Doraemon Group

Home Content Contact

presenting...

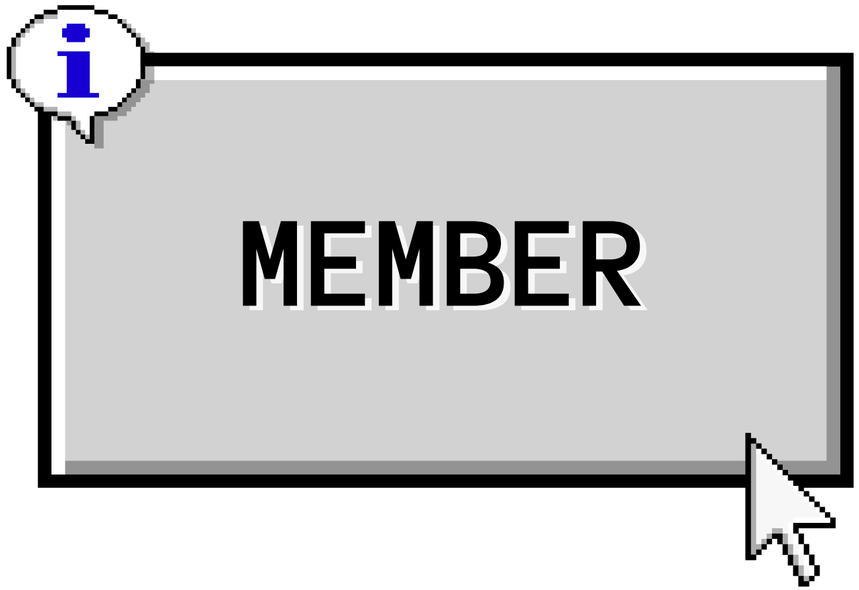
SPACE SHOOTER GAME

CPE-213 PROJECT

Start

Doraemon Group

Home **Content** Contact



Chinapat Suphanapong
id: 2311311480

Chisanupong Sukati
id: 2311310508

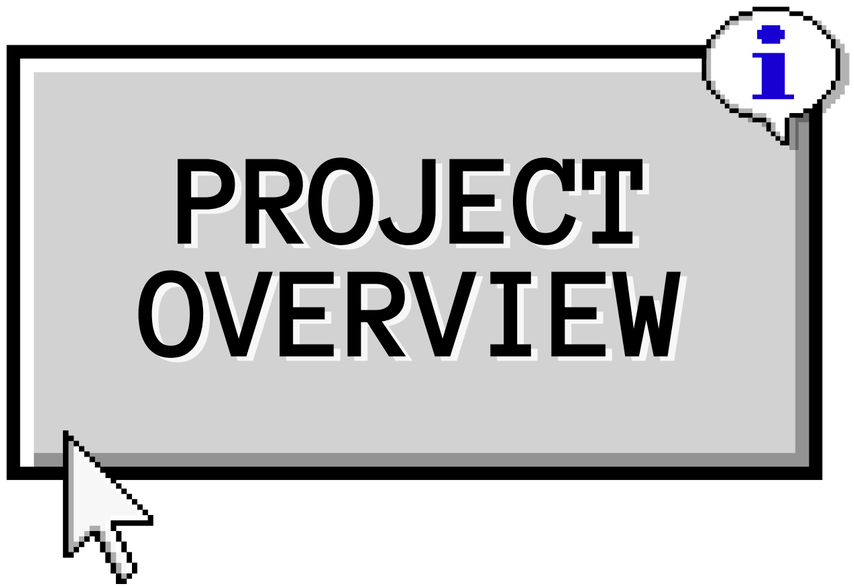
Doraemon Group

Home **Content** Contact

This project is a console-based space shooter game implemented on ESP32.
It uses:

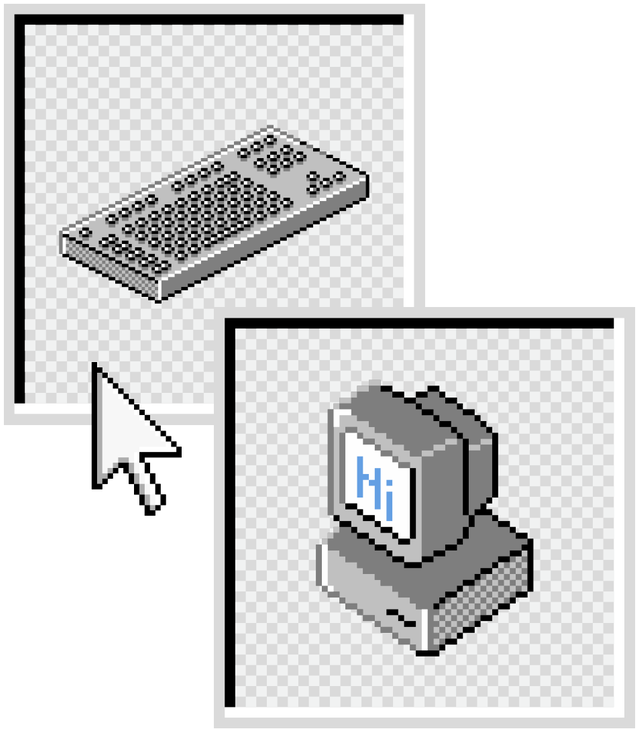
- Potentiometer to control the spaceship movement.
- Button to shoot bullets.
- Two ultrasonic sensors to detect a person within 10 cm — the game only runs if a player is detected.
- LED indicators to display lives remaining.

