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

// ---  腳位與硬體常數定義  ---
#define PIN_SERVO_LEFT      10
#define PIN_SERVO_RIGHT     13
#define PIN_CDS             12

// ❗  循跡感測器腳位  (I²C / TWI) ❗
//  循跡感測器使用  I²C  協定通訊  (透過  Wire.h  函式庫)。
// Arduino UNO/NANO  的  I²C  預設腳位為：SDA: A4, SCL: A5。
const uint8_t SENSOR_BAR_ADDRESS = 0x3E;   // I²C  循跡感測器條的地址
SensorBar mySensorBar(SENSOR_BAR_ADDRESS);

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]);

// ---  全域變數宣告  (用於循跡與狀態追蹤) ---
unsigned long turnStartTime = 0;     //  記錄開始轉向計時的時間點
unsigned long turnPassTime = 0;      //  記錄轉向計時經過的時間
int lastTurnDirection = 0;           //  記錄上次車子偏離線路的傾向  (1:  偏左, 2:  偏右)
boolean isTurningAction = 0;         //  標誌位，指示當前循跡動作是否為轉向動作
byte sensorRawValue = 0;             //  循跡感測器條讀取的  8  位元原始值
boolean isRobotStarted = false;      //  狀態旗標：追蹤機器人是否已啟動

// ---  函式原型宣告  (Function Prototypes) ---
void move(int actionIndex);
void check_turn(int direction);
byte check_Qit(void);

// ============================================================
// --- SETUP ---
// ============================================================
```

```
void setup(void)
{
  servoLeft.attach(PIN_SERVO_LEFT);
  servoRight.attach(PIN_SERVO_RIGHT);
  Wire.begin(); //  初始化 I²C 匯流排
  move(0); //  初始設定為停止

  //  初始化循跡感測器條
  mySensorBar.setBarStrobe();
  mySensorBar.clearInvertBits();
  uint8_t returnStatus = mySensorBar.begin();
}

// ========================================================
// --- LOOP ---
// ========================================================

void loop(void)
{
    //  階段 1: 檢查啟動狀態
    if (!isRobotStarted)
    {
        if (!(digitalRead(PIN_CDS)))
        {
            isRobotStarted = true; //  設置旗標為已啟動
            move(1);                  //  前進
            delay(200);               //  延遲
```

```
    }
    else
    {
        move(0);
        return;
    }
}

// 階段 2: 主循跡邏輯

// --- 循跡感測與動作 ---
sensorRawValue = check_Qit();

switch(sensorRawValue)
{
    // --- 左轉微調(單側停) ---
    case B10000000:
    case B11000000:
    case B11100000:
    case B11110000:
    case B11111000:
    case B01000000:
    case B01100000:
    case B01110000:
    case B01111000:
        move(7);
        isTurningAction = 1;
```

```
    break;
// ---  左轉微調(單側緩) ---
case B00100000:
case B00110000:
case B00111000:
    move(5);
    isTurningAction = 1;
    break;
// ---  右轉微調(單側停) ---
case B00000001:
case B00000011:
case B00000111:
case B00001111:
case B00011111:
case B00000010:
case B00000110:
case B00001110:
case B00011110:
    move(8);
    isTurningAction = 1;
    break;
// ---  右轉微調(單側緩) ---
case B00000100:
case B00001100:
case B00011100:
    move(6);
    isTurningAction = 1;
```

```
        break;
    // ---  找不到線  (Missing Line) ---
    case B00000000:
        check_turn(lastTurnDirection);
        lastTurnDirection = 0;
        break;

    // ---  路口/全黑線  (Intersection) ---
    case B11111111: case B01111111: case B11111110:
        move(1);
        delay(300); //多跑一點點
        break;

    default:
        move(1);
        sensorRawValue = check_Qit();
}

turnPassTime = millis() - turnStartTime;

//  紀錄轉向傾向
if(!isTurningAction)
{
    switch(sensorRawValue & B11000011)
    {
        case B10000000:
        case B11000000:
```

```
            case B01000000:
                lastTurnDirection = 1; //  傾向左轉
                turnStartTime = millis();
                break;
            case B00000001:
            case B00000011:
            case B00000010:
                lastTurnDirection = 2; //  傾向右轉
                turnStartTime = millis();
                break;
        }
    }
    isTurningAction = 0;
}

// ============================================================
// ---  輔助函式定義  (Function Definitions) ---
// ============================================================

//  核心動作函式
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
  }
}

////////////////////////////////////////////////////////////////////
//  檢查轉向函式：找不到線時執行，使用上次傾向來大幅轉彎
void check_turn(int direction)
{
  if(turnPassTime > 200)   //  如果時間過長，清除傾向
  {
    direction = 0;
  }
  if(direction == 1) //  傾向左轉
  {
    do
    {
      //  原地左轉  (左後退，右前進)
      move(3);
    }while(!(check_Qit()));
    delay(58);   //多轉一點點
    return;
  }
```

```
  else if(direction == 2) //  傾向右轉
  {
    do
    {
      //  原地右轉  (左前進，右後退)
      move(4);
    }while(!(check_Qit()));
    delay(66);   //多轉一點點
    return;
  }
  turnStartTime = millis();
  return;
}

//////////////////////////////////////////////////////////////////
//  讀取循跡感測器原始數值
byte check_Qit(void)
{
  uint8_t rawValue = mySensorBar.getRaw();
  return rawValue;
}
//////////////////////////////////////////////////////////////////
```