电子工艺实习总结

刘治妤
北京邮电大学

日期：2020 年 9 月 5 日

摘　　要

为期六天的电子工艺不知不觉中结束了，我和我的队友在老师的耐心的教导下，终于完成了电子工艺实习需要完成的任务。在代码调试和参数调整中我们积累了经验，培养了自己的初步的工程设计能力和创新意识以及解决问题的能力。在学习焊接的过程中我们学习了焊接的基本操作和方法和焊接元件的使用。在智能小车的调试过程中，我们学习了 PID 控制原理和对于小车行进时钟的设定。整个工艺实习培养了我的兴趣，更愿意作为一名不断进步追求创新的电子方向从业人员。

关键词: 智能平衡小车，电子工艺实习

1　课程目标

1. 实际学习基本的电子工艺，掌握一般的电子工艺技能，了解电子产品的生产流程。

2. 掌握电子技术安全常识、元器件基础知识，电子技术中的焊接、装配工艺、常用仪表的使用。

3. 培养初步的工程设计能力和创新意识，以及严谨、踏实、科学的工作作风，提高解决实际问题的能力和素质。

2　实习任务要求

1. 熟悉焊接用工具，了解焊接用材料，正确使用电烙铁，熟练掌握手工焊接技术，按照正确的焊接操作手法进行反复练习，知道什么是高质量的焊点，保障焊点的质量，学会手工拆焊技术。

2. 焊接一个发光二极管阵列，通过单片机控制阵列显示图形或字符

3. 安装调测智能平衡小车，基本要求实现平衡（静态平衡或动态平衡），提升要求实现直行、旋转、拐弯等。

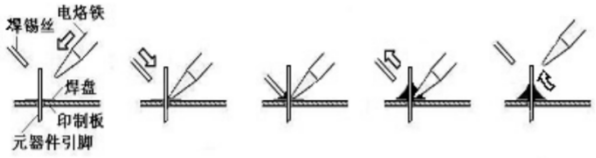

图 2.2.3 手工焊五步操作法图示

图 1: 标准焊接步骤

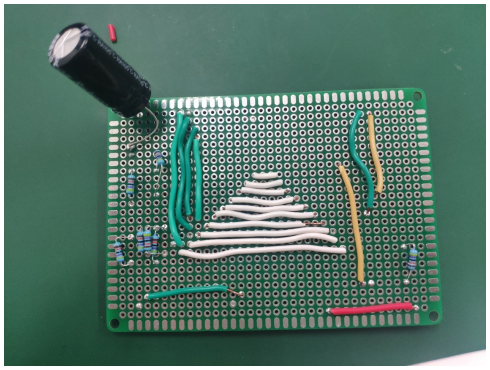

图 2: 进行焊接练习使用的电路板

3 实验题目及实现过程

3.1 焊接

3.1.1 焊接的注意事项

1、电烙铁通电前首先检查电烙铁的电源线有无破损，如有破损及时处理，防止发生短路、短路或触电事故。

2、焊接时，如果烙铁头挂锡太多影响焊接质量，应通过在湿布上擦拭来去处过多的锡。

3、焊接间隙电烙铁应放置在烙铁架上，不得随意放置，一般右手持电烙铁情况下，烙铁架放置在前方靠右处。

4、使用电烙铁进行焊接操作时，还应养成良好的习惯，操作台面上应整齐有序，不放置过多的东西。图 1 为焊接的标准操作图。

3.1.2 实验心得

学习了焊接的基础知识和用法，通过老师的讲解和阅读实习讲义，学习了基础的焊接元件（电烙铁，吸锡器）的使用，并且在板子上进行了练习，学习了基本的焊接手法，在和同伴交流和实践中，提高了自己对于焊接技术的认识和自己的焊接技术水平，为接下来的项目打下了良好的基础。图 2 为我们进行焊接练习时候的焊接效果图。