The game logic runs on the ESP32 and is displayed via the Serial Monitor.



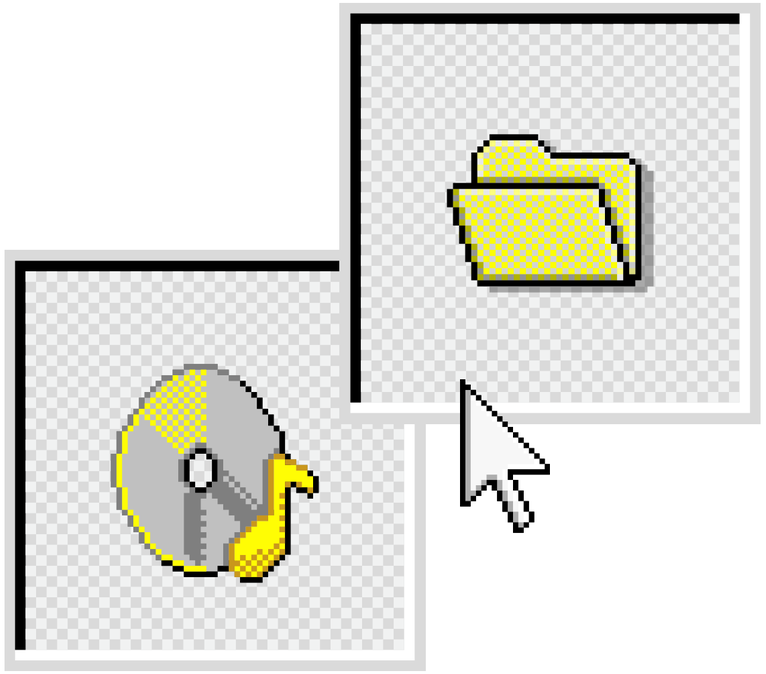
PROJECT DETAIL/REQUIREMENT

- Hardware: ESP32, Ultrasonic Sensor ×2, Potentiometer, Push button, LED ×3, Resistors
- Software: Arduino IDE
- Requirement:
  - เกมต้องแสดงผลได้ใน Serial Monitor
  - ผู้เล่นสามารถควบคุมยานซ้าย–ขวา และยิงกระสุนได้
  - ศัตรู (asteroids) ต้องตกลงมาและมีหลายพฤติกรรม
  - ระบบตรวจจับคนทำให้เกม pause/resume ได้



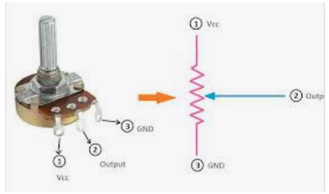
Doraemon Group

Home **Content** Contact



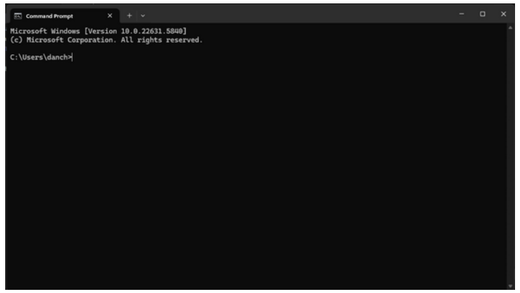
PROJECT SPECIFICATION

- Serial Monitor แสดงพื้นที่เกม 20×10 ช่อง
- ความเร็วเกมปรับตามคะแนน (ยิ่งเล่นนาน ยิ่งเร็ว)
- กระสุนยิงได้หลายลูกพร้อมกัน (สูงสุด 5 ลูก)
- อุกกาบาตสูงสุด 10 ลูกพร้อมกัน
- LED 3 สี แสดงชีวิต (3 → เขียว, 2 → เหลือง, 1 → แดง)
- Bullet cooldown = 200 ms



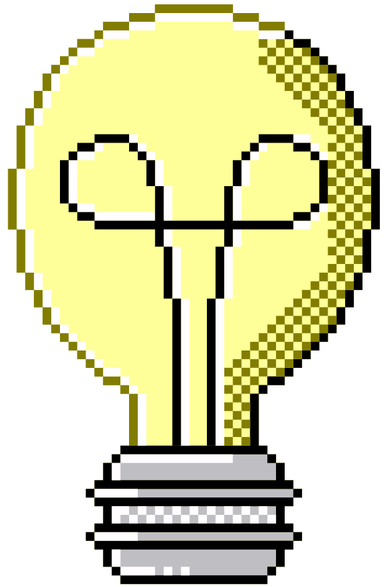
ARCHITECTURAL DESIGN

Input → Potentiometer, Button, Ultrasonic sensors
Processing → ESP32 (C++ game logic)
Output → Serial Monitor (game screen), LEDs (life status)



