第 5 章 Arduino 和 Processing 互动

Arduino 和 Processing 之间可以通过串行接口（简称串口）进行通信。本章首先分别说明 Processing 如何控制 Arduino 和 Arduino 如何控制 Processing，然后详细介绍一个两者连接互动的作品——交互艺术装置 Colorful Melody 的实现，最后介绍导电墨水。

5.1 Processing 控制 Arduino

案例 5-1 展示如何使用 Processing 控制 Arduino 开发板上的小灯亮灭。首先在 Processing 端通过 import processing.serial.* 语句导入串口库，然后声明一个 Serial 变量 port。在 Arduino 端初始化程序中，通过 Serial.begin(9600) 语句设置波特率为 9600b/s，插入 Arduino 开发板后计算机分配的端口是 COM3，Processing 端在初始化程序中通过 new Serial(this,"COM3",9600) 语句，就可以在 COM3 串口上以波特率 9600 b/s 发送和接收数据。

接下来实现发送和接收数据。在 Processing 端创建一个 300 × 300 像素的画布，在画布上绘制一个矩形，当鼠标单击矩形内区域，则向串口发送字符 a，同时打印 LED turn ON 方便调试，这样如果出现异常，可以判断是 Processing 程序出了问题没有运行到这里，还是串口发送的问题。当鼠标单击矩形外的画布区域，则向串口发送字符 b。Arduino 端一直从串口读数据，当判断读到的字符为 a（ASCII 码为 97）时，点亮 Arduino 开发板上的 LED 灯；字符为 b（ASCII 码为 98）时，熄灭

Arduino 开发板上的 LED 灯。

【案例 5-1】 Processing 控制 Arduino 板上的 LED 灯

Processing 端的代码：

```
import processing.serial.*;
Serial port;
void setup(){
  port = new Serial(this,"COM3",9600);
  size(300,300);
}
void draw(){
  rect(100,100,50,50);
}
void mouseClicked(){
  if((mouseX>=100) & (mouseX<=150) & (mouseY>=100) & (mouseY<=150)){
    println("LED turn ON");
    port.write("a");
  }
  else{
    println("LED turn OFF");
    port.write("b");
  }
}
```

Arduino 端的代码：

```
int c=0;
void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  if(Serial.available()){
    c=Serial.read();
    if(c==97)
      digitalWrite(LED_BUILTIN, HIGH);
    else if(c==98)
```

```
        digitalWrite(LED_BUILTIN, LOW);
    }
}
```

同时运行 Arduino 和 Processing 程序，这时 Processing 弹出一个画布窗口，如图 5-1 所示，单击矩形框内区域和矩形外画布区域，控制 Arduino 开发板上 LED 灯的亮灭。

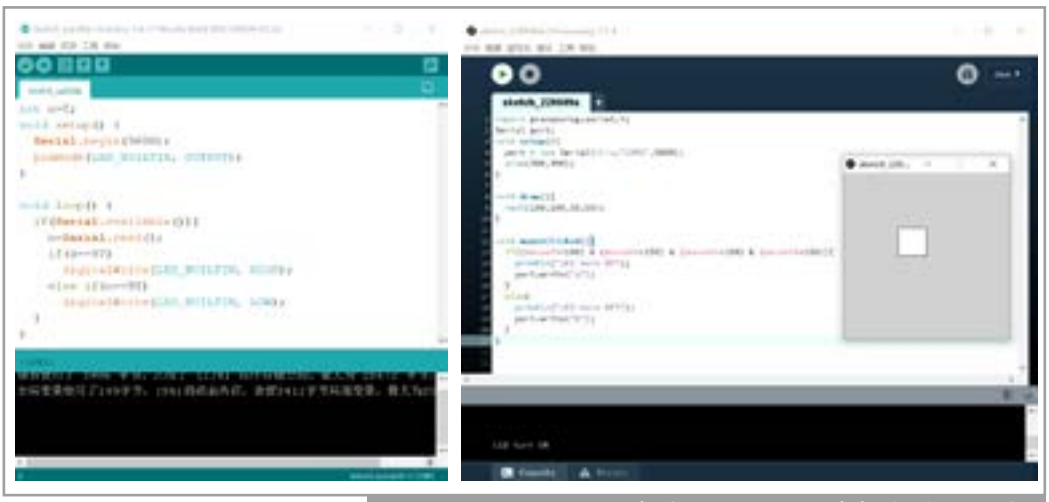


图 5-1 Processing 控制 Arduino 开发板上的 LED 灯

5.2 Arduino 控制 Processing

案例 5-2 为 Arduino 开发板向 Processing 发送字符串。在 Arduino 端初始化设置波特率为 9600b/s，不断向串口发送字符串“a,b,c”。Processing 端从串口接收字符串，可以通过 split() 函数对字符串进行分割。

