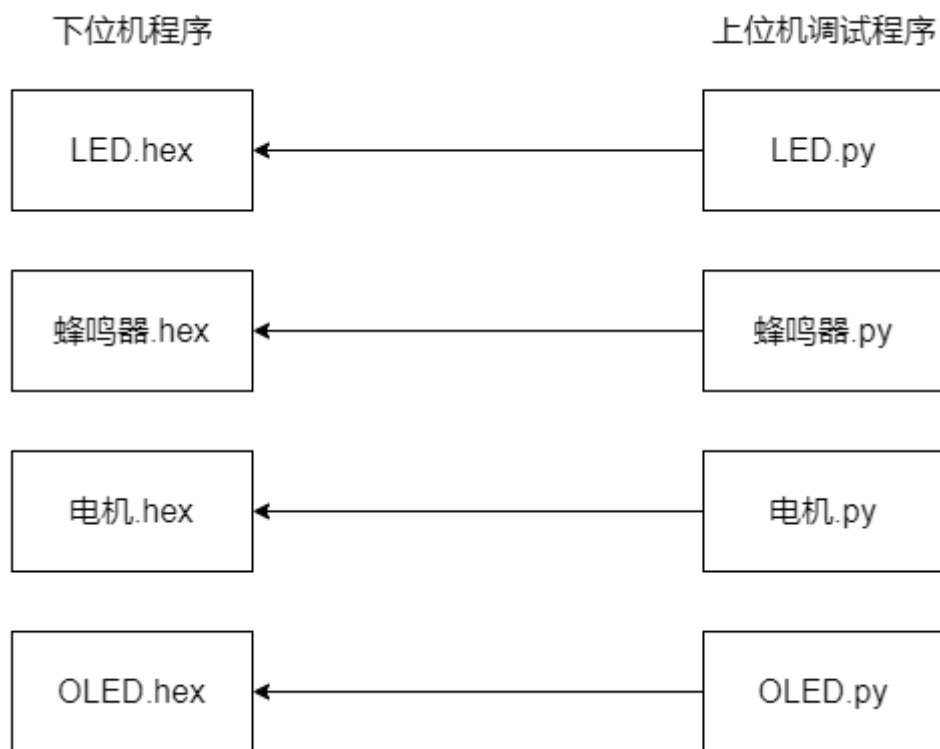
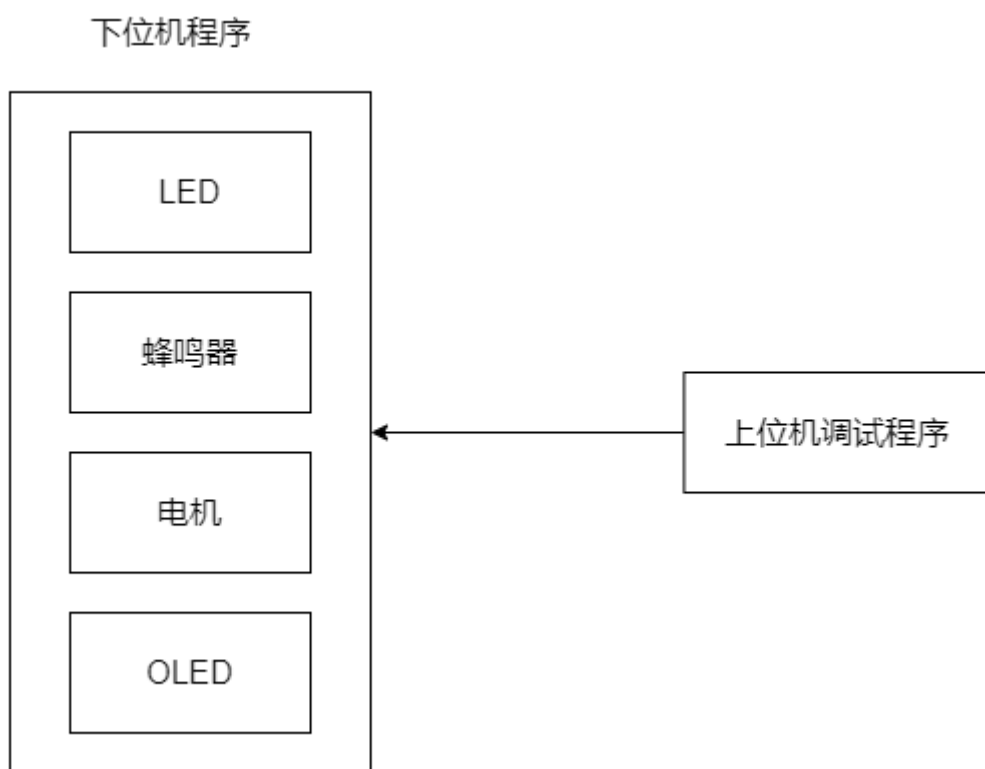