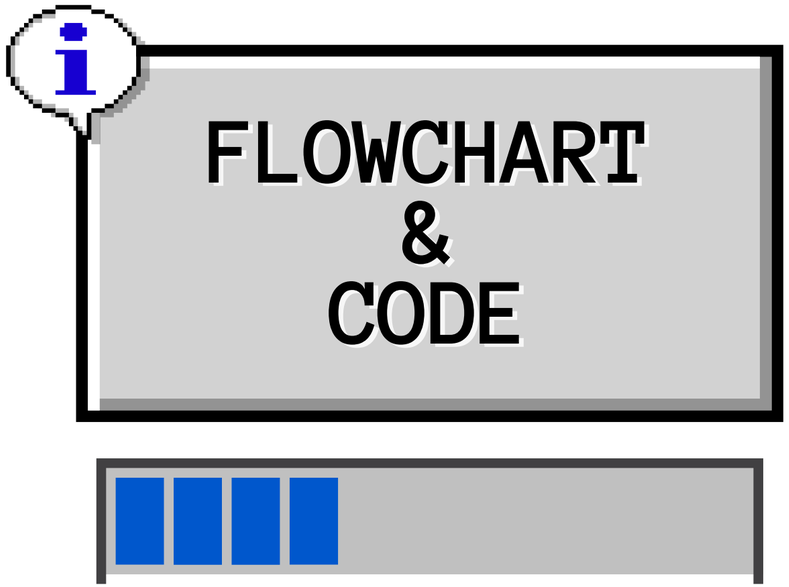
DETAILED DESIGN

- Ship movement: อ่านค่า analog จาก potentiometer แล้ว map ให้ตรงกับตำแหน่งบนหน้าจอ
- Shooting: เมื่อกดปุ่ม → ยิงกระสุนใหม่ (ถ้า cooldown ผ่านแล้ว)
- Asteroids: มี 3 ประเภท (Normal, Zigzag, Fast)
- Collision detection: ตรวจว่ากระสุนตรงกับอุกกาบาต หรืออุกกาบาตชนยาน → อัปเดตคะแนน/ลดชีวิต
- Game state: เมื่อไม่มีคนในระยะ → เกมหยุด



