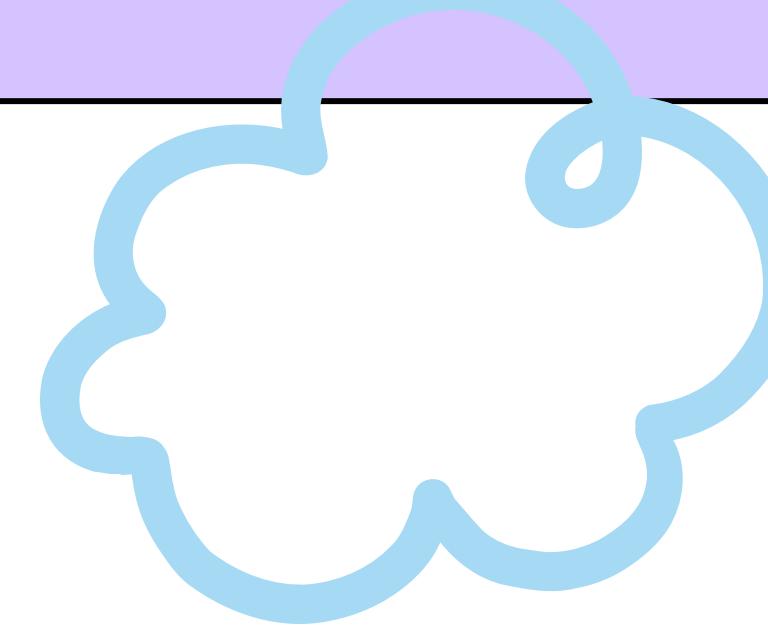
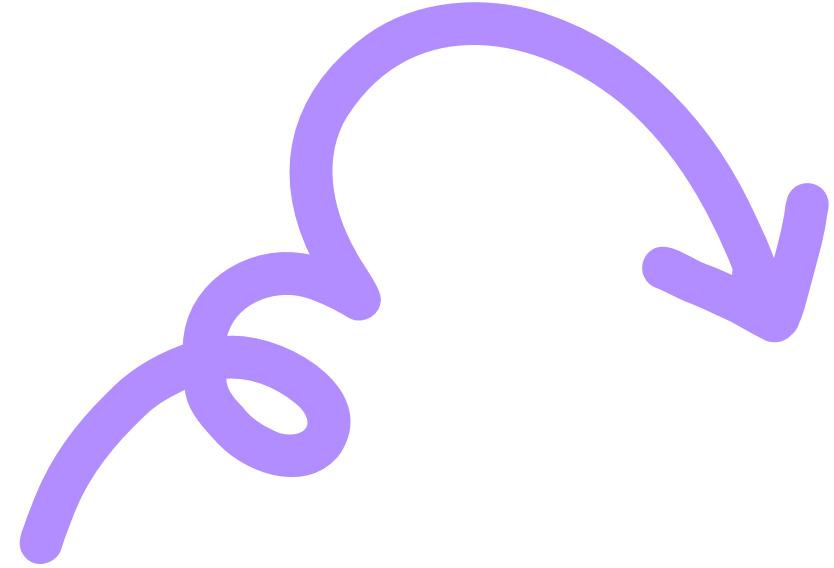
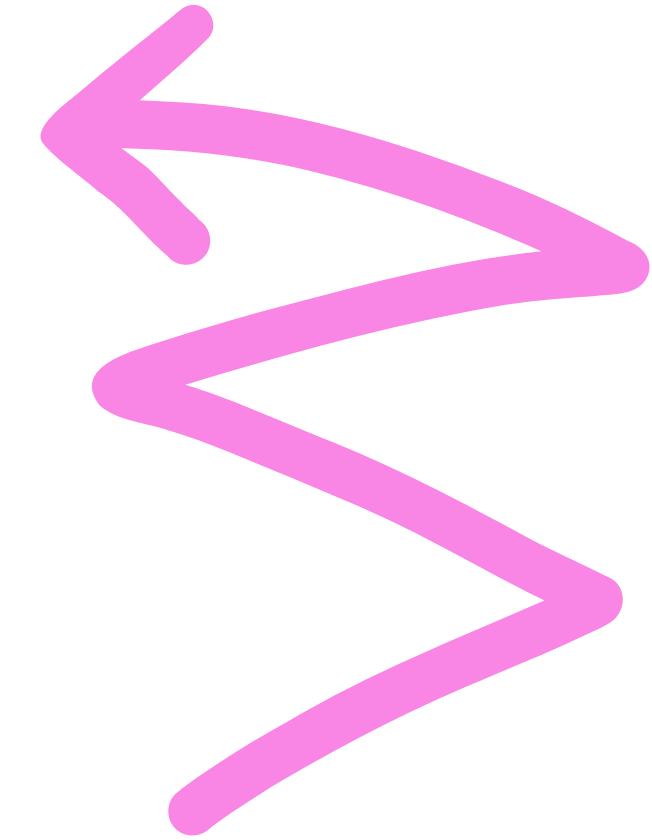
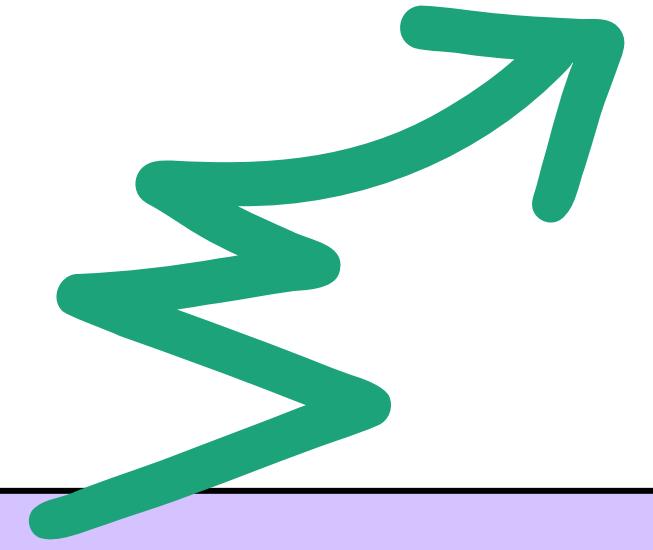


