EE1301：计算系统简介

*物联网实验室2*

*传感器和执行器入门*

作者：大卫·奥瑟、基亚·巴扎根和约翰·萨托里



感谢学生、助教和教职员工不断改进本文件。我们一起做更好的实验！

请将意见和建议发送到orser@umn.edu邮箱

                                                                                        版权所有2018

# 背景

正如物联网实验室1所讨论的，智能设备使用传感器和执行器与世界互动。传感器是检测环境变化的组件。温度计、光探测器和土壤湿度探测器都是传感器的例子。执行器是一种可以操纵智能设备周围世界的设备。阀门、灯、电机或显示器都是执行器。

# 目的

在这个实验室里，你将熟悉光子的输入和输出。使用这些输入/输出，我们将探索光子如何与周围的世界相互作用。这个想法是给你一个什么是可能的概述，反过来，激发你的项目可能包含什么想法。

# 补充资源

设备说明-光传感器（光电晶体管）

设备说明-温度传感器（TMP36）

设备描述-单个可寻址LED（8mm 2811 RGB LED）设备描述-简单扬声器（压电扬声器）

# 实验室前要求

在来实验室之前，你应该复习大量的阅读材料。阅读材料是以“快速课程”的形式提供的，即涵盖单个主题的独立文档。请仔细阅读以下实验前检查表上的所有材料。然后，在线完成物联网实验室2的Moodle预测试。

|  |
| --- |
| 实验前检查表  阅读快速课程-电路  阅读快速课程-了解您的引脚：电源、模拟和数字引脚  阅读Sparkfun Breadboard教程-  https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard网站  阅读Sparkfun教程的后半部分-“如何阅读示意图”https://learn.sparkfun.com/tutorials/how-to-read-a-schematic；名称指示符和值完成“物联网实验室2”的画布预实验测试 |

**所需组件：**

光电晶体管（光传感器）

200欧姆电阻器

4.7k欧姆电阻器

100k欧姆电阻器

3个可单独寻址的LED（WS2811S）

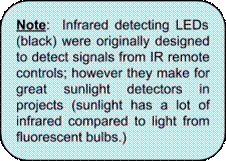
按钮开关

|  |  |  |
| --- | --- | --- |
|  | **EE 1301物联网实验室2** | **欧洲经委会部门** |
|  | |  |
| 电位计  去藕电容 | |  |

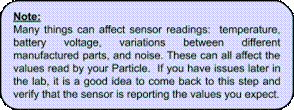
# 实验室程序

## 第一传感器-光线传感器

光传感器对未来的项目非常有用。例如，光传感器可以告诉我们某人何时进入实验室（打开灯），何时阳光照射到办公室，或者何时打开橱柜或储物柜。

这个实验室使用的光传感器叫做光电晶体管。光电晶体管根据照射到它的光量来传导一定量的电流。因为我们的粒子装置只测量电压，所以我们使用一个电阻将电流转换成我们可以读取的电压。图1中的示意图显示了右上角的感应电路。当光照到光电晶体管（Q1）上时，它被转换成电流。电流流过电阻器r1，使感测引脚a1上的电压下降。因此，更多的光意味着更低的测量电压。

|  |
| --- |
|  |
| 图1：光电晶体管测试电路示意图 |

测试和检索传感器数据是评估传感器的首要任务。在下面的示例中，我们将使用串行端口来评估传感器的响应并校准最终代码（如果“串行端口”一词似乎不熟悉，请参阅“快速课程-电源、模拟和数字管脚”）。