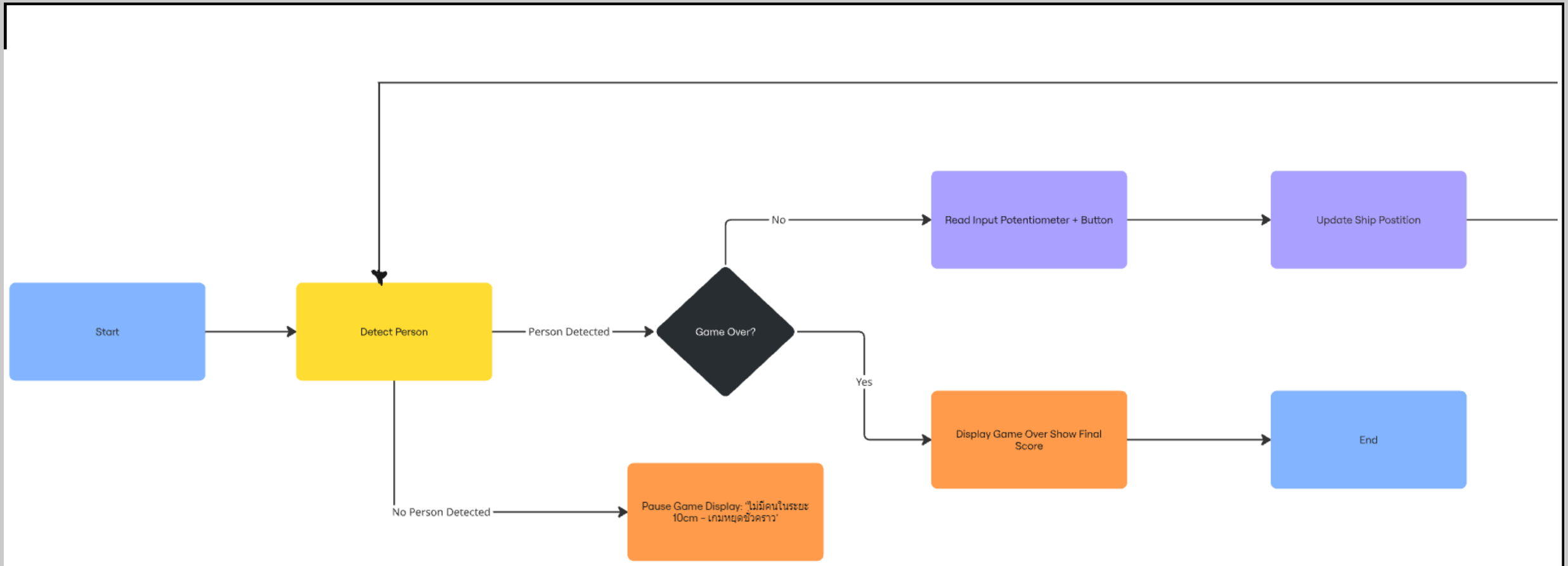
Doraemon Group

Home **Content** Contact



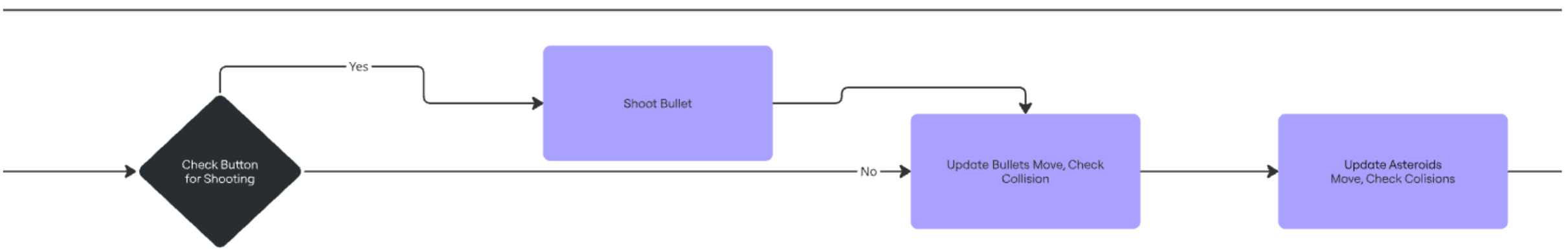
Doraemon Group

Home **Content** Contact



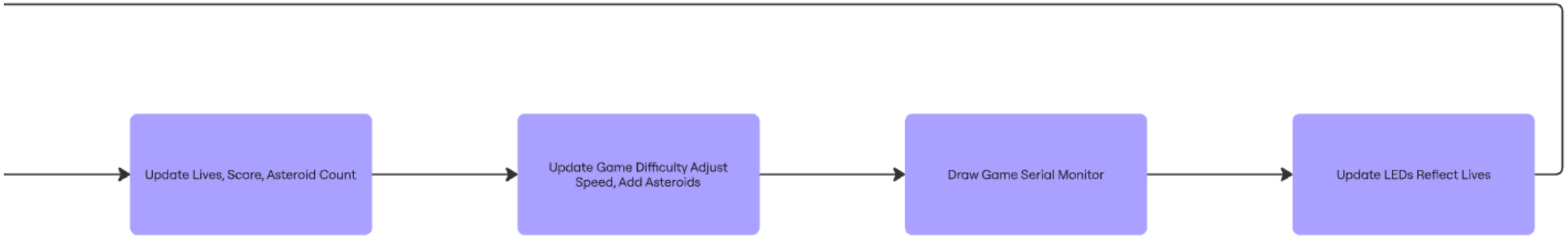
Doraemon Group

Home **Content** Contact



Doraemon Group

Home **Content** Contact



FLOWCHART & CODE

```
#include <Arduino.h>
#include <stdlib.h>

// === Pin Definitions ===
const int potPin = 34;
const int buttonPin = 25;
const int ledGreen = 26;
const int ledYellow = 27;
const int ledRed = 14;

// Ultrasonic Sensor 1
const int trigPin1 = 32;
const int echoPin1 = 33;

// Ultrasonic Sensor 2
const int trigPin2 = 4;
const int echoPin2 = 5;

// === Game Constants ===
const int SCREEN_WIDTH = 20;
const int SCREEN_HEIGHT = 10;
int GAME_UPDATE_RATE = 800;        // asteroid speed (dynamic)
const int BULLET_UPDATE_RATE = 100;
const int BULLET_COOLDOWN = 200;   // ยิงได้ทุก 200ms
const int DISPLAY_UPDATE_RATE = 100;
const int SHIP_MOVE_RATE = 5;
const int DETECT_DISTANCE = 10;

const int MAX_ASTEROIDS = 10;    // สูงสุด 10 ลูก
const int MAX_BULLETS = 5;       // กระสุนสูงสุด 5 ลูกบนจอ

// === Game Variables ===
int shipPosition = SCREEN_WIDTH / 2;

// กระสุน
int bulletX[MAX_BULLETS];
int bulletY[MAX_BULLETS];
bool bulletActive[MAX_BULLETS];
unsigned long lastBulletShot = 0;
unsigned long lastBulletUpdate = 0;
```

