第三章 基于 arduino 的前端开发

目录

本章导读

上一章讨论了四个层次的逻辑与思维，由于同学们没有太多系统开发的体验，在理解上，相信有一定难度。这就应了那句俗语：“不当家不知柴米贵”，只有真正参与系统开发，才会有所体会。与上章四个独立的游戏程序不同，从本章开始（因篇幅所限，完整项目分配在 3-5 章中），用三周的篇幅，为同学们设计了一个比较完整的“系统级”实训项目——“基于树莓派的物联网应用系统”。

本章 1-3 小节是实训项目的导论，介绍项目目标和开发方法。本章以下各节，以及第四章、第五章，分别完成整个项目的一个子系统的项目开发。后一个子系统会应用前一个子系统的开发成果，最终完成的是一个基于树莓派的“系统级”综合性应用系统项目，并由此来体验一个小型应用系统项目的开发与集成的方方面面，以及开发与集成过程。

相比前一章的四个小游戏，虽然项目大了、涉及的东西多了，系统相对复杂了，但每个子系统（每个环节）的功能需求、实现算法和代码量都不大，系统的“集成度”和“复杂性”相对也不算很高，离真正的“实用”就差得更远了。

“基于树莓派的物联网应用系统”涉及多种传感器信息采集、数码管显示、Arduino 开发平台、树莓派 USB 接口与数据传输、在树莓派上对信息进行初加工和反向控制、简单的用户界面设计、树莓派与物联网数据服务器的交互，直到在云上，建立自己的物联网应用服务器，让 IE 或手机客户端 APP，可以直接运用云服务器上的数据或应用。这一长串的“应用链”，包含了很多当前应用开发的热点，同时，涵盖了很多应用系统开发与集成的“关键知识点”，是同学们了解、理解、体验和学习实践系统开发与集成的很好例子。

在实训项目开发之前，教程还简单讨论了参加物联网大赛的项目创意、实现技术的可行性等课题。参赛即是对课程学习成果的检验，更是启发学生创新意识、提高学习兴趣、调动学生自主学习、深入探究，并与未来的专业课程学习、就业创业“对接”的好方式。

3.1 从“码工”到架构师

在软件开发的职业生涯中，如果从来没有带过“团队”（哪怕带一个人，而不是只是自

己“管”自己)、从来没有“管”过一个“系统级”项目的“需求”、“系统设计”等软件开发过程环节，这个人可能就是一个标准的“码工”。很多号称有 5-8 年、甚至更长软件开发从业经历的人，其实从来没有真正做过“系统工程师”（管事）或“项目经理”（管人）的工作。因此，一旦涉及“系统”、“项目”，他们是心虚的。国外企业还真的能看见 50 岁以上的编程大叔，国内因为“人口红利”，暂时还不会出现这样的情况。

系统工程师和项目经理要管人和管事（有所区别），但并不一定意味着就是“当官”的，无官，却必须由你拿“决策”。这句话涉及两个东西：“团队”和“系统”。前者是人，后者是“物”。现代软件开发早已不是个人“单打独斗”式的“写小说”，也不是师傅带徒弟的“作坊式”开发，现在最“时髦”的开发组织形式，已经不是 CMM/CMMI，也不是“敏捷”，而是“群体软件工程”了。软件开发中，人的组织和管理不是本课程讨论的内容，在此不多说。“系统级”的开发与组织能力，则是软件工程师的职业生涯中，跨越“码工”、成为真正系统工程师、必须经历的、饶不过去的门槛。

3.1.1 “系统级”应用的概念

从“码工”上升到“系统工程师”，是不是“软考”拿个证，就是系统工程师了？

1）系统工程师的“系统”是指什么？

软件专业的同学都知道，系统工程师的“系统”二字，首先指的是软件工程中“系统设计”一环（瀑布模型中），而主持并负责这个环节的人，就是系统工程师。在相应的教科书上，有关“软件系统”知识学习，包括各种考试，通常都是指“系统设计”。这可能是上个世纪 60 年代软件工程概念创立之后，一直延续至今的认知。

“系统”的另一个含义，应该指软件开发目标，即系统设计的对象——软件系统。例如：操作系统（系统软件）、数据库系统（支撑软件系统）、开发平台 IDE（工具系统），而另一类软件系统，就是应用软件系统。很多年前，中国软件行业协会有过一个统计：中国的软件企业，98%以上，是做应用软件的。估计这几年，这个比例还是变化不大。显然，作为中国的软件学生，毕业后从事软件开发的领域也将主要集中在应用软件开发领域。在校学习的目标对象，当然主要是应用软件系统。所以，教与学所关注的“系统”，即：应用软件系统。

前后两个“系统”有什么区别？前者（软件系统设计的“系统”）是软件工程过程，后者是软件工程过程的开发目标和对象。在本教程中，一般称前者为“架构”，而称后者为“软件系统”或“系统”。

在国内软件专业的相关课程教材中，不论是编程、还是软件工程类教材，一般只涉及前者（如：主题为《软件系统分析与设计》类），很少或几乎没有一个较完整地介绍、分析、帮助学生实际体验一个应用系统全貌的教材。有一些针对专门应用系统平台的、如 Android、iOS 的教材，但这里的“系统”指的是 Android 或 iOS 系统，而非针对一般的应用系统，更特特别缺乏针对特定行业和领域背景的应用系统。因为只有在比较复杂的业务背景下，才能真正涉及关键需求，否则确实有“无病呻吟”之嫌。

以简单系统（如：图书管理、仓库管理）为对象的系统分析和设计（软件工程过程），以及应用系统开发（结果）能够表现系统架构分析和设计的复杂性、综合性、完整性、灵活性、可扩展性——所有与系统架构有关的哪些关键质量属性吗？MVC、设计模式在图书管理、仓库管理中，能发挥什么作用？甚至有必要把这样的简单应用（数据库的增删改查）做成三层结构吗？

所以，“系统级”的概念，首先是要研究“系统”本身，这是“物质基础”，然后才是“系统设计”，系统设计是基于复杂系统的设计、是基于“系统”的设计和实现过程，是“上层建筑”。脱离了物质基础的上层建筑，无疑是“空中楼阁”、“闭门造车”，结果是教的人无聊，学的人无趣。

2）“系统级”系统的商业价值：

为了避免误解和啰嗦，一般所谓“系统级”的系统，通常称为“企业级”系统。“企业级”不是指只在企业使用，而是表示软件系统更专业（行业热点）、更深入、更全面、更完整，更复杂，也是价格更高。Windows 软件、数据库、各种工具，都有普通版、企业版、豪华版等等，除了商业“噱头”之外，软件产品本身在功能、性能等方面的差异，构成了不同版本的基本特征。

从这个意义上看，图书管理、仓库管理系统无论如何也不能称之为“企业级”的系统。哪怕它确实用在企业、确实有很多零碎的功能。如果用一个简单的指标来度量一下这样的“系统”的价值或“份量”，可以看看这种系统，作为产品的真正“市场报价”。如果看不到，还看看项目发包网上，做一个这种项目的发包价格。用 MIS 生成工具，10 分钟给你做出来，你说打算卖多少钱？

从学生的角度看，在有限的学习时间和精力条件下，应该学习有用的、有价值的知识，而不是陈旧的、过时的伪知识，甚至是“垃圾”知识。

3）“企业级”应用的特点：

企业级应用是指那些为商业组织、大型企业而创建并部署的解决方案及应用。这些企业级应用一般具有拓扑结构和系统逻辑结构复杂、涉及其他系统、外部资源接口众多；事务密集；数据量大；在线用户数多；用户体验要求高；系统可用性、可管理性、可维护性、可扩展性、安全性等要求高等特点。这些特点，无一遗漏地需要在最后完成的应用系统中，得到响应和体现。包括哪些不在开发期、也不在运行期、属于“未来性”的需求，如：可维护性、可扩展性、灵活性等。

4）架构师的任务与责任：

为此，架构师的责任是：

1、 领导并负责架构设计；
2、 实际参与架构原型的开发实现；
3、 讲解架构、指导开发、协调冲突；
4、 支持项目经理，如技术可行性研究、任务划分、人员招聘等；
5、 了解所在组织的业务目标，令架构更好地支持组织的业务目标；
6、 评估新技术，并提出采用建议等。

5）为什么学习的目标是“企业级”的：

道理很简单，因为我的职业目标是“架构师”，是设计“企业级”系统的人。当然，也可能我未来并不一定能做架构师、甚至未必去到软件企业，但是，现在我的专业就是软件，“不想当将军的士兵不是好士兵”。

3.1.2 从“码工”到架构师

由于“企业级”与“代码级”开发具有那么巨大的差异，设计“企业级”系统的架构师要知道得更多、更全面、更系统、更长远。

作者曾经帮助微软组织过一次高校教师软件架构培训班，课后有老师问微软讲师，架构师不是“官”（提问老师的意思是架构师至少应该挂个什么经理之类），架构设计约束（架构纪律）如何能让项目团队成员遵守？微软讲师的回答是三个字：靠沟通。提问的老师仍然不能理解，沟通什么能让别人放弃自由，老老实实地受你约束?微软老师只是微微一笑，不再回答。提问老师依然一头雾水，而个中意味大概我是读懂了的。这就是：从校门到校门的老师，如果没有在软件开发团队、在那种不靠行政命令，靠相互信任和配合做事的环境中呆过，如果没有真正接触过需要“架构”（而不是“无病呻吟”）的“企业级”应用开发，还真的是难于给你说清楚这个问题，所以他们也不想再多解释（说你也不懂）。

一般认为，软件架构的作用，主要体现为 4 个方面：在系统设计层面上（1）表达系统的功能需求是如何被实现的；（2）表达关键需求和设计约束的实现方案；（3）作为迭代开发的基础；（4）成为过程管理的依据。前两点反映的是对软件系统本身的作用，后两点，反映的是对软件开发过程的作用。

为了能够同时在上述四个方面，承担起架构师的责任，发挥出“有架构设计”的作用，架构师应该具备什么样的知识结构和能力？一个普通的“码工”是如何成长为架构师？

虽然作者自己也是教《软件架构设计》这门课的，但我还是要大声地说：架构师不是在大学里培养出来的，如果没有一定年限的实际开发经验积累，博士毕业也不可能立即成为架构师，这就如 MBA 毕业，并不能马上就当厂长一样。

1）架构师需要什么知识和经验？

一名合格的架构师主要需要两方面的知识和经验。专业领域方面：架构的第一个作用是实现用户的需求，架构的另一个作用是预见用户未来的需求，并在架构上，进行迭代实现。因此，如果没有专业领域的知识和经验，则好像李逵在黑夜中舞板斧。软件开发方面：不言而喻，如果没有参与过实际开发，不知道构件、连接、连接关系实际是什么，代表什么、意味着什么，则简直无法想象他能够理解架构的含义，以及架构师如何承载需求、能够理解关键质量属性需求，架构如何响应，并为之进行设计。

2）编码工程师可以自然成长为架构师吗？

答案是否定的。理由是：如果你在业务和编码两个方面，都有了很多年的实际锻炼，积累了不少经验，应该说具备了成为一名合格架构师的基础。但是，只有基础是不够的。你还需要慢慢培养自己作为架构师的领导意识和领导能力。

3）什么是架构师的领导意识？

在《组织行为学（管理心理学）》课程（作者也曾教过这门课，因为这是软件工程学生的必修课）中，“领导”有名词和动词之分。名词的含义是：“领导”是一个职位（职务）、是一个“位子”、是一种权力。而动词的含义是：领导是决定目标和方向，并激励众人实现目标的活动。二者不是简单等同。领导意识首先是目标意识。在架构师的眼里，组织的目标、用户的目标、系统要实现的目标，都要成为团队自觉的追求和行动的方向。相反，哪些过分追求完美、追求个人兴趣、强调困难的行为，都必须从被迫地被“抑制”，到自觉地被“打消”。领导的另一个意识就是“大局”意识，作为架构师，没有大局观是无法想象的。因为架构师的基本工作，就是“布局”。

4）什么是架构师的领导能力？

软件架构设计能力是架构师的基本能力，除此之外，作为领导者（动词），还需要具备组织、计划、沟通、平衡、协调等能力。因为架构师不是编码工程师，后者只要按要求把自己的事情做好就可以了。而架构师的工作是承前启后、他的工作成果要成为别人工作任务的依据和继续“迭代”的基础和平台，他要通过有形和无形的“手”，约束大家遵守架构设计约束，没有这些领导能力怎么可能。

5）什么时候开始提升自己的架构师领导能力？

不是领导宣布你成为架构师的时候。有时，架构师是自发形成的，就看谁觉醒得早。架构师是一个软件开发团队的客观需要，是“刚性需求”，即使没有人任命，也会客观存在，就像黑社会组织中，一定有一个“带头”的大哥存在一样。

3.1.3 “企业级”系统学习与实践的缺失

简单讨论了“企业级”系统之后，再看一下什么不是“企业”级的？所有编程语言课程所教的、练习的、编写的代码、《数据结构与算法》、《操作系统》、《数据库原理与应用》、《网络原理与应用》等等课程中所编写的代码，都不是“企业级”的，当然，这些课程也可以说，

我们不是教这个的。有课程教这个吗？《软件工程》？看看课程大纲，作者的《软件架构设计实践教程》课也不是专门教“企业级”应用开发的，但我要求学生有“较大型”的软件系统的开发经历和体验（《软件工程》、《软件项目管理》等高端课程都是有这个要求的），否则无法理解架构设计。显然这个知识基础和先修条件，在现有情况下，是不可能满足的（不但不可能，当我来上《软件架构设计与实践》课的时候，不少学生连 VS2010 菜单的按钮是干什么的都不知道）。有老师教“企业级应用系统”开发吗？很遗憾，可能很少。

放在国家高等教育和学校的大环境下看，出现这种问题的原因是很容易理解。

长期以来，我国高等工程教育虽然一直在探索课程内容、结构和形式的改革，但是由于目标不明确，未能以工程教育的最终目标作为课程改革的方向，所以并没有从根本上解决课程体系中存在的实践性与综合性缺乏的问题。

1996 年国家教委工程教育考察团赴美考察，中国开始接触“大工程观”的教育理念，2006 年 6 月，教育部战略研究重大专项“面向创新型国家建设的中国工程教育改革”研讨会就提出：要使课程设置从单一的“工程专业课程”传授，转变为“大工程观、大系统观”为指导的课程架构体系，“大工程观”的教育理念早在 20 年前就已经提出，不是“新时髦”。

这20年多来，有相当多的专家和学者，从理论到实践，对“大工程观”的本质和内涵、实践的可行性，进行了大量探讨，认为“回归工程实践”和整体性的“综合集成”，正是切实解决我国高等工程教育“实践”和“综合”二大根本问题的有效良方。

在实际教改实践中，通过开展多种形式的教学活动、增加学生的实验和项目实践、开设实训课程等形式，包括提出“做中学（CDIO）”战略等，在强化实践性教学方面，取得了一定的成效。

但是，由于可以理解的原因，工程课程体系和教学方式的综合性改革，由于涉及面广、受现有的教学方式、师资队伍、行政化的教育管理体制、高校教育目标与考核指向与社会需要的脱节等诸多因素影响，实际实行起来，难度很大，进展有限。这个情况，可以类比 30 多年前改革开放初期，计划经济时代下国营企业的改革。工资和奖金激励等浅层次的改革，虽然部分调动了职工的积极性，但并不能从根本上解决企业在市场经济环境下因所有制等深层次问题，导致的竞争力缺乏的问题。

就大部分院校而言，目前的高校组织管理机制，已经形成行政管理（占有教育行政资源）和专业导向（以个人的社会知名度为基础、以小范围的学术体系为核心、以非工程化为指向）相背离的两张皮运行机制。这样的体系，更适合“做课题、写论文”，而不适合教学、更不适合学科综合的教学。

院系的管理者出于自身的专业和管理能力的限制，更多从事的是行政事务性的日常管理。因此，在制定教学和培养目标、整合资源、协调教学内容等方面，难于发挥主导和总揽全局的作用。而教授们多为“清客”，脱离社会实际，也少有校内资源。因而，形成上面说的多、下面做的少；领导花架子多，教学实际成效少的局面。而这种局面的改变，根本不可能靠教育部、各级教育管理部门，立几个教改项目、专业建设项目，可以改变的。

所有这一切的结果，才能够解释教学效果为什么那么差的问题。

3.2 小型系统开发与集成实训

多说无益，还是提升自己的实际能力是最关键的。

本教程分三个阶段，进行架构师素质、能力的培养和训练。第一个阶段是上一章的《软件的逻辑与思维训练》，是从思想方法上，分四个层次，对未来的架构师，进行思想启蒙。第二个阶段是本章到第五章，希望通过一个基于树莓派的小型物联网应用系统的搭建，体验“系统”的概念。这一阶段所涉及的“系统”，还不是“企业级”的，而是“小型”的。因

为你需要先知道并体验什么是“系统”、什么是系统开发，然后才能进入到“企业级”的、大型系统的分析和开发过程中。否则，你会“消化不良”的。在最后的四章中，教程花了大量的篇幅（全书的一半篇幅，约 14 万字），带着你分析一个真正的“企业级”系统——《策略为王》股票软件系统，并指导你在此系统代码基础上，进行二次开发。

3.2.1 本次实训课程的目标

作者开设本阶段课程的最初目的，并不是为了给同学们增加小型应用“系统”开发的体验，而是为了提高或调动大家学习软件开发的兴趣。因为理由很简单，在上一章 2.2.9 节提到：C 语言课程这样教法、这样的上机能力和水平，到了高年级，上课不睡觉那才怪了呢。为此，需要在“编程”中添加“兴趣”内容。但是，学生的业务背景、编程能力那么差，有兴趣、没能力也是白费！不能只看老师“演示”吧？问题明摆着，光发牢骚也没有用。于是想到了“树莓派开发”。开设《树莓派开发》课程，可能在全国高校中，作者是第一个吧？

1）培养兴趣：

兴趣是学习的动力来源，现在学校的课，实在是太无聊了。兴趣是什么？那方面的兴趣？老师在课堂上像郭德纲才有兴趣？（也有老师真的是这么做的）。作者认为，只要努力挖掘，不论学生基础多差，培养学习的兴趣、探索的兴趣、创新的兴趣，还是有可能的。本课就是一次成功的尝试。

先画一个“饼”：给你一个平台（树莓派）；给你一个工具（基于树莓派的开发工具）；给你一些成功的例子（开发代码）；给你一个思路（参赛项目训练），最后，告诉你一个机会：2016 微软创新杯大赛、2016 IBM 软件设计大赛、2016 物联网创新大赛，……（中国现在已经是大学生大赛热啦！）。到美国、到“西雅图”参赛，你没有兴趣？只怕你没有这个胆量和能力罢了。但是，我告诉你，2015 年及再往前 10 年的参赛主题、获奖团队项目，你看看？不过如此嘛！我再告诉你，当你做完了本课程的实训项目，哪些参赛的课题，对你来说，也不是什么了不起，高不可攀的项目。

老师，你早点这样告诉我啊！

2）制订一个走向成功的 N 年计划：

老实说，你现在还不行。别人也不是“吃素”的。更何况，中国的学生够聪明，也是蛮拼的。04 年的时候，微软的创新杯大赛，当时的中国学生不需要国内选拔，可以直接去西雅图参加决赛（作者的儿子，也是作者的学生，当年大三的时候就去了西雅图）。后来，到西雅图的项目团队，有一半是来自中国的，微软没办法了。现在中国学生要“自相残杀”，拼到中国区第一、才能去美国。所以，你的对手不是老外学生，而是中国学生。也公平也不公平是吧？

不要指望通过一次实训，就出去参赛获奖，有的是时间。我们的计划是：第 1 年：启蒙（培养兴趣）；第 2 年：基础（找课题内容）；第 3 年：实战（想点子、出创新课题、准备参赛）。

3）3 周实训课程的目标：

根据 3 年计划，第一年是培养兴趣、开拓眼界，知道外面的世界原来是很精彩滴。3 周实训的目的是：第 1 周：感受一下树莓派这个平台是什么？为什么选这个平台而不是其他平台；第 2 周：体会一下基于这个平台可以做什么？初步掌握基于这个平台的扩展开发的能力；第 3 周：试试如果要做课题，可以朝什么方向发展？如果有想法、准备继续……

在课题方面，老师会给出课题需求、给出基本的技术思路，可能你现在还不具备实现课题的能力，那么就怀揣这个梦想，继续学习吧。课程结束以后，你会不会说：我自己也去买一个树莓派？

所以，不论是为了参赛，还是为了以后的课程学习，一切从培养兴趣开始。通过这些实

训，你可能看见“希望”，但是，你更多的是感到“心有余而力不足”。因而，是否可以把兴趣、好奇、希望、野心，转化为学习的动力?