3.2 发光二极管阵列焊接与调试

3.2.1 操作步骤

1. 根据元器件清单准备元器件
2. 根据元器件从低到高的顺序进行焊接
3. 安装芯片到管座
4. 电路检测
5. 单片机编程、电路调试

3.2.2 代码与设计

1. 点阵扫描原理

点阵扫描的原理上学期我们已经在数字电路与逻辑设计课程中学习过，8X8 点阵共由 64 个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点上，当对应的某一行置 1 电平，某一列置 0 电平，则相应的二极管就亮；如要将第一个点点亮，则 9 脚接高电平 13 脚接低电平，则第一个点就会被点亮。在这里使用的是一样的原理。

```
//点 阵 扫 描
  for ( i = 0 ; i < COL_NUM ; i++ )
  {
//    p = g_ShowData[i ];
    j = g_dot_start + i ;

    if ( j > ( MATRIX_COL - 1 ) ) j -= MATRIX_COL ;

    HAL_SPI_Transmit( &hspi1, g_ShowData + j  , 1, 0 );

    if ( i== 0 ) {
      HAL_GPIO_WritePin( DOT_EN_GPIO_Port  , DOT_EN_Pin , GPIO_PIN_RESET );
      HAL_GPIO_WritePin( DOT_SHIFT_GPIO_Port  , DOT_SHIFT_Pin , GPIO_PIN_RESET
          );
      HAL_GPIO_WritePin( DOT_SHIFT_GPIO_Port  , DOT_SHIFT_Pin , GPIO_PIN_SET );
      HAL_GPIO_WritePin( DOT_EN_GPIO_Port  , DOT_EN_Pin , GPIO_PIN_SET );
    }
    else
    {
      HAL_GPIO_WritePin( DOT_SHIFT_GPIO_Port  , DOT_SHIFT_Pin , GPIO_PIN_RESET
          );
      HAL_GPIO_WritePin( DOT_SHIFT_GPIO_Port  , DOT_SHIFT_Pin , GPIO_PIN_SET );
    }
    HAL_GPIO_WritePin( DOT_LAT_GPIO_Port  , DOT_LAT_Pin , GPIO_PIN_RESET );
    HAL_GPIO_WritePin( DOT_LAT_GPIO_Port  , DOT_LAT_Pin , GPIO_PIN_SET );
    for ( k = 0 ; k < SHOW_DELAY ; k++);
  }
```

2. 显示字符

显示想要产生的字符，则需要将想要生成的字符在 8*8 点阵上的明暗进行控制，我们先将想要显示的字符在 excel 中进行了整理，将每一列的明暗情况的二进制字符串转化为十六进制字符，写入代码即可。图 3 为 excel 中整理的截图和对应的代码。

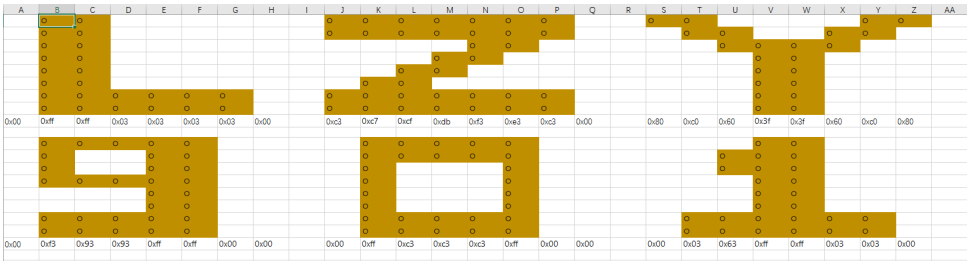

图 3: 用于设计字符的 excel 截图

```
//显示字符 显示姓名字母缩写LZY 学号901
uint8_t g_ShowData[ MATRIX_COL ]  = {
  0x00,0x03,0x03,0x03,0x03,0xff,0xff,0x00,
  0x00,0xc3,0xe3,0xf3,0xdb,0xcf,0xc7,0xc3,
  0x80,0xc0,0x60,0x3f,0x3f,0x60,0xc0,0x80,
  0x00,0x00,0xff,0xff,0x93,0x93,0xf3,0x00,
  0x00,0x00,0xff,0xc3,0xc3,0xc3,0xff,0x00,
  0x00,0x03,0x03,0xff,0xff,0x63,0x03,0x00,
  0};
```