VARIABLE DECLARATION

```
// อุกกาบาต
int asteroidX[MAX_ASTEROIDS];
int asteroidY[MAX_ASTEROIDS];
int asteroidDir[MAX_ASTEROIDS];   // สำหรับ zigzag
char asteroidType[MAX_ASTEROIDS]; // '*' = normal, '@' = zigzag, '#' = fast
int asteroidCount = 1;            // เริ่มมี 1 ลูก

// เกม
int lives = 3;
int score = 0;
bool gameOver = false;
bool gameOverPrinted = false;

unsigned long lastUpdate = 0;
unsigned long lastShipUpdate = 0;
unsigned long lastDisplayUpdate = 0;
int targetShipPosition = SCREEN_WIDTH / 2;

int lastMilestone = 0;  // สำหรับเพิ่มอุกกาบาตทีละลูก
```

FLOWCHART & CODE

```
// === Ultrasonic Functions ===
long readUltrasonic(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH, 30000);
  long distance = duration * 0.034 / 2;
  return distance;
}

long stableReadUltrasonic(int trigPin, int echoPin) {
  long sum = 0;
  int count = 5;
  for (int i = 0; i < count; i++) {
    sum += readUltrasonic(trigPin, echoPin);
    delay(5);
  }
  return sum / count;
}

bool isPersonDetected() {
  long dist1 = stableReadUltrasonic(trigPin1, echoPin1);
  long dist2 = stableReadUltrasonic(trigPin2, echoPin2);
  return (dist1 > 0 && dist1 < DETECT_DISTANCE) ||
         (dist2 > 0 && dist2 < DETECT_DISTANCE);
}
```

ULTRASONIC FUNCTION

FLOWCHART & CODE

GAME FUNCTION

```
// === Game Functions ===
void updateLEDs() {
  digitalWrite(ledGreen, lives >= 3);
  digitalWrite(ledYellow, lives >= 2);
  digitalWrite(ledRed, lives >= 1);
}
```

FLOWCHART & CODE

GAME FUNCTION

```
void drawGame() {
  for(int i = 0; i < 16; i++) Serial.println("                    ");
  Serial.print("+--------------------+\n");

  for(int y = 0; y < SCREEN_HEIGHT - 1; y++) {
    Serial.print("|");
    for(int x = 0; x < SCREEN_WIDTH; x++) {
      bool printed = false;

      // วาดอุกกาบาตทั้งหมด
      for (int a = 0; a < asteroidCount; a++) {
        if(y == asteroidY[a] && x == asteroidX[a]) {
          Serial.print(asteroidType[a]);
          printed = true;
          break;
        }
      }

      // วาดกระสุนทั้งหมด
      if(!printed) {
        for (int b = 0; b < MAX_BULLETS; b++) {
          if(bulletActive[b] && y == bulletY[b] && x == bulletX[b]) {
            Serial.print("^");
            printed = true;
            break;
          }
        }
      }

      if(!printed) Serial.print(" ");
    }
    Serial.print("|\n");
  }
```

```
  // วาดยาน
  Serial.print("|");
  for(int x = 0; x < SCREEN_WIDTH; x++) {
    Serial.print(x == shipPosition ? "W" : " ");
  }
  Serial.print("|\n");

  Serial.print("+--------------------+\n");
  Serial.print("Lives: ");
  Serial.print(lives);
  Serial.print(" Score: ");
  Serial.print(score);
  Serial.print(" Asteroids: ");
  Serial.print(asteroidCount);
  Serial.print("\n");
}

int getAsteroidScore(char type) {
  if(type == '*') return 10;
  if(type == '@') return 15;
  if(type == '#') return 20;
  return 5;
}

void shootBullet() {
  for(int i = 0; i < MAX_BULLETS; i++) {
    if(!bulletActive[i]) {
      bulletX[i] = shipPosition;
      bulletY[i] = SCREEN_HEIGHT - 2;
      bulletActive[i] = true;
      break;
    }
  }
}
```

FLOWCHART & CODE

```
void updateBullets() {
  for(int i = 0; i < MAX_BULLETS; i++) {
    if(bulletActive[i]) {
      int prevY = bulletY[i];
      bulletY[i]--;

      for (int a = 0; a < asteroidCount; a++) {
        if(bulletX[i] == asteroidX[a] &&
           ((prevY >= asteroidY[a] && bulletY[i] <= asteroidY[a]) ||
            (bulletY[i] == asteroidY[a]))) {
          score += getAsteroidScore(asteroidType[a]);  // คะแนนตามประเภท
          asteroidY[a] = 0;
          asteroidX[a] = random(0, SCREEN_WIDTH);
          asteroidDir[a] = random(0, 2) == 0 ? -1 : 1;
          int r = random(0, 3);
          asteroidType[a] = (r == 0 ? '*' : (r == 1 ? '@' : '#'));
          bulletActive[i] = false;
        }
      }

      if(bulletY[i] < 0) bulletActive[i] = false;
    }
  }
}
```