### 程序

1.）将粒子和光电晶体管连接起来，如下所示（注意：如果在计算机实验室工作，光电晶体管应该是透明的（所有波长）。（如果在有阳光的空间工作，可以使用有色黑色光电晶体管（仅红外）。

|  |
| --- |
|  |
| 图2：光电晶体管测试电路的面板布局 |

2.）将USB电缆从计算机连接到粒子设备。

3.）打开粒子IDE并创建一个新应用程序。

4.）在声明任何函数之前，我们应该声明一个变量（类型：int）来保存度量结果。

INT数据0；

5.）函数，我们首先需要设置串行端口。**在设置中（）**

//打开与计算机串行通信的串行口，开始（9600）；

注意：上面的那一行可能看起来有点晦涩。（函数serial.begin（）是从哪里来的？）粒子ide被设计成一个非常用户友好的编程环境，并且

只设计用于粒子线的器件；因此，它做了许多假设。其中两个假设是串行端口始终可用，并且库始终由编译器预加载。因此，我们不需要声明“serial”或指定其类型，只需初始化它。

6.）将包含我们程序的工作负载。首先，我们将模拟管脚读入变量data0。loop（）函数

//从模拟管脚读取数据（返回0到4095之间的数字）data0=AnalogRead（A1）；

7.）接下来，我们将这些数据以可读格式打印到串行端口。

//将数据打印到串行端口

serial.print（“我的数据是：”）；

串行打印（数据0）；

serial.println（“；”）；

8.）添加心跳LED

heartbeat led是一个有用的构造，用于验证程序是否已成功加载（或在更改后重新加载）。在我们的应用程序中添加三段代码将允许我们轻松实现心跳LED。

A）在SETUP（）中

//设置D7 pin输出心跳pinmode（D7，输出）；

循环开始时（）

//heartbeat，表示我们还活着digitalwrite（d7，high）；delay（250）；

循环结束时（）

//heartbeat，show we&apos;re alive digitalwrite（d7，低）；delay（250）；

注意：当你的粒子光子行为异常时，心跳LED非常有用。尝试显著改变LED的心跳频率并重新闪烁光子。您将立即知道代码是否正在运行以及是否已更改。

9.）保存、验证并将代码闪存到粒子中。

10.）当您的粒子重置并“呼吸青色”时，打开油灰并连接到相应的串行端口（例如，COM3）。

使用传感器获取一些数据以证明其工作正常（例如，放置传感器，使其能够看到光线，用手盖住并揭开传感器）。

|  |  |
| --- | --- |
| 条件 | 数模转换器读数 |
|  |  |
|  |  |
|  |  |

|  |
| --- |
|  |
| 图：AnalogRead2Serial.ino的一个可能版本 |

## 练习1-温度传感器

这一次，您可以自己使用“设备描述-温度传感器”文档（画布上提供链接）为tmp36温度传感器构建一个串行评估设置。

从传感器获取一些数据以证明其工作正常（例如，被挤压20秒后的空气温度和温度）。注意：小心不要接触包装底部的导线，因为您的身体阻抗可能会改变您的测量值。）

|  |  |
| --- | --- |
| 条件 | 数模转换器读数 |
|  |  |
|  |  |
|  |  |

**注意：如果本周你只做传感器，请看本文档末尾关于如何编写实验室报告的指南。**

## 执行机构-显示元件

在本节中，我们将致力于从微控制器中获取一些信息。廉价集成电路和新封装技术的出现，使得编码器和解码器可以直接嵌入显示元件中。不再需要8+导线以非常基本的方式与显示器通信。如今，使用上述串行端口（4线）的变体将配置信息（通常是几十个变量）传输到外部设备是很常见的。

### 可单独寻址的LED

这些LED是一类称为可单独寻址的设备的一部分。这意味着当连接在一个链上时，每个LED可以单独设置为不同的颜色。每个LED都需要共享电源和接地引脚。此外，每个LED都有一个数据输入和数据输出管脚。当串接（端到端）时，字符串中的第一个LED抓取前24位（颜色设置），然后将其余数据传递到链下的下一个LED，该LED抓取下一个24位（第二个颜色设置），将其余数据传递到上一个LED，等等。但为什么我们有24位来表示颜色？嗯，每个LED内部都有三个小LED：红色、绿色和蓝色。每一个小LED都可以显示其颜色，亮度从0（关）到255（最亮）。因此，24位包含8位来指定每个颜色强度。如果将红色值设置为128，将绿色和蓝色设置为零，则会得到半亮的红色。如果我设置红色=255，绿色=0，蓝色=255，我得到最亮的紫色。

