第十七届全国大学生
智能车竞赛

技 术 报 告

学　　校　:清华大学

队 伍 名 称　:THU-IRT

参 赛 队 员　:付宇辉 吕昕航 谭俊晗 吴宗桓 赵柳权

（按照姓氏拼音排序）

带 队 教 师　：高博麟

关于技术报告和论文使用授权的说明

本人完全了解全国大学生智能汽车竞赛关于保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会可以在相关主页上收录并公开参赛作品的的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会初版论文集里。

参赛队员签名：吴宗桢

带队教师签名：

日　　期：2022/08/19

目录

第一章 引言

全国大学生智能汽车竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，旨在促进高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神，为优秀人才的脱颖而出创造条件。智能车设计内容涵盖了控制、模式识别、传感技术、汽车电子、电气、计算机、机械等多个学科的知识，对学生的知识融合和实践动手能力的培养，具有良好的推动作用。大赛以竞速赛为基本竞赛形式，辅助以创意赛和技术方案赛等多种形式。竞速赛以统一规范的标准硬软件为技术平台，制作一部能够自主识别道路的模型汽车，按照规定路线行进，并符合预先公布的其他规则，以完成时间最短者为优胜。本竞赛过程包括理论设计、实际制作、整车调试、现场比赛等环节，要求学生组成团队，协同工作，初步体会一个工程性的研究开发项目从设计到实现的全过程。竞赛融科学性、趣味性和观赏性为一体，是以迅猛发展、前景广阔的汽车电子为背景，涵盖自动控制、模式识别、传感技术、电子、电气、计算机、机械与汽车等多学科专业的创意性比赛。本竞赛规则透明，评价标准客观，坚持公开、公平、公正的原则，保证竞赛向健康、普及、持续的方向发展。

我们使用*I*型车模，基本没有进行大规模改装，使用的传感器，如摄像头，都是根据赛方要求及验证过的版本。下面，我们从车模制作及主要思路、机械部分安装与改装、系统电路板的固定及连接、电路系统设计思路、微处理器控制软件主要理论及算法这几个方面进行介绍。我们的设计完全原创，秉持高度创新的原则，没有搬移或参考任何已有成果。

第二章 设计思路及技术方案概要说明

硬件电路部分采用按功能分板印刷的思路，利用赛方提供的车模孔位信息分别设计下位机、驱动板、电源板。这样的设计，相比于所有功能集成于同一电路板，有以下优点。首先，比较容易损坏的驱动板和电源板与下位机分离，有利于减小损耗，方便修复、更换、迭代。其次，驱动板给电机供电时，若驱动板提供的 PWM 信号占空比突然增加，会导致驱动板的电源局部降低。如果电压过低，电源板将开启欠压保护，导致下位机断电，一瞬间后上电将重启，产生意外的输出。最后，由于电机驱动在开关大电流，驱动产生的噪音会影响电源部分的模拟反馈信号和上下位机某些长信号线的电压，同时上下位机的数字信号变化也会影响电源部分的模拟反馈。

第三章 传感器、电路系统机械连接说明

在车模安装的基础上，我们用 M3 螺丝把驱动板固定在小车左翼，电源板固定在右翼，下位机固定在上位机与车模之间，如图 3.1 与图 3.2 所示。舵机、驱动板、编码器的信号线连接到下位机上，供电线直接连到电源板上。另外，我们计划在车内安装一个路由器用于发车和调试。上位机和下位机的信号接口是一对排针座，既能提供机械支撑又能保证低阻抗触点并防止针脚错位。

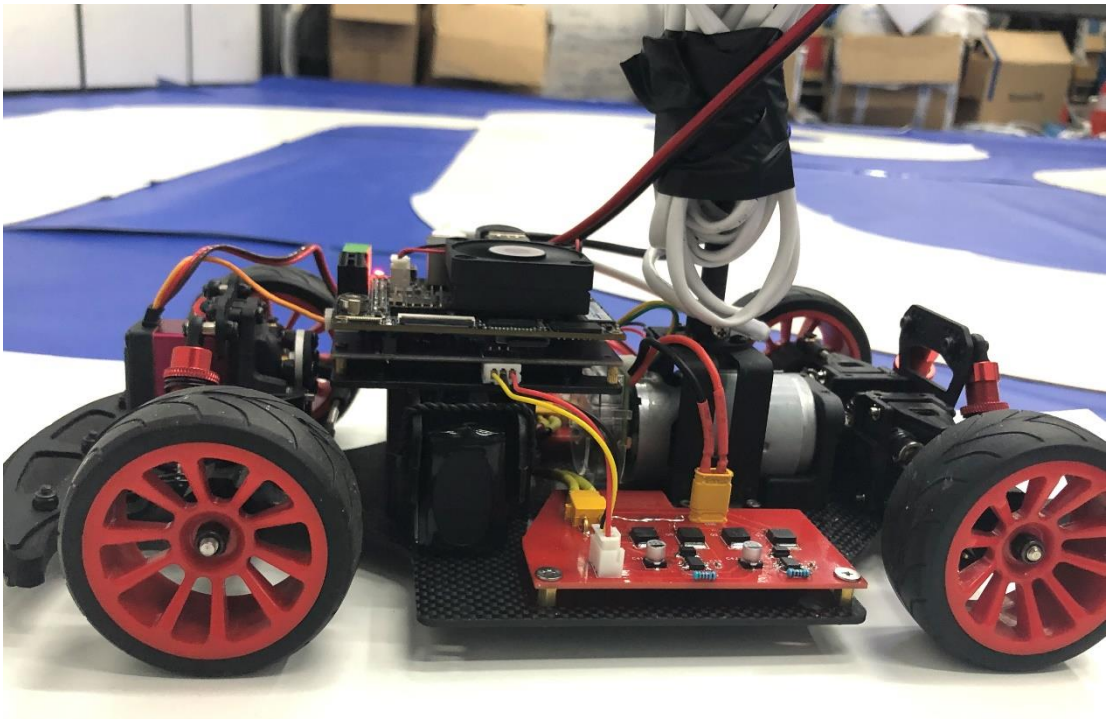


Figure 3.1: 驱动板、上位机、下位机位置关系

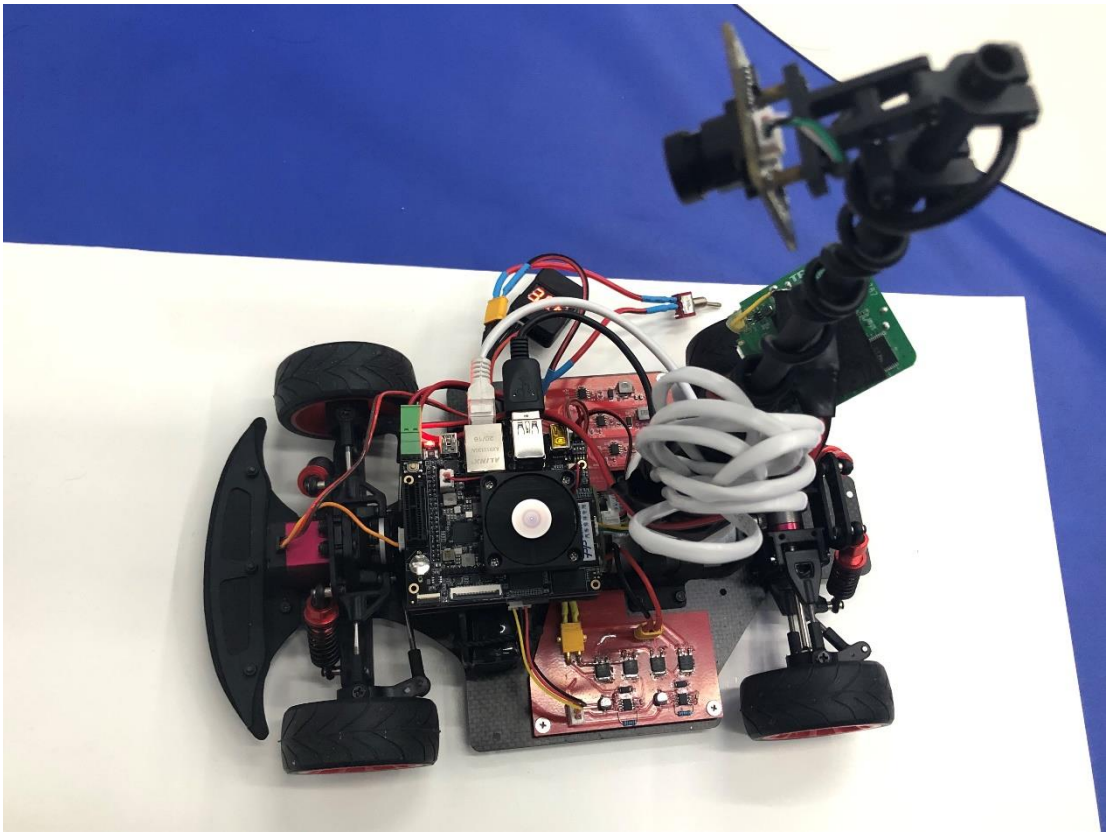


Figure 3.2: 驱动板、上位机、电源板位置关系以及路由器、摄像头

第四章 电路系统设计说明

除了上文提到的多板设计的方案以外，电路系统的设计还有一些方面需要进行详细说明。下位机使用英飞凌 TriCore 系列的 SAK-TC264D-40F200N_BC 芯片作为主控，主要功能是对电机的转速做 PID 控制、将上位机输出的舵机角度变成 PWM 信号给到舵机。除此之外，下位机上还配有 9 颗调试用的发光二极管和与上位机对应的 GPIO 插座、UART 插座。上位机的编程口采用 DAP 的接口，预留在靠电源板的一侧，并配有一个跳线帽用来选择板子是否由编程器供电。

