(Gas & Fire Alert System)

Prepared by group 4พระกาฬแต่มี3คน

Project overview

โปรเจคต์นี้ แจ้งเตือนแก๊สรั่วไหลและไฟไหม้ โดยใช้ เซนเซอร์ MQ-6 สำหรับตรวจจับแก๊สไวไฟ เซนเซอร์ MQ-2 สำหรับตรวจจับควัน ร่วมกับ DHT11 สำหรับตรวจวัดอุณหภูมิและความชื้น ระบบจะส่งสัญญาณเสียงหรือไฟแจ้งเตือนเมื่อค่าที่ตรวจวัดเกินเกณฑ์ ช่วยป้องกันอันตรายจากการระเบิดและไฟไหม้

Project detail/requirement

Project detail:

โปรเจคต์นี้เป็นระบบแจ้งเตือนแก๊สรั่วไหลและไฟไหม้สำหรับตรวจจับแก๊สไวไฟ ควัน และอุณหภูมิสูงโดยใช้

- เซนเซอร์MQ-6 ในการตรวจจับแก๊สปิโตรเลียมเหลว (LPG)
- เพนเซนเซอร์ MQ-2 ในการตรวจจับความเข้มข้นของควัน
- DHT11 สำหรับวัดอุณหภูมิและความชื้น

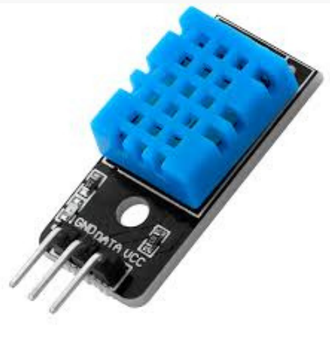
เมื่อค่าที่วัดได้เกินระดับปลอดภัย

- Buzzer จะเปิดเสียงเตือนและ เพื่อแจ้งให้ผู้ใช้ทราบ
- Oled จะแสดงค่าของแก๊ส ควัน และ อุณหภูมิ

Requirement

Hardware :

- เซนเซอร์ MQ-2 (ตรวจจับควัน)
- เซนเซอร์ MQ-6 (ตรวจจับแก๊ส)
- เซนเซอร์ DHT11 (วัดอุณหภูมิและความชื้น)
- ESP32+Shield
- Buzzer (สำหรับเสียงเตือน)
- จอOLED (สำหรับแสดงสถานะ)
- Breadboard และสาย Jumper
- แหล่งจ่ายไฟ 5V



Software :

- VS Code PlatformIO IDE (สำหรับเขียนและอัปโหลดโปรแกรม)

  Library ที่ใช้

  Arduino.h , Adafruit_GFX.h ,Adafruit_SSD1306.h

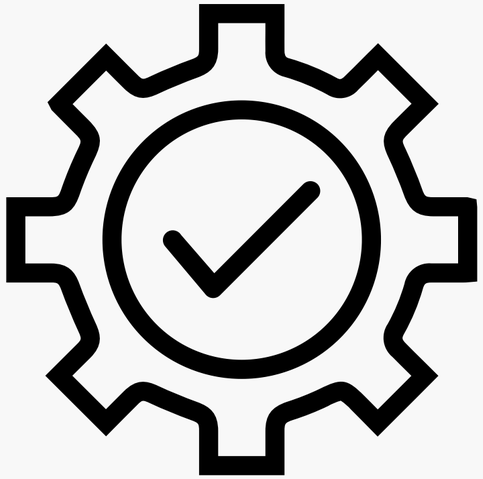
  Wire.h , driver/adc.h ,DHT.h
- Serial Monitor (ใช้ดูค่าที่อ่านจากเซนเซอร์)

Project specification

ฟังก์ชันหลัก:

อ่านค่าจาก MQ-2, MQ-6 และ DHT11 แบบเรียลไทม์

- เปรียบเทียบค่ากับเกณฑ์ที่ตั้งไว้
- แจ้งเตือนผู้ใช้ด้วย Buzzer หากค่าที่ตรวจพบเกินระดับปลอดภัย
- แสดงผลข้อมูลผ่าน Serial Monitor หรือOLED