Generic LED



Basic Speaker

# Detailed design



# Model design : Detailed design

- Function LED : จะสว่างตามระดับเสียง
- Function OLED : จะแสดงสถานะของระดับเสียง



## Top Down design



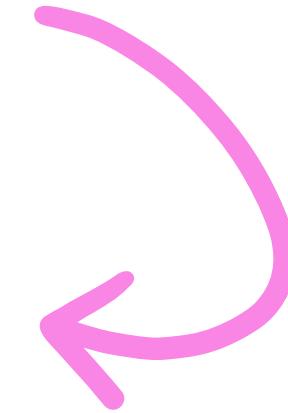
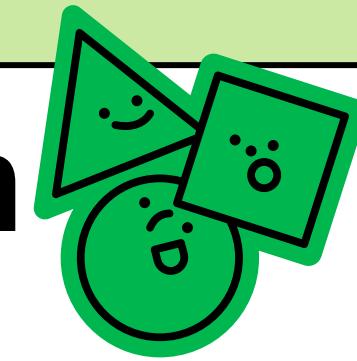
**PWM** (การควบคุม Speaker/LED)

ปรับเสียงหรือความสว่างของ LED:

- ตั้งค่าสัญญาณ PWM ตามระดับเสียงที่กำหนด
- ควบคุมการทำงานของ Speaker
- ควบคุมความสว่างของ LED (ในกรณีที่ต้องการ)



# Top Down design



## Potentiometer (การควบคุมอินพุต)

การปรับระดับเสียง:

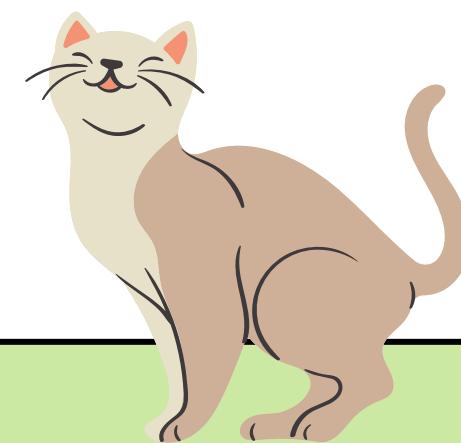
- อ่านสัญญาณอนาล็อก
- แปลงเป็นสัญญาณดิจิตอล (ADC)
- แปลงค่าให้อยู่ในช่วง 0-100% สำหรับควบคุมระดับเสียง