上位机和下位机采用重叠的摆放方式，也就是上位机用过针脚直接插在下位机的插座上，既减少了不必要的引线，还保证了数字信号的传输线对其它电路的影响降到最低。在布局下位机的元器件时，考虑到电路板的安装方向和电源板、驱动板、舵机、电机编码器的位置，这几个接口的连接器被放到距离部件最近的边缘。不仅如此，选择引脚时也让所有信号线尽可能的短，以减少串扰。

由于前期不确定是否应用超声波传感器以及陀螺仪，所以板子上预留了 HC-SR04 和 JY62 的接口，以便后期迭代。

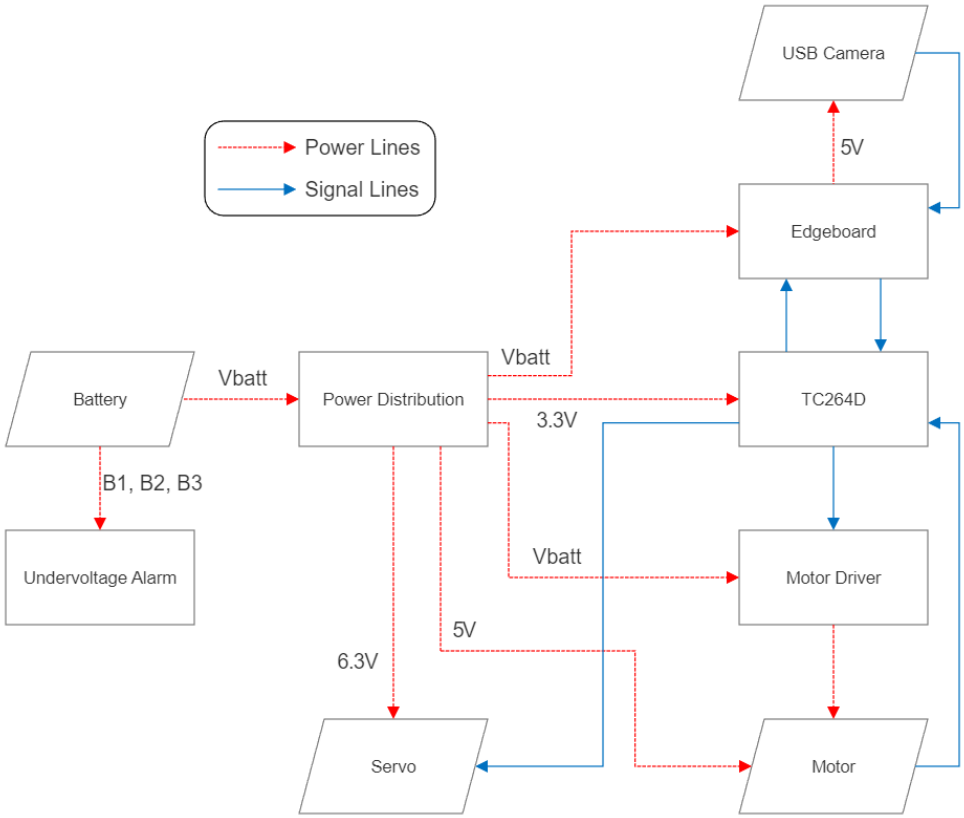


Figure 4.1: 小车电路系统整体的供电(虚线)与数据(实线)的连接关系

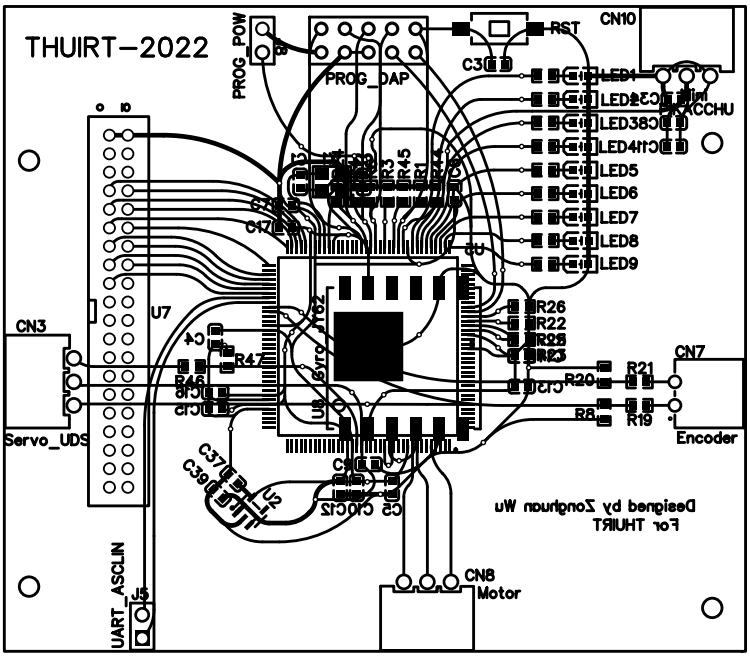


Figure 4.2: 上位机版图（顶层、底层、顶丝印层、底丝印层、多层）

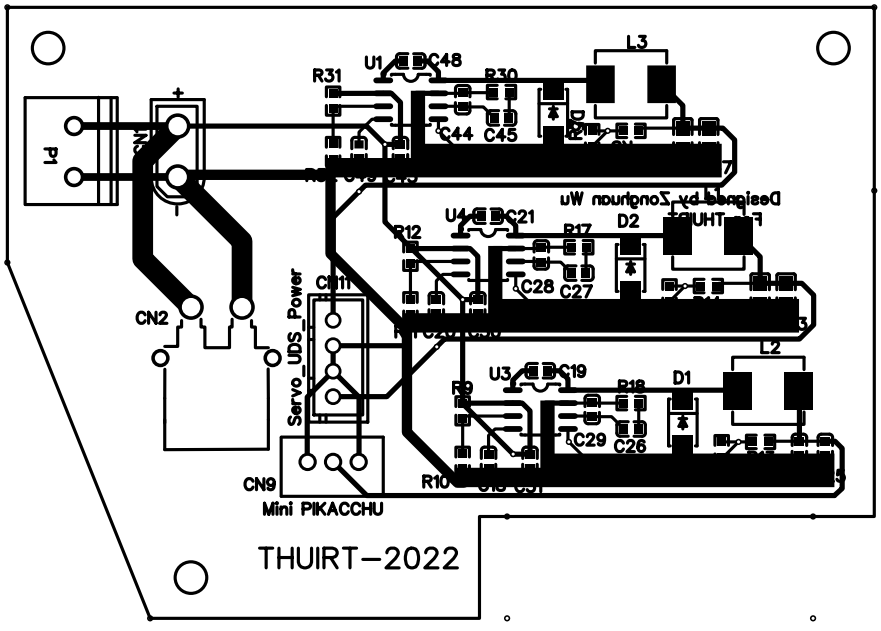


Figure 4.3: 电源板版图（顶层、底层、顶丝印层、底丝印层、多层）

第五章 控制软件主要理论、算法简略说明

车模当中软件算法主要分为上位机部分和下位机部分，上位机负责发送控制指令，下位机负责控制舵机和电机等运动控制。上位机通过摄像头采集到图像，传给单独线程的 Mobilenet 进行模型识别，检测到元素之后，将代码逻辑跳转到相应任务的处理模式下，完成相应的任务。