3. 级联显示

进行级联显示时候，可以将队友的 led 板子和自己的板子级联起来，在这个过程中注意连接线的方向。并将代码中的原来的一块板子显示的列数为 8 改成两块板子现实的列数为 16 即可完成显示级联。下图是我们组的级联显示示意图，图中为字母 RX（队友名字的缩写）。

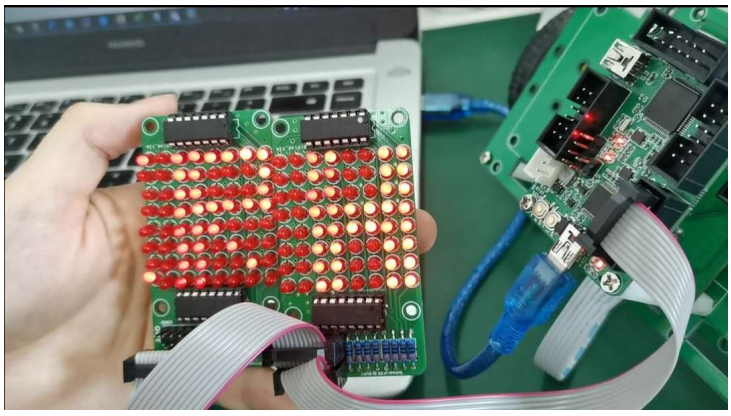

图 4: 级联显示示意图，图中为字母 RX

3.2.3 实验心得

1. 关于焊接，注意焊接的技巧和顺序。要根据据元器件从低到高的顺序进行焊接，好的焊接顺序可以使得焊接过程更轻松。

2. 单片机调试代码的时候，要弄明白例程中的扫描原理，对点阵扫描有比较好的认识，才可以对代码进行修改。在第一次试验里，由于我们弄错了点阵扫描的方向，导致字母显示为镜像，在对字符的顺序进行改动后，可以正常显示字符。级联要求更高的理解，需要读懂代码，弄清楚级联的原理，避免盲目修改代码，程序无法正常运行。

3.3 智能平衡车安装与调试

3.3.1 操作步骤

1. 智能平衡车装配
2. 智能平衡车的平衡调试
3. 实现智能小车的前进，后退，停止，左移，右移
4. 实现小车的循迹

3.3.2 例程代码理解

整个例程采用了前后端分离的模式，前台是一个 while(1) 的死循环，使得 usertask 不断循环执行，后台程序是中断控制，通过 TIM5/TIM6/TIM7 设置的中断频率周期调用，中断 5 可以用于动作执行，通过核心板上的 button1 可以控制调整例程运行的模式。其中模式一对应的是平衡车例程，模式二对应的是点阵控制模式。模式三对应的是串口示波器模式，可以用于调试串口和更好的了解串口情况。

```
//后端模式控制部分
  switch ( g_SysMode )
    {
      case 0 :   // balance control
        System_Init();
        IR_Sensor_Init();
        HAL_TIM_Base_Start_IT(&htim5); //time for applicationn
        HAL_TIM_Base_Start_IT(&htim6); //read sensor data and  control moto
        HAL_TIM_Base_Stop_IT(&htim7);  //read sensor data and  do not control
            moto
        break;
      case 1 :   // dot matrix
        System_Init();
        dot_matrix_init();
        HAL_TIM_Base_Start_IT(&htim5);
        HAL_TIM_Base_Stop_IT(&htim6);
        HAL_TIM_Base_Stop_IT(&htim7);
        break ;
      case 2 :   // show osc wave
        System_Init();
        IR_Sensor_Init();
        HAL_TIM_Base_Start_IT(&htim5);
        HAL_TIM_Base_Start_IT(&htim6);
```

```
        HAL_TIM_Base_Stop_IT(&htim7);
        break ;
    }
```

