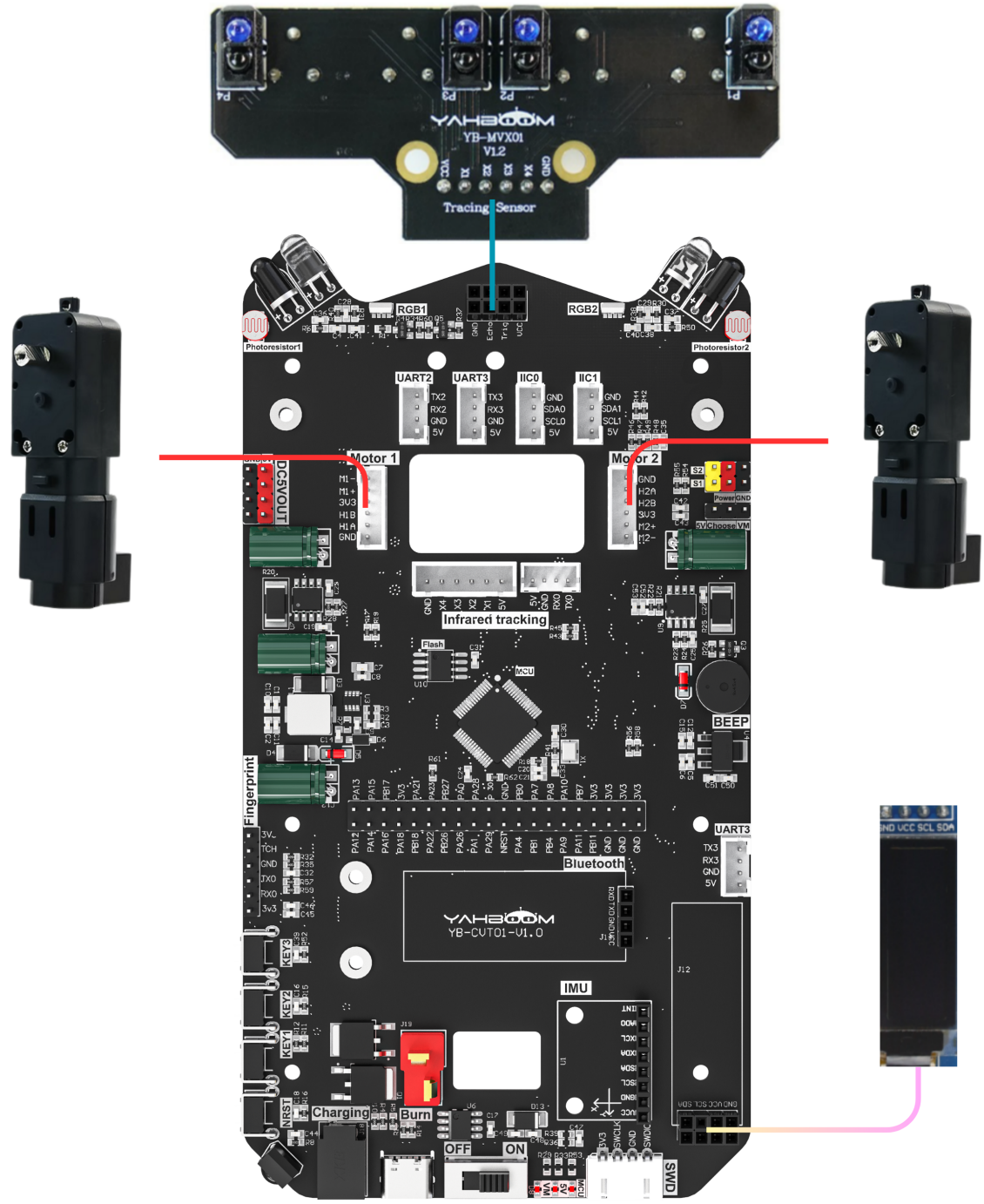
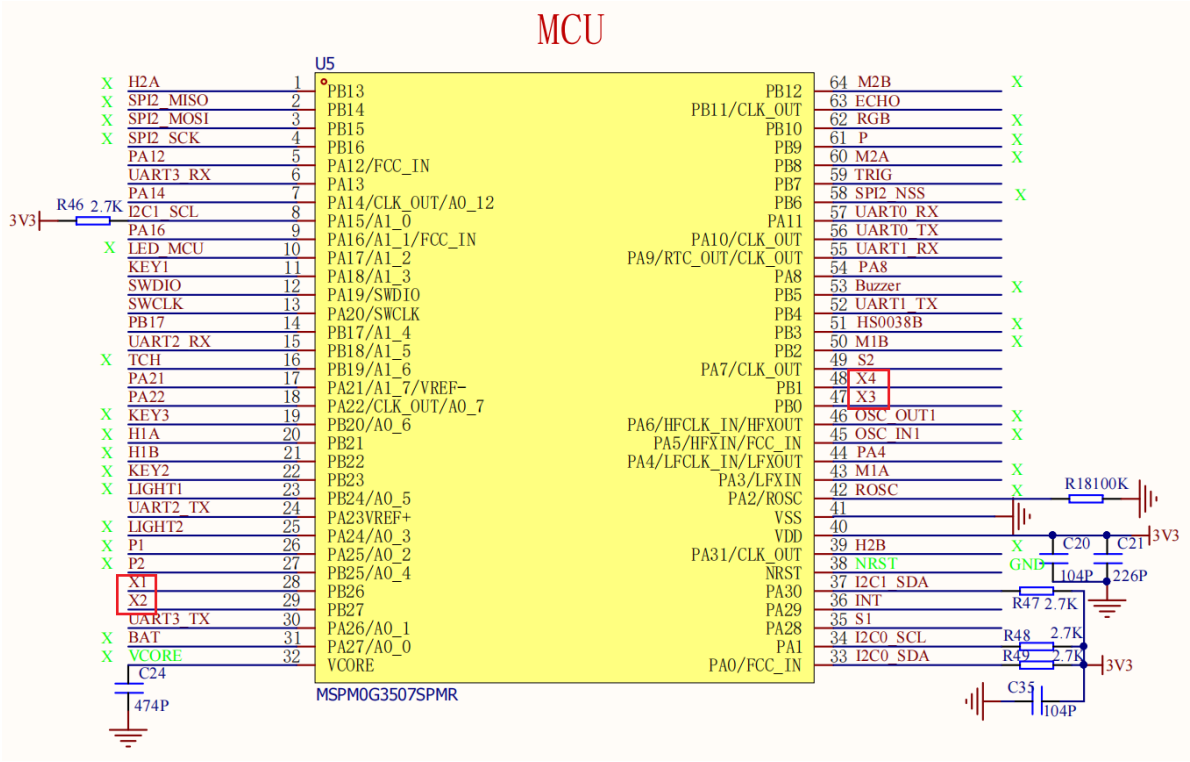