舵机的控制算法由上位机完成，电机的控制算法由下位机完成。下面分别从上位机和下位机两部分介绍我们的设计。

上位机采用位置式 PD 控制计算舵机的偏角：输入一个误差值，如图像左右两侧白色像素数量之差，结合上一个误差e，计算舵机偏角。假设采样为近似周期采样，离散自变量为 n，比例系数为K_p，微分系数为K_d，则离散 PD 控制的具体实现方法为：

$$u[n] = K_p \cdot e[n] + K_d\{e[n] - e[n-1]\}$$

下位机采用增量式 PID 控制计算输出给电机驱动的 PWM，与上位机的算法类似，只是在此基础上增加了积分项。若引入积分系数K_i，那么由上式改变可以得到离散 PID 控制算法：

$$u[n] = K_p \cdot e[n] + K_i \cdot \sum_{i=1}^{n} e[i] + K_d \cdot \{e[n] - e[n-1]\}$$

进而可以得到 n-1 时刻的控制量：

$$u[n-1] = K_p \cdot e[n-1] + K_i \cdot \sum_{i=1}^{n-1} e[i] + K_d \cdot \{e[n-1] - e[n-2]\}$$

设

$$\Delta u[n] = u[n] - u[n-1]$$

则

$$\Delta u[n] = K_p \cdot \{e[n] - e[n-1]\} + K_i \cdot e[n] + K_d \cdot \{e[n] - 2e[n-1] + e[n-2]\}$$

上位机通过 UART 将电机输出量和舵机转角发送给下位机。下位机不对舵机转角进行额外处理，而是将它转换成计数器的对应数字透传到寄存器上产生对应的 PWM。

第六章 开发工具链说明

下位机的开发工具为 Aurix Development Studio。上位机开发在本地完成，然后上传到 edge board 进行相应的调试。

第七章 结论

在整个备赛过程中，我们每个人都学到了很多，软件、硬件的知识，各种想法的数学验证、模型建立与实践，理性辩证思考问题的方式，当然还有与人合作、交流、相处的方法，学会相信他人，学会无论发生了什么，都不放弃。通过这次比赛，我们实验室团队每个成员之间都建立了深厚的友谊。

感谢比赛方提供的这样的一个比赛的机会，这既是让我们将自己的所思所学转化为真正的实践的机会，在这个过程中可以收获到比赛调试反复 debug、一点一点发现问题、解决问题的解决方法，也可以感受到一种只有付诸实践才能够学习到的知识，这是一段弥足珍贵的经历。

感谢英飞凌的大力帮助，I 型车模使用起来十分不错，感谢赛方卖给我们的车模，让我们顺利完成比赛，希望全国智能车比赛能够越办越好。

附录 A 模型车主要技术参数说明

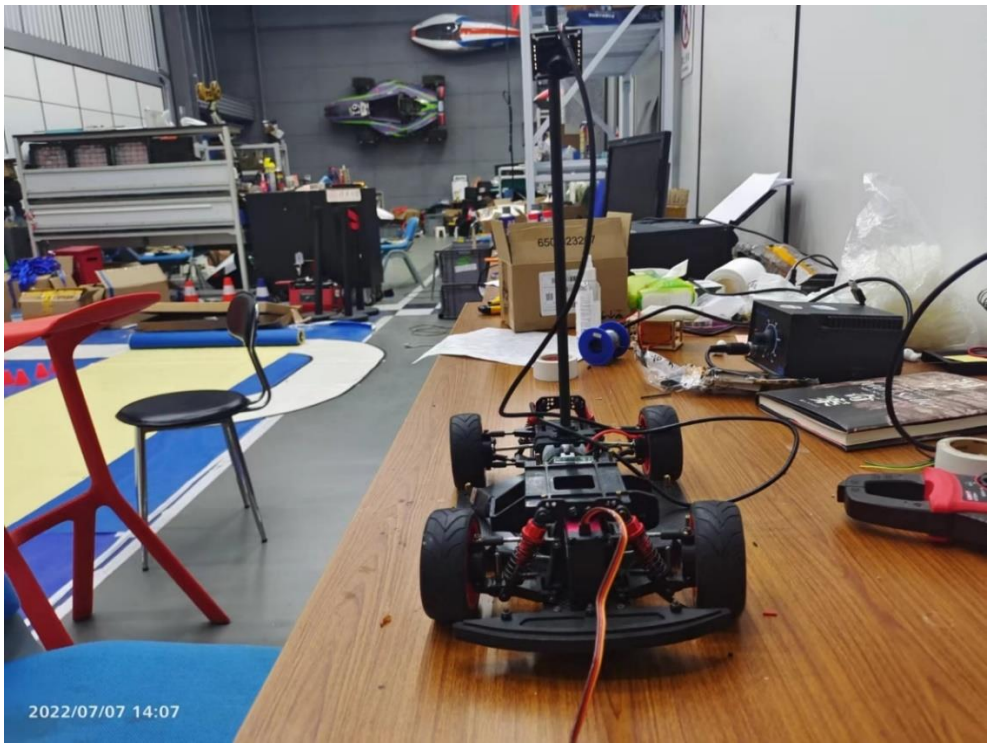
队伍名称	THU-IRT			
参赛学校	**清华大学**			
赛题组组别	**□ 四轮电磁　□ 四轮摄像头　□ 多车编队** **□ 平衡单车　□ 无线充电　□ 平衡信标** **□ 智能视觉　□ 极速越野　√ 完全模型**			
检查项目	**规格** (选手自行填写)	**符合** **(√)**	**不符合** **(×)**	**备注**
1.车模类型是什么?	赛事 I 型车模	√		如果是自制车模，请标明自制。
车模整体尺寸: 1.（包括传感器在内）长，宽，高(mm) 2. 对于无线充电组: 显示电能的 LED 板尺寸	313×192×370 完全模型组无限制	√		在填写是，请将所在组别规则对于车模尺寸限制同时进行填写。
1. 传感器种类、规格(型号)数量。	编码器 CSPE5-500×1 摄像头×1	√		
1. 控制转向舵机型号是否自行改装舵机? 2. 防伪易损标签是否完整?	CS-3120 否 是	√		
1. 是否增加伺服电机?	否	√		

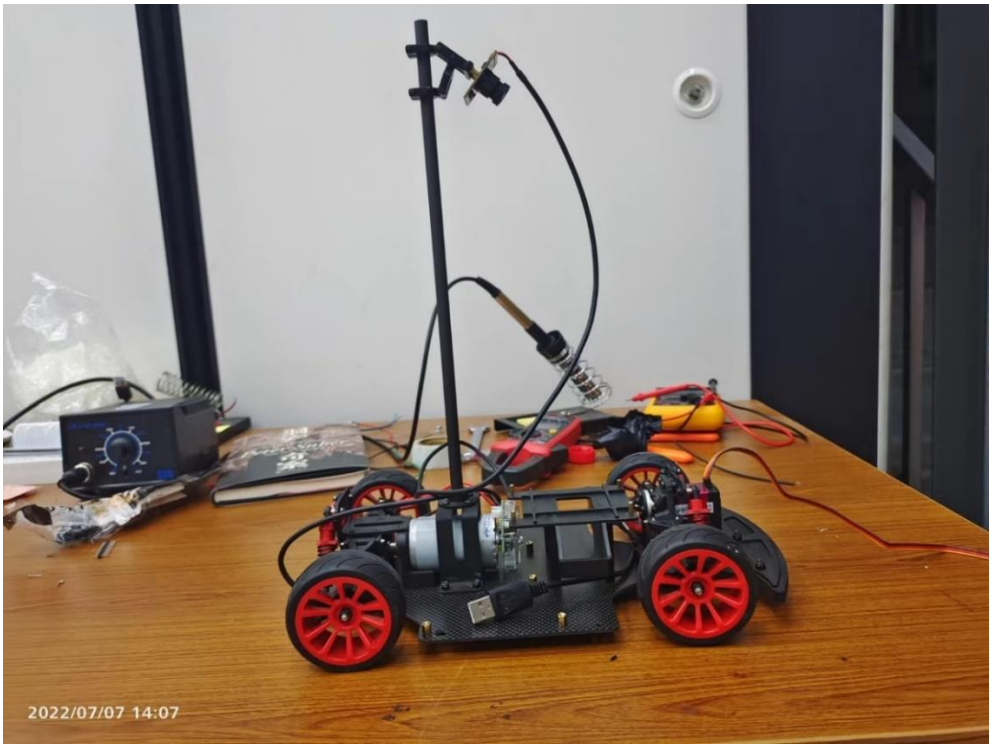
2. 如果有那么种类、个数和作用?				
1. 微处理器型号和个数? 2. 是否复合所在比赛组别要求?	TC264×1 FZ3B-Edgeboard×1 符合	√		
1. 是否具有其它可编程器件, 个数与作用?	否	√		
1. 是否有无线通讯装置? 2. 如果有, 那么种类和个数?	有,路由器×1	√		用于发车
1. 电池的种类、规格和数量?	CB-22003 3S1P 25C 24.42Wh 1个	√		
1. 是否有升压电路驱动舵机和后轮电机?	否	√		
1. 后轮驱动电机是否是原车模电机? 2. 是否具有防伪易损标签?	是 是	√		
1. 车模轮胎是否原有的纹理可辨析? 2. 轮胎表面是否具有粘性物质? 3. 对于麦克纳姆轮是否更换过小轮胶皮?	是 否	√		
1. 车模底盘是否是原车模底盘?	是	√		