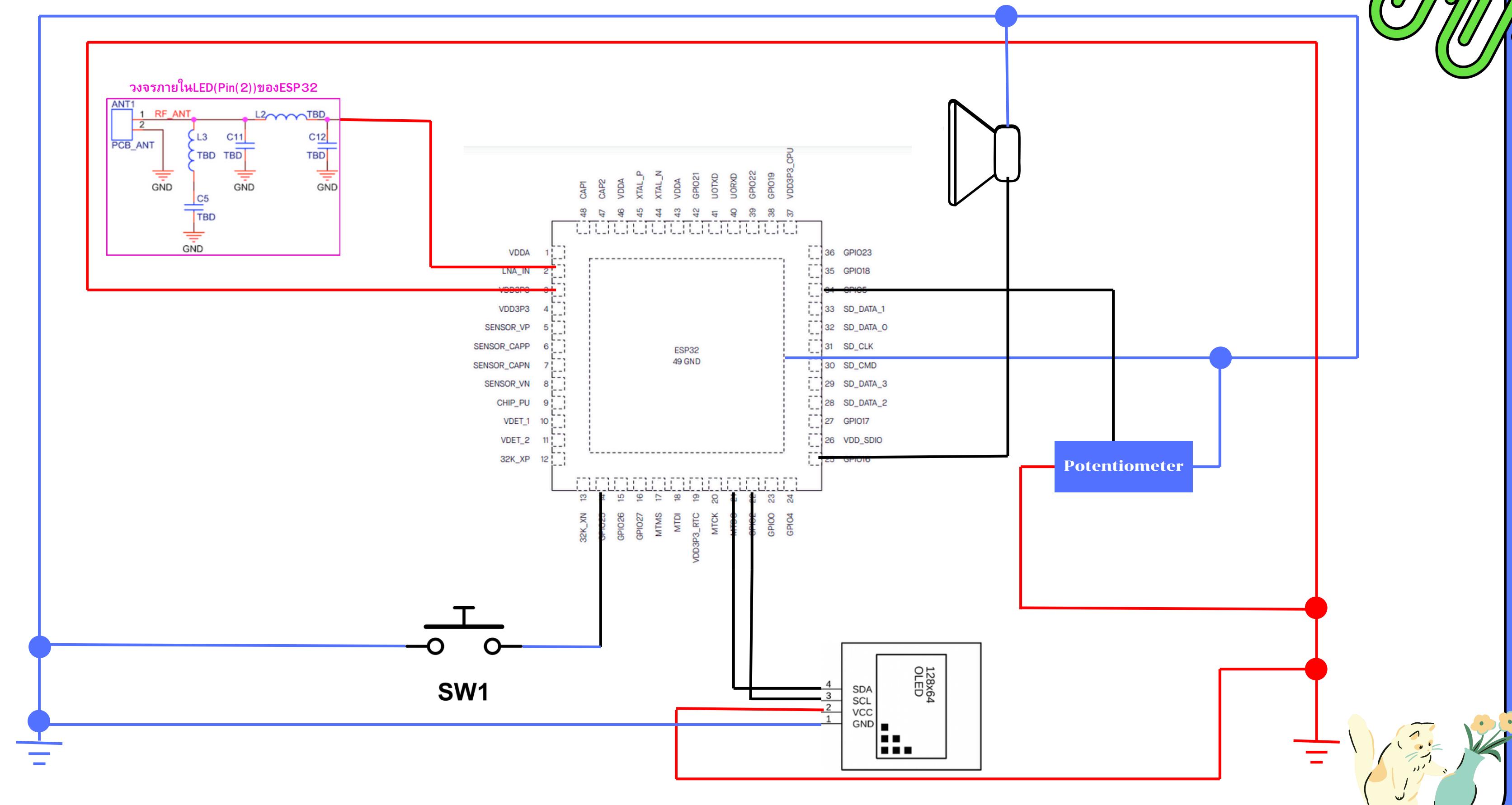


## หน้าจอ OLED (การแสดงผล)

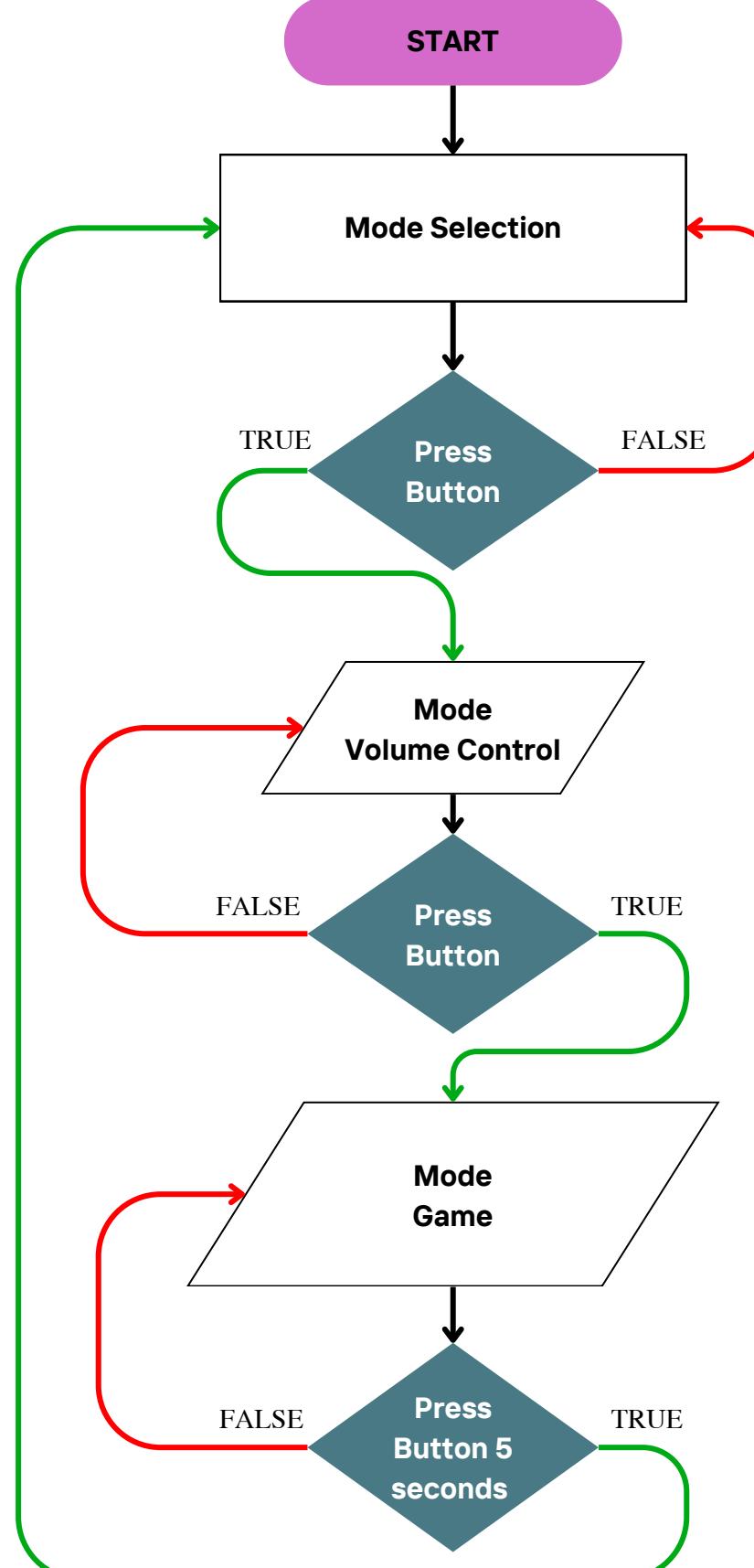
แสดงระดับเสียง:

- เริ่มต้นใช้งานライบรารี SSD1306
- แสดงระดับเสียงเป็นเปอร์เซ็นต์
- อัปเดตการแสดงผลแบบเรียลไทม์

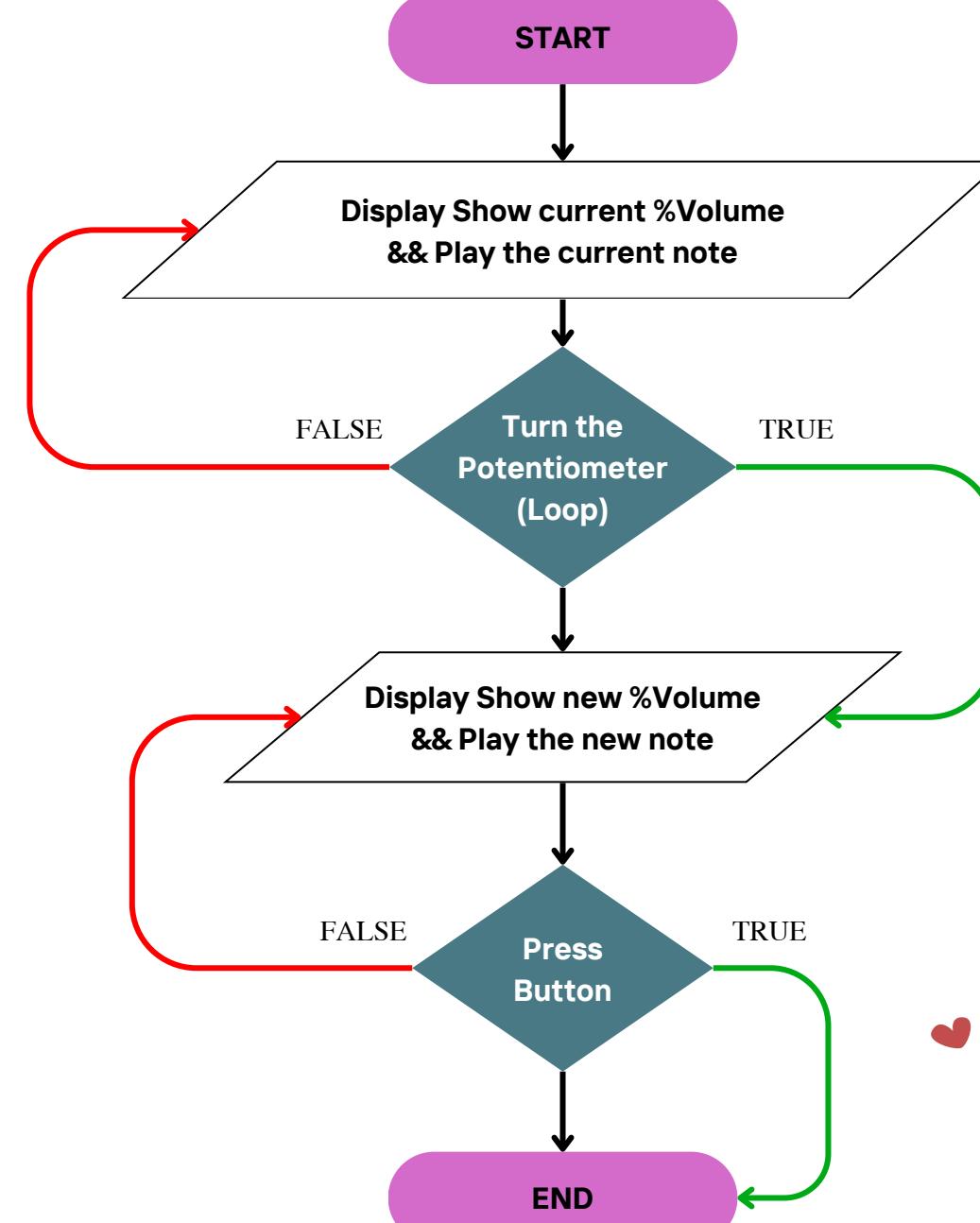




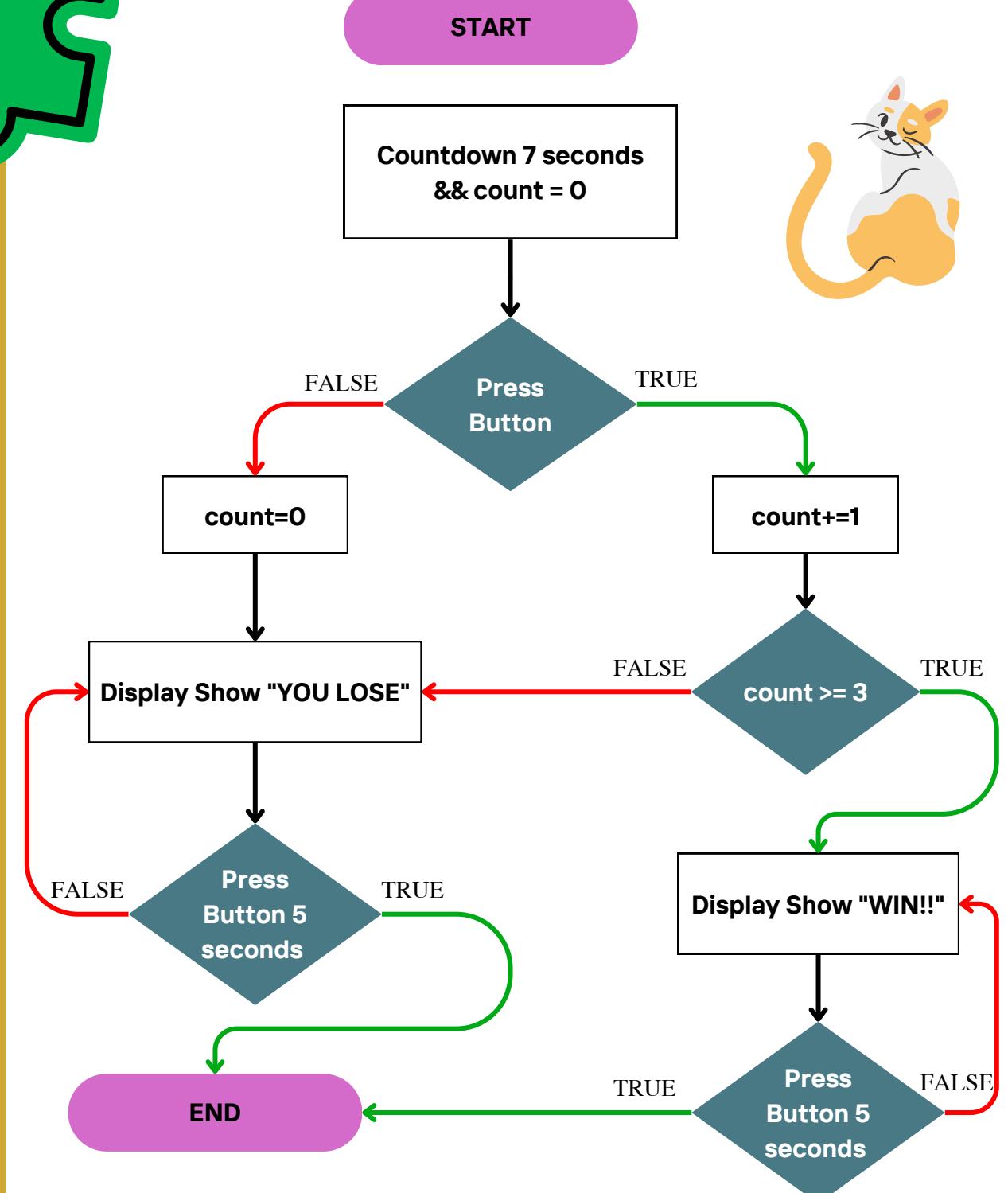
# MAIN

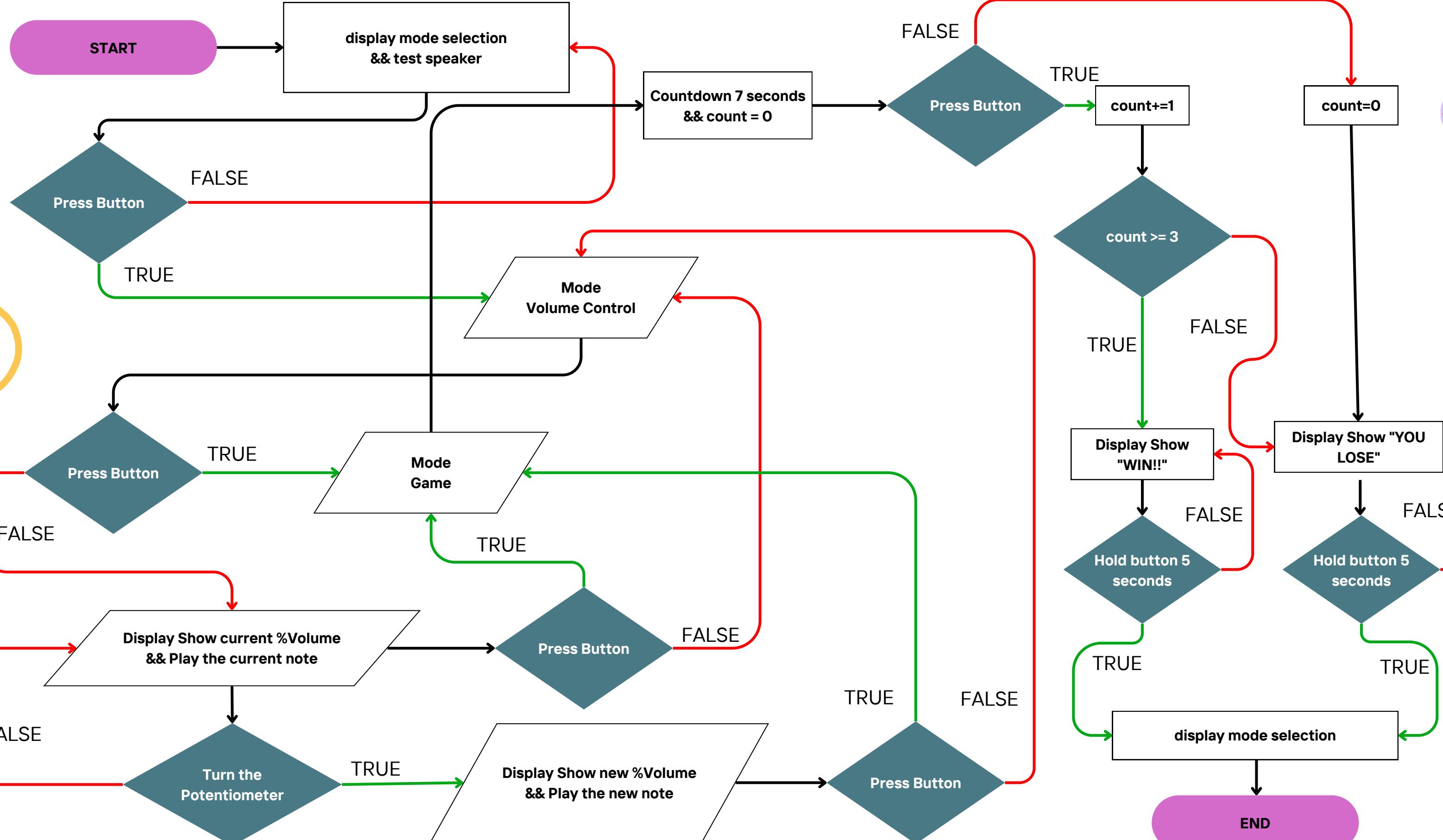