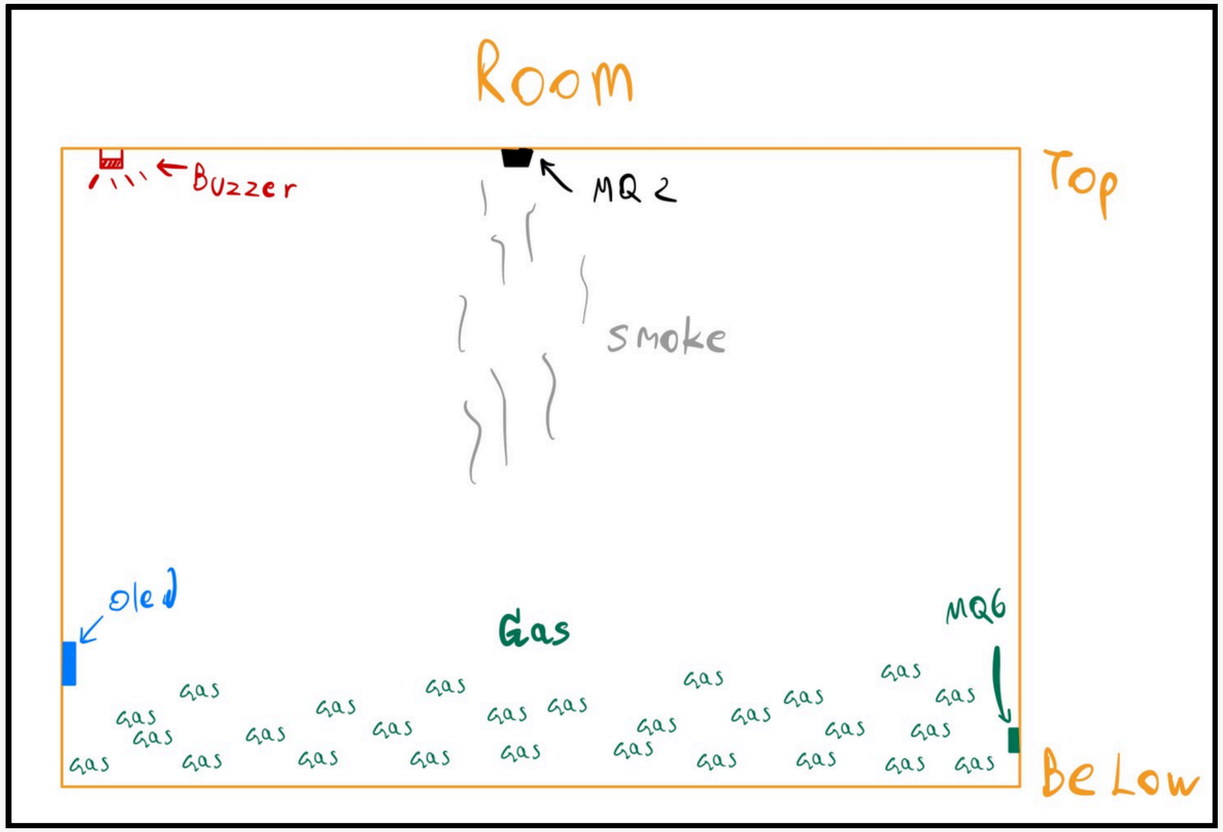


ประสิทธิภาพที่คาดหวัง:

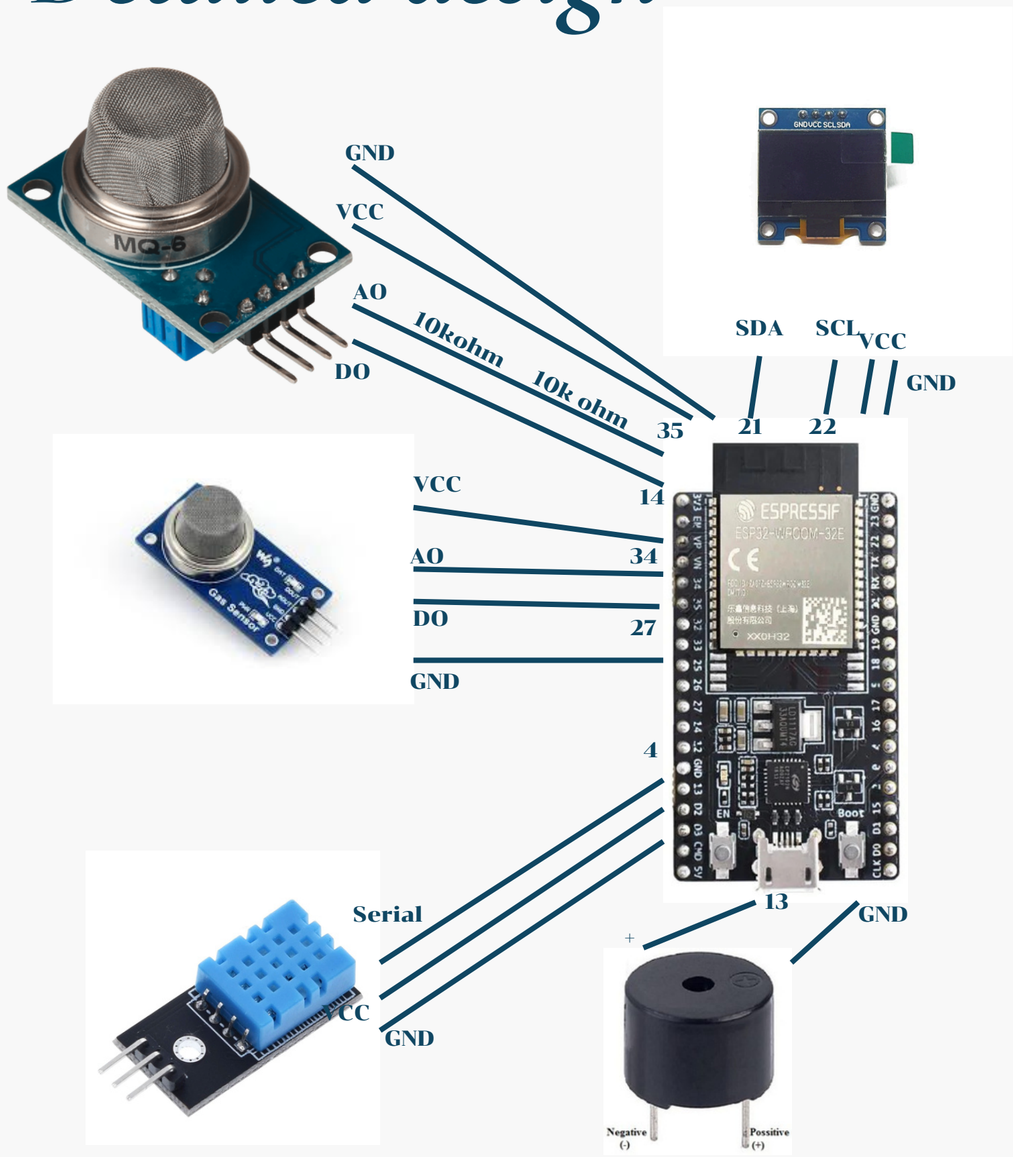
- ตอบสนองต่อค่าที่เกินเกณฑ์ภายใน 1–2 วินาที
- ความแม่นยำของเซนเซอร์เป็นไปตามสเปกของผู้ผลิต
- ระบบทำงานต่อเนื่องได้หลายชั่วโมงโดยไม่ต้องรีสตาร์ท