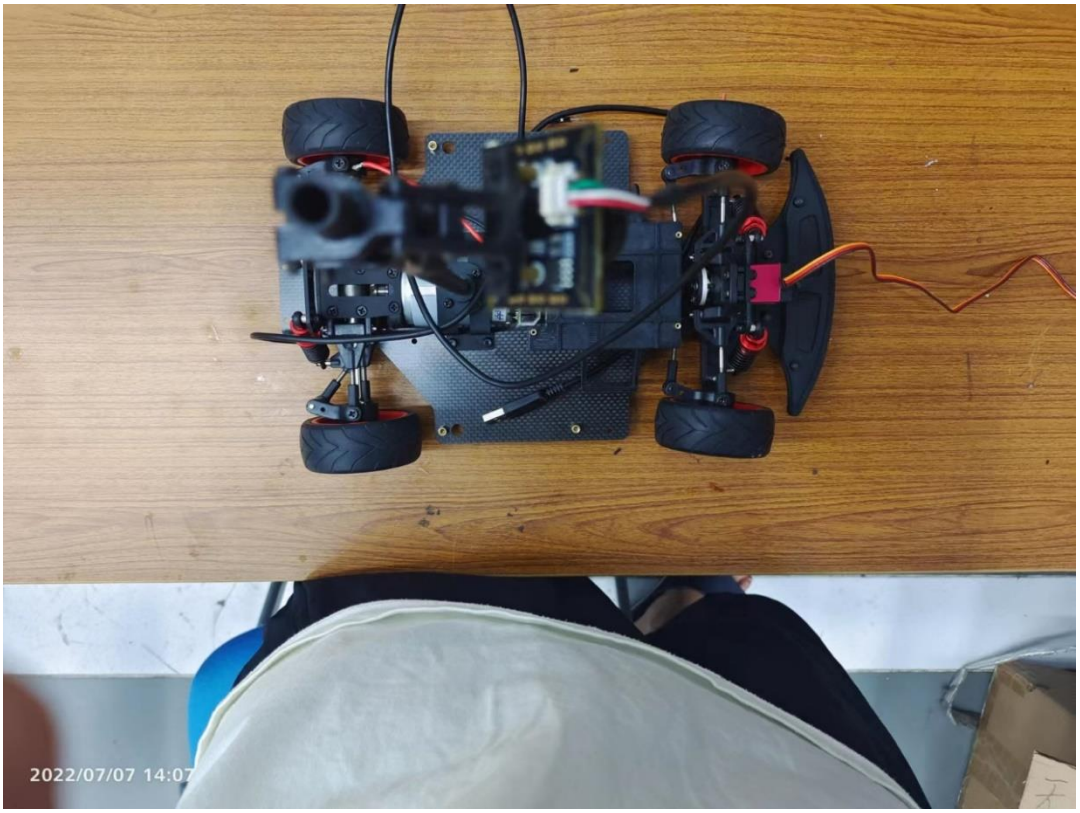
2. 是否有大面积切割?	否			
1. 车轮轴距、轮距是否改装? 2. 改装参数是什么?	否	√		
1. 车模驱动轮传动机构是否改装? 2. 改装方式是什么?	否	√		
1. 车模差速器是否改装? 2. 改装方式是什么?	否	√		
1. 车模零件是否更换或改装? 2. 更换和改装的方式什么?	否	√		。
1. 车模电路板个数及功能。 2. 其中是否有购买成品、哪一些?	四个: 上位机 Edgeboard, 下位机 TC264, 电源板, 驱动板 购买成品 FZ3B-Edgeboard	√		
1. 自制电路板是否标记有学校名称、队伍名称、制作日期等信息? 2. 标示信息在 PCB 的哪一层?	是 顶层和顶丝印层 内容: 名称与年份及作者	√		请在表格中注明电路板队伍信息的内容。
其它待说明内容		√		

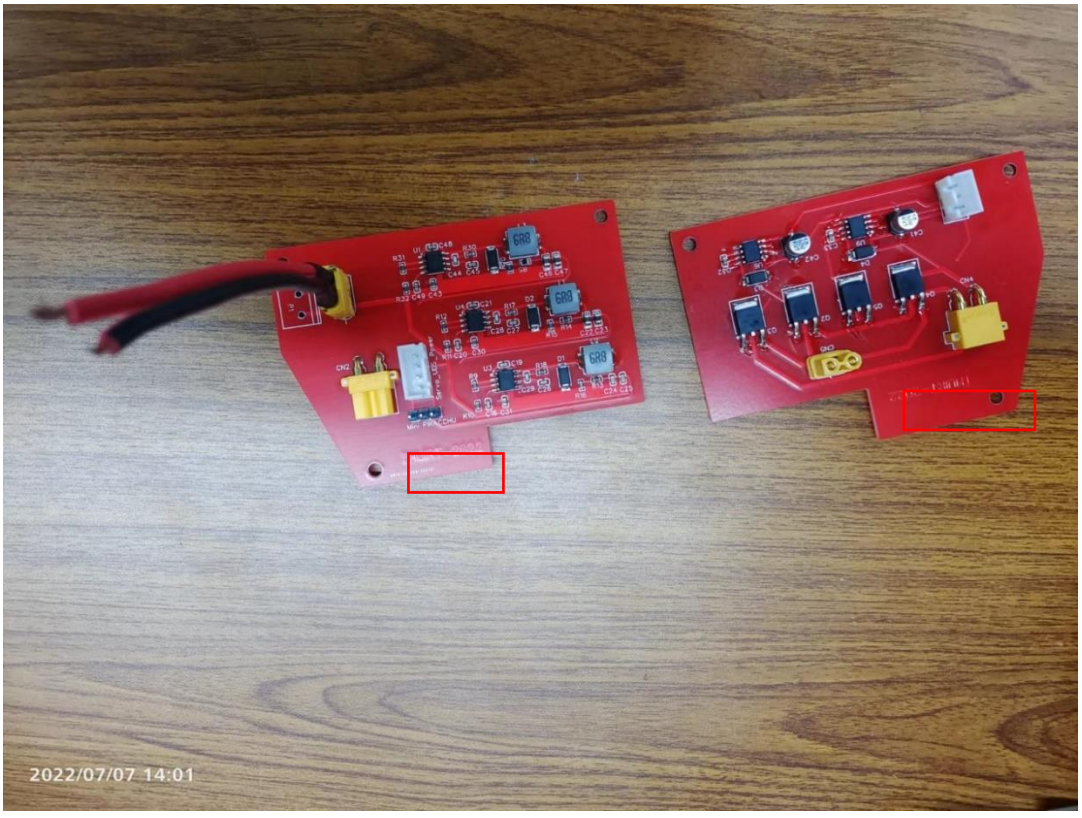
检查人员签名:	检查意见:			