## 问题

目前需要通过烧录多个程序，通过多套代码来驱动控制板。



我们更希望的是下位机一个文件就可以，然后调试代码也是一套就行



# 思考

单一功能的下位机程序，只能接收有限的指令，不能通过指令来区分到底是让谁干活，或者说只能让固定的部件干活。

要解决这个问题，只需要上位机和下位机沟通好，到底怎么干活就行。沟通的话术就是我们俗称的协议。

大家按照统一的协议规定，去干具体的活就行。

## 协议制定分析

功能	原始协议	现有协议
LED	0x01(开),0x02(关),0x03(切换)	0x01 + 0x01(开),0x02(关),0x03(切换)
蜂鸣器	0x01(开),0x02(关),0x03(切换)	0x02 + 0x01(开),0x02(关),0x03(切换)
电机	0xb8 0x0b(pwm值)	0x03 + 0xb8 0x0b(pwm值)
OLED	0x69 0x74 0x63 0x61 0x73 0x74(显示的内容)	0x04 + 0x69 0x74 0x63 0x61 0x73 0x74(显示的内容)

!!!note  
通过一个唯一的标记值取标定指令，后续根据实际操作来控制

## 控制代码实现

### 基本代码实现

```
1  import serial
2
3
4  def test_led():
5      # 字节数据
6      data = bytearray([0x01, 0x02])
7      ser.write(data)
8
9
10 def test_buzzer():
11     # 字节数据
12     data = bytearray([0x02, 0x02])
13     ser.write(data)
14
15
16 def test_motor():
17     # 字节数据 pwm 3000
18     data = bytearray([0x03, 0xb8, 0x0b])
19     ser.write(data)
20
21
22 def test_oled():
23     # 字节数据 itcast
```

```
24     data = bytearray([0x04, 0x69, 0x74, 0x63, 0x61, 0x73, 0x74])
25     ser.write(data)
26
27
28 if __name__ == '__main__':
29     ser = serial.Serial(port='/dev/ttyUSB0', baudrate=115200)
30
31     test_led()
32     test_buzzer()
33     test_motor()
34     test_oled()
```

## 异常

程序在运行过程中，发生了未知的事件，影响到了程序的正常运行。

异常是一种事件 异常会影响到程序正常运行。

格式如下：

```
1  try:
2      代码
3  except:
4      出现异常后的逻辑
```

```
1  a = 10
2  b = 0
3  c = a/b
4  print(c)
```

!!!warning

执行以上代码就会出现错误。

```
1  出现了ZeroDivisionError异常
2
3  出现异常会造成程序停止
```

## GUI实现

---



## 布局拆分



整个窗体可以分为一个水平布局，包含两个部分。



第一个部分，又可以分为一个垂直布局，包含连个部分



第二个部分，又可以分为一个垂直布局，包含连个部分

按照这种方式依次类推，就可以更细化的布局。

布局的原则就是由外向内。

## 分组控件

Qt中 `QGroupBox` 是一个分组控件，可以帮助我们将一个布局包括在一起。



`QGroupBox` 包含了title和内部的布局两个部分.

```
1 group = QGroupBox('我是title')
2 layout = QVBoxLayout(group)
```