GAME FUNCTION

FLOWCHART & CODE

```
void setup() {
  Serial.begin(921600);
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledYellow, OUTPUT);
  pinMode(ledRed, OUTPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  // init asteroid แรก
  asteroidX[0] = random(0, SCREEN_WIDTH);
  asteroidY[0] = 0;
  asteroidDir[0] = random(0, 2) == 0 ? -1 : 1;
  int r = random(0, 3);
  asteroidType[0] = (r == 0 ? '*' : (r == 1 ? '@' : '#'));

  for(int i = 0; i < MAX_BULLETS; i++) bulletActive[i] = false;

  Serial.println("\n\n=== Space Shooter Game ===");
  Serial.println("Use potentiometer to move left/right");
  Serial.println("Press button to shoot (can shoot multiple bullets)");
  Serial.println("Game will start when someone is detected within 10 cm...");
}
```

SETTING UP

FLOWCHART & CODE

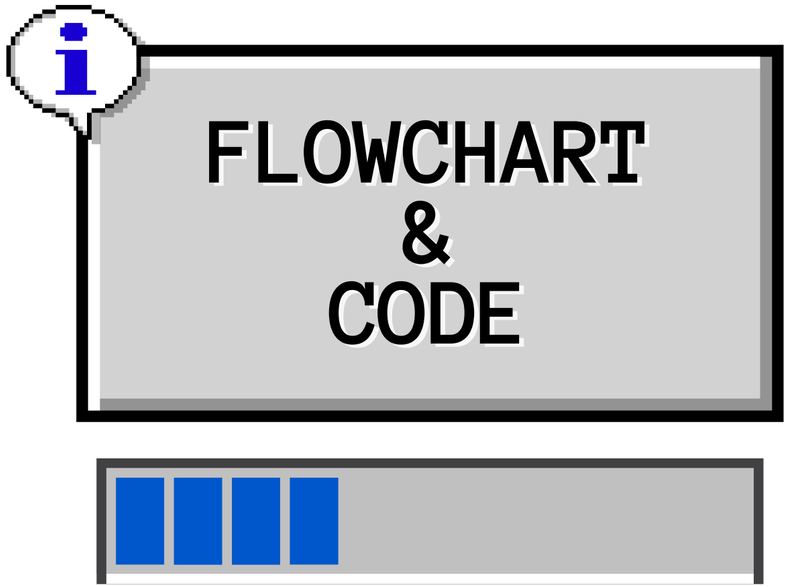
```
void loop() {
  if (!isPersonDetected()) {
    Serial.println("⏸ ไม่มีคนในระยะ 10cm - เกมหยุดชั่วคราว");
    return;
  }

  if(gameOver) {
    if(!gameOverPrinted) {
      Serial.println("\nGAME OVER - Final Score: " + String(score));
      gameOverPrinted = true;
    }
    return;
  }

  unsigned long currentTime = millis();

  // Ship movement
  if(currentTime - lastShipUpdate >= SHIP_MOVE_RATE) {
    lastShipUpdate = currentTime;
    int potValue = analogRead(potPin);
    targetShipPosition = map(potValue, 0, 4095, 0, SCREEN_WIDTH - 1);
    if(shipPosition < targetShipPosition) shipPosition++;
    else if(shipPosition > targetShipPosition) shipPosition--;
  }

  // Shooting
  if(digitalRead(buttonPin) == LOW && currentTime - lastBulletShot >= BULLET_COOLDOWN) {
    shootBullet();
    lastBulletShot = currentTime;
  }
```

LOOP

FLOWCHART & CODE

```
// Bullet update
if(currentTime - lastBulletUpdate >= BULLET_UPDATE_RATE) {
  lastBulletUpdate = currentTime;
  updateBullets();
}

// Asteroid update
if(currentTime - lastUpdate >= GAME_UPDATE_RATE) {
  lastUpdate = currentTime;

  for (int a = 0; a < asteroidCount; a++) {
    // ความเร็วตกต่างกันตามประเภท
    if(asteroidType[a] == '#') {
      asteroidY[a] += 2;  // fast asteroid
    } else {
      asteroidY[a]++;
    }

    if(asteroidType[a] == '@') { // zigzag only
      asteroidX[a] += asteroidDir[a];
      if(asteroidX[a] <= 0 || asteroidX[a] >= SCREEN_WIDTH-1) {
        asteroidDir[a] *= -1;
      }
    }

    if(asteroidY[a] >= SCREEN_HEIGHT - 1) {
      if(abs(asteroidX[a] - shipPosition) <= 1) {
        lives--;
      }
      asteroidY[a] = 0;
      asteroidX[a] = random(0, SCREEN_WIDTH);
      asteroidDir[a] = random(0, 2) == 0 ? -1 : 1;
      int r = random(0, 3);
      asteroidType[a] = (r == 0 ? '*' : (r == 1 ? '@' : '#'));
    }
  }
```