Architectural Design

in room



Detailed design



Input

- MQ2: ตรวจจับควัน (GPIO 34, GPIO 27)
- MQ6: ตรวจจับแก๊ส (GPIO 35, GPIO 14)
- DHT11: วัดอุณหภูมิและความชื้น (GPIO 4)

Output

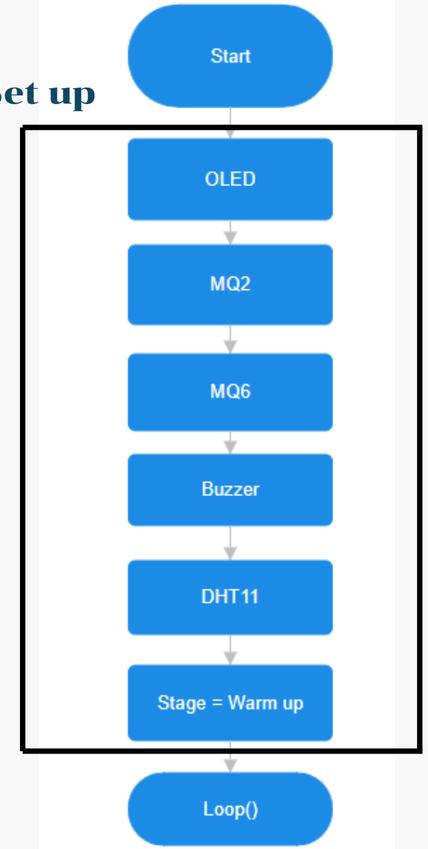
- OLED : แสดงผลสถานะและค่าต่าง ๆ (GPIO 21, GPIO 22)
- Buzzer: เตือนเมื่อค่าเกินเกณฑ์ (GPIO 13)

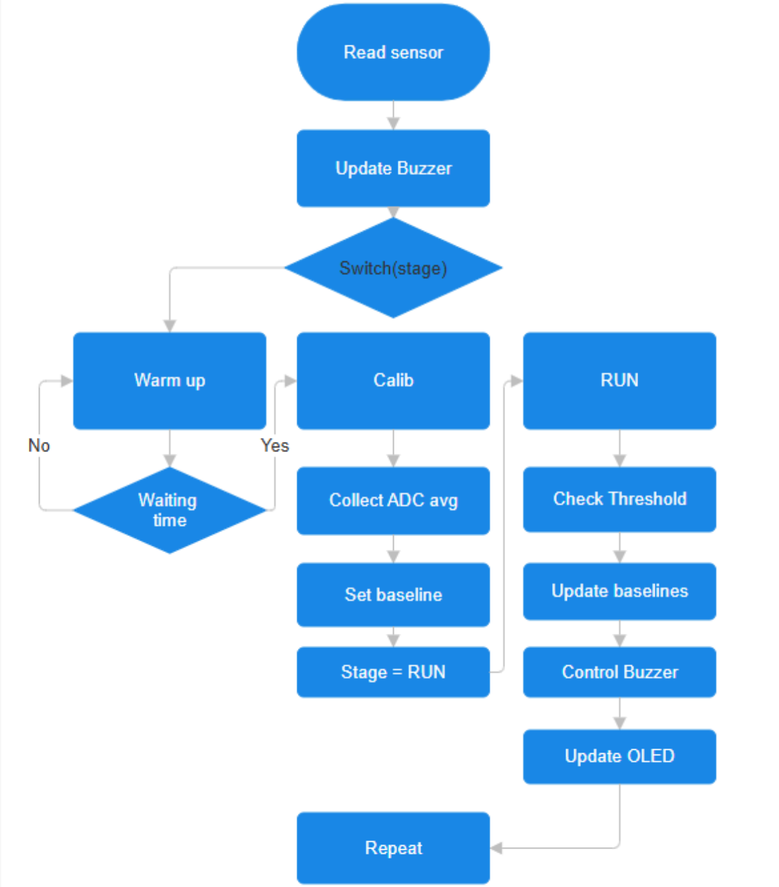
การประมวลผลเซนเซอร์

- MQ2/MQ6 อ่านค่าผ่าน AO แล้วใช้ EMA filter
  เทียบค่าปัจจุบันกับ baseline
  ใช้ DO สำหรับสัญญาณ threshold-triggered
- DHT11 ใช้ไลบรารี DHT.h
- OLED ใช้ Adafruit_SSD1306 และ Adafruit_GFX
- Buzzer ใช้ PWM
  เสียงตามที่กำหนด

