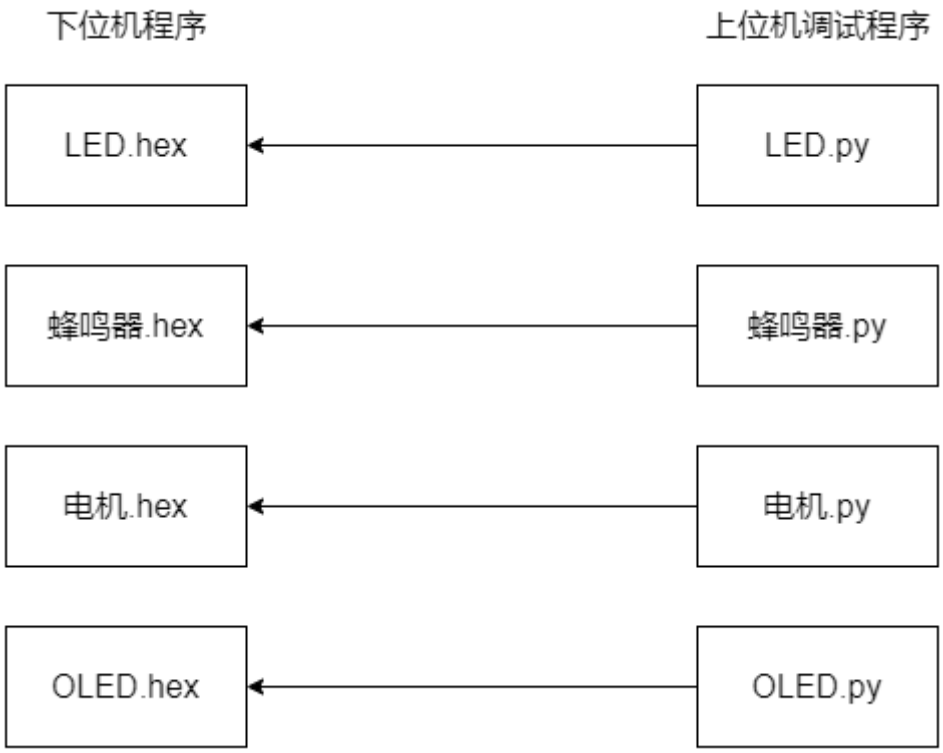
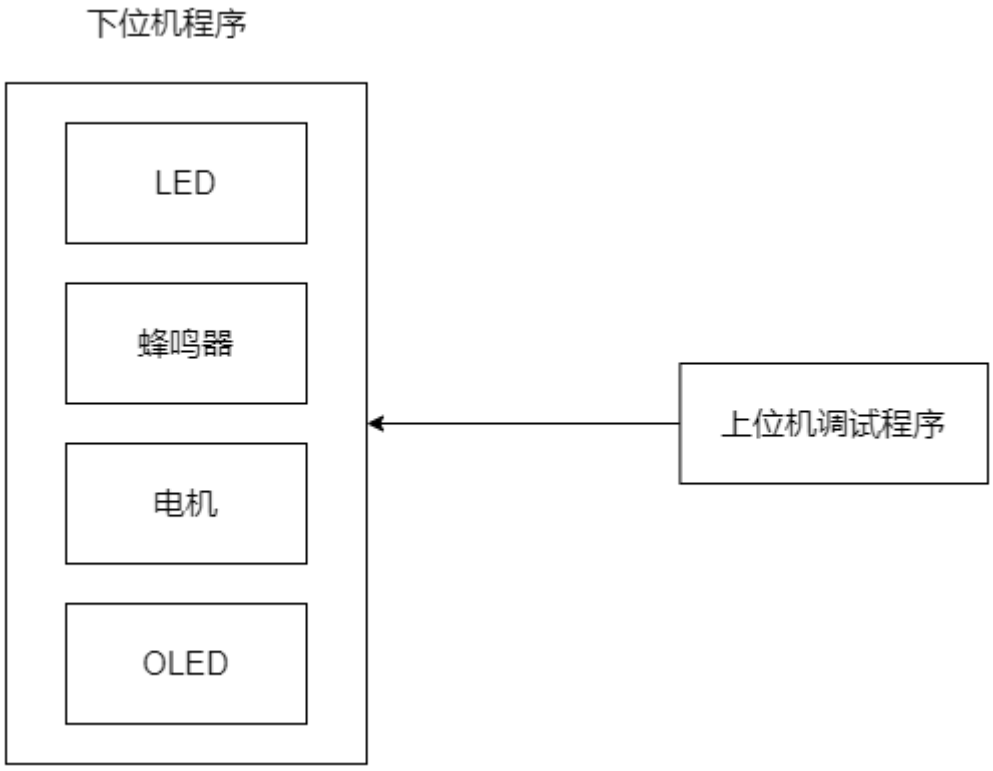
问题

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

```
import serial


def test_led():
    # 字节数据
    data = bytearray([0x01, 0x02])
    ser.write(data)


def test_buzzer():
    # 字节数据
    data = bytearray([0x02, 0x02])
    ser.write(data)


def test_motor():
    # 字节数据 pwm 3000
    data = bytearray([0x03, 0xb8, 0x0b])
    ser.write(data)


def test_oled():
    # 字节数据 itcast
```

```
    data = bytearray([0x04, 0x69, 0x74, 0x63, 0x61, 0x73, 0x74])
    ser.write(data)


if __name__ == '__main__':
    ser = serial.Serial(port='/dev/ttyUSB0', baudrate=115200)

    test_led()
    test_buzzer()
    test_motor()
    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:

```
try:
    代码
except:
    出现异常后的逻辑
```

```
a = 10
b = 0
c = a/b
print(c)
```

!!!warning
    执行以上代码就会出现错误。

```
出现了ZeroDivisionError异常

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

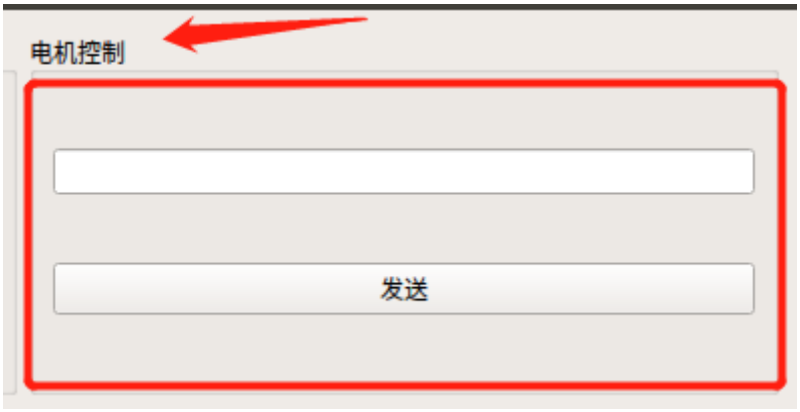
第二个部分，又可以分为一个垂直布局，包含连个部分

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.

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

完整代码实现

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
from PyQt5.QtCore import *
import sys

import serial
import struct


class MyWindow(QWidget):

    def __init__(self):
        super(MyWindow, self).__init__()

        self.__init_ui()

        count = 0
        while count < 10:
            count += 1
            try:
                self.ser = serial.Serial(port='/dev/ttyUSB0',
baudrate=115200)
                # 如果出错了，后面的代码就不执行了
                # 能到达这个位置说明，链接成功
                break
            except Exception as e:
                print(e)

    def __init_ui(self):
        self.setWindowTitle('下位机控制')

        layout = QHBoxLayout()
        self.setLayout(layout)

        first_layout = QVBoxLayout()
        second_layout = QVBoxLayout()

```

```
        layout.addLayout(first_layout)
        layout.addLayout(second_layout)

        self.__init_first(first_layout)
        self.__init_second(second_layout)

    def __init_first(self, layout):
        led_group = QGroupBox('LED控制')
        layout.addWidget(led_group)
        led_layout = QVBoxLayout(led_group)
        self.__init_led_layout(led_layout)

        buzzer_group = QGroupBox('蜂鸣器控制')
        layout.addWidget(buzzer_group)
        buzzer_layout = QVBoxLayout(buzzer_group)
        self.__init_buzzer_layout(buzzer_layout)

    def __init_led_layout(self, layout):
        btn_open = QPushButton('打开LED')
        btn_close = QPushButton('关闭LED')
        btn_toggle = QPushButton('开关LED')

        layout.addWidget(btn_open)
        layout.addWidget(btn_close)
        layout.addWidget(btn_toggle)

        btn_open.clicked.connect(self.led_open)
        btn_close.clicked.connect(self.led_close)
        btn_toggle.clicked.connect(self.led_toggle)

    def __init_buzzer_layout(self, layout):
        btn_open = QPushButton('打开蜂鸣器')
        btn_close = QPushButton('关闭蜂鸣器')
        btn_toggle = QPushButton('开关蜂鸣器')

        layout.addWidget(btn_open)
        layout.addWidget(btn_close)
        layout.addWidget(btn_toggle)

        btn_open.clicked.connect(self.buzzer_open)
        btn_close.clicked.connect(self.buzzer_close)
        btn_toggle.clicked.connect(self.buzzer_toggle)

    def __init_second(self, layout):
        motor_group = QGroupBox('电机控制')
        layout.addWidget(motor_group)
        motor_layout = QVBoxLayout(motor_group)
        self.__init_motor_layout(motor_layout)

        oled_group = QGroupBox('OLED控制')
        layout.addWidget(oled_group)
        oled_layout = QVBoxLayout(oled_group)
        self.__init_oled_layout(oled_layout)

    def __init_motor_layout(self, layout):
        self.__le_motor = QLineEdit()
        btn = QPushButton("发送")

```

```
        layout.addWidget(self.__le_motor)
        layout.addWidget(btn)

        btn.clicked.connect(self.motor_spin)

    def __init_oled_layout(self, layout):
        self.__le_oled = QLineEdit()
        btn = QPushButton("发送")

        layout.addWidget(self.__le_oled)
        layout.addWidget(btn)

        btn.clicked.connect(self.oled_show)

    def led_open(self):
        # 字节数据
        data = bytearray([0x01, 0x01])
        self.ser.write(data)

    def led_close(self):
        # 字节数据
        data = bytearray([0x01, 0x02])
        self.ser.write(data)

    def led_toggle(self):
        # 字节数据
        data = bytearray([0x01, 0x03])
        self.ser.write(data)

    def buzzer_open(self):
        # 字节数据
        data = bytearray([0x02, 0x01])
        self.ser.write(data)

    def buzzer_close(self):
        # 字节数据
        data = bytearray([0x02, 0x02])
        self.ser.write(data)

    def buzzer_toggle(self):
        # 字节数据
        data = bytearray([0x02, 0x03])
        self.ser.write(data)

    def motor_spin(self):
        text = self.__le_motor.text()
        pwm = int(text)
        pack = struct.pack('h', pwm)
        data = bytearray([0x03, pack[0], pack[1]])
        self.ser.write(data)

    def oled_show(self):
        text = self.__le_oled.text()
        data = bytearray([0x04])
        data.extend(text.encode())
        self.ser.write(data)


```

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