【案例 5-2】 Arduino 开发板向 Processing 发送字符串

Arduino 端的代码：

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial. println("a,b,c");
}
```

Processing 端的代码：

```
import processing.serial.*;
Serial port;
String rec;
void setup(){
  port = new Serial(this,"COM3",9600);
}
void draw(){
  if(port.available()>0){
    rec = port.readStringUntil('\n');
    String[] a = split(rec,',');
    println(rec);
    println(a[0]);
  }
  else
    println("no");
}
```

5.3 *Colorful Melody* 案例分享

音乐总会引起人们情感的共鸣。交互艺术装置 *Colorful Melody*（见图 5-2）通过与自然元素相近的形式和沉浸式氛围，将每首歌的情感色彩与旋律基调可视化。歌曲的风格以流动的粒子为载体，使心情的表达方式不局限于文字记录。人们通过

图 5-2 *Colorful Melody* 效果图

色彩的展示，在与歌曲中的情感产生共鸣的过程中，逐渐发掘自我的本心，渐渐地用音乐可视化的方式，谱写出属于自己的内心写照。

Colorful Melody 交互艺术装置创作流程如图 5-3 所示。

图 5-3 *Colorful Melody* 系统架构图

5.3.1 创意阶段

1. 和弦色彩归纳

常见的联觉现象是“色—听”联觉，对色彩的感觉能引起相应的听觉，音乐色彩的概念便是由此产生的。音区有低音、中音、高音之分，与色彩中的低明度、中明度、高明度相对应，又如调式音级具有倾向性，与色相的倾向性相对应。每一首歌都有组成它的和弦。通过文献调研，将每个和弦归纳出属于它的颜色，希望通过和弦的颜色，去表达一首歌的情感，如图 5-4 所示。总体来看，大调鲜艳而昂扬，小调沉郁而悠长。

图 5-4 *Colorful Melody* 和弦色彩归纳

2. 交互载体设计

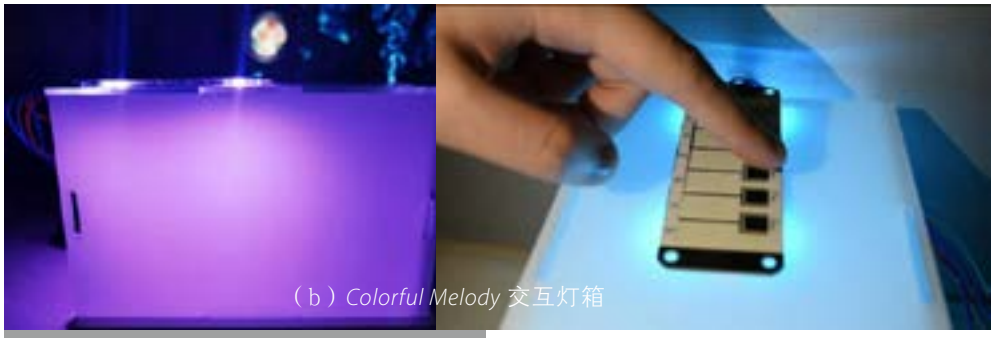
参考 Teamlab 的作品《台风气球・失重森林・共鸣人生》，营造出的氛围为参与者提供沉浸式体验，如图 5-5（a）所示。将灯箱作为一个小的元件，通过氛围感的营造，使用户置身于歌曲的情感色彩中，与作者的内心产生共鸣。根据歌曲的和弦按下灯箱上对应按键，琴键传感器上的 LED 灯会根据和弦亮出其对应的颜色，如图 5-5（b）所示。

5.3.2 Arduino 端实现

1. 硬件搭建

如图 5-6（a）所示，将写好和弦字母的纸条贴在琴键上，避免影响琴键上的触摸传感器，纸条制作要尽可能节省面积。灯箱的制作如图 5-6（b）所示，购买适当尺寸的亚克力板，进行拼接。内部的硬件连接如图 5-6（c）。

（a）*Typhoon Balls and Weightless Forest of Resonating Life* 效果图



（b）*Colorful Melody* 交互灯箱

图 5-5 *Colorful Melody* 交互载体设计

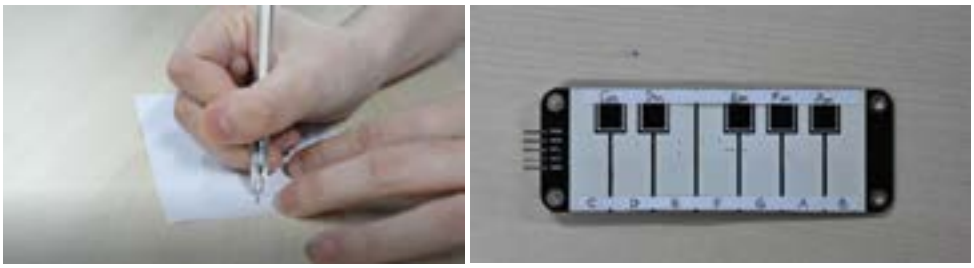
通过琴键（触摸传感器）输入信号，分别向 LED、蜂鸣器、PC 端输出信号，实现在弹奏的同时显示灯光颜色、播放音乐，以及 PC 端鼠标的单击和位移，最后通过鼠标动作实现 Processing 端的视觉效果。

2. 灯箱发光颜色控制

案例 5-3 为控制灯箱发光颜色的 Arduino 端代码，将每个琴键作为一个和弦，赋予其之前归纳好的颜色，其中 LED 灯的 RGB 色彩赋值语句以第一键为例，其余以此类推。

【案例 5-3】灯箱发光颜色控制