Flowchart

Set up





Code Blame 404 lines (369 loc) · 9.36 KB

```ino
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "driver/adc.h"
#include <DHT.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_ADDR 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

const int MQ2_AO_PIN = 34;
const int MQ2_DO_PIN = 27;
const int MQ6_AO_PIN = 35;
const int MQ6_DO_PIN = 14;
const int BUZZER_PIN = 13;
const int BUZZ_CH = 0;

#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

const bool USE_DO_MQ2 = true;
const bool USE_DO_MQ6 = true;
const bool USE_BUZZER = true;

const unsigned long WARMUP_MS = 180000UL;
const unsigned long CALIB_MS = 20000UL;
const unsigned long SAMPLE_PERIOD_MS = 250UL;
const unsigned long REPORT_PERIOD_MS = 1000UL;
const unsigned long DHT_PERIOD_MS = 3000UL;

const float EMA_ALPHA = 0.05f; // หนืดขึ้นเพื่อลดแกว่ง

const float RISE_MQ2 = 1.25f, FALL_MQ2 = 1.12f;
const float RISE_MQ6 = 1.40f, FALL_MQ6 = 1.18f;

// หน่วงเวลา/ค้างสถานะ
const unsigned long ASSERT_HOLD_MS = 1500;
const unsigned long CLEAR_HOLD_MS = 7000;
const unsigned long MIN_ON_MS = 3000;

enum Stage
{
  WARMUP,
```

Code

1. อุปกรณ์และเซนเซอร์

MQ-2 (ควัน/แก๊สไวไฟ)

a. อ่านค่า Analog (AO) และ Digital (DO)

b. ใช้ในการตรวจจับควันหรือแก๊สไวไฟ

MQ-6 (แก๊ส LPG/แอลกอฮอล์)

a. อ่านค่า Analog (AO) และ Digital (DO)

b. ใช้ตรวจจับแก๊สรั่ว

DHT11 (อุณหภูมิ/ความชื้น)

a. อ่านอุณหภูมิเพื่อแสดงบน OLED

OLED 128x64

a. แสดงสถานะ: SAFE, SMOKE, GAS, SMK+GAS และค่าอุณหภูมิ

Buzzer

a. ส่งเสียงเตือนแบบ on/off เป็นช่วง (non-blocking)

```
// --- Stage ---
enum Stage
{
  WARMUP,
  CALIB,
  RUN
};
Stage stage = WARMUP;
unsigned long t_start = 0, t_lastSamp = 0;

// --- EMA / baseline ---
float ema2 = NAN, base2 = NAN;
float ema6 = NAN, base6 = NAN;
const float EMA_ALPHA = 0.05f; // ลดแกว่ง

// --- Digital output debounce ---
volatile bool do2_trig = false, do6_trig = false;
volatile unsigned long do2_last = 0, do6_last = 0;
const unsigned long DO_DEBOUNCE_MS = 50;

// --- ฟังก์ชันอ่าน ADC และ EMA ---
int readADC(int pin) { return analogRead(pin); }

void emaUpdate(float x, float &e)
{
  if (isnan(e))
    e = x;
  else
    e = EMA_ALPHA * x + (1 - EMA_ALPHA) * e;
}

// --- ISR สำหรับ Digital output ---
void IRAM_ATTR isr_mq2()
{
  unsigned long now = millis();
  if (now - do2_last >= DO_DEBOUNCE_MS)
  {
    do2_trig = true;
    do2_last = now;
  }
}

void IRAM_ATTR isr_mq6()
{
  unsigned long now = millis();
  if (now - do6_last >= DO_DEBOUNCE_MS)
  {
    do6_trig = true;
    do6_last = now;
  }
}

}
```

```
// --- ส่วน loop อ่านค่าเซนเซอร์ ---
void loop()
{
 unsigned long now = millis();

 if (now - t_lastSamp >= SAMPLE_PERIOD_MS)
 {
  t_lastSamp = now;

  // อ่านค่าอนาล็อก
  int adc2 = readADC(MQ2_AO_PIN);
  int adc6 = readADC(MQ6_AO_PIN);

  // EMA เพื่อลดแกว่ง
  emaUpdate((float)adc2, ema2);
  emaUpdate((float)adc6, ema6);

  // --- Stage ---
  switch (stage)
  {
  case WARMUP:
   if (now - t_start >= WARMUP_MS)
   {
    ema2 = NAN;
    ema6 = NAN;
    t_start = now;
    stage = CALIB;
    Serial.println("Stage: CALIB...");
   }
   break;

  case CALIB:
   if (now - t_start >= CALIB_MS)
   {
    base2 = ema2; if (base2 < 1) base2 = 1;
    base6 = ema6; if (base6 < 1) base6 = 1;
    stage = RUN;
    Serial.printf("Stage: RUN (base2=%.1f, base6=%.1f)\n", base2, base6);
   }
   break;

  case RUN:
   // คำนวณ ratio r2, r6
   float r2 = (!isnan(ema2) && base2 > 0) ? (ema2 / base2) : 1.0f;
   float r6 = (!isnan(ema6) && base6 > 0) ? (ema6 / base6) : 1.0f;
   // อ่าน DO
   bool do2_low = (digitalRead(MQ2_DO_PIN) == LOW);
   bool do6_low = (digitalRead(MQ6_DO_PIN) == LOW);
   do2_trig = do6_trig = false;
   break;
  }
 }
}
```

