现代电子系统设计

创新实验

智能物流分拣系统

学　　号　　**2017011589**

姓　　名　　**吾尔开希**

专　　业　　**自动化**

日　　期　　**2019.7.1---7.12**

目录

1. 实验内容

小车上载着配送物（用海绵代替），小车自动巡航到达机械臂旁，同时利用超声检测距离，向机械臂发送指令，以及小车的 xy 轴坐标位置。机械臂根据小车的位置解算出各个舵机的旋转角，抓取配送物，并将其放到旁边的容器中。

我在该实验中完成的内容为：超声波测距、小车电机驱动、机械臂基础驱动、陀螺仪模块测角度、小车整体逻辑的完成，其他部分请参见我队友的报告。

2. 设计方案

2.1 超声波测距确定小车位置

机械臂抓取物体需要物体的准确坐标，因此小车需要在 xy 轴方向测量距离。首先，小车沿着 x 轴方向前进，遇到障碍物时停止，并记录此时与障碍物的距离。原地左转，沿着 y 轴方向前进，遇到障碍物停止，并记录距离。将这两个距离发送给机械臂，机械臂就能确定小车的具体方位，而配送物的高度是确定的，所以机械臂就能准确抓到配送物。

2.2 陀螺仪模块测角度

我们使用的是 jy61 陀螺仪，其角度以 100Hz 的频率通过串口发送，有确定的通信协议包，因此我们通过串口与之通信，并在看门狗时间中断内对接收到的包进行解码。

2.3 小车的方向控制

小车电机使用 L298N 模块进行驱动，一个电机有两个 PWM 输入口，不同的 PWM 占空比可以让电机以不同的速度转动。两个 PWM 输入口的高低电平关系可以控制电机的方向。

