```
#include <Wire.h>
#include <Servo.h>

// I/O 腳位定義
#define PIN_SERVO_LEFT      10
#define PIN_SERVO_RIGHT     13
#define PIN_CDS             12    // 光敏電阻 (CDS) 腳位，用於啟動/停止
#define PIN_SONAR_L         9     // 左側超音波感測器腳位
#define PIN_RC_L            8     // RC Time 左感測器腳位 (循光用)
#define PIN_BUMPER_LEFT     7     // 左側碰撞/觸碰感測器腳位
#define PIN_RC_M            6     // RC Time 中感測器腳位 (循光用)
#define PIN_BUMPER_RIGHT    5     // 右側碰撞/觸碰感測器腳位
#define PIN_RC_R            4     // RC Time 右感測器腳位 (循光用)
#define PIN_SONAR_R         3     // 右側超音波感測器腳位

Servo servoLeft; // 左側伺服電機物件
Servo servoRight; // 右側伺服電機物件

// --- 伺服馬達動作數據中心 ---
// 動作代碼 [Left Servo Pulse, Right Servo Pulse]:
const int MovementPulses[][2] =
{
  /* [0] 停止 */       { 1500, 1500 },
  /* [1] 前進 */       { 1700, 1300 },
  /* [2] 後退 */       { 1300, 1700 },
  /* [3] 原地左轉*/      { 1300, 1300 },
  /* [4] 原地右轉*/      { 1700, 1700 },
```

```
  /* [5]  左轉微調(單側緩)*/  { 1600, 1300 },
  /* [6]  右轉微調(單側緩)*/  { 1700, 1400 },
  /* [7]  左轉微調(單側停)*/  { 1500, 1300 },
  /* [8]  右轉微調(單側停)*/  { 1700, 1500 }
};

int pulseLeft;
int pulseRight;
const int NUM_MOVEMENT_ACTIONS = sizeof(MovementPulses) / sizeof(MovementPulses[0]);

/* ---  全域變數定義  --- */
long sonarDistanceLeft, sonarDistanceRight;       //  左右超音波距離
long sonarDistancePrevLeft = 0;
long sonarDistancePrevRight;
long rcTimeValueLeft, rcTimeValueRight;           //  左右循光感測器讀數
long rcTimeValueMid;                              //  中央循光感測器讀數
long rcTimeValuePrevMid = 0;                      //  上一次中央循光讀數
byte navigationEntryFlag = 0;
byte navigationStateCounter = 0;                  //  導航狀態機計數器
bool isRobotStarted = false;                      //  機器人啟動狀態旗標
int i;

/* ------------------------------------------------------------- */
//                          函式宣告  (Function Prototypes)
/* ------------------------------------------------------------- */
void move(int actionIndex);                //  核心動作控制函式
void execute_safe_forward(void);           //  帶碰撞檢測的前進
```

```
long ultrasound(int Pin);                  // 超音波測距
void ultrasoundr_10(void);                 // 右側循牆邏輯
long rctime(int Pin);                      // RC Time 循光感測器讀取
void blackbox(void);                       // 永久循光主邏輯
void left_90(void);                        // 執行左轉 90 度
void right_90(void);                       // 執行右轉 90 度

/* ------------------------------------------------------------ */
//                          Setup 和 Loop
/* ------------------------------------------------------------ */

void setup(void)
{
  // 設定輸入腳位模式
  pinMode(PIN_CDS, INPUT);
  pinMode(PIN_BUMPER_RIGHT, INPUT);
  pinMode(PIN_BUMPER_LEFT, INPUT);

  // 初始化伺服馬達
  servoLeft.attach(PIN_SERVO_LEFT);
  servoRight.attach(PIN_SERVO_RIGHT);

  move(0); // 初始停止
  Serial.begin(9600);
}

void loop(void)
```

```
{
  // 階段 1: 檢查啟動狀態 (優先級最高)
  if (!isRobotStarted)
  {
    // 檢查光敏電阻是否滿足啟動條件
    if (!(digitalRead(PIN_CDS)))
    {
      // 執行啟動序列 (只執行一次)
      isRobotStarted = true; // 設置旗標為已啟動
    }
    else
    {
      // 保持停止並等待啟動
      move(0);
      return; // 立即結束本次 loop 循環，等待下次檢查
    }
  }

  // 階段 2: 正式動作 (僅當 isRobotStarted 為 true 時執行)
  sonarDistanceLeft = ultrasound(PIN_SONAR_L);

  if (sonarDistanceLeft > 1400) // 1. 遇到轉角/空曠區域
  {
    switch (navigationStateCounter)
    {
      case 0:
        left_90();
```

```
        break;
      case 1:
        left_90();
        break;
      case 2:
        right_90();
        break;
      case 3:
        right_90();
        //ultrasoundr_10();
        break;
      case 4:
        // 執行預設距離的前進，然後進入永久循光
        for (i = 0; i < 990; i++)
        {
          execute_safe_forward();