定时器 6 将时间分为了 5 个间隔，每一个间隔可以执行不同的工作，分别用于 DirectionControl(方向控制),MPU6050Dataprocess(电机控制),GetIRSensor(获得传感器数据)... 等工作，使得小车可以在模式一下可以完成循迹和通过角度和方向来控制电机从而使小车保持平衡状态。

```
\\TIM6时间分割进行各个任务的处理
    if ( period_5ms > 4 ) {
      period_5ms =0;
      period_100ms++;
      if ( period_100ms > 19 ) {
        period_100ms = 0 ;
      }
    }
    if ( period_5ms == 0 )
    {
      if ( period_100ms == 0 )
      {
        CalculateSpeed();
        SpeedControl();
      }
      SpeedControlOutput( 5 );
      DirectionControlOutput( 5 );
      MotoOutput();
      if ( g_SysMode == 0)
      {
        MotoSpeedOut();
      }
      LoopLED();
      return ;
    }
    else if ( period_5ms == 1 )
    {
      MPU6050_Get_Accel_Gyro_Temp();
      return ;
    }
    else if ( period_5ms == 2 )
    {
      MPU6050_Data_Process();
      AngleControl();
      return ;
    }
    else if ( period_5ms == 3 )
    {
      if ( (period_100ms % 2) == 1 )
```

```
      {
        Get_IR_Sensor();
        DirectionControl();
        Power_IR_Sensor(0);
      }
      else
      {
        Power_IR_Sensor(1);
      }
      return ;
    }

  }
```

小车平衡主要使用了 PID 控制算法，对方向和角度进行微分和积分控制。在程序中体现在作为 angleControl(角度控制),SpeedControl(速度控制) 的参数，只有合适的参数才可以确保我们的小车趋于平稳，所以就需要我们不断控制小车进行测试找到合适的参数。

```
//angle control
  g_moto_ctrl.left_ctrl = g_moto_ctrl.angle_ctrl - g_moto_ctrl.speed - g_moto_
      ctrl.direction;
  g_moto_ctrl.right_ctrl = g_moto_ctrl.angle_ctrl - g_moto_ctrl.speed + g_moto_
      ctrl.direction;
```

```
//speed control
  g_moto_ctrl.moto_pulse = ( g_moto_ctrl.left_moto_pulse + g_moto_ctrl.right_
      moto_pulse ) / 2 ;
  g_moto_ctrl.speed =  g_moto_ctrl.moto_pulse * CAR_SPEED_CONSTANT;
  diff =  g_moto_ctrl.speed - g_moto_ctrl.speed_set ;
  g_moto_ctrl.positon += ( diff * SPEED_CONTROL_I ) ;

  g_moto_ctrl.speed_ctrl_last = g_moto_ctrl.speed_ctrl_next;
  g_moto_ctrl.speed_ctrl_next = diff * SPEED_CONTROL_P + g_moto_ctrl.positon  ;
  g_moto_ctrl.speed_diff = g_moto_ctrl.speed_ctrl_next - g_moto_ctrl.speed_ctrl
      _last;
```