这是否太“功利”？对于国内二本及以下的学生，“仰望星空”那是在做白日梦。

4）本次实训课程的两个关注点：

与上一章的实训内容不同，本次实训课程希望同学们关注两个东西：（1）小型系统开发与集成；（2）物联网应用。前者是实训课程的核心，是框架；后者是内容、是问题域和问题的解。

与上一章类似的是：本次实训课，也是通过若干小节课程，从“平台选择”、“外部扩展与数据采集”、“串口通讯与前端集成”、“Web Socket 传输与后端集成”、“云应用开发”等几个阶段、循序渐进地介绍一个基于树莓派的小型物联网应用系统的开发与集成过程。各个阶段分别涉及了一般应用系统开发的一些主要方面，而从传感器数据采集、直到云服务器和用户客户端应用，则涵盖了物联网应用这个特定目标系统的“全过程”链。因此，在整个实训课程中，要同时关注这两个方面。只关注前者，那是“空中楼阁”、“纸上谈兵”；而如果只了解后者，则最多只能是一个用户、一个产品经理，而不是一个架构师。

3.2.2 本次实训课程的课时计划

有关实训课程的时间安排，可参考如下计划。

1）先修基础：

本次实训课程，对学生的编程能力要求并不高，但需要的知识比较“杂”，面比较广。包括基础的 C/C++、java、python 程序设计、简单的网络与编程、基本的 linux 操作系统命令和 VI 知识、简单的 Web 网页和后台开发，甚至还需要一点逻辑电路的开发知识和动手能力。

2）实训时机安排：

笔者建议本次实训时间安排在大一或大二两个暑假中的一个。在这个时间段，学生至少已经学完了一些基础课程，开始学习专业基础课。与学校的某些专业课程不同，本阶段实训课程所涉及的面比较广，需要的是综合能力，所涉及的技术，并不能简单地归到某门课程中，因为这里训练的是“系统”开发。

3）课时安排计划：

建议教学计划与内容安排（以 3 周、每天 6 小时为例）如下：

天	课程内容	知识点与学习目的
1	实训课程介绍、学生组建团队；树莓派平台介绍；完成镜像 SD 下载、烧制、安装。	了解实训课程； 了解开发环境； 完成系统软件准备。
	配置树莓派，启动系统。 实现 SSH 或 VNC 访问。	点亮树莓派，建立可运行的树莓派系统（Linux），实现 SSH、VNC 等远程访问。
2	学习和使用基本的 Linux 系统命令。	了解最基本的 Linux 操作命令和基本操作、系统命令行和图形界面等。
3	为系统添加 U 盘、有线/无线网络，让树莓派成为 Wifi 热点。	为树莓派添加一些实用的功能，了解与网络、网络协议有关的知识。
	让树莓派成为一个 Apache 服务器。	建立一个基于 Web 的应用系统，了解搭建 WEB 服务器的基本知识。
4	让树莓派成为一个媒体播放中心。	实现多种媒体播放。

4	在树莓派上玩大型 Quake3 街机游戏。	体验树莓派的互动游戏。
5	将树莓派改造为一个视频监控平台。	实现多种远程视频监控，了解与视频播放、视频控制有关的知识。
	休息	
6	树莓派 Linux 系统内核定制改造（可选）。	学习 Linux 内核定制方法，进一步深入理解 Linux。
7	树莓派 GPIO 扩展开发介绍。	学习有关GPIO接口硬件、接口控制编程知识，为进一步的开发做准备。
	使用树莓派的 GPIO 接口实现点亮 LED 灯泡。	学习外设控制的软件实现技术和方法，体验树莓派控制外部设备的能力。
8	树莓派点亮数码管。	学习用软件方法，实现更为复杂的外部设备控制，体会程序设计的逻辑与实际作用。
9	用树莓派远程控制家中的电灯、家中的温度湿度。	学习用远程访问和服务的方式，实现远程的控制，包括：传感器采集、树莓派控制、网络服务器应用、手机客户端应用等一系列环节。
10	用树莓派控制其他外设体验。	包括：红外、温度、湿度、气压、光敏、气体检测、声音识别、烟雾识别、火焰识别、振动、方向、重力、夜视、加速度、射频、步进电机、陀螺仪、超声波测距等 20 多种传感器应用。
	休息	
11	Arduino、IDE 介绍。	了解一个与树莓派配合使用（外设控制）的单片机的硬件和软件开发平台 IDE、开发语言等。
12	树莓派与 arduino 结合的应用系统相关技术介绍。	包括：自动控制、图形图像识别、传感器、数据处理、网络技术等，为以后的学习，打开兴趣的窗口。
13	其他应用技术介绍。	树莓派上的 java、交叉编译、Hadoop 等。
14	基于树莓派的参赛创意设计与实现。	学习参加大赛的创意、实现技术和手段、可行性验证、实现过程、参赛经验和技巧训练等。
15	考试、课程总结。	课程考试、总结，撰写实训报告。

内容安排说明：

第 1 周（第 1-5 天）为课程基础部分，主要目的是让学生了解树莓派，并建立“玩转”它的兴趣。

第 2 周（第 6-10 天）为简单应用部分，主要目的是通过一些简单的设置，少量的代码开发，实现基于树莓派 GPIO 的扩展控制功能，让学生初步体验自己对树莓派的控制能力。

第 3 周（第 11-15 天）是本实训的最主要部分，通过在 arduino 上的扩展开发，让学生具备初步的树莓派应用开发基础。并为最后“从创意到实现”（参赛），提供技术路线和方法的可行性。

3.2.3 实验设备配置

（1）实验设备采购与试验环境准备（标配）

	设备名称	配置
1	树莓派主板（主机）	树莓派 / 树莓派 Mode B+ ，512M Ram，700M 赫兹；
2	外壳	树莓派 B+亚克力外壳 机盒 组装式 透明 可配螺丝螺帽 可配风扇；
3	无线网卡	EDUP USB 无线网卡 EP-N8508GS 仅树莓派 B/B+免驱 raspbian pidora；
4	电源	树莓派专用电源 充电器+开关电源线 套装 足额 5v 2a；
6	SD 卡	闪迪 8G TF 卡 CLASS10 高速 MicroSd 卡 含卡套；
7	HDMI 转 VGA 线	树莓派 B/B+适用 HDMI 转 VGA 带音频/DC 口 送音频/电源线 可不供电使用；
8	散热片	树莓派 B/B+适用的散热片 超频 纯铜 2 枚装 可放进外壳；
	合计	以上为标配

（2）实验设备采购与试验环境准备（GPIO 扩展）

	设备名称	配置
1	GPIO 面包板	面包板 165×55×10mm 830 孔；
2	面包板电源	面包板电源模块 5 伏 3.3V 给面包板供电 不用再从开发板引出电压；
3	公对公/母对母杜邦线，10CM	1 排 40 根（出厂默认数量），杜邦线是可以随意撕开拆分的，你比如要用两根（2p）杜邦线，那么你从 40p 中撕掉 2 根下来就行。当然可以随意拆分得到；
4	超声波测距模块	HC-sr04 超声波测距模块 避障传感器；
5	红外感应模块	人体红外感应 模块 热释电 红外传感器 进口探头 HC-SR501；
6	温度模块	DS18B20 测温模块 温度传感器模块；
7	湿度感应模块	DHT11 温湿度模块 传感器 电子积木 Arduino DHT11 模块；
8	声音传感模块	声音传感器 FC-04 检测识别有无声音 口哨模块 声控开关；
9	光敏模块	光敏电阻模块 光电传感器 光线检测 光敏二极管 ；
10	土壤湿度模块	土壤湿度传感器 YL-69 土壤湿度计检测模块；
11	红外夜视模块	红外夜视监控摄像头模块 500W 像素；
12	无线收发模块	NRF24L01+ 无线模块 功率加强版 2.4G 无线接收发 通信；
13	烟雾感应模块	烟雾气敏传感器 模块 甲烷 MQ-2 液化气可燃气体；
14	雨滴感应模块	雨滴传感器 下雨 天气模块 大面积雨滴模块；
15	气压感应模块	GY-68 BMP180 BOSCH 气压传感器模块；
16	红外接收	红外线接收器 TSOP4838 VISHAY；
17	数模转换模块	PCF8591 AD/DA 模数/数模 转换模块；
18	数码管	数码管 2 位共阳红色 0.36 寸 3261BH 15*14*7.2mm；
19	蓝牙模块	蓝牙 4.0HM-10 模块串口引出 ；
20	300 万像素摄像头	300 万像素 USB 摄像头免驱动；

21	GPS 接收器	USB GPS 接收器导航；
22	光强测试模块	GY-30 数字光强度检测模块 光照传感器；
23	倾斜传感器	倾斜/倾倒传感器模块倾斜开关角度模块/开关；
24	颜色识别传感器	TCS3200 颜色传感器颜色识别颜色感应模块；
25	水位传感器	水位传感器水分液滴水深检测；
26	其他	电阻、电位器等。

以上部件可根据学生试验需要采购部分或分组采购。

3.2.4 本次实训的二次开发项目选择

与上一章 4 个独立的游戏程序不同，本章虽然也是若干节“开发实践”课，却为同学们设计了一个比较完整、相互关联的小型“系统级”项目——“基于树莓派的物联网应用系统”。通过本章的 4 节实践课，每节课完成一个完整系统开发的子项目，每一节课运用前一节课的开发成果、第 4 节课运用前面各节课的成果，最终完成的是一个基于树莓派的小型“系统级”综合性项目，由此来体验一个小型应用系统项目的开发与集成过程。

相比前一章，虽然项目大了、系统复杂了，但每节课（每个环节）的功能需求、实现算法和代码量都不大，系统的“集成”相对也比较简单，但涉及的面比较广。“基于树莓派的物联网应用系统”涉及多种传感器信息采集、数码管显示、Arduino 开发平台、树莓派 GPIO/USB 接口与数据传输、在树莓派上对信息进行初加工、基于树莓派的控制、树莓派与物联网的交互，直到在云上，建立自己的物联网应用服务器，让 IE 或手机客户端 APP，可以直接运用云服务器上的数据或应用。这一长串的“应用链”，包含了很多当前运用开发的热点，同时，涵盖了很多应用系统开发的“关键知识点”，是同学们了解、理解、体验和学习系统开发与集成的很好例子。

3.2.5 本次实训课程的授课方法

本次实训，拟安排 15 天（3 周）或更多时间。从上一章的实训教学中，已经体会到，实训课程是老师一边讲，学生一边做。由于本次实训的大部分项目还包括硬件，所以，还是很“好玩”的。

与上一次课一样，老师会先讲一讲，也会用代码演示一下，接下来，会要求同学们模仿老师的方法，换一种采集头、自己到网上去找相应的硬件使用说明书、逻辑图、接线图和程序代码，自己实现信息采集和加工、应用。

老师会一步步加“内容”、加“需求”，直到构成一个“大项目”。每一步，老师都会做些讲解和演示，但是，同学们绝对只能模仿，不能照抄。因为老师讲的，一定不是你要做出来的东西。老师的责任是通过需求和示范，进行“引导”，而你需要听懂老师讲的内容和示范（这是“地图”，不看就要多走一点弯路），自己按地图，走到目的地。中间过程可能没有正确答案，可能老师也帮不了你。在软件企业做开发，哪里有正确答案在等你啊？有谁可以一直帮你到底啊？早点适应这样的生活吧！

3.2.6 实训课程的意义、价值与检验

本章实训课有两个目标：1）培养兴趣；2）体验小型应用系统开发过程，为下两章更复杂的项目开发做准备。

1）分组：

原则上不分组，每个同学独立完成项目开发任务；

2）考试：

3 周课程，中间有 6 次项目成果交付验收，最后有一次综合考，全部在机器上完成，按

交付时间早晚、快慢算成绩，老师也是够狠的！

3）成绩：

1、平时成绩

- ✓ 上课签到（上午、下午），占总成绩的 20%；
- ✓ 6 个小项目的提交，老师亲自验收到每个人，占总成绩的 30%；

2、课程结束机考，占 50%。

4）课程要求：

与一般课程相比，老师的作用更多的是引导，在课程中，同学们一定要自己动手、一定不要怕困难、一定要课后继续努力。培养兴趣和钻研精神，不是一次课程能够完成的任务，但是，这对你的一生非常重要。

3.3 树莓派应用开发与参赛

作为实训项目的内容，作者选择了“树莓派”平台，并在此平台上，引入“物联网”、“云计算”、“大数据”等应用课题。因为这些课题不但是“热门”，学生会比较有兴趣、有很多“题材”可以挖掘、并继续深入做下去、以成为学生参赛的基础，而且也因为这些课题的开发内容，可以相对比较完整地涵盖一个小型系统（当然也可以做得很大）的方方面面。

本节将从软件需求和系统架构基础的角度，简单讨论这些应用的基本问题。关于软件系统开发与集成的内容，将在其他小节中讨论。

3.3.1 树莓派与物联网

先简略了解一下实训课题项目有关的背景知识。

1）有关物联网：

所谓“物联网”（The Internet of things），就是“物物相连”的互联网。基于互联网、传统电信网等信息承载体，是让所有能够被独立寻址的普通物理对象，实现互联互通的网络。

从技术实现的角度看，上述描述有两层意思：第一：物联网的核心和基础仍然是互联网，是在互联网基础上的延伸和扩展的网络；第二：其用户端延伸和扩展到了任何物品与物品之间，进行信息交换和通信。因此，物联网就是这样的“物物相连的互联网”。

物联网的概念是 1999 年由 MIT 的 Auto-ID 中心提出来，当时的希望是将各种信息传感设备，如：射频（RFID）的电子标签、红外感应器、全球定位系统（地理信息）、激光扫描器（二维码）等，与互联网结合成一个巨大的网络，从而为“物体”赋予智能。所以，从实现技术（信息采集、传输、计算和处理、应用等）来看，物联网并没有什么太多的新技术，它只是一种新的应用模式。

2）物联网的整体系统结构：

一般物联网应用系统的网络层次结构如图 3-1 所示：

图 3-1 物联网的网络层次结构

在物联网的最底层是信息采集层，一般由多种传感器组成。信息传输层包括各种有线、无线、工业控制总线等网络。应用实现层包括应用服务器等应用基础设施和应用系统构成。在本实训项目中，这三个层次都会涉及。

3）传感器数据采集层：

在本实训课程中，让同学们分别尝试使用树莓派 GPIO 方式和使用 arduino 两个方式进行数据采集，所采集的信息内容包括：红外、温度、湿度、气压、光敏、气体检测、声音识别、烟雾识别、火焰识别、振动、方向、重力、夜视、加速度、射频、步进电机、陀螺仪、超声波测距等，近 30 多种传感器。

4）树莓派的定位：

很显然，在物联网的整体结构中，树莓派将自己定位在下端的数据采集层。更严格地说：如果没有 arduino 作传感器的前端采集平台的话，树莓派就是最底层的数据采集平台。同时，树莓派还担负着数据采集层与物联网应用服务层之间进行网络信息传输的作用，这是 arduino 所不具有的功能。

5）应用层（基于 Yeelink 或苏宁云）：

作为物联网信息应用的一种方式，实训课程在物联网信息应用环节，作为应用体验，选择了免费的物联网数据应用平台——Yeelink（类似平台还有很多）和苏宁云平台。YeeLink 号称可以把传感器所采集的信息完全释放出来，不需要编写一行代码，无需繁琐的服务器编程技术，就能够将传感器数据通过网络发布出来，并能随时随地的将数据从服务器中取回，通过插件或 APP，向你的朋友或是社会分享。

Yeelink 平台具有简单的应用服务功能，支持双向传输和控制，它不仅能够提供数据的“上行”，还能够实现简单的“下行”，例如：用于对家庭电器的控制。Yeelink 也可以“嵌入”到社交网络上，使数据不再是孤单的节点，存储在 Yeelink 的数据，可以简单的被 API 取回，放置到你的个人博客上，或者根据规则自动转发到您指定的微博上，在这里，您将会感受到数据和人之间的全面融合。

图 3-2 是在 iOS 和 Android 手机运行 YeeLink APP 的结果：

图 3-2 YeeLink 的 iOS 和 Android 手机客户端 APP 的运行结果

图 3-3 是 APP 显示的从 YeeLink 接收到的、实时的室内光线强度传感器采集到的信息。

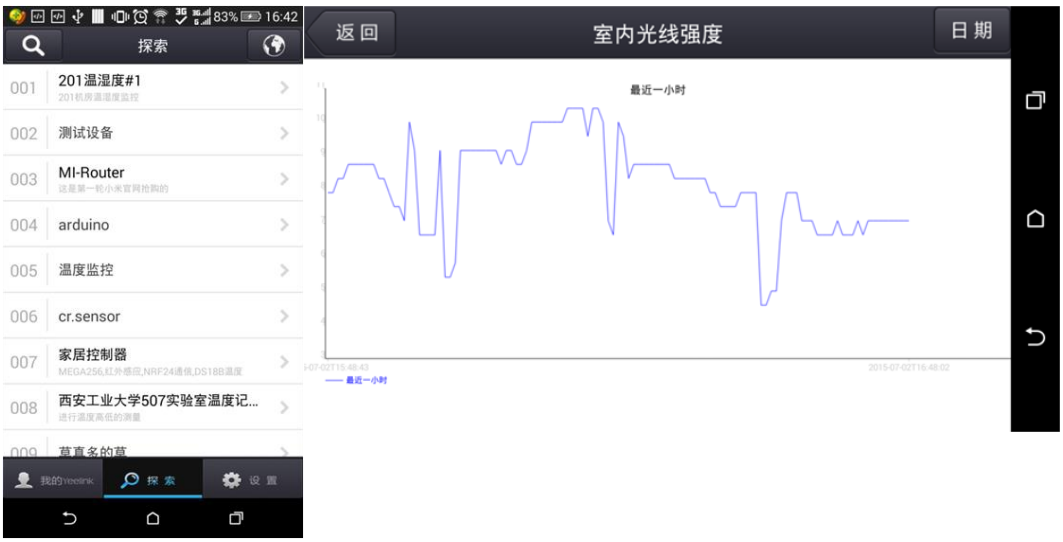

图 3-3 YeeLink 的 iOS 和 Android 手机客户端 APP 显示实时数据曲线的运行结果

参考这些案例，开发自己更“炫”的应用

6）YeeLink 应用服务器的局限与苏宁云：

Yeelink 服务器可以实时保存上传的数据信息，可以按一定方式和规范（由 YeeLink 制定），向用户客户端发布该信息。YeeLink 规定了网络服务器与客户端 APP 之间的通信协议，甚至直接提供 iOS 或 Android 的 APP 包，因此，要想为特定用户订制私人应用，甚至要对上传的数据，进行特定加工处理，是有一定限制的。这也是本次实训课程最后一节，讨论基于苏宁云服务器，开发自己的物联网应用服务器的理由。

3.3.2 树莓派与云计算

“云计算”是一个热门的技术名词，云计算技术也比较复杂。从“IT 基础设施”的虚拟服务器/虚拟桌面技术，到 Web 服务器、应用的交付与使用模式的转变等等。如果把这些都

包括进来，3 个月实训时间都不够。但是，作为物联网信息应用层，还是可以利用“云”，做点什么，感受一下“云”带来的好处。

利用云服务器可以做点什么？

1）云服务器的定位：

在物联网应用架构内，云服务器的定位当然是信息应用层，并扮演着向手机等移动用户提供应用服务的角色。云服务器显然不能用于直接的传感器采集，而更适合作为一般意义上的 WEB 服务器。在本实训课程的最后一节，让同学们体验在苏宁公有云上，搭建自己的应用服务器的过程。

2）树莓派的定位：

由于树莓派自身物理和系统的特点，决定了树莓派是处于物联网的下端（相比 Web 服务器）、小型、便携、移动、廉价等等。从软件系统架构上看，它更像 MVC 模型中的 C（假定传感器采集头是 M、web 服务器及其客户端 APP 为 V，此定义不甚严谨）。那么，基于 C 的应用，可以做什么？

3）一个简单的基于“云服务”的应用：

例如：设计一个个人专有的、订制化的媒体管理器应用。与一般媒体播放器的区别是，这是订制化、可控制管理的播放器。它的播放资源获得与保存、播放服务与控制，都是私人订制的。就好像是需要吃鱼的时候，不是在海里去捞，而是在自己家的鱼池中去抓鱼。海、私人养鱼池、餐桌形成三级架构。显然，很容易理解 Web 服务器、树莓派、播放器在海、养鱼池、餐桌三者之间的对应关系。

海是广大的网络，养鱼池是云服务器；鱼从海里到养鱼池里，是在云服务器上，实现自动扒取、下载、整理、存储。与传统方式比较，如果没有鱼池，要看某部影片的时候，需要临时到海里去捞，是否能捞得到是不一定的；热门的片子，你想看的时候，别人也想看，速度当然就慢。所以，我在白天上班的时候，家里，慢慢给我下，我不着急。

餐桌就是树莓派。等我回家以后，吃完饭、洗完澡。从我自己的云上，“直播”我的“大片”。由于有自家的鱼池，想吃什么吃什么，想什么时候吃，什么时候吃。与传统播放器相比，什么“自动接续播放”、“自动连续剧追踪”，“自动提醒”等等，都是“小菜”啦。

3.3.3 树莓派与大数据

大数据比云计算更热。

1）大数据的特点：

大数据的特点是“4V”，即：体量（Volume）、即非结构化数据的超大规模和增长；多样性（Variety）、即大数据的异构和多样性；价值（Value）、即大量的不相关信息和速度（Velocity）、即实时分析而非批量式分析。

有国外媒体报道：有人用 100 多个树莓派，搭建了一个大数据服务器集群，进行分布计算。这种实验，在娱乐性方面很有趣，但没有什么实际价值。因为大数据的价值在对大量高并发、非结构化的数据，进行并行计算，并找出有价值的数据之间的关系与模型，而不是计算平台（多主机）本身。从这个意义上说，用树莓派搭建大数据服务器，有点“矫情”。

2）树莓派在大数据应用中的作用：

与真正的云服务器集群比较，由树莓派本身作为服务器，构成大数据计算集群的意义和价值不大。那么，由树莓派构成的大数据采集——树莓派提供海量数据来源，具有多样性、分布、实时、并发、廉价等等特点，倒是有价值的。在这种模式下的应用，可以做什么？ 同时，还可以与物联网、云计算结合，这是一个更大的题目。（在网上查：传感器/物联网与大

数据/智慧城市，内容很多），同学们可以慢慢考虑。

3）物联网（树莓派）与大数据的应用案例（智能电网）：

有资料报道：有人用树莓派，对全国重点压缩机、发电机、涡轮机、鼓风机、石油钻采设备、传送带、内燃机车和医疗成像扫描仪等高能耗设备，安装传感器，进行采集。嵌入式传感器在这些机器和设备中，利用物联网来传输度量为震动、温度、湿度、风速、位置、燃料消耗、辐射水平的这些数据；检测中心利用这些设备的实时大数据分析，找到最佳的供电、供热、运输等能量转换方案，实现能量转换资源配置的最佳化。