## 完整代码实现

```
1 from PyQt5.Qtwidgets import *
2 from PyQt5.QtGui import *
3 from PyQt5.QtCore import *
4 import sys
5
6 import serial
7 import struct
8
9
10 class Mywindow(Qwidget):
11
12     def __init__(self):
13         super(Mywindow, self).__init__()
14
15         self.__init_ui()
16
17         count = 0
18         while count < 10:
19             count += 1
20             try:
21                 self.ser = serial.Serial(port='/dev/ttyUSB0',
baudrate=115200)
22                 # 如果出错了, 后面的代码就不执行了
23                 # 能到达这个位置说明, 链接成功
24                 break
25             except Exception as e:
26                 print(e)
27
28     def __init_ui(self):
29         self.setWindowTitle('下位机控制')
30
31         layout = QHBoxLayout()
32         self.setLayout(layout)
33
34         first_layout = QVBoxLayout()
35         second_layout = QVBoxLayout()
36
```

```
37         layout.addLayout(first_layout)
38         layout.addLayout(second_layout)
39
40         self.__init_first(first_layout)
41         self.__init_second(second_layout)
42
43     def __init_first(self, layout):
44         led_group = QGroupBox('LED控制')
45         layout.addWidget(led_group)
46         led_layout = QVBoxLayout(led_group)
47         self.__init_led_layout(led_layout)
48
49         buzzer_group = QGroupBox('蜂鸣器控制')
50         layout.addWidget(buzzer_group)
51         buzzer_layout = QVBoxLayout(buzzer_group)
52         self.__init_buzzer_layout(buzzer_layout)
53
54     def __init_led_layout(self, layout):
55         btn_open = QPushButton('打开LED')
56         btn_close = QPushButton('关闭LED')
57         btn_toggle = QPushButton('开关LED')
58
59         layout.addWidget(btn_open)
60         layout.addWidget(btn_close)
61         layout.addWidget(btn_toggle)
62
63         btn_open.clicked.connect(self.led_open)
64         btn_close.clicked.connect(self.led_close)
65         btn_toggle.clicked.connect(self.led_toggle)
66
67     def __init_buzzer_layout(self, layout):
68         btn_open = QPushButton('打开蜂鸣器')
69         btn_close = QPushButton('关闭蜂鸣器')
70         btn_toggle = QPushButton('开关蜂鸣器')
71
72         layout.addWidget(btn_open)
73         layout.addWidget(btn_close)
74         layout.addWidget(btn_toggle)
75
76         btn_open.clicked.connect(self.buzzer_open)
77         btn_close.clicked.connect(self.buzzer_close)
78         btn_toggle.clicked.connect(self.buzzer_toggle)
79
80     def __init_second(self, layout):
81         motor_group = QGroupBox('电机控制')
82         layout.addWidget(motor_group)
83         motor_layout = QVBoxLayout(motor_group)
84         self.__init_motor_layout(motor_layout)
85
86         oled_group = QGroupBox('OLED控制')
87         layout.addWidget(oled_group)
88         oled_layout = QVBoxLayout(oled_group)
89         self.__init_oled_layout(oled_layout)
90
91     def __init_motor_layout(self, layout):
92         self.__le_motor = QLineEdit()
93         btn = QPushButton("发送")
94
```

```
95         layout.addWidget(self.__le_motor)
96         layout.addWidget(btn)
97
98         btn.clicked.connect(self.motor_spin)
99
100     def __init_oled_layout(self, layout):
101         self.__le_oled = QLineEdit()
102         btn = QPushButton("发送")
103
104         layout.addWidget(self.__le_oled)
105         layout.addWidget(btn)
106
107         btn.clicked.connect(self.oled_show)
108
109     def led_open(self):
110         # 字节数据
111         data = bytearray([0x01, 0x01])
112         self.ser.write(data)
113
114     def led_close(self):
115         # 字节数据
116         data = bytearray([0x01, 0x02])
117         self.ser.write(data)
118
119     def led_toggle(self):
120         # 字节数据
121         data = bytearray([0x01, 0x03])
122         self.ser.write(data)
123
124     def buzzer_open(self):
125         # 字节数据
126         data = bytearray([0x02, 0x01])
127         self.ser.write(data)
128
129     def buzzer_close(self):
130         # 字节数据
131         data = bytearray([0x02, 0x02])
132         self.ser.write(data)
133
134     def buzzer_toggle(self):
135         # 字节数据
136         data = bytearray([0x02, 0x03])
137         self.ser.write(data)
138
139     def motor_spin(self):
140         text = self.__le_motor.text()
141         pwm = int(text)
142         pack = struct.pack('h', pwm)
143         data = bytearray([0x03, pack[0], pack[1]])
144         self.ser.write(data)
145
146     def oled_show(self):
147         text = self.__le_oled.text()
148         data = bytearray([0x04])
149         data.extend(text.encode())
150         self.ser.write(data)
151
152
```



```
153 if __name__ == '__main__':  
154     app = QApplication(sys.argv)  
155  
156     window = Mywindow()  
157     window.show()  
158  
159     sys.exit(app.exec_())
```