循迹控制是通过小车底部的循迹板，循迹板上有六个红外发射管，检测到黑线的时候，发送数据 1，对应的指示灯灭，否则发射 0，对应的指示灯量。对于不同的循迹情况比如当偏左侧（001111）检测不到黑线的时候，就会控制 direction 进行方向的调整，从而使得小车向右侧移动。在代码中对于不同的情况，不同的偏转方向，不同的偏转程度，使用了不同的系数，从而控制小车可以偏离合适的角度。

```
  g_moto_ctrl.direction_ctrl_last = g_moto_ctrl.direction_ctrl_next;

  if ( g_moto_ctrl.track_in == 0x08 ||  // 110111
       g_moto_ctrl.track_in == 0x0C ||  // 110011
       g_moto_ctrl.track_in == 0x04 )   // 111011
```

```
{
  g_moto_ctrl.direction_ctrl_next = 0 ;
}
else if ( g_moto_ctrl.track_in == 0x10 ||  // 101111
          g_moto_ctrl.track_in == 0x18 )   // 100111
{
  g_moto_ctrl.direction_ctrl_next = 10 * DIR_CONTROL_P + g_mpu6050.gyro_scale
      _z * DIR_CONTROL_D;
}
else if ( g_moto_ctrl.track_in == 0x20 ||    // 011111
          g_moto_ctrl.track_in == 0x30 ||    // 001111
          g_moto_ctrl.track_in == 0x38 )   // 000111
{
  g_moto_ctrl.direction_ctrl_next = 20 * DIR_CONTROL_P + g_mpu6050.gyro_scale
      _z * DIR_CONTROL_D;
}
else if ( g_moto_ctrl.track_in == 0x02 ||    // 111101
          g_moto_ctrl.track_in == 0x06 )     // 111001
{
  g_moto_ctrl.direction_ctrl_next = -10 * DIR_CONTROL_P + g_mpu6050.gyro_
      scale_z * DIR_CONTROL_D;
}
else if ( g_moto_ctrl.track_in == 0x01 ||    // 111110
          g_moto_ctrl.track_in == 0x03 ||    // 111100
          g_moto_ctrl.track_in == 0x07 )     // 111000
{
  g_moto_ctrl.direction_ctrl_next = -20 * DIR_CONTROL_P + g_mpu6050.gyro_
      scale_z * DIR_CONTROL_D;
}
else if ( g_moto_ctrl.track_in == 0x3F )
{
  g_moto_ctrl.speed_set = 0 ;
  g_moto_ctrl.direction_ctrl_next = 0 ;
}
else
{
  g_moto_ctrl.direction_ctrl_next = 0 ;
}
g_moto_ctrl.direction_diff = g_moto_ctrl.direction_ctrl_next - g_moto_ctrl.
    direction_ctrl_last;
```

中断 5 在所给的例程中除了让小车保持静止和平衡外，没有进行其他的操作。我们可以在其中进行我们的自主控制小车完成指定动作程序的编写。

3.3.3 代码与设计

1. 完成小车的组装

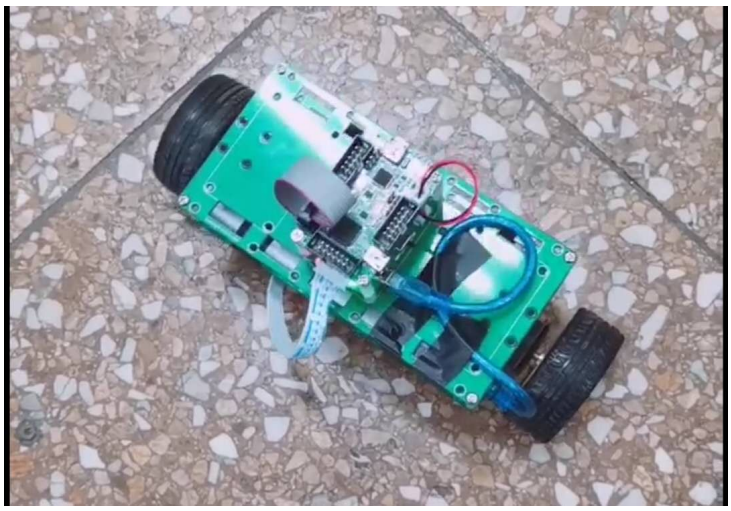

图 5: 小车安装完成示意图

按照讲义内容，我们完成了对应的焊接和组装工作。图 5 为我们小车安装完成的示意图。

2. 调整 pid，控制小车保持平衡

对于一个智能小车，我们最先的工作就是让小车先保持平衡。要想让小车保持平衡，我们就要使得他的角度和方向都可以保持平衡。通过上面我们对于代码的分析，我们可以很清楚的认识到，希望改变我们小车的平衡状态需要分别对我们的 SPEEDCONTROLP,SPEEDCONTROLD，控制小车的速度,ANGLECONTROLP,ANGLECONTROLD 控制小车的角度。我们的工作就是调整这四个参数。