这个意义的例子，给我们提供了一个利用树莓派，进行大数据收集、分析、应用的“参考概念模型”。

4）一个物联网（树莓派）与大数据应用的参赛项目构想（智能交通类）：

参赛创意：基于“集装箱”运输模式的未来“公交”运行模式。

愿景描述：过去海上货物运输大多采用“散装”模式，集装箱出现以后，实现了所谓“门（客户）到门（客户）”的运输，极大地提高了货物运输的效率，降低了运输成本。集装箱化也导致货运船舶的改变，散装货轮变成了集装箱货轮。那么，类似“散装”的公交车，是否能够集装箱化呢？目前，也出现了早期的集装箱式公交，规模最小的是“滴滴打车”，但成本太高，不宜推广。再大一点就是“订制公交”：个人预约线路和时间，公交不再接收其他客人，也不停中间站台。但是，由于预定客人少，这种模式规模做不大，难于达到效益最大。如果所有公交，都是“私人订制”的，类似集装箱模式，取消公交车的“线路”限制，完全根据“需求”组织运行和调度，情况如何？

实现设想：由网上获得所有乘客、车辆运行、道路、甚至天气的信息，进行大数据计算，并进行调度，规划最佳的运行效果。如：车辆状态，包括：位置、行驶路线、载客情况（单车）；请求状态：去向、人数；调配状态：城市需要与供给的总体平衡、要求达到最优化配置和运行调度。

创意：可以取消公交线路、站点的旧运行模式，如集装箱号和船期一样，按客户和运行需要，组装成一个具有相同始点和终点的“集装箱流”，与某“船期”配载，在规定的开始和到达时间内，接受并被送达目的地。乘客只要在规定时间和地点，上了预定的“集装箱”，就可以保证准时到达要去的地方。根据乘客需求和地点，这个集装箱有可能是一艘“巨轮”，也可能就是载一个人的出租车，视“流量”情况而定。

由此，全市的交通资源和交通需求，在这个平台上得到了统一调配。集装箱的集散——挂载、解挂、行驶路线、小车化、专车化等等，都是“效率”最佳的。

软件创意大赛并不是开发一个真正能够实用的产品，但是，一定需要一点“超前”的创意。上面的设想，是否有点意思。

具体怎么实现呢？

在这个模式中，树莓派承担了“公交车”状态采集的作用，采集的信息包括：行驶位置、去向、速度，上车或下车的乘客情况。在参赛“模拟”小车上，上述信息的采集是可以“模拟”实现的。

客户端是手机 APP：包括乘车请求、时间和地点预订、与管理中心的交互等；

云服务器的功能：接收乘客请求、接收车辆、接收路况/天气/交通指挥等等要素；然后进行数据分析、关联调度协调，采用神经网络、线性规划等等技术。

参赛情景与用户体验效果：将用户的请求、运行的状况都显示在模拟的调度屏上，实时观察请求与资源的配置情况。模拟突发事件，例如：大型运动会散场、局部暴雨、交通事故等，观察是否能够达到系统的最优配置。 在正常情景下，展现不同时段的资源利用率的平稳变化情况（类似 CPU/内存），但突发事件出现时，资源曲线没有发生很大的变化，没有发生拥塞。在原有模式下：公交车需要 3-5 小时，才能消化拥堵的人流，而非突发事件时，车

辆乘员有很大的冗余。在新模式下：当突发事件发生时，系统自动汇集大量车辆（集装箱），很快即消化人流，平时也无冗余。

项目的新颖性：以前有没有人做过类似的课题？前已分析：“滴滴打车”只能解决出租车这样的特定的、个体的资源调配问题，不是公交，这样的大众化的交通工具。手机查公交更是只是单向的为个人提供了乘车信息。本课题的创新关键是：每个公交车辆的运行，过去只受驾驶该车的司机、预订的线路、当时的路况所支配，地铁也没有摆脱“线路”的束缚。奥体没比赛，机场没航班，奥体或机场的地铁就没人乘，大流量时，地铁也拥塞。资源只是有限利用。

本课题提供了公共交通资源在大数据统一调度的大格局下的统一支配，也带来了车辆、路线、乘坐方式的大变革（散装变集装），如果用上谷歌的自动无人驾驶，将是一种什么局面？效率极大提高，资源极大利用，投资比地铁省很多。

参赛课题的关键：把物联网的“采集”、用户的 APP、云计算结合起来。当然，这个构想要真的实现，还有很多问题。一定有人会说，老太太不会用手机，难道就不能出门上车了吗？科技的进步不会使每一个人都满意，这不是本课题考虑的问题。 同学们现在要考虑的问题是：看到了创新的可能，更看到自己的不足——怎么把这个“点子”变成大赛评委面前的展品！

5）回到课程中来：

3 年后，带着这个构想参赛，现在要做什么？从什么地方开始？现在的任务，就是先把本次实训的任务完成，通过这次实训，选择平台、了解工具、知道现有平台和工具能做什么？还有哪些做不到。同时，站在现有的平台和技术基础之上，了解外面的世界（更多的问题和创意空间），了解自己（自己还不具备的更多的基础知识、能力、经验）。先为自己定一个模糊的目标（现在暂时不需要清晰），带着目标——上好课、自习、参加项目训练。 这就是本次实训课程希望带给你的收获！

3.4 应用平台选择与体验

平台指开发平台与运行平台。一般两种是相同的，区别是运行平台是真实的网络、主机和存储、用户的环境，而开发平台只是这个环境的“模拟”。平台还指由主机、OS、支撑软件等所组成的、提供给应用系统运行的环境，包括：各子系统之间的连接、交互。

3.4.1 物联网应用系统的逻辑与物理架构

在讨论一个系统架构的时候，一般有五个架构视角（详见《软件架构设计实践教程》第三章软件架构的描述与可视化），即：物理、开发、逻辑、运行和数据的架构。图 3.4 是物联网应用系统的逻辑与物理架构视图，它说明，物联网一般由信息采集、信息传输和信息应用三个逻辑层次组成。这三个层次的划分，首先会被“落实”到具体的物理架构上。采集、传输和应用被“物理”地分解到三个不同物理设备上。有时，这个“落实”可能仅仅是逻辑意义，而不是三个具体的物理设备，例如：开发的时候，开发环境的三个逻辑层次，同在一个物理设备上。

这三个层次的划分，是逻辑概念的划分，具体承载相应逻辑功能的设备，会由具体实现方案的不同而不同。例如：信息应用层可能就是一台 PC 服务器，也可能是租用电信的服务器（空间），也可能是自己的服务器集群，完全看需要和可能。在本阶段实训的最后两小节课程中，使用了 YeeLink 的服务器以及苏宁的公有云服务器。使用者（我们）并不知道这两个服务器的型号、CPU 和内存大小、甚至放在什么地方等，这些我们都不关心。当然，如果是真的应用系统，即使租用别人的服务器，也是要考虑的，因为要考虑服务器的性能是否能

够满足用户实际运行的需要，苏宁云估计不会无限制的帮你满足这个要求。

图 3-4 是本章实训项目——物联网应用系统的、总体的逻辑与物理架构图。

图 3-4 物联网应用系统的逻辑与物理架构图

3.4.2 物联网应用系统的逻辑和运行架构

物联网应用系统有什么样的物理架构，某种意义上说，是“刚性的”、受各种物理条件限制不由我们选择的，这个无需多说。而应用系统的逻辑架构和运行架构，则可以根据用户的需要、开发的需要，做出选择。而本小节所讨论的平台选择，就是有关架构方案的选择和比较。

有关选择情况如下：

1）云计算服务器平台的选择：

在本次实训课程中，采用了两种云服务器：YeeLink 和苏宁云服务器。前者是一个部分开放的数据存储和下载应用型服务器，不提供用户（指开发者）上传数据后，在服务器上运行自己开发应用（自行处理自己的数据）的功能，因此，YeeLink 是 “定制（由 YeeLink）”应用型的服务器。而后者可以根据租用者的需要，提供一个“干净”的 linux 系统+网络的环境。至于你在这个 linux 系统环境下“干什么”，它是不管的。本次实训根据这两个服务器的特点，分别实现了两个信息层应用，体验各自的不同。

2）传输方式的选择：

传输方式主要指网络信息的传输方式，这在现在来说，已经很简单，没有什么“选择”的余地。只要能上网，什么方式都可以，包括有线或无线。

3）采集信息平台的集成：

信息采集平台主要通过收集采集设备（传感器）传来的信息，经处理或不处理，上传到网络服务器上。由于上层的物联网服务器的信息传输方式和协议不同，这个平台与服务器的信息传输方式也有不同，这在以下具体实现时介绍。

承担这个任务的平台是树莓派，当然也可以选择 PC、笔记本电脑或其他设备。选择树莓派的目的，是想让同学们直观地接触一下“硬件”，使得实训更有趣一点。树莓派的强大功能、廉价性、便携性等优点，也是未来做项目时的一大优势。

4）采集设备（传感器）的选择：

项目罗列了 30 多种功能不同的传感器，各有所用。在后面的课程中再介绍。

本小节的平台选择看起来有点“矫情”，实际上并没有选择什么，在真正做项目的时候，不论最终是否无可选择，这个环节还是不能缺省的，因为平台本身对于项目的限制是决定性的（局限）。

3.4.3 配置树莓派

从这里开始，是本实训课程动手的第一课。首先，安装树莓派的基本系统，配置好必要的参数，点亮树莓派。

1） 树莓派启动方式及支持的 OS 系统：

由于树莓派开发板没有配置板载 FLASH，它只支持 SD 卡启动，所有需要下载相应 OS 镜像，并将其烧写在 SD 上，然后启动系统即可（这个镜像里包含了通常所说的 linux 的 bootloader、kernel、文件系统等）。树莓派由于其开源特性，支持非常多的 OS 系统类型，包括：RaspbianOS（debian）、RISC OS、Arch Linux、XBMC、RASPBMC、OpenELEC、PIDORA（Fedora linux）、FreeBSD，甚至包括 Android。最近微软出了 Windows10，也可以在树莓派上跑，但 windows 系统与 Linux 相比，还是过于庞大，即使再加以裁剪，估计运行速度也好不到那里去。同学们可以试试看！

树莓派支持的各种 OS 系统的主要区别是：Raspbian 是单纯的 Arm 版的 Linux 系统，是基于 Debian 的；Pidora 也是单纯的 Arm 版的 Linux 系统，是基于 Fedora 的；Arch Linux ARM 是单纯的 Arm 版的 Linux 系统，是基于 Arch Linux 的。Raspbmc 是在 Raspbian 基础上定制的 XBMC 影音播放系统，XBian 也是一个 XBMC 系统。OpenELEC 是用得最多的 XBMC 跨平台分支，也有树莓派版本。

熟悉 Linux 系统的可以用 Raspbian，单纯用作多媒体播放器的可以用 Openelec。本次实训也会在这两个系统上跑一下，两个都很优秀。基本上上述两类系统是 OS 的代表。前者用来进行 Linux 日常操作，后者主要用做媒体播放。

有关系统软件的下载地址，用百度一下就可以知道了。下载后，文件放在一个没有中文名的文件夹下，作者用的是 2014-06-20-wheezy-raspbian.img。

2）制作 SD：

制作镜像 SD 的工具可以用 DiskImager.zip，下载后解压、运行界面如图 3-5。

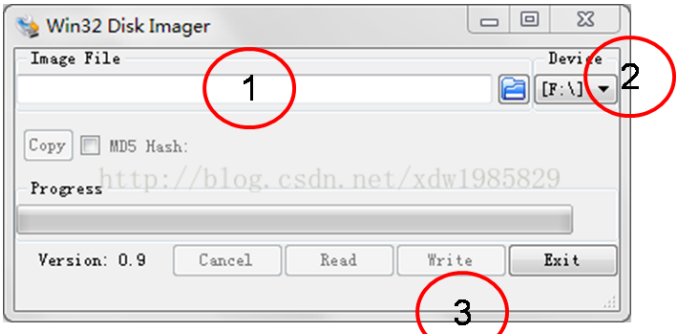

图 3-5 制作镜像 SD 的工具 DiskImager

准备一张 2G 以上的 SD 卡及读卡器，最好是高速卡，现在一般都是 Class10 的卡了，卡的速度直接影响树莓派的运行速度。在本实训课程中，只要有 4G 就足够了，当然大一点更好，因为所有的系统运行空间，都在这张 SD 卡上。

将 SD 卡使用读卡器（或卡套）连上电脑，运行解压后的 DiskImager.exe。在软件①中选择下载的树莓派系统 img 文件，在②处“Device”下选择 SD 的盘符，然后在③处选择“Write”。然后程序就开始烧写系统了，根据 SD 卡接口的速度，烧写过程有快有慢。烧写结束后会弹出完成对话框，说明烧写已经完成了，如果不成功，请关闭防火墙一类的软件，重新插入 SD 进行安装。烧写完成后，在 win 系统下看到 SD 卡只有 74MB 了，这是正常现象，因为 linux 下的分区内容，win 下是看不到的！

3）在 windows 下修改 config 文件：

在 windows 下进入 SD 卡，找到 config.txt 文件，使用“写字板”（注意：使用“记事本”工具时，看到的信息可能是不分行的）工具打开，修改以下几处：
依次将：

```
# hdmi_force_hotplug=1
# hdmi_group=1
# hdmi_mode=1
# hdmi_drive=2
# config_hdmi_boost=4
# sdtv_mode=2
# arm_freq=800
```

几行前面的“#”去掉。并且将 hdmi_group=1 改为 2，将 hdmi_mode=1 改为 18，保存修改后的文件后退出。 如果不做以上修改，树莓派连接 VGA 后，将出现黑屏。

在 PC 上修改完后，取出 SD 卡，插入到树莓派的 SD 卡槽中。外接好树莓派的 HDMI 或者 AV 显示器、USB 鼠标、键盘、无线网卡等相关配件。检查无误后，打开树莓派的电源。

树莓派的启动过程会显示在 HDMI 或者 AV 显示器上。第一次启动可能需要稍微花一点时间，第一次接触 linux 的同学也会发现，linux 的启动与 windows 的启动是不一样的，它把所有的启动信息，都呈现给你看，而且显示的很多、很快，你甚至都来不及看。

4）执行树莓派系统的配置程序：

树莓派第一次启动的时候，会自动进入如图 3-6 的配置程序界面，在命令行模式下输入 $sudo raspi-config，也可以回到这个界面。新旧版本的配置界面可能不太一样，下面介绍 2014-06-20-wheezy-raspbian.img 的配置程序操作。

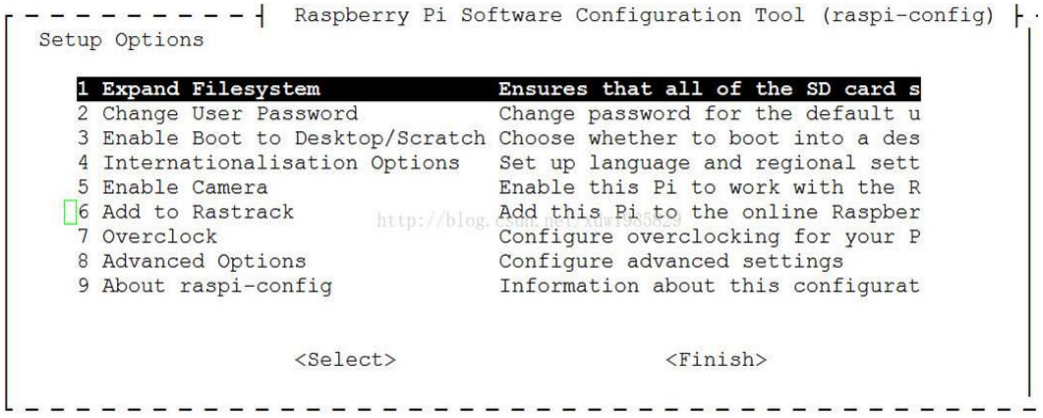

图 3-6 2014-06-20-wheezy-raspbian 的配置程序界面

因为现在是在 linux 的命令行模式下运行，因此，鼠标暂时不起作用。用键盘上的上下左右键移动光标，用 Tab 键操作确认键。

首先看选项 1，这是唯一必选操作。（扩展根分区：将根分区扩展到整张 SD 卡，树莓派默认不使用 SD 卡的全部空间）此步操作很简单，一路回车即可。

以下依次介绍各选项的设置与修改：

2、Change User Password：改变默认用户名 pi 的用户密码，按回车后输入 pi 用户的新密码。默认的用户是 pi，密码为 raspberry。 系统 root 用户的使用，在后面介绍。

3、Enable Boot to Desktop/Scratch：选择系统启动时，是直接进入字符控制台模式，还是进入图形桌面环境。即是 Console Text console 还是 requiring login（default），默认项为 LXDE 图形界面。 另一个是：Scratch Start the Scratch programming environment upon boot 启动时进入 Scratch 编程环境。不熟悉 Linux 的同学可以先选择进入默认的图形界面（类似 windows）。

4、Internationalisation Options：国际化选项，可以更改系统的默认语言。

1）Change Locale：语言和区域设置，默认为英文。树莓派默认是英文界面，但系统还是预装了中文字库，所以如果选择中文，也不需要再手动安装中文字体（实际已自带）。如果没有字库，请参考相关资料自行安装。

1、在命令行输入以下命令：

$sudo apt-get install ttf-wqy-zenhei 安装字库（系统已自带，可不安装），安装过程中如果碰到(Y/n)，都选择 y；

2、安装中文拼音输入法：

$sudo apt-get install scim-pinyin 一般系统已安装中文字库，可不必再安装；

$sudo raspi-config 此步一定要做，否则不出中文。

3、选择 change_locale：

在 Default locale for the system environment 中，选择删去 en_GB.UTF-8 UTF-8 前的*（使用空格键）；找到 en_US.UTF-8 UTF-8，用空格键在它前面添加* ；找到 zh_CN.UTF-8 UTF-8，用空格键在它前面添加* ；在 Default locale for the system environment:中，选择 zh_CN.UTF-8，选择 OK，退出。

4、配置完成之后，输入命令 sudo reboot 重启系统：

重启完成后，就可以在显示屏上看到中文显示和中文输入法了，切换中文输入法一样也是 ctrl+space

2）Change Timezone：设置时区，如果不进行设置，PI 的时间就显示不正常。在获取实时时间进行显示的时候，一定要把这里设置一下。选择 Asia（亚洲）再选择 Chongqing（重庆）即可。3）Change Keyboard Layout：选择合适的键盘类型和布局，有很多键盘类型，如果类型不匹配，键盘上的有些键与实际输入值不符，那是很难受的事情。

5、Enable Camera：启动 PI 的摄像头模块，如果想启用，选择 Enable，禁用选择 Disable 就行了。

6、Add to Rastrack：把你的 PI 的地理位置添加到一个全世界开启此选项的地图，建议还是不要开了，免得被跟踪。

7、Overclock：超频（不要玩超频，烧掉系统你自己买单）。None：不超频，运行在 700Mhz，核心频率 250Mhz，内存频率 400Mhz，不增加电压 。Modest：适度超频，运行在 800Mhz，核心频率 250Mhz，内存频率 400Mhz，不增加电压。Medium：中度超频，运行在 900Mhz，核心频率 250Mhz，内存频率 450Mhz，增加电压 2。High：高度超频，运行在 950Mhz，核心频率 250Mhz，内存频率 450Mhz，增加电压 6。Turbo：终极超频，运行在 1000Mhz，核心频率 500Mhz，内存频率 600Mhz，增加电压 6。

8、Advanced Options 高级设置：

A1 Overscan：是否让屏幕内容全屏显示，当然要；

A2 Hostname：在网上邻居或者路由器能看到的主机名，可以让别人看到；

A3 Memory Split：内存分配，选择给 GPU 多少内存，就用缺省值；要改以后再说；

A4 SSH：是否运行 SSH 登录，要开启，以后还是用 SSH 登录比较方便。

A5 SPI：是否默认启动 SPI 内核驱动，暂时不用管。

A6 Audio：选择声音默认输出到模拟口还是 HDMI 口，一般还是用模拟口，因为不是用的 HDMI 设备；可选项是：

- ✓ 0 Auto：自动选择；
- ✓ 1 Force 3.5mm ('headphone') jack：强制输出到 3.5mm 模拟口；
- ✓ 2 Force HDMI： 强制输出到 HDMI；

A7 Update：把 raspi-config 这个工具自动升级到最新版本，可以不做，后面会介绍系统

总升级；

9、About raspi-config 关于 raspi-config 的信息。

树莓派一般可以自动检测目标设备的分辨率。但对于 HDMI-VGA 转换器，分辨率可能会明显不对，这时需要给树莓派指定一个分辨率。修改树莓派的显示分辨率，需更改树莓派 FAT32 分区里的 config.txt，修改的方法是将 SD 卡拿到 PC 上直接修改，然后再插回树莓派。在树莓派上也可以修改，但要先修改文件的读写权限等，比较麻烦。前述在烧制 SD 卡之后，就修改了 config.txt（将 hdmi_mode=1 改为 18）。

其他对 config.txt 的修改含义如下：

- ✓ hdmi_group= 2 显示器参数分组，常用电脑显示器为 2；
- ✓ hdmi_mode=18 分辨率，具体见下表：
  - ■ hdmi_mode=16 1024x768 60Hz；
  - ■ hdmi_mode=17 1024x768 70Hz；
  - ■ hdmi_mode=18 1024x768 75Hz；
  - ■ hdmi_mode=19 1024x768 85Hz；
- ✓ hdmi_ignore_edid=0xa5000080 命令树莓派不检测 HDMI 设备的任何信息，只按照指定的分辨率输出。

如果上述参数不加，树莓派可能会“自作聪明”地检测 HDMI 设备的分辨率，造成设置的分辨率无效。显示器仍然会出现系统画面，但会有一个提示显示模式不支持的漂浮提示。配置完成后确认会重启系统，选项即可生效。