为了使小车沿特定的方向前进，我们对陀螺仪测到的角度进行 PID，利用 PID 结果对电机 PWM 的输入进行校正，从而使小车方向稳定。

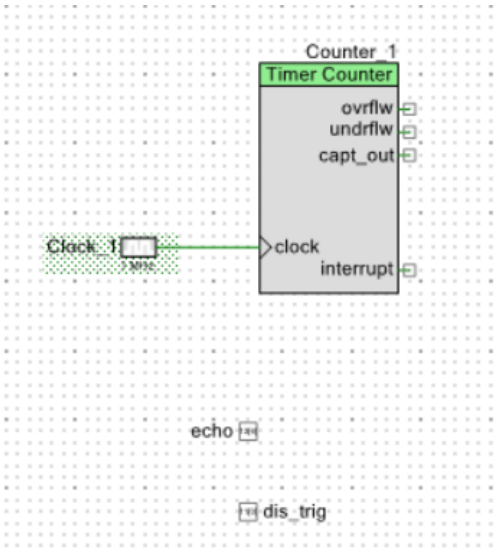
2.4 机械臂基础驱动

舵机与 Psoc 可以通过串口进行通信，按照特定的协议向舵机发送串口信息，即可驱动舵机转到特定的角度。

3. 电路图

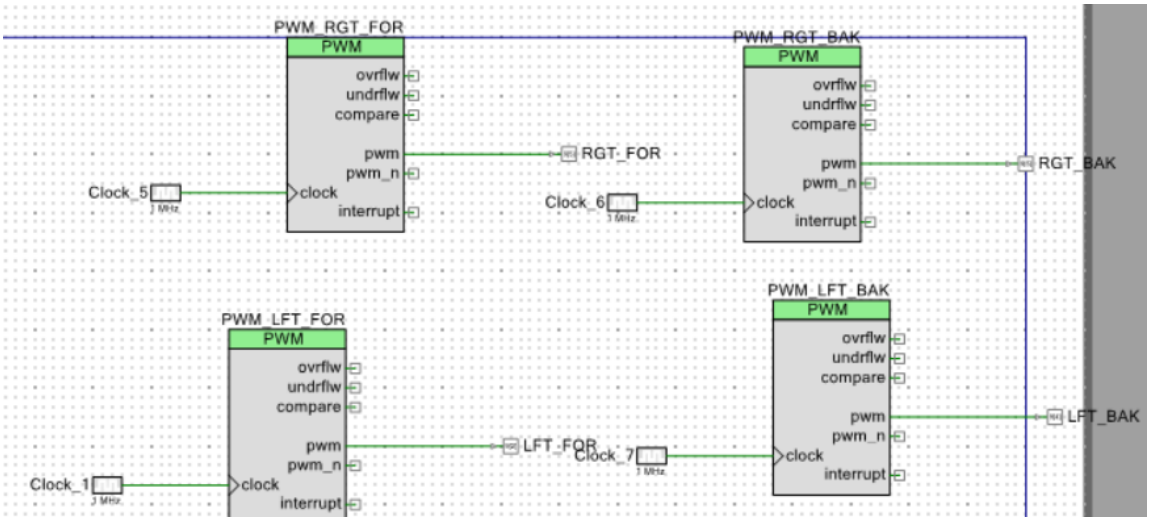
3.1 超声波模块

超声波测距需要控制 dis_trig 输出引脚、echo 输入引脚与一个计数器。计数器用来测量 echo 信号高电平时长。



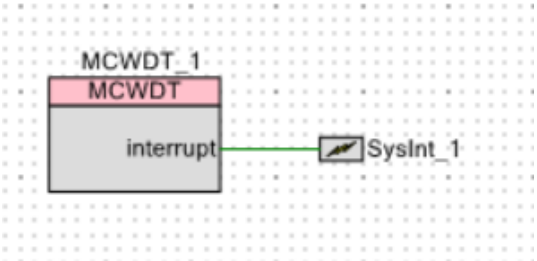
3.2 小车电机驱动

小车电机分为左电机与右电机，分别用两个 PWM 输出进行控制，所以一共有四个 PWM 输出，四个输出引脚。



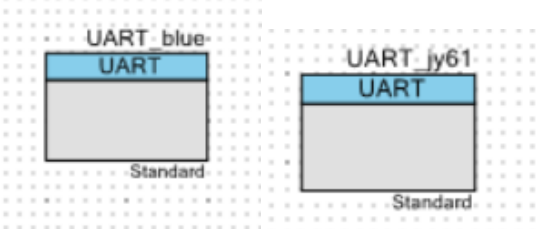
3.4 陀螺仪的时间中断

陀螺仪的输出需要间隔一段时间来进行读取与解码，我们使用看门狗时间中断来实现，在中断函数中完成上述操作。



3.3 串口通信

机械臂舵机、陀螺仪模块、电脑串口输出均使用 UART 串口通信，波特率均为 115200.



4. 模块选择与参数设置

4.1 超声波模块

超声波测距需要控制 dis_trig 输出引脚、echo 输入引脚与一个计数器。

dis_trig 驱动模式为 strong drive；超声模块的 echo 输出驱动能力较弱，所以 echo 引脚的驱动模式为 resistive pull up。

超声测距模块分辨率为 1mm，对应的 Echo 信号高电平时间：$T_h = 2 \times \frac{0.001}{340} \approx 5.88 \times 10^{-6} s$因此驱动计数器的时钟频率 f>170068Hz，取 f=1MHz。

距离的计算公式$\mathrm{x} = \mathrm{T_h} \times \frac{340}{2} m = 0.017n\ cm$，n 指的是 Echo 信号高电平时间内计数器的计数次数。

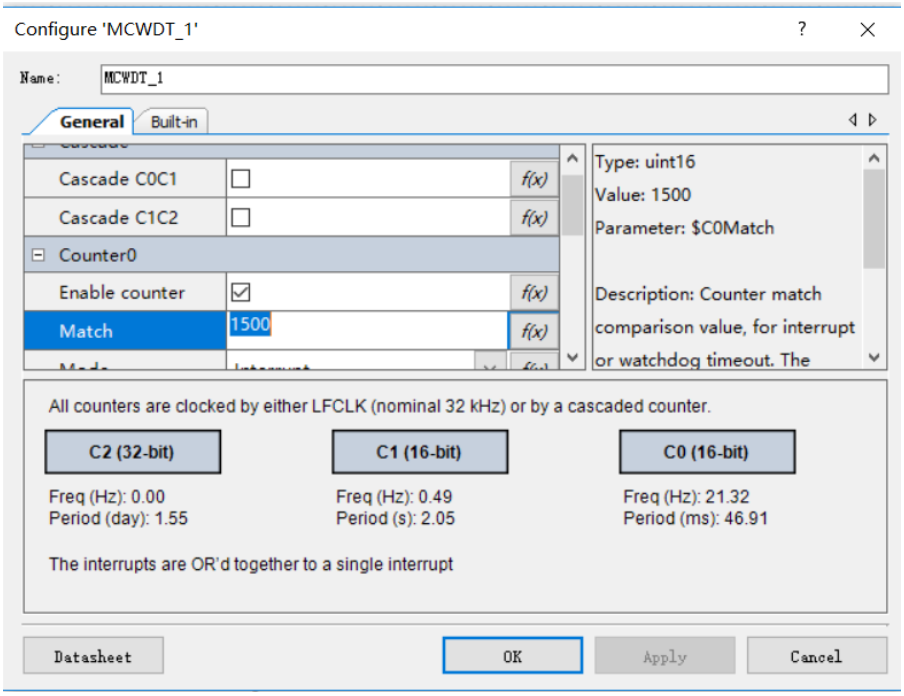
超声测距模块测量最大值 550cm 对应 n=32352，最小值 2cm 对应 n=118，因此计数器的寄存器位数足够（32 位，最大值 65535）