Code

2. โหมดการทำงาน (Stage)
   - WARMUP – รอให้เซนเซอร์พร้อมใช้งาน (3 นาที)
   - CALIB – เก็บค่า baseline ของ MQ-2 และ MQ-6 (20 วินาที)
   - RUN – เริ่มตรวจจับควันและแก๊สจริง
3. การอ่านค่าและประมวลผล
   - Analog EMA
     - ใช้ emaUpdate() ลดการแกว่งของค่าเซนเซอร์
     - เปรียบเทียบกับค่า baseline เพื่อคำนวณ ratio r2, r6
   - Digital DO
     - ใช้ interrupt และ debounce เพื่อตรวจจับเหตุการณ์เร็ว

```
// --- ฟังก์ชันแสดงผล OLED ---
void drawUI(bool smoke_, bool gas_, float tempC)
{
  display.clearDisplay();

  // กำหนดข้อความตามสถานะ
  bool both = smoke_ && gas_;
  const char *msg = both ? "SMK+GAS" : (smoke_ ? "SMOKE" : (gas_ ? "GAS" : "SAFE"));

  // เปลี่ยนสีพื้นหลังหากมีเหตุการณ์
  if (smoke_ || gas_)
  {
    display.fillRect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT, SSD1306_WHITE);
    display.setTextColor(SSD1306_BLACK);
  }
  else
  {
    display.setTextColor(SSD1306_WHITE);
  }

  // ขนาดตัวอักษร
  display.setTextSize(both ? 2 : 3);
  int16_t x1, y1;
  uint16_t w, h;
  display.getTextBounds(msg, 0, 0, &x1, &y1, &w, &h);
  int x = (SCREEN_WIDTH - (int)w) / 2, y = 8;
  display.setCursor(x, y);
  display.print(msg);

  // แสดงอุณหภูมิ
  display.setTextSize(1);
  char line[24];
  int y2 = 48;
  if (isnan(tempC))
    snprintf(line, sizeof(line), "Temp: --.- C");
  else
    snprintf(line, sizeof(line), "Temp: %.1f C", tempC);
  display.setCursor(8, y2);
  display.print(line);

  display.display();
}
```

```
// --- ฟังก์ชันควบคุม Buzzer แบบ non-blocking ---
void buzzerStop()
{
  if (!USE_BUZZER) return;
  ledcWrite(BUZZ_CH, 0);
  buzzer_on = false;
}

void buzzerTick(bool alarm)
{
  if (!USE_BUZZER) return;

  if (!alarm) // ไม่มีเหตุการณ์
  {
    buzzerStop();
    return;
  }

  unsigned long now = millis();
  if (!buzzer_on) // เริ่ม buzzer
  {
    ledcWrite(BUZZ_CH, BUZZ_DUTY);
    buzzer_on = true;
    t_buzz = now;
  }
  else
  {
    // สลับ ON/OFF ตาม BUZZ_ON_MS / BUZZ_OFF_MS
    if ((ledcRead(BUZZ_CH) > 0) && (now - t_buzz >= BUZZ_ON_MS))
    {
      ledcWrite(BUZZ_CH, 0);
      t_buzz = now;
    }
    else if ((ledcRead(BUZZ_CH) == 0) && (now - t_buzz >= BUZZ_OFF_MS))
    {
      ledcWrite(BUZZ_CH, BUZZ_DUTY);
      t_buzz = now;
    }
  }
}

}
```

Code

5. การแสดงผล
- drawUI()
- แสดงสถานะ SAFE/SMOKE/GAS/SMK+GAS
- แสดงอุณหภูมิล่าสุดจาก DHT11