5）其他主要事项：

有几个重要的东西，首先要告诉你：

重启系统的命令是：reboot。

系统默认用户是 pi 密码为 raspberry，当需要 root 权限时，通常由默认账户经由 sudo 执行。所以，大家最好习惯使用 sudo 的方式，来执行 root 权限的命令，而不是自己充当 root。因为与 windows 系统不同，在 Linux 系统下，root 只有系统管理员才可以用，一般人是不可以的，要熟悉这样的系统使用和管理模式。系统的 root 账户默认没有密码，但账户锁定，重新开启 root 账号，可由 pi 用户登录后，在命令行下执行：

sudo passwd root

此时要求输入 root 的密码，再输入一次，即可。

执行一次 apt-get update，对系统所有软件，进行一次更新。如果没有最新版本，系统会告诉你已经是最新版本了。然后什么事也不做。

3.4.3 实现树莓派的远程登录

所谓系统的基本功能，是指当安装完树莓派自带的系统软件（操作系统+缺省应用系统），并完成基本设置（网络、中文等）之后，可以实现的系统功能。由于篇幅所限，有关基本概念部分，只点到为止，不做详细介绍。

1） 有线局域网：

现在的世界第一是不能没有电、第二就是不能没有网。联网的方式包括：有线和无线。

有线联网：树莓派系统缺省是支持有线网络的，但是，系统的缺省设置是动态 IP，即 HDCP 模式，但是在学校，却是不支持 HDCP，每个学生的机器都有自己固定的 IP，所以，树莓派要修改为固定 IP。可能有同学没有学过 linux，不会 vi 命令，也不知道 linux 的文件系统结构，暂时先根据老师的要求模仿，以后再补这方面的内容吧。

修改的方法如图 3-7（如果是动态联接，无需修改），命令是：

$sudo vi /etc/network/interfaces

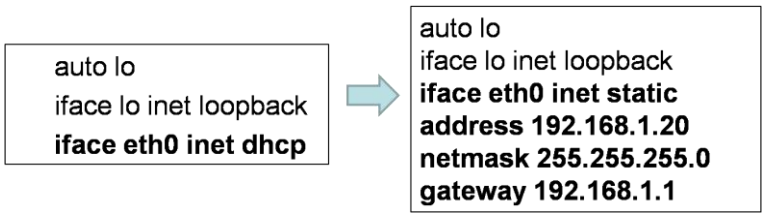

图 3-7 将树莓派的动态 IP 修改为固定 IP

即将/etc/network/interfaces 文件内容的：

iface eth0 inet dhcp 改为：iface eth0 inet dhcp；

并添加如下的内容：

address 192.168.1.20 定义一个自己的树莓派静态地址，这个地址是你可以用的 IP；

netmask 255.255.255.0 添加掩码；

gateway 192.168.1.1 添加树莓派连接的路由器地址。

同时，删除下面这一行：

iface default inet dhcp。

修改完成之后需要重启。重启系统以后，在图形界面上，点击图 3-8 左边的图标，就可以看到 WiFi Config 的界面了（图 3-8 右）。其中显示了树莓派实际联网的 WiFi 地址，被改为固定的 192.168.1.20，而不是早先自动获得的某个 IP 地址。此处的修改，相当于设置路由器 lan 口 IP，即访问路由器通常使用的默认网关 192.168.1.1。当然，记住：以后远程连接树莓派时，都要使用这个地址了。

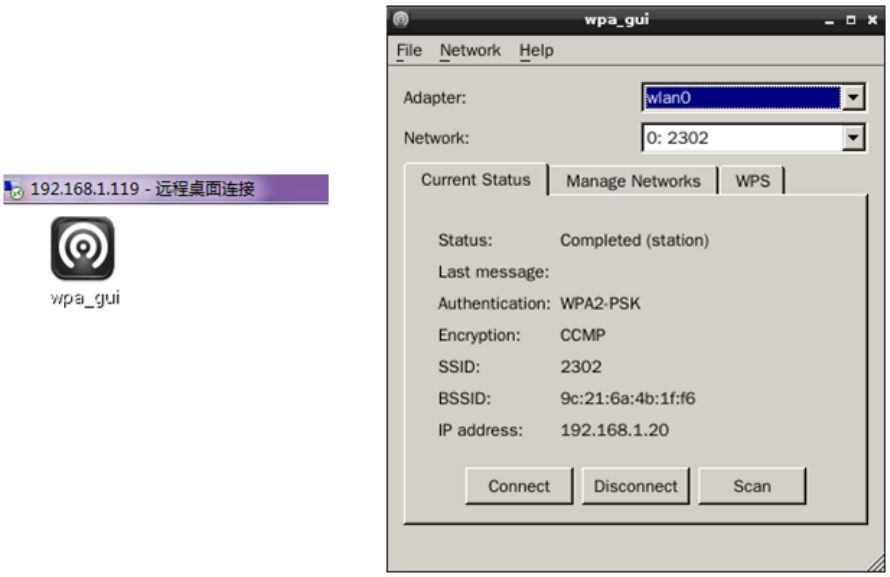

图 3-8 树莓派网络 config 图形界面看到的固定 IP 地址

使用固定 IP 的一个意外好处是以后使用远程终端登录树莓派的时候，IP 地址是固定不变的。

2）无线局域网网络：

树莓派缺省是支持 DHCP 无线上网，无需做什么修改的。直接运行图 3-8 的 IP 管理界面，扫描“热点“并输入密码等，确定并联接即可。如果是需要固定无线 IP，也像有线一样，进行修改，只是所针对的设备不同而已。如果找不到无线网卡，则需要查看无线网卡的设备、驱动等是否正确，不再详细介绍。

将树莓派做成一个“热点”供别人使用当然是可以的，但是，设置有点麻烦。

3）使用 SecureCRT 命令行远程终端：

支持 SSH 的远程终端软件很多，主要有命令行模式和图形模式。命令行模式的有：PuTTY、xshell4.0、SecureCRT（6.5.0）等，就命令行终端功能而言，使用上差别不大，主要的差别是一些辅助功能。这里主要以如图 3-10 的 SecureCRT（6.5.0）为例，进行介绍。

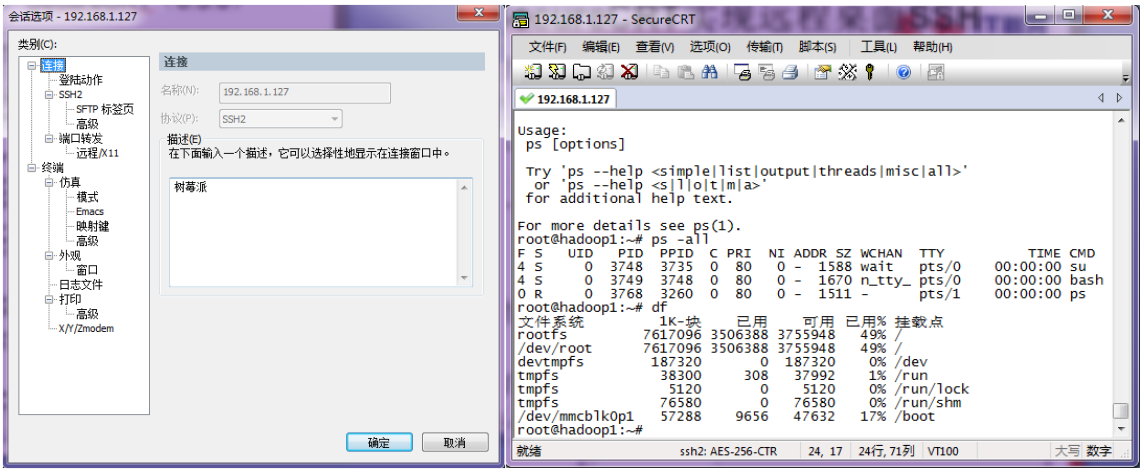

图 3-10 SecureCRT 的参数选择和登录后的界面

选择 SecureCRT 的理由是它可以直接在 windows 和 Linux 之间，通过图形界面操作，传输文件，而不需要在命令行上输入命令。

在 SecureCRT 的 host 选项中填入树莓派 IP，输入远程登录的用户名、密码，如默认的：pi/raspberry，选择使用 SSH2 协议。在外观选项中，选择字符编码为 UTF-8，以支持中文。然后选择“连接”。

4）使用 windows 的远程桌面软件：

通过 windows 自带的远程桌面软件，也可以远程访问树莓派桌面应用，并且还是图形桌面系统。在 PC 上运行系统附件目录下的“远程桌面连接程序”，在如图 3-11 左的界面上输入树莓派的 IP 地址，稍等片刻，在图 3-11 右的界面上，输入用户名和密码。

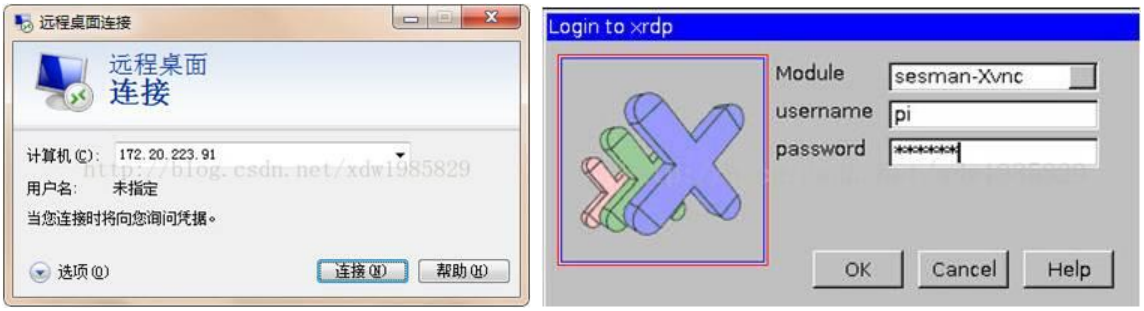

图 3-11 windows 远程桌面登录界面

登录验证成功后，就出现了如图 3-12 左的树莓派图形桌面。

与其他客户端工具不同，这个软件是可以同时实现图形和命令行仿真的客户端。点开桌面上的 LX 终端图标，出现的就是命令行客户端。

有同学提问：为什么我的桌面上，除了一个树莓，什么都没有？这是因为树莓派的桌面，跟 Windows 一样，也是根据每个用户的情况不同，呈现出不同的桌面的。在树莓派桌面的左下角，点击类似 windows 的“开始”按钮，在弹出的菜单中（图 3-12 右），选择自己想放到桌面（在树莓派中，桌面被称为“菜单”）的应用程序，右键选择“添加到菜单”。看看，有什么了？ 当然你也可以把它从“菜单”中删掉。

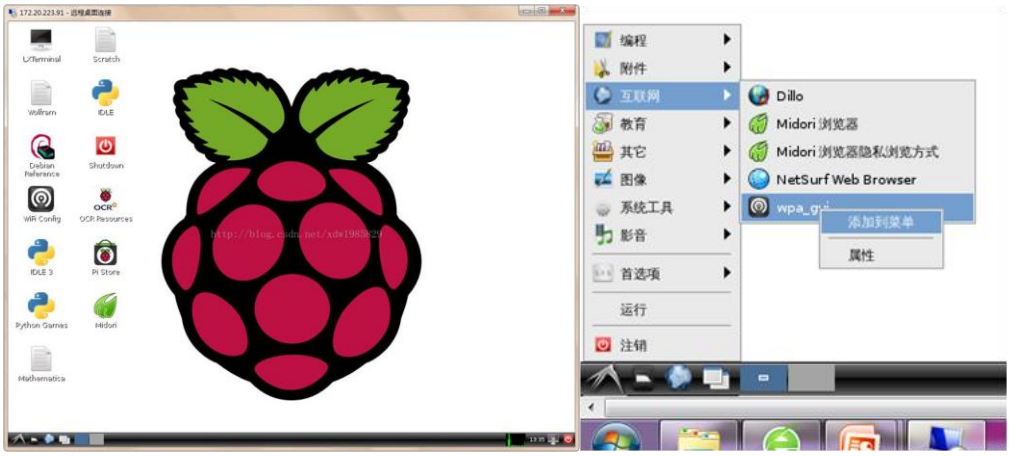

图 3-12 远程登录后看到的树莓派界面

进入树莓派命令行模式下，就可以像在一般的 linux 系统命令行一样操作了，而图形终端则像 windows 桌面上一样，但是 linux 的图形桌面与 windows 相比，“丑”很多，大家不要奇怪。另一个明显的感觉就是：用 windows 的远程终端登录树莓派后的操作，要慢很多。

5）赶快修改你的缺省密码：

现在可以远程登录树莓派了，不但可以远程登录自己的树莓派，也可以登录别人的树莓派！只要输入别人的 IP。在机房里，同学之间的 IP 相互都是知道的。对那些懒得修改密码的同学，管他谁，登上去试试。开始有同学不安分了——帮人家“关机”还算是“厚道”的，偷偷改了人家密码，害的人家只能重新烧 SD，当一回“黑客”。

赶快换密码吧！这是在 windows 下，从来没有碰到过的情况。

6）抛掉树莓派的键盘鼠标：

可以远程登录以后，原则上，是否就不再需要树莓派上的键盘、鼠标了。确实如此。前提是每次远程登录树莓派的 IP 是事先知道的。

3.4.4 让树莓派成为 samba 文件服务器

1）树莓派 samba 文件服务器：

通过 samba，可以将树莓派设置为一个文件服务器，如果家里有一堆不再用的旧硬盘，收集起来，再安装一个 hadoop 软件，就可以做成一个很“廉价（200 元以下）”、移动的“硬盘库”，供手机、IPAD 使用，还是很有适用价值的。

a) 安装 samba：

$sudo apt-get install samba

$sudo apt-get install samba samba-common-bin

b) 修改配置文件：

$sudo vi /etc/samba/smb.conf

下面的配置是让用户可以访问自己的 home 目录：

I) 开启用户认证，找到“##### Authentication #####”，将“# security = user ”的#号去掉（设置安全级别为 user 级/最低）；

ii)配置用户可以读写自己的 home 目录，在“[homes]”节中，把 “read only = yes”改为“read only = no”；

iii)找到 browseable=no 改为 yes 否则等会访问时此文件夹会隐藏。

c)重启 samba 服务：

sudo /etc/init.d/samba restart

d)把系统默认用户 pi 添加到 samba ：

sudo smbpasswd -a pi 或者 sudo smbpasswd -a <新用户名>，会提示你输入密码。

e)windows 访问：

在 windows 上，任意打开一个文件浏览器，输入 ip 地址，就可以从 PC 上访问树莓派 home 目录（图 3-13 左）。图 3-13 右图是在 PC 上看到的共享目录内容。这个共享目录在树莓派的什么地方？缺省位置在/tmp 目录下。

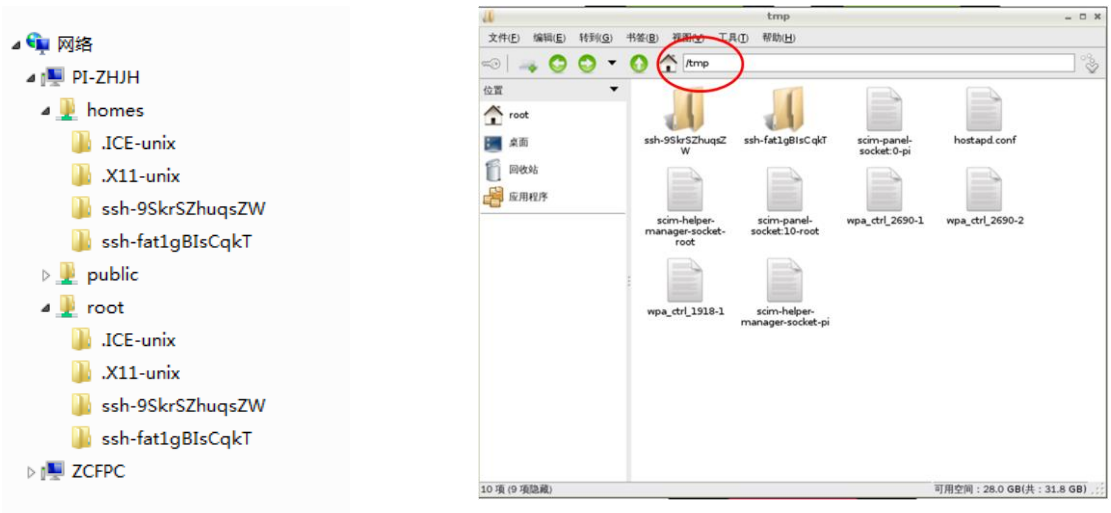

图 3-13 在 windows 的资源管理器或 IE 上可以看到树莓派的共享文件夹

上述方法是开放了树莓派某一个用户名下的共享目录，还可以配置一个公共文件区，任何用户都可以访问，命令如下：

$sudo mkdir /home/shares

$sudo mkdir /home/shares/public

$sudo chown -R root:users /home/shares/public

$sudo chmod -R ug=rwx,o=rx /home/shares/public

然后编辑 /etc/samba/smb.conf 文件，在文件的末尾添加：

[public]

comment = Public Storage

path = /home/shares/public 共享目录位置

valid users = public 用户可在网上看见的共享名

create mask = 0660 用户建立新文件的默认权限

directory mask = 0771 用户建立新目录的默认权限

read only = no

browseable = yes 用户可浏览

writable = yes 用户可写

保存文件，并重启 samba 服务。

2）挂载 U 盘：

目前的系统版本是自动挂载 U 盘的，不需要再行设置。如果不能使用 U 盘（U 盘插入 USB 接口后不能识别），可参照以下介绍，进行挂载。

通常 U 盘挂载在 /mnt 或 /media 目录下。为此，新建一个目录作为挂载点，比如：

$sudo mkdir /mnt/udisk

a) 手动挂载：

挂载命令：

$sudo mount -o uid=pi,gid=pi /dev/sda1 /mnt/udisk

挂载后，可以看到/mnt 下多了一个 udisk 设备，如图 3-14 左图：

图 3-14 挂载 U 盘

直接访问/dev/mnt/udisk 就可以看见 U 盘上的内容，如图 3-14 右图。

注意：sda1 是取决于你的实际情况，a 表示第一个硬盘，1 表示第一个分区。FAT 格式 U 盘 mount 本身就能支持，但如果你的 U 盘或移动硬盘使用的是 exFAT 格式，mount 会说不支持。没关系，安装 exfat-fuse 软件之后 mount 就支持了。命令是：

$sudo apt-get install exfat-fuse

U 盘用完之后的卸载命令是：

$sudo umount /mnt/1GB_USB_flash

b) 开机挂载：

如果想开机自动挂载，而不是每次手工执行，可以编辑文件：/etc/fstab，在其末尾添加一行：/dev/sda1 /mnt/udisk vfat rw,defaults 0 0，这样每次开机就会自动挂载。

c) 热插挂载:

需要像 windows 一样，U 盘插上时，能够自动识别并自动挂载在某一目录下，拔下自动 umount，请执行以下操作：

$sudo vi /etc/udev/rules.d/10-usbstorage.rules（此文件默认没有，需要新建），赋值以下内容即可，树莓派就会自动在/mnt/udisk 目录下挂载 U 盘。

```
KERNEL!="sd*", GOTO="media_by_label_auto_mount_end"
SUBSYSTEM!="block",GOTO="media_by_label_auto_mount_end"
IMPORT{program}="/sbin/blkid -o udev -p %N"
ENV{ID_FS_TYPE}=="", GOTO="media_by_label_auto_mount_end"
ENV{ID_FS_LABEL}!="", ENV{dir_name}="%E{ID_FS_LABEL}"
ENV{ID_FS_LABEL}=="", ENV{dir_name}="Untitled-%k"
ACTION=="add", ENV{mount_options}="relatime,sync"
ACTION=="add", ENV{ID_FS_TYPE}=="vfat", ENV{mount_options}="iocharset=utf8,
umaskk
=000"
ACTION=="add", ENV{ID_FS_TYPE}=="ntfs", ENV{mount_options}="iocharset=utf8,
umaskk
=000"
ACTION=="add", RUN+="/bin/mkdir -p /mnt/udisk/", RUN+="/bin/mount -
o $env{mount__
options} /dev/%k /mnt/udisk/"
ACTION=="remove", ENV{dir_name}!="", RUN+="/bin/umount -
l /mnt/udisk/}", RUN+="//
bin/rmdir /mnt/udisk/"
LABEL="media_by_label_auto_mount_end"
```

3）为树莓派安装 Office 软件：

选择 LibreOffice（还有很多类似软件），使用 get-apt install libreoffice 命令自动安装，装

完后，看一下装到哪里去了。执行：Which libreoffice，显示结果：/usr/bin/libreoffice。安装完之后，在工具菜单上，就可以看见几个 libreoffice 工具。注意：如果在要在远程客户端使用此 LibreOffice 软件，只能使用 windows 远程桌面软件（图形终端），不能使用其他命令行终端软件。因为这些办公软件都是基于图形界面的。

Windows 下的 office 软件在 libreoffice 下基本上都是兼容的，可以自动打开（图 3-15），但界面和使用方式差别较大。在树莓派上运行 libreoffice 的另一个感觉就是——慢！这时才感觉到树莓派的“小”了。

图 3-15 树莓派上安装的 libreoffice 界面

3.4.5 让树莓派成为 Web 服务器

常见的 Web 服务器有 apache 和 Tomcat，后者更常用一些，具体区别不再介绍，请参阅相关资料。

1） 配置树莓派的 java 环境：

本系统版本是缺省安装了 java7 的，如果需要升级，可下载 Java 8 for 树莓派，下载地址：http://jdk8.java.net/fxarmpreview/。可在树莓派上下载，也可在你的 windows 笔记本上下载。下载 jdk-8-ea-b36e-linux-arm-hflt-29_nov_2012.tar.gz 完成后，上传到安装的目录下。习惯上，自己下载安装的软件，都放到/usr/local/下。