4.2 电机驱动 PWM 输出

电机驱动接收四个 PWM 的输出，每个 PWM 的时钟信号为 1MHz，周期为 100，所以占空比的控制范围即为 0~100.

4.3 看门狗时间中断模块

Jy61 的角度大概以 100Hz 的频率发送，即每间隔 10ms 能得到一次新的角度值，于是我们将看门狗时间中断的中断间隔设置为 47ms，基本可以保证每次解码都能得到新的角度值，且满足了基本要求。

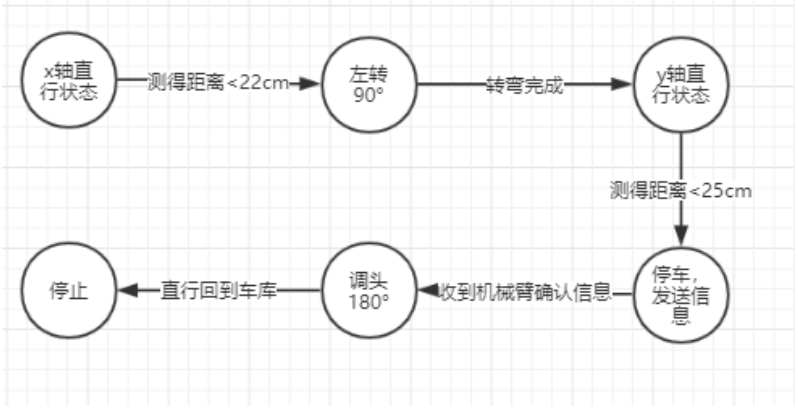


4.3 其他模块

其他模块包括机械臂通信用的串口 UART 模块，陀螺仪通信用的串口 UART 模块等。

5. 流程图

5.1 小车主逻辑流程



首先，状态 0 小车沿着 x 轴方向前进，遇到障碍物时停止，并记录此时与障碍物的距离，进入状态 1。原地左转 90 度，进入状态 2。沿着 y 轴方向前进，遇到障碍物停止，并记录距离，将这两个距离发送给机械臂，进入状态 3。当收到

机械臂确认信息，进入状态 4，开始后退 1s 并调头。当调头 180°完成，进入状态 5，直行回到车库。

小车主逻辑代码如下：

```
void controlMain()
{
    char state=0;
    int distance;
    int base_speed;
    int angle_delta;
    int aimAngle = 0;
    int send_x_dis = 0;

    while(1)
    {
        if(state == 0) //x 轴直行状态
        {
            angle_delta = (z_angle - aimAngle);
            angle_delta = PIDcontrol(angle_delta);
            base_speed = 35;

            setSpeed(base_speed - angle_delta, base_speed + angle_delta);

            distance = measurementDis();
            CyDelay(100);


            if(distance < 22)
            {
                send_x_dis = distance;
                state = 1;
            }

        }
        else if(state == 1)//左转
        {
            if((z_angle >= -105) && (z_angle <= -83))
            {
                aimAngle = -90;
                state = 2;
            }
            else
```

```
            {
                int turnSpeed = (z_angle-(-90))*0.6 + 22;
                turnSpeed = turnSpeed<40 ? turnSpeed : 40;
                turnLeft(turnSpeed);
            }
        }
        else if(state == 2)//y 轴直行状态
        {
            angle_delta = (z_angle - aimAngle);
            if(angle_delta>180)
            {
                angle_delta -= 360;
            }
            angle_delta = PIDcontrol(angle_delta);
            base_speed = 40;

            setSpeed(base_speed - angle_delta, base_speed + angle_delta);

            distance = measurementDis();
            //printf("y distance:%d", distance);
            CyDelay(100);

            //static int distance_count2=0;
            if(distance <= 25)
            {
                //distance_count2++;

                //if(distance_count2==1)
                {
                    //distance_count2 = 0;
                    setSpeed(0, 0);
                    state=3;
                }
            }
            //else{distance_count2=0;}

        }
        else if(state == 3) //停止，发送信息
        {
            //static int distance_count3 = 0;

            distance = measurementDis();
            //distance_count3 ++;