如果你想了解普通LED的功能，请查看CSE冬季灯光秀的YouTube视频。想象一下你可以用一个rgb发光二极管做什么！

https://www.youtube.com/watch？V=1ZJRMUFZTWI型

|  |
| --- |
|  |
| 管脚说明  注：这些设备需要+5V电源。 |

**警告：将这些LED反向插入（即使是一秒钟）也会破坏它们！！！！**

现在，按如下方式连接LED：

|  |
| --- |
|  |
| 图1:RGB LED的电路接线图示例 |

### 图书馆

库是常用的代码集合。它们通常由公司或感兴趣的个人维护。它们使复杂的任务变得更容易。在粒子开发环境中，它们是以一种特殊的方式处理的。

1.）返回build.particle.io开发环境。首先创建一个新的应用程序（单击“代码”按钮，单击

2.）要查看受支持库的列表，请单击按钮。

3.）在本例中，我们将使用一个名为neopix库的库。单击此库。您当前的程序窗口将暂时隐藏，库代码将显示出来供您查看。请注意，窗口顶部有一些选项卡；库通常由几个文件组成，包括供您查看的示例代码。花点时间看看文件。

4.）查看完代码后，单击“包含在项目中”按钮，选择新创建的应用程序，然后单击“确认”。

5.）你会注意到以下几点：

a.）首先，将以下行添加到主文件中

//include语句是由particle ide自动添加的。

#包括<neopixel.h>

第二，在左窗格中，您会注意到一个标有“includelibraries”的部分，其中包括“neopixel”。这很重要，因为库包含程序运行所需的许多文件。

6.）你的应用现在应该是这样的：



### 对象模型

详细介绍面向对象编程超出了本文的范围。理解类是C++编程中的另一种数据类型就足够了。对象是包含唯一嵌入数据和函数的类的实例。这些嵌入式函数中的每一个都称为方法。

我们需要一些基本信息来设置neopix对象：

●支持脉冲宽度调制的数字管脚编号（D4），像素串附在该编号上

●字符串中的像素数（3）

●控制器芯片类型（ws2811，https://www.adafruit.com/products/1734）

然后，您可以创建一个新的neopix对象，类似于我们如何声明一个新整数，只是我们使用一个函数用数据初始化对象。像这样：

**//important:设置像素计数、pin和类型**

**#定义像素4**

**#定义像素计数3**

**#定义像素类型ws2811**

**adafruit\_neopixel strip=adafruit\_neopixel（像素计数，像素引脚，像素类型）；**

这将创建一个名为“strip”的对象，该对象基于对象类“adafruit\_neopixel”，该类在“neopixel”库中定义（还不清楚？，抓住你的短裤……）

在C++（以及许多其他的编程语言）中，我们使用语法“Obj.Mult.（））访问对象的方法。把这些方法简单地看作函数。我们调用方法“strip.begin（）”来初始化strip。我们只需要这样做一次，所以最好的地方是在光子代码的setup（）函数中。如此：

**void setup（）{strip.begin（）；**

**}**

最后，我们已经声明并初始化了我们的对象；我们准备使用它来做一些有用的事情！

我们将使用Adafruit\_Neopixel类对象中的三种方法。下表包含我们将使用的方法的定义。

|  |  |
| --- | --- |
| **方法（函数）定义** | **描述** |
| **void=strip.setpixelcolor（<uint16>，<uint32>）[1]示例：strip.setpixelcolor（pixelid，packedcolor）** | 将第n个像素（零索引整数）的packedcolor（32位“压缩”rgb整数）存储在内存中 |
| **<uint32==条带颜色（<uint8>，<uint8>，<uint8>）**  **例子：**  **packedcolor=条带颜色（红色、蓝色、绿色）** | 返回表示rgb值的32位“压缩”整数  （即红色\*256^2+蓝色\*256+绿色） |
| **void=strip.show（）**  **例子：**  **显示（）** | 在一次突发传输中将存储的配置发送到led条带 |