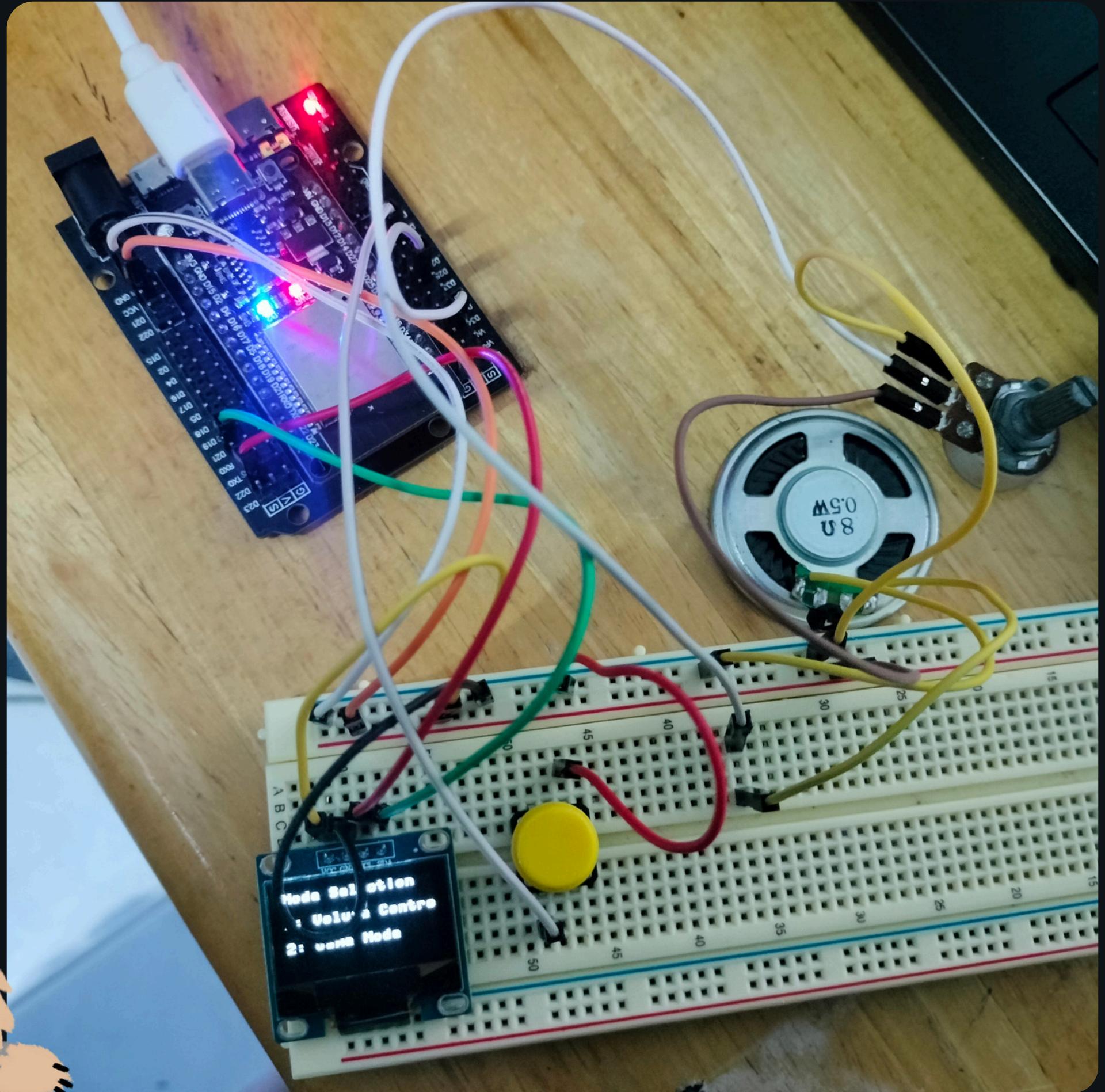
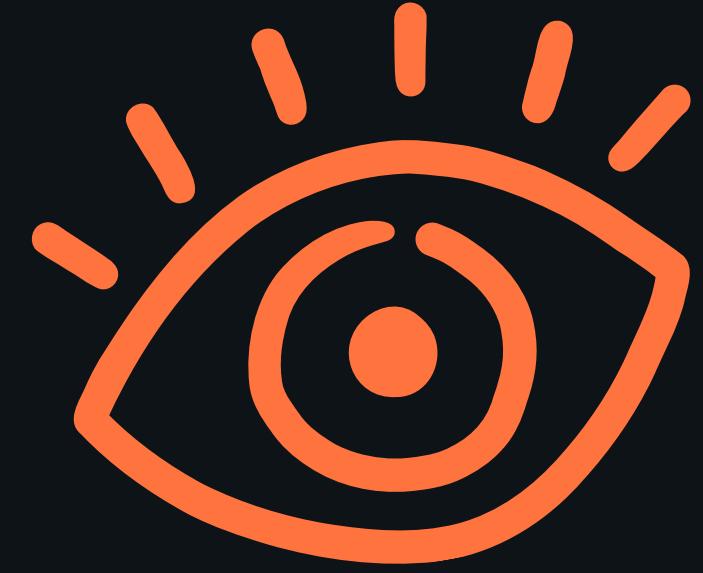
# Volume Mode