6. Buzzer
- ทำงานแบบ non-blocking
- เปิด/ปิดเป็นช่วงตาม BUZZ_ON_MS / BUZZ_OFF_MS
- หยุดเมื่อไม่มีสถานะควันหรือแก๊ส

Code

```
// --- ฟังก์ชันเสริม ---
int readADC(int pin) {
  return analogRead(pin); // อ่านค่าอนาล็อกจากขาเซนเซอร์
}

void emaUpdate(float x, float &e) {
  if (isnan(e))
    e = x;        // เริ่มค่า EMA
  else
    e = EMA_ALPHA * x + (1 - EMA_ALPHA) * e; // ปรับค่า EMA เพื่อลดแกว่ง
}

void buzzerStop() {
  if (!USE_BUZZER) return;
  ledcWrite(BUZZ_CH, 0);  // ปิดเสียง buzzer
  buzzer_on = false;
}

void buzzerTick(bool alarm) {
  if (!USE_BUZZER) return;

  if (!alarm) {
    buzzerStop();    // ปิด buzzer ถ้าไม่มี alarm
    return;
  }

  unsigned long now = millis();
  if (!buzzer_on) {
    ledcWrite(BUZZ_CH, BUZZ_DUTY);
    buzzer_on = true;
    t_buzz = now;
  } else {
    if ((ledcRead(BUZZ_CH) > 0) && (now - t_buzz >= BUZZ_ON_MS)) {
      ledcWrite(BUZZ_CH, 0);
      t_buzz = now;
    } else if ((ledcRead(BUZZ_CH) == 0) && (now - t_buzz >= BUZZ_OFF_MS)) {
      ledcWrite(BUZZ_CH, BUZZ_DUTY);
      t_buzz = now;
    }
  }
}
```

```
// --- ตัวอย่างหลักการทำงานใน loop() ---
switch(stage) {
 case WARMUP: // 1. รอเซนเซอร์พร้อมใช้งาน
 if (millis() - t_start >= WARMUP_MS) {
 ema2 = ema6 = NAN;
 t_start = millis();
 stage = CALIB;
 }
 break;
 case CALIB: // 2. เก็บ baseline
 if (millis() - t_start >= CALIB_MS) {
 base2 = (ema2 < 1) ? 1 : ema2;
 base6 = (ema6 < 1) ? 1 : ema6;
 stage = RUN;
 }
 break;
 case RUN: // 3. เริ่มตรวจจับจริง
 float r2 = (!isnan(ema2) && base2>0) ? ema2/base2 : 1.0f;
 float r6 = (!isnan(ema6) && base6>0) ? ema6/base6 : 1.0f;

 bool hazard2 = (r2 >= RISE_MQ2) || (USE_DO_MQ2 && digitalRead(MQ2_DO_PIN)==LOW);
 bool quiet2 = (r2 < FALL_MQ2) && (!USE_DO_MQ2 || digitalRead(MQ2_DO_PIN)==HIGH);
 bool hazard6 = (r6 >= RISE_MQ6) || (USE_DO_MQ6 && digitalRead(MQ6_DO_PIN)==LOW);
 bool quiet6 = (r6 < FALL_MQ6) && (!USE_DO_MQ6 || digitalRead(MQ6_DO_PIN)==HIGH);

 // 4. อัปเดตสถานะ smoke / gas พร้อมหน่วงเวลา
 // ... (โค้ดอัปเดต smoke/gas ตาม ASSERT_HOLD_MS, CLEAR_HOLD_MS, MIN_ON_MS)

 // 5. แสดงผล OLED + buzzer
 bool any_new = smoke || gas;
 buzzerTick(any_new);
 drawUI(smoke, gas, last_tempC);

 // 6. ปรับ baseline อัตโนมัติทุก 5 นาที หากไม่มี alarm
 if (!any_new && millis() - t_lastRebase >= SAFE_REBASE_INTERVAL_MS) {
 base2 = 0.9f*base2 + 0.1f*ema2;
 base6 = 0.9f*base6 + 0.1f*ema6;
 t_lastRebase = millis();
 }
 break;
```

7. ฟังก์ชันเสริม

- readADC() – อ่านค่าอนาล็อกจากขาเซนเซอร์
- emaUpdate() – ปรับค่า EMA ของเซนเซอร์
- buzzerStop() – ปิดเสียง buzzer
- buzzerTick() – ควบคุม buzzer ตามสถานะ alarm

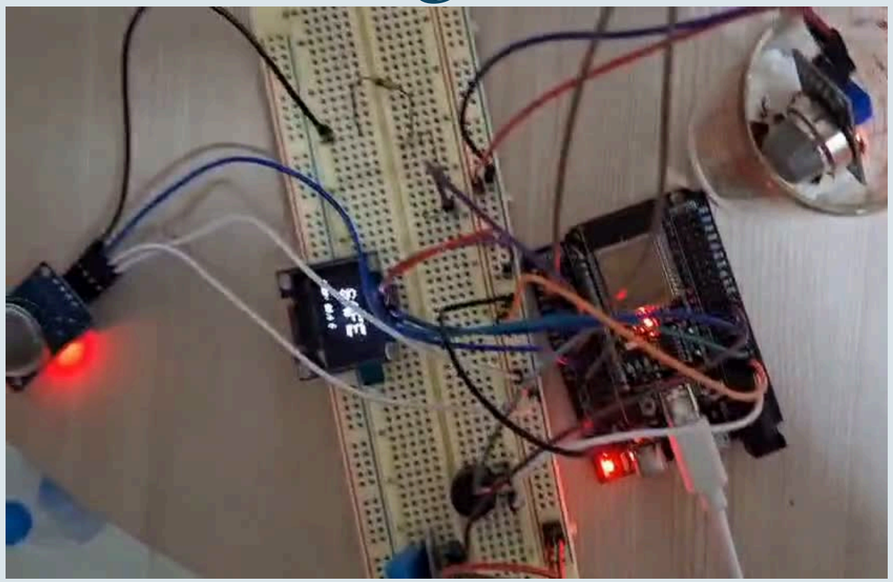
---

8. สรุปหลักการทำงาน

1. ESP32 เริ่ม WARMUP → CALIB → RUN
2. อ่านค่า MQ-2 / MQ-6 / DHT11
3. ตรวจสอบ hazard / quiet พร้อมหน่วงเวลา
4. อัปเดตสถานะ smoke / gas
5. แสดงผลบน OLED และ ส่งเสียงเตือน หากมีเหตุการณ์
6. ค่า baseline ปรับอัตโนมัติทุก 5 นาที ถ้าไม่มีสถานะเตือน