```
#include "FastLED.h"          //LED 灯光控制
#define NUM_LEDS 4
#define DATA_PIN 4
const int SCL_PIN = 2;
const int SDO_PIN = 3;
const int buzzer_pin = A0;
CRGB leds[NUM_LEDS];
```



（a）琴键外观制作

（b）灯箱外观制作

（c）硬件连接

图 5-6 *Colorful Melody* 硬件搭建

```
void setup()
{
  FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);
  Serial.begin(115200);
  Serial.println("Start Touching Several Keys Simultaneously!");
  LEDS.showColor(CRGB(150, 0, 0));
```

```
      pinMode(buzzer_pin, OUTPUT);
    digitalWrite(buzzer_pin, LOW);
}
void loop()
{
uint8_t sum=0;
uint16_t keys=ttp229.ReadKeys16();
for (uint8_t i=0; i<16; i++)
if (keys&(1<<i))
      sum += i + 1;
 if(sum==1)     // 第一键 LED 灯光颜色赋值
  {
     LEDS.showColor(CRGB(150, 0, 0));
     QhBuzzerplay(11);
     delay(400);
  }
Serial.println(sum);
}
```

3. 蜂鸣器控制

案例 5-4 是让每个琴键响起对应音调，以控制音准，其中音高赋值语句以第一键为例，其余以此类推。

【案例 5-4】蜂鸣器控制

```
#include <TTP229.h>                          // 琴键音高控制
void QhBuzzerplay(int num)
{
  switch(num){
    case 1: tone(buzzer_pin,261, 400);      // 第一键音高赋值
   break;
   default: break;
  }
}
```

4. 鼠标控制

案例5-5将鼠标控制代码与琴键传感器进行连接，将触碰琴键转化为鼠标单击，

实现 Arduino 与 Processing 的连接互动，为后续粒子的变化提供帮助，其中鼠标方向移动控制语句以第一键为例，其余以此类推。

【案例 5-5】触碰琴键转化为鼠标单击

```
#include <Mouse.h>                    // 鼠标方向移动控制
void QhBuzzerplay(int num)
{
  switch(num){                        // 第一键鼠标方向控制
    case 1:
    if (!Mouse.isPressed(MOUSE_LEFT)) {
      Mouse.press(MOUSE_LEFT);}
else {
    // if the mouse is pressed, release it:
    if (Mouse.isPressed(MOUSE_LEFT)) {
      Mouse.press(MOUSE_LEFT);Mouse.move(10, 0, 0);
    }
  } break;
default: break;
  }
}
```

5. 代码整合

通过 Arduino 开发板将 12 个触摸感应器（对应 12 键，7 白 5 黑）感应信号传出，调用 FastLED.h 库使 LED 灯接收每个信号并分别处理，控制 RGB 显示相对应的颜色；同时输出蜂鸣器发声信号，调用 TTP229.h 库调整代码声音频率使其与琴键音调相符；最后调用 Mouse.h 库，调整代码控制每次触摸琴键的鼠标单击和移动距离，实现传感器硬件结构和对屏幕的总体控制。代码如案例 5-6，其中各部分功能代码仍以第一键为例。

【案例 5-6】Arduino 代码整合