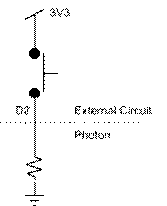
我们将定义两种颜色，然后将所需的数据存储到strip对象中。设置好所有内容后，我们将调用show（）方法将数据通过数字数据链接转储到像素字符串。

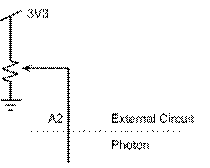
|  |
| --- |
| **void循环{**  **//设置一些颜色**  **int pixelcolorblue=strip.color（0，0，128）；int pixelcolorred=strip.color（80，0，4）；int pixelcolorgold=strip.color（60，50，5）；**  **//将第一个像素设置为蓝色**  **条带。setpixelcolor（0，pixelcolorblue）；**  **//将第二个像素设置为红色**  **条带。setpixelcolor（1，pixelcolorred）；**  **//将第三个像素设置为gopher gold！strip.setPixelColor（2，PixelColorGold）；strip.show（）；**  **延迟（1000）；//等待1秒**  **//设置第二和第三个像素为gopher gold！分别是红色**  **strip.setPixelColor（2，pixelColorRed）；strip.setPixelColor（1，pixelColorGold）；strip.show（）；**  **延迟（1000）；//wait 1sec}** |

验证并向光子闪烁代码。检查LED是否按预期工作。

需要注意的是，neopixels会保留其颜色设置，直到下一个show（）方法调用或5v电源关闭。所以，没有必要不断地更新它们。

## 传感器-人工输入设备

接受人类的输入是微控制器的一个有用特性。虽然这看起来是一个简单的任务，但有几个陷阱。我们将首先设置一个按钮和电位计（旋钮）作为输入设备。

按钮是一个常开瞬时开关；这意味着，如果我们将按钮的一个端子连接到3v3，将另一个端子连接到设置为“input\_pulldown”的光子输入端，则会产生右侧的电路。当按钮“未按下”时，会导致光子的内部“下拉”电阻将输入拉至接地、0.0V或“低”状态。将此视为默认状态。当按下按钮时，输入以低电阻连接到3v3。电阻有一场拉锯战，低电阻路径获胜，将输入拉到3.3V或“高”状态。

电位器是一个三端装置。它在物理上是一个长电阻，两端各有一个触点（端子1和3）和一个“开关”，可以在两者之间的任何地方（端子2）接触。当两个固定端子连接到电源轨（GND和3V3）时，“开关”端子将输出0.0V到3.3V之间的电压，具体取决于开关的位置。这可以很容易地连接到一个模拟输入终端，采样电压值。

如上文所述，在以下电路中连接按钮和电位计：

|  |  |
| --- | --- |
|  |  |

我们现在有两个输入。注意，按钮提供数字输入，电位计提供模拟输入。我们希望对这些输入的状态进行采样，并将其输出到串行端口以进行调试。下面的代码显示了一种方法。

|  |
| --- |
| **int按钮pin=d2；int potpin=a2；**    **int potout=0；bool buttonout=false；int buttoncount=0；**    **无效设置{**  **pinmode（buttonpin，input\_下拉菜单）；pinmode（potpin，input）；**  **序列开始（9600）；**  **}**  **void循环{**  **buttonout=数字读取（buttonpin）；**  **potout=模拟读取（potpin）；**    **if（buttonout==高）{**  **buttonCount=按钮计数+1；**    **serial.print（“button count=”）；**  **串行打印（按钮计数）；**  **serial.print（“，level=”）；**  **串行打印（potout）；**  **}**  **}** |

继续在光子上运行此代码，然后使用终端程序（putty、coolterm等）连接到光子。按下按钮几次，转动电位计，然后再次按下按钮。你应该马上注意到一些事情。

### 事件与状态

您可能已经注意到，按一次按钮会产生多个按钮计数。实际上，按钮计数的数量取决于微控制器的速度和正在运行的代码的复杂性（这不是好事！）.