LOOP




```
  // Difficulty scaling
  GAME_UPDATE_RATE = max(200, 800 - (score / 50) * 50);

  if(score >= lastMilestone + 100 && asteroidCount < MAX_ASTEROIDS) {
    asteroidX[asteroidCount] = random(0, SCREEN_WIDTH);
    asteroidY[asteroidCount] = 0;
    asteroidDir[asteroidCount] = random(0, 2) == 0 ? -1 : 1;
    int r = random(0, 3);
    asteroidType[asteroidCount] = (r == 0 ? '*' : (r == 1 ? '@' : '#'));
    asteroidCount++;
    lastMilestone += 100;
  }

  // Display update
  if(currentTime - lastDisplayUpdate >= DISPLAY_UPDATE_RATE) {
    lastDisplayUpdate = currentTime;
    drawGame();
  }
}
```

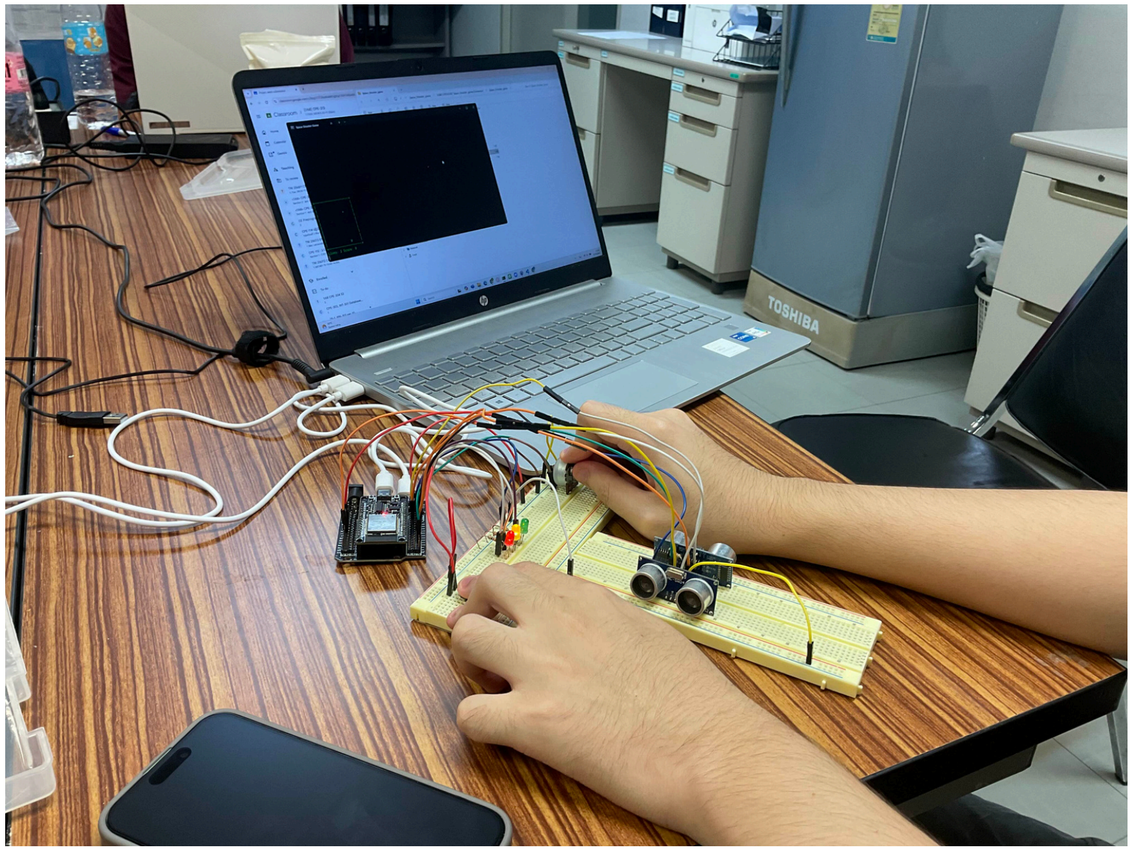
LOOP

Doraemon Group

Home **Content** Contact

Doraemon Group

Home **Content** Contact

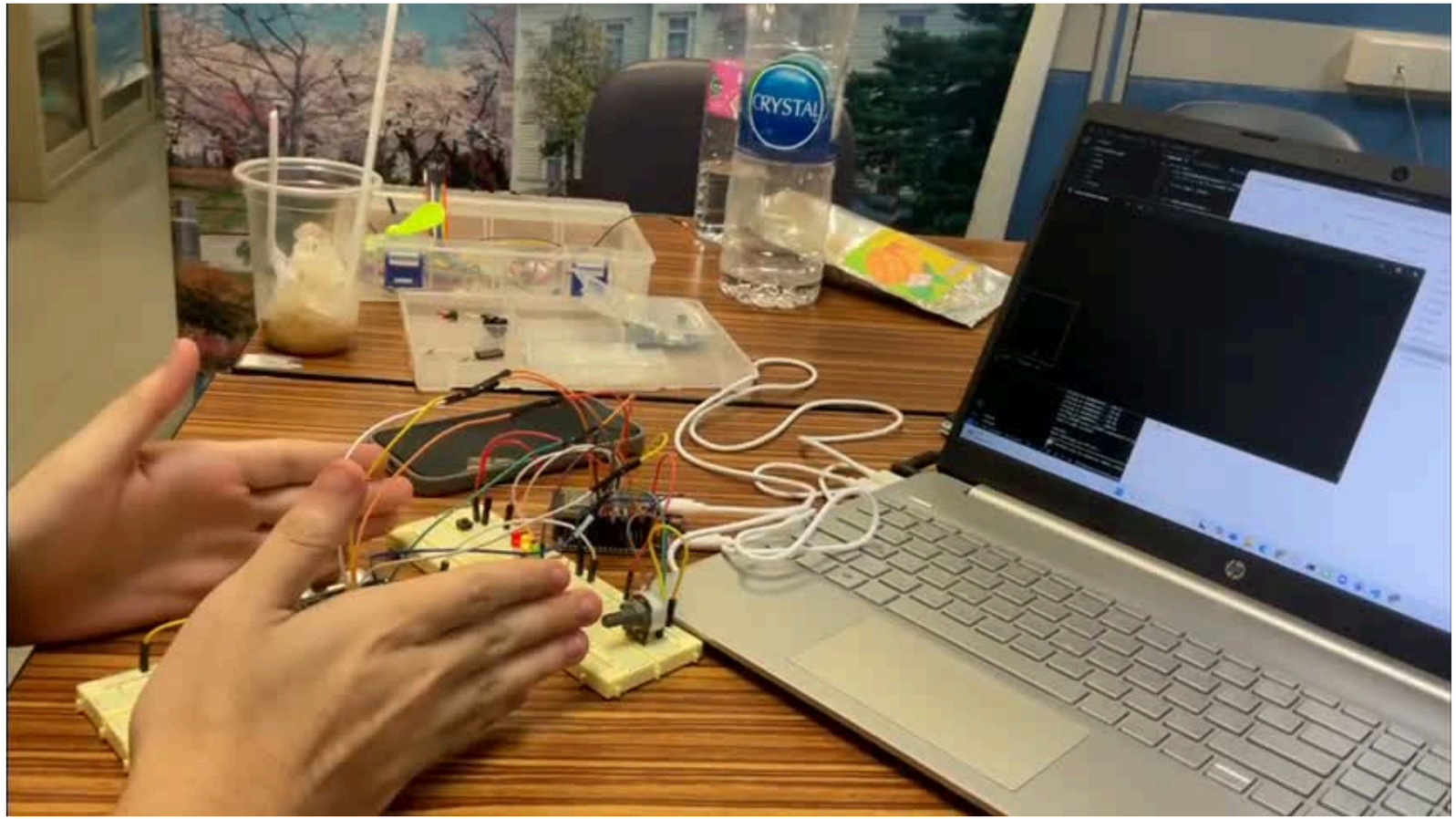


PICTURE OF PROJECT

Doraemon Group

Home **Content** Contact

VDO DEMONSTRATION



Doraemon Group

Home **Content** Contact

PROBLEM AND SOLUTION

Space_Shooter_game/launch_game.bat

```
@@ -2,4 +2,4 @@
2  2  mode con: cols=30 lines=20
3  3  title Space Shooter Game
4  4  color 0A
5     - pio device monitor --port COM7 --baud 115200
      5 + pio device monitor --port COM7 --baud 921600
```