```

```
            //if(distance_count3 == 3)
            {
                //distance_count3 = 0;
                BLUE_TOTH((int)distance*100, (int)send_x_dis*100);

                goBack(30);
                CyDelay(1300);

                state = 4;
            }
        }
        else if(state == 4) //调头
        {
            if(z_angle >75 && z_angle < 105)
            {
                aimAngle = 90;
                state = 5;
             }
             else
             {
                 int turnSpeed2;
                 turnSpeed2 = -(z_angle-90)*0.6+20;
                 turnSpeed2 = turnSpeed2 < 50 ? turnSpeed2:50;
                 turnRight(turnSpeed2);
             }
         }
         else if(state  == 5) //回到车库
         {
             angle_delta = (z_angle - aimAngle);
             if(angle_delta>180)
             {
                 angle_delta -= 360;
             }
             angle_delta = PIDcontrol(angle_delta);
             base_speed = 40;

             setSpeed(base_speed - angle_delta, base_speed + angle_delta);

             distance = measurementDis();
             CyDelay(100);

             //static int distance_count4=0;
             if(distance < 40)
             {
```

```
                //distance_count4++;

                //if(distance_count4==3)
                {
                    //distance_count4 = 0;
                    setSpeed(0, 0);
                    break;
                }
            }
            //else{distance_count4 = 0;}
        }
    }
}
```

5.2 看门狗时间中断

看门狗时间中断大概每 47ms 被触发一次，中断函数中，扫描 jy61 串口的缓冲区，根据包头与校验位判断是否收到角度信息。具体实现如下：

```
void WATCHDOW_interrupt()
{
    uint32_t mask = Cy_MCWDT_GetInterruptStatus(MCWDT_1_HW);

    if(0u != (mask & CY_MCWDT_CTR0))
    {
        Cy_MCWDT_ClearInterrupt(MCWDT_1_HW, CY_MCWDT_CTR0);

        //UART_1_Put('a');
        if(0UL == (CY_SCB_UART_RECEIVE_ACTIVE  & Cy_SCB_UART_GetReceiveStatus (UART_jy61_HW, &UART_jy61_context)))
        {
            //UART_1_PutString("get one\r\n");
            char getDataFlag = 0;
            for(int i=0;i<RING_BUFFER_SIZE-1-10;i++)
            {
                if((uartBuffer[i]==0x55) && (uartBuffer[i+1]==0x53))
                {
                    uint8_t sum=0;
                    for(int j=0;j<10;j++)
                    {
                        sum += uartBuffer[i+j];
```

```
                    }
                    if(sum==uartBuffer[i+10])
                    {
                        z_angle = (uartBuffer[i+6] + ((uartBuffer[i+7])<<8))/32768.0*180.0;

                        if(z_angle>180)
                        {
                            z_angle -= 360;
                        }

                        getDataFlag = 1;
                    }
                }
            }

            if(getDataFlag==1)
            {
                char output[15];
                sprintf(output, "z angle:%d\r\n", z_angle); //向右转时角度增大
                //UART_1_PutArray(output, strlen(output));
            }

            Cy_SCB_UART_Receive(UART_jy61_HW, uartBuffer, RING_BUFFER_SIZE, &UART_jy61_context);
        }
    }

}
```

5.3 PID 控制

为了使小车沿特定的方向前进，我们对陀螺仪测到的角度进行 PID，利用 PID 结果对电机 PWM 的输入进行校正，从而使小车方向稳定。PID 函数如下：

```
double PIDcontrol(double e)
{
    double p_part = PID_kp*(e-e_1);
    double i_part = PID_ki*e;
    double d_part = PID_kd*(e - 2*e_1 + e_2)/time_record*1000.0;

    e_2 = e_1;
    e_1 = e;

    return (p_part + i_part + d_part);
}
```