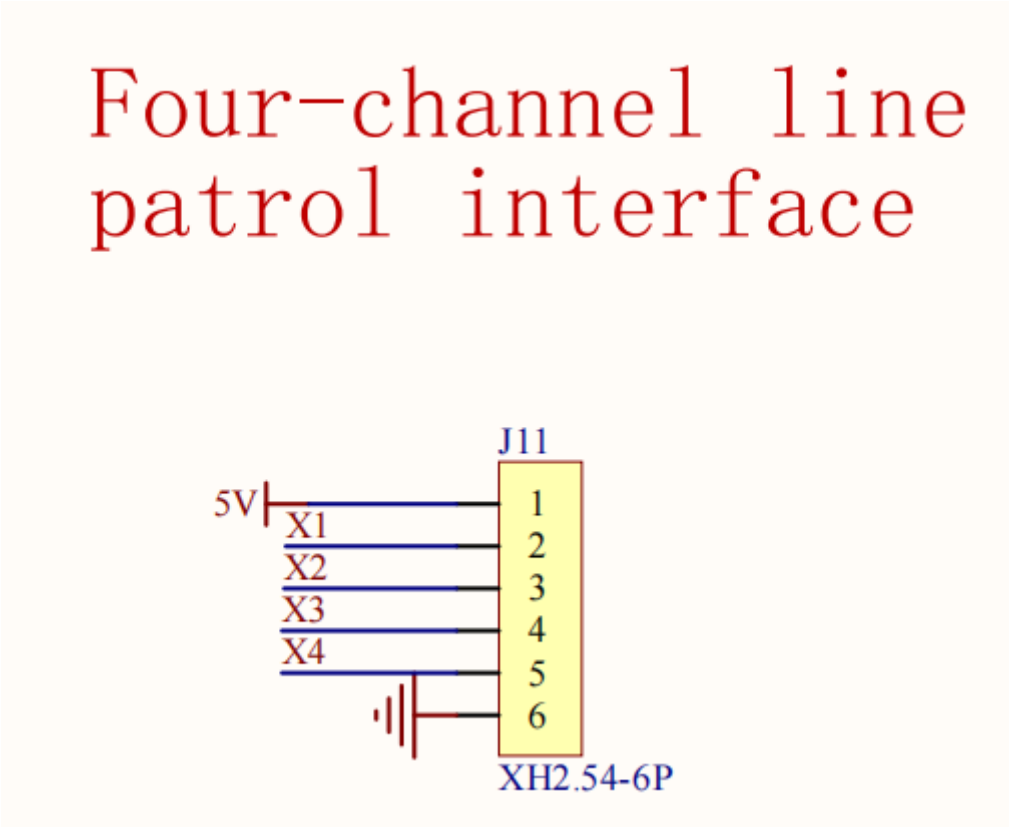
Car Line Tracking - 4 Channel Infrared Tracking Sensor Module