对下载的压缩文件执行解压缩，然后运行安装程序：

#tar zxvf jdk-8-ea-b36e-linux-arm-hflt-29_nov_2012.tar.gz

注意：下载 java8 后，不要忘了修改环境变量，否则，环境变量还是系统原来的 java7 的。需要修改的内容如下：

#vi /etc/profile

JAVA_HOME=/usr/local/jdk1.8.0（这是你的安装目录）;

CLASSPATH=.JAVA_HOME/lib/tools.jarJAVA_HOME/lib/dt.jar（安装目录下的 java 库目录）;

PATH=$JAVA_HOME/binPATH （安装目录下的 bin 目录）;

export JAVA_HOME CLASSPATH PATH

执行#source /etc/profile，这个目录是使 profile 文件修改生效，而不需要重新启动系统。

再查看：#java –version，显示：

java version "1.8.0-ea"

Java(TM) SE Runtime Environment (build 1.8.0-ea-b36e)

Java HotSpot(TM) Client VM (build 25.0-b04, mixed mode)

说明 java8 安装成功。一般情况下，没有必要将 java7 升级为 java8。

2）安装 tomcat6：

Tomcat 可不是预安装的，首先下载 tomcat6，地址是：

http://tomcat.apache.org/download-60.cgi

下载 apache-tomcat-6.0.36.tar.gz 上传后，执行：

#tar zxvf apache-tomcat-6.0.36.tar.gz

#cd /apache-tomcat-6.0.36/bin

./startup.sh （Tomcat 启动命令）

显示：

Using CATALINA_BASE: /root/apache-tomcat-6.0.36

Using CATALINA_HOME: /root/apache-tomcat-6.0.36

Using CATALINA_TMPDIR: /root/apache-tomcat-6.0.36/temp

Using JRE_HOME: /root/jdk1.8.0

Using CLASSPATH: /root/apache-tomcat-6.0.36/bin/bootstrap.jar

至此，Tomcat6 安装完成，在树莓派或 PC 浏览器上输入 http://ip:8080/，其中 ip 为树莓派的 IP 地址，令人兴奋的“变态的小猫”（图 3-16）就出现了！

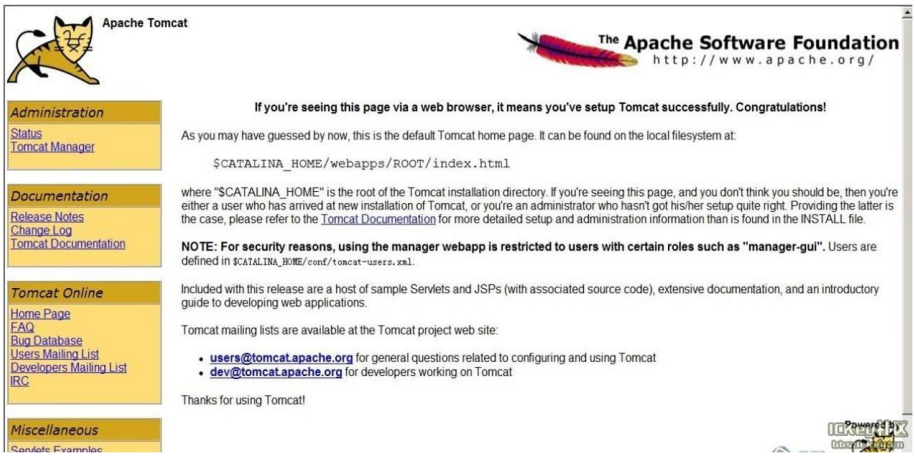

图 3-16 Tomcat 的登录界面

可以试运行一下 Tomcat 自带的网页，有关创建自己的网站的内容，将在以后再介绍。

3.4.6 让树莓派成为媒体服务器

树莓派的另一个应用就是娱乐功能，媒体播放器甚至是树莓派的“强项”。

1）树莓派媒体中心版本：

目前，树莓派媒体播放中心系统有三个主要版本：Raspbmc 是官方推荐的系统，功能比另外两个要多一些，例如：支持 LIRC、XBMC-PVR、自动调整分区、更快速的缓存和显示电影缩略图等；Xbian 则更新频繁，不过在某些软件使用上，违反了 GPL 版权；OpenELEC 以小巧、速度快著称，一张 1G 的 SD 卡就可以安装整个系统，并能通过 sh 脚本自动更新，2012 年 8 月以后的版本解决了大部分速度问题，已经成为一个优秀的媒体播放系统。上述三个系统本质上都是 XBMC 程序的（是在 Linux 上包装了一个 XBMC 应用程序）操作、甚至界面都大同小异。

XBMC 是一个优秀的开源媒体中心软件，最初是为 Xbox 游戏机开发的（目前的版本已经不支持 Xbox 了），XBMC 可以运行在 Linux、MacOS、Windows 系统上。XBMC 支持多种媒体应用，包括：音频、视频，还可以直接播放网络媒体，支持各种网络协议。只要指定网络的位置和协议，XBMC 就可以自动扫描媒体文件，建立资料库。如果没有封面、内容介绍、海报剧照，XBMC 可以自动在网上为你找到。当然，歌词也是这样。包括定时录制视频节目

等。而穿越 XBMC（如使用远程登录方式），其后面还是 Linux，稍后介绍。

2）树莓派媒体播放中心 OpenELEC 安装：

首先需要在 PC 上下载 OpenELEC-Eden 通用安装包：http://openelec.tv/get-openelec，以后的解压、将 SD 卡使用读卡器连上电脑，运行 DiskImager.exe、config.txt 文件的修改等等，与上一个系统“2014-06-20-wheezy-raspbian.img”的方法是一样的，不再重复。

3）中文语言设置说明：

1、在主界面选 SYSTEM，接着选 Appearance；

2、设定皮肤字体 Skin-Fonts 为 Arial based；

3、设定界面语言 International-Language 为 Chinese （Simple），OK 完成。如果出现方块或无字，则再设置一下 Skin-Fonts 为 Arial based，就行了。

4）设置自动更新：

一旦有新版 openelec 出来，系统可以自动下载升级。在系统界面上，将 system updata 设置为 auto。

5）SSH 支持：

OpenELEC 是缺省支持 SSH 的，因此，在 OpenELEC 的系统信息窗口查看其 IP 地址后，就可以用远程终端直接链接了。远程终端软件的使用，与上一个系统是一样的。OpenELEC 的缺省用户名：root，密码：openelec。

6）其他：

OpenELEC 也是自动支持 Samba 服务器、移动硬盘等，所以，使用上述功能不需要再进行设置。在 PC 的网络邻居上查找 OPENELEC（OpenELEC 的缺省机器名），就可以看见 OpenELEC 上的各共享目录，而且，移动硬盘直接把要播放的东西放到相应目录下，只要你的网够快）移动硬盘的位置：用 df –h 看一下，在/var/media 下面。

7）使用遥控器：

使用电视机的遥控器不用介绍，用 PC 或手机也可以做遥控器，方法也很简单。

知道 OpenELEC 的 IP 地址（如果把树莓派设为静态 IP，就可以省去这个问题了），然后在 PC 或手机的浏览器上，输入这个 IP 地址，就出现了如图 3-17 的界面，点击：Romote，遥控你的树莓派吧！

图 3-17 在 PC 上的 OpenELEC 遥控器界面

8）玩游戏：

《我的世界》（Minecraft）是一款游戏史上具有里程碑意义的作品：如果涵盖其所有版本的话，迄今已售出了大概三千万份的拷贝；对于一个看起来似乎缺乏明确“目标”的游戏，这实在是个了不起的成绩。它是一款独立的沙箱游戏，就像前面说的，没有一个明确的游戏

目标，你所做的一切都是关于“创造”（人们已经创造了很多实实在在的作品）：从运行中的计算机系统到企业号飞船的精细部件。

现在《我的世界》已经登陆到树莓派平台，而且让人高兴的两点是：《我的世界：树莓派版》是免费的。下载并解压：

wget https://s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz

tar -zxvf minecraft-pi-0.1.1.tar.gz

cd mcpi 解压目录

运行：./minecraft-pi 开始玩吧！

3.4.7 课程小结

本章第一节实训课没有做什么“开发”，而是“装”了两个系统（比 windows 安装还要简单），下载了几个软件，做了一些设置，玩了一个游戏。

实际上，由于篇幅关系，本节课没有展开的一个重要内容——你可能是第一次“玩”了一把 linux 系统。Linux 之庞大和复杂，岂是一节课能够说清楚、玩透彻的。但是，一点不知道也是不行的，因为现在已经“用”上 Linux 了。那些命令、配置文件修改、参数设置等等，都需要了解 Linux 的基本知识。

通过这节课，只想让同学们得到一个印象：树莓派是一个几乎跟我的笔记本电脑功能一样、基于 linux 的“电脑”，它不但“小”、“可移动”、“便宜”之外，几乎“无所不能”。这就是我们的开发和运行平台。有时间的话，同学们一定想要再仔细“研究”一下它。

实训课程到目前为止的另一个目的，是让同学们熟悉一下一个与 windows 完全不同的 OS 环境，特别是命令行环境，以及命令行下的操作，包括：设置、修改配置文件、下载软件并安装等。如果实训时间够的话，建议授课老师，至少用半天时间，为同学们补充 Linux 的基本知识。

3.5 用树莓派 GPIO 控制 LED

3.5.1 认识树莓派的 GPIO

从本节课开始，要涉及到具体的树莓派“硬件”了，对于不熟悉计算机硬件的同学，除了“好奇”，还会感觉：学软件与学硬件确实有所不同。

1）认识 GPIO 位置：

树莓派从 A 型、B 型，到现在最新的 B+型，都配置了 GPIO（General Purpose Input Output，通用输入/输出）接口。树莓派上的 GPIO 接口在哪里？它是怎么定义的？先看两幅图。图 3-18 的左侧是是树莓派的正面图，可见 PGIO 的 26 针（B+为 40 针），在左图的方框中，左上角为 1 脚、右上角为 2 脚，左排为单数脚、右排为双数。

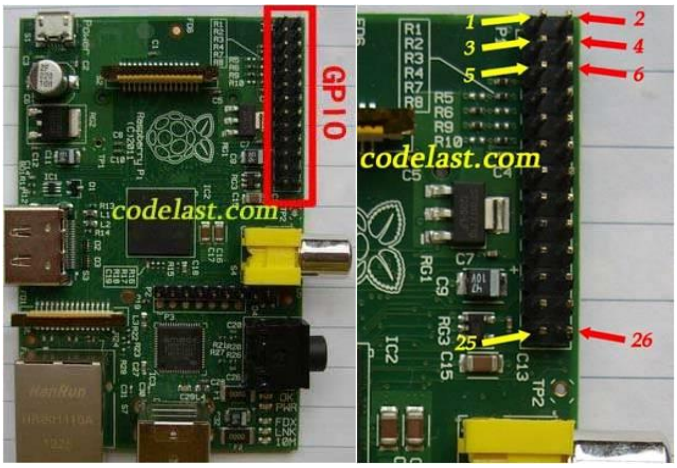

图 3-18 树莓派的 GPIO 脚

将 PI 的电路板翻到反面，在 26 个针脚中（图 4-18 右图），焊点为方形的那个针脚就是“1”，在它旁边同一排（两个针脚一排）的那一个就是“2”，依此类推。

所以：在树莓派的电路板上，GPIO 的物理引脚的位置从左到右，从上到下是：左边基数，右边偶数：1-26（树莓派版本不同，个数不同，A、B 型早期为 26 个，B+为 40 个）。

2）GPIO 编号：

树莓派 GPIO 的逻辑编号方式，有两种编号方法。即：BCM 编号法和 wpi 编号法。BCM 编号侧重 CPU 寄存器，根据 BCM2835 的 GPIO 寄存器编号。Wpi 编号侧重实现逻辑，把扩展 GPIO 端口从 0 开始编号，这种编号方便编程。

图 3-19 是两种编号方法的对照表：

BCM 编码方式 | wpi 编码方式 | 功能名 | 物理接口 | 功能名 | wpi 编码方式 | BCM 编码方式

BCM	wPi	Name	Mode	V	B Plus Physical	V	Mode	Name	wPi	BCM
		3.3v			1 || 2			5v		
2	8	SDA.1	ALT0	1	3 || 4			5V		
3	9	SCL.1	ALT0	1	5 || 6			0v		
4	7	GPIO. 7	IN	1	7 || 8	0	ALT0	TxD	15	14
		0v			9 || 10	1	ALT0	RxD	16	15
17	0	GPIO. 0	IN	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13 || 14			0v		
22	3	GPIO. 3	IN	0	15 || 16	0	IN	GPIO. 4	4	23
		3.3v			17 || 18	1	OUT	GPIO. 5	5	24
10	12	MOSI	ALT0	0	19 || 20			0v		
9	13	MISO	ALT0	1	21 || 22	1	OUT	GPIO. 6	6	25
11	14	SCLK	ALT0	1	23 || 24	1	ALT0	CE0	10	8
		0v			25 || 26	1	ALT0	CE1	11	7
0	30	SDA.0	ALT0	1	27 || 28	1	ALT0	SCL.0	31	1
5	21	GPIO.21	IN	1	29 || 30			0v		
6	22	GPIO.22	IN	1	31 || 32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33 || 34			0v		
19	24	GPIO.24	IN	0	35 || 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 || 38	0	IN	GPIO.28	28	20
		0v			39 || 40	0	IN	GPIO.29	29	21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM

图 3-19 BCM 与 wpi 编号对照表

从表中，可以看到：实际的物理接口，对应着两种编号：BCM 和 wpi，当然，还有一个重要的内容——接口的真正含义（功能）。功能图才是那个脚、应该接哪里的真正决定性因素。

3）GPIO 功能定义：

图 3-20 是树莓派 B 的 GIO26 脚的功能定义图。

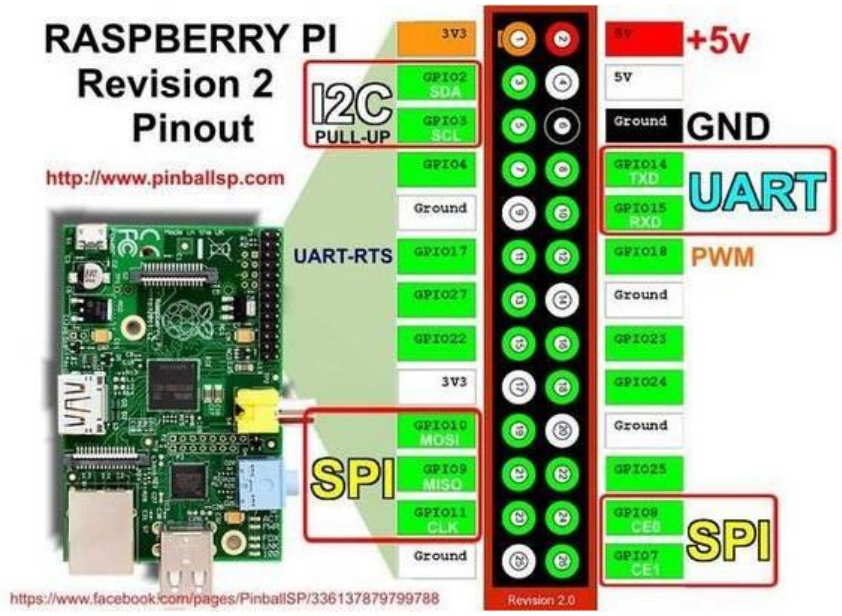

图 3-20 树莓派 B 的 GPIO 的功能定义

首先要知道的是：Pin01（Power 电源正）、Pin06（Ground 电源接地）、某一个控制脚，如：Pin11（GPIO17 控制信号）。知道了脚的定义，换一个相同定义的脚也没有问题了。

在树莓派的 PIN 脚编号旁边，还有其他的字母，这是这些脚的更“专业”名称。例如：3 和 5 是 SDA 和 SCL，它们分别是 I2C 串行总线的时钟和数据线。I2C 用于温度传感器和 LCD 显示器。8 和 10 用于串行通信的 RX 和 TX（接收发送），另一种串行通信（我们现在用的 SPI）则使用 11、13、15）。

4）针脚编号与常用编程命令：

以下部分暂时无法理解也没有关系，实际使用的时候，还会再介绍。

在实际编程时，目前有两种方式，可以对树莓派上的 IO 针脚进行编号。

第一种方式是使用 BOARD 编号系统。该方式参考树莓派主板上 P1 接线柱的针脚编号。使用该方式的优点是无需考虑主板的修订版本，你的硬件始终都是可用的状态，你将无需从新连接线路和更改你的代码。

第二种方式是使用 BCM 编号。这是一种较低层的工作方式。该方式参考 Broadcom SOC 的通道编号。使用过程中，你始终要保证主板上的针脚与图表上标注的通道编号相对应。你的脚本可能在树莓派主板进行修订版本更新时无法工作。

一般情况下，要指定使用哪种模式，否则会报错！

指定所使用的方式的代码是：

GPIO.setmode(GPIO.BOARD) //BOARD 模式；

或者

GPIO.setmode(GPIO.BCM) //BCM 模式；

警告：可能树莓派的 GPIO 上，同时有多个脚本在运行。因此，如果树莓派 GPIO 检测到某个针脚被设置为其它用途而非默认的状态（默认为输入），你会在尝试配置某脚本时，得到警告消息。禁用该警告消息的代码是：GPIO.setwarnings(False) 。

以下介绍一些常用的 GPIO 命令：

通道配置：

需要为每个用于输入或输出的针脚配置通道。

配置为输入通道的代码是：GPIO.setup(channel, GPIO.IN)，通道编号是基于所使用的编号

系统所指定的（BOARD 或 BCM）。

配置为输出通道的代码是：GPIO.setup(channel, GPIO.OUT)， 通道编号同上。

还可以指定输出通道的初始值：GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH) 。

输入命令：

读取 GPIO 针脚值的代码是：GPIO.input(channel)，这将返回 0 / GPIO.LOW / False 或者 1 / GPIO.HIGH / True。

输出命令：

设置 GPIO 针脚的输出状态：GPIO.output(channel, state)，状态可以为 0 / GPIO.LOW / False 或者 1 / GPIO.HIGH / True。

清理：

在任何程序结束后，请养成清理用过的资源的好习惯。使用树莓派.GPIO 也同样需要这样。恢复所有使用过的通道状态为输入，你可以避免由于短路意外损坏你的树莓派。注意，该操作仅会清理你的脚本使用过的 GPIO 通道。

在你的脚本结束后进行清理的命令是：GPIO.cleanup() 。

验证 GPIO 接口

同时，为了保险，还需要用万用表测量一下刚才的判断：在树莓派的工作状态下，将万用表的“+”接到“1”口上，万用表的“-”接到“6”口上，再看看万用表的显示是不是大约为正 3.3V，如果不是，那么说明弄错了针脚顺序，需要回头再检查！

5）常用 GPIO 开源工程：

树莓派内核中已经自带了 GPIO 的驱动，也可以通过一些第三方库函数，来完成具体的操作。这是程序员可以利用的最“底层”资源，一直往上，可以到达所谓“应用框架”层次。本实训除了教你树莓派的硬件和软件开发之外，另一个要教会你的技能，就是利用已有资源，进行开发。所以，那些喜欢“吹嘘”自己是编码“牛人”，“炫耀”一切自己搞定，有多么地无知。

比较常见的操作库函数有 3 种：

1、python GPIO：

【开发语言】——python（类似 C 语言）；

【简单介绍】——树莓派官方资料中推荐且容易上手。python GPIO 是一个小型的 python 库，可以帮助用户完成 raspberry 相关 IO 口操作，但是 python GPIO 库还没有支持 SPI、I2C 或者 1-wire 等总线接口。

【官方网站】—— https://code.google.com/p/raspberry-gpio-python/

本小节的第一个例子，就是用 python GPIO 实现的。

2、wiringPi：

【开发语言】——C 语言；

【简单介绍】——wiringPi 适合那些具有 C 语言基础，在接触树莓派之前，已经接触过单片机或者嵌入式开发的人群。wiringPi 的 API 函数和 arduino 非常相似，这也使得它广受欢迎。网上有大量的库代码说明和示例代码，这些示例代码也包括 UART 设备，I2C 设备和 SPI 设备等。

【官方网站】—— http://wiringpi.com/

如果有时间的话，也用 wiringPi，把第一个例子再做一遍。

3、BCM2835 C Library：

【开发语言】——C 语言；

【简单介绍】BCM2835 C Library 可以理解为使用 C 语言实现的相关底层驱动，BCM2835 C Library 的驱动库包括 GPIO、SPI 和 UART 等，可以通过学习 BCM2835 C Library，熟悉 BCM2835

相关的寄存器操作。如果有机会开发树莓派上的 linux 驱动，或自主开发 python 或 PHP 扩展驱动，可以从 BCM2835 C Library 找到不少的“灵感”。

【官方网站】—— http://www.airspayce.com/mikem/bcm2835/

最后，这个库也请有兴趣的同学尝试一下吧。

3.5.2 GPIO 接口编程体验

现在开始做第一个小实验。

1）与硬件有关的问题：

在计算机硬件中，所有开关信号，只有 1（通）和 0（断），这就是所谓的数字信号。电源开关，就是一个开关信号的例子。温度值通常不是开关信号，而被叫做模拟信号。它比数字信号要复杂，需要额外的设备（A/D、D/A 数模转换）来处理。 树莓派的 GPIO 没有模拟设备。只能处理数字信号，模拟信号需要经过数模转换，才能处理。

需要注意的是，树莓派上的 GPIO 是低功率的（电压是固定的 3.3V，功率变化体现为电流大小），所以有时你可能需要一块扩展板，借助额外的供电，来满足需要高功率的场合。功率是负载能力，当树莓派联接的 LED 灯太多的时候，负载不够，亮度就会明显变暗。这是学软件的同学从来没有遇到过的问题，更不要说“限流”保护，以后介绍。