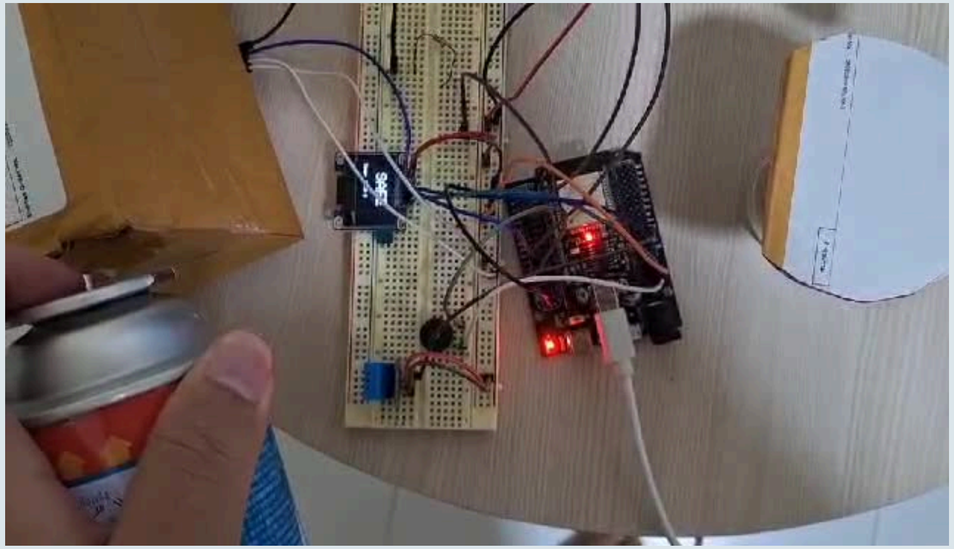
ตามนี้

Video Testing and result (all cases)