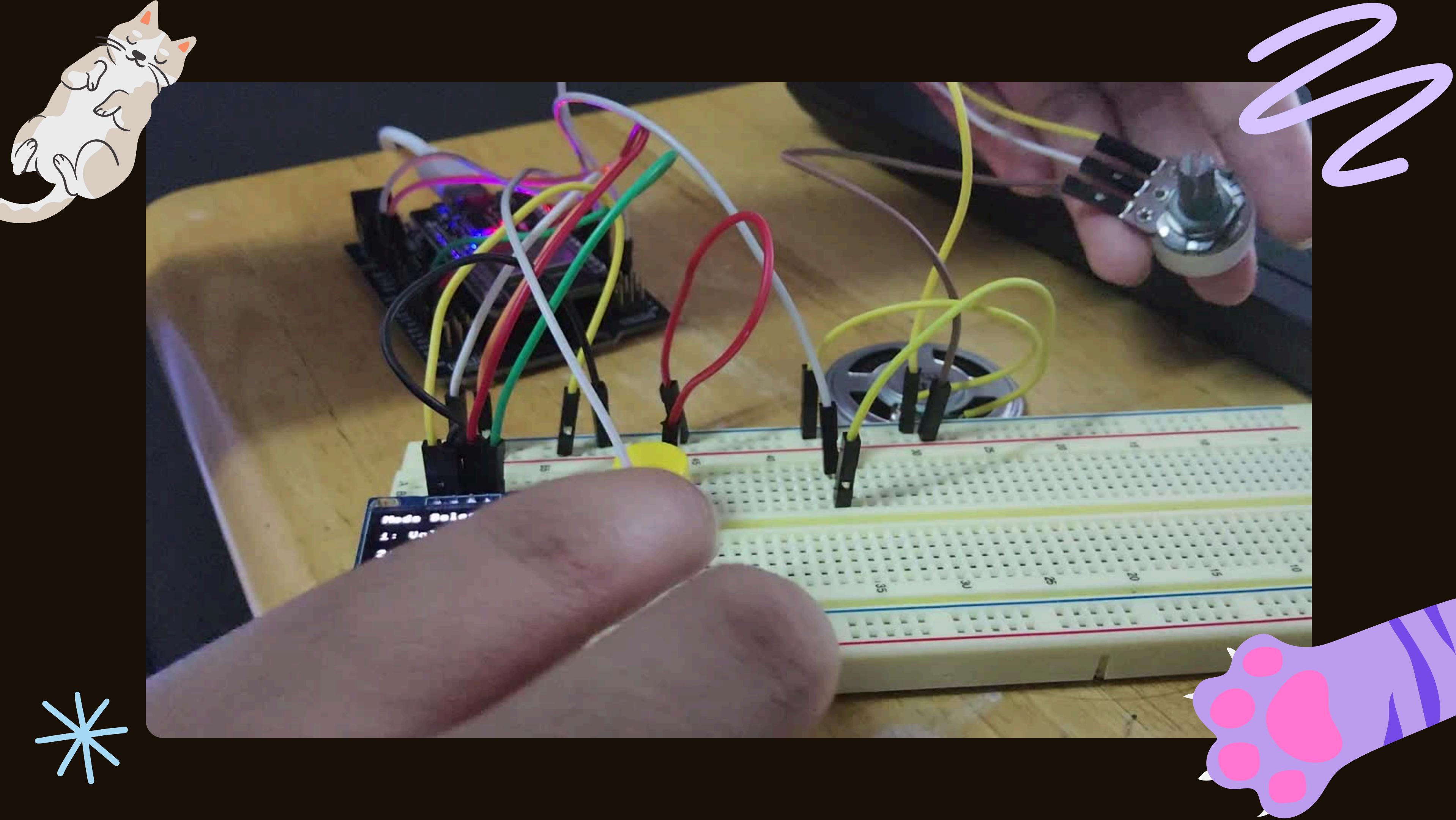


# Game Mode +







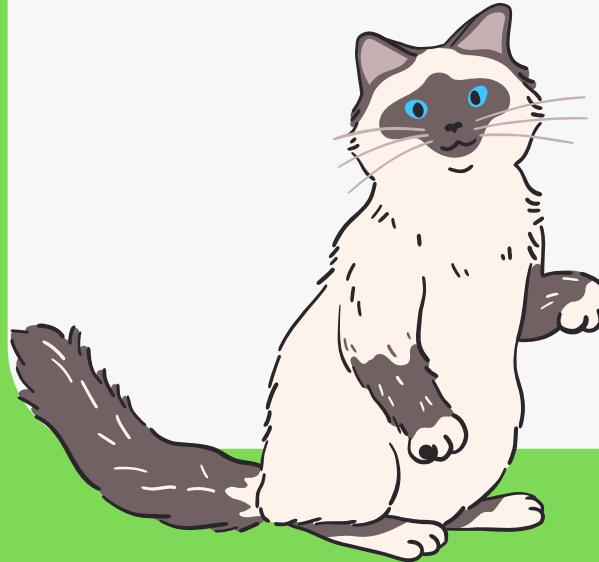




# Problem and solution

## Timer

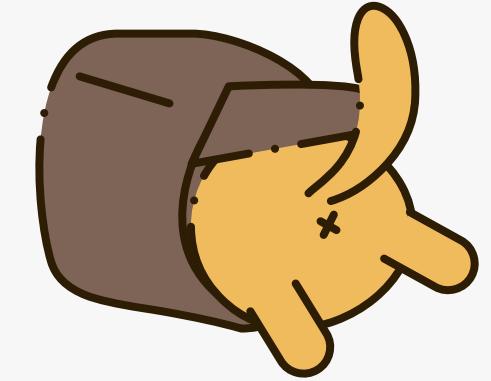
ตอนแรกเลือกใช้ timer ในการเปลี่ยนโหมดจากโหมด 2 ไปที่โหมด 0 แต่ว่ามันทำงานช้าเกินไปตั้งค่าการทำงานอยู่ที่ 5 วินาที แต่เวลาที่ใช้จริงๆ ประมาณ 8-9 วินาที



## Solution

### Time

เราจึงเปลี่ยนมาใช้โมดูล time แทน timer การทำงานของ time คือการรับค่าเวลาไปเรื่อยๆ เราจึงทำการนำตัวแปรมากำหนดเวลาเริ่มต้นที่จะบันทึกไว้ค่อยมาลบกับเวลาจริงที่ผ่านมาแล้ว