调试过程如下 (以 angle control 为例)

首先控制 d=0, 调整 p 的值，使得小车可以稳定起来，p 过大的时候会导致小车震荡，快速的来回摆动，出现超调。

然后控制 p 的值，使小车趋于平稳，d 加大后，就会来回抖动，我们通过控制参数调整合适的值，使得小车静止下来。

以下为我们的小车最终调整的 pd 值

```
ANGLE_CONTROL_P = 0.14
ANGLE_CONTROL_D = 0.01
SPEED_CONTROL_P = 0.1
SPEED_CONTROL_I = 0.0005
```

图 6 是我和我的队友调试的现场图

3. 编写程序代码，控制小车完成指定动作在小车可以保证基础的平衡之后，通过上面我们对于代码的分析，我们可以直接把我们想要控制的代码放到定时器 5 中进行执行，也感谢袁老师的代码对我们的提示。

首先程序是个 43s 的循环，对于直行和停止，只需要对他们的速度的值进行设计即可。前进时候设置为 5, 后退时候设置为-5，停止时候设置为 0 即可。

对于实现小车的左转右转，我们需要控制小车两边电机速度不同的转速，当一个轮子转速快，另一个轮子转速慢的时候，小车就会进行转弯。

```
//整个过程为43s的循环
```

图 6: 现场调试参数图

```
if ( period_30s < 429 ) period_30s++;
  else period_30s = 0 ;
  if ( period_30s < 30) {    //前3s保持静止
    g_moto_ctrl.speed_set = 0;
  }
  else if( period_30s < 80) //3s-8s以speed=5前进
    g_moto_ctrl.speed_set = 5 ;
  else if( period_30s < 110)//8s-11s停止
    g_moto_ctrl.speed_set = 0 ;
  else if( period_30s < 160)//11s-16s后退
    g_moto_ctrl.speed_set = -5 ;
  else if( period_30s < 190)//16s-19s停止
    g_moto_ctrl.speed_set = 0 ;
  else if( period_30s < 240 ){//19s-24s左转
      //控制电机两边转速不同，从而控制小车转弯
    g_moto_ctrl.speed_set = 3 ;
    g_moto_ctrl.left_ctrl=g_moto_ctrl.left_ctrl*0.5;
    g_moto_ctrl.right_ctrl=g_moto_ctrl.right_ctrl*1;
  }
  else if( period_30s < 270) //24s-27s停止
    g_moto_ctrl.speed_set = 0 ;
  else if( period_30s < 320) //27s-32s前进
    g_moto_ctrl.speed_set = 5 ;
  else if( period_30s < 350)//32s-35s停止
    g_moto_ctrl.speed_set = 0 ;
  else if( period_30s < 400 ){ //35s-40s右转
    g_moto_ctrl.speed_set = 3 ;
    g_moto_ctrl.left_ctrl=g_moto_ctrl.left_ctrl*1;
    g_moto_ctrl.right_ctrl=g_moto_ctrl.right_ctrl*0.5;

  }
  else{ //40s-43s控制小车静止
    g_moto_ctrl.speed_set = 0 ;
  }
```