1. Hardware Connection

4 Channel Infrared Tracking Sensor Module	MSPM0G3507
5V	5V
GND	GND
X1	X1
X2	X2
X3	X3
X4	X4

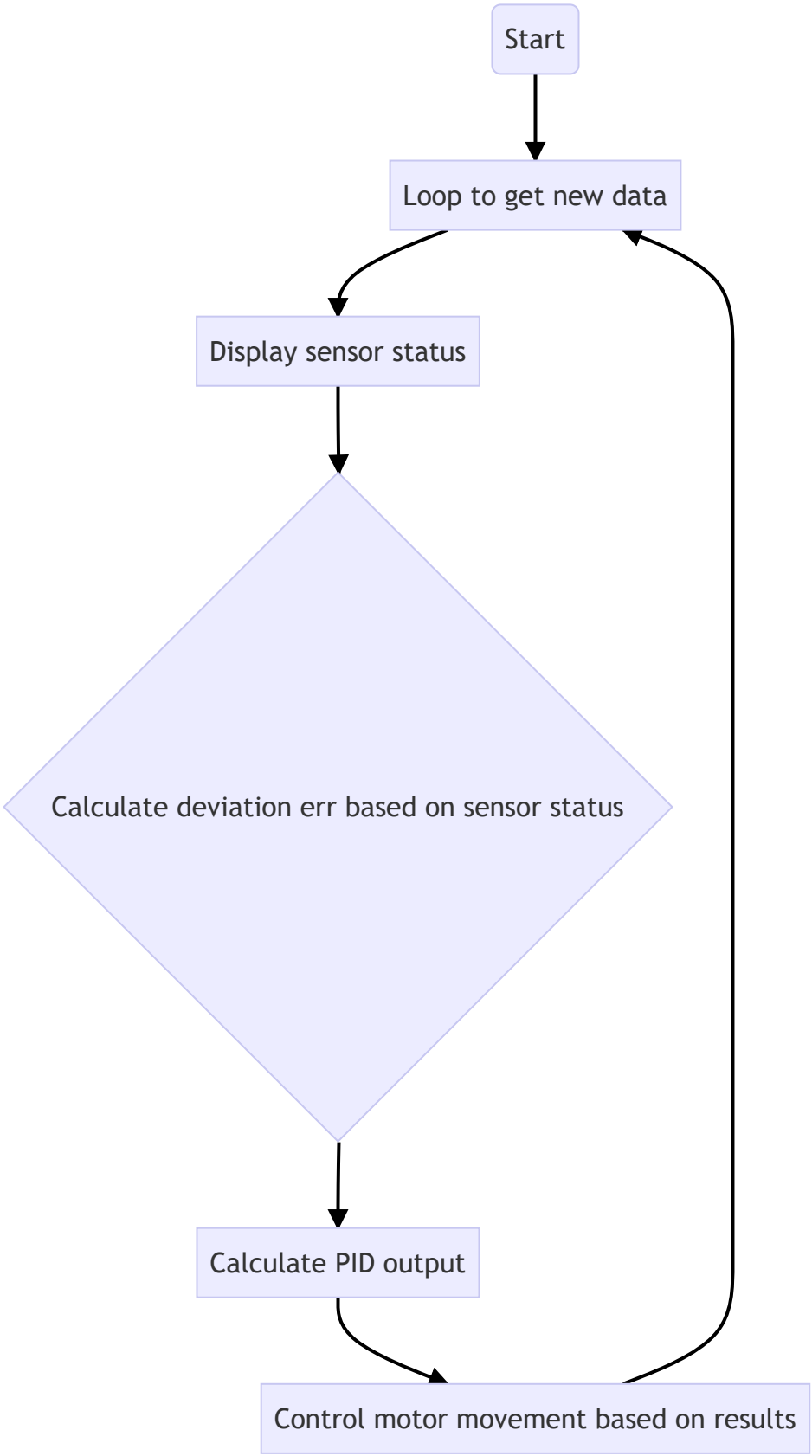

YAHBOOM
YB-MVX01
V1.2
Tracing Sensor
Photoresistor1
Photoresistor2
RGB1
RGB2
UART2
UART3
IIC0
IIC1
Motor 1
Motor 2
DC5VOUT
Infrared tracking
Flash
MCU
BEEP
Fingerprint
Bluetooth
YAHBOOM
YB-CUT01-V1.0
IMU
Charging
Burn
OFF
ON
SWD
KEY1
KEY2
KEY3
NRST