5.4 舵机控制

舵机基础控制只需要按照通信协议，向舵机串口发送特定的信号即可。

```
//用速度v转到目标角度，write模式，立即动
//全速1024
//aimAngle为0~360
void turnToAimAngleVelocity(unsigned char id, unsigned int aimAngle, unsigned int v)
{
    aimAngle = (unsigned int)(aimAngle*1023.0/300.0);

    unsigned char aimA_L = (unsigned char)aimAngle;
    unsigned char aimA_H = (unsigned char)(aimAngle >> 8);
    unsigned char v_L = (unsigned char)v;
    unsigned char v_H = (unsigned char)(v >> 8);

    char command[11] = {0xFF, 0xFF, 0, 7, COMMAND_WRITE_DATA, ADDR_AIM_ANGLE, 0, 0, 0xFF, 1, 0};

    command[2] = id;
    command[6] = aimA_L;
    command[7] = aimA_H;
    command[8] = v_L;
    command[9] = v_H;
    command[10] = ~ (id + 7 + COMMAND_WRITE_DATA + ADDR_AIM_ANGLE + aimA_L + aimA_H + v_L + v_H);

    UART_1_PutArray(command, 11);
    Angles[id] = aimAngle;
}
```

6. 实验结果

达到了实验内容的要求，车上载着配送物（用海绵代替），小车自动巡航到达机械臂旁，同时利用超声检测距离，向机械臂发送指令，以及小车的 xy 轴坐标位置。机械臂根据小车的位置解算出各个舵机的旋转角，抓取配送物，并将其放到旁边的容器中。

7. 实验中遇到的问题与解决方法

7.1 Echo 引脚的输入问题

在实验过程中我们发现 Echo 引脚输入无法正确检测，结果一直为低电平。于是用示波器单独检查了超声模块上的 Trig 信号与 Echo 信号，发现二者均正常工作。猜测可能是引脚驱动模式的问题，于是将超声模块上的 Echo 信号与 Psoc 上的引脚连接起来，再用示波器观测，发现 Echo 信号会有一个小幅上升，但远

没有达到 Psoc 中高电平的要求，猜想得到验证。所以将 Psoc 原理图中 Echo 信号的驱动模式从 strong drive 改为 resistive pull up，问题解决。

7.2 机械臂运动的问题

在验收前，我们突然发现机械臂不动了，即使烧录之前正确的程序也无法让机械臂动作。我们怀疑是 Psoc 串口的问题，于是换了个 Psoc，问题解决。

7.3 超声波被吸收的问题

我们最初在实验室内测试了超声波，确认没问题后在楼道里进行测试，但是这时超声波测距似乎出了问题，小车在墙壁前不会按程序中要求的那样停下来，但是当我们把手挡在超声模块前，小车就会停下来。由于小车在移动，不能连接调试串口，而且小车蓝牙与机械臂蓝牙锁定了，所以我们让小车通过蓝牙把距离发送给机械臂，机械臂打印出距离。发现小车在墙壁前计算出的距离都很大，可能是没有收到回波。我们分析认为是黑色的墙壁吸收了超声波，于是在墙壁前放了一个纸箱，问题迎刃而解。

7.4 车臂通信的问题

小车需要通过蓝牙向机械臂发送位置信息，机械臂也需要向小车发送确认信息。位置信息加上包头，校验位一共有六字节，我们起初是将缓冲区长度设为六字节，但发现这么做会出现问题：包头可能不在缓冲区第一个字节，这属于丢包现象，于是我们将缓冲区长度设置为 18 字节，同时提高发送频率，在这 18 字节中扫描寻找有效的信息包，解决了问题。

8. 体会、收获与建议

这次实验遇到了很多问题，我和我的队友们独立地逐一解决了这些问题，并

最终完成了任务，我想我们都从这次合作中学到了很多，也得到了很多锻炼。虽然四天的调试过程很辛苦，但非常值得，感谢这个课程的老师与助教给我们提供的平台与机会。

创新实验是在之前所学知识的基础上完成的，将理论知识应用到实际，并自己独立完成一项任务，能带给人莫大的成就感。唯一的遗憾就是开始动手前没有做好完整的规划与分工。