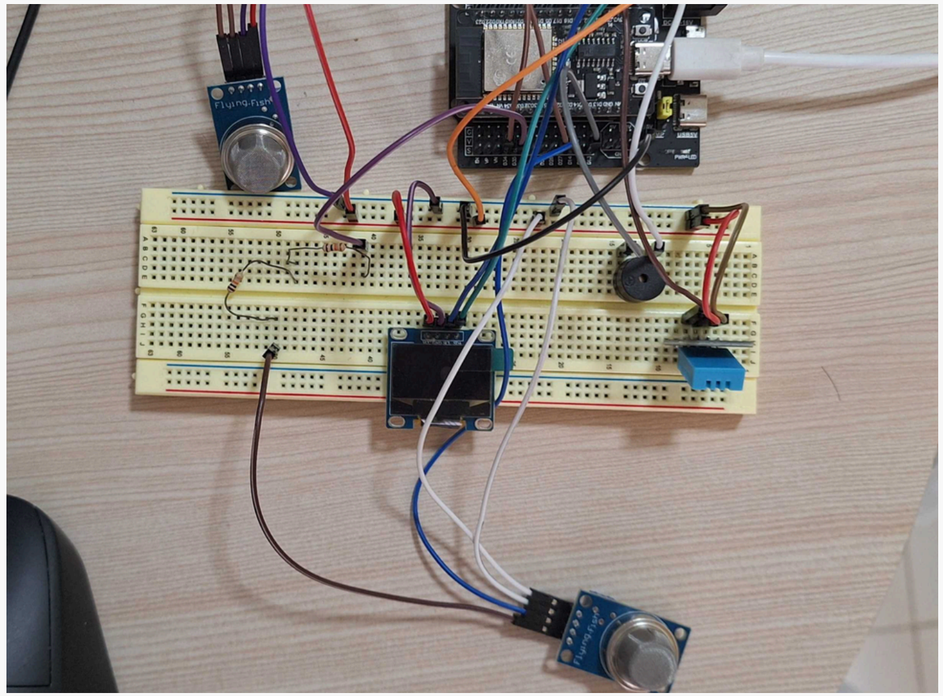


smoke

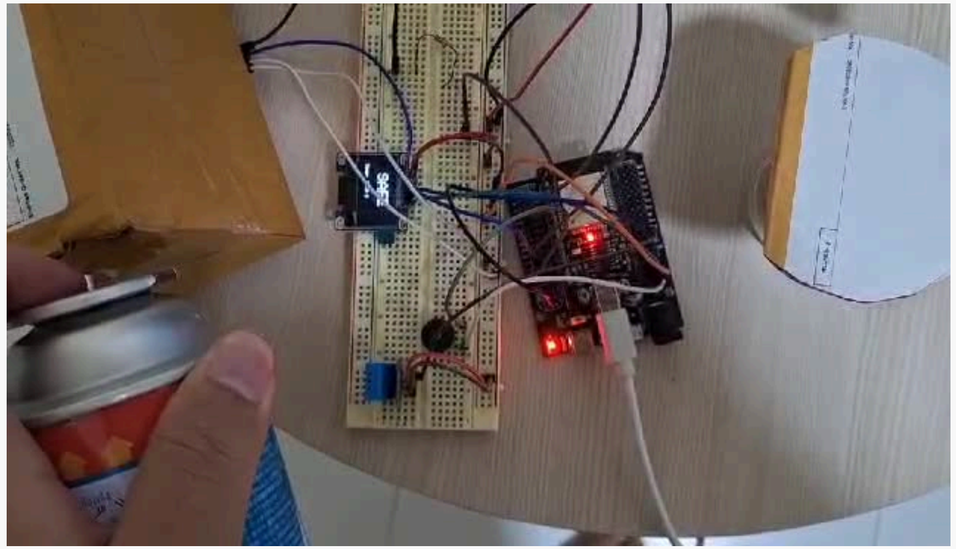
Gas



Picture of your project



Video



Problem and solution

บอร์ดร้อน / เซนเซอร์ร้อน

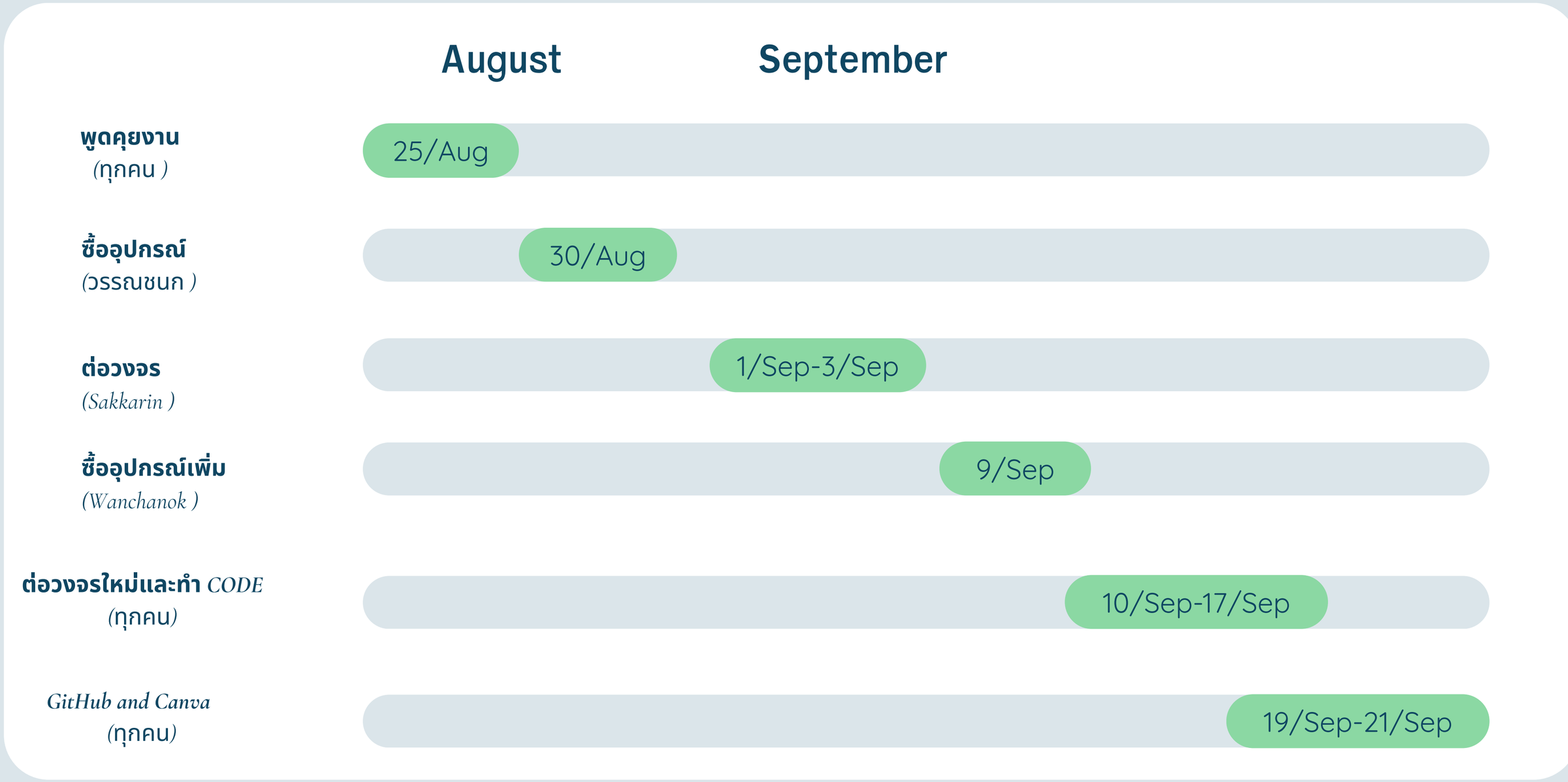
วิธีแก้ปัญหา

- วางบอร์ดและเซนเซอร์ในที่มีอากาศเย็น
- วางเซนเซอร์ห่างจากแหล่งความร้อนอื่น

- เพิ่ม debounce
- EMA → ปรับให้ ช้าลง เช่น 0.05
  EMA ช้า → sensor ลดค่าเมื่อแก๊สหาย
- เพิ่ม resistor 2ตัว ต่อเข้ากับขา AO ของMQ-6 เพื่อลดแรงดัน

Gantt chart

(Gas & Fire Alert System)



Thank you