```
#include <Mouse.h>     // 鼠标控制
#include <TTP229.h>    // 蜂鸣器控制
#include "FastLED.h"   // 灯光控制
#define NUM_LEDS 4
```

```
#define DATA_PIN 4
const int SCL_PIN = 2;
const int SDO_PIN = 3;
const int buzzer_pin = A0;
CRGB leds[NUM_LEDS];
TTP229 ttp229(SCL_PIN, SDO_PIN);
void setup()
{
  Mouse.begin();
  FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);
  Serial.begin(115200);
  Serial.println("Start Touching Several Keys Simultaneously!");
  LEDS.showColor(CRGB(150, 0, 0));
      pinMode(buzzer_pin, OUTPUT);
    digitalWrite(buzzer_pin, LOW);
}
void loop()
{
uint8_t sum=0;
  uint16_t keys=ttp229.ReadKeys16();
  for (uint8_t i=0; i<16; i++)
 if (keys&(1<<i))
      sum+=i+1;
 if(sum==1)
  {
     LEDS.showColor(CRGB(150, 0, 0));
     QhBuzzerplay(11);
     delay(400);
  }
  Serial.println(sum);
}
void QhBuzzerplay(int num)
{
  switch(num){
    case 1: tone(buzzer_pin,261, 400);
```

```
    if (!Mouse.isPressed(MOUSE_LEFT)) {
      Mouse.press(MOUSE_LEFT);} else {
    // if the mouse is pressed, release it:
    if (Mouse.isPressed(MOUSE_LEFT)) {
      Mouse.press(MOUSE_LEFT);Mouse.move(10, 0, 0);
    }
  } break;
   default: break;
  }
}
```

5.3.3 Processing 端实现

运用 Processing 实现音乐播放器界面和粒子界面两部分效果。

1. 音乐播放器

案例 5-7 为一个简易的音乐播放器界面的技术实现部分代码，效果如图 5-7 所示。

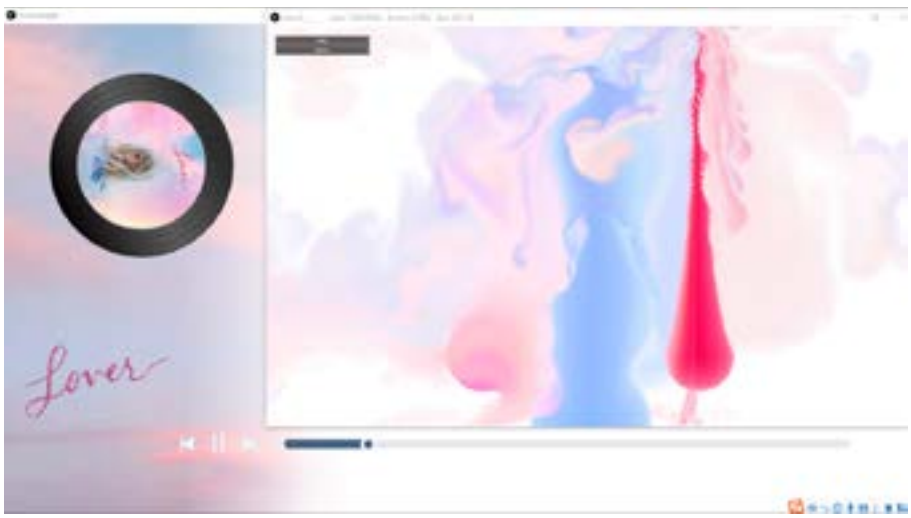


图 5-7 *Colorful Melody* 音乐播放器界面

【案例 5-7】音乐播放器

```
import ddf.minim.*;                    // 播放音乐
Minim minim;
AudioPlayer[] music=new AudioPlayer[2];
```