附录 B 模型车及电路板照片

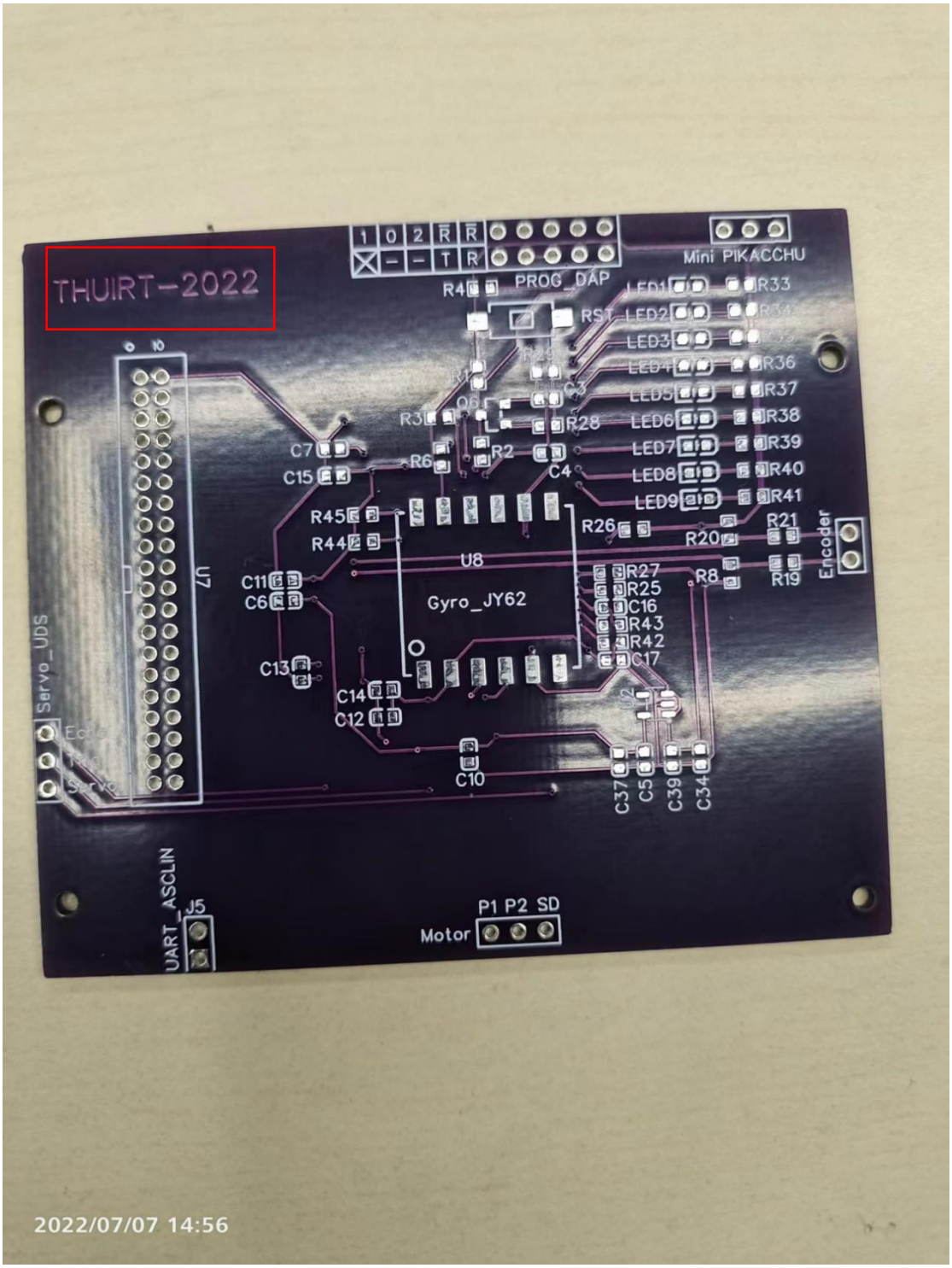


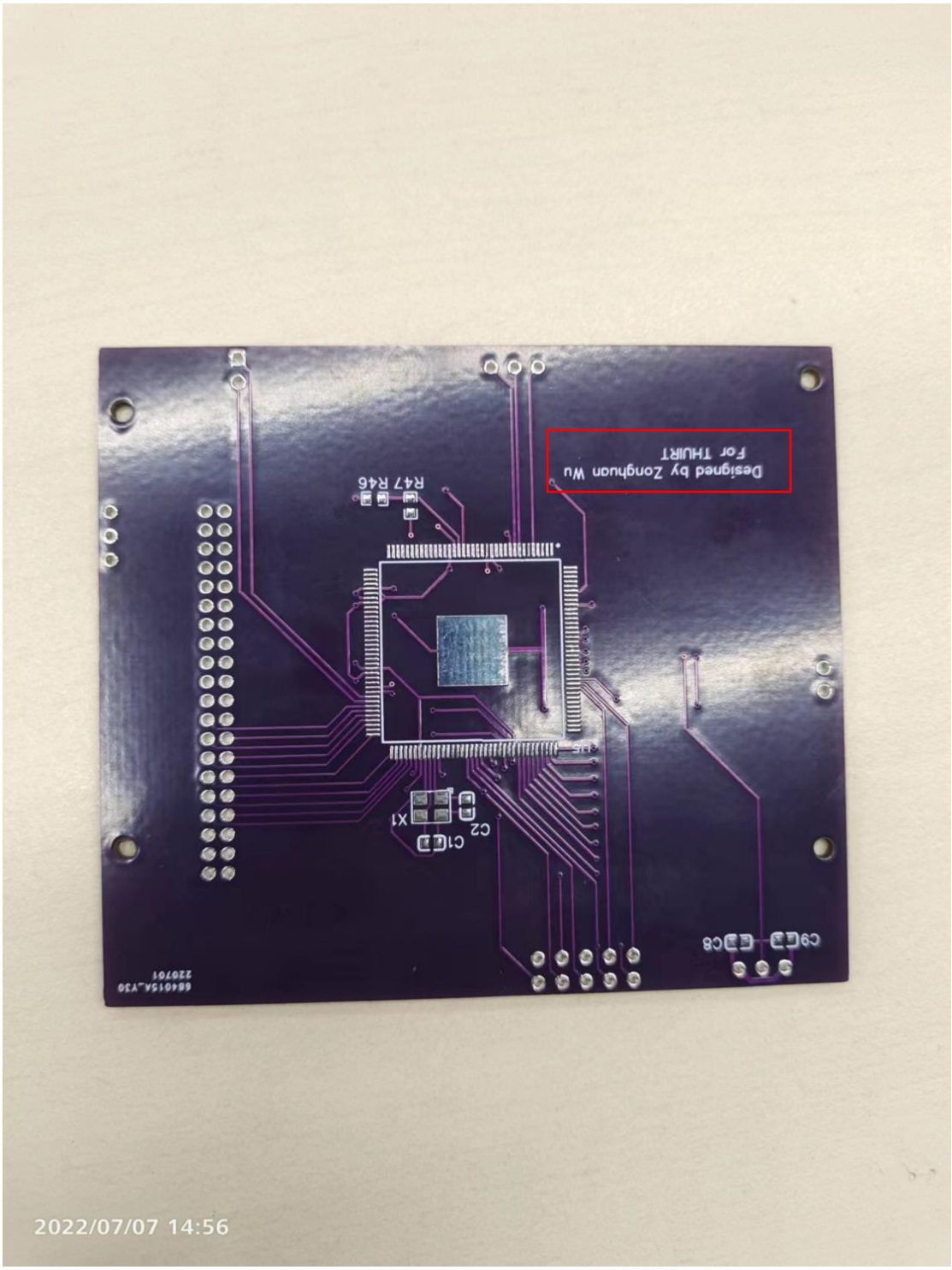


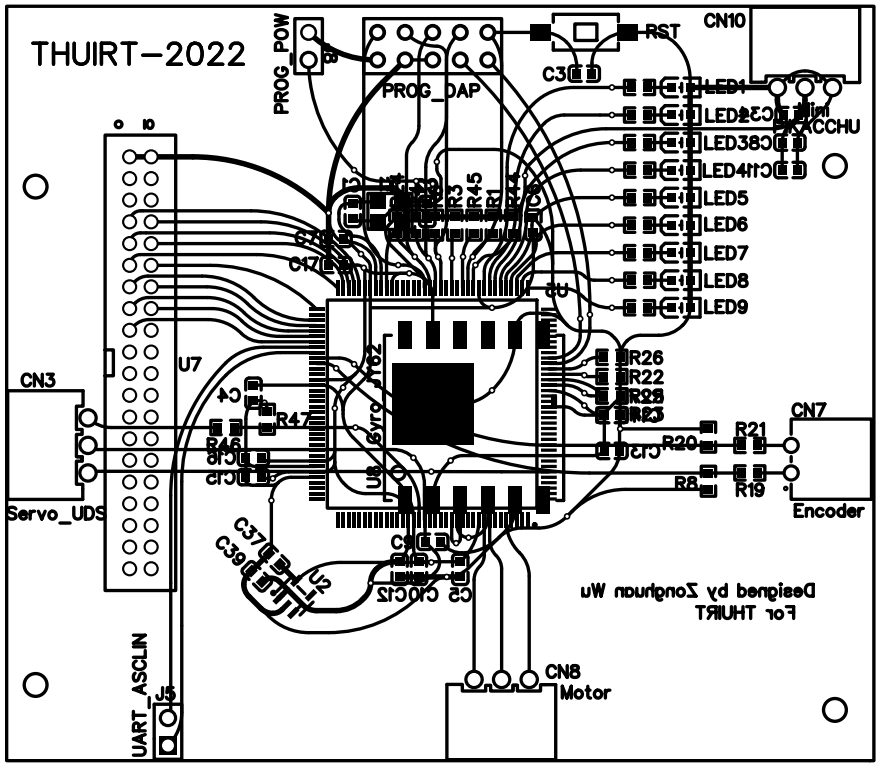

2022/07/07 14:07

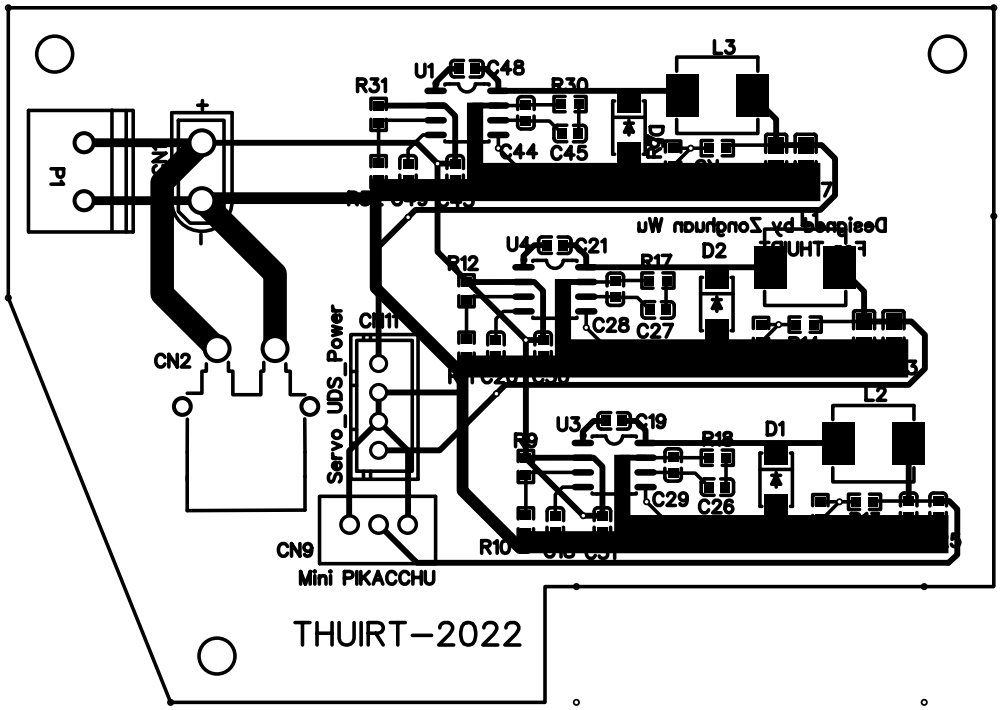

2022/07/07 14:01

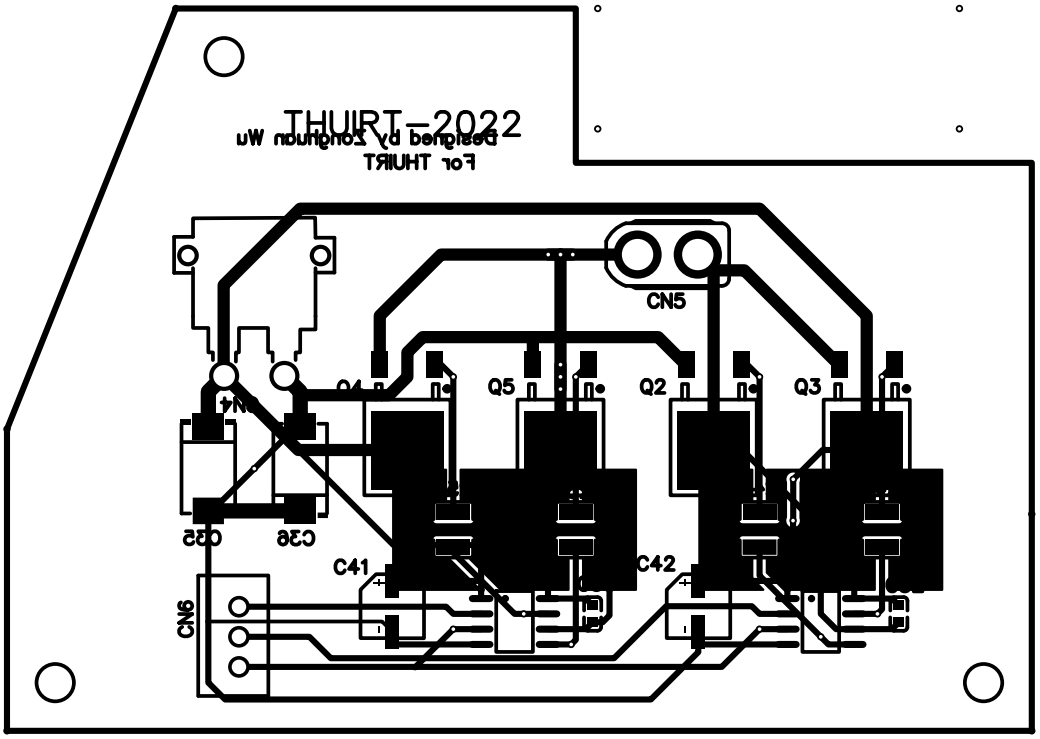

2022/07/07 14:02


THUIRT-2022
Mini PIKACCHU
PROG_DAP
Gyro_JY62
U8
U7
Servo_UDS
UART_ASCLIN
J5
P1 P2 SD
Motor
Encoder
2022/07/07 14:56


Designed by Zonghuan Wu
For THUIRT
R47 R46
X1
C2
C1
C9
C8
2022/07/07 14:56


THUIRT-2022
PROG_POW
PROG_DAP
RST
CN10
C3
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
U7
CN3
Servo_UDS
R47
R46
R26
R22
R21
R20
R8
R19
CN7
Encoder
U2
U5
CN8
Motor
UART_ASCLN
J5
Designed by Zongjun Wu
For THUIRT


L3
U1
C48
R31
R30
C44
C45
P1
U4
C21
R12
R17
D2
C28
C27
CN11
CN2
Servo_UDS_Power
L2
U3
C19
R9
R18
D1
C29
C26
CN9
Mini PIKACCHU
R10
THUIRT-2022


THUIRT-2022
CN5
Q4
Q5
Q2
Q3
C41
C42
CN6

附录 C 程序源代码

下位机 CPU1

```
#include "./Libraries/user_lib.h"
IfxCpu_syncEvent g_cpuSyncEvent = 0;

char c;
float dist;
//IfxGtm_Tim_In echo;
int core0_main(void)
{
    get_clk();
    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    gpio_init_pins();
    uart_init(2, 115200, UART_TX, UART_RX);
    gpt12_init(COUNTER, DIRECTION);
    gtm_pwm_init(&PWM0, 25000);
    gtm_pwm_init(&PWM1, 25000);
    gtm_pwm_init(&PWM2, 50);
    pwm_set_duty(&PWM0, 0);
    pwm_set_duty(&PWM1, 0);
    set_pin_high(IFXCFG_PORT_MOTOR_SDN);

    pit_interrupt_ms(0, 0, PID_PERIOD);
    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
    IfxCpu_enableInterrupts();
    while(1)
    { }
    return 1;
}
```

下位机 CPU0

```
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"

extern IfxCpu_syncEvent g_cpuSyncEvent;

int core1_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG1 IS DISABLED HERE!!
     * Enable the watchdog and service it periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    while(1)
    {
    }
    return (1);
}
```