        }
//        move(0);
//        while(1);
        blackbox(); // 進入 blackbox() 後，程式將被鎖定在循光無限迴圈中
        break;
    }
    navigationStateCounter++;
  }
  else // 2. 距離適中或近：執行基本安全移動
  {
    execute_safe_forward();
```

```
  }
}

/* ---------------------------------------------------------------- */
//                          函式定義 (Function Definitions)
/* ---------------------------------------------------------------- */

// 核心動作函式 (內建 delay(5) 作為最小動作時間片)
void move(int actionIndex)
{
  if (!(actionIndex >= 0 && actionIndex < NUM_MOVEMENT_ACTIONS))
  {
    actionIndex = 0;
  }
  pulseLeft = MovementPulses[actionIndex][0];
  pulseRight = MovementPulses[actionIndex][1];
  servoLeft.writeMicroseconds(pulseLeft);
  servoRight.writeMicroseconds(pulseRight);
  if (actionIndex != 0)
  {
    delay(2); // 為減輕馬達抖動、控制不穩定、CPU 負荷情況。常用 2ms~4ms 之間
  }
}

// 基本移動/避障函式 (在沒有碰撞時執行前進)
void execute_safe_forward(void)
{
```

```
  if (!(digitalRead(PIN_BUMPER_RIGHT)))
  {
    i=i-2;
    move(3); //  右側碰撞，左轉
  }
  else if (!(digitalRead(PIN_BUMPER_LEFT)))
  {
    i=i-2;
    move(4); //  左側碰撞，右轉
  }
  else
  {
    move(1); //  無碰撞，前進
  }
}

/* ---  感測器與輔助函式  --- */

long ultrasound(int Pin)
{
  pinMode(Pin, OUTPUT);
  digitalWrite(Pin, LOW);
  delayMicroseconds(2);
  digitalWrite(Pin, HIGH);
  delayMicroseconds(5);
  digitalWrite(Pin, LOW);
  pinMode(Pin, INPUT);
```

```
  long sonar = pulseIn(Pin, HIGH);
  return sonar;
}

// 用於右側循牆，直到距離超過 1200 或讀數不穩定
void ultrasoundr_10(void)
{
  do
  {
    sonarDistanceRight = ultrasound(PIN_SONAR_R);

    // 檢查讀數是否穩定。若變化過大 (>= 200)，則立即跳出迴圈
    if (abs(sonarDistancePrevRight - sonarDistanceRight) >= 200)
    {
      break;
    }

    if (sonarDistanceRight < 400)
    {
      move(3); // 離牆太近，左轉遠離
    }
    else if (sonarDistanceRight > 600)
    {
      move(4); // 離牆太遠，右轉靠近
    }
    else
    {
```

```
      execute_safe_forward(); //  距離適中，繼續前進
    }

    sonarDistancePrevRight = sonarDistanceRight;

  } while (sonarDistanceRight < 1200); //  距離超過  1200  則脫離
}

long rctime(int Pin)
{
  long rcTimeValue = 0;
  pinMode(Pin, OUTPUT);
  digitalWrite(Pin, HIGH);
  delay(1); //  確保充電時間
  pinMode(Pin, INPUT);
  digitalWrite(Pin, LOW);
  while (digitalRead(Pin))
  {
    rcTimeValue++; //  測量放電時間
  }
  return rcTimeValue;
}

/* ---  複雜動作函式  (迴圈次數已調整為  5ms  基礎時間) --- */

void left_90(void)
{
```

```
  // 參數可能需要根據實際機器人調整
  for (i = 0; i < 170; i++)   //多走一點
  {
    execute_safe_forward();
  }
  for (i = 0; i < 150; i++) //原地自轉
  {
    move(3); // 左轉
  }
  for (i = 0; i < 370; i++)   //多走一點
  {
    execute_safe_forward();
  }
  move(0);
}

void right_90(void)
{
  // 參數可能需要根據實際機器人調整
  for (i = 0; i < 170; i++) //多走一點
  {
    execute_safe_forward();
  }
  for ( i = 0; i < 150; i++) //原地自轉
  {
    move(4); // 右轉
  }
```

```
  for (i = 0; i < 370; i++)   //多走一點
  {
    execute_safe_forward();
  }
  move(0);
}

// blackbox 函式 (高度反應性循光)
void blackbox(void)
{
  while (1) // 無限迴圈，用於終局循光
  {
    // 註解：若循光感測器有硬體差異，可在此處對 rcTimeValueRight/Left/Mid
    // 自行進行預設修正值的加減（例如，發現某個值固定偏高 30，則減去 30），以利平衡使用。
    rcTimeValueRight = rctime(PIN_RC_R);
    rcTimeValueMid = rctime(PIN_RC_M);
    rcTimeValueLeft = rctime(PIN_RC_L);

    // 循光主判斷邏輯：如果滿足以下任一條件，則執行安全前進：
    // 1. (左右平衡 AND 中心反射較低)：左右差異小於 300 且中心讀數相對低 (不在黑線核心區)
    // 2. 中央讀數於有光情況

    if ((abs(rcTimeValueRight - rcTimeValueLeft) > 300 ) && (rcTimeValueMid > 250) )
    {
      if (rcTimeValueRight > rcTimeValueLeft)
      {
        move(3); // 右邊讀數高 (可能偏向左邊)，修正左轉
```

```
        }
        else
        {
          move(4);
        }
      }
      else
      {
        execute_safe_forward();
      }
    }
}
```