在我们的例子中，我们真正感兴趣的是事件“按钮被按下”，而不是按钮“向下”的当前状态。从本质上说，事件“button is pushed”需要了解两条信息——按钮“up”的前一个状态和按钮“down”的当前状态。我们可以构建代码来捕获这些信息并在此基础上进行构建。例如：

**buttonNow=数字读取（buttonpin）；**

**if（buttonNow==高&&buttonLast==低）{**

**/在这里做我们的工作；**

**buttonLast=高；**

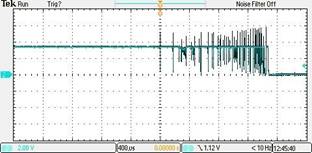
**}否则if（buttonNow==low）{**

**buttonLast=低；**

**}**

修改前一个示例，使每个按钮按一次仅将信息转储到串行端口。

### 去抖动

通常情况下，人机界面会变得更加复杂，因为在确定稳定的输出值之前，实际使用的物理按钮会“弹跳”几次（见右边的示波器输出，显示按下按钮时从按钮采样的电压。）这可能导致在一次按下时检测到多个按钮按下，或一项技术释放按钮时按按钮。

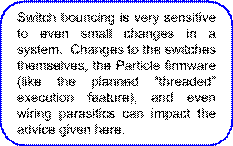
通常，这种反弹持续时间小于2毫秒（但有时可以持续10毫秒）。

光子器件循环函数的最大执行速率为1毫秒，这可以很好地掩盖反弹效应。不幸的是，这有时会导致错误的安全感。所以，如果你的应用程序：

●使用ECE车辆段提供的PCB安装瞬时开关以外的开关

●操作PCB安装开关的“释放”，而不是“按下”

●在单个循环（）函数中多次采样相同的输入

●在系统模式下操作光子（手动），您需要在项目中处理开关去抖动。否则，你现在可能会忽略它。

工具包中包含的PCB安装瞬时开关与loop（）函数的处理时间搭配起来相当健壮，但仍有可能遇到问题。

# 练习2-微型项目

阅读扬声器和伺服电机的设备说明。利用你在这个实验室学到的知识，写一个能感知某些东西（按钮、罐子、温度、灯光等）并能以某种方式响应（扬声器、伺服、LED等）的应用程序。使用实验室里的任何一个执行器或者你自己发现的设备。潜在微型项目的两个例子是：

●音乐盒：按下按钮→播放歌曲或多音警报器

●自动灯：灯位上升→关闭LED灯

●自动风扇：温度升高→打开风扇

在实验室的这一部分中，您可以选择一个特定的微型项目，但有两个概念对嵌入式系统设计非常有用。现在可能是阅读快速课程-编程结构的好时机。看看它是否与你有关。

# 实验报告

对这个实验室的工作做一个非常简短的报告。它应该包括以下内容：

●一段关于您使用所连接的每个设备（传感器、执行器）的经验。

○你第一次连接并编程时，它工作了吗？你犯了什么错误？你是怎么解决问题的？如果你没有任何问题，就这么说。

○包括用于使其工作的相关代码。在描述代码时，将代码分成几个部分并描述如何以及为什么选择使用的特定实现是很重要的。

○对于传感器，包括显示不同条件下DAC值的表格。

●描述你的微型项目。

●实验室助教必须观看电路演示。确保他/她在你离开实验室前检查你的工作（或者，如果你没有时间完成，在实验室外安排一段时间演示）。

[〔1〕](" \l "_ftnref1" \o ")您可以在699行的“neopix.c”文件中看到这个方法的定义。如果需要了解未记录的库是如何工作的，可以在这里查看。