下位机中断

```
#include "./Libraries/user_lib.h"

/* PID */
sint16 cnt = 0;
int goal = 0;
float Kp = 5, Ki = 1, Kd = 1;
float err = 0, derr = 0, dderr = 0;
float speed = 0;
float output = 0;

/* servo */
uint8 speed_ticks_per_tenms = 0;
sint8 theta = 0;
float duty;

/* uart */
int state = 0;
uint8 data[3];

int test = 0;
#define cnt_per_cycle 340.0
#define radius 3.0f

/* communication with edgeboard */
// speed, theta, speed, theta^speed(check)
IFX_INTERRUPT(uart2_rx_isr, 0, 10)
{
    //test++;
    IfxCpu_enableInterrupts();
    IfxAsclin_Asc_isrReceive(&uart_handle);
    int data = IfxAsclin_Asc_blockingRead(&uart_handle);
    if (data <= 20)   speed_ticks_per_tenms = (data ? data * 2 + 50 : 0);
    else if (data >= 30 && data <= 210){
        duty = Servo_Mid_Val * GTM_ATOM0_PWM_DUTY_MAX + (data - 120) / 90.0 *
150.0;
        if (duty > Servo_Mid_Val * GTM_ATOM0_PWM_DUTY_MAX + 150)
            duty = Servo_Mid_Val * GTM_ATOM0_PWM_DUTY_MAX + 150;
```

```
            if (duty < Servo_Mid_Val * GTM_ATOM0_PWM_DUTY_MAX - 150)
                duty = Servo_Mid_Val * GTM_ATOM0_PWM_DUTY_MAX - 150;
            pwm_set_duty(&PWM2, duty);
        }
        IfxPort_togglePin(IFXCFG_PORT_LED_1.port, IFXCFG_PORT_LED_1.pinIndex);
        state = (state + 1) % 3;
        //state = -1;
}

IFX_INTERRUPT(uart2_tx_isr, 0, 9)
{
        IfxCpu_enableInterrupts();
        IfxAsclin_Asc_isrTransmit(&uart_handle);
}

IFX_INTERRUPT(uart2_er_isr, 0, 11)
{
        IfxCpu_enableInterrupts();
        IfxAsclin_Asc_isrError(&uart_handle);
}

IFX_INTERRUPT(cc60_pit_ch0_isr, 0, 8)
{
        IfxCpu_enableInterrupts();
        PIT_CLEAR_FLAG(0, 0);
        cnt = gpt12_get();
        gpt12_clear();
        speed = cnt;
        goal = speed_ticks_per_tenms * 3;
        //if (goal != 0) set_pin_high(IFXCFG_PORT_LED_2);
        char speed_str[20];
        //sprintf(speed_str, "%f", speed);
        //uart_put_string(0, (uint8*)speed_str);
        dderr = goal - speed - err - derr;
        derr = goal - speed - err;
        err = goal - speed;
        output += Kp * derr + Ki * err + Kd * dderr;


        if (output > 3000)
```

```
    {
        output = 3000;
    }
    else if (output < 0)
    {
        output = 0;
    }

    pwm_set_duty(&PWM0, (uint32)output);
}

IFX_INTERRUPT(cc60_pit_ch1_isr, 0, 29)
{
    IfxCpu_enableInterrupts();
    PIT_CLEAR_FLAG(0, 0);
    IfxPort_setPinLow(ultrasound_trig.port, ultrasound_trig.pinIndex);
    systick_delay_us(0, 8);
    IfxPort_setPinHigh(ultrasound_trig.port, ultrasound_trig.pinIndex);
    systick_delay_us(0, 20);
    IfxPort_setPinLow(ultrasound_trig.port, ultrasound_trig.pinIndex);
}
```