2）实验材料准备：

第一个小实验是用树莓派点亮一个 LED 灯，这个实验的硬件，需要一个开关、一个 LED 灯、一个面包板和若干电阻和导线。如图 3-21。

开关：

需要一个如 3-21 图最左的按键开关。一般开关可以分为两种，闭锁开关（latching switch）和瞬时开关（momentary switch）。前者可以保持开或关的状态，就像电灯开关那样；后者则只有你按下它的时候才接通，比如键盘按键。在本实验中用的是一个普通的轻触开关（后者）。

图 3-21 点亮 LED 灯所需的材料

LED 灯：

LED 灯又被称为发光二级管，它们通常被用作指示灯。这里是一支普通的 3mm LED 灯。要注意的是：正因为 LED 灯是一只会发光的“二极管”，所以，它是有“正负极”的。

电阻：

需要 10 千欧、1 千欧和 47 欧的电阻三支，电阻的作用是对 LED 灯起“限流”保护作用。

面包板：

为方便实验，还需要一块面包板和一些导线。它们可以使你非常方便的连接和修改电路。

导线：

最后，还需要三条母对公的杜邦线，用来连接树莓派和面包板。也可以各使用三根公对公、母对母的导线对接起来使用。

3）面包板介绍：

如图 3-21 最右图。在面包板的最上部和最下部分别有两条平行走向的金属条，通常将它们分别设置为电源线（红色）和地线（黑色）。 上部或下部的电源线是联通的，也就是说，

在上部或下部一行的任何一个孔插入，都可以接到电源正或地线上。这样，面包板中部的电子器件可以方便地“就近”连接到电源（5V 等）或者地。“就近”可不是简单地图方便，硬件电路设计中，器件之间相连接的导线距离长短，有时是非常“讲究”的。

在面包板二组（上下）电源条之间的部分，是放置电子元器件的“中间地带”。中间地带分为上下二组，上组和下组的同列 5 个插孔是联通的，而同行是不通的（如图彩色连线）。所以，元器件放置或者是“跨列”，或者是同列“跨组”否则是无效的。大的面包板则还会分为更多的“组”。

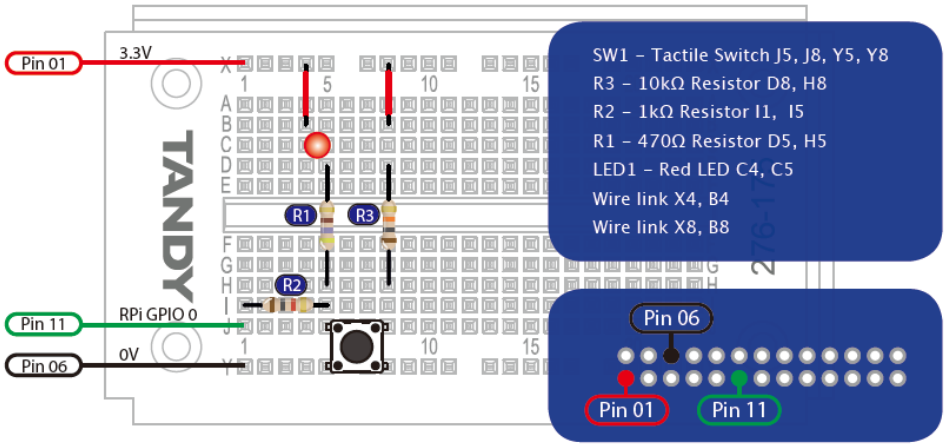

图 3-22 点亮 LED 的接线图

以马上要做的实验项目为例，如图 3-22：R1、R3 是同列跨组，而 R2 是跨列，LED 则是跨行跨列。两根电源线（正）从电源引向元器件，而开关则跨在地和元器件之间。另外 3 根导线则连接面包板和树莓派的相应针脚上。

4）面包板的外接电源：

面包板也可以不用树莓派的电源，而是用一个专用的电源模块，这是兼容树莓派、可提供 5V、3.3V 的电源。这个电源是单独给面包板上面的设备供电的。所以，可以不用再从树莓派上引出电源（可以省掉的是图 3-22 上的 pin01 和 pin11）。其最大的好处，就是减少了烧坏树莓派的风险。因为树莓派现在只提供控制信号，电源是隔离的。

面包板电源的主要参数是：输入电压：6.5-12V（直流）或 USB 供电；输出电压：3.3V、5V 可切换；最大输出电流：<700ma；上下两路两路独立控制，可切换为 0V、3.3V、5V，分别由板载两组 3.3V、5V 直流输出插针输出。外接电源插在面包板上的效果如图 3-23：

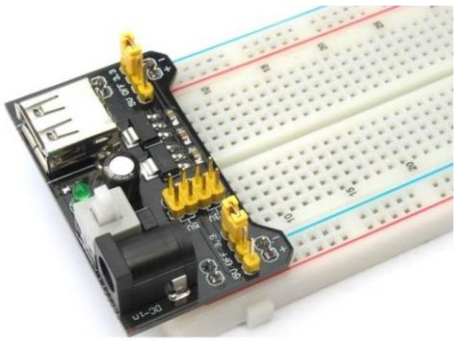

图 3-23 面包板与独立电源的连接

5）实验电路：

硬件工程师都是先看电路图（原理图），然后再搭电路的。这是先搞清原理，再动手实现。同学们可能第一次接触电路，往往直接看“线是怎么接的（哪根接哪根）”，不管为什么。

错了以后，当然是“一头雾水”。图 3-24 是用树莓派点亮 LED 灯的电路图。

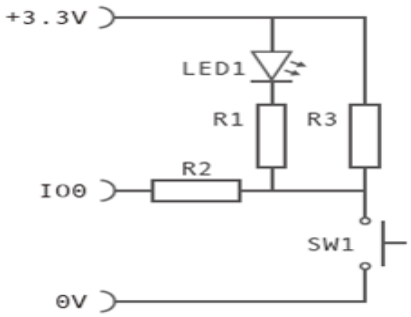

图 3-24 树莓派点亮 LED 的电路图

这个图非常简单，似乎不相应解释了。按照电路图，把所有元件插上面包板。要插在面包板上的电子零件分别是：1 个 LED 灯、1 个开关、3 个电阻：R1、R2、R3，以及 5 根导线，2 根板上连接导线、3 根外接导线（树莓派一头的接法暂时不管）。具体链接方式，请先看懂图 3-24，再对照图 3-22 的实际接线图，进行连接。注意，除 LED 外，其他零件没有正负极。开关两脚必需跨在电源槽和空插槽之间。LED 的正负极不要接错，一般而言，正极引脚比较长，插在 C4 孔，负极引脚短，插在 C5。三条外接的杜邦线接树莓派的 pin1、pin6 和 pin11。

如果没有开关，可以用导线把开关的位置接上，即：连接 Y5 与 J5、Y8 与 J8。用什么当“开关”呢？自己想一下，是否有替代方法。

6）启动树莓派：

连接好电路（包括与树莓派相连的 3 根导线、如果使用外部面包板电源，则不需要连接树莓派电源的那 2 根导线、只连接 1 根信号线），再三检查无误后，打开树莓派，按下开关，LED 应该亮起，否则请检查电路。

如果 LED 在开关放开的情况下仍然发光，可以把开关取下旋转 90 度插回（开关脚位置错）。因为树莓派的接口不带 buffer 保护，所以接入 R2 作为保护。而 10 千欧的 R3 作为上拉电阻，确保 pin11 始终处在高电位，当按下开关时，pin11 就被拉到低电位。

7）安装 python GPIO 包：

到目前为止，严格意义上说，LED 并没有受树莓派的控制（只是提供了一个环境），项目要做的是：使用 python 语言，控制 LED。在上一小节有关 GPIO 开源工程的介绍中，已经提到 python。Python（英语发音：/ˈpaɪθən/）是一种面向对象的、解释型的计算机程序设计语言，由 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年。Python 是纯粹的自由软件， 源代码和解释器遵循 GPL 协议、语法简洁清晰，具有丰富和强大的库。在我们的系统中，python 是预安装的，因此，不需要再装了。

为了保险起见，也可以再安装一次 python-dev，输入以下指令：

$ sudo apt-get install python-dev

或通过安装 python 程序包来赋予树莓派控制 GPIO 的能力。从 http://pypi.python.org/pypi/树莓派.GPIO 可以下载最新的版本。安装过程如下：

gunzip RPi.GPIO-0.2.0.tar.gz（下载的版本号可能不同）；

tar -xvf RPi.GPIO-0.2.0.tar （解压）；

cd RPi.GPIO-0.2.0

sudo python setup.py install （自动安装）；

至此，安装过程结束。

8）编写编程：

现在开始写一个程序，来监视面包板上的开关状态，并在屏幕上显示些东西。

$cd ..　（回到上一个目录）；

$nano mybutton.py （创建一个名为 mybutton.py 的 python 新文件，也可以用 vi 作编辑器）；

在新文件中输入：（注意空格，否则报错，这是 python 特色。中文可能成为乱码，暂时不用！）。有关 python 编程语言的学习，采用“用到哪里、学到哪里”的方法进行，而且如果老师没有介绍的话，就自己上网查该语句的含义。

```
#！/usr/bin/python  //python 特色语句，用来指定用解释器运行脚本以及解释器所在的位置
import time   //引入 time 和树莓派.GPIO 以使用其中的函数
import RPi.GPIO as GPIO//设置模式
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN) //把 pin11 配置为 input 模式以接收开关状态，注意与实际接线相符
while True: // while True 是一个死循环
    mybutton = GPIO.input(11) //读开关状态是否按下
  If mybutton == False: // if 语句判断 pin11 取回的状态，当它变成低电位，即按下开关时，就在屏幕上输出一个 OK
        print “OK”
        time.sleep(.2) //让程序休息 0.2 秒，时间可以自己定
  GPIO.cleanup() //退出时清理通道
```

保存上述内容，运行程序（python 是解释型的编程语言）、输入：

$sudo python mybutton.py

程序开始运行，按下开关，应该可以看见屏幕上出现的 OK。按 ctrl+c 可以终止程序运行（程序是在 while True 死循环中的）。

9）让 LED 闪烁起来：

把程序改一下：

```
GPIO.setup(11,GPIO.OUT)
while True：
 GPIO.output(11, GPIO.HIGH)
time.sleep(1)
GPIO.output(11, GPIO.LOW)
time.sleep(1)
```

LED 灯常亮？——把地线拔掉——好了，闪了！ 让它在后台运行吧！运行命令改为：

python mybutton.py&

此时，系统返回一个进程号 ****，程序开始在后台运行。要杀掉此进程（此时用 ctrl+c 已经没有用了）：Kill **** （****为刚才显示的进程号）。

10）计数器：

请同学们自己为程序添加一个计数器，把死循环改成计数器，并每按一次开关，就“报数”一次。当达到一定次数后，程序就自动终止运行。没有开关的话，也可以用拔插电源线的方式代替。

3.5.3 使用函数库实现 GPIO 接口控制

还可以有其他方法，实现对树莓派的 GPIO 接口进行控制。

1）用 wiringPi GPIO 实现对 LED 的控制：

前已介绍，WiringPi 是应用于树莓派平台的 GPIO 控制函数库，使用 wiringPi GPIO 的好处是程序实现更简单，因为一些复杂的控制，都由 WiringPi 帮你实现了。所以，它有点类似于 windows 下的设备驱动程序。WiringPi 遵守 GUN Lv3 规范、使用 C 或者 C++开发，并且可以被其他语言包使用，例如 python、ruby 或者 PHP 等。

wiringPi 包括一套 GPIO 控制命令，使用 GPIO 命令可以控制树莓派 GPIO 管脚，用户可以利用 GPIO 命令，通过 shell 脚本控制或查询 GPIO 管脚。wiringPi 是可以扩展的，可以利用 wiringPi 的内部模块扩展模拟量输入芯片，可以使用 MCP23x17/MCP23x08（I2C 或者 SPI）扩展 GPIO 接口。另外，用户可以自己编写扩展模块并把自定义的扩展模块集成到 wiringPi 中。

WiringPi 支持模拟量的读取和设置功能，不过在树莓派上并没有模拟量设备。但是使用 WiringPi 中的软件模块却可以轻松地应用 AD/DA 芯片。

wiringPi GPIO 安装：

1)安装方案 1——使用 GIT 工具：

通过 GIT 获得 wiringPi 的源代码：

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

build 脚本会帮助你编译和安装 wiringPi

2)方案 2——直接下载：

在 https://git.drogon.net/?p=wiringPi;a=summary 上直接下载最新版本，编译使用：

```
tar xfz wiringPi-xx.tar.gz
cd wiringPi-xx
./build
```

3)make

确保你已经安装了 make，如果没装，很简单：pacman -S make。

下载 wiringPi 安装包后，解压，编译：

```
mkdir temp
cd temp
wget http://project-downloads.drogon.net/files/wiringPi.tgz //下载
tar xf wiringPi.tgz //解压
cd wiringPi/wiringPi/
make
make install //编译
```

这样 wiringPi 就算安装完了。

wiringPi GPIO 测试：

wiringPi 包括一套 GPIO 命令，使用 GPIO 命令可以控制树莓派上的各种接口，通过以下指令可以测试 wiringPi 是否安装成功：

```
$gpio -v
$gpio readall
```

即可出现测试 IO 图 3-25：

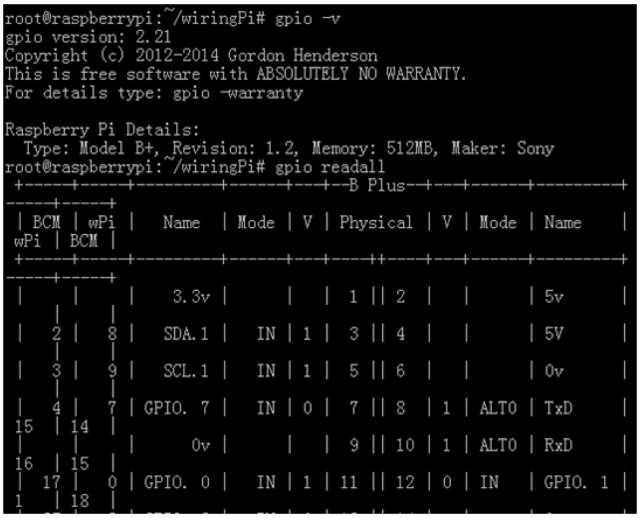

图 3-25 wiringPi 测试图

2）WiringPi 程序的例子：

以下是 wiringPi GPIO 程序例子，创建如下程序：

```
#include <wiringPi.h>
int main(void)
{
wiringPiSetup() ;
pinMode (0, OUTPUT) ;
for(;;)
    {
digitalWrite(0, HIGH) ; delay (500) ;
digitalWrite(0,  LOW) ; delay (500) ;
    }
}
```

编译运行，在树莓派上输入命令：

```
$gcc -Wall -o test test.c -lwiringPi
$sudo ./test
```

注意事项：

（1）IO 的编号方式略有不同，采用 wiring 编码方式。

（2）-lwiringPi 表示动态加载 wiringPi 共享库。

3）使用 WiringPi 实现点亮 LED：

电路图如图 3-26：

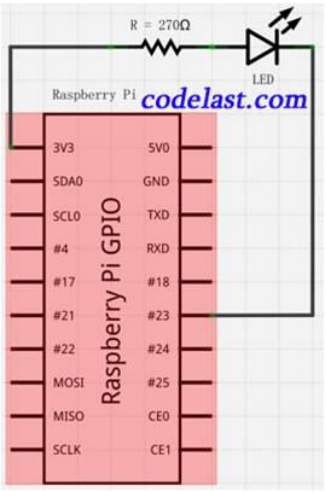

图 3-26 wiringPi 点亮 LED

在本例中，随意选了一个 GPIO 口——GPIO 23（即端口 16）来接 LED。需要注意的是，一旦你用 digitalWrite()函数置了高电平，那么只要你不把它置为低电平，它将一直维持在高电平（这就是库函数的作用，不需要你“费神”监视着）。

相应代码如下，注意 wiringPi 的代码是 C 语言：

```
// led.c
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc,char* argv[])
{
if (argc < 2) {
printf("Usage example: ./led 4 \n");
return 1;
}
int pinNumber = atoi(argv[1]);
if (-1 == wiringPiSetup()) {
printf("Setup wiringPi failed!");
return 1;
}
pinMode(pinNumber, OUTPUT); // 设置为 output 模式，设置方法不同
while(1) {
digitalWrite(pinNumber, 1); // 置为高电平
delay(800); //暂停
digitalWrite(pinNumber, 0); // 置为低电平
delay(800);
}
return 0;
}
```

编译程序：

```
gcc led.c -o led –lwiringPi
```

运行程序：

./led 4

显示结果。

现在可以看到 LED 开始闪烁了。

这里向程序传入了一个参数 4，它代表你要置高电平的是 GPIO 的哪个脚。为什么是 4 呢？因为以前已经说了，树莓派的 GPIO 口有两种命名方式，一种是树莓派的方式，另一种是 Broadcom 的方式，当使用 WiringPi 时，应参考前者。因此，对应到 Broadcom 方式的 GPIO 23 上，那就应该是 GPIO 4，所以应该向程序输入参数 4。

4）作业：

请同学们自己实现：点亮多个 LED，让它们轮流闪烁！ 提示：

（1）参考图 3-24 的电路图，将控制一个 LED“复制”为控制多个 LED；

（2）思考一下发生了什么改变？为什么？

（3）如果用 WiringPi GPIO 直接控制 LED 的话，至少可以接多少 LED？

5）BCM2835 C Library 的用法：

1、下载:

$wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.35.tar.gz

2、解压缩: $tar xvzf bcm2835-1.35.tar.gz

3、进入压缩之后的目录:$cd bcm2835-1.35

4、 配置: $./configure

5、从源代码生成安装包:$make

6、执行检查: $sudo make check

7、安装 bcm2835 库: $sudo make install

2、实验代码：

```
#include <bcm2835.h>
// P1 插座第 11 脚
#define PIN 树莓派_GPIO_P1_11
int main(int argc, char **argv)
{
if (!bcm2835_init())
 return 1;
// 输出方式
bcm2835_gpio_fsel(PIN, BCM2835_GPIO_FSEL_OUTP);
while (1)
{
bcm2835_gpio_write(PIN, HIGH);
 bcm2835_delay(100);
bcm2835_gpio_write(PIN, LOW);
bcm2835_delay(100);
 }
 bcm2835_close();
 return 0;
}
```

第一个小实验到此结束，请同学们总结并体会一下三种方法的异同。

3.6 了解 arduino 开发平台

把多个 LED 灯点亮以后，同学们的桌上，顿时“活跃”起来，好像圣诞来临一样。下面将把 LED 灯，换成温度探头，并用另一个平台——arduino 来进行采集，采集到的数据，再通过串口，送到树莓派上，进行显示。

3.6.1 arduino 平台介绍

1）Arduino 的环境：

Arduino，是一个基于开放源码的软硬件开发平台，构建于开放源码 simple I/O 介面版，并且具有使用类似 Java、C 语言的 Processing/Wiring 开发环境。Arduino 平台包含硬件（有多种型号、如图 3-27），以及软件 IDE（图 3-28）两个部分组成。Arduino 的硬件部分可以用来做电路连接，在本小节实验中，用来连接各种传感器；Arduino 的软件部分则是 IDE——你的程序开发环境。

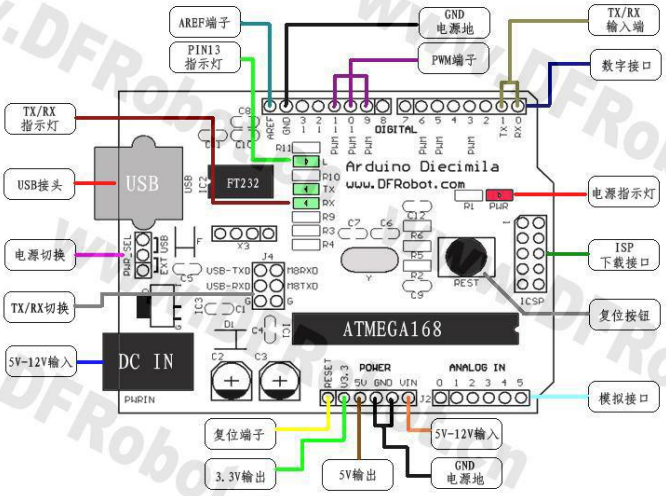

图 3-27 Arduino 的硬件

Arduino 板子上的微控制器可以通过 Arduino 的编程语言来编写程序，编译成二进制文件，写进微控制器（类似传统的 51 单片机）。基于 Arduino 的项目，可以只包含 Arduino，也可以将 Arduino 与 PC、树莓派集成，在他们之间进行通信来实现。

2）树莓派与 arduino 的区别：

1、树莓派和 Arduino 是两个完全不同运算水平的平台，毕竟树莓派使用的是一颗运行在 700MHZ 的 ARM11 CPU；而常用的 Arduino UNO 则是一颗在工业控制领域常见的 8-bit ATmega328，最高运行频率仅 20MHz。

2、树莓派可以运行完整的操作系统，如 Debian 等常见 Linux 发行版 。这意味着你可以使用你熟练的语言（如 Python、Java）和熟悉的库来进行开发，同时后台运行多个进程也毫无压力。而 Arduino 作为一款真正传统意义上的单片机系统，一次只能运行一个你烧进去的程序，功能也相对单一。

3、树莓派自带的接口比较全面，USB-host、RJ45、HDMI、SD 读卡器等常用接口都有；而 Arduino 比较单纯，与外部设备打交道需要另外采购接口板。

4、树莓派因为是一个相对完整的“电脑”，其成本肯定比 Arduino 高，目前 B+版在淘宝上约为 200 元左右；而常用的 Arduino UNO 只需要 40 元。