2. Partial Code Analysis

Control Principle



Four_linewalking.c

```
//带死区处理的位置式 / Positional PID with dead zone processing
float APP_IR_PID_Calc(int8_t actual_value)
{
```

```
    float IRTrackTurn = 0;
    int8_t error;
    static int8_t error_last=0;
    static float IRTrack_Integral;//积分 / Integral

    error=actual_value;

    IRTrack_Integral +=error;

    //位置式pid / Positional PID
    IRTrackTurn=error*IRTrack_Trun_KP
                          +IRTrack_Trun_KI*IRTrack_Integral
                          +(error - error_last)*IRTrack_Trun_KD;



    if (IRTrackTurn > (MAX_SPEED - MOTOR_DEAD_ZONE))
            IRTrackTurn = (MAX_SPEED - MOTOR_DEAD_ZONE);
    if (IRTrackTurn < (MOTOR_DEAD_ZONE - MAX_SPEED))
            IRTrackTurn = (MOTOR_DEAD_ZONE - MAX_SPEED);
    return IRTrackTurn;
}
//得到四路循迹模块的数据 / Get data from 4-channel line tracking module
void Four_GetLineWalking(int *LineL1, int *LineL2, int *LineR1, int *LineR2)
{
    *LineL1 = LineWalk_L1_IN;
    *LineL2 = LineWalk_L2_IN;
    *LineR1 = LineWalk_R1_IN;
    *LineR2 = LineWalk_R2_IN;
}
//
void Four_LineWalking(void)
{

    Four_GetLineWalking(&LineL1, &LineL2, &LineR1, &LineR2);//获取黑线检测状态 /
Get black line detection status
        sprintf((char *)oledbuf, "L2:%d L1:%d R1:%d R2:%d",
        LineWalk_L2_IN,LineWalk_L1_IN,LineWalk_R1_IN,LineWalk_R2_IN);
        OLED_ShowString(0, 10, (uint8_t *)oledbuf, 8, 1);

    // 0 0 X 0 / 0 0 X 0
    // 1 0 X 0 / 1 0 X 0
    // 0 1 X 0 / 0 1 X 0
    //处理右锐角和右直角的转动 / Handle right sharp angle and right straight angle
turns
    if( (LineL1 == LOW || LineL2 == LOW) && LineR2 == LOW)
    {
        err=13;

    }
   // 0 X 0 0 / 0 X 0 0
   // 0 X 0 1 / 0 X 0 1
   // 0 X 1 0 / 0 X 1 0
   //处理左锐角和左直角的转动 / Handle left sharp angle and left straight angle turns
    else if ( LineL1 == LOW && (LineR1 == LOW || LineR2 == LOW))
    {
        err=-13;
```

```
    }
    // 0 X X X / 0 X X X
   //最左边检测到 / Leftmost sensor detected
    else if( LineL1 == LOW )
    {
        err=-9;

    }
    // X X X 0 / X X X 0
   //最右边检测到 / Rightmost sensor detected
    else if ( LineR2 == LOW)
    {
        err=9;


    }
    // X 0 1 X / X 0 1 X
   //处理左小弯 / Handle left small turn
    else if (LineL2 == LOW && LineR1 == HIGH) //中间黑线上的传感器微调车左转 /
Sensors on the middle black line slightly adjust car to turn left
    {
        err=-1;
    }
    // X 1 0 X / X 1 0 X
   //处理右小弯 / Handle right small turn
    else if (LineL2 == HIGH && LineR1 == LOW) //中间黑线上的传感器微调车右转 /