4. 进行小车循迹的调试我们通过阅读循迹部分的代码，不难看出，对于循迹的控制和 pd 有

图 7: 循迹测试图

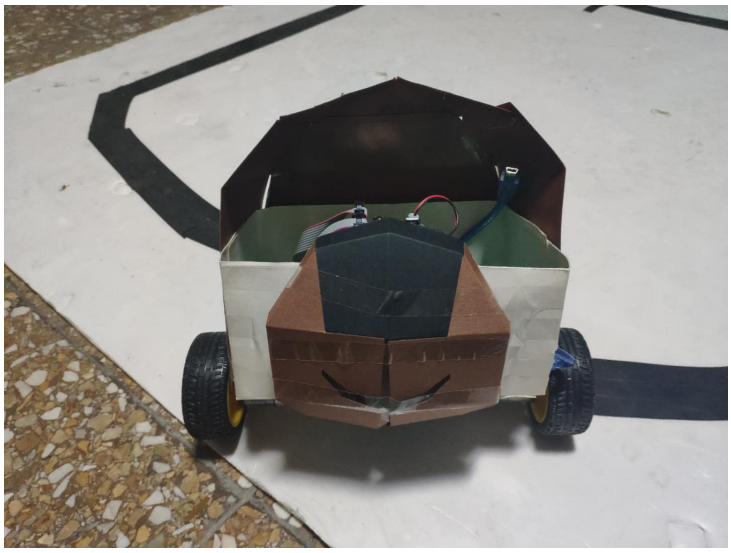

图 8: 小车外观展示图

关，还有就是灵敏度的调试。我们首先通过调整电位器调整元件的灵敏度，确保元件可以很灵敏的感受到黑线，然后进行不断的测试，不断更改和确定循迹的 pd 对应的参数值。以下为我们最终确定的循迹 pd 参数值。

```
DIR_CONTROL_P       0.1
DIR_CONTROL_D       0.01
```

图 7 为我们的小车正在进行循迹测试

5. 外观设计外观我们使用的是淘宝上购买的纸模进行改造对我们的小车外观进行了装饰，对我们的小车进行了美化。图 8 图 9 为我们的小车的外观设计

3.3.4 实验心得

这是我和队友第一次参与 stm32 代码的编写，也是我本人第一次做一个比较大的项目。在这短短几天时间里，我学会了单片机开发的一般流程，在老师的讲解下，学习了 stmcubeMX 和 keli 编程软件的使用。并且学习了平衡小车的基本工作原理。

图 9: 小车外观展示图

4 本人所担任的工作

1. 在点阵扫描实验中，我仔细阅读了代码，研究了如何控制点阵显示字母图案，给出了代码修改的方案，和队友合作将 led 灯的明暗转为了十六进制数，使得字母可以正常显示。

2. 在智能车的拼接中，我担任了小组元件的大部分焊接工作，并合作队友完成了小车的组装。

3. 担任了自定义动作程序部分的代码编写主要工作，和小车运动状态的姿态矫正。

4. 协助队友进行参数调试。

5 总结和反思

1. 在小车组装的焊接问题上，我们出现了一些问题，首先是霍尔元件器件焊接方向反向，然后是霍尔元件管脚有点偏长，在这上面我们浪费了宝贵的一天半的问题，差点导致我们的工作无法顺利继续下去。从这次实习的教训中，我认识到工业调试是一个很科学严谨的工作，容不得半点的差错。作为电子方向从业人员，我们每天接触的电子元器件都很小，但是只有保持科学严谨，不放过每一个细节，哪怕每一步都很慢，但是只有认真了才是不浪费时间，慢工才能出细活。

2. 参数调整。参数调整是一个漫长的工作，这时候需要一个好的心态。由于我和我的队友之前的失误导致后来我们的进度慢于周围的同学，但是当我们沉下心，理性分析，参数的控制其实也不是那么的困难。在我们以后的实验中，也会有很多需要对参数进行控制的地方，我们需要一个沉着理性的心态，慢慢调试，仔细观察现象，分析参数对于现象的影响，就可以找得到参数调整的方向。在参数调试上，可以选择微调法，也可以选择二分法，极限法，提高参数调试的效率。

3. 循迹测试。小车循迹对于环境的光线有很高的要求。由于我们一开始没有注意到这个重要的因素，使得我们浪费了很多时间在对于 pd 的调试上面，浪费了时间，后来在同学的提醒下，改变了环境，pd 的调试变得更有效率了，加快了我们的实验错误。在进行测试的时候，及时的交流是胜利的法宝，团队拥有更多的智慧。

4. 感谢老师在我们遇到问题的时候给我们的建议和讲解，帮助我们顺利完成了实习任务。

6 参考资料

《2020 电子工艺实习讲义》