5、树莓派拥有更完整的操作系统，这也意味着每次复电后所需的启动时间较长，包括

需要有些相关的服务也能正常启动；但 Arduino 因为用途单一，重启速度超快，而且重启后就立即直接运行你的程序（没有操作系统）。

3）树莓派与 Arduino 的分工：

如果让树莓派作为中央控制服务器，负责与互联网的通信、采样存储 Arduino 上报的状态数据、处理数据量大的工作（如音频、视频、图片相关）、提供 API 给 iOS 及 Android 以方便用手机控制家居。树莓派 与 Arduino 间通过以太网或 Zigbee 进行数据传输。

而 Arduino 负责传感器采样，如光线强度、温度、人体红外感应等。有的模块比较独立，比如人体感应的夜间走廊灯，单个 Arduino 可以自己实现监控加控制，就不需要上报数据给树莓派了。需要上报和接收数据的 Arduino，通过 Zigbee 及以太网和树莓派进行通信。

两者的结合发挥了各自的优点，规避了缺点，可谓扬长避短、各得其所，算是物尽其用。还有一个更大的便利就是，简化了树莓派对外设的控制编程。

4）Zigbee 协议：

ZigBee 是基于 IEEE802.15.4 标准的低功耗局域网协议。根据国际标准规定，ZigBee 技术是一种短距离、低功耗的无线通信技术。ZigBee 又称“紫蜂”协议，来源于蜜蜂的八字舞，由于蜜蜂（bee）是靠飞翔和“嗡嗡”（zig）地抖动翅膀的“舞蹈”来与同伴传递花粉所在方位信息，也就是说蜜蜂依靠这样的方式构成了群体中的通信网络。其特点是近距离、低复杂度、自组织、低功耗、低数据速率。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。简而言之，ZigBee 就是一种便宜的，低功耗的近距离无线组网通讯技术。

3.6.2 windows 环境下的 arduino IDE

Arduino IDE 是 Arduino 开放源代码的集成开发环境，其界面友好，语法简单以及能方便的下载程序，使得 Arduino 的程序开发变得非常便捷。作为一款开放源代码的软件，Arduino IDE 也是由 Java、Processing、avr-gcc 等开放源码的软件写成，其另一个最大特点是跨平台的兼容性，适用于 Windows、Max OS X 以及 Linux。

2011 年 11 月 30 日 Arduino 官方正式发布了 Arduino1.0 版本，可以下载不同系统下的压缩包，也可以在 github（开源代码库）上下载源码重新编译自己的 IDE。使用方法之一是在 Windows 下安装 Linux 环境，然后再安装 Arduino IDE。这种模式必须先安装开发工具 Cygwin、Java JDK、Ant。由于可以在 windows 下可以直接使用 Arduino IDE，所以，就没有必要“多此一举”，再安装 Linux 虚拟环境了。即采用另一种模式，就是在 Windows 下直接装 Arduino IDE。

1）windows 下的 Arduino IDE 安装：

下载 Windows 版 IDE：http://arduino.googlecode.com/files/arduino-1.0.1-windows.zip。

可以直接找最新中文版，但有的下载不带驱动，需要自己安装驱动。安装驱动的方法是：Windows 环境：XP、WIN7 32 位需要手工更新驱动，驱动在 IDE 目录下 drivers\FTDI USB Drivers 目录中，指定目录为自动搜索即可。WIN7 64 位，WIN8 环境：自动在线更新驱动即可。

将 arduino 方型接头的 usb 数据线连接到电脑，以 win7 32 位为例，windows 会出现识别到新硬件显示。然后，输入下载的驱动目录。

如果没有自动识别新硬件，则双击“计算机|属性|设备管理器|其他设备|USB Serial Port”，选择“更新驱动程序软件”。在弹出的窗口中，输入下载的驱动目录。

进入下载软件解压后的目录，运行 arduino.exe 文件，打开 IDE，出现如图 3-28 的界面。

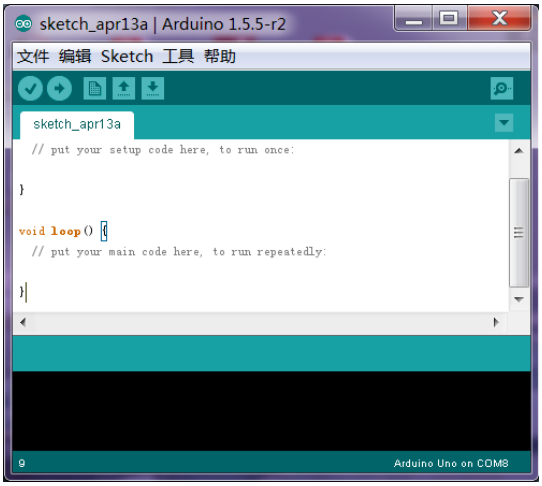

图 3-28 Arduino 的 IDE 界面

IDE 的几个快捷图标工具依次是：编译、上传、新建程序、打开程序、保存程序、串口监视器等。

2）菜单项的主要功能介绍：

（1）IDE 菜单功能说明——文件：

文件是 IDE 的第一个选项，主要功能如图 3-29。其中 Upload 非常重要，在 IDE 上编译完成的程序，就是通过这个 Upload 上传（有人称为：下载）到 arduino uno 上运行的。

图 3-29 Arduino IDE 的 File 功能

（2）IDE 菜单功能说明——编辑：

编辑功能比较简单，易于理解，如图 3-30：

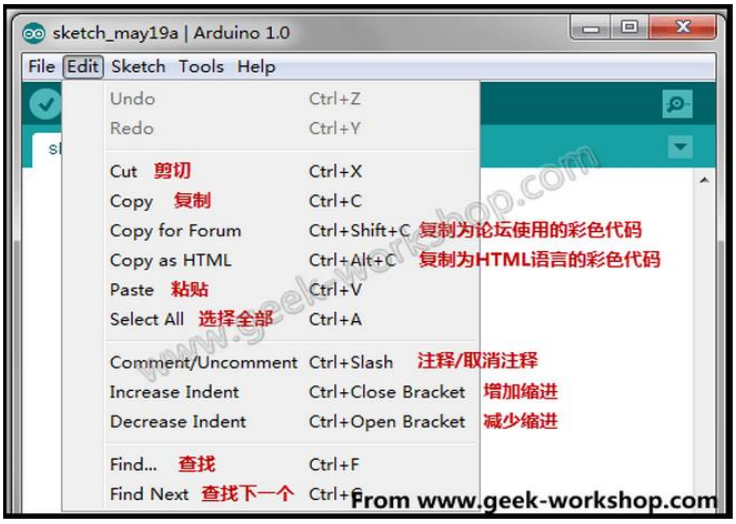

图 3-30 Arduino IDE 的 Edit 功能

（3）IDE 菜单功能说明——草稿：

这个菜单下，有几个重要的功能，包括：验证和编译——这是 IDE 编译并查错的主要选项，编译验证正确后，才能用 Upload 上传到片子上执行，所以称为：草稿。

图 3-31 Arduino IDE 的 Sketch 功能

该菜单的另一个功能就是 Improt Library（导入库），这是 IDE 自带的源代码资源，可以直接使用。网上有很多例库，可以下载，也可以自己编写后扩充。常用的导入库包括：

- EEPROM - EEPROM 读写程序库
- Ethernet - 以太网控制器程序库
- LiquidCrystal - LCD 控制程序库
- Servo - 舵机控制程序库
- SoftwareSerial - 任何数字 IO 口模拟串口程序库
- Stepper - 步进电机控制程序库
- Wire - TWI/I2C 总线程序库
- Matrix - LED 矩阵控制程序库
- Sprite - LED 矩阵图象处理控制程序库

（4）IDE 菜单功能说明——工具：

工具菜单的功能如图 3-32：

图 3-32 Arduino IDE 工具菜单

工具菜单中，有一个非常重要的参数设置项——选择开发版：点击“工具|板”右键，选择与所使用的 arduino 一致的板子——例如本项目使用的是 arduino uno。然后，再选择端口：点击“工具|端口”右键，看到本项目使用的是 COM6 口（不同 PC 的 USB 口，可能会有不同，因注意 USB 插入 PC 主机时的 windows 系统提示）。

点击 IDE 菜单最右边的窗口监视器，在输入栏输入 123456，应得到如图 3-33 的回应，说明 arduino uno 工作正常。

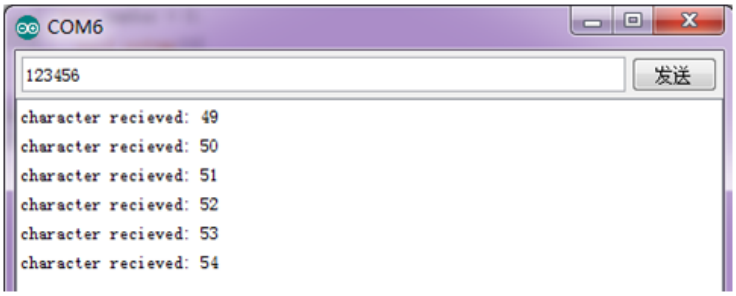

图 3-33 验证 Arduino 与 PC 的 USB 连接正确

3.6.3 树莓派环境下的 arduino IDE

目前的树莓派已自带 arduino IDE，因此，完全不需要自己再安装。

在树莓派图形界面上，点击 arduino IDE 图标，或在命令行终端，依次输入以下命令：

cd arduino-1.0/ （arduino 所在的目录）

./arduino

此时，你就会发现 Arduino IDE 已经成功运行，并与在 Windows 下的界面是完全一样的。由于 arduino 编写的程序，是直接在 arduino IDE 上编译、下载到 arduino 芯片上运行的，因此，在 windows 环境下的 arduino IDE 编译生成的代码，与在树莓派上生成的是完全一样的。那么，我们当然熟悉并喜欢在 windows 下，使用 arduino IDE。

3.6.4 树莓派与 arduino IDE 通信

树莓派连接 Arduino Uno 的方式有很多种，主要有三种：GPIO 方式、USB 方式以及 I2C 方式。

1） GPIO 方式：

简单介绍一下 GPIO 方式：

1.安装 python（系统已自带，可跳到下一步）：

$sudo aptitude install python-dev

2.安装 python 的 GPIO 模块，用于控制 LED、电机等（安装过的请跳到下一步）：

下载 GPIO 库：

wget http://raspberry-gpio-python.googlecode.com/files/树莓派.GPIO-0.3.1a.tar.gz；

tar 解压：

tar xvzf 树莓派.GPIO-0.3.1a.tar.gz；

进入解压后的文件夹：

cd RPi.GPIO-0.3.1a

安装 GPIO 库文件：

sudo python setup.py install

3.安装 serial，用于串口通信及 USB 通信：

$sudo apt-get install python-serial

4.当然你要是想在树莓派装串口调试工具就装：

$sudo apt-get install minicom

配置 minicom：sudo minicom –s

命令 minicom 是进入串口超级终端画面，而 minicom -s 为配置 minicom。/dev/ttyAMA0 对应为串口 0，为你连接开发板的端口。Minicom 命令的界面如图 3-34：

图 3-34 minicom 命令的界面

启动出现配置菜单：选 serial port setup，进入串口配置：

输入 A：配置串口驱动为/dev/ttyAMA0

输入 E：配置速率为 9600 8N1

输入 F：将 Hardware Flow Control 设为：NO

回车：退出

输入结果如图 3-35：

图 3-35 minicom 命令的设置结果

由于这里使用了 minicom 作为超级终端控制路由器等设备，而不是控制 modem，所以需要修改 Modem and dialing，将 Init string, Reset string, Hang-up string 设置为空。设置后的

结果如图 3-36：

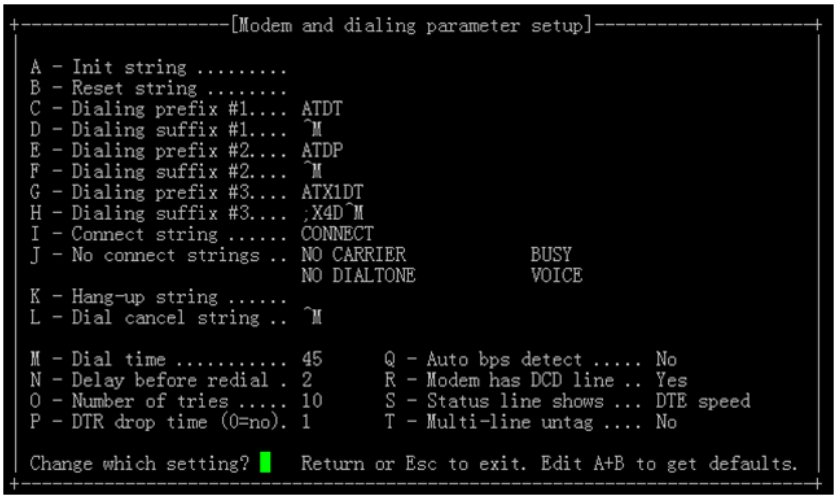

图 3-36 修改设置后的结果

设置完成后，选择 Save setup as dfl 将当前设置保存为默认设置。选 Exit 退出。下次再输入 minicom（没有参数-s），即可直接进入串口调试工具。

测试一下环境设置是否 OK，在树莓派上：

sudo nano test.py 或在 window 下用 python IDE 打开一个空的文件，输入以下内容：

import serial

import RPi.GPIO

保存退出，然后运行代码：

python test.py

如果没有报错，那就说明 RPi.GPIO 与 serial 两个库安装成功。

2）USB 对接方式：

USB 对接方式，需要一根 USB 对 USB 的对接线，这根线是购买 arduino 时自带的。用这根线的“方头”接 arduino（如图 3-37），另一 USB 头接树莓派。对接完成后，在树莓派和 Arduino 两边，都是把 USB 接口当成串口使用。

图 3-37 arduino USB 连接

连接好 USB 线以后，在树莓派输入：ls /dev/tty* ，查看有没有 ttyACM0 这个文件，命令结果如图 3-38：

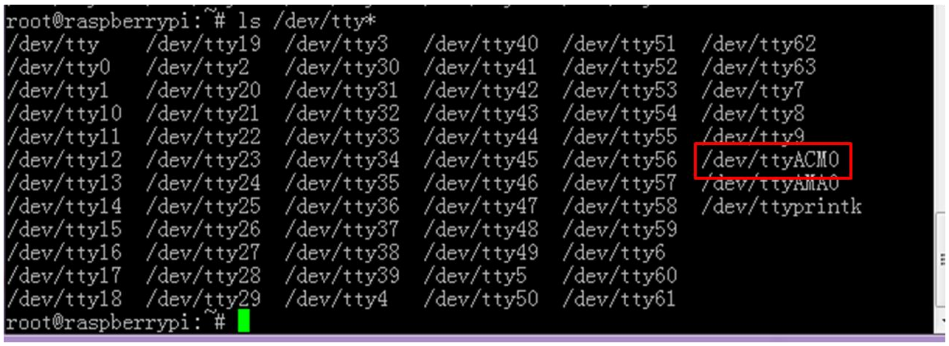

图 3-38 查看 USB 串口设备/dev/ttyACM0

只有在两个硬件 USB 互连的情况下，才会有这个 ttyACM0 设备，否则不会出现。看到列表中有 ttyACM0 设备，就说明二者可以通讯了，接下来运行测试代码。

在 windows 的 arduino IDE 上，建立如下代码：

```
byte number = 0;
void setup(){
Serial.begin(9600); //设置串口通讯速率为 9600bps/s
}
void loop(){
if (Serial.available()) {
      number = Serial.read(); //读取串口（来自树莓派）
      Serial.print("character recieved:"); //送回串口接收到的字符
      Serial.println(number, DEC);
      }
}
```

在 IDE 上编译完成上述程序后，通过 Upload 上传到 arduino 上。

在树莓派上，建立如下的 python 代码程序。

```
import serial //导入串口库
ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1) //设置串口
ser.open() //打开串口
ser.write("testing") //写串口
try:
    while 1:
          response = ser.readline() //读串口
          print response //显示内容
except KeyboardInterrupt: //接收键盘中断
    ser.close() //关闭串口
```

保存 serial.py 退出。注意：树莓派下的 serial 以及后面谈到的 GPIO 库都要在 root 帐户下才能运行。然后运行树莓派代码 python serial.py（这里的 serial.py 就是刚刚保存的树莓派代码）。

运行结果显示：

```
character recieved: 116 -- t
character recieved: 101 -- e
character recieved: 115 -- s
...
```

character recieved: 103 -- g

说明两者通过 USB 通迅成功了，上述收到的代码是什么意思？

就是 serial.py 程序的一行代码 ser.write(“testing”)（向串口写 testing），以及 Arduino 的两行显示代码：

Serial.print("character recieved: "); // 显示一行接收字符

Serial.println(number, DEC); // 显示接收的每个字符（字符代码）

从上述程序例子中，可以大致了解树莓派与 arduino 之间是如何通过串口方式（实际为 USB 转）进行信息交互的。在本案例中，树莓派有发送和接收，Arduino 有接收和发送，构成“交互”。双方的串口读写，代码实现很简单。

简单小结一下：

Arduino 的程序是以 setup()开头，loop()作为主体的一个程序构架。setup()：用来初始化变量，管脚模式，调用库函数等等，此函数只运行一次。loop()函数是一个循环函数，函数内的语句“周而复始“地循环执行，功能类似 c 语言中的 main()。

树莓派的代码首先导入串口库，然后设置串口读写模式，并对串口进行读写。串口读写有多种模式，以后还会继续介绍。

3）GPIO 方式：

GPIO 方式就是利用树莓派的 pin8（GPIO14）、pin10（GPIO15）与 Arduion pin0(rx) pin1(tx) 的进行对接。看一下树莓派的接口定义（图 4-19）可以知道：GPIO14-15 就是串口的读和写口。而 arduino 的 pin0 和 pin1 也是串口的读和写，两边都是真正的“串口”。

在连线的时候，千万注意：双方的读/写要“交叉”，我的读是连你的写，反之亦然。具体 GPIO 的连接方式，不再详细介绍。

3.7 用 DHT11 获取温度

传感器的第一个实验是用 arduino 做平台，采集 DHT11 温度传感器的信息，并上传到树莓派上显示。

3.7.1 DHT11 温湿度传感器简介

温度传感器有很多种类型。DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比高等优点。图 3-39 是 DHT11 的外形图（正反面）。

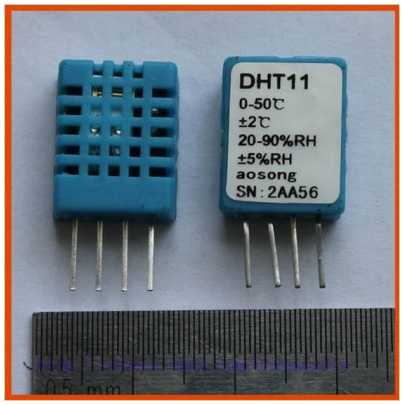

图 3-39 DHT11 正反面图

DHT11 数字温湿度传感器模块为 3 针 PH2.0 封装，读取数据只需要占用一个 IO 口。能够同时测量温度和相对湿度。

3.7.2 DHT11 传感器的技术指标

由于本实验只是简单地获取温度信息，对传感器本身的其他技术指标，并不太关心，但是，作为一名“专业人士”，多少还是要知道一些基本的内容。

DHT11 的主要性能指标描述如下：

1. 供电电压：3-5.5V；
2. 供电电流：最大 2.5Ma；
3. 温度范围：0-50℃ 误差±2℃；
4. 湿度范围：20-90%RH 误差±5%RH；
5. 响应时间: 1/e(63%) 6-30s；
6. 测量分辨率分别为 8bit（温度）、8bit（湿度）；
7. 采样周期间隔不得低于 1 秒钟；
8. 模块尺寸：30x20mm。

3.7.3 连接电路

常规的 DHT11 与上位机（目前是 arduino）的连接如图 4-40。注意：在自动控制专业，一般按控制设备的作用划分层次，传感器与信息采集处理设备（如：arduino）相比，处于更底层的位置，所以，称 arduino 为“上位机”。

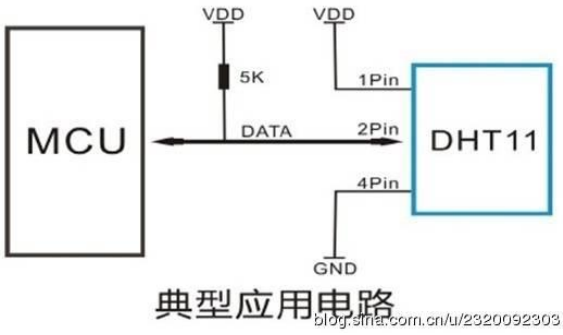

图 3-40 DHT11 的连接电路图

DHT11 传感器硬件连接将 DHT11 温湿度传感器的 VCC、GND 分别连接至 Arduino 控制器的+5V、GND，以给 DHT11 提供电源。DHT11 模块的 DOUT 引脚接至 ArduinoUno 控制器数字

引脚 D2，且并联 5kΩ 的上拉电阻，DHT11 模块的 NC 引脚也连接至 GND。上拉电阻不用也可以，主要起保护作用。

DHT11 的引脚定义如表 3-1：

Pin	名称	注释
1	VDD	供电 3－5.5VDC
2	DATA	串行数据，单总线
3	NC	空脚，请悬空
4	GND	接地，电源负极

表 3-1：DHT11 的引脚定义

3.7.4 DHT11 的时序

由于 DHT11 传感器是采用单线制串行通讯的方法进行采样数据的，要配合时序一位位地从单条通讯线传过来，再合成 8 位字节，然后还要进行校验和，判断数据传送是否正确。

DATA 用于微处理器与 DHT11 之间的通讯和同步，采用单总线数据格式。一次通讯时间 4ms 左右。数据分小数部分和整数部分。具体格式在下面说明。当前小数部分用于以后扩展，读出为零。

一次读取的操作流程如下:

一次完整的数据传输为 40bit，高位先出。

数据格式:

8bit 湿度整数数据+8bit 湿度小数数据；

+8bi 温度整数数据+8bit 温度小数数据；

+8bit 校验和。

数据传送正确时，校验和数据等于“8bit 湿度整数数据+8bit 湿度小数数据+8bi 温度整数数据+8bit 温度小数数据”所得结果的末 8 位。

用户（MCU）发送一次开始信号后，DHT11 从低功耗模式转换到高速模式，等待主机开始信号结束后，DHT11 发送响应信号，送出 40bit 的数据，并触发一次信号采集。用户可选择读取部分数据。DHT11 接收到开始信号触发一次温湿度采集。如果没有接收到主机发送开始信号，DHT11 不会主动进行温湿度采集。采集数据后，DHT11 转换到低速模式。具体时序见图 3-41。

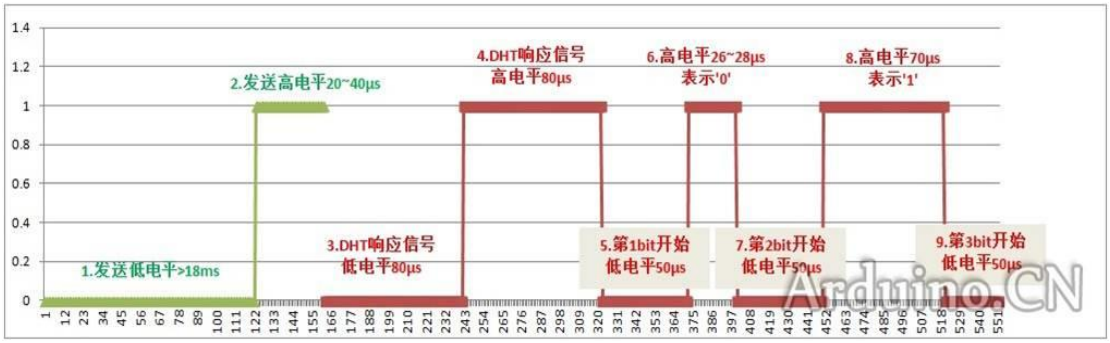

图 3-41 DHT11 的信息采集操作时序

作为软件编程人员，需要从上述介绍中知道：（1）接口管脚的位置和含义；（2）高/低电平有效；（3）操作时序（顺序）；（4）数据格式和含义。好在，所有这些都有现成的相关器件函数库，为你做好了，你只要按相关函数定义，调用就行。这也就是“抽象”的好处。

3.7.5 硬件接线

根据图 4-40 的定义，具体接线方法如下：（arduino 蓝色正面向上）：
VCC （左） →3.3V/5V 电源正极；
GND （右） →电源负极；
DATA （中） →arduinoIO 口；
注意：切勿将 VCC 与 GND 接反，接反必烧！ 接线效果见图 3-42：

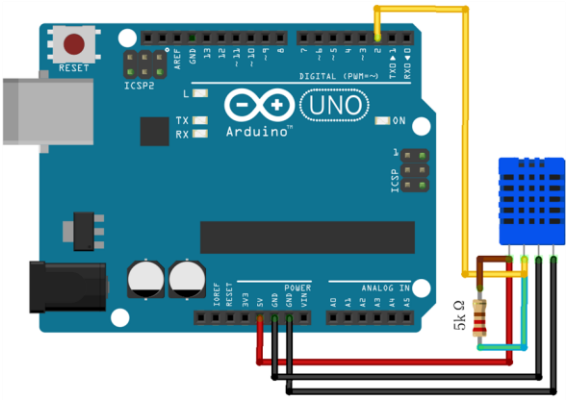

图 3-42 arduino 连接 DHT11 的接线图

3.7.6 DHT11 传感器信息采集软件

（1）下载库文件（帮你处理 DHT11 时序细节），解压到 arduino IDE 的 libraries 目录下：
http://www.arduino.cn/forum.php?mod=attachment&aid=ODEwfGFhMjQwNzJifDE0MTg3MTI5NzF8MzE3MTd8MTQyOQ%3D%3D
（2）或用 arduinoIDE 上的 Sketch 的导入库，直接导入该压缩文件，如图 3-43：

图 3-43 导入 DHT11 库文件

（3）创建采集软件：
在 IDE 上，创建以下程序：

```
double Fahrenheit(double celsius) //摄氏温度度转化为华氏温度函数
{
 return 1.8 * celsius + 32;
}
double Kelvin(double celsius) //摄氏温度转化为开氏温度函数
{
return celsius + 273.15;
}
// 露点计算函数（点在此温度时，空气饱和并产生露珠）
```

```
        double dewPoint(double celsius, double humidity)
        {       double A0= 373.15/(273.15 + celsius);
                double SUM = -7.90298 * (A0-1);
                SUM += 5.02808 * log10(A0);
                SUM += -1.3816e-7 * (pow(10, (11.344*(1-1/A0)))-1) ;
                SUM += 8.1328e-3 * (pow(10,(-3.49149*(A0-1)))-1) ;
                SUM += log10(1013.246);
                double VP = pow(10, SUM-3) * humidity;
                double T = log(VP/0.61078);   // temp var
                return (241.88 * T) / (17.558-T);
        }
        // 快速计算露点函数，速度是 5 倍 dewPoint()
        double dewPointFast(double celsius, double humidity)
        {      double a = 17.271;
                double b = 237.7;
                double temp = (a * celsius) / (b + celsius) + log(humidity/100);
                double Td = (b * temp) / (a - temp);
                return Td;
        }
#include <dht11.h> //导入的库文件
dht11 DHT11;
#define DHT11PIN 2 //使用 arduino uno D2 脚，如果使用其他脚，要修改这里；
void setup() //设置串口（此串口是 arduino 串口监视的串口）
{ Serial.begin(9600); //串口 9600/s
  Serial.println("DHT11 TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT11LIB_VERSION);
  Serial.println();
}
void loop() //温度采集主程序
{  Serial.println("\n");
   int chk = DHT11.read(DHT11PIN); //读 DHT11，DHT11.read 是 dht11.cpp 提供的例程
   Serial.print("Read sensor: ");
  switch (chk)
  {
    case DHTLIB_OK:
                Serial.println("OK");
                break;
    case DHTLIB_ERROR_CHECKSUM:
                Serial.println("Checksum error");
                break;
    case DHTLIB_ERROR_TIMEOUT:
                Serial.println("Time out error");
                break;
```

```
  default:
          Serial.println("Unknown error");
          break;
 }
 Serial.print("Humidity (%): "); //湿度
 Serial.println((float)DHT11.humidity, 2);
  Serial.print("Temperature (oC): "); //摄氏温度
 Serial.println((float)DHT11.temperature, 2);
  Serial.print("Temperature (oF): "); //华氏温度
 Serial.println(Fahrenheit(DHT11.temperature), 2);
  Serial.print("Temperature (K): "); //K 氏温度
 Serial.println(Kelvin(DHT11.temperature), 2);
  Serial.print("Dew Point (oC): ");//露点
 Serial.println(dewPoint(DHT11.temperature, DHT11.humidity));
  Serial.print("Dew PointFast (oC): "); //快速计算的露点
 Serial.println(dewPointFast(DHT11.temperature, DHT11.humidity));
 delay(2000);
}
```

3.7.7 编译运行

在 arduino IDE 菜单上，选择验证/编译。编译通过后，将代码上传到 arduino。将 arduino 重启一下。在 arduino IDE 的串口监视器上，就可以看到如图 3-44 的运行结果：

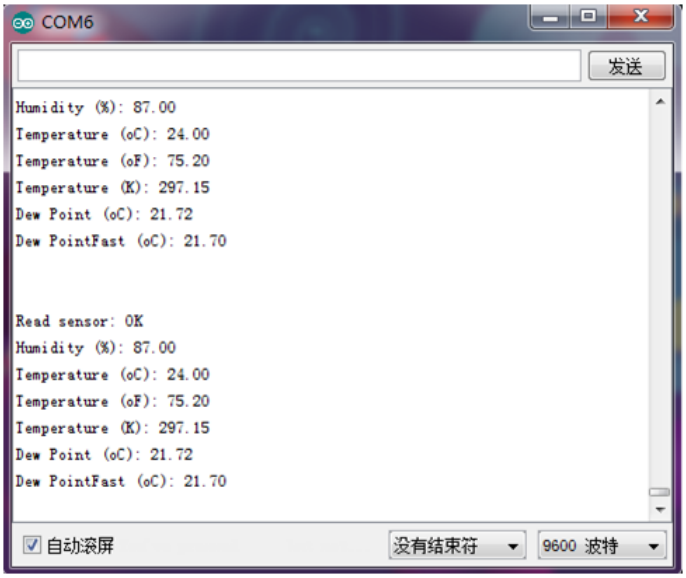

图 3-44 采集到的温度结果显示

把探头放到冒着热气的茶杯上面，看看有什么变化？

3.8 用四位数码管显示温度

用 arduino 的串口监控，显示传感器的温度，毕竟不规范。如果 arduino 脱离 IDE，怎么看温度呢？那就为 arduino 加一个显示器件吧。这个显示器件，当然不是电脑显示器，这

里为同学们选择的是：四位共阴数码管。

3.8.1 四位共阴数码管介绍

这个数码管的简单情况如下，外形如图 3-45：

1、名称：4 位数码管显示模块，采用 74HC595+8550 三极管驱动；

2、数码管型号：0.56 英寸共阳数码管；

3、工作电压：3.3V-5V；

4、设有两个固定螺栓孔方便安装，孔间距 34MM；

5、底板尺寸：5cm*2.7cm。

图 3-45 四位共阴数码管的实物图（正反面）

3.8.2 数码管接口说明

该数码管的输入与输出（见背板两列针接口）是对称的，即可拼接多块单元板，作串联、连续显示。

各脚定义如图 3-46：

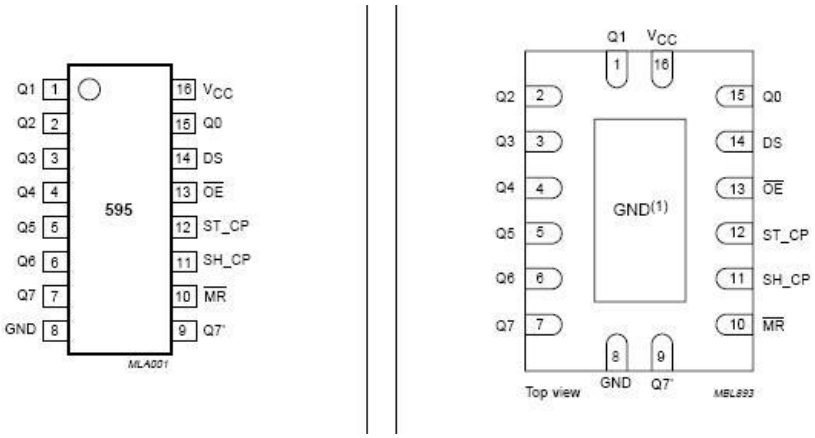

图 3-45 四位共阴数码管的管脚信号定义

引脚说明如表 3-2：

符号	引脚	描述
Q0…Q7	15， 1， 7	并行数据输出
GND	8	地
Q7’	9	串行数据输出

MR	10	主复位（低电平）
SH_{CP}	11	移位寄存器时钟输入
ST_{CP}	12	存储寄存器时钟输入
OE	13	输出有效（低电平）
D_S	14	串行数据输入
V_{CC}	16	电源

关键各脚的信号说明如下：

1、16 脚正电源接口，+3-5V。可直接与单片机的 3.3V-5V 电压连接，脚 8 为 GND，为电源负极。

2、脚四针位选择 1-4 输入，低电平位选，可直接与单片机 IO 口相连。

3、三针串入数据引脚，DAT,CLK,ST，为 74HC595 串行输入控制，可直接与单片机 IO 口相连。

注：模块与单片机 IO 口相连需要 7 个 IO 口，不管你拼接多少块单元，都只要 7 个 IO 口连接，扩展灵活

74HC595 的控制逻辑 74HC595 是 8 位串行输入并行输出移位寄存器，也就是串行转并行，来驱动数码管，否则，就需要如前例，每个数码管的每笔驱动。

项目中用到的输入分别接 74HC595 的：

- ✓ DAT_IN：DS；
- ✓ CLK：SHCP；
- ✓ ST：STCP。

3.8.3 74HC595 的控制逻辑

74HC595 的 DS（14）为串行数据输入口；SH_CP（11）为串行时钟输入口；ST_CP（12）为寄存器移位。现分别介绍其控制逻辑：

SH_CP：

每个上升沿到来时，芯片内部的移位寄存器会左移一位，最低位由 DS 决定，最高位移出丢失，次高位成为最高位，并在 Q7'体现出来（根据 Q7'可以看出，74HC595 也有串行输出功能）；

ST_CP：

每个上升沿会将移位寄存器的值输出到存储寄存器,存储寄存器直接和引脚 Q0~Q7 相连，所以存储寄存器的值会直接反映在引脚 Q0~Q7 上，从而实现串行转并行功能；

OE 是输出使能，高电平时 Q0~Q7 为高阻态，低电平时 Q0~Q7 为存储寄存器的值；MR 为低电平时，移位寄存器会被清 0，高电平时无效；VCC 接电源；GND 接地。

3.8.4 数码管的片选与刷新

74HC595 数码管的驱动方式是动态驱动。即每个数码管的 a,b,c,d,e,f,g,dp 同名片段连在一起，同时每个数码管有自己的独立的控制 I/O，用于控制是否显示。比如 4 位数码管，总共有 4 个针脚用于控制每个数码管的显示。

74HC595 是“共阴”数码管，那么控制片段显示的针脚为 LOW 的时候，对应的数码管才会显示（低电平起作用）。由于 74HC595 每次只能显示一个数字，如果需要每个数码管显

示不同的数字，那么必须通过在 1～2ms 内反复刷新。

3.8.5 代码说明

每个数码管都是由一个 8 位笔画构成的，要显示 0-9（以及小数点 DP），需要分别告诉数码管具体的 8 位信息。这就是所谓数码管的“真值表”，如表 3-3：

数码管实现0-9的真值表：

数字	Pin7	Pin6	Pin5	Pin4	Pin3	Pin2	Pin1	Pin0
0	HIGH	HIGH	LOW	LOW	LOW	LOW	LOW	LOW
1	HIGH	HIGH	HIGH	HIGH	HIGH	LOW	LOW	HIGH
2	HIGH	LOW	HIGH	LOW	LOW	HIGH	LOW	LOW
3	HIGH	LOW	HIGH	HIGH	LOW	LOW	LOW	LOW
4	HIGH	HIGH	LOW	HIGH	HIGH	LOW	LOW	HIGH
5	HIGH	LOW	LOW	HIGH	LOW	LOW	HIGH	LOW
6	HIGH	HIGH	LOW	LOW	LOW	LOW	HIGH	HIGH
7	HIGH	HIGH	HIGH	HIGH	HIGH	LOW	LOW	LOW
8	HIGH	LOW	LOW	LOW	LOW	LOW	LOW	LOW
9	HIGH	LOW	LOW	HIGH	HIGH	LOW	LOW	LOW
DP	LOW	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH

表 3-3：数码管的真值表

数码管显示程序代码如下：

```
// 0-9 10 个显示数字与小数点（DP）
byte seven_seg_digits[10] =
        { B00000011,   // = 0
          B10011111,   // = 1
          B00100101,   // = 2
          B00001101,   // = 3
          B10011001,   // = 4
          B01001001,   // = 5
          B01000001,   // = 6
          B00011111,   // = 7
          B00000001,   // = 8
          B00001001,     // = 9
          B111111110       // = dp };
//数码管初始化
int latchPin = 6;    //ST_CP 移位
int clockPin = 5;    //SH_CP 时钟
int dataPin = 4;    //DS 数据串口
void setup() {
   pinMode(latchPin, OUTPUT);   //移位输出
   pinMode(clockPin, OUTPUT); //时钟输出
   pinMode(dataPin, OUTPUT); //数码管串口输出
```

```
  pinMode(8, OUTPUT); //片选 1 输出
  pinMode(9, OUTPUT); //片选 2 输出
  pinMode(10, OUTPUT); //片选 3 输出
  pinMode(11, OUTPUT); //片选 4 输出
  pinMode(2, INPUT);  //温度传感器采集口
}
void sevenSegWrite(byte digit, int b) {
  digitalWrite(8, HIGH);  //全暗
  digitalWrite(9, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(latchPin, LOW); //移位低
  shiftOut(dataPin, clockPin, LSBFIRST, seven_seg_digits[digit]); //移位
  digitalWrite(latchPin, HIGH); //移位高，完成写入
  digitalWrite(b, LOW); //点亮 b（片选）
}
//SH_CP 每个上升沿到来时，芯片内部的移位寄存器会左移一位，最低位由 DS 决定，最高
位移出丢失，次高位成为最高位，并在 Q7'体现出来（根据 Q7'可以看出，74HC595 也有串
行输出功能）；
//ST_CP 每个上升沿会将移位寄存器的值输出到存储寄存器,存储寄存器直接和引脚 Q0~Q7
相连，所以存储寄存器的值会直接反映在引脚 Q0~Q7 上，从而实现串行转并行功能；
shiftOut(dataPin,clockPin,bitOrder,val)
//shiftOut 函数能够将数据通过串行的方式在引脚上输出，相当于一般意义上的同步串行通
//信，这是控制器与控制器、控制器与传感器之间常用的一种通信方式。
//shiftOut 函数无返回值，有 4 个参数：dataPin、clockPin、bitOrder、val，具体说明如下：
//dataPin：数据输出引脚，数据的每一位将逐次输出。引脚模式需要设置成输出。
//clockPin：时钟输出引脚，为数据输出提供时钟，引脚模式需要设置成输出。
//bitOrder：数据位移顺序选择位，该参数为 byte 类型，有两种类型可选择，分别是高位先
//入 MSBFIRST 和低位先入 LSBFIRST。
//val：所要输出的数据值。
//项目代码：
  digitalWrite(latchPin, LOW); //移位低
  shiftOut(dataPin, clockPin, LSBFIRST, seven_seg_digits[digit]);
  digitalWrite(latchPin, HIGH); //移位高，完成写入
  digitalWrite(b, LOW); //点亮 b（片选）
//**说明：
shiftOut 函数原型如下：
void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val)
{
    uint8_t i;
    for (i = 0; i < 8; i++)
    {
        if (bitOrder == LSBFIRST)
            digitalWrite(dataPin, !!(val & (1 << i)));
```

```
            else
                digitalWrite(dataPin, !!(val & (1 << (7 - i))));
            digitalWrite(clockPin, HIGH);
            digitalWrite(clockPin, LOW);
    }
}
*//
void showNum(int num) { //显示一个数字
  int b = 11; //数码管移位操作次数-实际以数据位数决定
  while(1) {
    sevenSegWrite(num % 10, b); //将 num 逐位写到 b 位数码管
    b -= 1;   //数码管移一位
    if(num/10 > 0){ //数值移一位
      num /= 10;
    }
    else {
      break;
    }
  }
}
 void loop() { //主程序
  showNum(analogRead(A5)); //显示树莓派 A5 口（温度）的采集信息，反复写（刷新）
}
```

至此，脱离 IDE，在 arduino 上，可以直接用数码管显示实时温度了，本阶段实训项目也完成了在 arduino 平台上，DHT11 温度传感器与四位数码管的集成。这里暂且称为“前端”的基于 arduino 的温度采集集成。

3.9 课程小结

传感器数据采集和数码管显示的开发，是本节的重点。对没有接触过硬件的同学来说，理解传感器和数码管的原理、特别是信号电平、触发、时序等与电路控制有关的概念，有一定难度，毕竟没有学过多少“硬件”知识。但是，回顾这几个器件的开发、集成过程，可以发现一个基本的开发方法和思路。即：老师给一个“范例”，然后同学们自己上网找资料，包括相关传感器的电路图、接线图、接口说明、android 例库等，甚至可以找到现成的代码。而下面要做的事情，就是学习这些资料，可能有些代码直接就可以拿过来用，有时要再稍做修改，也可以完成不是太熟悉的软硬结合的开发。

本节课做了三件事：（1）熟悉和了解基于 arduino 平台的开发；（2）熟悉 DHT11 温度传感器和四位数码管的使用；（3）将 arduino 作为前端平台，在其上，集成了温度传感器和数码管两个部件。本阶段实训项目的最大的意义和价值是：尝试并体验了传感器、数码管与 arduino 的开发过程，并成为以后继续此类开发的成功案例。

显然，这样的“集成”，除了实现了温度传感器信息采集和数码管显示的功能外，还将温度传感器的数据，通过 arduino 代码，传送给数码管。虽然这次暂时没有对温度传感器的数据，进行任何处理；也没有对数码管添加更多的显示变化，但是想想：如果需要，在现有基础上，这些需求都是完全可以做到的。这就是说，到目前为止，同学们已经具有了在这个

“前端平台”上继续“做点什么”的基础。

实际情况也是这样，同学们可以把实训课程中所购买的其他 30 多种传感器，都尝试集成一遍，并用数码管显示它们采集到的数据。甚至可以把这些“器件”，按其功能，互相组合，创造出各种“应用”。

例如：温度、湿度、气压、亮度、光强度、人体感应、红外等信号的采集，达到一定“阀值（控制）”值之后，就可以引发 LED 闪亮、声音报警、信息显示等。由此看来，那些非常“炫”的物联网应用，其基础，不过如此而已。

下一节课将讨论如何将目前已经完成的“前端”部分，与树莓派、甚至与网上的数据服务器（Yeelink）、与自己开发的 Web 应用服务器（苏宁云），进行更大的“后端”集成。当然，这里所谓“前端”与“后端”的划界，只是为了说明方便，并不存在严格的“前与后”的划界原则和规定。如果非要有一个“理由”的话，那么“前端”是软硬结合的，“后端”是“纯软件”实现的。

由于考虑每章的篇幅限制，“后端”集成放在第四、五章中。虽然分在不同章节中，但它们应该被视为一个整体，就像全书也是一个整体一样。