Sensors on the middle black line slightly adjust car to turn right
    {
        err=1;
    }
    // X 0 0 X / X 0 0 X
   //处理直线 / Handle straight line
    else if(LineL2 == LOW && LineR1 == LOW) // 都是黑色，加速前进 / Both are black,
accelerate forward
    {
        err=0;
    }
    // 0 0 0 0 / 0 0 0 0
    else if(LineL1 == LOW && LineL2 == LOW && LineR1 == LOW && LineR2 == LOW) //
都是黑色，加速前进 / All are black, accelerate forward
    {
        err = 0;
    }

    //当为1 1 1 1时小车保持上一个小车运行状态 / When it's 1 1 1 1, the car maintains
the previous running state
    pid_output_IRR = (int)(APP_IR_PID_Calc(err));

        Motion_Car_Control(IRR_SPEED, 0, pid_output_IRR);

}
```

3. Main Functions

APP_IR_PID_Calc

Function Prototype	**float APP_IR_PID_Calc(float actual_value)**
Function Description	Implements infrared line tracking positional PID calculation, calculates control amount based on input deviation value, and applies output limiting
Input Parameters	actual_value: actual deviation value (used for PID output calculation)
Return Value	float type, control amount after PID calculation (with limiting applied)

Four_GetLineWalking

Function Prototype	**void Four_GetLineWalking(int *LineL1, int *LineL2, int *LineR1, int *LineR2)**
Function Description	Gets pin level status of four line tracking sensors (LineL1, LineL2, LineR1, LineR2) and returns through pointers
Input Parameters	LineL1: pointer to variable storing left first sensor status; LineL2: pointer to variable storing left second sensor status; LineR1: pointer to variable storing right first sensor status; LineR2: pointer to variable storing right second sensor status
Return Value	None

Four_LineWalking

Function Prototype	**void Four_LineWalking(void)**
Function Description	Four-sensor line tracking main function: reads sensor status, calculates deviation value err based on different detection conditions, calls PID to calculate control amount, and finally controls car movement through Motion_Car_Control (straight line speed is IRR_SPEED, steering determined by PID output)
Input Parameters	None
Return Value	None

4. Experimental Phenomenon

After connecting the car properly, connecting the OLED module, and burning the program to MSPM0, place the car on a white background with black lines map, the car will follow the line, and sensor data will be displayed on the OLED

Where L2, L1, R1, R2 represent the four sensor data from left to right


L211 L111 R111 R211