Space_Shooter_game/platformio.ini

```
@@ -12,4 +12,4 @@
12  12  platform = espressif32
13  13  board = esp32dev
14  14  framework = arduino
15     - monitor_speed = 115200
      15 + monitor_speed = 921600
```

OLD

NEW

```
22  - const int GAME_UPDATE_RATE = 800;     // Asteroid speed
23  - const int BULLET_UPDATE_RATE = 100;   // Bullet speed
24  - const int DISPLAY_UPDATE_RATE = 250;  // Screen refresh
25  - const int SHIP_MOVE_RATE = 10;        // Ship movement
26  - const int DETECT_DISTANCE = 10;       // ตรวจจับ 10 cm
22  + const int GAME_UPDATE_RATE = 500;
23  + const int BULLET_UPDATE_RATE = 100;
24  + const int DISPLAY_UPDATE_RATE = 100;
25  + const int SHIP_MOVE_RATE = 5;
26  + const int DETECT_DISTANCE = 10;
```

OLD

NEW

Doraemon Group

Home Content **Contact**

PROJECT GANTT CHART

Task	week 1	week 2	week 3	week 4	Responsible
Study hardware & sensor	25/8/2025 - 27/8/2025				Both
Basic game config	29/8/2025 - 5/9/2025				Chinapat
Add ultrasonic detection		4/9/2025 - 5/9/2025			Both
Debug and testing		6/9/2025 - 14/9/2025			Chisanupong
Prepare document and slides			10/9/2025 - 20/9/2025		Both

su.chinapat_st@tni.ac.th
su.chisanupong_st@tni.ac.th

link to github