上位机

```
#include <opencv2/opencv.hpp>
#include "core/predictor.hpp"
#include "core/uart_test.hpp"
#include <unistd.h>
#include <cstdio>
#include <string>
#include <ctime>
#include <chrono>
#include <thread>
using namespace cv;

float mat[3][3] = {2.09508877e+00, 4.21555464e+00, -3.56540157e+02,
                   2.77740867e-02, 8.15669436e+00, -6.55965763e+02,
                   8.55618540e-05, 1.31347491e-02, 1.00000000e+00};
Mat perspectiveMat(3, 3, CV_32F, mat);
Mat mask;

#define length 640
#define width 480
Mat img;
std::shared_ptr<Driver> driver;
const int bias = 300;
const int threshold_cnt = 50;
const int max_gap = 10;
const int default_start_row = 400;
const int default_end_row = 320;
bool roundabout_enable = true;
bool outgarage_enable = true;
bool ingarage_enable = false;
bool flooded_area_enable = false;
bool end_predict_flag = false;
int colLeftBoundary[440], colRightBoundary[440];
int round_frame = 0;
long long frameCnt = 0;

int speed = 0;
```

```
struct RoundaboutRow
{
    int len, id, idLeftBound;
};

struct GarageRow
{
    int len, id, idRightBound;
};

void perspective(Mat img, Mat &dist)
{
    warpPerspective(img, dist, perspectiveMat, Size(length, width), 1, 0, Scalar(0, 0, 0));
}

void predict()
{
    std::string config_path = "/root/workspace/mobilenet-ssd";
    Predictor predictor(config_path);
    predictor.init();
    while (1)
    {
        Mat tmp_img = img;
        std::vector<PredictResult> results = predictor.run(tmp_img);
        for (int i = 0; i < results.size(); i++)
        {
            PredictResult result = results[i];
            if (result.type == 4)
            {
                if (result.y > 200 && result.y < 400 && result.score > 0.9)
                {
                    std::this_thread::sleep_for(std::chrono::milliseconds(3 * result.y));
                    flooded_area_enable = true;
                    printf("state: enable flooded area!\n");
                }
            }
        }
        if (end_predict_flag)
            return ;
    }
```

```
}


int Process(Mat img)
{
    Mat imgSplit[3];
    split(img, imgSplit);                                                    //绿色域灰度化
    threshold(imgSplit[1], imgSplit[1], 120, 255, THRESH_BINARY); //二值化
    perspective(imgSplit[1], imgSplit[1]);                                   // 透视

    int sumWhites = 0, differenceWhites = 0;
    if (outgarage_enable)
    {
        for (int i = default_end_row; i <= default_start_row; i++)
        {
            if (imgSplit[1].at<uchar>(i, colLeftBoundary[i]) && imgSplit[1].at<uchar>(i,
colRightBoundary[i]))
            {
                sumWhites += 290 - colLeftBoundary[i];
                differenceWhites += colLeftBoundary[i] - 260;
            }
        }
    }

    if (!sumWhites)
    {
        int rightCnt = 0, leftCnt = 0;
        for (int i = bias; i < 100 + bias; i++)
        {
            if (imgSplit[1].at<uchar>(i, colLeftBoundary[i]))
                leftCnt++;
            if (!imgSplit[1].at<uchar>(i, colRightBoundary[i]))
                rightCnt++;
        } //初筛

        if (rightCnt > threshold_cnt && leftCnt > threshold_cnt)
        {
            int totCnt = 0;
            for (int i = bias; i <= bias + 100; i++)
            {
```

```
                    if (imgSplit[1].at<uchar>(i, colRightBoundary[i]) || !imgSplit[1].at<uchar>(i,
colLeftBoundary[i]))
                        continue;
                    int curColor = 0, changeCnt = 0, gap = 0;
                    for (int j = colRightBoundary[i]; j > colLeftBoundary[i]; j--)
                    {
                        if (imgSplit[1].at<uchar>(i, j) < 255)
                            imgSplit[1].at<uchar>(i, j) = 0;
                        if (imgSplit[1].at<uchar>(i, j) != curColor && gap >= 8)
                            changeCnt++, curColor = imgSplit[1].at<uchar>(i, j), gap = 0;
                        else
                            gap++;
                        if (gap >= 20)
                            changeCnt = 0, curColor = curColor ? 0 : 255;
                        if (changeCnt > 12)
                        {
                            totCnt++;
                            break;
                        }
                    }
                }
                if (totCnt >= 30 && ingarage_enable)
                {
                    printf("state: get into garage!\n");
                    driver->senddata(205);
                    driver->senddata(5);
                    waitKey(1200);
                    driver->senddata(0);
                    driver->senddata(120);
                    driver->senddata(0);
                    driver->senddata(120);
                    driver->senddata(0);
                    driver->senddata(120);
                    end_predict_flag = true;
                    printf("frameCnt: %lld\n", frameCnt);
                    exit(0);
                }
            else if(totCnt >= 30 && end_predict_flag)
            {
            end_predict_flag = false;
```

```
            std::thread predictThread_1(predict);
            predictThread_1.detach();
            }
            }
        }
        if (roundabout_enable && !sumWhites) //是否考虑环岛
        {
            int rightCnt = 0;
            int leftCnt = 0;
            for (int i = bias; i < 100 + bias; i++)
            {
                if (imgSplit[1].at<uchar>(i, colRightBoundary[i]))
                    rightCnt++;
                if (!imgSplit[1].at<uchar>(i, colLeftBoundary[i]))
                    leftCnt++;
            } //初筛

            if (rightCnt > threshold_cnt && leftCnt > threshold_cnt)
            {
                std::vector<RoundaboutRow> roundaboutRows; //保存环岛行的相关信息
                for (int i = bias; i <= bias + 100; i++)
                {
                    if (imgSplit[1].at<uchar>(i, colLeftBoundary[i]) || !imgSplit[1].at<uchar>(i,
colRightBoundary[i]))
                        continue;
                    RoundaboutRow tmp = {0, i, 0};
                    int gap = 0;
                    for (int j = colLeftBoundary[i] + 1; j <= colRightBoundary[i]; j++)
                    {
                        if (imgSplit[1].at<uchar>(i, j))
                        { //白色像素
                            tmp.len++, gap = 0;
                        }
                        else if (tmp.len)
                            gap++; //非白色像素且前已有白色像素
                        if (gap > max_gap)
                            break; //判断为白线中断
                    }
                    tmp.idLeftBound = colRightBoundary[i] - tmp.len;
                    // imgSplit[1].at<uchar>(i, tmp.idLeftBound) = 128;
```

```
                    if (tmp.idLeftBound <= 200 || tmp.idLeftBound >= 260)
                        continue;
                    //左端点落到需要落到合适范围
                    // if (!roundaboutRows.empty() && roundaboutRows.back().id - tmp.id >
max_gap)
                    //      break;
                    //环岛行也要求连续
                    roundaboutRows.push_back(tmp);
                }
                // printf("#satisfied rows: %d\n", (int)roundaboutRows.size());
                if (roundaboutRows.size() > threshold_cnt)
                {
                    int sz = roundaboutRows.size();
                    RoundaboutRow beginRow, endRow, midRow;
                    beginRow = roundaboutRows.front();
                    endRow = roundaboutRows.back();
                    midRow = *(roundaboutRows.begin() + sz / 2);
                    int l1 = beginRow.idLeftBound, l2 = midRow.idLeftBound, l3 =
endRow.idLeftBound;
                    if (abs(l1 - l2) + abs(l2 - l3) + abs(l1 - l3) < 3 * max_gap)
                    {
                driver->senddata(7);
                        if (round_frame < frameCnt - 10)
                            round_frame = frameCnt;
                        end_predict_flag = true;
                        printf("state: being roundabout!\n");
                        /* 识别为环岛 */
                        if (round_frame >= frameCnt - 3)
                            waitKey(1450);
                if (round_frame == frameCnt - 4) waitKey(600);
                        return -75;
                    }
                }
            }
        }

        /* 非环岛的情况 */
        if (!sumWhites)
        {
            for (int i = default_start_row; i >= default_end_row-(speed-10)*5; i--)
```

```
            {
                for (int j = 0; j < imgSplit[1].cols; j++)
                {
                    // printf("%d %d\n", i, j);
                    if (imgSplit[1].at<uchar>(i, j))
                    {
                        sumWhites++;
                        j > 319 ? differenceWhites++ : differenceWhites--;
                    }
                }
            }
        }

        int ret;
        if (sumWhites == 0)
            ret = 0;
        else
        {
        float tmp = (float)differenceWhites / (float)sumWhites;
        tmp = tmp < 0? - tmp * tmp: tmp * tmp;
            ret = int(-90 * tmp);
        }
        //char c[50];
        //sprintf(c, "%d", ret);
        //imshow(c, imgSplit[1]);
        //waitKey(0);
        return ret;
}

int FloodedAreaProcess(Mat img)
{
        if (!flooded_area_enable)
            return 0;
        Mat imgSplit[3];
        split(img, imgSplit);                                                    //红色，蓝色域
灰度化
        threshold(imgSplit[2], imgSplit[2], 120, 255, THRESH_BINARY_INV); //二值化
        // threshold(imgSplit[0], imgSplit[0], 55, 255, THRESH_BINARY);
        // multiply(imgSplit[0], imgSplit[2], img);
        img = imgSplit[2];
```

```
perspective(img, img); //  透视

float times = 1;
int RsumWhites = 0, RdifferenceWhites = 0;
int Rsumlen = 0, Rrows = 0;
for (int i = 430; i >= 250; i--)
{
    int Rlen = 0, Rgap = 0, j = colRightBoundary[i];
int blackLen = 0;
    for (; j >= colLeftBoundary[i]; j--)
    {
        if (img.at<uchar>(i, j) < 255)
            img.at<uchar>(i, j) = 0;
        if (!img.at<uchar>(i, j)&&!Rlen)
    {
    blackLen ++;
    }
    // printf("%d %d\n", i, j);
        if (img.at<uchar>(i, j))
    {
    if (Rgap > max_gap && Rgap < 3*max_gap)
    {
        times = 2;
        break;
    }
    else if (Rgap > 3*max_gap){
        times = 1.25;
        break;
    }
            Rgap = 0, Rlen++, RsumWhites++, j < 319 ? RdifferenceWhites-- :
RdifferenceWhites++;
    }
        else if (Rlen)
            Rgap++;
    }
    if (Rlen > 80)
        Rrows++, Rsumlen += Rlen;
    if (Rrows > 5 && Rrows < 20    && blackLen > 250 && !img.at<uchar>(i, 319))
    {
        printf("%d, %d\n", i, j);
```

```
                printf("state: get out of flooded area\n");
            imwrite("1.jpg", img);
                flooded_area_enable = false;
                return 0;
            }
        }
        //imshow("1", img);
        //waitKey(0);
        //imshow("2", imgSplit[0]);
        //waitKey(0);
        //imshow("3", imgSplit[2]);
        //waitKey(0);
        int sumWhites, differenceWhites;
        sumWhites = RsumWhites, differenceWhites = RdifferenceWhites * times;

        if (sumWhites < abs(differenceWhites))
            sumWhites = abs(differenceWhites);
        if (sumWhites == 0)
            return 0;
        int ret = int(-90 * (float)differenceWhites / (float)sumWhites);
        return ret;
}

int main()
{
        waitKey(5000);
        VideoCapture capture("/dev/video0");
        if (!capture.isOpened())
        {
            printf("camera not opened\n");
            return 1;
        }

        driver = std::make_shared<Driver>("/dev/ttyPS1", BaudRate::BAUD_115200);
        if (driver == nullptr)
        {
            printf("create driver error\n");
            return 1;
        }
        if (driver->open())
```

```
    {
        printf("driver open failed\n");
        return 1;
    }
    std::thread predictThread(predict);
    predictThread.detach();
    mask = imread("/root/workspace/image/mask.jpg", IMREAD_GRAYSCALE);
    for (int i = 0; i < 440; i++)
        for (int j = 0; j < mask.cols; j++)
            if (!mask.at<uchar>(i, j))
                colLeftBoundary[i] ? colRightBoundary[i] = j - 1 : colLeftBoundary[i] = j +
1;

    // resize(mask, mask, Size(length, width), 0, 0, INTER_NEAREST);
    char i = 45;
    char file[100];
    while (1)
    {
        // auto begin = std::chrono::high_resolution_clock::now();
        frameCnt++;
        if (frameCnt > 1100)
            ingarage_enable = true;
        if (frameCnt - round_frame > 30 || frameCnt - round_frame < 8)
    {
            roundabout_enable = true;
        //if (frameCnt - round_frame == 31 && round_frame)
        //{
        //    end_predict_flag = false;
        //      std::thread predictThread_1(predict);
        //    predictThread_1.detach();
        //}
    }
    else
            roundabout_enable = false;
        if (frameCnt < 100)
            outgarage_enable = true;
        else
            outgarage_enable = false;

        // sprintf_s(file, "D:\\VS_HW\\smartcar\\Project1\\test1\\test1\\img1\\%d.jpg", i);
```

```
            // printf("%s", file);
            // img = imread(file);
            while (!capture.read(img))
                ;
            resize(img, img, Size(length, width), 0, 0, INTER_NEAREST);
            int angle = 0;
            if (flooded_area_enable)
                angle = FloodedAreaProcess(img);
            else
                angle = Process(img);
            // if (error_flooded && !flooded_area_enable)
            //         imwrite("error.jpg", img);
            speed = 10 - abs(angle) / 90.0 * 4.5;
        // if (speed < 6) speed = 6;
            angle += 120;
            if (abs(angle - 120) < 60) speed += 5;
        if (abs(angle - 120) < 10) speed += 5;
        //driver->senddata(20);
        //waitKey(0);
        driver->senddata(speed);
            // driver->senddata(0);
            driver->senddata(angle);
        }

        return 0;
}
```