# การปรับเสียงเล่นเพลง

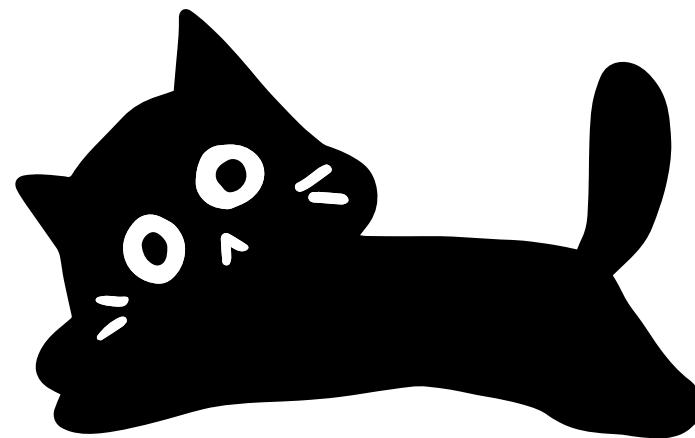
ในโค้ดตัวนี้ ต้องทำการเล่นเพลงให้จบเป็นลูปฯ ไป ก่อน speaker ถึงจะทำการปรับเสียง แต่ถ้ายังอยู่ในระหว่างที่เล่นเพลงอยู่นั้น จะยังไม่สามารถปรับเสียงได้ ณ ขณะนั้น

# ความดังเสียง

ในแต่ละตัวโน้ตนั้นจะมีบางตัวโน้ตที่จะเสียงดังกว่าตัวอื่นๆ ต้องทำการตั้ง duty cycle ให้เหมาะสมแต่ละตัวโน้ตนั้นๆ ตามความเหมาะสม



# Project gantt chart



## Project status & responsible person



## PROJECT TITLE

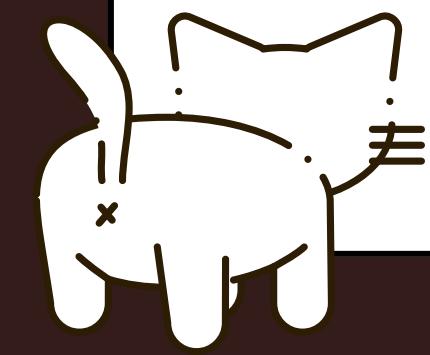
## Volume Control System

## PROJECT NAME

P'Nuang

DATE

9/30/24



# THANKS

